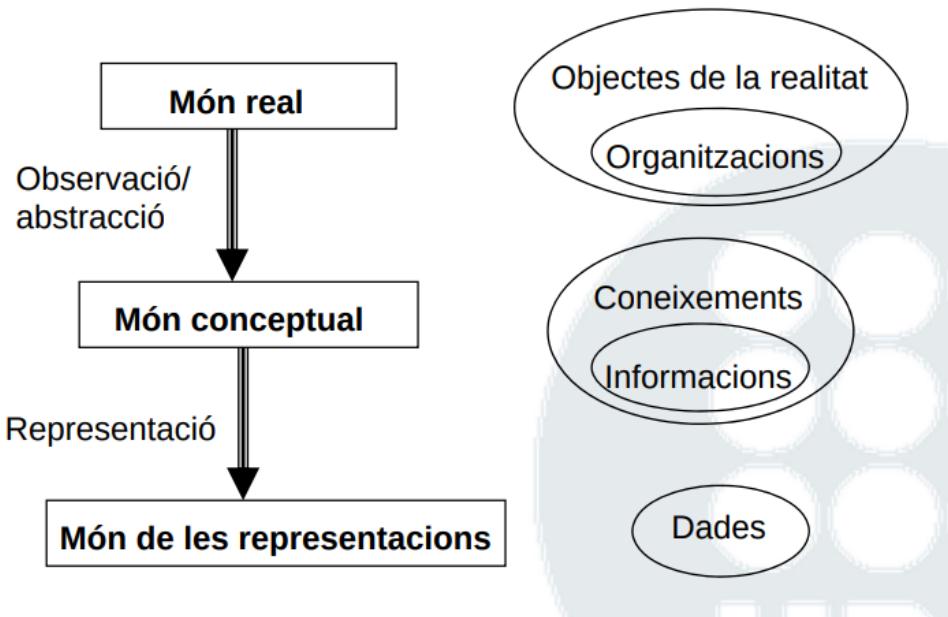
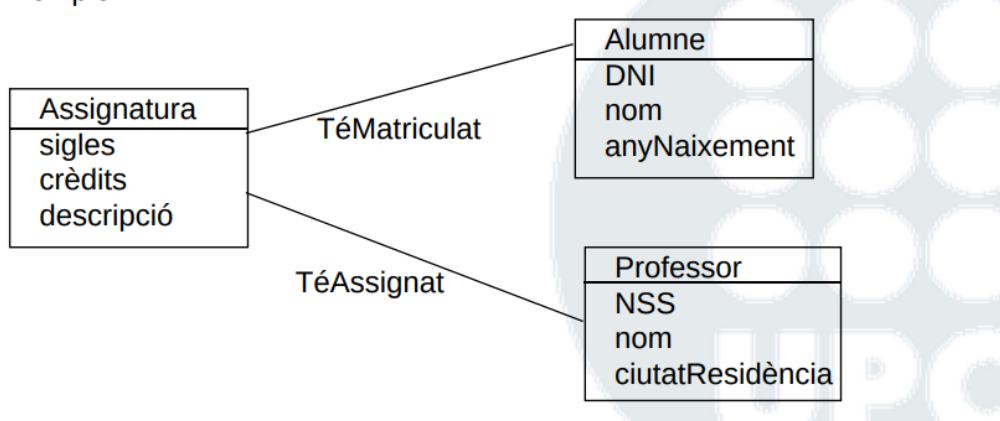


Els tres mons



El món conceptual

- Classe d'objectes:** Descriu un conjunt d'objectes que comparteixen propietats, associacions amb altres objectes i tenen una semàntica comuna.
- Atribut:** Propietat compartida pels objectes d'una classe.
- Associació:** Interrelació entre classes d'objectes.
- Exemple:**



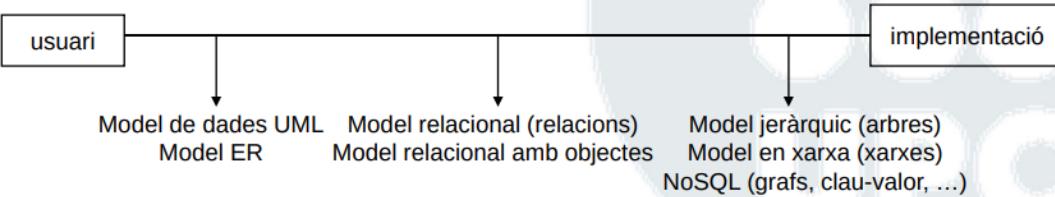
Concepte de BD

- Una base de dades és una col·lecció de dades que permet representar classes d'objectes, els seus atributs i les seves associacions i que està gestionada per un SGBD.

Model de BD: Conjunt de components o eines conceptuals que proporciona un SGBD per estructurar i manipular les dades.

- **Estructures de dades** amb les quals es pot construir la BD.
- **Operacions** per manipular i consultar les dades.
- **Regles d'integritat** que l'SGBD haurà de fer complir a les dades.

• Models de BD



Els SGBD ens proporcionen:

- Persistència**
- Eficiència en l'accés**
- Emmagatzematge de quantitats massives de dades**
- Accés multi-usuari**
- Seguretat**
- Fiabilitat**
- Conveniència**

Accés multi-usuari

- Un objectiu fonamental dels SGBD és permetre que diversos usuaris puguin accedir concurrentment a la mateixa BD.
- Per a tractar els accessos concurrents els SGBD fan servir el concepte de transacció:
 - **Transacció:** conjunt d'operacions simples que s'executen com una unitat.
 - Ex: Transferència de diners d'un compte X a un compte Y
 - Càrrec a X de la quantitat q de diners
 - Abonament a Y de la mateixa quantitat q de diners
- Una transacció pot acabar de dues maneres:
 - COMMIT: Transacció confirmada.
 - ROLLBACK: L'SGBD ha de desfer la transacció.
- Poden donar-se **problemes de concorrència** entre transaccions.

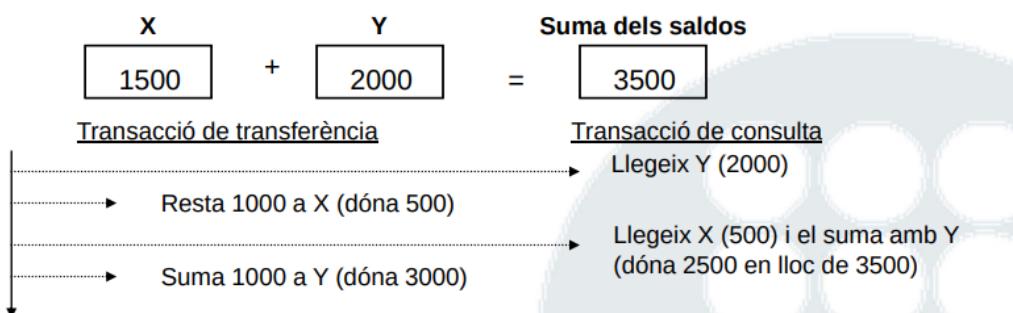
Seguretat

- El terme **seguretat** se sol utilitzar per fer referència als temes relatius a la confidencialitat, les autoritzacions, els drets d'accés, etc.
- Un objectiu dels SGBD és el de garantir que les dades emmagatzemades només poden ser accedites per les persones autoritzades i de la forma autoritzada.
- Per assolir aquest objectiu cal:
 - **Identificació i autentificació dels usuaris:** mitjançant una paraula clau, targeta, tècniques biomètriques, ...
 - **Possibilitat de definir autoritzacions o drets d'accés:** Una autorització dóna dret a un usuari o grup d'usuaris a accedir a unes determinades dades de la BD per fer unes determinades operacions.
 - La definició de les **dades** que s'autoritzen s'ha de poder fer amb diferents nivells de granularitat: al nivell global de tota la BD, al nivell de classe, al nivell d'atribut.
 - S'han de poder autoritzar **operacions** concretes sobre les dades: consulta, inserció, esborrat, modificació.

Fiabilitat

- **Exemple** de problema de concorrència entre transaccions:

Transferència de 1000 euros del compte X al compte Y concurrentment amb una consulta de la suma dels saldo d'X i Y.



- Per aconseguir que les transaccions s'executin com si estiguéssin aïllades, els SGBD usen diverses tècniques, la més coneguda de les quals és el **bloqueig** (lock)
- El bloqueig d'unes dades per una transacció consisteix a posar limitacions als accessos que les altres transaccions podran fer a aquestes dades.

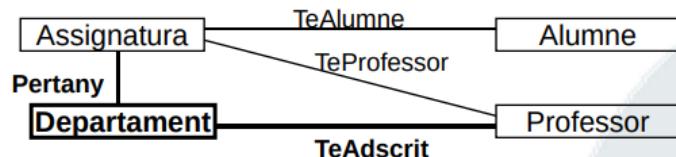
- **Redundàncies controlades:**
 - Pot interessar tenir una dada calculable redundant si es vol millorar el rendiment de la seva consulta.
 - Caldrà refer el càlcul de la dada redundant cada vegada que hi ha una modificació a les dades que serveixen per fer el càlcul.
 - És aconsellable que el mateix SGBD refaci el càlcul o que controli que el càlcul es refaci adequadament per evitar inconsistències.
- **Transaccions:**
 - No s'han de perdre els efectes d'una transacció confirmada.
 - Els canvis produïts per una transacció fallida s'han de desfer.
- **No pèrdua de dades.**

Conveniència

- **Flexibilitat als canvis:** Ha de ser fàcil fer canvis.

Canvis possibles:

- Canvis conceptuais: afegir/ suprimir atributs, classes d'objectes o associacions.



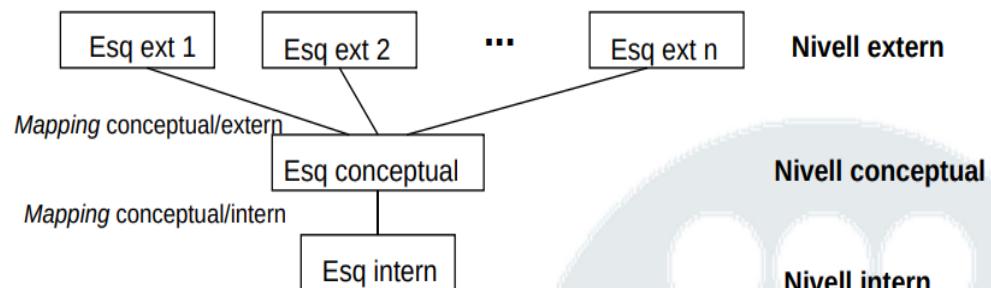
Afegim la classe
Departament i les seves
associacions

- Canvis a l'enregistrament físic de les dades.

- L'arquitectura ANSI/SPARC està orientada a l'obtenció de la independència física i lògica de les dades.
- Els usuaris han de poder fer consultes de qualsevol tipus i **complexitat** directament a l'SGBD.
- Aquestes consultes poden **no ser preestablertes** i l'SGBD ha de respondre immediatament, és a dir, sense que s'hagi d'escriure, compilar i executar un programa específic per a cada consulta.
- L'usuari ha de poder formular la consulta en un **llenguatge senzill i declaratiu**:
 - Que no requereixi coneixements de la representació física per a la formulació de la consulta.
 - El sistema l'ha d'interpretar directament.
- El llenguatge estàndard relacional, **SQL**, facilita l'assoliment d'aquest objectiu.
- Els SGBD inclouen la funció del **processament de consultes** que fa la transformació de la consulta a passos de baix nivell que la implementen i alhora l'optimitzen.

- L'SGBD ha de **proporcionar mecanismes** per:
 - La **definició** de les restriccions d'integritat dels usuaris.
 - El **manteniment** de les regles d'integritat del model i de les restriccions d'integritat dels usuaris.
 - Facilitar el control de la redundància de les dades
- **Causes** externes a les transaccions que poden interferir en els objectius anteriors:
 - **Fallades del sistema.**
 - **Fallades del sistema d'emmagatzematge.**
- Per fer front a aquests casos, és necessari que l'SGBD pugui fer una **recuperació** de les dades.
- Els mecanismes de recuperació dels SGBD es basen en:
 - Obtenció de **còpies de seguretat** periòdiques (backup).
 - Manteniment continu d'un **dietari** (log) on l'SGBD va enregistrant totes les operacions d'actualització que fan les transaccions.

Arquitectura ANSI/SPARC d'un SGBD



- **Nivell intern:** correspon a l'emmagatzematge de les dades.
- **Nivell conceptual:** correspon a l'estrucció de la base de dades per a tota la comunitat d'usuaris.
- **Nivell extern:** correspon a les diferents visions dels diferents tipus d'usuaris o aplicacions que utilitzen la BD. Hi haurà un esquema extern per cadascun.
- **Esquema:** descripció d'elements que pertanyen a un determinat nivell.

Arquitectura ANSI/SPARC: Contingut dels esquemes

- **Esquema conceptual:** Conté la descripció de:

- Classes d'objectes
- Atributs
- Associacions
- Restriccions d'integritat

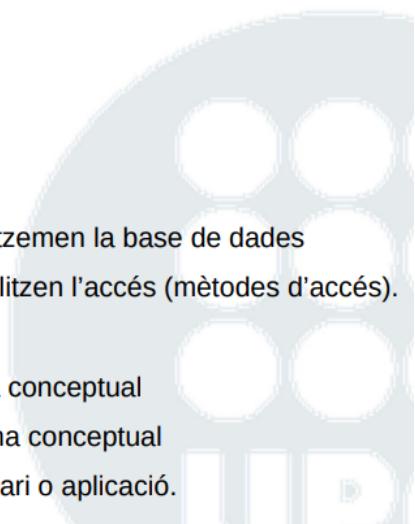
en termes del model de dades de l'SGBD.

- **Esquema intern:** Conté la descripció de:

- Organització dels fitxers que emmagatzemem la base de dades
- Estructures de dades auxiliars que agilitzen l'accés (mètodes d'accés).

- **Esquema extern:** Conté:

- Un subconjunt de dades de l'esquema conceptual
- Dades calculables a partir de l'esquema conceptual que interessen a un determinat tipus d'usuari o aplicació.



Arquitectura ANSI/SPARC: Independència de les dades

- L'arquitectura de tres nivells ANSI/SPARC ens permet aconseguir:
 - Independència física de les dades.
 - Independència lògica de les dades.

Arquitectura ANSI/SPARC: Independència de les dades lògica i física

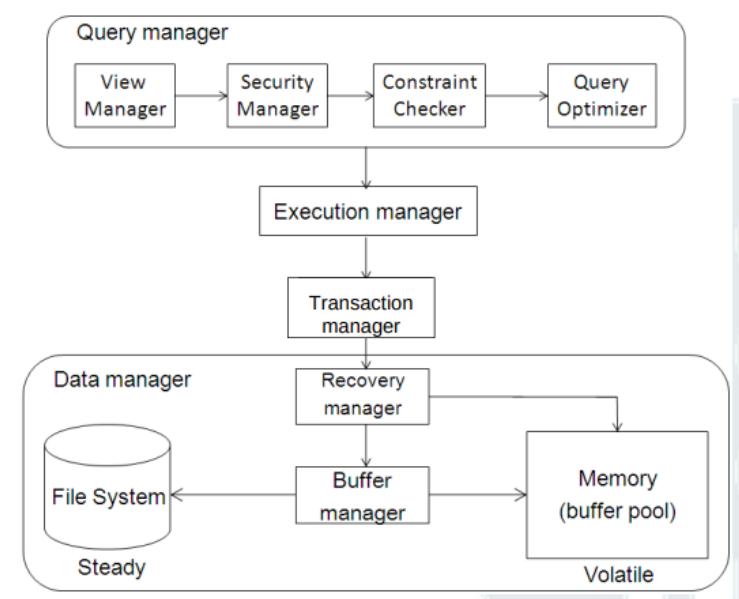
- **Independència física de les dades:**

- **Canvis a l'esquema intern:** No afecten a l'esquema conceptual ni als esquemes externs
 - Canvis possibles a l'esquema intern: canvis als mètodes d'accés, mida de les pàgines, etc.
 - Els programes d'aplicació i els usuaris directes no es veuran afectats per aquests canvis. En canvi, sí que caldrà refer les correspondències entre l'esquema conceptual i l'intern i, en la majoria de casos, caldrà refer la BD física.

- **Independència lògica de les dades:**

- **Canvis a l'esquema conceptual:** Un canvi d'aquest tipus no afectarà als esquemes externs que no facin referència a les classes, atributs o associacions modificats.
- **Canvis als esquemes externs:** Un canvi en un esquema extern no afectarà als altres esquemes externs, ni l'esquema conceptual, ni l'esquema intern.

Arquitectura d'un SGBD



Usuaris de BD

• Usuaris informàtics:

- Dissenyadors de bases de dades
- Programadors d'aplicacions
- Administradors de la BD
- Implementadors de SGBD

• Usuaris no informàtics:

– Usuaris paramètrics:

- Usen programes d'aplicacions prèviament dissenyats, implementats i provats pels usuaris informàtics.
- No han de conèixer la BD ni cap llenguatge de BD.

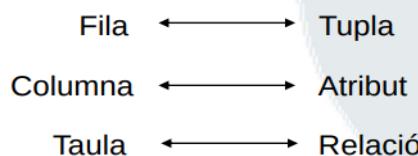
– Usuaris finals:

- Necesiten realitzar ocasionalment consultes (no previstes) a la BD i la informació a accedir pot ser diferent en cada ocasió.
- Usen un llenguatge de consulta d'alt nivell per programar-les ells mateixos.
- Han de conèixer part de l'estructura de la BD i el llenguatge de consulta.

Estructura de dades: Visió informal d'una relació

EMPLEATS	DNI	Nom	Sou
	40.444.255	Maria Domínguez	1500
	33.567.711	Pere Roca	2000
	55.898.425	Carles Bueno	2500
	77.232.144	Elena Pla	2000

- Informalment, cada relació pot visualitzar-se com una **taula**
- **Fila** de la taula: Col·lecció de valors de dades relacionats entre ells
- El nom de la **taula** i els noms de les **columnes** ajuden a interpretar el significat dels valors
- Tots els valors d'una columna són d'un mateix **domini**



Estructura de dades: Components d'una relació i esquema de la relació

EMPLEATS	DNI	Nom	Sou
	40.444.255	Maria Domínguez	1500
	33.567.711	Pere Roca	2000
	55.898.425	Carles Bueno	2500
	77.232.144	Elena Pla	2000

- Una **relació** es compon de:

- esquema de la relació o intensió de la relació (capçalera)
- extensió de la relació (cos)

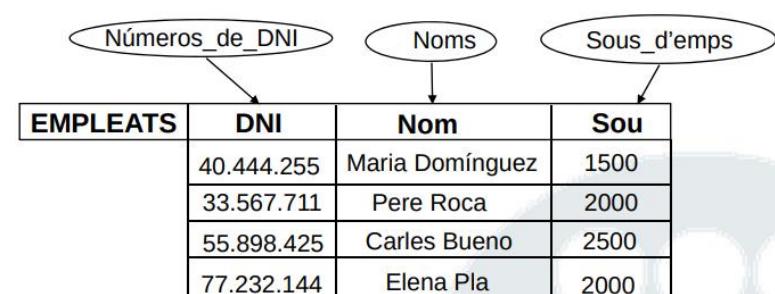
- **L'esquema de la relació** es compon de:

- nom de la relació. Ex: EMPLEATS
- conjunt d'atributs. Ex: {DNI, Nom, Sou}

- L'esquema de la relació es denota $R(A_1, A_2, \dots, A_n)$

– Ex: EMPLEATS(DNI, Nom, Sou)

Estructura de dades: Atribut i domini



- Un **atribut** indica el rol o paper que exerceix un domini en un esquema de relació

- Un **domini** és un conjunt de valors **atòmics**

- Els dominis poden ser:

- Predefinits. Ex.: INTEGER
- Definitos per l'usuari
Ex: Números_de_DNI, Noms, etc.

