DOSSIER DE PRACTIQUES DE BASES DE DADES

(VERSIÓ PER ALUMNES)

Sergio Sayago
Departament de Matemàtiques i Informàtica
Universitat de Barcelona
Curs acadèmic 2017/18

1. Introducció

Aquest document és el **dossier de pràctiques** de l'assignatura Bases de Dades del Grau en Enginyeria en Informàtica de la Universitat de Barcelona corresponent al curs acadèmic 2017/18.

1.1 Continguts

El dossier consisteix en 9 pràctiques. **Les pràctiques estan dividides en dos blocs**.

- **El bloc A està molt lligat als primers 4 temes de teoria**. Aquest bloc té una pràctica per cada tema. La pràctica 1 és sobre el TEMA 1 (Bases de dades relacionals). La pràctica 2 és sobre el TEMA 2 (Model relacional). La pràctica 3 és sobre el TEMA 3 (Àlgebra relacional), i la pràctica 4 és sobre el TEMA 4 (Entitat-Relació). L'objectiu d'aquestes pràctiques és el de reforçar la teoria. Aquestes pràctiques consisteixen en una combinació de preguntes obertes i exercicis.
- **El bloc B és sobre SQL** (*Structured Query Language*). SQL no forma part dels continguts teòrics de l'assignatura perquè he preferit tractar-lo a les pràctiques. Aquest bloc ens ocuparà tota la segona meitat de la part pràctica de l'assignatura. Aquest bloc consisteix en 5 pràctiques. L'objectiu fonamental d'aquestes pràctiques és el d'oferir-vos una introducció pràctica guiada a SQL, de tal manera que acabeu l'assignatura sent capaços de crear i gestionar bases de dades relacionals, i realitzar consultes de diferents nivells de complexitat. Utilitzareu una base de dades que vosaltres mateixos creareu i manipulareu. Utilitzareu **PostgreSQL 10 a Windows** com a SGBD.

1.2 Objectius

Les pràctiques tenen dos objectius principals:

- Les pràctiques estan dissenyades per ajudar-vos a desenvolupar habilitats importants per dissenyar i treballar amb bases de dades relacionals, des del disseny del diagrama Entitat Relació i el seu corresponent pas a taules, passant per la creació de la base de dades, inserció, eliminació i consultes de dades, combinant exercicis amb paper i altres amb ordinador.
- Les pràctiques també intenten reforçar una habilitat professional molt important: aprendre a aprendre. Això ho fem perquè durant la vostra vida professional, haureu d'aprendre moltes coses, i us haureu d'enfrontar a problemes i reptes vosaltres mateixos; per tant, heu de ser capaços d'aprendre d'una manera autònoma i raonadament independent.

1.3 Dinàmica

- Les pràctiques les podeu realitzar individualment o en grup.
- Les pràctiques no les lliurareu durant el semestre; sou estudiants de tercer i heu de ser capaços d'aprendre a auto-gestionar el vostre temps i feina a fer. D'altra banda, us recomanem que les

feu seguint el calendari, perquè us ajudarà a preparar els parcials (2) de la part pràctica de l'assignatura.

- Les correccions de les pràctiques es faran durant les sessions de pràctiques o mitjançant tutories.

BLOC A) Pràctica 1: Bases de dades relacionals

Objectius

- Aprofundir en els conceptes de bases de dades (relacionals) del TEMA 1 de teoria.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Exercicis / Preguntes

Exercici 1

A teoria hem parlat dels principals avantatges dels SGBD. Segurament, també tinguin **inconvenients**. Quins penses que són, o que poden ser, els seus **principals inconvenients**? Raona la teva resposta.

Exercici 2

Explica, en les teves paraules, la principal diferència entre la **independència lògica i la física** de dades.

Exercici 3

A teoria hem comentat que les bases de dades modelen una part del món que volem tractar. De la següent llista d'elements relacionats amb les bases de dades, indica els que juguen – en cas que hi hagi més d'un - un paper important en aquesta **representació de la informació del món real**. Explica breument la teva resposta.

a. El llenguatge de definició de dades
b. El llenguatge de manipulació de dades
c. El nivell físic
d. El model de dades

Exercici 4

Una base de dades és un dipòsit compartit de dades relacionades lògicament i una descripció de les dades (metadades). Què són les **metadades**? Comenta un parell d'exemples.

Exercici 5

Comenta breument **dos limitacions dels sistemes basat en fitxers** envers els sistemes de bases de dades.

Exercici 6

Explica en les teves paraules què és una **instància** o exemplar d'una base de dades.

Exercici 7

Les bases de dades relacionals, **únicament** guarden dades corresponent al món que volem modelar? Raona la teva resposta.

BLOC A) Pràctica 2: Model relacional

Objectius

- Aprofundir en conceptes importants del TEMA 2 de teoria.
- Realitzar exercicis del model relacional.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Exercicis / Preguntes

Exercici 1

Descriu la principal diferència entre **relació i esquema.**

Exercici 2

Hi ha diferents **motius que justifiquen la utilització de les vistes** a les bases de dades. Indica un parell de motius i raona la teva resposta.

Exercici 3

Explica, en les teves paraules, el significat del valor **NULL** a les bases de dades.

Exercici 4

Explica, en les teves paraules, **dues situacions** en las que ens pot interessar introduir valors NULL a un camp d'una relació d'una base de dades.

Exercici 5

Quina és la principal diferència entre **clau candidata i clau primària**?

Exercici 6

Analitza el següent esquema de relacions (base de dades d'accidents de cotxes) i contesta a les preguntes:

```
PERSONA (id-conductor, nom, direcció)
COTXE (matricula, any, model)
ACCIDENT (num-informe, lloc, data)
TE (id-conductor, matrícula)
PARTICIPAT (num-informe, id-conductor, matricula, danys)
```

- Per què la relació TE té una clau primària formada per dos atributs?
- La clau primària de la relació PARTICIPAT és correcte?

Exercici 7

Identifica i justifica les **claus primàries i foranies** dels següents esquemes de relacions:

```
ESTUDIANT (sid: integer, nom: string, login: string, edat:
integer)
PROFESSOR (pid: string, pnom: string, pcategoria: string)
CURS (cid: string, cnom: string, crèdits: integer)
AULA (aid: integer, planta: integer, capacitat: integer)
```

```
MATRICULATS (sid: integer, cid: string, nota: integer)
IMPARTEIX (pid: string, cid: string)
```

Exercici 8

Per què les **claus primàries no poden ser NULL**?

Exercici 9

Determina les **superclaus i claus candidates** per cada relació:

```
USUARI (id: integer, nom: string, DNI: text)
LLIBRE (isbn: string; títol: string, any: integer)
BIBLIOTECA (id: integer, nom: string, direcció: string)
```

Exercici 10

El debat de las claus primàries. Si podem assignar un ID únic a totes les files (per exemple, 1, 2,...), perquè penseu que necessitem dedicar temps i esforços per a determinar la clau primària?

BLOC A) Pràctica 3: Àlgebra relacional

Objectius

- Aprofundir en aspectes importants del TEMA 2 de teoria.
- Realitzar exercicis sobre àlgebra relacional (demostracions i consultes).
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Exercicis / Preguntes

Exercici 1

Tenim les següents relacions:

`SUPPLIER (sid: integer, sname: string, address: string)`: informació de proveïdors

`PART (pid: integer, pname: string, color: string)`: informació de parts / components

`CATALOG (sid: integer, pid: integer, cost: real)`: guarda el preu dels components servits pels proveïdors

a. Escriu l'operació d'àlgebra relacional per a la següent consulta: troba parells de sids (parells de proveïdors) tals que pel mateix component, el proveïdor amb el primer sid és més car que el segon

b. Escriu en les teves paraules que fa la següent operació d'àlgebra relacional:

$$\rho(R1, Catalog)$$
$$\rho(R2, Catalog)$$
$$\pi_{R1.pid}\sigma_{R1.pid=R2.pid \wedge R1.sid \neq R2.sid}(R1 \times R2)$$

Exercici 2

Considera les següents relacions sobre informació d'una companyia aèria:

`VOLS (volid, origen, destí, distancia, sortida, arribada)`

`AVIO (aid, anom, autonomia)`. El camp autonomia indica els kilòmetres que pot volar un avió sense repostar.

