Bases de dades relacionals

Novembre de 2020

Sumari

1. Bases de dades

Persistència

Anomenem persistència a la qualitat de les dades de romandre al sistema fins i tot si el desconectem

Gràcies a la persistència, quan tanquem l'ordinador les nostres dades no es perden. I això és gràcies a què les dades es desen fora del sistema:

- En dispositius auxiliars com discs, memòries USB, etc
- En d'altres sistemes. És el cas en el que les dades es mantenen en un servidor remot
- Al núvol

Bases de dades

Les Bases de dades (BD) són un mecanisme d'implementació de la persistència

El primer objectiu d'una BD és implementar la persistència. Però no de qualsevol manera, sinó que cal que es compleixi la propietat de la indpendència de les dades. Aquesta propietat diu que les dades persistents han de poder ser usades per diferents sistemes; és a dir, les dades han de ser independents del sistema o aplicació que les utilitza.

Exemple 1. Independència de les dades

El sistema de vendes necessita conèixer els clients i els productes. El sistema de control d'estocs necessita conèixer els proveïdors i els productes. El que es pretén és que els productes accedits per ambdós sistemes siguin els mateixos.

La BD es pot veure com un sistema comú de persistència al que poden accedir tant el sistema de vendes com el de control d'estocs.

Assolir la propietat de la independència de les dades té un cost elevat, i més si volem permetre l'accés simultani a les dades per part de diferents sistemes. Entre d'altres caldrà tenir en compte :

- Control d'accés
  - Qui pot accedir a la BD?
    - Es pot resoldre amb un accés per usuari i clau, per exemple
  - Qui pot accedir a cada informació de la BD?
    - Es pot resoldre amb permisos d'accés. Cada element de la BD pot tenir una llista dels usuaris que hi poden accedir, juntament amb la informació de si a part de visualitzar les dades també les poden alterar. També es pot resoldre fent que cada usuari tingui una llista amb els elements de la BD que pot accedir. En el primer cas parlem de permisos per element de la BD; en el segon, de permisos per usuari.
- Protecció de les dades
  - Perdurabilitat
    - Les dades no es poden destruir. I en el cas que un usuari tingui permís per fer-ho, enlloc de destruir-les les apartem del sistema, però en mantenim una còpia
  - Consistència
    - En tot moment s'ha de mantenir la consistència de la informació present. És a dir, totes les operacions fetes sobre la BD han de mantenir els invariants de la informació
      - Exemples d'invariants: les restriccions semàntiques (RS), atributs derivats
  - Recuperació
    - Què passa si una dada es corromp?
      - Un mal funcionament del sistema extern d'emmagatzematge, o una caiguda de tensió, són causes que poden corrompre les dades
      - Si mantenim un historial de les alteracions fetes a les dades (modificacions i insercions), llavors podem accedir a l'historial per a recuperar la informació malmesa
    - Què passa si tota la BD es corromp?

        - En alguns casos la fallada dels sistemes és tan gran, que l'accés a l'historial no resol el problema
        - En aquest casos, disposar d'una còpia de seguretat de les dades, generada automàticament, pot ser una solució

- Gestor de transaccions
  - Quan més d'un procés pot accedir simultàniament a les mateixes dades, apareixen tota una sèrie de problemes nous. Anomenem transacció a l'accés a una BD que fa un procés.[1] En aquest accés es poden consultar o alterar dades.
  - Les transaccions han de complir les anomenades propietats ACID:
    - Atomicitat. La transacció és una unitat indivisible
      - Per exemple, no pot ser que en una transacció de reintegre apuntem en el diari de fons que s'han donat els diners, sense que aquesta operació també quedi reflectida en el saldo del compte. O es fan ambdues anotacions, o no se'n fa cap.
    - Consistència. Tota transacció deixa les dades en un estat consistent
    - Independència o isolament. Tot i que una transacció s'executi en paral·lel a una altra, el resultat ha de ser equivalent al que obtindríem si les executéssim una després de l'altra
      - Suposem que estem fent dos reintegres de 100€ concurrentment sobre un compte amb un saldo de 150€.
      - Si executem una transacció darrere l'altra, la segona no s'admetrà per manca de saldo.
      - Imaginem que els executem a la vegada, i que ambdues consulten el saldo de 100€. Llavors totes dues es veuen en capacitat de tirar endavant! El resultat serà que els dos reintegres s'hauran fet, i el saldo resultant serà de -50€. En aquest cas les transaccions no són independents.
      - Si en lloc d'un reintegre pensem en una reserva en un restaurant, la no independència de les transaccions pot generar overbooking; és a dir, que una mateixa taula sigui reservada per més d'un comensal
    - Durabilitat. Els canvis fets per una transacció no es poden desfer, a no ser que es demani explícitament
      - Suposem un compte amb 100€, i li fem un reintegre de 50€ i un ingrés de 25€. El resultat voldríem que fos un saldo de 75€
      - Imaginem que ambdues transaccions consulten el saldo en el mateix moment. A continuació la transacció del reintegre apunta la disminució de saldo (que d'aquesta manera queda a 50€). Tot seguit la transacció d'ingrés modifica el saldo tenint en compte el saldo que ha consultat (100€) i el saldo ingressat (25€). El saldo final serà de 125€

[1] La definició real de *transacció* és un pèl més complexa. Però pels propòsits expositius pretesos aquí, la definició simplificada presentada és més que suficient.

· Les dues transaccions, quan s'executen concurrentment, no són isolades. És més, en l'execució presentada la transacció d'ingrés desfà o oculta els canvis realitzats per la transacció de reintegre. El resultat és que el reintegre no és durable.

– El mecanisme més habitual per assegurar que les transaccions són ACID és emprar un mecanisme de bloqueig: quan un element de la BD està bloquejat, ningú més hi pot accedir. El bloqueig pot ser de lectura (ningú ho pot llegir) o d'escriptura (ningú ho pot modificar); el bloqueig pot afectar un element atòmic de la BD o tot un grup d'elements. Una jerarquia habitual de bloquejos és atribut-registre-taula-base de dades

Les BD són eines extremadament complexes

Els programes encarregats de gestionar una BD s'anomenen sistemes gestors de bases de dades (SGBD, DBMS en anglès).

2. Introducció a les bases de dades relacionals (BDR)

2.1 Una mica d'història

El 1970 Edgar Frank Codd diu que una BD ha de seguir les següents regles:

1. Regla fundacional. Qualsevol operació sobre una base de dades es pot fer emprant només operacions relacionals
2. Regla de la informació. Només hi ha una manera de representar qualsevol tipus d'informació: un valor dins d'una taula
3. Accés garantit. Tota informació és accessible
4. Regla del tractament sistemàtic dels valors nuls. La manca d'informació es representa amb nulls
5. Catàleg dinàmic. L'estructura de la BDR es representa internament en la mateixa BD, de la mateixa manera que es fa amb les dades. I per tant es pot modificar dinàmicament, com qualsevol altra dada
6. Llenguatge comprensiu. Hi ha d'haver un un llenguatge simple, de sintaxi lineal, interactiu que permeti tant la definició de dades (crear i modificar l'estructura de la BD) com la seva manipulació (inserir, modificar i eliminar dades de la BD)
7. Actualització de vistes. Tot conjunt d'informació obtingut de la BD i que teòricament pot ser actualitzat, ha de poder ser actualitzat pel sistema

8. Alt nivell de manipulació. La manipulació de les dades s'ha de poder fer tant a nivell individual, com a nivell de grup

9. Independència física de les dades. La representació interna de les dades no ha d'afectar les aplicacions

10. Independència lògica de les dades. El canvi en el model lògic d'una BD no ha d'afectar les aplicacions que teòricament no estan afectades per aquest canvi. Per exemple, si afegim la informació de l'estoc d'un producte, l'aplicació de generació d'albarans ha de funcionar sense problemes; una altra cosa és que ara vulguem afegir un requeriment que no permeti vendre productes si no tenim prou estoc.

11. Independència de la integritat. Les restriccions d'integritat s'han de poder definir amb el llenguatge relacional, i emmagatzemar-les en la mateixa BD.

12. Independència de la distribució. Les aplicacions no s'han de veure afectades pel fet que les dades estiguin centralitzades o distribuïdes.

13. Regla de la no subversió. No ha de ser possible usar mecanismes de baix nivell per saltar-se el nivell relacional. En cas de permetre-ho ningú no pot assegurar que es mantenen les restriccions d'integritat.

Una BD que segueixi les regles de Codd l'anomenem base de dades relacional (BDR).

Els anys '80 apareixen els primers productes comercials que, en certa mesura, intenten seguir les regles de Codd (R, Oracle, DB2, Informix,...). Cap d'ells, però, implementa totes les regles de Codd.

Avui en dia no hi ha cap producte comercial que sigui purament relacional, en el sentit de Codd. De fet, actualment es considera que una BD és una base de dades relacionals si segueix el model relacional: l'únic element d'emmagatzematge de dades és la relació; i hi ha operadors, anomenats operadors relacionals, que permeten combinar relacions o extreure informació d'elles.

