

SQL

Master Data Analytics para para la empresa

Pedro Nieto

Agenda

1. SQL Introducción

2. Tipos de Instrucciones

3. DML

1. Select

2. Insert

3. Update

4. Delete

4. DDL

1. Create

2. Drop

3. Alter

5. Joins

6. Public Dataset

SQL Introducción



SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (**RDBMS**), or for stream processing in a relational data stream management system (**RDSMS**). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables.



Google BigQuery

Bigquery el almacén de datos como servicio



Google BigQuery



- ✓ El Almacén de datos corporativos en la nube de Google
- ✓ Escala de Petabyte y la facilidad del SQL
- ✓ Datos Cifrados, Durables y Áltamente disponibles
- ✓ Servicio completamente gestionado

Beneficios de Bigquery



Escalable

Escalabilidad horizontal real, con alto rendimiento para Petabytes de datos

En aplicaciones de producción en Google durante más de una década



Simple

Servicio gestionado: escalado automático de almacenamiento y computación

Analiza tus datos en la plataforma de Google Cloud usando SQL



Compatible

Comparte el acceso a datos y resultados a un grupo mayor de usuarios

Conjuntos de datos públicos y comerciales para enriquecer los análisis



Seguro

Datos cifrados en movimiento y almacenamiento

Gestión de acceso granular con Google Cloud IAM

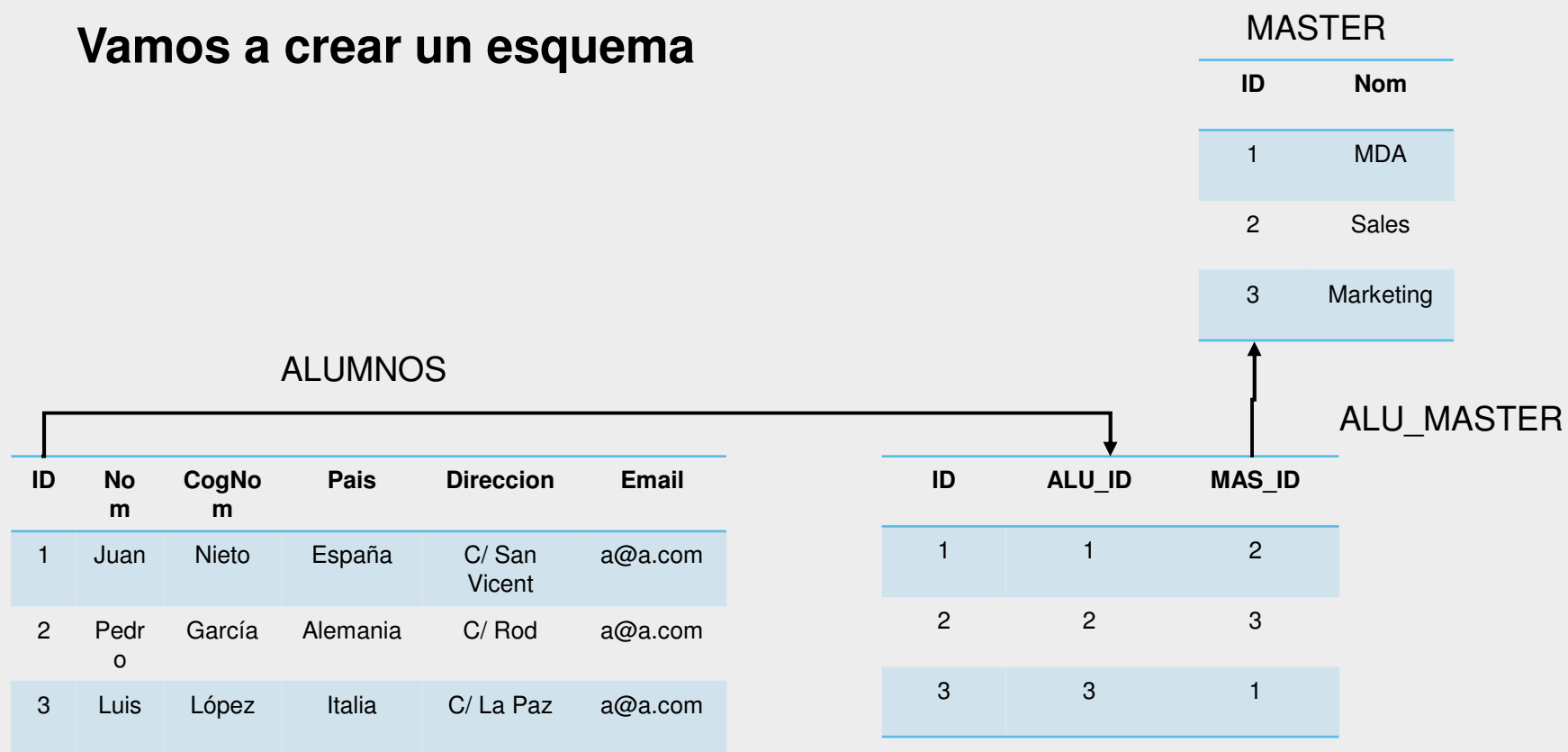


Ahorras

Servicio completamente gestionado, con un Coste de Propiedad bajo para organizaciones de todo tipo