`CERTIFICAT (tid, aid)`. Aquesta relació indica els pilots (que són treballadors) que estan certificats per pilotar avions.

`TREBALLADOR (tid, tnom, salari)`. Tots els treballadors de la companyia – pilots inclosos.

a. Escriu un expressió en àlgebra relacional que trobi tots els aids dels avions que poden fer el viatge Bonn-Madrid sense parar.

b. Escriu una expressió en àlgebra relacional que mostri tots els eids dels empleats que estan certificats per a pilotar exactament tres avions.

Exercici 3

Demostra que l'operació **intersecció** es pot expressar en funció de l'operació **diferència**. La demostració es pot fer en un parell de línies.

Exercici 4

Re-escriu el THETA JOIN en funció de les operacions de selecció i producte cartesià.

Exercici 5

Calcula R ⋈ S

R				S		
A	B	C		B	C	D
1	2	3		2	3	4
6	7	8		2	3	5
9	7	8		7	8	10

Exercici 6

Tenim tres relacions: NAVEGANTS, RESERVES i VAIXELLS (Taula 1, 2 i 3, a continuació). Escriu i comenta les operacions d'àlgebra relacional per a les següents consultes.

a. Mostrar **el nom de tots els navegants** que han **reservat el vaixell** amb identificador **103**
b. Mostrar **el nom de tots els navegants** que han **reservat un vaixell** de color **roig**
c. Mostrar els **colors dels vaixells** que el navegant **LUBBER** ha reservat
d. Mostrar els **noms de tots els navegants** que han reservat **al menys un vaixell**
e. Mostrar el **nom dels navegants** que han reservat un **vaixell de color vermell o verd**
f. Mostrar els **nids** d'aquells **navegants** que tenen **més de 20 anys** i que **no han reservat** cap **vaixell** de color **vermell**
g. Mostrar el **nom dels navegants** que han reservat **tots els vaixells**.

Table 1: NAVEGANTS

nid	nnom	puntuació	edat
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	45.0
64	Horatio	7	35.0

71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Table 2: RESERVES

nid	bid	Data
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Table 3: VAIXELLS

bid	bnom	Color
101	Interlake	Azul
102	Interlake	Rojo
103	Clipper	Verde
104	Marine	Rojo

BLOC A) Pràctica 4: Entitat-Relació (1/2)

Objectius

- Aprofundir en aspectes importants del TEMA 4 de teoria.
- Realitzar diagrames E-R.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Exercici 1

Hem realitzat una sèrie d'entrevistes a professors i estudiants del departament. Hem identificat els següents aspectes:

- els professors tenen un codi que els identifica com a professor, una categoria i àrea d'investigació,
- els professors poden treballar en projectes com a investigadors,
- els professors també poden coordinar projectes,
- els projectes tenen un únic (professor) coordinador. Un professor pot coordinar un o més d'un projecte,
- un professor pot treballar en un projecte o més,
- en els projectes treballen diferents professors,
- els projectes tenen un codi, una entitat finançadora, una data d'inici i fi, i un pressupost,
- els estudiants també poden treballar en projectes,
- els estudiants tenen un codi que els identifica com a estudiant, un nom i un títol (de grau, de màster, de doctorat...),
- un mateix estudiant pot treballar en diferents projectes,
- els estudiants que treballen en projectes, tenen un supervisor (un professor),
- un professor pot supervisar a uns quants estudiants,
- els estudiants poden tenir tutors, que són estudiants de cursos superiors.

Realitza el model Entitat-Relació utilitzant la informació proporcionada i seguint les convencions del PDF disponible al campus. Documenta les principals decisions de disseny i totes les suposicions que facis.

Exercici 2

La empresa "Xupi Colònies per a nens i nenes de Catalunya" ens ha encarregat dissenyar la base de dades per gestionar les seves cases de colònies.

- Cada casa de colònies té un nom (que la identifica), i una direcció postal, que consisteix en el nom del carrer i el codi postal.

- De cada casa de colònies volem saber, a més del seu nom i direcció, la capacitat (el nombre de nens i nenes que es poden allotjar), la comarca en la que està situada i les activitats que es realitzen.
- Una casa de colònies té un nombre d'habitacions. Cada habitació s'identifica amb un codi alfanumèric. Aquest codi és el mateix en totes les cases de colònies. Volem poder distingir les habitacions que pertanyen a cada casa.
- Una casa de colònies pot oferir diferents activitats: natació, pintura, música... La mateixa activitat es pot realitzar en diferents cases. La empresa ens ha comunicat que volen tenir una avaluació de les activitats que s'ofereixen a cada casa. L'avaluació és una nota numèrica que identifica el grau de satisfacció dels nens amb les activitats.
- Les cases de colònies estan gestionades per tutors (un o més d'un). Volem saber el nom dels tutors, el seu telèfon de contacte, i la casa en la que treballen. En aquest sentit, l'empresa ens ha dit que pot haver una casa sense tutor, però no en cas contrari – si tenen un tutor a la seva base de dades, llavors aquest gestiona alguna casa.
- Els tutors, a més de supervisar nens i nenes, poden ser directors de cases de colònies. No tenim clar el número de directors per casa.
- En cada casa s'allotgen nens i nenes. Es vol tenir un control dels nens que estan a cada casa. Cada nen s'allotja a una única casa. Dels nens ens interessa saber, a més de la casa en la que s'allotgen, el seu nom, telèfon dels pares, i comarca de residència.
- De les comarques volem tenir el seu nom i número d'habitants.

Realitza el model Entitat-Relació utilitzant la informació proporcionada i seguint les convencions del PDF disponible en el campus. Documenta les principals decisions de disseny i totes les suposicions que facis.

BLOC A) Pràctica 4: Entitat-Relació (2/2)

Objectius

- Aprofundir en aspectes importants del TEMA 4 de teoria.
- Realitzar el pas a taules de diagrames E-R.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Exercici 1

Passa a taules el diagrama E-R de l'exercici 1.

Exercici 2

Passa a taules el diagrama E-R de l'exercici 2.

BLOC B) Pràctica 5: Introducció a SQL i PostgreSQL

Objectius

- Començar a familiaritzar-te amb SQL.
- Connectar-te a un servidor PostgreSQL.
- Aprendre a crear i eliminar bases de dades.
- Aprendre a crear, eliminar i modificar taules.
- Aprendre a inserir, eliminar i modificar dades a una taula.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Introducció (molt breu) a SQL

Segons ANSI (*American National Standards Institute*), SQL (*Structured Query Language*) és el llenguatge estàndard dels sistemes de bases de dades relacionals. Amb SQL podem realitzar consultes, recuperar dades, inserir, actualitzar i esborrar registres, crear bases de dades, taules, vistes i procediments, afegir permisos...SQL es basa en l'àlgebra relacional.

1) Connecta't a PostgreSQL (versió 10)

Per a connectar-te a PostgreSQL, engega l'ordinador amb Windows. Ves al menú INICI i selecciona l'entrada PostgreSQL 10. Aquí pots accedir al Shell (*psql*) i al gestor gràfic *pgAdmin* IV. Nosaltres utilitzarem **psql**. La contrasenya, *12345.*

Per desconnectar-te, pots fer servir la instrucció **\q**

2) Crear (i elimina) la bases de dades dreamhouse

Crea la base de dades *dreamhouse.* Per això, escriu la següent instrucció al *psql*

```
CREATE DATABASE dreamhouse;
```

DreamHouse és una empresa (fictícia) de lloguer de pisos i cases al Regne Unit.

Per esborrar una base de dades, escriu la següent instrucció al *psql*

```
DROP DATABASE nom-base-de-dades;
```

Nota: Per esborrar una base de dades, no ho pots fer un cop t'hagis connectat. Ho has de fer des de fora. És a dir, el cursor de PostgreSQL ha de ser `psql>` i **no** `nom-base-de-dades>`.

3) Connecta't a la base de dades

```
\connect dreamhouse;
```

Un cop connectat, el cursor del *psql* canvia i mostra el nombre de la base de dades.

