Més enllà del model relacional: marc actual i noves tendències

Jordi Conesa Caralt
Àngels Rius Gavidia

PID_00171649

Universitat Oberta de Catalunya

Índex

Introducció

Actualment, el model relacional és el més utilitzat pels sistemes de gestió de bases de dades i pels sistemes d'informació que ens envolten. Tot i que aquest model és útil, l'aparició d'Internet i dels dispositius mòbils, a més de la internacionalització i la compartició d'informació, han imposat nous requisits a les aplicacions que els models relacionals no poden satisfer; si més no, amb l'eficiència que podríem esperar. Per aquest motiu, en el darrers temps han aparegut diversos models de bases de dades i extensions al model relacional que permeten tenir en compte altres necessitats: el tractament d'informació amb finalitats analítiques (magatzem de dades, mineria de dades...), el tractament d'informació geogràfica, les bases de dades mòbils, el tractament de documents XML, etc.

En aquest mòdul volem oferir una visió dels problemes més importants del model relacional i mostrar altres bases de dades prou consolidades que trobem actualment. Per acabar, presentarem les tendències actuals en el món de les bases de dades; és a dir, els models o sistemes gestors de bases de dades que són interessants, però que considerem que encara no estan prou consolidats.

Així doncs, tal com hem comentat, en aquest mòdul no volem fer una explicació exhaustiva dels models de dades no relacionals, sinó oferir un primer punt de vista dels avantatges, les aplicacions d'ús i les peculiaritats que tenen tant en l'emmagatzematge de dades com en l'accés a aquestes, el panorama comercial actual i les mancances que hi trobem. En cas que vulgueu aprofundir en un o més d'aquests nous models, us animem a utilitzar les referències proporcionades en l'aula.

Objectius

En el material didàctic d'aquest mòdul, l'estudiant trobarà les eines bàsiques per a assolir els objectius següents:

1. Obtenir una visió general de les limitacions més rellevants del model de base de dades relacional.

2. Presentar una descripció breu de les alternatives més destacades al model relacional, i indicar, per a cadascuna, l'àmbit d'aplicació, el funcionament general i els productes corresponents.

3. Presentar algunes de les noves tendències actuals en bases de dades i els àmbits als quals són aplicables.

1. Limitacions del model relacional

En les últimes dècades s'han utilitzat les bases de dades relacionals com a solució per a qualsevol tipus de problema. No obstant això, les bases de dades relacionals estan concebudes i optimitzades per al tractament d'informació transaccional, el que en anglès s'anomena *online transactions processing*. Emprar una base de dades relacional per a resoldre qualsevol problema és equivalent a fer servir un martell per a enroscar un cargol. Probablement, per a enroscar un cargol podríem usar una eina més adequada.

A mesura que la tecnologia avança, es creen noves aplicacions amb nous requisits i que afronten problemes diversos: magatzem de dades (*data warehouse*), tractament d'informació semiestructurada, necessitat de fer cerques textuals, tractament d'informació geogràfica, etc. Tot i que els sistemes relacionals han evolucionat i han donat solució també a aquests problemes nous, el fet és que no estan expressament dissenyats per a fer-ho.

En els darrers anys han aparegut altres tipus de bases de dades que, a diferència dels sistemes relacionals, s'enfoquen a aplicacions i necessitats concretes. Aquests nous sistemes permeten millorar l'eficiència quan s'apliquen als problemes concrets per als quals han estat definits.

Un exemple de base de dades més eficient que les bases de dades relacionals són les bases de dades orientades a columnes, que estan dissenyades per a millorar l'eficiència en processos que impliquin un ús intensiu d'operacions de lectura contra la base de dades. Els últims estudis mostren que, en alguns casos, l'ús de bases de dades orientades a columnes en processos de gestió de dades permeten millorar l'eficiència en un factor de cinquanta. De fet, en els darrers anys, l'ús d'aquests sistemes basats en columnes s'està imposant en el mercat de la intel·ligència empresarial (*business intelligence*).

Per tant, es pot dir que, fins fa relativament poc temps, un sol sistema gestor de bases de dades horitzontal era aplicable a qualsevol problema i domini. La tendència actual apunta cap a diferents sistemes gestors de bases de dades verticals. Òbviament, aquests nous sistemes permetran tractar problemes específics del domini d'una manera més eficient i més còmoda. Actualment ja hi ha alguns sistemes específics de domini, com ara les bases de dades per a processos analítics (orientades a columnes o gestors de dades), les bases de dades XML, les bases de dades temporals, les bases de dades geogràfiques, etc. Alguns d'aquests tipus de bases de dades s'han repensat des de zero, com, per

exemple, les bases de dades XML natives; en altres casos, en canvi, s'ha adaptat el model relacional per a oferir les noves funcionalitats, com en les bases de dades geogràfiques o espacials.

A banda d'aquests nous tipus de bases de dades, l'increment de la diversitat, tant de nous reptes com de noves classes de dispositius, ha fet que les bases de dades hagin d'oferir noves funcionalitats, com la integració d'aquestes bases en dispositius mòbils, l'emmagatzematge i el tractament de les dades que tenen en memòria (les anomenades *bases de dades de memòria*) i la relaxació de les propietats ACID en alguns casos.

En l'apartat següent presentem una descripció dels tipus de bases de dades que podem trobar avui dia. Posteriorment, en el tercer apartat veurem les noves tendències en el món de les bases de dades que poden donar lloc a nous sistemes de bases de dades en el futur.

2. Altres models de base de dades

A continuació presentem una descripció dels tipus de bases de dades no relacionals que podem trobar actualment. De cadascun, n'estudiarem els avantatges respecte del model relacional, el domini d'aplicació, les peculiaritats d'emmagatzematge i de recuperació d'informació, les mancances i els productes que els implementen. Finalment farem un quadre resum de tots els tipus per tal de mostrar, d'una manera visual, les diferències més destacades i les oportunitats d'ús.

Nota

Recordem que en aquest apartat no oferim una explicació exhaustiva de cada tipus de base de dades, sinó que en presentem les característiques bàsiques amb l'objectiu de donar-les a conèixer i facilitar-vos l'aprofundiment en el tema, si ho considereu necessari.

2.1. Bases de dades orientades a objectes

Les bases de dades orientades a objectes (BDOO) permeten emmagatzemar i recuperar col·leccions d'objectes persistents, d'acord amb un model de dades orientat a objectes.

Enllaç d'interès

Podeu baixar-vos l'especificació de l'ODMG a l'adreça d'Internet següent: http://www.odbms.org/odmg/.

Un model de BDOO permet capturar la semàntica dels objectes suportats pels llenguatges de programació orientats a objectes (LPOO) i, per tant, utilitza els mateixos conceptes que fan servir els LPOO.

El model ODMG, desenvolupat pel consorci que porta aquest mateix nom, és l'estàndard per a bases de dades orientades a objectes. Té com a objectiu aconseguir transportabilitat entre aplicacions que treballen contra bases de dades orientades a objectes i proporciona indicacions per a definir els elements d'aquest model de bases de dades per a fer-los persistents. Bàsicament està constituït a partir dels llenguatges ODL (*object definition language*) i OQL (*object query language*).

2.1.1. Avantatges respecte del model relacional

En relació amb les bases de dades relacionals, els avantatges més importants de les BDOO són els següents:

1) Disposen de més capacitat per a representar objectes del món real, la qual cosa permet la definició d'objectes complexos necessaris per a altres tipus d'aplicacions més sofisticades que les aplicacions de negocis tradicionals.

Recordem que el model relacional es basa en estructures simples i senzilles i els valors que emmagatzema són atòmics. A més, en el model relacional, el conjunt d'operacions possibles està preestablert i lligat als dominis, i no és possible l'herència d'objectes ni d'operacions per a dominis específics.

2) Faciliten la integració de base de dades en aplicacions informàtiques, ja que, en les BDOO, els objectes es representen de la mateixa manera que en els LPOO i, per tant, no cal cap mena de transformació[1].

(1)Les diferències entre la representació dels objectes de la base de dades i els objectes dels llenguatges de programació utilitzats s'anomena *desadaptació d'impedàncies* (en anglès, *impedance mismatch*).

Com que, en el model relacional, la informació es representa mitjançant taules i claus i les aplicacions acostumen a fer servir altres formes de representació, cal una transformació explícita de les dades.

3) Ofereixen suport a la recuperació de dades representades com a objectes complexos i definits recursivament, cosa que millora el rendiment de les aplicacions que gestionen aquest tipus d'objectes i fa més eficients els sistemes que les gestionen.

En el cas del model relacional, no és possible definir dades complexes, només dades simples, i, per tant, es complica la definició i la recuperació de dades en termes d'altres.

4) Possibiliten l'expressió de restriccions relatives a la semàntica de les dades i les seves interrelacions, com, per exemple, la restricció de pertinença d'herència o les de cardinalitat de relacions.

Cal tenir en compte que el model relacional no ofereix un suport adequat per a expressar restriccions i que, per aquest motiu, les restriccions s'han d'incloure en els programes d'aplicacions.

2.1.2. Aplicacions d'ús

Les BDOO van néixer a conseqüència de l'aparició de noves aplicacions que requerien la gestió d'objectes complexos i persistents. Fer persistents els objectes implicava, d'una banda, la definició de nous tipus de dades i noves operacions lligades als tipus i, de l'altra, la necessitat de transaccions de durada llarga en dominis específics.

Les aplicacions CAD/CAM en l'àmbit de l'enginyeria industrial, les aplicacions CASE en el camp de l'enginyeria del programari, els sistemes de gestió en xarxa pel que fa a comunicacions, els sistemes multimèdia i les aplicacions d'autoedició digital en l'àmbit multimèdia i, en general, les aplicacions que utilitzen persistència per a emmagatzemar les dades i l'estat dels objectes són exemples d'aplicacions que fan servir les BDOO.

2.1.3. Peculiaritats d'emmagatzematge

L'esquema d'una BDOO permet definir objectes complexos, tant a nivell d'estat com de comportament, i també les seves interrelacions amb altres objectes. Definir objectes implica especificar, entre d'altres, tipus de dades, constructors o herència mitjançant el llenguatge ODL proposat pel model ODMG.

Els constructors essencials d'ODL són els literals i els objectes. Els literals no tenen identificador i els objectes sí, i per aquest motiu permeten al sistema localitzar i recuperar objectes amb independència del seu contingut.

Els literals, però, tenen associat un valor i poden ser simples o estructurats. Hi ha tres tipus de literals estructurats: atòmics, col·leccions i estructurats. Els tipus de literals bàsics del model d'objectes són: `long`, `short`, `unsigned short`, `unsigned long`, `float`, `double`, `boolean`, `char`, `string`, `enum`, etc. Els literals estructurats estan formats per valors construïts mitjançant el constructor de tuple, cosa que permet crear tipus de dades com ara `date`, `interval`, `time` i `timestamp`. Els literals de tipus col·lecció es construeixen utilitzant recursivament els constructors `set`, `bag`, `list`, `array` i `dictionary`.

Els objectes es caracteritzen per un identificador, un nom, un temps de vida i una estructura. Poden ser de tipus atòmic o col·lecció, i aquests últims són els que es construeixen recursivament a partir d'altres constructors. Per a definir-los s'utilitza la paraula reservada `class` i, sovint, l'usuari és qui crea els objectes establint dues parts: estat i comportament. L'*estat* defineix el valor concret que representa l'objecte mitjançant els atributs que el caracteritzen i les relacions que té amb altres objectes. El *comportament* especifica les operacions que són aplicables a cada tipus d'objecte mitjançant la signatura, en què s'indica el nom de l'operació, el tipus d'argument i el tipus de retorn.

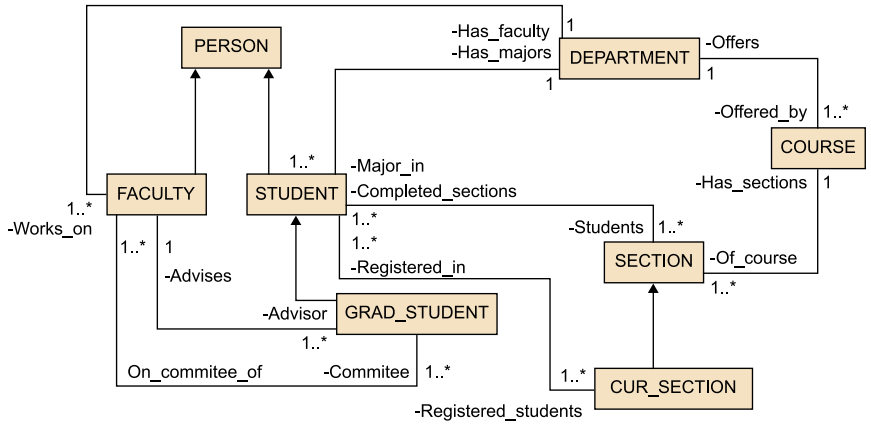
El model ODMG permet especificar tipus d'objectes per herència utilitzant dos mecanismes: 1) l'extensió d'una classe –en aquest cas s'empra la paraula reservada `extends` en l'esquema seguida del nom de la classe a partir de la qual es defineix– i 2) les operacions d'alguna interfície, cas en què el nom de la classe que es defineix va seguit de dos punts i del nom de la interfície de la qual hereta operacions. En el primer cas es poden definir tant propietats d'estat com operacions per mitjà de l'herència, i en el segon, únicament operacions.

Les classes són instanciables i les interfícis, no. Que sigui instanciable vol dir que permet la creació d'objectes individuals d'una classe i els corresponents supertipus i subtipus corresponents. En el model ODMG, el dissenyador de la base de dades pot associar un `extent` a cada classe que declara. Cada `extent` té associada una clau, o més, i un nom que serveix per a referir-se al conjunt d'objectes persistents de la classe. Es podria dir que mitjançant l'`extent` és possible forçar automàticament les relacions entre supertipus i subtipus.

Exemple d'herència per extensió de classes

Suposem que disposem de la base de dades `UNIVERSITY` descrita pel diagrama de classes UML següent:

Diagrama de classes de la base de dades `UNIVERSITY`



R. Elmasri; S.B. Navarthe. *Fundamentals of database systems*, pàg. 726.

A partir del diagrama de classes UML presentat prèviament, i utilitzant les paraules reservades `extend`/`extends`, podem crear l'esquema de bases d'objectes en ODL, tal com es descriu a continuació (Elmasri i Navarthe, 2006, pàg. 727-728).

```
class PERSON
(  extent     PERSONS
   key        Ssn )
{  attribute  struct Pname {  string   Fname,
                              string   Mname,
                              string   Lname } Name;
   attribute  string                          Ssn;
   attribute  date                            Birth_date;
   attribute  enum Gender{M, F}               Sex;
   attribute  struct Address { short    No,
                               string   Street,
                               short    Apt_no,
                               string   City,
                               string   State,
                               short    Zip }   Address;
   short     Age(); };

class FACULTY extends PERSON
(  extent      FACULTY )
{  attribute   string        Rank;
   attribute   float         Salary;
   attribute   string        Office;
   attribute   string        Phone;
   relationship DEPARTMENT   Works_in inverse DEPARTMENT::Has faculty;
   relationship set<GRAD_STUDENT> Advises inverse GRAD_STUDENT:: Advisor;
   relationship set<GRAD_STUDENT> On_committee_of inverse GRAD_STUDENT::Committee;
   void      give_raise(in float raise);
   void      promote (in string new rank); };

class GRADE
(  extent     GRADES )
```

```
{
   attribute   enum GradeValues{A,B,C,D,F,I, P) Grade;
   relationship SECTION Section inverse SECTION::Students;
   relationship STUDENT Student inverse STUDENT:: Completed_sections; };

class STUDENT extends PERSON
(  extent     STUDENTS )
{  attribute  string        Class;
   attribute  Department     Minors_in;
   relationship Department Majors_in inverse DEPARTMENT::Has_majors;
   relationship set<GRADE> Completed_sections inverse GRADE::Student;
   relationship set<CURR_SECTION> Registered_in inverse CURR_SECTlON::Registered_students;
   void        change_major(in string dname) raises(dname_not_valid):
   float       gpa();
   void        register(in short secno) raises(section_not_valid);
   void        assign_grade(in short secno; in GradeValue grade)
                raises(section_not_valid,grade_not_valid); };

class DEGREE
{  attribute   string       College;
   attribute   string       Degree;
   attribute   string       Year;     };

class GRAD_STUDENT extends STUDENT
(  extent    GRAD_STUDENTS )
{  attribute set<Degree>  Degrees;
   relationship Faculty_advisor inverse FACULTY::Advises;
   relationship set<FACULTY> Committee inverse FACULTY:: On_committee_of;
   void        assign_advisor(in string Lname; in string Fname)
                   raises(faculty_not_valid);
   void        assign_committee_member(in string Lname; in string Fname)
                   raises(faculty_not_valid); };

class DEPARTMENT
(  extent          DEPARTMENTS
   key             Dname)
{  attribute       string           Dname;
   attribute       string           Dphone;
   attribute       string           Doffice;
   attribute       string           College;
   attribute       FACULTY          Chair;
   relationship    set<FACULTY>     Has_faculty inverse FACULTY::Works_in;
   relationship    set<STUDENT>     Has_majors inverse STUDENT::Majors_in;
   relationship    set<COURSE>      Offers inverse COURSE::Offered_by; };

class COURSE
(  extent      COURSES
```

```
   key         Cno )
{  attribute   string       Cname;
   attribute   string       Cno;
   attribute   string       Description;
   relationship set<SECTlON> Has_sections inverse SECTION::Of_course;
   relationship <DEPARTMENT> Offered_by inverse DEPARTMENT::Offers;};

class SECTION
(  extent      SECTIONS )
{  attribute   short        Sec_no;
   attribute   string       Year;
   attribute   enum Quarter{Fall, Winter, Spring, Summer}
                     Qtr;
   relationship set<Grade> Students inverse Grade::Section;
   relationship COURSE Of_course inverse COURSE::Has_sections; };

class CURR_SECTION extends SECTION
(  extent     CURRENT_SECTIONS )
{  relationship set<STUDENT> Registered_students
                     inverse STUDENT::Registered_in
   void       register_student(in string Ssn)
              raises(student_not_valid. sectionjull); );
```

En l'esquema presentat anteriorment podem veure que per cada classe del model conceptual es defineix una classe en ODL. Algunes classes, les que es defineixen per extensió de classe, utilitzen la paraula reservada `extends`. Per exemple, les classes `FACULTY` i `STUDENT` hereten de `PERSON` la classe `GRAD_STUDENT` de la classe `STUDENT` i la classe `CUR_SECTION` de la classe `SECTION`. Les relacions entre objectes o els atributs multivaluats també es descriuen mitjançant noves classes. Per exemple, per a emmagatzemar la qualificació que cada estudiant obté en un curs es crea la classe `GRADE` que representa la relació entre les classes `STUDENT` i `SECTION`. També es pot observar que es defineix una nova classe per a definir un atribut multivaluat, la classe `DEGREE`.