Estructura de dades: Valors nuls

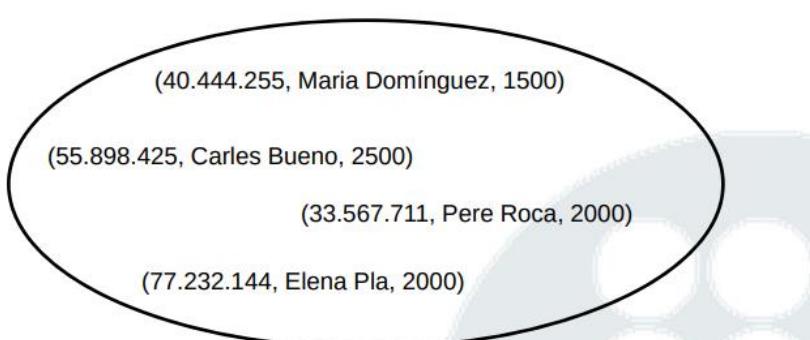
- El valor d'un atribut per a una tupla concreta d'una relació pot ser **desconegut o inaplicable**

EMPLEATS	DNI	Nom	Telèfon
	40.444.255	Maria Domínguez	937885146
	33.567.711	Pere Roca	desconegut
	55.898.425	Carles Bueno	inaplicable
	77.232.144	Elena Pla	934452435

- En aquests casos s'usa un **valor nul (NULL)**

EMPLEATS	DNI	Nom	Telèfon
	40.444.255	Maria Domínguez	937885146
	33.567.711	Pere Roca	NULL
	55.898.425	Carles Bueno	NULL
	77.232.144	Elena Pla	934452435

Estructura: Extensió de la relació



- **Extensió de la relació:** L'extensió d'una relació d'esquema $R(A_1, A_2, \dots, A_j, \dots, A_n)$ és un conjunt de tuples $t_i = <v_{i1}, v_{i2}, \dots, v_{ij}, \dots, v_{in}>$ on v_{ij} és un valor del domini d' A_j o bé un **valor nul**
- Una **tupla** és un element de l'extensió d'una relació

Estructura de dades: Cardinalitat i grau

EMPLEATS	DNI	Nom	Sou
	40.444.255	Maria Domínguez	1500
	33.567.711	Pere Roca	2000
	55.898.425	Carles Bueno	2500
	77.232.144	Elena Pla	2000

- **Cardinalitat d'una relació:** Es el nombre de tuples de la seva extensió

Ex: cardinalitat 4

- **Grau d'una relació:** Es el nombre d'atributs del seu esquema de relació

Ex: grau 3

Estructura de dades: Parany de la visió informal d'una relació

- Informalment, cada relació pot visualitzar-se com una **taula** o fitxer simple
- La visualització tabular suggereix algunes idees falses sobre les relacions
- Per tant convé precisar que en una relació:
 - Els valors dels atributs són atòmics
 - No hi ha tuples repetides
 - No hi ha ordre entre les tuples
 - No hi ha ordre entre els atributs

Estructura de dades: Claus d'una relació

- **Superclau d'una relació:** Subconjunt dels atributs de l'esquema de la relació que identifica les tuples de l'extensió de la relació
- **Clau d'una relació:** Superclau de la relació que no té cap subconjunt propi que sigui també superclau
També s'anomena **clau candidata** de la relació
- Exemples: EMPLEAT(DNI, NSS, Nom, Telèfon, Sou)
Superclaus: {DNI, NSS, Nom, Telèfon, Sou}
{DNI, Nom}, {DNI}, etc.
Claus candidates: {DNI}, {NSS}
DESPATX(Edifici, Número, Superfície)

C6	119	15
C6	120	15
D4	119	10

Clau candidata: {Edifici, Número}

Estructura de dades: Claus d'una relació - Clas foranes

- Les tuples de les relacions d'una base de dades poden requerir **connexions** entre elles
- Exemple: EMPLEAT i DEPARTAMENT
cada empleat està assignat a un departament
un empleat pot tenir un altre empleat que li fa de "cap"

DEPARTAMENT(NomDep, ...)	
Producció	Vendes

EMPLEAT(DNI, ...)	
40.444.255	NULL
33.567.711	40.444.255
55.898.425	33.567.711
77.232.144	NULL

, EmpleatCap,	DepAssig)
Producció	Producció
Producció	Vendes
- Una **clau forana d'una relació** és un subconjunt dels atributs de l'esquema de la relació que **referencia** una clau primària d'una altra relació o de la pròpia relació

Estructura de dades: Claus d'una relació - Clau primària i alternatives

- Una de les claus candidates es designa **clau primària**
- **Clau alternativa:** Clau candidata no designada primaria
- Exemple:

EMPLEAT(DNI, NSS, Nom, Telèfon, Sou)

Claus candidates: {DNI}, {NSS}

Clau primària: {DNI}

Clau alternativa: {NSS}

- Convenció: Se subratllen els atributs que formen la clau primària

Estructura de dades: Claus d'una relació - Clas foranes

- Una **clau forana** ha de complir que:
 - Té el mateix nombre d'atributs que la clau primària referenciada
 - Els atributs que la formen han de tenir dominis compatibles amb els de la clau primària referenciada

- Exemple:

DESPATX(Edifici, Número, Superfície)

EMPLEAT(DNI, ... ,
EdDespatx, NumDespatx)

Estructura de dades: Connexions al model relacional

- El model relacional és un model **uniforme** en el sentit que té només un tipus d'elements: les relacions
- Les **connexions** al model relacional es fan mitjançant les **claus foranes** de les pròpies relacions

Operacions

- Les **operacions** del model relacional han de permetre **manipular** les dades emmagatzemades a una base de dades relacional
- La **manipulació** inclou:
 - Actualització
 - Consulta
- **Actualització**:
 - Inserció de tuples a una relació
 - Esborrat de tuples d'una relació
 - Modificació de tuples d'una relació
- **Consulta**: Obtenció de dades deduïbles a partir de les relacions
- Segons la manera d'especificar les consultes els **llenguatges relacionals** poden classificar-se en:
 - Llenguatges basats en l'**àlgebra relacional**
 - Llenguatges basats en el **càlcul relacional**
- **SQL** és un llenguatge basat en el càlcul relacional però que incorpora també elements de l'àlgebra relacional

Regles d'integritat

- Regla d'integritat d'**entitat**
- Regla d'integritat **referencial**
- Regla d'integritat de **domini**

Regles d'integritat: Regla d'integritat d'entitat

- Fa referència a les **claus primàries** de les relacions
- Estableix que:
 - Els atributs que formen part de la clau primària han de tenir **valors únics** en conjunt (no repetits)
 - Cap atribut de la clau primària pot prendre el **valor nul**
- Exemple:

DESPATX(<u>Edifici</u> , Número, Superfície)		
C6	119	15
C6	120	15
D4	119	10
<u>C6</u>	<u>119</u>	<u>12</u>
<u>NULL</u>	<u>NULL</u>	<u>25</u>
<u>C6</u>	<u>NULL</u>	<u>10</u>
<u>NULL</u>	<u>119</u>	<u>8</u>

- **Motivació**: la clau primària ha de servir per identificar les tuples d'una relació

Regles d'integritat: Regla d'integritat referencial

- Fa referència a les **claus foranes** de les relacions
- Estableix que:
 - Els valors d'una clau forana poden ser només valors de la **clau primària referenciada** o **valors nuls**
- Exemple:

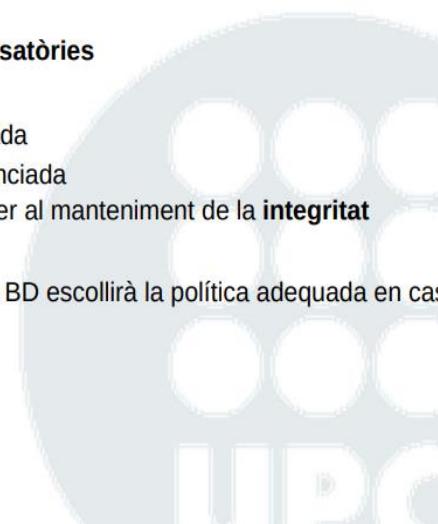
DEPARTAMENT(<u>NomDep</u> , ...)	
Producció	Vendes
Producció	Vendes
Vendes	NULL
Marketing	Marketing

EMPLEAT(<u>DNI</u> , ...)	
40.444.255	Producció
33.567.711	Producció
55.898.425	Vendes
77.232.144	NULL
25.250.333	Marketing

- **Motivació**: les claus foranes han de servir per establir connexions entre les tuples

Regles d'integritat: Manteniment de la integritat referencial

- Una **operació** d'inserció, esborrat o modificació pot condir a un estat que no satisfaci les regles d'integritat
- Manteniment de la integritat:
 - rebutjar** l'operació
 - acceptar-la i realitzar **accions compensatòries**
- En els casos següents:
 - esborrat** d'una clau primària referenciada
 - modificació** d'una clau primària referenciadapoden aplicar-se accions compensatòries per al manteniment de la **integritat referencial**
- Per a cada clau forana el dissenyador de la BD escollirà la política adequada en cas d'esborrat i en cas de modificació
- Algunes polítiques possibles:
 - Restricció**
 - Cascada**
 - Anul·lació**



Regles d'integritat: Integritat referencial - restricció

- No es permet** esborrar o modificar una clau primària referenciada en alguna clau forana
- Exemple:

CLIENT(NumClient, ...)	
10	
15	
18	

FACTURES_PENDENTS(NumFactura, ...)	
1234	10
1235	10
1236	15

Regles d'integritat: Integritat referencial - cascada

- En cas d'esborrar o modificar una clau primària referenciada en alguna clau forana, **s'esborren o modifiquen** totes les referencies
- Exemple:

EDIFICI(NomEdif, Direcció)		
C6		
D4		

DESPATX(Edifici, Número, Superficie)		
C6	119	15
C6	120	15
D4	119	10

- Aquesta política es pot aplicar recursivament

DESPATX(Edifici, Número, Superficie)		
A0	119	15
A0	120	15
D4	119	10

EMPLEAT(DNI, ...)		
	,EdDespatx, NumDespatx)	

Regles d'integritat: Integritat referencial - cascada

- En cas d'esborrar o modificar una clau primària referenciada en alguna clau forana, **s'esborren o modifiquen** totes les referencies
- Exemple:

EDIFICI(NomEdif, Direcció)
A0 ← C6
A0 ← D4

modificació de C6 per A0

DESPATX(Edifici, Número, Superficie)		
A0	119	15
A0	120	15
D4	119	10

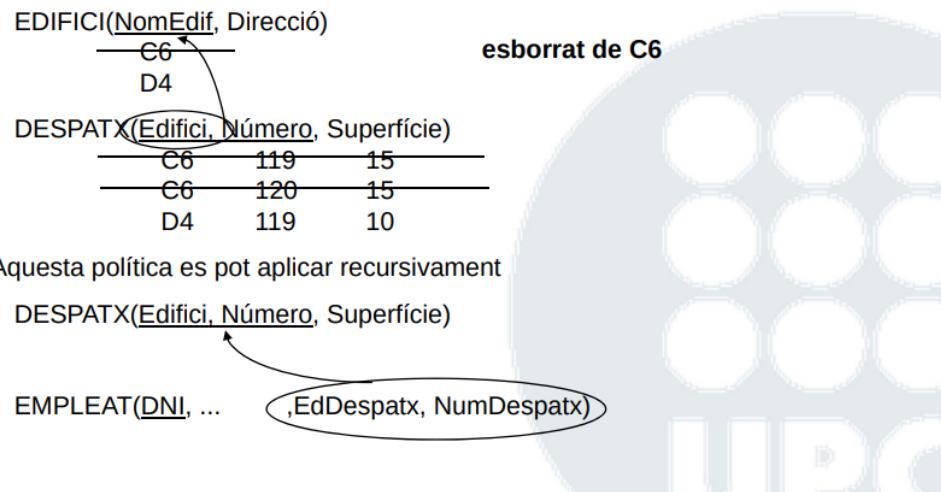
- Aquesta política es pot aplicar recursivament

DESPATX(Edifici, Número, Superficie)		
A0	119	15
A0	120	15

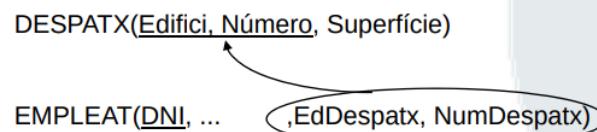
EMPLEAT(DNI, ...)		
	,EdDespatx, NumDespatx)	

Regles d'integritat: Integritat referencial - cascada

- En cas d'esborrar o modificar una clau primària referenciada en alguna clau forana, **s'esborren o modifiquen** totes les referències
- Exemple:

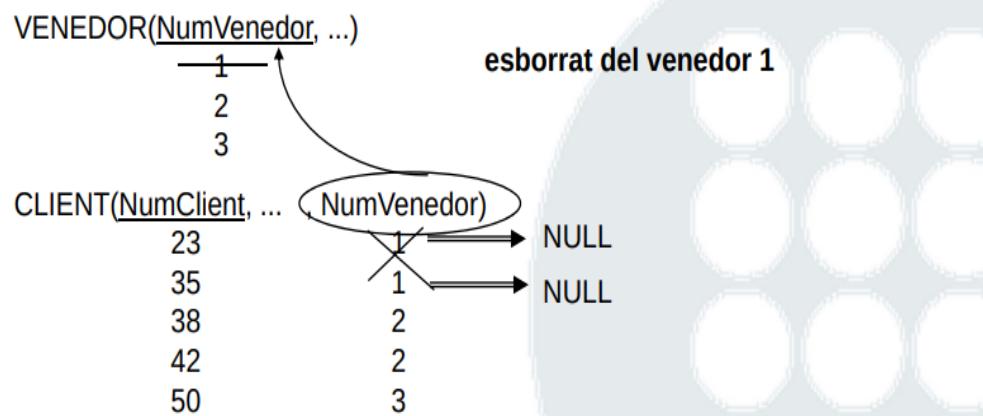


- Aquesta política es pot aplicar recursivament



Regles d'integritat: Integritat referencial - anul.lació

- En cas d'esborrar o modificar una clau primària referenciada en alguna clau forana, s'assignen **valors nuls** a totes les referències
- Exemple:



Regles d'integritat: Regla d'integritat de domini

Estableix que:

- Un valor no nul d'un atribut ha de pertànyer al domini de l'atribut

Exemple 1: PERSONA(DNI: Integer, ...)
2,5

NO

Exemple 2: domini EdatEmpl: Integer entre 16 i 65
EMPLEAT(...Edat: EdatEmpl...)
14

NO

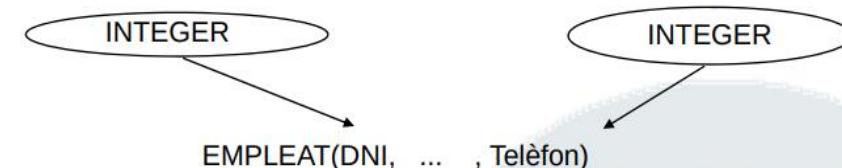
- Les operacions que es poden aplicar sobre els valors depenen del domini dels valors

Exemple: DNI = 'Pere Roca'

NO

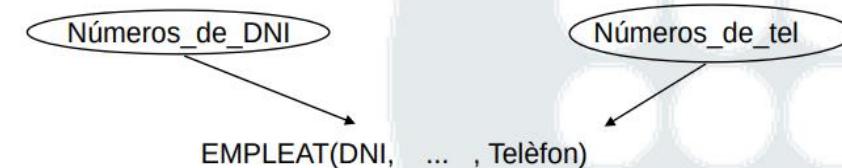
Regles d'integritat: Segona condició de la regla d'integritat de domini

- Dominis predefinits:



Consultar els empleats tals que DNI= Telèfon

- Dominis definits per l'usuari



Consultar els empleats tals que DNI=Telèfon

NO

- Supòrt en SQL: només per dominis predefinits

Base de dades exemple

- **Clau primària:** Cada taula té una clau primària que permet identificar les files de la taula. Per ex. **num_dpt** és la clau primària de la taula departaments. Això vol dir que cada departament té un num_dpt que ha de ser únic entre tots els departaments, és a dir mai hi haurà dos departaments amb el mateix número de departament.
- **Clau forana.** Una clau forana permet relacionar les files de dues taules. Per ex. **num_dpt** és una clau forana de la taula empleats que referencia la taula departaments. Això vol dir que entre les dades d'un empleat hi haurà també el departament al que pertany, i això ens permetrà saber el departament on treballa un empleat, i també els empleats que treballen a un departament.

departaments(<u>num_dpt</u> , nom_dpt, planta, edifici, ciutat_dpt)				
1	DIRECCIO	10	PAU CLARIS	BARCELONA
2	DIRECCIO	8	RIOS ROSAS	MADRID
3	MARQUETING	1	PAU CLARIS	BARCELONA
projectes(<u>num_proj</u> , nom_proj, producte, pressupost)				
1	IBDTEL	TELEVISIO	1000000	
2	IBDVID	VIDEO	500000	
empleats(<u>num_empl</u> , nom_empl, sou, ciutat_empl, (<u>num_dpt</u>), (<u>num_proj</u>))				
1	CARME	400000	MATARO	1 1
2	EUGENIA	350000	TOLEDO	2 2
3	JOSEP	250000	SITGES	3 1

Creació d'una taula

CREATE TABLE <nom_taula>

```
(<nom_columna> <tipus_dades> [<restriccions_col>] [<val_per_defecte>]
[, <nom_columna> <tipus_dades> [<restriccions_col>] [<val_per_defecte>]...]
[<restriccions_taula>]);
```

- **tipus_dades:** INTEGER, FLOAT(precisió), REAL, CHAR(n), NUMERIC(precisió, escala), DECIMAL(precisió, escala), SMALLINT, DOUBLE PRECISION, VARCHAR(n), DATE,....
- **val_per_defecte:** Valor per defecte d'una columna per a una fila que s'insereix a la taula.

DEFAULT { <literal> | NULL }.

Creació d'una taula: Restriccions de columna

- **restriccions_col:**

UNIQUE	La columna no pot tenir valors repetits
PRIMARY KEY	La columna és clau primària de la taula
REFERENCES <taula> [<col>]	La columna és clau forana que referencia la taula indicada.
CHECK (<condicions>)	La columna ha de complir les condicions especificades. Només pot referir-se a la columna per la que es defineix.
NOT NULL	La columna no pot tenir valors nuls

Les **restriccions de columna** es refereixen a una única columna.

Per exemple, en la condició d'un CHECK de columna no es pot fer referència a més d'una columna.

Creació d'una taula: Restriccions de taula

- **restriccions_taula:**

UNIQUE (<cols>)	El conjunt de les columnes especificades han de tenir valors únics entre les files de la taula
PRIMARY KEY (<cols>)	El conjunt de les columnes especificades formen la clau primària
FOREIGN KEY (<cols>) REFERENCES <taula> [<cols>]	El conjunt de columnes especificades formen una clau forana que referencia la taula indicada.
CHECK (<condicions>)	La taula ha de complir les condicions especificades. La condició pot referir-se a una o més columnes de la taula.

Les **restriccions de taula** poden referir-se a una o més columnes de la taula. Així, en cas de restriccions que tenen a veure amb més d'una columna cal usar una restricció de taula.

Per exemple, en cas de claus primàries compostes per més d'un columna o condició (CHECK) que tenen a veure amb més d'una columna.

Creació d'una taula: Exemple

CREATE TABLE empleats

```
( num_empl      INTEGER,  
  nom_empl     CHAR(30) NOT NULL,  
  sou          INTEGER DEFAULT 100000  
                  CHECK (sou>80000),  
  ciutat_empl   CHAR(30),  
  num_dpt       INTEGER,  
  num_proj      INTEGER,  
  PRIMARY KEY (num_empl),  
  FOREIGN KEY (num_dpt) REFERENCES departaments(num_dpt),  
  FOREIGN KEY (num_proj) REFERENCES projectes(num_proj));
```

Esborrat de files d'una taula

```
DELETE FROM <taula>  
WHERE <condicions>;
```

- S'eliminen de la **taula** les files que compleixen les **condicions** especificades a la clausula **WHERE**.

Inserció de files en una taula

```
INSERT INTO <nom_taula> [<columnes>]  
( VALUES {<valor1> | NULL}, ..., {<valorn> | NULL} ) | <consulta>;
```

- En cas de no posar les **columnes** a continuació del **nom_taula**, els valors han de correspondre exactament als valors de les columnes en el **CREATE TABLE** i en el mateix ordre.
- En cas de posar les **columnes**, els valors han de correspondre als valors de les columnes explicitades i en el mateix ordre.
- Els valors de les columnes de la fila o files a inserir es poden obtenir també com a resultat d'una **consulta** (veure subconsultes).

Inserció de files en una taula: Exemples

INSERT INTO empleats

```
VALUES (4, 'RICARDO', 400000, 'BARCELONA', 1, 1);
```

INSERT INTO empleats (num_empl, num_dpt, num_proj,
nom_empl)

```
VALUES (11, 3, 2, 'NURIA');
```

empleats(num_empl, nom_empl, sou, ciutat_empl, num_dpt, num_proj)					
1	CARME	400000	MATARO	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	400000	BARCELONA	1	1
11	NURIA	100000	NULL	3	2

■ Files inserides per la primera sentència
■ Files inserides per la segona sentència

Esborrat de files d'una taula: Exemples

```
DELETE FROM empleats  
WHERE num_dpt=2;
```

```
DELETE FROM empleats  
WHERE sou <= 250000;
```

empleats(num_empl, nom_empl, sou, ciutat_empl, num_dpt, num_proj)					
1	CARME	400000	MATARO	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	400000	BARCELONA	1	1
11	NURIA	100000	NULL	3	2

■ Files esborrades per la primera sentència
■ Files esborrades per la segona sentència

Modificació de files d'una taula

```
UPDATE <taula>
SET <col> = {expressió/ NULL} [,<col> = {expressió/ NULL}...]
WHERE <condicions>;
```

- Es modifiquen de la manera indicada a la clausula **SET** les columnes de les files de la **taula** que compleixen les **condicions** especificades a la clausula **WHERE**.

Modificació de files d'una taula: Exemples

```
UPDATE empleats
SET sou = sou +10000
WHERE num_dpt = 1;
```

```
UPDATE empleats
SET sou = sou + 50000, ciutat_empl = 'VIC'
WHERE num_empl = 11;
```

empleats(<u>num_empl</u> , nom_empl, sou, ciutat_empl, <u>num_dpt</u> , <u>num_proj</u>)					
1	CARME	410000	MATARO	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	410000	BARCELONA	1	1
11	NURIA	150000	VIC	3	2

■ Fils modificades per la primera sentència
■ Fils modificades per la segona sentència

Consultes sobre una taula: Format bàsic

```
SELECT <columnes_a_seleccionar> | *
FROM <taula_a_consultar>
[ WHERE <condicions> ] ;
```

- El resultat de la consulta és el valor de les **columnes_a_seleccionar** de la **taula_a_consultar** únicament per a la fila o files que acompleixen les **condicions** especificades a la clausula **WHERE**.
- En el cas de no posar la clausula **WHERE**, el resultat és el valor de les **columnes_a_seleccionar** per totes les files de la **taula_a_consultar**.
- Si posem un * en lloc de **columnes_a_seleccionar** indica que estem interessats en totes les columnes de la **taula_a_consultar**.