4) Crear taules

Ara crea la següents taules. Las claus primàries estan subratllades:

- `BRANCH (branch_id, street, city, postcode)`

La taula BRANCH emmagatzema el carrer, la ciutat i el codi postal de les oficines de l'empresa.

- `STAFF (staff_id, fname, lname, position, sex, dob, salary, branch_id)`

La taula STAFF emmagatzema el nom, cognom, càrrec, sexe, data de naixement i salari anual dels treballadors de les oficines.

- `PROPERTY4RENT (property_id, Street, city, postcode, type, romos, rent, owner_id, staff_id, branch_id)`

La taula PROPERTY4RENT té informació sobre els pisos i cases que estan en lloguer. També té dades sobre els treballadors i les oficines que gestionen les propietats.

- `CLIENT (client_id, fname, lname, telnumber, pref_type, max_rent)`

La taula CLIENT emmagatzema, per a cada client, el nom, cognom, número de telèfon, preferència d'immoble (casa o pis) i quantitat màxima de lloguer que està disposat a pagar.

- `PRIVATEOWNER (privateowner_id, fname, lname, address, telnumber)`

La taula PRIVATEOWNER emmagatzema el nom, cognom, carrer i número de telèfon dels propietaris de las cases i pisos en lloguer.

- `VIEWING (client_id, property_id, viewdate, comment)`

La taula VIEWING és un registre de les visites realitzades pels clients a les propietats en lloguer.

- `REGISTRATION (client_id, branch_id, staff_id, date_joined)`

La taula REGISTRATION emmagatzema la data en la que un client es va registrar com a tal en l'empresa.

Per a crear les taules, has de fer servir la instrucció **CREATE TABLE.** Pots consultar la documentació de PostgreSQL https://www.postgresql.org/docs/10/static/sql-createtable.html

Els principals tipus de dades els pots trobar al següent enllaç: https://www.postgresql.org/docs/10/static/datatype.html

Exemple: el següent codi crea la taula BRANCH:

```
CREATE TABLE "branch" (
"branch_id" VARCHAR(5) NOT NULL,
"street" VARCHAR (25) NOT NULL,
"city" VARCHAR (15) NOT NULL,
"postcode" VARCHAR (10) NOT NULL,
Constraint "branch_pkey" Primary Key ("branch_id")
);
```

Per esborrar una taula, fes servir la instrucció `DROP TABLE nom-de-la-taula;`

5) Inserir dades a les taules

Ara les taules estan buides. Per a omplir-les de dades, fes servir la instrucció **INSERT**.

Tenim dos sentències INSERT (https://www.postgresql.org/docs/10/static/dml-insert.html):

```
INSERT INTO products (product_no, name, price) VALUES
    (1, 'Cheese', 9.99),
    (2, 'Bread', 1.99),
    (3, 'Milk', 2.99);
```

Aquesta sentència ens permet afegir una fila a la taula PRODUCTS amb els valors indicats a la llista de valors. Els valors han d'estar en ordre – segons les columnes.

```
INSERT INTO products (product_no, name, price)
  SELECT product_no, name, price FROM new_products
    WHERE release_date = 'today';
```

Aquesta instrucció ens permet afegir més d'una fila a una tabla. Veuràs la clàusula SELECT més endavant.

Exemple: la següent instrucció afegeix una fila a la taula BRANCH:

```
INSERT INTO branch VALUES ('B005','22 Deer Rd', 'London', 'SW1 4EH');
```

Utilitzant aquest exemple, omple les taules amb les dades del fitxer PDF que trobaràs al campus.

Nota:

Per a fer el procés d'inserció més àgil, i per a futures pràctiques, podem executar fitxers (*scripts*) SQL en PostgreSQL. Segueix aquests passos:

1)Escriu les instruccions INSERT – de fet, també pots escriure qualsevol sentència SQL – en un fitxer de text.

2)Guarda el fitxer amb extensió SQL a la unitat **D** dels ordinadors de les aules.

3)Executa el fitxer amb la instrucció `\i 'D:\\path\\fitxer.sql';`

6) Modificar continguts

La instrucció **UPDATE** ens permet modificar els continguts de les files d'una taula (https://www.postgresql.org/docs/10/static/sql-update.html).

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
UPDATE [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    SET { column_name = { expression | DEFAULT } |
          ( column_name [, ...] ) = [ ROW ] ( { expression | DEFAULT } [, ...] ) |
          ( column_name [, ...] ) = ( sub-SELECT )
        } [, ...]
    [ FROM from_list ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]
```

La instrucció **DELETE** ens permet esborrar files d'una taula (https://www.postgresql.org/docs/10/static/sql-delete.html).

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
DELETE FROM [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    [ USING using_list ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]
```

Realitza les següents accions de modificació:

a) Puja un 5% el salari dels 'managers'.

b) Promou l'empleat David Ford (SG14) a Manager amb salari 18000.

c) Elimina totes les visites que s'hagin realitzat a la propietat PG4.

7) Modificar l'estructura de les taules

La instrucció **ALTER TABLE** ens permet (https://www.postgresql.org/docs/10/static/ddl-alter.html):

a) afegir / eliminar una columna d'una taula:

```
ALTER TABLE products ADD COLUMN description text;
ALTER TABLE products DROP COLUMN description;
```

b) afegir / eliminar una restricció:

```
ALTER TABLE products ALTER COLUMN product_no SET NOT NULL;
ALTER TABLE products DROP CONSTRAINT some_name;
```

c) assignar / canviar l'assignació d'un valor per defecte a una columna.

```
ALTER TABLE products ALTER COLUMN price SET DEFAULT 7.77;
ALTER TABLE products ALTER COLUMN price DROP DEFAULT;
```

Realitza les següents operacions de modificació:

a) Afegeix el camp “country” a la taula BRANCH.
b) Esborra el camp que has creat

Per veure els canvis a la taula, pots usar aquesta instrucció:

```
dreamhouse=# \d nom-de-la-taula;
```

Bloc B) Pràctica 6: Consultes senzilles, ORDER BY i funcions d'agregació

Objectius

- Realitzar consultes senzilles.
- Realitzar consultes que utilitzen la clàusula WHERE per a seleccionar files que compleixin certes condicions.
- Ordenar els resultats de consultes amb la clàusula ORDER BY.
- Realitzar consultes utilitzant funcions d'agregació.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Abans de començar, carrega la base de dades *dreamhouse*.

1) SELECT

SELECT és probablement la paraula clau més utilitzada de SQL. SELECT ens permet escriure i realitzar operacions d'àlgebra relacional (selecció, projecció) d'una manera 'més humana'.

SELECT recupera i mostra les dades d'una (o més d'una) taula. El resultat d'una consulta és una relació (taula). L'estructura d'una consulta SELECT és https://www.postgresql.org/docs/10/static/sql-select.html):

```
SELECT      [DISTINCT | ALL] {* | [columnExpression [AS newName]] [, . . . ]}
FROM        TableName [alias] [, . . . ]
[WHERE      condition]
[GROUP BY   columnList] [HAVING condition]
[ORDER BY   columnList]
```

- SELECT: especifica les columnes que volem recuperar i mostrar en el resultat.
- FROM: especifica les taules que utilitzarem en la consulta.
- WHERE: filtra les files de la consulta segons una condició.
- GROUP BY: agrupa les files del resultat que tinguin el mateix valor a una columna.
- HAVING: filtra els resultats segons una condició.

L'ordre no es pot canviar. **Una consulta SQL no pot començar amb FROM**.

SELECT i FROM són obligatoris. La resta, opcionals.