Les relacions entre classes es materialitzen mitjançant una referència a l'identificador d'objecte tipus amb el qual es relaciona la classe i s'introdueix amb la paraula reservada `relationship`.

Per a relacions binàries, es pot especificar la relació en un sentit o en dos, però, si es fa en tots dos sentits, cal indicar quines relacions són inverses. En l'exemple, la classe `SECTION` està relacionada amb la classe `COURSE` mitjançant una relació 1 – *N* que indica quines aules (`Has_sections`) té cada curs, i i de quin curs són (`Of_course`). Aquesta relació 1 – *N* s'ha traduït a ODL en dues clàusules **`relationship`**; l'una dins la classe `SECTION` anomenada `Of_course`, i l'altra, dins la classe `COURSE` anomenada `Has_sections`, definida aquesta última com un conjunt mitjançant la paraula reservada **`set`**. Totes dues definicions referencien la inversa mitjançant la paraula reservada `inverse` que va seguida del nom de la classe amb què es relaciona.

Les relacions de tipus M – N sempre es defineixen amb 2 clàusules `relationship` mitjançant el constructor `set` i fent referència a la classe inversa. En el cas d'exemple, en són una mostra les relacions establertes entre les classes `FACULTY` i `GRAD_STUDENT` (`On_commitee_of`, `commitee`) o entre les classes `STUDENT` i `SECTION` (`Completed_sections`, `Students`). També és important observar que, a causa de la creació de la classe `GRADE`, la relació *N* – *M* entre `STUDENT` i `SECTION` es descompon en dues relacions 1 – *N*, una entre `STUDENT` i `GRADE` i l'altra entre `SECTION` i `GRADE`.

A més, cal esmentar que, juntament amb la definició de propietats i interrelacions entre classes, se n'especifiquen els mètodes, però no la implementació d'aquestes operacions.

Exemple d'herència per mitjà d'interfícies

Suposem que volem definir figures geomètriques concretes (rectangles, triangles, cercles) a partir del concepte genèric de figura geomètrica definit per una interfície. És un exemple extret també de l'obra d'Elmasri i Navarthe (2006, pàg. 729).

```
interface GeometryObject
{  attribute   enum      Shape{RECTANGLE, TRIANGLE, CIRCLE,...} Shape;
   attribute   struct    Point {short x, short y} Reference_point;
   float       perimeter();
   float       area();
   void        translated(in short x_translation; in short y_translation);
   void        rotate(in float angle_of_rotation); };

class RECTANGLE : GeometryObject
(  extent      RECTANGLES)
{  attribute   struct    Point {short x, short y} Reference_point;
   attribute   short     Length;
   attribute   short     Height;
   attribute   float     Orientation_angle; };

class TRIANGLE : GeometryObject
(  extent      TRIANGLES )
{  attribute   struct    Point {short x, short y) Reference_point;
   attribute   short     Side_1;
   attribute   short     Side_2;
   attribute   float     Side1_side2_angle;
   attribute   float     Side1_orientation_angle; };

class CIRCLE : GeometryObject
(  extent      CIRCLES )
{  attribute   struct    Point {short x, short y) Reference_point;
   attribute   short     Radius; };
```

Convé recordar que l'herència mitjançant interfícies només permet heretar operacions, no propietats; per tant, en cas de necessitar una propietat de la classe pare, aquesta s'ha de repetir en la classe filla. D'altra banda, les operacions heretades admeten diferent implementació en la classe filla; per exemple, l'operació `area`, que serà implementada de diferent manera segons la figura geomètrica en qüestió. A més es permet l'herència múltiple en el cas de les interfícies, no per a les classes.

En definitiva, ODL permet descriure esquemes de BDOO d'acord amb un model d'objectes que permet dotar els objectes de la persistència necessària per a ser compartits per diverses aplicacions, a més de permetre'n una definició similar a la que fan servir els llenguatges de programació orientats a objectes (LPOO).

2.1.4. Accés a les dades

Quant a l'accés a dades, el model ODMG proposa l'OQL com a llenguatge estàndard d'accés. Aquest llenguatge ha estat dissenyat per a treballar amb llenguatges de programació i utilitza una sintaxi molt semblant a la de l'SQL. Podríem dir que estén l'SQL incorporant característiques típiques del model d'objectes, com ara la identitat dels objectes, els objectes complexos, les operacions, l'herència, el polimorfisme i les relacions entre objectes. A continuació veurem uns quants exemples de consultes en OQL sobre la base de dades `UNIVERSITY` que hem presentat anteriorment i que es poden trobar a l'obra d'Elmasri i Navarthe (2006, pàgs. 735, 737, 739).

Les sentències d'accés a la base d'objectes segueixen el patró de les sentències `SELECT` de SQL. No obstant això, la clàusula `FROM` accepta qualsevol col·lecció d'objectes persistents; així doncs és possible el nom d'un `extent` d'una classe en la clàusula `FROM`, com és el cas de `GRAD_STUDENTS`. Individualment, els objectes es referencien mitjançant variables d'iteració, igual que en SQL.

```
SELECT S
FROM S in GRAD_STUDENTS
WHERE S.Majors_in.Dname='Computer Science';
```

La sentència `select` retorna objectes de qualsevol dels tipus que reconegui el model ODMG. Així doncs, és possible retornar objectes complexos fent ús de la paraula reservada `struct` en la clàusula `select`. Com a novetat, una sentència `SELECT` pot incorporar algun `path expression` format per una seqüència d'atributs, relacions i noms de mètodes concatenats convenientment per a descriure l'accés a les dades.

Path expression

Una `path expression` ha d'entendre's com una ruta que facilita la navegació en la base de dades per a accedir a algun objecte que emmagatzema.

```
SELECT struct(fname:F.name.Lname, salary:F.Salary)
FROM F in FACULTY
ORDER BY F.Salary desc;
```

Pel que fa als operadors, a part dels que defineix l'SQL, l'OQL n'afegeix per al tractament de col·leccions; per exemple, per a obtenir un únic element d'una col·lecció, per a agregar elements d'una col·lecció emulant les funcions d'agregació de SQL, per a obtenir el primer o el darrer element d'una llista o per a establir agrupacions a l'estil de `COUNT` o `GROUP BY` en SQL, etc.

```
SELECT dept_name, avg_gpa: AVG( SELECT P.S.gpa
                                FROM P in partition)
FROM S in STUDENTS
GROUP BY dept_name:S.Majors_in.Dname
HAVING COUNT(partition) A> 100;
```

A més, l'OQL, igual que l'SQL, permet la definició de vistes, i ho fa mitjançant l'ús de la paraula reservada `define`.

```
DEFINE Has_minors (Dept_name) AS
          SELECT S
          FROM S in STUDENTS
WHERE S.Minors_in_Dname=Dept_name;
```

2.1.5. Mercat

Encara que les BDOO van sorgir com a resposta a les necessitats de gestió eficient d'objectes complexos, el terme BDOO com a tal data del 1985. S'han desenvolupat nombrosos prototipus experimentals entre els quals destaquen:

- Encore-Ob/Server (Brown University),
- EXODUS (University of Wisconsin-Madison),
- IRIS (Hewlett-Packard),
- ODE (Bell Labs),
- ORION (Microelectronics and Computer Technology Corporation o MCC),
- Vodak (GMD-IPSI) i
- Zeitgeist (Texas Instruments).

Des del punt de vista comercial, els primers productes que van aparèixer van ser:

- Gemstone (Servio Logic, el nom va ser canviat a GemStone Systems),
- Gbase (Graphael) i
- Vbase (Ontologic).

Després hi va haver dos períodes importants en què van aparèixer en el mercat una gran quantitat de sistemes de BDOO; d'una banda, la primera meitat de la dècada dels noranta, amb:

- ITASCA (Itasca Systems),
- Jasmine (Fujitsu, comercialitzat per Computer Associates),
- Matisse (Matisse Software),
- Objectivity/DB (Objectivity, Inc.),
- ObjectStore (Progress Software, adquirit per eXcelon que, anteriorment, era Object Design),
- ONTOS (Ontos, Inc., nom modificat de Ontologic),
- O2 (O2 Technology, fusionada amb diverses companyies, adquirida per Informix, que, al seu torn, va ser comprada per IBM),
- POET (ara FastObjects, de Versant, que ha estat comprada per Poet Software),
- Versant Object Database (Versant Corporation),

- VOSS (Logic Arts) i
- JADE (Jade Software Corporation).

De l'altra, a partir del 2004, juntament amb l'emergència dels productes de codi obert, van entrar en el mercat db4o (db4objects), DTS/S1 de Obsidian Dynamics i Perst (McObject); aquest últim, també disponible en llicència comercial.

No obstant això, encara que les BDOO són més eficaces que les BDR pel que fa a emmagatzematge d'objectes complexos, i més segures quant a accés a dades, no han aconseguit substituir les bases de dades relacionals. A escala comercial, el que realment ha tingut més sortida són les bases de dades objecte-relacional (BDOOR), ja que són una extensió de les bases de dades relacionals amb característiques de l'orientació a objectes. Els primers productes comercials objecte-relacional que van aparèixer són:

- Illustra (Illustra Information Systems, comprat per Informix Software, que, alhora, va ser adquirit per IBM),
- Omniscience (Omniscience Corporation, comprat per Oracle Corporation i ha esdevingut Oracle Lite), i
- UniSQL (UniSQL, Inc., comprat per KCOMS).

Més recentment, en el terreny comercial han destacat com a bases de dades objecte-relacional: PostgreSQL, SQL Server i Oracle.

2.1.6. Problemàtica

Entre les mancances més importants de les BDOO, destaquen la falta de maduresa i, també, la inexistència d'un model teòric matemàtic subjacent. Actualment es disposa d'un model de dades estàndard, l'ODMG, però, no obstant això, aquest encata no pot competir amb l'experiència del model relacional.

I, malgrat l'activitat experimental existent entorn de les BDOO, a nivell comercial no estan tenint gaire incidència, possiblement, pels riscos que comporta la implantació de noves tecnologies.

D'altra banda, encara hi ha uns quants temes pendents de resoldre, com ara l'optimització de consultes en BDOO, ja que l'encapsulament de consultes en els mètodes fa que la seva implementació quedi amagada i, per tant, impossibilita de millorar l'eficiència de les consultes.

2.2. Bases de dades deductives

Les bases de dades deductives (BDD) permeten emmagatzemar i recuperar informació d'acord amb un model relacional estès. Es tracta d'un model relacional que té molt present el formalisme del càlcul relacional i que s'estén amb predicats de la lògica matemàtica.

L'extensió del model relacional amb axiomes, tant de base com deductius, és el que permet capturar la semàntica de la informació i proveir la BDR de mecanismes d'inferència. Així doncs, la BDD està formada per una base de dades relacional i per un conjunt de regles de deducció.

Els components d'una base de dades deductiva es representen gràficament a la figura següent:

Components d'una base de dades deductiva

La base de dades relacional constitueix la part *extensional* de la BDD, ja que permet la representació explícita de la informació mitjançant axiomes de base o fets.

Les regles deductives (conjunt d'axiomes deductius o regles de deducció) formen la part *intensional* de la BDD i en ser interpretades adequadament sobre els fets emmagatzemats a la base de dades permeten generar nova informació o comprovar restriccions.

2.2.1. Avantatges respecte del model relacional

Els avantatges principals de les BDD respecte de les bases de dades relacionals són els següents:

1) Possibiliten la inferència d'informació a partir de les dades emmagatzemades i les regles de deducció que els són aplicables.

El model relacional no disposa de mecanismes d'inferència automàtics; el que més s'hi assembla són els disparadors, però cal implementar-los.

2) Suporten objectes complexos i conjunts definits a partir de la composició de predicats de la lògica, de manera que són recuperables mitjançant consultes recursives i algorismes eficients per a aquest tipus de dades.

El model relacional només admet estructures de dades simples que permeten la uniformitat en el tractament de la informació mitjançant un llenguatge bàsicament declaratiu, que no resulta idoni per a la recursivitat.

3) Tenen capacitat per a expressar consultes mitjançant regles lògiques, fet que permet automatitzar les especificacions d'aquest tipus de consultes.

El model relacional disposa de vistes per a deduir nova informació, però la seva especificació i la implementació es duen a terme separadament.

2.2.2. Aplicacions d'ús

Les BDD van sorgir a conseqüència de l'ampliació de les bases de dades relacionals amb mecanismes d'inferència. Per tant, aquest tipus de base de dades és especialment adequat per a les aplicacions que requereixen deduir nova informació a partir d'altra informació existent i, també, per a formalitzar models de dades en l'àmbit de l'enginyeria del programari.

A escala comercial no han tingut gaire repercussió. No obstant això, s'han fet servir força des del punt de vista experimental, sobretot en aplicacions específiques dins l'àmbit de la intel·ligència artificial per a la construcció de sistemes experts, per a la representació de coneixement i per a aplicacions que fan servir la tecnologia d'agents, com també per a la integració de dades en els sistemes d'informació.

Actualment, en l'àmbit de la recerca s'utilitzen les ontologies, que, si bé no són BDD, s'hi poden assimilar, ja que permeten la representació de dominis compartits i la recuperació de fets base i derivats. Per a efectuar-ne la implementació, es fa servir un llenguatge de descripció lògica ampliat amb algun altre llenguatge de regles, i la persistència s'aconsegueix quan es pobla l'ontologia.

2.2.3. Peculiaritats d'emmagatzematge

La composició de la BDD –d'una banda, la BDR, i, de l'altra, el conjunt de regles de deducció– fa que sigui necessària la definició i l'emmagatzematge de fets i regles de deducció.

Els fets s'especifiquen d'una manera semblant als tuples d'una relació relacional, però amb la diferència que no cal introduir-hi el nom dels atributs, ja que el significat del valor dins del tuple queda definit per la posició que ocupa dins d'aquest. Per tant, els fets estan constituïts per valors constants i l'especificació corresponent queda descrita per l'esquema de base de dades relacionals.

Les regles de deducció tenen una certa semblança amb les vistes relacionals, ja que permeten especificar relacions virtuals, que no estan emmagatzemades físicament, però que es poden formar a partir dels fets aplicant mecanismes d'inferència basats en l'especificació de regles.

A diferència de les vistes, les regles permeten recursivitat i, en general, s'especifiquen de la manera següent:

Predicat :- Conjunt de predicats,

El predicat de l'esquerra de l'expressió, s'anomena *cap de la regla*, se satisfà si es compleixen tots els predicats del cos de la regla, que està format per una seqüència de predicats que componen la la banda dreta de l'expressió. En cas que la regla es pugui satisfer de diverses maneres, es pot definir més d'una regla, cosa que s'interpretaria com una disjunció lògica.

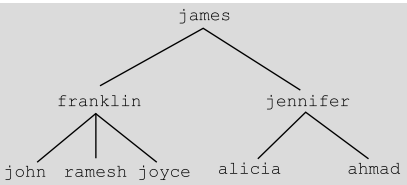
Les regles contenen valors variables i constants. La correspondència entre valors constants dels fets i valors variables del conjunt de regles de deducció fa que se satisfaci el cos de la regla i, per tant, que el cap de la regla sigui interpretat com a veritat.

Exemple de base de dades deductiva

Suposem una base de dades que emmagatzema informació sobre la jerarquia d'empleats d'una organització, de manera que permeti conèixer quins empleats en supervisen d'altres. En una base de dades deductiva aquesta informació es representaria mitjançant un conjunt de fets i regles tal com es mostra a continuació.

Exemple de base de dades deductiva com a conjunt de fets, regles i consultes expressades en Prolog

```
          Facts

SUPERVISE(franklin, john).
SUPERVlSE(franklin, ramesh).
SUPERVISE(franklin, joyce).
SUPERVISE(jennifer, alicia).
SUPERVISE(jennifer, ahmad).
SUPERVISE(james, franklin).
SUPERVISE(james, jennifer).
...

          Rules

SUPERIORS(X,Y):-SUPERVISE(X,Y).
SUPERIORS(X,Y):-SUPERVISE(X,Z),SUPERIOR(Z,Y).
SUBORDINATE(X,Y) :-SUPERIOR(Y,X).

         Queries

SUPERIOR(james, Y)?
SUPERlOR(james, joyce)?
```



A la dreta hi ha una mostra de l'arbre de supervisió.
Font: R. Elmasri; S. B. Navathe. Fundamentals of database systems, pàg. 839.

L'emmagatzematge de fets es realitzaria mitjançant la base de dades relacional. Els fets es materialitzen mitjançant relacions relacionals, i les regles de deducció amb sentències expressades en algun llenguatge de càlcul de predicats com a part d'una vista relacional.

2.2.4. Accés a les dades

La recuperació de dades en el cas d'una BDD comporta l'execució d'una consulta. Una consulta pot retornar predicats de dos tipus: *fets*, definits explícitament en la base de dades com a relacions, i *regles de deducció*, definides implícitament mitjançant vistes.

El llenguatge de predicats de la lògica més utilitzat per a bases de dades deductives és el Datalog[2], que és un subconjunt de Prolog. Per a poder implementar consultes en Datalog, cal conèixer-ne el funcionament o, el que és el mateix, el mecanisme que fa servir per a interpretar les regles.

(2)Web del manual de Datalog: http://www.ccs.neu.edu/home/ramsdell/tools/datalog/datalog.html

A grans trets, hi ha diversos enfocaments per a interpretar les regles de deducció d'una BDD, entre els quals destaquen els basats en la *teoria de demostracions* i la *teoria de models*. Quan s'utilitza l'enfocament de la teoria de les demostracions, es parteix de la premissa que els fets i les regles són predicats certs i, a continuació, s'apliquen les regles sobre els fets per a deduir nova informació.

Exemple de provisió d'un nou fet mitjançant la teoria de demostracions

Partint de la base de dades presentada anteriorment, es mostren un parell de regles que permeten fer una consulta i, a continuació, el conjunt de fets inferits aplicant la teoria de demostracions.

```
1. SUPERIOR(X, Y) :- SUPERVISE(X, Y). (rule 1)
2. SUPERIOR(X, Y) :- SUPERVISE(X, Z), SUPERIOR(Z, Y). (rule 2)

3. SUPERVISE(jennifer, ahmad). (ground axiom, given)
4. SUPERVlSE(james, jennifer). (ground axiom, given)
5. SUPERIOR(jennifer, ahmad). (apply rule 1 on 3)
6. SUPERIOR(james, ahmad). (apply rule 2 on 4 and 5)
```

En la pràctica per a interpretar regles es fan servir dos mecanismes d'inferència, basats en la teoria de les demostracions. El primer, anomenat *ascendent* o *de resolució ascendent*, que a partir d'un conjunt de fets determinats, i utilitzant regles d'inferència, permet generar nous fets en comparar-los amb l'objectiu de la consulta. En el segon, anomenat *d'inferència cap enrere* o *de resolució descendent*, a partir de la consulta es busquen els valors constants que la satisfan. Concretament, aquest darrer mecanisme és el que empren Prolog i Datalog.

