dadavalangege / itinerario_itacademy_ds_apuntes Private

Code Issues Pull requests Actions Projects Security Insight

main

itinerario_itacademy_ds_apuntes / Bases_de_datos.ipynb

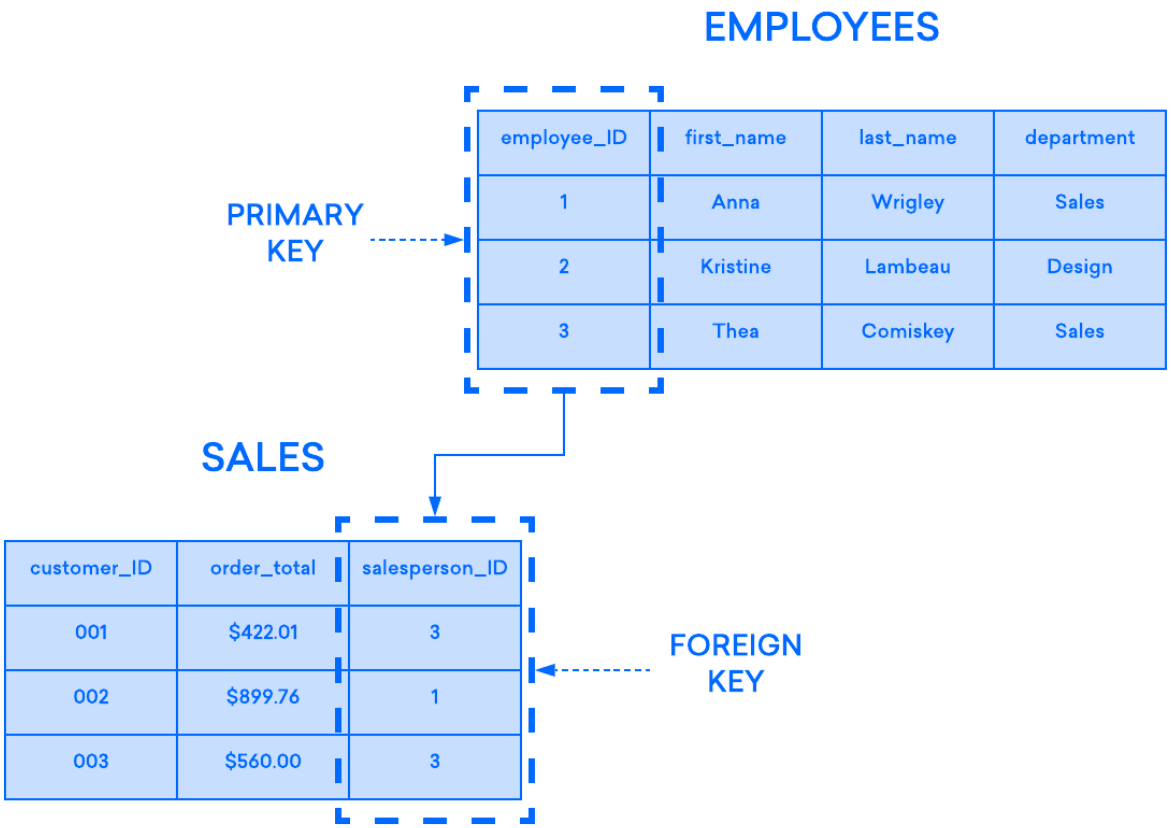
dadavalangege 19/10 History

1 contributor

1.57 MB

SQL

https://www.digitalocean.com/community/tutorials/understanding-relational-databases (https://www.digitalocean.com/community/tutorials/understanding-relational-databases)



Advantages and Limitations of Relational Databases

Horizontal scaling, or scaling out, is the practice of adding more machines to an existing stack in order to spread out the load and allow for more traffic and faster processing. This is often contrasted with vertical scaling which involves upgrading the hardware of an existing server, usually by adding more RAM or CPU.

