BD Segon parcial

5. Procediments emagatzemats i disparadors

5.1 Procediments emagatzemats en PL/pgSQL

Resum de les sentències

CREATE FUNCTION: Per a crear una funció *(procediment emagatzemat)*.

END: Per a marcar el final d'una funció.

DECLARE: Per declarar i assignar valors a variables.

IF, ELSE IF, ELSE: Son les sentències condicionals de les que disposem.

LOOP, FOR, WHILE: Son les sentències interatives de les que disposem.

EXCEPTION, RAISE EXCEPTION: Sentències per a la gestió d'errors.

Glossari de les sentències

```
create function nom_funcio (nom_param_in tipus) returns tipus_ret as $$
    declare variable_ret tipus_var /*not null*/;
    declare i integer := 0;
begin
    loop
    i:=i+1;
    if condició then
        /* Codi en SQL */
        return next i;
    else
        /* Codi en SQL */
    end if;
    exit when i = 19;
    end loop;
    return variable_ret;
end;
$$language plpgsql;
```

> **nom_funcio**: Nom de la funció
>
> **nom_param_in tipus**: Nom del parametre d'entrada i el seu tipus (int, char...)
>
> **tipus_ret**:
>
> - void: No retorna res
> - int, char, varchar(x)...: Retorna el tipus especificat.
> - **setof** tipus: Retorna un conjunt de tuples de "tipus" (int, char...).
>
> **variable_ret tipus_var**: Nom i tupus de la variable declarada.
>
> ":=": Operador d'assignació de valors.
>
> **return next**: No acaba el procediment, sino que va retornant els valors.

Tuples

```
create type nom_tupla as (
    nom_var_1 varchar(10);
    nom_var_2 int;
);
/* Per accedir després a cada valor ho fem com a c++ nom.nom_var_1*/
```

Variable Found

Found es un boolea que **al inici sempre te valor false**, pero pot ser true si:

- Select: Si obté almenys una fila.
- Update, Insert, Delete: Si insereix, acutalitza o esborra almenys una fila.
- For: Al final d'un for si aquest ha fet **almenys una** iteració.

Sentències iteratives

```
for target in query end loop;
for variable in expression..expression end loop;
while expression loop /* statements */ end loop;
```

Gestió d'errors

```
begin
    /* Sentències */
exception
    when expressió then
        raise exception '%', SQLERRM;
        return;
    when expressió then
        raise exception "Missatge d'error";
        return;
    (...)
end;
```

5.2 Disparadors

Sintaxis

Els triggers amb parametres:

- **row**: Executen la funció per a cada tupla modificada/inserida/esborrada
- **statement**: Executen la funció només un cop per operació realitzada.

```
create trigger nom_trigger {before | after}
{insert | delete | update [of column], ...} on nom_taula
[for each {row | statement}]
execute procedure nom_funcio
```

Funcions amb triggers

Per tal que una funció pugui usar un trigger aquesta ha de tenir com a variable de retorn `trigger`, això comporta les seguents coses:

- Només pot tornar o bé: NULL, NEW o OLD (`return new;` , `return null;` , ...)
    - **NULL**: No retorna cap dada.
    - **NEW**: Retorna les dades modificades/inserides. Nomes per rows.
    - **OLD**: Retorna les dades abans de modificar-les/esborrar-les. Nomes per rows.
- **TG_OP** es una varaible que conté el tipus d'operació que ha fet saltar el trigger en majuscules: {INSERT, DELETE o UPDATE}.
- Si: "after+for each row" o "for each statement" normalment retornen **NULL**.

6. Programació usant SQL

// Mirar una mica com es programa i tal amb java i ja

8. Transaccions i concurrència

Definició

Transacció: Una transacció és un conjunt d'operacions de lecutra i/o actuyalització de la BD que acaba confirmant o cancel·lant els canvis que s'han dut a terme.

Tota **transacció** ha de complir amb les propietats **ACID**:

- **A**tomicitat
- **C**onisitència
- A**ï**llament
- **D**efinitivitat

Interferències

Les interferències es produeixen si les transacciojns no s'aïllen, hi han quatre tipus:

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

T1	T2
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 es llegeix un valor que eencara no ha estat confirmat (Pertant hi ha perill de ABORT).

T1	T2
	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.

T1	T2
Saldo := Llegir_saldo(Compte)	
	Saldo := Llegir_saldo(Compte)
	escriure_saldo(Compte, Saldo - 25)
	COMMIT
Saldo := Llegir_saldo(Compte)	
COMMIT	

- **Anàlisi inconsistent**: Els tres tipus d'interferencies anterior tenen lloc respecte a una única dada de la BD, pero a mes a mes es produeixen interferències respecte a la visió que dues transaccions tenen d'un conjunt de dades.