Lectura recomanada

Per a un coneixement més detallat del tema, podeu consultar els manuals de Prolog i Datalog.

En canvi, si s'empra la teoria de models, s'assigna a cada predicat totes les possibles combinacions de valors constants del domini i es determina el conjunt de valors per als quals el predicat és cert i el conjunt de valors per al qual és fals. Si el predicat és cert per a tots els valors del domini en qüestió, es diu que és un model. En el següent exemple (Elmasri i Navathe, 2006, pàg. 843) i utilitzant la mateixa base de dades es presenta un exemple que il·lustra aquest segon mecanisme d'inferència.

Exemple d'interpretació a partir de la teoria de models

```
Rules

SUPERIOR(X, Y) :- SUPERVISE(X, Y).
SUPERIOR(X, Y) :- SUPERVISE(X, Z), SUPERIOR(Z, Y).

Interpretation

Known Facts:

SUPERVISE(franklin, john) is true.
SUPERVISE(franklin, ramesh) is true.
SUPERVISE(franklin, joyce) is true.
SUPERVISE(jennifer, alicia) is true.
SUPERVISE(jennifer, ahmad) is true.
SUPERVISE(james, franklin) is true.
SUPERVISE(james, jennifer) is true.
SUPERVISE(X, Y) is false for all other possible (X, Y)
combinations

Derived Facts:

SUPERIOR(franklin, john) is true.
SUPERIOR(franklin. ramesh) is true.
SUPERIOR(franklin, joyce) is true.
SUPERIOR(jennifer, alicia) is true.
SUPERIOR(jennifer, ahmad) is true.
SUPERIOR(james, franklin) is true.
SUPERIOR(james, jennifer) is true.
SUPERIOR(james, john) is true.
SUPERIOR(james, ramesh) is true.
SUPERIOR(james, joyce) is true.
SUPERIOR(james, alicia) is true.
SUPERIOR(james, ahmad) is true.
SUPERIOR(X, Y) is false for all other possible (X, Y)
combinations
```

Finalment, convé remarcar que l'ús de llenguatges basats en la lògica facilita tota classe de consultes recursives, fet que permet incrementar l'eficiència dels sistemes de gestió de bases de dades relacionals per a resoldre certs tipus de consultes basades en la recursivitat.

2.2.5. Mercat

Els primers prototipus i sistemes comercials de BDD daten de la dècada dels vuitanta, quan la recerca al voltant del processament de consultes recursives estava en un punt àlgid. D'aquesta època són Nail! (Universitat de Standford), LDL (Microelectronics and Computer Technology Corporation, MCC) i Megalog o DedGin, que són prototipus del treball de recerca dut a terme a ECRC fora d'un àmbit universitari.

Altres projectes sobre BDD són Aditi (Universitat de Melbourne), ConceptBase system (Universitat de Passau), CORAL (Universitat de Wisconsin), DECLARE (MAD Intelligent Systems), LOGRES (Politecnico de Milà), LOLA (TU München) i Starbust (IBM).

Alguns d'aquest prototipus han estat implementats en Lisp, com LOLA o RDL; d'altres, en Prolog, com LDL, Nail! o Aditi.

Avui, la recerca i el desenvolupament d'aquest tipus de sistemes està més aviat en punt mort i, en l'àmbit de la recerca, com s'ha comentat anteriorment, es fan servir les ontologies com a mecanisme d'inferència, modelatge i integració de dades.

2.2.6. Problemàtica

La problemàtica principal d'aquest tipus de bases de dades està relacionada amb el processament i l'optimització de consultes. Això és degut al fet que les regles s'expressen declarativament i, per tant, no és possible determinar com s'haurà de dur a terme la consulta i, en conseqüència, no es pot optimitzar.

Un altre tipus de problemes de les BDD són els relacionats amb els llenguatges de predicats de la lògica i les limitacions que tenen; aquestes limitacions afecten els SGBD en la seva capacitat per a expressar regles deductives, tant per a derivar nova informació com per a definir restriccions. Per exemple, no és fàcil saber si un procediment desenvolupat amb regles és eficaç o no, amb el perill consegüent que representa entrar en un bucle infinit, o si un context deductiu ha estat plantejat correctament per poder donar resposta a una pregunta. També és difícil establir criteris per, donada una regla, decidir si s'ha de fer servir com a regla de deducció o com a regla de coherència.

2.3. Bases de dades temporals

Les bases de dades temporals (BDT) permeten emmagatzemar i recuperar informació transaccional i històrica, de manera que enregistren l'evolució de la informació. La informació es considera temporal, és a dir, que sempre depèn del temps.

Una BDT organitza les dades tenint en compte que hi ha una associació indissoluble entre *informació* i *temps*. Si bé és cert que les aplicacions informàtiques han considerat sempre la dimensió temporal, històricament, els dissenyadors i els desenvolupadors són els que l'han tingut en compte en el disseny i la implementació de les aplicacions.

Les BDT incorporen aquesta dimensió temporal de la informació i, sovint, s'implementen a partir de BDR. Per tant, per a oferir suport al tractament de la informació temporal, ha calgut disposar d'un llenguatge que estengui l'SQL, el TSQL2. Aquest llenguatge forma part de l'SQL3.

2.3.1. Avantatges respecte del model relacional

Entre els avantatges principals de les bases de dades temporals (BDT) respecte de les relacionals destaquen els següents:

1) Disposen de més capacitat per a representar objectes del món real, i afegeixen una perspectiva històrica o d'evolució de la informació.

Recordeu que el model relacional permet representar la informació que es considera vàlida, i que en possibilita l'actualització i l'esborrament en el moment en què deixa de ser vàlida, de manera que es perd tota la història de canvis.

2) Incorpora nous tipus de dades per a oferir suport a la gestió de la informació temporal.

El model relacional disposa de tipus bàsics per a definir el temps amb una certa granularitat (*date*, *time*, entre d'altres) però no ofereix tipus per a especificar altres dimensions temporals com ara intervals o períodes.

3) Estén el llenguatge d'accés a dades mitjançant operadors que permeten el tractament de la informació temporal i la definició de restriccions associades al temps.

El model relacional no disposa d'operadors per a gestionar intervals o períodes de temps, ni controlar restriccions que afecten la temporalitat de les dades.

2.3.2. Aplicacions d'ús

Les bases de dades temporals han sorgit per donar suport a nous tipus d'aplicacions que mantenen i gestionen dades que varien amb el temps i de les quals interessa conservar-ne tota l'evolució. No és rellevant la granularitat que tingui el component temps, però sí que ho és l'existència d'un fort component temporal en la representació i la manipulació de la informació.

Podem trobar exemples d'aplicacions que fan servir BDT en l'àmbit financer (gestió de la cartera electrònica, de comptes o de bancs), en l'àrea de les assegurances (reclamacions, accidents), en el manteniment de registres (personal, historials clínics, gestió d'inventaris) i en les aplicacions de planificació i control (reserves d'hotel, companyies aèries, tren, lloguer de vehicles). També són útils en la gestió de projectes i d'aplicacions científiques, com, per exemple, en aplicacions que monitoren el temps.

2.3.3. Peculiaritats d'emmagatzematge

Per a poder entendre les peculiaritats de l'emmagatzematge de les BDT, cal conèixer els conceptes rellevants que s'han de representar en aquest tipus de bases de dades.

En primer lloc, es consideren les dimensions del temps: *esdeveniments*, o punts de temps, i *períodes*, o intervals de temps. Els primers es corresponen amb la representació habitual del temps en qualsevol altra base de dades, indepen-

dentment de la granularitat. Els segons, en canvi, es representen mitjançant dos punts de temps, un punt de temps inicial i un punt de temps final, que determinen una durada.

La unitat mínima de temps associada a la informació de qualsevol BDT s'anomena *chronos*. Per tant, qualsevol informació representada en aquesta unitat no ho pot ser mitjançant altres punts de temps.

El fet d'emmagatzemar informació històrica implica registrar tant informació vàlida com no vàlida. Per tant, s'utilitza el concepte *temps de validesa*. El temps de validesa permet indicar en quins períodes la informació és vàlida, i possibilita capturar els diferents estats pels quals passen els objectes del món real que són modelats. D'altra banda, el sistema ha d'enregistrar tots els canvis d'estat de la informació que emmagatzema, i permet la traçabilitat de les actualitzacions. Per a això, cal el *temps de transacció*; és a dir, el temps durant el qual ocorre la transacció. Les bases de dades temporals que suporten aquests dos tipus de temps s'anomenen *bases de dades bitemporals*.

En conseqüència, la creació d'un esquema de bases de dades temporals es complica a causa de les referències temporals que ha d'emmagatzemar. Els tipus de temps que hem comentat –temps de validesa i temps de transacció– sovint es representen mitjançant períodes o intervals de temps; per tant, es defineixen amb un temps inicial i un temps final. Així doncs, el TSQL2 amplia l'SQL i permet la definició de la granularitat de les dimensions de temps considerades en temps de creació de les taules o relacions.

Incorporar el temps en una BDR per convertir-la en BDT requereix diverses adaptacions. El primer aspecte que cal considerar és la clau primària. Atès que es conserven els diferents estats de la informació, la clau primària ha de canviar o ser una subrogació; si més no, ha d'estar formada per més d'un atribut, un dels quals sigui el temps d'inici o de finalització del temps de validesa. Les versions dels tuples diferents corresponen a canvis d'estat que es produeixen al llarg del temps i es representen tenint en compte el temps de validesa. El temps de validesa se sol representar mitjançant dos atributs addicionals per a cada tuple, de manera que es puguin conservar els punts de temps inicial i final de l'interval de validesa. Anàlogament, es representa el temps de transacció per a mantenir el registre de transaccions efectuades sobre la informació i cal tenir en compte que, de vegades, l'actualització d'alguna dada pot representar l'actualització del punt final d'algun interval de validesa d'una altra dada ja existent.

Si es tracta d'incorporar el temps en una base de dades orientada a objectes, aleshores les versions diferents d'un mateix atribut variable amb el temps es poden representar com una estructura de dades complexa, formada pel valor de l'estat de l'atribut i pels temps de validesa inicial i final, i es pot representar

com un triplet. En el cas de bases de dades bitemporals, l'atribut variable temporalment es representaria d'una manera semblant, mitjançant tuples de cinc valors, que inclourien a més els temps de transacció inicial i final.

Definició en ODL de la classe `EMPLOYEE` incorporant temps de validesa i considerant versions

A continuació es presenta part d'un esquema de bases de dades que defineix la classe `EMPLOYEE` incorporant el temps de validesa i considerant versions.

```
class TEMPORAL_SALARY
{   attribute    Date            Valid_start_time;
    attribute    Date            Valid_end_time;
    attribute    float           Salary;
};

class TEMPORAL_DEPT
{   attribute    Date            Valid_start_time;
    attribute    Date            Valid_end_time;
    attribute    DEPARTMENT_VT   Dept;
};

class TEMPORAL_SUPERVISOR
{   attribute    Date            Valid_start_time;
    attribute    Date            Valid_end_time;
    attribute    EMPLOYEE_VT     Supervisor;
};

class TEMPORAL_LIFESPAN
{   attribute    Date            Valid_start time;
    attribute    Date            Valid end time;
};

class EMPLOYEE_VT
(   extent EMPLOYEES )
{   attribute    list<TEMPORAL_LIFESPAN>       lifespan;
    attribute    string                        Name;
    attribute    string                        Ssn;
    attribute    list<TEMPORAL_SALARY>         Sal_history;
    attribute    list<TEMPORAL_DEPT>           Dept_history;
    attribute    list<TEMPORAL_SUPERVISOR>     Supervisor_history;
};
```

2.3.4. Accés a les dades

Accedir a una BDT i recuperar informació que emmagatzema implica gestionar un gran volum d'informació històrica.

Tal com hem vist, sovint es tracta d'una base de dades relacional adaptada a la que s'afegeixen nous tuples o columnes per a incorporar el component temporal de la informació, fet que complica la resolució i l'optimització de les consultes. La implementació d'aquesta BDT es pot fer utilitzant particions horitzontals (per a separar tuples en relacions diferents) o verticals (per a separar columnes de cada tuple) considerant aspectes temporals.

La recuperació de la informació en una BDT requereix principalment tenir en compte els períodes de temps de validesa, principalment, i això vol dir que calen nous operadors per a destriar la informació necessària entre el cúmul d'informació històrica que emmagatzema. Amb aquest objectiu, d'acord amb

la proposta de Richard Snodgrass, s'ha estès l'SQL per a poder tractar informació temporal i dóna lloc al llenguatge TSQL2, el qual, una vegada incorporat en SQL:1999, ha donat origen a SQL3.

Per tal de permetre el tractament de la informació temporal ha estat necessari crear nous operadors. Per exemple, per a reconèixer períodes de temps que se superposen o que estan inclosos en d'altres. En l'àlgebra d'Allen es recullen el conjunt d'operadors relacionats amb intervals que són comuns en diferents BDT implementades amb BDR, a més del conjunt d'operadors de combinació temporals que són necessaris (`time-join`, `time-equijoin`, `event-join`, etc.).

2.3.5. Mercat

Més que sistemes comercials, s'han desenvolupat nombrosos prototipus com a programari intermediari de sistemes de gestió de bases de dades existents. Les primeres implementacions daten de la dècada dels noranta.

Alguns dels prototipus desenvolupats són els següents: ARCADIA (Politecnico de Milà) sobre una plataforma orientada a objecte com és ONTOS, CALANDA (UBILAB), ChronoLog Institut für Informations systems de Zuric) sobre Oracle, HDBMS (Indian Institute of Technology) per a UNIX, TempIS i TimeBD (Universitat d'Arizona), el primer sobre VAX i el segon sobre Oracle, etc.

Quant a implementacions comercials de BDT sobre BDR, destaquen Oracle Workspace Manager, per a Oracle, la darrera versió del qual concorda amb TSQL2 i TimeDB, desenvolupat per TimeConsult, que és lliure, creat per a la plataforma Oracle amb ús de TSQL2.

2.3.6. Problemàtica

Las primeres limitacions de les bases de dades temporals estaven relacionades amb la capacitat d'emmagatzematge, atesa la gran quantitat d'informació que calia emmagatzemar. Posteriorment els problemes a resoldre han estat relacionats amb el processament de consultes i la seva optimització.

D'altra banda, el fet que les BDT es construeixin sobre sistemes ja existents implica dissenyar un programari intermediari per a cada plataforma en particular i, en conseqüència, estan molt lligats a les implementacions de cada moment.

2.4. Bases de dades geogràfiques

Les bases de dades geogràfiques permeten representar tant informació geogràfica com informació alfanumèrica. Per *informació geogràfica* entenem la representació digital d'entitats, objectes o fenòmens que s'esdevenen a la superfície de la Terra o a prop d'aquesta. Les bases de dades geogràfiques de vegades també s'anomenen *bases de dades espacials*.

La representació d'un objecte geogràfic inclou la descripció de la seva geometria i els atributs alfanumèrics que en descriuen les propietats. En el cas de la representació d'una casa, podríem representar-ne la geometria amb una figura rectangular; la posició, amb la latitud i la longitud de cadascun dels vèrtexs del rectangle, i les dades alfanumèriques (any de fabricació, carrer, color, etc.), utilitzant atributs i relacions. Els tipus de dades que permeten representar aquesta informació geogràfica s'anomenen *geometria* (*geometry*) o *característica* (*feature*). L'Open Geospatial Consortium (OGC) ha definit l'especificació *simple features specification* (en endavant, SFS) per a estandarditzar els tipus de dades geogràfiques i les operacions que s'hi poden aplicar. Tot i que la major part de SGBD geogràfics no segueixen al cent per cent aquesta especificació, sí que ho fan en gran mesura.

Open Geospatial Consortium

L'OGC és una organització mundial que treballa per definir estàndards per a serveis geoespacials i serveis basats en localització.

Enllaç d'interès

Podeu consultar l'especificació de l'SFS a l'adreça d'Internet següent: http://www.opengeospatial.org/standards/sfs.

2.4.1. Avantatges respecte del model relacional

Els avantatges principals de les bases de dades geogràfiques són els següents:

1) Permeten emmagatzemar i processar informació geogràfica més eficientment. Representar informació geogràfica en un model relacional implicaria generar una gran quantitat de taules. La recuperació d'informació geogràfica comportaria unir (*join*) taules diverses, cosa que complicaria els processos de recuperació i en reduiria l'eficiència.

2) Proporcionen facilitats per importar dades geogràfiques creades amb un gran nombre de formats i representacions diferents.

3) Proporcionen un conjunt d'operadors i funcions que permeten gestionar d'una manera més natural informació geogràfica.

2.4.2. Aplicacions d'ús

L'ús d'aquest tipus de base de dades està orientat a sistemes d'informació geogràfica, tant generalistes (programes SIG d'escriptori de propòsit general, com ara gvSIG, ESRI, geoServer...) com sistemes *ad hoc* creats per qualsevol empresa. Les facilitats i l'eficiència que ofereixen aquest tipus de bases de dades per

tractar la informació geogràfica, juntament amb les noves tecnologies mòbils amb funcionalitats de geoposicionament, fan d'aquestes bases de dades uns productes molt utilitzats i amb grans expectatives de futur.

2.4.3. Peculiaritats d'emmagatzematge

Per a emmagatzemar informació geogràfica, la base de dades ha de disposar de tipus de dades que permetin representar-la. La representació d'informació geogràfica es pot fer bàsicament de dues maneres: mitjançant gràfics vectorials o mitjançant gràfics *raster*.

En la representació vectorial, els tipus d'objectes que es poden representar mitjançant gràfics vectorials es classifiquen en objectes puntuals, objectes lineals i objectes d'àrea:

- Els objectes puntuals, anomenats *de dimensió zero*, són objectes que es troben un punt concret, com, per exemple, un arbre o una persona.

- Els objectes lineals, anomenats *de dimensió un*, són objectes que es distribueixen linealment (que no vol dir en línia recta) sobre el territori, com ara un riu, la via del tren o una carretera.

- Els objectes d'àrea, anomenats *de dimensió dos*, són objectes que ocupen l'àrea d'un territori, com, per exemple, un terreny, una piscina o una casa.

Cal tenir en compte que un objecte es pot voler representar d'una manera o d'una altra segons l'ampliació (*zoom*) en què es vulgui mostrar; així doncs, un riu es pot representar com una línia, quan el mirem des de molt lluny, o com una àrea, si el mirem de més a prop.