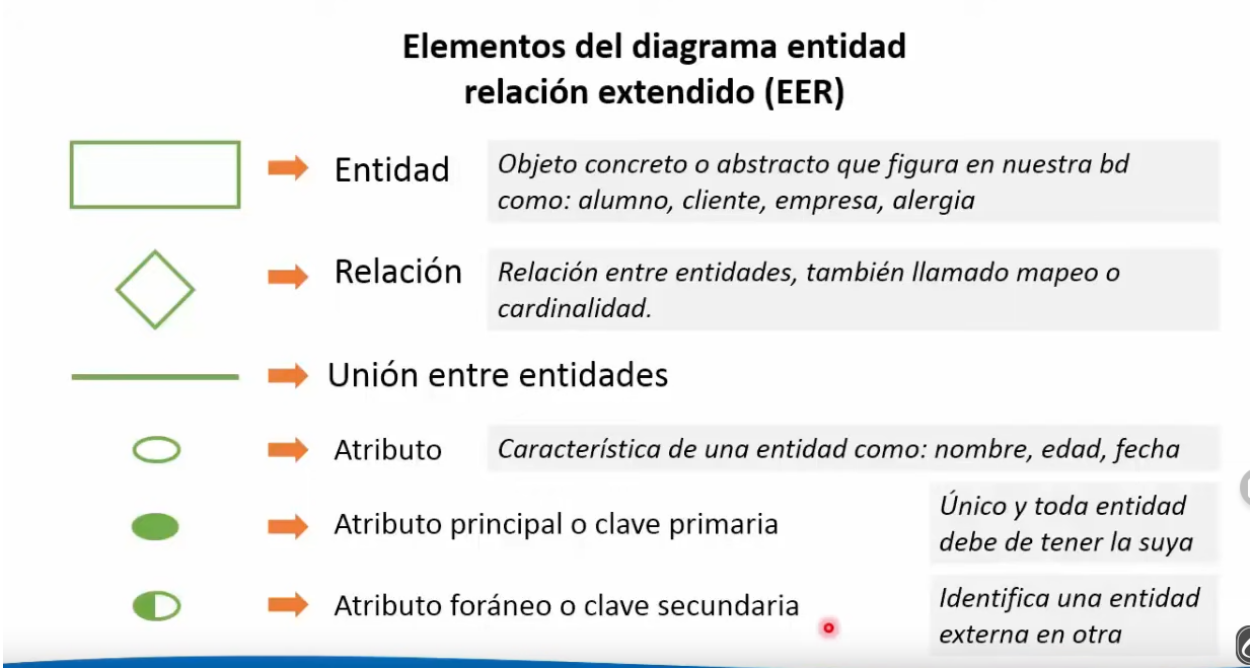
The reason it's difficult to scale a relational database horizontally has to do with the fact that the relational model is designed to ensure consistency, meaning clients querying the same database will always retrieve the same data. If you were to scale a relational database horizontally across multiple machines, it becomes difficult to ensure consistency since clients may write data to one node but not the others. There would likely be a delay between the initial write and the time when the other nodes are updated to reflect the changes, resulting in inconsistencies between them.

Another advantage of relational databases is that almost every RDBMS supports transactions. A transaction consists of one or more individual SQL statements performed in sequence as a single unit of work. Transactions present an all-or-nothing approach, meaning that every SQL statement in the transaction must be valid; otherwise, the entire transaction will fail. This is very helpful for

https://ioc.xtec.cat/materials/FP/Recursos/fp_dam_m02_/web/fp_dam_m02_htmlindex/WebContent/u2/a1/continguts.html (https://ioc.xtec.cat/materials/FP/Recursos/fp_dam_m02_/web/fp_dam_m02_htmlindex/WebContent/u2/a1/continguts.html)

https://ioc.xtec.cat/materials/FP/Recursos/fp_dam_m02_/web/fp_dam_m02_htmlindex/media/fp_dam_m02_u3_pdfindex.pdf (https://ioc.xtec.cat/materials/FP/Recursos/fp_dam_m02_/web/fp_dam_m02_htmlindex/media/fp_dam_m02_u3_pdfindex.pdf)

Modelo entidad relación / relacional / normalización



Las entidades y las relaciones se determinan en función de las **reglas del negocio**, es decir, a partir de las necesidades de la empresa.

Ejercicio diagrama entidad-relación

Cardinalidad o mapeo

Mapeo

- Relación uno a uno entre entidades
- Uno a muchos (**1:N**)
- Muchos a uno
- Muchos a muchos (**N:M**)

Cuando la relación es uno a muchos se asigna la CLAVE PRIMARIA de UNO como CLAVE

Cuando tenemos una relación muchos a muchos se crea una TABLA INTERMEDIA. La tabla intermedia va a tener DOS CLAVES FORÁNEAS de las claves primarias de las dos entidades de la relación.

