Departamento de Computación
UNIVERSIDADE DA CORUÑA

TESE DE DOUTORAMENTO

Corpus lingüísticos estruturados de grandes dimensións: Metodoloxía e sistemas de recuperación de información

AUTOR: Fco. Mario Barcala Rodríguez
DIRECTORES: Dr. Manuel Vilares Ferro
Dr. Jorge Graña Gil

Setembro de 2009

A Cristina

Agradecementos

Non quixera deixar pasar esta oportunidade sen darlles as grazas a todas as persoas que tiveron algo que ver na realización desta tese de doutoramento.

A Manuel Vilares e Jorge Graña, polo seu labor de dirección e corrección.

A Miguel, Fran, Víctor, Suso, Moli, Juan, Nieves, Erica, Mila, Sara, Vanesa, Adrián e Carlos, compañeiros desta viaxe. Dun ou doutro xeito puxeron o seu grao de area.

Ao Centro Ramón Piñeiro para a Investigación en Humanidades, por facilitarme o acceso aos recursos necesarios para desenvolver este traballo, especialmente a Guillermo Rojo, María Sol López e Eva Domínguez, porque das moitas reunións que tivemos xurdiron os aspectos clave aquí tratados. Tamén a Fernando e Carlos, por esas achegas espontáneas de encontros improvisados.

Ao Servizo de Normalización Lingüística da Universidade da Coruña e o Servizo de Terminoloxía de Galicia, especialmente a Xesús e Inés, polo seu asesoramento lingüístico.

A meus pais e sogros, polo seu apoio constante en todos os ámbitos.

A toda a comunidade do software libre, sen a cal este traballo non podería ser levado a cabo dentro duns custos económicos razoables.

E finalmente, e máis importante, a Cristina, sen quen todo isto non sería posible.

A todos, moitas grazas.

Índice xeral

Índice de figuras

Parte I

Introdución

A recente evolución de Internet provocou a posibilidade de acceso a un volume de información enorme, pero toda ela non resulta útil se non existe unha maneira precisa de atopar o que se necesita nun momento dado. Por iso, case paralelamente ao crecemento de Internet fóronse desenvolvendo sistemas de recuperación de información (RI) que permitían localizar a información relevante en cada caso, dando lugar ao que hoxe coñecemos como buscadores. Mais un dos principais problemas que presentaban radicaba en que, en xeral, a información que utilizaban estaba moi pouco estruturada, o que limitaba en certa modo as súas posibilidades: non se podían delimitar as seccións dos documentos, nin aplicar filtros de busca etc., é dicir, só se permitía introducir unha expresión de busca que se intentaba atopar en toda a base documental.

Debido a estas carencias, ao mesmo tempo tamén se foron desenvolvendo sistemas de RI que requirían que a información estivera organizada dalgún xeito particular. Estes sistemas non estaban deseñados para facer buscas en Internet en xeral, senón que actuaban sobre un conxunto máis grande ou máis pequeno de información dispoñible e ofrecían máis oportunidades de busca.

Estas dúas vertentes evolutivas, a de utilizar información desestruturada e a de tela organizada, chegaron á actualiade propiciando a aparición de distinas ferramentas de busca. Por unha banda, temos os buscadores de Internet, que permiten localizar documentos que satisfán unha busca concreta e, pola outra, os sistemas que utilizan información estruturada, que cobren aspectos como a obtención de datos de clientes, facturación, control de stock etc. Finalmente, mesmo hai contornos que poden combinar en diferente grao estas dúas vertentes (ferramentas de prospección de datos, sistemas de predición etc.).

Neste traballo tratamos un caso particular dos sistemas de RI que utilizan información estruturada: o dos sistemas lingüísticos que traballan con grandes coleccións de documentos (corpus), o que encadra a presente tese de doutoramento dentro da lingüística computacional e, máis concretamente, na lingüística de corpus. Malia que neste eido tamén hai un amplo espectro de posibilidades, centrámonos naqueles en que a información que necesitan os usuarios, normalmente lingüistas, está relacionada coa frecuencia de ocorrencia de palabras ou coa visualización de exemplos no seu contexto.

A evolución destes sistemas foi practicamente simultánea ao desenvolvemento da informática. Desde as primeiras ferramentas de busca monolíticas que utilizaban coleccións textuais, consideradas agora de reducidas dimensións, foise evolucionando grazas ao incremento da capacidade dos computadores, ata os actuais sistemas de consulta a través da rede que manexan corpus de gran tamaño[1]. Centrarémonos nestes últimos, analizando as diferentes posibilidades e tecnoloxías dispoñibles actualmente para desenvolvelos pero, ademais, tamén faremos unha proposta metodolóxica xenérica sobre a creación de corpus[2], que son o sustento de datos destes sistemas de RI.

Ofrecemos, pois, unha visión de conxunto que abarca, tanto a construción de corpus como a súa posterior explotación, mentres temos en mente en todo momento o emprego dos estándares máis actuais. Con isto esperamos conseguir que, por unha banda, se utilice este traballo como guía para outros proxectos futuros e, pola outra, se garanta unha evolución adecuada no tempo das propostas que facemos.

[1] Na actualidade consideramos que un corpus de gran tamaño neste ámbito é aquel que inclúe varias decenas ou centenas de millóns de palabras.

[2] A ausencia de información publicada ao respecto motivounos a afrontarmos este reto, que rematou por ser unha das liñas de investigación fundamental.

Motivación e obxectivos da tese

O punto de partida do presente traballo poderiamos situalo no desenvolvemento de diversas ferramentas para o procesamento da linguaxe natural (PLN) que se tiveron que realizar no marco dun convenio de colaboración entre o Centro Ramón Piñeiro para a Investigación en Humanidades [20] e o grupo COLE [46] da Universidade da Coruña.

Unha vez se profundou nas necesidades do centro con respecto ao proxecto Corpus de referencia do galego actual (CORGA) [21, 55], comprobouse que era necesario reestruturalo en dúas frontes: por un lado, cumpría enriquecer a estrutura dos documentos xa existentes e, polo outro, desenvolver un novo sistema de consultas que puidera facer uso desta nova estrutura.

A partir das investigacións realizadas para resolver estas dúas necesidades e do interese propio que tiñamos en xeneralizalas para que puideran ser empregadas noutros contextos, xurdiu este traballo. Os seus obxectivos son:

- Definir unha proposta metodolóxica para a creación de corpus estruturados de grandes dimensións.
- Definir os requirimentos que se lle poden esixir a un sistema de RI que consulte un corpus.
- Avaliar as tecnoloxías dispoñibles para desenvolver este tipo de sistemas.
- Propoñer unha arquitectura para os mesmos.
- Aplicar as propostas anteriores a un caso particular, o do proxecto CORGA.

Contexto de traballo

O traballo desenvolvido é consecuencia da participación en diversos contratos, proxectos e bolsas de investigación que enumeramos a continuación por orde cronolóxica inversa:

Bolsas e contratos de investigación

- Contrato para o desenvolvemento das aplicacións que permitan a construción dos textos etiquetados e lematizados do Corpus de Referencia do Galego Actual. Consellería de Presidencia (04/07/2007-03/07/2009).
- Contrato como investigador para a realización de traballos de lingüística de corpus, computacional, elaboración de ontoloxías específicas e bases de datos terminolóxicas no proxecto de investigación *Creación e integración multilingüe de recursos lingüísticos en gallego para* RI *mediante estrategias de control terminológico y discursivo en ámbitos comunicativos especializados.* Universidade de Santiago de Compostela (09/06/2005-08/12/2007).
- Contrato de servizos para o deseño e desenvolvemento de ferramentas para a análise automática do galego actual. Consellería de Educación e Ordenación Universitaria (04/06/2003-31/05/2007).

- Bolsa para a colaboración nun proxecto de investigación sobre o desenvolvemento dun desambiguador de textos en lingua galega no Centro Ramón Piñeiro para a Investigación en Humanidades. Consellería de Educación e Ordenación Universitaria (01/05/2000-30/04/2003).

Proxectos I+D

- *Consolidación e estruturación de unidades de investigación competitivas.* Xunta de Galicia (DOG num. 203 do 20 de outubro de 2008), 2008.
- *Mellora na recuperación de noticias e no acceso a información financeira: recuperación de textos sobre bases documentais de axencias de noticias.* Xunta de Galicia (07SIN005206PR), 2007-2010.
- *Búsqueda de respuestas empleando metagramáticas.* Ministerio de Educación y Ciencia (HUM2007-66607-C04-02), 2007-2010.
- *Rede galega de procesamento da linguaxe e recuperación de información.* Xunta de Galicia (DOG núm. 238 do 13 de decembro de 2006), 2006-2009.
- *Herramientas para el análisis y la clasificación automática de documentos legislativos.* Deputación de Ourense (2005-INOU-09), 2006.
- *Aplicación de técnicas de procesamiento del lenguaje natural en sistemas de búsqueda de respuestas sobre documentos técnicos.* Universidade de Vigo, 2005-2006.
- *Generación, extracción y estructuración de información legislativa mediante técnicas de inteligencia artificial.* Xunta de Galicia e Telémaco, Información, Documentación y Sistemas (PGIDIT05SIN044E), 2005-2006.
- *Extracción de Información Económica Multilingüe.* Xunta de Galicia (PGIDIT05PXIC30501PN), 2005-2008.
- *Proyecto GARI-COTER: Creación e integración multilingüe de recursos lingüísticos en gallego para Recuperación de Información mediante estrategias de control terminológico y discursivo en ámbitos comunicativos especializados.* Ministerio de Educación y Ciencia (HUM2004-05658-C02-02), 2004-2007.
- *Aplicaciones de la Ingeniería lingüística a los sistemas colaborativos y desktop publishing.* Universidade da Coruña, 2003.
- *Evaluación interactiva de la pertinencia en entornos de recuperación automática de información.* Secretaría Xeral de Investigación e Desenvolvemento da Xunta de Galicia (PGIDIT02PXIB30501PR), 2002-2004.
- *Extracción de información de noticias bursátiles para a avaliación do sentimento do mercado.* Universidade da Coruña, 2002.
- *Análisis sintáctico robusto para portugués, gallego y español.* Ministerio de Ciencia y Tecnología (HP2001-0044), 2002-2003.

- *Interrogación y recuperación de información sobre Web: Una arquitectura distribuida.* Ministerio de Ciencia y Tecnología (TIC2000-0370-C02-01) e Secretaría Xeral de Investigación e Desenvolvemento da Xunta de Galicia (PGIDT01PXI10506PN), 2001-2003.

- *Cluster sobre arquitectura x86.* Secretaría Xeral de Investigación e Desenvolvemento da Xunta de Galicia (PR405A/00/27), 2001-2002.

- *Extracción y Recuperación de Información mediante Análisis Lingüístico.* Unión Europea (1FD97-0047-C04-02), 1998-2000.

Estrutura da memoria

O presente traballo divídese en seis partes. Introdución, definición e construción de corpus, sistemas de recuperación de información, conclusión e liñas de traballo futuro, apéndices e bibliografía.

O núcleo da tese constitúeno as partes II e III. A primeira delas, titulada "Definición e construción de corpus" engloba os capítulos relacionados coa definición e construción de corpus. Primeiramente, no capítulo 1, proponse unha metodoloxía de traballo en cinco fases: a de definición, onde se establecen os diferentes criterios que hai ter en conta para a construción dun corpus, así como as fases de procesamento dos documentos; a de construción, en que se describen os elementos máis relevantes que son necesarios para poñer en práctica a metodoloxía; a de comprobación, que valida se os recursos informáticos e metodolóxicos construídos dan cabida aos criterios da primeira fase; a de planificacion temporal, en que se organiza cronoloxicamente o traballo; e a de desenvolvemento, onde os procesos definidos para tratar os documentos se implantan no equipo de traballo de xeito definitivo.

No capítulo 2 da parte II materialízanse os conceptos metodolóxicos para o CORGA: concrétanse os tipos de documentos que serán tidos en conta, as decisións tomadas nos diferentes ámbitos de actuación, materialízanse as fases do procesamento, compróbase a metodoloxía, faise unha planificación temporal e ponse en práctica esta última para, finalmente, explicarmos a maneira en que se trataron os textos que existían no proxecto con antelación.

A parte III, titulada "Sistemas de recuperación de información", analiza aspectos variados relacionados coa construción de sistemas de RI baseados en corpus. No capítulo 1 xeneralízanse os requirimentos comúns a este tipo de sistemas de RI e explícase en que consiste cada un deles. No capítulo 2 avalíanse diferentes tecnoloxías que poden ser utilizadas na construción destes sistemas: indexadores de texto, aplicacións lingüísticas, bases de datos relacionais e xestores XML [99]. A continuación, elíxense as que a priori parecen ser máis prometedoras e, por último, compárase o grao de cumprimento dos requirimentos establecidos no capítulo 1 en cada unha delas. No capítulo 3 defínense algunhas consultas tipo que cobren moitos dos aspectos considerados no capítulo 1 para, posteriormente, facermos unha avaliación de rendemento das tecnoloxías máis prometedoras. A descrición polo miúdo das consultas utilizadas nesta avaliación atópase no apéndice D. Finalmente, no capítulo 4 descríbense os aspectos clave da arquitectura do sistema de RI construído para a consulta do corpus CORGA.

Adicionalmente, na parte IV expóñense as conclusións que se obtiveron e as liñas de traballo futuro que barallamos, na parte V inclúense os apéndices (A, B e C dan exemplos de protocolos seguidos no CORGA: adquisición de documentos en papel, estruturación dun xornal e revisión

dun xornal, D amosa pormenorizadamente as sentenzas SQL correspondentes, tanto á creación das táboas como á execución das buscas, utilizadas na avaliación de rendemento do capítulo 3 e, o E presenta unha listaxe descritiva das siglas utilizadas ao longo de todo o traballo) e, para rematar, na parte VI recóllese a bibliografía utilizada.

Difusión dos resultados

Debido ás diferentes investigacións realizadas durante a preparación desta tese de doutoramento fixéronse diversas publicacións, tanto de ámbito nacional como internacional, que inclúen resultados parciais relacionados co presente traballo. Incluímolas a continuación por orde cronolóxica inversa:

1. LORENTE, Mercè [et al.], *Vocabulario multilingüe de economía*, Barcelona: Institut Universitari de Lingüística Aplicada, 2008. Sèrie Materials, 7.

2. ROJO, Guillermo, GARCÍA, Francisco, BARCALA, Fco. Mario e DOMÍNGUEZ, Eva, *Corpus de Referencia do Galego Actual (CORGA), versións 1.2 e 1.3*, Centro Ramón Piñeiro para a Investigación en Humanidades, 2007 e 2008.

3. BARCALA, Fco. Mario, MOLINERO, Miguel A. e DOMÍNGUEZ, Eva, "XML rules for enclitic segmentation". En Roberto Moreno Díaz, Franz Pichler, Alexis Quesada Arencibia (eds.), *Computer Aided Systems Theory - EUROCAST 2007: 11th International Conference on Computer Aided Systems Theory, Las Palmas de Gran Canaria, Spain, February 12-16, 2007, Revised Selected Papers*, Berlin / Heidelberg: Springer, 2007. p. 273-281. Lecture Notes in Computer Science (LNCS), volume 4739.

4. BARCALA, Fco. Mario [et al.], "El proyecto Gari-Coter en el seno del proyecto RICOTERM2". En Víctor J. Díaz Madrigal, Fernando Enríquez de Salamanca Ros (eds.), *XXIII Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural: Universidad de Sevilla, 10, 11 y 12 de septiembre de 2007*, España: 2007, p. 295-296. Procesamiento del Lenguaje Natural, número 39.

5. BARCALA, Fco. Mario [et al.], "A Corpus and Lexical Resources for Multi-word Terminology Extraction in the field of Economy in a Minority Language". En Zygmunt Vetulani (ed.), *Human Language Technologies as a Challenge for Computer Science and Linguistics: Proceedings of 3rd Language & Technology Conference, October 5-7, 2007, Poznań, Poland*, Poznań: Wydawnictwo Poznańskle Sp. z o.o., 2007, p. 359-363.

6. MOLINERO, Miguel A. BARCALA, Fco. Mario, OTERO, Juan e GRAÑA, Jorge, "Practical application of One-Pass Viterbi Algorithm in Tokenization and Part-of-Speech Tagging". En Galia Angelova [et al.] (eds.), *International Conference Recent Advances in Natural Language Processing Proceedings: 27-29 September 2007, Borovets, Bulgaria*, Showmen: INCOMA Ltd.,2007, p. 35-40.

7. BARCALA, Fco. Mario, BLANCO, Cristina e DARRIBA, Víctor Manuel, "Metodología para la construcción de córpora textuales estructurados basados en XML". En Manuel Palomar Sanz [et al.] (eds.), *Procesamiento del Lenguaje Natural*, número 36, Alicante: Universidad de Alicante, 2006, p. 9-16.

8. BARCALA, Fco. Mario, MOLINERO, Miguel A. e DOMÍNGUEZ, Eva, "Information Retrieval and Large Text Structured Corpora". En Roberto Moreno Díaz, Franz Pichler e Alexis Quesada Arencibia (eds.), *Computer Aided Systems Theory - EUROCAST 2005: 10th International Conference on Computer Aided Systems Theory, Las Palmas de Gran Canaria, Spain, February 7 - 11, 2005: Revised Selected Papers*, Berlin / Heidelberg: Springer, 2005, p. 91-100. Lecture Notes in Computer Science (LNCS), volume 3643.

9. VILARES, M., Otero, J., BARCALA F.M. e DOMÍNGUEZ, E., "Automatic Spelling Correction in Galician". En José Luis Vicedo [et al.] (eds.), *Advances in Natural Language Processing: 4th International Conferente, EsTAL 2004, Alicante, Spain, October 2004 Proceedings*, Berlin / Heidelberg: Springer, 2005, p. 45-57. Lecture Notes in Artificial Intelligence (LNAI), volume 3230.

10. VILARES, Jesús, BARCALA, Fco. Mario, FERNÁNDEZ, Santiago e OTERO, Juan, "Manejando la variación morfológica y léxica en la Recuperación de Información Textual". En Manuel Palomar Sanz [et al.], *Procesamiento del Lenguaje Natural*, número 30, España: 2003, p. 99-106.

11. VILARES, Jesús [et al.], "Practical NLP-Based Text Indexing". En Francisco J. Garijo, José C. Riquelme, Miguel Toro (eds.), *Advances in Artificial Intelligence - IBERAMIA 2002: 8th Ibero-American Conference on AI, Seville, Spain, November 2002 Proceedings*, Berlin / Heidelberg: Springer, 2002, p. 635-644. Lecture Notes in Artificial Intelligence (LNAI), volume 2527.

12. BARCALA, Fco. Mario [et al.], "Tokenization and Proper Noun Recognition for Information Retrieval". En A Min Tjoa e Roland R. Wagner (ed.), *Proceedings: 13th International Workshop on Database and Expert Systems Applications, 2-6 September 2002, Aix-en-Provence, France*, Los Alamitos: IEEE Computer Society, 2002, p. 246-250.

13. BARCALA, Fco. Mario [et al.], "Una aplicación de RI basada en PLN: el proyecto ERIAL". En Emilio Sanchis, Lidia Moreno e Isidoro Gil (eds.), *JOTRI 2002: I Jornadas de Tratamiento y Recuperación de Información (JOTRI), Facultad de Informática, Valencia, 4 y 5 de Julio de 2002*, Valencia: Editorial UPV, 2002, p. 165-172.

14. BARCALA, Fco. Mario [et al.], "El sistema ERIAL: LEIRA, un entorno para RI basado en PLN". En Emilio Sanchis, Lidia Moreno e Isidoro Gil (eds.), *JOTRI 2002: I Jornadas de Tratamiento y Recuperación de Información (JOTRI), Facultad de Informática, Valencia, 4 y 5 de Julio de 2002*, Valencia: Editorial UPV, 2002, p. 173-174.

15. VILARES, Jesús, BARCALA, Fco. Mario, ALONSO, Miguel A., "Using syntactic dependency-pairs conflation to improve retrieval performance in Spanish". En Alexander Gelbuk (ed.), *Computational Linguistics and Intelligent Text Processing: Third International Conference, CICLing 2002, Mexico City, Mexico, February 2002 Proceedings*, Berlin / Heidelberg: Springer, 2002, p. 381-390. Lecture Notes in Computer Science (LNCS), volume 2276.

16. GRAÑA, Jorge, BARCALA, Fco. Mario, VILARES, Jesús, "Using syntactic dependency-pairs conflation to improve retrieval performance in Spanish". En Alexander

Gelbukh (ed.), *Computational Linguistics and Intelligent Text Processing: Third International Conference, CICLing 2002, Mexico City, Mexico, February 2002 Proceedings*, Berlin / Heidelberg: Springer, 2002, p. 240-249. Lecture Notes in Computer Science (LNCS), volume 2276.

17. GRAÑA, Jorge, BARCALA, Fco. Mario, ALONSO, Miguel A., "Compilation Methods of Minimal Acyclic Finite-State Automata for Large Dictionaries". En Bruce W. Watson, Derick Wood (eds.), *Implementation and Application of Automata: 6th International Conference, CIAA 2001, Pretoria, Soth Africa, July 2001, Revised Papers*, Berlin / Heidelberg: Springer, 2002, p. 135-148. Lecture Notes in Computer Science (LNCS), volume 2494.

18. GRAÑA, Jorge, BARCALA, Fco. Mario, VILARES, Jesús, "Etiquetación Robusta del Lenguaje Natural: Preprocesamiento y Segmentación". En L. Alfonso Ureña López (ed.), *XVII Congreso de la SEPLN: Sociedad Española para el Procesamiento del Lenguaje Natural, Universidad de Jaén, 12-14 septiembre 2001*, España: 2001, p. 173-180. *Procesamiento del Lenguaje Natural*, número 27.

19. VILARES, Jesús, BARCALA, Fco. Mario, ALONSO, Miguel A., "Normalización de términos multipalabra mediante pares de dependencia sintáctica". En L. Alfonso Ureña López (ed.), *XVII Congreso de la SEPLN: Sociedad Española para el Procesamiento del Lenguaje Natural, Universidad de Jaén, 12-14 septiembre 2001*, España: 2001, p. 123-130. *Procesamiento del Lenguaje Natural*, número 27.

20. BARCALA, Fco. Mario, SACRISTÁN, Oscar, GRAÑA, Jorge, "Stochastic Parsing and Parallelism". En Alexander Gelbukh (ed.), *Computational Linguistics and Intelligent Text Processing: Second International Conference, CICLing 2001, Mexico City, Mexico, February 18-24, 2001 Proceedings*, Berlin / Heidelberg: Springer, 2001. Lecture Notes in Computer Science (LNCS), volume 2004.

Notación

Para simplificarmos a lectura do traballo seguimos os seguintes criterios tipográficos:

- Empregamos a cursiva para marcar estranxeirismos, títulos de proxectos, libros, exemplos e referencias a elementos que, doutro xeito, poderían causar confusión (por exemplo, os nomes de campos no modelo entidade-relación e os nomes de elementos e/ou atributos no caso de referírmonos a documentos XML).

- Para o caso dos nomes de obxectos e de entidades e relacións, dado que a súa propia nomenclatura establece que deben escribirse coa primeira letra en maiúscula, non empregaremos marca tipográfica ningunha.

- Utilizamos un tipo de letra monoespazado para os exemplos de SQL, descricións do contido de arquivos e exemplos de *scripts*.

Parte II

Definición e construción de corpus

Capítulo 1

Metodoloxía para a construción de corpus textuais estruturados

O desenvolvemento de corpus non é unha tarefa tan sinxela como nun primeiro momento pode parecer. Habitualmente, cando está definida a estrutura que teñen os documentos e xa se incorporaron varios ao corpus, xorde o problema de que un novo texto que desexamos introducir non se amolda á estrutura fixada. Cando isto sucede nunha fase inicial, non é excesivamente complexo modificala e depurala, de xeito que se poidan representar os novos elementos que non foran considerados nun primeiro momento. Pola contra, cando xorde en etapas avanzadas, é máis difícil abordar os cambios e pode supoñer un consumo de recursos difícil de asumir.

Neste capítulo propoñemos unha secuencia de fases que se deben levar a cabo para a construción de corpus textuais baseados na linguaxe extensible de marcado (*Extensible Marckup Language*, XML) [99] no ámbito da lingüística ou lexicografía de corpus, que abrangue desde a definición da metodoloxía de traballo ata o desenvolvemento do corpus propiamente dito. Existen numerosos estudos onde se explican os procedementos seguidos para o desenvolvemento dun corpus particular [13, 18, 77], e outros onde se intentan definir algúns criterios xerais que hai que ter en conta para o deseño e desenvolvemento de corpus [9, 15, 64, 78], pero non atopamos traballos onde se xeneralicen e organicen, tanto os conceptos xerais, coma os concretos, que poidan servir de guía completa para diferentes proxectos.

Os principios tratados a continuación describen os aspectos máis importantes que hai que considerar, no contexto da lingüística computacional, para o desenvolvemento de case calquera tipo de corpus textual estruturado monolingüe que non inclúa ningún tipo de información morfosintáctica. Evitaremos, polo tanto, mencionar tarefas e conceptos que teñan que ver con estoutra información.

Na figura II.1.1 aparecen as cinco fases necesarias para desenvolver un corpus destas características, que deseguido describimos brevemente e analizamos polo miúdo en seccións posteriores.

1. Definición da metodoloxía: Fíxanse as decisións que atinxen aos obxectivos que se perseguen, ao tipo de corpus que se vai desenvolver, de documentos que se incluirán, á estrutura que terán estes, aos procesos que deben seguir antes da súa incorporación ao corpus etc.

2. Construción da metodoloxía: Conséguense os recursos materiais, económicos e humanos necesarios para poñer en marcha a metodoloxía definida. Entre as tarefas incluídas

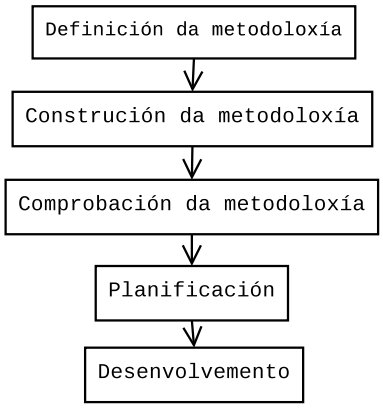

Figura II.1.1: Fases para a construción dun corpus

nesta fase atópanse as seguintes: adquisición das ferramentas necesarias, desenvolvemento doutras non existentes, creación de *scripts* e creación e/ou refinamento de definicións de tipos de documentos (*Document Type Definition*, DTD) [97] ou esquemas XML [100].

3. Comprobación da metodoloxía: Próbase o funcionamento desta con algúns documentos tipo. Trátase de procesar completamente uns poucos textos para comprobar que todos os elementos funcionan correctamente. Se se atopa algún problema, corríxense os defectos e fanse de novo as comprobacións pertinentes.

4. Planificación: Organízase o traballo que se vai realizar. Faise unha selección dos documentos que queremos que pasen a formar parte do corpus e establécese unha temporalización, é dicir, decídese en que momento vai ser procesado cada un.

5. Desenvolvemento: Trátase o procesamento dos documentos propiamente dito. Deben aplicarse todas as tarefas que defina a metodoloxía para os textos ata seren incluídos no corpus.

Aínda que as fases propostas seguen unha orde cronolóxica ás veces é necesario volver a algunha das precedentes para refinar ou readaptar as decisións tomadas con anterioridade. A volta a fases anteriores pode producirse, ben porque nos conveña organizar o traballo por partes, ben porque detectemos algún erro que debamos corrixir. Cantas menos veces se dea esta segunda circunstancia, menos recursos humanos e económicos consumiremos.

Daquela, ás veces é conveniente volver atrás voluntariamente para organizar o traballo ou refinar algunha decisión xa tomada. Por exemplo, pode ser que desexemos enfocar a metodoloxía por tipos de documentos: en primeiro lugar, abordamos as fases de definición, construción e comprobación da metodoloxía para un tipo determinado; posteriormente, afrontamos a súa planificación e, finalmente, mentres está en marcha a fase de desenvolvemento, completamos as fases anteriores para os demais tipos de documentos. Deste xeito podemos conseguir acurtar o tempo que se tarda en comezar o traballo de desenvolvemento.

Outras veces, como xa dixemos, estes retrocesos teñen carácter obrigatorio debido a que foi detectado un erro que hai que corrixir. Por exemplo, podería darse o caso de que esteamos na fase de desenvolvemento procesando un documento e atopemos un problema na estrutura

textual elixida na fase de definición, debido a que aparezan elementos estruturais que non estaban contemplados inicialmente. O normal nesta situación sería volver á fase de definición da metodoloxía para refinar a estrutura que tiñamos prevista inicialmente, retocar os elementos necesarios na fase de construción e procesar o texto problemático novamente. As consecuencias deste tipo de problemas poden ser moi diversas e, nos casos máis graves, implicarían reprocesar moitos textos xa tratados con anterioridade.

De xeito transversal, resulta de vital importancia recompilar e rexistrar toda a información posible concernente a cada unha das fases propostas, que debe quedar patente nun ou varios documentos. A construción dun corpus adoita alongarse durante un período temporal amplo e, se non se anotan todas as decisións tomadas en cada momento, iranse esquecendo e haberá un alto risco de que xurdan incoherencias.

1.1 Definición da metodoloxía

Na fase de definición da metodoloxía concrétase cada un dos factores que afectan ao desenvolvemento do corpus (véxase a figura II.1.2). Pasamos a detallalos a seguir:

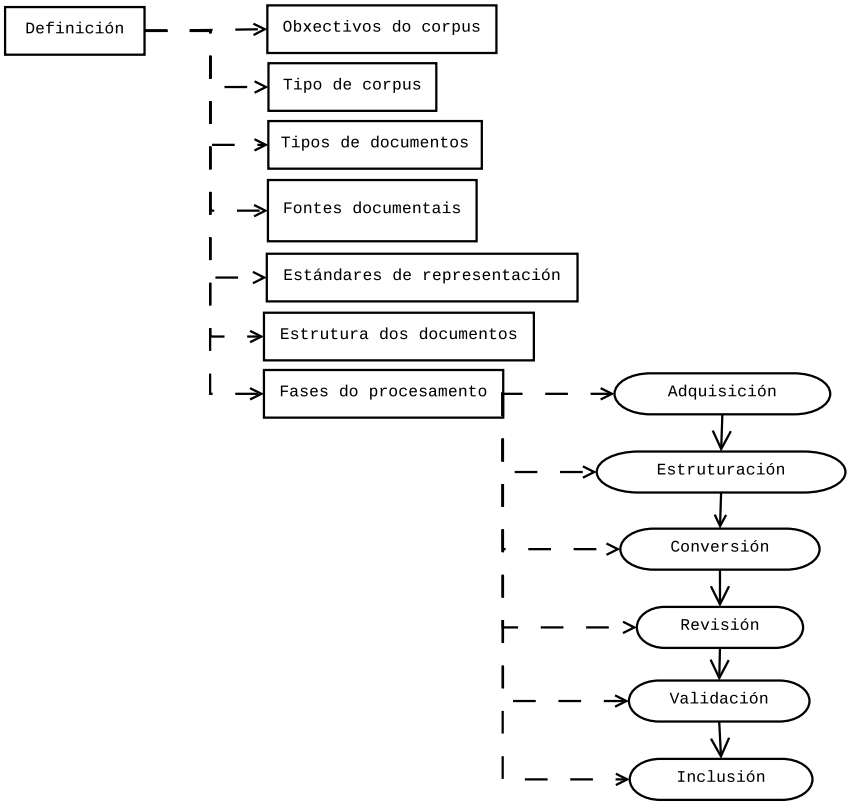

Figura II.1.2: Fase de definición da metodoloxía

1.1.1 Obxectivos do corpus

Os obxectivos que se perseguen co desenvolvemento dun corpus constitúen o primeiro factor que se debe ter en conta para definir a metodoloxía. As motivacións que levan a construír un corpus poden ser de moi diversa índole pero, habitualmente, enmárcanse dentro do obxectivo xeral de servir de soporte a algunha ferramenta que, ben permita facer diferentes estudos filolóxicos que o utilicen[1], ben posibilite facer tarefas que requiran o seu emprego[2].

Cando un equipo de traballo baralla desenvolver un corpus é porque previamente pensou nos obxectivos ou necesidades que se perseguen mediante a súa explotación. Deste xeito, o seu desenvolvemento non debe conformar un obxectivo final, senón que actúa como un medio grazas ao cal se poderán facer posibles os obxectivos reais, é dicir, os diferentes estudos filolóxicos, a tradución automática de textos etc.

Ademais, estes obxectivos serán determinantes á hora de definir os outros elementos da metodoloxía, xa que, por exemplo, dependendo do tipo de estudos que se queiran facer, o tipo de corpus será un ou outro e a estrutura dos documentos que o compoñan será dun xeito ou doutro, polo que se debe reflexionar concienzudamente sobre este factor para asegurármonos de non escoller un camiño erróneo desde o comezo.

Para a identificación dos obxectivos concretos resulta de utilidade pensar no uso que se vai facer do corpus e nas posibilidades, tanto das ferramentas existentes, como dun posible sistema de RI que se constrúa ad hoc, pero debe terse especial coidado en delimitarmos o alcance das nosas aspiracións. Inicialmente podemos pensar en multitude de obxectivos particulares, pero deberemos acoutalos e priorizalos para evitar o naufraxio cando se leven a cabo todos os traballos necesarios para o seu desenvolvemento.

1.1.2 Tipo de corpus

A segunda decisión consiste en determinar que tipo de corpus imos desenvolver. En función dos criterios que se utilicen, existen multitude de clasificacións que definen diferentes tipos de corpus [9, 31]. Baseándonos nestes traballos, e nalgunhas recompilacións [13, 66], incluímos aquí os tipos de corpus textuais que consideramos máis relevantes canto á influencia na definición da metodoloxía:

- **Corpus de referencia**: Constitúe unha mostra representativa das principais variedades dunha lingua, polo que xeralmente debe incluír gran cantidade de documentos para que ofreza información o máis ampla posible.
- **Corpus especializado**: Corpus que se crea para que sexa representativo dunha variedade lingüística específica ou lingua especializada, é dicir, deséñase con algún propósito específico.
- **Corpus monitor**: Orixinalmente [23] corpus de tamaño constante en que se van incluíndo novos materiais ao tempo que se eliminan outros máis vellos. Deste xeito ofrécese a posibilidade de observar cambios recentes no uso da lingua. Dados os avances tecnolóxicos actuais, xa non é necesaria a eliminación dos textos máis antigos.

[1] A consulta dalgún caso concreto de ocorrencia, a proba ou refutación dalgunha teoría, a sondaxe de casos que apoien ou rebatan algunha tese etc.

[2] Por exemplo, traballos relacionados coa tradución automática ou a corrección ortográfica.

- **Corpus de fragmentos textuais**: Corpus constituído por fragmentos de textos (fronte a corpus de textos completos).

- **Corpus de textos completos**: Os textos inclúense na súa totalidade.

- **Corpus bilingüe ou multilingüe**: Corpus que contén textos de dúas ou máis linguas relacionados dalgún xeito. Poden ser textos que estean traducidos a varias linguas (corpus paralelo), ou textos que compartan algunhas características (corpus comparable).

O tipo de corpus que construamos tamén condicionará a materialización dalgúns elementos da metodoloxía. As explicacións dadas ao longo deste capítulo baséanse na construción de corpus textuais monolingües constituídos por textos completos estruturados utilizando XML. Aínda que estas tamén son válidas para outros tipos de corpus textuais, nalgúns casos deben adaptarse e/ou completarse para que a metodoloxía se adecúe totalmente ao seu desenvolvemento.

1.1.3 Tipos de documentos

Antes de definirmos a estrutura dos documentos que serán traballados é importante decidir que tipos de textos imos incorporar ao corpus. Por iso, haberá que ter en conta o xénero (literario, xornalístico etc.), as áreas temáticas, o soporte da publicación (revista, libro ou xornal) e outros criterios específicos de relevancia para o conxunto.

Algúns criterios que se deben considerar para determinar os tipos de documentos que se utilizarán son os seguintes:

- **Lingua**: O corpus estará constituído por textos dunha soa lingua (monolingüe)? Cal? Por varias (multilingüe)? Que linguas abarcará?

- **Variedade lingüística**: Necesitamos representar a variedade lingüística ou, pola contra, buscamos documentos con algunha característica que os singularicen lingüisticamente?

- **Períodos**: Que franxa temporal queremos abranguer coa selección dos textos?

- **Temática**: Empregamos documentos de temática variada ou específica?

- **Xénero**: É pertinente seleccionarmos textos de diferentes xéneros?

- **Tipo de publicación**: Centrámonos nun determinado tipo de publicación ou temos diferentes tipos?

- **Autor**: Trátase dun corpus constituído por documentos dun único autor ou de varios? De autores que cumpran algún tipo de característica común? É irrelevante para o corpus quen sexan os autores dos documentos?

Todos estes criterios, e outros particulares que quedan fóra da análise aquí desenvolvida, afectarán e condicionarán a nosa decisión sobre que tipo de documentos incorporar ao corpus. É necesario analizar polo miúdo cada un dos elementos clave para que a decisión final sexa acertada.

1.1.4 Fontes documentais

Debe confeccionarse unha listaxe inicial das fontes de que tiraremos o material. Haberá que comprobar que, efectivamente, se poderán obter polas canles que determinemos, e que respectaremos en todo momento a lei de propiedade intelectual cando os procesemos e os incluamos no corpus.

Obteranse xa neste momento algúns dos documentos que pretendemos incluír no corpus. Deberemos conseguir, como mínimo, un ou dous de cada tipo, que utilizaremos como modelos orientativos á hora de definir a estrutura.

1.1.5 Estándares de representación

Para o desenvolvemento de corpus utilizando XML existen dúas tendencias principais:

1. Definir un XML propio para a representación de documentos.
2. Utilizar algún dos estándares dispoñibles como, por exemplo, XCES [92] ou TEI [89].

A decisión de optar por unha das aproximacións depende de varios factores. Como norma xeral, resulta de máis utilidade empregar algún estándar xa definido, xa que, ao seren moitos os colectivos que utilizan ese estándar, presenta certas vantaxes:

1. Maior compatibilidade entre corpus, xa que se facilita o intercambio de información entre proxectos diferentes.
2. Mellórase a formación dos profesionais. Minimízase o impacto do custo desta cando unha persoa se cambia dun proxecto a outro ao se propiciar a mobilidade entre proxectos. Do mesmo xeito, favorécese a existencia de cursos de formación, alleos aos diferentes proxectos, onde se expliquen eses estándares.
3. Optimízase o soporte de ferramentas. Ao haber un estándar común, pódense desenvolver ferramentas para seren utilizadas en diferentes proxectos.

Porén, existen casos onde un desenvolvemento propio da estrutura dos documentos pode resultar vantaxoso:

- Se chegamos á conclusión de que é mellor para o proxecto que as propias etiquetas XML estean nunha lingua diferente do inglés. Por exemplo, a lingua de traballo do persoal do proxecto.
- Se o estándar non permite facer un refinamento da estrutura que propón, e non queremos que haxa máis etiquetas das que xustamente nós necesitamos.

En calquera caso, e aínda que nos decidamos pola segunda opción, resulta de utilidade estudar e analizar a estrutura documental que propoñen os estándares para poder definir axeitadamente a nosa particular. Eses estándares son creados por expertos, o que normalmente garante que as decisións que neles se toman sexan acertadas.

Nas explicacións posteriores suporemos que nos decantamos pola opción de definir un XML propio, que é a máis complexa. Todas as explicacións serán tamén válidas no caso de utilizarmos algún estándar, e puntualizaremos aquelas en que isto non ocorra.

1.1.6 Estrutura dos documentos

Unha vez temos claros os obxectivos, o tipo de corpus, os tipos de documentos que imos incorporar e a tecnoloxía de representación que utilizaremos e, posuíndo xa, algúns exemplos de textos candidatos a formaren parte do corpus, é necesario definir a súa estrutura. Nun principio convén pensar en todas as posibilidades de uso futuras do corpus, xa que é conveniente non esquecerse de considerar ningún elemento que sexa necesario nun tratamento posterior. Porén, tampouco é recomendable afrontar absolutamente todas as posibilidades e detalles para utilidades que vaian ser afrontadas a moi longo prazo desde un comezo. Normalmente intentar abranguelo todo desde o principio leva asociado que o desenvolvemento sexa demasiado lento, e que non se aprecien avances significativos durante moito tempo. Iso si, para permitir posibles variacións futuras sen un custo excesivo, deben terse en conta estes aspectos durante a concepción da estrutura dos documentos e amoldala de maneira consecuente.

O principio básico que se debe seguir para estruturar axeitadamente os documentos dun corpus consiste en que hai que dividir ou marcar os elementos que sexa necesario tratar, manipular ou recuperar. Por exemplo, se queremos construír un corpus en que, mediante unha ferramenta ou un sistema de RI, poidamos facer buscas de topónimos nel, estes deberán estar marcados dalgún xeito dentro dos textos.

Na maioría das ocasións, aos deseñadores ocórrenselles moitas estruturas diferentes para os documentos. Escoller a máis axeitada non é doado, pero serve de axuda ter en conta o seguinte:

- **A presenza física do documento**: Dado que en moitas ocasións os documentos están orixinariamente impresos en papel, esa aparencia pode levarnos a non estuturalos axeitadamente.

  É común que nunha primeira análise sobre cal debería ser a estrutura dos textos, se chegue á conclusión de que estes están organizados en páxinas e estas, á súa vez, en liñas. Malia que esta forma de división nalgúns casos está xustificada, porque, por exemplo, son moi importantes os datos exactos de localización na edición, o normal é que presente máis inconvenientes que vantaxes no transcurso do desenvolvemento do corpus.

  Para chegar a unha boa solución de compromiso deben terse en conta todas as variantes de representación dos documentos, así como as diferentes técnicas para levalas a cabo. Por exemplo, no caso de utilizar a tecnoloxía XML, que é o caso que nos ocupa, existen varias técnicas para representar múltiples xerarquías [30, 89]. A aplicación destas técnicas permitiríanos considerar información adicional de xeito que non se vexa modificada a estrutura principal dos documentos.

  En relación ao exemplo que acabamos de introducir, podemos supoñer que o noso corpus formará parte dun sistema de RI. Imaxinemos que nese sistema se queren amosar os exemplos de ocorrencia dunha cadea de busca xunto coa páxina e a liña en que aparece, pero a nós parécenos interesante que os documentos estean organizados en capítulos e non en páxinas e liñas. Utilizando estas técnicas poderíanse incluír etiquetas en liña na estrutura dos documentos onde haxa saltos de páxina ou liña, que nos permitan obter o sistema desexado sen perder a estrutura que cremos máis axeitada para estes.

- **O sistema de** RI: En moitos casos, a finalidade do corpus que deseñamos é a de construír un sistema de RI que permita facer buscas sobre el. Como apuntamos noutro traballo

anterior [11], a estrutura dos documentos para este sistema non só pode ser diferente á axeitada para o corpus, senón que moitas veces é recomendable que así sexa.

Polo tanto, aínda que si é bo ter presente a estrutura dos documentos que teñamos pensada para o sitema de RI, non debemos deixar que inflúa demasiado na nosa decisión. Se así for tenderiamos a simplificar demasiado os documentos e perderiamos expresividade e flexibilidade.

Dado que a organización interna dos documentos é crucial para un desenvolvemento acertado do corpus, debemos prestar especial atención a non cometer ningún erro grave de deseño neste punto. Se a estrutura definida inicialmente é boa, non a volveremos refinar, ou só o faremos en cuestións moi puntuais de pouca importancia. Cando se atopa algún erro, normalmente é cando xa estamos en fases avanzadas, mesmo cando xa temos textos completamente procesados, e ás veces é bastante custoso rectificar. A magnitude deste custo dependerá, tanto da importancia do erro detectado, como do automático que for o proceso de arranxo, que normalmente incluirá un refinamento na estrutura dos documentos.

Aínda que a utilización dun estándar facilita un pouco a tarefa, tamén hai que decidir cales dos elementos que define este son os que imos utilizar para os diversos tipos de documentos. Os estándares definen elementos de estruturación para unha variedade ampla de textos pero deberemos utilizar soamente os que precisemos.

1.1.7 Fases do procesamento

Nesta epígrafe defínense as diferentes fases polas que pasarán os documentos, desde o seu formato e medio orixinal ata adaptalos á estrutura electrónica que teñamos definida para o corpus. Para facelo, deberemos ter en conta:

- **O medio en que se atopa o documento**: Se o documento orixinal está en papel ou se o temos en soporte electrónico.
- **O seu formato**: Dentro dos documentos que están impresos en papel referímonos á súa maquetación (tamaño de páxina, dúas caras nunha ou non ou catacterísticas relacionadas co encolumnado), e nos que se atopan en soporte electrónico, á clase de arquivo en que os temos (PDF [1], HTML [98], ODT [61] etc.).
- **O tipo de documento**: Dependendo das clases de textos que teñamos definidos, o procesamento dun tipo vai ser algo diferente ao doutros, aínda que poidan ter bastantes transformacións comúns.

Para cada medio, formato e tipo de texto cómpre definir e documentar un protocolo de actuación que describa polo miúdo as diferentes fases, tanto as automáticas como as manuais, que van seguir os textos para que pasen a formar parte do corpus. O protocolo será útil de cara a que queden patentes todas as decisións tomadas e a que se formen novos membros do equipo de lingüistas que aplicará as tarefas manuais.

Adoita ocorrer que, co paso do tempo, poden esquecerse algúns detalles destas fases. A existencia dos protocolos evita volver discutir decisións que xa se tomaron no seu momento e, por tanto, atrasos e incoherencias no avance do proxecto.

Entre as diferentes fases que se inclúen neste procesamento podemos distinguir as manuais, que serán executadas polo equipo de linguistas na fase de desenvolvemento utilizando ferramentas informáticas, e as automáticas, que serán realizadas por *scripts*[3] creados polo equipo informático.

Identificamos seis fases de procesamento xenéricas[4], que se amosaron na figura II.1.2 e describimos a continuación:

Adquisición

Trátase dunha fase manual que consiste en adquirir unha versión electrónica do documento que se vai procesar. A dificultade desta tarefa pode ser moi dispar, dependendo, principalmente, de se partimos de textos impresos en papel ou dixitais.

Incorporar un xornal dixital accesible a través da web pode ser relativamente sinxelo. Abondaría con descargar as páxinas que o constitúen (xa sexa en formato HTML ou texto). Máis sinxelo aínda sería que algunha editorial nos cedese unha versión electrónica do que queremos incorporar (un ODT, un PDF...). Neste caso habería que comprobar que o documento dixital coincide exactamente coa versión impresa que queremos procesar, pero xa disporiamos directamente da versión electrónica.

Porén, un proceso máis laborioso sería incorporar un texto que só estiver dispoñible en papel. Neste caso habería que utilizar un programa de escaneamento e ir metendo, páxina a páxina, todo o contido no formato dixital que corresponda. Así, ao finalizar a adquisición, o documento estaría constituído, ou por un arquivo no formato da ferramenta de escaneamento, ou por un conxunto de imaxes con cada unha das súas páxinas.

Estruturación

É unha fase manual que consiste en estruturar, nun formato común que debe ser definido e documentado para cada tipo de texto, os documentos dixitais que resultaron da adquisición. Este formato facilitará o traballo que teñen que realizar os equipos informático e lingüístico tanto nesta fase como nas seguintes.

Normalmente, canto máis sinxelo for, maior será o traballo requirido nas fases posteriores, e viceversa. Polo tanto, o criterio máis importante á hora de definilo é conseguir un bo compromiso entre o traballo que supón crealo nesta fase, e o traballo que implicará manexalo posteriormente.

Aínda que este pode variar dun sistema de traballo a outro, unha posible proposta sería utilizar unha organización en directorios con arquivos de texto. Por exemplo, para estruturar un xornal poderiamos ter un directorio co nome e a data do xornal. Nel podemos crear un subdirectorio para cada unha das seccións do xornal e dentro as noticias, constituídas por un arquivo co texto e outro cos datos sobre os autores.

Conversión

Consiste nunha transformación automática dos documentos que resultaron da fase de estruturación co obxectivo de adaptalos ao formato XML que se definiu con anterioridade (sección

[3] Programas informáticos que, neste caso, realizan transformacións nos documentos.

[4] Para cada corpus pode haber variacións e/ou modificacións dalgunha(s) das fases, ou incluso omisión doutra(s). Neste caso, propoñemos unhas fases xenéricas que se poden adaptar para a súa utilización en diferentes tipos de corpus e sistemas de traballo.

1.1.6). Xa que logo, será necesario crear algún *script* que faga esta conversión. O resultado desta fase será unha primeira versión do documento en XML.

Revisión

Dada a dificultade das tarefas que ten que facer o *script* da fase anterior, e tendo en conta a solución de compromiso que se debe tomar na fase de estruturación para o formato intermedio, debemos supoñer que este non será capaz de representar adecuadamente todos os fenómenos necesarios no formato XML definido para os documentos do corpus.

Novamente, resulta de vital importancia atopar unha boa solución de compromiso para o formato común xa que, canto máis detallado for este formato, menor será o traballo a realizar na fase de revisión[5] e viceversa.

Tamén se deben definir agora os protocolos de actuación que seguirá o equipo de lingüistas na revisión dos textos para adecualos á estrutura deseñada. A existencia dos protocolos é de vital importancia para manter a coherencia do corpus ao longo do tempo e, neles, deben quedar reflectidas pormenorizadamente todas as revisións que van realizar as persoas involucradas e a orde en que se han executar.

Validación

Hai ocasións en que a ferramenta que se utiliza para levar a cabo a fase de revisión non permite facer todas as comprobacións que desexamos antes de que os documentos pasen a formar parte do corpus. É por isto polo que resulta de utilidade definir unha fase de validación onde os documentos xa revisados se comproben, ben a través de ferramentas xa existentes, ben mediante *scripts* automáticos construídos ad hoc.

No caso de que se detecte algún problema nesta fase, volverase á anterior para solucionar o problema e proceder novamente a validar o documento.