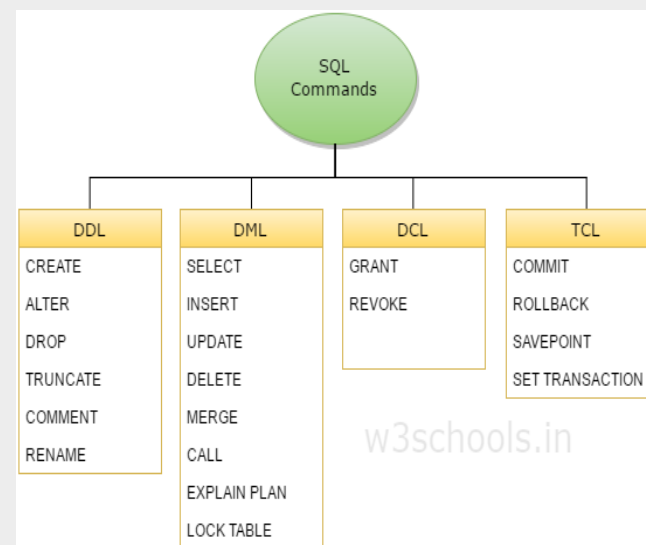
Facturación flexible, con detalle de todos los costes por recurso

Vamos a crear un esquema



Tipos de Operaciones SQL

- Las diferentes operaciones que se pueden realizar en una **base de datos relacional** se pueden agrupar en:
 - DDL (Data Definition Language)**
 - Permite crear y modificar la estructura de una **base de datos**.
 - DML (Data Manipulation Language):**
 - Permite recuperar, almacenar, modificar, eliminar, insertar y actualizar datos de una **base de datos**.
 - DCL (Data Control Language):**
 - Permite crear roles, permisos e integridad referencial, así como el control al acceso a la **base de datos**.
 - TCL (Transactional Control Language):**
 - Permite administrar diferentes transacciones que ocurren dentro de una **base de datos**.



DML - Data Manipulation Language

- Permite recuperar, almacenar, modificar, eliminar, insertar y actualizar datos de una **base de datos**.
 - **SELECT**: Utilizado para consultar registros de la **base de datos** que satisfagan un criterio determinado.
 - **INSERT**: Utilizado para cargar de datos en la **base de datos** en una única operación.
 - **UPDATE**: Utilizado para modificar los valores de los campos y registros especificados
 - **DELETE**: Utilizado para eliminar registros de una tabla de una **base de datos**.



Sintaxis SELECT

- **SELECT:** Utilizado para consultar registros de la **base de datos** que satisfagan un criterio determinado.
- **ORDER BY**
 - **ASC:** De menor a mayor
 - **DESC:** De mayor a menor
- **SELECT DISTINCT y SELECT DISTINCT ON**
- **WHERE**
 - **AND y OR**
 - **= y <>**
 - **IN y NOT IN**
- **LIKE y NOT LIKE**
 - Operadores: **% y _**
- **BETWEEN**



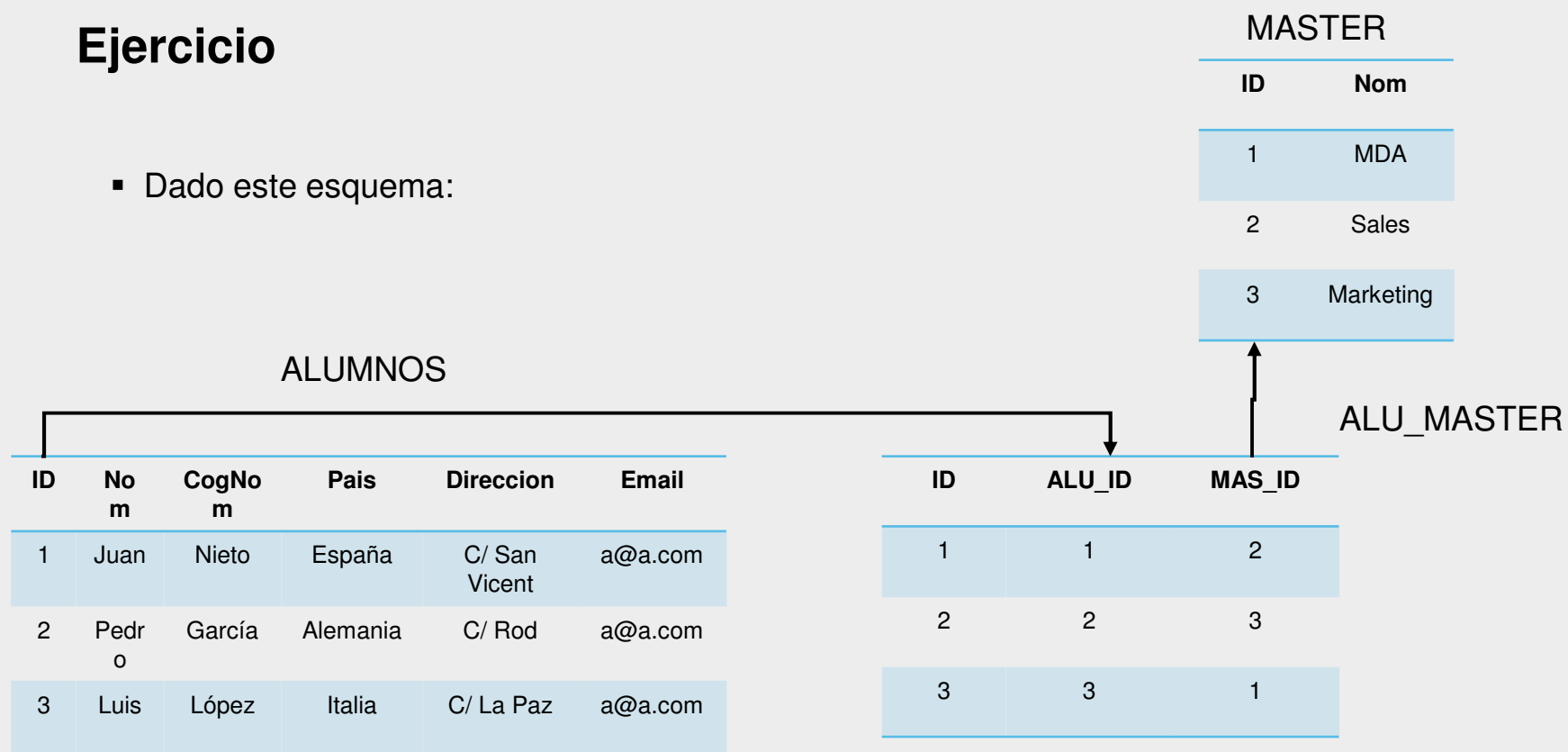
***“ Select * From Customers where
first_name like “%Ant%”***

