BASE DE DADES

Misericordia Avila Irigaray

Àlgebra relacional

Cal que T i S siguin relacions compatibles (tenen esquemes amb un conjunt d'atributs idèntic, i els dominis de cada parella d'atributs són els mateixos a T i a S)=

-UNIÓ

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

-INTERSECCIÓ

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

-DIFERENCIA

R =PERSONAL-ADM $-$ PERSONAL-LAB

No fa falta que siguin compatibles=

-PRODUCTE CARTESIÁ

R =MODUL-CN $\times$ OFICINA

-REANOMAMENT

R =PERSONAL-DOC {modul-de -> modul, num-de -> num}

-SELECCIÓ

R =OFICINA(modul-de='B2' AND superfície>16)

-PROJECCIÓ

R =PERSONAL-ADM[modul, num]

-COMBINACIÓ

R =MODUL-CN[modul=modul-de]OFICINA

R =MODUL-CN[modul=modul-de, sup-promig-of<=superfície]OFICINA

-NATURAL JOIN

R =PERSONAL-ADM[modul*modul-de, num*num-de]OFICINA

R =PERSONAL-ADM * MODUL-CN

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, superfície)		
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

Model relacional

Fila = Tupla

Columna = Atribut

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

Components lògics d'una base de dades

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];

Les assersions son restriccions d´integritat que afecten a més d'una taula.

CREATE ASSERTION nom CHECK (condició);

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

WITH CHECK OPTION = Si se modifica una la vista las modificaciones no puede alterar el número de filas que te da la vista

Vistes no actualitzables:

• 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

Procediments

Exemples:

CREATE FUNCTION nom_proc (param_entrada tipus) RETURNS tipus_retorn AS $$ .....

RETURN variable_retorn;

END;

$$LANGUAGE plpgsql;

Example:

Obtenir l'adreça (concretament el carrer, num_carrer i ciutat) d'un client:

CREATE TYPE TAdressa AS (

carrer varchar(20),

num_carrer varchar(4),

```
    ciutat varchar(15)
);
CREATE FUNCTION trobar_adressa_client (dni_client clients.dni%type) RETURNS TAdressa AS $$ DECLARE dadesCli TAdressa;
BEGIN
    SELECT carrer,num_carrer,ciutat INTO dadesCli FROM clients WHERE dni=dni_client;
    RETURN dadesCli;
END;
$$LANGUAGE plpgsql;
select * FROM trobar_adressa_client('45678900');
```

Bool FOUND:

• Una sentència SELECT ... INTO posa FOUND a True si el select obté una fila, i a False si no s'obté cap fila.

• Una sentència UPDATE, INSERT o DELETE posa FOUND a True si com a mínim una fila es veu afectada per la sentència, i a False si no queda afectada cap fila.

• Una sentència FOR. Dintre de cada iteració del FOR, el valor de la variable pot canviar segons les sentències que s'hi executen. Però en sortir del FOR és posa FOUND a True si s'ha iterat una o més vegades, sinó es posa a False.

Modifica l'import de la comanda a la taula comandes per cadascuna de les comandes d'un determinat client:

```
CREATE FUNCTION import_totes_com(dni_client clients.dni%type) RETURNS void AS $$ DECLARE
    num_comanda comandes.num_com%type;
    import comandes.import_total%type;
BEGIN
```

```
    FOR num_comanda IN SELECT num_com FROM comandes WHERE dni=dni_client
    LOOP
        SELECT SUM(ic.quantitat*i.preu_unitat) INTO import FROM items_comanda ic, items i WHERE i.num_item=ic.num_item AND ic.num_com=num_comanda;
        UPDATE comandes SET import_total=import WHERE num_com=num_comanda;
    END LOOP;
END;
$$LANGUAGE plpgsql;
```

Errors:

```
EXCEPTION
    WHEN raise_exception THEN RAISE EXCEPTION '%', SQLERRM;
    WHEN foreign_key_violation THEN RAISE EXCEPTION 'La comanda o el item no existeixen';
    WHEN OTHERS THEN RAISE EXCEPTION 'Error intern';
```