2.2 Primera aproximació a les bases de dades relacionsls

Sense entrar en els detalls, una BDR és una base de dades que segueix, fins a un cert punt, les regles de Codd, que anomenem regles relacionals.

Els sistemes gestors de bases de dades que implementen el model relacional se'ls anomena SGBDR.

Una BDR es pot definir com un conjunt de relacions més un conjunt de restricicons sobre aquestes relacions. Les operacions sobre una BDR es fan amb àlgebra relacional o càlcul relacional

Base de dades relacional

Anomenem Base de dades relacional a un conjunt de relacions més un conjunt de restriccions, manipulable exclusivament a través de l'àlgebra i càlcul relacional:

BDR = Relacions + Restriccions

En els apartats que segueixen analitzem tots aquests elements: relacions, àlgebra relacional, càlcul relacional, i restriccions.

3. Relació

3.1 El concepte de relació

Les bases de dades relacionals es basen en l'anomenat model relacional. Aquest és un model matemàtic, però d'enunciació molt simple, i amb una representació visual molt intuïtiva.

Definició de relació

- Domini. Conjunt de valors
- Relació. Subconjunt del producte cartesià d'un conjunt de dominis:

$$R \subseteq D_1 \times D_2 \times \ldots \times D_n$$

- Tuple. Element d'una relació

$$t =< a_1, a_2, \ldots, a_n > \text{ on } a_i \in D_i$$

Exemple 2. Dominis

Alguns exemples de domini:

- $D_1 = \{1, 2, 3, 4\}$
- $D_2 =$ conjunt dels nombres parells = $\{2n \mid n \in \mathcal{N}\}$
- $D_3 = Colors = \{$ $\}$
- $D_4 = Treballadors = \{Anna, Berta, Carles, Diana\}$
- $D_5 = \mathcal{N} =$ conjunt dels nombres naturals

Exemple 3. Relació

La relació *Edat*, que interrelaciona el conjunt dels treballadors D_4 amb la seva edat, és:

$$Edat \subseteq Treballadors \times \mathcal{N}$$

Observem que $Edat$ és realment un subconjunt del producte cartesià.

Si l'*Anna* té 32 anys, tindrem que $< Anna, 32 > \in Edat$.

Però llavors necessàriament, per exemple, $< Anna, 46 > \notin Edat$, tot i que $< Anna, 46 > \in Treballadors \times \mathcal{N}$

Exemple 4. Tuple

El tuple $< Berta, 35 >$ és un tuple de la relació $Edat \subseteq Treballadors \times \mathcal{N}$, que diu que l'edat de la Berta és de 35 anys

Tot model conceptual es pot expressar emprant exclusivament les relacions

3.2 Predicat de la relació

Tota relació té associat un predicat, el predicat de la relació, que expressa la semàntica dels tuples de la relació. És a dir, diu com cal interpretar semànticament els diferents tuples.

Exemple 5. Predicat de la relació

Sigui la relació $Edat \subseteq Treballadors \times \mathcal{N}$.

Els tuples de R són de la forma $< treballador, n >$, on $treballador$ és un valor vàlid pel domini $Treballador$ i $edat$ és un valor vàlid dins del domini de l'$Edat$:

$$treballador \in Treballador$$
$$n \in \mathcal{N}$$

El predicat de la relació diu que, donat un tuple qualsevol $< treballador, n >$, aquest tuple significa que "el *Treballador* de nom *treballador* té l'edat de *n* anys".

Per exemple $< Anna, 32 >$ diu que l'Anna té 32 anys

Predicat de la relació

- Una relació és un concepte matemàtic, però buit de significat interpretatiu
- Una relació només es pot interpretar si disposem del seu predicat de la relació

En una relació, un mateix domini pot aparèixer més d'un cop. En cada aparició, però, el significat del valor concret del domini serà diferent; qui el defineix és el predicat de la relació.

Exemple 6. Domini repetit

Sigui $R \subseteq Treballador \times \mathcal{N} \times \mathcal{N}$.

El predicat de la relació diu que el treballador té l'edat indicada en el primer domini natural, i que porta tants anys a l'empresa com indica el segon domini natural.

Com es pot veure l'orde dins dels tuples és important: $< Carles, 28, 10 >$ no és el mateix que $< Carles, 10, 28 >$.

3.3 Atribut

Sabem que una relació és el subconjunt d'un producte cartesià; així, per expressar una relació n'hi ha prou en donar la seqüència ordenada dels seus dominis. La seqüència ha de ser ordenada per tal que la interpretació donada pel predicat de la relació sigui correcta.

Si a cada aparició d'un domini dins d'aquesta seqüència li donem un nom únic, llavors la relació es pot expressar com un conjunt. I aquesta és la idea dels atributs.

Atribut

Un atribut és un parell $< nom, Domini >$, sota la condició que en una mateixa relació no hi hagi dos atributs amb el mateix nom

Exemple 7. Atribut

Sigui $R \subseteq Treballador \times \mathcal{N} \times \mathcal{N}$.

Una manera alternativa de definir R és la següent, on (a,b) indica una seqüència ordenada de dos elements:

$$R = (Treballador, \mathcal{N}, \mathcal{N})$$

Definim els següents atributs:

- $nom : Treballador$
- $edat : \mathcal{N}$
- $antiguitat : \mathcal{N}$

Gràcies a aquests atributs podem definir la relació R com:

$$R = \{nom, edat, antiguitat\}$$

El predicat de la relació diu que el *nom* té l'*edat* i l'*antiguitat* indicades; i això és indiferent de la posició on aparegui cada atribut dins de la relació. Així, gràcies als atributs, l'ordre ha desaparegut:

$$R = \{nom, edat, antiguitat\} = \{antiguitat, edat, nom\}$$

En general els atributs els definim al mateix moment que la relació:

$$R = \{nom : Treballador, edat : \mathcal{N}, antiguitat : \mathcal{N}\}$$

- Els atributs permeten expressar les relacions en termes de conjunt, sense ordre
- Els predicats del a relació s'expressen en termes de la relació

3.4 Definició d'una relació

Definició d'una relació

Les relacions les definirem en termes d'atributs:

$$R = \{atribut_1 : Domini_1, \ldots, atribut_n : Domini_n\}$$

Formalment:

- Una relació és un predicat
- Les variables lliures d'aquest predicat són els seus atributs
- Un tuple és el resultat de substituir cadascuna de les variables lliures d'aquest predicat per un valor del domini corresponent
- En conseqüència, els tuples són enunciats
- La interpretació d'aquests enunciats és la que proporciona el predicat de la relació
- Els tuples, per tant, representen fets de la propietat expressada pel predicat de la relació

Semàntica d'una relació

Tota relació representa una propietat; els fets que compleixen aquesta propietat són els tuples

Exemple 8. Fets i predicats

La relació $Edat = \{nom : Treballador, n : \mathcal{N}\}$ és un predicat $edat(nom : Treballador, n : \mathcal{N})$

En substituir les variables lliures del predicat $edat()$ per valors dels dominis pertinents, obtenim un enunciat. Per exemple $edat(Diana, 23)$ és un enunciat que diu que la propietat $edat$ és certa amb els valors $Diana$ i 23.

El predicat de la relació dóna la interpretació d'aquest enunciat: és un fet que la $Diana$ va néixer fa 23 anys.

Els diferents fets s'expressen introduïnt els tuples corresponents a la relació $Edat$. Així, l'enunciat $edat(diana, 23)$ s'expressa mitjançant el tuple $< Diana, 23 >$.

Què és una relació

- Una relació és un predicat sobre atributs
- Tota relació ha d'anar acompanyada del predicat de la relació, que és qui li dóna el valor interpretatiu
- Un tuple és un enunciat que expressa un fet: diu que determinats valors del domini compleixen el predicat de la relació

3.5 Representació visual de les relacions

Les relacions tenen una representació visual molt simple. Gràcies a aquesta representació es poden fer raonaments matemàtics complexs de manera simple i intuïtiva.

Tota relació es pot expressar en forma de taula, on les columnes són els atributs, i les files són els tuples (o fets).

Exemple 9. Representació visual d'una relació

Sigui la relació $Edat = \{nom : Treballador, n : \mathcal{N}\}$.

Llavors podem construir la següent taula:

nom:Treballador	n:$\mathcal{N}$
Anna	32
Carles	28
Diana	23

La taula mostra el conjunt actual de tuples de la relació *Edat*. Tenim tres fets, que ens diuen que coneixem que en aquests moments l'Anna té 32 anys; en Carles en té 28; i la Diana, 23. De la Berta no en sabem res.

És important d'observar que les taules expressen els fets coneguts. I a tots els efectes aquests són els únics fets reals.

Que per la *Berta* no hi hagi cap entrada al a taula tan pot ser perquè no té edat (encara no ha nascut), o bé perquè la desconeixem. A tots els efectes, però, per a la taula *Edat* la *Berta* no existeix.