***** significa totes les columnes d'una taula. Per exemple, si tenim la relació:

```
CLIENT (id, nom, cognom, ciutat, país, telèfon)
```

I fem la consulta

```
SELECT * FROM client;
```

Ens retornarà totes les columnes de la taula – amb els seus valors corresponents. Si volem el nom i el número de telèfon, la consulta hauria de ser:

```
SELECT nom, telèfon FROM client;
```

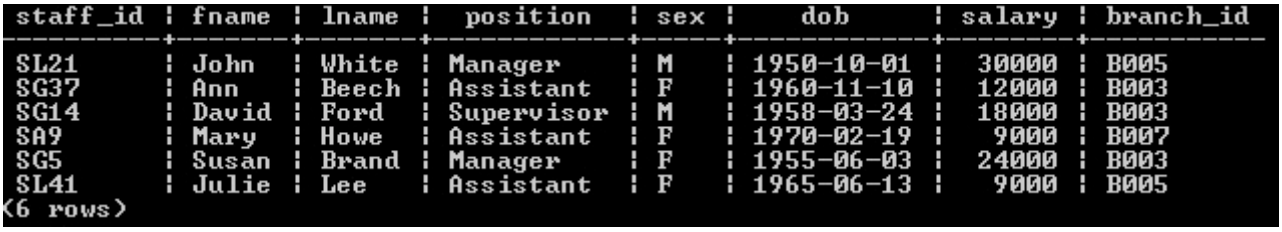
Si volem que a la taula resultat, la columna telèfon es digui "tel", ho farem així:

```
SELECT no, telèfon AS tel FROM client;
```

L'operador AS és un operador de renombrament.

Exercicis:

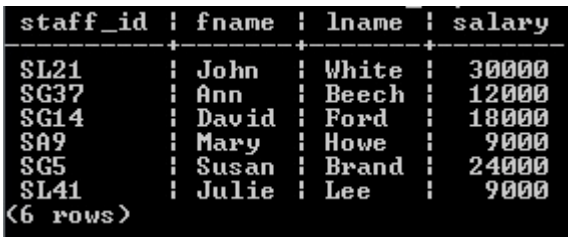
1) Mostra tots els detalls dels empleats.



staff_id	fname	lname	position	sex	dob	salary	branch_id
SL21	John	White	Manager	M	1950-10-01	30000	B005
SG37	Ann	Beech	Assistant	F	1960-11-10	12000	B003
SG14	David	Ford	Supervisor	M	1958-03-24	18000	B003
SA9	Mary	Howe	Assistant	F	1970-02-19	9000	B007
SG5	Susan	Brand	Manager	F	1955-06-03	24000	B003
SL41	Julie	Lee	Assistant	F	1965-06-13	9000	B005

(6 rows)

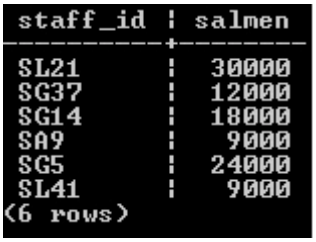
2) Per a cada empleat, mostra el seu número d'identificació, nom, cognom i salari.



staff_id	fname	lname	salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

(6 rows)

3) Per cada empleat, mostra el seu número d'identificació i el seu salari mensual. La columna de salari mensual apareixerà com "salmen".



staff_id	salmen
SL21	30000
SG37	12000
SG14	18000
SA9	9000
SG5	24000
SL41	9000

(6 rows)

2) DISTINCT

SELECT no elimina duplicats. Per eliminar duplicats, utilitzem **DISTINCT**.

La següent consulta mostra totes les propietats que s'han visitat:

```
SELECT property_id FROM viewing;
```

El resultat, com veuràs, té propietats duplicades. Re-escriu la consulta per a que no apareguin duplicats.

```
 property_id
-------------
 PG4
 PG36
 PA14
(3 rows)
```

3) WHERE

Habitualment necessitem restringir les files de les consultes. Això ho podem fer amb la clàusula WHERE, seguida de la condició de cerca. Les condicions bàsiques de cerca són:

- Comparació: compara el valor d'una expressió amb el valor d'una altra expressió
- Rang: comprova si el valor d'una expressió està dins d'un rang de valors
- Conjunt: comprova si un valor es troba dins d'un conjunt de valors
- NULL: comprova si una columna té un valor desconegut
- Patrons: comprova si un patró alfanumèric es troba a una cadena de caràcters

Per realitzar les comparacions, podem utilitzar:

- = igual
- < > diferent
- < menor que; > major que; <= menor o igual que; >= major o igual que
- AND, OR, IN, NOT, BETWEEN, IS (NOT) NULL
- LIKE (per patrons)

Exercicis

1) Mostra tots els empleats amb un salari > 10.000.

```
 staff_id | fname | lname |  position  | salary
----------+-------+-------+------------+--------
 SL21     | John  | White | Manager    |  30000
 SG37     | Ann   | Beech | Assistant  |  12000
 SG14     | David | Ford  | Supervisor |  18000
 SG5      | Susan | Brand | Manager    |  24000
(4 rows)
```

2) Mostra la direcció de totes les oficines a Londres o Glasgow.

```
 branch_id |    street    |  city   | postcode
-----------+--------------+---------+----------
 B005      | 22 Deer Rd   | London  | SW1 4EH
 B003      | 163 Main St  | Glasgow | G11 9QX
 B002      | 56 Clover Dr | London  | NW10 6EU
(3 rows)
```

3) Mostra el nom, cognom, càrrec i salari d'aquells empleats amb un salari entre 20000 i 30000.

```
 fname | lname | position | salary
-------+-------+----------+--------
 John  | White | Manager  |  30000
 Susan | Brand | Manager  |  24000
(2 rows)
```

4) Mostra el nom, cognom i càrrec de tots els managers i supervisors. Utilitza IN.

```
 fname | lname |  position
-------+-------+------------
 John  | White | Manager
 David | Ford  | Supervisor
 Susan | Brand | Manager
(3 rows)
```

5) Mostra tots els propietaris que tinguin la paraula 'Glasgow' a la seva direcció. Per realitzar aquesta consulta, necessites treballar amb l'operador LIKE.
   a. address LIKE 'H%' significa que la direcció comença amb una H (en majúscula) i la resta pot ser qualsevol cosa,
   b. address LIKE 'H_ _': significa que la direcció té 3 caràcters; el primer d'ells, una H.

```
 fname | lname  |            address
-------+--------+--------------------------------
 Carol | Farrel | 6 Achray St, Glasgow G32 9DX
 Tina  | Murphy | 63 Well St, Glasgow G42
 Tony  | Shaw   | 12 Park Pl, Glasgow G4 0QR
(3 rows)
```

6) Per a les propietats sense comentaris en la seva visita, mostra l'identificador dels clients i la data en la que la vam visitar. Recorda que NULL no és el mateix que '' en SQL.

```
 client_id |  viewdate
-----------+------------
 CR56      | 2016-05-26
(1 row)
```

4) ORDER BY

En general, les files d'una consulta SQL no estan ordenades. D'altra banda, però, a vegades ens interessa que sí que estiguin ordenades – per exemple, de menor a major en el cas dels salaris. Per això, utilitzem la clàusula ORDER BY (https://www.postgresql.org/docs/10/static/queries-order.html).

ORDER BY va seguit de les columnes, separades per comes, que s'utilitzaran per ordenar els resultats. L'ordre de presentació pot ser ascendent (ASC) o descendent (DESC).

```
SELECT select_list
    FROM table_expression
    ORDER BY sort_expression1 [ASC | DESC] [NULLS { FIRST | LAST }]
             [, sort_expression2 [ASC | DESC] [NULLS { FIRST | LAST }] ...]
```

Exercicis

1) Mostra un llistat dels empleats ordenat pel seu salari en ordre descendent.

staff_id	fname	lname	salary
SL21	John	White	30000
SG5	Susan	Brand	24000
SG14	David	Ford	18000
SG37	Ann	Beech	12000
SA9	Mary	Howe	9000
SL41	Julie	Lee	9000

(6 rows)

2) Mostra un llistat de les propietats per llogar ordenades pel seu tipus (casa o pis).

property_id	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

(6 rows)

3) Mostra un llistat de les propietats per llogar ordenades pel seu tipus i lloguer en ordre ascendent.

property_id	type	rooms	rent
PG4	Flat	3	350
PG36	Flat	3	375
PL94	Flat	4	400
PG16	Flat	4	450
PG21	House	5	600
PA14	House	6	650

(6 rows)

5) Funcions d'agregació

Les funcions d'agregació ens permeten realitzar operacions que no siguin únicament de consulta.