Ejemplos

- `SELECT NOM,PAIS FROM EDEM.ALUMNOS;`
- `SELECT NOM FROM EDEM.ALUMNOS WHERE ID BETWEEN 1 AND 10;`
- `SELECT NOM FROM EDEM.ALUMNOS WHERE NOM LIKE '%O%';`
- `SELECT NOM FROM EDEM.ALUMNOS WHERE ID IN (1,2,3);`

Ejercicio

- Dado este esquema:



Ejercicio

- Proporciona una Select que de los Alumnos de Portugal
- Proporciona una Select de los Masters que lleven una D
- Proporciona una Select con los Alumnos cuyo ID sea 37 y 45

Sintaxis INSERT

- **INSERT:** Utilizado para cargar de datos en la **base de datos** en una única operación.
- All columns by order:
`INSERT INTO table_name VALUES(data1, data2, ...)`
- Specific columns:
`INSERT INTO table_name(field1, field2) VALUES(data1, data2, ...)`

COMMIT



```
INSERT INTO student VALUES(101, 'Adam', 15);
```


Ejemplos

- insert into EDEM.ALUMNOS (id, Nom, CogNom, Pais, Direccion, Email) values (1001, "PEDRO", "Robottham", "Philippines", "3620 Graedel Court", "mrobotthamrr@reddit.com");
- insert into EDEM.ALUMNOS (id, Nom) values (1002, "PEDRO");

Ejercicio

- Dado este esquema:

ALUMNOS

ID	No m	CogNo m	Pais	Direccion	Email
1	Juan	Nieto	España	C/ San Vicent	a@a.com
2	Pedr o	García	Alemania	C/ Rod	a@a.com
3	Luis	López	Italia	C/ La Paz	a@a.com

MASTER

ID	Nom
1	MDA
2	Sales
3	Marketing

ALU_MASTER

ID	ALU_ID	MAS_ID
1	1	2
2	2	3
3	3	1

Ejercicio

- Genera tu Insert para incluir en la base de datos

Sintaxis UPDATE

- **UPDATE:** Utilizado para actualizar registros de la **base de datos** que satisfagan un criterio determinado.

- One column:

```
UPDATE table_name SET column_name = new_value WHERE  
some_condition;
```

- Multiple Columns:

```
UPDATE table_name SET column_name = new_value, column_name =  
new_value WHERE some_condition;
```

- Using custom values:

```
UPDATE table_name SET column_name = column_name + 1;
```



```
UPDATE student SET age=18 WHERE  
student_id=102;
```

Ejemplos

- UPDATE EDEM.ALUMNOS SET NOM="ESTEBAN" WHERE NOM="PEDRO";
- UPDATE EDEM.ALUMNOS SET NOM="ESTEBAN" WHERE ID IN (SELECT ID FROM EDEM.ALUMNOS WHERE NOM="PEDRO");
- UPDATE EDEM.ALUMNOS SET NOM="ROBERTO",PAIS="CHINA" WHERE NOM="ESTEBAN";

Ejercicio

- Dado este esquema:

ALUMNOS

ID	No m	CogNo m	Pais	Direccion	Email
1	Juan	Nieto	España	C/ San Vicent	a@a.com
2	Pedr o	García	Alemania	C/ Rod	a@a.com
3	Luis	López	Italia	C/ La Paz	a@a.com

MASTER

ID	Nom
1	MDA
2	Sales
3	Marketing

ALU_MASTER

ID	ALU_ID	MAS_ID
1	1	2
2	2	3
3	3	1

Ejercicio

- Actualiza tu país de origen a Mexico
- Inserta una fila con tu asistencia al master de MDA

Sintaxis DELETE

- **DELETE:** Utilizado para eliminar registros de la **base de datos** que satisfagan un criterio determinado.
- No filters:
`DELETE FROM table_name;`
- With Filter:
`DELETE FROM table_name where column_name>1;`

COMMIT



DELETE FROM student WHERE s_id=103;

Ejemplos

- DELETE FROM EDEM.ALUMNOS WHERE NOM="PEDRO";
- DELETE FROM EDEM.ALUMNOS WHERE ID IN (SELECT ID FROM EDEM.ALUMNOS WHERE NOM="PEDRO");
- DELETE FROM EDEM.ALUMNOS WHERE ID>1000;

