

CFGS Desenvolupament d'Aplicacions Multiplataforma
Mòdul 2 – Gestió de bases de dades
UF3 – Llenguatges SQL: DCL i extensió procedimental
Unitat 7 – Programació de bases de dades
TEA1
(Curs 2018-19 / 2on semestre)

Presentació i resultats d'aprenentatge

Aquest exercici d'avaluació continuada (EAC) es correspon amb els continguts treballats a la unitat **"U7 Programació de bases de dades"** de la **"UF3 – Llenguatges SQL: DCL i extensió procedimental"**

Els **resultats d'aprenentatge** que es plantegen són:


- Identifica les eines disponibles en el sistema gestor de bases de dades per a editar guions.
- Defineix guions per automatitzar tasques que gestionen la base de dades.
- Identifica els tipus de dades, identificadors, variables i constants.
- Utilitza estructures de control de flux i llibreries de funcions.
- Desenvolupa procediments i funcions d'usuari
- Gestiona els possibles errors dels procediments i funcions i controla les transaccions.
- Utilitza cursors per manipular les dades d'una base de dades.
- Utilitza les funcions incorporades en el sistema gestor de bases de dades.
- Desenvolupa disparadors.

Criteris d'avaluació

La puntuació màxima assolible a cada activitat s'indica a l'enunciat respectiu.

Els criteris que es tindran en compte per avaluar el treball de l'alumnat són els següents:

- La correcció i la completeness de les respostes
- La coherència i la bona estructuració de les respostes, així com la seva pulcritud

	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 1 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

Forma i data de lliurament

Un cop finalitzat l'exercici d'avaluació continuada s'ha d'enviar el document a la bústia de **"Lliurament EAC7"** de l'aula, dins del termini establert. Tingueu en compte que el sistema no permetrà fer lliuraments després de la data i hora indicades.

El nom del fitxer tindrà el següent format: **"DA2_M02B2_TEA1_Cognom1_InicialDelCognom2.odt"**. Per tant, heu de lliurar l'arxiu en el format original: **document de text OpenOffice (.odt)**.

Els cognoms s'escriuran sense accents. Per exemple, l'estudiant Joan García Santos posaria el següent nom al seu fitxer de l'EAC6: **"DA2_M02B2_TEA1_Garcia_S.odt"**.
Substituïu també **"Nom i cognoms"** de la capçalera per les vostres dades personals.

Podeu fer servir captures de pantalla i qualsevol altre recurs per documentar el vostre treball i aclarir les vostres proves, però és imprescindible que afegiu la **transcripció literal de les vostres respostes**, sigui codi PostgreSQL o llenguatge natural.


El termini de lliurament finalitzarà a les **23:55 h** del dia **12/04/2019**. La proposta de solució de l'EAC és publicarà el dia 23/04/2019 i les qualificacions el dia 02/05/2019.

Enunciat

Per a realitzar aquest TEA1 necessiteu tenir correctament instal·lats PostgreSQL i l'eina pgAdmin III.

Tornem a fer servir la base de dades **enjoyElVostreNom**.

Si algú no ha fet l'EAC 6, pot consultar com generar la base de dades i una explicació sobre la mateixa a als enunciats de l'EAC5 i EAC6, o a la publicació de la solució dels respectius EAC's.

	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 2 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

Cursors i control d'errors

Les següents activitats les fareu a l'espai «públic» de la base de dades.

Activitat 1 – treball per parelles

L'objectiu d'aquesta activitat és implementar, mitjançant la sobrecàrrega de funcions, una funció que, depenent del tipus de paràmetre que li passem, ens retorni un tipus de dada o una altra. Aquesta funció treballareu els cursors en operacions de selecció simples:

- La funció s'anomenarà **dades_treballador**.
- En cas que se li passi com a paràmetre el **sou** d'un treballador, retornarà els cognoms d'aquells treballadors amb un sou més alt que el seu. **La implementació de la funció es farà emprant cursors.**
- En cas que se li passi com a paràmetre un **nom**, retornarà els cognoms d'aquells treballadors que coincideixin amb aquell nom. **La implementació de la funció es farà emprant cursors.**


