
Bases de datos clase 19: Torpedo

Leonardo Bravo Illanes
Escuela de Informática y Telecomunicaciones
Universidad Diego Portales

CRUD en SQL



CRUD

CREATE

```
INSERT INTO video (titulo, media, descrip, fecha)
values (
    'Arch Enemy - Sunset Over The Empire',
    'https://youtu.be/E3mOvCMG24I',
    'New single "Sunset Over The Empire" out everywhere!',
    '2022-05-25');
```

```
test2=# INSERT INTO video (titulo, media, descrip, fecha) values ('Arch Enemy - Sunset Over The Empire', 'https://youtu.be/E3mOvCMG24I', 'New single "Sunset Over The Empire" out everywhere!', '2022-05-25');
```

```
INSERT 0 1
```

```
test2=# select * from video;
```

id_video	titulo	media	descrip	fecha	place
1	Arch Enemy - Sunset Over The Empire	https://youtu.be/E3mOvCMG24I	New single "Sunset Over The Empire" out everywhere!	2022-05-24	
2	Arch Enemy - Sunset Over The Empire	https://youtu.be/E3mOvCMG24I	New single "Sunset Over The Empire" out everywhere!	2022-05-25	
3	Arch Enemy - Sunset Over The Empire	https://youtu.be/E3mOvCMG24I	New single "Sunset Over The Empire" out everywhere!	2022-05-25	

```
(3 rows)
```

CRUD

READ

```
SELECT id_video, titulo, media, to_char(fecha, 'YYYY-MM-dd') as fecha  
FROM video;
```

(3 rows)

```
test2=# select id_video, titulo, media, to_char(fecha, 'YYYY-MM-dd') as fecha from video;
```

id_video	titulo	media	fecha
1	Arch Enemy - Sunset Over The Empire	https://youtu.be/E3m0vCMG24I	2022-05-24
3	Arch Enemy - Sunset Over The Empire	https://youtu.be/E3m0vCMG24I	2022-05-25
2	Hans Zimmer performs INCEPTION "Time"	https://youtu.be/xdYYN-4ttDg	2022-05-25

(3 rows)

CRUD

UPDATE

```
UPDATE video
  set titulo='Hans Zimmer performs INCEPTION "Time"',
      media='https://youtu.be/xdYYN-4ttDg',
      descrip='at Hollywood in Vienna 2018'
 where id_video=2;
```

```
test2=# UPDATE video set titulo='Hans Zimmer performs INCEPTION "Time"', media='https://youtu.be/xdYYN-4ttDg', descrip='at Hollywood in Vienna 2018' where id_video=2;
UPDATE 1
```

```
test2=# select * from video;
```

id_video	titulo	media	descrip	fecha	place
1	Arch Enemy - Sunset Over The Empire	https://youtu.be/E3mOvCMG24I	New single "Sunset Over The Empire" out everywhere!	2022-05-24	
3	Arch Enemy - Sunset Over The Empire	https://youtu.be/E3mOvCMG24I	New single "Sunset Over The Empire" out everywhere!	2022-05-25	
2	Hans Zimmer performs INCEPTION "Time"	https://youtu.be/xdYYN-4ttDg	at Hollywood in Vienna 2018	2022-05-25	

```
(3 rows)
```

CRUD

DELETE

```
DELETE
```

```
FROM student
```

```
WHERE stud_id = 11
```

```
AND stud_name = PQR;
```

Cosas que pueden salvar vidas



Cosas que pueden salvar vidas

Vista

Corresponden a una forma de almacenar en el servidor alguna query de selección en SQL para poder utilizarla como una relación. **NO ES UNA TABLA**

```
CREATE VIEW nombre_vista AS
```

```
SELECT A1,...,Ak
```

```
FROM t1,t2,t3
```

```
WHERE p
```


¿Que es ACID o una base de datos ACIDA?



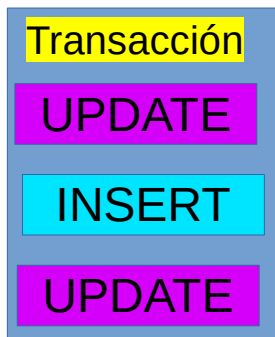
A tener presente

ACID o Base de Datos Acida

El acrónimo para los conceptos Atomicity, Consistency, Isolation y Durability

Atomicidad:

Esta propiedad indica que, para que una transacción se dé por «completada», deben haberse realizado todas sus partes o ninguna de ellas.