Ejercicio

- Dado este esquema:

ALUMNOS

ID	No m	CogNo m	Pais	Direccion	Email
1	Juan	Nieto	España	C/ San Vicent	a@a.com
2	Pedr o	García	Alemania	C/ Rod	a@a.com
3	Luis	López	Italia	C/ La Paz	a@a.com

MASTER

ID	Nom
1	MDA
2	Sales
3	Marketing

ALU_MASTER

ID	ALU_ID	MAS_ID
1	1	2
2	2	3
3	3	1

Ejercicio

- Borra todos los alumnos que se llamen Juan

Combina tus DML

- **INSERT + SELECT**

INSERT INTO Table_Name SELECT * FROM Table_Name2;

- **DELETE + SELECT**

DELETE FROM Table_Name where Id in (select ID from Table_Name)

- **SELECT + SELECT**

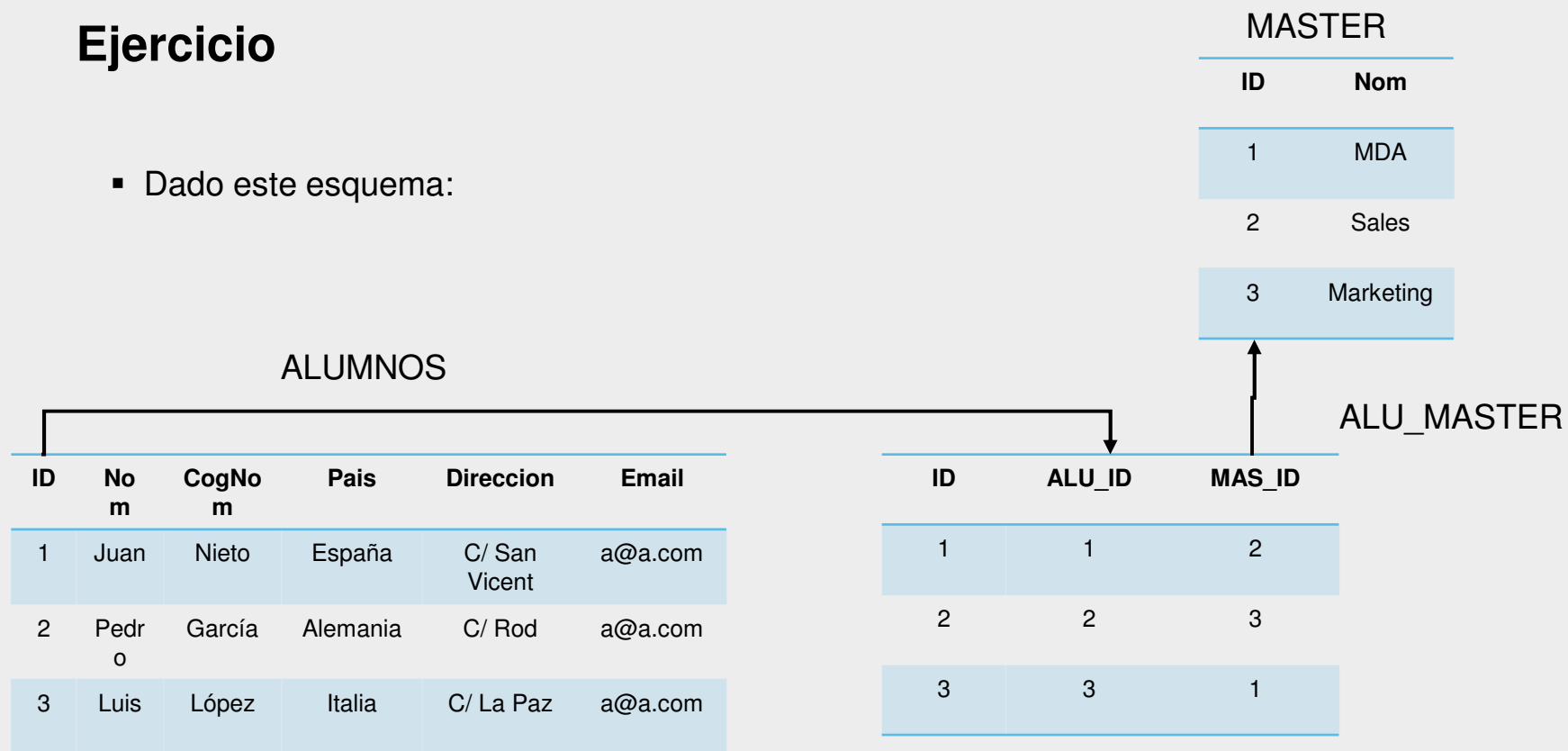
SELECT * FROM Table_Name where Id in (select ID from Table_Name)

- **UPDATE + SELECT**

UPDATE table_name SET column_name = new_value WHERE Id in (select ID from Table_Name);

Ejercicio

- Dado este esquema:



Ejercicio

- Borra los alumnos que no asistan a ningún master
- Borra los masters que no tengan alumnos
- Actualiza a Nulo los emails de los alumnos del master de MDA