Desenvolupament de l'activitat:

- Aquesta activitat s'ha de fer per parelles. Organitzeu-vos a través del fòrum de l'aula o a través del correu intern de l'aula. En cas que algú es quedi sense parella, excepcionalment es podrà afegir a un grup de tres.
- Ompliu la següent plantilla:

Solució: (1,5 punts)

- Nom i cognoms dels components del grup:
- Anoteu aquí:
 - com heu interpretat l'enunciat
 - què heu entès que heu de fer
 - què heu pactat
 - com ho heu dut a terme (quina tasca heu fet cadascú)
- Copieu aquí a sota el codi de la funció que tracta la primera casuística, juntament amb una captura de pantalla del resultat de provar la vostra funció, passant-li com a paràmetre el sou d'un treballador de la vostra base de dades:
- Copieu aquí a sota el codi de la funció que tracta la segona casuística, juntament amb una captura de pantalla del resultat de provar la vostra funció, passant-li com a paràmetre un nom d'un treballador de la vostra base de dades:
- Com us heu intercanviat les funcions desenvolupades cadascú? Heu entès tots dos el funcionament de les dues funcions, encara que una no l'hagueu desenvolupat vosaltres?

Solució: (1,5 punts)

	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 3 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

Una manera de fer-ho serà utilitzant la sobrecàrrega de funcions. En aquest cas, dues funcions amb el mateix nom, però a una li passem com a paràmetre un número (real), i a l'altra una cadena de caràcters (varchar). Quan cridem a la funció passant-li un número executarà i buscarà la primera funció, i quan cridem a la funció passant-li una cadena de caràcters executarà i buscarà la segona funció.

```
CREATE OR REPLACE FUNCTION dades_treballador (ent_sou real)  
RETURNS setof varchar AS
```

```
$BODY$  
DECLARE
```

```
cognoms_treb comercial.treballadors.cognoms%type;  
cursor1 CURSOR FOR SELECT cognoms  
FROM comercial.treballadors WHERE sou > ent_sou;
```

```
BEGIN
```

```
OPEN cursor1;  
LOOP  
FETCH cursor1 INTO cognoms_treb;  
EXIT WHEN NOT FOUND;  
RETURN NEXT cognoms_treb;  
END LOOP;  
CLOSE cursor1;
```

```
RETURN;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION dades_treballador (identif varchar(30))  
RETURNS setof varchar AS
```

```
$BODY$  
DECLARE
```

```
cognoms_treb comercial.treballadors.cognoms%type;  
cursor2 CURSOR FOR SELECT cognoms  
FROM comercial.treballadors WHERE nom LIKE identif;
```

```
BEGIN
```

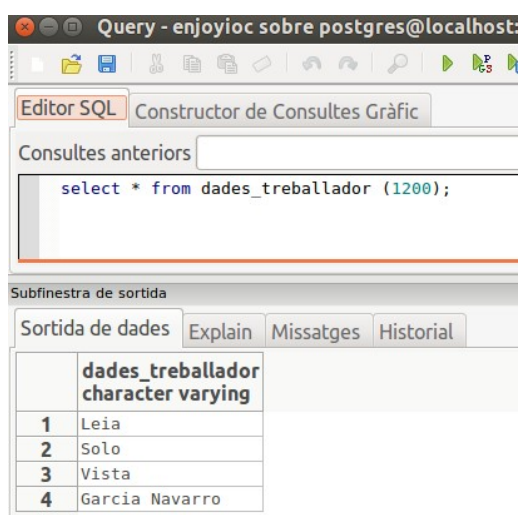
```
OPEN cursor2;  
LOOP  
FETCH cursor2 INTO cognoms_treb;  
EXIT WHEN NOT FOUND;  
RETURN NEXT cognoms_treb;  
END LOOP;  
CLOSE cursor2;
```

```
RETURN;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

Taula treballadors:

	dni [PK] character varying(10)	nom character varying(30)	cognoms character varying(60)	carrec character varying(25)	email character varying(20)	sou real
1	12312312A	Princesa	Leia	gerent	leia@gmail.com	2200
2	2222222B	Han	Solo	comercial	han@gmail.com	1500
3	42222222X	Santi	Fede Nas	comercial	santi@gmail.com	1180
4	45645666X	Paula	Garcia Navarro	gerent	paula@gmail.com	1289
5	789789D	Prova	Vista	guia	prova2@gmail.com	1350
*						

Provem d'introduir com a paràmetre un sou:



Provem a introduir com a paràmetre un nom:



Activitat 2

L'objectiu d'aquesta activitat és implementar una funció emprant cursors en la iteració d'una operació de selecció de més d'un camp amb múltiples resultats mitjançant un tipus compost.

- Implementeu una funció anomenada **sous_treballadors**. Se li passarà com a paràmetre el sou d'un treballador, i retornarà un tipus compost anomenat **tinfo** que contindrà els camps corresponents al nom, cognoms, sou i email d'aquells treballadors amb un sou més alt o igual que el seu. **La implementació de la funció es farà emprant cursors.**

El tipus de retorn serà podria ser:


```
CREATE TYPE tinfo AS (  
  nom_mon character varying(30),  
  cog_mon character varying(60),  
  sou_mon real,  
  email character varying(20));
```

- Copieu aquí a sota el codi de la vostra funció juntament amb una captura de pantalla del resultat de provar la vostra funció, passant-li com a paràmetres el sou d'un treballador de la vostra base de dades.

Solució: (1,5 punts)

La implementació de la funció corresponent ha de permetre treballar amb el tipus de dades definit, i per això la consulta que tractarà el cursor tindrà els mateixos atributs.

```
CREATE TYPE tinfo AS (  
  nom_mon character varying(30),  
  cog_mon character varying(60),  
  sou_mon real,  
  email character varying(20));  
  
CREATE OR REPLACE FUNCTION sous_treballadors (entrada real)  
RETURNS setof tinfo AS  
  
$BODY$  
DECLARE  
  
info_treb tinfo;  
  
cursor3 CURSOR FOR SELECT nom, cognoms, sou, email  
  FROM comercial.treballadors WHERE sou >= entrada;  
  
BEGIN  
  
OPEN cursor3;  
LOOP  
FETCH cursor3 INTO info_treb;  
EXIT WHEN NOT FOUND;  
RETURN NEXT info_treb;  
END LOOP;  
CLOSE cursor3;  
  
RETURN;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 6 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

Query - enjoyioc sobre postgres@localhost:5432 *

Editor SQL Constructor de Consultes Gràfic

Consultes anteriors

```
select * from sous_treballadors (1200);
```

Subfinestra de sortida

Sortida de dades Explain Missatges Historial

	nom_mon character varying(30)	cog_mon character varying(60)	sou_mon real	email character varying(20)
1	Princesa	Leia	2200	leia@gmail.com
2	Han	Solo	1500	han@gmail.com
3	Prova	Vista	1350	prova2@gmail.com
4	Paula	Garcia Navarro	1289	paula@gmail.com

Activitat 3

L'objectiu d'aquesta activitat és implementar tipus compostos segons les necessitats del tractament dels cursors.

- Desenvolpeu una funció anomenada **usuaris_inscripcions()** que permeti obtenir les dades següents: el nom i cognoms d'aquelles usuaris que s'hagin inscrit a més de dos activitats. Cal implementar el tipus de dada compost amb el qual es donarà suport al tractament del cursor.
- Copieu aquí a sota el codi de la vostra funció juntament amb una captura de pantalla del resultat de provar la vostra funció.