Disparadors

	ROW			STATEMENT
	INSERT	UPDATE	DELETE	INSERT, UPDATE, DELETE
BEFORE	■ OLD: No té valor definit. ■ NEW: Valors de la tupla després de l'insert	■ OLD: Valors de la tupla abans de l'update. ■ NEW: Valors de la tupla després de l'update	■ OLD: Valors de la tupla abans del delete. ■ NEW: No té valor definit.	Valor NULL
AFTER				

	ROW	STATEMENT
BEFORE	■ RETURN NEW – A continuació s'executa la sentència que activa el disparador en triggers INSERT, UPDATE. I si es modifica el valor de la variable NEW dins del procediment, s'executa la sentència amb el valor modificat. – No fa res en triggers DELETE ■ RETURN OLD – A continuació s'executa la sentència que activa el disparador en triggers DELETE, UPDATE. Si és un trigger UPDATE, no fa el canvi establert en la sentència. – No es fa res en triggers INSERT. ■ RETURN NULL – No executa la sentència que activa el disparador	Valor de retorn ignorat. Millor: RETURN NULL
AFTER	Valor de retorn ignorat. Millor: RETURN NULL	Valor de retorn ignorat. Millor: RETURN NULL

En el cas de l'UPDATE si el procediment retorna OLD el UPDATE no fa la modificació de la fila actual.

Introducció al disseny de bases de dades relacionals

Associacions binaries:

-1 a molts/1 a 1:

1 referencia a l'altra

-molts a molts:

Taula extra amb claus primàries de les dues altres taules

Associacions ternàries:

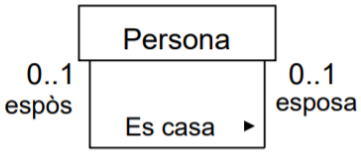
-molts a molts a molts:

Taula extra amb clau primària de les altres 3

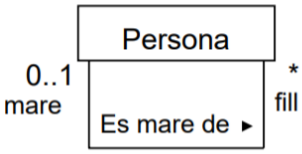
-molts a molts a un/molts a un a un:

Taula extra amb claus primàries les que tenen els molts + atribut primari de l'altra taula

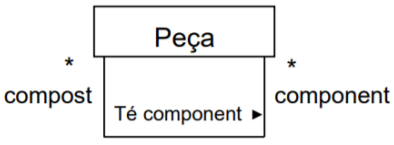
Associacions recursives:



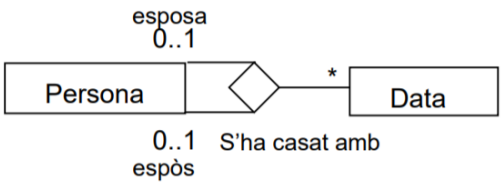
PERSONA(Codi-per, ..., Codi-conjuge)
on {Codi-conjuge} referencia PERSONA



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

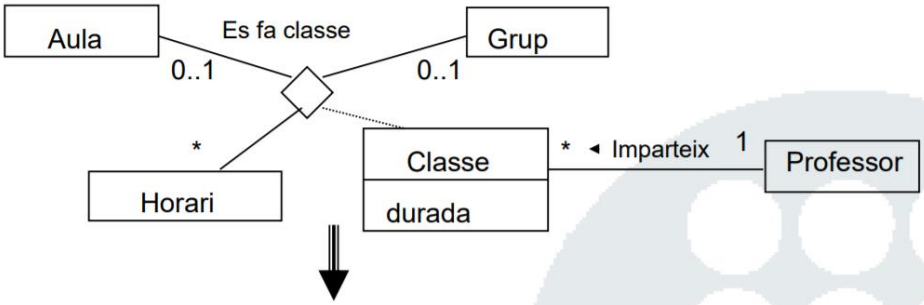


PEÇA(Codi-peça, ...)
COMPONENT(Codi-compost,Codi-component)
on {Codi-compost} referencia PEÇA
i {Codi-component} referencia PEÇA