DDL - Data Definition Language

- Un Data Definition Language o Lenguaje de descripción de datos (DDL) es un lenguaje de programación para definir [estructura de datos](#) .
 - **CREATE**: Sirve para crear una nueva base de datos, tabla, índice, o procedimiento almacenado..
 - **DROP**: Sirve para borrar en forma sencilla distintos objetos dentro del [SGBD] como por ejemplo base de datos, tablas, índices.
 - **ALTER**: La sentencia ALTER TABLE es usada para agregar, borrar o modificar columnas en una tabla existente



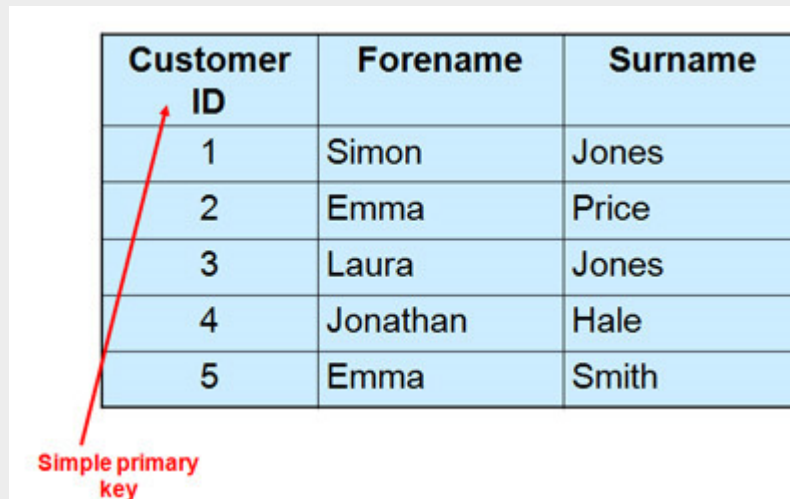
DataTypes

	postgresql	sqlite	sqlserver	sybase
:binary	bytea	blob	image	image
:boolean	boolean	boolean	bit	bit
:date	date	date	date	datetime
:datetime	timestamp	datetime	datetime	datetime
:decimal	decimal	decimal	decimal	decimal
:float	float	float	float(8)	float(8)
:integer	integer	integer	int	int
:string	(note 1)	varchar(255)	varchar(255)	varchar(255)
:text	text	text	text	text
:time	time	datetime	time	time
:timestamp	timestamp	datetime	datetime	timestamp

Casi todo es standard...

Constraints

- **Primary key: (PK):** A primary key is a column or a group of columns used to identify a row uniquely in a table.



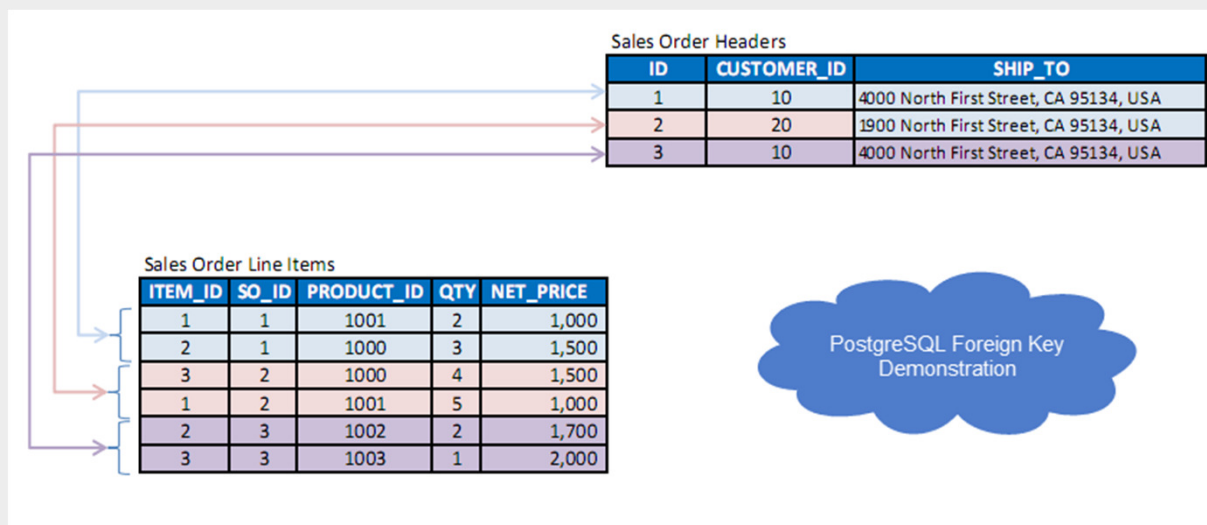
A diagram illustrating a primary key constraint. It features a table with three columns: 'Customer ID', 'Forename', and 'Surname'. The 'Customer ID' column contains values 1, 2, 3, 4, and 5. A red arrow points from the text 'Simple primary key' to the 'Customer ID' column header, indicating that this column is the primary key.

Customer ID	Forename	Surname
1	Simon	Jones
2	Emma	Price
3	Laura	Jones
4	Jonathan	Hale
5	Emma	Smith

Simple primary key

Constraints

- **Foreign KEY (FK):** A foreign key is a field or group of fields in a table that uniquely identifies a row in another table. In other words, a foreign key is defined in a table that references to the primary key of the other table.



