

INTRODUCCIÓ A SQL



1. INTRODUCCIÓ A SQL (STRUCTURED QUERY LANGUAGE)

- El llenguatge estructurat de consultes (SQL) és un llenguatge de base de dades normalitzat*, utilitzat per la gran majoria dels servidors de bases de dades que manegen bases de dades relacionals o objecte-relacionals.
 - **SQL és el llenguatge universal de les bases de dades relacionals.**
 - normalitzat: és un llenguatge de programació estàndard per al maneig d'informació des d'una base de dades relacional.
- En aquest curs aprendrem a utilitzar SQL en Postgres.

1. INTRODUCCIÓ A SQL (STRUCTURED QUERY LANGUAGE)

- SQL: 1970 Codd.
- Consultes, actualitzacions, definició de dades i control a BD.
- Per a administradors, desenvolupadors i usuaris.
- Llenguatge **declaratiu**: S'especifica que es vol, no els passos que cal fer per obtenir-lo.
- Embegut en PL/SQL o Java, per exemple.

1. INTRODUCCIÓ A SQL (STRUCTURED QUERY LANGUAGE)

- Interactivament, introduint les ordres des de terminal, s'obtenen resultats.
 - Senzill, les paraules clau permeten escriure les ordres com si foren frases en què s'especifica (en anglès) que és el que volem obtenir.
 - Per exemple: `SELECT nombre FROM municipios WHERE poblacion>5000 ORDER BY poblacion;`

1.1. COMPONENTS D' SQL

- El llenguatge SQL està compost per:
 - Ordres,
 - Clàusules,
 - Operadors i
 - Funcions d'agregat.
- Aquests elements es combinen a les instruccions per crear, actualitzar i manipular les bases de dades.

1.1. COMPONENTS D' SQL

- **Ordres:** de 2 tipus
 - Les que permeten crear i definir noves bases de dades, camps i índexs: CREATE, DROP, ALTER
 - Les que permeten generar consultes per ordenar, filtrar i extreure dades de la base de dades: SELECT, INSERT, UPDATE, DELETE
- **Clàusules:** condicions utilitzades per concretar quines dades són les que es vol seleccionar o manipular: FROM, WHERE, GROUP BY, HAVING, ORDER BY
- **Operadors** : Lògics (AND, OR, NOT), de Comparació (>, <, >=, <=, =, <>, BETWEEN o LIKE)
- **Funcions d'agregat:** AVG, COUNT, SUM, MAX, MIN

1.2. TIPUS DE SENTÈNCIES D' SQL

- **DDL**: CREATE, DROP i ALTER
- **DML**: SELECT, INSERT, UPDATE i DELETE
- **DCL**: GRANT, REVOKE (concedir, suprimir privilegis)
i COMMIT, ROLLBACK (transaccions)

2. TIPUS DE DADES POSTGRES

- PostgreSQL admet els tipus de dades següents:
 - Booleà
 - Tipus de caràcters com ara caràcters, varchar i text.
 - Tipus numèrics com nombres enters i en coma flotant.
 - Tipus temporals com ara data, hora, marca de temps (timestamp) i interval
 - UUID per emmagatzemar identificadors únics universals
 - Arrays per emmagatzemar cadenes de caràcters, números, etc.
 - JSON per emmagatzemar dades JSON
 - hstore per emmagatzemar parells clau-valor
 - Tipus especials com ara adreces de xarxa i dades geomètriques.

3. DDL

- El llenguatge de definició de dades es basa en tres sentències: CREATE, ALTER i DROP, que s'apliquen a cada objecte de la base de dades.
- Cada objecte es pot correspondre amb un objecte físic (és a dir, un arxiu als sistemes de fitxers) o a un objecte lògic (és a dir, una definició emmagatzemada al catàleg de la base de dades).

3.1. CREATE

- Crear una base de dades:
- **CREATE DATABASE** nom;

```
Query Editor  Historial de Consu...  
1  -- Database: BD1  
2  
3  -- DROP DATABASE "BD1";  
4  
5  CREATE DATABASE "BD1"  
6      WITH  
7      OWNER = postgres  
8      ENCODING = 'UTF8'  
9      LC_COLLATE = 'C.UTF-8'  
10     LC_CTYPE = 'C.UTF-8'  
11     TABLESPACE = pg_default  
12     CONNECTION LIMIT = -1;
```

3.1. CREATE

- **Crear una taula**

- Per crear una taula nova, utilitzeu la sentència CREATE TABLE. La sintaxi bàsica de la sentència CREATE TABLE és:

- **CREATE TABLE** opcional [IF NOT EXISTS] taula_nom (
 - columna1_nom tipus(length) columna_constraint,
 - columna2_nom tipus(length) columna_constraint,
 - columna3_nom tipus(length) columna_constraint,
 - table_constraints
-);

3.1. CREATE

- Les **restriccions de columna** especifiquen regles que han de seguir les dades emmagatzemades a la columna. Les posem al final de la definició de la columna i abans de la següent columna.
- **Restriccions**
- PostgreSQL inclou les restriccions de columna següents:
 - **NOT NULL**: garanteix que els valors d'una columna no poden ser NULL.
 - **UNIQUE**: garanteix els valors únics d'una columna a les files de la mateixa taula.
 - **PRIMARY KEY**: clau principal, la columna que identifica de manera única les files d'una taula. Una taula pot tenir una i només una clau principal. La restricció de clau principal us permet definir la clau primària d'una taula.
 - **CHECK**: una restricció CHECK garanteix que les dades han de satisfer una expressió booleana.
 - **FOREIGN KEY**: la clau externa o forana assegura que existeixen valors en una columna o en un grup de columnes d'una taula en una columna o grup de columnes d'una altra taula. A diferència de la clau principal, una taula pot tenir moltes claus externes
- I finalment, especifiquem les **restriccions de la taula**, incloent la clau primària, la clau externa i les restriccions de check.

3.1.1. TIPUS DE DADES PRINCIPALS

● Numèrics

Nombre	Tamaño de almacenamiento	Descripción	Distancia
smallint	2 bytes	Entero de rango pequeño	-32768 a +32767
integer	4 bytes	Elección típica para entero	-2147483648 a +2147483647
bigint	8 bytes	Entero de gran alcance	-9223372036854775808 a 9223372036854775807
decimal	variable	Precisión especificada por el usuario, exacta	Hasta 131072 dígitos antes del punto decimal; hasta 16383 dígitos después del punto decimal
numeric	variable	Precisión especificada por el usuario, exacta	Hasta 131072 dígitos antes del punto decimal; hasta 16383 dígitos después del punto decimal
real	4 bytes	Precisión variable, inexacta	Precisión de 6 dígitos decimales
double precision	8 bytes	Precisión variable, inexacta	Precisión de 15 dígitos decimales
smallserial	2 bytes	Pequeño entero autoincrementador	1 a 32767
serial	4 bytes	Entero autoincrementador	1 a 2147483647
bigserial	8 bytes	Gran entero autoincrementador	1 a 9223372036854775807

3.1.1. TIPUS DE DADES PRINCIPALS

- **Cadena de caràcters**

Nombre	Descripción
<code>Character varying(n), varchar(n)</code>	Longitud variable con limite
<code>Carácter(n), char(n)</code>	Longitud fija, relleno los espacios en blancos
<code>Text</code>	Longitud variable ilimitada

- `char`: longitud fija, especificada por `n`. Una columna con un tipo, por ejemplo, `char(20)` utiliza 20 caracteres por columna independientemente si se están utilizando o no. Por ejemplo, "gat" ocupará 20 espacios de memoria.
- `varchar`: longitud variable, donde la máxima longitud es especificada por `n`. A diferencia del tipo anterior, si se tiene una columna de este tipo, se utiliza solo la longitud de la dato que se quiere guardar. Por ejemplo, "gat" solo ocupará 4 espacios de memoria.
- `text`: Una columna de longitud variable, donde no hay limitación para la longitud de esta.

3.1.1. TIPUS DE DADES PRINCIPALS

- **Booleans:**

Nombre	Tamaño de almacenamiento	Descripción
boolean	1 byte	El estado es true o false

- **Dades binàries:** El tipus de dades **bytea** permet emmagatzemar cadenes binàries

Nombre	Tamaño de almacenamiento	Descripción
bytea	1 o 4 bytes más la cadena binaria real	cadena binaria de longitud variable

3.1.1. TIPUS DE DADES PRINCIPALS

- **Data i hora**

Nombre	Tamaño de almacenamiento	Descripción	Valor vajo	Valor alto
timestamp [(p)] [without time zone]	8 bytes	Tanto la fecha como la hora (sin zona horaria)	4713 BC	294276 AD
TIMESTAMP TZ	8 bytes	Tanto la fecha como la hora, con zona horaria	4713 BC	294276 AD
date	4 bytes	Fecha (sin hora del día)	4713 BC	5874897 AD
time [(p)] [without time zone]	8 bytes	Hora del día (sin fecha)	00:00:00	24:00:00
time [(p)] with time zone	12 bytes	Solo horas del día, con zona horaria	00:00:00+1459	24:00:00-1459
interval [fields] [(p)]	12 bytes	Intervalo de tiempo	-178000000 years	178000000 years

- **Cadena de bits:** Els tipus de cadenes de bits s'utilitzen per emmagatzemar màscares de bits. Són 0 o 1. Hi ha dos tipus de bits SQL: bit (n) i bit variable (n), on n és un enter positiu.