La representació visual d'una relació posa clarament en evidència el doble ús del terme relació: per una banda és un conjunt d'atributs; però per l'altre també és un conjunt de tuples.

La relació, com a conjunt d'atributs, és una abstracció: diu els dominis que intervenen, i quina semàntica cal donar a les seves relacions.

La relació, com a conjunt de tuples, és una realització: diu quins fets es tenen, en un moment determinat, que compleixen el predicat de la relació$_{abstracció}$.

Per evitar confusions usarem la següent nomenclatura:

- Relació o variable de relació: Abstracció; conjunt d'atributs
- Estat de relació: Conjunt de tuples consistents amb una relació donada; és la realització d'una *relació*. Ho notarem amb $r(R)$, on R és una relació.

Nomenclatura

- Capçalera o esquema: Sinònim de relació (o variable de relació)

  Notació: R

- Taula: Sinònim d'estat de relació

  Notació: $r(R)$

- Fila: Sinòmim de tuple o fet

  Notació: $t(r)$ o bé $t \in r(R)$

- Columna: Sinònim d'atribut

  Notació: $R.atr$ o $t.atr$

- Semàntica de la taula: Sinònim de predicat de la relació

  Notació: $P(R)$ o bé $sem(R)$

4. Àlgebra i càlcul relacional

4.1 Àlgebra relacional

Les relacions s'han de poder manipular. L'àlgebra relacional és un conjunt d'operadors sobre estats de relació que sempre retornen un altre estat de relació. Sigui *op* un operador relacional; llavors:

$$op : Relació_1 \times \ldots \times Relació_n \rightarrow Relació'$$
$$op : r(Relació_1) \times \ldots \times r(Relació_n) \rightarrow r(Relació')$$

L'àlgebra relacional consta de cinc operadors, que en son la seva base. Sovint però també s'usen uns altres operadors que es poden explicar en termes dels operadors bàsics; són els operadors derivats. Les dues taules següents presenten tots aquests operadors relacionals.

Tota expressió formada amb els operadors relacionals rep el nom de consulta relacional. Tota consulta es pot veure com una funció que donats diferents estats de relació retorna un nou estat de relació:

$$consulta : Relació_1 \times \ldots \times Relació_n \rightarrow Relació'$$
$$consulta : r(Relació_1) \times \ldots \times r(Relació_n) \rightarrow r(Relació')$$

Operadors relacionals

- Producte: $A \times B$

$$A \times B = \{< a, b > | a \in A \wedge b \in B\}$$

- Unió: $A \cup B$

$$A \cup B = \{x | x \in A \vee x \in B\}$$

- Diferència: $A - B$

$$A - B = \{x | x \in A \wedge x \notin B\}$$

- Projecció i-èssima: $\prod_i(A)$

$$\prod_i(A) = \{< x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n > \\ | < x_1, \ldots, x_i, \ldots, x_n > \in A\}$$

  - L'operador de projecció es pot generalitzar indicant més d'una columna:

    $\pi_X(A)$, $X \subseteq \{1, \ldots, n\}$ on n és el nombre d'atributs de A

- Selecció segons la restricció P: $\sigma_P(A)$

$$\sigma_P(A) = \{a | a \in A \wedge P(a)\}$$

L'operació de projecció el que fa és un tall vertical de la taula. El resultat d'una projecció és una taula idèntica a la de partida però només amb les columnes indicades.

L'operació de selecció el que fa és un tall vertical de la taula. El resultat d'una projecció és una taula idèntica a la de partida però només amb les files indicades.

El producte cartesià augmenta la longitud dels tuples; és a dir, genera una taula resultant amb més amplada que qualsevol de les taules de partida. La unió, genera una taula amb més tuples; és a dir, amb més alçada que qualsevol de les taules de partida.

La diferència i selecció disminueixen el nombre de files. La projjecció disminueix el nombre de columnes.

Exemple 10. Consulta relacional

Sigui la relació següent:

$$Venda =< nom, preu, pes, proveïdor >$$

Un estat vàlid per a aquesta relació és:

Nom	Preu	Pes	Proveïdor
taula	3000	8	moblesBonics
cadira	250	1.5	cadirairesModerns
armari	2500	4	moblesBonics

El predicat de la relació diu que hem fet alguna venda del producte amb el nom *nom*, que pesa *pes* kg, que costa els euros indicats per *preu*, i que el proveïdor és *proveïdor*.

Per saber quins són els mobles que hem venut de *moblesBonics*, construim la seqüent consulta:

$$\pi_{nom}(\sigma_{proveïdor="moblesBonics"}(Vendes))$$

Observem la consulta: seleccionem tots els tuples que com a proveïdor tenen *moblesBonics* (tall horitzontal de la taula); i dels tuples resultants ens quedem només amb l'atribut *nom* (projecció o tall vertical de la taula).
El resultat és la següent taula:

Nom
taula
armari

Les definicions presentades dels operadors relacionals són definicions simplificades: només hem expressat cada operador en termes dels estats de la relació; manca expressar les exigències de cada operador sobre les variables de relació involucrades (la capçalera de la taula), tant pel que fa als operands com al resultat. En la següent taula expressem aquestes condicions.

Operadors relacionals en termes dels esquemes

- Producte: $A \times B$
  - Els operands poden tenir qualsevol esquema. L'esquema resultant és la unió dels esquemes dels operands

$$A \times B = A \cup B$$

- Unió: $A \cup B$
  - Els operands han de tenir el mateix esquema. L'esquema resultant és l'esquema dels operands

$$A \cup B = A = B$$

- Diferència: $A - B$
  - Els operands han de tenir el mateix esquema. L'esquema resultant és l'esquema dels operands

$$A - B = A = B$$

- Projecció: $\prod_i(A)$
  - L'operand pot tenir qualsevol esquema. L'esquema resultant és el conjunt format per les columnes projectades

$$\prod_{i_1,\ldots,i_k}(A) = A_i, \ldots, A_k$$

- Selecció: $\sigma_P(A)$
  - L'operand pot tenir qualsevol esquema. L'esquema resultant és l'esquema de l'operand

$$\sigma_P(A) = A$$

Operadors relacionals derivats

- Intersecció: $A \cap B$

$$A \cap B = \{x | x \in A \wedge x \in B\}$$

- Unió natural (join) segons els atributs a i b: $A \bowtie_{a,b} B$

$A \bowtie_{a,b} B =$

$= \{< x_1, \ldots, x_p, \ldots, x_n, y_1, \ldots, y_{q-1}, y_{p+1}, \ldots, y_m > |$

$< x_1, \ldots, x_n > \in A \wedge < y_1, \ldots, y_n > \in B \wedge x_q = y_p\}$

on x_p és el valor de l'atribut a de A, i y_q és el valor de l'atribut b de B

- Divisió: A/B

$A/B = \{< x_1, \ldots, x_k > |$
$\forall < y_1, \ldots, y_n > \in B \Rightarrow \exists < x_1, \ldots, x_k, y_1, \ldots, y_n > \in A\}$

La unió natural és un dels operadors més emprats. El seu objectiu és unir dues taules usant com a nexe d'unió un atribut de cada taula: unim un tuple de A amb un tuple de b quan els atributs que defineixen el nexe tenen exactament el mateix valor. Al tuple resultant aquest valor repetit només el fem aparèixer un cop.

Operadors derivats en termes dels esquemes

- Intersecció: $A \cap B$
  - Els operands han de tenir el mateix esquema. L'esquema resultant és l'esquema dels operands

$$A \cap B = A = B$$

- Unió natural: $A \bowtie_{a,b} B$
  - Els operands poden tenir qualsevol esquema, sota la condició que entre els dos hi hagi un domini comú. L'esquema resultant és la unió de l'esquema dels operands, menys un dels atributs que defineixen la unió

$$A \bowtie_{a,b} B = A \cup B - \{b\}$$

- Unió natural: A/B
  - L'esquema del segon operand ha de ser un subconjunt de l'esquema del primer operand. L'esquema resultant és la diferència dels esquemes-dels operands

$$A/B = B - A$$
$$A \subseteq B$$

4.2 Càlcul relacional

Els operadors relacionals constitueixen l'àlgebra relacional. Amb l'àlgebra relacional, en fer una consulta explicitem quins són els pasos per a obtenir el resultat; i ho fem indicant quins operadors relacionals cal aplicar i en quin ordre.

El càlcul relacional, en canvi, expressa les consultes en termes del resultat desitjat, sense necessitat d'indicar el procediment per a obtenir-lo.

Distingim entre dos tipus de càlcul relacional: el basat en tuples, i el basat en dominis.