En la representació *raster* es defineix una espècie de malla que representa la superfície que volem. Cada cel·la de la malla conté un valor segons les dades que es vol representar. Normalment, s'utilitzen per a representar cobertures, és a dir, fenòmens que varien en l'espai, com ara la humitat en un punt concret o la densitat de CO_2 en una zona determinada. Un altre tipus de representació *raster* a la qual estem força acostumats són les imatges *raster*, és a dir, imatges o fotografies aèries que se situen sobre el territori. En aquesta categoria trobem les ortofotografies, que són les imatges aèries que observem, per exemple, al Google Maps.

La majoria de les bases de dades geogràfiques emmagatzemen i tracten la informació en format vectorial. A continuació explicarem, respecte d'aquest format, els tipus de dades geogràfiques més habituals que ens trobarem en bases de dades geogràfiques:

1) Punts: és el tipus de dades més simple que permet representar un objecte geomètricament discret i mínim sobre el territori. El punt pot tenir diferents representacions depenent del sistema de coordenades utilitzat, si bé el més comú és descriure'l indicant-ne la latitud i la longitud.

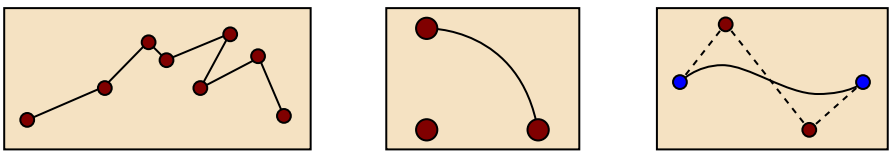
Independentment del sistema de coordenades emprat, també pot caldre representar la coordenada *Z* del punt (la cota o l'altura sobre el terreny del punt) o la direcció del punt. Representar la direcció del punt pot ser útil en alguns casos, com, per exemple, si representem una parada de bus, per a indicar la part de la carretera on queda, i la podem dibuixar de manera perpendicular a la carretera.

2) Línies: els objectes lineals es poden representar mitjançant un conjunt de segments, en què cada segment es pot definir com:

- **Una polilínia**: conjunt ordenat de punts units per un segment recte entre cada punt i el següent.
- **Un arc de circumferència**: es compon de tres punts, dos dels quals defineixen els extrems dels arcs i el tercer indica el centre de la circumferència.
- **Una corba de Bézier**: està definida per un conjunt de punts de pas i punts de control que en determinen el grau.

En la figura següent es mostra un exemple de cada tipus de línia.

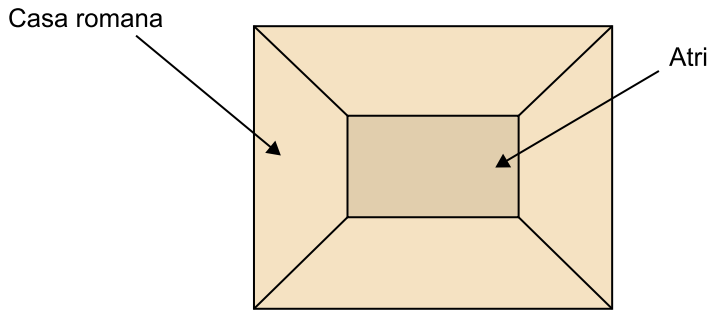
Exemple d'una polilínia, d'un arc de circumferència i d'una corba de Bézier



Atesa la complexitat en el tractament dels arcs de circumferència i les corbes de Bézier, molts sistemes no els suporten, i les polilínies són les més utilitzades actualment.

3) Polígons: la definició d'aquest terme s'ha relaxat en el context dels sistemes d'informació geogràfica per tal de permetre que els polígons continguin forats i tinguin arestes corbes. Per a un sistema d'informació geogràfic, un polígon és una superfície plana delimitada per una frontera exterior i una o més fronteres interiors que en defineixen els forats. Cada frontera és un conjunt ordenat de línies connectades, on l'extrem inicial de la primera es connecta a l'extrem final de la darrera. En la figura següent podem observar com s'ha representat una casa romana amb un polígon que té un forat al mig per a indicar el pati o atri.

Casa romana vista des de l'aire



Per a poder explotar les dades geogràfiques representades en la base de dades, necessitem tenir informació geogràfica de suport que indica, bàsicament, la informació que es representa, el sistema de coordenades que s'utilitza, els camps de la base de dades on es guarda la geometria, etc. Aquesta informació representaria metadades sobre la informació geogràfica emmagatzemada.

Normalment, els SGBD geogràfics són sistemes relacionals que permeten definir informació geogràfica afegint columnes de tipus geogràfic en les taules de la base de dades.

Exemple

En la figura següent, es mostra com es crea en PostGIS una taula per a representar informació sobre les parròquies d'Andorra. Aquesta taula conté tres camps: l'identificador de la parròquia, el nom de la parròquia i un polígon que representa els límits de la parròquia.

```
-- Es crea la taula parròquies sense la informació geogràfica
CREATE TABLE parròquies
(
  id  int,
  nom varchar(50)
);

-- S'afegeix una nova columna geogràfica anomenada geometria
-- a la taula Parròquies de la BD anomenada bd_prova.
-- Aquesta nova columna és de tipus POLYGON, de dimensió 2.
-- També s'hi indica que el sistema de referència utilitzat
-- en aquesta columna és l'identificat pel codi 423
SELECT AddGeometryColum('bd_prova','parròquies','geometria',423,
       'POLYGON',2).
```

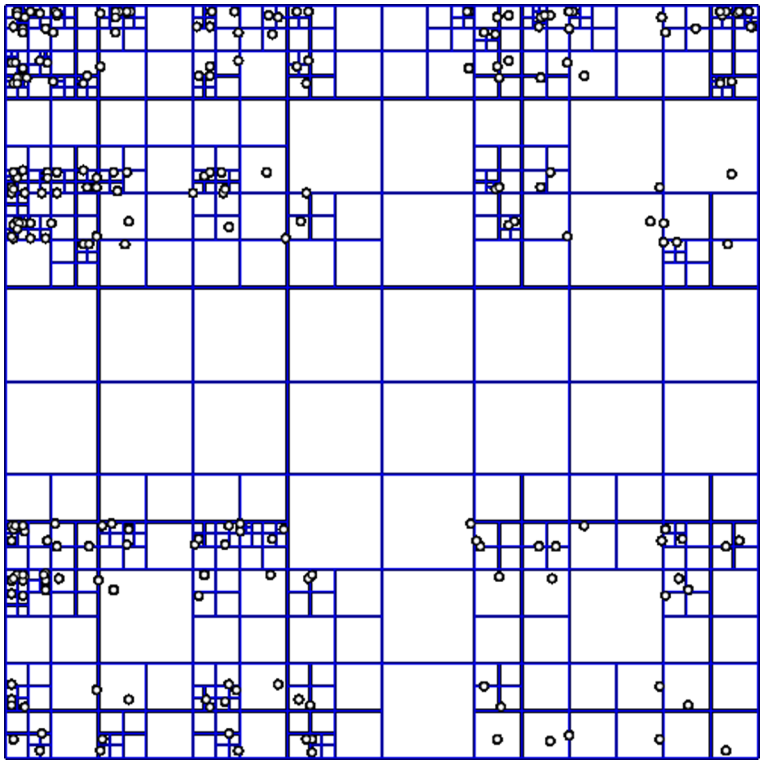
PostGIS

PostGIS és una extensió de PostgreSQL que permet emmagatzemar i tractar informació geogràfica segons l'estàndard SFS d'OGC.

Tal com passa amb la informació alfanumèrica, quan una base de dades conté molta informació geogràfica calen uns índexs espacials per a optimitzar l'accés a les dades. Els índexs espacials permeten optimitzar les cerques sobre informació geogràfica basades en criteris espacials. Com que les dades contingudes en bases de dades geogràfiques són tant alfanumèriques com geogràfiques, s'utilitzen índexs tradicionals (basats en arbres B+, mapa de bits, inversos...), juntament amb índexs espacials per a optimitzar-ne l'accés. Malauradament, els índexs tradicionals no serveixen per a optimitzar consultes espacials. Tot i que hi ha diferents tipus d'índexs espacials, a continuació en comentem dos dels més utilitzats actualment: els basats en arbres quaternaris i els basats en arbres R.

L'arbre quaternari (*quadtree*) és una estructura en forma d'arbre en què cada node de l'arbre té fins a quatre nodes. Aquests tipus d'arbres normalment s'utilitzen per a dividir espais de dues dimensions en parts més petites d'una manera recursiva, com podem veure en la figura següent. Aquest índex és molt eficient per a dividir objectes puntuals, si la nostra geografia conté polígons o línies que poden estar distribuïts entre quadrants diferents.

Exemple d'arbre quaternari per a indexar punts en l'espai



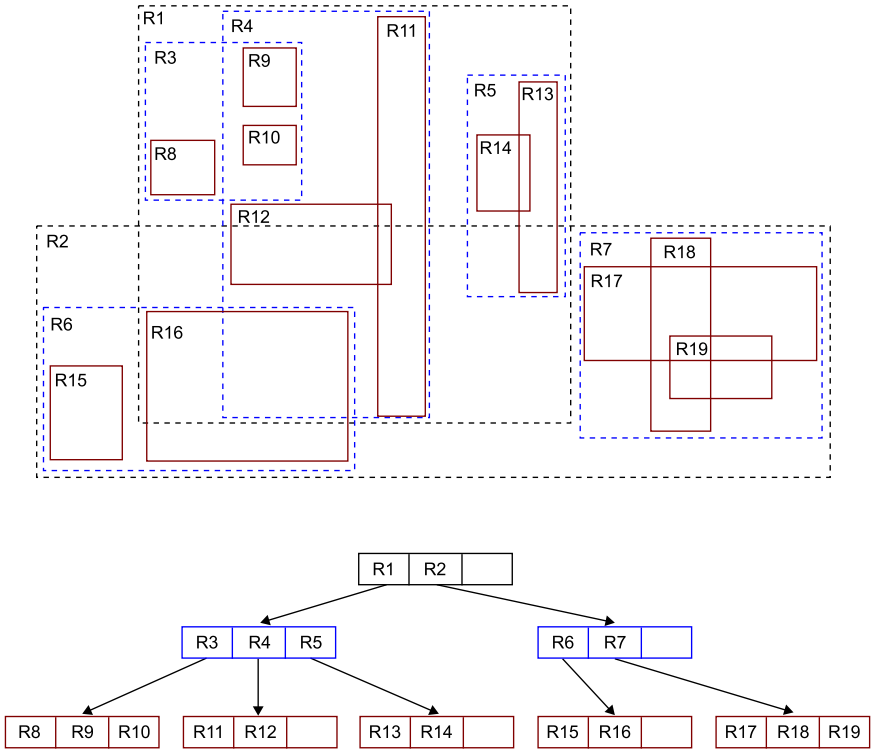
Els índexs basats en arbres R (*R-tree*) possiblement són els més emprats avui, atès que tenen un rendiment superior. Aquest tipus d'índex és molt útil per a optimitzar consultes que requereixin conèixer la proximitat de diferents objectes geogràfics.

Un arbre R permet dividir l'espai en caixes que contenen les geometries emmagatzemades en la base de dades. Un dels paràmetres de l'arbre és el nombre d'entrades màxim per node, cosa que es traduirà en el nombre màxim de geometries que podrà contenir cada caixa contenidora.

Exemple d'arbre R

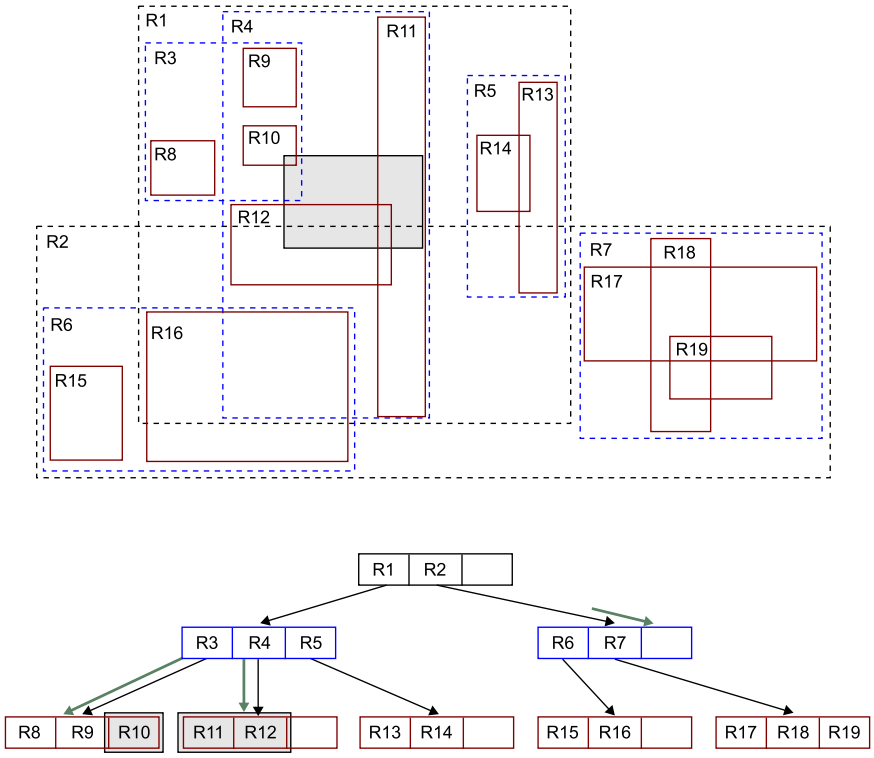
En la figura següent podem veure un exemple de com es divideix l'espai en un arbre R de fins a tres entrades per node. Els rectangles R8 a R19 de la figura són les geometries que s'han d'indexar, mentre que els R3 a R7 són les caixes contenidores que s'han creat per a indexar aquestes geometries bàsiques. Observeu que cada caixa contenidora conté tres objectes geogràfics, com a màxim, i que els objectes s'agrupen per proximitat. Els algoritmes d'inserció i esborrament de nodes de l'arbre han de garantir que els objectes continguts en una caixa contenidora són els més propers. Finalment, les caixes R1 i R2 són les caixes contenidores de nivell més alt i que, per tant, contenen fins a tres caixes contenidores de nivell inferior.

Exemple d'arbre R per a indexar geometries



Tot seguit, en la figura següent es mostra un exemple de com es fan servir els arbres R per a accedir eficientment a la informació geogràfica.

Exemple de cerca fent servir arbres R



Suposem que l'usuari vol saber els objectes que s'han de dibuixar (d'una manera total o parcial) per a mostrar l'àrea marcada amb un rectangle gris). Per resoldre la consulta, l'SGBD comprovaria si els contenidors del node arrel de l'arbre (R1 i R2) s'intersequen amb la zona afectada. Com que R1 conté parcialment la zona d'interès, l'SGBD faria la mateixa comprovació per a les caixes contenidores de R1, situades en el node esquerre de l'arbre (R3, R4 i R5), i es detecta que l'àrea d'interès es talla amb els contenidors R3 i R4, però no amb el contenidor R5. Com que per sota de R3 i R4 no hi ha més caixes contenidores,

sinó els objectes de la base de dades, se seleccionarien els objectes continguts en R3 (R8, R9 i R10) que s'intersequen amb l'àrea que ens interessa (tan sols, R10). Posteriorment, es faria la mateixa comprovació per als objectes R11 i R12, i se seleccionarien tots dos. En aquest punt, l'SGBD tornaria a l'arrel, ja que R2 conté la zona d'interès, i comprovaria que R6 i R7 no són rellevants per a la consulta, ja que no s'intersequen amb la zona d'interès i, per tant, el procés finalitzaria. Al final, l'SGBD retornaria els objectes geogràfics R10, R11 i R12.

2.4.4. Accés a dades

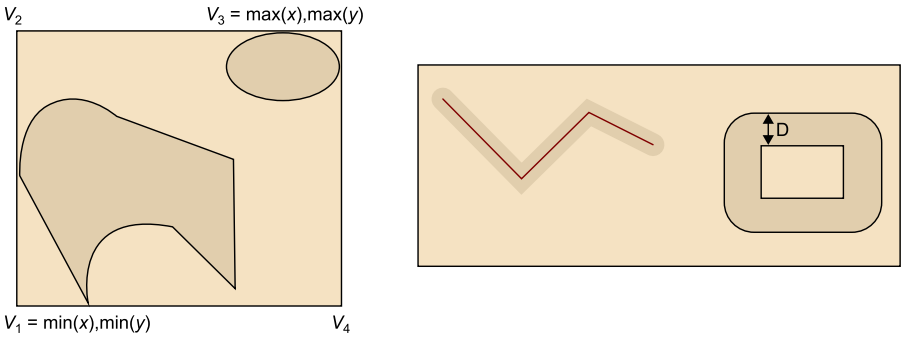
A part de permetre emmagatzemar informació geogràfica, una base de dades geogràfica també ha de permetre executar operacions i funcions geogràfiques sobre les dades. L'estàndard de l'OGC *simple feature type* proposa estendre SQL amb diferents funcions i operadors que permeten treballar amb informació geogràfica. Segons SFT, els operadors són mètodes que tenen com a paràmetres dades geogràfiques i retornen un valor booleà, mentre que les funcions prenen com a paràmetres dades geogràfiques i retornen una nova geometria. A continuació, enumerem la llista d'operadors i funcions espacials definida per l'estàndard SFT:

- **Operadors**: permeten comprovar les relacions espacials entre dos elements amb geometries sobre el pla. Per a entendre les relacions espacials més complexes, cal considerar que cada geometria pot tenir un interior, una frontera, un exterior i una dimensió. Els operadors espacials són:
  - **Igualtat**: retorna cert, si dues geometries són iguals.
  - **Disjunció**: retorna cert, si dues geometries no tenen cap punt en comú en el pla.
  - **S'interseca / S'encreua / Inclou / Conté / Se solapa**: aquests operadors retornen cert, si dues geometries es toquen en algun punt. La semàntica concreta de cada operador varia segons les dimensions dels operands i el grau en què s'intersequen.
- **Funcions**: permeten analitzar geometries i responen amb una nova geometria fruit d'un càlcul:
  - **Centre**: retorna la geometria que defineix el centre de gravetat d'una geometria.
  - **Caixa contenidora**: retorna el rectangle mínim que inclou una geometria (vegeu la figura següent).
  - ***Buffer***: ens retorna una nova geometria que cobreix tots els punts de la geometria, des de la frontera fins a una certa distància definida per l'usuari (vegeu la figura següent).
  - **Unió/Intersecció/Diferència**: retorna la unió, la intersecció o la diferència de dues geometries.

– **Distància**: retorna la distància entre els punts més propers de dues geometries.

– **Veí més proper**: a partir d'una geometria i un conjunt de geometries, retorna la geometria del conjunt més propera a la geometria donada.

Exemple de les funcions caixa contenidora i *buffer*



A banda d'aquestes funcions i operadors, n'hi ha d'altres que permeten dur a terme manipulacions diverses sobre les dades geogràfiques.