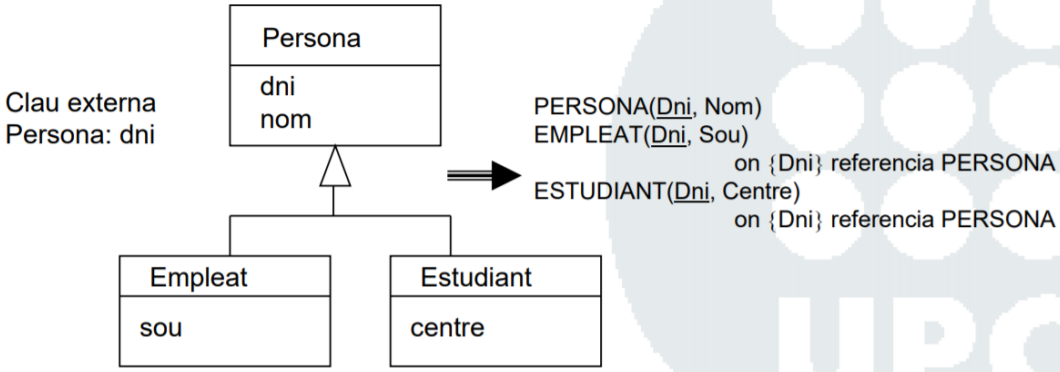
PERSONA(Codi-per, ...)
DATA(Dat)
CASAMENT(**Codi-espòs**,Codi-esposa,**Dat**)
o bé
CASAMENT(Codi-espòs,**Codi-esposa**,**Dat**)
on {Codi-espòs} referencia PERSONA,
{Codi-esposa} referencia PERSONA
i {Dat} referencia DATA

Classes associatives:



HORARI(Dia-set,Hora)
AULA(Codi-aula, ...)
GRUP(Gr, ...)
PROFESSOR(Prof,…)
CLASSE (Dia-set,Hora, Codi-aula, Gr, Durada, Prof)
o bé
CLASSE (Dia-set,Hora, Codi-aula, Gr, Durada, Prof)
on {Dia-set,Hora} referencia HORARI,
{Codi-aula} referencia AULA i
{Gr} referencia GRUP
{Prof} referencia PROFESSOR

Superclasses:



Transaccions i concurrència

-Actualització perduda: Aquesta interferència es produeix quan es perd el canvi que ha efectuat una operació d'escriptura.

Transacció T_1 (reintegrament de 10)	Transacció T_2 (reintegrament de 25)
Saldo := llegir_saldo(Compte)	
	Saldo := llegir_saldo(Compte)
escriure_saldo(Compte, Saldo − 10)	
	escriure_saldo(Compte, Saldo − 25)
COMMIT	
	COMMIT

-Lectura no confirmada: Aquesta interferència es produeix quan una transacció llegeix una dada que ha estat modificada per una altra transacció que després avorta.

Transacció T_1 (reintegrament de 10)	Transacció T_2 (reintegrament de 25)
	Saldo := llegir_saldo(Compte)
	escriure_saldo(Compte, Saldo − 25)
Saldo := llegir_saldo(Compte)	
escriure_saldo(Compte, Saldo − 10)	
COMMIT	
	ABORT

-Lectura no repetible: Aquesta interferència es produeix si una transacció llegeix dues vegades la mateixa dada i obté valors diferents, a causa d'una modificació efectuada per una altra transacció.

Transacció T_1 (lectura de saldo)	Transacció T_2 (reintegrament de 25)
Saldo := llegir_saldo(Compte)	
	Saldo := llegir_saldo(Compte)
	escriure_saldo(Compte, Saldo − 25)
	COMMIT
Saldo := llegir_saldo(Compte)	
COMMIT	

-Anàlisi inconsistent: Anàlisi (afecta a 2 gràuls).