Les funcions d'agregació que veurem a l'assignatura són:

- COUNT: retorna el número de files d'una columna.
- SUM: retorna la suma de valors d'una columna.
- AVG: retorna la mitjana dels valors d'una columna.
- MIN / MAX: retorna el valor mínim i / o màxim d'una columna.

Les funcions d'agregació treballen sobre una única columna i retornen un únic valor.

AVG, MIN i MAX únicament treballen amb camps numèrics.

Per eliminar duplicats, escrivim DISTINCT abans del nom de la columna. DISTINCT no té cap efecte amb MIN i MAX.

Les funcions d'agregació es poden utilitzar amb la clàusula SELECT i HAVING. **No es poden utilitzar en qualsevol altre lloc de la consulta**.

Exercicis

1)Quantes propietats que tenen un lloguer superior a 350?

resultado
5

(1 row)

2)Quantes propietats (diferents) van ser visitades entre Març i Abril del 2003?

```
 resultado
-----------
         2
(1 row)
```

3)Quants managers hi ha a la base de dades, i quina és la suma dels seus salaris?

```
 count |  sum
-------+-------
     2 | 54000
(1 row)
```

4)Mostra el salari mínim, màxim, i mitjà dels treballadors.

```
 min  |  max  |        avg
------+-------+--------------------
 9000 | 30000 | 17000.000000000000
(1 row)
```

6) Transaccions

Tot i que les transaccions no formin part de les consultes senzilles, són part d'aquesta pràctica perquè les podem relacionar amb les consultes.

PostgreSQL té AUTOCOMMIT per defecte. Això significa que sempre es realitza un COMMIT després d'executar instruccions SQL. Per indicar que volem iniciar una transacció de manera explícita – i, per tant, haurem d'acabar amb un COMMIT o ROLLBACK – hem d'escriure BEGIN. Realitza les següents transaccions i comenta el que succeeix.

```
BEGIN;
DELETE FROM branch WHERE branch_id='B005';
ROLLBACK;
```

Què penses que ha passat? S'ha esborrat l'oficina o no? Fes un SELECT per esbrinar-ho.

```
BEGIN;
DELETE FROM branch WHERE branch_id='B005';
SELECT * FROM branch;
ROLLBACK;
```

I ara, surt l'oficina B005 al SELECT o no? Perquè?

BLOC B) Pràctica 7: vistes, agrupació de resultats, i consultes niades

Objectius

- Aprendre a crear vistes.
- Escriure consultes amb GROUP BY i HAVING.
- Aprendre a fer consultes niades (*nested queries*).
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Abans de començar, carrega la base de dades *dreamhouse*.

A) Vistes

Les vistes són una relació virtual. Per als usuaris, les vistes són taules, amb les seves files i columnes. D'altra banda, a diferència de les taules, les vistes no sempre existeixen físicament a la base de dades com un conjunt de valors. Tampoc formen part del model Entitat-Relació.

Per crear una vista, utilitzem la clàusula **CREATE VIEW** nom-de-la-vista AS (...) El contingut de la vista serà, habitualment, el resultat d'executar una sentència SELECT. La consulta pot ser senzilla o una consulta niada (https://www.postgresql.org/docs/10/static/sql-createview.html).

```
CREATE [ OR REPLACE ] [ TEMP | TEMPORARY ] [ RECURSIVE ] VIEW name [ ( column_name [, ...] ) ]
    [ WITH ( view_option_name [= view_option_value] [, ... ] ) ]
    AS query
    [ WITH [ CASCADED | LOCAL ] CHECK OPTION ]
```

Per exemple:

```
CREATE VIEW manager3staff AS
SELECT * FROM staff WHERE branch_id = 'B003';
```

El codi anterior crea una vista amb nom *manager3staff* que té tots els treballadors de l'oficina amb identificador B003. Si consultes les taules a la base de dades (\d) veuràs que la vista apareix com a taula.

Ja que les vistes són taules, podem fer:

```
SELECT * FROM manager3staff;
```

¿Què penses que retornarà aquesta consulta? Clar! Els empleats de l'oficina B003.

Exercici

1)Crea una vista amb els empleats de l'oficina amb identificador B003 sense el seu salari.

Les vistes es guarden a la base de dades. Per esborrar-les, utilitzem:

DROP VIEW nom-de-la-vista.

Les vistes s'actualitzen si les taules base – de les que recupera les dades – s'actualitzen. El contrari no sempre és cert. Veiem un exemple.

Crea la següent vista:

```
CREATE VIEW staffpropcnt (branch_id, staff_id, cnt) AS
SELECT s.branch_id, s.staff_id, COUNT (*)
FROM staff s, property4rent p
WHERE s.staff_id = p.staff_id
GROUP BY s.branch_id, s.staff_id;
```

Pots definir què fa la vista?

Ara volem fer un INSERT sobre la vista. Com és una taula, en principio, ho podem fer:

```
INSERT INTO staffpropcnt VALUES ('B003', 'SG5', 2);
```

Què passa? Et dono una pista: intentem afegir dos propietats gestionades pel mateix treballador, però...quines són les propietats?

Tot i que podem fer que una vista es pugui actualitzar, com et suggereix PostgreSQL, les vistes són realment útils per recuperar informació i ocultar part/s d'una base de dades als usuaris.

B) GROUP BY, HAVING

Les consultes que has realitzat fins ara produeixen un llistat de dades sense agrupar. D'altra banda, moltes vegades necessitem realitzar consultes de l'estil 'calcula quants empleats treballen en cada oficina'. Per fer això, necessitem agrupar les files (els treballadors) i produir un únic resultat pel grup (empleats de l'oficina A, B...). Això ho podem fer amb **GROUP BY**.

Quan utilitzem GROUP BY, totes les columnes que apareixen en el SELECT han d'aparèixer a la clàusula GROUP BY. Una excepció són les columnes que s'utilitzen a les funcions d'agregació de la mateixa consulta (https://www.postgresql.org/docs/10/static/queries-table-expressions.html#QUERIES-GROUP).

Exercici

1)Calcula el nombre de treballadors de cada oficina i la suma dels seus salaris.

```
 branch_id | count |  sum
-----------+-------+-------
 B003      |     3 | 54000
 B005      |     2 | 39000
 B007      |     1 |  9000
(3 rows)
```

Podem afegir més control al GROUP BY amb la clàusula **HAVING**. HAVING ens permet restringir els grups que es mostraran a la taula resultat.

HAVING i WHERE semblen semblants, però no ho són. **WHERE treballa sobre files individuals. HAVING ho fa sobre grups.**

La condició que segueix a HAVING sempre inclou al menys una funció d'agregació.

Exercici

1)Per cada oficina amb més d'un empleat, calcula quants empleats treballen en cadascuna d'elles i la suma dels seus salaris.

```
 branch_id | count |  sum
-----------+-------+-------
 B003      |     3 | 54000
 B005      |     2 | 39000
(2 rows)
```

C) Consultes niades

Les consultes niades són consultes "dins" de consultes. Les consultes niades sempre les trobarem entre parèntesis.

Podem utilitzar consultes niades a les clàusules WHERE i HAVING. També les podem utilitzar a les operacions INSERT, UPDATE i DELETE.