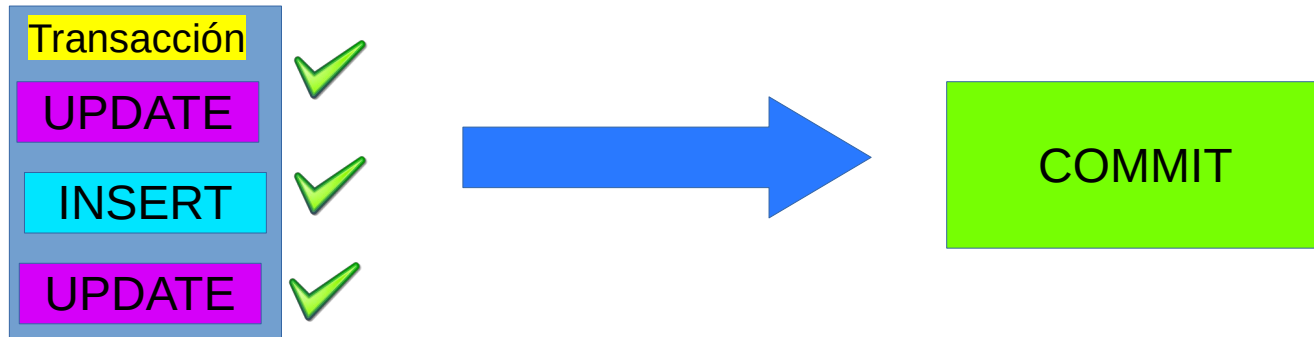
A tener presente

ACID o Base de Datos Acida

El acrónimo para los conceptos Atomicity, Consistency, Isolation y Durability

Atomicidad:

Esta propiedad indica que, para que una transacción se dé por «completada», deben haberse realizado todas sus partes o ninguna de ellas.



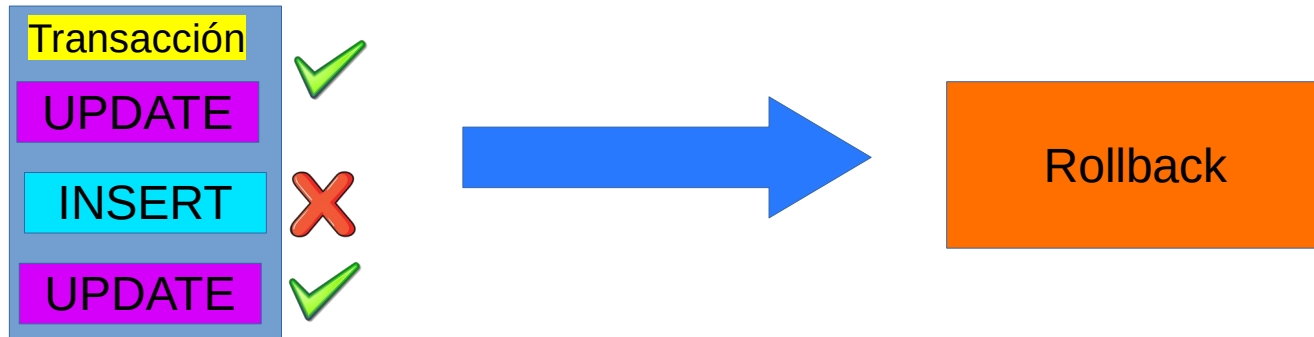
A tener presente

ACID o Base de Datos Acida

El acrónimo para los conceptos Atomicity, Consistency, Isolation y Durability

Atomicidad:

Esta propiedad indica que, para que una transacción se dé por «completada», deben haberse realizado todas sus partes o ninguna de ellas.