Consultes sobre una taula: Format bàsic - Exemple 1

```
SELECT *
FROM empleats;
```

empleats(<u>num_empl</u> , nom_empl, sou, ciutat_empl, <u>num_dpt</u> , <u>num_proj</u>)					
1	CARME	410000	MATARO	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	410000	BARCELONA	1	1
11	NURIA	150000	VIC	3	2

■ Dades obtingudes com a resultat de la consulta

Consultes sobre una taula: Format bàsic - Exemple 2

```
SELECT num_empl, nom_empl, sou  
FROM empleats;
```

empleats(num_empl, nom_empl, sou, ciutat_empl, num_dpt, num_proj)					
1	CARME	410000	MATARO	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	410000	BARCELONA	1	1
11	NURIA	150000	VIC	3	2

Dades obtingudes com a resultat de la consulta

Consultes sobre una taula: Format bàsic - Exemple 3

```
SELECT num_empl, nom_empl, sou  
FROM empleats  
WHERE num_dpt = 3;
```

empleats(num_empl, nom_empl, sou, ciutat_empl, num_dpt, num_proj)					
1	CARME	410000	MATARO	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	410000	BARCELONA	1	1
11	NURIA	150000	VIC	3	2

Dades obtingudes com a resultat de la consulta

Operadors en les condicions

operadors

- aritmètics: *, +, -, /
- de comparació: =, <, >, <=, >=, <>
- lògics: NOT, AND, OR
- altres:
 - <columna> BETWEEN <límit₁> AND <límit₂>
 - <columna> IN (<valor₁>,<valor₂> [...,<valor_N>])
 - <columna> LIKE <característica>
 - <columna> IS [NOT] NULL

Aquests operadors poden sortir a les **condicions**

- En la clausula **WHERE** de les sentències d'esborrat (**DELETE**), modificació (**UPDATE**) i consulta (**SELECT**)
- En la clausula **CHECK** de les sentències de creació d'una taula (**CREATE TABLE**).

Operadors en les condicions: Exemple

```
SELECT num_empl, nom_empl  
FROM empleats  
WHERE NOT(num_dpt = 2) AND  
( ciutat_empl IN ('MATARO', 'SITGES', 'BARCELONA') OR  
ciutat_empl LIKE 'V%') AND  
num_proj IS NOT NULL AND  
sou BETWEEN 400000 AND 500000;
```

empleats(num_empl, nom_empl, sou, ciutat_empl, num_dpt, num_proj)					
1	CARME	410000	MATARO	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	410000	BARCELONA	1	1
11	NURIA	150000	VIC	3	2

Dades obtingudes com a resultat de la consulta

Consultes sobre una taula: Ordenació

```
SELECT <columnes_a_seleccionar> | *  
FROM <taula_a_consultar>  
[ WHERE <condicions> ]  
ORDER BY <columna> [DESC | ASC].... ;
```

- En el resultat s'obté les dades ordenades segons les columnes que s'explicitin a la clausula **ORDER BY**.
- Si per a una columna no es posa **DESC** s'enten que la classificació, segons els seus valors, es vol que sigui ascendent. Cosa que també es pot demanar explícitament amb la paraula clau **ASC**.

Consultes sobre una taula: Ordenació - Exemple

```
SELECT num_empl, nom_empl, sou  
FROM empleats  
WHERE num_dpt IN (1,2)  
ORDER BY sou DESC, nom_empl;
```

num_empl	nom_empl	sou
1	CARME	410000
4	RICARDO	410000
2	EUGENIA	350000
12	NURIA	150000

emplats	num_empl	nom_empl	sou	ciutat_empl	num_dpt	num_proj
	1	CARME	410000	MATARO	1	1
	2	EUGENIA	350000	TOLEDO	2	2
	3	JOSEP	250000	SITGES	3	1
	4	RICARDO	410000	BARCELONA	1	1
	11	NURIA	150000	VIC	3	2
	12	NURIA	150000	MATARO	1	5

■ Files que compleixen la condició del WHERE

■ Dades tal com s'obtenen (ordenades) com a resultat de la consulta

Consultes sobre una taula: Resultats sense repeticions

```
SELECT [ DISTINCT | ALL ] <columnes_a_seleccionar>  
FROM <taula_a_consultar>  
[ WHERE <condicions> ] ;
```

- Si volem que el resultat d'una consulta se'n doni sense repeticions, cal utilitzar la paraula clau **DISTINCT**.
- Si no es posa res se'n donarà el resultat amb repeticions (en cas de que n'hi hagin). Cosa que també es pot demanar explícitament amb la paraula clau **ALL**.

Consultes sobre una taula: Resultats sense repeticions

```
SELECT DISTINCT nom_empl, sou  
FROM empleats  
WHERE num_dpt IN (1,3);
```

resultat

nom_empl	sou
CARME	410000
JOSEP	250000
RICARDO	410000
NURIA	150000

emplats	num_empl	nom_empl	sou	ciutat_empl	num_dpt	num_proj
	1	CARME	410000	MATARO	1	1
	2	EUGENIA	350000	TOLEDO	2	2
	3	JOSEP	250000	SITGES	3	1
	4	RICARDO	410000	BARCELONA	1	1
	11	NURIA	150000	VIC	3	2
	12	NURIA	150000	MATARO	1	5

■ Files que compleixen la condició del WHERE

■ Dades obtingudes com a resultat de la consulta

Consultes sobre una taula: Funcions d'agregació

```
SELECT <funcions_d'agregació>
FROM <taula_a_consultar>
[ WHERE <condicions> ] ;
```

- Són funcions que s'apliquen sobre el conjunt de files de la **taula_a_consultar** que compleixen les **condicions** especificades a la clàusula **WHERE**.

- COUNT:

- COUNT(*)** - número de files que compleixen la condició del where
- COUNT(DISTINCT <columna>)** – número de valors diferents de la columna, sense comptar valors NULL, per a les files que compleixen la condició del where.
- COUNT(<columna>)** – número de valors de la columna, sense comptar valors NULL, per a les files que compleixen la condició del where.

- SUM (expressió), MIN(expressió), MAX(expressió), AVG(expressió):

- expressió pot ser simplement una columna, o pot ser una càlcul a partir del valor de diferents columnes i constants.
- SUM**: dóna la suma dels valors resultants de calcular l'expressió per a les files que compleixen la condició del WHERE
- MIN**: dóna el valor mínim dels resultats de calcular l'expressió per a les files que compleixen la condició del WHERE
- MAX**: dóna el valor màxim dels resultats de calcular l'expressió per a les files que compleixen la condició del WHERE
- AVG**: dóna el valor promig dels resultats de calcular l'expressió per a les files que compleixen la condició del WHERE

Consultes sobre una taula: Funcions d'agregació - Exemple

```
SELECT COUNT(*) AS quantEmpl ,
COUNT(DISTINCT nom_empl) AS
quantNoms,
SUM(sou*0.1) AS partSou      resultat → 

|  | quantEmpl | quantNoms | partSou |
|--|-----------|-----------|---------|
|  | 5         | 4         | 137000  |

FROM empleats
WHERE num_dpt IN (1,3);
```

empleats(<u>num_empl</u> , <u>nom_empl</u> , <u>sou</u> , <u>ciutat_empl</u> , <u>num_dpt</u> , <u>num_proj</u>)					
1	CARME	410000	MATARÓ	1	1
2	EUGENIA	350000	TOLEDO	2	2
3	JOSEP	250000	SITGES	3	1
4	RICARDO	410000	BARCELONA	1	1
11	NURIA	150000	VIC	3	2
12	NURIA	150000	MATARÓ	1	5

■ Files que compleixen la condició del WHERE

■ Resultat de la consulta

Consultes sobre una taula: Agrupació de files

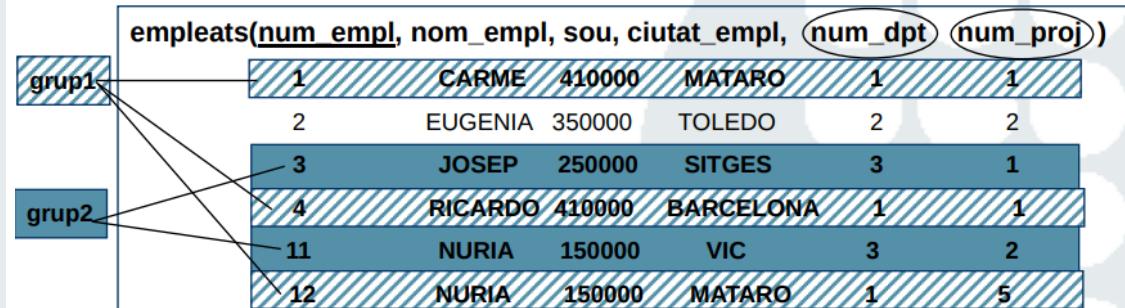
```
SELECT <columnes_a_seleccionar> [,<funcions_d'agregació>]
FROM <taula_a_consultar>
[ WHERE <condicions> ]
GROUP BY <columnes_segons_les_que_agrupar>;
```

- S'organitza en grups les files de la **taula_a_consultar** que compleixen les **condicions** especificades a la clàusula **WHERE**, segons el seu valor per les **columnes_segons_les_que_agrupar**.
- El resultat de la consulta és el valor de les **columnes_a_seleccionar** per cadascun dels grups de files obtinguts.
- Les **columnes_a_seleccionar** han de ser columnes que continguin el mateix valor per totes les files dins d'un grup.
- En el resultat es pot demanar també el valor de **funcions_d'agregació** que es calculen per cadascun dels grups de files obtinguts.

Consultes sobre una taula: Agrupació de files - Exemple

```
SELECT num_dpt,
COUNT(*) AS quantEmpl
FROM empleats
WHERE num_dpt IN (1,3) → resultat
GROUP BY num_dpt;
```

num_dpt	quantEmpl
1	3
3	2



■ Files que compleixen la condició del WHERE, agrupades segons indica la clàusula GROUP BY

■ Dades obtingudes com a resultat de la consulta. Hi ha un resultat per cada grup

Consultes sobre una taula: Condicions sobre grups

```
SELECT <columnes_a_seleccionar> [,<funcions_d'agregació>]
FROM <taula_a_consultar>
[ WHERE <condicions> ]
GROUP BY <columnes_segons_les_que_agrupar>
HAVING <condicions_per_grups>;
```

- En el cas de posar la clàusula **HAVING**, només apareix un resultat per cada grup que compleix les **condicions_per_grups**.
- Les **condicions_per_grups** seran comparacions entre constants, columnes amb un valor únic per cada grup i valors de funcions d'agregació (també amb un únic valor per cada grup). *Per exemple: si s'agrupa els empleats per departament, el sou de cada empleat d'un departament pot ser diferent, però el MAX del sou dels empleats d'un departament té valor únic.*
- Les **funcions d'agregació** té sentit aplicar-les a columnes que poden tenir valors diferents entre les files que componen els grups.

Consultes sobre una taula: Condicions sobre grups - Exemple 2

```
SELECT DISTINCT num_dpt
FROM empleats
GROUP BY num_dpt, ciutat_empl
HAVING COUNT(*) >= 2;
```

empleats(num_empl, nom_empl, sou, ciutat_empl, num_dpt, num_proj)					
grup1	1	CARME	410000	MATARO	1 1
grup2	2	EUGENIA	350000	TOLEDO	2 2
grup3	3	JOSEP	250000	SITGES	3 1
grup4	4	RICARDO	410000	BARCELONA	1 1
grup5	11	NURIA	150000	VIC	3 2
	12	NURIA	150000	MATARO	1 5
	13	ALBERT	150000	BARCELONA	1 5

■ En no haver-hi WHERE, són totes les files les que s'agrupen segons indica la clàusula GROUP BY
 ■ Dades obtingudes com a resultat de la consulta. Un resultat per cada grup que compleix la condició del HAVING, sense resultats repetits degut a la clàusula DISTINCT.

Consultes sobre una taula: Condicions sobre grups - Exemple 1

```
SELECT num_dpt, SUM(sou) AS sumaSous
FROM empleats
GROUP BY num_dpt
HAVING COUNT(*) >= 3;
```

resultat → num_dpt sumaSous

1	970000
---	--------

Consultes sobre més d'una taula: Format bàsic

```
SELECT <columnes_a_seleccionar> | *
FROM <taules_a_consultar>
[ WHERE <condicions> ] ;
```

- El resultat de la consulta és el valor de les **columnes_a_seleccionar** de les **taules_a_consultar** únicament per a la fila o files que compleixen les **condicions** especificades a la clausula **WHERE**.
- En el cas de no posar la clausula **WHERE**, el resultat és el valor de les **columnes_a_seleccionar** per a les files obtingudes del producte cartesià de les files a les **taules_a_consultar**.
- Si posem un * en lloc de les **columnes_a_seleccionar** indica que estem interessats en totes les columnes de les **taules_a_consultar**.

■ En no haver-hi WHERE, totes les files agrupades segons indica la clàusula GROUP BY
 ■ Dades obtingudes com a resultat de la consulta, un resultat per cada grup que compleix la condició del HAVING.

Consultes sobre més d'una taula: Format bàsic – Exemple 1

SELECT *
FROM empleats e, projectes p;

e.num_empl	e.nom_empl	e.num_proj	p.num_proj	p.nom_proj
1	CARME	1	1	IBDTEL
1	CARME	1	2	IBDVID
1	CARME	1	3	IBDTEF
3	JOSEP	1	1	IBDTEL
3	JOSEP	1	2	IBDVID
3	JOSEP	1	3	IBDTEF

empleats(num_empl, nom_empl, num_proj) projectes(num_proj, nom_proj)

1	CARME	1	IBDTEL
3	JOSEP	1	IBDVID

Dades obtingudes com a resultat de la consulta. Cal notar que, de totes les files, les que segurament podem estar interessats, són aquelles marcades en negreta (són aquelles en què el número de projecte en què treballa l'empleat coincideix amb el número de projecte del projecte).

Consultes sobre més d'una taula: Format bàsic – Exemple 2

SELECT e.num_empl, p.num_proj,
p.nom_proj
FROM empleats e, projectes p
WHERE e.num_proj = p.num_proj;

e.num_empl	p.num_proj	p.nom_proj
1	1	IBDTEL
3	1	IBDTEL

empleats(num_empl, nom_empl, num_proj) projectes(num_proj, nom_proj)

1	CARME	1	IBDTEL
3	JOSEP	1	IBDVID

Dades obtingudes com a resultat de la consulta. Cal notar que només apareixen aquelles combinacions en les que coincideix el número de projecte en què treballa l'empleat, amb el número de projecte del projecte.

Consultes sobre més d'una taula Sintaxis alternativa– Clàusula Inner Join - Exemple 3

La condició de combinació de les taules es pot escriure:

- O bé en la clàusula WHERE
- O bé usant la clàusula JOIN en el FROM
 - INNER JOIN requereix la condició de combinació de manera explícita
 - NATURAL INNER JOIN fa la combinació per les columnes amb el mateix nom en les taules implicades.

SELECT e.num_empl, p.num_proj, p.nom_proj
FROM empleats e, projectes p
WHERE e.num_proj = p.num_proj;

SELECT e.num_empl, p.num_proj, p.nom_proj
FROM empleats e **INNER JOIN** projectes p **ON** e.num_proj = p.num_proj;

SELECT e.num_empl, p.num_proj, p.nom_proj
FROM empleats e **NATURAL INNER JOIN** projectes p;

Les tres sentències anteriors són totalment equivalents.

Consultes sobre varis taules: Exemple amb agrupació de files

SELECT p.num_proj, p.nom_proj
FROM projectes p, empleats e
WHERE p.num_proj = e.num_proj
GROUP BY p.num_proj
HAVING p.pressupost < SUM(e.sou);

resultat	p.num_proj	p.nom_proj
	1	IBDTEL

grup1	e.num_empl	e.nom_empl	e.sou	e.num_proj	p.num_proj	p.nom_proj	p.pressupost
	1	CARME	410000	1	1	IBDTEL	1000000
	2	EUGENIA	350000	2	2	IBDVID	500000
grup2	3	JOSEP	250000	1	1	IBDTEL	1000000
	4	RICARDO	410000	1	1	IBDTEL	1000000
	11	NURIA	150000	2	2	IBDVID	500000

Files resultants de la combinació dels empleats amb els projectes, agrupades segons la clàusula GROUP BY
 Dades obtingudes com a resultat de la consulta. Un resultat per cada grup que compleix la condició del HAVING. Només hi ha un projecte tal que el seu pressupost és inferior a la suma del sou dels empleats que hi treballen.

Cal notar que l'atribut p.nom_proj pot estar entre els atributs del select perquè té un valor únic per cada grup. Igualment, l'atribut p.pressupost pot estar en la condició del having perquè té un valor únic per cada grup.

Consultes: Unió

```
SELECT <columnes_a_seleccionar> | *
  FROM <taules_a_consultar>
  [ WHERE <condicions> ]
UNION
SELECT <columnes_a_seleccionar> | *
  FROM <taules_a_consultar>
  [ WHERE <condicions> ]
[ ORDER BY <columna> [DESC|ASC],... ];
```

- El resultat és la unió del resultat obtingut de les dues sentències **SELECT**.
- Les **columnes_a_seleccionar** en les dues sentències **SELECT** han de ser semànticament compatibles
- Sempre s'obté resultats sense repeticions (en molts SGBDR ja surten ordenats).
- Les columnes que apareixen a clausula **ORDER_BY** han de ser un subconjunt de les **columnes_a_seleccionar** del primer **SELECT**.

Consultes: Unió - Exemple

```
SELECT ciutat_empl
  FROM empleats
UNION
SELECT ciutat_dpt
  FROM departaments
 ORDER BY ciutat_empl DESC;
```

ciutat_empl
SITGES
MATARÓ
MADRID
BARCELONA

resultat

empleats(<u>num_empl</u> , nom_empl, ciutat_empl)	departaments(<u>num_dpt</u> , ciutat_dpt)
1 CARME MATARÓ	1 BARCELONA
3 JOSEP SITGES	2 MADRID
	3 BARCELONA

■ Dades obtingudes com a resultat de la consulta.

Consultes: Diferència

```
SELECT <columnes_a_seleccionar> | *
  FROM <taules_a_consultar>
 WHERE <columna_taula> NOT IN ( SELECT <columna_a_seleccionar>
  FROM <taules_a_consultar>
  [ WHERE <condicions> ]);
```



```
SELECT <columnes_a_seleccionar> | *
  FROM <taules_a_consultar>
 WHERE NOT EXISTS ( SELECT *
  FROM <taules_a_consultar>
  WHERE <condicions> );
```

- Hi ha dos formes alternatives de fer una diferència amb NOT IN o bé amb NOT EXISTS.
- Un NOT IN és cert si el valor de la columna *columna_taula* no està en el resultat de la subconsulta.
- Un NOT EXISTS és cert si la subconsulta no dóna cap resultat.
- Hi ha altres maneres de fer una diferència (operador Except en SQL estàndard), encara que hi ha diferents noms de l'operador en diferents sistemes.

Consultes: Diferència – Exemples

```
SELECT p.num_proj, p.nom_proj
  FROM projectes p
 WHERE p.num_proj NOT IN (SELECT e.num_proj
  FROM empleats e);
```

resultat

p.num_proj	p.nom_proj
2	IBDVID
3	IBDTEF
4	IBDCOM

```
SELECT p.num_proj, p.nom_proj
  FROM projectes p
 WHERE NOT EXISTS (SELECT * FROM empleats e
  WHERE p.num_proj = e.num_proj);
```

empleats(<u>num_empl</u> , nom_empl, <u>num_proj</u>)	projectes(<u>num_proj</u> , nom_proj)
1 CARME 1	1 IBDTEL
3 JOSEP 1	2 IBDVID

- Dades obtingudes com a resultat de la consulta, en qualsevol de les dues alternatives.
En qualsevol cas, la consulta dóna aquells projectes que no tenen cap empleat assignat.
- NOT IN: Un projecte és al resultat de la consulta si el seu número no està entre els números de projecte dels empleats.
- NOT EXISTS: Un projecte és al resultat de la consulta si no existeix cap empleat que tingui el seu número de projecte.

Subconsultes en sentències Delete, Update i Select

- Poden aparèixer en aquelles sentències on hi ha la clausula **WHERE**:

- Esborrat de files d'una taula

```
DELETE FROM <taula>
WHERE .... (SELECT ..... );
```

- Modificació de files d'una taula

```
UPDATE <taula>
SET ....
WHERE .... (SELECT ..... );
```

- Consultes una taula o més taules

```
SELECT <columnes_a_seleccionar>
FROM <taules_a_consultar>
WHERE .... (SELECT ..... );
```

Subconsultes en sentències Delete - Exemple

```
DELETE FROM projectes
WHERE NOT EXISTS (SELECT *
                  FROM empleats e
                  WHERE e.num_proj = projectes.num_proj);
```

empleats(num_empl, nom_empl, num_proj)			projectes(num_proj, nom_proj)	
1	CARME	1	1	IBDTEL
3	JOSEP	1	2	IBDVID
			3	IBDTEF
			4	IBDCOM

■ Files esborrades com a resultat de la sentència. La sentència elimina aquells projectes que no tenen cap empleat assignat. Cal notar que la subconsulta obté resultats per aquells projectes que tenen com a mínim un empleat.