Solució: (1 punts)

Primerament cal implementar el tipus de dades amb què treballarem:


```
CREATE TYPE tusuaris AS (
  nom varchar(30),
  cognoms varchar(60));
```

La funció implementada emprant aquest tipus de dades tindrà la codificació següent:

```
CREATE OR REPLACE FUNCTION usuaris_inscripcions ()
RETURNS setof tusuaris AS
```

```
$BODY$
DECLARE
```

```
usuari tusuaris;
cursor5 CURSOR FOR SELECT nom, cognoms FROM comercial.usuaris
WHERE id_usuari IN (SELECT id_usuari FROM comercial.inscripcio
                    group by id_usuari
                    having count (*) >2);
```

	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 7 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

BEGIN

```

OPEN cursor5;
LOOP
FETCH cursor5 INTO usuari;
EXIT WHEN NOT FOUND;
RETURN NEXT usuari;
END LOOP;
CLOSE cursor5;

```

```

RETURN;
END;
$BODY$
LANGUAGE plpgsql;

```

Taula inscripcions:

	id_inscripcio [PK] serial	id_usuari integer	id_act integer	data_inici date	data_fi date
1	1	1	1	2019-02-23	2019-02-24
2	2	2	2	2019-02-26	2019-02-26
3	3	3	3	2019-06-15	2019-06-21
4	5	1	2	2019-06-21	2019-06-23
5	6	1	3	2019-05-15	2019-05-17
*					

Taula usuaris:

id_usuari [PK] serial	dni character varying(10)	nom character varying(30)	cognoms character varying(60)	adreca character varying(30)	email character varying(20)
1	123321A	Maria	Garcia Gomez	C/Nau 9, Tarragona	mgomez@gmail.com
2	321123B	Jan	Herrera Fred	C/Mig 3, Lleida	jherrera@gmail.com
3	44444S	Pol	Jota Max	C/del mig	pol@gmail.com
4	66666677A	Carol	Khern Fysser	C/del mig	carol@gmail.com

L'usuari amb identificador '1' compleix la condició de l'enunciat.

nom character varying(30)	cognoms character varying(60)
1 Maria	Garcia Gomez

Activitat 4


Interpreteu i adapteu la següent funció anomenada **primers_usuaris()**, per tal que permeti obtenir un conjunt de dades corresponent als 3 primers usuaris que s'hagin donat d'alta a la base de dades (donarem per suposat que l'antiguitat bé relacionada amb l'identificador de l'usuari, a mesura que es van introduint usuaris a la base de dades l'identificador va creixent, per tant, els més antics seran aquells amb un identificador més baix). Traurem el seu identificador, nom, cognoms i email (cal que definiu prèviament el tipus t_usuari).

```
CREATE OR REPLACE FUNCTION xxxxxxxxxxxx()  
    RETURNS setof t_usuari AS  
$BODY$  
DECLARE  
    usuari t_usuari;  
    numero integer;  
    curs1 CURSOR FOR SELECT id_usuari, nom, cognoms, email  
        FROM xxxxxxxx ORDER BY id_usuari DESC;  
BEGIN  
    OPEN curs1;  
    FOR numero IN 1..X LOOP  
        FETCH curs1 INTO usuari;  
        EXIT WHEN NOT FOUND;  
        RETURN next usuari;  
    end LOOP;  
    close curs1;  
END;  
$BODY$  
LANGUAGE plpgsql;
```

- Copieu aquí a sota el codi de la funció juntament amb una captura de pantalla del resultat de provar la vostra funció.

Solució: (0,5 punts)

```
CREATE TYPE t_usuari AS (  
id int4,  
nom varchar(30),  
cognoms varchar(60),  
email varchar(20));  
  
CREATE OR REPLACE FUNCTION primers_usuaris()  
    RETURNS setof t_usuari AS  
$BODY$  
DECLARE  
    usuari t_usuari;  
    numero integer;  
    curs1 CURSOR FOR SELECT id_usuari, nom, cognoms, email  
        FROM comercial.usuaris ORDER BY id_usuari ASC;  
BEGIN  
    OPEN curs1;  
    FOR numero IN 1..3 LOOP  
        FETCH curs1 INTO usuari;  
        EXIT WHEN NOT FOUND;  
        RETURN next usuari;  
    end LOOP;
```