A tener presente

ACID

Atomicidad:

```
BEGIN;  
  
INSERT INTO cuentas (n_cuenta, nombre, balance) VALUES (0679259, 'Pepe', 200);  
  
UPDATE cuentas SET balance = balance - 137.00 WHERE nombre = 'Pepe';  
  
UPDATE cuentas SET balance = balance + 137.00 WHERE nombre = 'Juan';  
  
SELECT nombre, balance FROM cuentas WHERE nombre = 'Pepe' AND nombre = 'Juan';  
  
COMMIT;
```

A tener presente

ACID

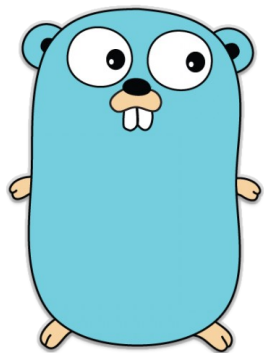
Atomicidad:

```
BEGIN;  
UPDATE accounts SET balance = balance - 100.00  
  WHERE name = 'Alice';  
SAVEPOINT my_savepoint;  
UPDATE accounts SET balance = balance + 100.00  
  WHERE name = 'Bob';  
-- oops ... forget that and use Wally's account  
ROLLBACK TO my_savepoint;  
UPDATE accounts SET balance = balance + 100.00  
  WHERE name = 'Wally';  
COMMIT;
```

A tener presente

ACID

Atomicidad:



```
// Create a new context, and begin a transaction
ctx := context.Background()
tx, err := db.BeginTx(ctx, nil)
if err != nil {
    log.Fatal(err)
}

// 'tx' is an instance of '*sql.Tx' through which we can execute our queries

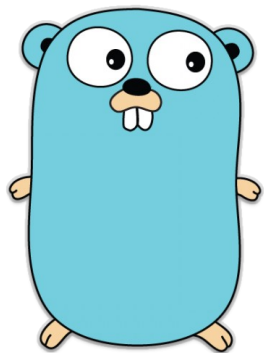
// Here, the query is executed on the transaction instance, and not applied to the database yet
_, err = tx.ExecContext(ctx, "INSERT INTO pets (name, species) VALUES ('Fido', 'dog'), ('Albert', 'cat')")
if err != nil {
    // In case we find any error in the query execution, rollback the transaction
    tx.Rollback()
    return
}

// The next query is handled similarly
_, err = tx.ExecContext(ctx, "INSERT INTO food (name, quantity) VALUES ('Dog Biscuit', 3), ('Cat Food', 3)")
if err != nil {
    tx.Rollback()
    return
}
```

A tener presente

ACID

Atomicidad:



```
// Finally, if no errors are recieved from the queries, commit the transaction
// this applies the above changes to our database
err = tx.Commit()
if err != nil {
    log.Fatal(err)
}
}
```

- <https://go.dev/doc/database/execute-transactions>
- <https://www.sohamkamani.com/golang/sql-transactions/>

A tener presente

ACID o Base de Datos Acida

El acrónimo para los conceptos Atomicity, Consistency, Isolation y Durability

Consistencia:

Hace referencia a la capacidad que tiene un sistema para iniciar solo operaciones que puede concluir. Esto implica que solo se pueden ejecutar pasos de la transacción que no incumplan con las reglas o directrices de integridad definidas, incluyendo los **triggers**, **cascades** y **constraints**, así como sus combinaciones.