Inclusión

Unha vez que se comproba que un documento cumpre todos os requirimentos necesarios para formar parte do corpus, hai que incluílo no lugar que teñamos definido para o caso.

1.2 Construción da metodoloxía

Unha vez definida a metodoloxía de traballo é necesario:

- Adquirir e/ou construír todas as ferramentas e os recursos necesarios para poder levala a cabo.
- Instalar e configurar estas ferramentas, así como o contorno de traballo das persoas involucradas.

Trátase de ter todos os elementos necesarios dispostos para poder afrontar o desenvolvemento do corpus. A continuación, distinguimos varias áreas de actuación sobre as que traballar.

[5] Debido a que o *script* responsable de realizar a conversión poderá estruturar mellor os documentos XML.

1.2.1 DTD ou esquemas XML

Dado que xa se definiu a estrutura que terán os documentos na fase de definición da metodoloxía (1.1.6), será necesario construír unha ou varias DTD ou esquemas XML que reflictan esa estrutura. No caso de empregar un estándar habería que escoller as DTD ou esquemas XML que utilizaremos e, de ser o caso, crear outras que inclúan o subconxunto de etiquetas que desexemos usar. Pode ser que na fase de definición da metodoloxía xa se empregaran estes estándares para definir a estrutura dos diferentes documentos. No caso contrario, é nesta fase onde se deben concretar.

Aínda que os documentos XML poderían non ter DTD ou esquema asociado, resulta de utilidade especificalos e agrupar os textos dun mesmo tipo baixo a mesma definición. Deste xeito poderase comprobar máis facilmente que os documentos cumpren co formato que temos definido, evitaranse erros na súa estruturación e facilitarase a creación de calquera tipo de aplicación que os manexe.

Canto á decisión de utilizar DTD ou esquemas, será mellor traballar con esquemas sempre e cando atopemos ferramentas que funcionen correctamente con eles. Estes permiten especificar un maior detalle nas definicións que as DTD, ao poderen por exemplo, restrinxir o tipo de dato que debe conter cada etiqueta dos documentos.

1.2.2 Adquisición dos documentos

Para levar a cabo a fase de adquisición descrita na sección 1.1.7 é necesario adquirir e configurar as ferramentas que permiten obter unha versión electrónica dos documentos. Como apuntamos ao comezo deste capítulo, nalgunhas ocasións xa dispoñemos dunha versión dixital do documento, polo que non é necesaria ningunha ferramenta especial que nos permita realizar esta tarefa. En cambio, noutras moitas, teremos que adquirir nós esa versión, ben cando o documento orixinal estiver en papel, ben cando teñamos que descargalo dalgunha páxina web. Xa que logo, debemos adquirir, instalar e configurar dous tipos de ferramentas que nos axuden neste proceso:

- Unha aplicación de descarga de webs que permita obter os documentos en liña, sobre todo cando están formados por varios arquivos HTML.
- Unha ferramenta de escaneamento que permita procesar documentos en papel para obter a súa versión dixital.

Para lle facilitar o traballo ao equipo de lingüistas tamén resulta de utilidade crear modelos na aplicación de escaneamento que permitan cargar automaticamente os parámetros do escáner de acordo coa diferente aparecia dos textos: a dobre cara, dúas caras na mesma folla, follas grandes, follas pequenas, dúas columnas etc.

1.2.3 Edición

Para realizar as tarefas da fase de estruturación son necesarias ferramentas que axuden ao equipo de lingüistas a transformar a versión electrónica dos documentos, resultado da fase de adquisición, no formato común de representación. En particular, é imprescindible dispoñer dun editor de textos axeitado para realizar esas tarefas e dun software de recoñecemento óptico de

caracteres (*Optical Character Recognition*, OCR) que posibilite realizar a extracción do texto no caso de que se traballe con documentos escaneados.

Ademais, tamén na fase de revisión é necesaria algunha ferramenta que nos permita revisar e corrixir os posibles problemas que aparecen despois da fase de conversión. Neste caso é necesario un editor de documentos XML que permita manipulalos ata deixalos na súa versión definitiva.

Para facer o manexo do editor XML máis sinxelo, resulta de utilidade desenvolver algún tipo de personalización da vista dos documentos no editor, por exemplo, a través de follas de estilos en cascada (*Cascade Style Sheets*, CSS) [96] que melloren a visualización e simpliquen a manipulación.

1.2.4 *Scripts*

A realización de *scripts* afecta, principalmente, á fase de conversión, descrita na sección 1.1.7. Deben desenvolverse os que, a partir do formato común resultado da fase de estruturación, xeren unha primeira versión do documento XML que poida ser revisada e manipulada manualmente ata obter o resultado definitivo.

Adicionalmente, tamén resulta de utilidade desenvolver *scripts* que validen a versión final do documento. Estes complementarían as comprobacións que xa fai o editor XML sobre a DTD ou os esquemas, e cubrirían os aspectos que non poidan afrontarse desa maneira.

1.2.5 Recursos económicos, humanos e hardware

A dispoñibilidade de recursos económicos, humanos e hardware é un factor clave á hora de construír un corpus. Debemos comprobar se temos os recursos necesarios para desenvolvelo e, en caso contrario, adquirilos nesta fase: mercar os ordenadores, contratar as persoas que realizarán o desenvolvemento que deben ir formándose nas diferentes tarefas que deberán realizar etc.

1.2.6 Almacenamento

Na fase de construción da metodoloxía, e con respecto ao almacenamento, hai que responder dúas preguntas que están relacionadas entre si:

1. Onde estarán almacenados os arquivos.

2. Que arquivos se conservarán e por canto tempo.

Con respecto á primeira cuestión, a decisión dependerá moito da infraestrutura informática con que se conte. Se se dispón dun acceso a un disco común entre os diferentes ordenadores de traballo, resulta de utilidade definir unha estrutura de directorios que debe quedar reflectida nos protocolos de actuación. Por exemplo, podemos crear un directorio para as DTD ou os esquemas, outro para os documentos con que se está traballando e aínda non están rematados, e un terceiro para os que xa están corrixidos, ou incluso complementar esta clasificación coa subdivisión interna en clases de documentos.

Con respecto á segunda, debe terse en conta que existen diferentes versións de cada documento. Tense a versión inicial froito da fase de adquisición, a versión correspondente ao resultado da estruturación, a que se obtén despois da fase de conversión e, finalmente, a definitiva, resultado da fase de revisión.

Nun primeiro momento téndese a pensar que só se necesita almacenar e xestionar a versión definitiva. Porén, existen casos onde é moi útil dispoñer das diferentes versións dos documentos:

- Se nalgunha das fases de procesamento atopamos algo que parece ser un erro do OCR, e queremos comprobar se efectivamente é así. Pode tratarse dun erro que xa estaba no documento orixinal ou introducido por algún revisor nalgunha fase. Mantendo as diferentes versións poderiamos localizar cal é a orixe do erro.

- Se estamos afrontando a fase de conversión dun grupo de documentos e, nese momento, non é posible acometer a fase de revisión de todos eles[6]. Posteriormente, pódese detectar que os *scripts* involucrados na conversión non actuaron axeitadamente, e temos que modificalos para refacer a conversión dalgúns dos documentos que xa foron revisados. Neste caso, se non dispoñemos dos documentos resultantes da fase de estruturación, habería que repetir o traballo feito nela, o cal tería un custo moi elevado.

Polo tanto, é desexable manter unha copia de todas as versións dos documentos indefinidamente pero, dado que algunha destas versións poden ocupar demasiado espazo, hai que chegar a unha solución de compromiso sobre qué se almacena e por canto tempo.

Por outra banda, os diferentes tipos de soportes axudarán a manter unha política de almacenamento aceptable, xa que podemos manter algúns arquivos nun soporte en liña, como pode ser o disco duro, durante certo tempo e pasalos posteriormente a un soporte fóra de liña, como o CD ou o DVD, por tempo indefinido.

En definitiva, é na fase de construción da metodoloxía cando se definen os conceptos relacionados coas políticas de almacenamento e seguridade dos diferentes arquivos que xorden no procesamento dos documentos.

1.2.7 Outras áreas

Aínda que xa analizamos algúns dos elementos clave que se deben ter en conta na construción da metodoloxía, existen outros que quedan fóra dos obxectivos deste traballo, pero que poden ser de utilidade en función das necesidades e requirimentos particulares de cada equipo de desenvolvemento de corpus.

Un exemplo claro pode constituílo a utilización dun navegador web. Este pode ser necesario, por exemplo, na fase de estruturación se traballamos con documentos que son páxinas HTML obtidas en Internet. Unha vez descargados os contidos, coas funcións de copiado e pegado, tanto do navegador como do editor de texto, pódese ir creando o formato intermedio resultado desta fase.

1.3 Comprobación da metodoloxía

Antes de metérmonos no desenvolvemento do corpus propiamente dito, compre facer unhas probas exhaustivas para asegurármonos de que a metodoloxía funciona correctamente. Nelas só intervirán algúns membros do equipo de desenvolvemento, recomendablemente os coordinadores

[6] Por problemas de que os recursos humanos estean dedicados a outras tarefas ou, simplemente, porque non se poden revisar todos os documentos á vez.

da parte lingüística e informática, que revisarán todos os elementos que a compoñen: programas para a adquisición dos documentos, *scripts*, editores etc.

O mellor xeito de levar isto a cabo é collermos un ou varios documentos de cada tipo e aplicarlles todas as fases de actuación. Desta forma iremos detectando problemas nos diferentes procesos de transformación de documentos: *scripts* que teñen imperfeccións, problemas co manexo e configuración das ferramentas de edición etc. Se percibimos algunha anomalía, corríxese, e volven realizarse as probas ata que todos os compoñentes funcionen correctamente.

Como xa se apuntou na fase de definición da metodoloxía, hai que poñer especial atención en comprobar o correcto funcionamento de todos os procesos e ferramentas. Se comezamos coa fase de desenvolvemento sen ter ben probada a metodoloxía, pode ocorrer que aparezan erros en etapas avanzadas e que isto supoña levar a cabo unha corrección moi custosa.

1.4 Planificación temporal

Chegados a este punto débese facer unha escolla de textos entre as fontes documentais preseleccionadas na definición e planificar o seu procesamento ao longo do tempo. Os mesmos criterios que usamos para definir os tipos de documentos son os que empregramos para planificar a súa incorporación ao corpus. A modo de exemplo pódese considerar:

- O **período**: En moitos corpus un factor importante é a data de publicación ou de creación dos documentos. Pode resultar interesante comezar por documentos dun determinado período para que o corpus vaia collendo consistencia nesas datas. Así, obteriamos un corpus moi completo para a análise dunha época concreta.

  Porén, tamén pode ser interesante escoller uns poucos textos de cada período, se o que buscamos é obter rapidamente unha maneira de facer comparacións entre épocas.

- O **tipo de publicación**: Pode ser interesante comezar polos documentos dun tipo de publicación, de entre todos os que decidimos incorporar no corpus. Isto é útil, por exemplo, para economizar a formación, xa que adquirir destreza na manipulación e transformación dun único tipo de documento é máis sinxelo que tratar simultaneamente varios.

  Se o que se busca é obter rapidamente unha gran variedade tipolóxica, haberá que coller soamente uns poucos textos de cada clase, aínda que o custo de formación e probablemente a calidade do procesamento non sexan óptimos.

En definitiva, é necesario planificar o desenvolvemento do corpus e, por suposto, esta planificación debe irse adaptando a medida que avance o proxecto.

1.5 Desenvolvemento

Finalmente, na fase de desenvolvemento aplícase a metodoloxía definida sobre os diferentes documentos que se decidiron procesar na planificación. Nesta fase xa son todos os membros do equipo de desenvolvemento os que participan na incorporación de textos ao corpus, e cobra especial relevancia a realización de reunións periódicas que permitan unificar criterios e resolver conxuntamente as dúbidas que xurdiren.

Capítulo 2

O corpus CORGA

Neste capítulo exporemos como aplicamos a metodoloxía anterior ao noso caso de estudo: o corpus CORGA. Para simplificar as explicacións, neste capítulo faise referencia ás seccións 1.1 e 1.2 do anterior conxuntamente, é dicir, comentamos as particularidades da metodoloxía ao tempo que concretamos como desenvolvemos cada unha das partes.

2.1 Definición e construción da metodoloxía

O obxectivo principal do CORGA é o de servir como referente para os estudos, principalmente léxicos e morfolóxicos, da lingua galega nos medios escritos actuais. Polo tanto, un elemento crucial para posibilitar o acceso ao corpus é o desenvolvemento dalgún tipo de ferramenta accesible á comunidade que permita a realización de diferentes tipos de consultas.

O CORGA encaixa perfectamente nun dos tipos de corpus mencionados na sección 1.1.2, o corpus de referencia, xa que pretende ser unha mostra representativa das principais variedades da lingua escrita. Está formado unicamente por textos en galego publicados desde o ano 1975, xa que é o ano en que se comezan a albiscar indicios de normalización en diferentes eidos. Dado que se pretende recoller a riqueza lingüística desde esa data, non hai restricións canto a xénero, temática, tipo de publicación ou autor. Non se inclúen varias edicións dun mesmo documento e, nos casos en que é posible, incorpórase sempre a primeira edición dos que teñan varias.

Dado que o desenvolvemento do proxecto comezou no ano 1995 e que as tecnoloxías evolucionaron enormemente na última década, a técnica de representación documental variou desde a súa concepción inicial. Nun comezo, os documentos estaban divididos en dous arquivos: un cos datos da cabeceira do documento (autor, ano de publicación, editorial etc.) e outro co texto.

As dificultades que esta estrutura producía, tanto á hora de construír un sistema de buscas[1], como á de incluír unha codificación adicional de elementos que puidera permitir consultas moito máis versátiles, leváronnos a acometer unha transformación ao estándar XML no ano 2002 que perdura na actualidade. A continuación, centrarémonos na descrición do sistema actual de codificación, e deixaremos para un pouco máis adiante as explicacións relacionadas coa transformación dos documentos antigos ao novo estándar.

Canto ás fontes documentais, inicialmente seleccionáronse algunhas de fácil acceso para favorecer o arranque do proxecto.

[1] Debido, principalmente, a que non se podían considerar elementos aniñados.

2.1.1 Estrutura dos documentos

Despois dun estudo pormenorizado dos estándares XCES e TEI, decidimos definir un XML propio para representar a estrutura dos documentos. Isto permitiunos utilizar etiquetas en galego e, acelerar así, a formación dos lingüistas que foron formando parte do proxecto.

Finalmente, consideramos seis tipos de documentos: xornais, revistas, libros, coleccións de relatos ou ensaios, obras teatrais e coleccións de obras teatrais. Para estas seis clases definimos cadansúa DTD, que reflicte a súa estrutura interna. Nas figuras II.2.1 e II.2.2 amósanse algunhas definicións comúns a todas as DTD e, nas figuras II.2.3, II.2.4, II.2.5, II.2.6, II.2.7 e II.2.8, as correspondentes a cada un dos tipos de documentos.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- Elementos comúns para todas as dtds. -->

<!-- Elementos por enriba de parágrafo. -->

<!ELEMENT identificador (#PCDATA)>
<!ELEMENT medio (#PCDATA)>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT lingua (#PCDATA)>
<!ELEMENT soporte (#PCDATA)>
<!ELEMENT ano_publicación (#PCDATA)>
<!ELEMENT lugar_publicación (#PCDATA)>
<!ELEMENT editorial (#PCDATA)>
<!ELEMENT depósito_legal (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT issn (#PCDATA)>
<!ELEMENT número_edición (#PCDATA)>
<!ELEMENT ano_primeira_edición (#PCDATA)>
<!ELEMENT versión_electrónica (#PCDATA)>
<!ELEMENT ano_versión (#PCDATA)>
<!ELEMENT responsable_creación (#PCDATA)>
<!ELEMENT responsable_versión (#PCDATA)>
<!ELEMENT comentarios (#PCDATA)>
<!ELEMENT título (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT área_temática (código, subcódigo)>
<!ELEMENT código (#PCDATA)>
<!ELEMENT subcódigo (#PCDATA)>
<!ELEMENT sistema_ortográfico (#PCDATA)>
<!ELEMENT dedicatoria (parágrafo+)>
<!ELEMENT cita (parágrafo+)>
<!ELEMENT pé_de_foto (parágrafo+)>
<!ELEMENT nota (parágrafo+)>
<!ATTLIST nota identificador ID #IMPLIED
               tipo (referencia_bibliográfica) #IMPLIED>
```

Figura II.2.1: Definicións comúns a todos os tipos de documentos (I)

Se nos fixamos nas ditas figuras, todas as clases de documentos comparten unha característica común: teñen unha cabeceira xeral e un contido. A partir desta característica compartida especialízanse segundo a información que conteñan en cada caso. Ao longo das seguintes seccións explicamos o significado dos elementos da estrutura, así como as diferenzas principais entre os diferentes tipos de documentos.

```
<!-- Elementos por debaixo de parágrafo -->

<!ENTITY % valores_distinto "outra_lingua|outro_período|non_normativo|descoñecido">
<!ENTITY % texto "#PCDATA|referencia_nota|fórmula|novo_verso|inicio_poema|fin_poema">

<!ELEMENT fórmula EMPTY>
<!ELEMENT novo_verso EMPTY>
<!ELEMENT inicio_poema EMPTY>
<!ELEMENT fin_poema EMPTY>

<!ELEMENT referencia_nota EMPTY>
<!ATTLIST referencia_nota referencia IDREF #REQUIRED>

<!ELEMENT distinto (%texto;)*>
<!ATTLIST distinto tipo (%valores_distinto;) #REQUIRED>

<!ELEMENT oración (%texto;|distinto)*>
<!ATTLIST oración distinto (%valores_distinto;)
                  tipo (acoutación) #IMPLIED>

<!ELEMENT parágrafo (oración+)>
<!ATTLIST parágrafo interlocutor (Entrevistador|Entrevistado) #IMPLIED
                    nome_interlocutor CDATA #IMPLIED
                    distinto (%valores_distinto;) #IMPLIED
                    tipo (acoutación) #IMPLIED
                    referencia IDREF #IMPLIED>
```

Figura II.2.2: Definicións comúns a todos os tipos de documentos (II)

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- Definición dos elementos comúns a todas as dtds. -->

<!ENTITY % comuns SYSTEM "comuns.ent">
%comuns;

<!-- Estrutura dun xornal -->

<!ELEMENT documento (cabeceira_documento, contido_documento)>

<!ELEMENT cabeceira_documento (identificador, medio, nome, lingua, soporte,
                               ano_publicación, lugar_publicación, editorial,
                               depósito_legal?, issn?, isbn*, número_edición,
                               versión_electrónica, ano_versión, responsable_creación,
                               responsable_versión, comentarios?)>

<!ELEMENT contido_documento (sección+)>
<!ELEMENT sección (nome, noticia+)>
<!ELEMENT noticia (cabeceira_noticia, contido_noticia)>
<!ELEMENT cabeceira_noticia (identificador, autor+, área_temática+,
                             sistema_ortográfico, comentarios?)>

<!ELEMENT contido_noticia (titular?, resumo?, pé_de_foto*, corpo)>
<!ELEMENT titular (parágrafo+)>
<!ELEMENT resumo (parágrafo+)>
<!ELEMENT corpo (parágrafo+, nota*)>
```

Figura II.2.3: DTD de xornal

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- Definición dos elementos comúns a todas as dtds. -->

<!ENTITY % comuns SYSTEM "comuns.ent" >
%comuns;

<!-- Estrutura dunha revista -->

<!ELEMENT documento (cabeceira_documento, contido_documento)>

<!ELEMENT cabeceira_documento (identificador, medio, nome, lingua, soporte
                               ano_publicación, lugar_publicación, editorial,
                               depósito_legal?, issn?, isbn*, número_edición,
                               versión_electrónica, ano_versión, responsable_creación,
                               responsable_versión, comentarios?)>

<!ELEMENT contido_documento (noticia+)>
<!ELEMENT noticia (cabeceira_noticia, contido_noticia)>
<!ELEMENT cabeceira_noticia (identificador, autor+, área_temática+,
                             sistema_ortográfico, comentarios?)>

<!ELEMENT contido_noticia (titular?, resumo?, pé_de_foto*, corpo)>
<!ELEMENT titular (parágrafo+)>
<!ELEMENT resumo (parágrafo+)>
<!ELEMENT corpo (parágrafo+, nota*)>
```

Figura II.2.4: DTD de revista

2.1.2 Xornais

Se observamos detidamente a DTD da figura II.2.3, podemos apreciar que un xornal ten unha cabeceira xeral cos seguintes campos:

- **identificador**: Identificador único para o xornal. Consta das iniciais do nome do xornal seguidas do ano, o mes e o día da publicación, con cada compoñente da data separado cun guión. Así, por exemplo, para o xornal *O Correo Galego* teriamos o identificador CGAAAA-MM-DD, onde CG son as iniciais do nome e AAAA, MM e DD son o ano, o mes e o día da publicación do xornal.
- **medio**: Especifica o medio de publicación, neste caso Xornal.
- **nome**: O nome do xornal. Por exemplo, *O Correo Galego*.
- **lingua**: Contén Galego.
- **soporte**: Pode conter os valores Papel ou Electrónico, dependendo de se o xornal está obtido a partir dunha edición en papel ou electrónica.
- **ano_publicación**: Ano de publicación do xornal.
- **lugar_publicación**: Lugar de publicación.
- **editorial**: Editorial.
- **depósito_legal**: Depósito legal das publicacións que o teñan. É un campo opcional.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- Definición dos elementos comuns a todas as dtds. -->

<!ENTITY % comuns SYSTEM "comuns.ent">
%comuns;

<!-- Estrutura dun libro -->

<!ELEMENT documento (cabeceira_documento,contido_documento)>

<!ELEMENT cabeceira_documento (identificador, medio, título, autor+, lingua,
                               soporte, ano_publicación, lugar_publicación,
                               editorial, depósito_legal?, issn?, isbn*,
                               número_edición, ano_primeira_edición, área_temática+,
                               versión_electrónica, ano_versión, responsable_creación,
                               responsable_versión, sistema_ortográfico, comentarios?)>

<!ELEMENT contido_documento (prólogo*, dedicatoria*, cita*, pé_de_foto*, corpo, apéndice*)>

<!ELEMENT prólogo (cabeceira?, contido)>
<!ELEMENT cabeceira (autor+, lingua?, área_temática*, sistema_ortográfico, comentarios?)>
<!ELEMENT contido (encabezamento?, pé_de_foto*, corpo)>

<!ELEMENT corpo (división+, nota*)>
<!ELEMENT apéndice (cabeceira?, contido)>

<!ELEMENT división (encabezamento?, dedicatoria*, cita*, pé_de_foto*, corpo_división)>
<!ATTLIST división tipo (capítulo|parte) #IMPLIED>
<!ELEMENT encabezamento (parágrafo+)>
<!ELEMENT corpo_división (parágrafo+ | división+)>
```

Figura II.2.5: DTD de libro

- **issn**: É opcional.
- **isbn**: É opcional e podería ter varios.
- **número_edición**: Número de edición a partir da que se obtén o documento. No caso dos xornais sempre contén o valor 1.
- **versión_electrónica**: Identificador da versión do documento. Inicialmente ten o valor 1.0 e modifícase cada vez que hai nel unha transformación relevante.
- **ano_versión**: Ano en que se creou esa versión electrónica do documento.
- **responsable_creación**: Responsable da creación do documento, é dicir, o nome do lingüista que se encarga de crear a primeira versión dixital do documento.
- **responsable_versión**: Responsable da versión do documento. Pode diferir da persoa responsable da creación, xa que un mesmo documento pode ser creado por unha persoa e revisado por outra diferente.
- **comentarios**: Comentarios que poden ser incluídos polo creador ou o revisor para sinalar tendencias de lingua do xornal ou erros significativos sistemáticos que poida haber.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- Definición dos elementos comúns a todas as dtds. -->

<!ENTITY % comuns SYSTEM "comuns.ent">
%comuns;

<!-- Estrutura dunha colección -->

<!ELEMENT documento (cabeceira_documento, contido_documento)>

<!ELEMENT cabeceira_documento (identificador, medio, título?, autor*, lingua,
                               soporte, ano_publicación, lugar_publicación,
                               editorial, depósito_legal?, issn?, isbn*,
                               número_edición, ano_primeira_edición,
                               área_temática+, versión_electrónica, ano_versión,
                               responsable_creación, responsable_versión,
                               sistema_ortográfico?, comentarios?)>

<!ELEMENT contido_documento (prólogo*, dedicatoria*, cita*, pé_de_foto*,
                             corpo_documento, apéndice*)>

<!ELEMENT prólogo (cabeceira?, contido)>
<!ELEMENT cabeceira (autor*, lingua?, área_temática*, sistema_ortográfico, comentarios?)>
<!ELEMENT contido (encabezamento?, pé_de_foto*, corpo)>

<!ELEMENT corpo_documento ((agrupación_elementos | elemento)+, nota*)>

<!ELEMENT apéndice (cabeceira?, contido)>

<!ELEMENT agrupación_elementos (encabezamento, dedicatoria*, cita*, corpo_agrupación)>
<!ELEMENT corpo_agrupación ((división | elemento)*, nota*)>

<!ELEMENT corpo (división+, nota*)>

<!ELEMENT elemento (cabeceira_elemento, contido_elemento)>
<!ELEMENT cabeceira_elemento (identificador, título, autor*, lingua?,
                              área_temática*, sistema_ortográfico, comentarios?)>
<!ELEMENT contido_elemento (prólogo*, dedicatoria*, cita*, pé_de_foto*, corpo, apéndice*)>

<!ELEMENT división (encabezamento?, dedicatoria*, cita*, pé_de_foto*, corpo_división)>
<!ATTLIST división tipo (capítulo|parte) #IMPLIED>
<!ELEMENT encabezamento (parágrafo+)>
<!ELEMENT corpo_división ((parágrafo* | división*), nota*)>
```

Figura II.2.6: DTD de colección

Dentro do contido (*contido_ documento*) podemos observar que un xornal se organiza nunha ou máis seccións e que, cada unha delas, contén un nome e unha ou máis noticias.

Os nomes de sección estandarizáronse, dando lugar aos posibles valores que se presentan na táboa da figura II.2.9. Todos os xornais deben ser adaptados a estas seccións en caso de que conteñan algún nome de sección diferente.

Cada noticia, á súa vez, tamén ten unha cabeceira e un contido. A cabeceira contén os seguintes campos:

- **identificador**: Consiste no identificador do xornal, unha barra (/), e un número identificador de noticia. Este segundo número correspóndese coa orde en que aparece a

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- Definición dos elementos comúns a todas as dtds. -->

<!ENTITY % comuns SYSTEM "comuns.ent">
%comuns;

<!-- Estrutura dunha obra teatral -->

<!ELEMENT documento (cabeceira_documento, contido_documento)>

<!ELEMENT cabeceira_documento (identificador, medio, título, autor+, lingua,
                               soporte, ano_publicación, lugar_publicación,
                               editorial, depósito_legal?, issn?, isbn*,
                               número_edición, ano_primeira_edición, ano_escritura,
                               área_temática+, versión_electrónica, ano_versión,
                               responsable_creación, responsable_versión,
                               sistema_ortográfico, comentarios?)>
<!ELEMENT contido_documento (prólogo*, dedicatoria*, cita*, pé_de_foto*, corpo, apéndice*)>

<!ELEMENT prólogo (cabeceira?, contido)>
<!ELEMENT cabeceira (autor+, lingua?, área_temática*, sistema_ortográfico, comentarios?)>
<!ELEMENT contido (encabezamento?, pé_de_foto*, corpo)>

<!ELEMENT corpo (división+, nota*)>

<!ELEMENT apéndice (cabeceira?, contido)>

<!ELEMENT división (encabezamento?, dedicatoria*, cita*, pé_de_foto*, corpo_división)>
<!ATTLIST división tipo (acto|parte|escena|cadro) #IMPLIED>
<!ELEMENT encabezamento (parágrafo+)>
<!ELEMENT corpo_división (parágrafo* | división*)>

<!ELEMENT ano_escritura (#PCDATA)>
```

Figura II.2.7: DTD de teatro

noticia dentro do xornal. Así, o formato do identificador de noticia para *O Correo Galego* sería CGAAAA-MM-DD/N, onde N é o número de noticia.

- **autor**: Autor ou autores da noticia. Ten que haber polo menos un.
- **área_temática**: Área ou áreas temáticas da noticia. Haberá como mínimo unha e, como máximo, tres. Cada área descomponse nun código e un subcódigo (figura II.2.1), que se corresponden cos valores que se amosan na figura II.2.10.
- **sistema_ortográfico**: Indica se a noticia está en galego normativo ou non, para o cal se entende por normativos os textos que intentan seguir algunha das normativas oficiais publicadas desde 1982 [70, 71, 72]. Pode ter os valores Normativo e Outros, polo que se deixará con este último valor se a noticia for anterior á data mencionada ou se, claramente, non segue a normativa oficial.
- **comentarios**: Comentarios que pode incluír o creador ou o revisor para mencionar tendencias de lingua da noticia ou erros significativos sistemáticos que puider haber.

E finalmente, o contido da noticia inclúe:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- Definición dos elementos comúns a todas as dtds. -->

<!ENTITY % comuns SYSTEM "comuns.ent">
%comuns;

<!-- Estructura dunha colección de obras teatrais -->

<!ELEMENT documento (cabeceira_documento, contido_documento)>

<!ELEMENT cabeceira_documento (identificador, medio, título?, autor*, lingua,
                               soporte, ano_publicación, lugar_publicación,
                               editorial, depósito_legal?, issn?, isbn*,
                               número_edición, ano_primeira_edición, área_temática+,
                               versión_electrónica, ano_versión, responsable_creación,
                               responsable_versión, sistema_ortográfico?, comentarios?)>

<!ELEMENT contido_documento (prólogo*, dedicatoria*, cita*, pé_de_foto*, corpo_documento, apéndice*)>

<!ELEMENT prólogo (cabeceira?, contido)>
<!ELEMENT cabeceira (autor*, lingua?, área_temática*, sistema_ortográfico, comentarios?)>
<!ELEMENT contido (encabezamento?, pé_de_foto*, corpo)>

<!ELEMENT corpo_documento ((agrupación_elementos | elemento)+, nota*)>

<!ELEMENT apéndice (cabeceira?, contido)>

<!ELEMENT agrupación_elementos (encabezamento, dedicatoria*, cita*, corpo_agrupación)>
<!ELEMENT corpo_agrupación ((división | elemento)*, nota*)>

<!ELEMENT corpo (división+, nota*)>

<!ELEMENT elemento (cabeceira_elemento, contido_elemento)>
<!ELEMENT cabeceira_elemento (identificador, título, autor*, lingua?, ano_escritura,
                              área_temática*, sistema_ortográfico, comentarios?)>
<!ELEMENT contido_elemento (prólogo*, dedicatoria*, cita*, pé_de_foto*, corpo, apéndice*)>

<!ELEMENT división (encabezamento?, dedicatoria*, cita*, pé_de_foto*, corpo_división)>
<!ATTLIST división tipo (acto|parte|escena|cadro) #IMPLIED>
<!ELEMENT encabezamento (parágrafo+)>
<!ELEMENT corpo_división ((parágrafo* | división*), nota*)>

<!ELEMENT ano_escritura (#PCDATA)>
```

Figura II.2.8: DTD de coleccións de teatro

- **titular**: Titular da noticia, que podería non aparecer.
- **resumo**: Resumo da noticia, que tamén é opcional.
- **pé_de_foto**: Pé da fotografía da noticia. Pode non aparecer ningún ou haber varios.
- **corpo**: Corpo da noticia.

Estes catro campos terán no seu interior un ou máis parágrafos. Particularmente, o campo *corpo* poderá ter ao final as notas ao pé de páxina da noticia, que poderán ser referenciadas no texto, polo que teñen un atributo *identificador* opcional que será utilizado na referencia. Tamén

Figura II.2.9: Seccións dun xornal

teñen un atributo *tipo* que só pode conter o valor *referencia_ bibliográfica*, e que se incorporará cando se tratar dunha referencia bibliográfica (véxanse as figuras II.2.1 e II.2.2).

Os elemenos máis internos da estrutura son os parágrafos e as oracións, que se amosan na figura II.2.2. Un parágrafo consta dunha ou máis oracións, fórmulas ou táboas, e contén os seguintes atributos:

- **interlocutor**: Aparece no caso dunha entrevista. Pode conter os valores Entrevistador ou Entrevistado en función de se o parágrafo lle corresponde ao entrevistador ou ao entrevistado.

- **nome_interlocutor**: Aparece no caso dunha entrevista a máis dunha persoa ou dun coloquio, e contén o nome do interlocutor.

  No caso dunha entrevista cun único entrevistado, non se utilizará este campo. Se, pola contra, é unha entrevista onde hai máis dun entrevistado, aparecerá Entrevistador en *interlocutor* cando se refira ao que fai a entrevista, e o nome do interlocutor cando interveñen os entrevistados (desbótanse, polo tanto, a utilización do atributo *interlocutor* nese caso). No caso dun coloquio emprégase *nome_ interlocutor* en todos os casos e non se usa *interlocutor*.

- **distinto**: Indica se o parágrafo está noutra lingua, na mesma pero noutro período[2], ou se está na mesma lingua mais segundo unha variedade non-normativa[3]. Así, os posibles valores que pode conter son: *outra_ lingua*, *outro_ período*, *non_ normativo* ou *descoñecido*.

- **referencia**: Cando hai unha nota ao pé que fai referencia a todo un parágrafo ou conxunto de parágrafos, a nota situarase no elemento *nota* correspondente, pero deberá aparecer o atributo *referencia* co valor do identificador da nota correpondente no parágrafo afectado.

Por outra banda, as oracións conterán o texto propiamente dito e poderán levar o atributo *distinto*. Ademais, poden incluír outros elementos, como son:

- **referencia_nota**: Cando hai unha nota ao pé, a nota aparecerá no elemento *nota* correspondente, pero no texto manterase a etiqueta *referencia_ nota* que ten o atributo *referencia* como obrigatorio, e que debe incluír o valor do identificador da nota correspondente.

[2]Texto escrito en galego anterior ao 1975.

[3]Inclúe o texto escrito en normativa denominada "de máximos" [6] ou "de mínimos" [38].

Código	Área temática
1	**ECONOMÍA E POLÍTICA**
101	Política
102	Desenvolvemento e infraestruturas
103	Emprego, empresa, industria
104	Sector servizos
105	Explotación primaria
106	Economía, facenda, bolsa
107	Ordenación sanitaria
108	Xustiza, lexislación, dereito
109	Asuntos sociais
110	Ordenación académica
2	**CULTURA E ARTES**
201	Audiovisuais e espectáculo
202	Medios de comunicación
203	Artes gráficas e plásticas
204	Patrimonio, arquitectura, arquivos
3	**CIENCIAS SOCIAIS**
301	Lingua
302	Literatura
303	Relixión
304	Historia e xeografía
305	Civilización, etnoloxía, arqueoloxía e antropoloxía
306	Pensamento, ética e filosofía
307	Socioloxía e psicoloxía
308	Erotismo e sexoloxía
309	Astroloxía
4	**CIENCIAS E TECNOLOXÍA**
401	Sanidade
402	Bioloxía, botánica, ecoloxía, zooloxía e paleontoloxía
403	Tecnoloxía e industria
404	Medio, astronomía e xeoloxía
405	Matemáticas e estatística
406	Química, bioquímica e farmacia
5	**OUTROS**
501	Deportes
502	Turismo
503	Afeccións e asuntos domésticos
504	Actualidade, sucesos, homenaxes, inauguracións...
505	Biografía
506	Nota prologal
6	**FICCIÓN**
601	Novela
602	Relato curto
603	Teatro

Figura II.2.10: Listaxe das áreas temáticas

- **fórmula**: Se no texto orixinal aparece unha fórmula, simplemente se mantén esta etiqueta, pero non se inclúe realmente o seu contido. Deste xeito déixase indicación de que nese punto había unha fórmula.

- **inicio_poema**: Indica o inicio dun poema.

- **novo_verso**: Sinala a finalización dun verso e o comezo doutro.

- **fin_poema**: Marca a finalización dun poema.

- **distinto**: Delimita fragmentos de texto que están noutra lingua, noutro período ou segundo unha variedade non normativa.

Fragmentos noutra lingua ou non normativos

Como se pode intuír, existen diferentes maneiras de representar porcións de texto noutra lingua ou non-normativas (etiqueta e atributo *distinto*), polo que establecemos os seguintes criterios:

- Se hai todo un parágrafo afectado, debe aparecer o atributo *distinto* no parágrafo, e non o das oracións que contén e, por suposto, tampouco a etiqueta *distinto*.
- Se hai unha oración completa non-normativa ou noutra lingua, debe incluírse o atributo *distinto* na oración, e non a etiqueta *distinto* dentro da oración.
- Se o que hai que marcar é unha porción dunha oración emprégase a etiqueta *distinto* para englobar ese fragmento, e desbótase a utilización de calquera dos dous atributos.

Ademais, decidimos non marcar as expresións de menos de tres palabras, xa que o beneficio que poderiamos obter non compensaría o esforzo que suporía facer esta marcaxe.

2.1.3 Revistas

Como se pode ver na figura II.2.4, unha revista ten exactamente a mesma estrutura que un xornal, pero sen estar organizada en seccións. É dicir, no contido do documento xa aparecen as noticias directamente.

A outra diferenza con respecto aos xornais radica en que no identificador non figurará o día (DD) e o campo *medio* conterá Revista en lugar de Xornal.

2.1.4 Libros

No caso dos libros xa existen máis diferenzas, como se pode extraer da figura II.2.5:

- **identificador**: É arbitrario e estará baseado nas palabras do título do libro. Normalmente úsanse estas sen tiles, substitúense os espazos por guións baixos e o *ñ* por *nn*, e elimínanse corchetes, parénteses, interrogantes, exclamacións, puntos, e comiñas dobres e simples. Se este nome sobrepasa as cinco palabras, é adaptado manualmente para que sexa o máis representativo posible.
- **medio**: Neste caso conterá Libro.
- **título**: Sería o equivalente a *nome* no caso de xornais e revistas. Contén o título do libro.
- **autor**: Pode haber varios, e especifica o autor ou autores do mesmo.
- **área_temática**: Pode haber varias, ata un máximo de tres, e especifica a área ou áreas temáticas.
- **sistema_ortográfico**: Indica se o libro está en galego normativo ou non. Véxase a sección 2.1.2 para máis detalles sobre este campo.

No contido do documento pode haber prólogos, dedicatorias, citas, pés de foto, o corpo do documento, e apéndices. Só é obrigatorio o elemento *corpo*, que está formado por unha ou máis divisións (capítulos, partes, seccións etc.) e poderá estar sucedido por notas ao pé de páxina.

Cada división ten un corpo, que pode ser un conxunto de parágrafos ou doutras divisións. Ademais, cada división podería ter un encabezamento, dedicatorias, citas, e pés de foto. Especificarase o tipo de división (atributo *tipo*) no caso de capítulos e partes sempre que estiver indicado explicitamente no libro.

Tanto os prólogos como os posibles apéndices poden ter unha cabeceira[4], que conterá o autor ou autores dese elemento, a área ou áreas temáticas[5], o sistema ortográfico, os comentarios e o contido, cun posible encabezado e pés de foto e, finalmente, o corpo, que poderá ter exactamente a mesma estrutura complexa que o corpo dun libro. No caso de que os prólogos ou apéndices sexan do mesmo autor, non levarían cabeceira.

Como se pode observar, o corpo do libro e, polo tanto, tamén o de prólogos e apéndices, ten obrigatoriamente unha división. É dicir, se o libro non está estruturado en divisións ou, o que é o mesmo, soamente contén parágrafos de texto, todos os parágrafos estarán metidos dentro dunha división. Se, polo contrario, no libro aparece un texto introdutorio, e despois inclúe divisións, este texto inicial deberá ir tamén dentro dunha división[6].

2.1.5 Coleccións

Englobamos neste tipo de documento todas as coleccións de relatos ou ensaios do mesmo ou de diferentes autores.

Este caso é bastante semellante ao dos libros, pero cambian algúns elementos de estruturación xerais. Así, mentres que nun libro o seu corpo está constituído por divisións, nas coleccións o corpo do documento está formado por elementos e/ou agrupacións de elementos (véxase a figura II.2.6).

Por un lado, un elemento, á súa vez, ten exactamente a mesma estrutura que un libro, coa excepción dos campos da cabeceira non aplicables neste caso e, polo outro, *agrupación_elementos* utilízase para aglutinar un conxunto de elementos con algún encabezamento común ou texto de unión. É por isto que no corpo dunha colección se poden intercalar divisións con elementos.

Finalmente, para este caso a etiqueta *medio* tamén conterá o valor Libro.

2.1.6 Teatro e coleccións de teatro

O único que cambia no caso de teatro e coleccións de teatro, en referencia a libros e a coleccións respectivamente, consiste en que na cabeceira do documento no caso de teatro, e na cabeceira de elemento no caso das coleccións, existe o elemento *ano_escritura*, que inclúe o ano en que se escribiu a obra.

Ademais, utilízase o atributo *tipo* dos parágrafos e oracións para marcar as acoutacións, e a etiqueta *división* pode conter os valores *acto*, *parte*, *escena* e *cadro*.

[4] No caso de que o autor dese elemento constitutivo sexa diferente ao autor do libro.

[5] En caso de que difira das da cabeceira xeral do libro.

[6] Consideraremos como máximo tres niveis de xerarquía nas divisións, deixando todos os demais, a partir do cuarto inclusive, no terceiro.

2.1.7 Criterios de decisión

Como pode extraerse da estrutura definida dos textos do corpus, a unidade mínima de estruturación que consideramos é a oración. A explicación desta decisión débese a que, aínda que no presente traballo nos centramos no CORGA como corpus non etiquetado[7], xa desde un inicio se pensou en que este puidera estalo, ou incluso analizado a modo de banco de árbores[8].

Se nalgún momento do desenvolvemento dun corpus se ten en mente a súa anotación, resulta de vital importancia esta estruturación en oracións, máis aínda se se pensa en representar a súa análise sintáctica, xa que é a unidade de traballo de moitas das ferramentas que o utilizarían[9].

A marcaxe de *inicio_poema*, *novo_verso*, e *fin_poema* dentro das oracións constitúe a materialización da técnica de marcaxe en liña que se presentou na sección 1.1.6. O obxectivo desta marcaxe no noso caso é meramente visual, é dicir, trátase de que, na ferramenta de consulta do corpus, se permita visualizar o texto en verso de xeito diferenciador.

Por outra banda, no noso caso desbotamos a utilización de esquemas XML en beneficio das DTD debido a que, cando se comezou a construción do corpus utilizando o estándar XML, non existían ferramentas suficientemente maduras que permitiran traballar axeitadamente con esquemas.

2.2 Fases do procesamento

No noso caso de estudo os documentos poden chegarnos a través dun exemplar en papel ou en soporte electrónico e, dentro dos documentos en soporte electrónico, distinguimos os textos formados por varias páxinas HTML dos que xa conseguimos en formato PDF. A seguir describimos cal é o proceso de transformación que seguen os documentos, desde que están ao noso dispor ata que pasan a formar parte do corpus, que variará en función do tipo de texto do que se tratar.

2.2.1 Adquisición

O caso máis sinxelo de adquisición da versión dixital dun documento constitúeno os arquivos PDF, que habitualmente se atopan nalgún sitio web para a súa descarga. Para conseguilos empregamos un navegador web e descargamos o arquivo do lugar onde está situado. Pola contra, para descargar arquivos HTML empregamos diversas estratexias en momentos diferentes. Para algúns documentos formados por varias páxinas resultan de utilidade ferramentas como *wget* [37] ou HTTrack [74] e, se non conseguimos descargar ben todas as páxinas asociadas, podemos facelo manualmente unha por unha empregando o navegador.

Un caso ben distinto é o dos documentos en papel, que deben ser dixitalizados usando un escáner, o que supón bastante máis traballo. Utilizamos o Omnipage Pro 12[10] para realizar esta tarefa. Configuramos e definimos diferentes perfís dentro da ferramenta para poder traballar de maneira automática coas diferentes formas en que podemos recibir os documentos: úsase o alimentador automático ou o modo manual, introdúcense follas a unha ou dobre cara, diferentes

[7] Entendemos por corpus etiquetado aquel que ten información morfosintáctica asociada a cada unha das palabras que inclúen os documentos que o conforman (o que orixinalmente en inglés se denomina un *Part-of-Speech tagged corpus* [44]).

[8] Do inglés *treebank* [44].

[9] No momento da redacción da presente tese estanse desenvolvendo as ferramentas e procedementos para proceder á anotación dos textos do corpus.

[10] Omnipage é un producto rexistrado de Nuance Communications, Inc. [59].

tamaños de papel etc. Isto deu lugar a varios arquivos de configuración cos diferentes perfís, que poden cargarse na ferramenta antes de tratar un documento. Deste modo evítase especificar manualmente cada un dos parámetros en cada ocasión e, por tanto, a perda de tempo conseguinte e a posibilidade de cometer equivocacións.

Cada documento en papel que se procesa gárdase nun arquivo co formato da ferramenta Omnipage e almacénase no lugar que lle corresponde. Do mesmo xeito, os arquivos PDF son almacenados no propio formato e, para cada documento HTML, créase un directorio que aglutina as súas páxinas HTML. A metodoloxía de almacenamento será revisada en profundidade na sección 2.4, polo que aparcaremos a discusión sobre onde se deben gardar os arquivos ata ese momento.

Por outra banda, para evitar confusións no emprego das diferentes ferramentas que se utilizan nesta fase, definimos o protocolo de cada un dos pasos que debe seguir o lingüista encargado de realizar a adquisición dun documento. A existencia dunha guía non só aforra tempo na formación do novo persoal, senón tamén na resolución de dúbidas recorrentes. Canto máis polo miúdo se describan os pasos definidos neste protocolo, máis se axilizará o traballo en fases posteriores.

No apéndice A amosamos un exemplo de definición de protocolo para realizar a adquisición dos documentos.

2.2.2 Estruturación

Unha vez almacenados, é necesario estruturar os documentos resultantes da adquisición para adaptalos ao formato de representación intermedia que decidamos (véxase sección 1.1.7 para máis detalles). No noso caso utilizamos arquivos e directorios que variarán en función do tipo de documento do que se trate.

Segundo a orixe do documento, as ferramentas informáticas que se usarán para levar a cabo esta tarefa serán diferentes. Así, se o documento que imos estruturar está formado por un ou varios arquivos HTML, empregaranse o navegador web e un editor de texto para crear o formato desexado utilizando as funcións de copiado e pegado. Calquera editor de texto con funcionalidades avanzadas como TextPad [47] ou Emacs [34] resulta adecuado para realizar estas tarefas.

Se o documento estaba orixinariamente en papel e, daquela, temos a versión dixital en formato Omnipage, é necesario pasarlle un programa de OCR (no noso caso o propio Omnipage Pro 12): delimítanse, unha a unha, as rexións que haxa que exportar a texto e revísanse e corríxense os erros que se producen no recoñecemento dos caracteres, para o que é de utilidade ir confeccionando unha listaxe de erros típicos que permitirá automatizar parcialmente este proceso. Finalmente, complétase a estrutura para que se adecúe ao formato definido mediante o editor de textos.

Se, en cambio, o documento que se vai estruturar está en PDF, haberá que actuar de maneira similar aos textos en papel. Con Omnipage exportamos o texto, unha vez delimitadas as zonas pertinentes pero, neste caso, non haberá que estar pendente de corrixir os erros de OCR, xa que non os haberá.

Por outra banda, dado que Omnipage Pro 12 non considera o galego como unha das linguas de traballo, foinos de utilidade crear un dicionario galego personalizado de 2000 formas, o máximo que permite a versión do programa coa que traballamos, extraídas das máis frecuentes dos textos

que tiñamos. Isto axudounos a mellor sensiblemente os resultados provenientes das tarefas de OCR.

A continuación amósase a estrutura de directorios e arquivos utilizada para cada un dos tipos de documentos.

Xornais

Se observamos a estrutura elixida para os xornais, representada na figura II.2.11, podemos ver como é necesario crear un directorio para o nome do xornal co identificador deste (*vid.* sección 2.1.2).

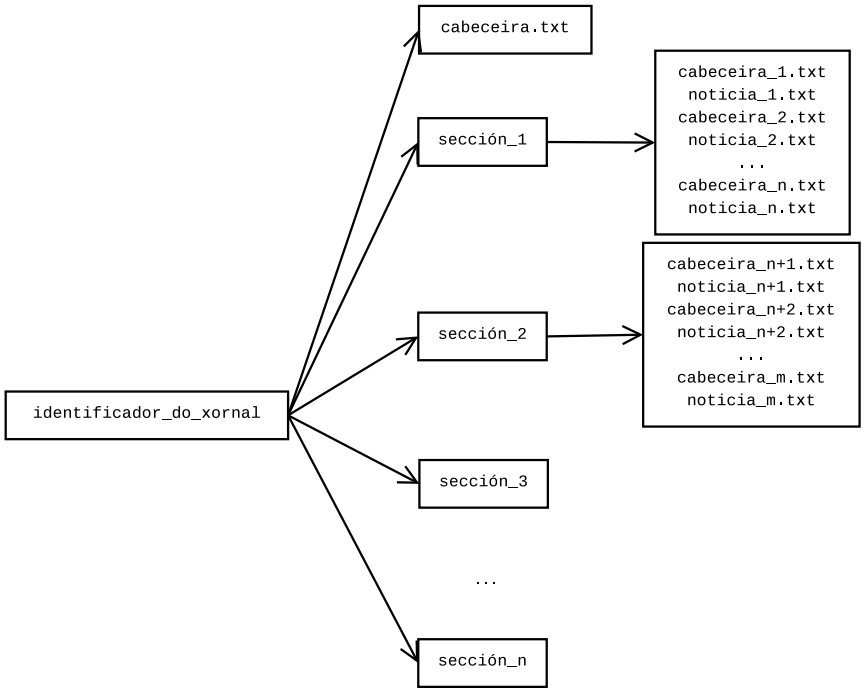

Figura II.2.11: Estrutura dos directorios dun xornal

Este directorio contén un arquivo coa cabeceira xeral do xornal e un subdirectorio por cada sección e, dentro de cada unha delas, temos dous arquivos por cada noticia: cabeceira_x.txt e noticia_x.txt, numerados consecutivamente segundo a orde en que aparecen no xornal. Na cabeceira xeral figuran os datos relativos á publicación, mentres que na cabeceira das noticias, a información correspondente a cada unha delas.

Pola contra, os arquivos noticia_x.txt son os que teñen o contido da noticia propiamente dito, pero deben verificar unha serie de regras para facilitarlles o traballo aos *scripts* que procesarán eses datos en fases posteriores:

- En primeiro lugar irá o titular.
- A continuación, o resumo, precedido por unha liña en branco.
- E, finalmente, o corpo da noticia, precedido tamén por unha liña en branco.

- Se algún destes elementos non existe déixase unha liña en branco adicional.
- En cada un dos elementos anteriores ponse un retorno de carro ao final de cada parágrafo.
- Polo medio do texto poden aparecer as etiquetas <COMMENT> e </COMMENT> englobando conxuntos de palabras que están noutra lingua.

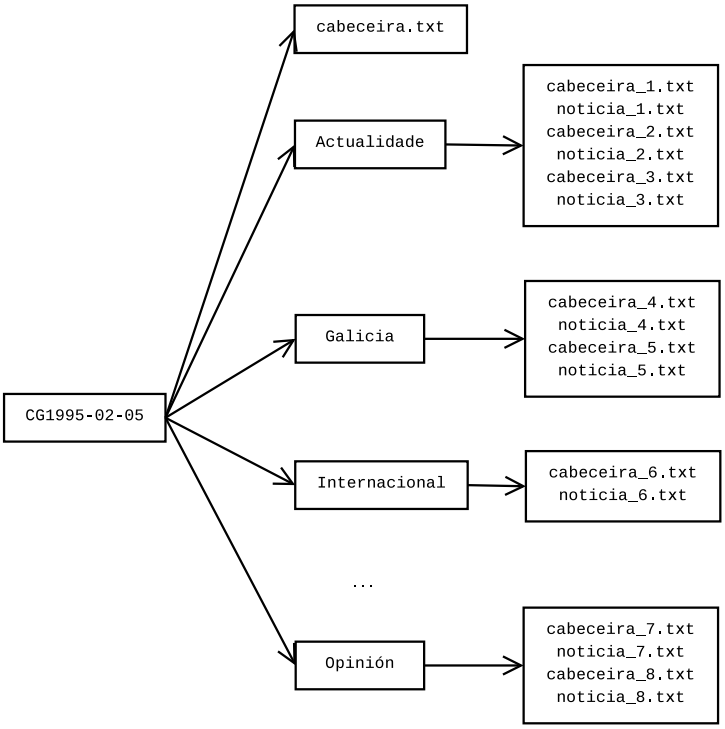

Figura II.2.12: Estrutura dos directorios de *O Correo Galego* do 5 de febreiro de 1995

Na figura II.2.12 exemplifícase a estrutura de directorios para o xornal *O Correo Galego* do 5 de febreiro de 1995, na figura II.2.13 amósase o contido correspondente á súa cabeceira xeral e na II.2.14 a relativa á súa segunda noticia. Para facilitar o traballo, creouse un arquivo modelo para cada un dos dous tipos de cabeceira con todos os campos baleiros de contido, de xeito que o equipo de lingüistas vai copiando e modificando ese modelo para cada documento que procesa. Así conséguense eliminar os erros causados porque se esqueza a inclusión dalgún campo.

No apéndice B amósase un exemplo de protocolo para executar a fase de estruturación dun xornal.

Revistas

O caso das revistas é exactamente igual ao dos xornais. A única diferenza consiste en que, nas primeiras, non se consideran as seccións, polo que os elementos cabeceira_x.txt e noticia_x.txt irían directamente dentro do directorio constituído polo identificador da revista. Este identificador estaría constituído polas iniciais da revista, o ano e o mes, xa que as revistas empregadas son de publicación mensual.