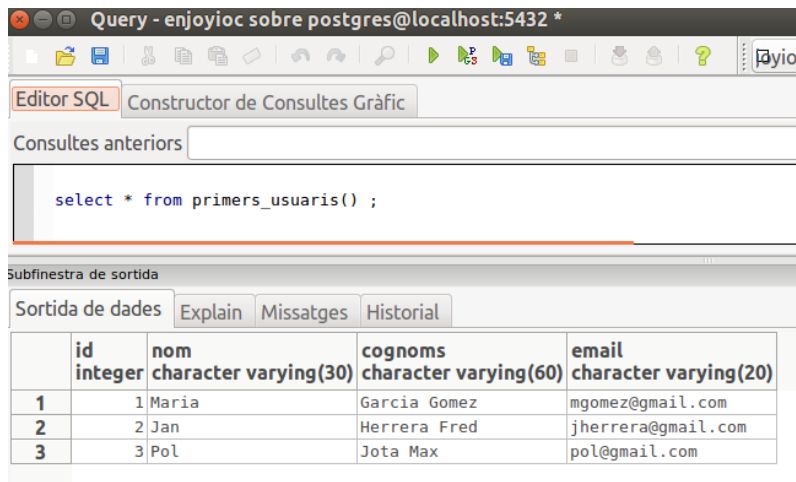
	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 9 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

```

        close curs1;
    END;
$BODY$
    LANGUAGE plpgsql;

```

Prova:



Ara voleu extreure la mateixa informació (identificador, nom, cognoms i email dels usuaris més antics) però sense limitar-ho a 3. El número límit s'introduirà com a paràmetre. Voleu que quan es cridi la funció **primers_usuaris** sense paràmetres, actuï com s'ha definit anteriorment, limitat a 3, i si es crida amb un paràmetre (en aquest cas un número) actuï respectant el límit introduït per l'usuari. Això és possible? Com?

- Implementa-ho i justifica la resposta.
- Copieu aquí a sota el codi de la funció juntament amb una captura de pantalla del resultat de provar la vostra funció.

Solució: (1 punts)

Si que és possible, amb la sobrecàrrega de funcions.

Implementarem una funció sobrecarregada que faci el mateix que la funció anterior `usuaris_antics()`, però que el nombre d'usuaris a mostrar s'introdueixi com un paràmetre.

Definir una nova funció, amb el mateix nom, i amb un paràmetre d'entrada que serà de tipus numèric.

```

CREATE OR REPLACE FUNCTION primers_usuaris(num integer)
    RETURNS setof t_usuari AS
$BODY$
DECLARE
    usuari t_usuari;
    numero integer;
    curs1 CURSOR FOR SELECT id_usuari, nom, cognoms, email
        FROM comercial.usuaris ORDER BY id_usuari ASC;
BEGIN
    OPEN curs1;
    FOR numero IN 1..num LOOP

```

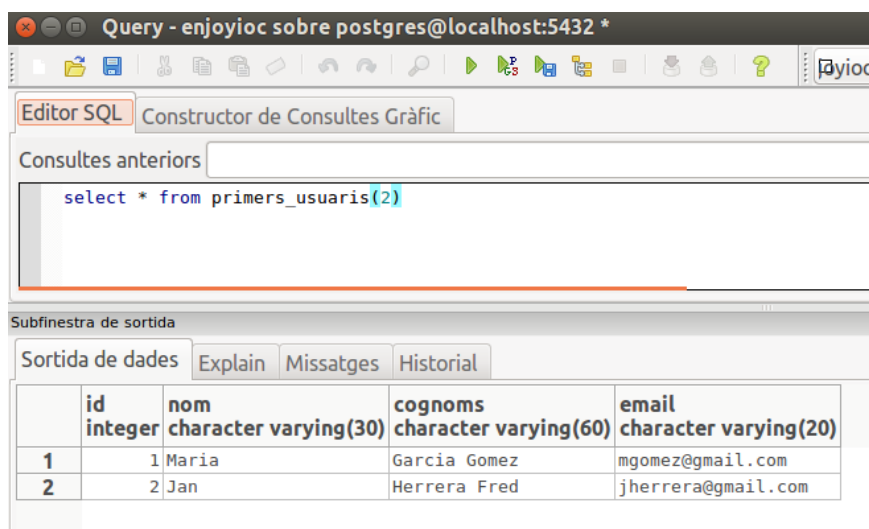
```

        FETCH curs1 INTO usuari;
        EXIT WHEN NOT FOUND;
        RETURN next usuari;
    end LOOP;
    close curs1;

END;
$BODY$
LANGUAGE plpgsql;

```

Prova: li diem que només volem treure dos.