Subconsultes en sentències Update - Exemple

```
UPDATE projectes
SET pressupost = pressupost + (pressupost * 0,1)
WHERE 2 <= (SELECT COUNT(*)
             FROM empleats e
             WHERE projectes.num_proj = e.num_proj);
```

empleats(num_empl, nom_empl, num_proj)		projectes(num_proj, pressupost)	
1	CARME	1	1 1100000
3	JOSEP	1	2 500000
			3 4500000
			4 2000000

■ Files modificades com a resultat de la sentència. La sentència puja el pressupost d'aquells projectes que tenen dos o més empleats assignats. Cal notar que la subconsulta el que fa és calcular el nombre d'empleats del projecte que s'està considerant si cal modificar o no.

Subconsultes en sentències Select – Exemple 1

```
SELECT e.num_empl, e.nom_empl
      FROM empleats e
      WHERE e.sou > ( SELECT AVG(e1.sou)
                      FROM empleats e1);
```

empleats(num_empl, nom_empl, sou, ciutat_empl, num_dpt, num_proj)				
1	CARME	410000	MATARO	1 1
2	EUGENIA	350000	TOLEDO	2 2
3	JOSEP	250000	SITGES	3 1
4	RICARDO	410000	BARCELONA	1 1
11	NURIA	150000	VIC	3 2

■ Dades obtingudes com a resultat de la sentència. La sentència dóna els empleats que tenen un sou superior a la mitjana del sou de tots els empleats. Cal notar que la subconsulta retorna la mitjana del sou de tots els empleats de la taula empleats.

Subconsultes en sentències Select – Exemple 2

```
SELECT p.num_proj, p.nom_proj
FROM projects p
WHERE p.pressupost < (SELECT SUM(e.sou)
    FROM empleats e
    WHERE e.num_proj=p.num_proj);
```

resultat → p.num_proj,p.nom_proj

1	IBDTEL
---	--------

projectes(num_proj,nom_proj,p.pressupost)			empleats(num_empl, nom_empl, sou, num_proj)		
1	IBDTEL	1000000	1	CARME	410000 1
2	IBDVID	500000	2	EUGÉNIA	350000 2
			3	JOSEP	250000 1
			4	RICARDO	410000 1
			11	NURIA	150000 2

- Dades obtingudes com a resultat de la sentència. La sentència dóna els projectes que tenen un pressupost inferior a la suma del sou dels empleats que hi estan assignats. Cal notar que la subconsulta retorna la suma del sou dels empleats assignats al projecte que s'està considerant.
- Filas seleccionades en la subconsulta quan es considera el projecte número 1. La suma dels sous és 1070000.
- Filas seleccionades en la subconsulta quan es considera el projecte número 2. La suma dels sous és 500000

Subconsultes en clàusules Having – Exemple

```
SELECT d.num_dpt, d.nom_dpt, 100*SUM(e.sou)/d.pressupost AS percSous
FROM departaments d, empleats e
WHERE d.num_dpt = e.num_dpt
GROUP BY d.num_dpt
HAVING SUM(e.sou) > ( SELECT SUM(e1.sou)
    FROM empleats e1
    WHERE e1.num_dpt = 3);
```

resultat → d.num_dpt d.nom_dpt percSous

1	DIRECCIO	82
---	----------	----

empleats(num_empl, nom_empl, sou, num_dpt)				departaments(num_dpt, nom_dpt, presupost)		
1	CARME	410000	1	1	DIRECCIO	1000000
2	EUGENIA	350000	2	2	DIRECCIO	2000000
3	JOSEP	350000	3	3	MARQUETING	2500000
4	RICARDO	410000	1			
11	NURIA	150000	3			

- Dades obtingudes com a resultat de la sentència. La sentència dóna els departaments tals que la suma del sou dels seus empleats és superior a la suma del sou dels empleats del departament número 3. Cal notar que la subconsulta retorna la suma del sou dels empleats del departament número 3.
- Filas seleccionades en la subconsulta sigui qui sigui el departament que es considera. La suma dels sous és 500000.

Subconsultes en sentències Insert – Exemple

```
INSERT INTO clients
(SELECT num_empl,nom_empl,200000
FROM empleats
WHERE num_dpt IN (2,3));
```

Subconsultes en sentències Insert i clàusules Having

- En aquelles sentències on hi ha la clausula HAVING:

```
SELECT <columnes_a_seleccionar>
FROM <taules_a_consultar>
WHERE <condicions>
GROUP BY <columnes_agrupacio>
HAVING ..... (SELECT ..... );
```

- Finalment, en sentències d'inserció de files a una taula (en aquest cas el resultat de la subconsulta és un conjunt de files que ha de ser compatible amb la definició de la taula en el CREATE TABLE)

```
INSERT INTO <taula>
(SELECT ..... );
```

clients(num_client, nom, credit)			empleats(num_empl, nom_empl, num_dpt)		
2	EUGENIA	200000	1	CARME	1
3	JOSEP	200000	2	EUGENIA	2
			3	JOSEP	3
			4	RICARDO	1

- Filas seleccionades a la subconsulta.
- Filas inserides a la taula clients.

PERSONAL-ADM(num-per, nom, cognom, modul, num)				
100	Joan	Soler	B6	25
150	Clara	Bellsolà	B6	25

PERSONAL-LAB(num-per, nom, cognom, modul, num)				
150	Clara	Bellsolà	B6	25
110	Núria	Nogué	B2	25
200	Jordi	Moles	B6	27
230	Pere	Roig	NULL	NULL

$$R = PERSONAL-ADM \cup PERSONAL-LAB$$

R(num-per, nom, cognom, modul, num)				
100	Joan	Soler	B6	25
150	Clara	Bellsolà	B6	25
110	Núria	Nogué	B2	25
200	Jordi	Moles	B6	27
230	Pere	Roig	NULL	NULL

No hi ha tuples repetides!!!

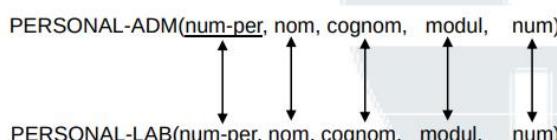
- Els atributs de l'**esquema** de la relació resultant de $T \cup S$ coincideixen amb els atributs de l'esquema de la relació T o de la relació S .
- L'**extensió** de la relació resultant de $T \cup S$ és el conjunt de tuples que pertanyen a l'extensió de T o que pertanyen a l'extensió de S o que pertanyen a l'extensió d'ambdues relacions
- Per fer la unió de dues relacions T i S cal que T i S siguin **relacions compatibles**.
- En cas de que els atributs de T i S no coincideixin cal reanomenar els atributs d'una de les dues relacions per tal de que siguin compatibles.

Relacions compatibles

- Algunes operacions de l'àlgebra relacional, com ara la unió, només té sentit que s'apliquin a relacions que siguin compatibles (que tinguin tuples "similars")
- Exemple: pot fer-se la unió
 $PERSONAL-ADM \cup PERSONAL-LAB$
perquè les tuples de les dues relacions s'assemblen en canvi no té sentit fer la unió
 $PERSONAL-ADM \cup OFICINES$
- Diem que **dues relacions T i S són compatibles** si:
 - tenen esquemes amb un conjunt d'atributs idèntic, i els dominis de cada parella d'atributs són els mateixos a T i a S .

Exemple:

PERSONAL-ADM i PERSONAL-LAB són clarament compatibles:



Reanomenament

PERSONAL-DOC(num-per, nom, cognom, modul-de, num-de)				
400	Jaume	Cases	Omega	119
500	Pau	Pou	B6	123

$$R = PERSONAL-DOC \{modul-de \rightarrow modul, num-de \rightarrow num\}$$

R(num-per, nom, cognom, modul, num)				
400	Jaume	Cases	Omega	119
500	Pau	Pou	B6	123

- L'**esquema** de la relació resultant és el mateix, exceptuant el canvi de nom dels atributs que han estat reanomenats.
- L'**extensió** de la relació resultant no canvia.

Intersecció

PERSONAL-ADM(num-per, nom, cognom, modul, num)				
100	Joan	Soler	B6	25
150	Clara	Bellsolà	B6	25

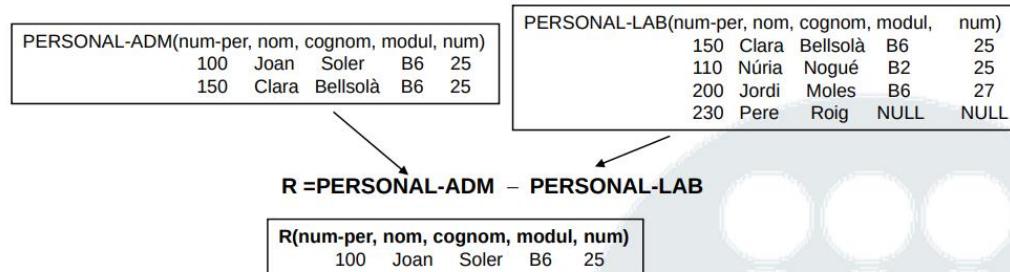
PERSONAL-LAB(num-per, nom, cognom, modul, num)				
150	Clara	Bellsolà	B6	25
110	Núria	Nogué	B2	25
200	Jordi	Moles	B6	27
230	Pere	Roig	NULL	NULL

$$R = PERSONAL-ADM \cap PERSONAL-LAB$$

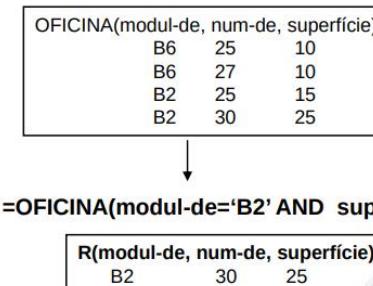
R(num-per, nom, cognom, modul, num)				
150	Clara	Bellsolà	B6	25

- Els atributs de l'**esquema** de la relació resultant de $T \cap S$ coincideixen amb els atributs de l'esquema de la relació T o la relació S .
- L'**extensió** de la relació resultant de $T \cap S$ és el conjunt de tuples que pertanyen a l'extensió d'ambdues relacions
- Per fer la intersecció de dues relacions T i S cal que T i S siguin **relacions compatibles**.
- En cas de que els atributs de T i S no coincideixin cal reanomenar els atributs d'una de les dues relacions per tal de que siguin compatibles.

Diferència

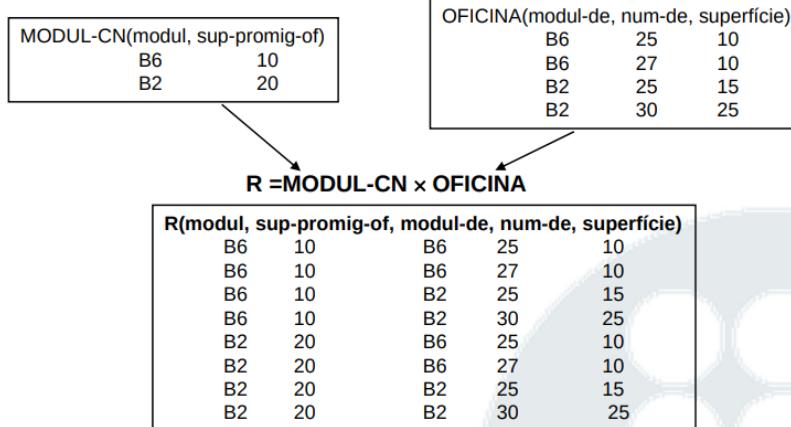


Selecció



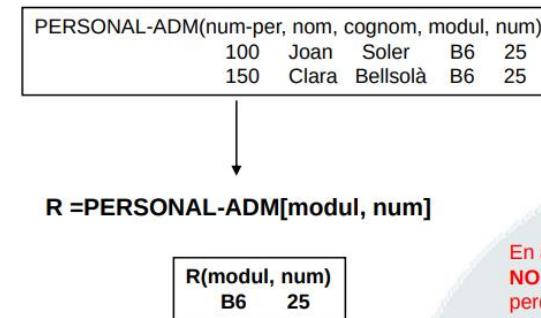
- Els atributs de l'**esquema** de la relació resultant de T - S coincideixen amb els atributs de l'esquema de la relació T o la relació S.
 - L'**extensió** de la relació resultant de T - S és el conjunt de tuples que pertanyen a l'extensió de T però no a la de S.
 - Per fer la diferència de dues relacions T i S cal que T i S siguin **relacions compatibles**.
 - En cas de que els atributs de T i S no coincideixin cal reanomenar els atributs d'una de les dues relacions per tal de que siguin compatibles.
- **T(C)** indica la selecció de T amb la condició C, essent C una condició de selecció
 - La condició C està formada per una o més comparacions de la forma:
- $$A_i \theta V_j \text{ o bé } A_i \theta A_j$$
- on A_i i A_j són atributs de la relació T, V_j és un valor constant, i θ és un operador de comparació ($=, <, >, \leq, \geq$). Les comparacions han d'estar relacionades entre elles per un dels operadors lògics AND (\wedge), OR (\vee).
- Els atributs de l'**esquema** de la relació resultant de $T(C)$, coincideixen amb els atributs l'esquema de la relació T
 - L'**extensió** de la relació resultant de $T(C)$ és el conjunt de tuples que pertanyen a l'extensió de T i que satisfan la condició de selecció C.

Producte Cartesià



- Els atributs de l'**esquema** de la relació resultant de $T \times S$ són tots els atributs de T i tots els atributs de S.
- Si T i S tenen algun nom d'atribut idèntic, s'haurà de fer prèviament una operació de reanomenament d'una de les dues relacions per eliminar l'ambigüïtat.
- L'**extensió** de la relació resultant de $T \times S$ és el conjunt de totes les tuples de la forma $\langle v_1, v_2, \dots, v_n, w_1, w_2, \dots, w_m \rangle$ on es compleix que $\langle v_1, v_2, \dots, v_n \rangle$ pertany a l'extensió de T i que $\langle w_1, w_2, \dots, w_m \rangle$ pertany a l'extensió de S.
- Per fer el producte cartesià de dues relacions T i S **no cal** que T i S siguin **relacions compatibles**

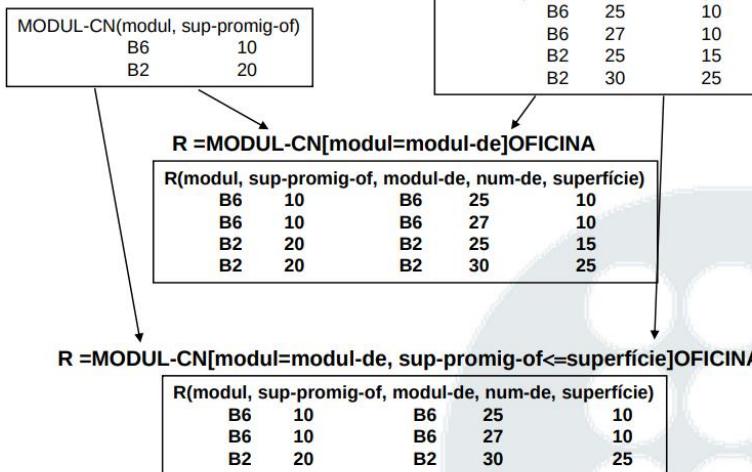
Projecció



En àlgebra relacional,
NO HI HA TUPLES REPETIDES
perquè el resultat de les operacions són conjunts.

- **T[A_i, A_j, ..., A_k]** indica la projecció de T sobre {A_i, A_j, ..., A_k}, essent {A_i, A_j, ..., A_k} un subconjunt dels atributs de l'esquema de la relació T.
- Els atributs de l'**esquema** de la relació resultant de $T[A_i, A_j, \dots, A_k]$, són els atributs {A_i, A_j, ..., A_k}
- L'**extensió** de la relació resultant de $T[A_i, A_j, \dots, A_k]$ és el conjunt de totes les tuples de la forma $\langle t.A_i, t.A_j, \dots, t.A_k \rangle$ on es compleix que t és una tupla de l'extensió de T i on $t.A_p$ denota el valor per l'atribut A_p de la tupla t.

Combinació (join)



- **T[B]S** indica la combinació de T i S amb la condició B

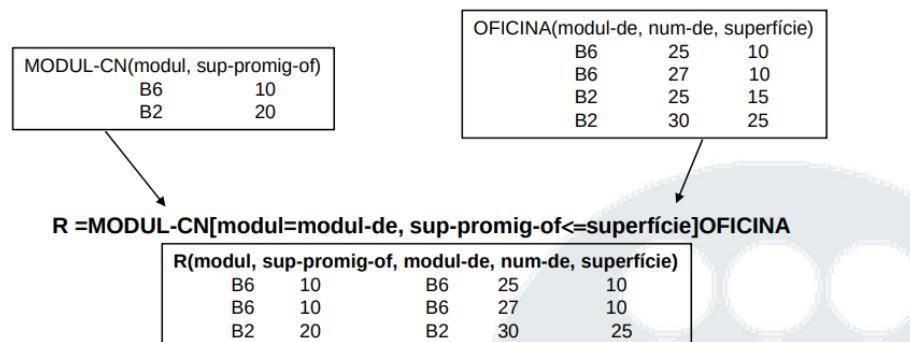
- La condició B d'una combinació **T [B] S** està formada per una o més comparacions de la forma:

$$A_i \theta A_j$$

on A_i és un atribut de la relació T, A_j és un atribut de la relació S, θ és un operador de comparació ($=, \neq, <, \leq, >, \geq$) i es compleix que A_i i A_j tenen el mateix domini.

- Les comparacions d'una condició de combinació se separen per comes.

Combinació (join)



- Els atributs de l'**esquema** de la relació resultant de **T[B]S** són tots els atributs de T i tots els atributs de S.

- Si T i S tenen algun nom d'atribut idèntic, s'haurà de fer prèviament una operació de reanomenament d'una de les dues relacions per eliminar l'ambigüïtat.

- L'**extensió** de la relació resultant de **T [B] S** és el conjunt de tuples que pertanyen a l'extensió del producte cartesià **T × S** i que satisfan totes les comparacions que formen la condició de combinació B.

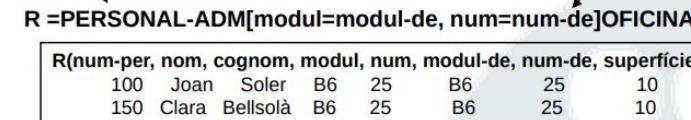
Combinació (join): Tipus de “joins”

- **“θ-join”**: La “join” s'anomena també “θ-join”

- **“Equi-join”**: Cas particular de “join” en què totes les comparacions de la condició tenen l'operador ‘=’.

PERSONAL-ADM(num-per, nom, cognom, modul, num)		
100	Joan	Soler
150	Clara	Bellsolà

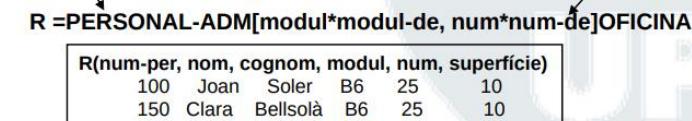
OFICINA(modul-de, num-de, superficie)		
B6	25	10
B6	27	10
B2	25	15
B2	30	25



- **“Natural join”**: Variant de la “equi-join” en la qual s'eliminen els atributs superflus. Es denota mitjançant un *. La diferència amb la “equi-join” és en l'esquema de la relació resultant, ja que no hi apareix el segon atribut de la comparació.

PERSONAL-ADM(num-per, nom, cognom, modul, num)		
100	Joan	Soler
150	Clara	Bellsolà

OFICINA(modul-de, num-de, superficie)		
B6	25	10
B6	27	10
B2	25	15
B2	30	25



Combinació (join): “Natural join” implícita

PERSONAL-ADM(num-per, nom, cognom, modul, num)		
100	Joan	Soler
150	Clara	Bellsolà

MODUL-CN(modul, sup-promig-de)		
B6	10	
B2	20	

R = PERSONAL-ADM * MODUL-CN

R(num-per, nom, cognom, modul, num, sup-promig-de)					
100	Joan	Soler	B6	25	10
150	Clara	Bellsolà	B6	25	10

- La “natural join” ímplicita: Variant de la “natural-join” en la qual no s'especifica la condició de combinació i aleshores s'assumeix per defecte que la condició de combinació correspon a la d'una “natural join” on s'igualen tots els parells d'atributs que tenen el mateix nom a les dues relacions.

- **T * S** denota la “natural join” ímplicita de T i S.

Seqüències d'operacions de l'àlgebra relacional

Exemple: Obtenir les oficines (modul i número) dels moduls que tenen una superfície promig més gran que 15.

MODUL-CN(modul, sup-promig-of)	
B6	10
B2	20

OFICINA(modul-de, num-de, superficie)		
B6	25	10
B6	27	10
B2	25	15
B2	30	25

A = MODUL-CN(sup-promig-of > 15)

B = A{modul -> modul-de}

C = OFICINA * B

R = C[modul-de, num-de]

R(modul-de, num-de)	
B2	25
B2	30

• Les **consultes** a una BD relacional es poden expressar en termes de **seqüències d'operacions** de l'àlgebra relacional.

• Les seqüències d'operacions ens permeten definir una relació que conté precisament allò que es desitja consultar.