Exemple

Aquest exemple mostra la manera d'obtenir l'àrea i el perímetre de les parròquies d'Andorra.

```
SELECT nom as Parròquia,
    ST_Àrea(geometria)/1000 AS "Àrea (hectàrees)",
    ST_Perímetre(geometria)/1000 AS "Perímetre (km)"
FROM parròquies
```

Com a resultat d'aquesta consulta, obtindríem el següent:

	Parroquia character varying(50)	Área (hectáreas) double precision	Perimetro (Km) double precision
1	Ordino	8172.69954088764	44.5990749394133
2	Canillo	10599.3382606277	57.1567993913074
3	Encamp	7006.86569735069	45.4695272150604
4	La Massana	6439.54877985706	43.0359895920158
5	Escaldes-Engordany	4808.25631655273	36.513675724385
6	Andorra La Vella	2012.99846212931	20.07362300846
7	Sant Julià de Lória	6183.03843277001	37.142993711361
8	Terreny de Concordia	1547.0131482954	19.0172548681042

2.4.5. Mercat

PostGIS és l'SGBD geogràfic de programari lliure més utilitzat actualment. Es tracta d'una extensió de PostgreSQL que permet emmagatzemar i tractar informació geogràfica. És un producte en evolució constant, i que té un rendiment i una potència similars als seus homòlegs propietaris. A més a més, aquesta base de dades incorpora pgRouting, una extensió de PostGIS que permet calcular rutes i executar altres operacions geogràfiques avançades, tenint en compte la informació geogràfica emmagatzemada en la base de dades. PostGIS segueix al peu de la lletra les especificacions SFT d'OGC; això fa que sigui molt fàcil enllaçar-la amb altres eines geogràfiques, com, per exemple, eines SIG d'escriptori i servidors de mapes.

Oracle Spatial és la versió d'Oracle per a tractar informació geogràfica, i és la que molts consideren la millor base de dades geogràfica del mercat. Oracle no segueix l'estàndard SFT d'OGC al cent per cent, i incorpora alguns canvis en la manera d'emmagatzemar la informació geogràfica i el nom de les funcions i els operadors geogràfics. Oracle permet també emmagatzemar la informació geogràfica en format vectorial i *raster* i utilitzar regles de topologia.

Altres SGBD potents que cal tenir en compte a l'hora de tractar dades geogràfiques són Microsoft SQL Server Spatial i Spatial Extender d'IBM DB2.

MySQL també disposa d'una extensió geogràfica, però és menys utilitzada que PostGIS. El motiu és que MySQL no segueix l'estàndard OGC i ofereix menys funcionalitats i un rendiment pitjor que PostGIS.

2.4.6. Problemàtica

Les bases de dades geogràfiques fa temps que existeixen i estan molt consolidades. Potser la problemàtica principal que presenten és la poca formació que es fa actualment d'aquest tipus de base de dades en les titulacions tecnològiques. Això fa que, per simple desconeixement, es perdin moltes oportunitats de fer servir bases de dades geogràfiques que poden ser rellevants.

2.5. Bases de dades distribuïdes

> Una base de dades distribuïda permet emmagatzemar i recuperar dades que es troben lògicament distribuïdes en ubicacions diferents i interconnectades mitjançant alguna xarxa de comunicació de manera que l'usuari pot treballar contra aquesta com si es tractés d'una única base de dades.

Així doncs, no es pot parlar d'un model d'una base de dades distribuïdes, ja que es tracta d'un conjunt de bases de dades possiblement suportades per models diferents de bases de dades. Òbviament, aquesta distribució de les dades comporta una certa complexitat addicional quant a disseny i implementació del sistema, que no té pas una base de dades centralitzada.

2.5.1. Avantatges respecte del model relacional

Atesa la composició d'un sistema distribuït de bases de dades, no té massa sentit fer una comparativa respecte dels sistemes de bases de dades relacionals. No obstant això, sí que pot ser interessant enumerar-ne els avantatges respecte de les bases de dades centralitzades.

1) Reflecteix l'estructura organitzativa i, al mateix temps, facilita l'autonomia local mitjançant la distribució de dades pròpia. L'accés a les dades, el processament de consultes o la gestió de transaccions en una base de dades distribuïda es pot efectuar amb transparència; és a dir, sense que l'usuari se n'assabenti.

2) Incrementa la fiabilitat i la disponibilitat de les dades. El fet de tenir les dades distribuïdes incrementa la probabilitat que el sistema funcioni i ho faci en un interval de temps determinat, ja que una fallada en una part del sistema afectarà només un fragment de la base de dades.

3) Augmenta el rendiment. La localització de la informació en llocs diferents facilita la ubicació de la informació en el lloc de més demanda i permet que els sistemes treballin en paral·lel, cosa que permet distribuir més bé la càrrega dels servidors i optimitzar el rendiment del sistema.

4) Facilita l'extensió modular del sistema. És més fàcil incorporar noves dades, augmentar la mida de la base de dades i afegir nous processadors sense que es vegin afectats la resta de sistemes.

2.5.2. Aplicacions d'ús

Les aplicacions d'ús de les bases de dades distribuïdes van sorgir per una motivació doble. D'una banda, la necessitat de sistemes integrats que donin resposta als objectius globals de les organitzacions, i de l'altra, la voluntat d'atorgar autonomia als usuaris perquè siguin més eficients en les tasques. No obstant això, aquest ús en contextos organitzatius es donava a l'inici de les bases de dades distribuïdes.

Posteriorment, l'adveniment d'Internet ha canviat força el panorama, ja que el que abans era tendència a la distribució ha esdevingut una necessitat per a la majoria d'aplicacions web.

2.5.3. Peculiaritats d'emmagatzematge

La distribució de les dades en llocs diferents fa que en aquest tipus de bases de dades s'utilitzin unes tècniques d'emmagatzematge determinades, entre les quals destaquen la replicació i la fragmentació de dades. Com que la distribució de dades s'ha gestat des de les bases de dades relacionals i a conseqüència de restriccions d'espai, es farà èmfasi en les tècniques que s'empren en la distribució de dades emmagatzemades en bases de dades relacionals. Es podria procedir d'una manera anàloga per a altres models de bases de dades.

La replicació de dades possibilita que les mateixes dades s'emmagatzemin en diversos llocs al mateix temps, cosa que permet millorar el rendiment de consultes globals a canvi d'haver d'actualitzar les diverses rèpliques per a mantenir la consistència de la base de dades.

En canvi, la fragmentació permet dividir la base de dades d'acord amb una unitat lògica de distribució que pugui ser adequada per a reduir el nombre d'accessos remots i incrementar el grau de concurrència. Per contra, comporta una degradació del rendiment del sistema i una certa complexitat per continuar mantenint la integritat referencial en la base de dades.

Hi ha diversos tipus de fragmentació:

- **Horitzontal**. La *fragmentació horitzontal* es dóna quan una relació es divideix en unes quantes, d'acord amb un cert criteri lògic de distribució de tuples i, per tant, la recuperació de la relació inicial s'obtindrà a partir de la unió de fragments.
- **Vertical**. La *fragmentació vertical* es produeix quan les dades d'una relació se separen per columnes, fet que implica com a mínim la duplicació de la clau primària per a possibilitar la combinació de fragments en la recuperació de dades.
- **Mixta**. La *fragmentació mixta*, tal com indica el nom, s'obté barrejant els dos tipus de fragmentació anteriors, l'horitzontal i el vertical.

Exemple de fragmentació de dades horitzontal i vertical

Considerem la taula Projectes amb dades de projectes. Si les dades es fragmenten verticalment considerant quan el valor de la columna Pressupost, és superior a 200.000 o no, obtenim dues taules: la taula Projecte_H1, que conté les dades de projectes amb pressupost inferior o igual a 200.000 i la taula Projecte_H2, que conté les dades amb un pressupost superior a 200.000.

Taula Projecte

Num_proj.	Nom_projecte	Pressupost	Ubicació
1	Instrumentació	150.000	Monterrey
2	Desenvolupament de bases de dades	135.000	Mèxic
3	CAD/CAM	250.000	Puebla
4	Manteniment	310.000	Mèxic
5	CAD/CAM	500.000	Guadalajara

Taula Projecte_H1

Num_proj.	Nom_departament	Pressupost	Ubicació
1	Instrumentació	150.000	Monterrey
2	Desenvolupament de bases de dades	135.000	Mèxic

Taula Projecte_H2

Num_proj.	Nom_departament	Pressupost	Ubicació
3	CAD/CAM	250.000	Puebla
4	Manteniment	310.000	Mèxic
5	CAD/CAM	500.000	Guadalajara

Si, en canvi, es fragmentessin les dades verticalment es podrien repartir les dades en dues taules: Projecte_V1, amb les dades relatives a pressupost, i Projecte_V2, amb el nom i ubicació de projecte.

Taula Projecte_V1

Num_proj.	Pressupost
1	150.000
2	135.000
3	250.000
4	310.000
5	500.000

Taula Projecte_V2

Num_proj.	Nom_projecte	Ubicació
1	Instrumentació	Monterrey
2	Desenvolupament de bases de dades	Mèxic
3	CAD/CAM	Puebla
4	Manteniment	Mèxic
5	CAD/CAM	Guadalajara

En conseqüència, la creació d'un esquema de BDD implica la creació d'un esquema de bases de dades convencional ampliat amb l'esquema de localització i l'esquema de fragmentació.

L'*esquema de localització* descriu el lloc on es localitzen els diversos fragments de la base de dades i, en cas que existissin rèpliques, on es troben. D'altra banda, l'*esquema de fragmentació* descriu els fragments de la base de dades que inclouen les columnes o tuples de la base de dades que satisfan alguna condició i, per tant, com es poden reconstruir mitjançant operacions d'unió o combinació per a donar una resposta global.

2.5.4. Accés a dades

L'organització de les dades sempre ve condicionada pel mode d'accés, i, en aquest cas, també per la distribució.

Suposem que tenim un sistema de bases de dades distribuïdes format per diferents tipus de sistemes de gestió de bases de dades i que, per a poder funcionar com a si es tractés d'un sistema centralitzat, necessita un esquema que ofereixi una visió global de la base de dades. Si, a més, els sistemes són independents (és a dir, funcionen autònomament i cooperen entre si gràcies a un esquema de base de dades global), aleshores la base de dades distribuïda s'anomena també *base de dades federada* o *sistema multibase de dades*. Concretament, en aquest cas, la resolució de consultes encara es pot complicar més; per exemple, perquè s'utilitzin diversos models de dades i entrin en contradicció algunes restriccions o perquè es facin servir diversos llenguatges d'accés a la base de dades amb tipus de dades i operadors diferents.

El processament de consultes en un sistema distribuït permet l'execució paral·lela en llocs de la xarxa diferents, de manera que es distribueix la petició de consulta i es pot dur a terme el processament en llocs diversos. En aquest cas, doncs, convé reduir el temps de transferència per la xarxa, que és el coll d'ampolla, i, en la mesura que sigui possible, establir alguna estratègia d'optimització local. Amb el primer objectiu, reduir el trànsit per la xarxa, s'ha de procurar distribuir el mínim d'informació i, amb aquesta finalitat, cal una nova operació de l'àlgebra, la *semijoin*. Una *semijoin* pretén reduir el nombre de tuples d'una relació que s'ha de combinar amb les d'una altra relació que es troba en una altra ubicació, i s'aconsegueix que es transfereixi el mínim d'informació possible per la Xarxa.

A més a més, quan les operacions d'accés a la base de dades formen part d'una transacció, en els sistemes de bases de dades distribuïdes sorgeixen nous problemes que cal resoldre: 1) el control de concurrència ha de mantenir les rèpliques perquè la base de dades sigui sempre consistent; 2) el sistema ha de continuar treballant quan fa fallida algun dels diversos punts de la xarxa o es malmet algun enllaç de comunicació, i 3) s'han d'establir mecanismes per a solucionar possibles problemes derivats de confirmacions de transaccions en entorns distribuïts o d'abraçades mortals entre punts diferents de la xarxa.

2.5.5. Mercat

Aquest tipus de bases de dades ha proliferat pel fet que s'adequa a entorns distribuïts. Actualment hi ha molts productes comercials, projectes de codi obert i tecnologies en fase d'investigació que suporten grans volums de dades que s'han de distribuir i que fan servir múltiples servidors per al processament de consultes.

D'entre els productes comercials més utilitzats avui dia, destaquen: Microsoft SQL Server 2008 i Oracle 11g. Tanmateix, també són rellevants altres productes de codi obert, com ara Apache Cassandra, que inicialment va ser desenvolupat per Facebook.

Cal destacar que Internet i el gran nombre d'aplicacions web, o per a entorns distribuïts, ha fet avançar molt aquest tipus de bases de dades ateses les diferents necessitats d'emmagatzematge, de recuperació i de tractament de la informació. Aquest fet ha donat lloc a altres tipus de bases de dades que encara no estan del tot consolidades i que es poden considerar tendències, però que en el fons són simplement tipus de bases de dades distribuïdes, com ara bases de dades clau-valor, bases de dades documentals, bases de dades per columnes, etc. No les comenten en aquest apartat, ja que les tractarem individualment més endavant.

2.5.6. Problemàtica

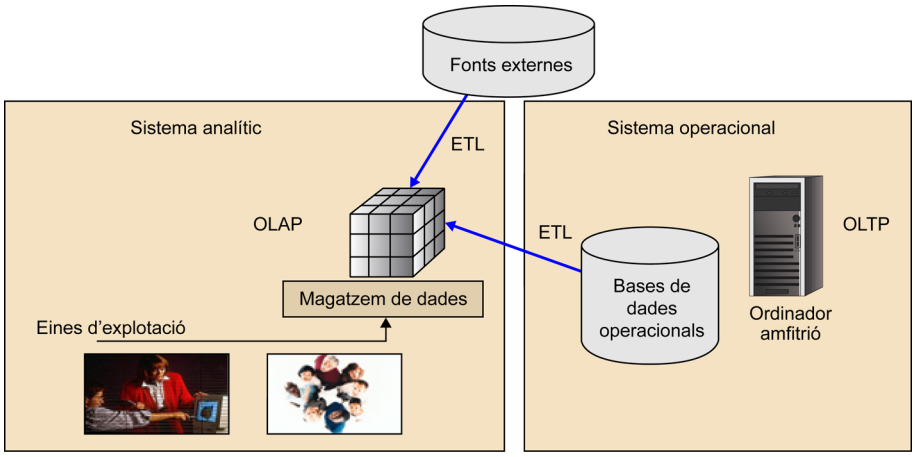
No hi ha estàndards, eines ni metodologies que ofereixin suport als usuaris en la conversió d'un sistema de gestió de bases de dades centralitzat a sistema de gestió de bases de dades distribuït, de la mateixa manera que hi ha mancança de personal qualificat i experimentat en bases de dades distribuïdes.

La complexitat d'aquest tipus de bases de dades implica la resolució de molts problemes relacionats amb la distribució de dades, l'accés a les dades, el control de concurrència i la recuperació. Per exemple, des del punt de vista de la recuperació, s'han de resoldre situacions en què es produeixi fallida per part d'algun node implicat, cal fer còpies de seguretat de nombrosos fragments de dades en formats ben diversos, s'han d'executar transaccions distribuïdes, etc.

2.6. Bases de dades analítiques

Les bases de dades analítiques (BDA) permeten emmagatzemar i recuperar informació amb l'objectiu d'oferir suport a la presa de decisions. Aquestes bases de dades també s'anomenen *magatzems de dades* (*data warehouse*) i formen part d'un sistema més ampli.

Un magatzem de dades forma part de l'arquitectura de sistemes següent:



Arquitectura de sistemes bàsica quan es treballa amb magatzems de dades

Un dels principals components de les bases de dades analítiques són les fonts de dades que s'integren en aquest tipus de sistemes, en concret les bases de dades operacionals, que són bases de dades transaccionals, les quals emmagatzemen el dia a dia de les empreses, després d'haver estat tractades convenientment. D'altra banda, les eines OLAP (*online analytical process*) permeten l'anàlisi i explotació de bases de dades analítiques.

Tot i que aquestes bases de dades es poden construir utilitzant el model relacional, també hi ha la possibilitat de crear-les a partir d'un model de dades multidimensional o d'una combinació de tots dos.

2.6.1. Avantatges respecte del model relacional

Els avantatges principals d'un magatzem de dades respecte d'una base de dades relacional són els següents:

1) Permet representar informació que pot ser útil per a la presa de decisions, en comptes de ser-ho per a l'operativa diària en l'organització.

La representació de la informació en el model relacional es fa a partir de requisits funcionals. Quan es dissenya un magatzem de dades, es desconeixen aquests requisits i, per tant, la representació de la informació es fa tenint en compte les àrees d'interès o els conceptes.

D'altra banda, el model relacional gestiona informació detallada, mentre que el magatzem de dades utilitza informació agregada, que és més útil per a l'estratègia empresarial.

2) Suporta la integració de dades provinents de fonts diverses, cosa que permet l'obtenció d'un model de dades comú dissenyat per a treure rendiment de grans volums de dades històriques.

El model relacional és utilitzat per a bases de dades operatives, és a dir, bases de dades orientades a gestionar transaccions de negoci que són quotidianes en les organitzacions. A més, les bases de dades relacionals estan pensades per a emmagatzemar informació del moment, no informació històrica.

3) És pensat per a l'anàlisi de dades amb eines OLAP, no per a actualitzacions en línia.

El model relacional està especialment indicat per a sistemes que requereixen actualització de dades en línia i recuperació de dades mitjançant consultes *ad hoc* per mitjà d'algun llenguatge d'accés a la base de dades.

4) Facilita la integració de magatzems de dades en entorns de treball d'usuaris que tenen un perfil analític o gerencial.

El model relacional facilita la integració de la base de dades en aplicacions informàtiques i els usuaris disposen de coneixements informàtics.

2.6.2. Aplicacions d'ús

El disseny d'aquest tipus de bases de dades orientades a l'anàlisi de grans volums de dades ha convertit els magatzems de dades en peces clau per a totes les aplicacions relacionades amb la presa de decisions.

Sobretot es fan servir en àmbits de la indústria en què l'estratègia es fonamental per al negoci. Concretament, en la presa de decisions es poden fer servir per a aplicacions d'anàlisi de tendències, de prediccions financeres, d'anàlisi de frau, d'anàlisi de registres de trucades, per a la gestió logística d'inventaris, etc.

2.6.3. Peculiaritats d'emmagatzematge

La informació que s'ha d'emmagatzemar en un magatzem de dades és informació integrada, històrica, no volàtil i orientada a conceptes, organitzada convenientment per a oferir suport a la presa de decisions. Aquesta informació prové de les bases de dades operacionals de l'organització i de fonts externes a aquesta, les quals són transformades convenientment abans que la informació sigui incorporada en el magatzem de dades.