Activitat 5

Modifiqueu la funció **inscripcions_activitats** (desenvolupada en l'EAC6), de manera que controli els casos següents, indicant-ho en el missatge que es mostra a l'usuari:

- L'activitat que ens han passat com a paràmetre no existeix.
- No hi ha cap inscripció d'aquella activitat.

Copieu aquí a sota el codi de la funció juntament amb les captures de pantalla del resultat de provar la vostra funció. Proveu com a mínim les dues casuístiques anomenades en el paràgraf anterior.

Solució: (1 punts)

```

CREATE OR REPLACE FUNCTION inscripcions_activitats
(identif comercial.activitats.id_activitat%type) RETURNS varchar(70) AS $$
DECLARE
    missatge varchar(70);
    quantitat integer;
BEGIN

```

```

IF (identif NOT IN (SELECT id_activitat FROM comercial.activitats))
THEN missatge='Aquesta activitat no existeix a la nostra base de dades';
ELSE

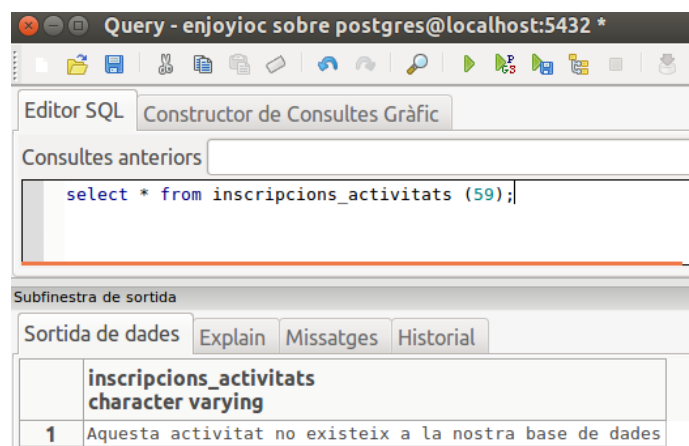
SELECT COUNT(*) INTO quantitat
FROM comercial.inscripcio
WHERE id_act = identif;
IF (quantitat = 0) THEN missatge= 'No hi ha cap inscripció amb aquesta
activitat';
ELSIF (quantitat = 1) THEN missatge = 'D''aquesta activitat només hi ha una
inscripció';
ELSIF ((quantitat >= 1 ) AND (quantitat <= 4)) THEN missatge = 'D''aquesta
activitat hi ha menys de 5 inscripcions';
ELSIF (quantitat >4) THEN missatge = 'D''aquesta activitat hi ha 5 o més
inscripcions';
END IF;
END IF;
RETURN missatge;
END;
$$ LANGUAGE plpgsql;

```

Taula activitats:

Edita Dades -IOC (localhost:5432) - enjoyioc - comercial.activitats			
00 registre			
id_activitat [PK] serial	nom_activitat character varying(20)	descripcio text	preu real
1	Visita al Zoo	Visita al parc de la ciutadella i	45
2	Ruta barri Born	Passeig amb gui pel barri del Born	35
3	Ruta gastronòmica	Passeig i tapes pel barri del Born	80

Provem la primera casuística, que l'activitat no existeixi:



Taula inscripcions:

Edita Dades -IOC (localhost:5432) - enjoyioc - comercial.inscripcio					
	id_inscripcio [PK] serial	id_usuari integer	id_act integer	data_inici date	data_fi date
1	1	1	1	2019-02-23	2019-02-24
2	2	2	2	2019-02-26	2019-02-26
3	3	3	3	2019-06-15	2019-06-21
4	5	1	2	2019-06-21	2019-06-23
5	6	1	3	2019-05-15	2019-05-17
*					