Esquemes

- Des d'aquest punt de vista, el component esquema és una eina específica de l'SGBD que serveix com a unitat administrativa per agrupar un conjunt d'altres components.
- Aquesta és la utilitat principal del component esquema: ens permet centralitzar tasques administratives que d'altra manera hauríem de repetir individualment. Podem engegar, aturar, o atorgar privilegis d'un conjunt de components lògics a un usuari.
- Hi ha diversos criteris per a decidir quins components formaran part d'un esquema. Per exemple:
 - Per aplicacions
 - Per usuaris finals

Esquemes

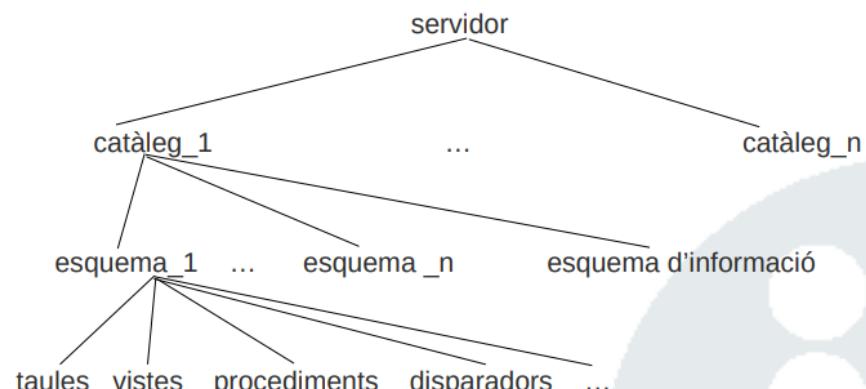
Els SGBD agrupen els components de dades (taules, vistes, ...) i els components de control (procediments, disparadors, ...), que veurem més endavant, en un altre component lògic anomenat **esquema**

Un esquema, per a un SGBD, és un conjunt **definit** i **relacionat** de components de dades i de components de control

perquè si una taula pertany a un esquema totes les vistes definides sobre ella, totes les restriccions, índexs, disparadors normalment també estarán definits a dins

perquè l'SGBD disposa d'un mecanisme per definir-lo exactament:
- create schema nom_esquema

Servidor, Catàleg i Esquema



- Un catàleg (catalog) és un grup d'esquemes, un dels quals es anomenat esquema d'informació (information schema)
- L'esquema d'informació és un conjunt de vistes que conté la descripció de totes les dades SQL que pertanyen al catàleg corresponent
- Un servidor (cluster) pot contenir zero o més catàlegs

Esquemes

- No hi ha una instrucció del SQL estàndar per crear, destruir o modificar Catàlegs. Dependrà de cada implementador
- La sentència **CREATE SCHEMA** dóna nom a un nou esquema, identifica a l'usuari propietari de l'esquema i pot donar la llista d'elements de l'esquema

CREATE SCHEMA [[nom_cat.]nom_esq] [**AUTHORIZATION** ident_usuari]
[llista d'elements de l'esquema];

- La sentència **DROP SCHEMA** amb l'opció **RESTRICT** esborra un esquema només si aquest està completament buit. En canvi amb l'opció **CASCADE** l'esborra encara que contingui elements

DROP SCHEMA nom_esquema [**RESTRICT** | **CASCADE**];

Connexions, Sessions i Transaccions

- Una **connexió** es pot definir com l'associació que es crea entre un client SQL i un servidor SQL quan el client manifesta que té la intenció de treballar amb la BD sol·licitant una connexió.

Sentència per establir una connexió:

CONNECT TO nom_servidor [**AS** nom_connexio] [**USER** ident_usuari];
SET SCHEMA nom_esq;

I que es destrueix quan acaba:

DISCONNECT nom_connexio | **DEFAULT** | **CURRENT** | **ALL**;

- Les sentències SQL que s'executen mentre hi ha una connexió activa a un servidor formen una **sessió**. Per tant una sessió és el context en el que un usuari o aplicació executa una seqüència de sentències SQL mitjançant una connexió:

Sentència per establir les característiques de les transaccions que s'executen en una sessió:

SET SESSION CHARACTERISTICS AS mode_transac [, mode_transac ...];

Connexions, Sessions i Transaccions

- Una transacció és un conjunt de sentències SQL de consulta i actualització de la BD que s'executen com una unitat. Les seves característiques es poden definir prèviament amb:

SET TRANSACTION mode_transac [, mode_transac ...];
on mode_transac pot ser: mode_d'accés | nivell d'aïllament
on mode_d'accés pot ser: **READ ONLY** | **READ WRITE**
on nivell d'aïllament pot ser: **READ UNCOMMITTED** |
READ COMMITTED | **REPEATABLE READ** | **SERIALIZABLE** (*)

És pot fer explícit l'inici d'una transacció amb:

START TRANSACTION mode_transac [, mode_transac ...];

- Una **transacció** finalitza confirmant o cancel·lant els canvis que s'hi han fet.

Sentència que confirma tots els canvis de la transacció:

COMMIT [WORK] [AND [NO] CHAIN];

Sentència que desfà tots els canvis de la transacció, deixant la BD com abans de començar l'execució de la transacció:

ROLLBACK [WORK] [AND [NO] CHAIN];

Dominis

- Un esquema pot contenir zero o més dominis. Un domini és un conjunt de valors vàlids definits per l'usuari

CREATE DOMAIN nom_domini **AS** tipus_dades
[**DEFAULT** literal | temps | **USER** | **NULL**]
[llista_restriccions_domini];

- Restriccions de domini

llista_restriccions_domini poden ser:

CONSTRAINT nom_restricció **CHECK** condició
[, **CONSTRAINT** nom_restr **CHECK** condició ...]

CREATE DOMAIN ciutat **AS** char(15)
DEFAULT 'BCN'
CONSTRAINT vàlides **CHECK VALUE IN** ('BCN', 'MAD');

CREATE TABLE empleats (nempl integer, ciutat_naix ciutat,
ciutat_resid ciutat, ciutat_treb ciutat);

Taules

- El component lògic principal d'una base de dades és la **taula**

```
CREATE TABLE nom_taula  
(definició_columna [, definició_columna ... ][, restriccions_taula]);
```

- Definició de columna

definició_columna pot ser:

nom_columna tipus_dades [def_defecte] [restricció_columna]

on tipus_dades pot ser: DATE, TIME, TIMESTAMP, INTERVAL, BINARY LARGE OBJECT(BLOB), DECIMAL (NUMERIC), FLOAT, DOUBLE PRECISION, REAL, INTEGER (INT), SMALLINT, CHARACTER (CHAR), CHARACTER LARGE OBJECT(CLOB), VARCHAR, BOOLEAN

on def_defecte (definicions per defecte) pot ser:

DEFAULT literal | funció | NULL

on funció pot ser:

funcions de valors de temps: CURRENT_DATE, CURRENT_TIME,
CURRENT_TIMESTAMP, ...

funcions de valors d'usuaris: CURRENT_USER, SESSION_USER,
SYSTEM_USER ...

Taules: Restriccions de columna

restricció_columna pot ser:

- **NOT NULL**
 - La columna no pot tenir valors nuls
- **UNIQUE**
 - La columna no pot tenir valor repetits.
- **PRIMARY KEY**
 - La columna és clau primària de la taula.
- **REFERENCES taula [(nom_columna)]**
 - [ON DELETE
NO ACTION | RESTRICT | CASCADE | SET NULL | SET DEFAULT]
 - [ON UPDATE
NO ACTION | RESTRICT | CASCADE | SET NULL | SET DEFAULT]
 - La columna és clau forana que referència la taula especificada (amb clau primària nom_columna).
- **[CONSTRAINT nom_restricció] CHECK (condicions)**
 - La columna ha de complir les condicions especificades.
 - Són típicament de rang del domini, encara que permeten posar qualsevol expressió.
 - És opcional el donar nom a aquestes restriccions.

Taules: Restriccions de taula

restriccions_taula poden ser:

- **UNIQUE (columna,)**
 - El conjunt de columnes especificades no pot tenir valors repetits.
- **PRIMARY KEY(columna,...)**
 - El conjunt de les columnes especificades és clau primària de la taula.
- **FOREIGN KEY(columna,...) REFERENCES taula (nom_columna,...)**
 - [ON DELETE
NO ACTION | RESTRICT | CASCADE | SET NULL | SET DEFAULT]
 - [ON UPDATE
NO ACTION | RESTRICT | CASCADE | SET NULL | SET DEFAULT]
 - El conjunt de les columnes especificades és una clau forana que referencia la taula especificada (amb clau primària nom_columna,...).
- **[CONSTRAINT nom_restricció] CHECK (condicions)**
 - La taula ha de complir les condicions especificades.
 - La condició pot referir-se a una o més columnes de la taula.
 - És opcional el donar nom a aquestes restriccions.

Segons el estàndard SQL, en el cas que una restricció faci referència a més d'una columna, s'ha de definir com a restricció de taula.

Taules: Restriccions

- En el cas que una restricció faci referència a un únic atribut, podem escollir si la definim com a restricció de taula o de columna.
- En el cas que una restricció faci referència a més d'una columna, s'ha de definir com a restricció de taula.
- Exemples de definició de restriccions:

```
CREATE TABLE persona (  
Dni char(8) PRIMARY KEY,  
Nom varchar(30) NOT NULL,  
Data_naixement date NOT NULL  
CONSTRAINT data_naix CHECK (data_naixement>='01/01/1900'),  
Data_defuncio date,  
Ciutat_naixement varchar(30) NOT NULL,  
Ciutat_residencia varchar(30) NOT NULL,  
Estudis char(1) DEFAULT '1'  
CONSTRAINT estudis CHECK (estudis BETWEEN '1' and '5'),  
Telefon decimal(9) UNIQUE,  
CONSTRAINT dates CHECK ((data_naixement<data_defuncio) OR  
(data_defuncio is NULL)));
```

Taules: Problemes en la definició de restriccions

- Una restricció no es viola mai si la taula està buida

- **1er Problema** (Ramakrishnan & Gehrke)

```
CREATE TABLE emp ( nemp integer,  
                   CHECK ((SELECT COUNT(*) FROM emp) > 0)
```

- Una restricció només es comprova quan s'actualitza la taula on està definida

- **2on Problema** (Garcia-Molina, Ullman & Widom)

```
CREATE TABLE emp  
( nemp integer,  
  dept char(10) CHECK ( dept IN SELECT dept FROM dept))
```

INSERT, UPDATE empleat **OK**
DELETE, UPDATE dept **NO**

Assercions

- Restriccions d'integritat que afecten a més d'una taula
- A diferència de les restriccions de columna o de taula es comproven sempre
- En la majoria dels sistemes actuals no es poden definir. Cal usar altres mecanismes: disparadors
- CREATE ASSERTION** nom **CHECK** (condició);
- Exemple:**

empleat(nemp, ciutat_e, ndept) dept(ndept, ciutat_d)

Cal assegurar que tots els empleats treballin a un departament que estigui situat a la ciutat on resideixen!

CREATE ASSERTION ciutat_emp_dept **CHECK**
(NOT EXISTS (SELECT *

```
FROM empleat e, dept d  
WHERE e.ndept = d.ndept and  
      e.ciutat_e <> d.ciutat.d));
```

Tema 4-1. Definir en SQL estàndard les següents assercions

1. Tot empleat ha de tenir un sou més gran o igual que el sou mínim de la seva categoria

categories (nom_categoria, soumínim)

empleats (num_emp, dni, nom, sou, nom_categoria)

{nom_categoria} referencia categories

2. El nombre d'empleats d'administració ha de ser menor o igual al nombre d'empleats de producció

empleats_adm (num_emp, dni, nom, sou, nom_categoria)

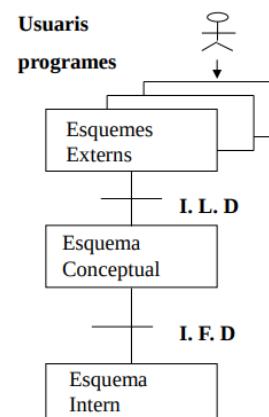
empleats_prod (num_emp, dni, nom, sou, nom_categoria)

```
Create assertion ex411 check (NOT EXIST(SELECT* FROM  
empleats e, categories c where e.nom_categoria =  
c.nom_categoria and e.sou < c.sou_minim));
```

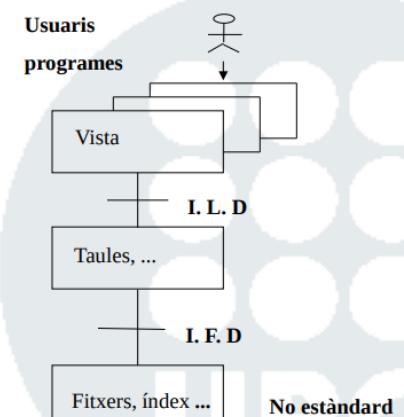
Vistes

Visió general

Arquitectura ANSI/SPARC



Arquitectura Relacional



Vistes

Una vista és una relació derivada: el seu esquema i contingut es deriven d'altres relacions (bàsiques o derivades) a partir d'una consulta relacional

Es pot consultar i actualitzar amb SQL

La seva extensió no existeix físicament

Potència SQL per definir vistes, excepte ORDER BY

CREATE VIEW nom_vista [(nom_columna, ...)] **AS** sentència_select
[**WITH CHECK OPTION**];

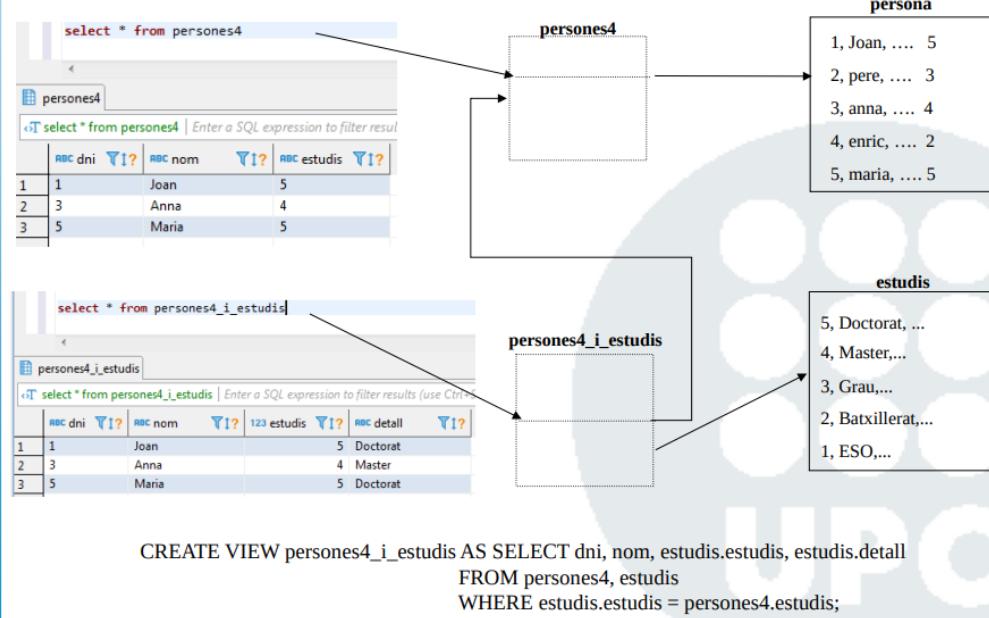
Exemple:

persona (dni, nom, data_naixement, data_defuncio,
ciutat_naixement, ciutat_residencia, estudis, telefon)

```
CREATE VIEW persones4 AS SELECT dni, nom, estudis  
FROM persona  
WHERE estudis >= 4
```

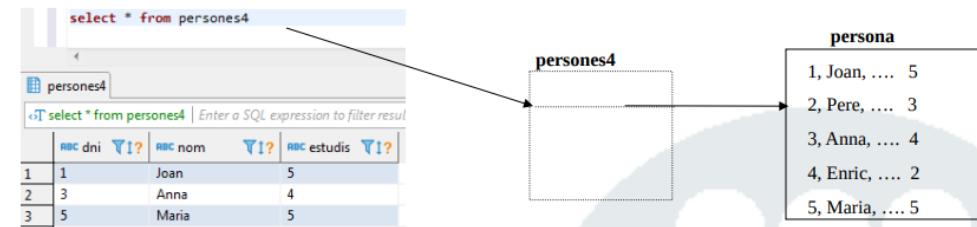
Vistes: Consultes

Les vistes són consultables amb SQL. De fet, de cara a l'usuari final és transparent el fet de que la relació que consulta sigui una vista



Vistes: Actualitzacions

- Les vistes són actualitzables amb SQL. De fet, de cara al usuari final és transparent el fet que la relació que actualitzi sigui una vista



1. UPDATE persones4 SET nom='Anna Maria' WHERE dni=5;
2. SELECT * FROM persones4;

	RBC dni	RBC nom	RBC estudis
1	1	Joan	5
2	3	Anna	4
3	5	Anna Maria	5

3. UPDATE persones4 SET estudis=2 WHERE dni = 5;
4. SELECT * FROM persones4;

	RBC dni	RBC nom	RBC estudis
1	1	Joan	5
2	3	Anna	4

Vistes: With Check Option

```
CREATE VIEW persones4 AS SELECT dni, nom, estudis  
FROM persona  
WHERE estudis >= 4  
WITH CHECK OPTION;
```

1. UPDATE persones4 SET nom='Anna Maria' WHERE dni=5;
2. SELECT * FROM persones4;

	RBC dni	RBC nom	RBC estudis
1	1	Joan	5
2	3	Anna	4
3	5	Anna Maria	5

3. UPDATE persones4 SET estudis=2 WHERE dni = 5;

"ERROR: new row violates check option for view "persones4"

Vistes: No totes les vistes són actualitzables

- De totes maneres, no totes les vistes són actualitzables. L'estàndard defineix amb precisió quines ho són i quines no. De manera simplificada, s'admeten actualitzacions d'aquelles vistes definides com:
 - SELECT sobre una única relació R (o vista actualitzable), sense agregats ni DISTINCT
 - Els atributs del SELECT han d'incloure tots els atributs amb restriccions not null que no tinguin valor per defecte.
 - Sense GROUP BY
- El motiu de fons és que amb aquest tipus de vista qualsevol actualització de la vista sempre es pot traduir a una única actualització de la taula de base i, per tant, sense ambigüïtat.
- L'estàndard expandeix les vistes actualitzables incloent certs tipus de vistes amb més d'una taula en el FROM o amb subconsultes (Ramakrishnan & Gehrke)

Vistes: No totes les vistes són actualitzables - Exemples

subministraments(nprov, nmat, qtt)
p1 m1 100
p2 m1 200
p2 m2 200

```
CREATE VIEW sumaqtt(material,suma) AS SELECT nmat, sum(qtt)
                                         FROM subministraments
                                         GROUP BY nmat
```

1. SELECT * FROM sumaqtt => m1 300
m2 200

2. DELETE de "m1,300" de la vista => DELETE "p1, m1, 100"
DELETE "p2, m1, 200"

Una única solució! Hauria de ser possible traduir-la.

Però no ho és ni a l'estàndard, ni als SGBD concrets

3. UPDATE de "m1, 300" per "m1, 301" =>

Com es tradueix? Hi ha múltiples solucions !!!

Vistes: No totes les vistes són actualitzables - Exemples

proveidors(nprov, nom, ...)
100 joan
subministraments(nprov, nmat, qtt, ...)
100 m1 5230

```
CREATE VIEW prov_subprov AS SELECT *
                                         FROM proveidors, subministraments
                                         WHERE proveidors.nprov = subministraments.nprov
```

1. SELECT * FROM prov_subprov => 100 joan ... 100 m1 5230

2. DELETE "100, joan,...m1,..." de la vista =>
1.1 DELETE "100, m1, 5230, ..." de subministraments
1.2 DELETE de proveidors, DELETE de subministraments
1.3 DELETE "100, joan," de proveidors
....

Múltiples solucions!!! No és possible

3. INSERT "200, pere, ..., 200, m2, ..." a la vista =>
INSERT "200, pere, ..." a proveidors, INSERT "200, m2, ..." a subministraments

Una solució! Hauria de ser possible

Esquema d'Informació

- Cada catàleg conté un esquema d'informació (*Information Schema*), a més dels esquemes definits pel propi usuari
- Tota la informació dels esquemes definits pels usuaris: noms i atributs de les taules, índexs, restriccions de columna, taula,... s'emmagatzema a la seva vegada a l'esquema d'informació. Així l'esquema d'informació és un esquema sobre esquemes: meta-dades !!
- L'esquema d'informació consisteix en un conjunt de vistes accessibles pels usuaris:
 - SCHEMATA: Informació de cada esquema del catàleg
 - DOMAINS: Informació sobre els dominis
 - TABLES: Informació sobre les taules
 - VIEWS: Informació sobre vistes
 - ASSERTIONS: Informació sobre restriccions
 - TRIGGERS: Informació sobre disparadors
 - ...
- Les vistes de l'esquema d'informació estan definides sobre un conjunt de *taules de sistema (definition schema)* accessibles només per l'administrador
- Altres, baixes i modificacions en aquestes taules de sistema són indirectes!!!

Privilegis

- Una Base de Dades té molts objectes, molts usuaris i molts grups d'usuari. No tots els usuaris han d'accedir a tots els objectes. L'SGBD ha d'establir un mecanisme de control d'accés dels usuaris sobre aquests objectes.
- Aquest mecanisme es basa en el concepte de PRIVILEGI:

L'autorització que es dóna a un
usuari / grup d'usuaris
per realitzar una
operació
sobre un
objecte d'un esquema

- Els privilegis s'assignen i es revoquen amb les sentències GRANT i REVOKE.