PK

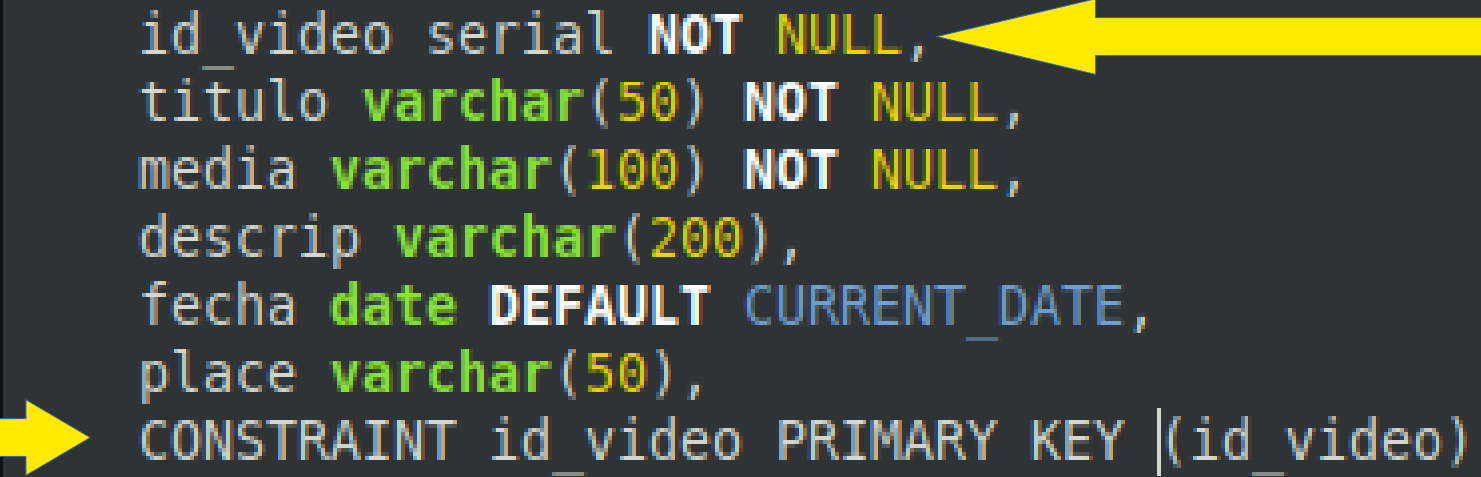
FK

NOT NULL

Cosas de buena crianza

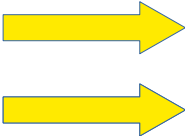
Consistencia:

```
CREATE TABLE public.video (  
  id_video serial NOT NULL,  
  titulo varchar(50) NOT NULL,  
  media varchar(100) NOT NULL,  
  descrip varchar(200),  
  fecha date DEFAULT CURRENT_DATE,  
  place varchar(50),  
  CONSTRAINT id_video PRIMARY KEY (id_video)  
);
```

Two yellow arrows are present. One arrow points from the right edge of the image towards the 'id_video serial NOT NULL,' line. The other arrow points from the left edge of the image towards the 'CONSTRAINT id_video PRIMARY KEY (id_video)' line.

Cosas de buena crianza

Consistencia:



```
-- ALTER TABLE public.percepcion DROP CONSTRAINT IF EXISTS id_video CASCADE;  
ALTER TABLE public.percepcion ADD CONSTRAINT id_video FOREIGN KEY (id_video)  
REFERENCES public.video (id_video) MATCH SIMPLE  
ON DELETE NO ACTION ON UPDATE NO ACTION;  
-- ddl-end --
```

Cosas de buena crianza

Consistencia:

statements:

```
1 ALTER TABLE employee ADD FOREIGN KEY (process_fk)
2 REFERENCES process(emp_id) ON DELETE CASCADE;
```

In the statement above, we specified **ON DELETE CASCADE** — this means that if the parent table is deleted, the child table will also be deleted.