```
<identificador>CG1995-02-05</identificador>
<medio>Xornal</medio>
<nome>O Correo Galego</nome>
<lingua>Galego</lingua>
<soporte>Papel</soporte>
<ano_publicación>1995</ano_publicación>
<lugar_publicación>Santiago de Compostela</lugar_publicación>
<editorial>Editorial Compostela</editorial>
<depósito_legal>C 31/1994</depósito_legal>
<número_edición>1</número_edición>
<versión_electrónica>1.0</versión_electrónica>
<ano_versión>2002</ano_versión>
<responsable_creación>Nome do encargado de crear o documento</responsable_creación>
<comentarios></comentarios>
```

Figura II.2.13: Cabeceira xeral na fase de estruturación de *O Correo Galego* do 5 de febreiro de 1995

```
<identificador>CG1995-02-05/2</identificador>
<autor>Redacción</autor>
<autor>Axencia</autor>
<área_temática>101</área_temática>
<área_temática>504</área_temática>
<sistema_ortográfico>Normativo</sistema_ortográfico>
```

Figura II.2.14: Cabeceira na fase de estruturación da segunda noticia de *O Correo Galego* do 5 de febreiro de 1995

Libros

O caso dos libros varía substancialmente. A estrutura de arquivos e directorios para este propósito amósase na figura II.2.15. Un libro pode ter dedicatorias, citas, prólogos, e apéndices, pero o único que debe aparecer obrigatoriamente en todos os casos é o corpo. Tanto os prólogos e apéndices como o corpo do libro estrutúranse en divisións. Cada división pode ter un encabezamento (encamezamento.txt) e un corpo (corpo.txt) ou, pola contra, ter o corpo estruturado en subdivisións.

O encabezamento representa o título da división. Se un prólogo, un apéndice ou o corpo do libro non está estruturado en divisións, crearase igualmente unha división artificial, que conterá soamente o arquivo corpo.txt. Adicionalmente, os prólogos e apéndices incluirán un arquivo cabeceira.txt cando a súa autoría difira da do libro.

Opcionalmente poden existir notas ao pé, tanto no corpo coma nos elementos periféricos (prólogos e apéndices), así que se creará un directorio denominado Notas no lugar correspondente cando sexa necesario, que incluirá un arquivo de texto notas.txt co seguinte formato[11]:

```
<numero_identificador>. contido da nota<retorno_de_carro>
<numero_identificador>. contido da nota<retorno_de_carro>
<numero_identificador>. contido da nota<retorno_de_carro>
```

[11] Os elementos que están entre < e > deben substituírse polo que representan.

Todos os arquivos, exceptuando os que conteñen as cabeceiras, deberán levar un retorno de carro ao final de cada parágrafo.

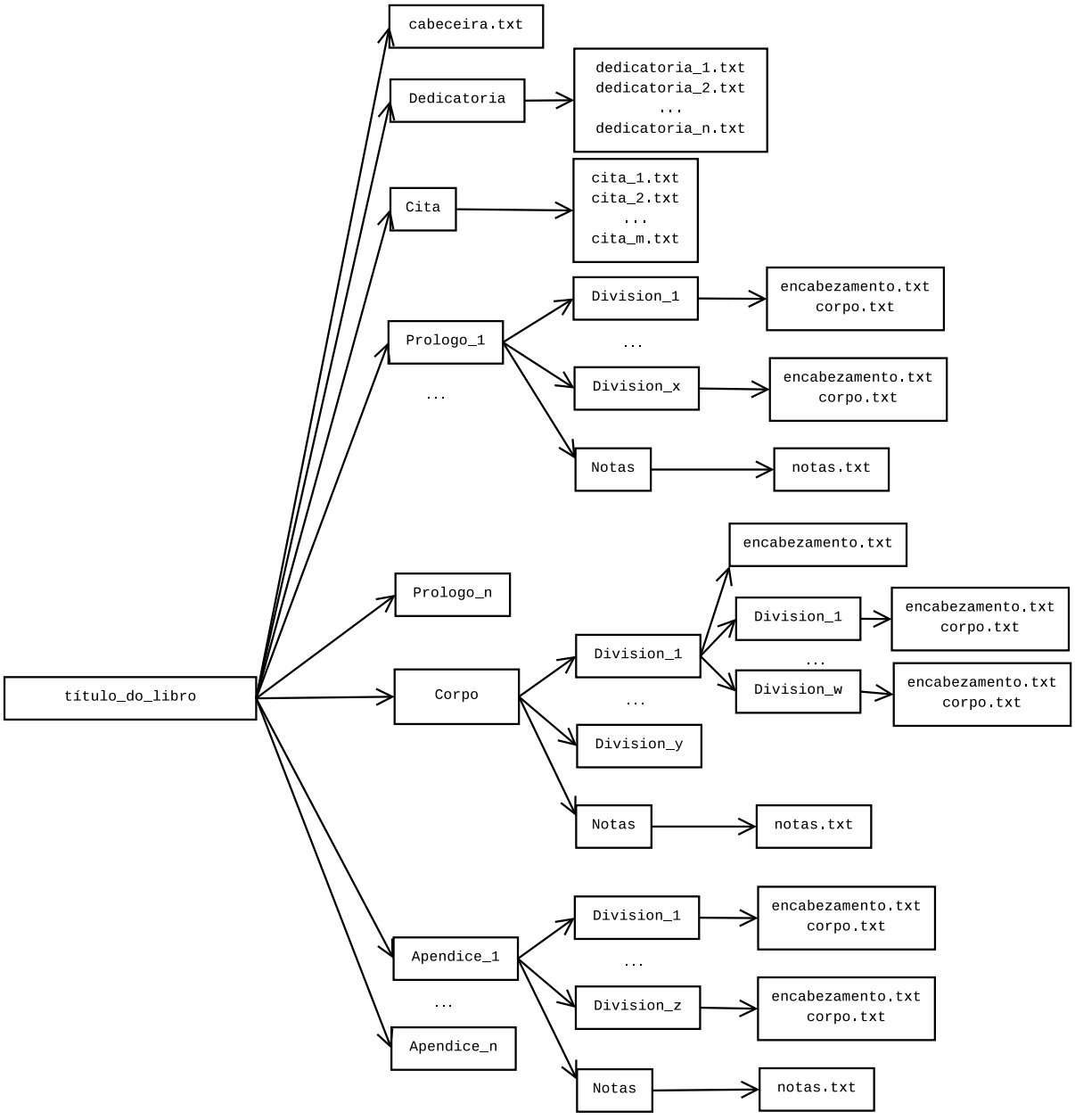

Figura II.2.15: Estrutura dos directorios dun libro

Coleccións

A estrutura de arquivos e directorios para coleccións amósase na figura II.2.16. Pode observarse que hai que crear un directorio Elementos que conterá un subdirectorio Elemento para

cada compoñente da colección. Cada un destes, á súa vez, pode ter unha estrutura tan complexa como a dos libros, con dedicatorias, citas etc. O único que hai que reseñar con respecto aos libros son os títulos dos elementos da colección, situados nos arquivos titulo.txt correspondentes.

Teatro

A estrutura de directorios de teatro é similar á de libros, coa excepción de que os arquivos que conteñen o corpo do texto poden incoporar identificacións de personaxes. Así, por exemplo, se no texto aparece algo como

```
XUÍZ. (Érguese enfadado) É a última vez que o digo!
```

incluiríase no texto marcado da seguinte maneira:

```
<Xuíz>
(Érguese enfadado) É a última vez que o digo!
```

É dicir, utilízase o nome do personaxe, enmarcado cos signos < e >, entre parágrafo e parágrafo para identificar o personaxe co que se corresponde o parágrafo situado a continuación.

Do mesmo xeito sinálanse os parágrafos que son acoutacións con <ac> diante.

Coleccións de teatro

Nas coleccións de teatro séguense as mesmas pautas que no caso conxunto de coleccións e obras teatrais.

2.2.3 Conversión

Unha vez que se teñen os documentos estruturados segundo o exposto na sección anterior, hai unha fase de conversión, que xera unha primeira versión do documento XML.

No noso caso construímos unha serie de *scripts*[12] de transformación que collen a estrutura de directorios definida para cada documento e reconstrúen esta versión XML. Entre as funcións que realizan os *scripts* atópanse:

- Crear a estrutura principal do XML.
- Introducir os datos da cabeceira xeral.
- Estruturar correctamente as divisións no caso de libros ou coleccións, e as seccións no caso de xornais.
- Para xornais e revistas, inserir no XML as noticias na orde establecida polos números dos nomes de arquivo.
- Estruturar e encher correctamente as cabeceiras dos elementos no caso de coleccións.
- Identificar e cubrir correctamente os personaxes nas obras teatrais.

[12] No proxecto CORGA, cando falamos de *scripts* referímonos a pequenos programas desenvolvidos utilizando, fundamentalmente, Perl [65, 93], XSLT [88, 103], libxml [40] e bash [33, 58].

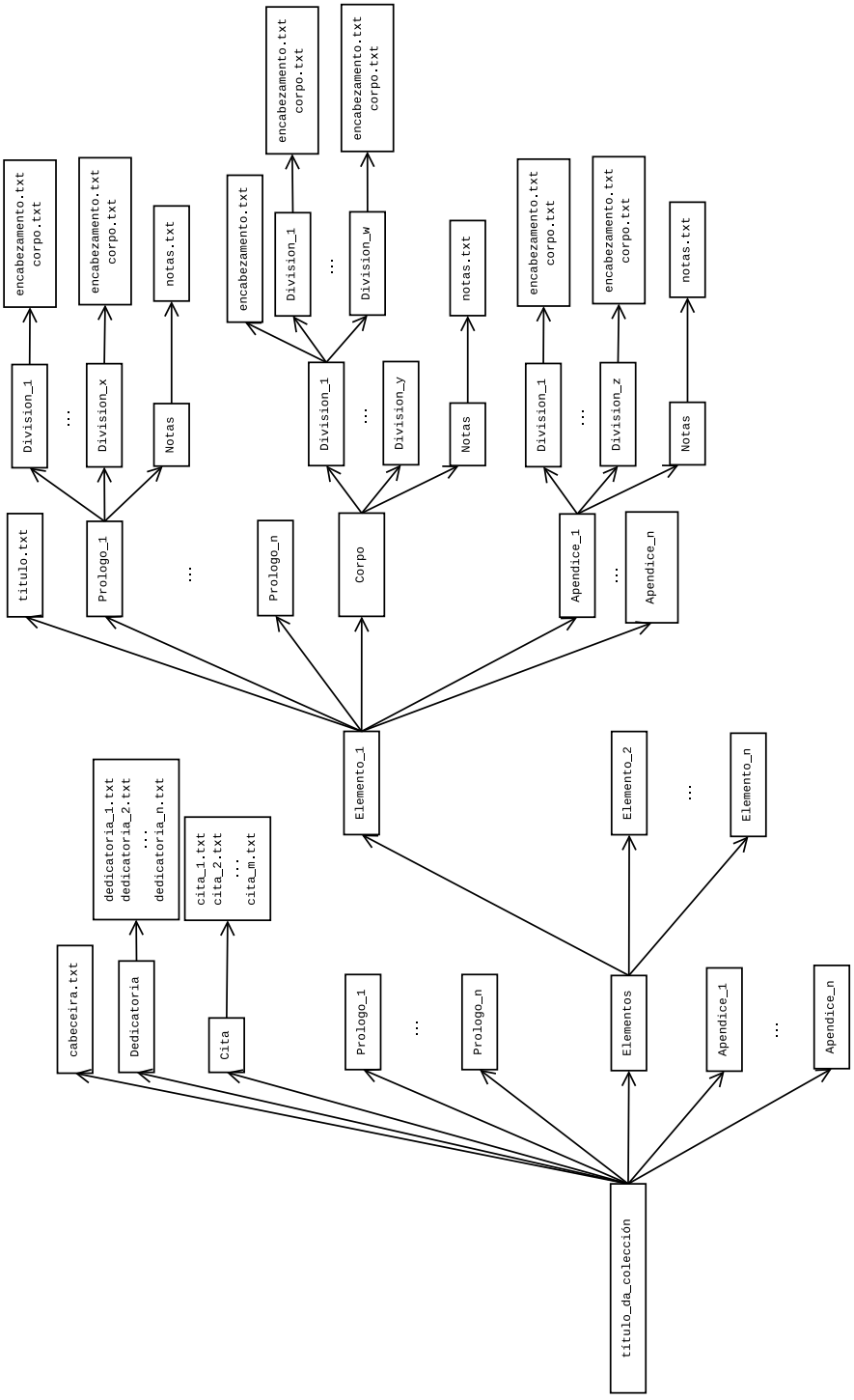

Figura II.2.16: Estrutura dos directorios dunha colección

- Segmentar o texto en parágrafos, e estes, á vez, en oracións. Dado que na fase de estruturación só están delimitados os parágrafos mediante retornos de carro, desenvolveuse

un segmentador de oracións que ten en conta abreviaturas, siglas e outros fenómenos lingüísticos que dificultan unha correcta segmentación.

2.2.4 Revisión

Nesta fase revísanse e corrínxese manualmente os documentos XML resultado da fase anterior xa que, como é de agardar, os *scripts* mencionados non poden deixalos perfectamente estruturados[13]. Entre as tarefas que deben realizar os membros do equipo lingüístico nesta fase atópanse:

- Revisar que as oracións estean ben segmentadas: Aínda que o segmentador de oracións é bastante sofisticado, non sempre fai esta segmentación correctamente e é necesario corrixir manualmente os erros.
- Colocar os pés de foto no lugar que lles corresponda: Como non foron marcados na fase de estruturación, é necesario recolocalos no documento.
- Corrección de erratas: Pode suceder que haxa erratas nos documentos[14] e que cheguen intactas ata esta fase, así que haberá que corrixilas.
- Introducir as referencias ás notas: Aínda que as notas están marcadas e son detectadas, non sucede o mesmo coas referencias a elas, polo que hai que introducilas.

Para realizar isto, decidimos utilizar o editor XMLMind XML Editor [67], debido a que é moi sinxelo o seu manexo e que dispón, ademais, dunha versión gratuíta. Aínda que permite manipular o documento XML sen que teña unha folla de estilos asociada, o traballo faise bastante caótico: a vista do documento é moi homoxénea xa que non se resaltan unhas partes sobre outras, non permite cambiar o tamaño da fonte de visualización, ten por defecto un tamaño de fonte bastante pequeno para unha resolución de pantalla habitual e os cambios de todos os atributos fanse nunha zona independente á parte. É por isto polo que consideramos necesario definir unha folla de estilos que permite visualizar e manipular os documentos dun xeito moito máis sinxelo desde o editor.

Mediante a definición desta folla de estilos conseguimos mellorar o aspecto visual do documento no editor e adaptalo ás nosas necesidades específicas. Ademais, fixo posible manipular algunhas partes do documento sen ter que utilizar os menús externos, co que o traballo se converteu en algo moito máis ameno e sinxelo.

Dado que no momento de definir esta fase para o CORGA o soporte dos esquemas XML aínda non estaba moi estendido entre as ferramentas de edición XML, utilizamos varias DTD para definir os tipos de documentos que tiñamos. Isto provocou que non se puidera validar desde o propio editor o contido dalgunhas etiquetas que debían incluír un valor de entre un conxunto de elementos posibles.

Polo tanto, para evitar que nas mencionadas etiquetas puidera introducirse accidentalmente algún valor non permitido, modificáronse lixeiramente as DTD definidas nas figuras II.2.1, II.2.2, II.2.3, II.2.4, II.2.5, II.2.6, II.2.7 e II.2.8, de maneira que algúns contidos de etiquetas pasaron a

[13] Entre outras cousas porque non todos os fenómenos que se consideraron nas DTD se marcaron na fase de estruturación.

[14] Por exemplo, porque o programa de OCR cometa erros no recoñecemento dos caracteres.

ser atributos multiavaliados. Deste xeito o editor XML só deixa seleccionar un valor de entre os definidos nas DTD e así, evitamos que se produzan erros.

Este é o caso, por exemplo, da etiqueta *editorial* da cabeceira do documento, onde para evitar que o nome dunha mesma editorial se escriba de diferentes xeitos, pasamos a considerala unha etiqueta baleira cun atributo nome cun conxunto predefinido de valores posibles.

Ocorre algo similar coas seccións dos xornais. As noticias dun xornal aparecen organizadas en seccións na estrutura documental pero, dado que resulta moi laborioso andar metendo as noticias en cadansúa sección, no XML de traballo eliminouse esa organización en seccións e incorporouse unha etiqueta *sección* co atributo multiavaliado *nome* en cada noticia, o que facilitou considerablemente o traballo do proceso de revisión.

No apéndice C amósase un exemplo de protocolo para a realización das tarefas relacionadas con esta fase.

2.2.5 Validación

Dado que tivemos que adaptar un pouco as DTD para que a fase anterior puidera realizarse dunha maneira máis cómoda, o primeiro que facemos xusto antes de validar un documento é transformalo para que cumpra as DTD orixinais, é dicir, as que amosamos anteriormente. Isto realízase de xeito automático mediante un *script* que desfai esas pequenas adaptacións.

Unha vez feito isto e, aínda que na ferramenta utilizada na revisión xa se fan algunhas validacións para evitar que se cometan erros, construímos unha serie de *scripts* encargados de validar o contido dalgunhas etiquetas que non se puideron verificar na revisión. Estes encárganse de facer diversas comprobacións e, se apareceren mensaxes de erro, comunícanselle ao lingüista responsable do documento para que faga as correccións oportunas. Algunhas cousas que comproban son:

- Revisión das áreas temáticas: Constátase que os códigos que se introduzan sexan válidos (é dicir, algún dos da figura II.2.10), que os xornais e as revistas non teñan códigos de área temática que comecen por 6 (xa que este díxito inicial só é válido para libros) etc.

- Boa marcaxe dos textos noutra lingua: Como xa comentamos anteriormente existen diferentes xeitos de delimitar as zonas que están noutra lingua. Neste caso os *scripts* comproban que non haxa unha oración co atributo *distinto* dentro dun parágrafo que xa ten ese atributo e que non exista unha oración englobada completamente dentro da etiqueta *distinto*, xa que para isto empregamos o atributo *distinto* de oración.

- Marcaxe axeitada dos poemas: Dado que, como xa comentamos, os poemas se marcan en liña de maneira transversal á estrutura principal dos documentos, é fácil que haxa despistes durante a revisión. Os *scripts* encárganse de detectar as incoherencias.

2.2.6 Inclusión

Os documentos ao chegar a esta fase xa cumpren todos os requirimentos para formar parte do corpus, así que simplemente hai que metelos no directorio que lles corresponda.

2.3 Recursos hardware, económicos e humanos

Esta epígrafe está condicionada pola organización e o funcionamento do Centro Ramón Piñeiro para a Investigación en Humanidades. O proxecto habitualmente está formado por oito persoas: o director do proxecto, o coordinador lingüístico, o coordinador informático, cinco bolseiros da parte lingüística e un bolseiro da parte informática. Os bolseiros permanecen no proxecto ata un máximo de tres anos e, cando marchan, convócanse novamente as vacantes para que entren a formar parte do proxecto novos investigadores.

Esta rotación fai que sexa moi importante axilizar a fomación inicial dos bolseiros, que deberán interiorizar rapidamente os coñecementos sobre a dinámica de traballo e os protocolos de actuación do proxecto, para que este siga avanzando.

Por cada membro do equipo existe un ordenador para realizar o traballo. Actualmente os ordenadores do equipo lingüístico teñen Windows XP[15] instalado aínda que, no momento de escritura da presente tese, estase a barallar a posibilidade de migrar todos os ordenadores a un sistema operativo GNU/Linux [36, 53, 54].

Adicionalmente, hai un ordenador no centro conectado a un escáner con alimentador, destinado a facer as tarefas relativas á fase de adquisición para os documentos dispoñibles en papel, e outros dous co sistema operativo Debian estable [81], en que o equipo informático desenvolve e executa os diferentes *scripts* de transformación.

2.4 Almacenamento

Canto á política de almacenamento dispoñemos dun servidor que comparte unha unidade de disco con todas as máquinas de traballo. Todos os ordenadores téñena como unidade J:. Nela temos estes directorios:

- corga_xml_adquisicion: Contén os documentos na versión dixital máis primitiva de que se dispón (a que se ten xusto ao rematar a fase de adquisición). Distinguimos os subdirectorios HTML, PDF e escaneados, que conteñen os documentos dos tipos respectivos (os que están en HTML, en PDF, e os arquivos de Omnipage resultado do escaneamento dos que están en papel).

  Dado que o resultado do escaneamento dun documento ocupa moito, decidimos que o directorio escaneados sexa un almacenamento temporal, de xeito que imos arquivando en CD ou DVD eses documentos a medida que imos tendo outros novos. Dependendo da actividade que haxa na fase de adquisición baléirase dun xeito máis ou menos frecuente. O resto de directorios que describimos seguidamente témolos dispoñibles sempre no disco duro do servidor.

- corga_xml_estruturacion: Inclúe os documentos despois do resultado da estruturación. Identificamos aquí un subdirectorio por cada tipo de texto: xornais, revistas, coleccions, teatro e coleccions_teatro. Esta subdivisión aparecerá tamén dentro de todos os directorios que describimos a continuación.

  Dentro de cada un destes subdirectorios crearase outro co nome do título do documento que se está estruturando, tal e como explicamos na sección 2.2.2.

[15] Windows XP é unha marca rexistrada por Microsoft Corporation.

- corga_xml_conversion: Contén os documentos resultantes despois de pasalos polos *scripts* da fase de conversión.

- corga_xml_revision: Inclúe os documentos que están en revisión.

- corga_xml_inclusion: Contén os documentos na súa versión definitiva, os que xa compoñen o corpus e que pasaron sen erros a fase de validación.

Canto á política de copias de seguridade, facemos unha copia diaria do disco do servidor, utilizando para iso un robot apropiado para manexar un volume tan grande de información.

2.5 Comprobación da metodoloxía

Unha vez construída a metodoloxía e, sempre antes de que o equipo de traballo ao completo se poña a traballar no desenvolvemento do corpus, foi necesario probala. Para isto, os coordinadores lingüístico e informático fixeron o proceso completo de conversión dun texto para cada tipo de documento definido, desde a fase de adquisición ata a inclusión.

Desta forma, detectáronse algúns erros nos diferentes *scripts* de conversión e algunhas deficiencias e complicacións na manipulación dos textos que foron solucionados. Ademais tomouse nota das dificultades xurdidas en certas operacións de revisión, para seren detalladas na documentación dos protocolos de actuación.

Cando se remataron as comprobacións da metodoloxía de maneira exitosa para un tipo de documento, comezouse coa planificación temporal e o desenvolvemento dese tipo de documentos, que xa incluía todo o equipo do proxecto.

2.6 Planificación temporal

A máxima do proxecto á hora de facer a planificación temporal foi, e segue a ser, intentar ter equilibrado o corpus no que respecta a tres parámetros: temporal (lustros), temático (áreas) e medio da publicación. Aínda que na maior parte do tempo o corpus si se atopaba equilibrado, houbo varios momentos en que isto non sucedeu:

- Ao comezo do proxecto, cando se remataba a comprobación da metodoloxía para un tipo de documentos e se iniciaba con eles a fase de desenvolvemento. As comprobacións da metodoloxía para algún tipo de texto levaron máis tempo do esperado, polo que, ata que se remataron as probas para todas as tipoloxías textuais e se puido incorporar un número razoable de textos variados, pasou bastante tempo. Isto fixo que, nunha fase inicial, a base documental do corpus non estivera moi compensada.

- Durante etapas intermedias do proxecto, debido a que durante algúns períodos de tempo había poucos bolseiros no proxecto[16] e, neses momentos, a dedicación de esforzos a compensar un parámetro descompensaba os outros.

[16] Nalgún caso coincidiu a finalización dun período de traballo para varios bolseiros á vez e tardou en facerse efectiva a súa substitución.

Actualmente faise unha replanificación temporal anual que intenta, na medida do posible, ir equilibrando o corpus canto aos tres parámetros expostos. De todos os xeitos, algo que hai que ter especialmente en conta é que debe planificarse a adquisición de documentos con certa antelación para non ter problemas á hora de dispoñer deles cando chegue o momento de procesalos.

Un caso especialmente relevante neste sentido é a adquisición de xornais ou revistas electrónicos que, unha vez pasado o día de publicación, é moi complicado conseguir. Isto témolo en conta na planificación anual e nela programamos a descarga automática (ou cando menos, semiautomática) dos documentos dixitais que han de incorporarse ao corpus.

2.7 Desenvolvemento

Na fase de desenvolvemento xa está implicado todo o equipo de traballo. Antes de que unha persoa nova poida involucrarse na fase de desenvolvemento é necesario un tempo de formación, que require que esta estude a fondo os protocolos de actuación[17], e que o coordinador lingüístico revise durante un período de tempo o resultado do traballo deste membro.

Logo de comprobarmos que o traballo dunha nova persoa cumpre os criterios de calidade esixibles, xa non é revisado. En calquera caso, os *scripts* de validación evitan que moitos erros pasen ao corpus sen seren antes revisados, polo que se garante unha calidade aceptable nos textos. Ademais, cunha periodicidade anual, fanse varias sesións dirixidas a atopar erros nos documentos que puideran non ser detectados.

Por outra banda, durante toda a fase de desenvolvemento, o equipo lingüístico fai reunións semanais para poñer en común os problemas atopados nas distintas revisións, as dificultades en casos particulares e casuística complexa de afrontar relacionada coa estruturación ou a segmentación de oracións. Tómanse decisións consensuadas para a resolución dos casos expostos e rexístranse nos protocolos de actuación para que non se volvan discutir no futuro.

Finalmente, tamén é importante resaltar que a interacción na transición de fases manuais a automáticas, e á inversa, que require a comunicación dos equipos lingüístico e informático, sempre se fai a través dos coordinadores, que coñecen algúns detalles dos protocolos de actuación irrelevantes para o resto do equipo (establecemento do espazo en disco onde se fai o intercambio dos arquivos, onde deben ser colocados nun primeiro momento etc).

2.8 Conversión dos documentos antigos do CORGA

Ata aquí as explicacións da metodoloxía para a construción do CORGA se desde un comezo se seguise unicamente este proceso. Pero a realidade foi ben distinta. O proxecto naceu no ano 1995 e, nese momento, as tecnoloxías XML non estaban suficientemente maduras como para seren utilizadas tal e como se definen na presente tese. É por isto polo que toda a metodoloxía proposta está completamente operativa soamente desde o ano 2000.

Xa que logo, xurdiu nese momento, e de xeito natural, unha pregunta que repercutiría en diferentes graos sobre a metodoloxía proposta: como facer cos documentos que xa tiñamos dispoñibles para adaptalos á nova estrutura.

[17] De aí a importancia da existencia desta documentación para que todo o equipo faga as cousas do mesmo xeito.

2.8.1 Descrición do formato antigo dos documentos

Desde que comezou o proxecto en 1995 e ata o ano 2000, dispoñíase de dous arquivos para incluír a información correspondente a un único documento. Como se pode ver no exemplo da figura II.2.17, o primeiro arquivo (o arquivo cabeceira), contiña os datos sobre o título, o autor, o ano de publicación, a editorial etc. do documento. A seguir describimos brevemente os dezaseis campos que podía chegar a ter:

```
<#FIELD NAME = TITULO>VG1995-06-29/010</#FIELD>
<#FIELD NAME = AUTOR>Redacción</#FIELD>
<#FIELD NAME = AREF>1995</#FIELD>
<#FIELD NAME = MEDIO>P</#FIELD>
<#FIELD NAME = TEMAS>105</#FIELD>
<#FIELD NAME = LPUB>A Coruña</#FIELD>
<#FIELD NAME = EDITORIAL>La Voz de Galicia</#FIELD>
<#FIELD NAME = NED>01</#FIELD>
<#FIELD NAME = DED>1995/06/16</#FIELD>
<#FIELD NAME = A1ED>1995</#FIELD>
<#FIELD NAME = ISBN>C-2-1958</#FIELD>
<#FIELD NAME = VERSION>01</#FIELD>
<#FIELD NAME = AVER>1999</#FIELD>
<#FIELD NAME = RESP>V</#FIELD>
<#FIELD NAME = SORT>N</#FIELD>
<#FIELD NAME = NOTAS></#FIELD>
```

Figura II.2.17: Exemplo dun documento cabeceira no formato antigo

- TITULO: Contén o título no caso dun libro, un identificador no caso de noticias, *Prólogo a* e nome do libro no caso dun prólogo doutro autor[18], *Apéndice a* e nome do libro no caso dun apéndice, e *Epílogo a* e nome do libro no caso dun epílogo.

- AUTOR: Inclúe o autor ou autores do documento. En caso de que haxa máis dun, cada un está nunha liña e, ao final, hai unha liña co último autor e coa etiqueta de peche (véxase a figura II.2.18).

```
<#FIELD NAME = AUTOR>M.L.
S.D.
R.G.</#FIELD>
```

Figura II.2.18: Exemplo de autores múltiples no formato antigo

- AREF: Contén o ano de referencia, que consiste no ano da edición que se manexa, é dicir, o ano do campo DED.

- MEDIO: Indica o medio e pode ser: P = Prensa escrita, Pe = Prensa electrónica, R = Revista, Re = Revista electrónica, L ou Li = Libro.

[18] Tamén pode aparacer *Limiar a*, *Presentación a* e *Prefacio a*.

- TEMAS: Contén as diferentes áreas temáticas ás que pertence o documento. A sintaxe utilizada é igual á de autor, é dicir, cando hai máis dunha (ata un máximo de tres), hai unha por liña e na última área temática a etiqueta de peche.
- LPUB: Lugar de publicación.
- EDITORIAL: Nome da editorial.
- NED: Número de edición. Normalmente trabállase coa primeira, e nunca se incorporan ao corpus dúas edicións do mesmo documento.
- DED: Data de edición ou publicación.
- A1ED: Ano da primeira publicación (normalmente coincide co ano do campo DED).
- ISBN: ISBN do documento.
- VERSION: Sempre contén o valor 01 e indica que é a primeira versión do documento.
- AVER: Ano da versión do documento.
- RESP: Código identificador da persoa responsable do documento.
- SORT: Indica se o texto é normativo ou non. No caso de que o sexa o campo contén un N.
- NOTAS: Contén algunhas notas sobre o texto, como tendencias do autor ou erros intencionados.

O segundo arquivo, polo contra, contiña o texto do documento cun retorno de carro ao final de cada parágrafo. Adicionalmente, podían estar marcadas as porcións de texto noutra lingua, englobadas entre <COMMENT> e </COMMENT>, e tamén aparecer as notas a pé de páxina ao final do documento despois dunha liña que contivera o texto [NOTAS].

2.8.2 Deficiencias

A estrutura documental antiga presenta certas deficiencias que obrigaron a facer un cambio de tecnoloxía para asegurar que o corpus, e os sistemas que o utilicen, evolucionen axeitadamente no tempo.

Por unha banda, é evidente que existe un problema serio coa composición de documentos. Por exemplo, hai un documento (formado por dous arquivos) por cada noticia dun xornal, pero non existe ningún lugar onde se poida introducir información relativa ao xornal no seu conxunto. O mesmo sucede, por exemplo, coas coleccións de relatos ou ensaios, onde se representan de forma separada cada un dos relatos ou ensaios, o que se fai que se perda a información relativa á colección.

Aínda que moitos sistemas de RI seguen tendo certas dificultades para tratar con estruturas complexas de documentos, xa non existen na representación documental. A perda de información pode limitar as posibilidades de explotación do corpus no futuro, polo que sería un erro prescindir dela.

Por outra banda, pode apreciarse que se utilizan códigos e/ou abreviaturas como valores de certos campos da cabeceira do documento (por exemplo nos campos MEDIO, RESP ...)

e que a estrutura do texto propiamente dita é moi pobre comparada coa que describimos con anterioridade[19]. Isto débese, principalmente, á herdanza de tecnoloxías dispoñibles noutro tempo e xa obsoletas actualmente. Co paso dos anos e, sobre todo, coa chegada do estándar XML, é posible facer operativas representacións documentais complexas, sen que sexa necesario utilizar símbolos ou acoutar a lonxitude dos campos, que tanto tempo limitaron as posibilidades para representar calquera tipo de información.

Por todos estes motivos afrontamos unha transformación dos documentos ao estándar XML, o que nos permitiu considerar todas estas relacións xerárquicas inherentes aos textos e deixar de lado limitacións herdadas de sistemas antigos de representación e procesamento, co fin de garantirmos a conservación de toda esta valiosa información.

2.8.3 Modificación da metodoloxía

O cambio na estrutura documental de que acabamos de falar precisou dun esforzo importante de traballo e influíu, como era de esperar, en diferentes fases da nova metodoloxía.

En primeiro lugar, no momento deste cambio tecnolóxico priorizouse a conversión dos textos que xa tiñamos dispoñibles fronte a unha posible incorporación doutros. Xa que logo, aínda que a metodoloxía exposta tamén é válida para adaptar estoutros documentos ao novo formato, a súa materialización presentou certas peculiaridades. Así, por exemplo, nun primeiro momento a fase de adquisición só consideraba a adaptación de documentos xa existentes no sistema antigo do CORGA; non existía fase de estruturación; na fase de conversión os *scripts* automáticos só tiñan en conta a estrutura que presentaban eses documentos; e na fase de revisión había que comprobar moitas máis cousas, xa que a estrutura documental era moito máis pobre da que posteriormente pasarían a ter os novos despois da fase de estruturación.

En segundo lugar, e como consecuencia do anterior, os protocolos de actuación eran algo diferentes aos que xurdiron posteriormente. Polo tanto, cando se decidiu introducir novos documentos existían dous protocolos diferenciados: un para procesar os xa existentes no formato antigo, e outro para afrontar a conversión dos novos. Estes dous protocolos estiveron vixentes simultaneamente desde o ano 2003, data en que se decide ir incorporando novos documentos de xeito simultáneo á transformación dos antigos, ata o ano 2006, momento en que se rematou con estes últimos.

Ademais, durante todo este período foi moi complicado manter o corpus compensado. Dada a magnitude do traballo, priorizouse facer a reconversión do xeito máis rápido posible e deixar para datas posteriores, é dicir, para a etapa actual, a compensación do corpus. Cómpre destacar que a descompensación máis grande se produciu na información correspondente ao lustro 2004-2008, xa que foi case imposible incorporar documentos desa etapa debido a que todo o equipo de traballo estaba dedicado a converter textos antigos. Porén, os poucos esforzos dispoñibles adicáronse a corrixir descompensacións que xa existían en lustros anteriores.

Daquela, e como cabe esperar, nas primeiras planificacións temporais, e durante un período considerable de tempo, só figuraba a adaptación tecnolóxica de textos xa dispoñibles. Durante estes anos fóronse construíndo os elementos da metodoloxía encamiñados a procesar novos documentos, tales como *scripts* automáticos ou a definición de protocolos e, posteriormente, fóronse poñendo en práctica.

[19] Só se anotan seccións noutra lingua e permítese a inclusión de notas, pero non se poden delimitar oracións ou seccións dentro do texto.

No ano 2006 rematouse completamente coa transformación dos textos antigos do proxecto e, actualmente, xa están en fase de desenvolvemento textos para cubrir a lagoa temporal existente desde o 2004, polo que é previsible que nos próximos anos o equipo do proxecto traballe ao día incorporando documentos do lustro vixente e incluso poida incorporar outros novos que compensen pequenas deficiencias dos lustros anteriores.

Xa a modo de conclusión, debemos dicir que o equipo de traballo está convencido de que o camiño que seguimos garante a evolución axeitada do corpus. Ben é certo que se investiron moitos esforzos para facer o cambio ao estándar XML, pero os beneficios obtidos fixeron que pagase a pena.

Malia que a nova estrutura documental utilizando XML deixa ver parte destes beneficios, como puidemos comprobar anteriormente, onde quedan máis patentes é nos sistemas de RI que permiten facer buscas sobre o corpus, que é no que se centra a segunda parte desta tese de doutoramento. Como se poderá apreciar, existirán diferenzas cualitativas considerables entre o sistema de RI desenvolvido para consultar a base documental nova do CORGA e a antiga, e confirmarase que, esta nova maneira de representar os textos ofrece moitas posibilidades de busca que doutro xeito serían inabarcables.

Parte III

Sistemas de recuperación de información

Capítulo 1

Requirimentos xerais

Nesta parte da tese tratamos a construción de sistemas de RI lingüísticos que utilizan corpus de grandes dimensións, en que os documentos están estruturados seguindo o estándar XML e non se inclúe ningunha información morfosintáctica.

Terán especial relevancia, entón, a obtención de frecuencias de ocorrencia de palabras e a visualización dos exemplos concretos no seu contexto, mais tamén será crucial manexar todos os ingredientes axeitadamente para que o rendemento do sistema cumpra as expectativas dos usuarios, normalmente lingüistas. Algúns destes corpus son moi extensos e inclúen centos ou miles de documentos que, á súa vez, conteñen decenas ou centos de millóns de palabras, polo que se fai necesaria a construción de sistemas de RI eficientes e flexibles que traballen con eles.

En calquera caso, os asuntos tratados nesta investigación tamén serán útiles, non só á hora de afrontar problemas similares en corpus etiquetados, senón tamén para outros que non estean no formato XML, xa que se os documentos están ben estruturados poden ser convertidos de maneira sinxela a este estándar.

Neste traballo, cando falamos de sistemas de RI englobamos un amplo abano de tecnoloxías que teñen por obxecto obter, a partir dunha busca, a información relevante para un usuario dentro dunha base documental. É dicir, empregamos o concepto de sistema de RI no seu sentido máis amplo, polo que analizamos catro tipos de tecnoloxías que poden ser utilizadas:

1. Por unha banda temos os sistemas de RI convencionais [32], tamén chamados indexadores de texto, que tratan cos documentos planos ou cunha estrutura moi sinxela. Este tipo de sistemas son capaces de executar consultas dunha maneira moi eficiente grazas á utilización de índices invertidos [32, 94], pero non teñen en conta posibles marcas que delimiten diferentes seccións ou se o fan, presentan esta capacidade moi limitada.

2. Pola outra, están as aplicacións lingüísticas propiamente ditas. Trátase de tecnoloxías en que, xa desde un comezo, se tiveron en conta as necesidades dos sistemas que aquí nos ocupan. Realmente poderían considerarse un subtipo dos anteriores, xa que normalmente utilizan índices invertidos para levar a cabo as buscas pero, dado que presentan algunhas peculiaridades propias dos sistemas lingüísticos, analizarémolas individualmente.

3. A continuación, están os xestores de bases de datos relacionais, que permiten xestionar e consultar practicamente calquera tipo de información. Resolven as necesidades dunha gran cantidade de sistemas de información e, cada vez máis, inclúen funcionalidades non asociadas ao mundo relacional, como a posibilidade de utilizar índices textuais.

4. E, por último, os modernos xestores de documentos XML [22] constitúen a base para o desenvolvemento de sistemas de xestión e busca documental, polo que tamén son incluídos nesta clasificación. Ao presentaren a particularidade de tratar cos documentos XML directamente, non é necesario facermos ningún tipo de conversión no caso de dispoñer os documentos estruturados seguindo ese estándar.

O que motivou a realización deste segundo gran bloque nos termos en que nós o propoñemos foi que, aínda que si existen diferentes exemplos de sistemas de RI baseados en corpus [21, 26] e de métodos de busca que se poden empregar en corpus extensos [26], ou incluso algúns traballos [22] introducen unha revisión de diferentes tecnoloxías que poden utilizarse para construír sistemas de RI xenéricos baseados en XML, non atopamos ningunha análise comparativa ou estudo contrastivo sobre tecnoloxías que poidan ser utilizadas para construír sistemas de RI baseados en corpus estruturados de grandes dimensións, nin ningún traballo que marque unhas pautas a seguir para desenvolver este tipo de sistemas.

Polo tanto, neste capítulo comezaremos por expor cales son os requirimentos que adoitan compartir os sistemas de RI que estudamos para, nos posteriores, analizar diferentes tecnoloxías que poden ser empregadas para construílos (coas súas vantaxes e limitacións), facer un estudo de rendemento das máis representativas e, finalmente, explicar os elementos clave para o seu desenvolvemento nun caso concreto, o do proxecto CORGA.

1.1 Requirimentos

O que se pretende conseguir coa construción de sistemas de RI que utilicen un corpus varía para cada caso particular e abrangue propostas de moi diversa complexidade. Un caso sinxelo sería o desenvolvemento dunha ferramenta que permita localizar o conxunto de documentos do noso corpus que conteñan unha palabra de busca, mentres que outro máis complexo podería ser o de crear un contorno que nos dea moitas e variadas posibilidades de busca diferentes incluída, por exemplo, a de consultar algún caso de ocorrencia nunha sección concreta dos documentos, combinada con diferentes criterios de busca por data, autor ou área temática. No noso caso centrarémonos en sistemas de RI orientados a lingüistas que desexaren facer diferentes estudos a partir da consulta dun corpus.

Dado que cantas máis posibilidades de busca se ofrezan, máis información poden obter os usuarios, no noso caso lingüistas, o ideal en moitos casos é construír ferramentas versátiles que permitan unha gran variedade de consultas. A seguir describimos a nosa proposta das características que é esperable que teñan moitos sistemas de RI deste tipo e que nos servirá, nos capítulos seguintes, para ver en que grao as tecnoloxías máis representativas na actualidade poden desenvolver cada unha delas:

1. **Buscas textuais**[1]: Dentro das buscas textuais, distinguimos:
   (a) Buscas de coincidencia exacta. É dicir, amosan os casos que coincidan exactamente coa cadea de busca.
   (b) Buscas sensibles e non sensibles aos tiles. Permiten decidir se para facer as buscas se deben ter en conta ou non os tiles. Por exemplo, nunha busca non sensible aos tiles

[1] Do inglés *full-text*.

onde se busca a palabra *camión*, deberían obterse tamén os casos onde apareza escrita a palabra *camion* sen o til. A utilidade principal das buscas non sensibles aos tiles radica na posibilidade de incluír erros ortográficos nos resultados dunha busca.

(c) Buscas sensibles ou non ás maiúsculas. É o mesmo caso que o anterior, pero referido ás maiúsculas e ás minúsculas. As consultas sensibles ás maiúsculas permiten detectar nomes propios no texto, mentres que as non sensibles evitan o problema da imposibilidade de que as palabras ao comenzo de oracións saian nos resultados.

(d) Unificación de caracteres. Posibilita unificar diferentes caracteres, é dicir, trátase de facer o mesmo que nas buscas sensibles ou non sensibles aos tiles, pero sen que un carácter teña que ser o mesmo que outro con til. Sería o caso, por exemplo, dun corpus de textos en latín en que se desexan facer consultas unificando os caracteres *u* e *v*.

(e) Buscas booleanas. Permiten utilizar operadores booleanos, como por exemplo buscar os documentos en que aparezan dúas palabras concretas.

(f) Buscas de proximidade. Isto é, buscas sobre palabras que están separadas menos dunha distancia dada, como por exemplo, buscar os documentos onde aparezan as palabras *camión* e *tonelaxe* separadas por unha distancia de menos de cinco palabras.

(g) Exclusión de texto marcado das buscas. É dicir, permite ignorar fragmentos de texto á hora de facer as buscas. Tipicamente esta opción utilízase para excluír das buscas palabras, expresións, ou incluso fragmentos de texto de certa extensión que están nunha lingua diferente á considerada polo corpus. Se a frecuencia de aparición de fragmentos de textos noutra lingua é elevada e hai palabras comúns entre as dúas linguas, a obxecto do corpus e a dos fragmentos diferentes, as conclusións extraídas a partir das consultas poden ser erróneas. A marcaxe destes fragmentos para evitar a súa indexación arranxaría este problema.

(h) *Stop words*. Posibilita excluír certas palabras de uso frecuente á hora de facer as buscas. Á diferenza doutro tipo de sistemas de busca, nos lingüísticos adoita ser útil poder especificar unha lista de *stop words* baleira xa que, ás veces, se desexa información relacionada con palabras que a miúdo están incluídas nestas listaxes.

(i) Múltiples modos de busca simultáneos. Hai tecnoloxías que permiten múltiples modos de busca simultáneos e ofrecen a posibilidade de que o usuario poida escoller. Outras, en cambio, aínda que ofrecen a posibilidade de utilizar varios modos de busca, estes non poden estar en funcionamento simultaneamente e teñen que ser establecidos a priori polo equipo informático.

(j) Configuración dos caracteres que forman parte das palabras. En ocasións hai caracteres que interesa que non formen parte das palabras do índice de busca e, noutros casos, que si. Pode ser o caso do guión (-), que no galego préstase a que forme parte das palabras para poder buscar formas verbais coa segunda forma do artigo e, no castelán, en ocasións non, xa que así se poden localizar individualmente os dous compoñentes dunha palabra composta por dúas separadas co guión.

Outro exemplo podería ser a utilización desta característica para un sistema de consulta de textos antigos. A miúdo nestes textos aparecen marcas dentro das palabras entre corchetes ou chaves para reflectir que se fixo unha reconstrución desa parte da

palabra. O que interesa habitualmente nestes casos é poder buscar a palabra completa reconstruída (sen os corchetes), pero á hora de amosar o texto ver esas marcas.

(k) Utilización de comodíns: Utilización de comodíns ou expresións regulares nas buscas.

(l) Resaltado: Existe a posibilidade de resaltar as palabras que coincidiron cunha busca concreta. Habitualmente, cando se fan consultas a sistemas deste tipo, deséxase ver as palabras que verificaron a expresión da consulta destacadas sobre as que a arrodean. Isto fai que a exploración visual dos resultados sexa moito máis fluída.

2. **Capacidades estatísticas**: Consisten en obter valores numéricos a diferentes niveis como, por exemplo, contar o número de casos e de documentos dunha consulta.

3. **Información adicional**: Trátase de ofrecer información adicional sobre os resultados obtidos. Por exemplo, non só obter os casos que verifican os criterios da busca, senón tamén amosar o autor ou a editorial, é dicir, trátase de visualizar datos adicionais obtidos da estrutura dos documentos do corpus.

4. **Contexto**: Alén dos casos coincidentes dunha busca, pode ser posible amosar o contexto en que aparecen. Denomínase tamén palabra no seu contexto (*Key Word In Context*, KWIC) e permítenos ver as n palabras á esquerda e á dereita da expresión coincidente coa busca.

5. **Independencia de seccións**: Consiste en poder facer buscas unicamente en determinadas seccións dos documentos, ou mesmo en zonas marcadas dentro do propio texto. Un exemplo do primeiro caso sería ofrecer ao usuario a posibilidade de facer buscas, ben nos titulares das noticias do xornal, ben no texto completo. O segundo caso sería útil, por exemplo, para facer buscas de nomes propios que están marcados no texto.

6. **Xogos de caracteres**: Os sistemas deben traballar axeitadamente co xogo de caracteres que se necesita para os documentos. Entres os xogos de caracteres máis demandandos están os estándares ISO (ISO-8859) [50] e Unicode (UTF-8) [91].

7. **Navegación polos resultados**: Posibilita navegar entre as diferentes páxinas de resultados. É habitual que as ferramentas de RI requiran algún tipo de navegación entre páxinas e que non sexa factible devolver todos os resultados á vez. É por isto polo que resulta interesante ter un mecanismo para ir obtendo os datos páxina a páxina.

8. **Ordenación**: Deixa ordenar os resultados utilizando un ou varios criterios a un tempo. Tamén é interesante que ofreceza a opción de ordenar os resultados alfabeticamente segundo as cadeas que coinciden coa busca cando se utilizan comodíns.

9. **Relacións estruturais**: Flexibiliza os sistemas de xeito que se poden incluír relacións estruturais de certa complexidade nas consultas. Por exemplo, pódense obter os casos que coincidan cunha expresión de busca dentro dun subconxunto de documentos dun certo período e área temática.

10. **Refinamento de consultas**: Posibilita refinar as buscas, é dicir, facer unha busca que actúe sobre o resultado doutra.

11. **Organizar resultados**: Permite configurar como se visualizan os resultados, por exemplo, determinar o número de casos que se amosan por cada páxina.

12. **Gardar resultados**: Fai que os resultados poidan ser gardados para un posterior procesamento local.

13. **Eficiencia**: Deixa que o sistema sexa suficientemente eficiente como para que os usuarios estean satisfeitos co seu comportamento. Debe sinalarse que este tipo de sistemas se caracteriza por requirir eficiencia unicamente na consulta dos datos, de modo que resulta pouco importante o tempo de introdución ou actualización. Normalmente actualízanse os datos periodicamente e dunha soa vez e, a partir de aí, fanse unicamente consultas.

14. **Nómina**: Permite obter datos relativos ao subconxunto de documentos que se está a consultar. Nesta epígrafe inclúese a posibilidade de obter a relación de documentos que contén o corpus, o número de palabras total dos documentos que verifican uns parámetros etc. é dicir, trátase de poder acadar uns referentes sobre os que comparar os resultados dunha busca para dispoñer dunha maneira de obter frecuencias normalizadas.

Capítulo 2

Avaliación de tecnoloxías

Aínda que no noso caso partimos de documentos que están estruturados segundo o estándar XML, isto non ten por que condicionar que a tecnoloxía que se utilice para construír o sistema de RI trate directamente con este estándar. Un documento XML pode ser convertido a outras moitas e diversas estruturas [22, 88, 102], polo que hai que avaliar ferramentas de moi diversa índole.

Para facermos esta avaliación concentrámonos nalgúns dos principais expoñentes no mercado e intentamos primar os baseados en software libre. Como referente comercial utilizamos Oracle [60] que, como comprobaremos, cobre en gran medida os requirimentos propostos no capítulo anterior e, a partir de aí, intentamos atopar unha tecnoloxía de software libre que igualase ou superase as súas prestacións.

En primeiro lugar, analizamos cada unha das tecnoloxías por separado e, posteriormente, facemos unha comparativa global que nos dea unha visión de conxunto sobre cales son as máis axeitadas.

Ademais, á hora de construír o sistema de RI son necesarios dous elementos: a tecnoloxía que dará soporte aos datos e a aplicación que accederá a ela para obter os resultados. Algúns dos requirimentos presentados no capítulo anterior dependen en certa medida da segunda máis ca da primeira, concretamente os relacionados coas epígrafes 12 e 13, polo que non serán utilizados para a comparativa. En calquera caso estudaranse as facilidades que se ofrezan para materializar estas necesidades.

2.1 Indexadores de texto

Aínda que a súa orixe conceptual é un pouco anterior, os indexadores de texto propiamente ditos non comezan a ser útiles ata a década dos 80, en plena explosión dos computadores persoais. Baséanse na utilización de índices invertidos, que xorden como unha necesidade para sacar partido da información contida en textos non estruturados mediante a localización neles das palabras de busca.

No mercado actual existen diversos indexadores de texto: Lucene [2], dtSearch [29], Swish-e [49, 69], SMART [76], Sphinx [82] etc. Porén, a maioría deles céntrase na obtención dunha listaxe ordenada de documentos relevantes con respecto á busca realizada. É dicir, o resultado principal destes sistemas é un conxunto de documentos que trata sobre algún tema, organizado segundo a relevancia dos textos para a consulta. Isto fai que, con eles, en moitos casos resulte especialmente complicado que se satisfagan diferentes aspectos dos incluídos na nosa

proposta de requirimentos, xa que esta non se centra na obtención dunha listaxe de documentos senón de casos.

Por exemplo, se tratamos de obter o número de palabras que coinciden coa busca, suporía obter primeiro os documentos que coinciden coa busca para, posteriormente, percorrer cada un deles e ir contando os termos coincidentes, o que normalmente ten un custo elevado de rendemento. Isto ocorre, por exemplo, en Lucene.

Pola contra, existen uns poucos sistemas de indexación de texto, como dtSearch, que incorporan unha funcionalidade básica para darlles resposta ás condicións expostas. Iso si, incluso estes sistemas teñen un inconveniente: debido a que a estrutura documental que manexan é moi simple non permiten, entre outras cousas, xestionar axeitadamente estruturas aniñadas neles.

Xa que logo, optar por algunha destas tecnoloxías para desenvolver os sistemas que nos ocupan, sobre todo se é unha plataforma de pagamento, ten un certo risco. A súa principal limitación é perfectamente coñecida: non permiten traballar con documentos de relativa complexidade estrutural e se, como ocorre en ocasións, nalgún momento da evolución dos corpus cremos que é necesaria unha estrutura un pouco máis elaborada dos documentos, a tecnoloxía empregada limitaranos as posibilidades de busca a non moi longo prazo.

2.2 Aplicacións lingüísticas

As primeiras aplicacións lingüísticas de consulta de corpus nacen practicamente ao tempo que a informática doméstica, na década dos 80, pouco despois da aparición dos primeiros computadores persoais. Foron moitas e moi variadas as aplicacións que se utilizaron neste ámbito: Tact [90], OCP [62], MonoConc [7], WordSmith tools [63] etc., a maioría delas monolíticas, que non naceron na era de Internet e que, polo tanto, non tiveron en mente o concepto de aplicación en rede que se ten agora.

Nesta epígrafe describimos dúas destas aplicacións: unha pola súa relevancia histórica (WordSmith tools) e outra polo seu potencial (Sketch Engine [51, 52]), xa que amplía de diversos xeitos os postulados da primeira.

2.2.1 WordSmith tools

WordSmith tools é un clásico dentro das ferramentas de análise de corpus. Foi unha das primeiras empregadas para facer estudos lingüísticos similares aos que se pretenden conseguir cos sistemas de RI tratados no presente traballo. Publicada por Oxford University Press desde 1996, WordSmith tools foi evoluíndo incorporando novas características ata a súa última versión, a 5.0, que saíu en 2007.

Efectivamente, esta ferramenta cobre moitos dos aspectos relacionados con amosar os casos dunha palabra no seu contexto de aparición, pero está pensada como aplicación autónoma monousuario. Ademais, en xeral, obvia os datos de alto nivel dos documentos de feito que, por exemplo, se nos textos temos un campo indicando o seu ano de publicación e necesitamos facer unha busca en documentos comprendidos nun rango de anos específico, é necesario que nós, fóra do seu contorno, realicemos unha preselección dos textos que encaixen nese rango.

Daquela, aínda que esta ferramenta é das máis versátiles en moitos aspectos relacionados co tratamento puramente textual, o certo é que non cumpre dúas premisas fundamentais que

buscamos no presente estudo: que sexa un sistema de consulta en rede e que contemple a estrutura de alto nivel dos documentos.

2.2.2 Sketch Engine

O sistema Sketch Engine xa é moito máis recente que o anterior. Dáse a coñecer no ano 2004 e cobre aspectos que xa trataba WordSmith tools máis outras moitas posibilidades relacionadas coa etiquetaxe morfosintáctica. En definitiva, non só se centra na explotación de corpus textuais sen anotar, senón que o seu maior potencial é o aproveitamento da información morfosintáctica asociada ás palabras.

O Sketch Engine si foi concibido para ser utilizado en rede e a súa licenza dá acceso a unha serie de corpus preestablecidos. Ademais permite traballar cun corpus propio. Porén, esta ferramenta centra a súa funcionalidade ao redor do que na súa terminoloxía denominan *word sketches*, ou contextos de aparición das palabras[1] e, ao igual que no caso de WordSmith, para facer buscas sobre un conxunto de documentos que cumpra uns criterios é necesario crear ese subcorpus de consulta.

2.3 Bases de datos relacionais

A construción dun sistema de RI utilizando un xestor relacional pode facerse de moi diversas formas. Aínda así, podemos clasificar todas estas posibilidades en dúas alternativas: as que non necesitan dividir as palabras dos documentos illadamente nunha táboa, e as que si o fan.

A estrutura básica da primeira opción estaría formada por unha única táboa que incluiría, tanto os datos da cabeceira dos documentos, coma o texto propiamente dito. Na segunda alternativa habería, polo menos, unha táboa adicional relacionada coa primeira que incluiría as palabras dos textos. A partir destas dúas arquitecturas básicas hai outras moitas intermedias que combinarán en diferentes grao as cuestións aquí tratadas para resolver cada caso particular.

Polo tanto, para o primeiro caso teriamos a seguinte definición de táboas[2]:

```
CREATE TABLE documento (
 identificador INT,
 editorial VARCHAR(128),
 texto CLOB,
 CONSTRAINT doc_pk PRIMARY KEY (identificador)
);
```

E, para o segundo, esta[3]:

```
CREATE TABLE documento (
  identificador INT,
  editorial VARCHAR(128),
  texto CLOB,
```

[1] Neste caso o concepto de *word sketch* é un pouco máis amplo que o de KWIC, xa que implica tamén a información morfosintáctica das palabras.

[2] A sintaxe SQL utilizada neste caso é a utilizada por Oracle, que será adaptada ás demais tecnoloxías presentadas en caso de ser necesario.

[3] Neste caso materializamos unha relación 1:N entre as táboas *palabra* e *documento*, pero sería perfectamente válida tamén unha relación N:M.

```
  CONSTRAINT doc_pk PRIMARY KEY (identificador)
);

CREATE TABLE palabra (
  identificador INT,
  identificador_documento INT,
  palabra VARCHAR(128),
  posicion INT,
  CONSTRAINT foreign_documento FOREIGN KEY (identificador_documento)
    REFERENCES documento (identificador),
    ON DELETE CASCADE,
    CONSTRAINT palabra_pk PRIMARY KEY (identificador)
);
```

2.3.1 Oracle

Oracle Corporation [60] é unha das principais compañías intrinsicamente relacionadas cos xestores de bases de datos. Podemos situar o seu nacemento no ano 1977 e a súa evolución estivo intimamente relacionada co desenvolvemento de xestores de bases de datos, particularmente de xestores relacionais.

Desde a súa orixe ata a actualidade, o seu xestor de base de datos (Oracle) foi evoluíndo tecnoloxicamente ata a versión actual, a 11g, que xa presenta numerosas características non relacionadas directamente cos xestores relacionais por necesidades tecnolóxicas e do mercado.

As novas tecnoloxías están avanzando a pasos axigantados e hai moitas funcionalidades que se van engadindo ás versións modernas deste prolífico xestor. Nesta tese utilizamos a versión 11g Release 1 (11.1.0.6.0) do xestor, que é a última dispoñible no momento en que escribimos estas liñas (febreiro de 2009).

O xestor Oracle presenta a alternativa de poder traballar cos dous enfoques expostos anteriormente, polo que expoñemos aquí as posibilidades que ofrece para cada un deles.

Buscas textuais

(a) Buscas de coincidencia exacta

A mellor maneira de facer buscas de coincidencia exacta con Oracle é empregando a función `CONTAINS`, que presenta a particularidade de que non busca as cadeas no propio texto, senón sobre un índice que contén as súas palabras, polo que resulta especialmente útil nos sistemas de RI que nos ocupan.

Se nos situamos no primeiro suposto presentado anteriormente, podemos introducir os documentos pertinentes na táboa e definir un índice que nos permita facer estas buscas como segue:

```
begin
  ctx_ddl.create_preference('lexico','BASIC_LEXER');
  ctx_ddl.set_attribute('lexico', 'BASE_LETTER', 'NO');
end;
/

CREATE INDEX textindex ON documento(texto)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('LEXER lexico');
```

Primeiro defínese un `LEXER` que indica como se construirá o índice (neste caso sinalando que non se faga ningunha transformación nas palabras) e despois créase o propio índice. Poderemos consultar os identificadores de documento que conteñan unha expresión de busca do seguinte xeito:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (texto, 'expresión')>0;
```

Se, pola contra, estamos no segundo suposto e queremos obter non só os documentos, senón tamén a posición en que ocorren as palabras que coinciden coa consulta para facer o resaltado posterior do resultado na aplicación, hai que utilizar a columna *posicion*. Así, se buscásemos unha expresión formada por *termo1* seguido de *termo2*, a consulta sería a seguinte:

```
SELECT documento.identificador, palabra1.posicion
FROM   documento, palabra palabra1, palabra palabra2
WHERE  palabra1.identificador_documento = documento.identificador
       AND palabra2.identificador_documento = documento.identificador
       AND palabra2.posicion=palabra1.posicion+1
       AND CONTAINS (palabra1.palabra, 'termo1')>0
       AND CONTAINS (palabra2.palabra, 'termo2')>0;
```

Ademais, para axilizar o tempo de resposta da consulta resulta de utilidade, tanto crear un índice sobre a columna *posicion*, como crear un índice funcional baseado en *posicion+1*:

```
CREATE INDEX palabra_posicion
  ON palabra (posicion);

CREATE INDEX palabra_posicion_func
  ON palabra (posicion+1);
```

Isto tamén se pode conseguir se utilizamos para a posición das palabras números consecutivos, de xeito que non se volva comezar cando se cambia de documento e que salte un número cando si cambie. Así podemos especificar un índice de tipo `UNIQUE` para *posicion* que permita resolver máis rapidamente certo tipo de consultas.

(b) Buscas sensibles e non sensibles aos tiles

Para que se poidan facer buscas sensibles ou non aos tiles hai que especificalo no `LEXER` utilizado na creación do índice de texto. Así, no exemplo anterior, as buscas que se fixeron son sensibles aos tiles, xa que o parámetro `BASE_LETTER` está posto a `NO`. Este parámetro é o encargado de decidir se na construción do índice se cambia cada letra pola súa letra base (sen til). Para realizar buscas non sensibles aos tiles habería que poñelo a `YES` e indicarlle a través da variable de contorno `NLS_LANG` que a lingua é `SPANISH` [4]. Se, ademais, queremos que cando fagamos consultas non sensibles aos tiles non trate o *ñ* e o *n* como a mesma letra, é necesario incorporar na definición do `LEXER` a variable `BASE_LETTER_TYPE`. Xa que logo, a definición do índice do exemplo anterior para este caso quedaría:

[4] A versión actual de Oracle non inclúe o galego entre as súas linguas, pero as regras que ten incorporadas para o español no tocante á codificación e manipulación de caracteres son perfectamente válidas para o galego.

```
begin
  ctx_ddl.create_preference('lexico','BASIC_LEXER');
  ctx_ddl.set_attribute('lexico', 'BASE_LETTER', 'YES');
  ctx_ddl.set_attribute('lexico','BASE_LETTER_TYPE', 'SPECIFIC');
end;
/
```

(c) Buscas sensibles ou non ás maiúsculas

De maneira similar, pódese utilizar o atributo `MIXED_CASE` do `BASIC_LEXER` para especificar se o índice incluirá as palabras tal cal aparecen no texto ou se son convertidas a minúsculas. Se non se indica nada para o parámetro ou se pon a `NO`, o índice non será sensible ás maiúsculas, é dicir, poderanse atopar indistintamente palabras que están en maiúsculas ou minúsculas.

(d) Unificación de caracteres

Oracle permite definir xogos de caracteres propios e regras particulares de conversión a letra base utilizando o construtor de xogos de caracteres (Locale Builder). Así, por exemplo, para algún sistema de RI que utilice textos medievais ou manuscritos antigos podería interesar tratar os caracteres u e v indistintamente. Polo tanto, o que habería que facer utilizando esta ferramenta sería crear un novo xogo de caracteres onde a letra base para u, U, v e V sexa u en todos os casos. Finalmente só especificariamos na variable `NLS_LANG` o nome que lle demos a este xogo de caracteres cando se acceda á base de datos, estableceriamos os atributos `BASE_LETTER` e `BASE_LETTER_TYPE` tal e como fixemos anteriormente e volveriamos crear o índice de texto. Deste xeito pódense facer correspondencias de ata dous caracteres, como por exemplo homoxeneizar $\tilde{n}$ e nh.

(e) Buscas booleanas

A función `CONTAINS` pode conter expresións booleanas. Así, para obter os identificadores dos documentos que conteñen *termo1* e *termo2*, fariamos:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (texto, 'termo1 AND termo2')>0;
```

(f) Buscas de proximidade

Se traballamos co primeiro suposto, pódense facer buscas que consideren a proximidade entre dúas palabras utilizando o operador `NEAR`. Por exemplo, a busca:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (texto,'NEAR((termo1,termo2),2)')>0;
```

serviríanos para procurar os documentos en que aparezan *termo1* e *termo2* a unha distancia de dúas ou menos palabras.

No caso de estar traballando no segundo suposto, podemos xestionalo actuando sobre os valores de posición:

```
SELECT identificador
FROM   documento, palabra palabra1, palabra palabra2
```

```
WHERE  palabra1.identificador_documento = documento.identificador
       AND palabra2.identificador_documento = documento.identificador
       AND CONTAINS (palabra1.palabra,'termo1')>0
       AND CONTAINS (palabra2.palabra,'termo2')>0
       AND palabra1.posicion < palabra2.posicion+2;
```

(g) Exclusión de texto marcado das buscas

Para prescindir destas porcións de texto delimitadas é necesario facelo na construción do índice. Así, para evitar que se poida buscar en zonas de texto delimitadas polas etiquetas <d> e </d>, habería que establecer a seguinte definición:

```
begin
   ctx_ddl.create_section_group ('seccion', 'BASIC_SECTION_GROUP');
   ctx_ddl.add_field_section ('seccion','noindex','d',FALSE);
end;
```

e utilizala na construción do índice do seguinte xeito:

```
CREATE INDEX textindex ON texto(texto)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('SECTION GROUP seccion');
```

(h) *Stop words*

Oracle permite facer listaxes de *stop words* personalizadas. Para que se poidan facer buscas referentes a calquera palabra, é dicir, utilizar unha listaxe de *stop words* baleira, débese crear a seguinte definición:

```
begin
  ctx_ddl.create_stoplist('lista_stop', 'BASIC_STOPLIST');
end;
/
```

que será necesario utilizar na construción do índice:

```
CREATE INDEX textindex
  ON documento(texto)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('STOPLIST lista_stop');
```

(i) Múltiples modos de busca simultáneos

Anteriormente vimos como en Oracle se poden definir índices sensibles e non sensibles aos tiles pero, que ocorre se queremos que sexan os usuarios os que decidan que tipo de consulta queren facer?

Unha alternativa sería duplicar a columna que contén o texto do documento, crear un índice sensible aos tiles sobre unha columna, outro índice non sensible aos tiles sobre a outra e, finalmente, que a aplicación consulte unha ou outra columna en función da entrada do usuario.

Porén, Oracle inclúe un mecanismo mediante o cal se poden utilizar dous índices sen termos que repetir todo o contido da columna relevante. Consiste na utilización dunha columna artificial que obtén o seu contido doutra. Para definir dous índices de texto deste xeito, primeiramente engadímoslle unha columna á táboa dos documentos, de modo que o código para a súa creación queda tal e como se amosa a continuación:

```
CREATE TABLE documento (
  identificador INT,
  texto CLOB,
  textononsensible CHAR(1),
  CONSTRAINT documento_id PRIMARY KEY (identificador)
);
```

A seguir creamos un `LEXER` non sensible aos tiles e outro sensible aos tiles tal e como fixemos anteriormente:

```
begin
  ctx_ddl.create_preference('lexicons','BASIC_LEXER');
  ctx_ddl.set_attribute('lexicons', 'BASE_LETTER', 'YES');
  ctx_ddl.set_attribute('lexicons','BASE_LETTER_TYPE', 'SPECIFIC');
end;
/

begin
  ctx_ddl.create_preference('lexico','BASIC_LEXER');
  ctx_ddl.set_attribute('lexico', 'BASE_LETTER', 'NO');
end;
/
```

Logo de definirmos a táboa é necesario indicarlle ao sistema que a columna *textononsensible* obterá os seus datos da columna *texto*. En primeiro lugar creamos o procedemento que nos vai permitir facelo e dámoslle permisos de execución ao usuario `CTXSYS`:

```
CREATE PROCEDURE obtentexto(rid in rowid, tlob in out clob) is
  begin
      for c1 in (SELECT texto
           FROM   documento
           WHERE rowid = rid)
      loop
        dbms_lob.writeappend(tlob, length(c1.texto), c1.texto);
      end loop;
  end;
/

GRANT EXECUTE ON obtentexto TO CTXSYS;
```

Agora debemos desenvolver outro procedemento como usuario `CTXSYS` que chama ao anterior, e dámoslle permisos de execución ao usuario que vaia crear o índice, no noso caso *probas*:

```
CREATE PROCEDURE s_obtentexto(rid in rowid, tlob in out clob) is
begin
  probas.obtentexto(rid, tlob);
end;
/

GRANT EXECUTE ON s_obtentexto TO probas;
```

A continuación, definimos a preferencia `USER_DATASTORE`, xa como usuario *probas*:

```
begin
  ctx_ddl.create_preference('almacenamento', 'USER_DATASTORE');
  ctx_ddl.set_attribute('almacenamento',
                        'procedure',
                        'CTXSYS.s_obtentexto');
  ctx_ddl.set_attribute('almacenamento', 'output_type', 'CLOB');
end;
/
```

inserimos os datos correspondentes na táboa dos documentos e, finalmente, construímos o índice non sensible aos tiles

```
CREATE INDEX textindexnonsensible
  ON documento(textononsensible)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE almacenamento LEXER lexicons');
```

Na columna *texto* pódese construír o índice sensible de xeito normal:

```
CREATE INDEX textindex
  ON documento(texto)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('LEXER lexico');
```

Polo tanto, para facer en tempo de execución unha consulta sensible aos tiles fariamos:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (texto,'expresión')>0;
```

e para facela non sensible aos tiles:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (textononsensible,'expresión')>0;
```

Deste xeito podemos crear diferentes índices textuais sobre unha mesma columna e permitimos así, trasladarlle ao usuario a opción do tipo de busca que desexa facer en cada momento. Todo isto sen repetirmos o texto en diferentes columnas.

(j) Configuración dos caracteres que poden forman parte das palabras

Oracle ofrece varios atributos para especificar no `LEXER` como deben tratarse algúns caracteres especiais. Destacamos dous: `SKIPJOINS`, que permite definir os que poden aparecer dentro das palabras e que deben ignorarse á hora de construír o índice textual, e `PRINTJOINS`, que permite especificar os que poden aparecer dentro das palabras e que deben considerarse como parte súa.

Así, para definir un índice textual que indexe as palabras ignorando os corchetes e considerando o guión como parte delas, definiriamos un `LEXER` do seguinte xeito:

```
begin
  ctx_ddl.create_preference('lexico','BASIC_LEXER');
  ctx_ddl.set_attribute('lexico', 'BASE_LETTER', 'YES');
```

```
  ctx_ddl.set_attribute('lexico','BASE_LETTER_TYPE', 'SPECIFIC');
  ctx_ddl.set_attribute('lexico','SKIPJOINS','[]');
  ctx_ddl.set_attribute('lexico','PRINTJOINS','-');
end;
/
```

E, posteriormente, deberiamos construír o índice textual utilizando este `LEXER` tal e como xa fixemos con anterioridade.

(k) Utilización de comodíns

Oracle ofrece dous comodíns que poden empregarse dentro da sentencia `CONTAINS`: % e _. O primeiro deles substitúe cero ou máis caracteres e o segundo un único carácter. Así, a consulta

```
SELECT id
FROM   documento
WHERE  CONTAINS (texto, 'desenvolv\%')>0
```

obtería os identificadores dos documentos que conteñen no seu texto as palabras: *desenvolver*, *desenvolvendo*, *desenvolvido*, *desenvolto*, *desenvolvo* etc. Mentres que a consulta

```
SELECT id
FROM   documento
WHERE  CONTAINS (texto, 'desenvolv\_')>0
```

só obtería os identificadores dos documentos que conteñen no seu texto as palabras: *desenvolvo*, *desenvolva* etc., pero non outras como *desenvolto* ou *desenvolvido*.

Aínda que o xestor presenta posibilides de busca de expresións regulares, estas non están integradas dentro dos índices textuais que se poden utilizar coa función `CONTAINS`, polo que non poden empregarse con ela. Se resultase imprescindible a súa utilización, teriamos que recorrer á segunda aproximación para facer as consultas sobre a táboa de palabras, e empregar a cláusula `LIKE`, que permite o uso de expresións regulares. Se así for habería que buscar solucións alternativas aos problemas expostos anteriormente (exclusión de texto das buscas, *stop words* etc.) que puideran interferir nos resultados esperados. Debemos recordar que a lexicalización do texto só se pode utilizar coa función `CONTAINS` que, desde logo, é o mecanismo máis axeitado para resolver a problemática dos sistemas de RI obxecto de estudo.

(l) Resaltado

No primeiro suposto Oracle ofrece tres posibilidades para destacar as palabras que coinciden cunha busca. A primeira, `CTX_DOC.MARKUP`, permite obter unha versión resaltada do documento que se lle pasa, a segunda, `CTX_DOC.HIGHLIGHT`, unha serie de desprazamentos que poden ser utilizados para facer o resaltado na aplicación e, a terceira, `CTX_DOC.SNIPPET`, as palabras destacadas que coinciden coa busca dentro do contexto en que aparecen. Tanto `CTX_DOC.MARKUP` como `CTX_DOC.SNIPPET` permiten definir as marcas que utilizará o xestor para facer o resaltado dos resultados.

Se estamos baixo o segundo suposto o resaltado pode resolverse no código da aplicación, que deberá manexar axeitadamente os valores da columna *posicion* da táboa de palabras e o texto do documento.

Capacidades estatísticas

Para obter diferentes valores estatísticos Oracle ofrece a función `COUNT`. Con ela obtense o número de tuplas que verifican unha busca. Porén, se estamos no primeiro suposto, as palabras non están illadamente nas tuplas, polo que `COUNT` non resulta de axuda para averiguar o número de termos coincidentes coa consulta. Aínda que existe a función `CTX_QUERY.COUNT_HITS` para poder obter o número de veces que aparece unha palabra no texto incluído nunha columna dunha táboa, esta función non pode ser incluída nunha busca un pouco máis complexa que conteña algunha clausula `WHERE` para filtrar a base documental sobre a que se desexa buscar, polo que tampouco aporta unha solución.

Polo tanto, para dispoñer desta funcionalidade en todas as súas posibilidades só podemos optar pola segunda proposta, é dicir, descompoñer as palabras dos textos. Se, porén, nos conformamos cunha aproximación dos valores reais, e queremos seguir apostando pola primeira alternativa e, porque nalgún momento futuro, a función `CTX_QUERY.COUNT_HITS` se poida combinar coa función `WHERE`, o que podemos facer é estruturar os documentos en oracións ou parágrafos, de xeito que obteñamos o número de oracións ou de parágrafos en que apareza a cadea de busca.

Así, por exemplo, se estamos baixo o primeiro suposto, podemos obter o número de documentos que conteñen a palabra *termo1* que pertenzan á editorial *editorial1* do seguinte xeito:

```
SELECT COUNT(*)
FROM   documento
WHERE  CONTAINS(texto, 'termo1')>0
       AND editorial='editorial1';
```

Ademais, podemos saber o número de veces que aparece a palabra *termo1* nos documentos, neste caso sen poder facer a restrición de editorial, do seguinte xeito:

```
SET SERVEROUTPUT ON;

declare
  contador NUMBER;
begin
 contador := ctx_query.count_hits('textindex', 'proba', TRUE);
 dbms_output.put_line('Coincidencias:');
 dbms_output.put_line(contador);
end;
/
```

Pola contra, só baixo o segundo suposto podemos obter o número de veces que aparece a palabra *termo1* nos documentos que pertenzan á editorial *editorial1* do seguinte xeito:

```
SELECT COUNT(*)
FROM   documento, palabra
WHERE  palabra.id_documento=documento.id
AND    documento.editorial='editorial1'
AND    CONTAINS(palabra.palabra, 'termo1')>0;
```

Adicionalmente, resulta de utilidade combinar a función `COUNT` coa cláusula `GROUP BY` para calcular varios valores estatísticos simultaneamente e aumentar así a eficiencia do sistema.

Información adicional

Se a estrutura das táboas que almacenan a información dos documentos é axeitada, ningún xestor relacional terá problemas con esta epígrafe. Pódense obter datos de calquera información referida ao resultado dunha busca sen moitos problemas.

Por exemplo, se a táboa cos documentos que tiñamos anteriormente ten en consideración algunha información adicional, como a súa área temática, e definímola do seguinte xeito:

```
CREATE TABLE documento (
  identificador INT,
  editorial VARCHAR (128),
  area_tematica VARCHAR (128),
  texto CLOB
);

begin
  ctx_ddl.create_preference('lexico','BASIC_LEXER');
  ctx_ddl.set_attribute('lexico', 'BASE_LETTER', 'NO');
end;
/

CREATE INDEX textindex
  ON documento(texto)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('LEXER lexico');
```

obteremos, non só os documentos que coincidan cunha busca, senón a editorial ou a área temática á que pertence do seguinte xeito:

```
SELECT identificador, editorial, area_tematica
FROM   documento
WHERE  CONTAINS(texto, 'termo1')>0;
```

Igualmente, se temos relacións entre estas táboas e outras, pódese conseguir calquera valor utilizando as unións (`JOIN`) correspondentes.

Contexto

O contexto é un dos requirimentos básicos para os sistemas de RI que nos ocupan. Todos estes sistemas teñen un elemento en común: amosan a expresión de busca no seu contorno de aparición. Como xa mencionamos anteriormente, se estamos no primeiro suposto, dispoñemos da función `CTX_DOC.SNIPPET`, que permite resaltar as palabras que coincidiron cunha busca e ver as que as arrodean.

O que se bota en falta na función `CTX_DOC.SNIPPET` é poder especificar o tamaño do contexto, polo que en ocasións esta non é suficiente, e será necesario desenvolver a visualización na aplicación que accede ao xestor. Se, finalmente, temos segmentado o texto en oracións e/ou parágrafos, resultará moi sinxelo amosar a oración ou o parágrafo que contén unha cadea de busca[5] e xestionar o contexto desde a propia aplicación.

[5] Incluso as oracións e/ou parágrafos anteriores e/ou posteriores á que presenta a coincidencia.

Baixo o segundo suposto, do mesmo xeito que faciamos co resaltado dos resultados, habería que tratar o contexto desde a aplicación e utilizar de forma correcta o atributo *posicion* da táboa de palabras.

Independencia de seccións

En Oracle podemos facer buscas que só afecten a un determinado tipo de seccións de dúas maneiras: ben estruturando as seccións de busca en columnas diferentes, ben delimitando estas seccións dentro do texto mediante a función `CTX_DDL.ADD_FIELD_SECTION` á hora de crear o índice, e empregando o operador `WITHIN` para buscar unicamente dentro das seccións marcadas.

Un exemplo do primeiro caso podería ser desdobrar en dúas columnas un documento: unha para o título e outra para o seu corpo. Deste xeito a definición da táboa quedaría:

```
CREATE TABLE documento (
  identificador INT,
  titulo VARCHAR (128),
  texto CLOB
);
```

A continuación, poderiamos definir dous índices textuais, un para o título e outro para o texto:

```
CREATE INDEX tituloindex
  ON documento(titulo)
  INDEXTYPE IS CTXSYS.CONTEXT;

CREATE INDEX textindex
  ON documento(texto)
  INDEXTYPE IS CTXSYS.CONTEXT;
```

Para buscar na sección título fariamos simplemente:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS(titulo, 'termo1')>0;
```

Pola contra, un exemplo do segundo caso sería a discriminación dunha sección dentro do texto denominada, por exemplo, *seccion1*. Deste xeito, definiríase e crearíase o índice da táboa documento tal como segue[6]:

```
begin
  CTX_DDL.CREATE_SECTION_GROUP ('seccion', 'BASIC_SECTION_GROUP');
  CTX_DDL.ADD_FIELD_SECTION ('seccion','seccion1','seccion1',FALSE);
end;
/

CREATE INDEX textindex
  ON documento(texto)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('SECTION GROUP seccion');
```

[6]O último parámetro da función `CTX_DDL.ADD_FIELD_SECTION` indica se o contido desa sección debe considerarse tamén no índice textual xeral ou non.

Así, para buscar algo dentro do texto que estea englobado polas etiquetas <seccion1> e </seccion1> empregaremos a cláusula `WITHIN` tal como segue:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (texto,'termo1 WITHIN seccion1')>0;
```

Xogos de caracteres

Oracle permite traballar con calquera xogo de caracteres (ISO-8859, UTF-8 etc.), e mesmo permite definir xogos de caracteres propios en caso de seren necesarios.

Navegación polos resultados

Oracle ofrece a posibilidade de utilizar `ROWNUM` para devolver soamente un subconxunto de resultados. Xa que logo, pódense ir obtendo as diferentes páxinas de resultados, unha de cada vez. Por exemplo, se queremos acceder á segunda páxina de documentos supoñendo que se poden visualizar dez en cada páxina, actuariamos do seguinte xeito:

```
SELECT identificador
FROM   documento
WHERE  ROWNUM > 10 AND ROWNUM <= 20
ORDER BY identificador;
```

Porén, o problema é que cada vez que obtemos unha páxina de resultados estamos a refacer a consulta. Aínda que Oracle incorpora un sistema caché para almacenar resultados recentes, por se tratar de corpus de grandes dimensións e de consultas complexas, este non resulta de moita utilidade. É por iso polo que, normalmente, se necesita xestionar a paxinación no código da aplicación a través do desenvolvemento dunha caché propia, para que non se teñan que refacer as consultas cando se navega polos resultados.

Ordenación

Oracle permite ordenar calquera campo que estea estruturado. Ademais deixa utilizar varios criterios de ordenación simultaneamente. Por exemplo, poderiamos obter os documentos que conteñan o termo *termo1* ordenados por editorial e identificador do documento así:

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (texto, 'termo1')>0
ORDER BY editorial, identificador;
```

Porén, o xestor non nos ofrece posibilidade ningunha para ordenar os resultados polas cadeas que coinciden coa busca sempre e cando as cadeas aparezan no medio do texto e non estean nun campo textual illadamente (o equivalente ao primeiro suposto). Para que se entenda mellor, se na nosa táboa documento do exemplo facemos a busca

```
SELECT identificador
FROM   documento
WHERE  CONTAINS (texto, 'term\%')>0;
```

non temos xeito de indicar que nos devolva os resultados ordenados alfabeticamente segundo a cadea que coincide coa busca, é dicir, non podemos dicirlle que os documentos que conteñan a palabra *termodinámica* saian despois dos que conteñan *termal*.

Pola contra, isto si pode facerse no segundo suposto, realizando a ordenación pola columna das palabras que coinciden coa busca.

Relacións estruturais

A riqueza dos modelos relacionais fai posible que, estruturando ben a información asociada aos documentos, se poida xestionar practicamente calquera tipo de relación estrutural.

Refinamento de consultas

Oracle permite que o resultado dunha consulta se almacene nunha nova táboa. Porén, non é sinxelo xestionar esta característica desde unha aplicación que acceda ao xestor. Alén diso, a nova táboa xerada non inclúe inicialmente os índices necesarios para facer buscas nela, e construílos en tempo real pode tardar demasiado tempo se houber numerosos resultados.

Por todo isto, na maioría dos sistemas obxecto do presente estudo non se pode aproveitar esta característica, e o refinamento de consultas tradúcese nunha nova busca que contén uns criterios de selección máis restritivos.

Organizar resultados

A organización dos resultados é máis ben responsabilidade da aplicación que accede ao xestor, pero este debe ofrecelos do xeito adecuado. Os xestores relacionais en xeral, e Oracle en particular, presentan moita versatilidade na obtención dos resultados que, na maioría dos casos, satisfán as necesidades organizativas das aplicacións. Iso si, en ocasións hai que ter un coidado especial coa aparición de duplicados de tuplas cando se atravesan relacións N:M, para o que existen diversas solucións (acceder primeiro aos datos dunha parte da relación e despois aos da outra ou utilizar algún campo que permita facer a unificación de tuplas repetidas).

Gardar resultados

Esta característica resólvese no nivel da aplicación e non nos sistemas de xestión dos datos propiamente ditos, polo que non se utilizará na comparativa das diferentes tecnoloxías analizadas.

Eficiencia

Trataremos esta epígrafe a fondo e individualmente no seguinte capítulo.

Nómina

A nómina non supón un problema para os xestores relacionais e, xa que logo, tampouco para Oracle. Iso si, se necesitásemos datos relacionados co número total de palabras que hai nos documentos, estes deben ser calculados previamente e introducidos nas táboas que lles correspondan.

2.3.2 PostgreSQL

PostgreSQL [68] é un sistema de bases de datos relacional de código aberto que ten máis de 15 anos de desenvolvemento. O sistema actual provén dun primeiro proxecto denominado Postgres que comezou no ano 1986. Desde ese primeiro proxecto ata a versión estable actual do PostgreSQL, a 8.3, o desenvolvemento pasou por diferentes etapas e cambios que desenvocaron no proxecto baseado en software libre que coñecemos hoxe en día.

Para as probas do presente traballo (febreiro de 2009) utilizouse a versión 8.3.7, a última estable dispoñible durante a escritura destas liñas. Trátase da primeira que inclúe funcionalidades de busca de texto integradas, o que fai posible traballar con este xestor baixo os dous supostos expostos anteriormente.

A definición das táboas para o primeiro suposto no caso de PostgreSQL quedaría da seguinte forma:

```
CREATE TABLE documento (
  identificador INT,
  editorial VARCHAR,
  texto VARCHAR,
  CONSTRAINT doc_pk PRIMARY KEY (identificador)
);
```

e para o segundo:

```
CREATE TABLE documento (
  identificador INT,
  editorial VARCHAR,
  texto VARCHAR,
  CONSTRAINT doc_pk PRIMARY KEY (identificador)
);

CREATE TABLE palabra (
  identificador INT,
  identificador_documento INT,
  palabra VARCHAR,
  posicion INT,
  CONSTRAINT foreign_documento FOREIGN KEY (identificador_documento)
    REFERENCES documento (identificador),
  CONSTRAINT palabra_pk PRIMARY KEY (identificador)
);
```

Buscas textuais

(a) Buscas de coincidencia exacta

Para facer unha busca de coincidencia exacta en PostgreSQL de xeito que se consulte un índice que inclúa as palabras do texto (e non o texto en si mesmo), e traballando baixo o primeiro suposto, en primeiro lugar debemos definir un perfil básico para construír os índices textuais[7]:

[7] Previamente hai que crear no directorio de instalación de PostgreSQL, share/tsearchdata/, o arquivo galician.stop baleiro, para que non elimine ningunha palabra do índice.

```
CREATE TEXT SEARCH CONFIGURATION public.galician (
  PARSER = pg_catalog.default
);

CREATE TEXT SEARCH DICTIONARY public.galician_dict (
  TEMPLATE = pg_catalog.simple,
  STOPWORDS = galician
);

ALTER TEXT SEARCH CONFIGURATION galician
  ALTER MAPPING FOR asciiword, asciihword, hword_asciipart,
                    word, hword, hword_part
  WITH galician_dict;
```

A continuación, creamos un índice de tipo `gin` na columna *texto* como segue:

```
CREATE INDEX textindex
  ON documento
  USING gin(to_tsvector('galician',texto));
```

E, finalmente, poderemos obter os documentos que coinciden cunha consulta determinada do seguinte xeito:

```
SELECT identificador
FROM   documento
WHERE  to_tsvector('galician',texto) @@
       to_tsquery('galician','termo');
```

O problema é que PostgreSQL non permite facer buscas de expresións que conteñan máis dun termo, é dicir, non funcionaría unha consulta deste estilo:

```
SELECT identificador FROM documento
WHERE  to_tsvector('galician',texto) @@
       to_tsquery('galician','termo1 termo2');
```

Se, pola contra, estamos no segundo suposto, tamén podemos empregar as funcións de busca de texto, que esta vez actuarán sobre a táboa de palabras:

```
CREATE INDEX textindexpalabra
  ON palabra
  USING gin(to_tsvector('galician',palabra));

SELECT documento.identificador
FROM   documento, palabra palabra1, palabra palabra2
WHERE  palabra1.identificador_documento = documento.identificador
       AND palabra2.identificador_documento = documento.identificador
       AND palabra2.posicion = palabra1.posicion+1
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician','termo1')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician','termo2');
```

Deste modo, xa podemos facer buscas dunha expresión multipalabra utilizando os índices correspodentes relacionados coa posición das palabras. Para que estas consultas non sexan demasiado lentas con expresións multipalabra e grandes volumes de información, necesitamos crear o índice relativo a *posicion* e o índice funcional relativo a *posicion+1*.

```
CREATE INDEX palabra_posicion
  ON palabra(posicion);

CREATE INDEX palabra_position_func
  ON palabra ((position+1));
```

A utilización de números consecutivos non repetidos para a posición das palabras tamén podería optimizar este último tipo de buscas.

(b) Buscas sensibles e non sensibles aos tiles

Para poder realizar consultas non sensibles aos tiles é necesario facerlle un procesamento previo ao texto antes de pasarllo á función `to_tsvector` que será utilizada na construción do índice de tipo `gin`. Para isto será necesario crear un procedemento PL/SQL[8] antes da creación do índice.

En primeiro lugar definimos a función:

```
CREATE OR REPLACE FUNCTION nonsensible (text) RETURNS text AS $$
declare
  var1 varchar;
begin
  var1=replace($1, 'á', 'a');
  var1=replace(var1, 'é', 'e');
  var1=replace(var1, 'í', 'i');
  var1=replace(var1, 'ó', 'o');
  var1=replace(var1, 'ú', 'u');
  var1=replace(var1, 'Á', 'A');
  var1=replace(var1, 'É', 'E');
  var1=replace(var1, 'Í', 'I');
  var1=replace(var1, 'Ó', 'O');
  var1=replace(var1, 'Ú', 'U');
  return var1;
end;
/
$$LANGUAGE plpgsql immutable;
```

A continuación, a configuración para as buscas de texto:

```
CREATE TEXT SEARCH CONFIGURATION public.galiciannonsensible (
  PARSER = pg_catalog.default
);

CREATE TEXT SEARCH DICTIONARY public.galician_dict (
  TEMPLATE = pg_catalog.simple,
  STOPWORDS = galician
);
```

[8] Linguaxe de programación embebida utilizada por Oracle e PostgreSQL.