Una consulta en el càlcul relacional basat en tuples és de la forma:

$$\text{consulta} = \{t|\{P(t)\}$$

Una consulta en el càlcul relacional basat en dominis és de la forma:

$$\text{consulta} = \{< x_1, \ldots, x_k > |\{P(x_1, \ldots, x_k)\}$$

El el càlcul relacional basat en dominis es cerquen els dominis (atributs) que compleixen la propietat desitjada, i amb els valors obtinguts es crea el tuple resultant. En el càlcul relacional basat en tuples es demanen directament les propietats desitjades al tuple resultant. En ambdós càlculs, el predicat que defineix la consulta s'expressa en la lògica de primer ordre o lògica de predicats.

Es pot demostrar que el càlcul relacional (en qualsevol de les seves versions) i l'àlgebra relacional són equivalents.

El càlcul relacional i l'àlgebra relacional són equivalents

4.3 L'àlgebra i càlcul relacional en els sistemes comercials

4.3.1 L'àlgebra com a llenguatge del motor relacional

En els SGBDR comercials, en general l'usuari defineix les consultes amb el càlcul relacional; i aquestes consultes es converteixen a àlgebra relacional abans de ser executades. (De fet, un cop després de ser expressades en àlgebra relacional, s'optimitzen, i finalment s'executen).

4.3.2 Llenguatges relacionals: SQL i QBE

Els llenguatges comercials basats en el càlcul relacional que són més habituals són el SQL i el QBE. El llenguatge Structured Query Language (SQL) es basa en el càlcul relacional basat en tuples; el llenguatge Query By Example (QBE) es basa en el càlcul relacionat basat en dominis.

En una BDR, les consultes, generalment en SQL, s'executen en termes dels operadors relacionals

La majoria dels SGBDR comercials usen com a llenguatge d'interacció l'SQL ampliat, que no és més que una versió de SQL amb operacions per definir l'estructura de la BD. Cal tenir present, però, que el llenguatge SQL més que un llenguatge és una família de llenguatges; cada SGBDR comercial té la seva pròpia versió amb les seves particularitats, la qual cosa fa difícil parlar d'un estàndard SQL .Ho remarquem: la "S" significa *Structured*, i no pas *Standard*.

El llenguatge SQL no és estàndard

4.3.3 Estats de relació que no aparenten ser un conjunt

No hi ha cap SGBDR comercial que sigui 100% relacional en el sentit de Codd. La majoria de SGBDR comercials per a tota consulta generen una columna extra que conté la posició del tuple dins la taula. En alguns casos aquest atribut és explícit, en d'altres implícit. Una conseqüència desagradable de l'ús implícit d'aquesta columna és que en les visialitzacions normals ens pot donar la sensació que hi ha un tuple repetit, quan realment no hi és.

Exemple 11. Falsos tuples repetits

Exemple 10, pàgina 15

Siqui l'exemple 10.→ Suposem que l'estat de relació que tenim ve donat per la següent taula:

Nom	Preu	Pes	Proveïdor
taula	3000	8	moblesBonics
cadira	250	1.5	cadirairesModerns
armari	2500	4	moblesBonics
taula	2500	6	moblesBonics

Llavors la consulta hauria de donar:

Nom
taula
armari

En canvi, en alguns SGBDR comercials el resultat obtingut sembla que tingui una fila repetida:

Nom
taula
armari
taula

Això és així perquè l'operador de selecció retorna tres files. Però aquestes tres files tenen els quatre atributs de *vendes*, més un atribut que diu el número del tuple:

NumTuple	Nom	Preu	Pes	Proveïdor
1	taula	3000	8	moblesBonics
2	armari	2500	4	moblesBonics
3	taula	2500	6	moblesBonics

Ara projectem sobre l'atribut explícit *nom*, però també sobre l'atribut implícit *numTuple*. El resultat és el següent, que com es pot observar no conté files repetides:

NumTuple	Nom
1	taula
2	armari
3	taula

Alguns SGBDR, per+o, tot i generar la consulta amb aquest atribut implícit, no el mostren en el resultat.

En els SGBDR la visualització de les consultes poden fer creure que contenen files repetides, però en realitat no és així

5. Clau d'una relació

5.1 Notació

Per a l'estudi de les claus, emprarem la següent notació:

- Notarem els atributs amb les majúscules de l'inici de l'alfabet: A, B, C
- Amb les majúscules del final de l'alfabet expressarem conjunts d'atributs: X,Y, Z
- El resultat de projectar un tuple t al conjunt d'atributs X l'expressarem amb $t[X]$

5.2 Súper-clau

5.2.1 Definició de *súper-clau*

En el model relacional el concepte de clau és molt rellevant. Per definir-lo, però, primer cal introduir què és una súper-clau.

Súper-clau

Donat un esquema R anomenem súper-clau de R a aquell conjunt d'atributs $X \subseteq R$ tal que donat un estat de R qualsevol, $r(R)$, per a tot t_1, t_2 de $r(R)$ es compleix:

$$t_1[X] = t_2[X] \Rightarrow t_1 = t_2$$

Una súper-clau és aquell conjunt d'atributs que permet diferenciar una fila qualsevol de qualsevol altra fila. I ho permet en qualsevol estat de la relació.

Exemple 12. Súper-clau

Sigui el següent estat d'una relació, on hem numerat les diferents files:

	U	V	W
1	x	a	1
2	x	b	2
3	y	a	1
4	x	b	1

Cap de les tres columnes és una súper-clau. Vegem-ho:

- Suposem per exemple l'atribut `U`
- Si ens quedem amb només aquesta columna obtenim la llista $\{x, x, y, x\}$, que té repetits
- Compte: no estem projectant; la projecció sempre retorna un estat de relació, i per tant no té files repetides; així, $\prod_U(R) = \{x, y\}$
- L'observació que la columna `U` conté valors repetits diu que hi ha tuples diferents tals que la seva projecció sobre `A` coincideix. I per tant no es compleix la condició de súper-clau. Per exemple:

$$t_1[U] = t_2[U] = \{x\} \text{ amb } t_1 \neq t_2$$

- El mateix passa amb les altres columnes.

El conjunt $X = \{U, W\}$ tampoc és una súper-clau.

- **Anàlisi 1**. Si de la taula ens quedem amb les columnes `U` i `W` veiem que hi ha files repetides; i per tant el resultat no és un estat de relació.
- **Anàlisi 2**. Podem observar que $t_1[U, W] = t_4[U, W]$, tot i que $t_1 \neq t_4$

El conjunt $Y = \{U, V\}$ tampoc és una súper-clau; ens ho demostren t_2 i t_4.

El conjunt $Y = \{V, W\}$ tampoc és una súper-clau; ens ho demostren t_1 i t_3.

Conclusió: Cap subconjunt propi dels atributs de R és una súper-clau.

Exemple 13. Súper-clau enganyosa

Sigui el següent estat d'una relació, on hem numerat les diferents files:

	U	V	W
1	x	a	1
2	x	b	2
3	y	a	1
4	z	b	1

Ara els conjunts $X = \{U, V\}$ i $Y = \{U, W\}$ no generen files repetides. Per tant es compleix:

$$t_1[X] = t_2[X] \Rightarrow t_1 = t_2 \; t_1[Y] = t_2[Y] \Rightarrow t_1 = t_2$$

Podem afirmar llavors que X i Y són súper-claus de R??

La resposta és NO. La definició de súper-clau exigeix que la condició de súper-clau es compleixi per a tot estat de relació.

Aquí tenim un estat de relació que compleix la condició de súper-clau; però això no impedeix que en un altre estat $r'(R)$ de la mateixa relació, la condició no es compleixi.

Per exemple, si canviem la `z` de la columna `U` per una `x`, obtenim l'estat de relació de l'exemple 12$^{\rightarrow}$; i en ell hem vist que cap subconjunt propi d'atributs és una súper-clau.

Exemple 12, pàgina 22

L'anàlisi d'un estat de relació pot dir-nos quins conjunt no són súper-clau. Però no permet detectar les *súper-clau*

5.2.2 Detecció de les *súper-clau*

Com podem detectar les súper-clau d'una relació? Des d'un estat de relació qualsevol és impossible; caldria analitzar tots els estats de relació, que poden ser infinits. L'única manera de determinar les súper-clau és mitjançant el predicat de la relació i els requeriments del problema.

Exemple 14. Detecció de súper-claus

Suposem que ens exigeixen que en cada ciutat hi hagi com a molt un magatzem.

Sigui la relació `Magatzem={NomMagatzem, CiutatUbicació-Magatzem, Responsable}`.

Llavors conjunt $X = \{$`CiutatUbicacióMagatzem`$\}$ és una súper-clau de la relació.

El raonament és que el predicat de la relació diu que donat un tuple $< m, c, r >$ el que modela és el fet que el magatzem `m` està ubicat a la ciutat `c`, i el seu responsable és `r`. Si aquesta relació modela el nostre problema, ha de complir els requeriments, i per tant no hi pot haver

una ciutat en la que hi hagi dos magatzems; és a dir, no hi poden haver dos tuples $< m, c, r >$ i $< m', c, r' >$ amb $m \neq m'$.