Privilegis

- Quan un usuari crea un esquema s'identifica amb la clàusula AUTHORIZATION i té tots els privilegis sobre ell. L'esquema serà inaccessible a altres usuaris fins que el propietari d'aquest esquema autoritzi privilegis a altres usuaris amb la sentència GRANT.
 - Quan una sessió es comença amb una connexió tenim l'oportunitat d'indicar l'usuari amb la clàusula USER
- CONNECT TO nom_servidor AS nom_connexio USER ident_usuari**
- Per tant, podrem executar una operació SQL només si l'usuari identificat té tots els privilegis necessaris per fer l'operació sobre els objectes involucrats

Privilegis

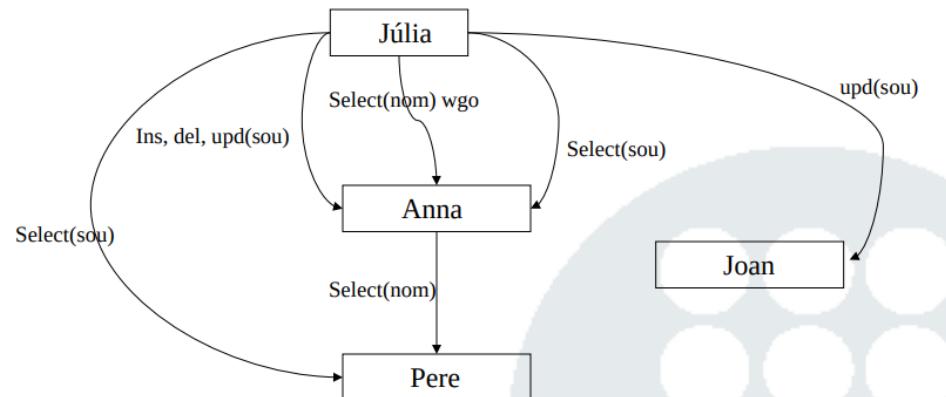
- SQL defineix 9 tipus de PRIVILEGIS:

- SELECT	Poden tenir associada una llista d'atributs	Aplicables a una taula o vista
- INSERT		
- UPDATE		
- DELETE		
- REFERENCES		
És el dret a fer referència a una taula en una restricció d'integritat		
- USAGE		
És el dret a utilitzar altres objectes en les pròpies declaracions		
- TRIGGER		
És el dret a definir disparadors sobre una taula		
- EXECUTE		
És el dret a executar una peça de codi, per exemple procediments		
- UNDER		
És el dret a crear subtipus d'un tipus donat		
- ALL		

Privilegis: Sentències GRANT i REVOKE

- Autoritzant privilegis:
GRANT privilegis ON objectes TO usuaris [WITH GRANT OPTION];
 - Revocant privilegis:
REVOKE [GRANT OPTION FOR] privilegis ON objectes FROM usuaris {CASCADE | RESTRICT};
 - CASCADE: Es revoquen també tots els privilegis concedits a partir dels privilegis revocats, excepte que estiguin autoritzats per una altra via
 - RESTRICT: No es permet revocar el privilegi si amb CASCADE es revoqués algun altre privilegi
- Exemple:
Júlia:
CREATE TABLE empleats (n integer, nom char(10), sou integer);
GRANT insert, delete, update(sou) ON empleats TO Anna;
GRANT select(nom) ON empleats TO Anna WITH GRANT OPTION;
GRANT select(sou) ON empleats TO Anna, Pere;
GRANT update(sou) ON empleats TO Joan;
Anna:
GRANT select(nom) ON empleats TO Pere;
Joan:
UPDATE empleats SET sou=4;

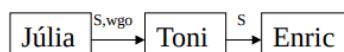
Privilegis: Diagrama d'autoritzacions



Júlia: GRANT insert, delete, update(sou) ON empleats TO anna;
 GRANT select(nom) ON empleats TO anna WITH GRANT OPTION;
 GRANT select(sou) ON empleats TO anna, pere;
 GRANT update(sou) ON empleats TO joan;
Anna: GRANT select(nom) ON empleats TO pere;
Júlia: REVOKE select(nom) ON empleats FROM anna CASCADE;

Privilegis: Moltes Subtileses

- Si la Júlia en lloc de GRANT select(sou) ON empleats TO Pere hagués fet
`GRANT select(nom) ON empleats TO Pere`
 després del revoke el pere conservaria el privilegi select(nom)
- Si la Júlia en lloc de GRANT select(sou) ON empleats TO Pere hagués fet
`GRANT select ON empleats TO pere`
 després del revoke el pere conservaria el privilegi select
- El Joan necessita privilegis diferents per:
`UPDATE empleats SET sou=4;`
`UPDATE empleats SET sou=sou-1;`
- Júlia: GRANT select ON empleats TO Toni WITH GRANT OPTION;
 Toni: GRANT select ON empleats TO Enric;
 Júlia: REVOKE GRANT OPTION FOR select ON empleats FROM Toni CASCADE;



Privilegis i Vistes

■ Exemple:

Donada una taula empleats (nemp, adreça, sou) es vol que l'empleat Joan només pugui veure el seu sou.

GRANT + VISTES = precisió en les autoritzacions

`CREATE VIEW sou_joan AS`

`SELECT sou FROM empleats WHERE nemp='joan';`
`GRANT SELECT ON Sou_joan TO joan;`

■ Els privilegis que es poden donar sobre una vista són els mateixos que es poden donar a una taula: SELECT, INSERT, UPDATE, DELETE

Privilegis: ROLS

- Nombre elevat de GRANT per implantar la BD !!
- ROL:** és una agrupació de privilegis definida per a un grup d'usuaris específics.
 - permét al DBA estandarditzar i canviar els privilegis de molts usuaris tractant-los com a membres d'una classe
 - Similar al concepte de grup dels SO
- DBA :**

<code>CREATE ROLE lector</code>	<code>CREATE ROLE escriptor;</code>
<code>GRANT select ON T1 TO lector</code>	<code>GRANT update ON T1 TO escriptor</code>
<code>GRANT select ON T2 TO lector</code>	<code>GRANT update ON T2 TO escriptor</code>
	<code>GRANT lector TO escriptor</code>
- DBA o un usuari amb grant option**

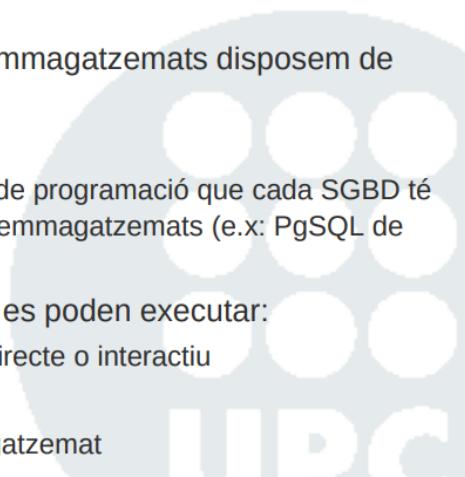
<code>GRANT lector TO Joan, Anna WITH GRANT OPTION</code>
<code>GRANT escriptor to Toni</code>
- DBA:** Manipulació dinàmica de privilegis

<code>GRANT select on T3 to lector</code>
<code>GRANT update on T3 to escriptor</code>
<code>REVOKE update on T2 from escriptor CASCADE</code>
<code>GRANT update(a) on T2 to escriptor</code>
- Usuari:** Han d'activar els rols

<code>Anna: SET ROLE lector</code>

Procediments Emmagatzemats

- Un procediment emmagatzemat és una funció definida per un usuari que, un cop creat, s'emmagaatzema a la BD i es tractat com un objecte més de la BD.
- Per poder escriure procediments emmagatzemats disposem de dos tipus de sentències:
 - Sentències pròpies de l'SQL
 - Sentències pròpies del llenguatge de programació que cada SGBD té per implementar els procediments emmagatzemats (e.x: PgSQL de PostgreSQL)
- Els procediments emmagatzemats es poden executar:
 - De manera interactiva, amb SQL directe o interactiu
 - Des d'una aplicació
 - Des d'un altre procediment emmagatzemat



Procediments emmagatzemats: Exemple de creació

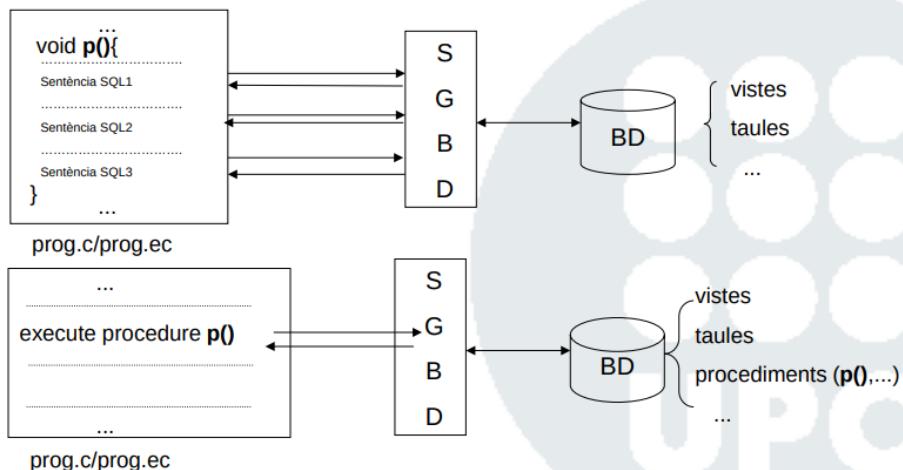
```
CREATE FUNCTION trobar_ciutat(dni_client varchar(9)) RETURNS varchar(15) AS $$  
DECLARE  
    -- DEFINICIÓ DE LES VARIABLES LOCALS DEL PROCEDIMENT  
    ciutat_client varchar(15);  
  
BEGIN  
    SELECT ciutat into ciutat_client  
    FROM clients  
    WHERE dni=dni_client;  
  
    RETURN ciutat_client;  
END;  
$$ LANGUAGE plpgsql;
```

Si es vol esborrar el procediment emmagatzemat cal fer:

```
DROP FUNCTION trobar_ciutat(varchar(9));
```

Procediments emmagatzemats

- Serveixen per:
 - Simplificar el desenvolupament d'applicacions
 - Millorar el rendiment de la BD
 - Controlar les operacions que els diferents usuaris realitzen contra la BD



Procediments emmagatzemats: Exemple d'execució

```
select * from trobar_ciutat('45678900');
```

Panel de Salida

	trobar_ciutat	character varying
1	Barcelona	

En aquest cas es tracta d'una execució de manera interactiva. Però com s'ha indicat abans també es pot fer execucions des de dins d'altres procediments o des d'applicacions en general.

Tema 4-7. Privilegis (ex2.2 - juny 2017)

Considereu la taula professors(*dni,nomProf,telefon*), propietat d'en Toni, i la seqüència de sentències:

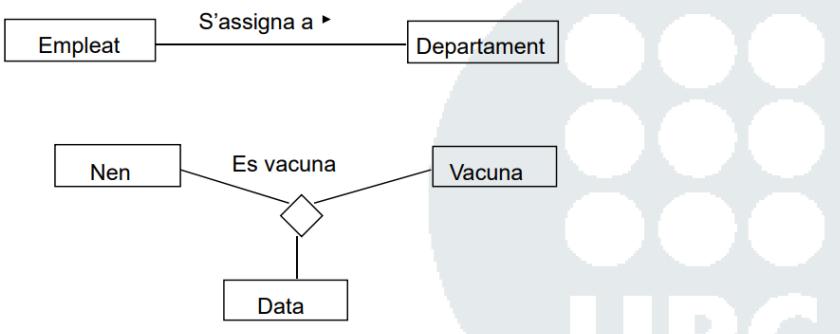
- 1 - Toni: GRANT SELECT ON professors TO Albert WITH GRANT OPTION
- 2 - Albert: GRANT SELECT ON professors TO Carme WITH GRANT OPTION
- 3 - Carme: GRANT SELECT(*dni,telefon*) ON professors TO Dolors WITH GRANT OPTION
- 4 - Carme: GRANT SELECT(*dni,nomProf*) ON professors TO Se WITH GRANT OPTION
- 5 - Toni: GRANT SELECT ON professors TO Se
- 6 - Toni: GRANT SELECT(*telefon*) ON professors TO Xavi
- 7 - Dolors: GRANT SELECT(*dni,telefon*) ON professors TO Xavi WITH GRANT OPTION
- 8 - Se: GRANT SELECT(*dni,telefon*) ON professors TO Xavi
- 9 - Dolors: GRANT SELECT(*dni*) ON professors TO Elena
- 10 - Se: GRANT SELECT(*dni*) ON professors TO Elena
- 11 - Toni: REVOKE SELECT ON professors FROM Se RESTRICT
- 12 - Carme: REVOKE SELECT(*dni,telefon*) ON professors FROM Dolors RESTRICT
- 13 - Dolors: REVOKE SELECT(*dni*) ON professors FROM Se CASCADE
- 14 - Albert: REVOKE SELECT ON professors FROM Carme CASCADE
- 15 - Toni: REVOKE SELECT ON professors FROM Albert RESTRICT

Quines d'aquestes sentències, si n'hi ha cap, no s'executaran amb èxit o no tindran cap efecte sobre la base de dades? Raoneu la resposta. Assumirem que les sentències que no s'executin amb èxit, no tindran cap efecte i es continuará amb la sentència següent.

- (8) Falla perquè se no té S(t) wqo.
- (12) Falla perquè en cascada algú més a part de la Dolors perdria el privilegi (raon)
- (13) Falla perquè la Dolors no havia fet cap G a la Se.

Associacions

- Una associació és la representació d'una relació entre dos o més objectes.
- Poden ser binàries o d'ordre superior a dos.



Introducció: etapes del disseny de BD

• Disseny conceptual:

- S'obté una estructura de la informació de la futura BD independent de la tecnologia que cal emprar.
- S'obté un esquema en un llenguatge com ara UML, ER, etc.

• Disseny lòtic:

- El resultat del disseny conceptual s'adapta al model de l'SGBD amb el qual es desitja implementar la BD.
- En el cas del model relacional s'obtenen les relacions amb els seus atributs, claus primàries i claus foranes.

• Disseny físic:

- Es transforma l'estructura obtinguda a l'etapa del disseny lòtic amb l'objectiu d'aconseguir una major eficiència i, a més, es completa amb aspectes d'implementació física que dependran de l'SGBD concret a utilitzar (p.e.: índexos).

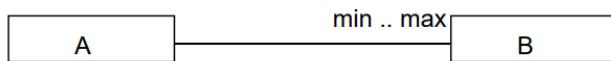
Classes d'objectes i atributs

Empleat
dni
nom
cognom
sou

Clau externa Empleat: dni

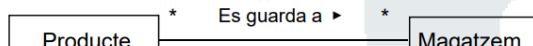
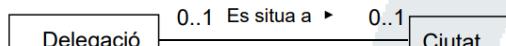
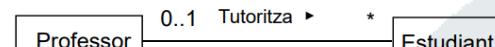
- **Classe d'objectes:** Descriu un conjunt d'objectes similars dels quals interessen les mateixes propietats.
- Els objectes tenen **identitat** (OID) i són distingibles entre ells.
- **Atribut:** Propietat compartida pels objectes d'una classe.
- **Clau externa**
 - Conjunt d'atributs (mínim) que permeten identificar els objectes d'una classe.
 - UML no proporciona notació gràfica per assenyalar les claus externes de les classes d'objectes.
 - Es poden descriure amb restriccions textuales.

Multiplicitat de les associacions binàries (1)

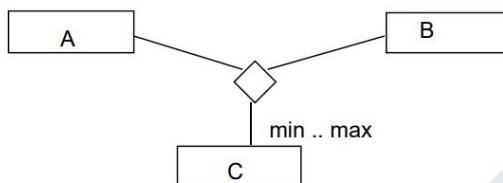


- Defineix quantes instàncies d'una classe B es poden associar amb una instància de la classe A.
- Si max és * indica que el màxim no està limitat.
- Si min=max es pot indicar amb un únic valor, per exemple: 1 en comptes d'1..1.
- La multiplicitat 0..* s'indica abreviadament *.
- Per simplificar, considerarem només els valors 0, 1 i * com a valors mínims i/o màxims de la multiplicitat.

Multiplicitat de les associacions binàries: exemples

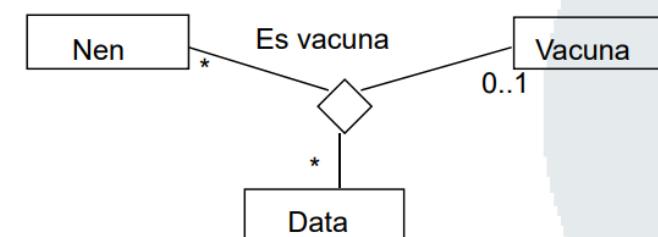
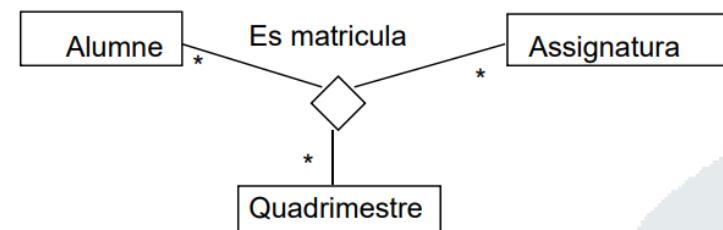


Multiplicitat de les associacions ternàries

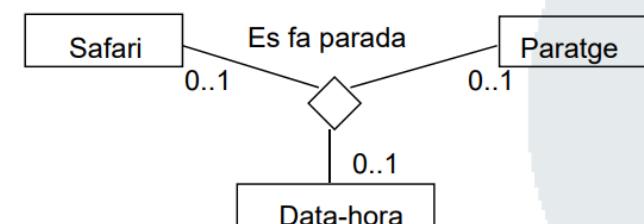
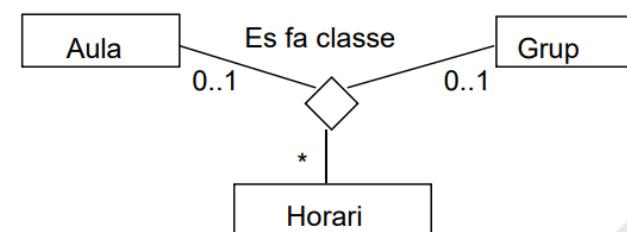


- Donades una instància *a* d'A i una instància *b* de B qualsevol, la multiplicitat al costat C defineix quantes instàncies de C es poden associar amb la parella formada per *a* i *b*.

Multiplicitat de les associacions ternàries: exemples (1)

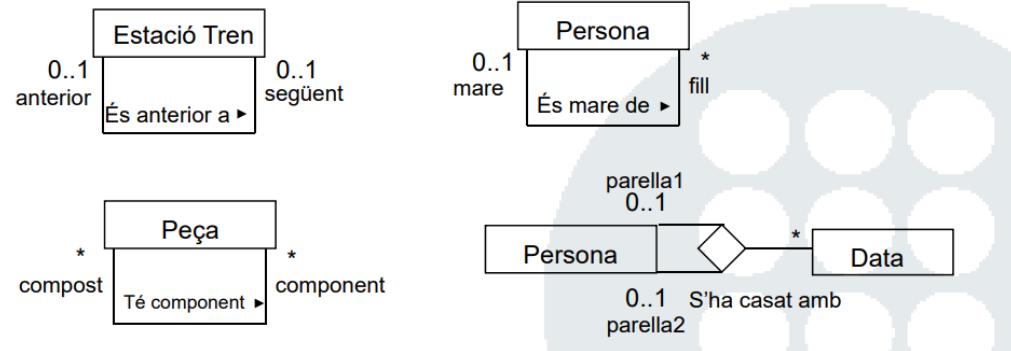


Multiplicitat de les associacions ternàries: exemples (2)



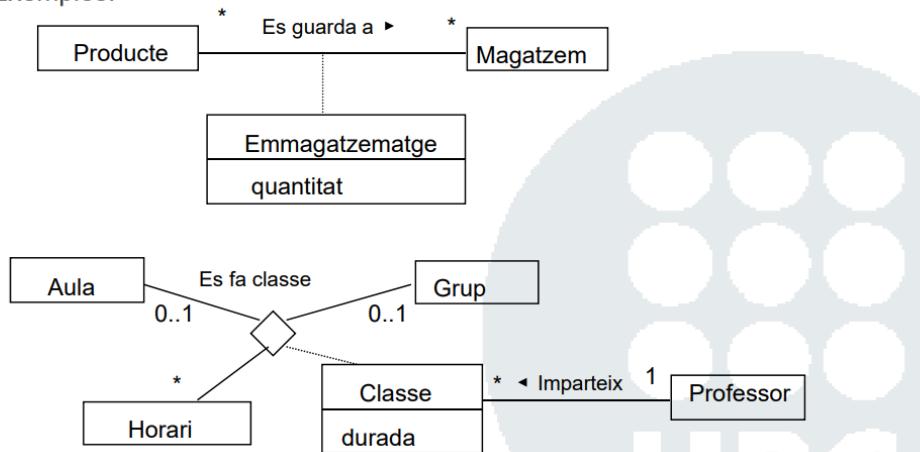
Associacions recursives

- Associacions en les que una mateixa classe d'objectes hi participa més d'una vegada.
- Exemples:



Classes associatives

- Representa una associació que es pot veure com una classe (així pot tenir atributs i associacions).
- Exemples:

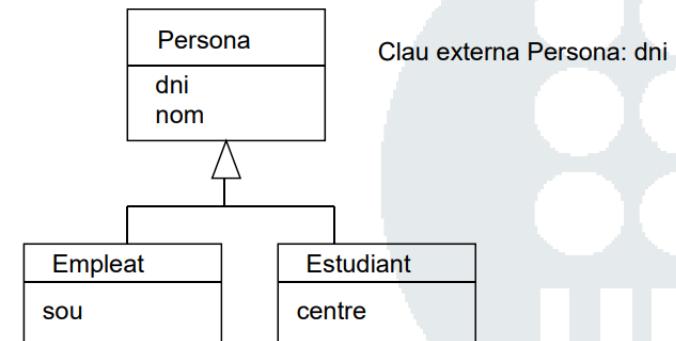


Transformació de les associacions binàries

- Per transformar una associació (binària o n-ària) cal prèviament haver transformat totes les classes d'objectes que associa.
- Casos a considerar per les associacions binàries:
 - Cas un a molts.
 - Cas un a un.
 - Cas molts a molts.