Les consultes niades les podem dividir en tres tipus:

- les escalars retornen una única columna i una única fila (un valor)
- les de fila retornen múltiples valors, però una única fila
- les de taula, que retornen múltiples files i columnes

Exemple:

Escriu una consulta que mostri tots els empleats de l'oficina ubicada al carrer '163 Mian St'. Necessites dos SELECT. Un SELECT és per obtenir l'identificador de l'oficina al carrer donat. L'altre SELECT és per obtenir els detalls que et demanen.

```
SELECT staff_id, fname, lname, position
FROM staff
WHERE branch_id = (SELECT branch_id FROM branch WHERE street = '163 Main
St');
```

Exercicis

1)Mostra el nom, cognom i càrrec dels treballadors que tinguin un salari més gran que el salari mitjà, i mostra també la diferència respecte al salari mitjà. Pista: necessitaràs funcions d'agregació. Si vols, pots utilitzar vistes.

```
 fname | lname |  position  |       saldif
-------+-------+------------+--------------------
 John  | White | Manager    | 13000.000000000000
 David | Ford  | Supervisor |  1000.000000000000
 Susan | Brand | Manager    |  7000.000000000000
(3 rows)
```

2)Mostra totes les propietats que estan gestionades pels treballadors de l'oficina al carrer '163 Main St'. Utilitza l'operador IN i tres SELECT.

```
 property_id |   street    |  city   | postcode | type  | rooms | rent
-------------+-------------+---------+----------+-------+-------+------
 PG36        | 2 Manor Rd  | Glasgow | G32 4QX  | Flat  |     3 |  375
 PG21        | 18 Dale Rd  | Glasgow | G12      | House |     5 |  600
 PG16        | 5 Novar Dr  | Glasgow | G12 9AX  | Flat  |     4 |  450
(3 rows)
```

Els operadors **ANY (SOME)** i **ALL** es poden utilitzar a les consultes niades que retornen una única columna. Si la consulta està precedida per ALL, tots el valors retornats per la consulta niada han de complir la condició per a sigui certa. Si la consulta està precedida per ANY, és suficient amb que un valor dels retornats per la consulta niada compleixi la condició per a que sigui certa.

Exercicis

1)Mostra tots els treballadors amb salari més gran que el salari d'almenys un treballador de l'oficina B003.

```
 staff_id | fname | lname |  position  | salary
----------+-------+-------+------------+--------
 SL21     | John  | White | Manager    |  30000
 SG14     | David | Ford  | Supervisor |  18000
 SG5      | Susan | Brand | Manager    |  24000
(3 rows)
```

2)Mostra tots els treballadors amb salari més gran que el salari de tots el treballadors de l'oficina B003.

```
 staff_id | fname | lname | position | salary
----------+-------+-------+----------+--------
 SL21     | John  | White | Manager  |  30000
(1 row)
```

BLOC B) Pràctica 8: consultes multi-taula

Objectius

- Aprendre a realitzar consultes multi-taula.
- Conèixer el JOINS de SQL
- Realitzar consultes utilitzant operacions sobre conjunts.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Abans de començar, carrega la base de dades *dreamhouse*.

A) Joins

Tots els exercicis que has realitzat fins ara tenen una característica - limitació: les columnes de la taula de resultats són d'una única taula. En molts casos, això no és suficient. Per combinar columnes de diferents taules necessitem fer un JOIN.

L'operació JOIN combina columnes de dos taules. Podem tenir més d'un JOIN a una mateixa consulta SQL.

La forma més senzilla de fer un JOIN és incloure més d'una taula a la clàusula FROM, i especificar el criteri d'unió a la clàusula WHERE.

Per exemple, si volem mostrar el nom de tots els clients que han visitat una propietat, i els seus comentaris, podem escriure la consulta:

```
SELECT c.client_id, fname, lname, property_id, comment
                FROM client c, viewing v
            WHERE c.client_id = v.client_id;
```

Aquesta consulta és multi-taula. És equivalent a la següent consulta:

```
SELECT c.client_id, fname, lname, property_id, comment
FROM client c INNER JOIN viewing v ON c.client_id = v.client_id;
```

Exercici

1)Per cada oficina, mostra quants empleats gestionen propietats, el nom d'aquests empleats i l'identificador de les propietats.

```
branch_id | staff_id | fname | lname | property_id
----------+----------+-------+-------+------------
B003      | SG14     | David | Ford  | PG16
B003      | SG37     | Ann   | Beech | PG21
B003      | SG37     | Ann   | Beech | PG36
B005      | SL41     | Julie | Lee   | PL94
B007      | SA9      | Mary  | Howe  | PA14
(5 rows)
```

SQL defineix tres tipus de JOIN. Us deixo com a exercici establir la relació d'aquests JOINS amb els que heu vist al tema d'àlgebra relacional.

- **INNER JOIN**: selecciona les files de dos taules en les que hi ha una correspondència de valors en la condició (ON) del JOIN. La resta de files no es mostren.

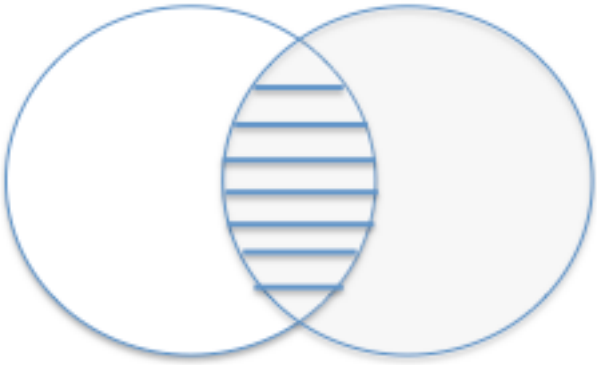


Figura 1: inner Join

- **LEFT JOIN**: retorna totes les files de la taula de l'esquerra, i les files de la taula de la dreta que compleixen la condició del JOIN. El resultat és NULL per les files de la dreta que no tenen parella a l'esquerra.

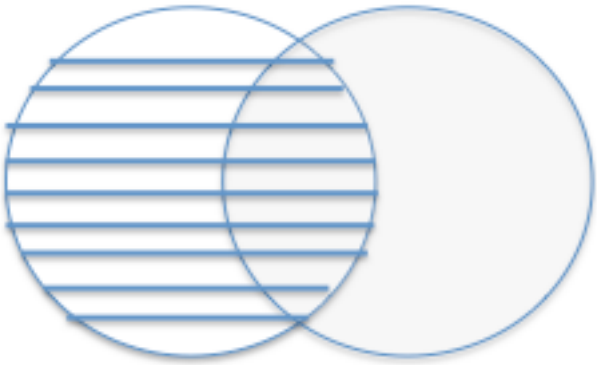


Figura 2: Left Join

- **RIGHT JOIN**: retorna totes les files de la taula de la dreta, i les files de la taula de l'esquerra que compleixen la condició del JOIN. El resultat és NULL per a les files de l'esquerra sense parella a la dreta.

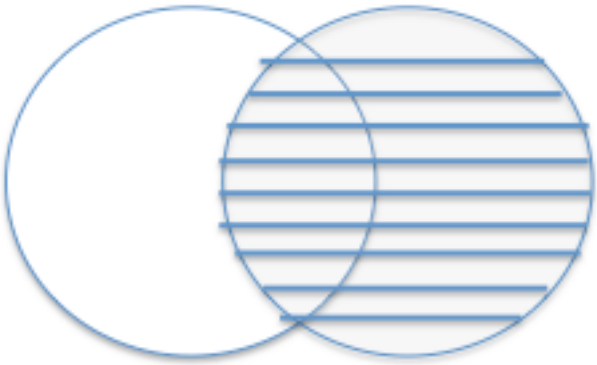


Figura 3: Right Join

- **FULL OUTER JOIN**: retorna totes les files de la taula de l'esquerra i de la dreta.