Detecció de súper-claus

Per detectar les súper-clau cal accedir als requeriments i al predicat de la relació

5.2.3 Existència i no unicitat

Donada una relació R qualsevol, sempre existeix una súper-clau. Però aquesta no necessàriament és única.

Exemple 15. Existència de la súper-clau

Donada una relació R qualsevol, el conjunt X format per tots els atributs de R és una súper-clau de R.

Per veure-ho n'hi ha prou en recordar que un estat de relació $r(R)$ qualsevol és un conjunt; i que per tant no hi poden haver dues files iguals.

Exemple 16. No unicitat de la súper-clau

En una ciutat només s'hi pot ubicar un magatzem; i una mateix persona només pot responsable d'un magatzem. Així tan el conjunt `{Ubicació}` com el conjunt `{Responsable}` són súper-claus de la relació $Magatzem$.

Súper-clau

- **Existència**. El conjunt de tots els atributs d'una relació sempre és una súper-clau
- **No unicitat**. La súper-clau d'una relació no necessàriament és única
- **Encapsula la semàntica**. Per determinar les súper-claus cal accedir a la semàntica del problema

5.2.4 Utilitat de les súper-claus

Per definició, donat un estat qualsevol d'una relació, no hi pot haver dos tuples diferents que coincideixin de valors simultàniament en tots els atributs de la súper-clau. D'aquest manera podem usar els valors dels atributs d'una súper-clau per a identificar un tuple.

En una BDR el mecanisme d'identificació d'un tuple és qualsevol de les súper-clau de la relació

5.3 Clau

Una clau és aquella súper-clau tal que si li treiem un atribut deixa de ser súper-clau:

Clau d'una relació

Donada una relació R i un conjunt d'atributs $X \subseteq R$ direm que X és una clau de R si i només si:

- X és una súper-clau de R

  $\forall r(R)\ \forall t_i, t_j \in r(R)$ es compleix que:
  $$t_i[X] = t_j[X] \Rightarrow t_i = t_j$$

- És mínim en el sentit que per a tot $Y \subset X$ es compleix que Y no és una súper-clau de R

L'existència de la súper-clau implica l'existència de la clau: en el pitjor dels casos, el conjunt format per tots els atributs de R és una clau.

Anàlogament al que passa a les súper-claus, la clau no té perquè ser única.

Exemple 17. Clau

Sigui el següent estat d'una relació, on hem numerat les diferents files:

	U	V	W
1	t	a	1
2	x	b	2
3	y	a	2
4	z	b	1

Suposem que tots els estats possibles de la relació es comporten com

aquesta taula. Aquesta suposició ens permet detectar les *súper-claus* analitzant només aquest estat de la relació. Sabem que la hipòtesi no es pot afirmar sense coneixement de la semàntica del problema, però ens serveix per a simplificar l'exemple.

L'anàlisi del es files, sota la hipòtesi de treball, ens permet afirmar que els següents conjunts són súper-claus:

- $X_1 = \{U, V, W\}$
- $X_2 = \{U, V\}$
- $X_3 = \{V, W\}$
- $X_4 = \{U, W\}$
- $X_5 = \{U\}$

Sobre aquests conjunt tenim les següents inclusions (només n'indiquem algunes):

- $X_2 \subset X_1$
- $X_5 \subset X_2$
- $X_5 \subset X_4$

La primera inclusió diu que X_1 no és una clau: si a X_1 hi traiem atributs, obtenim x_2, que també és una súper-clau.

Les següents inclusions descarten X_2 i X_4 com a clau.

Un cop eliminades totes les súper-claus que són un súper-conjunt d'una altra súper-clau, el que ens queda són les claus. En concret, les claus de R són:

- $X_3 = \{V, W\}$
- $X_5 = \{U\}$

5.4 Clau primària

En molta bibliografia es parla de clau primària, i es presenta com a sinònim de clau. L'error rau en qüestions històriques.

En l'època que es defineixen les BDR els sitemes d'emmagatzematge persistent eren molt cars; i això duia a emmagatzemar cada informació només en un lloc, sense reduplicació. Com a conseqüència, la millor manera d'identificar una informació era per la seva posició dins d'aquest emmagatzematge.

En el cas de les BDR, el problema era determinar quina informació del tuple era imprescindible per poder calcular (potser mitjançant una taula auxiliar o índex) quina era aquesta posició d'emmagatzematge en el suport extern. Aquesta informació necessària per a calcular la posició d'emmagatzematge era el que s'anomenava clau primària. Evidentment, donat un tuple la clau primària és única.

La clau primària, per tant, no és una definició relacional; es tracta d'una

definició en relació a la tecnologia de l'emmagatzematge.

El problema apareix perquè en els seus primers articles en Codd no s'adona de la no unicitat de les claus (tal i com hem definit *clau* més amunt). I això li permet usar com a clau primària la clau de la relació.

Compte però: l'ús de la clau com a clau primària implica que una relació té una única clau, cosa que no és certa. De fet aquest va ser l'error d'en Codd: adonar-se massa tard del fet que una relació pot tenir més d'una clau. (Una altra de les conseqüències d'adonar-se de l'error va ser la necessitat d'introduir la forma normal anomenada BCNF).

Un cop som conscients que tenim més d'una clau, no podem dir que la clau primària i la clau són sinònims. El màxim que podem dir és que escollim una de les claus, pel criteri que sigui (i aquest criteri pot ser l'atzar), i que usem aquesta clau com a clau primària.

Sigui com sigui, però, el concepte de clau primària continua lligat a la tecnologia de l'emmagatzematge, i no té res a veure amb el model relacional.

Fixem-nos que fins i tot el concepte de clau primària deixa de tenir sentit si usem un emmagatzematge on els tuples no s'accedeixen seqüencialemnt (per exemple, en fitxers invertits) o un emmagatzematge on cada tuple apareix en més d'una posició.

Per tot plegat, bandegem el terme de clau primària.

Clau i no pas clau primària

- Una clau és un concepte relacional
- La clau primària és un concepte d'emmagatzematge

5.5 Clau forana

Per tal d'introduir les *claus foranes* hem de començar per definir la *compatibilitat* entre els atributs.

Definició de *compatible*

- Direm que un domini D és compatible amb el domini D' si i només si $D' \subseteq D$.
- Direm que un atribut atr és compatible amb un atribut atr' si el domini de l'atribut atr és *compatible* amb el domini de l'atribut atr'

$$Domini(atr) \subseteq Domini(atr')$$

- Direm que un conjunt d'atributs X és compatible amb un conjunt d'atributs Y si es compleix que existeix una funció bijectiva $f : X \to Y$, anomenada funció de compatibilitat, tal que si $f(x) = y$ s'ha de complir que x és compatible amb y:

$$\forall x \in X : Domini(x) \subseteq Domini(f(x))$$

Una clau forana és un conjunt d'atributs en una relació tals que els seus valors es poden usar per a identificar un tuple d'una altra relació. És a dir, els valors de la clau forana coincideixen amb els valor d'una clau d'una altra relació.

Clau forana

Siguin dues relacions R i R'; sigui un conjunt d'atributs $X \subseteq R$ i sigui una funció de compatibilitat $f()$ entre R i R'.

Direm que X és una clau forana de R' (segons $f()$) si i només si $f(X)$ és una clau de R'.

Ho expressem amb:

$$X \overset{f(X)}{\longmapsto} R'$$

En moltes ocasions els noms dels atributs de $X \subseteq R$ i de $Y \subseteq R'$ són els mateixos, així com els seus dominis. En aquests casos no cal expressar la funció

de compatibilitat: $X \longmapsto R'$.

Sovint, donat el conjunt $X \subseteq R$ només hi ha un conjunt $Y \subseteq R'$ tal que X sigui compatible amb Y. En aquests casos, abusem del llenguatge i diem que X és una clau forana de R', sense indicar quina és la clau emprada de R. També ho expressem amb $X \longmapsto R'$.

Un altre cas molt habitual és considerar els conjunts d'atributs $X \subseteq R$ i $Y \subseteq R'$ com a seqüències. Llavors la funció de compatibilitat fa correspondre l'atribut i-èssim de X amb l'atribut i-èssim de Y. Sota aquestes condicions tota la informació de la compatibilitar recau en el conjunt Y, i per tant escriurem $X \longmapsto Y$.

En qualsevol de les anotacions, el nom de l'atribut pot anar prefixat pel nom de la relció, per tal d'evitar confusions: `taula.atribut`

Exemple 18. Clau forana

Siguin les següents relacions:

- $Productes = \{Nom, Codi, Preu, UbicacióMagatzem\}$
- $Magatzems = \{Ciutat, Responsable\}$

Siguin els següents estats de relació:

Nom	Codi	Preu	Ubicació
Cadira	P1	1200	Vilafranca
Taula	P2	2000	Cubelles
Llum	P3	350	Vilafranca
Telèfon	P4	175	Vilanova

Ciutat	Responsable
Vilanova	Zacaries
Vilafranca	Xènia
Cubelles	Valentí

Sigui la funció de compatibilitat de $Productes$ a $Magatzems$ següent:

$$f(UbicacióMagatzem) = Ciutat$$

Llavors tenim que l'atribut `UbicacióMagatzem` de $Productes$ és una clau forana, segons $f()$, de $Magatzems$

$$\{UbicacióMagatzem\} \overset{f()}{\longmapsto} Magatzems$$

Això significa que el valor de l'atribut `UbicacióMagatzem` correspon al valor de la clau d'un $Magatzem$.