```
ALTER TEXT SEARCH CONFIGURATION galiciannonsensible
  ALTER MAPPING FOR asciiword, asciihword, hword_asciipart,
                    word, hword, hword_part
  WITH galician_dict;
```

Logo, creamos o índice utilizando a función definida

```
CREATE INDEX textindexnonsensible
  ON documento
  USING gin(to_tsvector('galiciannonsensible',nonsensible(texto)));
```

e, finalmente, facemos a consulta:

```
SELECT identificador
FROM   documento
WHERE  to_tsvector('galiciannonsensible',texto) @@
       to_tsquery('galiciannonsensible','termo');
```

(c) Buscas sensibles ou non ás maiúsculas

O caso da sensibilidade ás maiúsculas nesta versión de PostgreSQL é un pouco máis complexo de resolver, xa que o modelo pg_catalog.simple utilizado no exemplo anterior cambia a minúsculas as entradas e non permite modificar este comportamento mediante parámetros de configuración. Polo tanto, habería que programar un novo modelo na linguaxe C para pasarllo á configuración de buscas de texto e compilalo e instalalo no sistema. A grandes trazos o proceso sería:

```
1) Crear un novo modelo (arquivo .c)
   baséandose, por exemplo, no arquivo contrib/dict_int.c.
2) Compilar e instalar o modelo
3) Crear unha configuración textual con ese modelo.
4) Usar esta configuración para crear o índice e facer as
   consultas.
```

(d) Unificación de caracteres

PostgreSQL posibilita a unificación de calquera caracter a través do procesamento do texto antes da creación do índice correspondente, do mesmo xeito que para conseguir buscas sensibles e non sensibles aos tiles.

(e) Buscas booleanas

Pódense utilizar os operadores booleanos & (`AND`), | (`OR`) e ! (`NOT`):

```
SELECT identificador
FROM   documento
WHERE  to_tsvector('galician',texto) @@
       to_tsquery('galician','termo1 & termo2');
```

(f) Buscas de proximidade

A versión de PostgreSQL non permite facer consultas que impliquen proximidade entre termos, polo que habería que xestionalas a través da táboa de palabras da segunda proposta. Por exemplo, para obter os documentos que conteñan *termo1* e *termo2* a unha distancia menor de tres palabras, poderiamos facer:

```
SELECT documento.identificador
FROM   documento, palabra palabra1, palabra palabra2
WHERE  palabra1.identificador_documento = documento.identificador
       AND palabra2.identificador_documento = documento.identificador
       AND to_tsvector ('galician', palabra1.palabra) @@
           to_tsquery('galician','termo1')
       AND to_tsvector ('galician', palabra2.palabra) @@
           to_tsquery('galician','termo2')
       AND palabra1.posicion < palabra2.posicion+3;
```

(g) Exclusión de texto marcado das buscas

Novamente hai que facer uso da facilidade que dá PostgreSQL para procesar o texto antes de incluílo no índice para materializar estas buscas. Deste xeito pode transformarse o texto para que non se indexen as palabras que se atopan incluídas dentro dalgún tipo de marca.

(h) *Stop words*

Pódense facer listaxes de *stop words* personalizadas. Só hai que crear un arquivo coas palabras que non se desexan indexar (coa extensión .stop) e colocalo no directorio share/tsearchdata/ de instalación. Despois só é necesario indicar na configuración do dicionario o nome do arquivo. Por exemplo:

```
CREATE TEXT SEARCH DICTIONARY public.nome_diccionario (
  TEMPLATE = pg_catalog.simple,
  STOPWORDS = nome_de_arquivo
);
```

(i) Múltiples modos de busca simultáneos

Para poder ofrecerlle ao usuario varios tipos de busca, pódese crear un duplicado da columna que se vai indexar para, por exemplo, facer un índice sensible aos tiles nunha delas e non sensible aos tiles na outra. Finalmente, consultaríase unha columna ou outra en función da busca desexada polo usuario.

Porén, este método implica unha duplicación da columna de texto que se debería evitar cando traballamos cun corpus moi grande. Para non ter que duplicala con PostgreSQL hai que especificar dous índices textuais sobre a mesma columna. Así, se baixo o primeiro suposto queremos que se poidan facer índices sensibles e non sensibles aos tiles, crearemos dúas definicións de configuración de buscas textuais

```
CREATE TEXT SEARCH CONFIGURATION public.galiciannonsensible (
  PARSER = pg_catalog.default
);
```

```
CREATE TEXT SEARCH DICTIONARY public.galician_dict (
  TEMPLATE = pg_catalog.simple,
  STOPWORDS = galician
);

ALTER TEXT SEARCH CONFIGURATION galiciannonsensible
  ALTER MAPPING FOR asciiword, asciihword, hword_asciipart,
                    word, hword, hword_part
  WITH galician_dict;

CREATE TEXT SEARCH CONFIGURATION public.galiciansensible (
  PARSER = pg_catalog.default
);

ALTER TEXT SEARCH CONFIGURATION galiciansensible
  ALTER MAPPING FOR asciiword, asciihword, hword_asciipart,
                    word, hword, hword_part
  WITH galician_dict;
```

onde a primeira alternativa define os parámetros para as consultas non sensibles aos tiles e a segunda os das sensibles aos tiles. A continuación, habería que facer os índices de texto para os dous casos, tendo en conta que para o primeiro deles se procesaría o texto para eliminarlle os tiles:

```
CREATE INDEX textindexnonsensible
  ON documento
  USING gin(to_tsvector('galiciannonsensible',nonsensible(texto)));

CREATE INDEX textindexsensible
  ON documento
  USING gin(to_tsvector('galiciansensible',texto));
```

Podemos comprobar como se utiliza a función *nonsensible* na creación do índice que definimos anteriormente. Xa só quedaría, en tempo de execución, facer unha consulta sensible ou non sensible dependendo do nome da configuración que se utilice nela:

```
SELECT identificador
FROM   documento
WHERE  to_tsvector('galiciannonsensible',texto) @@
       to_tsquery('galiciannonsensible','termo');

SELECT identificador
FROM   documento
WHERE  to_tsvector('galiciansensible',nonsensible(texto)) @@
       to_tsquery('galiciansensible','termo');
```

No caso da segunda proposta actuaríase do mesmo xeito sobre a táboa de palabras.

(j) Configuración dos caracteres que poden formar parte das palabras

PostgreSQL incorpora un analizador textual moi versátil que permite, por exemplo, localizar tanto partes dunha palabra composta como a forma completa. Porén, hai casos en que poderiamos preferir que, por exemplo, non se recoñezan as partes dunha palabra que teña un

guión, como pode ser o caso da segunda forma do artigo no galego. Neste caso haberá que cambiar a configuración das buscas de texto que temos do seguinte xeito:

```
ALTER TEXT SEARCH CONFIGURATION galician
  ALTER MAPPING FOR asciiword, asciihword,
                    word, hword
  WITH galician_dict;
```

Isto é, non hai que incluír no `MAPPING`: `hword_asciipart` e `hword_part`, que son os tipos de definicións que utiliza o analizador para recoñecer os compoñentes dunha palabra composta.

Por outra banda, se o que queremos é que certos caracteres non afecten á indexación dos termos como, por exemplo, algunha marca da edición que indique un fragmento reconstruído dunha palabra[9], podemos usar, novamente, o preprocesamento antes da utilización da función `to_tsvector`. Así, eliminariamos eses caracteres antes de que os reciba a función `to_tsvector` para a creación do índice.

(k) Utilización de comodíns

Un dos problemas principais da versión de PostgreSQL analizada é que non permite usar comodíns cando se fai unha consulta empregando as características de análise textual (`to_tsquery`). Xa que logo non se poderán facer buscas de prefixos e/ou sufixos de palabras, o que limita considerablemente as posibilidades do xestor. Parece que na versión 8.4 si virá incluída esa posibilidade.

(l) Resaltado

A función `ts_headline` permite facer o resaltado dos termos que verifiquen unha consulta nos documentos e especificar, ademais, as marcas que delimitarán o texto coincidente e o tamaño do contexto que se quere visualizar. Por tanto, baixo o primeiro suposto utilizarase esta función do seguinte xeito:

```
SELECT ts_headline ('galician','texto do documento',
       to_tsquery('termo de busca'));
```

No segundo suposto non é necesario ter esta funcionalidade xa que, coa utilización da información da posición das palabras e o texto do documento, pódese facer o resaltado desde a aplicación sen moitas complicacións.

Capacidades estatísticas

Tampouco PostgreSQL ten unha función para contar as palabras que coinciden cunha busca sen ter que optar pola segunda proposta. As estatísticas deben facerse a través da función `COUNT` de SQL, polo que se poderá contar calquera elemento que estea almacenado individualmente nunha columna. Daquela, haberá que elixir a segunda opción, é dicir, ter estruturadas as palabras de xeito illado para poder explotar ao máximo estas capacidades, ou utilizar algunha subdivisión de texto que nos poida servir de aproximación (oracións ou parágrafos).

[9] Nese caso o normal é que desexemos que a palabra se indexe sen eses caracteres e, cando se visualicen os resultados, se poidan amosar tal cal aparecen no texto.

Información adicional

As bases de datos relacionais, en xeral, non presentan limitacións neste punto, xa que se pode obter calquera información asociada aos casos coincidentes se está ben estruturada.

Contexto

Como xa mencionamos anteriormente, a función `ts_headline` debería cubrir todas as necesidades de resaltado baixo o primeiro suposto e, no segundo, a aplicación podería utilizar a información da posición das palabras para facer o resaltado.

Independencia de seccións

En PostgreSQL unicamente podemos facer buscas que só afecten a unha determinada sección dun documento se esta se almacena nunha columna independente. Por exemplo, se os textos conteñen noticias de xornal e queremos que se permita buscar só dentro dos titulares, no corpo ou en ambas as zonas, estas seccións deberían estar almacenadas en columnas independentes. É dicir, non se permite especificar seccións dentro do propio texto desestruturado.

Xogos de caracteres

PostgreSQL permite traballar cos xogos de caracteres máis coñecidos (ISO-8859, UTF-8 etc.).

Navegación polos resultados

As cláusulas SQL `OFFSET` e `LIMIT` en PostgreSQL permiten obter un conxunto limitado de resultados. Porén, dado que estamos a falar de corpus de grandes dimensións e potencialmente podería haber moitos resultados, a caché que incorpora o xestor non adoita ser suficiente para que non se teña que refacer a consulta ao acceder ás diferentes páxinas de resultados. Por isto acostuma ser moi útil a implementación dunha caché na aplicación de consultas que permita facer esta navegación moito máis eficiente.

Ordenación

PostgreSQL tamén presenta todas as vantaxes e os inconvenientes das posibilidades de ordenación dos xestores relacionais. Por unha banda, poderase ordenar por calquera campo estruturado e, pola outra, temos a limitación de non poder facelo polas coincidencias da busca no caso do primeiro suposto, como xa comentamos para Oracle (sección 2.3.1).

Relacións estruturais

Se a información está ben estruturada no modelo subxacente non haberá problemas con esta epígrafe.

Refinamento de consultas

En PostgreSQL pódese almacenar o resultado dunha consulta nunha táboa. Non obstante, é bastante complexo xestionar o refinamento de consultas no caso de acceso concorrente. Débese ter en conta que estamos ante sistemas que traballan con grandes dimensións de datos. O resultado dunha busca pode ser demasiado grande para facer outra sobre el e indexalo podería ser moi custoso.

É por isto polo que o normal, no caso de querer facer unha subconsulta, sexa realizar unha nova busca desde cero con criterios de selección máis restritivos que a orixinal para obter os resultados desexados.

Organizar resultados

Os xestores relacionais ofrecen moita flexibilidade para que a aplicación organice correctamente os resultados. O único problema que pode aparecer en ocasións é o da eliminación de tuplas repetidas cando se atravesan relacións N:M (véxase o dito para Oracle en relación ás solucións posibles).

Gardar resultados

Resólvese satisfactoriamente ao nivel de aplicación.

Eficiencia

Véxase o seguinte capítulo para datos relativos a esta epígrafe.

Nómina

PostgreSQL non tería problemas con esta epígrafe.

2.3.3 MySQL

MySQL [83] é un xestor de base de datos relacional de código aberto que naceu en 1995 na empresa sueca MySQL AB, absorbida por Sun Microsystems en 2008 e esta, á vez, por Oracle Corporation en 2009. O proxecto foi evolucionando coa intención de crear un xestor de base de datos que cumprira o estándar SQL, pero sen sacrificar velocidade, fiabilidade ou usabilidade.

Desde a súa orixe deica os nosos días foi cambiando ata a versión estable actual, a 5.1.32, que é a última dispoñible no momento en que escribimos isto (febreiro 2009). Esta foi a versión que se utilizou para facer a nosa avaliación.

Xa é coñecido que as capacidades de indexación textual son relativamente recentes nos xestores relacionais e, en moitos casos, estas están asociadas a funcións de busca estilo web. Estas últimas manexan conceptos como a relevancia, as listaxes documentais ordenadas (*ranking*) e outros elementos que carecen de interese para o obxecto do presente traballo, polo que tamén en MySQL ignoraremos ese tipo de funcionalidades.

A definición das táboas baixo os dous supostos presentados quedan para MySQL do seguinte xeito:

```
CREATE TABLE documento (
  identificador INT,
  editorial VARCHAR,
  texto TEXT,
  CONSTRAINT doc_pk PRIMARY KEY (identificador)
);
```

para o primeiro caso, e:

```
CREATE TABLE documento (
  identificador INT,
  editorial VARCHAR,
  texto TEXT,
  CONSTRAINT doc_pk PRIMARY KEY (identificador)
);

CREATE TABLE palabra (
  identificador INT,
  identificador_documento INT,
  palabra VARCHAR (150),
  posicion INT,
  CONSTRAINT foreign_documento FOREIGN KEY (identificador_documento)
    REFERENCES documento (identificador),
  CONSTRAINT palabra_pk PRIMARY KEY (identificador)
);
```

para a segunda proposta.

Buscas textuais

(a) Buscas de coincidencia exacta

MySQL presenta varios modos de busca de texto. O modo de busca en linguaxe natural, que nun principio pode parecer máis axeitado para os nosos propósitos, permite facer buscas de expresións multipalabra, pero presenta o inconveniente de que, por defecto, non inclúe no índice as palabras que aparecen en máis do 50% das tuplas. Para modificar este comportamento é necesario cambiar no arquivo fonte myisam/ftdefs.h a definición de `GWS_IN_USE` a `GWS_FREQ` e recompilar e reinstalar o xestor.

Porén, para o obxecto dos sistemas que nos ocupan, resulta moito máis interesante o segundo modo de busca, as consultas de texto boolenas. Este tipo de consultas non se ve afectado polo límite do 50% anterior e permitirá habilitar a mellor solución aos requirimentos aquí propostos.

Polo tanto, para facermos unha busca de coincidencia exacta, non hai máis que introducir os datos na táboa e, a continuación, definir un índice sobre a columna textual que queremos utilizar:

```
ALTER TABLE documento ADD FULLTEXT documento_text(texto);
```

Facemos a consulta do seguinte xeito:

```
SELECT identificador
FROM   documento
WHERE  MATCH (texto)
       AGAINST ('"expresión de busca"' IN BOOLEAN MODE);
```

As comiñas dobres, neste caso, indican que se desexa buscar a expresión multipalabra situada dentro delas. Se só fora unha palabra non serían necesarias.

Baixo o segundo suposto, se necesitamos obter a posición en que se atopa unha expresión de busca, a consulta quedaría:

```
SELECT documento.identificador, palabra1.posicion
FROM   documento, palabra palabra1, palabra palabra2
WHERE  palabra1.identificador_documento = documento.identificador
       AND palabra2.identificador_documento = documento.identificador
       AND palabra2.posicion=palabra1.posicion+1
       AND MATCH (palabra1.palabra) AGAINST('termo1' IN BOOLEAN MODE)
       AND MATCH (palabra2.palabra) AGAINST('termo2' IN BOOLEAN MODE);
```

Iso si, a diferenza dos casos de Oracle e PostgreSQL, en MySQL non se poden construír índices funcionais que permitan acelerar as buscas que utilizan o campo *posicion*. Unha posible solución para esta limitación de optimización podería ser incorporar unha columna extra na táboa das palabras que conteña o valor de *posicion* incrementado nun e creando un índice sobre ela. Deste xeito, se supoñemos que esa columna se chama *posicionmais1*, a consulta quedaría do seguinte xeito:

```
SELECT documento.identificador, palabra1.posicion
FROM   documento, palabra palabra1, palabra palabra2
WHERE  palabra1.identificador_documento = documento.identificador
       AND palabra2.identificador_documento = documento.identificador
       AND palabra2.posicion=palabra1.posicionmais1
       AND MATCH (palabra1.palabra) AGAINST('termo1' IN BOOLEAN MODE)
       AND MATCH (palabra2.palabra) AGAINST('termo2' IN BOOLEAN MODE);
```

(b) Buscas sensibles e non sensibles aos tiles

MySQL permite facer consultas sensibles ou non sensibles aos tiles de diversas maneiras, pero a máis recomendable é o emprego da cláusula `COLLATE` á hora da creación da táboa. No caso de termos codificación ISO-8859, por defecto as consultas non son sensibles aos tiles e pódense facer sensibles especificando `latin1_bin` na cláusula. Para UTF-8 o valor `utf8_bin` ten o mesmo efecto.

(c) Buscas sensibles ou non ás maiúsculas

Este caso resólvese do mesmo xeito que o anterior. Por defecto as consultas non son sensibles ás maiúsculas, pero poden selo se especificamos na cláusula `COLLATE`, `latin1_bin` ou `utf8_bin`.

(d) Unificación de caracteres

A versión 5.1 de MySQL posibilita a unificación de calquera caracter, ben a través do sistema de conectores, ben ao modificar o conxunto de caracteres empregado para a base de datos. No primeiro caso, o xestor permite crear un analizador de texto que pode cambiar as palabras antes de pasarllas ao indexador para que sexan incluídas no índice de texto. Pola contra, no segundo,

deben definirse as correspondencias do `COLLATION` utilizado polo conxunto de caracteres para que aquelas reflictan a unificación.

(e) Buscas booleanas

MySQL inclúe os operadores +(`AND`) -(`NOT`) e a ausencia de operador (`OR`) para facer as consultas booleanas:

```
SELECT identificador
FROM   documento
WHERE  MATCH(texto)
       AGAINST('+termo1 -termo2' IN BOOLEAN MODE)
```

(f) Buscas de proximidade

Este xestor non permite facer consultas que impliquen proximidade entre palabras, polo que o único xeito de facelas é baixo o segundo suposto ao utilizarmos o parámetro de posición.

(g) Exclusión de texto marcado das buscas

MySQL non inclúe por defecto esta funcionalidade, pero presenta mecanismos para desenvolvela cun certo esforzo. Si inclúe unha interface de programación de aplicacións (*Application Programming Inferface*, API) para crear conectores que permitirían construír un analizador de texto na linguaxe C. Mediante estes, o xestor pode evitar, por exemplo, introducir no índice as palabras que estean entre determinadas marcas. Iso si, habería que investir un tempo considerable no desenvolvemento do conector que, ademais, é probable que teña que ser revisado en versións posteriores[10].

(h) *Stop words*

Nesta versión de MySQL pódese utilizar unha listaxe de *stop words* diferente da que trae por defecto para o inglés. Porén, esta é única para todo o xestor, o que pode dar algúns problemas no caso de que teñamos diferentes proxectos no mesmo sistema. Para que se indexen todas as palabras é necesario inicializar cunha cadea baleira a variable do sistema `ft_stopword_file`.

(i) Múltiples modos de busca simultáneos

Se desexamos que os usuarios poidan decidir se utilizan unha consulta sensible ou non aos tiles ou ás maiúsculas, só queda a solución de duplicar a columna cos datos e construír índices independentes para columnas diferentes. O xestor non presenta un mecanismo para definir dous índices sobre unha mesma columna.

(j) Configuración dos caracteres que poden formar parte das palabras

MySQL ofrece dúas posibilidades para especificar os caracteres que poden formar parte das palabras: modificar as macros `true_word_char()` e/ou `misc_word_char()` do arquivo storage/myisam/ftdefs.h para incluír/eliminar aí os símbolos que poden formar ou non parte das palabras, o que supón a recompilación e reinstalación do xestor, e modificar un arquivo de

[10] A 5.1 é a primeira versión en que aparece esta posibilidade de crear conectores e na propia documentación faise esa advertencia.

conxunto de caracteres cos cambios que se desexen para usalo na construción do índice de texto, o cal non obriga a reinstalar o sistema.

En calquera dos casos e, como se pode comprobar, as dúas alternativas son tarefas de moito máis baixo nivel que as equivalentes dos outros xestores e xeralmente requiren máis tempo para levalas a cabo.

(k) Utilización de comodíns

MySQL inclúe soamente o comodín * nas consultas booleanas para poder facer buscas de prefixos. Polo tanto, non permite empregar este elemento ao comezo das palabras e, xa que logo, non se poderan facer consultas sobre sufixos.

Ademais, por defecto MySQL só indexa as palabras que teñen máis de tres caracteres. Dado que nos sistemas que nos ocupan ás veces é necesario consultar palabras de menos lonxitude, é necesario definir no arquivo de configuración principal my.conf a variable `ft_min_word_len` e poñerlle o valor 1.