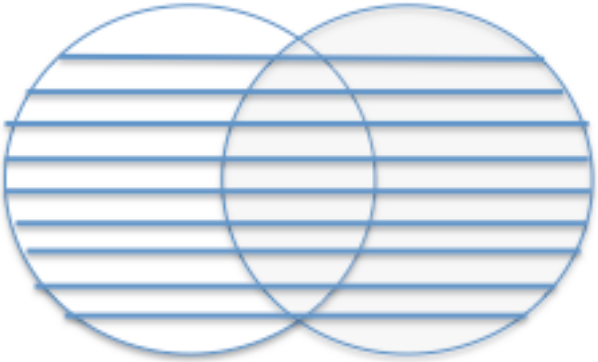


Figura 4: FULL OUTER JOIN

Exercicis

1) Crea les següent taules:

- table_left (id, valor);
- table_right (id, valor);

Afegeix els següents registres:

table_left		table_right	
ID	Valor	ID	Valor
1	FOX	1	TROT
2	COP	2	CAR
3	TAXI	3	CAB
6	WASHINGTON	6	MONUMENT
7	DELL	7	PC
5	ARIZONA	8	MICROSOFT
4	LINCOLN	9	APPLE

10	LUCENT	10	SCOTCH

Executa les següents consultes i analitza els resultats:

Consulta A)

```
SELECT table_left.id, table_left.value, table_right.id,
table_right.value FROM table_left INNER JOIN table_right ON
table_left.id = table_right.id;
```

Consulta B)

```
SELECT table_left.id, table_left.value, table_right.id,
table_right.value FROM table_left LEFT JOIN table_right ON table_left.id
= table_right.id;
```

Consulta C)

```
SELECT table_left.id, table_left.value, table_right.id,
table_right.value FROM table_left RIGHT JOIN table_right ON
table_left.id = table_right.id;
```

Consulta D)

```
SELECT table_left.id, table_left.value, table_right.id,
table_right.value FROM table_left FULL OUTER JOIN table_right ON
table_left.id = table_right.id;
```

Consulta E)

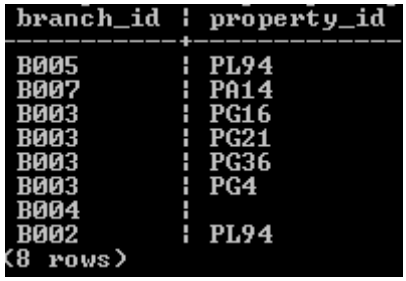
```
SELECT table_left.id, table_left.value, table_right.id,
table_right.value FROM table_left LEFT JOIN table_right ON table_left.id
= table_right.id WHERE table_right.id IS NULL;
```

Consulta F)

```
SELECT table_left.id, table_left.value, table_right.id,
table_right.value FROM table_left RIGHT JOIN table_right ON
table_left.id = table_right.id WHERE table_left.id IS NULL;
```

2) Carrega la base de dades *dreamhouse* i escriu les següents consultes:

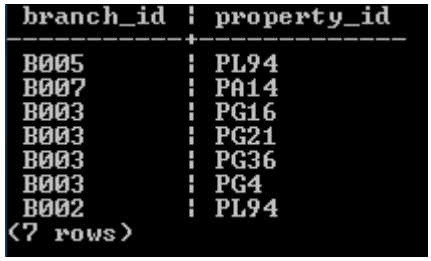
- Mostra totes les oficines, i propietats que estiguin (o no) a la mateixa ciutat.



branch_id	property_id
B005	PL94
B007	PA14
B003	PG16
B003	PG21
B003	PG36
B003	PG4
B004	
B002	PL94

(8 rows)

- Mostra totes les propietats, i oficines que estiguin (o no) a la mateixa ciutat.



branch_id	property_id
B005	PL94
B007	PA14
B003	PG16
B003	PG21
B003	PG36
B003	PG4
B002	PL94

(7 rows)

B) UNION, INTERSECT I EXCEPT

SQL també ens ofereix les operacions de conjunt UNION, INTERSECT i EXCEPT (diferència). Si et fixes, semblen molt similars a les operacions de conjunt que has vist a àlgebra relacional.

UNION: unió de dos taules, A i B. El resultat és una taula amb les files de les dos taules.

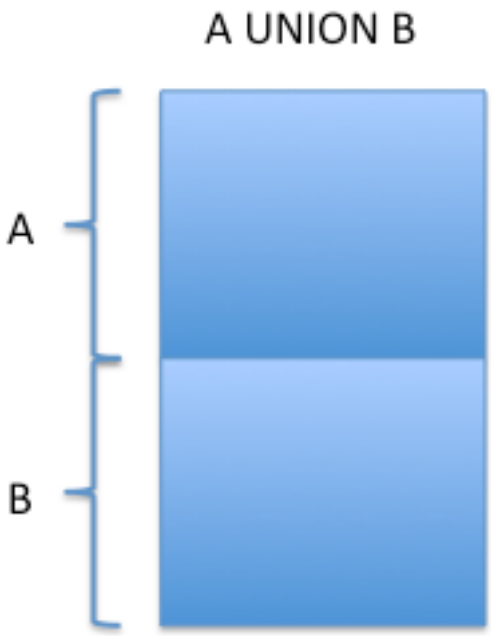


Figura 5: UNION

INTERSECT: intersecció de dos taules, A i B. El resultat és una taula amb les files que comparteixen A i B

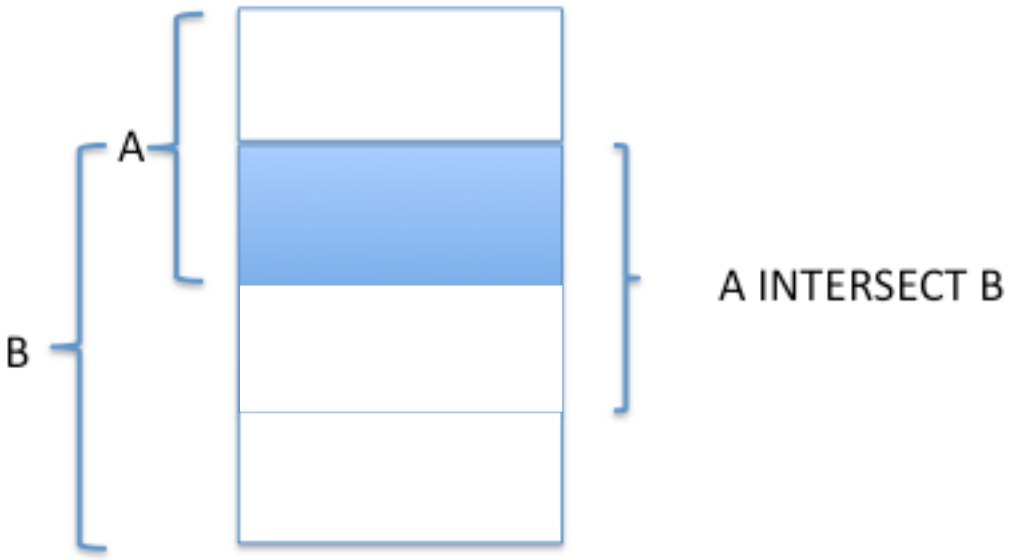


Figura 6: INTERSECT

EXCEPT: diferència de dos taules, A i B. El resultat és una taula amb les files de la taula A que no estan a la taula B.

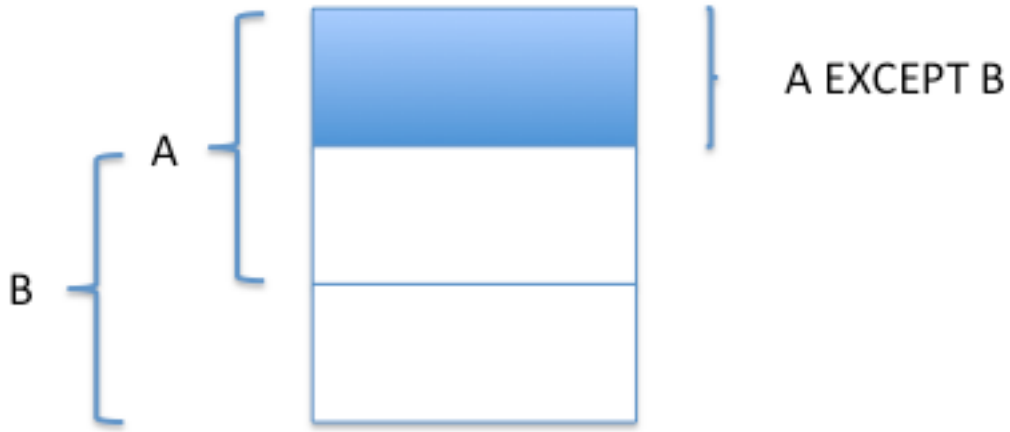


Figura 7: EXCEPT

Recorda que per realitzar aquestes operacions les taules A i B han de ser compatibles. Dos taules són compatibles si tenen la mateixa estructura: nombre i tipus de columnes.