T1	T2
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**: Quan es llegeixen unes dades per una transacció, una altra les modifica i després la primera transacció torna a llegir/utilitzar la informació.

T1	T2
Llegir tots els comptes del banc: (Compte1,2)	
	crear_compte(Compte3)
	escriure_saldo(Compte3, 100)
Sumar els saldos de tots: (C1+C2+100)	
COMMIT	
	COMMIT

Serialitzabilitat

Definicions:

- La **teoria de la serialitzabilitat** defineix les condicions que han de complir per considerar que unes dades estan correctament aillades.
- El **gránul** es la unitat, definida per el SGBD, minima.
- **L'Horari o història** representa l'ordre d'accions dins de cada transacció.
    - Un **horari serial** es aquell on primer s'executa una i despres l'altre.

- Un horari compleix el **criteri de recuperabiltiat** si cap transacció que llegeixi o escrigui dades modificades per una primera transacció confirma abans que la primera.
- El **nivell de pararel·lisme** és la feina efectiva realitzada, és a dir, la feina realment útil per als usuaris.
- Tipus d'operacions:

SELECT	INSERT/UPDATE/DELETE
R(G)	RU(G) W(G)

- Exemple d'horari

#temps	T1	T2
1	R(A)	
2		RU(A)
3	R(B)	
4		W(A)
5	R(C)	
6	commit	
7		commit

Control de concurrència amb reserves

Per a controlar la concurrencia els SGBD utiltizen les reserves, abns de poder efectuar accions (lectures o escriptures) sobre un gràNul. Es demanen amb:

- **LOCK(G,m)**: on G es el granul i m es el mode, hi distingim dos:
  - Modalitat compartida (S): permet fer lectures.
  - Modalitat exclusiva (X): permet fer lectures i escriptures.
- **UNLOCK(G)**: Allibera la reserva del granul G.
- PR2F+reserves fins l'acabament de les transaccions: Sempre sera serialitzable

Reserves i nivells d'aïllament

- **Read Uncommitted**:
  - Reserves X fins l'acabament de la transacció.
  - No es fan reserves S per lectura.
- **Read Commited**
  - Reserves X dins l'acabament de la transacció.
  - Reserves S dins després de la laectura.
- **Repetable Read**
  - Reserves X i S fins l'acabament de la transacció.
- **Serialitzable**
  - Totes les reserves fins l'acabament de les transaccions

9. Emmagatzematge

Nivell Físic

- **Fitxer**: És la unitat global a partir de la qual el sistema operatiu gestione les dades.
- **Extensió**: Es la unitat d'aquisició d'espai per a cada fitxer. És un mutilple de pàgina.
- **Pàgina**: És el component físic més petit. Conté les dades del nivell lògic.
  - **Fila**: Conté una capçalera amb informació sobre la longitud de la fila i l'identificador de la taula a la qual pertany, i després tots els camps que la formen.
  - **Camp**: Conté una capçalera amb informació sobre si es null o no i la longitud d'aquest camp, **si es de longitud fixa i not null no té capçalera**.

Mètodes d'accés

- **Directe**: S'accedeix directament a una posició en concret. Cost (1)
- **Seqüencial**: S'accedeixen a totes les posicions fins arribar a la desitjada. Cost (N)

$$Cost = \frac{\#tuples}{\#tuples/pagina}$$

Arbres B+

$$Apuntadors\ Nodes\ Interns(ANI) = (d \cdot ocupació \cdot 2) + 1$$

- Si el valor buscat es **clau primaria**:

$$Cost = \lceil \log_{ANI} \#tuples \rceil + 1$$

- Si el valor buscat **NO es calu primaria**:
  - Amb Index agrupat

$$Cost = \lceil \log_{ANI} \#tuples \rceil + \frac{\#tuples}{fiels/pagina}$$

  - Sense Index agrupat

$$Cost = \lceil \log_{ANI} \#tuples \rceil + \lceil \frac{\#pag\ a\ visitar}{\%ocupació} \rceil - 1 + \#tuples$$

10. Criteris de qualitat

SQL

- Només s'usen les clàusules explicades a les transparències
- **Si i només si** una consulta pot donar un **resultat amb repeteicions** s'usa **distinct**
- La sentència SQL **no te parts redundants**, com consultes innecessàries, etc...
- No usar subconsultes si no cal.
- No usar count si no fa falta comptar alguna cosa. (osigui en lloc de not exists).

Procediments / Disparadors

- S'apliquen tots els criteris de SQL
- Escollir el tipus de disparadaor correctament (before, after, row, statement) per evitar fer solucions poc eficients o redundants.

Programació amb SQL: JDBC

- S'apliquen tots els criteris de SQL
- **Utilitzar adequadament el Statement i el PreparedStatement**.
  - PreparedStatement s'usa només quan hi han bucles que l'executin varios cops.