A tener presente

ACID o Base de Datos Acida

El acrónimo para los conceptos Atomicity, Consistency, Isolation y Durability

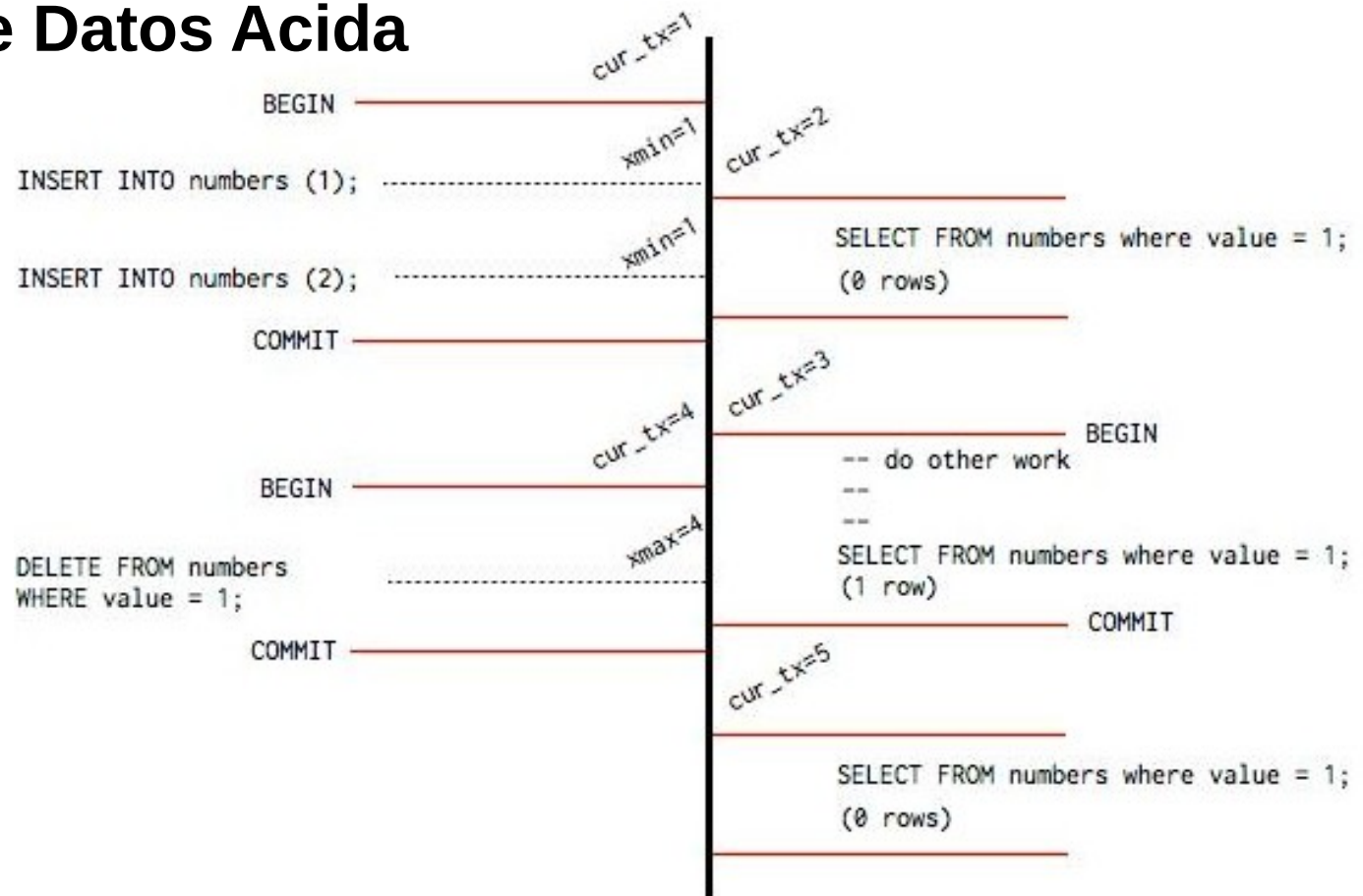
Aislamiento:

Es decir, la realización de una operación no debería afectar a las otras, debido a que cada una de las transacciones debe ser ejecutada en aislamiento total