Exemple 19. Clau forana composta

Seguim l'exemple 18, $^{\rightarrow}$ però hi introduïm un **canvi en els requeriments**

Suposem que en una ciutat hi pugui haver més d'un magatzem; i que dins d'una ciutat els magatzems es numeren. Així podem tenir els magatzems 1 i 2 de Vilanova, que són diferents dels magatzems 1 i 2 de Vilafranca. Una manera d'introduir aquest nou requeriment és

Exemple 18, pàgina 29

afegint un atribut `Num` a $Magatzems$. Llavors la[2] clau de $Magatzem$ és $< Ciutat, Num >$.

Ara, per tal que $Productes$ pugui contenir una clau forana de $Magatzems$ cal introduir l'atribut `NumMagatzem`. La nova funció de compatibilitat es:

$$f(\{UbicacióMagatzem, NumMagatzem\}) = \{Ciutat, Num\}$$

En conseqüència és el parell $\{UbicacióMagatzem, NumMagatzem\}$ que és clau forana de $Magatzems$:

$$\{UbicacióMagatzem, NumMagatzem\} \overset{f()}{\longmapsto} Magatzems$$

Exemple 20. Simplificació de la notació de les claus foranes

Exemple 19, pàgina 29

Seguim l'exemple 19.→

Tant `Nom` com `UbicacióMagatzem` són compatibles amb `Ciutat`; però també són compatibles amb `Responsable`, ja que tots quatre atributs comparteixen el mateix dominis, el de les cadenes de caràcters. Això obliga a explicitar d'alguna manera la funció de compatibilitat.

Per explicitar la funció de compatibilitat podem veure els conjunts de compatibilitat com a seqüències. Llavors, enlloc d'indicar la relació referenciada el que indiquem és el conjunt d'atributs:

$$\{UbicacióMagatzem, NumMagatzem\} \overset{f()}{\longmapsto} \{Ciutat, Num\} \subseteq Magatzems$$

Els noms `UbicacióMagatzem` i `NumMagatzem` incorporen una referència explícita a $Magatzem$. És a dir, hem encapsulat dins dels noms dels atributs la funció de compatibilitat. Gràcies a això, podem escriure:

$$\{UbicacióMagatzem, NumMagatzem\} \overset{f()}{\longmapsto} Magatzems$$

Una altra manera d'encapsular la funció de compatibilitat dins dels noms dels atribut és usar els mateixos noms. Per exemple, podem fer els següents rebateigs: `UbicacióMagatzem`→`Ciutat` i `NumMagatzem`→`Num`. I llavors:

$$\{Ciutat, Num\} \longmapsto Magatzem$$

Encapsular la funció de compatibilitat dins dels noms dels atributs té sentit quan només hi ha un possible conjunt d'atributs compatible entre ambdues relacions. En cas contrari, cal alguna manera d'indicar quins són els atributs que pertanyen al mateix conjunt.

Si volem explicitar quina és la taula de l'atribut, per claredat o per desambiguar (en el cas que un mateix nom d'atribut es repeteixi en taules diferents), prefixarem l'atribut amb el nom de la taula. Per exemple:

[2] Diem *la* clau perquè en aquest cas particular només n'hi ha una, de clau.

$$\{Productes.Ciutat, Productes.Num\} \longmapsto Magatzem$$

Si explicitem les taules, podem expressar la funció de compatibilitat explícitament, segons la notació $X \longmapsto Y$. Per exemple:

$$\{Productes.Ciutat, Productes.Num\} \longmapsto \{Magatzems.Ciutat, Magatzems.Num\}$$

Exemple 21. Combinació de taules

Seguim amb l'exemple 18.→

Exemple 18, pàgina 29

Sigui la unió natural (join) de les dues taules:

$$\prod_{Ubicació=Responsable}(Productes, Magatzems)$$

Aquesta consulta retorna una taula (o vista) on a cada producte se li ha afegit la informació del magatzem on està ubicat:

Nom	Codi	Preu	Ubicació	Responsable
Cadira	P1	1200	Vilafranca	Xènia
Taula	P2	2000	Cubelles	Valentí
Llum	P3	350	Vilafranca	Xènia
Telèfon	P4	175	Vilanova	Zacaries

Les claus foranes, juntament amb l'operador d'unio natural (join) són el mecanisme emprat per combinar taules

6. Restriccions d'integritat

6.1 Catàleg de restriccions d'integritat

Una BDR és un conjunt de relacions, i un conjunt de restriccions d'integritat sobre els tuples que poden existir en un estat qualsevol de la BD:

Definició de Bases de dades relacionals (BDR)

BDR = {R} + restriccions(r(R))

Aquestes restriccions poden afectar una sola taula, o més d'una. Algunes restriccions són exigides pel model relacional; d'altres, pel problema modelat amb la BDR. Els següents quadres són un resum dels diferents tipus de restriccions d'integritat:

Restriccions d'integritat exigides pel model relacional

Sobre una taula

- Restriccions de domini
  - Tot domini només accepta valors atòmics i monoavaluats
  - El valorsd'un atribut ha de ser consistent amb el domini de l'atribut
- Unicitat de les claus

  En cap moment hi pot haver dos tuples amb el mateix valor de la clau
- Integritat d'entitats

  En un tuple, les claus no poden ser nul·les ni contenir atributs amb valor nul

Sobre dues taules

- Integritat referencial

  Tota referència que des d'un tuple $t_1(R_1)$ es faci a un tuple $t_2(R_2)$, s'ha de referir a un tuple t_2 existent en l'estat actual $r(R_2)$

Restriccions d'integritat exigides pel problema

- Dependència funcional

  La majoris de les restriccions exigides pel problema s'expressen en termes de dependència funcional$^{\rightarrow}$ sobre els atributs

Pàgina 35

6.2 Explicació de les restriccions d'integritat

Tot seguit intentem explicar el què i el perquè de cadascuna de les restriccions d'integritat. Algunes de les afirmacions que farem, però, necessiten una anàlisi més detallada., que deixem pels següents apartats.

Les restriccions sobre el domini són a dos nivells: abstracció i realització.

A nivell de l'abstracció, les restriccions sobre el domini defineixen els dominis possibles: només són possibles els dominis que els seus valors siguin atòmics i monoavaluats; descartem dominis amb valors estructurats o multiavaluats. Formalment:

$$\forall A \in R, A \text{ és atòmic i monoavaluat}$$

A nivell de la realització, les restriccions sobre el domini diuen valors són vàlids en cada columna d'un tuple: només hi pot aparèixer un dels valors admesos pel domini corresponent. Formalment:

$$\forall r(R), \forall t \in r(R), \forall A \in R, t[A] \in Domini(A)$$

Així, les restriccions sobre el domini formalitzen la idea d'atribut, tan a nivell d'abstracció, com a nivell de realització.

Tota relació que compleixi les restriccions sobre el domini direm que està en primera forma normal o 1FN.

La restricció de la unicitat de les claus correspon a la definició de súper-clau. La restricció de la integritat d'entitats correspon a la definició de clau.

La restricció de la integritat referencial assegura que les claus foranes realment referenciïn algun tuple en una altra taula, i que d'aquesta manera la combinació de taules sigui possible.

Finalment, les restriccions del problema concret s'expressen en termes de dependència funcional,$^{\rightarrow}$ o en termes de condicions sobre els valors dels atributs. Una dependència funcional expressa una relació que hi ha entre els valors de dos conjunts atributs de la mateixa taula; i per tant limita els tuples que són considerats vàlids.

Pàgina 35

6.3 Integritat d'entitats

Sigui X el conjunt d'atributs que constitueixen una clau; sigui x el conjunt de valors que un tuple qualsevol t dóna a X:

$$t[X] = x$$

Si X és una súper-clau, llavors x identifica un tuple t, en el sentit que per a qualsevol altre $t' \in r(R)$ s'ha de tenir que $t'[X] \neq x$.

Si X és una clau, qualsevol $y \subseteq x$ no serà suficient per a identificar t.

Suposem que el valor d'identificació de t és: $x = \{a, b, nul\}$. Com ho interpretem?