Exercicis

1)Construeix una llista de totes les ciutats en les que podem trobar oficines o propietats en lloguer.

```
   city
----------
 Aberdeen
 London
 Glasgow
 Bristol
(4 rows)
```

2)Construeix una llista de totes les ciutats en les que hi hagin oficines i propietats en lloguer.

```
   city
----------
 Aberdeen
 London
 Glasgow
(3 rows)
```

3)Construeix una llista de totes les ciutats en la que podem trobar oficines però no propietats en lloguer.

```
  city
--------
 Bristol
(1 row)
```

BLOC B) Pràctica 9: JDBC i funcions SQL

Objectius

- Conèixer aspectes importants i bàsics de JDBC
- Aplicar els coneixements adquirits a Programació II (orientació a objectes i interfícies gràfiques amb Java Swing) en el context del desenvolupament de programes d'aplicació en Bases de Dades.
- Ser capaç de programar funcions (stored procedures) en SQL.
- Millorar i / o reforçar habilitats d'aprendre a aprendre.

Abans de començar, carrega la base de dades *dreamhouse*.

A) JDBC

Passos previs

- Descarrega't del campus virtual el fitxer **JDBC1718.zip**. Aquest fitxer és un projecte Netbeans – un IDE per desenvolupar programes en JAVA (i altres llenguatges de programació).
- Descarrega't el driver JDBC de PostgreSQL: https://jdbc.postgresql.org/download.html
- Assegurat que la base de dades dreamhouse està carregada. No cal que tinguis PostgreSQL obert.
- Obre el projecte amb Netbeans.
- Afegeix el jar del driver JDBC a "Libraries" del projecte.
- Comprova que compila correctament.

Sobre l'aplicació

L'aplicació és d'escriptori i treballa amb la base de dades dreamhouse. L'aplicació es divideix en el següents fitxers:

- **Main.java**. És la classe principal (JFRAME). Ens permet connectar-nos a la base de dades i realitzar les tres operacions principals: consultar, inserir i eliminar dades.
- **Query.java**. És una classe secundaria (JDIALOG) que ens permet realitzar consultes a la base de dades.
- **Insert.java**. És una classe secundaria de tipus formulari (JDIALOG) que ens permet introduir noves dades.

Nota: A la classe Main, modifica la línia de la connexió per a introduir la contrasenya necessària per a connectar-se a la base de dades.

Feina a fer

1. Analitza el codi font per a intentar entendre el funcionament i les relacions entre les classes abans de llegir la breu introducció a JDBC de l'enunciat de la pràctica. Potser necessites revisar una mica els teus apunts de Programació II.
2. Al fitxer Query.java, seguint com a model el codi proporcionat, implementa la funcionalitat de consultar els clients de la base de dades. Pots ser creatiu en el disseny de la interfície gràfica.
3. Al fitxer Insert.java, seguint com a model el codi proporcionat, implementa la funcionalitat per a inserir nous clients. Pots ser creatiu en el disseny de la interfície gràfica.
4. Implementa el JDialog amb nom Delete.java. Pots servir el mateix model que Insert, ja que totes dues operacions son UPDATEs al JDBC. Volem esborrar oficines i clients.

Breu introducció a JDBC

JDBC (*Java Database Connectivity*) és una API de JAVA que ens permet connectar-nos a bases de dades relacionals. Amb JDBC podem connectar-nos a una base de dades i executar sentències SQL. Els resultats de les sentències SQL són accessibles des del programa JAVA.

Els principals components JDBC són:

- *Driver Manager*. És un gestor que s'ocupa de connectar l'aplicació JAVA amb el driver de la base de dades.
- *Driver*. És una interfície que s'ocupa de gestionar les comunicacions amb el servidor de la base de dades.
- *Connection*. És la interfície amb el mètodes per a connectar-se a la base de dades.
- *Statement*. La utilitzem per enviar sentències SQL a la base de dades.
- *ResultSet*. Els resultats d'una consulta SQL són objectes del tipus ResultSet.
- *SQLException*. Classe pròpia que gestiona els errors en la connexió amb la base de dades i el programa Java.

Mirem el codi font de l'aplicació per a entendre millor el funcionament d'aquestes classes.

Per a més informació sobre JDBC, pot seguir aquest tutorial:

https://www.tutorialspoint.com/jdbc/

B) PL/pgSQL: alguns aspectes

PL/pgSQL (https://www.postgresql.org/docs/10/static/plpgsql.html) és un llenguatge de programació procedimental (que està instal·lat per defecte a PostgreSQL) que ens permet estendre la funcionalitat de la base de dades amb funcions definides per nosaltres. Aquestes funcions habitualment no les podem fer, o ens resulta molt difícil fer-les, amb instruccions SQL com les que heu vist durant el curs. Aquestes funcions es coneixen amb el nom de "stored procedures".

La meva primera funció

Les funcions es creen amb **CREATE FUNCTION**:

```
CREATE FUNCTION nom-funció (arg1 tipus, arg2 tipus)
RETURNS tipus AS $$
BEGIN
-- codi
END;
$$ LANGUAGE nom-llenguatge;
```

Si la funció és vàlida, PostgreSQL retorna CREATE FUNCTION. Per a cridar a la funció, fem: **SELECT** nom-funció(args); també podem fer SELECT * from nom-funció(args). Les funcions es guarden a la base de dades. Per a esborrar-la, fem: **DROP FUNCTION** nom-funció.

Exemple:

```
CREATE FUNCTION inc(val integer)
RETURNS integer AS $$
BEGIN
RETURN val + 1;
END;
$$ LANGUAGE plpgsql;
```

Exercici 1: Fes una funció que multipliqui dos nombres enters que rep com a paràmetres. El resultat de cridar a SELECT multiplica (2,3) hauria de ser 6.

Paràmetres

A l'exemple i exercici anterior has treballat amb paràmetres d'entrada. PL/pgSQL també ens permet tenir paràmetres de sortida, que són paràmetres d'entrada a on guardem la sortida. RETURNS i RETURN no són necessàries en aquest cas.

OUT nom-paràmetre tipus

Per a consultar els tipus de dades: http://www.postgresqltutorial.com/postgresql-data-types/

Exercici 2: re-escriu la funció multiplica amb OUT. Les assignacions (x = 2) es fan amb **:=**.

També tenim paràmetres INOUT – consulta la documentació.

Funcions que retornen una taula

Les funcions també les podem programar per a que ens retornin taules. Per exemple, la següent funció ens retorna l'id i el carrer de les oficines que tenen un cert patró de text al seu camp ciutat.

```
CREATE OR REPLACE FUNCTION get_branch (patro VARCHAR)
RETURNS TABLE (
  id_oficina VARCHAR,
  carrer VARCHAR)
AS $$
BEGIN
RETURN QUERY
  SELECT branch_id, street FROM branch WHERE city LIKE patro;
END;
$$ LANGUAGE plpgsql;
```

SELECT * from get_branch('L%') -> la funció retorna les oficines a ciutats amb que comencen amb L.

Indiquem que retorna una taula amb **RETURNS TABLE**. La taula resultant ha de tenir el mateix tipus i nombre d'atributs que el SELECT.

Exercici 3: escriu una funció que retorni una taula amb el nom i càrrec dels empleats amb salari més gran que un concret – que l'usuari introdueix quan crida a la funció

Condicionals

També tenim estructures condicionals – les iteratives les podeu consultar a la documentació. Una de les més senzilles és:

```
IF condició THEN
   Sentències;
END IF;
```

Exercici 4 (avançat): Crea una funció que donat el nom de dos empleats, mostri el nom de l'empleat amb el salari més gran i el seu salari.

Per a mostrar un missatge (ja sigui d'error o un *print*): **RAISE NOTICE** 'el missatge';

Per declarar variables a la funció, afegim el bloc **DECLARE** abans de BEGIN:

```
DECLARE
nom-variable tipus;
BEGIN (...)
```

Finalment, una de les funcions, potser més interessants de les funcions, són els *triggers*. En aquest curs no els veurem, però pots consultar la documentació de PostgreSQL (https://www.postgresql.org/docs/10/static/plpgsql-trigger.html) per a veure com es crea i s'executen. Et proposo l'exercici de fer un trigger que cada cop que es visiti una propietat, es guardi a una taula (un log) l'id del client i de la propietat, i la data.