```
[mysqld]
ft_min_word_len=1
```

(l) Resaltado

MySQL non ofrece ningunha función de resaltado de texto, o que nos obrigaría a optar pola segunda opción, é dicir, a ter estruturadas as palabras nunha táboa para, a través dunha posible columna coa posición, facer o resaltado dentro da propia aplicación.

Capacidades estatísticas

Tampouco ten unha función para contar as coincidencias sen ter que optar pola segunda proposta. As estatísticas deben facerse a través da función `COUNT` de SQL, polo que se poderá contar calquera elemento que estea almacenado individualmente nunha columna independente. Polo tanto, haberá que elixir a alternativa de ter estruturadas as palabras illadamente para explotar ao máximo estas capacidades ou utilizar algunha subdivisión de texto que nos poida servir de aproximación (oracións ou parágrafos).

Información adicional

Permite obter calquera información asociada aos casos coincidentes se a base de datos está ben estruturada.

Contexto

Do mesmo xeito que para o resaltado, MySQL non conta con funcións para amosar o contexto de ocorrencia dunha palabra, polo que haberá que facer uso da información de posición das palabras para levar a cabo o resaltado desde a propia aplicación.

Independencia de seccións

Ao igual que en PostgreSQL só podemos facer buscas que só afecten a unha sección do documento se esta se almacena nunha columna independente. Non hai maneira de determinar seccións independentes dentro do texto.

Xogos de caracteres

MySQL permite traballar cos xogos de caracteres máis coñecidos (ISO-8859, UTF-8 etc.).

Navegación polos resultados

A cláusula SQL `LIMIT` en MySQL posibilita obter un conxunto limitado de resultados. Porén, dado que estamos a falar de corpus de grandes dimensións, potencialmente podería haber moitos resultados. Neste caso a caché que incorpora o xestor non é suficiente para evitar refacer as consultas que acceden ás diferentes páxinas de resultados. Por isto adoita ser moi útil a implementación dunha caché na aplicación que permita facer esta navegación moito máis eficiente.

Ordenación

MySQL tamén presenta as vantaxes dos xestores relacionais neste aspecto e permite ordenar por calquera campo estruturado. Porén, do mesmo xeito que en Oracle e PostgreSQL, tampouco deixa ordenar polo termo de busca se os termos non están nunha táboa independente.

Relacións estruturais

Se a información está ben estruturada no modelo subxacente, non haberá problemas con esta epígrafe.

Refinamento de consultas

Tamén en MySQL se pode almacenar o resultado dunha consulta nunha táboa pero, dada a dificultade que presenta a súa xestión, xa que estamos falando de bases de datos de grandes dimensións, acostuma ser máis práctico refacer a busca cuns criterios máis restritivos.

Organizar resultados

Os xestores relacionais ofrecen moita flexibilidade co fin de que a aplicación organice correctamente os resultados. O único problema facilmente solucionable que pode aparecer, como xa comentamos nas tecnoloxías anteriores, é a eliminación de tuplas repetidas cando se atravesan relacións N:M.

Gardar resultados

Resólvese satisfactoriamente ao nivel de aplicación.

Eficiencia

Véxase o seguinte capítulo para os datos relativos a esta epígrafe.

Nómina

Tampouco MySQL terá problemas nesta epígrafe, dado que é un xestor relacional.

2.4 Xestores xml

Para o final deixamos os xestores xml que, malia que pode parecer que son a opción máis axeitada para facer este tipo de sistemas en que os documentos xa están en formato xml, na práctica é máis ben ao contrario, xa que a súa tecnoloxía aínda é moi recente e algunhas das funcionalidades que esiximos destes sistemas de ri aínda non foron afrontadas.

Quizais o exemplo máis representativo desta tecnoloxía sexa Tamino [80], para a que xa fixemos un estudo previo da súa adecuación aos sistemas aquí representados [11] que confirmou que algúns aspectos necesarios non son tratados axeitadamente.

Ademais, hai outras alternativas no software libre, como eXist [57] ou Xindice [5] que tampouco cumpren, de momento, as especificacións propostas. O problema de base dunhas e doutras radica en que aínda son xestores moi novos que intentan resolver problemas de sistemas de ri moito máis básicos e, polo tanto, a eficiencia relacionada cos aspectos que nos ocupan non está á altura doutros sistemas xa presentados aquí. Debe terse en conta, por exemplo, que a primeira versión do estándar XQuery [101], que parece ser unha das alternativas máis axeitadas para a consulta de bases de datos xml nativas publicouse no ano 2007, de modo que dar un soporte eficiente a ese tipo de consultas levará aínda un certo tempo.

2.5 Outras alternativas

No presente traballo seleccionamos as tecnoloxías que, ao noso entender, son as máis relevantes para a construción de sistemas que cumpran os requirimentos propostos. Non obstante, existen moitas e cada día aparecen novas. En calquera caso, no momento en que escribimos estas liñas non atopamos outras tecnoloxías que teñan a relevancia e madurez suficiente para afrontar un reto deste tipo con garantías.

2.6 Comparativa

Ata aquí a análise individual de cada unha das tecnoloxías seleccionadas para este estudo. Nesta sección estudamos conxuntamente as súas posibilidades co obxectivo de termos unha visión global e determinar a súa adecuación aos sistemas de ri que nos ocupan.

Características	Oracle 11g	PostgreSQL 8.3	MySQL 5.1
1a	Si	1ª(1 termo), 2ª(Non)	Si
1b	Si	Si	Si
1c	Si	+	Si
1d	+, <= 2 caract.	Si	+
1e	Si	Si	Si
1f	Si	1ª(Non), 2ª(Si)	1ª(Non), 2ª(Si)
1g	Si	+	+
1h	Si	Si	Si(global)
1i	Si	Si	duplicación
1j	Si/Si	Si/+	Si(global)/+
1k	%,_	%(non sufixos)	*(non sufixos)
1l	1ª(Si: -tamaño), 2ª(Si)	Si	1ª(Non), 2ª(Si)
2	1ª(Si: -palabras), 2ª(Si)	1ª(Si: -palabras), 2ª(Si)	1ª(Si: -palabras), 2ª(Si)
3	Si	Si	Si
4	1ª(Si: -tamaño), 2ª(Si)	Si	1ª(Non), 2ª(Si)
5	Si	Só estruturado	Só estruturado
6	Si	Si	Si
7	Caché aplicación	Caché aplicación	Caché aplicación
8	1ª(Si: -palabras), 2ª(Si)	1ª(Si: -palabras), 2ª(Si)	1ª(Si: -palabras), 2ª(Si)
9	Si	Si	Si
10	Si(refacer)	Si(refacer)	Si(refacer)
11	Si	Si	Si
14	Si	Si	Si

Figura III.2.1: Comparativa do cumprimento dos requirimentos propostos

En primeiro lugar, no cadro da figura III.2.1[11] poñemos en común as posibilidades dos xestores relacionais. Xa nunha primeira ollada podemos comprobar que é na columna de Oracle onde aparecen máis sis sen condicións e ningún non. Pasamos, a continuación, a facer a análise comparativa por epígrafes.

Buscas textuais

Como pode observarse, as únicas limitacións de Oracle canto ás buscas textuais son que só se permite a unificación de ata dous caracteres, o que habitualmente é suficiente, e que impide definir o tamaño do contexto cando o texto non está estruturado en palabras.

En cambio, en PostgreSQL non se poden facer buscas de expresións multipalabra, algo moi básico para os sistemas que consideramos. Tampouco buscas de proximidade cando o texto non está dividido en palabras, nin consultas de prefixos/sufixos de palabras e non é sinxelo excluír caracteres das palabras do índice. Ademais, necesítase programar un novo analizador en linguaxe C para poder realizar buscas sensibles ás maiúsculas, e excluír texto marcado das buscas require moito traballo extra.

Finalmente, MySQL esixe moito traballo extra para facer a unificación de caracteres, excluír texto marcado das buscas e ignorar caracteres das palabras na construción dos índices. A listaxe de *stop words* e a configuración dos caracteres que poden formar parte das palabras configúranse

[11] A ocorrencia do símbolo + indica que se require un traballo máis laborioso do esperable para poder levar a cabo esa característica. Isto inclúe calquera traballo que requira programar máis alá do que supoña establecer algún parámetro de configuración. Do mesmo xeito, cando aparece 1ª ou 2ª referímonos ás primeira e segunda alternativas, respectivamente, presentadas con anterioridade. É dicir, non estruturar as palabras de xeito illado ou si facelo. A numeración das características correspóndese coa utilizada no capítulo anterior.

globalmente para todo o xestor, o que non permite ter configuracións diferentes para varios sistemas de RI. Tampouco deixa construír varios índices de texto para unha mesma columna de datos sen duplicar a súa información e, baixo o primeiro suposto, impide facer buscas de proximidade e resaltar o texto que coincide cunha busca.

Capacidades estatísticas

Ningunha das tres tecnoloxías ordena polas palabras coincidentes da busca se non están estruturadas separadamente e as tres permiten facelo cando as palabras si están separadas.

Contexto

Só PostgreSQL deixa definir o tamaño do contexto cando o texto non está estruturado, pero as tres permiten abordar esta epígrafe no segundo suposto.

Independencia de seccións

Só Oracle dá a posibilidade de buscar en seccións marcadas dentro de texto desestruturado de maneira sinxela.

Navegación polos resultados

Debido ao tamaño potencial do resultado das consultas, resulta recomendable nas tres tecnoloxías ter unha caché na aplicación de buscas que permita facermos unha navegación páxina a páxina dos resultados sen refacer a consulta.

Ordenación

Ningunha das tres tecnoloxías consinte ordenar polas cadeas de coincidencia da busca no primeiro suposto, polo que é necesario para este caso que as palabras estean illadamente nunha táboa.

Refinamento de consultas

Nos tres casos é recomendable refacer a consulta con criterios de filtrado máis restritivos en vez de actuar contra un subconxunto de resultados. Isto débese a que o primeiro resultado pode ser moi grande e a súa xestión pode ser aínda máis custosa que refacer a busca.

A modo de conclusión diremos que, como se aprecia na figura III.2.1, Oracle é a tecnoloxía relacional que menos problemas presenta de entrada e, nos casos problemáticos, a que máis facilidades ofrece para resolvelos nun tempo razoable. Xa que logo, e aínda que as licenzas para produción de Oracle teñen un custo considerable, hoxe por hoxe é a tecnoloxía que cómpre elixir se se queren evitar problemas no desenvolvemento deste tipo de contornos. É máis, Oracle está moi preto de conseguir que non sexa necesario utilizar a táboa de palabras para estes sistemas. O aspecto clave que debe resolver é o da combinación da función `count_hits` con cláusulas `WHERE` que permitiría resolver as consultas que, doutro xeito, necesitarían desta táboa. En calquera caso, todo parece indicar que as alternativas de software libre igualarán as prestacións das propietarias dentro de non moito tempo.

Tecnoloxía	**Motivo principal**
WordSmith tools	Aplicación monolítica
Sketch Engine	Carencia de filtros de alto nivel
DtSearch Swish-e	Estrutura documental pobre
Lucene	Centrado na obtención de documentos
Sphinx	Experimental
Tamino eXist	Rendemento pobre
Xindice	Experimental

Figura III.2.2: Tecnoloxías desbotadas

Na figura III.2.2, amosamos, por contra, os motivos principais polos que o resto de tecnoloxías non son axeitados para afrontar os problemas propostos. A grandes trazos, ningunha destas tecnoloxías ofrece actualmente un nivel aceptable canto ás posibilidades de busca correspondentes aos nosos requirimentos, aínda que tamén neste caso hai algunhas delas que están avanzando moi rapidamente e é preciso non perdelas de vista ante posibles evolucións que as fagan máis competitivas.

Capítulo 3

Rendemento

Neste capítulo avaliamos o rendemento das tres tecnoloxías relacionais seleccionadas (Oracle, PostgreSQL e MySQL) e utilizamos, como banco de probas, a estrutura básica da base de datos desenvolvida para o proxecto CORGA (figura III.3.1). Aínda que os principios de deseño serán analizados no seguinte capítulo, explicamos a continuación os conceptos básicos do esquema da figura anterior.

Como pode apreciarse no esquema, o núcleo dun documento XML represéntase mediante dúas entidades: Agrupacion e Documento. Por exemplo, para un xornal, a entidade Agrupacion representa o xornal completo, mentres que a entidade Documento só as súas noticias. Cada documento ten asignadas as súas oracións e, finalmente, cada oración asóciase coas palabras que a forman. Esta relación, con cardinalidade N:M, almacena as posicións en que ocorren as palabras dentro das oracións. Alén diso, os documentos teñen un medio de publicación, unha editorial, uns autores e están catalogados en ata, como máximo, tres áreas temáticas. Tamén se almacena asociado a cada documento o número de palabras (*num_pal*) que contén para atender aos requirimentos das consultas da nómina.

Por tanto, para facer a avaliación de rendemento imos definir varias consultas tipo que cobren os aspectos analizados anteriormente, e executarémolas en cada unha das tecnoloxías sobre a versión 1.3 do proxecto CORGA, que inclúe 23.059.543 de palabras distribuídas en 1.366.257 oracións e estas, á vez, en 437 xornais, 86 revistas e 390 libros.

3.1 Consultas tipo

Definimos, a continuación, as consultas que serán empregadas na avaliación, dividindo as probas en baterías segundo o que se pretenda probar:

Batería 1

Con este conxunto de consultas trátase de termos un referente base para comparar os outros resultados:

Q1a: Obter o número de documentos e de oracións que coinciden coa expresión *desenrolo*. O obxectivo é ver o rendemento dunha consulta sinxela.

Q1b: Igual que a anterior buscando o prefixo *des-*. O obxectivo é ver o custo que supón a busca de prefixos.

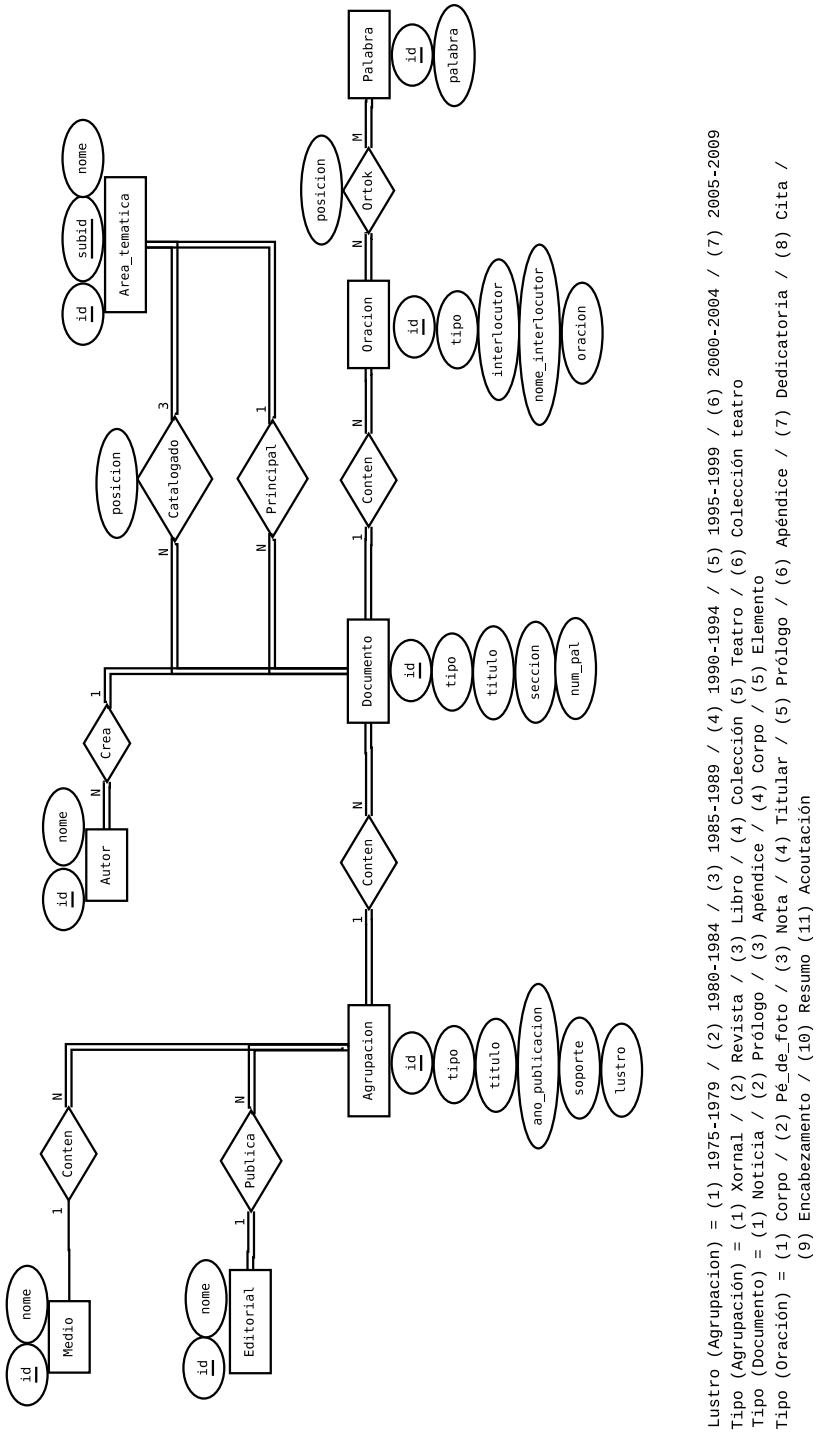

Figura III.3.1: Modelo entidade-relación da estrutura de base de datos do CORGA

Q1c: O mesmo para o sufixo *-ado*. Trátase de ver a penalización adicional na busca de sufixos.

Batería 2

Neste caso trátase de contrastar os tempos de resposta cos da batería anterior para medir a penalización que supón traballar coa táboa de palabras, que non é necesario utilizar na batería 1:

Q2a: Obter o número de documentos e de casos que coinciden coa expresión *desenrolo*.

Q2b: Idem para o prefixo *des-*.

Q2c: Idem para o sufixo *-ado*.

Batería 3

Con este conxunto de probas pretendemos obter a penalización na busca de expresións con dúas palabras, sobre todo cando se utiliza a táboa que contén as palabras almacenadas individualmente:

Q3a: Obter o número de documentos e de oracións que coinciden coa expresión *sen embargo*.

Q3b: Obter o número de documentos e de casos que coinciden coa expresión *sen embargo*.

Batería 4

Neste caso intentamos ver a penalización que supón a utilización de filtros nas buscas. Haberá que facer tamén a unión con todas as táboas periféricas.

Q4aa: Obter o número de documentos e de oracións que coinciden coa expresión *sen embargo* que teñan como área temática *Economía e política*.

Q4ab: Obter o número de documentos e de oracións que coinciden coa expresión *sen embargo* nas noticias de revistas.

Q4ac: Obter o número de documentos e de oracións que coinciden coa expresión *sen embargo* de documentos cuxo autor conteña no nome *Rivas*.

Q4ba: O mesmo que en Q4aa, pero obtense o número de casos no lugar do de oracións.

Q4bb: O mesmo que en Q4ab, pero obtense o número de casos no lugar do de oracións.

Q4bc: O mesmo que en Q4ac, pero obtense o número de casos no lugar do de oracións.

Q4ca: Obter o número de documentos e de oracións que coinciden coa expresión *sen embargo* nas noticias de revistas con área temática *Economía e Política* e cuxo autor sexa alguén que conteña no nome *Rivas*.

Q4cb: O mesmo que o anterior, pero obtense o número de casos en lugar do número de oracións.

Batería 5

Neste caso comprobamos a facilidade do xestor para compartir cálculos:

Q5a: Obter o número de documentos e de oracións que teñen a expresión *sen embargo* para cada tipo de medio, área temática principal e lustro. Dado que hai 3 tipos de medio, 6 áreas temáticas principais e 7 lustros, deben calcularse 32 valores diferentes.

Q5b: O mesmo que o anterior, pero obtense o número de casos en vez do número de oracións.

Batería 6

Trátase de ver o custo que ten facer a unión de todas as táboas só para obter valores das periféricas.

Q6a: Obter as oracións en que aparece a expresión *sen embargo*.

Q6b: Refacer a consulta para obter tamén o título dos documentos, a editorial, os autores, o medio, os temas, o ano de publicación e o tipo de oración.

Batería 7

Neste caso avaliamos o custo que supón a ordenación dos resultados.

Q7a: Repetir a consulta Q6a coa palabra *desenvolvemento*.

Q7b: Repetir a consulta Q6b coa palabra *desenvolvemento* e ordenar o resultado por ano de publicación, medio e área temática principal.

Batería 8

Esta batería proba o rendemento de obtención de valores da nómina de obras.

Q8a: Obter o número de palabras e de documentos de toda a base de datos.

Q8b: Obter o número de palabras e de documentos relativos a xornais con área temática *Economía e Política*.

Q8c: Obter o número de palabras e de documentos para cada medio, área temática principal e lustro (32 valores diferentes).

Q8d: O mesmo que o anterior pero só para os documentos da área temática *Economía e Política*.

3.2 Hardware e sistema operativo

O hardware utilizado para realizar as probas correspóndese cun servidor de gama media Prolliant de Hewlett-Packard [48] coas seguintes características:

Modelo: HP Prolliant ML310 G4

CPU: Dual-Core Intel Xeon 2,13 GHz

Bus: 1066 MHz

Caché: 2 Mb L2

Memoria: 3Gb

Disco duro: SAS de 136 Gb a 15.000 rpm

Sistema Operativo: CentOS 5.2

Para o sistema operativo elixiuse CentOS 5.2 [19] na versión de 64 bits cunha configuración gráfica e de servizos mínima para o servidor, de maneira que se atenuou, no posible, o impacto doutros procesos no rendemento das consultas. Os motivos principais polos que se escolleu foron a facilidade de instalación de Oracle neste sistema e que está pensado como unha solución seria para servidores. CentOS tamén sería unha opción factible no caso de querer facer a instalación do sistema de RI nun servidor en explotación e non soamente para as probas que nos ocupan.

3.3 Avaliación

Dado que nestes sistemas raramente se fai a mesma busca nun espazo curto de tempo, para obter os resultados correspodentes ás diferentes consultas das baterías definidas anteriormente, executouse cada unha tres veces xusto despois do arranque do xestor e colleuse a media destes tres valores. Deste xeito, evitamos que os tempos de resposta se vexan influenciados pola caché da tecnoloxía empregada.

Como podemos comprobar na figura III.3.2, en que se amosan os tempos medios de resposta en segundos obtidos para cada unha das baterías de consultas[1], xa a simple vista podemos confirmar que Oracle é o xestor que, en xeral, proporciona mellores tempos de resposta cun número elevado de buscas nun tempo de resposta por debaixo do segundo.

Están marcados en negriña os tempos de resposta que resultan claramente inaceptables para os sistemas que nos ocupan. A aparición dun x significa que non se puido obter resposta do sistema para a busca[2], polo que podemos ver como tamén Oracle ten problemas para resolver algunhas das buscas propostas (aínda que en menor medida que as outras tecnoloxías): presenta problemas graves con tres consultas fronte a oito de MySQL e de PostgreSQL. Se analizamos máis polo miúdo os resultados, podemos concluír tamén o seguinte:

1. O acceso a prefixos e sufixos, que devolven un número importante de entradas nos índices textuais, pon en apertos todos os xestores (consultas Q1b e Q1c). Na nosa opinión isto débese a que a perspectiva de integración dos índices textuais dentro dos sistemas relacionais aínda non está moi depurada.

[1] Pódense ver os detalles pormenorizados da definición das táboas de cada tecnoloxía e das consultas realizadas no apéndice D.

[2] Consideramos dúas causas posibles: que a consulta tarde máis de cinco minutos ou que, simplemente, o xestor non permita facela.

Un indexador textual puro non tería problema ningún con este tipo de consultas, polo que todo parece indicar que é a maneira en que se están a desenvolver eses índices no contexto relacional o que está a causar problemas.

Quizais a aplicación de índices baseados en autómatas [45] e/ou *arrays* de sufixos [10], é dicir, o emprego de técnicas propias dos indexadores textuais, así como a utilización de múltiples índices[3] sexan algúns camiños a seguir para mellorar eses tempos de resposta, pero o certo é que os desenvolvementos actuais non parecen ir por esa vía.

Vemos ademais como, cando se combina este acceso aos índices textuais con unións de táboas de relativa complexidade (consultas Q2b e Q2c), se pode agravar aínda máis esta problemática.

2. Comprobamos que, en xeral, o cálculo do número de casos fronte ao de oracións adoita ser bastante máis custoso, debido, sobre todo, ao `JOIN` que hai que facer coa táboa de palabras. Isto pode verse nas consultas Q3a fronte á Q3b ou na Q5a fronte á Q5b.

3. Dado que en PostgreSQL non se pode acceder ao índice textual cunha expresión multipalabra, funcionalidade en que se está traballando para a versión 8.4, non só é necesario utilizar a táboa das palabras para a consulta de casos, senón que tamén hai que facelo para obter os valores correspondentes ás oracións (Q3a, Q4aa etc.). Este feito implica que, aínda que para expresións dunha ou dúas palabras se obteñan valores aceptables, a medida que a expresión ten máis palabras, os tempos de resposta van empeorando. Isto non sucede cos outros xestores, xa que o acceso ao índice textual de Oracion cunha expresión multipalabra de dous ou máis elementos axuda a que se resolva a consulta en moito menos tempo.

   Adicionalmente, o feito de que PostgreSQL non permita a utilización de comodíns nas súas expresións de busca textual supón unha limitación moi importante para os sistemas que nos ocupan.

4. Todo parece indicar tamén que o planificador e optimizador das consultas dos diferentes xestores aínda non traballan todo o ben que deberan con respecto aos índices de texto, o que ás veces provoca que se obteñan tempos de resposta desproporcionadamente altos.

   É o caso de MySQL para as consultas Q4bc, Q4ca e Q4cb, onde parece que ao accederen a dous índices textuais, o da táboa de autores para filtrar polo nome do autor e o da expresión de busca sobre Oracion, o planificador non sabe priorizar e/ou optimizar ben o acceso aos índices textuais. Problemas desta índole tamén foron detectados en PostgreSQL (Q5a e Q6b) e en versións de Oracle anteriores á que aquí se está a avaliar.

   Tamén suceden cousas un pouco estrañas na consulta Q3b no caso de MySQL. Curiosamente, se eliminamos o filtro da expresión textual que actúa sobre Oracion, o que en teoría debe penalizar o tempo de resposta da consulta, obtéñense tempos de resposta moito máis aceptables.

Como conclusión a estas probas de rendemento podemos dicir que, aínda que Oracle ten, en xeral, os mellores tempos de resposta, tamén presenta problemas para resolver certas consultas de vital importancia neses contornos.

[3] Por exemplo, un para o acceso aos prefixos e outro diferente para os sufixos.

Consulta	Oracle 11g	PostgreSQL 8.3	MySQL 5.1
Q1a	1	<1	<1
Q1b	**90**	**x**	**85**
Q1c	**50**	**x**	**x**
Q2a	1,5	4,5	2
Q2b	20	4,5	**191**
Q2c	**x**	**x**	**x**
Q3a	1	4,7	2
Q3b	11	**x**	**30**
Q4aa	<1	4,3	2
Q4ab	<1	4,3	<1
Q4ac	<1	**31**	7
Q4ba	<1	4	4
Q4bb	<1	4	2
Q4bc	<1	3,6	**64**
Q4ca	<1	5	**78**
Q4cb	<1	5	**64**
Q5a	1	**37,5**	9
Q5b	18	21	13
Q6a	<1	7,5	<1
Q6b	3	**60**	1
Q7a	<1	<1	<1
Q7b	<1	<1	<1
Q8a	<1	<1	<1
Q8b	<1	<1	<1
Q8c	<1	<1	<1
Q8d	<1	<1	<1

Figura III.3.2: Tempos medios de resposta en segundos para cada unha das consultas de proba e tecnoloxías

Parece que os xestores de licenza libre non están lonxe de igualar as prestacións dos de licenza propietaria, pero o certo é que, nin as persoas que desenvolven os uns, nin os outros, teñen en mente os sistemas de RI que tratamos nesta investigación, onde o custo da introdución e modificación de datos é pouco relevante, a utilización dos índices textuais é vital e a redundancia podería ser unha vía para obter un aumento de rendemento neles.

Capítulo 4

O sistema de RI do CORGA

Previamente ao desenvolvemento deste traballo, o proxecto CORGA xa dispoñía dun sistema de RI que permitía abordar algunhas das consultas presentadas no capítulo 1 da parte III. Este sistema utilizaba a ferramenta dtSearch para facer buscas sobre a base documental no formato textual orixinal[1], pero non podía resolver algunhas cousas relevantes para o proxecto, entre as que destacan as seguintes:

1. Á hora de visualizar o contexto dun caso de ocorrencia, incorporar o concepto de oración, e non só facer a visualización en función dun número de palabras antes e/ou despois do caso que coincide coa busca.

2. Permitir amosar as secuencias de palabras que están noutro idioma á vez que as buscas non actúen sobre eses elementos.

3. Habilitar un criterio de busca para determinar as seccións dos textos en que se desexa buscar (só títulos dos xornais, resumos de noticias, prólogos, apéndices etc.).

4. Incorporar flexibilidade na combinación de criterios de ordenación dos resultados de xeito que poidan ser utilizados varios a un tempo.

É por isto polo que, paralelamente á adaptación dos documentos, se construíu un novo sistema de RI que traballa con Oracle, en que se aplicaron os coñecementos adquiridos para o desenvolvemento da presente tese.

4.1 Análise de requirimentos

Os requirimentos completos que cumpre o novo sistema son os seguintes:

Modos de busca básicos

- Frase exacta: Busca dos casos que coinciden coa expresión especificada.
- Calquera palabra: Busca das oracións que conteñen algunha das palabras da expresión.
- Todas as palabras: Busca das oracións que inclúen todas as palabras da expresión.

[1] Véxase a sección 2.8 da parte II para máis detalles sobre este formato.

- Booleana: Busca das oracións que verifican a expresión booleana definida na busca.

Ademais, os modos de busca permiten que o usuario poida decidir se desexa facer unha consulta sensible aos acentos ou non.

Comodíns

Tamén poden utilizarse dous comodíns:

* para indicar a substitución de cero ou máis letras.

? para indicar a substitución dunha única letra.

Criterios de busca

Os criterios de busca inclúen os seguintes: área e subárea temáticas, cuxos posibles valores se amosaron na figura II.2.10; intervalo de anos, que van desde o 1975 ata a actualidade; medio, que pode ser xornal, revista ou libro; e, por último, sección do documento, en que se poden especificar as zonas do documento onde se desexa buscar (corpo, pés de foto, notas ao pé, titulares de noticias, resumos de noticias, acoutacións no teatro, prólogos, apéndices, dedicatorias, citas e encabezamentos de seccións).

Estatísticas

Para os catro modos de busca básicos e os diferentes criterios de busca especificados, poden calcularse o número de casos/oracións e documentos. Os casos só son relevantes para o modo de busca básica *frase exacta*, mentres que para o resto se amosa o número de oracións.

Adicionalmente, o sistema obtén estatísticas relativas a medios, lustros e áreas temáticas principais. Así, para as estatísticas de medios, os valores deben incluír o número de oracións/casos e documentos para xornais, revistas e/ou libros; no caso dos lustros, datos de oracións/casos e documentos para os intervalos de anos 1975-1979, 1980-1984, 1985-1989, 1990-1994, 1995-1999, 2000-2004 e 2005-2009; e, para as áreas temáticas, datos de oracións/casos e documentos dos bloques temáticos economía e política, cultura e artes, ciencias sociais, ciencias e tecnoloxía, ficción e outros.

Contexto

O sistema amosa as palabras que coinciden coa expresión de busca, resaltándoas convenientemente no contexto. Este incorpora o concepto de oración. Así, nunha primeira listaxe dos resultados vese a oración completa en que aparece a expresión convenientemente resaltada e, nun segundo momento, pódese ampliar este contexto coas oracións anteriores e/ou posteriores.

Ordenación

Pódense ordenar os resultados por data, área temática, e medio. Para a ordenación por área temática e medio utilízase, como criterio de ordenación complementario, a data.

Exclusións textuais

As secuencias marcadas no texto como pertencentes a outra lingua vense de maneira diferenciada cando aparecen no contexto da expresión buscada. Porén, non se poden obter casos de palabras que coindiden coa expresión de busca nesas seccións, é dicir, as palabras delimitadas dese xeito quedan fóra das buscas.

Información adicional

Cando se amosan os casos coincidentes dáse tamén información relativa ao documento ao que pertencen (título, autor, editorial, área ou áreas temáticas, medio e ano de publicación).

Rendemento

Os tempos de resposta do sistema son razoables para os usuarios e, na maioría dos casos, limítanse a uns poucos segundos.

Aplicación web

O sistema é accesible desde un navegador, é dicir, é unha aplicación web.

Nómina

A aplicación inclúe un sistema de buscas para a nómina de obras que permite utilizar os criterios de busca descritos anteriormente. É dicir, pódese obter o número de casos e de documentos total con respecto aos criterios de busca que se seleccionan, así como os metadatos relacionados coas obras escollidas.

4.2 Análise e deseño

Nesta sección describiremos os elementos máis relevantes relacionados coa análise e o deseño da ferramenta. Non pretendemos abordar neste traballo todos os elementos propios da análise e deseño dunha aplicación, senón centrármonos naqueles máis relevantes que permitan comprender as particularidades desta.

4.2.1 Arquitectura da aplicación

A aplicación desenvolveuse utilizando a linguaxe Java [41] seguindo o patrón Modelo-vista-controlador (*Model-View-Controller*, MVC) [39, 84]. Aínda que na actualidade hai diferentes contornos de traballo que se desenvolven ao redor deste patrón [3], o nacemento do noso sistema é anterior á estabilidade destes proxectos, polo que se seguiu un esquema propio de despregamento.

Diagrama de clases da análise

Na figura III.4.1 amósase o diagrama de clases UML [16, 17] de alto nivel do sistema de buscas. Nel pode apreciarse a distinción entre as clases do controlador, as do modelo e as da vista.

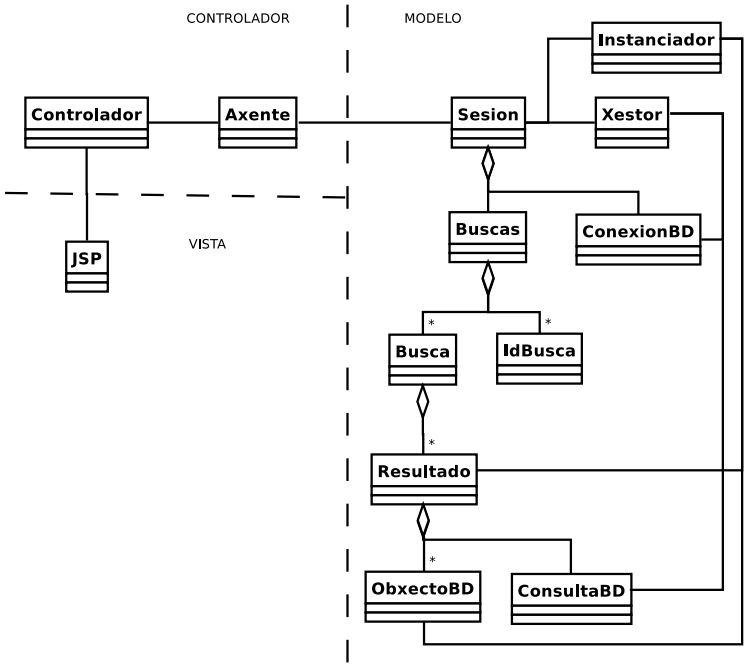

Figura III.4.1: Diagrama de clases da análise

Describimos a continuación as responsabilidades de cada un dos obxectos da análise:

- **Controlador, Axente e** JSP **[14]**: O controlador encárgase de recibir as diferentes solicitudes do navegador e delega no axente para que realice as accións oportunas. Unha vez que o axente remata, o primeiro delega, á súa vez, no JSP que corresponda para amosar os resultados da solicitude.

  Para que esta comunicación entre o controlador e os JSP sexa posible, o axente introduce na sesión HTTP, tanto os obxectos necesarios para amosar o resultado, como os valores necesarios para futuras intervencións do usuario.

- **Sesion**: Representa unha sesión de traballo dun usuario e materializa a caché de consultas de tal modo que, ao navegar polos resultados dunha busca, esta non ten que refacerse. É por isto polo que o obxecto de tipo Sesion aglutina os diferentes obxectos imprescindibles para manter a información da sesión, é dicir, os datos necesarios para resolver sucesivas peticións do usuario.

- **Xestor**: Clase encargada de crear as conexións coa base de datos e de facer as consultas sobre ela.

- **ConexionBD**: Clase que representa a conexión coa base de datos. En xeral, cada usuario ten unha conexión asociada á súa sesión de traballo.

- **Instanciador**: Clase encargada de instanciar os obxectos resultado dunha consulta.

- **Busca e Resultado**: O tipo de obxecto Busca representa unha consulta dun usuario e aglutina os diferentes resultados que se poden obter dela. É dicir, hai unha busca asociada a un conxunto único de criterios que define o usuario no formulario pero, para esa consulta, poden terse varios resultados (máis dun).

  Por exemplo, un usuario fai unha busca dunha palabra e obtén o número de casos e de documentos que conteñen esa palabra. A continuación quere ver os casos no seu contexto. Desta forma estamos ante unha única consulta que ten dous resultados: nun primeiro momento un valor numérico e, despois, os contextos de aparición da palabra.

- **Buscas**: Aglutina as diferentes buscas simultáneas que pode ter un usuario. Isto pode ocorrer cando un mesmo usuario ten abertos varios navegadores en que está a realizar diferentes consultas. Haberá un obxecto de tipo Busca por cada navegador conectado ao sistema. No noso caso impuxemos un límite de cinco consultas simultáneas por usuario.

- **ObxectoBD e ConsultaBD**: Un resultado pode estar formado por un ou máis obxectos de tipo ObxectoBD ou por un de tipo ConsultaBD.

  O primeiro caso dáse cando o resultado dunha busca está formado por un conxunto pequeno de valores (da orde de unidades ou decenas) de modo que o obxecto Resultado almacena os ObxectoBD correspondentes aos resultados indiviuais. Se o usuario necesita obter novamente eses datos, a aplicación non volvería facer a consulta debido a que xa dispón deles.

  Por contra, no segundo caso, o obxecto Resultado retén unha representación da execución da consulta (ConsultaBD). Isto sucede cando o resultado dunha busca é un número potencialmente grande de datos (centos ou miles) e necesítase navegar páxina a páxina a través deles. Ao mantermos unha representación da consulta permitimos acceder ás diferentes páxinas do resultado sen que haxa que refacela.

O diagrama de clases do subsistema da nómina é exactamente igual que o do sistema de buscas amosado na figura III.4.1. Só habería que engadir o sufixo *Nomina* ao nome de todas as clases excepto a do controlador, que é compartido polos dous subsistemas.

Diagramas de secuencia

Desenvolvemos nesta sección os diagramas de secuencia correspondentes aos casos de uso *autenticar* e *facer consulta* que nos servirán para descubrir os aspectos máis relevantes da arquitectura da aplicación.

Autenticar

Na figura III.4.2 amósase o diagrama de secuencia correspondente ao caso de uso *autenticar*. A secuencia descríbese a seguir:

1. Un usuario utiliza o navegador e accede á páxina para autenticarse no sistema.

2. Cobre os datos relativos ao usuario e ao contrasinal e preme no botón Enviar.

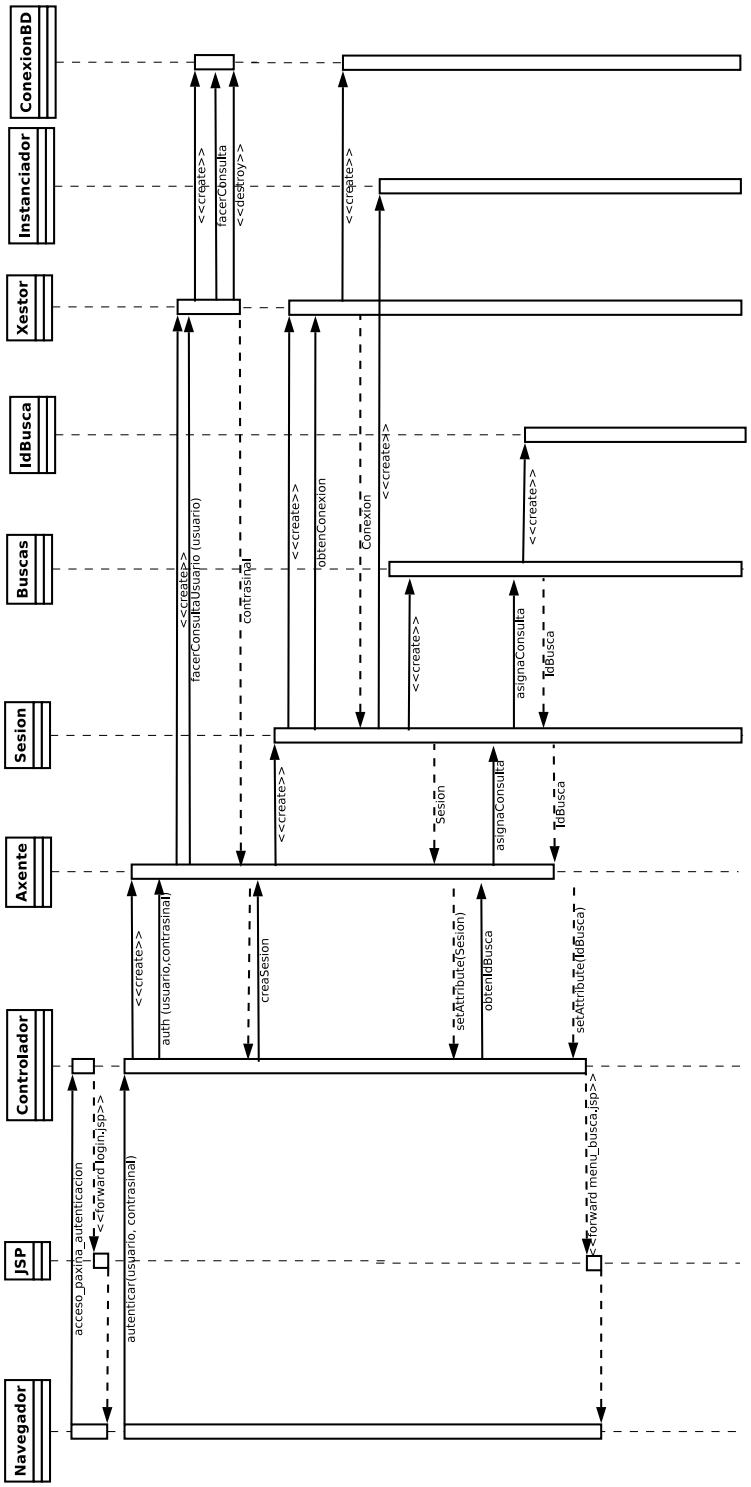

Figura III.4.2: Diagrama de secuencia da análise para o caso de uso *autenticar*

3. A solicitude de autenticación chega ao controlador, que delega nun axente para xerar a sesión do usuario.

4. O axente crea un obxecto de tipo Xestor para que este, á súa vez, faga unha consulta á base de datos e comprobe que o usuario está rexistrado no sistema.

5. O axente crea a sesión do usuario (Sesion), e esta crea todos os obxectos necesarios para as súas futuras interaccións: Xestor, Instanciador, a conexión coa base de datos (ConexionBD) e o obxecto que xestiona o acceso ás diferentes buscas que se poderán facer na sesión (Buscas).

6. O axente introduce o obxecto Sesion na sesión HTTP.

7. O controlador delega no axente para que se asigne un identificador de busca necesario para xestionala. Finalmente, encárgalle a un JSP amosar o formulario de busca no navegador.

Facer consulta

Neste punto o usuario ten na súa pantalla o formulario para facer unha busca e entra en xogo o caso de uso *facer consulta*, que se describe na figura III.4.3:

1. O usuario cobre os campos do formulario de busca e a solicitude chega ao controlador.

2. O controlador crea un axente e delega nel a execución da consulta.

3. O axente interactúa co obxecto Sesion para que este, á súa vez, localice o obxecto Busca correspondente ao formulario do usuario co fin de que que se encargue da execución e instanciación da consulta.

4. O obxecto Busca crea un obxecto Resultado, que obtén unha representación da consulta (ConsultaBD) do xestor e a executa.

5. O obxecto Busca instancia o resultado a través do obxecto Resultado, que utiliza o instaciador para realizar esta tarefa.

6. O axente interacciona novamente co obxecto Sesion para obter o resultado e introducilo na sesión HTTP.

7. Finalmente, o controlador delega nun JSP para que amose o resultado na sesión HTTP.

Estes diagramas de secuencia son tamén válidos para o subsistema da consulta da nómina.

4.2.2 Base de datos

Cando se trata con corpus estruturados de grandes dimensións, a elección da estrutura da base de datos que dará soporte a todas as consultas do sistema é un dos aspectos máis complexos de resolver. Un servidor de gama media ou alta, xunto cun xestor de base de datos de primeira liña, é fácil que teñan problemas para mover con soltura a resolución de consultas dos sistemas que nos ocupan.

Aínda que existen infinidade de posibilidades para materializar a base de datos, o certo é que, a grandes trazos, podemos clasificalas en dúas:

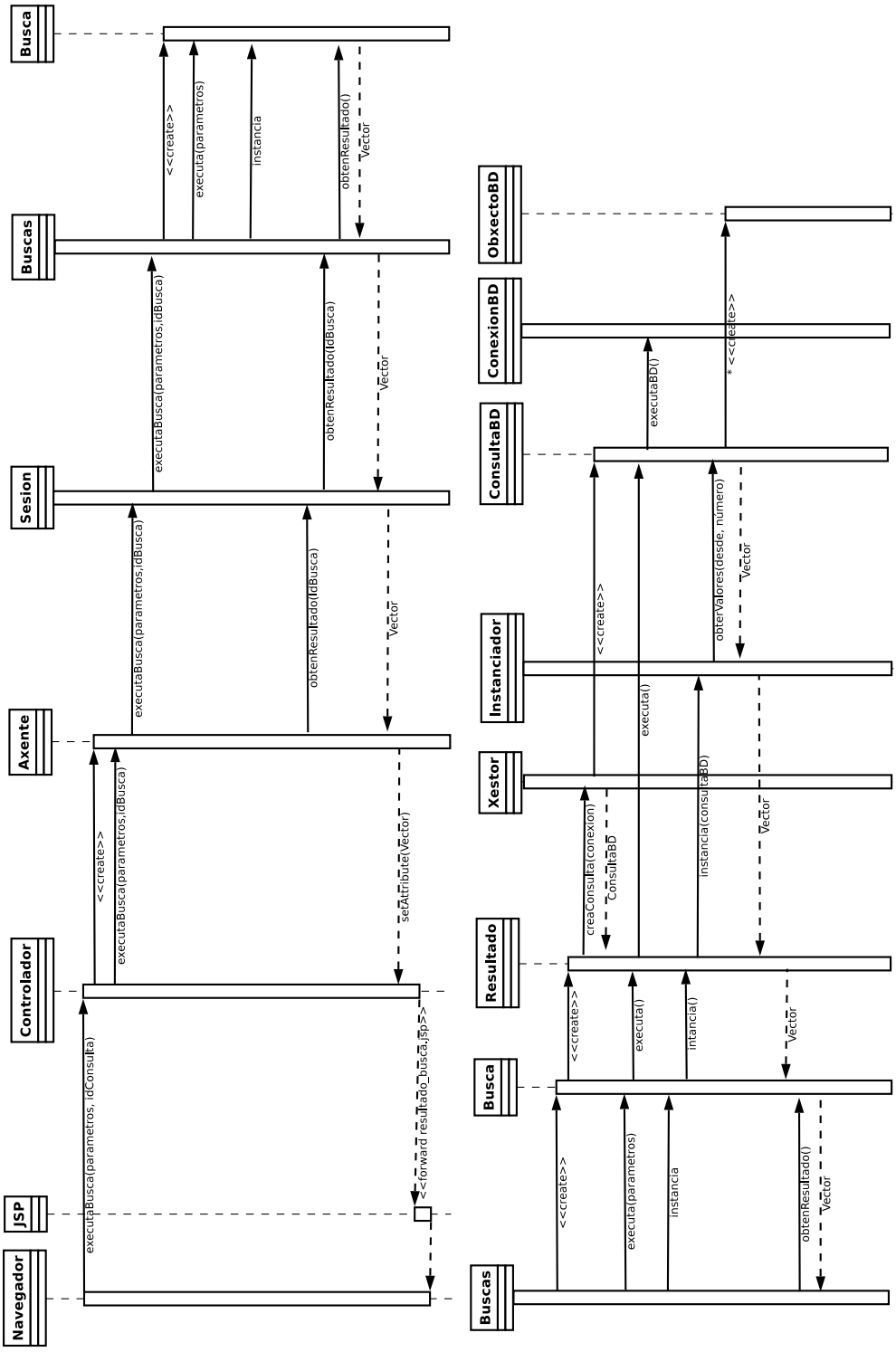

Figura III.4.3: Diagrama de secuencia da análise para o caso de uso *facer consulta*

1. As que priorizan o contido non estruturado, é dicir, o texto propiamente dito dos documentos. É, por exemplo, a que utiliza Mark Davies en diferentes traballos [26, 27, 28].

   Este tipo de sistemas acostuman desenvolver as consultas puramente lingüísticas dun xeito moi eficiente, pero non permite utilizar criterios de máis alto nivel para delimitar o seu ámbito de aplicación. Para intentar solucionalo parcialmente, créanse bases de datos paralelas que teñen datos precalculados para diferentes elementos de interese pero, aínda así, seguen presentando problemas, xa que os criterios para os que se poden obter eses valores son definidos a priori por quen desenvolve o sistema e non a posteriori polo usuario, como sería desexable.

2. As que sitúan ao mesmo nivel o texto e a estrutura dos documentos, que é a proposta que defendemos neste traballo. Aínda que algúns tipos de consultas poden ser un pouco máis lentas que no primeiro caso, a súa flexibilidade é moito maior, e os usuarios poden definir criterios de busca que abranguen elementos estruturais e de metainformación dos textos.

Neste traballo, xa que logo, centrámonos na posibilidade de manexar a metainformación dos textos, o que loxicamente repercutirá na estrutura da base de datos escollida.

De maneira recíproca a cando falamos da estruturación dos documentos na sección 1.1.6 da parte II, a estrutura da base de datos tampouco debe estar condicionada unicamente pola dos documentos. Evidentemente, deberemos ter en conta a segunda para construír o modelo de datos, pero non nos deixaremos levar pola estrutura rica e flexible que teñen os textos, xa que, do contrario, construiriamos un sistema moi ineficiente.

Para o caso do proxecto CORGA, en que hai varios tipos de documentos (xornais, revistas, libros, coleccións, obras teatrais e coleccións de obras teatrais), sería inviable termos un conxunto de táboas independente para cada tipo de documento. Aínda que deste xeito o sistema de buscas sería moi flexible, para facer unha consulta en toda a base documental necesitariamos executala unha vez para cada tipo de texto, o que resultaría extremadamente ineficiente no mellor dos casos. Mesmo podería haber problemas para amosar un único conxunto de resultados (independentemente do tipo de documento).

Daquela, o máis axeitado para este caso é extraer factor común das diversas tipoloxías textuais e construír unha estrutura compartida que concentre os aspectos relevantes dos diferentes documentos. É probable que deste modo algún elemento moi específico non encaixe e se sacrifique algún matiz da flexibilidade, mais é esperable que estas perdas sexan poucas e de menor importancia, co que se obterá un sistema moito máis eficiente sen que deixe de ser moi flexible.

O modelo entidade-relación da aplicación é o que xa se amosou para as probas na figura III.3.1. Analizámolo embaixo.

Representación dos documentos na base de datos

A estrutura básica de representación dos documentos está formada polas entidades Agrupacion, Documento e Oracion, e concentra nas dúas primeiras o almacenamento dos metadatos dos documentos.

Así, no caso dun xornal ou revista, os metadatos relativos ao xornal sitúanse na entidade Agrupacion, e os relativos ás diferentes noticias, na Documento. No caso dun libro, os datos estarán en Agrupacion e repetiranse en Documento. Adicionalmente, un libro poderá ter

prólogos e apéndices doutros autores que incluirán os seus metadatos tamén en Documento. Finalmente, para as coleccións, os elementos relativos á colección irán en Agrupacion, mentres que os metadatos asociados aos diferentes elementos dela situaranse en Documento.

Esta forma de organizar a información fai que as consultas se fagan contra un único banco de datos, de xeito que se optimiza o tempo de resposta das buscas máis frecuentes, as que actúan sobre todo o conxunto de documentos.

Por outra banda, dado que o que constitúe un documento neste modelo entidade-relación pode ter ata tres áreas temáticas e, en ocasións, é necesario ordenar por este criterio, faise necesaria unha relación adicional (Principal) entre Documento e Area_tematica. Esta representa a asociación dos documentos coa súa área temática principal e permite que nos resultados non saian documentos repetidos.

Destacamos o atributo *num_pal* de Documento, que inclúe o número de palabras dos documentos para dar soporte ás consultas da nómina e *sección*, cuberto coas seccións dos xornais en que aparecen as noticias. Neste último caso conseguimos incluír información específica dun só tipo de texto sen deixar de utilizar unha base de datos común.

Compoñentes textuais

A entidade Oracion dá cabida ás diferentes oracións dos documentos. Nela inclúese, ademais do texto da propia oración, a zona do documento en que aparece[2] e, en caso de obras teatrais ou de coloquios, o nome do interlocutor asociado a esta[3]. No caso de entrevistas tamén se fai referencia a se a oración corresponde ao entrevistador ou ao entrevistado[4].

Por outra banda, a entidade Palabra contén as palabras das oracións do texto. Estas únense coas oracións en que aparecen a través da relación Ortok, que inclúe información sobre a posición de cada palabra na oración. Esta relación utilízase nas consultas que fan uso do atributo *posicion* en expresións multipalabra.

Materialización en táboas

Son varios os aspectos susceptibles de ser analizados no modelo entidade-relación presentado con respecto a como se materializa este modelo no esquema relacional subxacente. Como cabe imaxinar, debe prestarse especial atención a non esquecerse de xerar os diferentes índices da base de datos, especialmente os que se corresponden coas claves foráneas a outra táboas que permiten resolver as unións de maneira eficiente. Mais hai outros aspectos non tan triviais que deben terse en conta:

(a) Entidades débiles

En primeiro lugar, podemos observar que concretamos as entidades Documento e Oracion como entidades fortes. De entrada sería natural pensar que Documento é unha entidade débil dependente de Agrupacion e que Oracion, á súa vez, é dependente de Documento. Deste xeito, a táboa Documento incorporaría como campo o identificador de Agrupacion e Oracion dous campos correspondentes aos identificadores de Agrupacion e Documento respectivamente.

[2] É o campo *tipo*, que contén valores numéricos que se identifican coas zonas: corpo do documento, pé de foto, nota ao pé, titular de noticia etc.

[3] Campo *nome_interlocutor*.

[4] Campo *interlocutor*.

O problema desta aproximación radicaría en que, dado que é posible a aparición de tuplas repetidas en diferentes consultas debido, principalmente, a que un documento pode ter ata tres áreas temáticas, a cláusula `DISTINCT` de SQL xoga un papel importante para obter os resultados desexados de maneira eficiente. Xa que esta cláusula só pode recibir como parámetro unha única columna, resultan moito máis sinxelas de realizar as consultas se esas entidades se tratan como fortes, incorporando un campo único coa clave primaria de cada unha.

Polo tanto, resulta de moita axuda materializar as entidades como fortes, aínda que, se o desexamos, podemos incorporar igualmente as claves foráneas das outras entidades para optimizar algunhas consultas[5].

(b) Relación N:M coa táboa Palabra

En segundo lugar, pode observarse que a relación entre as entidades Oracion e Palabra, chamada Ortok, ten cardinalidade N:M, polo que a táboa Palabra incluirá unha única tupla por cada palabra diferente que hai no texto. Facelo así ten a vantaxe de que o índice de texto desa táboa é moito máis pequeno e, en consecuencia, as consultas a el son máis eficientes. Pola contra, despois o xestor ten que resolver a unión con Ortok e Oracion para facer as consultas.

Se materializásemos esa relación mediante unha cardinalidade 1:N, a táboa Palabra incluiría unha entrada por cada ocorrencia dunha palabra no texto. Isto faría esa táboa moito máis grande e, polo tanto, a consulta do índice textual moito máis lenta, pero a unión con Oracion sería moito máis sinxela.

Tras varias probas utilizando as dúas alternativas non se atoparon diferenzas claras entre apostar por unha ou outra, xa que o custo da resolución da unión no primeiro caso é equiparable á resolución do índice textual do segundo.

(c) Xustificación da táboa Palabra2

Coa solución que se amosou na figura III.3.1, existe un problema cando se fai unha busca de casos dunha expresión que ten exactamente dúas palabras. Para resolver estas consultas, o xestor debe acceder ás táboas Palabra e Ortok dúas veces de xeito que se compara o atributo *posicion* da primeira palabra da expresión con *posicion+1* da segunda. Mais a resolución pode tardar un tempo inaceptable por tres motivos:

1. É habitual que, polo menos, un dos dous compoñentes da expresión de dúas palabras sexa moi frecuente, co que hai moitas tuplas resultantes da unión entre Ortok e Palabra que coindiden coa busca textual.

2. Aínda que nos filtros se actúe sobre o índice de texto da táboa Oracion para filtrar pola expresión completa de dúas palabras, tamén é moi probable que esa expresión verifique un número elevado de tuplas da táboa Oracion.

3. Facer o `JOIN` con Ortok dúas veces, que é unha táboa con moitas tuplas (unha por cada palabra do corpus), é moi custoso e a comparación das posicións, aínda que se utilicen índices funcionais, non resulta tan eficiente como cabería esperar.

[5]Por exemplo, se unha busca que actúa sobre a táboa Documento filtra os resultados unicamente por un tipo de medio, non é necesario facer o `JOIN` con Agrupacion e pode conectarse directamente a táboa Documento con Medio utilizando o identificador de Agrupacion, que neste caso estaría accesible en Documento.

Estes tres motivos combinados fan, como dixemos, que estas consultas poidan resultar excesivamente ineficientes. Para solucionalo creamos unha nova táboa, denominada Palabra2, que contén as secuencias diferentes de dúas palabras e está relacionada con Oracion a través dunha nova asociación (Ortok2). Deste xeito, só se utiliza unha vez Ortok2 e non se empregan os atributos *posicion*.

Para tres ou máis palabras o problema xa non se dá. Esas expresións de tres ou máis palabras teñen moita menos frecuencia que as dunha lonxitude menor. Por isto, se a busca utiliza o índice textual da táboa Oracion para localizar a expresión completa, redúcese considerablemente a expansión da unión con Ortok e Oracion de modo que a consulta se realiza nun tempo aceptable.

Igual ocorre coas buscas dun único termo, onde non é necesario utilizar o atributo *posicion* e utilízanse Ortok e Palabra unha soa vez, resultando consultas de resolución eficiente.

Tratamento do atributo *posicion*

Para a materialización do atributo *posicion* da táboa Ortok utilizáronse números consecutivos non repetidos, é dicir, non son números en que para cada oración empecen en un e rematan no número de palabras da oración, senón que seguen incrementándose entre oración e oración.

Desta forma conséguese que, cando se utiliza *posicion* para facer comparacións, só haxa unha tupla na táboa que verifique o valor de *posicion*. Se definimos un índice único (`UNIQUE`) optimizamos o acceso á información.

Eliminación de unións innecesarias

Finalmente, compre comentar algo relacionado simultaneamente co modelo relacional e a aplicación que fai as buscas: debe terse especial coidado en non utilizar táboas e unións innecesarias cando se fan as consultas á base de datos. Habitualmente, o obxecto que se encarga de construír estas últimas (no noso caso Xestor) implementa un sistema xenérico que pode tender a usar táboas de máis para a resolución das buscas.

Por exemplo, se unha consulta se pode facer utilizando unicamente as táboas Documento e Oracion, non ten sentido que estea tamén a táboa Agrupacion nin a unión correspondente, xa que o servidor empregaría máis tempo en resolvela. É por isto polo que deberemos ter especial coidado na formulación de consultas para que só se utilicen as táboas relevantes en cada caso.

Como se pode apreciar, na práctica resulta de utilidade saírse un pouco dos conceptos académicos dos modelos relacionais, aínda que é necesario ser consciente de como e por que os estamos modificando. Noutro tipo de sistemas estas optimizacións son unha opción, mais aquí son unha necesidade, xa que do contrario o sistema podería resultar inservible.

4.2.3 Interface do usuario

Para o desenvolvemento da interface do usuario seguimos dúas máximas:

1. Dado que o cliente da aplicación debe ser o navegador web, esta aplicación debería funcionar en calquera navegador máis ou menos recente.

2. Tería que verse correctamente nas resolucións de pantalla máis utilizadas (desde 1024x768 ata 1280x1024).

Tendo en conta isto, a aplicación debe incluír os criterios de busca desagregados na análise de requirimentos:

- Campo textual para os termos de busca.
- Marca de sensibilidade ou non aos acentos.
- Selección do tipo de busca: todas as palabras, calquera palabra, frase exacta e busca booleana.
- Selección de área e subárea temática.
- Medio: xornal, revista ou libro.
- Rango de datas en que se desexa realizar a busca.
- Ámbito da busca: corpo, pé de foto, nota ao pé, titular de noticia, resumo de noticia, acoutación, prólogo, apéndice, dedicatoria, cita ou encabezamento de sección.
- Criterio de ordenación: data, medio ou tema.
- Facilidades de navegación polos resultados.

Ademais, considerouse interesante que, malia ser unha aplicación web, tivese un mecanismo de interacción similar ao dunha aplicación non web, polo que se deseñou unha barra de ferramentas cos botóns para realizar as diferentes accións. Na figura III.4.4 amósase a páxina principal de buscas.

Pódese observar como, preto da zona superior, hai unha barra onde aparecen os botóns que se poden utilizar en cada momento. Neste caso son, de esquerda a dereita os seguintes: pechar a sesión de traballo, ver a páxina de axuda, ir á páxina principal e cancelar a consulta actual.

Unha vez que se introduce a expresión de busca e se seleccionan os criterios de filtrado que se desexan, prémese en Buscar e aparece un primeiro resultado co número de casos/oracións[6] e o número de documentos, tal e como se reflicte na figura III.4.5.

Observamos que na botoeira aparecen un selector para definir o criterio de ordenación que se desexa, e dous novos botóns que serven para, de esquerda a dereita, ver os casos de ocorrencia e as estatísticas respectivamente. Ao premermos no primeiro dos botóns obtemos os casos de ocorrencia, e aparecerá o que se amosa na figura III.4.6, é dicir, os exemplos resaltados no seu contexto.

Logo, xorden na barra de ferramentas novos botóns, de esquerda a dereita: gardar os resultados da páxina en formato XML, volver ao resultado inicial, ir ás estatísticas (similar ao que había tamén no resultado inicial) e os que se corresponden co desprazamento polas diferentes páxinas de casos de ocorrencia.

Finalmente, se prememos no botón correspondente ás estatísticas, obtemos o resultado dos diferentes medios, áreas temáticas e lustros que se amosa na figura III.4.7. Obviamente, os criterios de selección da busca modifican estas estatísticas, é dicir, os datos calcúlanse en tempo real e non están prealmacenados. Para ver os casos relacionados con algún dos campos presentados, non hai máis que premer sobre o número subliñado que lles corresponda.

[6]Casos no tipo de busca *frase exacta* e oracións no resto de tipos de busca.

Desde aí, pode irse novamente, a través dos botóns superiores, ao resultado inicial ou á consulta dos casos e, en calquera momento, introducir unha nova busca e volver comezar todo o proceso.

Por outra banda, o subsistema de consulta da nómina de obras e autores presenta un formulario inicial como o que se amosa na figura III.4.8. Introdúcense os datos polos que se desexa facer a busca e prémese en Buscar (figura III.4.9). Neste caso amósanse o número de palabras e de documentos da área temática Economía e Política e subárea Política que hai no corpus. Isto permite obter frecuencias normalizadas, é dicir, en todo momento podemos saber o tamaño do subcorpus subxacente que temos para as buscas que estamos a facer.

Desde ese resultado, podemos consultar os datos relativos aos documentos que compoñen esa base documental (figura III.4.10) ou ás estatísticas por medio, lustro e área temática (figura III.4.11). Isto é, a navegación no sistema de buscas da nómina é moi similar ao de expresións.

Ademais, conectamos o sistema de busca de expresións co da nómina, de xeito que se poden obter os valores da nómina correspondentes aos filtros da consulta que estamos facendo dunha maneira máis sinxela. Isto faise desde o resultado inicial da busca dunha expresión (figura III.4.5), en que aparece un interrogante xusto ao lado do número de documentos que é, precisamente, a conexión coa nómina.

4.3 Desenvolvemento e implantación

Aínda que para as probas do capítulo 3 da parte III se utilizou o hardware alí descrito, o certo é que actualmente o sistema en explotación [21] está funcionando sobre un computador un pouco menos recente. A súa descrición é a seguinte:

Modelo: AlphaServer ES40 68/833 M1

CPU: 2 a 833 MHz

Bus: 100 MHz

Memoria: 3Gb

Disco duro: 8 discos SCSI de 36,4 Gb a 15.000 rpm.

Sistema operativo: TRUE64 Unix 5.1B

Ademais, a versión do xestor Oracle nese servidor é a 9ir2 e, dado que a 11g non está dispoñible para esa arquitectura e que a versión 9ir2 presenta algúns problemas coa funcionalidade dos sistemas que aquí presentamos, é previsible que este hardware non tarde en renovarse. En calquera caso, os tempos de resposta da maioría das consultas que se realizan no sistema son considerados aceptables, e xa se leva dando este servizo varios anos.

Por outra banda, a aplicación web desenvolvida fíxose utilizando Java 1.5 [86] como linguaxe de programación, JDBC [85] para o acceso á base de datos e Tomcat 5.5 [4] para dar o servizo. A computadora que ten o servidor Tomcat non é a mesma que a que inclúe o xestor, polo que describimos a continuación as súas características técnicas:

Modelo: HP dc7600C

CPU: Dual-Core Intel Pentium 4 a 3,20 GHz

Bus: 100 MHz

Caché: 2 Mb

Memoria: 1Gb

Disco duro: 2 discos SATA de 80 Gb

Sistema operativo: Debian 5.0 Lenny

Comprobamos que un servidor de gama baixa é suficiente para aloxar a aplicación, xa que o tapón está na base de datos, que é a que ten que facer as consultas.

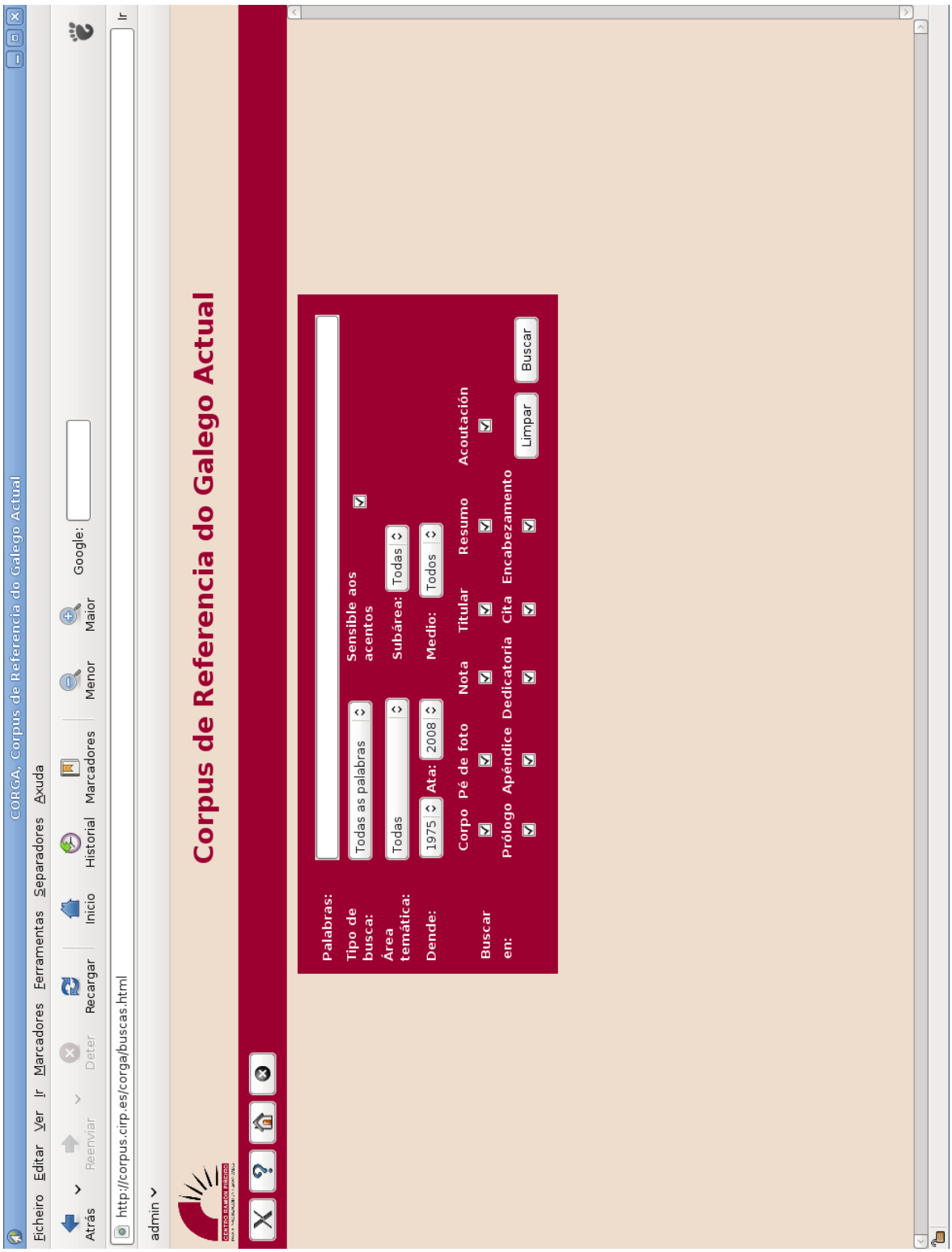

Figura III.4.4: Formulario de buscas

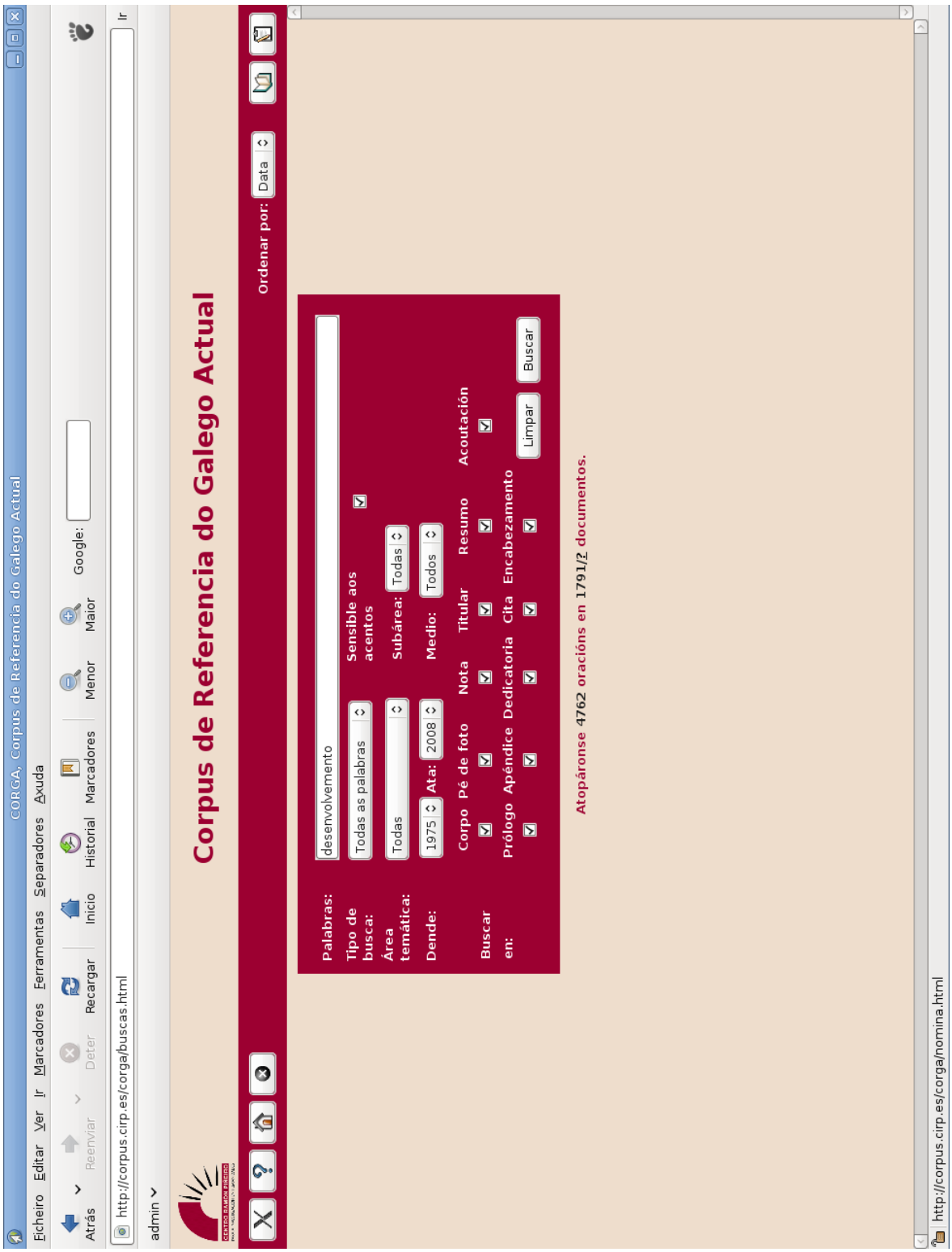

Figura III.4.5: Resultado inicial

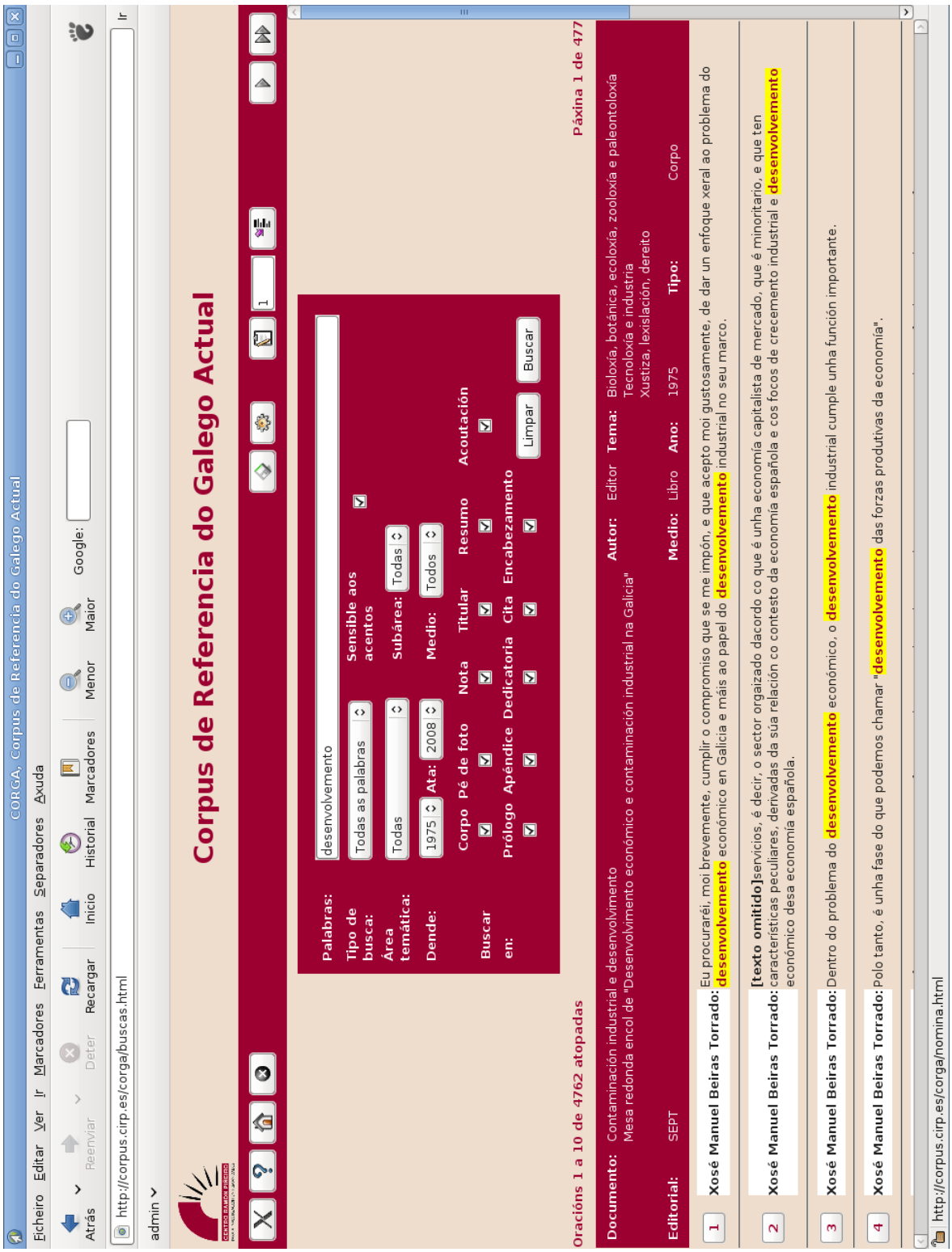

Figura III.4.6: Consulta dos casos

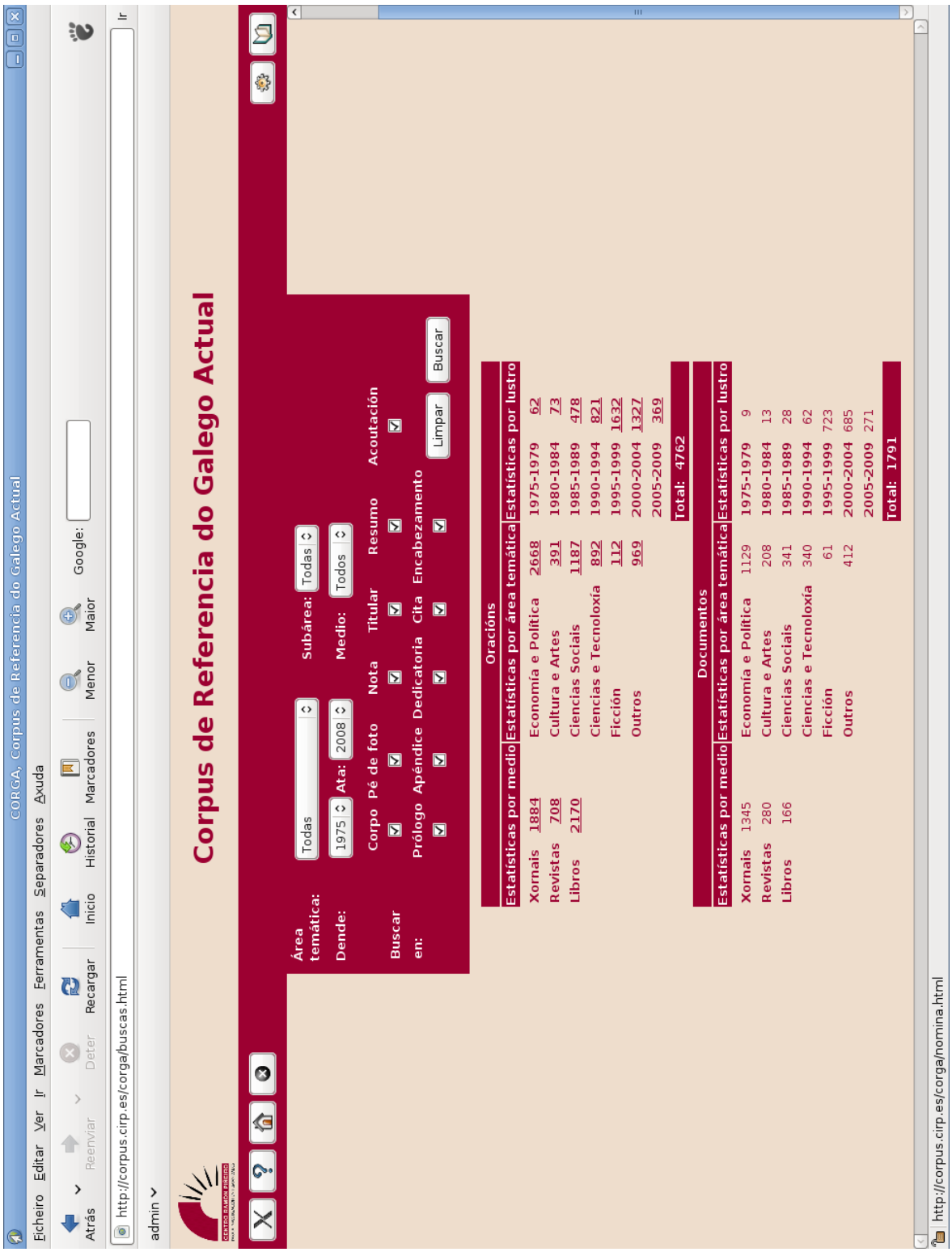

Figura III.4.7: Consulta das estatísticas

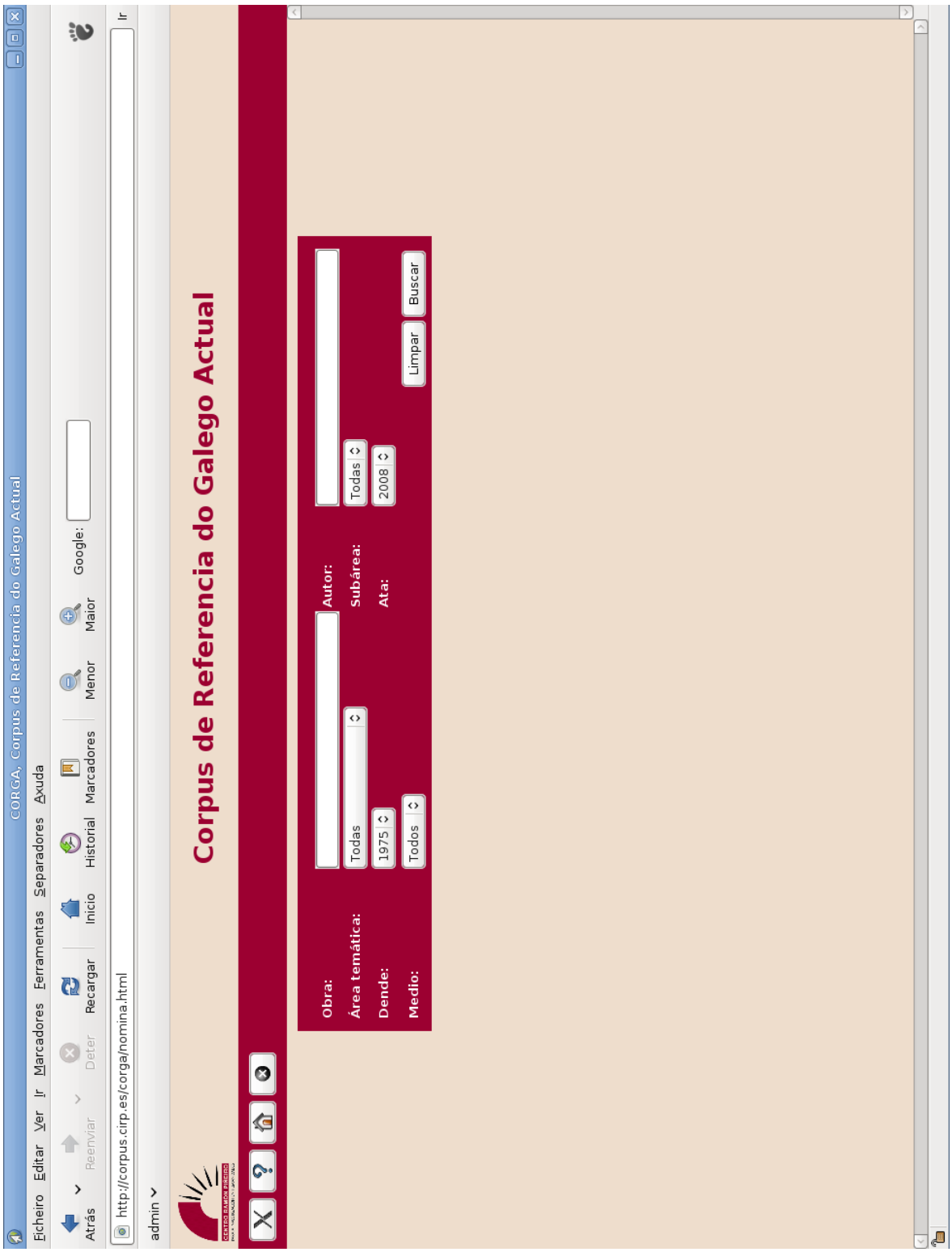

Figura III.4.8: Formulario da nómina

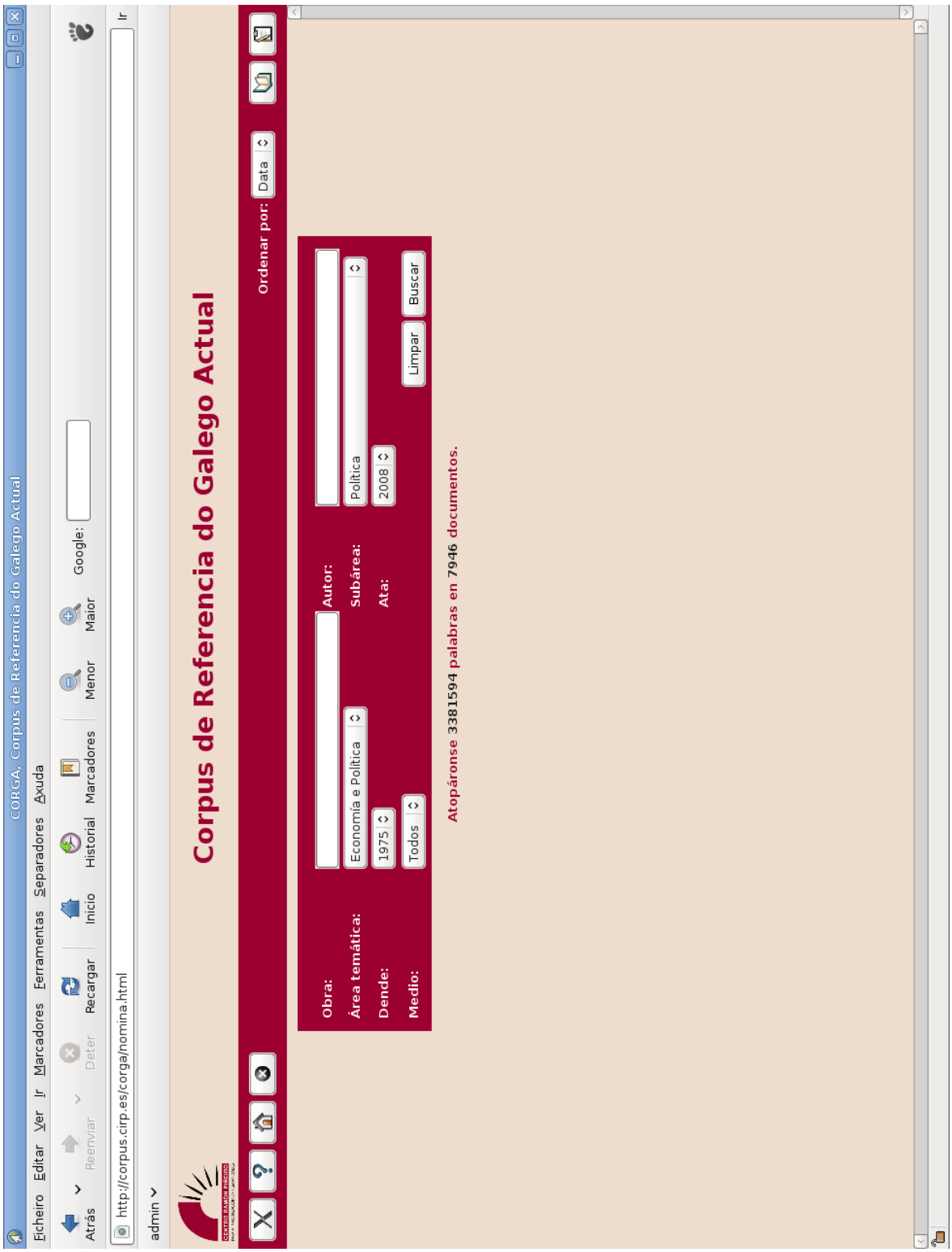

Figura III.4.9: Resultado inicial da nómina

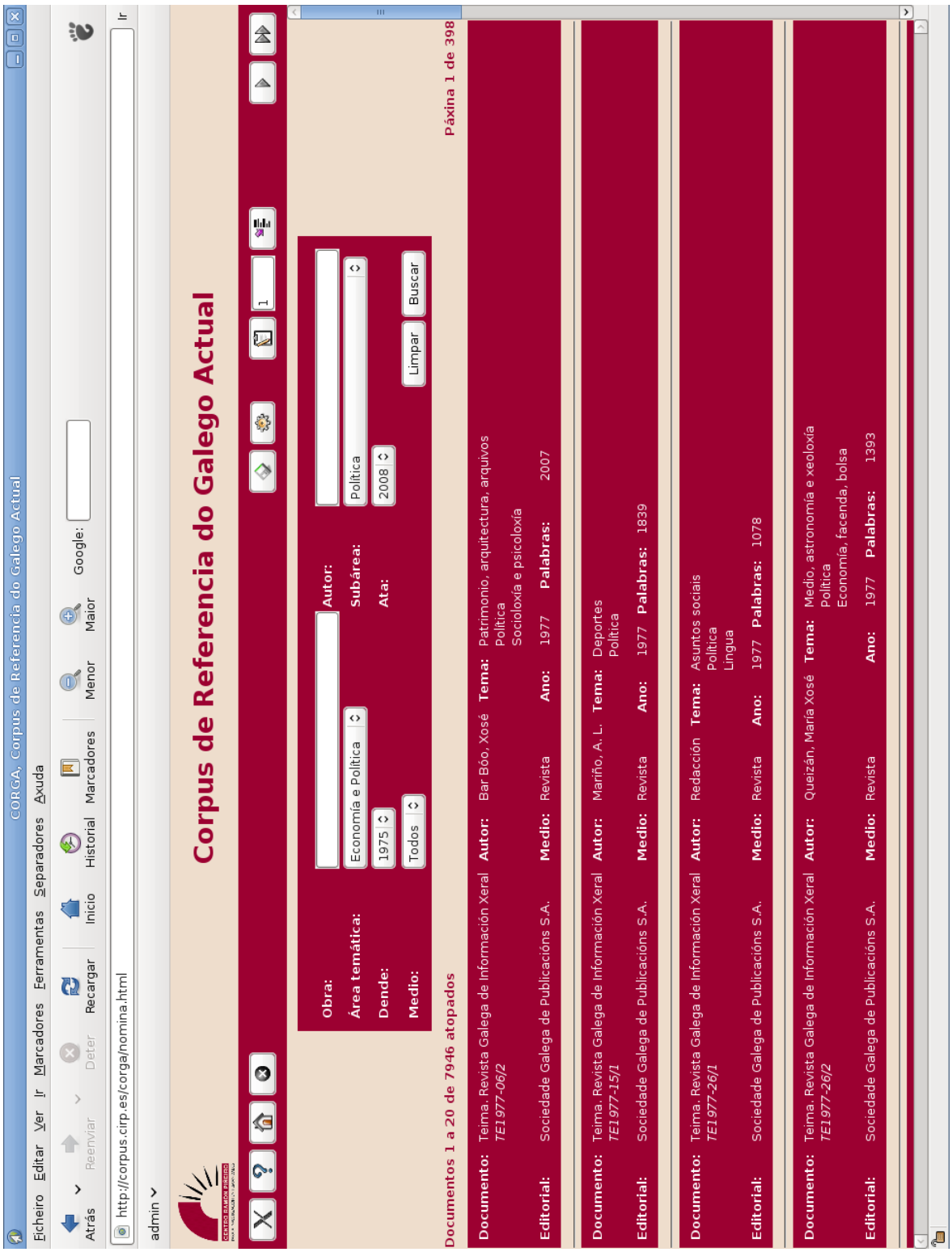

Figura III.4.10: Resultado dos documentos que figuran na nómina

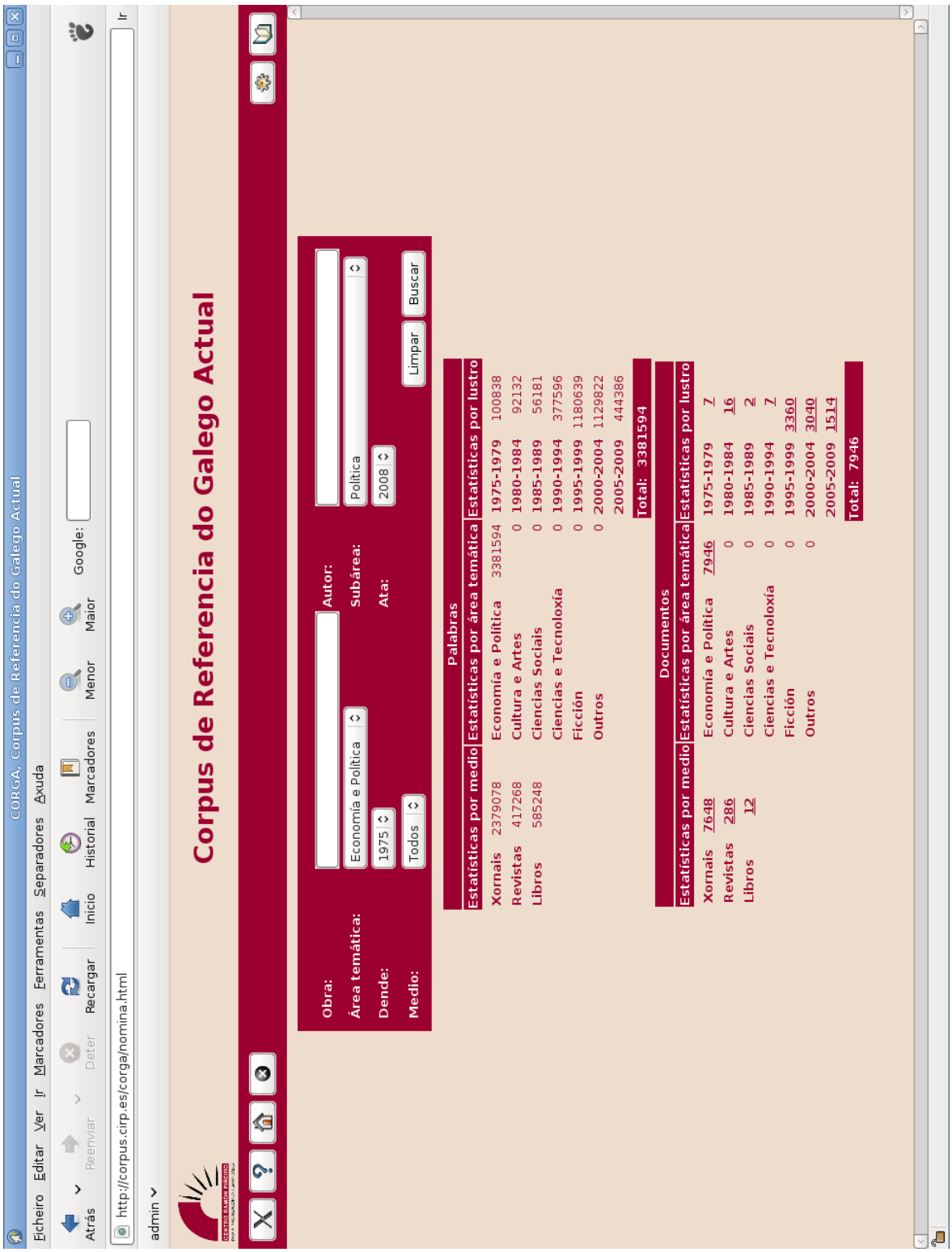

Figura III.4.11: Estatísticas da nómina

Parte IV

Conclusións e liñas de traballo futuro

Ao longo do presente traballo de investigación tratáronse diversos temas relacionados coa lingüística de corpus e, máis concretamente, coa creación e explotación de corpus textuais de grandes dimensións.

No tocante á construción de corpus, son moitos os traballos que tratan a problemática dun concreto, e moi pouca a información dispoñible relativa ás fases necesarias polas que debe pasar un documento, desde o seu medio orixinal ata que está incluído no corpus. É máis, se falamos de traballos que tenten xeneralizar os procesos de cada caso particular para que sexan de utilidade en proxectos que compartan algunhas características, a información é practicamente nula.

Desde a construción dos primeiros corpus ata a actualidade, os sistemas informáticos foron evolucionando moito, mentres que a aparición de traballos que procuraron xeneralizar os procesos de construción de corpus foi moi escasa. O expoñente máis representativo é do ano 1992 [9] e, desde aquela, case non houbo intentos por definir unha nova metodoloxía.

Verbo da construción dos sistemas de RI concentramos os nosos esforzos no estudo dos que actúan sobre corpus de grandes dimensións xa que, para bases documentais pequenas, practicamente todo é posible, e non aparecen as limitacións tecnolóxicas aquí presentadas.

Existen moitos e diversos puntos de vista sobre as tecnoloxías que se poden utilizar e sobre como estruturar os datos para construír sistemas de RI que permitan facer consultas nun corpus. Algúns traballos abordan o problema primando o acceso á información de máis baixo nivel (o texto propiamente dito), en detrimento da de máis alto rango (metainformación, compoñentes estruturais etc.), mentres que no noso caso consideramos igual de importantes os dous tipos de información.

Xa que logo, tendo en conta estas dúas frontes, tratamos o tema desde unha visión de conxunto, é dicir, propuxemos unha metodoloxía concreta para a construción de corpus á vez que determinamos a tecnoloxía máis axeitada para desenvolver os sistemas de consulta, e todo iso ilustrado cos exemplos dun caso concreto: o CORGA.

Conclusións

Con respecto á proposta dunha metodoloxía para construír corpus estruturados, fixemos especial fincapé na xeneralización dos diferentes procesos a que se debe someter un documento ata que pasa a formar parte dun corpus, debido precisamente á ausencia de traballos recentes sobre o tema. O resultado foi unha proposta metodolóxica que esperamos resulte de utilidade para outros proxectos.

Ademais, esta proposta foi acompañada dunha aplicación concreta ao proxecto do CORGA, o que enriqueceu o estudo cunha visión práctica que complementa os conceptos máis abstractos. Aínda que quedou fóra do ámbito de discusión desta investigación, cómpre salientar que o feito de utilizarmos noutros proxectos con resultados satisfactorios esta mesma metodoloxía, corroborounos que, efectivamente, é suficientemente versátil para ser empregada en diferentes contextos.

Por un lado, as alternativas para a construción de corpus son moi variadas, cada unha coas súas vantaxes e inconvenientes e, polo outro, as tecnoloxías que se poden empregar tamén o son. Polo tanto, na actualidade hai multitude de alternativas para levar a cabo os contidos que tratamos aquí, mais cremos que o camiño que encetamos neste traballo é prometedor.

A metodoloxía proposta apóiase nas tecnoloxías e estándares máis actuais (que tamén están a dar os seus froitos en ámbitos diferentes ao aquí tratado), utiliza ferramentas sinxelas, promove

a documentación e define os ámbitos de comunicación interdepartamentais. Con estes catro compoñentes conseguimos que o custo da posta en práctica desta metodoloxía ante un novo proxecto non sexa excesivo, fronte aos beneficios que achega: estandarización, detección de erros, coherencia, documentación de protocolos etc. Isto garante, ademais, que o corpus evolucione moito mellor no tempo e que os cambios sexan máis previsibles e menos traumáticos e, en definitiva, que o recurso sexa de maior calidade.

Na segunda parte da investigación, en cambio, tratamos a construción de sistemas de RI que utilizan corpus como o seu sustento de datos. Tamén nesta parte fixemos un esforzo por xeneralizar a análise de requirimentos, para o que utilizamos, á parte do CORGA, a información extraída doutros sistemas de busca que desenvolvemos.

A cantidade de tecnoloxías que, a priori, poderían ser candidatas ao desenvolvemento deste tipo de sistemas é realmente inimaxinable, polo que tivemos que facer un inventario e unha posterior clasificación. Finalmente, agrupámolas en indexadores de texto, aplicacións lingüísticas, bases de datos relacionais e xestores XML, e intentamos avaliar as que máis posibilidades tiñan dentro do contorno GNU/Linux para dar unha solución baseada totalmente en software libre, o que abarataría moito os custos.

En calquera caso, o certo é que as tecnoloxías probadas baseadas en software libre aínda non demostraron estar á altura doutras propietarias, cando menos non nos aspectos que nós necesitabamos. Aínda así, o que máis nos sorprendeu foi que, incluso coas propietarias atopamos máis dificultades das que nun comezo pensabamos.

Desde un punto de vista puramente técnico non parecen existir tantos problemas como os que deixaron patentes todas as tecnoloxías testadas, xa que cousas que están bastante ben resoltas nunhas tecnoloxías son os problemas doutras e viceversa.

Na nosa opinión isto sucede por dous motivos. Por unha banda, o desenvolvemento da funcionalidade que dá soporte ás buscas deste tipo de sistemas de RI é relativamente recente, co que aínda hai moita marxe para a mellora e, pola outra, os requirimentos que son imprescindibles para as ferramentas obxecto do estudo non son un obxectivo prioritario nas diferentes tecnoloxías. É evidente que estas necesidades son menos demandadas ca outras (movementos bancarios, facturación, xestión etc.), polo que as prioridades de desenvolvemento destes proxectos non coindiden coas dos sistemas que explotan os corpus.

Finalmente, confiamos en que tamén sexan de utilidade as pinceladas que ofrecemos sobre o esquema da análise do sistema de RI do CORGA. Neste caso, o emprego de patróns de deseño no paradigma obxectual, así como o seguimento das tendencias actuais da programación orientada a obxectos, permiten manexar unha estrutura clara e fácil de manter.

Liñas de traballo futuro

No futuro agardamos aplicar a metodoloxía proposta a novos proxectos. Dado que unha metodoloxía non debe ser algo estático, os casos vindeiros de aplicación contribuirán a desvelar as particularidades que non foron tratadas neste traballo. Xa que logo, haberá que ir actualizando, revisando e mellorando a proposta a medida que se vaian presentando elementos que, ou ben non se tiveron en conta, ou ben non encaixan totalmente nos esquemas expostos. A finalidade última desta adaptación dinámica será incorporar as necesidades que vaian aparecendo, tanto á materialización metodolóxica do caso concreto como á abstracción propiamente dita.

Pero ademais de facer que esta metodoloxía evolucione respecto á ampliación do seu radio de acción, facéndoa válida en cada vez máis tipos de corpus, tamén é necesario adaptala ás novas correntes tecnolóxicas. Dada a velocidade á que aparecen tecnoloxías máis modernas que substitúen outras que quedan rapidamente obsoletas, será necesario estarmos atentos a estes cambios para vermos de que forma poden afectarlle e, así, actualizala cando for oportuno.

Noutra orde de cousas, canto ao desenvolvemento dos sistemas de RI, é imprescindible seguir, tanto a evolución das ferramentas probadas, como a doutras que poidan ir xurdindo, para vermos en que grao poden darlles solución aos requirimentos aquí expostos. Prestaremos especial atención ás baseadas en software libre que, como dixemos, poden abaratar considerablemente os custos de construción e explotación destes sistemas.

Nesta liña, resultaría produtivo intervirmos directamente no desenvolvemento da tecnoloxía de software libre que consideremos máis prometedora. Cremos que, por exemplo, o emprego de índices textuais baseados en autómatas finitos ou *arrays* de sufixos, combinado coa redundancia de índices para mellorar o rendemento dos distintos tipos de acceso que se lles poden facer, pode axudar a conseguir algúns dos obxectivos perseguidos.

Por último, a extensión deste estudo ao ámbito dos corpus anotados será tamén unha das principais liñas do noso traballo no futuro. De feito, xa estamos avanzando neste sentido, establecendo fases de procesamento documental adicionais, seleccionando ferramentas que as permitan levar a cabo, estudando os requirimentos particulares deste tipo de sistemas e avaliando diferentes tecnoloxías que permitan dar cumprimento a estas novas necesidades.

Como se pode comprobar, aínda son moitas as frontes abertas en que se pode avanzar para conseguir unha mellora nas posibilidades de busca dos sistemas de RI baseados en corpus, polo que confiamos en que esta investigación sexa a primeira de moitas outras que dean continuación a esta vía aberta.

Parte V

Apéndices

Apéndice A

Exemplo de protocolo para a adquisición de documentos en papel

1. Desprazámonos ao ordenador que ten o escáner.
2. Acendemos o escáner.
3. Acendemos o ordenador conectado a el.
4. Arrancamos Omnipage premendo en *Inicio* > *Programas* > *ScanSoft Omnipage Pro 12.0*.
5. Cando arranca a ferramenta, prememos en *Herramientas* > *Opciones* > *Todo...*, despois en *Cargar parámetros...* e imos ao directorio C:\escaneados:
   (a) Se non dispoñemos do documento en follas soltas:
      i. Escollemos o arquivo documento_compacto.dat.
      ii. Seleccionamos *Escanear en ByN*, e prememos en *Obtener página(s)*.
      iii. Poñemos na superficie do escáner a primeira páxina, aliñada na parte superior dereita.
      iv. Prememos en *Obtener página(s)*.
      v. Colocamos a seguinte páxina no alimentador.
      vi. Prememos en *Obtener página(s)*.
      vii. Repetimos estes dous últimos pasos para cada unha das páxinas do documento.
   (b) Se dispoñemos do documento en follas soltas escritas por unha cara:
      i. Escollemos o arquivo follas_soltas_simple_cara.dat.
      ii. Colocamos o montón de follas no alimentador do escáner.
      iii. Seleccionamos *Escanear en ByN* e prememos en *Obtener página(s)*.
      iv. Esperamos a que se escaneen todas as follas.
   (c) Se dispoñemos do documento en follas soltas que están a dobre cara:
      i. Escollemos o arquivo de parámetros follas_soltas_dobre_cara.dat.
      ii. Colocamos o montón de páxinas no alimentador automático.
      iii. Seleccionamos *Escanear en ByN* e prememos en *Obtener página(s)*.
      iv. Seleccionamos a opción *1, 3, 5, ...* na ventá que nos sae e prememos en *Aceptar*.

v. Cando remate, viramos o montón de páxinas e volvémolas colocar no alimentador.

vi. Seleccionamos a opción *2, 4, 6, ...* na ventá que nos sae e prememos en *Aceptar*.

vii. Esperamos a que se escaneen todas as follas.

6. Gardamos o documento: Prememos en *Archivo* > *Guardar como...*, seleccionamos o formato *Documento de Omnipage (*.opd)* en C:\escaneados e poñémoslle como nome o título do documento utilizando guións baixos en lugar de espazos.

7. Copiamos o documento en J:\corga\corga_xml_adquisicion\escaneados, e eliminámolo de C:\escaneados.

Apéndice B

Exemplo de protocolo para a estruturación dun xornal

1. Creamos un directorio coas iniciais do xornal, o ano con catro díxitos, o mes con dous díxitos e, finalmente, o día con dous díxitos en J:\corga_xml_estruturacion\xornais. Referirémonos a este directorio como directorio principal.

2. Copiamos o arquivo cabeceira_xornal.txt modelo situado en J:\corga_xml_estruturacion e pegámolo no directorio principal.

3. Abrímolo co editor de texto e cubrimos os datos de todas as etiquetas.

4. Gardámolo.

5. Se o xornal está almacenado en varios arquivos HTML na fase de adquisición:

   (a) Abrimos o arquivo principal do xornal co navegador.

   (b) Creamos un subdirectorio dentro do directorio principal por cada sección que presente o xornal. O nome destes subdirectorios será algún dos seguintes: Actualidade, Galicia, Cultura e Sociedade, Opinión, Deportes, Suplemento, Internacional, Economía, España, Área de Compostela ou TV.

   (c) Abrimos o editor de texto.

   (d) Prememos nas ligazóns de cada unha das noticias por orde e, para cada unha delas:

      i. Creamos un arquivo novo co editor de texto (*Archivo* > *Nuevo*).

      ii. Seleccionamos no navegador o texto da noticia co rato.

      iii. Prememos en *Editar* > *Copiar* no navegador.

      iv. Prememos en *Editar* > *Pegar* no editor.

      v. No editor de texto:

         A. Separamos o titular do resumo cunha liña en branco. Se non hai titular debe aparecer igualmente unha liña en branco.

         B. Separamos o resumo do corpo da noticia cunha liña en branco. Se non hai resumo introducimos unha liña en branco extra, é dicir, aparecerán tres liñas en branco entre o titular e o corpo da noticia.

C. Separamos cada parágrafo cun retorno de carro e eliminamos as liñas en branco que puidera haber que non se correspondan coas dos dous apartados anteriores.

D. Englobamos entre <COMMENT> e </COMMENT> as secuencias de texto que claramente están noutra lingua.

E. Gardamos o arquivo de texto (*Arquivo > Gardar como...*) co nome noticia_x.txt, onde *x* debe substituírse polo número correlativo da noticia no xornal, dentro do subdirectorio da sección que lle corresponda.

vi. Copiamos o arquivo cabeceira_noticia.txt modelo situado en J:\corga_xml_estruturacion e pegámolo no subdirectorio onde acabamos de gardar a noticia.

vii. Renomeamos ese arquivo como cabeceira_x.txt, onde *x* debe corresponderse co número do arquivo que contén o texto da noticia.

viii. Abrímolo de novo e cubrimos todos os datos das etiquetas.

ix. Gardamos o arquivo.

6. Se dispoñemos do xornal en formato PDF:

   (a) Abrimos o arquivo do xornal co Omnipage.

   (b) Creamos un subdirectorio dentro do directorio principal por cada sección que presente o xornal. O nome destes subdirectorios escóllese da listaxe que aparece paso 2(b).

   (c) Seleccionamos o dicionario personalizado indo a *Proceso > Realizar OCR > Parámetros de OCR*, elixindo na pestana *Diccionario del usuario*, *galego*.

   (d) Para cada noticia:

      i. Seleccionamos no Omnipage as áreas que conteñan o texto[1]. Resulta de utilidade definir unha área independente por cada columna de texto da noticia.

      ii. Seleccionamos *Columna única, sin tabla* xusto debaixo do botón cos anteollos (*Realizar OCR*).

      iii. Prememos en *Realizar OCR*.

      iv. Prememos en *Exportar Resultados*, o botón que ten a carpeta debuxada.

      v. Gardamos o resultado como texto (*Arquivo de texto *.txt)* co nome noticia_x.txt, onde *x* debe substituírse polo número correlativo da noticia no xornal, dentro do subdirectorio da sección que corresponda.

      vi. Prememos co botón dereito sobre a área recoñecida e eliximos *Seleccionar todo*.

      vii. Prememos co botón dereito sobre a área recoñecida e escollemos *Borrar*.

      viii. Abrimos co editor de textos o arquivo gardado.

      ix. Unimos o texto de xeito que quede unha liña por cada parágrafo e eliminamos os guións correspondentes ás palabras que estaban divididas en dúas liñas.

      x. Separamos o titular do resumo cunha liña en branco. Se non hai titular aparecerá igualmente unha liña en branco.

      xi. Separamos o resumo do corpo da noticia cunha liña en branco. Se non hai resumo introducirase unha liña en branco extra.

[1] Icona laranxa cun T dentro.

xii. Separamos cada parágrafo cun retorno de carro e eliminamos as liñas en branco que puidera haber que non se correspondan coas dos dous apartados anteriores.

xiii. Englobamos entre <COMMENT> e </COMMENT> as secuencias de texto que claramente están noutra lingua.

xiv. Gardamos o arquivo.

xv. Copiamos o arquivo cabeceira_noticia.txt modelo situado en J:\corga_xml_estruturacion e pegámolo no subdirectorio onde acabamos de gardar a noticia.

xvi. Renomeamos o arquivo como cabeceira_x.txt, onde *x* debe corresponderse co número do arquivo que contén o texto da noticia.

xvii. Abrímolo e cubrimos os datos de todas as etiquetas.

xviii. Gardamos o arquivo.

7. Se o resultado da adquisición é un documento Omnipage, os pasos son exactamente os mesmos que no caso dos documentos PDF, coa excepción de que entre os pasos *xiii* e *xiv* hai que revisar o texto para corrixir os erros de OCR que se puideran producir. No arquivo J:\corga_xml_estruturacion\erros_OCR_tipicos.txt amósase unha listaxe dos erros máis comúns que se deben revisar.

Apéndice C

Exemplo de protocolo para a revisión dun xornal

1. Copiamos de J:\corga_xml_conversion\xornais a C:\corga_xml_revision\xornais o documento que se ten que revisar.

2. Abrímolo co editor XML (XMLmind XML Editor).

3. Comprobamos a súa validez premendo en *Herramientas* > *Comprobar la validez del documento*. Se a estrutura do documento non é válida, abrirase unha ventá cos problemas atopados. Iremos premendo en cada unha das mensaxes e corrixindo os problemas segundo corresponda en función da mensaxe de erro, tal e como describimos a continuación:

   - *El contenido del elemento <contido_noticia> no es válido.* Isto sucede cando unha noticia do xornal non ten titular e só ten un parágrafo como contido, que no documento XML aparece no titular da noticia. Neste caso o que hai que facer é transformar o titular da noticia no seu corpo.

     Para iso, seleccionamos o elemento *titular* que está a causar o problema, imos a *Editar* e prememos en *Convertir el elemento titular*. Apareceranos unha ventá coa opción *corpo*, seleccionámola e verificamos, de modo que o problema fica corrixido.
   - *El atributo requerido "tipo" del elemento "distinto" no está especificado.* Aparece esta mensaxe de erro cada vez que hai unha secuencia de palabras delimitadas coa etiqueta <COMMENT> na fase de estruturación. Trátase de especificar que tipo de distinción ten esa secuencia de palabras. Polo tanto, seleccionamos o elemento *distinto* que está a causar o erro, prememos en *Herramientas* > *Editar atributos* e, na frecha que aparece ao lado de *valor* escollemos a opción que corresponda: *outra_lingua*, *non_normativo*, *outro_período* ou *descoñecido*. Finalmente pechamos a ventá e continuamos.

4. Cando están corrixidos todos os erros, comprobamos novamente a validez do documento. Debe aparecer na barra inferior do editor unha mensaxe confirmándoa (*El documento es válido*).

5. Collemos a versión orixinal do xornal, ben en papel, ben en formato dixital (o arquivo ou arquivos resultantes da fase de adquisición).

6. Comprobamos a cabeceira do documento.

(a) Comprobamos que todos os campos están presentes e cubertos correctamente.

(b) Seleccionamos o noso nome en *responsable_ versión*.

(c) Comprobamos se a publicación ten ISBN ou ISSN, ademais do depósito legal, e inserímolos ou completámolos se for necesario. Para isto, prememos no elemento *depósito_ legal* e imos a *Editar* > *Insertar después*, seleccionando ISBN ou ISSN. Despois prememos no elemento en cuestión para cubrir o campo.

(d) Inserimos o elemento *comentarios* se temos algo que expor sobre o documento.

7. Para cada unha das noticias do xornal:

(a) Comprobamos a cabeceira da noticia.

i. Facemos unha comprobación xeral para asegurármonos de que estean todos os campos pertinentes cubertos correctamente.

ii. Inserimos o elemento *comentarios* se temos algo que expoñer sobre a noticia.

(b) Revisamos o contido da noticia.

i. Fixámonos no *titular* e nos primeiros parágrafos do *corpo*, e comprobamos se están correctamente estruturados. Pode ocorrer que:

- A noticia non teña orixinalmente titular, e apareza como titular o primeiro parágrafo do corpo. No caso de que a noticia só teña un único parágrafo (é dicir, no documento XML non aparece o corpo), podemos arranxalo premendo en *Edición* > *Convertir el elemento titular*, e seleccionando no menú que se despregа *corpo*. En caso contrario, cortamos o parágrafo situado en *titular*, inserímolo como primeiro parágrafo do corpo e eliminamos o elemento titular (*Editar* > *Eliminar el elemento titular*).
- O primeiro parágrafo que aparece no corpo sexa en realidade resumo da noticia. Deberemos seleccionar o elemento *corpo* e ir a *Editar* > *Insertar antes del elemento corpo*, seleccionamos *resumo* e despois cortamos o parágrafo e pegámolo dentro do elemento *resumo*.
- Os primeiros parágrafos que aparecen no corpo sexan pés de foto no documento orixinal. Neste caso haberá que crear tantos elementos *pé_ de_ foto* como pés de foto haxa, indo a *Editar* > *Insertar antes del elemento corpo* e seleccionando *pé_ de_ foto*. Despois cortamos os parágrafos un a un e pegámolos no elemento *pé_ de_ foto* que lles corresponda.
- Haxa combinacións diferentes da fenomenoloxía anterior, que haberá que resolver combinando o explicado anteriormente.

ii. Comprobamos a separación en parágrafos e oracións e corriximos as anomalías, é dicir, as segmentacións en oracións mal feitas, ben pola aparición de abreviaturas ou siglas, ben pola existencia de puntos e coma ou dous puntos que segmentan oracións.

Neste caso debemos realizar diferentes tarefas en función da problemática: crear parágrafos e/ou oracións utilizando as funcións do menú *Editar* > *Insertar después de* ou *Insertar antes de* tendo seleccionando un parágrafo ou unha oración (segundo proceda); dividir un parágrafo en dous, colocándonos no

lugar de segmentación e premendo a tecla *INTRO*; fusionar dous parágrafos, seleccionando o segundo e premendo as teclas *CTRL+INTRO*; eliminar un parágrafo, seleccionándoo e premendo a tecla *SUPR* etc.

iii. Corriximos os elementos *distinto*. Cando o elemento *distinto* englobe o texto de todas as oracións dun parágrafo, colocarase o atributo homónimo no parágrafo correspondente (seleccionando o parágrafo e indo a *Herramientas* > *Editar atributos* e elixindo o valor correcto do atributo) e eliminarase o elemento distinto dentro das oracións[1]. En cambio, cando o elemento *distinto* englobe o texto dunha oración completa, haberá que crear unha nova oración co atributo do mesmo nome, copiar o texto correspondente nesta nova oración e eliminar finalmente a oración orixinal.

iv. Incluímos as referencias a notas. Aínda que no documento XML aparecerán correctamente estruturados os elementos *nota*, haberá que engadir a referencia á nota no lugar en que se referencie no texto. Deberemos posicionarnos no lugar en que debe figurar, premer en *Editar* > *Insertar dentro*, seleccionar *referencia_nota* e, despois, en *Herramientas* > *Editar atributos* escoller o identificador da nota que lle corresponda.

v. Revisamos e corriximos os interlocutores no caso das entrevistas, por se houbo algún erro á hora de especificalos na fase de estruturación, ou por se se decidiu pospoñer a marcaxe dos nomes dos interlocutores ata esta fase. No caso dun coloquio cubrirase o atributo do parágrafo *nome_interlocutor*, e no dunha entrevista seleccionarase o valor *Entrevistador* ou *Entrevistado* do atributo *interlocutor* do parágrafo. Isto debe facerse con todas as oracións referidas a un coloquio ou entrevista.

[1] Para cada oración, creamos outra baleira xusto antes ou despois, copiamos o texto que contén o distinto nesa nova oración e, finalmente, eliminamos a orixinal, a que contén o elemento *distinto*.

Apéndice D

Modelo relacional e consultas realizadas

D.1 Oracle

D.1.1 Definición das táboas e índices

A continuación, amósanse as sentencias SQL utilizadas para a creación das táboas e índices:

Configuración do índice textual