Entrem una nova activitat de la que no farem cap inscripció, per tal de poder provar la segona casuística que se'ns demana:

Edita Dades -IOC (localhost:5432) - enjoyioc - comercial.activitats				
	id_activitat [PK] serial	nom_activitat character varying(20)	descripcio text	preu real
1	1	Visita al Zoo	Visita al parc de la ciutadella i	45
2	2	Ruta barri Born	Passeig amb gui pel barri del Born	35
3	3	Ruta gastronòmica	Passeig i tapes pel barri del Born	80
4	5	Sagrada Família	Visita guiada a la Sagrada Família	35
*				

Ho comprovem:

Query - enjoyioc sobre postgres@localhost:5432 *	
Editor SQL	Constructor de Consultes Gràfic
Consultes anteriors	
<pre>select * from inscripcions_activitats (s);</pre>	
Subfinestra de sortida	
Sortida de dades	Explain Missatges Historial
	inscripcions_activitats character varying
1	No hi ha cap inscripció amb aquesta activitat


Activitat 6

Control d'errors mitjançant la gestió d'excepcions: captureu qualsevol altre tipus d'errors dins de la funció ***inscripcions_activitats*** i reporteu-los al nivell superior mitjançant excepcions.

Solució: (0,5 punts)

Afegir al final de la funció, just sota el RETURN, les següent línies, per tal de reportar qualsevol altre tipus d'excepcions:

```
...  
RETURN missatge;  
EXCEPTION  
WHEN raise_exception THEN RAISE EXCEPTION' %: %',SQLSTATE, SQLERRM;  
WHEN others THEN RAISE EXCEPTION' P0001: Error intern';  
END;  
$$LANGUAGE plpgsql;
```

	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 14 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

Disparadors

Activitat 7 (1 punt)

En aquesta activitat se us donen un seguit de preguntes que haureu de definir sobre el material que teniu als apunts. No és necessari ni interessant que copieu tal com ho defineix el material didàctic sinó que en doneu una definició amb les vostres paraules i també sigueu capaços de donar exemples sobre el tema.

- Què són els disparadors (*triggers*)?

Els disparadors (*triggers*) són procediments emmagatzemats a la base de dades que s'executen automàticament quan es du a terme una operació INSERT, DELETE o UPDATE sobre alguna taula en concret i permeten als usuaris executar funcions introduïdes per altres usuaris sense conèixer-ne la implementació.

No es poden dur a terme en una operació SELECT.

Aquesta acció podria servir per fer una comprovació de consistència en un conjunt de valors per ser inserits, en el format de les dades abans de ser introduïdes, en una modificació en una taula o en una modificació d'un conjunt de files.

- Es poden dur a terme en una operació SELECT?

No, en una operació select no es poden dur a terme. Només es du a terme quan s'actualitza algun valor a la BBDD (insert, delete o update).

- Quan una funció PL/PgSQL és cridada per un disparador es creen diverses variables especials. Identifiqueu-ne 4 d'elles i descriu-les.

NEW	RECORD	Variable que conté la nova fila de la base de dades per a operacions INSERT / UPDATE en disparadors a escala de fila (<i>row-level</i>). Aquesta variable és NULL en els disparadors a escala d'instrucció (<i>statement-level</i>) i per a les operacions DELETE .
OLD	RECORD	Variable que conté el valor antic de la fila per a operacions UPDATE/DELETE en disparadors a escala de fila (<i>row-level</i>). Aquesta variable és NULL en els disparadors a escala d'instrucció (<i>statement-level</i>) i per a les operacions INSERT .
TG_NAME	name	Variable que conté el nom del disparador que en realitat s'ha disparat
TG_WHEN	text	Una cadena BEFORE o AFTER , depenent de la definició del disparador.
TG_LEVEL	text	Una cadena ROW o STATEMENT , depenent de la definició del disparador.
TG_OP	text	Una cadena INSERT , UPDATE , DELETE o TRUNCATE , que diu quina operació ha llençat el disparador.
TG_RELID	oid	L'OID (identificador d'objecte) de la taula que ha causat la invocació.
TG_RELNAME	name	El nom de la taula que va causar la invocació del disparador. Això ara està en desús, i podria desaparèixer en futures versions. Millor emprar TG_TABLE_NAME .
TG_TABLE_NAME	name	El nom de la taula que va causar la invocació del disparador
TG_TABLE_SCHEMA	name	El nom de l'esquema de la taula que va causar la invocació del disparador
TG_NARGS	integer	El nombre d'arguments donats en el procediment d'activació en la declaració CREATE TRIGGER