3.1.1. TIPUS DE DADES PRINCIPALS

- **Moneda:** emmagatzema la quantitat de moneda amb una precisió fraccionaria fixa.

Nombre	Tamaño Almacenaje	Descripción	Rango
money	8 bytes	Cantidad de moneda	-92233720368547758.08 a +92233720368547758.07

- **Enumerats:** són tipus de dades que comprenen un conjunt de valors estàtic i ordenat. Per exemple, adreces de la brúixola, és a dir, NORD, SUD, EST i OEST o dies de la setmana.
- A diferència d'altres tipus, els tipus enumerats s'han de crear utilitzant l'ordre CREATE TYPE, com es mostra a continuació:

```
CREATE TYPE week AS ENUM ('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun');
```

- Una vegada creat, es pot fer servir com qualsevol altre tipus.

3.1. CREATE

- Exemples:

create table libros(

codigo int not null,

titulo varchar(30) unique,

autor varchar(30) not null,

editorial varchar(20),

primary key(codigo)
);

```
CREATE TABLE films (  
  code      CHAR(05),  
  title     VARCHAR(40),  
  did       DECIMAL(03),  
  date_prod DATE,  
  kind      CHAR(10),  
  len       INTERVAL HOUR TO MINUTE,  
  CONSTRAINT code_title PRIMARY KEY(code,title)  
);
```

```
Query Editor  Query History  
1  CREATE TABLE empleados (  
2    ID SERIAL PRIMARY KEY,  
3    NOMBRE varchar(50),  
4    PUESTO varchar(50),  
5    SUELDO integer  
6  );
```

```
CREATE TABLE distributors (  
  did      DECIMAL(3),  
  name     VARCHAR(40)  
  CONSTRAINT con1 CHECK (did > 100 AND name > '')  
);
```

```
CREATE TABLE products (  
  product_no integer,  
  name text,  
  price numeric CHECK (price > 0)  
);
```

```
CREATE TABLE Pedidos (  
  PedidoID int NOT NULL,  
  NumeroPedido int NOT NULL,  
  PersonalID int,  
  PRIMARY KEY (PedidoID),  
  CONSTRAINT FK_PedidoPersona FOREIGN KEY (PersonalID)  
  REFERENCES Personas(PersonalID)  
);
```

3.2. DROP

- **Esborrar base de dades**: Des de pgAdmin4 podem utilitzar l'assistent, fent clic dret sobre la base de dades que volem esborrar i seleccionar l'opció Borrar/Eliminar.
- O bé, des de l'editor de SQL introduir:

1.- Comprovar que no hi ha usuaris connectats:

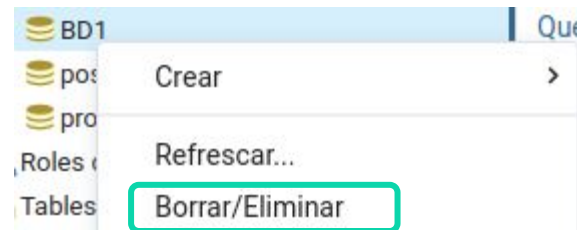
```
SELECT pg_terminate_backend(pg_stat_activity.pid)
```

```
FROM pg_stat_activity
```

```
WHERE pg_stat_activity.datname = 'basedades_a_esborrar'
```

```
AND pid <> pg_backend_pid();
```

2.- Esborrar la base de dades: **DROP DATABASE** basedades_a_esborrar;



3.2. DROP

- **Esborrar una taula**: Des de pgAdmin4 podem utilitzar l'assistent, fent clic dret sobre la taula que volem esborrar i seleccionar l'opció Borrar/Eliminar.
- O bé, des de l'editor de SQL introduir:

DROP TABLE [IF EXISTS] `nom_taula` [CASCADE | RESTRICT];

- **NOTA**: utilitzeu l'opció IF EXISTS per eliminar la taula només si existeix, ja que si elimineu una taula que no existeix, PostgreSQL produirà un error.
- En cas que la taula que vulgueu eliminar s'utilitzi en altres objectes com ara vistes, triggers, funcions i procediments emmagatzemats, la funció DROP TABLE no pot eliminar la taula. En aquest cas, tenim dues opcions:
 - L'opció CASCADE ens permet eliminar la taula i els objectes dependents.
 - L'opció RESTRICT rebutja l'eliminació si hi ha algun objecte que depèn de la taula. L'opció RESTRICT és la predeterminada si no s'especifica explícitament a la sentència DROP TABLE.