```
begin
  ctx_ddl.create_preference('lexico','BASIC_LEXER');
  ctx_ddl.set_attribute('lexico', 'INDEX_TEXT', 'YES');
  ctx_ddl.set_attribute('lexico', 'INDEX_THEMES', 'NO');
  ctx_ddl.set_attribute('lexico', 'PROVE_THEMES', 'NO');
  ctx_ddl.set_attribute('lexico', 'PRINTJOINS', '-');
  ctx_ddl.set_attribute('lexico', 'BASE_LETTER', 'NO');
end;
/

begin
ctx_ddl.create_preference('almacenamento', 'USER_DATASTORE');
ctx_ddl.set_attribute('almacenamento', 'procedure', 's_obtentexto_corga');
ctx_ddl.set_attribute('almacenamento', 'output_type', 'CLOB');
end;
/
begin
  ctx_ddl.create_preference('lista_palabras', 'BASIC_WORDLIST');
  ctx_ddl.set_attribute('lista_palabras', 'SUBSTRING_INDEX', 'YES');
  ctx_ddl.set_attribute('lista_palabras', 'STEMMER', 'NULL');
  ctx_ddl.set_attribute('lista_palabras', 'WILDCARD_MAXTERMS', 50000);
end;
/

begin
  ctx_ddl.create_stoplist('lista_stop', 'BASIC_STOPLIST');
end;
/
```

medio

```
CREATE TABLE medio (
```

```
  id NUMBER (1),
  nome VARCHAR (20),
  CONSTRAINT medio_pk PRIMARY KEY (id)
);
```

editorial

```
CREATE TABLE editorial (
  id NUMBER (2),
  nome VARCHAR (100) NOT NULL,
  CONSTRAINT editorial_pk PRIMARY KEY (id)
);
CREATE INDEX editorialtext ON editorial(nome)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE CTXSYS.DEFAULT_DATASTORE LEXER lexico
               WORDLIST lista_palabras STOPLIST lista_stop');
```

area_tematica

```
CREATE TABLE area_tematica (
  id NUMBER (1),
  subid NUMBER (2),
  nome VARCHAR (100) NOT NULL,
  CONSTRAINT area_tematica_pk PRIMARY KEY (id, subid)
);
```

autor

```
CREATE TABLE autor (
  id INT,
  nome VARCHAR (100),
  CONSTRAINT autor_pk PRIMARY KEY (id)
);
CREATE INDEX autortext ON autor(nome)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE CTXSYS.DEFAULT_DATASTORE LEXER lexico
               WORDLIST lista_palabras STOPLIST lista_stop');
```

agrupacion

```
CREATE TABLE agrupacion (
  id INT,
  tipo NUMBER(2) NOT NULL,
  titulo VARCHAR (300),
  ano_publicacion INT NOT NULL,
  lustro NUMBER(2) NOT NULL,
  soporte VARCHAR(15) NOT NULL,
  id_editorial NUMBER (2) NOT NULL,
  id_medio NUMBER (1) NOT NULL,
  num_pal INT NOT NULL,
  comentarios VARCHAR (800),
  CONSTRAINT agrupacion_pk PRIMARY KEY (id),
  CONSTRAINT foreign_editorial FOREIGN KEY (id_editorial)
    REFERENCES editorial (id)
```

```
    ON DELETE CASCADE,
  CONSTRAINT foreign_medio FOREIGN KEY (id_medio)
    REFERENCES medio (id)
    ON DELETE CASCADE
);

CREATE INDEX agrupacion_id_editorial
  ON agrupacion (id_editorial);

CREATE INDEX agrupacion_id_medio
  ON agrupacion (id_medio);

CREATE INDEX agrupacionltext ON agrupacion(titulo)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE CTXSYS.DEFAULT_DATASTORE LEXER lexico
               WORDLIST lista_palabras STOPLIST lista_stop');
```

documento

```
CREATE TABLE documento (
  id_agrupacion INT,
  id INT,
  tipo NUMBER (2) NOT NULL,
  titulo VARCHAR (300),
  seccion VARCHAR (20),
  id_area_tematica NUMBER(1),
  subid_area_tematica NUMBER(2),
  num_pal INT NOT NULL,
  comentarios VARCHAR (800),
  CONSTRAINT documento_pk PRIMARY KEY (id),
  CONSTRAINT foreign_agrupacion FOREIGN KEY (id_agrupacion)
    REFERENCES agrupacion(id)
    ON DELETE CASCADE,
  CONSTRAINT oracion_foreign_area_tematica FOREIGN KEY
    (id_area_tematica, subid_area_tematica)
    REFERENCES area_tematica (id, subid)
    ON DELETE CASCADE
);

CREATE INDEX documento_id_agrupacion
  ON documento (id_agrupacion);

CREATE INDEX documento_area_tematica
  ON documento (id_area_tematica, subid_area_tematica);

CREATE INDEX documentotext ON documento(titulo)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE CTXSYS.DEFAULT_DATASTORE LEXER lexico
               WORDLIST lista_palabras STOPLIST lista_stop');
```

oracion

```
CREATE TABLE oracion (
  id_agrupacion INT,
  id_documento INT,
```

```
  id INT,
  id_nota VARCHAR (5),
  id_contexto INT,
  tipo NUMBER (2) NOT NULL,
  interlocutor NUMBER (1),
  nome_interlocutor VARCHAR (80),
  oracion CLOB NOT NULL,
  oracionsenacentos CHAR(1),
  CONSTRAINT oracion_pk PRIMARY KEY (id),
  CONSTRAINT oracion_foreign_agrupacion FOREIGN KEY (id_agrupacion)
    REFERENCES agrupacion (id)
    ON DELETE CASCADE,
  CONSTRAINT oracion_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE
);

CREATE INDEX oracion_id_agrupacion
  ON oracion (id_agrupacion);

CREATE INDEX documento_id_documento
  ON oracion (id_documento);


CREATE INDEX textindex ON oracion(oracion)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE CTXSYS.DEFAULT_DATASTORE LEXER lexico
               WORDLIST lista_palabras STOPLIST lista_stop');
```

palabra

```
CREATE TABLE palabra (
  id INT,
  palabra VARCHAR(150),
  CONSTRAINT palabra_pk PRIMARY KEY (id)
);

CREATE INDEX textindexpalabra ON palabra(palabra)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('DATASTORE CTXSYS.DEFAULT_DATASTORE LEXER lexico
               WORDLIST lista_palabras STOPLIST lista_stop');
```

crea

```
CREATE TABLE crea (
  id_autor INT,
  id_documento INT,
  CONSTRAINT crea_pk PRIMARY KEY (id_autor, id_documento),
  CONSTRAINT crea_foreign_autor FOREIGN KEY (id_autor)
    REFERENCES autor (id)
    ON DELETE CASCADE,
  CONSTRAINT crea_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE
);
```

catalogado

```
CREATE TABLE catalogado (
  id_documento INT,
  id_area_tematica NUMBER (1),
  subid_area_tematica NUMBER (2),
  posicion NUMBER (1) NOT NULL,
  CONSTRAINT catalogado_pk PRIMARY KEY
    (id_documento, id_area_tematica, subid_area_tematica),
  CONSTRAINT catalogado_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE,
  CONSTRAINT catalogado_foreign_area_t FOREIGN KEY
    (id_area_tematica, subid_area_tematica)
    REFERENCES area_tematica (id,subid)
    ON DELETE CASCADE
);
```

ortok

```
CREATE TABLE ortok (
  id_oracion INT,
  id_palabra INT,
  posicion INT,
  CONSTRAINT ortok_foreign_oracion FOREIGN KEY (id_oracion)
    REFERENCES oracion (id)
    ON DELETE CASCADE,
  CONSTRAINT ortok_foreign_palabra FOREIGN KEY (id_palabra)
    REFERENCES palabra (id)
    ON DELETE CASCADE
);

CREATE INDEX ortok_idor
  ON ortok (id_oracion);

CREATE INDEX ortok_idto
  ON ortok (id_palabra);

CREATE UNIQUE INDEX ortok_posicion
  ON ortok (posicion);

CREATE INDEX ortok_posicion_func
  ON ortok (posicion+1);
```

D.1.2 Consultas

Para calcular o tempo de resposta dunha consulta fixemos o seguinte:

1. Cando as consultas devolven uns poucos valores numéricos poñemos `set timing on` no SQL*Plus e copiamos a consulta no terminal. Collemos o tempo que se amosa unha vez resolta a consulta.

2. Cando se devolven moitas tuplas e, por tanto, o tempo de visualización no terminal é relevante:

(a) Metemos o código da consulta nun *script*: proba.sql

(b) A seguir tecleamos nun editor de texto:

```
set termout off
timing start contador
start proba.sql
timing stop
```

(c) Finalmente, copiamos esas catro ordes en bloque no terminal SQL*Plus e utilizamos o tempo que se amosa cando finaliza a consulta.

Batería 1

Q1a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion
WHERE  oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'desenrolo')>0;
```

Q1b

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion
WHERE  oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'des%')>0;
```

Q1c

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion
WHERE  oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'%ado')>0;
```

Batería 2

Q2a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND CONTAINS (palabra1.palabra,'desenrolo')>0;
```

Q2b

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1
```

```
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND CONTAINS (palabra1.palabra,'des%')>0;
```

Q2c

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND CONTAINS (palabra1.palabra,'%ado')>0;
```

Batería 3

Q3a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion
WHERE  oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q3b

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1,
       ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND ortok2.posicion=ortok1.posicion+1
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Batería 4

Q4aa

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, catalogado, area_tematica
WHERE  oracion.id_documento = documento.id
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q4ab

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   medio, agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q4ac

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   autor, crea, documento, oracion
WHERE  crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND CONTAINS (autor.nome,'Rivas')>0
       AND oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q4ba

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   catalogado, area_tematica, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND ortok2.posicion=ortok1.posicion+1
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q4bb

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
```

```
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND ortok2.posicion=ortok1.posicion+1
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q4bc

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   autor, crea, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND CONTAINS (autor.nome,'Rivas')>0
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND ortok2.posicion=ortok1.posicion+1
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q4ca

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   catalogado, area_tematica, medio, autor, crea,
       agrupacion, documento, oracion
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND CONTAINS (autor.nome,'Rivas')>0
       AND oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q4cb

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   catalogado, area_tematica, medio, autor, crea, agrupacion,
       documento, oracion, ortok ortok1, palabra palabra1, ortok ortok2,
       palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
```

```
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND CONTAINS (autor.nome,'Rivas')>0
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND ortok2.posicion=ortok1.posicion+1
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Batería 5

Q5a

```
SELECT medio.id AS medio,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   medio, agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND medio.id = agrupacion.id_medio
       AND CONTAINS (oracion.oracion,'sen embargo')>0
GROUP BY medio.id;

SELECT area_tematica.id AS at,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, catalogado, area_tematica
WHERE  oracion.id_documento = documento.id
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'sen embargo')>0
GROUP BY area_tematica.id;

SELECT agrupacion.lustro AS lustro,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'sen embargo')>0
GROUP BY lustro;
```

Q5a

```
SELECT medio.id AS medio,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
```

```
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND medio.id = agrupacion.id_medio
       AND CONTAINS (oracion.oracion,'sen embargo')>0
GROUP BY medio.id;

SELECT area_tematica.id AS at,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, catalogado, area_tematica,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND CONTAINS (oracion.oracion,'sen embargo')>0
GROUP BY area_tematica.id;

SELECT agrupacion.lustro AS lustro,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND CONTAINS (palabra1.palabra,'sen')>0
       AND CONTAINS (palabra2.palabra,'embargo')>0
       AND CONTAINS (oracion.oracion,'sen embargo')>0
GROUP BY lustro;
```

Batería 6

Q6a

```
SELECT oracion
FROM   oracion
WHERE  CONTAINS (oracion.oracion,'sen embargo')>0;
```

Q6b

```
SELECT *
FROM   catalogado, area_tematica, medio, autor, crea,
       agrupacion, documento, oracion
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND oracion.id_documento = documento.id
       AND CONTAINS (oracion.oracion,'sen embargo')>0;
```

Batería 7

Q7a

```
SELECT oracion
FROM   oracion
WHERE  CONTAINS (oracion.oracion,'desenvolvemento')>0;
```

Q7b

```
SELECT oracion.id, agrupacion.ano_publicacion, medio.nome, area_tematica.nome
FROM   area_tematica, medio, agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND oracion.id_documento = documento.id
       AND documento.id_area_tematica = area_tematica.id
       AND documento.subid_area_tematica = area_tematica.subid
       AND CONTAINS (oracion.oracion,'desenvolvemento')>0
ORDER BY agrupacion.ano_publicacion, medio.nome, area_tematica.nome;
```

Batería 8

Q8a

```
SELECT COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS id,
              documento.num_pal as num_pal
       FROM   documento
     );
```

Q8b

```
SELECT COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS id,
              documento.num_pal as num_pal
       FROM   medio, agrupacion, documento,
```

```
              catalogado, area_tematica
       WHERE  documento.id_agrupacion = agrupacion.id
              AND medio.id = agrupacion.id_medio
              AND documento.id = catalogado.id_documento
              AND area_tematica.id = catalogado.id_area_tematica
              AND area_tematica.subid = catalogado.subid_area_tematica
              AND medio.id = 1
              AND area_tematica.id = 1
   );
```

Q8c

```
SELECT medio.id as medio,
       COUNT(DISTINCT(documento.id)) as num_documentos,
       SUM(documento.num_pal) as num_palabras
FROM   medio, agrupacion, documento
WHERE  documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
GROUP BY medio.id;

SELECT area_tematica.id,
       COUNT(DISTINCT (documento.id)) AS num_documentos,
       SUM(documento.num_pal) AS num_palabras
FROM   documento, catalogado, area_tematica
WHERE  documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
GROUP BY area_tematica.id;

SELECT lustro,
       COUNT(DISTINCT(documento.id)) as num_documentos,
       SUM(documento.num_pal) as num_palabras
FROM   agrupacion, documento
WHERE  documento.id_agrupacion = agrupacion.id
GROUP BY lustro;
```

Q8d

```
SELECT medio,
       COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) as id,
              medio.id as medio,
              documento.num_pal as num_pal
       FROM   medio, agrupacion, documento,
              catalogado, area_tematica
       WHERE  documento.id_agrupacion = agrupacion.id
              AND area_tematica.id = catalogado.id_area_tematica
              AND area_tematica.subid = catalogado.subid_area_tematica
              AND catalogado.id_documento = documento.id
              AND area_tematica.id = 1
              AND medio.id = agrupacion.id_medio
      )
GROUP BY medio;
```