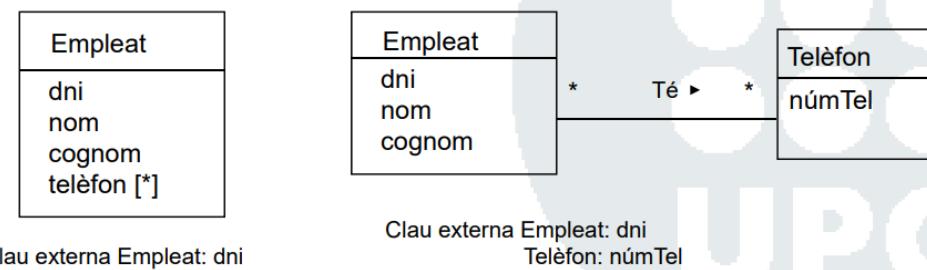
Generalització/especialització

- Permet reflectir el fet que hi ha una classe d'objectes general, que anomenem superclasse, que es pot especialitzar en entitats subclasses.
 - La superclass representa les característiques comunes a totes les subclasses.
 - Les subclasses representen les característiques pròpies de les especialitzacions.



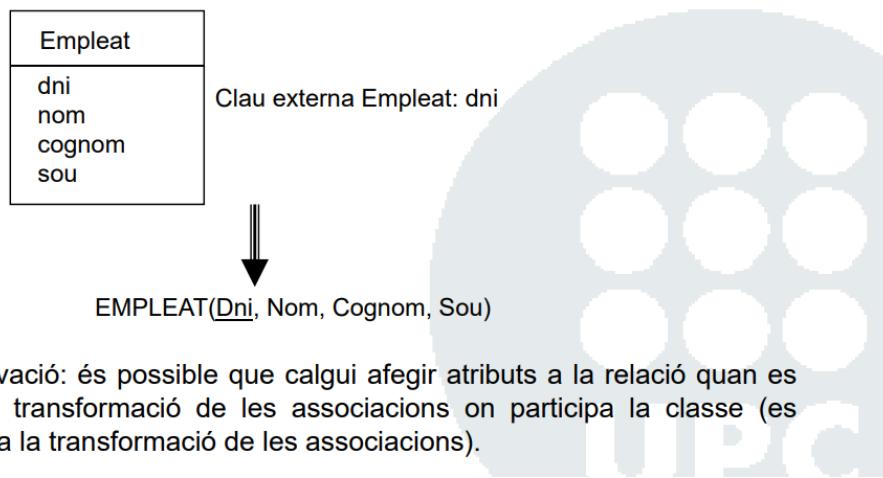
Transformacions inicials del model de partida

- Per poder aplicar les transformacions que veurem més endavant cal que el model de partida compleixi els requisits següents:
 - Totes les classes d'objectes (no associatives ni subclasses d'altres classes) han de tenir restricció de clau externa.
 - Tots els atributs han de ser univaluats (no poden ser multivaluats).
- Si algun d'aquests requisits no es compleix cal transformar el model de partida de manera que es compleixi:
 - Afegir atributs amb restriccions de clau externa a les classes que calgui.
 - Transformar els atributs multivaluats en associacions.
- Exemple de transformació d'un atribut multivaluat:



Transformació de les classes d'objectes i els seus atributs

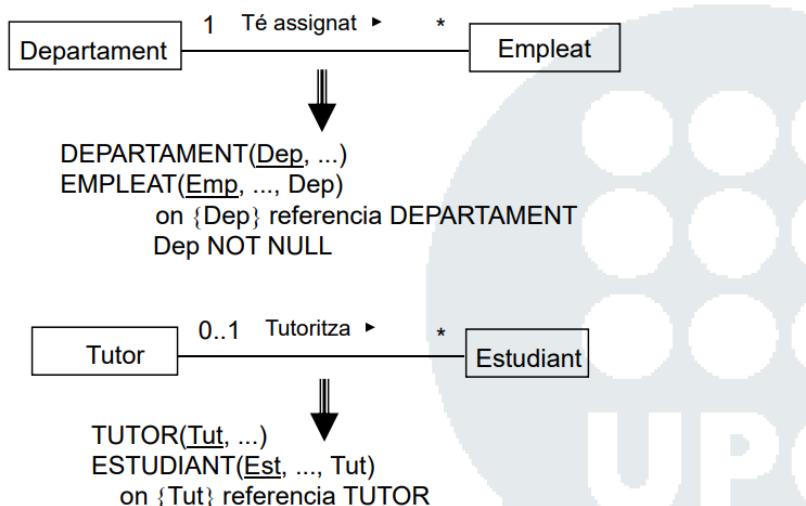
- Cada classe d'objectes es transforma en una relació
 - La clau primària de la relació serà la clau externa de la classe d'objectes.
 - Els atributs de la relació seran els atributs de la classe d'objectes.



- Observació: és possible que calgui afegir atributs a la relació quan es faci la transformació de les associacions on participa la classe (es veurà a la transformació de les associacions).

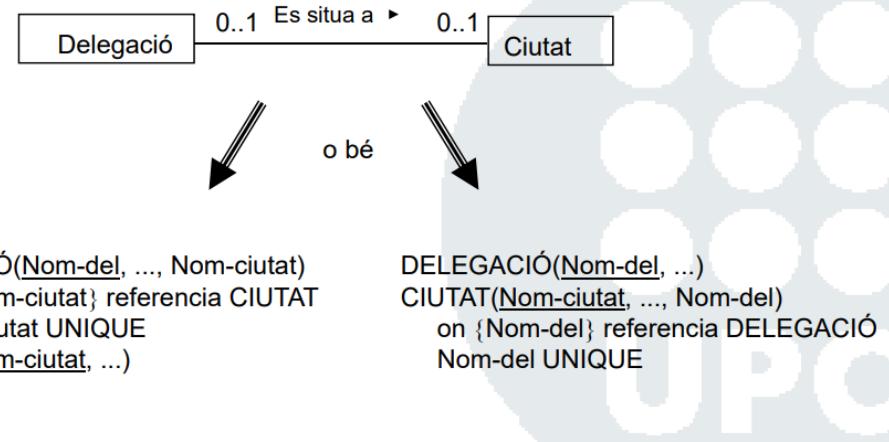
Associacions binàries: cas un a molts

- Cas un a molts: la multiplicitat és 0..1 o 1 a l'extrem 'un' i * a l'extrem 'molts' de l'associació.
- La transformació consisteix en afegir una clau forana a la relació que correspon a la classe de l'extrem 'molts' de l'associació per tal de referenciar a l'altra relació.



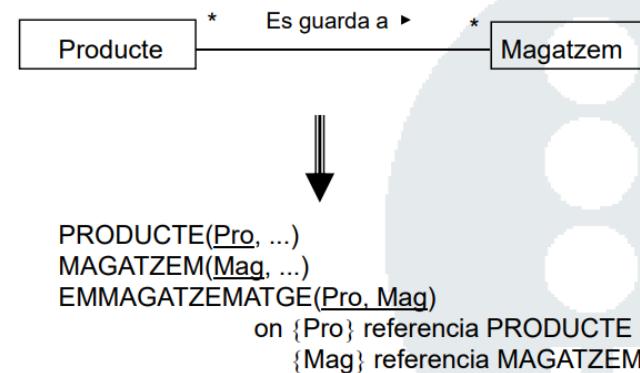
Associacions binàries: cas un a un

- Cas un a un significa que la multiplicitat és 0..1 a ambdós extrems de l'associació.
- La transformació consisteix en afegir a qualsevol de les dues relacions (correspondents a les classes associades) una clau forana que referenciï a l'altra relació.



Associacions binàries: cas molts a molts

- Cas molts a molts significa que la multiplicitat és * a ambdós extrems de l'associació.
- La transformació consisteix en definir una nova relació. La seva clau primària estarà formada pels atributs de la clau primària de les relacions corresponents a les classes dels dos extrems de l'associació.

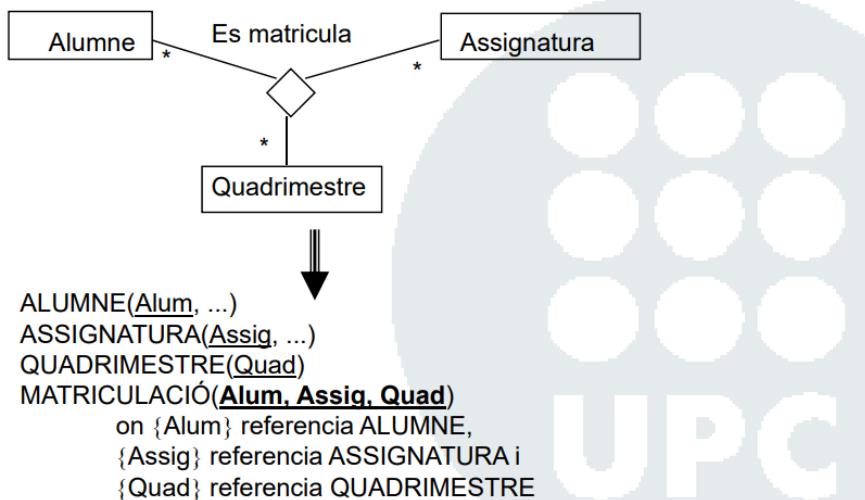


Transformació de les associacions n-àries

- Per transformar una associació (binària o n-ària) cal prèviament haver transformat totes les classes d'objectes que associa.
- Les associacions n-àries sempre es transformen en una **nova relació** que **conté els atributs que formen la clau de les n classes associades**.
- Analitzarem alguns casos que es poden donar en associacions ternàries i després descriurem el cas general:
 - Ternàries molts-molts-molts.
 - Ternàries molts-molts-un.
 - Ternàries molts-un-un.
 - Transformació general de les n-àries.
- Observació: per les associacions n-àries considerarem que el cas 'un' correspon només a 0..1 (no 1).

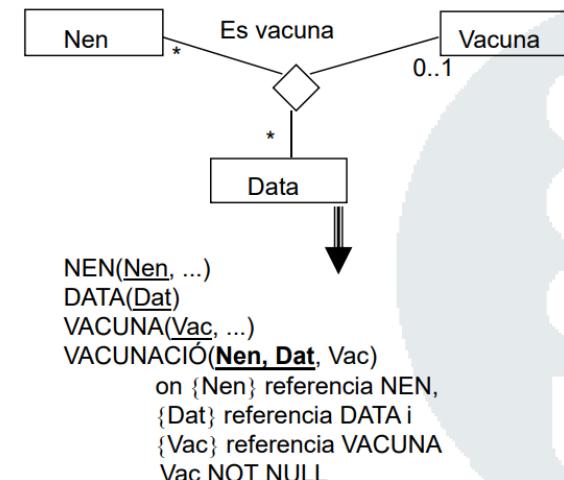
Associacions ternàries: molts-molts-molts

- En el cas molts-molts-molts la multiplicitat és * als tres extrems de l'associació.
- La clau primària de la nova relació està formada pels atributs de les claus de les classes corresponents als tres extrems de l'associació.



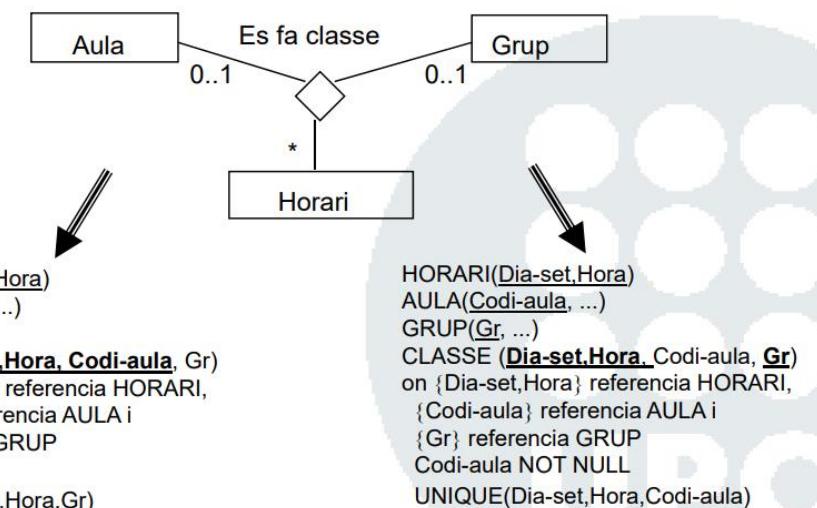
Associacions ternàries: molts-molts-un

- En el cas molts-molts-un la multiplicitat és * als dos extrems 'molts' i 0..1 a l'extrem 'un' de l'associació.
- La clau primària està formada pels atributs de les claus de les classes corresponents als dos extrems 'molts' de l'associació.



Associacions ternàries: molts-un-un

- En el cas molts-un-un la multiplicitat és * al l'extrem 'molts' i 0..1 als extrems 'un' de l'associació.
- La relació té dues claus candidates. Cadascuna està formada pels atributs de les claus de dues de les classes associades una de les quals és necessàriament la del costat 'molts'.

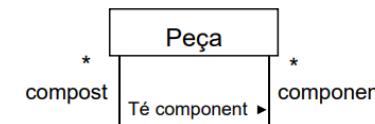


Transformació general de les nàries

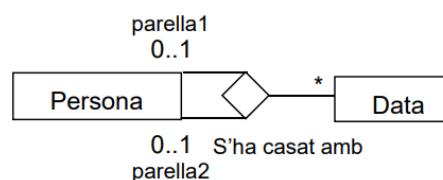
- En tots els casos, la transformació d'una interrelació nària consistirà en una **nova relació** que conté tots els atributs que formen les claus de les n classes associades.
- Si totes les classes estan connectades amb 'molts'**, la clau primària de la nova relació estarà formada per tots els atributs que formen les claus de les n classes associades.
- Si una o més classes estan connectades amb 'un'**, la clau primària de la nova relació estarà formada per les claus de n-1 de les classes associades amb la condició de que la classe, la clau de la qual no s'ha inclòs, ha de ser una de les que està connectada amb 'un'. Pot haver-hi diverses claus candidates.

Transformació de les associacions recursives (2)

- Exemples:



PEÇA(Codi-peça, ...)
COMPONENT(Codi-compost,Codi-component)
on {Codi-compost} referencia PEÇA
i {Codi-component} referencia PEÇA



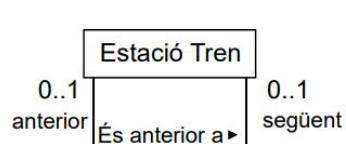
PERSONA(Codi-per, ...)
DATA(Dat)
CASAMENT(Codi-parella1,Codi-parella2,Dat)
on {Codi-parella1} referencia PERSONA,
{Codi-parella2} referencia PERSONA
{Dat} referencia DATA
Codi-parella2 NOT NULL
UNIQUE(Codi-parella2,Dat)

o bé

CASAMENT(Codi-parella1,Codi-parella2,Dat)
on {Codi-parella1} referencia PERSONA,
{Codi-parella2} referencia PERSONA
{Dat} referencia DATA
Codi-parella1 NOT NULL
UNIQUE(Codi-parella1,Dat)

Transformació de les associacions recursives (1)

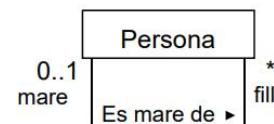
- Les **associacions recursives** es transformen de la mateixa manera que la resta d'associacions, és a dir, que cal tenir en compte si són binàries o nàries i també la seva multiplicitat i aplicar les transformacions corresponents.
- Exemples:



ESTACIO (Línia,Nom, ..., LíniaEstSeg,NomEstSeg)
on {LíniaEstSeg,NomEstSeg} referencia ESTACIO
UNIQUE(LíniaEstSeg,NomEstSeg)

o bé

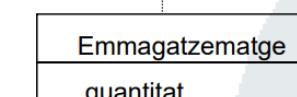
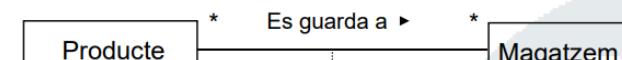
ESTACIO (Línia,Nom, ..., LíniaEstAnt,NomEstAnt)
on {LíniaEstAnt,NomEstAnt} referencia ESTACIO
UNIQUE(LíniaEstAnt,NomEstAnt)



PERSONA(Codi-per, ..., Codi-mare)
on {Codi-mare} referencia PERSONA

Transformació de les classes associatives (1)

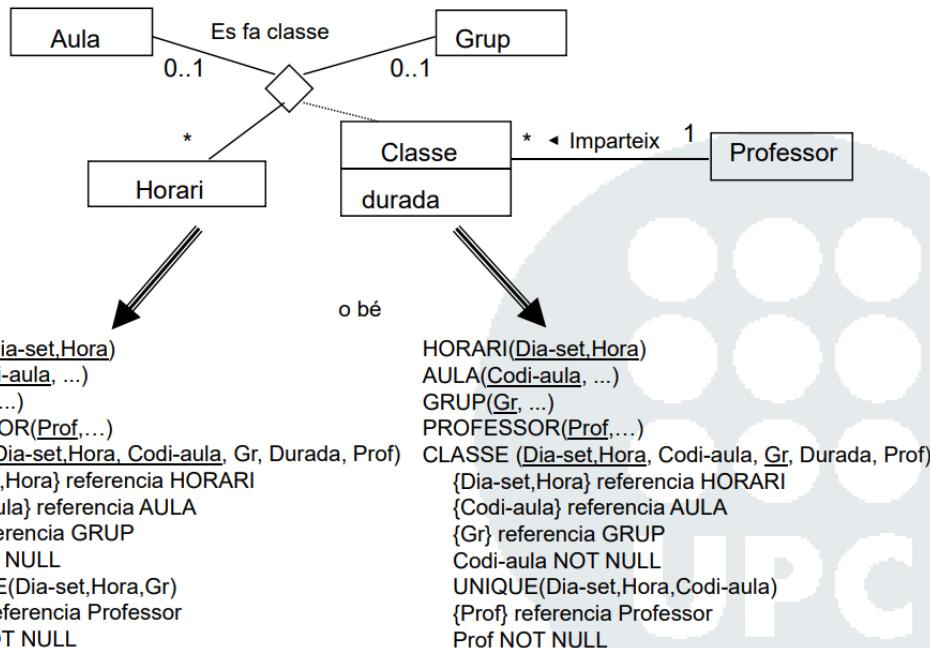
- La transformació de la seva associació és, alhora, la transformació de la classe associativa.
- Si la classe associativa té atributs, aleshores s'afegeixen com a atributs de la relació corresponent a la seva transformació.
- Exemple:



PRODUCTE(Pro, ...)
MAGATZEM(Mag, ...)
EMMAGATZEMATGE(Pro, Mag, Quantitat)
on {Pro} referencia PRODUCTE i
{Mag} referencia MAGATZEM

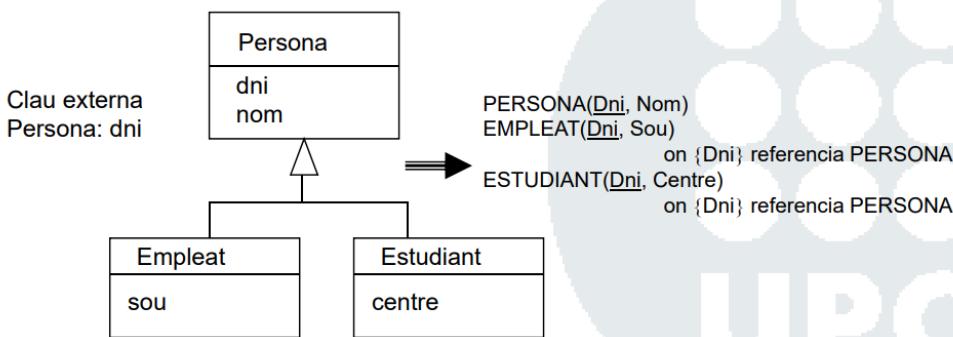
Transformació de les classes associatives (2)

- Exemple:



Transformació de la generalització/especialització

- **Cadascuna de les classes d'objectes** (superclasse i subclasses) que formen part d'una generalització/especialització **es transforma en una relació**.
- La relació corresponent a la **superclasse** té com a clau primària la clau de superclasse i conté tots els atributs de la superclasse (atributs comuns).
- Les relacions corresponents a les **subclasses** tenen com a clau primària la clau de la superclasse i contenen els atributs específics de la subclass.
- Exemple:



Resum de les transformacions

Elements del model de partida	Transformació al model relacional
Classe d'objectes	Relació
Associació binària: un a molts	Clau forana al costat molts
Associació binària: un a un	Clau forana a qualsevol costat
Associació binària: molts a molts	Relació
Associació n-ària	Relació
Associació recursiva	Com les no recursives: Clau forana per binàries un a un i un a molts Relació per binàries molts a molts i n-àries
Classe associativa	La transformació de la seva associació és, alhora, la transformació de la classe associativa.
Generalització/especialització	Relació per la superclasse i per cadascuna de les subclasses

Consideracions finals

- Les relacions corresponents a classes temporals com ara Data, Hora, Any, etc que no tenen atributs a part de la clau es poden eliminar.
- Quan hi ha diverses claus candidates per una relació, caldria prohibir les repetitions de valors per les claus alternatives (no primàries). En SQL es pot usar la clàusula UNIQUE.
- Restriccions que no es poden controlar mitjançant la definició de claus primàries i foranes de les relacions i que cal controlar amb altres mecanismes ('triggers', 'assertions', etc.).
 - Multiplicitats diferents de les considerades.
 - Restriccions de 'disjoint' i 'complete' que poden especificar-se en una generalització/especialització.
 - Restriccions textuals no expressables gràficament en UML poden requerir altres mecanismes de control (no és el cas de les restriccions de clau externa que es poden transformar en restriccions de clau primària de la relació corresponent).