Transacció T1 (consulta de saldos)	Transacció T2 (transferència)
saldo2:= llegir_saldo(compte2)	
	saldo1:= llegir_saldo(compte1)
	escriure_saldo(compte1, saldo1-100)
saldo1:= llegir_saldo(compte1)	
COMMIT	
	saldo2:= llegir_saldo(compte2)
	escriure_saldo(compte2, saldo2+100)
	COMMIT

-Fantasmes:

Transacció T_1 (llistat de comptes)	Transacció T_2 (creació de comptes)
Llegir tots els comptes del banc. Obtenim Compte1 i Compte2	
	crear_compte(Compte3)
	escriure_saldo(Compte3, 100)
Sumar els saldos de tots els comptes. Obtenim saldo de Compte1 + + saldo de Compte2 + 100 (El Compte3, amb saldo 100, és el fantasma)	
COMMIT	
	COMMIT

Nivells d'aïllament:

	Actualització perduda	Lectura no confirmada	Lectura no repetible i anàlisi inconsistent (tret de fantasmes)	Fantasmes
READ UNCOMMITTED	Sí	No	No	No
READ COMMITTED	Sí	Sí	No	No
REPEATABLE READ	Sí	Sí	Sí	No
SERIALIZABLE	Sí	Sí	Sí	Sí

Si una transacció intenta bloquejar un gränul que ja està bloquejat per una altra transacció, es quedarà a l'espera que es desbloquegi (tret que les dues l'hagin bloquejat per llegir).

Quan es bloqueja 1 gränul per escriure, aquest es desbloqueja en acabar la transacció.

Quan es vol llegir el gränul, depèn del nivell d'aïllament:

- Read Uncommited: Reserves X fins l'acabament de la transacció. No es fan reserves S per lectura.

- Read Commited: Reserves X fins l'acabament de la transacció. Reserves S fins després de la lectura.

- Repeatable read: Reserves X i S fins l'acabament de la transacció.

- Serialitzable: Totes les reserves fins l'acabament de les transaccions (incloent reserves d'informació de control).

Un horari no té equivalent serial si té interferències.

Mètodes d'accés

-Per posició:

Directe: Obtenir una pàgina que té un número de pàgina determinat.

Insert into Empleats values (25, 'Joan', 150, 200k)

Seqüencial: Obtenir les pàgines d'un espai seguint l'ordre definit pels seus números de pàgina.

Select * from Empleats

-Per valor:

Directe: Obtenir totes les files que contenen un valor determinat d'un atribut.

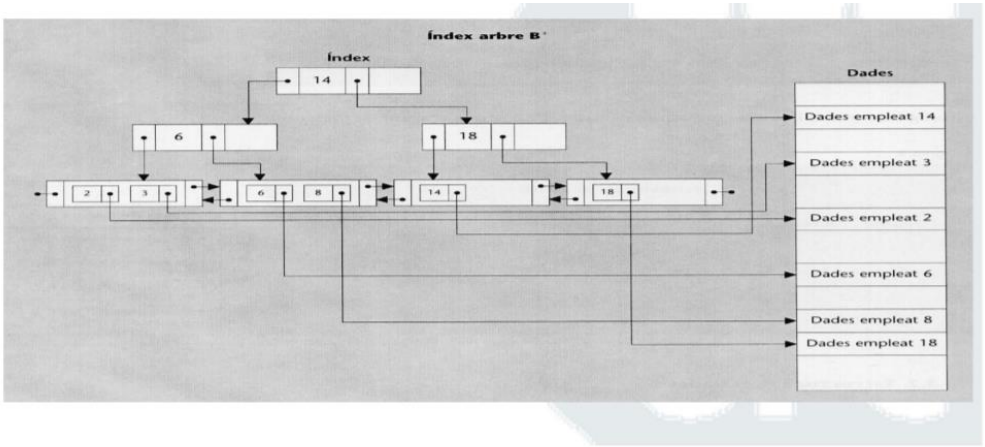
Select * from Empleats where despatx=150

Seqüencial: Obtenir totes les files per l'ordre dels valors d'un atribut.