1. `Nul` significa "valor prescindible"
   - Estem dient que amb $\{a, b\}$ n'hi ha prou per a identificar t, independentment del valor del tercer atribut de la clau
   - Així, si més no pel tupla t, el conjunt $X = \{A, B, C\}$ és una súper-clau, però no pas una clau
2. `Nul` és un valor
   - Amb aquest interpretació, $\{a, b, c\}$ identifica a un tuple diferent de l'identificat per $\{a, b, null\}$
   - Aquesta interpretació, però, exigeix que `null` sigui un valor del domini. Però justament `null` es vol definir com la no existència de valor

De tot plegat en resulta que un valor `null` en un dels atributs d'una clau s'ha d'interpretar com que, com a mínim per a aquest cas particular, el valor que hi pugui haver en aquest atribut no és rellevant de cara a identificar el tuple. I per tant la clau està mal definida.

D'aquí la necessitat de la restricció de la integritat de les entitats: exigim que totes les claus estiguin ben definides.

6.4 Integritat referencial

Una clau forana és un conjunt d'atributs X en una relació R tals que els seus valors permeten identificar un tuple en una altra relació R': $X \longmapsto R'$. La restricció d'integritat referencial exigeix que realment estiguem identificant un tuple a R'.

No complir amb la integritat referencial significa que tenim una *referència trencada*: estem intentant identificar un tuple que no existeix.

Exemple 22. Referència trencada

Sigui la BDR de l'exemple 18.→

A la nostra taula de `Magatzems` hi tenim els responsables de *Vilanova*, *Vilafranca* i *Cubelles*.

Suposem que en la taula de `Productes` tenim un tuple on en l'atribut `Ubicació` hi ha el valor *Cunit*. En aquest cas estem davant

Exemple 18, pàgina 29

d'una **referència trencada**: desconeixem quin és el magatzem de *Cunit*!!

Si tenim una clau forana $X \longmapsto R'$, i es compleix la integritat referencial llavors, donat un tuple qualsevol de R, hem de tenir que $t[X]$ és un identificador d'un tuple t' de R'. I per tant, per la integritat d'entitats, cap dels valors de $t[X]$ pot ser nul.

Clau forana

Els atributs d'una clau forana:

- O bé són tots `nulls`
- O bé identifiquen un tuple de la taula referenciada, en l'estat actual de la BDR

6.5 Dependències funcionals

Direm que B és dependència funcional de A quan, en qualsevol estat de la relació, no hi pot haver dos tuples diferents que coincideixin en els atributs de A, però no en els de B. Així, si trobem dos tuples que concideixen en els valors de A, també han de coincidir en els valors de B.

Definició de dependència funcional

Direm que existeix una dependència funcional de B sobre A, $A \rightarrow B$, on A i B són subconjunts d'atributs de R, si i només si per a qualsevol parell de tuples t_1 i t_2 de R es compleix que:

$$t_1[A] = t_2[A] \Rightarrow t_1[B] = t_2[B]$$

Donada una súper-clau, els atributs d'una relació són dependència funcional d'ella:

- Sigui $X \subseteq R$ una súper-clau de R
- Per definició de súper-clau tenim que $t_1[X] = t_2[X] \Rightarrow t_1 = t_2$
- Per definició de la projecció, tenim que un tuple és la projecció del tuple sobre tots els atributs de la relació: $t_1 = t_1[R]$ i $t_2 = t_2[R]$
- Per tant, $t_1[X] = t_2[X] \rightarrow t_1[R] = t_2[R]$
- En conseqüència, $R \rightarrow X$

Tot tuple depèn funcionalment de les seves súper-claus

Exemple 23.

Exemple 21, pàgina 31

Sigui la $r(R)$ taula obtinguda a l'exemple 21.$^{\rightarrow}$ I sigui R la relació que la qual aquesta taula n'és un estat.

De la manera com s'ha construït la taula (per la combinació de `Magatzems` i `Productes`) sabem que hi ha una dependència funcional:

$$Ubicació \rightarrow Responsable$$

És fàcil veure com a $r(R)$, la coincidència del valor de la `Ubicació` coincideix amb la coincidència del valor del `Responsable`.

Aquesta dependència té sentit perquè sabem que en cada ubicació hi ha un únic responsable. En canvi, desconeixem si la dependència $Responsable \rightarrow Ubicació$ es compleix o no; tot depèn de si un mateix responsable ho pot ser de diverses ubicacions, o no.

7. Base de dades relacional

Pàgina 31

Recordem la definció de BDR donada més amunt:$^{\rightarrow}$

Definició de Bases de dades relacionals (BDR)

BDR = {R} + restriccions(r(R))

Pàgina 5

A partir d'aquesta definició de base de dades relacional, ja podem veure el paper de les regles de Codd.$^{\rightarrow}$

La regla fundacional diu que sobre la BDR només usem els operadors relacionals.

La regla de la informació diu que hi ha un únic element per a representar la informació: la taula. Així, tota informació del model o és un tuple o és el valor d'un atribut dins d'un tuple.

En una BDR no hi ha informació visible i informació oculta: totes les taules són consultables; d'elles es pot obtenir qualsevol dels seus tuples; i d'aquests es pot accedir al valor de qualsevol dels seus atributs. D'aquesta manera obtenim l'accés garantit.

El tractament sistemàtic dels nuls és força complex. De moment ens podem quedar amb la idea que la manca d'informació es representa amb un `null`; i que aquest pròpiament no és un valor, ans l'absència d'aquest.

La regla del llenguatge comprensiu exigeix la presència d'un llenguatge que tant pugui fer consultes relacionals com manipular l'estructura de la BDR; és a dir, capaç de crear i modificar esquemes i restriccions. En la majoria dels productes comercials aquest llenguatge és l'SQL$^{\rightarrow}$, tot i que el motor de la BDR executa àlgebra relacional.

Pàgina 19

El resultat d'una consulta és un nou estat de relació, anomenat vista. Si l'usuari modifica les dades d'aquesta vista el sistema ha de poder modificar consistentment les dades de les taules de les que prové la vista. Això és el que diu la regla d'actualització de les vistes.

Les vistes també són el mecanisme que permet l'alt nivell de manipulació i la independència física de les dades.

L'alt nivell de manipulació s'aconsegueix permetent fer operacions sobre les vistes, i no només sobre les taules reals.

Sense entrar massa en detalls, la independència física de les dades s'obté a base de diferenciar l'esquema visible per cada aplicació (l'esquema extern de la BD) del seu esquema intern. Tot esquema extern és el resultat d'una consulta sobre l'esquema intern; això permet canviar l'esquema intern sense problemes, sempre i quan la consulta que genera l'esquema extern doni el mateix resultat.

Suposem que en una BDR es fa un canvi en el model lògic: afegim o traiem atributs; afegim o traiem taules. Sobre la BDR resultant podem construir unes vistes (consultes) que recuperin les taules del model original. Així, si les aplicacions antigues operen sobre aquestes vistes, i no pas sobre les taules reals, els canvis efectuats no afecten les aplicacions. En essència, aquest mecanisme per assolir la independència lògica de les dades és el mateix independència física de les dades; ara però, enlloc de dfiferenciar entre l'esquema intern de l'esquema extern, el que tenim és un conjunt d'esquemes externs diferents.

La independència de la integritat s'aconsegueix amb la arquitectura interna de les BDR. Aquest no és el lloc per entrar-hi, però la idea és que tota BDR té unes taules específiques que contenen la definició de la BD i de les seves restriccions. Més que una propietat de la BDR és una propietat del SGBDR, és a dir, del producte comercial que gestiona la BDR.

La independència de la distribució també és un tema tecnològic. Però la base és la mateixa: les aplicacions treballen amb un esquema extern centralitzat, que s'ha obtingut a partir d'esquemes distribuïts. És una propietat del SGBDR.

La regla de la no subversió és la complementària de la regla fundacional: volem usar eines relacionals, i només aquestes. És una propietat del SGBDR, que impedeix poder-se saltar l'operativa relacional.

8. Anomalies

8.1 Anàlisi d'anomalies

Anomenem anomalia aquella situació en què el disseny de la BD dificulta o impedeix de mantenir correctament la informació.

Una anomalia pot dificultar o impedir el manteniment de:

- La semàntica o consistència de les dades
- Les restriccions d'integritat de la BD
- La informació coneguda

La possibilitat d'anomalies significa que pot ser complicat, si no impossible, mantenir la informació i la seva consistència i integritat

L'estudi de les anomalies permet cercar millors dissenys per a les BD

Exemple 24. Enunciat comú pels exemples d'anomalies

Sigui una BDR formada per una sola relació:

Contractacions = {nomTreb, dataContractació, #Departament, adreçaDep}

El subratllat indica les claus.