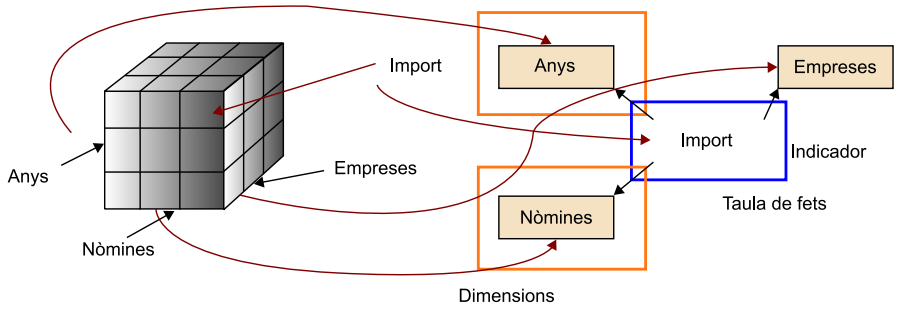
Atès que, per les característiques que té una base de dades analítica, no existeix la problemàtica associada a l'actualització de dades que s'esdevé en les bases de dades relacionals, no hi ha la necessitat d'evitar la redundància de dades i es fomenta l'agrupació i l'agregació de dades, fet que facilita la presa de decisions i permet accelerar un cert tipus de consultes. D'altra banda, un magatzem de dades pot estar format per diversos magatzems de dades departamentals, de

manera que la informació es troba repartida entre àrees de negoci o departaments, donant lloc a magatzems de dades més reduïts i detallats que es poden optimitzar més fàcilment.

En els sistemes analítics, les dades acostumen a estar estructurades d'acord amb un model de dades especial, anomenat *model multidimensional*, que, si bé no és l'únic, sí que és un dels que proporciona més bons resultats d'explotació de dades.

El model multidimensional es basa en la representació de la informació relativa a l'organització mitjançant cubs multidimensionals. Aquesta informació s'organitza en forma de fets, dimensions i mesures que posteriorment s'utilitzen per a l'anàlisi de la informació. Els *fets* representen l'activitat que es vol analitzar. Les *dimensions* caracteritzen l'activitat, és dir, materialitzen els diversos punts de vista des dels quals es pot analitzar l'activitat. Les *mesures* o *atributs dels fets* representen la informació rellevant sobre els fets i els *atributs de dimensió* representen la informació descriptiva de cada dimensió. Aquests conceptes, mitjançant el model multidimensional de dades, es representen per un cub de dades, de manera que les cel·les contenen valors, i les arestes són les dimensions.

Representació del model multidimensional de dades mitjançant un cub



Un cop determinat el model de dades, cal construir el magatzem de dades, fet que implica triar una arquitectura OLAP. Existeixen les següents arquitectures possibles: ROLAP (*relational on-line analytical processing*), MOLAP (*multidimensional on-line analytical processing*) i HOLAP (*hybrid on-line analytical processing*) o híbrida.

En l'arquitectura ROLAP, la informació s'emmagatzema d'acord amb un esquema de bases de dades relacional desnormalitzat, que representa els fets i les dimensions en taules relacionals, les quals es relacionen entre si, donant lloc a noves taules que contenen informació precalculada i agregada. Així doncs, els sistemes analítics que utilitzen aquest tipus d'arquitectura s'implementen amb sistemes de gestió de bases de dades relacionals; i, per tant, utilitzen com a estructura d'emmagatzematge taules amb files i columnes, i procurant

l'emmagatzematge per files. Per a la representació de les dades, siguin agregades o no, s'utilitza un conjunt de tipus de dades molt extens i, sovint, els models de dades que se n'obtenen són difícils d'entendre.

Alguns models típics que resulten a l'hora de construir un magatzem de dades ROLAP són els següents:

- el **model en forma d'estrella**, que no explicita cap mena de jerarquia entre atributs de diferents dimensions de l'esquema;

- el **model de dades floquet de neu**, que explicita jerarquies d'atributs de diferents dimensions de l'esquema i les emmagatzema en taules relacionals distintes, i

- el **model de constel·lació de fets**, que combina diversos esquemes d'estrella o de floquet de neu per compatir dimensions, de manera que dóna lloc a noves entitats en separar alguns atributs de dimensions, que poden ser compartits amb altres taules de fets.

En els sistemes analítics MOLAP, la informació s'implementa amb sistemes de gestió de bases de dades multidimensionals amb arquitectura propietària. Aquesta multidimensionalitat permet emmagatzemar la informació en cubs de manera que es facilita la visualització de les dades en diferents dimensions d'anàlisi. L'estructura de dades emprada per a aquest tipus de sistemes esta basada en vectors i matrius i les dades que admet són numèriques, cosa que comporta problemes amb les descripcions textuals i els fets no agregables. A més, la utilització d'estructures de dades poc flexibles complica també la comprensió del model i la modificació de model de dades, però en canvi no requereix cap transformació entre el model lògic i el model físic.

Hi ha altres arquitectures híbrides HOLAP i d'altres tipus, com les basades en columnes i el model associatiu.

2.6.4. Accés a dades

L'accés a dades d'un magatzem de dades ve condicionat per l'organització de les dades i, a més, exigeix una recuperació de dades tan dinàmica i eficient com sigui possible per tal de donar resposta a la presa de decisions.

En el cas d'un sistema ROLAP, la base de dades relacional s'encarrega de l'emmagatzematge de les dades i el motor ROLAP, de la seva funcionalitat analítica. Així doncs, un cop definit l'esquema de bases de dades, aquestes es pugen des d'un sistema operacional, s'executen rutines per a aconseguir agregació de dades, si escau, i es creen els índexs per a l'optimització dels temps d'accés en l'execució de consultes. En la pràctica, quan un usuari ordena una anàlisi de dades determinada, cal transformar la consulta que efectua a consulta SQL, de manera que s'estableixi una correspondència entre les metadades OLAP i

el model físic de la base de dades, de manera que s'estableix una correspondència entre dimensions-cubs-jerarquies i taules-columnes-relacions. El llenguatge d'accés a dades és l'SQL ampliat amb funcions addicionals afegides pel fabricant. Es poden fer servir tota classe d'eines proporcionades per la BDR, les vistes materialitzades per a l'optimització de consultes i els índexs d'unió per a reduir el temps de resposta.

En canvi, en el cas d'un sistema MOLAP, la representació de la informació mitjançant cubs resulta molt més clara visualment, ja que facilita la visualització de les dades en les diferents dimensions d'anàlisi. La càrrega dels cubs es duu a terme per lots mitjançant un conjunt de rutines que s'executen i que calculen les dades agregades considerant les dimensions del negoci. Després, es generen els índexs i les taules de dispersió necessàries per a optimitzar les consultes. A partir d'aquest moment, la base de dades està disponible per a ser consultada sense que calgui correspondència entre el model lògic i el model físic de la base de dades. Les consultes es realitzen amb llenguatges específics que proveeix cada fabricant i que, de vegades, accepta algunes funcionalitats de l'SQL. Un dels llenguatges desenvolupats i bastant acceptat és l'MDX (*multidimensional expressions*) de Microsoft i que incorpora SQL Server. Aquest llenguatge ha estat acceptat per altres fabricants, cosa que l'ha convertit en un possible futur estàndard de fet. En general, cada fabricant incorpora unes operacions pròpies per a facilitar l'anàlisi dinàmica de dades utilitzant els cubs multidimensionals, però ja hi ha un conjunt mínim d'operacions comunes en tots ells.

Les operacions comunes a qualsevol llenguatge d'accés a una BDA multidimensional són les següents: *Roll-up*, *Drill-Down*, *Drill-Across*, *Slice*, *Dice* i *Pivot*.

Roll-up és un operador que permet agregar dades en diferents nivells d'agregació prèviament definits en l'esquema multidimensional; *Drill-down* és l'operador contrari a l'anterior i, per tant, és un operador OLAP que permet desagregar dades i baixa a nivells més atòmics de l'esquema multidimensional; *Drill-across* permet moure les dades d'un fet cap a un altre; *Slice* permet extreure una part del cub corresponent a una dimensió donada, mentre que *Dice* defineix un subcub de l'espai original. D'altra banda, *Pivot* permet orientar el cub de manera que facilita una visualització més intuïtiva i natural de les dades.

Malgrat l'existència d'aquestes operacions i els índexs generats, encara es poden optimitzar més les consultes mitjançant la creació de subcubs que permeten la manipulació d'un conjunt més restringit de dades i, per tant, n'optimitzen l'accés.

2.6.5. Mercat

Al principi de la dècada dels anys noranta del segle XX, com a alternativa als sistemes d'informació per a executius (EIS), van aparèixer els primers sistemes OLAP. Eren sistemes basats en models multidimensionals que proporcionaven

una tecnologia més eficient per a l'anàlisi de dades. Actualment són una peça clau per a aplicacions de l'àmbit de la intel·ligència empresarial (*business intelligence*).

Hi ha un gran nombre de sistemes basats en magatzems de dades que, actualment, permeten transformar les dades de l'organització en coneixement útil per a obtenir avantatge competitiu. Com a productes líder potents, pel que fa a magatzems de dades, destaquen l'Oracle –un magatzem construït amb arquitectura ROLAP– i SQL Server –un exemple de MOLAP. Tanmateix, hi ha altres sistemes reconeguts, com Sybase, Illuminate Solutions, Kognitio, Ingres, Teradata, GreenPlum, 1010data, etc. També destaquen altres sistemes de codi obert, com Pentaho, de tipus ROLAP, i Palo, de tipus MOLAP.

2.6.6. Problemàtica

Els problemes més rellevants que presenten aquestes bases de dades són els següents:

- No són òptimes per a entorns amb dades no estructurades.
- Solen quedar obsoletes ràpidament i tenen un cost molt elevat, atès que tenen un temps de vida curt.
- De vegades, la informació que retorna una consulta no és prou bona per a prendre decisions, cosa que representa una pèrdua per a l'organització.
- Hi ha un llindar en què és difícil separar funcionalitats d'un sistema operacional i un magatzem de dades i, per tant, seria convenient determinar les funcionalitats que es poden reaprofitar i les que s'han d'implementar.

2.7. Bases de dades de columnes

Només hi ha dues opcions obvies per a emmagatzemar una taula relacional (que té dues dimensions) en una unitat d'emmagatzematge (que consta d'una sola dimensió): emmagatzemar la taula fila per fila, o columna per columna.

Els SGBD orientats a columnes són sistemes en què les dades s'emmagatzemen per columnes, i no per files. És a dir, en comptes d'emmagatzemar files senceres d'una taula, l'una darrera l'altra, com fan els sistemes relacionals clàssics, emmagatzemen cada columna separadament. Per tant, les dades de cada columna s'emmagatzemen de manera contínua, i normalment comprimides i densament empaquetades[3].

(3)L'expressió "densament empaquetades" prové del terme anglès *densely packed*, que significa codificar de manera diferent la mateixa informació fent servir menys bits. Per exemple, utilitzant informació densament empaquetada, en comptes de definir un dígit (0..9) i, per tant, deu possibles valors amb 4 bits, podríem definir-ne 16 (0..15).

2.7.1. Avantatges respecte del model relacional

Els avantatges principals dels SGBD orientats a columnes són els següents:

1) Són més eficients per a processos analítics que siguin intensius en lectures, ja que requereixen menys accessos a disc per a recuperar les dades d'interès. El nombre inferior d'accessos està condicionat pel fet que les dades de les columnes s'emmagatzemen comprimides i perquè tan sols s'obtindran les dades en les columnes implicades en la consulta.

2) Permeten representar més informació en menys espai a causa de la compressió de les dades.

2.7.2. Aplicacions d'ús

Les aplicacions més habituals d'aquestes bases de dades són els sistemes d'informació que fan un ús intensiu d'operacions de lectura, com, per exemple, els sistemes d'informació analítics (sistemes OLAP). Per tant, seran útils per a entorns de magatzem de dades, de mineria de dades, d'intel·ligència empresarial i d'aparador de dades, i en alguns casos també poden ser d'utilitat en la gestió de dades científiques.

2.7.3. Peculiaritats d'emmagatzematge

Cada columna s'emmagatzema de manera contínua en un espai separat del disc. Per millorar l'eficiència en la lectura de les columnes, els valors s'emmagatzemen densament empaquetats i comprimits. Normalment, la compressió en aquest tipus de SGBD és més gran, ja que, com que les dades estan disposades en columnes, les dades que s'emmagatzemen juntes pertanyen al mateix domini i, per tant, tenen menys entropia.

Alguns sistemes, com Vertica, ordenen els valors de les columnes abans de comprimir les dades, cosa que permet un grau de compressió molt més alt. Això és molt útil en columnes que tenen associat un domini discret amb pocs valors, com, per exemple, gènere (masculí i femení) o edat (de 0 a 150 anys, com a màxim).

Tanmateix, no tots els SGBD orientats a columnes utilitzen les mateixes estratègies d'emmagatzematge. Algunes bases de dades simulen l'orientació a columnes fent servir un model relacional clàssic, en el qual creen taules diferents (una per cada columna de la taula que no sigui clau primària). Cadascuna d'aquestes taules ha de contenir dues columnes, l'una amb un identificador (que pot ser la clau primària) per a identificar la fila, i l'altra, per a representar la columna en qüestió.

2.7.4. Accés a dades

El fet que les dades estiguin emmagatzemades en columnes implica que cada vegada que es vulgui recuperar un registre complet de la base de dades caldrà obtenir-ne totes les columnes i, després, ajuntar-les mitjançant una unió per formar el registre. Aquesta operació encara es complica més si tenim en compte que les dades solen estar comprimides i que abans s'hauran de descomprimir. No obstant això, alguns SGBD permeten executar certes operacions directament sobre les dades comprimides, de manera que s'evita descomprimir les dades i es millora el temps de resposta.

Les bases de dades orientades a columnes fan servir diferents tipus d'índexs per a millorar l'eficiència (arbres B+, mapes de bits, etc.). D'altra banda, els canvis d'aquestes bases de dades afecten sobretot la manera d'emmagatzemar la informació, però no el model de base de dades utilitzat. Per aquest motiu, l'accés a la base de dades es fa mitjançant SQL estàndard, com en qualsevol sistema relacional.

2.7.5. Mercat

L'origen de l'organització de les dades per columnes apareix en la dècada dels setanta del segle XX. No obstant això, no és fins l'any 2000 que apareixen els primers SGBD comercials orientats a columnes i la recerca en aquest tema esdevé més intensa. Actualment, hi ha un gran nombre de productes orientats a columnes, tant comercials (Vertica, Sybase IQ, SenSage i ParAccel) con de programari lliure (per exemple, MonetDB o C-Store).

2.7.6. Problemàtica

Els problemes principals d'aquest tipus de SGBD són els següents:

- **Operacions d'escriptura**: les dades que cal inserir s'han de separar segons els atributs que tenen, i cadascun s'ha d'emmagatzemar de manera separada. A més, el fet que algunes bases de dades ordenin les columnes i emmagatzemin la informació densament empaquetada pot provocar retards ocasionals.

- **Reconstrucció de files**: una fila s'emmagatzema en diferents llocs del disc; per tant, a l'hora de retornar una fila, s'ha de fer la combinació de totes les columnes que conté. Això és especialment problemàtic, ja que les API actuals (ODBC i JDBC) accedeixen als resultats fila per fila, en comptes de columna per columna.

Aquests obstacles fan que, per a sistemes d'informació transaccionals (OLTP), sigui més efectiu fer servir bases de dades amb sistemes d'emmagatzematge orientats a files.

2.8. Bases de dades documentals

Les bases de dades documentals permeten emmagatzemar i recuperar documents de tota classe. Aquestes bases de dades treballen amb documents, en comptes de fer-ho amb registres, i solen estar implementades sobre una BDR o una BDOO.

Generalment, s'han utilitzat per a emmagatzemar documents textuals i, darrerament, per a emmagatzemar documents XML.

2.8.1. Avantatges respecte del model relacional

Els avantatges més destacats de les bases de dades documentals són els següents:

1) Permeten emmagatzemar documents amb una estructura diferent, com si fossin del mateix tipus. Això permet estalviar espai quan els documents de la mateixa classe tenen estructures molt heterogènies.

2) Proporcionen una manera més natural de gestionar (consultar i modificar) documents.

3) Utilitzen una interfície força més directa amb documents web i XML.

2.8.2. Aplicacions d'ús

L'ús d'aquestes bases de dades està orientat al tractament de grans volums d'informació textual o poc estructurada. Per tant, han d'oferir operadors que permetin fer cerques més potents sobre el contingut dels documents, que normalment serà text. Per exemple, una base de dades d'aquest tipus hauria de permetre realitzar la consulta següent: "retorna tots els anuncis (documents) que continguin la paraula *venc* prop de *cotxe*, i que també continguin la paraula *ganga*".

2.8.3. Peculiaritats d'emmagatzematge

En una base de dades documental, cada registre es correspon amb un document, o amb la seva referència. Pot ser un document sencer (per exemple, un llibre) o un document que és part d'un altre (per exemple, un capítol de llibre).

Per cada registre (document), hi ha un conjunt de camps que en descriuen el contingut i que s'anomenen *metadades*. Són exemples de metadades els autors, el títol del document, l'ISBN i el resum. Segons el tipus de document, tindrem unes metadades disponibles o unes altres.

Els sistemes d'indexació clàssics dels sistemes relacionals (dispersió i arbres B +) no són gaire efectius per a indexar contingut textual. Per a això, aquestes bases de dades solen fer servir el que anomenem *índexs de text complet* (actualment, els índexs més utilitzats per a indexar text són els índexs invertits). Aquests índexs mantenen una correspondència entre les paraules contingudes en els documents i els documents on apareixen. Normalment, es fonamenten en diccionaris o tesaurus a fi d'augmentar-ne l'eficiència, cosa que permet trobar conceptes amb independència de la representació corresponent (temps verbal, grau o gènere que utilitzin). Fer servir diccionaris o tesaurus permetria que, si cerquem la paraula *compro*, el SGBD retorni les ocurrències de totes les conjugacions del verb *comprar* (compro, comprar, compraré, compraria...) en els documents de la base de dades.

2.8.4. Accés a dades

L'accés als documents, i per extensió a les dades que contenen, d'una base de dades documental es realitza mitjançant cerques a les metadades, que poden estar indexades en índexs tradicionals, o al contingut, indexat normalment en índexs de text complet.

En les cerques a continguts textuals de documents, es poden utilitzar els operadors booleans (AND, OR, NOT), operadors sintàctics i operadors de comodí. Normalment, els operadors sintàctics permeten establir condicions relatives a la proximitat (una paraula es troba a prop d'una altra) o l'adjacència de paraules (una paraula apareix juntament amb una altra). Els operadors de comodí permeten definir una expressió regular que indica quin patró volem que segueixi el text; per exemple, "Retornar tots els documents que continguin un codi postal de Barcelona (cinc caràcters, en què els dos primers han de ser *08*)".