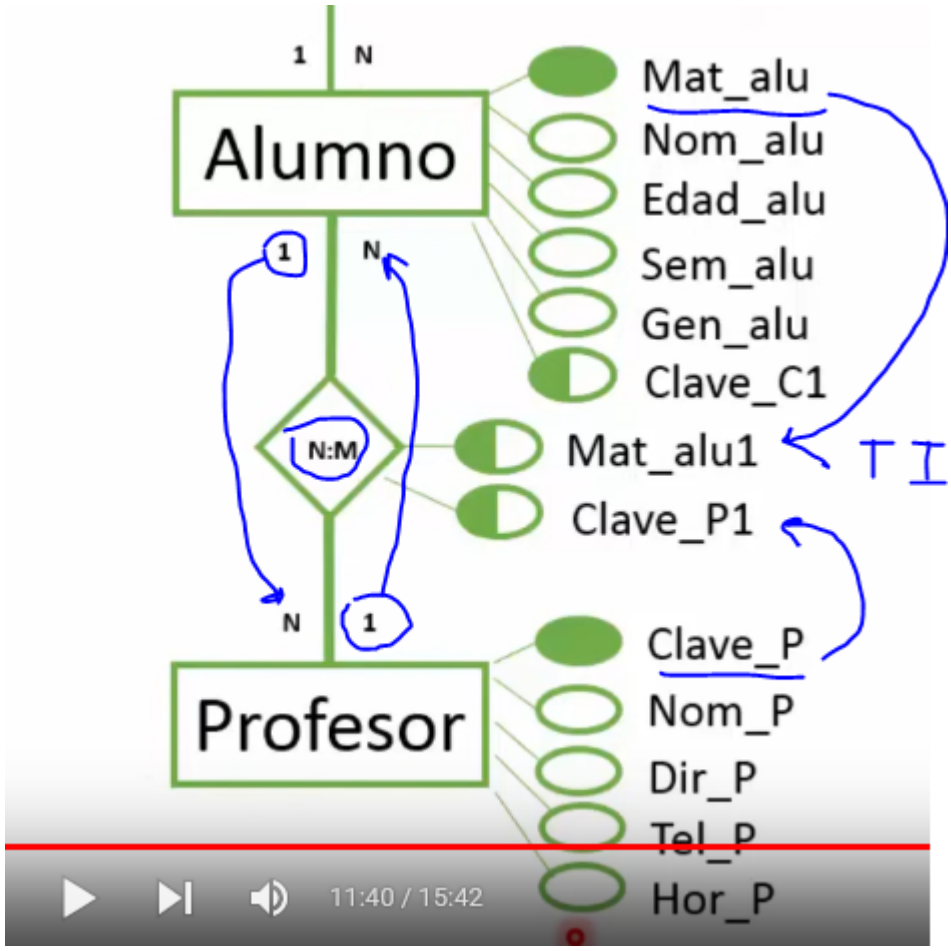


Diagrama entidad-relación extendido

Atributos en las tablas intermedias.

Normalización

Diagramas con MySQL Workbench

https://guru99.es/er-modeling/ (https://guru99.es/er-modeling/)

SQL

W3SCHOOLS / PLATZI

- https://www.w3schools.com/sql/sql_intro.asp (https://www.w3schools.com/sql/sql_intro.asp)

Some of The Most Important SQL Commands

```
SELECT - extracts data from a database
UPDATE - updates data in a database

        UPDATE table_name
        SER col1=value1, col2=val2, ...
        WHERE condition

DELETE - deletes data from a database, elimina entradas

        DELETE FROM col WHERE condition;

        DELETE FROM col WHERE 1=1 #elimina todas las entradas sin e
liminar la tabla de la base de datos

INSERT INTO - inserts new data into a database

        INSERT INTO table_name (col, col, col...)
        VALUES (value, value, value...)
```

```
        ALTER TABLE table_name
        MODIFY COLUMN column_name datatype;

DROP TABLE - deletes a table
CREATE INDEX - creates an index (search key)
DROP INDEX - deletes an index
```