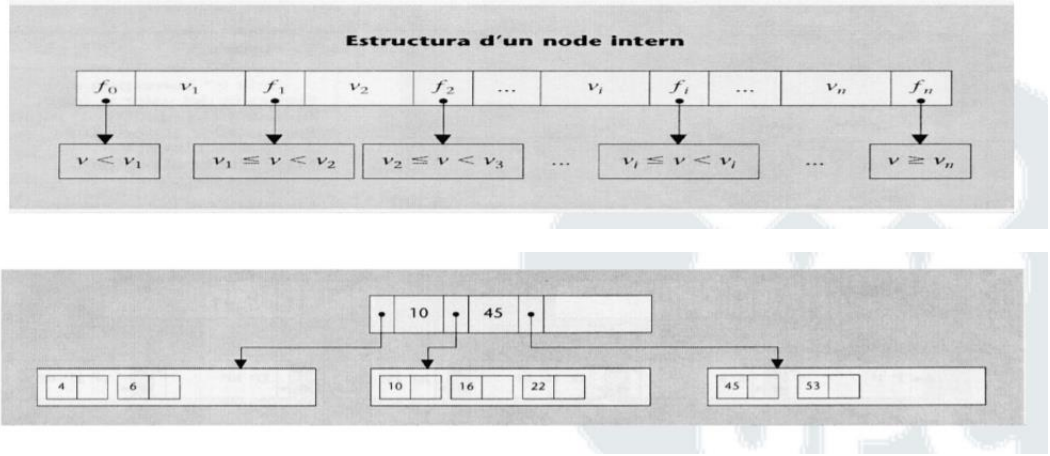
Update Empleats set sou=250k where despatx >100 and despatx<300

Arbre B+:

-Estructura



Arbre B+ d'ordre d, els nodes contenen com a màxim 2d valors.



Els nodes interns no tenen valors repetits.

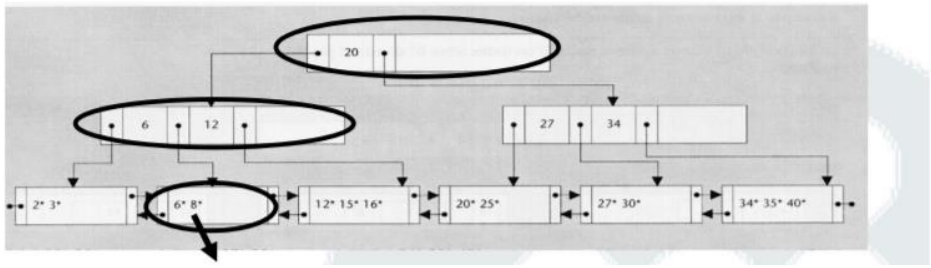
Els nodes fulla contenen: – Entrades formades per [valor, RID] que escriurem v*

– Apuntadors a la fulla anterior i següent

Cas màxim: 2d (valors+rid) + 2 apuntadors a fulla.

Lectures: El nombre de pàgines a les que cal accedir és igual al nivell de la fulla on hauria d'estar + 1 accés a la pàgina de dades (en cas d'existir el valor).

Localitzar el registre amb el valor 8



La mida habitual d'un node coincideix amb la mida de les pàgines. Cada node s'emmagatzema en una pàgina. Per tant, en un arbre de 3 nivells, calen 3 E/S per localitzar la fulla.

Costos=

-NO AGRUPAT:

C = (h + F) + |R(a>=X)|

h alçada índex B+

F nombre de nodes fulla addicionals (a més del primer node fulla) de l'índex B+ que cal recórrer

|R(a>=X)| nombre de files de la taula R que verifiquen la condició a>=X expressada a la consulta

El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

--Repetit:

C = (h + F) + |R(b=Y)|

h alçada índex B+

F nombre de nodes fulla addicionals (a més del primer node fulla) de l'índex B+ que cal recórrer

|R(b=Y)| nombre de files de la taula R que verifiquen la condició b=Y expressada a la consulta

El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

-AGRUPAT:

C = h + D

D = $\lceil |R(a>=X)|/f \rceil$

|R(a>=X)| nombre de files de la taula R que verifiquen la condició a>=X expressada a la consulta

f factor de bloqueig del fitxer de dades (nombre de registres per pàgina)

El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

--Repetit:

C = h + D

h alçada índex B+

D nombre de pàgines del fitxer de dades que cal recórrer

D = $\lceil |R(b=Y)|/f \rceil$

|R(b=Y)| nombre de files de la taula R que verifiquen la condició b=Y expressada a la consulta

f factor de bloqueig del fitxer de dades (nombre de registres per pàgina)

El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria

-NO REPETIT:

C = h + 1

h alçada de l'arbre B+ ν

El cost d'accés a l'índex es pot disminuir en 1 unitat si l'arrel està a memòria