El predicat de la relació és:

El treballador, que s'identifica amb el seu nom `nomTreb`, va ser contractat el dia `dataContractació`, i actualment està assignat al departament `#Departament`, que està en l'adreça `adreçaDep`

Exemple 25. Anomalia d'inserció

- Problemes per assegurar la consistència
  - Cal assegurar que en totes les contractacions d'un mateix departament haguem introduït la mateixa adreça del departament
  - El disseny de la BD fa que en la inserció d'una nova contractació sigui difícil assegurar la consistència de les dades

- Introducció de nuls
  - Què passa si hem contractat algú, però no està encara assignat a cap departament?
  - En aquest cas tant el `#Departament` com l'`adreçaDep` han de valdre `null`
  - El disseny de la BDR fa que en la inserció d'una nova contractació s'hagin d'introduir `nuls`, i no de qualsevol manera. El resultat és una dificultat en la inserció (a part de l'aparició de `nuls`, que per d'altres motius no és una bona idea)
- Violació de la restricció d'integritat
  - Com ho fem per a crear un departament?
  - Amb la BDR que tenim només hi ha una possibilitat: introduir una contractació falsa. És a dir, una `contractació` on tota la informació del treballador sigui `null`.
  - A part de la introducció dels `nulls` ara tenim un altre problema: la clau de la `contractació` té un valor `nul`!! I per tant es viola la restricció d'integritat de les entitats.

Exemple 26. Anomalia d'esborrat

- Pèrdua indesitjada d'informació
  - Què passa si eliminem la `contractació` que correspon a l'únic `treballador` d'un `departament`?
  - Com que la `contractació` suprimida és l'única amb aquest `departament`, perdem tota la informació que teníem associada al `departament`, com el seu nom i adreça.

Exemple 27. Anomalia d'actualització

- Exigència de propagar els canvis
  - Què passa si modifiquem, en una `contractació`, el nom d'un `departament`?
  - Si volem mantenir la consistència de les dades cal propagar el canvi a totes les `contractacions` sobre el mateix `departament`.
  - Per tant, el disseny de la BD dificulta mantenir la consistència de les dades davant d'una actualització

Una anomalia és el resultat d'un mal disseny de la BD. Aquest mal disseny és el resultat d'emprar relacions amb un predicat de relació anòmal.

Els predicats de relació anòmals són de fet la conjunció de predicats més simples. Per tal de millorar el disseny de la BD el que cal és *trencar* els predicats de relació, fins a aconseguir que cada relació tingui un predicat simple o

atòmic. Quan en una BD tenim totes les relacions amb predicats simples, que ja no es poden separar en d'altres predicats, direm que la BD està normalitzda.

Exemple 28. Normalització d'un predicat

Exemple 24, pàgina 38

La `contractació` de l'exemple 24→ diu tres coses diferents:

- Dóna informació del treballador, com el seu codi i el seu nom
- Dóna informació del departament, com el seu nom i la seva adreça
- Dona informació del lligam entre un treballador i un departament

Per tant, el predicat de la contractació és de fet la conjunció de tres predicats. Expressem-ho en forma relacional:

`Empleats` = {nomTreb, dataContractació} $Pred_1$

`Departaments` ={#Departament, adreçaDep} $Pred_2$

`Assignacions` = {#Dep,nomTreb} $Pred_3$

$Pred_1$ = El treballador que es diu `nomTreb` va ser contractat el dia `dataContractació`

$Pred_2$ = El departament de nom `#Dep` està ubicat a l'adreça `adreçaDep`

$Pred_3$ = El treballador de nom `nomTreb` està assignat al departament de nom `nomDep`

És a dir:

$$Contractacions = Empleats \wedge Departaments \wedge Assignacions$$

Les tres relacions que ara tenim, no són independents; estan lligades a través de claus foranes:

$$Assignació.\#Dep \longmapsto Departament.\#Departament$$

$$Assignació.\#Treb \longmapsto Empleat.nomTreb$$

Exemple 29. Desaparició de les anomalies

Exemple 28, pàgina 40

Seguim amb l'exemple 28.→

El procés de normalització ha implicat convertir una BD amb una única interrelació, en una BD amb tres interrelacions i dues claus foranes.

La nova BD és força més complexa, però s'han reduït les situacions

d'anomalia.

- Inserció d'una nova *contractació* →
  - Cal inserir un tuple a `Empleats` i una a `Assignacions`
  - La taula `Departaments` no hi està involucrada, i per tant no cal tornar a donar l'adreça del departament
- Contractació sense *assignació* →
  - Si contractem algú sense assignar-lo a cap departament, n'hi ha prou en imserir un tuple a `Empleats`
  - Per tant, no tenim la necessitat de mantenir atributs amb valor `null`
- Creació d'un *departament* →
  - N'hi ha prou en inserir un tuple a la taula `Departaments`
  - Les taules `Assignacions` i `Empleats` no hi estan involucrades
  - La creació d'un departament és possible, i no violem les restriccions d'integritat
- Esborrat d'una *contractació* →
  - Per esborrar una contractació cal eliminar un tuple d'`Assignacions` i un d'`Empleats`
  - La taula `Departaments` no hi està involucrada, i per tant manté tota la seva informació
- Modificació del nom d'un *departament* →
  - L'única taula que caldria involucrar és la de `Departaments`
  - Si el nom del departament és la clau forana emprada des de les `Assignacions` tenim una anomalia: caldria modificar tots els tuples d'*Assignacions* que tinguin aquest nom en la seva clau forana
  - Si la clau forana $Assignació \longmapsto Departaments$ no és el nom de departament, l'única taula involucrada és la de `Departaments`

Compareu amb l'exemple 25, pàgina 38
Compareu amb l'exemple 25, pàgina 38
Compareu amb l'exemple 25, pàgina 38
Compareu amb l'exemple 26, pàgina 39
Compareu amb l'exemple 27, pàgina 39

8.2 Bases de dades relacionals i anomalies

Sovint es diu que un dels principals objectius del es bases de dades és evitar la redundància. Però això no és cert; de fet una BDR esrtà carregada de redundàncies:

- Diari de fons (`log`)
  - Per tal de poder gestionar degudament les transaccions→

Pàgina 4

els SGBDR mantenen un diari de fons que enregistra totes les operacions fetes. Així si una transacció fallaes pot tornar a la situació inicial
  - També es pot usar el diari de fons per tal de recuperar-se d'una fallada del sistema
- Còpies de seguretat (`snapshot`)
  - Un mecanisme molt habitual de seguretat és mantenir una o més còpies de tota la BDR, per si de cas ocorre un desastre
- Mecanismes d'accés
  - Per facilitar els accessos, les BDR mantenen índexs, memòries cau, vistes precalculades, etc. Cadascuna d'aquestes estructures és del tot redundant

Redundància en una BDR

- Una BDR és altament redundant
- La redundància d'una BDR, però, és una redundància controlada

El que realment importa en una BDR és evitar al màxim les anomalies. Per aconseguir-ho, s'han definit un conjunt de restriccions d'integritat que ha de complir una BDR per tal que no tingui determinades anomalies. Quan una BDR compleix un d'aquests conjunts de restriccions d'integritat, diem que la BDR està en la forma normal descrita per aquest conjunt de restriccions.

Exemple 28, pàgina 40

Anomenem normalitzar una BDR a transformar-la fins a aconseguir una BDR equivalent però que estigui en una forma normal. L'exemple 28$^{\rightarrow}$ és un exemple de normalització (tot i que no sabem res de les formes normals, hem vist com el resultat ha perdut moltes anomalies).

Un dels mecanismes de reducció de les anomalies és el de suprimir de les taules tota aquella informació que es pugui calcular, com els atributs o les interrelacions derivades. Per això sovint es diu que una BDR conté la informació mínima necessària per a calcular allò que és rellevant. Però:

- **Observació 1**. Recordem que en especificació cal indicar tota la informació rellevant del problema
- **Observació 2**. Sovint d'interpreta aquesta reducció al mínim com el fet que no es permet la redundància. Però com hem vost això no és així:
  - Una BDR és altament redundant
  - El que no admet una BDR és aquella redundància que pot generar anomalies. Per tant, el que volem evitar no és la redundància, ans l'anomalia

Objectiu principal del disseny d'una BDR

L'objectiu principal del disseny d'una BDR és aconseguir reduir al màxim la possibilitat d'anomalies

9. A mode de resum

El següent llistat resumeix les característiques principals d'una BDR:

- Són um mecanisme d'implementació de la persistència, que cerca la independència de les dades
- Es basa en un model matemàtic, el model relacional
  - És un model simple
  - L'únic element estructural del model és la relació
    - Una relació té una presentació visual simple i entenedora, en forma de taules
  - Es poden definir una àlgebra relacional i un càlcul relacional
    - Són equivalents
- Els SGBDR acostumen a manipular les taules amb el llenguatge SQL i/o el llenguatge QBE
  - Les versions comercials no són 100% relacionals
  - L'SQL no és pas un llenguatge estàndard
  - Els motors executen àlgebra relacional
- Una BDR és un conjunt de relacions més un conjunt de restriccions d'integritat
  - Algunes d'aquestes restriccions estan imposades pel model relacional
    - Un cas especial és el de les clau, que són un tipus de dependència funcional
  - D'altres, són imposades pel problema. D'aquestes les més importants són les expressables en termes de dependència funcional
- L'objectiu principal del disseny d'una BDR és evitar les anomalies
  - La redundància és admesa, i especialment volguda
  - El que no es permet és aquella redundància que generi anomalies