```
SELECT area_tematica,
       COUNT(documento) AS num_documentos,
       SUM(num_pal) AS num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS documento,
              area_tematica.id AS area_tematica,
              documento.num_pal AS num_pal
       FROM   documento, catalogado, area_tematica
       WHERE  documento.id = catalogado.id_documento
              AND area_tematica.id = catalogado.id_area_tematica
              AND area_tematica.subid = catalogado.subid_area_tematica
              AND area_tematica.id = 1
      )
GROUP BY area_tematica;

SELECT lustro,
       COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) as id,
              agrupacion.lustro as lustro,
              documento.num_pal as num_pal
       FROM   agrupacion, documento, catalogado, area_tematica
       WHERE  documento.id_agrupacion = agrupacion.id
              AND area_tematica.id = catalogado.id_area_tematica
              AND area_tematica.subid = catalogado.subid_area_tematica
              AND catalogado.id_documento = documento.id
              AND area_tematica.id = 1
     )
GROUP BY lustro;
```

D.2 PostgreSQL

D.2.1 Definición das táboas e índices

A continuación describimos a creación das táboas e índices de PostgreSQL:

Configuración do índice textual

```
CREATE TEXT SEARCH CONFIGURATION public.galician (PARSER = pg_catalog.default);

CREATE TEXT SEARCH DICTIONARY public.galician_dict (
   TEMPLATE = pg_catalog.simple,
   STOPWORDS = galician
);

ALTER TEXT SEARCH CONFIGURATION galician
   ALTER MAPPING FOR asciiword, asciihword, hword_asciipart,
                     word, hword, hword_part
    WITH galician_dict;
```

medio

```
CREATE TABLE medio (
```

```
  id INT,
  nome VARCHAR (20),
  CONSTRAINT medio_pk PRIMARY KEY (id)
);
```

editorial

```
CREATE TABLE editorial (
  id INT,
  nome VARCHAR (100) NOT NULL,
  CONSTRAINT editorial_pk PRIMARY KEY (id)
);
CREATE INDEX editorialtext ON editorial(nome);
```

area_tematica

```
CREATE TABLE area_tematica (
  id INT,
  subid INT,
  nome VARCHAR (100) NOT NULL,
  CONSTRAINT area_tematica_pk PRIMARY KEY (id, subid)
);
```

autor

```
CREATE TABLE autor (
  id INT,
  nome VARCHAR (100),
  CONSTRAINT autor_pk PRIMARY KEY (id)
);

CREATE INDEX textindexautor ON autor
USING gin(to_tsvector('galician',nome));
```

agrupacion

```
CREATE TABLE agrupacion (
  id INT,
  tipo INT NOT NULL,
  titulo VARCHAR (300),
  ano_publicacion INT NOT NULL,
  lustro INT NOT NULL,
  soporte VARCHAR(15) NOT NULL,
  id_editorial INT NOT NULL,
  id_medio INT NOT NULL,
  num_pal INT NOT NULL,
  comentarios VARCHAR (800),
  CONSTRAINT agrupacion_pk PRIMARY KEY (id),
  CONSTRAINT foreign_editorial FOREIGN KEY (id_editorial)
    REFERENCES editorial (id)
    ON DELETE CASCADE,
  CONSTRAINT foreign_medio FOREIGN KEY (id_medio)
    REFERENCES medio (id)
```

```
    ON DELETE CASCADE
);

CREATE INDEX agrupacion_id_editorial
on agrupacion (id_editorial);

CREATE INDEX agrupacion_id_medio
on agrupacion (id_medio);

CREATE INDEX textindexagrupacion ON agrupacion
USING gin(to_tsvector('galician',titulo));
```

documento

```
CREATE TABLE documento (
  id_agrupacion INT,
  id INT,
  tipo INT NOT NULL,
  titulo VARCHAR (300),
  seccion VARCHAR (20),
  id_area_tematica INT,
  subid_area_tematica INT,
  num_pal INT NOT NULL,
  comentarios VARCHAR (800),
  CONSTRAINT documento_pk PRIMARY KEY (id),
  CONSTRAINT foreign_agrupacion FOREIGN KEY (id_agrupacion)
    REFERENCES agrupacion(id)
    ON DELETE CASCADE,
  CONSTRAINT oracion_foreign_area_tematica FOREIGN KEY
    (id_area_tematica, subid_area_tematica)
    REFERENCES area_tematica (id, subid)
    ON DELETE CASCADE
);

CREATE INDEX documento_id_agrupacion
on documento (id_agrupacion);

CREATE INDEX documento_area_tematica
on documento (id_area_tematica, subid_area_tematica);

CREATE INDEX textindexdocumento ON documento
USING gin(to_tsvector('galician',titulo));
```

oracion

```
CREATE TABLE oracion (
  id_agrupacion INT,
  id_documento INT,
  id INT,
  id_nota VARCHAR (5),
  id_contexto INT,
  tipo INT NOT NULL,
  interlocutor INT,
  nome_interlocutor VARCHAR (80),
  oracion VARCHAR NOT NULL,
```

```
  oracionsenacentos VARCHAR,
  CONSTRAINT oracion_pk PRIMARY KEY (id),
  CONSTRAINT oracion_foreign_agrupacion FOREIGN KEY (id_agrupacion)
    REFERENCES agrupacion (id)
    ON DELETE CASCADE,
  CONSTRAINT oracion_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE
);
CREATE INDEX oracion_id_agrupacion
on oracion (id_agrupacion);

CREATE INDEX documento_id_documento
on oracion (id_documento);

CREATE INDEX textindex ON oracion
USING gin(to_tsvector('galician',oracion));
```

autor

```
CREATE TABLE palabra (
  id INT,
  palabra VARCHAR(150),
  CONSTRAINT palabra_pk PRIMARY KEY (id)
);

CREATE INDEX textindexpalabra ON palabra
USING gin(to_tsvector('galician',palabra));
```

crea

```
CREATE TABLE crea (
  id_autor INT,
  id_documento INT,
  CONSTRAINT crea_pk PRIMARY KEY (id_autor, id_documento),
  CONSTRAINT crea_foreign_autor FOREIGN KEY (id_autor)
    REFERENCES autor (id)
    ON DELETE CASCADE,
  CONSTRAINT crea_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE
);
```

catalogado

```
CREATE TABLE catalogado (
  id_documento INT,
  id_area_tematica INT,
  subid_area_tematica INT,
  posicion INT NOT NULL,
  CONSTRAINT catalogado_pk PRIMARY KEY
    (id_documento, id_area_tematica, subid_area_tematica),
  CONSTRAINT catalogado_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE,
```

```
  CONSTRAINT catalogado_foreign_area_t FOREIGN KEY
    (id_area_tematica, subid_area_tematica)
    REFERENCES area_tematica (id,subid)
    ON DELETE CASCADE
);
```

autor

```
CREATE TABLE ortok (
  id_oracion INT,
  id_palabra INT,
  posicion INT,
  CONSTRAINT ortok_foreign_oracion FOREIGN KEY (id_oracion)
    REFERENCES oracion (id)
    ON DELETE CASCADE,
  CONSTRAINT ortok_foreign_palabra FOREIGN KEY (id_palabra)
    REFERENCES palabra (id)
    ON DELETE CASCADE
);
CREATE INDEX ortok_idor
ON ortok (id_oracion);
CREATE INDEX ortok_idto
ON ortok (id_palabra);
CREATE UNIQUE INDEX ortok_posicion
ON ortok (posicion);
CREATE INDEX ortok_posicion_func
ON ortok ((posicion+1));
```

D.2.2 Consultas

Para calcular o tempo de resposta dunha consulta fixemos o seguinte:

1. Cando as consultas devolven uns poucos valores numéricos poñemos `\timing` no terminal psql e copiamos a consulta nel. Empregamos o valor que se amosa unha vez resolta a busca.
2. Cando se devolven moitas tuplas e, polo tanto, o tempo de visualización é relevante: Poñemos `\timing` e `\o resultado.txt` no terminal psql e copiamos nel a consulta. Deste xeito os resultados métense no arquivo resultado.txt e sae no terminal o tempo de resolución que utilizamos.

Batería 1

Q1a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM documento, oracion
WHERE oracion.id_documento = documento.id
      AND to_tsvector('galician',oracion.oracion) @@
          to_tsquery('galician', 'desenrolo');
```

Q1b

Non se pode facer.

Q1c

Non se pode facer.

Batería 2

Q2a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM documento, oracion, ortok ortok1, palabra palabra1
WHERE oracion.id_documento = documento.id
      AND ortok1.id_oracion=oracion.id
      AND ortok1.id_palabra=palabra1.id
      AND to_tsvector('galician',palabra1.palabra) @@
          to_tsquery('galician', 'desenrolo');
```

Q2b

Non se pode facer.

Q2c

Non se pode facer.

Batería 3

Q3a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, ortok ortok1, palabra palabra1,
       ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q3b

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1,
       ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
```

```
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Batería 4

Q4aa

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   area_tematica, catalogado, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND area_tematica.id = catalogado.id_area_tematica
       AND documento.id = catalogado.id_documento
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4ab

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4ac

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
```

```
FROM   autor, crea, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND crea.id_documento = autor.id
       AND to_tsvector('galician',autor.nome) @@ to_tsquery('galician', 'Rivas')
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4ba

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   catalogado, area_tematica, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4bb

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4bc

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   autor, crea, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND to_tsvector('galician',autor.nome) @@ to_tsquery('galician', 'Rivas')
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4ca

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   catalogado, area_tematica, medio, autor, crea, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND to_tsvector('galician',autor.nome) @@ to_tsquery('galician', 'Rivas')
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4cb

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   catalogado, area_tematica, medio, autor, crea, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
```

```
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND to_tsvector('galician',autor.nome) @@ to_tsquery('galician', 'Rivas')
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND ortok2.posicion=ortok1.posicion+1;
```

Batería 5

Q5a

```
SELECT medio.id AS medio,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND medio.id = agrupacion.id_medio
GROUP BY medio.id;

SELECT area_tematica.id AS at,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, catalogado, area_tematica,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
```

```
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
GROUP BY area_tematica.id;

SELECT agrupacion.lustro AS lustro,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
GROUP BY lustro;
```

Q5b

```
SELECT medio.id AS medio,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
       AND medio.id = agrupacion.id_medio
GROUP BY medio.id;

SELECT area_tematica.id AS at,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, catalogado, area_tematica,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
```

```
            to_tsquery('galician', 'embargo')
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
GROUP BY area_tematica.id;

SELECT agrupacion.lustro AS lustro,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo')
GROUP BY lustro;
```

Batería 6

Q6a

```
SELECT oracion
FROM   oracion, ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo');
```

Q6b

```
SELECT *
FROM   catalogado, area_tematica, medio, autor, crea, agrupacion, documento,
       oracion, ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
```

```
       AND to_tsvector('galician',palabra1.palabra) @@
           to_tsquery('galician', 'sen')
       AND to_tsvector('galician',palabra2.palabra) @@
           to_tsquery('galician', 'embargo');
```

Batería 7

Q7a

```
SELECT oracion
FROM   oracion
WHERE  to_tsvector('galician',oracion.oracion) @@
       to_tsquery('galician', 'desenvolvemento');
```

Q7b

```
SELECT oracion.id, agrupacion.ano_publicacion, medio.nome, area_tematica.nome
FROM   area_tematica, medio, agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND oracion.id_documento = documento.id
       AND documento.id_area_tematica = area_tematica.id
       AND documento.subid_area_tematica = area_tematica.subid
       AND to_tsvector('galician',oracion.oracion) @@
           to_tsquery('galician', 'desenvolvemento')
ORDER BY agrupacion.ano_publicacion, medio.nome, area_tematica.nome;
```

Batería 8

Q8a

```
SELECT COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS id,
              documento.num_pal as num_pal
       FROM   documento
     ) AS documentos;
```

Q8b

```
SELECT COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS id,
             documento.num_pal as num_pal
       FROM medio, agrupacion, documento, catalogado, area_tematica
       WHERE documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND medio.id = 1
       AND area_tematica.id = 1
    ) AS documentos;
```

Q8c

```
SELECT medio.id as medio,
       COUNT(DISTINCT(documento.id)) as num_documentos,
       SUM(documento.num_pal) as num_palabras
FROM   medio, agrupacion, documento
WHERE  documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
GROUP BY medio.id;

SELECT area_tematica.id,
       COUNT(DISTINCT (documento.id)) AS num_documentos,
       SUM(documento.num_pal) AS num_palabras
FROM   documento, catalogado, area_tematica
WHERE  documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
GROUP BY area_tematica.id;

SELECT lustro,
       COUNT(DISTINCT(documento.id)) as num_documentos,
       SUM(documento.num_pal) as num_palabras
FROM   agrupacion, documento
WHERE  documento.id_agrupacion = agrupacion.id
GROUP BY lustro;
```

Q8d

```
SELECT medio,
       COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) as id,
              medio.id as medio,
              documento.num_pal as num_pal
       FROM medio, agrupacion, documento, catalogado, area_tematica
       WHERE documento.id_agrupacion = agrupacion.id
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND medio.id = agrupacion.id_medio
     ) AS documentos
GROUP BY medio;

SELECT area_tematica,
       COUNT(documento) AS num_documentos,
       SUM(num_pal) AS num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS documento,
              area_tematica.id AS area_tematica,
              documento.num_pal AS num_pal
       FROM documento, catalogado, area_tematica
       WHERE documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
```

```
       AND area_tematica.id = 1
      ) AS documentos
GROUP BY area_tematica;

SELECT lustro,
       COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) as id,
              agrupacion.lustro as lustro,
              documento.num_pal as num_pal
       FROM agrupacion, documento, catalogado, area_tematica
       WHERE documento.id_agrupacion = agrupacion.id
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
     ) AS documentos
GROUP BY lustro;
```

D.3 MySQL

D.3.1 Definición das táboas e índices

medio

```
CREATE TABLE medio (
  id INT,
  nome VARCHAR (20),
  CONSTRAINT medio_pk PRIMARY KEY (id)
);
```

editorial

```
CREATE TABLE editorial (
  id INT,
  nome VARCHAR (100) NOT NULL,
  CONSTRAINT editorial_pk PRIMARY KEY (id)
);
ALTER TABLE editorial ADD FULLTEXT editorial_text(nome);
```

area_tematica

```
CREATE TABLE area_tematica (
  id INT,
  subid INT,
  nome VARCHAR (100) NOT NULL,
  CONSTRAINT area_tematica_pk PRIMARY KEY (id, subid)
);
```

autor

```
CREATE TABLE autor (
```

```
  id INT NOT NULL AUTO_INCREMENT,
  nome VARCHAR (100),
  CONSTRAINT autor_pk PRIMARY KEY (id)
);
ALTER TABLE autor ADD FULLTEXT autor_text(nome);
```

agrupacion

```
CREATE TABLE agrupacion (
  id INT NOT NULL AUTO_INCREMENT,
  tipo INT NOT NULL,
  titulo VARCHAR (300),
  ano_publicacion INT NOT NULL,
  lustro INT NOT NULL,
  soporte VARCHAR(15) NOT NULL,
  id_editorial INT NOT NULL,
  id_medio INT NOT NULL,
  num_pal INT NOT NULL,
  comentarios VARCHAR (800),
  CONSTRAINT agrupacion_pk PRIMARY KEY (id),
  CONSTRAINT foreign_editorial FOREIGN KEY (id_editorial)
    REFERENCES editorial (id)
    ON DELETE CASCADE,
  CONSTRAINT foreign_medio FOREIGN KEY (id_medio)
    REFERENCES medio (id)
    ON DELETE CASCADE
);

CREATE INDEX agrupacion_id_editorial
on agrupacion (id_editorial);

CREATE INDEX agrupacion_id_medio
on agrupacion (id_medio);

ALTER TABLE agrupacion ADD FULLTEXT agrupacion_text(titulo);
```

documento

```
CREATE TABLE documento (
  id_agrupacion INT NOT NULL,
  id INT NOT NULL AUTO_INCREMENT,
  tipo INT NOT NULL,
  titulo VARCHAR (300),
  seccion VARCHAR (20),
  id_area_tematica INT,
  subid_area_tematica INT,
  num_pal INT NOT NULL,
  comentarios VARCHAR (800),
  CONSTRAINT documento_pk PRIMARY KEY (id),
  CONSTRAINT foreign_agrupacion FOREIGN KEY (id_agrupacion)
    REFERENCES agrupacion(id)
    ON DELETE CASCADE,
  CONSTRAINT oracion_foreign_area_tematica FOREIGN KEY
    (id_area_tematica, subid_area_tematica)
```

```
    REFERENCES area_tematica (id, subid)
    ON DELETE CASCADE
);
CREATE INDEX documento_id_agrupacion
on documento (id_agrupacion);

CREATE INDEX documento_area_tematica
on documento (id_area_tematica, subid_area_tematica);

ALTER TABLE documento ADD FULLTEXT documento_text(titulo);
```

oracion

```
CREATE TABLE oracion (
  id_agrupacion INT,
  id_documento INT,
  id INT NOT NULL AUTO_INCREMENT,
  id_nota VARCHAR (5),
  id_contexto INT,
  tipo INT NOT NULL,
  interlocutor INT,
  nome_interlocutor VARCHAR (80),
  oracion TEXT NOT NULL,
  oracionsenacentos TEXT,
  CONSTRAINT oracion_pk PRIMARY KEY (id),
  CONSTRAINT oracion_foreign_agrupacion FOREIGN KEY (id_agrupacion)
    REFERENCES agrupacion (id)
    ON DELETE CASCADE,
  CONSTRAINT oracion_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE
);

CREATE INDEX oracion_id_agrupacion
on oracion (id_agrupacion);

CREATE INDEX documento_id_documento
on oracion (id_documento);

ALTER TABLE oracion ADD FULLTEXT oracion_text(oracion);
```

palabra

```
CREATE TABLE palabra (
  id INT,
  palabra VARCHAR(150),
  CONSTRAINT palabra_pk PRIMARY KEY (id)
);

ALTER TABLE palabra ADD FULLTEXT palabra_text(palabra);
```

ortok

```
CREATE TABLE ortok (
  id_oracion INT,
```

```
  id_palabra INT,
  posicion INT,
  CONSTRAINT ortok_foreign_oracion FOREIGN KEY (id_oracion)
    REFERENCES oracion (id)
    ON DELETE CASCADE,
  CONSTRAINT ortok_foreign_palabra FOREIGN KEY (id_palabra)
    REFERENCES palabra (id)
    ON DELETE CASCADE
);

CREATE INDEX ortok_idor
ON ortok (id_oracion);

CREATE INDEX ortok_idto
ON ortok (id_palabra);

CREATE UNIQUE INDEX ortok_posicion
ON ortok (posicion);
```

crea

```
CREATE TABLE crea (
  id_autor INT,
  id_documento INT,
  CONSTRAINT crea_pk PRIMARY KEY (id_autor, id_documento),
  CONSTRAINT crea_foreign_autor FOREIGN KEY (id_autor)
    REFERENCES autor (id)
    ON DELETE CASCADE,
  CONSTRAINT crea_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE
);
```

catalogado

```
CREATE TABLE catalogado (
  id_documento INT,
  id_area_tematica INT,
  subid_area_tematica INT,
  posicion INT NOT NULL,
  CONSTRAINT catalogado_pk PRIMARY KEY
    (id_documento, id_area_tematica, subid_area_tematica),
  CONSTRAINT catalogado_foreign_documento FOREIGN KEY (id_documento)
    REFERENCES documento (id)
    ON DELETE CASCADE,
  CONSTRAINT catalogado_foreign_area_t FOREIGN KEY
    (id_area_tematica, subid_area_tematica)
    REFERENCES area_tematica (id,subid)
    ON DELETE CASCADE
);
```

D.3.2 Consultas

Para calcular o tempo de resposta dunha consulta fixemos o seguinte:

1. Cando as consultas devolven uns poucos valores numéricos collemos o tempo que amosa o terminal MySQL.

2. Cando se obteñen moitas tuplas e, polo tanto, o tempo de visualización no terminal é relevante utilizamos a cláusula `INTO` da sentencia `SELECT` para introducir o resultado nun arquivo. Tamén neste caso empregamos o tempo que nos amosa o xestor no terminal.

Batería 1

Q1a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion
WHERE  oracion.id_documento = documento.id
       AND MATCH(oracion) AGAINST ('desenrolo' IN BOOLEAN MODE);
```

Q1b

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion
WHERE  oracion.id_documento = documento.id
       AND MATCH(oracion) AGAINST ('des*' IN BOOLEAN MODE);
```

Q1c

Non se pode facer.

Batería 2

Q2a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND MATCH(palabra1.palabra) AGAINST ('desenrolo' IN BOOLEAN MODE);
```

Q2b

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND MATCH(palabra1.palabra) AGAINST ('des*' IN BOOLEAN MODE);
```

Q2c

Non se pode facer.

Batería 3

Q3a

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion
WHERE  oracion.id_documento = documento.id
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE);
```

Q3b

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   documento, oracion, ortok ortok1, palabra palabra1,
       ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
       AND ortok2.posicion=ortok1.posicion+1;
```

Batería 4

Q4aa

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, catalogado, area_tematica
WHERE  oracion.id_documento = documento.id
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND MATCH (oracion.oracion) AGAINST('"sen embargo"' IN BOOLEAN MODE);
```

Q4ab

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   medio, agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE);
```

Q4ac

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   autor, crea, documento, oracion
WHERE  crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND MATCH(autor.nome) AGAINST ('Rivas' IN BOOLEAN MODE)
       AND oracion.id_documento = documento.id
       AND MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE);
```

Q4ba

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   catalogado, area_tematica, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4bb

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4bc

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
```

```
FROM   autor, crea, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND MATCH (autor.nome) AGAINST('Rivas' IN BOOLEAN MODE)
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
       AND ortok2.posicion=ortok1.posicion+1;
```

Q4ca

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   catalogado, area_tematica, medio, autor, crea, agrupacion, documento, oracion
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND MATCH (autor.nome) AGAINST ('Rivas' IN BOOLEAN MODE)
       AND oracion.id_documento = documento.id
       AND MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE);
```

Q4cb

```
SELECT COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   catalogado, area_tematica, medio, autor, crea, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND medio.id = 2
       AND crea.id_autor = autor.id
       AND crea.id_documento = documento.id
       AND MATCH (autor.nome) AGAINST('Rivas' IN BOOLEAN MODE)>0
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
```

```
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
       AND ortok2.posicion=ortok1.posicion+1;
```

Batería 5

Q5a

```
SELECT medio.id AS medio,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   medio, agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND medio.id = agrupacion.id_medio
       AND MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
GROUP BY medio.id;

SELECT area_tematica.id AS at,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, catalogado, area_tematica
WHERE  oracion.id_documento = documento.id
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
GROUP BY area_tematica.id;

SELECT agrupacion.lustro AS lustro,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
GROUP BY lustro;
```

Q5b

```
SELECT medio.id AS medio,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(ortok1.posicion)) AS num_palabras
FROM   medio, agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
```

```
       AND medio.id = agrupacion.id_medio
GROUP BY medio.id;

SELECT area_tematica.id AS at,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   documento, oracion, catalogado, area_tematica,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
GROUP BY area_tematica.id;

SELECT agrupacion.lustro AS lustro,
       COUNT(DISTINCT(documento.id)) AS num_documentos,
       COUNT(DISTINCT(oracion.id)) AS num_oracions
FROM   agrupacion, documento, oracion,
       ortok ortok1, palabra palabra1, ortok ortok2, palabra palabra2
WHERE  documento.id_agrupacion = agrupacion.id
       AND oracion.id_documento = documento.id
       AND ortok1.id_oracion=oracion.id
       AND ortok1.id_palabra=palabra1.id
       AND ortok2.id_oracion=oracion.id
       AND ortok2.id_palabra=palabra2.id
       AND MATCH(palabra1.palabra) AGAINST ('sen' IN BOOLEAN MODE)
       AND MATCH(palabra2.palabra) AGAINST ('embargo' IN BOOLEAN MODE)
       AND MATCH(oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE)
GROUP BY lustro;
```

Batería 6

Q6a

```
SELECT oracion INTO OUTFILE 'log.txt'
FROM   oracion
WHERE  MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE);
```

Q6b

```
SELECT * INTO OUTFILE 'log.txt'
FROM   catalogado, area_tematica, medio, autor, crea, agrupacion, documento, oracion
WHERE  area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND crea.id_autor = autor.id
```

```
       AND crea.id_documento = documento.id
       AND oracion.id_documento = documento.id
       AND MATCH (oracion.oracion) AGAINST ('"sen embargo"' IN BOOLEAN MODE);
```

Batería 7

Q7a

```
SELECT oracion INTO OUTFILE 'log.txt'
FROM   oracion
WHERE  MATCH (oracion.oracion) AGAINST ('desenvolvemento' IN BOOLEAN MODE);
```

Q7b

```
SELECT oracion.id, agrupacion.ano_publicacion, medio.nome, area_tematica.nome
       INTO OUTFILE 'log.txt'
FROM   area_tematica, medio, agrupacion, documento, oracion
WHERE  documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND oracion.id_documento = documento.id
       AND documento.id_area_tematica = area_tematica.id
       AND documento.subid_area_tematica = area_tematica.subid
       AND MATCH (oracion.oracion) AGAINST ('desenvolvemento' IN BOOLEAN MODE)
ORDER BY agrupacion.ano_publicacion, medio.nome, area_tematica.nome;
```

Batería 8

Q8a

```
SELECT COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS id,
              documento.num_pal as num_pal
       FROM   documento
     ) AS documentos;
```

Q8b

```
SELECT COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS id,
             documento.num_pal as num_pal
       FROM medio, agrupacion, documento, catalogado, area_tematica
       WHERE documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
       AND documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND medio.id = 1
       AND area_tematica.id = 1
    ) AS documentos;
```

Q8c

```
SELECT medio.id as medio,
       COUNT(DISTINCT(documento.id)) as num_documentos,
       SUM(documento.num_pal) as num_palabras
FROM   medio, agrupacion, documento
WHERE  documento.id_agrupacion = agrupacion.id
       AND medio.id = agrupacion.id_medio
GROUP BY medio.id;

SELECT area_tematica.id,
       COUNT(DISTINCT (documento.id)) AS num_documentos,
       SUM(documento.num_pal) AS num_palabras
FROM   documento, catalogado, area_tematica
WHERE  documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
GROUP BY area_tematica.id;

SELECT lustro,
       COUNT(DISTINCT(documento.id)) as num_documentos,
       SUM(documento.num_pal) as num_palabras
FROM   agrupacion, documento
WHERE  documento.id_agrupacion = agrupacion.id
GROUP BY lustro;
```

Q8d

```
SELECT medio,
       COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) as id,
              medio.id as medio,
              documento.num_pal as num_pal
       FROM medio, agrupacion, documento, catalogado, area_tematica
       WHERE documento.id_agrupacion = agrupacion.id
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
       AND medio.id = agrupacion.id_medio
     ) AS documentos
GROUP BY medio;

SELECT area_tematica,
       COUNT(documento) AS num_documentos,
       SUM(num_pal) AS num_palabras
FROM (
       SELECT DISTINCT (documento.id) AS documento,
              area_tematica.id AS area_tematica,
              documento.num_pal AS num_pal
       FROM documento, catalogado, area_tematica
       WHERE documento.id = catalogado.id_documento
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
```

```
       AND area_tematica.id = 1
      ) AS documentos
GROUP BY area_tematica;

SELECT lustro,
       COUNT(id) as num_documentos,
       SUM(num_pal) as num_palabras
FROM (
       SELECT DISTINCT (documento.id) as id,
              agrupacion.lustro as lustro,
              documento.num_pal as num_pal,
              area_tematica.id as atid
       FROM agrupacion, documento, catalogado, area_tematica
       WHERE documento.id_agrupacion = agrupacion.id
       AND area_tematica.id = catalogado.id_area_tematica
       AND area_tematica.subid = catalogado.subid_area_tematica
       AND catalogado.id_documento = documento.id
       AND area_tematica.id = 1
     ) AS documentos
GROUP BY lustro;
```

Apéndice E

Listaxe de siglas

API: Interface de programación de aplicacións (*Application Programming Interface*)

COLE: Grupo de compiladores e linguaxes (*Grupo de Compiladores y Lenguajes*)

CSS: Follas de estilo en cascada (*Cascade Style Sheets*)

CORGA: Corpus de referencia do galego actual

DTD: Definición de tipo de documento (*Document Type Definition*)

HTTP: Protocolo de transferencia de hipertexto (*Hypertext Transfer Protocol*)

HTML: Linguaxe de marcaxe de hipertexto (*Hypertext Markup Language*)

ISO: Organización internacional para a estandarización (*International Organization for Standardization*)

JDBC: Conectividade de base de datos Java (*Java Database Connectivity*)

JSP: Páxinas de servidor Java (*Java Server Pages*)

KWIC: Palabra no seu contexto (*Key Word in Context*)

MVC: Modelo-vista-controlador (*Model-View-Controller*)

OCP: Programa de concordancias de Oxford (*Oxford Concordance Program*)

OCR: Recoñecemento óptico de caracteres (*Optical Character Recognition*)

ODF: Formato de documento aberto (*Open Document Format*)

ODT: Formato de documento aberto para texto (*Open Document Format for Text*)

PDF: Formato de documento portable (*Portable Document Format*)

PLN: Procesamento da linguaxe natural

PL/SQL: Linguaxe procedemental/SQL (*Procedural Language/SQL*)

RI: Recuperación de información

SQL: Linguaxe de consulta estruturada (*Structured Query Language*)

TEI: Iniciativa para a codificación de textos (*Text Encoding Initiative*)

UML: Linguaxe unificada de modelado (*Unified Modeling Language*)

UTF: Formato de transformación Unicode (*Unicode Transformation Unicode*)

XCES: Estándar de codificación de corpus para XML (*Corpus Encoding Standard for XML*)

XML: Linguaxe extensible de marcaxe (*Extensible Markup Language*)

Parte VI

Bibliografía

Bibliografía

[1] Adobe Systems Incorporated, *Portable Document Format*. Dispoñible en http://www.adobe.com/es/products/acrobat/adobepdf.html [consultado o 29 de xullo de 2009].

[2] The Apache Software Foundation, *Apache Lucene*. Dispoñible en http://lucene.apache.org, 25 de xuño de 2009 [consultado o 29 de xullo de 2009].

[3] The Apache Software Foundation, *Apache Struts*. Dispoñible en http://struts.apache.org, 29 de abril de 2009 [consultado o 29 de xullo de 2009].

[4] The Apache Software Foundation, *Apache Tomcat*. Dispoñible en http://tomcat.apache.org [consultado o 29 de xullo de 2009].

[5] The Apache Software Foundation, *Apache Xindice*. Dispoñible en http://xml.apache.org/xindice, 8 de decembro de 2007 [consultado o 29 de xullo de 2009].

[6] Associaçom Galega da Língua (AGAL), "Acordo ortográfico para a lusofonia", *Agália*, outono 1990, número 23, p. 360-370.

[7] Athelstan, *Concordancer: MonoConc*. Dispoñible en http://www.athel.com/mono.html [consultado o 29 de xullo de 2009].

[8] ATKINS, B.T.S e ZAMPOLLI, A. (eds.), *Computational approaches to the lexicon*, Oxford: Oxford University Press, 1994.

[9] ATKINS, Sue, CLEAR, Jeremy e OSTLER, Nicholas, "Corpus Design Criteria". En *Literary & Linguistic Computing: Journal of the Association for Literary and Linguistic Computing*, Oxford: Oxford University Press, 1992, volume 7, número 1, p. 1-16.

[10] BAEZA-YATES, Ricardo, RIBEIRO-NETO, Berthier, *Modern Information Retrieval*, Edinburgh Gate: Addison-Wesley e ACM press, 1999.

[11] BARCALA, Fco. Mario, MOLINERO, Miguel A. e DOMÍNGUEZ, Eva, "Information Retrieval and Large Text Structured Corpora". En Roberto Moreno Díaz, Franz Pichler e Alexis Quesada Arencibia (eds.), *Computer Aided Systems Theory - EUROCAST 2005: 10th International Conference on Computer Aided Systems Theory, Las Palmas de Gran Canaria, Spain, February 7 - 11, 2005: Revised Selected Papers*, Berlin / Heidelberg: Springer, 2005, p. 91-100. Lecture Notes in Computer Science (LNCS), volume 3643.

[12] BARCALA, Fco. Mario, BLANCO, Cristina e DARRIBA Víctor Manuel, "Metodología para la construcción de córpora textuales estructurados basados en XML". En Manuel Palomar Sanz [et al.] (eds.), *Procesamiento del Lenguaje Natural*, número 26, Alicante: Universidad de Alicante, 2006, p. 9-16.

[13] BERBER, Tony, *Lingüística de Corpus*, Barueri: Editora Manole Ltda., 2004.

[14] BERGSTEN, Hans, *Java Server Pages, 3rd Edition*, Sebastopol: O'Reilly, 2004.

[15] BIBER, Douglas, CONRAD, Susan, REPPEN, Randi, *Corpus linguistics: Investigating language structure and use*, Cambridge: Cambridge University Press, 1998.

[16] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar, *El lenguaje unificado de modelado: Guía del usuario, 2ª edición*, Madrid: Addison-Wesley, 2006.

[17] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar, *El lenguaje unificado de modelado, Manual de referencia, 2ª edición*, Madrid: Addison-Wesley, 2007.

[18] BURNAGE, Gavin, DUNLOP, Dominic, "Encoding the British National Corpus", [traballo orixinal publicado en 1992]. En Geoffrey Sampson e Diana McCarthy (eds.), *Corpus Linguistics: Readings in a Widening Discipline*, London / New York: Continuum, 2005, p. 149-159.

[19] CentOS Development Team, *The Community Enterprise Operating System (CENTOS)*. Dispoñible en http://www.centos.org [consultado o 29 de xullo de 2009].

[20] *Centro Ramón Piñeiro para a Investigación en Humanidades*. Dispoñible en http://www.cirp.es [consultado o 29 de xullo de 2009].

[21] Centro Ramón Piñeiro para a Investigación en Humanidades, *Corpus de Referencia do Galego Actual (CORGA)*. Dispoñible en http://corpus.cirp.es/corga, 10 de xullo de 2009 [consultado o 29 de xullo de 2009].

[22] CHAUDRI, Akmal B., RASHID, Awais e ZICARI, Roberto (eds.), *XML Data Management, Native XML and XML-Enabled Database Systems*, Boston: Addison-Wesley, 2003.

[23] CLEAR, J., "Trawling the language: Monitor Corpora". En Snell-Hornby (ed.), *ZuriLEX Proceedings*, Tübingen: 1987, p.243-252.

[24] COAD, Peter, NICOLA, Jill *Object-oriented Programming*, Englewood Cliffs: Prentice Hall PTR, 1993.

[25] DAVIES, Mark, "Semantically-based queries with a joint BNC/WordNet database". En Roberta Facchinetti (ed.), *Corpus Linguistics: 25 Years on*, Amsterdan: Editions Rodopi, 2007, p. 149-168. Language and Computers: Studies in Practical Linguistics, número 62.

[26] DAVIES, Mark, "Un corpus anotado de 10.000.000 palabras del español histórico y moderno". En *XVII Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural: Universidad de Valladolid, 11-13 de septiembre de 2002*, España: 2002, p. 21-27. Procesamiento del Lenguaje Natural, número 29.

[27] DAVIES, Mark, "Relational n-gram databases as a basis for unlimited annotation on large corpora". En *Workshop on Shallow Processing of Large Corpora*, Lancaster: 2003.

[28] DAVIES, Mark, FERREIRA, Michael J, *O Corpus do Português*. Dispoñible en http://www.corpusdoportugues.org [consultado o 29 de xullo de 2009].

[29] dtSearch Corp., *dtSearch*. Dispoñible en http://www.dtsearch.com [consultado o 29 de xullo de 2009].

[30] Expert Advisory Group on Language Engineering Standards (EAGLES), *Corpus Encoding Standard*. Dispoñible en http://www.cs.vassar.edu/CES/, 20 de marzo de 2000 [consultado o 29 de xullo de 2009].

[31] Expert Advisory Group on Lenguage Engineering Standards (EAGLES), *Preliminary Recommendations on Text Typology: EAGLES Document EAG-TCWG-TTYP/P*, [Birmingham]: 1996.

[32] FRAKES, William B., BAEZA-YATES, Ricardo (eds.), *Information Retrieval. Data Structures & Algorithms*, Englewood Cliffs: Prentice Hall PTR, 1992.

[33] Free Software Foundation Inc., *Bash*. Dispoñible en http://www.gnu.org/software/bash, 20 de novembro de 2006 [consultado o 29 de xullo de 2009].

[34] Free Software Foundation Inc., *GNU Emacs*. Dispoñible en http://www.gnu.org/software /emacs, 12 de marzo de 2009 [consultado o 29 de xullo de 2009].

[35] GIBBON, Dafydd, "First Steps in Corpus Building for linguistics and technology". En *LREC 2004 Proceedings: First Steps for Language Documentation of Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation*, Lisbon: 2004.

[36] Free Software Foundation Inc., *GNU Operating System*, http://www.gnu.org, 29 de xullo de 2009 [consultado o 29 de xullo de 2009].

[37] Free Software Foundation Inc., *GNU wget*, http://www.gnu.org/software/wget, 7 de febreiro de 2008 [consultado o 29 de xullo de 2009].

[38] FREIXEIRO, Xosé Ramón, "As novas normas do galego; Sermos reintegracionistas con ortografía oficial". En *A Nosa Terra*, Vigo: A Nosa Terra, 2004, número 1112, p. 16.

[39] GAMMA, Erich [et al.], *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading: Addison-Wesley, 1995.

[40] Gnome project, *The XML C parser and toolkit of Gnome, libxml*. Dispoñible en http://www.xmlsoft.org [consultado o 29 de xullo de 2009].

[41] GOSLING, James [et al.] *The Java Language Specification, Third Edition: The Java Series ...from the Source*, Santa Clara: Sun Microsystems, 2005.

[42] KENNEDY, Graeme, *An Introduction to Corpus Linguistics*, New York: Addison Wesley Longman, 1998.

[43] KOWALSKI, Gerald, *Information Retrieval Systems: Theory and Implementation*, Norwell / AH Dordrecht: Kluber Academic Publishers, 1997.

[44] GRAÑA, Jorge, *Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural*, [tese de doutoramento], A Coruña: Universidade da Coruña, 2000.

[45] GRAÑA, Jorge, BARCALA, Fco. Mario, ALONSO, Miguel A., "Compilation Methods of Minimal Acyclic Finite-State Automata for Large Dictionaries". En Bruce Watson e Derick Wood (eds.), *Implementation and Application of Automata: 6th International Conference, CIAA 2001, Pretoria, South Africa, July 23-25, 2001: Revised Papers*, Berlin / Heidelberg: Springer, 2002, p. 135-148. Lecture Notes in Computer Science (LNCS), volume 2494.

[46] *Grupo de Compiladores y Lenguajes*. Dispoñible en http://www.grupocole.org [consultado o 29 de xullo de 2009].

[47] Helios Software Solutions, *TextPad*. Dispoñible en http://www.textpad.com [consultado o 29 de xullo de 2009].

[48] Hewlett-Packard Development Company, L.P. Dispoñible en http://www.hp.es [consultado o 29 de xullo de 2009].

[49] HUGHES, Kevin, UC Berkeley [et al.], *Simple Web Indexing System for Humans - Enhanced (Swish-e)*, http://swish-e.org/, 7 de febreiro de 2007 [consultado o 29 de xullo de 2009].

[50] *International Organization for Standardization*. Dispoñible en http://www.iso.org [consultado o 29 de xullo de 2009].

[51] KILGARRIF, Adam [et al.], "The Sketch Engine". En G. Willians e S. Vessier (eds.), *Proceedings of the Eleventh EURALEX International Congress: EURALEX 2004, Lorient, France, July 6-10*, Lorient: 2004, p. 105-116.

[52] KILGARRIF, Adm, RYCHLÝ, Pavel, POMIKÁLEK, Jan, *Sketch Engine*. Dispoñible en http://www.sketchengine.co.uk [consultado o 29 de xullo de 2009].

[53] The Linux Foundation. Dispoñible en http://www.linuxfoundation.org [consultado o 29 de xullo de 2009].

[54] Linux Online Inc., *Linux Online!* Dispoñible en http://www.linux.org [consultado o 29 de xullo de 2009].

[55] LÓPEZ, María Sol, "CORGA (Corpus de Referencia del Gallego Actual)". En *Actas de Hizkuntza-corpusak: Oraina eta feroa*, Borovets: 2003, p. 500-504.

[56] MCENERY, Tony, WILSON, Andrew *Corpus Linguistics, 2nd Edition*, Edinburg: Edinburg University Press, 2001.

[57] MEIER, WOLFGANG, *eXist*. Dispoñible en http://exist.sourceforge.net [consultado o 29 de xullo de 2009].

[58] NEWHAM, Cameron, ROSENBLATT, Bill, *Learning the bash Shell, Third Edition*. Sebastopol: O'Reilly, 2005.

[59] Nuance Communications, Inc., Dispoñible en http://www.nuance.com [consultado o 29 de xullo de 2009].

[60] Oracle Corporation. Dispoñible en http://www.oracle.com [consultado o 29 de xullo de 2009].

[61] Organization for the Advancement of Structured Information Standards (OASIS), *OASIS Open Document Format for Office Applications*. Dispoñible en http://www.oasis-open.org/specs [consultado o 29 de xullo de 2009].

[62] Oxford University Press, *Oxford Concordance Program (OCP)*.

[63] Oxford University Press, *WordSmith Tools*. Dispoñible en http://www.lexically.net/wordsmith [consultado o 29 de xullo de 2009].

[64] PERES, José Henrique, *Introduçom à linguística com corpora (inclui estudo contrastivo cultural luso-brasileiro)*, Santiago de Compostela: Edicións Laiovento, 2000.

[65] The Perl Fountation. Dispoñible en http://www.perlfoundation.org [consultado o 29 de xullo de 2009].

[66] PÉREZ, M. Chantal, "Explotación de los córpora textuales informatizados para la creación de bases de datos terminológicas basadas en el conocimiento". En Carlos Subirats Rüggeberg e Emilia Victoria Enríquez (eds.), *Estudios de Lingüística Española (ELiEs)*, volume 18, España: 2002.

[67] Pixware, *XMLmind XML Editor*. Dispoñible en http://www.xmlmind.com, 1 de decembro de 2008 [consultado o 29 de xullo de 2009].

[68] PostgreSQL Global Development Group, *PostgreSQL*. Dispoñible en http://www.postgresql.org [consultado o 29 de xullo de 2009].

[69] RABINOWITH, Josh, "How to index anything". En *Linux Journal*, Houston: Belltown Media, Inc., xullo, 2003.

[70] Real Academia Galega e Instituto da Lingua Galega, *Normas Ortográficas e Morfolóxicas do Idioma Galego*. Santiago de Compostela / A Coruña: Real Acedemia Galega e Instituto da Lingua Galega, 1982.

[71] Real Academia Galega e Instituto da Lingua Galega, *Normas Ortográficas e Morfolóxicas do Idioma Galego*. Santiago de Compostela / A Coruña: Real Acedemia Galega e Instituto da Lingua Galega, 1995.

[72] Real Academia Galega e Instituto da Lingua Galega, *Normas Ortográficas e Morfolóxicas do Idioma Galego*. Santiago de Compostela / A Coruña: Real Acedemia Galega e Instituto da Lingua Galega, 2003.

[73] RENOUF, Antoinette, "Corpus development 25 years on: from super-corpus to cyber-corpus". En Roberta Facchinetti (ed.), *Corpus Linguistics 25 Years on*, Amsterdan: Editions Rodopi, 2007, p. 27-50. Language and Computers: Studies in Practical Linguistics, número 62.

[74] ROCHE, Xavier, *HTTrack Website Copier*. Dispoñible en http://www.httrack.com [consultado o 29 de xullo de 2009].

[75] RYCHLÝ, Pavel, "Building Corpora from Scratch". En *European Masters in Language & Speech, Turorial 8*, University of Edinburgh, 2005. Dispoñible en liña en http://www.fi.muni.cz/ pary/emasters-building-corpora.pdf [consultado o 29 de xullo de 2009].

[76] SALTON, Gerard, "The SMART document retrieval project", En Abraham Bookstein [et al.] (eds.),*Proceedings of the 14th annual international ACM SIGIR Conference on Research and Development in Informormation Retrieval: Chicago, Illinois, USA, October 13-16, 1991 (Special Issue of the SIGIR Forum)*, Chicago: ACM Press, 1991, p. 356-358.

[77] SÁNCHEZ, A. [et al.], *Cumbre: Corpus lingüístico del español contemporáneo: fundamentos, metodología y aplicaciones*, Madrid: Sociedad General Española de Librería, S.A. (SGEL), 1995.

[78] SINCLAIR, John, "Corpus creation" [traballo orixinal publicado en 1987]. En Geoffrey Sampson e Diana McCarthy (eds.), *Corpus Linguistics: Readings in a Widening Discipline*, London / New York: Contiuum, 2005, p. 78-85.

[79] SINCLAIR, J. M. (ed.), *Looking Up: An account of the COBUILD Project in lexical computing: COLLINS COBUILD: COLLINS Birmingham University International Language Database*, London: Collins Publishers e University of Birmingham, 1987.

[80] Software AG, *Tamino*. Dispoñible en http://www.softwareag.com [consultado o 29 de xullo de 2009].

[81] Software in the Public Interest, Inc., *Debian*. Dispoñible en http://www.debian.org, 29 de xullo de 2009 [consultado o 29 de xullo de 2009].

[82] Sphinx Technologies Inc., *Sphinx*. Dispoñible en http://www.sphinxsearch.com [consultado o 29 de xullo de 2009].

[83] Sun Microsystems, Inc., *MySQL Community Edition*. Dispoñible en http://www.mysql.com [consultado o 29 de xullo de 2009].

[84] Sun Microsystems, Inc., *Model-View-Controller pattern*. Dispoñible en http://java.sun.com/blueprints/patterns/MVC.html, [2002] [consultado o 29 de xullo de 2009].

[85] Sun Microsystems, Inc., *Java Database Connectivity (JDBC)*. Dispoñible en http://java.sun.com/javase/technologies/database [consultado o 29 de xullo de 2009].

[86] Sun Microsystems, Inc., *Java Platform, Standard Edition (Java SE)*. Dispoñible en http://java.sun.com/javase [consultado o 29 de xullo de 2009].

[87] SVARTVIK, Jan, "Corpus linguistics 25+ years on". En Roberta Facchinetti (ed.), *Corpus Linguistics: 25 Years on*, Amsterdan: Editions Rodopi, 2007, p. 11-26. Chirintian Mair, Charles F. Meyer e Nelleke Oostdijk (eds.), Language and Computers: Studies in Practical Linguistics, número 62.

[88] TENNISON, Jeni, *XSLT and XPath On the Edge, Unlimited Edition*, New York / Clevealand / Indianapolis: M&T Books, 2001.

[89] Text Encoding Initiative (TEI) Consortium, *TEI's Guidelines for Electronic Text Encoding and Interchange*. Dispoñible en http://www.tei-c.org [consultado o 29 de xullo de 2009].

[90] University of Toronto, *Textual Analysis Computing Tools (TACT)*. Dispoñible en http://chass.utoronto.ca/pub/cch [consultado o 29 de xullo de 2009].

[91] The Unicode Consortium, *The Unicode Standard: Version 3.0*, Reading: Addison-Wesley, 2000.

[92] Department of Computer Science (Vassar College) e Equipe Langue et Dialogue (LORIA/CNRS), *Corpus Encoding Standard for XML*. Dispoñible en http://www.xces.org, 20 de xuño de 2008 [consultado o 29 de agosto de 2009].

[93] WALL, Larry, CHRISTIANSEN, Tom, ORWANT, Jon, *Programming Perl, Third Edition*, Sebastopol: O' Reilly, 2000.

[94] WITTEN, Ian H., MOFFAT, Alistair, BELL, Timothy C., *Managing Gigabytes. Compressing and Indexing Documents and Images, Second Edition*, San Francisco: Morgan Kaufmann Publishers, Inc., 1999.

[95] World Wide Web Consortium. Dispoñible en http://www.w3.org, 29 de xullo de 2009 [consultado o 29 de xullo de 2009].

[96] World Wide Web Consortium, *Cascade Style Sheets*. Dispoñible en http://www.w3.org/Style/CSS, 3 de xullo de 2009 [consultado o 29 de xullo de 2009].

[97] World Wide Web Consortium, *Document Type Definition (DTD)*. Dispoñible en http://www.w3.org/XML, 16 de abril de 2009 [consultado o 29 de xullo de 2009].

[98] World Wide Web Consortium, *Hypertext Markup Language*. Dispoñible en http://www.w3.org/MarkUp, 29 de xaneiro de 2009 [consultado o 29 de xullo de 2009].

[99] World Wide Web Consortium, *Extensible Markup Language (XML)*. Dispoñible en http://www.w3.org/XML, 16 de abril de 2009 [consultado o 29 de xullo de 2009].

[100] World Wide Web Consortium, *XML Schema*. Dispoñible en http://www.w3.org/XML/Schema, 24 de decembro de 2008 [consultado o 29 de xullo de 2009].

[101] World Wide Web Consortium, *XQuery 1.0: An XML Query Language*. Dispoñible en http://www.w3.org/TR/xquery, 23 de xaneiro de 2007, [19 de xullo de 2009].

[102] World Wide Web Consortium, *The Extensible Stylesheet Language Family (XSL)*. Dispoñible en http://www.w3.org/Style/XSL, 24 de xuño de 2009 [consultado o 29 de xullo de 2009].

[103] World Wide Web Consortium, *XSL Transformations*. Dispoñible en http://www.w3.org/TR/xslt, 16 de novembro de 1999 [consultado o 29 de xullo de 2009].