Normalment, les bases de dades documentals també permeten fer cerques per paraules clau i per jerarquies, entre d'altres. La cerca per jerarquies permet retornar totes les ocurrències de paraules que representin un concepte més concret que el cercat. Per exemple, si demanem les ocurrències de "persona" i les concrecions corresponents, aleshores el sistema retornaria els documents on apareix "persona", "home" i "dona", perquè *home* i *dona* són concrecions de persona.

2.8.5. Mercat

En la dècada dels noranta del segle XX van sorgir alguns sistemes d'informació documentals que utilitzaven BDR o BDOO com a base per a implementar les funcionalitats que hem descrit. Amb els anys, però, els productes relacionals van començar a integrar funcionalitats de cerca de text complet i van oferir nous tipus de dades que permetien emmagatzemar grans blocs d'informació, i per tant fitxers, en els camps de la base de dades. Molt probablement això va

fer que alguns sistemes de gestió de bases de dades documentals fossin discontinus. Actualment, la majoria de BDR pot realitzar amb èxit cerques textuals del contingut de la seva base de dades.

D'entre les bases de dades documentals que hi ha actualment, destaquem FileMaker Pro, Knosis, CDS/ISIS i DB/Text d'INMAGIC. D'altra banda, hi ha un tipus de bases de dades molt present avui que es pot veure com una extensió de les bases de dades documentals; es tracta de les bases de dades XML. Tractarem d'aquestes bases de dades d'una manera separada, atès que tenen una importància cabdal.

2.8.6. Problemàtica

La problemàtica principal d'aquestes bases de dades és que estan poc pensades per a emmagatzemar informació molt ben estructurada. Per tant, són poc eficients a l'hora d'emmagatzemar, processar i consultar aquest tipus d'informació.

2.9. Bases de dades XML

Les bases de dades XML permeten emmagatzemar i recuperar dades en format XML[4]. Es poden classificar en **bases de dades XML natives** (NXD, *Native Xml Database*) i **bases de dades adaptades a XML** (*XML-enabled*). Les bases de dades natives emmagatzemen la informació directament en XML, i utilitzen documents XML com a unitat lògica d'emmagatzematge. Les bases de dades adaptades a XML reutilitzen un altre tipus de SGBD (normalment relacional) i proporcionen funcionalitats per llegir i escriure dades en format XML d'una manera fàcil.

(4)XML són les sigles de *extensible markup language*, i designen un metallenguatge creat pel World Wide Web Consortium (W3C) que permet crear llenguatges d'etiquetes específics. Actualment, el format XML és un dels estàndards més utilitzats per a intercanviar informació.

A partir d'ara, quan parlem de base de dades XML ens referirem només a bases de dades natives. Si en algun cas té sentit parlar de bases de dades adaptades a XML ho farem constar explícitament.

2.9.1. Avantatges respecte del model relacional

Els avantatges més importants de les bases de dades XML són els següents:

1) Tractament més acurat de dades semiestructurades. Les bases de dades XML permeten un tractament més eficient de dades que tenen una estructura poc rígida o variable en el temps, com ara el registre de salut d'un pacient. La informació d'aquest registre variaria depenent de la malaltia que tingui, de l'estat de la malaltia i de l'evolució d'aquesta. El tractament d'aquestes dades en un model relacional implicaria un nombre de columnes molt elevat, que, en la majoria dels casos, tindrien un gran nombre de valors nuls.

2) Més eficiència en uns tipus de consultes determinats. Els casos més comuns són:

- **Consultes recursives**: els llenguatges d'interrogació d'aquestes bases de dades tenen un tractament molt potent de la recursivitat. Per tant, si necessitem fer consultes que explotin la recursivitat de l'esquema XML, probablement, una base de dades XML serà més eficient. No obstant això, si les consultes sobre les dades són complexes, però no exploten la recursivitat del model, són més convenients els SGBDR, ja que l'SQL té més funcions de manipulació que XPath.

- **Consultes en el document XML**: algunes estratègies d'emmagatzematge de BDXML emmagatzemen físicament els documents XML sencers. Això permet que, quan se cerca pel concepte representat en el document, l'SGBD pugui retornar el resultat fent una sola consulta a disc i estalviant-se de fer cap unió. Considerem un exemple típic de factures, línies de factures, productes, clients, etc. En cas de voler obtenir totes les dades d'una factura, l'SGBD tan sols hauria de fer una lectura en el disc per obtenir el document XML que representa la factura.

3) Ús de format XML. Quan es volen obtenir (o proveir) les dades en format XML, és més natural l'ús de bases de dades natives no solament perquè l'entrada i la sortida es fa en XML, sinó perquè els llenguatges de consulta estan pensats per a consultar fitxers XML i, per tant, faciliten la feina al programador.

2.9.2. Aplicacions d'ús

Les bases de dades XML s'utilitzen per a tractar documents XML o esquemes semiestructurats.

2.9.3. Peculiaritats d'emmagatzematge

Les bases de dades XML poden emmagatzemar els documents XML en text o en models. Les basades en text emmagatzemen els documents XML d'una manera textual, com un fitxer en un sistema de fitxers o com un camp de tipus BLOB dintre d'una BDR. D'altra banda, les basades en models emmagatzemen les dades dels documents XML en un model intern de document que és una representació, prou genèrica, de l'estructura del document XML. La manera com s'emmagatzema aquest document dependrà de la base de dades de base que s'utilitzi. Un model molt emprat és el **document object model** (DOM), que permet representar les dades del document XML com una jerarquia d'objectes relacionats entre si, en què els objectes poden ser de diversos tipus: elements per a representar les etiquetes XML, atributs per a representar els atributs de les etiquetes, blocs de text, etc.

Enllaç d'interès

El *document object model* és un esquema conceptual proposat pel W3C que permet representar documents XML i HTML.

En trobareu més informació a l'adreça d'Internet següent: http://www.w3.org/DOM/

Respecte a l'eficiència, les bases de dades que emmagatzemen les dades en forma textual utilitzen índexs de text complet per al contingut dels fitxers XML i, per tant, poden recuperar documents o fragments de documents en format textual molt eficientment. En les bases de dades basades en models, l'eficiència depèn de com d'optimitzat estigui el model que s'utilitza i de la base de dades que es fa servir de base. Normalment, aquestes bases de dades són més eficients en retornar documents emmagatzemats com a models (per exemple, com un arbre DOM).

La majoria de bases de dades XML ofereixen el concepte de col·lecció. Es tracta d'una agrupació lògica de documents XML i té un paper homòleg a les taules en BDR. En algunes bases de dades es permet definir jerarquies de col·leccions, agrupant les col·leccions segons el tipus. Per exemple, en el cas que volguéssim portar un control de la música que tenim, es podria crear una col·lecció anomenada *cançó* per a emmagatzemar el conjunt de documents XML que descriu les nostres cançons. En cas que l'SGBD permetés definir jerarquies de col·leccions, podríem especialitzar *cançó* en les col·leccions *cançó pop*, *cançó rock*, *cançó blues*, etc.

Normalment, les bases de dades XML ofereixen tres tipus d'índexs diferents:

- **Per valor**. Els índexs per valor són els utilitzats en BDR, que s'implementen normalment utilitzant arbres B+ i permeten indexar els valors de les etiquetes XML i dels atributs XML corresponents.

- **Estructurals**. Els índexs estructurals permeten indexar les etiquetes XML i els atributs pertinents segons de quin tipus siguin.

- **De text complet**. Els índexs de text complet permeten indexar el text contingut en els documents XML.

Utilitzant aquests índexs, es pot donar suport a consultes del tipus "retornar totes les etiquetes XML que tinguin un atribut amb el valor *Barcelona*" utilitzant índexs per valor, "retornar totes les etiquetes de tipus Ciutat" fent servir índexs estructurals, o "retornar les etiquetes de tipus Ciutat que continguin el text *Restaurant* i el valor de l'atribut *província* sigui igual a *Barcelona*" emprant els tres tipus d'índexs.

Igual que en les BDR, la qualitat d'una BD XML depèn de la qualitat del disseny de l'estructura dels documents XML que s'han d'emmagatzemar i de les col·leccions. En conseqüència, el concepte de normalització és també rellevant en BD XML i té els mateixos objectius que en BDR: la necessitat de dissenyar documents XML de manera que s'eviti la redundància de dades. Malauradament, les diferències entre les BD XML i les BDR (com, per exemple, l'existència de valors multivalents) fan que els algoritmes de normalització en BDR no hi

siguin aplicables. La normalització de bases de dades XML és encara un concepte nou i, per tant, poc madur. En conseqüència, no hi ha cap mètode sistemàtic que permeti normalitzar bases de dades XML.

Un altre tema que encara no està resolt en les BD XML és la integritat referencial. En el context de BDR, les restriccions d'integritat referencial garanteixen que els valors de les claus foranes utilitzades existeixen com a claus primàries d'altres taules. En el context de XML, la integritat referencial hauria de garantir que els enllaços o els punters a altres documents (o fragments) XML adrecen a documents vàlids.

En un document XML es poden definir punters que adrecin a elements XML del mateix document o d'altres documents. En el primer cas direm que es tracta d'integritat referencial interna, mentre que, quan s'apunta a altres documents, direm que és integritat referencial externa. La majoria de BD XML permeten comprovar parcialment la integritat referencial interna quan s'utilitzen mecanismes estàndard XML per a definir els punters, com, per exemple, els atributs ID/IDREF o camps de tipus clau (*key* o *keyref*). Diem *parcialment*, perquè la majoria de bases de dades tan sols comproven la integritat referencial interna quan s'afegeixen nous documents a la base de dades. Si en algun moment els usuaris modifiquen o esborren nodes concrets d'un document, aleshores la comprovació de les restriccions d'integritat referencial no s'efectuen i, per tant, la base de dades pot quedar en un estat inconsistent. Actualment, la integritat referencial externa no es garanteix, ja que és molt difícil comprovar que el recurs apuntat (que es pot trobar al web) existeix i és vàlid. A més, no hi ha cap manera de saber si un element de la web apuntat per la nostra base de dades es modifica o s'elimina d'Internet.

2.9.4. Accés a dades

Gairebé totes les bases de dades XML suporten transaccions. No obstant això, les suporten en un nivell de granularitat massa gran: el document. Això implica que el grau de concurrència al qual es pot arribar en una base de dades XML és relativament baix. Aquesta limitació es pot minimitzar depenent del que el dissenyador entengui per *document*, ja que, com més informació contingui un document XML, més informació es bloquejarà en cada operació. Idealment, seria interessant fer bloquejos a nivell d'etiqueta XML, però això planteja un problema que encara no ha estat resolt: bloquejar una etiqueta implica bloquejar-ne l'etiqueta pare (l'etiqueta que la conté). Fixeu-vos que aquesta situació és recursiva; cada bloqueig implica bloquejar les etiquetes que la contenen fins a arribar al node arrel del document. Per exemple, considerem un document XML que representi la informació d'una factura. Bloquejar una línia de factura implicaria bloquejar la factura mateixa, ja que la línia de factura està inclosa dintre de la factura. Això té sentit, perquè, si no és així, un altre usuari podria eliminar la factura i, al seu torn, tot el contingut, incloent-hi la línia

de factura que es modifica. Actualment hi ha algunes solucions teòriques a aquest problema, però malauradament no estan prou madures per a ser implementades en els sistemes comercials.

Hi ha diferents llenguatges que permeten consultar dades emmagatzemades en format XML: Lorel, Quilt, XPath, XQL, XQuery, etc. No obstant això, els més utilitzats avui dia són el XPath i el XQuery.

1) Xpath: permet navegar i seleccionar parts d'un document XML a partir de rutes de localització i expressions. Les rutes de localització s'utilitzen per a navegar pel document XML d'una manera similar a la navegació en un sistema de fitxers. Les expressions s'avaluen sobre un node del document XML, que s'anomena *node context*, i retornen un objecte. El resultat d'una expressió pot ser de tipus booleà, numèric, cadena de caràcters o *node-set*. Un element de tipus *node-set* conté un conjunt de nodes (etiquetes) que satisfan la condició especificada.

Per exemple, donat el fragment XML següent (menu.xml):

```
<menu>
  <breakfast time="7:00 am">
    <food price="1">toasts</food>
    <food price="1">eggs</food>
    <food price="3">beans</food>
    <drink price="1">Orange juice</drink>
  </breakfast>
  <lunch time="12:00 pm">
    <food price="3">beans</food>
    <food price="1">bread</food>
    <drink price="2">Coke</drink>
  </lunch>
  <dinner time="8:00 pm">
    <food price="3">salad</food>
    <drink price="1">water</drink>
  </dinner>
</menu>
```

l'expressió `/menu/dinner/food` retornaria tots els nodes de tipus *food* que estan compresos dins un node de tipus *dinner*. En el cas d'exemple, l'ítem `<food price="3">salad</food>`.

D'altra banda, l'expressió `/menu/breakfast@time` retornaria l'atribut *time* de l'etiqueta *breakfast* compresa dins l'etiqueta *menu*. En el cas d'exemple, el resultat seria `"7:00 am"`.

2) XQuery: és l'evolució de XPath i ofereix una representació més propera al llenguatge natural. XQuery és un llenguatge funcional en què qualsevol consulta és una expressió. XQuery inclou XPath com a mètode per a seleccionar nodes i l'estén afegint funcionalitats de transformació. Per a referir-se a la sintaxi de XQuery, s'utilitzen les sigles FLWOR (pronunciat com *flower*, en anglès). Aquestes sigles provenen de les possibles clàusules de XQuery: FOR, LET, WHERE, ORDER BY i RETURN.

Per exemple, la consulta:

```
FOR $x IN doc("menu.xml")/menu/breakfast/food
WHERE $x@price<3
ORDER BY $x
RETURN $x
```

retornaria els menjars que es poden demanar d'esmorzar que costen menys de tres euros i ordenats alfabèticament pel nom.

XPath i XQuery tan sols permeten consultar dades. Si volem modificar les dades dels documents XML emmagatzemats en la base de dades, hem de fer servir altres tipus de llenguatges; per exemple, XUpdate o algunes extensions de XQuery. XUpdate és un llenguatge proposat per la iniciativa XML:DB[5] i està implementat en un gran nombre de SGBD XML. XUpdate utilitza XPath per a identificar el conjunt de nodes que cal modificar o eliminar, o per a definir el punt d'ancoratge des d'on s'ha d'afegir o eliminar un conjunt de nodes.

(5)XML:DB és una iniciativa que treballa per definir especificacions per a bases de dades XML, amb l'objectiu que aquestes especificacions siguin portades al mercat de bases de dades XML per facilitar-ne l'aprenentatge i millorar-ne l'adopció.

La majoria de SGBD XML suporten una API semblant a JDBC que permet accedir a les dades de la base de dades des de programes. Al principi, els SGBD feien diferents implementacions d'aquestes API, a causa de la falta d'estàndards. No obstant això, a partir del 2009, la comunitat Java va definir la XQuery API for Java Specification (XQJ), una API comuna que permet que una aplicació realitzi consultes sobre una base de dades mitjançant el llenguatge XQuery 1.0.

2.9.5. Mercat

En els últims anys han aparegut nombroses bases de dades XML, tant natives (eXist, Tamino, MonetDB, Xindice, etc.) com adaptades a XML (Oracle XML, DB2, PostgreSQL, MySQL, etc.). També podem trobar moltes utilitats que ens permeten un ús més senzill de documents XML i una interacció més bona amb les bases de dades. Aquestes utilitats inclouen, entre d'altres, motors que permeten realitzar consultes XQuery sobre documents XML, *wrappers* que tracten documents XML com si fossin taules relacionals i editors que possibiliten editar documents XML fàcilment. Tenint en compte l'evolució ràpida d'aquestes eines i el nombre d'eines noves que apareixen cada any, hem preferit no fer una llista de totes elles amb les funcionalitats corresponents.

Enllaç d'interès

Podeu trobar una llista actualitzada de les bases de dades XML disponibles, i les eines relacionades, a la web de Ronald Bourret, expert reconegut en el món de les bases de dades XML. L'adreça és la següent: http://www.rpbourret.com/xml/XMLDatabaseProds.htm.

2.9.6. Problemàtica

Els problemes més importants que presenten aquest tipus de bases de dades són els següents:

- El concepte d'integritat referencial no esta prou ben definit en el context XML. Això fa que moltes bases de dades derivin el control referencial de les dades a les aplicacions. Si la integritat referencial és un requisit que cal tenir en compte, una BD XML nativa no és una opció.

- Les estratègies d'emmagatzematge que utilitzen. Normalment, els documents XML s'emmagatzemen sencers i de manera consecutiva; això fa que,

en cas que una consulta requereixi obtenir les dades en un ordre diferent de com han estat emmagatzemades físicament, tingui una eficiència més baixa que en els sistemes relacionals.

- Tan sols retornen dades en format XML. Si les aplicacions que fan servir la base de dades necessiten les dades en un format diferent de l'XML, aquestes seran les encarregades de convertir-les en el format volgut. Això fa poc recomanable l'ús d'aquestes bases de dades quan hi ha diferents programes que accedeixen a les dades utilitzant altres protocols, com, per exemple, el JDBC.

- El concepte de normalització és necessari, però encara està massa immadur.

- Falta estandardització en els llenguatges que s'utilitzen per a accedir a les dades i gestionar-les. Tot i que el llenguatge XQuery està força acceptat, encara queda força feina per fer per a incloure-hi funcionalitats que permetin inserir, esborrar i modificar dades XML. També caldria definir llenguatges estàndard que permetin operacions del tipus DDL (*data definition language*).

2.10. Bases de dades incrustades

Els SGBD incrustats (*embedded*) són sistemes que s'integren amb les aplicacions que utilitzen la base de dades per a amagar la base de dades a l'usuari final de l'aplicació. Aquestes bases de dades formen un tot amb les aplicacions que les empren i, per tant, són distribuïdes amb aquestes. Això fa que no necessitin un administrador, ja que, per defecte, una base de dades incrustada s'instal·larà juntament amb l'aplicació que la conté i ja estarà configurada correctament i optimitzada d'acord amb els requisits de l'aplicació.

Normalment, aquests SGBD suporten interfícies de programació diferents (SQL, propietàries i API natives), arquitectures diverses de base de dades (client/servidor i en procés), models distints d'emmagatzematge (en disc, en memòria i combinacions de tots dos) i models diferents de base de dades (relacional, orientat a objecte, etc.).

2.10.1. Avantatges respecte del model relacional

Els avantatges més rellevants que tenen respecte de les bases de dades relacionals són els següents:

1) Permeten més integració amb les aplicacions que les utilitzen.

2) Tenen una instal·lació i una configuració més fàcils.

3) No necessiten una persona que creï la base de dades i la mantingui, ja que es crea i es configura amb la instal·lació del programa que la utilitza.