BACUP DATABASE

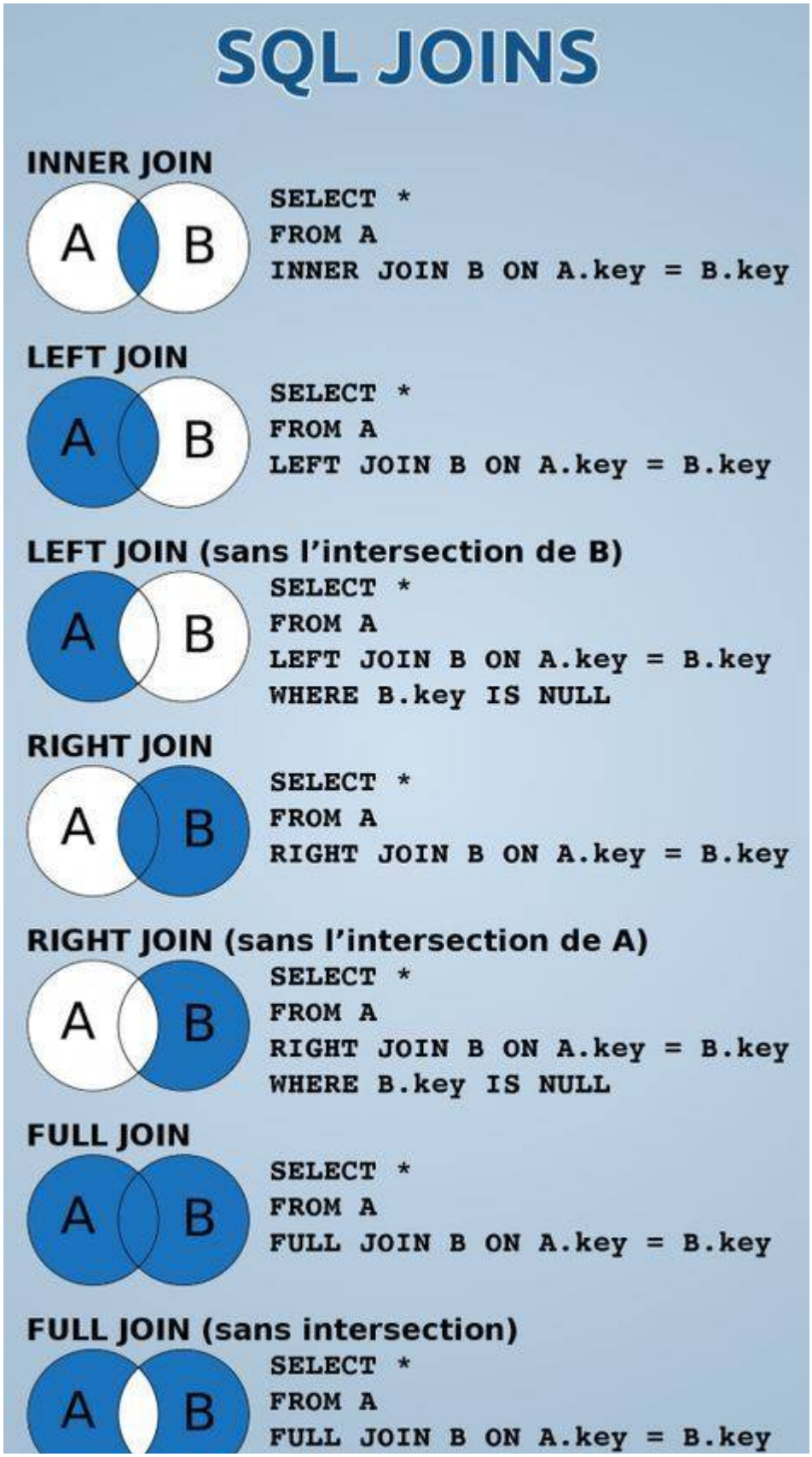
```
BACKUP DATABASE databasename
TO DISK = 'filepath';

BACKUP DATABASE databasename
TO DISK = 'filepath'
WITH DIFFERENTIAL;
```

SELECT

```
SELECT field AS alias;
```

Se puede hacer from desde diferentes bases de datos (extra, no copiado el código).

JOIN



```
FROM tabla

WHERE id = 1;

WHERE cantidad > 10
    AND cantidad < 100;

WHERE cantidad BETWEEN 10 #NOT BETWEEN
    AND 100;

WHERE name = x
    AND (
```

```
ASC/DESC
```

IN / NOT IN

```
#selects all customers that are located in "Germany", "France" or "
```

```
DROP VIEW [view_name];
```