Constraints

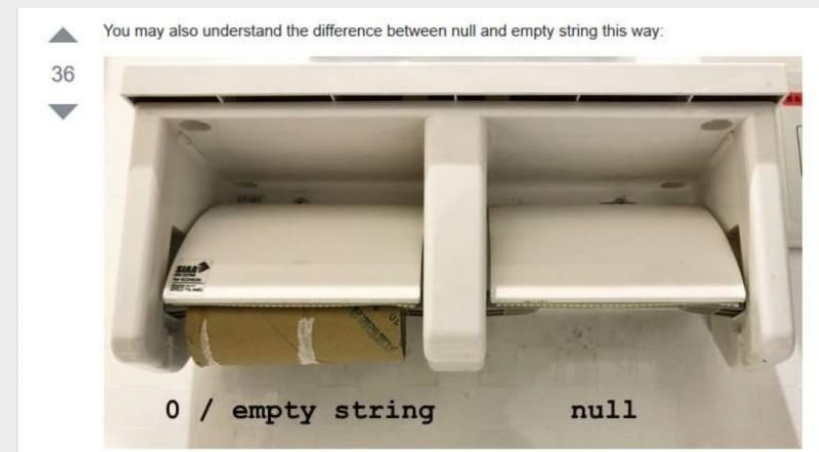
- **UNIQUE:** Sometimes, you want to ensure that values stored in a column or a group of columns are unique across the whole table such as email address and username. PostgreSQL provides you with the UNIQUE constraint to make sure that the uniqueness of the data is maintained correctly.

Student ID	First Name	Last Name	Email	Major	Faculty
200120	Kate	West	kwest@email.com	Music	Arts
200121	Julie	McLain	jmclain@email.com	Finance	Business
200122	Tom	Erlich	terlich@email.com	Sculpture	Arts
200123	Mark	Smith	msmith@email.com	Biology	Science
200124	Jen	Foster	jfoster@email.com	Physics	Science
200125	Matt	Knight	mknight@email.com	Finance	Business
200126	Karen	Weaver	kweaver@email.com	Music	Arts
200127	John	Smith	jsmith@email.com	Sculpture	Arts
200128	Allison	Page	apage@email.com	History	Humanities
200129	Craig	Cambell	ccambell@email.com	Music	Arts
200130	Steve	Edwards	sedwards@email.com	Biology	Science
200131	Mike	Williams	mwilliams@email.com	Linguistics	Humanities
200132	Jane	Reid	jreid@email.com	Music	Arts

Constraints

- **NOT NULL:** In database theory, NULL is unknown or missing information. The NULL value is different from an empty string or number zero. For example, you can ask a person for an email address, if you don't know, you use the **NULL** value for inserting into the email column. This indicates that the information at the time of inserting is unknown. In case the person does not have any email address, you can update it to an empty string.

The NULL value is very special. For example, NULL is not equal to anything even NULL. To check if a value is NULL or not, you use the Boolean operator **IS NULL** or **IS NOT NULL**. The expression `NULL = NULL` returns NULL.



Sintaxis CREATE

- Hay dos Statements disponibles de CREATE en SQL:
 - CREATE DATABASE (no la veremos)
 - CREATE TABLE

```
CREATE TABLE table_name  
(  
column1 data_type(size) constraint,  
column2 data_type(size) constraint,  
column3 data_type(size) constraint,  
....  
);
```



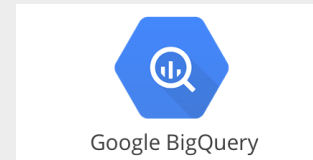
```
CREATE TABLE account(  
user_id serial PRIMARY KEY,  
username VARCHAR (50) UNIQUE NOT NULL,  
password VARCHAR (50) NOT NULL,  
email VARCHAR (355) UNIQUE NOT NULL,  
created_on TIMESTAMP NOT NULL,  
last_login TIMESTAMP  
);
```

Ejemplos



```
CREATE TABLE role(  
  role_id serial PRIMARY KEY,  
  role_name VARCHAR (255) UNIQUE NOT NULL  
);
```

```
CREATE TABLE account(  
  user_id serial PRIMARY KEY,  
  username VARCHAR (50) UNIQUE NOT NULL,  
  password VARCHAR (50) NOT NULL,  
  email VARCHAR (355) UNIQUE NOT NULL,  
  created_on TIMESTAMP NOT NULL,  
  last_login TIMESTAMP  
);
```



```
CREATE TABLE EDEM.role(  
  role_id int64,  
  role_name STRING  
);
```

```
CREATE TABLE EDEM.account(  
  user_id int64,  
  username STRING,  
  password STRING,  
  email STRING,  
  created_on DATETIME,  
  last_login DATETIME  
);
```

Ejercicio

- Crea la siguiente tabla en tu schema:
 - ID: Clave primaria único
 - Mes: No nulo
 - Cantidad: No nulo
 - Descripcion: Más de 250 caracteres

Id	Mes	Cantidad	Descripcion
1	Enero	100.4	Cuota Mensual
2	Febrero	240	Gasto

Sintaxis DROP

- Hay dos Statements disponibles de DROP en SQL:
 - DROP DATABASE (no la veremos)
 - DROP TABLE

**DROP TABLE [IF EXISTS] table_name
[CASCADE | RESTRICT];**

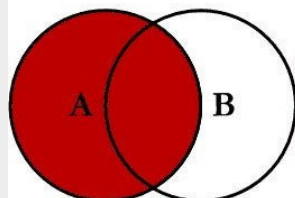


DROP TABLE IF EXISTS account;