Activitat 8

L'objectiu d'aquesta activitat és implementar un disparador d'auditoria que enregistri en una taula dissenyada amb aquest objectiu determinats valors que són necessaris per auditar operacions d'inserció i d'eliminació en una taula.

Se us demana crear un disparador d'auditoria, anomenat **audit_treballadors**, que permeti inserir els valors corresponents al dni d'un treballador, usuari que fa la modificació, hora de la modificació i l'acció realitzada, inserció o eliminació, en la taula **treballadors_log**, creada amb aquest objectiu, cada cop que s'insereixi o s'elimini un registre de la taula **treballadors**. La funció encarregada de fer aquestes accions s'anomenarà **modific_treballador_log()**

Solució: (1,5 punts)

Primer es crea la taula amb els requeriments d'auditoria esmentats:

```
CREATE TABLE treballadors_log (
  identif varchar(10),
  usuari varchar(15),
  hora_modif timestamp,
  accio char(15));
```


Seguint els requeriments esmentats a l'enunciat, el codi de la funció **modific_treballador_log()** associada al disparador seria el següent:

```
CREATE OR REPLACE FUNCTION modific_treballador_log()
RETURNS TRIGGER AS $$
BEGIN

IF TG_OP = 'INSERT' THEN
    INSERT INTO treballadors_log VALUES
    (NEW.dni, current_user, current_date, TG_OP);
END IF;

IF TG_OP = 'DELETE' THEN
    INSERT INTO treballadors_log VALUES
    (OLD.dni, current_user, current_date, TG_OP);
END IF;

RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

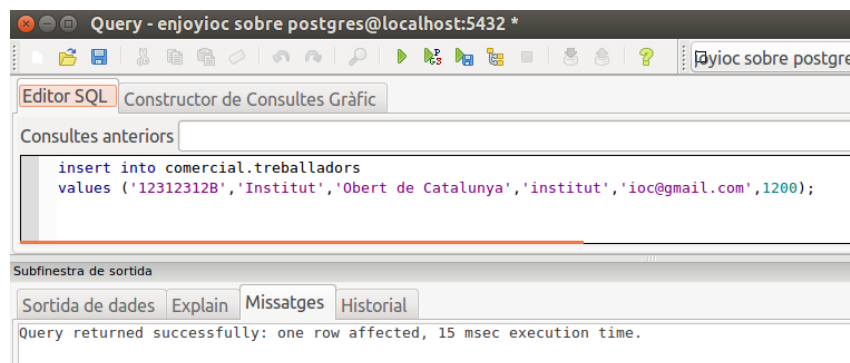
Es crea el disparador **audit_treballadors** perquè es dispari després de les accions d'inserció i eliminació:


```
CREATE TRIGGER audit_treballadors AFTER
INSERT OR DELETE ON comercial.treballadors
FOR EACH ROW EXECUTE PROCEDURE modific_treballador_log();
```

Per comprovar que funciona correctament feu una acció d'inserció i de supressió:

- Inserir un treballador que tingui els vostres cognoms.
- Suprimiu-lo després.
- Consulteu la taula **treballadors_log** per veure si s'han inserit els registres corresponents a les accions efectuades. Adjunteu les captures de pantalla.

Solució: (0,5 punts)



	Codi: I71	Exercici d'avaluació contínua 7	Pàgina 17 de 18
	Versió: 02	DA2_M02B2_TEA1_Enunciat_1819S2	Lliurament: 12/04/2019