3.3. ALTER

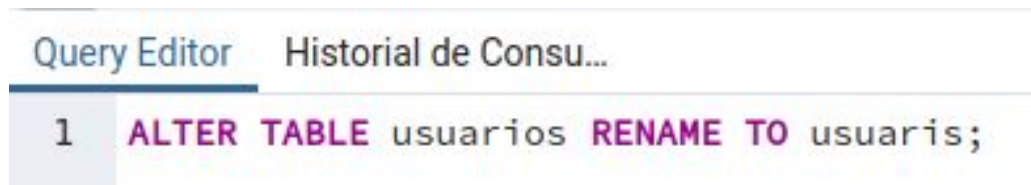
- **Canviar el nom d'una base de dades**
- Si per alguna raó necessitem canviar el nom d'una base de dades PostgreSQL, podem fer-ho amb la comanda **ALTER DATABASE**.
- Els passos que hem de donar són els següents:
- 1. Amb psql o pgAdmin ens connectem a una base de dades que no siga la que desitgem canviar el nom.
- 2. Ens assegurem que no hi ha usuaris connectats a la base de dades. Per a això executem la següent comanda (hauria de mostrar 0 usuaris connectats):
 - `SELECT COUNT(*) AS users_online FROM pg_stat_activity WHERE datname='nom_actual_bd';`
- **NOTA:** si el resultat d'usuaris connectats és distint a 0 (segurament serà 1 perquè estem connectats) podem seleccionar la base de dades, clic amb el botó dret i seleccionar: Desconnectar Base de Datos...
- 3. Reanomenem la base de dades amb la comanda ALTER DATABASE:
- **ALTER DATABASE nom_actual_bd RENAME TO nom_nou_bd;**

3.3. ALTER

- Canviar el nom d'una taula

ALTER TABLE *nom_actual* **RENAME TO** *nou_nom*;

- Exemple:



The screenshot shows a SQL Query Editor window with two tabs: 'Query Editor' and 'Historial de Consu...'. The 'Query Editor' tab is active and displays a single line of SQL code: '1 ALTER TABLE usuarios RENAME TO usuarios;'. The line number '1' is in a light blue box, and the SQL keywords are highlighted in purple.

```
1 ALTER TABLE usuarios RENAME TO usuarios;
```

→ Hem d'estar connectats a la base de dades que la conté!

3.3. ALTER

- Canviar el nom d'una columna

```
ALTER TABLE nom_taula RENAME COLUMN  
actual_nom_columna TO nou_nom_columna;
```

- Exemple:



The screenshot shows a SQL Query Editor window with two tabs: "Query Editor" and "Historial de Consu...". The "Query Editor" tab is active and displays a single line of SQL code: "1 ALTER TABLE usuaris RENAME COLUMN nombre TO nom;". The line number "1" is in a light blue box, and the keywords "ALTER", "TABLE", "RENAME", "COLUMN", and "TO" are highlighted in purple.

```
1 ALTER TABLE usuaris RENAME COLUMN nombre TO nom;
```

→ Hem d'estar connectats a la base de dades que la conté!

3.3. ALTER

- **Altres modificacions**

- Hem vist que podem canviar el nom d'una base de dades, d'una taula i també el nom d'una columna. A més, **podem canviar l'estructura d'una taula** existent.
- Per això utilitzem la sentència PostgreSQL **ALTER TABLE**.
- La sintaxi bàsica de la sentència ALTER TABLE és:
- **ALTER TABLE** acció nom_taula;
 - Veure en la següent transparència les accions possibles

3.3. ALTER

- Altres modificacions
- **ALTER TABLE** acció nom_taula;
- acció pot ser:
 - Afegir una columna
 - Esborrar una columna
 - Canviar el nom d'una columna
 - Canviar el tipus de dades d'una columna
 - Definir un valor per defecte per a la columna
 - Afegir una restricció a una columna
 - Canviar el nom d'una taula

3.3. ALTER

- **Altres modificacions**

- Sintaxi bàsica

- **Afegir una columna:** Per afegir una nova columna a una taula, utilitzeu la instrucció ALTER TABLE ADD COLUMN:
- **ALTER TABLE** nom_taula **ADD COLUMN** nom_columna tipus_dades columna_constraint;
- **Esborrar una columna:** Per esborrar una columna d'una taula, utilitzeu la instrucció ALTER TABLE DROP COLUMN:
- **ALTER TABLE** nom_taula **DROP COLUMN** [IF EXISTS] nom_columna;