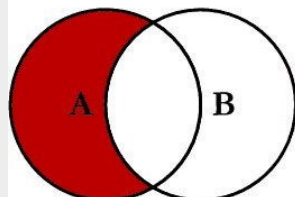
Ejercicio

- Genera un script que borre todas tablas del schema:
- **NO EJECUTEIS!!!!**

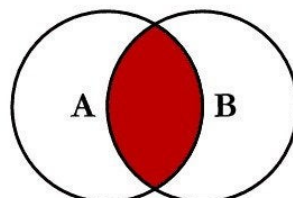
SQL JOINS



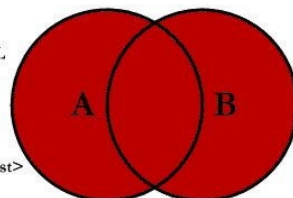
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



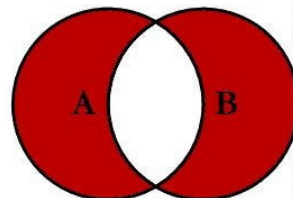
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



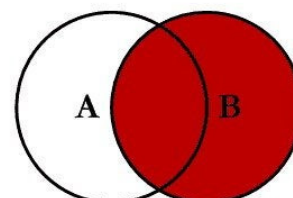
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