2.10.2. Aplicacions d'ús

L'ús d'aquestes bases de dades està orientat a aplicacions senzilles en què no té sentit crear i configurar una base de dades a part. Aquestes SGBD també són molt pràctiques quan les aplicacions que les empren han de ser distribuïdes en una gran quantitat i, per tant, es vol facilitar el procés d'instal·lació i reduir els costos de compra, d'instal·lació i de manteniment que tindria una base de dades tradicional.

Un altre àmbit d'aplicació possible són els dispositius mòbils (PDA, tauletes gràfiques o telèfons mòbils) que han de poder funcionar sempre, fins i tot quan no tinguin connectivitat de xarxa. Si no es pot garantir l'accés a una base de dades remota, l'ús d'aquest tipus de SGBD pot ser convenient.

2.10.3. Peculiaritats d'emmagatzematge

Tot i que, a primer cop d'ull, podria semblar que les funcionalitats d'aquestes dades haurien de ser molt inferiors que les bases de dades no incrustades, el cert és que actualment les bases de dades incrustades estan molt consolidades i ofereixen gairebé totes les funcionalitats que podem trobar en les bases de dades tradicionals, incloent-hi uns temps de resposta força bons. La majoria dels SGBD incrustats són transaccionals, comproven les restriccions d'integritat referencials, incorporen índexs B+, índexs de text, i índexs arbre R. D'altra banda, hi ha un gran nombre de funcionalitats addicionals que ofereixen aquests sistemes, com ara la possibilitat de carregar la base de dades en memòria i treballar directament des de memòria, extensions geogràfiques que permeten emmagatzemar, gestionar i consultar dades geogràfiques juntament amb dades alfanumèriques, etc.

2.10.4. Accés a dades

Com que aquestes bases de dades només són accessibles des d'un programa, tenen un mecanisme d'accés mitjançant API. La majoria d'elles implementen API en un format estàndard, com ara JDBC, ODBC o ADO.NET. No obstant això, també n'hi ha d'altres que implementen API propietàries en altres llenguatges, com, per exemple, en CLIPPER[6] o Delphi.

(6)Malgrat que CLIPPER és un llenguatge de programació de fa dues dècades, molts programes escrits en aquest llenguatge encara funcionen. Oferir suport per a aquest llenguatge permet que aquests programes s'actualitzin a un cost molt baix.

2.10.5. Mercat

Actualment hi ha molts sistemes gestors de bases de dades incrustats en el mercat. N'hi ha una gran diversitat i es diferencien en el model de base de dades utilitzat (relacional, orientat a objectes...), en les funcionalitats que ofereixen (emmagatzematge en memòria, consultes geogràfiques...) i en el tipus de llicència (gratuïta, en codi obert o de pagament).

La majoria de bases de dades incrustades segueixen el model relacional. En podem destacar les privatives: Advantage Database Server de Sybase, Empress Embedded Database, IBM Informix Dynamic Server, Interbase, SQL Server Compact de Microsoft i ScimoreDB. Respecte als sistemes de codi obert, destaquem Firebird o HSQLDB (*hyper structured query language database*), que està escrita en Java i permet que les taules s'emmagatzemin en memòria i en disc. Un altre producte semblant és RDM Embedded, que permet tenir en una mateixa base de dades diferents taules relacionades, independentment que estiguin emmagatzemades en memòria o en disc. InfinityDB és una base de dades Java incrustada que s'inclou en l'InfinityDB Developer's kit i està especialment dissenyada per a ser usada en dispositius mòbils.

Potser un dels productes incrustats més coneguts és SQLite. Es tracta d'un SGBD relacional que ocupa molt poc espai, és força eficient i es pot integrar completament en un programa. SQLite implementa la majoria de les característiques estàndard de SQL, tot i que no suporta vistes actualitzables i només té suport parcial a disparadors. Actualment, aquest SGBD es fa servir en nombrosos programes que utilitzem diàriament, com, per exemple, en el navegador web Firefox, en què s'utilitza per a gestionar les adreces d'interès (*bookmarks*), les galetes (*cookies*) i altres elements. SQLite té versions per a la majoria dels dispositius mòbils actuals, com ara iPhone, iPad i Android; tanmateix, les característiques de l'SGBD en aquests dispositius són més limitades que en les versions d'ordinador de sobretaula; per exemple, no contenen mecanismes d'extensió, ni índexs arbre R ni suport per a la gestió de dades geogràfiques.

2.10.6. Problemàtica

Aquests SGBD han de ser molt robustos, ja que no tenen la figura d'administrador, amb la qual cosa la base de dades és invisible des de fora de l'aplicació i qualsevol errada serà més difícil de solucionar. Per tant, si hi ha res que deixi de funcionar, no es podrà arreglar des de fora de l'aplicació com es faria amb una base de dades normal. D'altra banda, aquests SGBD s'han d'adaptar a una gran variabilitat de requisits tècnics, atès que han de poder ser instal·lats en un gran nombre d'entorns diferents: un ordinador personal, un portàtil, una PDA o una tauleta gràfica, per exemple, i en programes escrits en llenguatges de programació diferents.

A l'hora de fer servir una base de dades incrustada, s'ha de tenir molt en compte quines funcionalitats ofereix en el dispositiu de destinació. En el cas de dispositius mòbils, les bases de dades incrustades tenen versions amb menys funcionalitats, que poden fer-ne inaplicable l'ús segons els requisits del sistema que s'ha implementar.

2.11. Resum

A continuació es mostra una taula que recull les aportacions dels models de bases de dades que hem presentat en aquest apartat, les funcionalitats principals que afegeixen respecte del model relacional i, també, les limitacions.

	Què aporta respecte del model relacional	Funcionalitats	Limitacions
BDOO	Informació complexa. Herència.	Permet definir dades complexes. Permet definir noves operacions lligades als tipus. Utilitza llenguatges propis per a l'accés i la gestió de les dades i els esquemes: OQL i ODL.	L'optimització de consultes encara no està resolta.
BDD	Mecanismes d'inferència.	Permet definir fets i regles d'inferència. Permet inferir nova informació a partir de l'existent utilitzant les regles d'inferència. Suporta la recursivitat. Utilitza lògica de predicats per a accedir a la base de dades i definir-ne l'estructura.	L'optimització de consultes encara no està resolta. Limitacions dels llenguatges de predicats utilitzats.
BDT	Informació temporal. Evolució de la informació en el temps.	Permet representar informació des d'una perspectiva històrica o d'evolució de la informació. Incorpora nous tipus de dades per oferir suport a la informació temporal i operadors sobre aquests nous tipus. Permet definir restriccions d'integritat relatives al temps. Estén SQL per gestionar la informació temporal: TSQL2.	L'optimització de consultes encara no està resolta. Hi ha diversos problemes en el processament de consultes.
BD geogràfica	Informació espacial.	Incorpora nous tipus de dades per oferir suport a la informació geogràfica i operadors sobre aquests nous tipus. Permet definir índexs per optimitzar la cerca d'informació geogràfica. Permet importar dades geogràfiques creades amb altres programes SIG d'escriptori. Estén SQL per gestionar la informació geogràfica (estàndard *simple feature type*).	Tot i que hi ha un estàndard que diu com s'ha d'estendre SQL per tractar amb la informació geogràfica, poques bases de dades el fan servir al cent per cent.
BD distribuïda	Informació distribuïda entre diversos nodes.	Permet distribuir dades d'acord amb l'estructura organitzativa. Incrementa la fiabilitat i la disponibilitat de les dades i el rendiment del sistema. Facilita l'extensió modular del sistema.	No hi ha estàndards, ni eines ni metodologies que donin suport als usuaris en la conversió d'un sistema centralitzat a un sistema distribuït.

	Què aporta respecte del model relacional	Funcionalitats	Limitacions
BD analítica	Suport a la presa de decisions.	Està orientada a grans volums de dades. Permet representar informació útil per millorar l'estratègia empresarial. Suporta la integració de diverses fonts de dades. Millora el rendiment quan executa operacions de lectura. No segueix les formes normals.	Té un cost elevat i una vida molt curta. Es crea un model diferent per a cada organització i aspecte que s'estudia, per tant, és poc reaprofitable. El rendiment és menor quan s'executen operacions d'actualització, d'eliminació o d'inserció.
BD orientada a columnes	Suport a la presa de decisions. Rendiment superior en aplicacions que facin un ús intensiu d'operacions de lectura.	Emmagatzema les dades en disc agrupades per columnes, i no per files. Emmagatzema les dades d'una manera comprimida.	Rendiment inferior quan hi ha operacions d'escriptura.
BD documental	Suport en la gestió de documents.	Permet emmagatzemar i consultar documents. Permet indexar els documents per contingut, amb la qual cosa milloren les cerques per contingut.	Rendiment inferior en esquemes molt ben estructurats.
BD XML	Suport en la gestió de documents i dades en format XML.	Permet emmagatzemar i consultar documents XML. Permet indexar els documents per contingut, amb la qual cosa es milloren les cerques per contingut. Permet emmagatzemar directament informació XML sense haver de transformar-la a cap altre format. Permet recuperar la informació de la base de dades directament en format XML. És un suport més bo per a dades poc estructurades.	Rendiment inferior en esquemes molt ben estructurats. Manca d'estàndards en els llenguatges d'accés a la base de dades, sobretot en els llenguatges utilitzats per a modificar, esborrar i inserir dades a la base de dades. Problemes en la comprovació de la integritat referencial. Els criteris de disseny estan totalment en mans del dissenyador, ja que la normalització d'aquest tipus de base de dades està poc madura.
BD incrustada	Integració de la base de dades juntament amb el programa. Reducció de costos: no necessita administradors de base de dades, ja que el programa associat té aquesta funció.	Permet integrar una base de dades directament amb un programa. Quan s'instal·la el programa, també s'instal·la la base de dades. No cal permetre l'administració de la base de dades des de fora del programa. Implementa API propietàries en llenguatges diferents per facilitar-ne l'adopció.	Rendiment i funcionalitats més baixos que les bases de dades tradicionals. La manca d'administració des de fora del programa que l'inclou pot ser un problema si la base de dades falla i no es pot recuperar. Les funcionalitats de la base de dades depenen de la versió que utilitzem (algunes versions per a determinades plataformes són menys completes).

A partir d'aquesta taula, podem veure que l'elecció de la base de dades dependrà del tipus d'informació que s'hagi de tractar, de l'aplicació (o aplicacions) que l'hagi d'utilitzar i d'altres requisits.

Pel que fa als tipus de dades, si treballem amb dades temporals o històriques seran adients les bases de dades temporals; si tractem amb informació geogràfica, ens convindran les bases de dades geogràfiques, mentre que, si treballem amb documents, ens aniran bé les bases documentals (si bé també es poden simular utilitzant BDR) i les XML per a tractar dades en format XML.

Un altre factor que condicionarà la tria de la base de dades és el domini on s'aplicarà. Segons aquest domini, per a aplicacions orientades a l'estratègia empresarial caldrà utilitzar bases de dades analítiques o orientades a columnes; per a aplicacions distribuïdes, bases de dades distribuïdes; per a aplicacions sense connectivitat, amb dispositius poc potents o senzilles, tindrà sentit fer servir una base de dades incrustada, i, per a aplicacions transaccionals clàssiques, podrem utilitzar qualsevol tipus de base de dades, excepte les analítiques i orientades a columnes, atès el baix rendiment que tenen en aquest context.

Finalment, altres requisits que poden condicionar la tria d'una base de dades o una altra són els següents: tenir dades que segueixin esquemes complexos definits recursivament (BDOO), necessitar efectuar inferències sobre les dades (BDD) i haver de tractar una gran quantitat d'informació XML (BD XML), entre d'altres.

3. Noves tendències

La gran quantitat de dades disponibles a Internet, la necessitat d'utilitzar-les d'una manera eficient, la possibilitat de distribuir les dades i els processos que les tracten entre diferents núvols (*clouds*) a la xarxa, la necessitat de tenir dades en qualsevol lloc i en qualsevol moment i la possibilitat d'emmagatzemar-les en dispositius mòbils, són diversos dels factors que contribueixen a l'aparició de nombroses bases de dades, que permeten tractar problemes específics i en dominis particulars.

Moltes d'aquestes noves bases de dades són simplement petites modificacions de les que hi ha a fi de millorar-ne el rendiment per a unes aplicacions determinades. En els darrers anys, hi ha un terme que se sent força en el món de les bases de dades: NoSQL, utilitzat per a designar el conjunt de bases de dades que es diferencien de les bases de dades relacionals en algun aspecte; per exemple, no satisfer les propietats ACID. Recentment, hi ha una certa controvèrsia amb l'etiqueta emprada per a aquest tipus de base de dades; molta gent troba més raonable fer servir la denominació NoREL (de *no relacional*) en comptes de NoSQL (de *no SQL*), ja que l'SQL tan sols representa el llenguatge d'accés a la base de dades i, paradoxalment, una base de dades no relacional també el podria usar i anomenar-se *NoSQL*. Entre les bases de dades NoSQL, trobem el BigTable de Google, Haddop, Cassandra i MapReduce. BigTable, per exemple, és un sistema gestor de base de dades propietari, creat per Google, i dissenyat per a tractar grans quantitats de dades. Actualment, la majoria d'aplicacions de Google Apps fan servir aquesta base de dades.

Una de les altres tendències que utilitzen actualment algunes bases de dades per a millorar el rendiment és tenir totes les dades emmagatzemades en memòria. Aquestes bases de dades, anomenades IMBD (*in-memory database*), poden tenir un rendiment més bo que les que utilitzen el disc com a mecanisme d'emmagatzematge principal, ja que eviten els retards que impliquen els accessos a disc per recuperar informació. Òbviament, la quantitat de dades que s'han de tractar pot ser un factor determinant a l'hora de decidir si fem servir una base de dades com aquesta o no.

Un altre tipus de base de dades del qual es torna a parlar amb força en els últims anys són les bases de dades paral·leles, tot i que existeix des de fa més de dues dècades. Aquestes bases de dades poden ser molt eficients en entorns d'informàtica en núvol (*cloud computing*), en què diferents nodes poden executar parts dels mateixos processos, cosa que permet no solament una distribució de la informació en diversos nodes, sinó també una computació en paral·lel. Són exemples d'aquest tipus de bases de dades Teradata, Aster Data, Netezza, ParAccel, DB2 i Oracle.

Un altre terme que se sent bastant aquests darrers anys és el de bases de dades de tipus clau-valor. Podem trobar mencions a aquestes bases de dades amb altres termes, com ara *item-oriented*, *attribute-oriented*, etc. En aquest tipus de bases de dades s'emmagatzema informació associada a una clau que la identifica, com, per exemple, el DNI d'una persona com a camp clau, i la informació de la persona com a valor. Podríem veure aquestes bases de dades com un conjunt de taules de dispersió que ens permeten obtenir informació sobre objectes a partir de la clau. La informació emmagatzemada en aquest tipus de base de dades no ha de ser necessàriament atòmica, i tampoc no es dóna cap suport per a definir relacions entre diversos elements de la base de dades. Això vol dir que la base de dades probablement contindrà informació duplicada i l'usuari haurà de comprovar que aquesta duplicació és correcta i haurà de garantir la integritat de la informació emmagatzemada. D'altra banda, tot i que algunes bases de dades utilitzen variants de SQL per a consultar les dades, no hi ha cap llenguatge estàndard per a fer-ho i, normalment, l'accés s'acostuma a fer mitjançant API. Aquest tipus de base de dades pot ser molt útil per a emmagatzemar informació molt simple o documents XML, però imposa moltes restriccions quan tracta amb estructures de dades complexes, molt ben estructurades i força relacionades. Alguns SGBD d'aquest tipus són SimpleDB d'Amazon Webservices; Google AppEngine DataStore, construït a sobre de BigTable; Microsoft SQL Data Services; l'eina de programari lliure CouchDB; project Voldemort, i Drizzle.

Cal comentar que els termes presentats no són disjunts entre si; hi pot haver bases de dades NoSQL, de tipus clau-valor i que emmagatzemin les dades en memòria. Per tant, més que considerar aquests termes com a tipus de base de dades, de moment té més sentit veure'ls com a característiques diverses que poden satisfer els SGBD actuals.

El món de les bases de dades està guanyant en varietat de solucions i de productes, tal com ho demostra el fet que en els darrers anys han aparegut moltes tendències noves, productes nous i maneres diferents de tractar la informació més eficientment. En aquests moments, una gran part de la comunitat es qüestiona el fet que el model relacional sigui la veritat absoluta que hagin de seguir totes les aplicacions d'una manera dogmàtica. Una de les veus més crítiques és la de l'expert en bases de dades Michael Stonebraker, creador de bases de dades com PostgreSQL, Vertica, Ingres i, finalment, VoltDB, qui argumenta que, actualment, per a la majoria de dominis d'aplicació, hi ha SGBD no relacionals que es poden comportar d'una manera més eficient. La darrera aportació que ha fet en aquest sentit és la creació de VoltDB, que es presenta com una nova base de dades que vol superar les bases de dades tradicionals en aplicacions de tipus transaccional (OLTP). Per a fer-ho, aquesta base de dades es fonamenta en moltes de les tendències i models de base de dades que hem comentat en aquest mòdul: l'emmagatzematge de les dades en memòria, el

suport per a la distribució de les dades i la paral·lelització dels processos, el tractament de transaccions innovador que satisfà les propietat ACID i té en compte la distribució de dades en nodes diversos, etc.

Bibliografia

The Object Data Management Standard: ODMG 3.0. Edited by R. G.G. Cattell, Douglas K. Barry, Mark Berler, Jeff Eastman, David Jordan, Craig Russell, Olaf Schadow, Torsten Stanienda y Fernando Velez.

Botella, A.; Camps, R.; Muñoz, A. (2010). *Bases de datos geográficas*. Barcelona: UOC.

Elmasri, R.; Navarthe, S.B (2006). *Fundamentals of Database Systems* (5a. edició). Addison Wesley Iberoamericana. ISBN: 0-321-41506-X.

Inmon, W.H.; Hackathorn, R.D. (1994). *Using the data warehouse*. Nova York: Wiley.

Özsu, M. T.; Valduriez, P. (1991). *Principles of Distributed Databases* (2a edició). Englewood Cliffs, N. J.: Prentice-Hall. ISBN: 0-13-659707-6.

Ramakrishnan, R.; Ullman, J.D. (1995). *A Survey of Research on Deductive Database Systems*. J. Logic Programming.

Snodgrass, Richard T. (1999). *Developing Time-Oriented Database Applications in SQL* (Morgan Kaufmann Series in Data Management Systems). San Francisco: Morgan Kaufmann; 504 pàgines; ISBN 1-55860-436-7.

Snodgrass, Richard T. (Ed.) (1995). *The TSQL2 Temporal Query Language*. Kluwer. ISBN: 0-7923-9614-6.

Stonebraker et al. C-Store (2005). *A column-oriented DBMS*. Proceedings of the 31st VLDB Conference. Trondheim, Norway.