Doneu una sentència SQL per obtenir els números i els noms dels departament situats a MADRID, que tenen empleats que guanyen més de 200000.

Pel joc de proves que trobareu al fitxer adjunt, la sortida ha de ser:

NUM_DPT	NOM_DPT
5	VENDES

[Fitxer adjunt](#)

Solució:

```
SELECT DISTINCT D.NUM_DPT, D.NOM_DPT
FROM DEPARTAMENTOS D, EMPLEATS E
WHERE D.CIUTAT_DPT = 'MADRID' AND E.SOU > 200000 AND D.num_dpt = E.num_dpt;
```

Doneu una sentència SQL per obtenir el nom del departament on treballa i el nom del projecte on està assignat l'empleat número 2

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

Nom_dpt	Nom_proj
MARKETING	IBDVID

[Fitxer adjunt](#)

Solució:

```
select D.NOM_DPT, P.nom_proj
from departaments D, empleats E, projectes P
where E.num_empl=2 and E.num_proj=P.num_proj and E.num_dpt=D.num_dpt;
```

Obtenir per cada departament situat a MADRID la mitjana dels sous dels seus empleats. Concretament, cal donar el número de departament, el nom de departament i la mitjana del sou.

Pel joc de proves que trobareu al fitxer adjunt, la sortida ha de ser:

NUM_DPT	NOM_DPT	SOU
5	VENDES	250000

[Fitxer adjunt](#)

Solució:

```
select D.NUM_DPT, D.nom_dpt, AVG(E.SOU) as mitjana_Sou
from departaments D natural inner JOIN empleats E
where D.ciutat_dpt = 'MADRID'
group by D.num_dpt;
```

Doneu una sentència SQL per obtenir el nom dels empleats que guanyen el sou més alt. Cal ordenar el resultat **descendentment** per nom de l'empleat.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

NOM_EMPL
JOAN

[Fitxer adjunt](#)

Solució:

```
SELECT distinct e.nom_empl
FROM empleats e
WHERE e.sou = ( SELECT max(e1.sou) FROM empleats e1)
order by e.nom_empl desc;
```

Doneu una sentència SQL per obtenir els números i els noms dels projectes que tenen assignats dos o més empleats.

Cal ordenar el resultat **descendentement** per número de projecte.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

NUM_PROJ	NOM_PROJ
1	IBDTEL

[Fitxer adjunt](#)

Solució:

```
select distinct p.num_proj, p.nom_proj
from projectes p
where 2 <= (select count(*) from empleats e where e.num_proj = p.num_proj)
order by p.num_proj desc;
```

Doneu una sentència SQL per incrementar en 500000 el pressupost dels projectes que tenen algun empleat que treballa a BARCELONA.

Pel joc de proves que trobareu al fitxer adjunt, el pressupost del projecte que hi ha d'haver després de l'execució de la sentència és 1100000

[Fitxer adjunt](#)

Solució:

```
UPDATE projectes
SET pressupost = pressupost + 500000
where 1 <= (select count(*)
            from departaments d, empleats e
            where (projectes.num_proj = e.num_proj and e.num_dpt = d.num_dpt and d.ciutat_dpt = 'BARCELONA'));
```

Doneu una sentència SQL per obtenir la quantitat d'empleats que treballen a la ciutat de MADRID.

Pel joc de proves que trobareu al fitxer adjunt, la sortida ha de ser:

EMPL_MADRID
1

[Fitxer adjunt](#)

Solució:

```
select count(*) as EMPL_MADRID
from empleats e, departaments d
where e.num_dpt = d.num_dpt and d.ciutat_dpt = 'MADRID';
```

Obtenir per cada departament situat a MADRID quin és el sou més gran. Concretament, cal llistar el número de departament i el sou més gran. El resultat ha d'estar ordenat ascendentment per número de departament.

Pel joc de proves que trobareu al fitxer adjunt, la sortida ha de ser:

NUM_DPT	SOU
5	250000

[Fitxer adjunt](#)

Solució:

```
SELECT e.num_dpt, MAX(e.sou) AS SOU
FROM empleats e
INNER JOIN departaments d ON e.num_dpt = d.num_dpt
WHERE d.ciutat_dpt = 'MADRID'
GROUP BY e.num_dpt
ORDER BY e.num_dpt ASC;
```

Doneu una sentència SQL per obtenir el número i el nom dels departaments que no tenen cap empleat que visqui a MADRID.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

NUM_DPT	NOM_DPT
3	MARKETING

[Fitxer adjunt](#)

Solució:

```
select d.num_dpt, d.nom_dpt
from departaments d
where not exists (select * from empleats e where e.num_dpt = d.num_dpt and
e.ciutat_empl='MADRID');
```

Doneu una sentència SQL per obtenir les ciutats on hi viuen empleats però no hi ha cap departament.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

CIUTAT_EMPL
GIRONA

[Fitxer adjunt](#)

Solució:

```
select distinct e.ciutat_empl
from empleats e
where not exists (select * from departaments d where e.ciutat_empl = d.ciutat_dpt);
```

Doneu una sentència SQL per obtenir el número i nom dels departaments que tenen dos o més empleats que viuen a ciutats diferents.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

NUM_DPT	NOM_DPT
3	MARKETING

[Fitxer adjunt](#)

Solució:

```
select d.num_dpt, d.nom_dpt
from departaments d
where 2 <= (select count(distinct e.ciutat_empl) from empleats e where e.num_dpt =
d.num_dpt);
```

Tenint en compte l'esquema de la BD que s'adjunta, proposeu una sentència de creació de les taules següents:

comandes(numComanda, instantComanda, client, encarregat, supervisor)

productesComprats(numComanda, producte, quantitat, preu)

La taula *comandes* conté les comandes fetes.

La taula *productesComprats* conté la informació dels productes comprats a les comandes de la taula *comandes*.

En la creació de les taules cal que tingueu en compte que:

- No hi poden haver dues comandes amb un mateix número de comanda.
- Un client no pot fer dues comandes en una mateix instant.
- L'encarregat és un empleat que ha d'existeix necessàriament a la base de dades, i que té sempre tota comanda.
- El supervisor és també un empleat de la base de dades i que s'assigna a algunes comandes en certes circumstàncies.
- No hi pot haver dues vegades un mateix producte en una mateixa comanda. Ja que en cas de el client compri més d'una unitat d'un producte en una mateixa comanda s'indica en l'atribut quantitat.
- Un producte sempre s'ha comprat en una comanda que ha d'existeix necessàriament a la base de dades.
- La quantitat de producte comprat en una comanda no pot ser nul, i té com a valor per defecte 1.
- Els atributs numComanda, instantComanda, quantitat i preu són de tipus *integer*.
- Els atributs client, producte són *char(30)*, i *char(20)* respectivament.
- L'atribut instantComanda no pot tenir valors nuls.

Respecteu els noms i l'ordre en què apareixen les columnes (fins i tot dins la clau o claus que calgui definir). Tots els noms s'han de posar en majúscules/minúscules com surt a l'enunciat.

create table comandes

```
(    numComanda integer,  
    instantComanda integer not null,  
    client char(30),  
    encarregat integer not null,  
    supervisor integer,  
    primary key (numComanda),  
    unique (instantComanda, client),  
    foreign key (encarregat) references empleats (num_empl),  
    foreign key (supervisor) references empleats (num_empl));
```

Doneu una sentència SQL per obtenir el nom dels professors que o bé se sap el seu número de telèfon (valor diferent de null) i tenen un sou superior a 2500, o bé no se sap el seu número de telèfon (valor null) i no tenen cap assignació a un despatx amb superfície inferior a 20.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

NomProf
toni

Fitxer adjunt

Solució:

```
select p.nomprof  
from professors p  
where ((p.telefon is not null and p.sou > 2500) or (p.telefon is null and not exists(select * from  
assignacions a, despatxos d where p.dni=a.dni and a.numero=d.numero and a.modul=d.modul  
and d.superficie < 20)));
```

create table productesComprats

```
(    numComanda integer,  
    producte char(20),  
    quantitat integer not null default 1,  
    preu integer,  
    primary key (numComanda, producte),  
    foreign key (numComanda) references comandes (numComanda));
```

Donar una sentència SQL per obtenir els professors que tenen alguna assignació finalitzada (instantFi different de null) a un despatx amb superfície superior a 15 i que cobren un sou inferior o igual a la mitjana del sou de tots els professors. En el resultat de la consulta ha de sortir el dni del professor, el nom del professor, i el darrer instant en què el professor ha estat assignat a un despatx amb superfície superior a 15.

Pel joc de proves que trobareu al fitxer adjunt, la sortida ha de ser:

DNI NomProf Darrer_instant

111 toni 344

Fitxer adjunt

Solució:

```
select p.dni, p.nomprof, max(a.instantfi)
from professors p, assignacions a, despatxos d
where ((a.instantfi is not null)
and (a.dni = p.dni)
and a.numero = d.numero
and a.modul = d.modul
and d.superficie > 15)
and p.sou <= (select avg(sou) from professors))
group by p.dni;
```

Suposem la base de dades que podeu trobar al fitxer adjunt.

Suposem que aquesta base de dades està en un estat on no hi ha cap fila.

Doneu una seqüència de sentències SQL d'actualització (INSERTs i/o UPDATEs) sobre la taula que assignacions que violi la integritat referencial de la clau forana que referencia la taula Despatxos. Les sentències **NOMÉS** han de violar aquesta restricció.

Fitxer adjunt

Solució:

```
insert into professors values('111', 'Elias', 987654,999999);
insert into assignacions values ('111', 'OMEGA','118',109,344);
```

Doneu una seqüència d'operacions d'algebra relacional per obtenir el nom del departament on treballa i el nom del projecte on està assignat l'empleat número 2.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

Nom_dpt Nom_proj

MARKETING IBDVID

Fitxer adjunt

Solució:

```
A = empleats(num_empl = 2)
R = A * projectes
C = R * departaments
D = C[nom_dpt, nom_proj]
```

Suposeu la base de dades que podeu trobar al fitxer adjunt.

Doneu una seqüència de sentències SQL d'actualització (INSERTs i/o UPDATEs) de tal manera que, un cop executades, el resultat de la consulta següent sigui el que s'indica. **El nombre de files de cada taula ha de ser el més petit possible, i hi ha d'haver com a màxim un professor.**

Per a la consulta:

```
Select count(*) as quant  
From assignacions ass  
Where ass.instantInici>50  
Group by ass.instantInici  
order by quant;
```

El resultat haurà de ser:

```
quant  
1  
2
```

[Fitxer adjunt](#)

Solució:

```
insert into professors values ('111', 'juan carlos', 64672878, 3000);  
insert into despatxos values ('A5', '34', 50);  
insert into despatxos values ('omega', '108', 25);  
insert into assignacions values ('111', 'A5', '34', 55, null);  
insert into assignacions values ('111', 'omega', '108', 55, null);  
  
insert into assignacions values ('111', 'omega', '108', 60, null);
```

Doneu una seqüència d'operacions d'algebra relacional per obtenir el número i nom dels departaments tals que tots els seus empleats viuen a MADRID. El resultat no ha d'incloure aquells departaments que no tenen cap empleat.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

Num_dpt	Nom_dpt
3	MARKETING

[Fitxer adjunt](#)

Solució:

```
A = empleats(ciutat_empl <> 'MADRID')  
B = departaments*A  
C=B[num_dpt, nom_dpt, planta, edifici, ciutat_dpt]  
F = departaments*empleats  
G = F[num_dpt, nom_dpt, planta, edifici, ciutat_dpt]
```

D = G-C

E = D[num_dpt, nom_dpt]

Doneu una seqüència d'operacions de l'àlgebra relacional per obtenir el número i nom dels departaments que tenen dos o més empleats que viuen a ciutats diferents.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

Num_dpt	Nom_dpt
3	MARKETING

[Fitxer adjunt](#)

Solució:

```
A = empleats[ciutat_empl, num_dpt,num_empl]
B = A {ciutat_empl ->ciutat_empl1, num_dpt -> num_dpt1, num_empl -> num_empl1}
C = A[num_dpt = num_dpt1, ciutat_empl <> ciutat_empl1]B
D = C*departaments
E = D[num_dpt,nom_dpt]
```

Donar una seqüència d'operacions d'àlgebra relacional per obtenir informació sobre els despatxos que només han estat ocupats per professors amb sou igual a 100000. Es vol obtenir el modul i el numero d'aquests despatxos.

Pel joc de proves que trobareu al fitxer adjunt, la sortida ha de ser:

Modul	Numero
Omega	128

[Fitxer adjunt](#)

Solució:

```
A = professors(sou=>100000)
B = A*assignacions
C = B[modul, numero]
D = assignacions[modul, numero]
E = D-C
```

Doneu una sentència SQL per obtenir el número i nom dels departaments que tenen 2 o més empleats que viuen a la mateixa ciutat.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

NUM_DPT	NOM_DPT
3	MARKETING

[Fitxer adjunt](#)

Solució:

```
select d.num_dpt, d.nom_dpt
from departaments d
WHERE d.num_dpt IN (
    SELECT e.num_dpt
    FROM empleats e
    WHERE e.num_dpt = d.num_dpt
    GROUP BY e.ciutat_empl
    HAVING COUNT(*) >= 2
);
```

Doneu una sentència SQL per esborrar els departaments que no tenen cap empleat.

Pel joc de proves que trobareu al fitxer adjunt, els departaments que hi ha d'haver a la taula departaments després de l'execució de la sentència són els següents:

NUM_DPT

3

[Fitxer adjunt](#)

Solució:

```
DELETE FROM departaments  
WHERE num_dpt NOT IN (SELECT DISTINCT num_dpt FROM empleats);
```

Doneu una sentència d'inserció de files a la taula *cost_ciutat* que l'ompli a partir del contingut de la resta de taules de la base de dades. Tingueu en compte el següent:

Hi haurà una fila de la taula per cada ciutat on hi ha un departament. El valor de l'atribut cost serà la suma del sou dels empleats dels departaments situats a la ciutat.

Només han de sortir les ciutats on hi ha departament que tinguin empleats.

Pel joc de proves públic del fitxer adjunt, un cop executada la sentència d'inserció, a la taula *cost_ciutat* hi haurà les tuples següents:

CIUTAT_DPT COST

BARCELONA 100

[Fitxer adjunt](#)

Solució:

```
INSERT INTO cost_ciutat (ciutat_dpt, cost)  
SELECT DISTINCT d.ciutat_dpt, SUM(e.sou) as cost  
FROM departaments d  
INNER JOIN empleats e ON d.num_dpt = e.num_dpt  
GROUP BY d.ciutat_dpt;
```

Doneu una seqüència d'operacions d'àlgebra relacional per obtenir el nom dels empleats que guanyen més que l'empleat amb num_empl 3.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

Nom_empl

JOAN

PERE

[Fitxer adjunt](#)

Solució:

```
A = empleats(num_empl = 3)  
B = A{sou -> sou1, nom_empl -> nom_empl1}  
R = B[nom_empl1,sou1]  
C = empleats(num_empl <> 3)  
D = C[nom_empl,sou]
```

E = R[sou1 < sou]D

F = E[nom_empl]

Donar una sentència SQL per obtenir per cada mòdul on hi hagi despatxos, la suma de les durades de les assignacions finalitzades (instantFi different de null) a despatxos del mòdul. El resultat ha d'estar ordenat ascendentment pel nom del mòdul.

Pel joc de proves que trobareu al fitxer adjunt, la sortida ha de ser:

MODUL	SUMAA
Omega	235

[Fitxer adjunt](#)

Solució:

```
select a.modul, (sum(a.instantFi)-sum(a.instantInici)) as SUMAA
from assignacions a
where a.instantFi is not null
group by a.modul
order by a.modul asc;
```

Doneu una sentència SQL per obtenir els departaments tals que tots els empleats del departament estan assignats a un mateix projecte.

No es vol que surtin a la consulta els departaments que no tenen cap empleat.

Es vol el número, nom i ciutat de cada departament.

Cal resoldre l'exercici **sense fer servir funcions d'agregació**.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

Num_dpt	Nom_dpt	Ciutat_dpt
1	DIRECCIO	BARCELONA

[Fitxer adjunt](#)

Solució:

```
select distinct d.num_dpt, d.nom_dpt, d.ciutat_dpt
from departaments d, empleats e
where d.num_dpt = e.num_dpt
and not exists (
    select *
        from empleats e1
        where e1.num_dpt = e.num_dpt
        and e1.num_proj <> e.num_proj);
```

Doneu una seqüència d'operacions en àlgebra relacional per obtenir el nom dels professors que o bé tenen un sou superior a 2500, o bé que cobren menys de 2500 i no tenen cap assignació a un despatx amb superfície inferior a 20.

Pel joc de proves que trobareu al fitxer adjunt, la sortida seria:

NomProf
toni

[Fitxer adjunt](#)

Solució:

A = professors(sou>2500)	F = B[dni]
B = professors(sou<2500)	G = F-E
C = despatxos(superficie<20)	H = G*B
D = C*assignacions	I = A_u_H
E = D[dni]	J = I[nomprof]

Doneu una sentència d'inserció de files a la taula **cost_ciutat** que l'ompli a partir del contingut de la resta de taules de la base de dades. Tingueu en compte el següent:

Cal inserir una fila a la taula **cost_ciutat** per cada ciutat on hi ha un o més departaments, però no hi ha cap departament que tingui empleats.

Per tant, només s'han d'inserir les ciutats on cap dels departaments situats a la ciutat tinguin empleats.

El valor de l'atribut cost ha de ser 0.

Pel joc de proves públic del fitxer adjunt, un cop executada la sentència d'inserció, a la taula **cost_ciutat** hi haurà les tuples següents:

CIUTAT_DPT	COST
BARCELONA	0

[Fitxer adjunt](#)

Solució:

```
insert
into cost_ciutat (ciutat_dpt, cost)
select distinct d.ciutat_dpt, 0 as cost
from departaments d
where not exists
(select *
 from empleats e, departaments d2
 where e.num_dpt = d2.num_dpt and d.ciutat_dpt = d2.ciutat_dpt);
```

Tenint en compte l'esquema de la BD que s'adjunta, proposeu una sentència de creació de la taula següent:

```
presentacioTFG(idEstudiant, titolTFG, dniDirector, dniPresident, dniVocal, instantPresentacio, nota)
```

Hi ha una fila de la taula per cada treball final de grau (TFG) que estigui pendent de ser presentat o que ja s'hagi presentat.

En la creació de la taula cal que tingueu en compte que:

- No hi pot haver dos TFG d'un mateix estudiant.
- Tot TFG ha de tenir un títol.
- No hi pot haver dos TFG amb el mateix títol i el mateix director.
- El director, el president i el vocal han de ser professors que existeixin a la base de dades, i tot TFG té sempre director, president i vocal.
- El director del TFG no pot estar en el tribunal del TFG (no pot ser ni president, ni vocal).
- El president i el vocal no poden ser el mateix professor.
- L'identificador de l'estudiant i el títol del TFG són chars de 100 caràcters.
- L'instant de presentació ha de ser un enter diferent de nul.
- La nota ha de ser un enter entre 0 i 10.
- La nota té valor nul fins que s'ha fet la presentació del TFG.

Respecteu els noms i l'ordre en què apareixen les columnes (fins i tot dins la clau o claus que calgui definir). Tots els noms s'han de posar en majúscues/minúscules com surt a l'enunciat.

[Fitxer adjunt](#)

Solució:

```
create table presentacioTFG
(idEstudiant char(100),
titolTFG char(100) not null,
dniDirector char(50) not null,
dniPresident char(50) not null,
dniVocal char(50) not null,
instantPresentacio integer not null,
nota integer default NULL
check (nota <= 10 and nota >= 0),
primary key(idEstudiant),
unique(titolTFG, dniDirector),
foreign key(dniDirector) references professors,
foreign key(dniPresident) references professors,
foreign key(dniVocal) references professors,
check (dniDirector <> dniPresident and dniDirector <> dniVocal and dniPresident <> dniVocal))
```