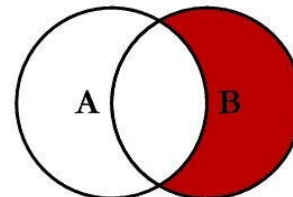
```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

© C.L. Moffatt, 2008



**DO
NOT
PANIC**

Ayuda

- Las siguientes opciones os harán la vida más fácil a la hora de construir una join:

Visual JOIN

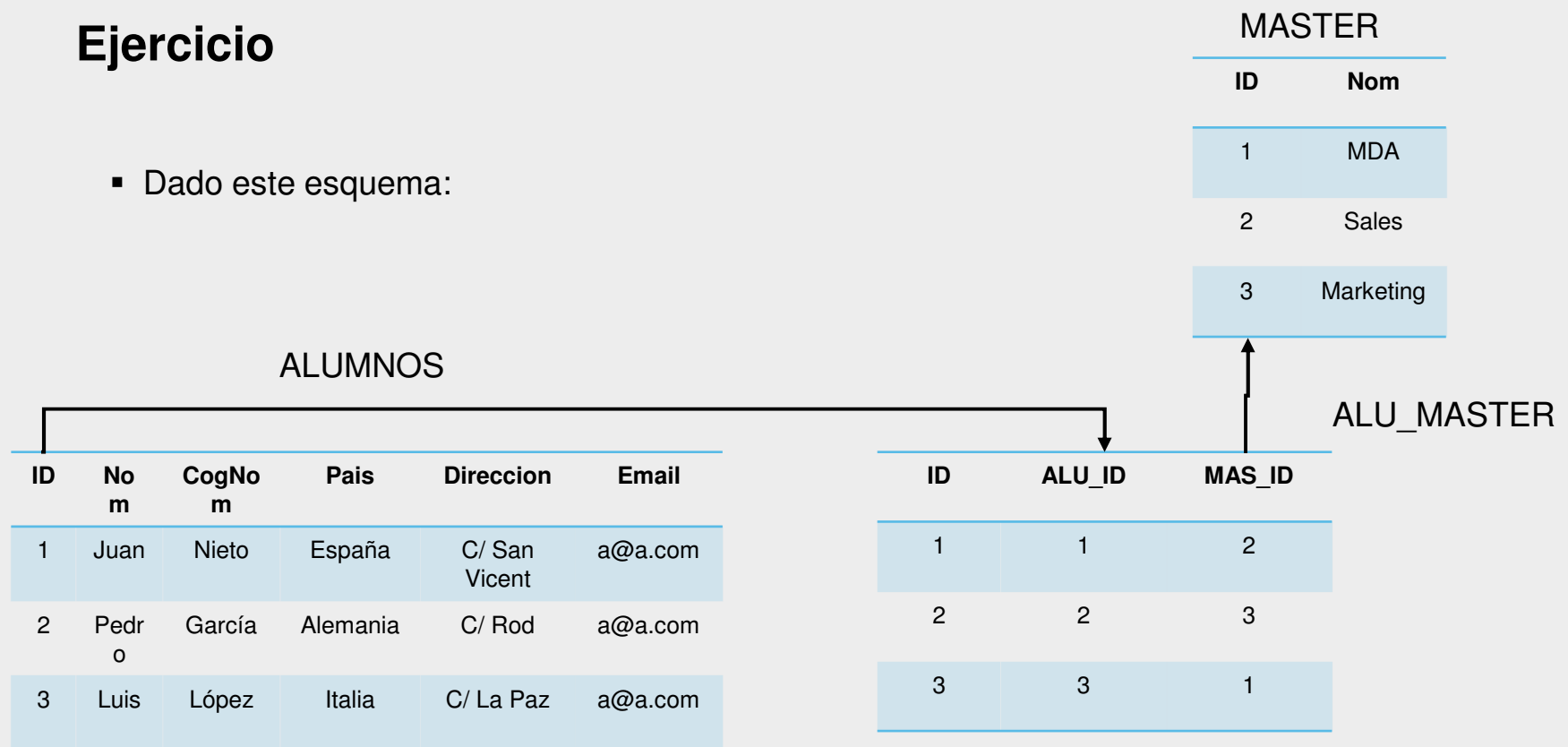
Understand how joins work by interacting and see it visually

SQL Joins Visualizer

SQL Joins Visualizer help to you build SQL JOIN between two tables by using of Venn diagrams.
Working offline and as mobile app.

Ejercicio

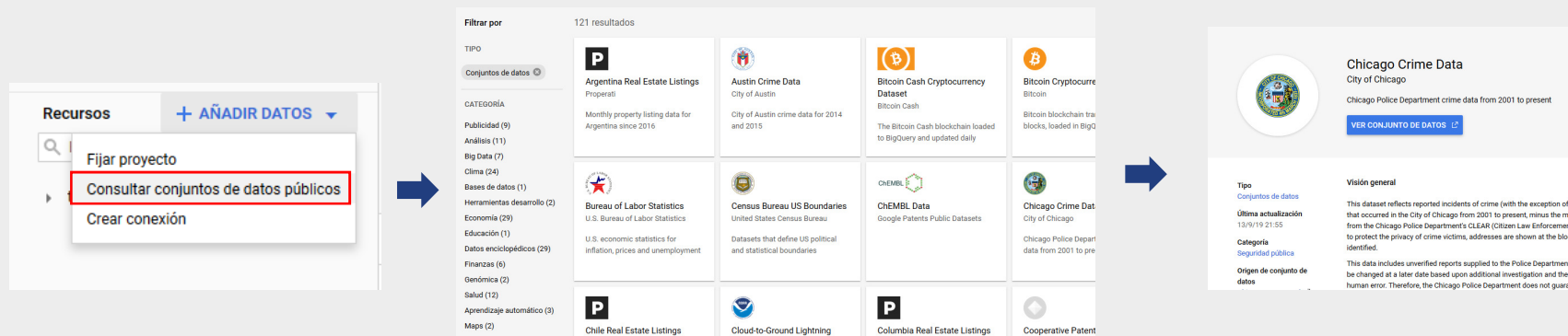
- Dado este esquema:



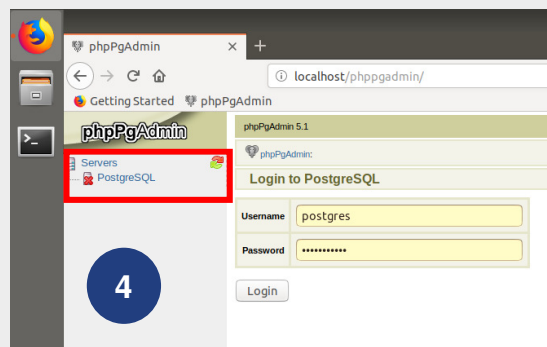
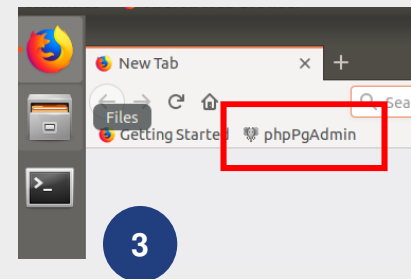
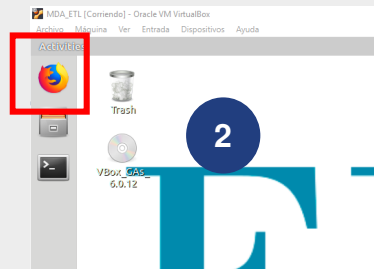
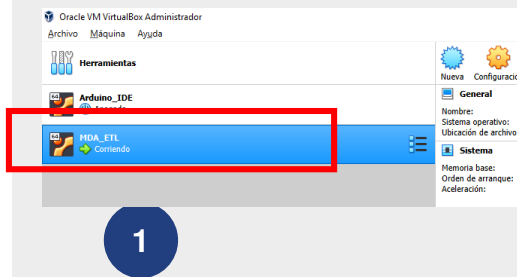
Example:

- Query que haga Join y muestre cada alumno el master que tiene asociado

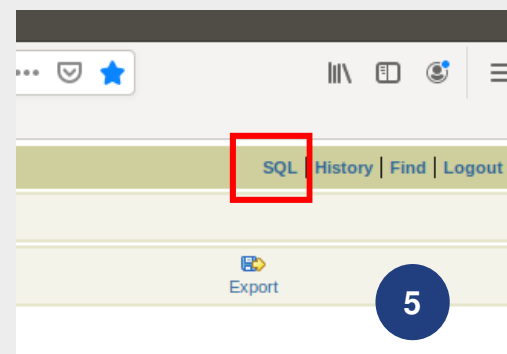
Dataset Publicos



Ejercicios de Repaso



postgres / postgres123



DVD Rental Database

Hagamos unos pocos de Ejercicios

- 1. Proporciona una SQL que muestre los siguientes datos:**
 - Nombre Actor
 - Apellido Actor
- 2. Proporciona una SQL que muestre los siguientes datos:**
 - Nombre Actor
 - Título de la Película
- 3. Proporciona una SQL que muestre los siguientes datos:**
 - Nombre Actor
 - Número de películas
 - Ordenar de mayor a menor
- 4. Proporciona una SQL que muestre los siguientes datos:**
 - Película
 - Numero de veces alquilada
- 5. Proporciona una SQL que muestre los siguientes datos:**
 - Película
 - Dinero recaudado por película
- 6. Proporciona una SQL que muestre los siguientes datos:**
 - Nombre del mejor cliente (mayor gasto)
- 7. Proporciona una SQL que muestre los siguientes datos:**
 - Nombre del mejor cliente (mayor num alquileres)