3.3. ALTER

- Altres modificacions

- Sintaxi bàsica

- **Canviar el nom d'una columna:** Per canviar el nom d'una columna, com ja hem vist, utilitzeu la instrucció ALTER TABLE RENAME COLUMN TO:

- **ALTER TABLE** nom_taula **RENAME COLUMN** nom_columna
- TO nou_nom_columna;

Query Editor

Historial de Consu...

```
1 ALTER TABLE usuaris RENAME COLUMN nombre TO nom;
```

- **Canviar el tipus de dades d'una columna:** Per canviar el tipus de dades d'una columna, utilitzeu la instrucció ALTER TABLE de la següent manera:
- **ALTER TABLE** nom_taula **ALTER COLUMN** nom_taula [SET DATA] **TYPE** nou_tipus_dades;

3.3. ALTER

- **Altres modificacions**

- Sintaxi bàsica

- Definir un valor per defecte per a la columna: Per canviar un valor per defecte de la columna, utilitzeu ALTER TABLE ALTER COLUMN SET DEFAULT o DROP DEFAULT:

ALTER TABLE nom_taula

ALTER COLUMN nom_columna

[SET DEFAULT valor | DROP DEFAULT];

3.3. ALTER

- **Altres modificacions**

- Sintaxi bàsica

- Per canviar la restricció NOT NULL, utilitzeu la instrucció ALTER TABLE ALTER COLUMN:

ALTER TABLE nom_taula **ALTER COLUMN** nom_columna
[SET NOT NULL | DROP NOT NULL];

- Per afegir una restricció CHECK, utilitzeu la instrucció ALTER TABLE ADD CHECK:

ALTER TABLE nom_taula **ADD CHECK** expressio_a_complir;

3.3. ALTER

- **Altres modificacions**
- Sintaxi bàsica
 - Afegir una restricció a una columna: Generalment, per afegir una restricció a una taula, utilitzeu la instrucció ALTER TABLE ADD CONSTRAINT:

ALTER TABLE nom_taula

ADD CONSTRAINT nom_constraint

definicio_constraint;

- També podem eliminar restriccions introduïdes amb:
- **ALTER TABLE** table_name **DROP CONSTRAINT** nom_constraint;

3.3. ALTER

- **Altres modificacions**
- Sintaxi bàsica
 - En el cas de la restricció de clau forana, la sintaxi és més específica (hem de indicar les accions d'esborrat/modificació).

ALTER TABLE nomtaula1

ADD CONSTRAINT nomrestricció

FOREIGN KEY (campclauforanea)

REFERENCES NOMTAULA2 (campclauprimaria)

ON DELETE ACCIO

ON UPDATE ACCIO;

Les possibles accions són: SET NULL, SET DEFAULT, RESTRICT, NO ACTION (mateix que RESTRICT) i CASCADE.

Per defecte, si no s'especifica cap acció s'aplica NO ACTION.

3.3. ALTER

- Exemples d'afegir restriccions:
 - Afegim una restricció d'unicitat a la columna "correu" de la taula "emails":

ALTER TABLE emails → nom de la taula

ADD CONSTRAINT unique_mail UNIQUE (correu);

nom de la restricció

tipus de restricció

nom de columna que no es pot repetir

3.3. ALTER

- Exemples d'afegir restriccions:
 - Afegim una restricció de clau primària composta a la taula “parquing”:

ALTER TABLE parquing

nom de la
taula

ADD CONSTRAINT fk_parquing

nom de la
restricció

PRIMARY KEY (matricula, horaarribada);

tipus de
restricció

noms de les columnes

3.3. ALTER

- Exemples d'afegir restriccions:
 - Afegim una restricció de clau forana “...” de la taula “llibres”:

ALTER TABLE llibres → nom de la taula

ADD CONSTRAINT FK_llibres_editorial → nom de la restricció

tipus de restricció FOREIGN KEY(codi_editorial) → nom de columna de la taula actual

references editorials(codi) → nom de taula de la qual depèn i la columna a la que volem “apuntar”

ON UPDATE CASCADE

ON DELETE RESTRICT;

3.3. ALTER

- **NOTES:** Si no especifiquem explícitament el nom de la restricció de clau principal, per defecte, PostgreSQL utilitza `nomtaula_pkey` per defecte com a nom per a la restricció de clau principal, `nomtaula_key` per a la restricció de unicitat (UNIQUE),...és a dir, li assigna un nom automàtic.
- En el cas de les claus primàries i foranes no és obligatori posar la clàusula CONSTRAINT a menys que volguem indicar un nom diferent al que PostgreSQL li assignarà per defecte.