A tener presente

ACID o Base de Datos Acida

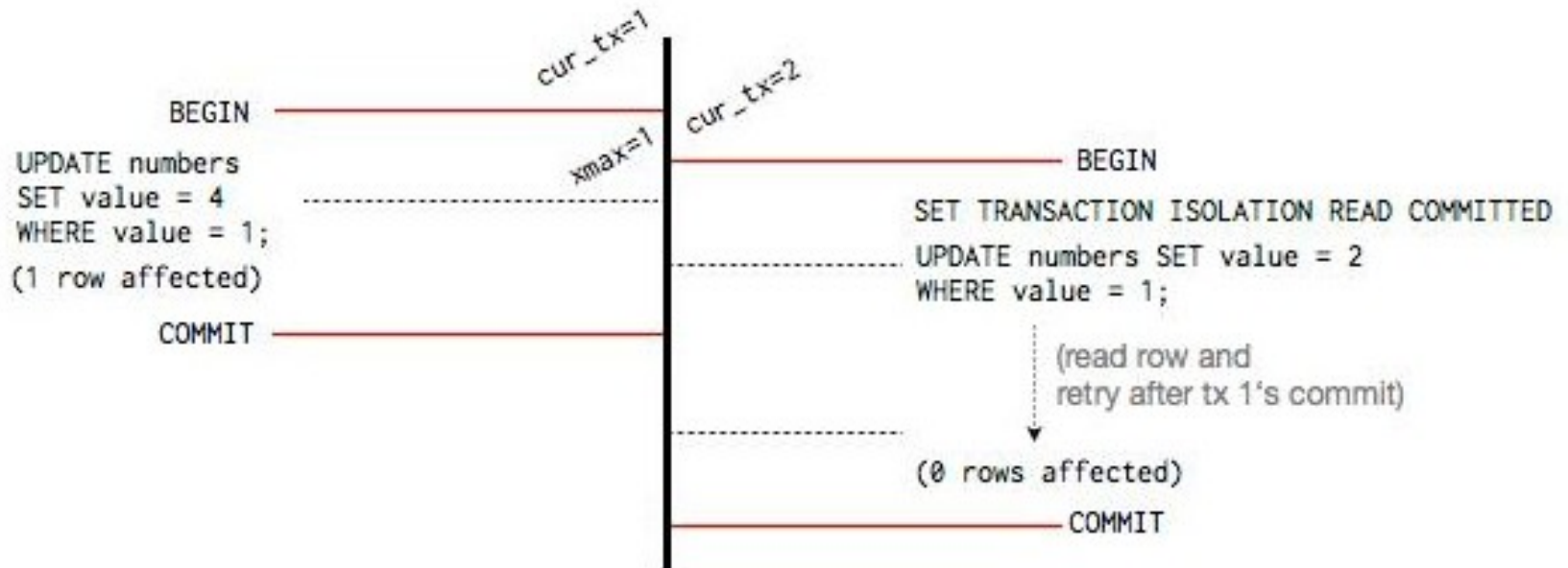
Aislamiento :



A tener presente

ACID o Base de Datos Acida

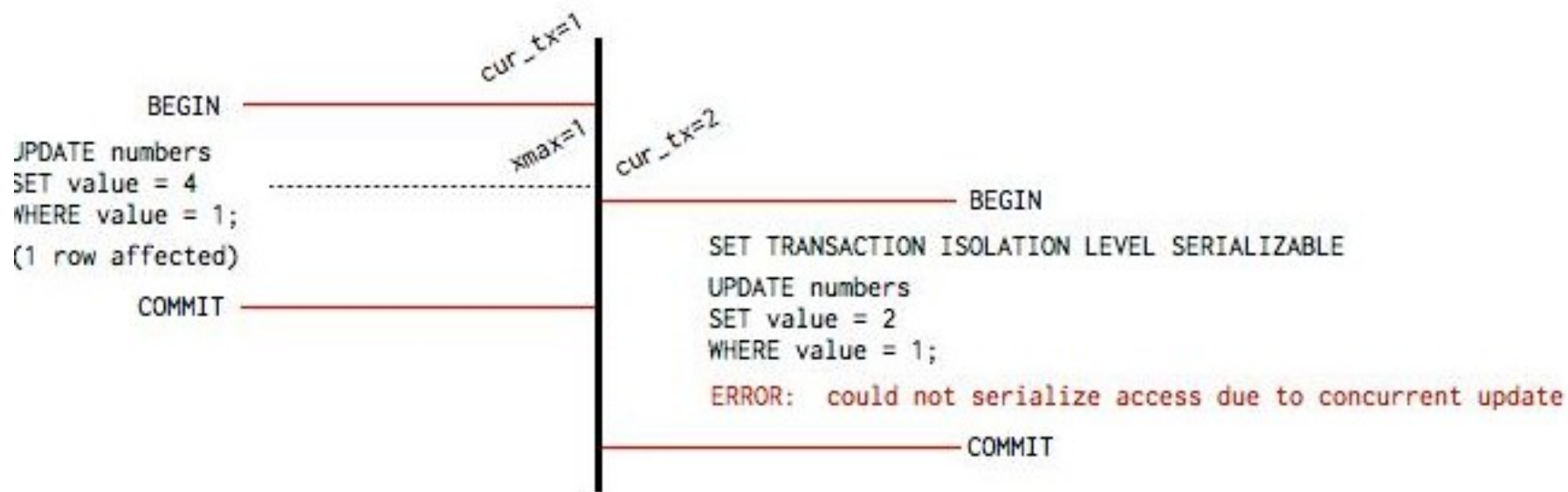
Aislamiento:



A tener presente

ACID o Base de Datos Acida

Aislamiento:



A tener presente

ACID o Base de Datos Acida

El acrónimo para los conceptos Atomicity, Consistency, Isolation y Durability

Durabilidad:

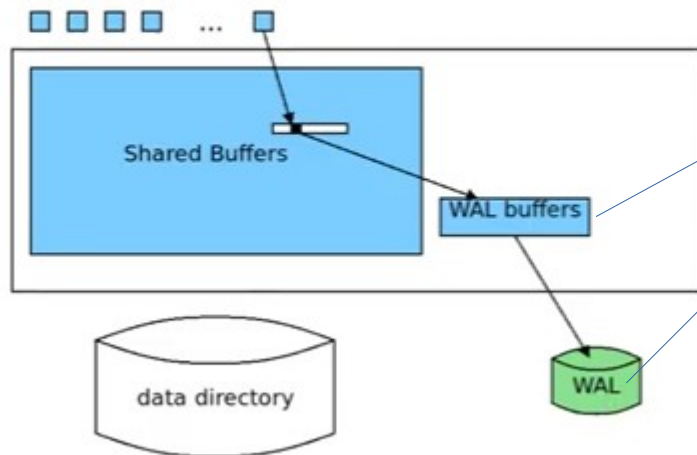
capacidad de persistir y no puedan ser deshechas incluso si el sistema falla

A tener presente

Durabilidad:

capacidad de persistir y no puedan ser deshechas incluso si el sistema falla

PostgreSQL resistente a fallas locales



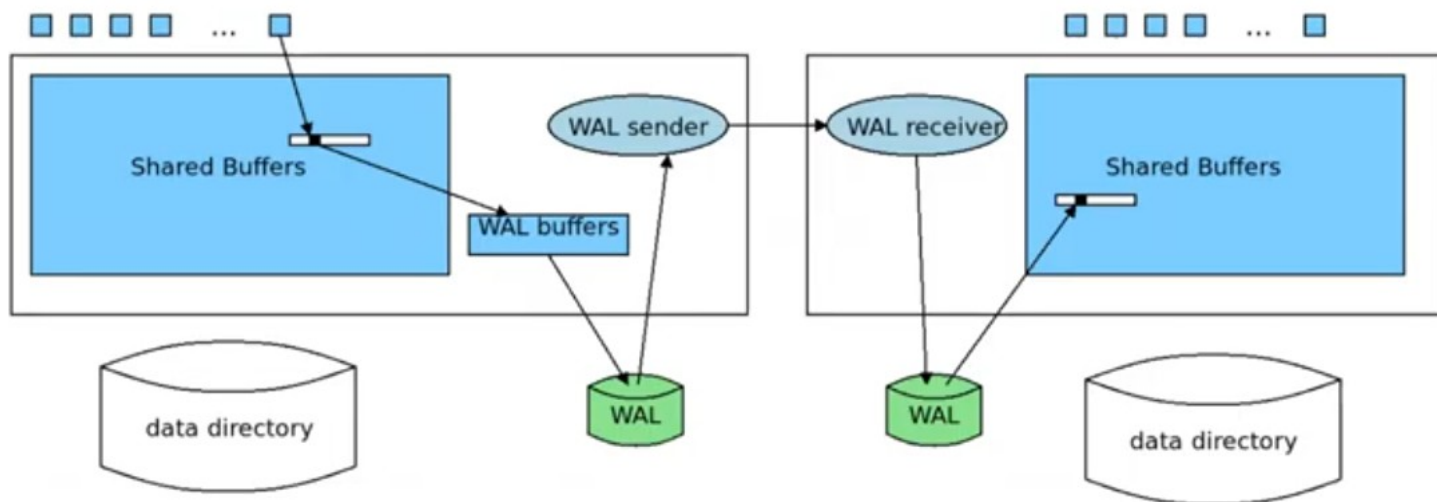
WAL (Write-Ahead Logging),
registro de los que se escribió,
persistencia anticipada

A tener presente

Durabilidad:

capacidad de persistir y no puedan ser deshechas incluso si el sistema falla

Replicación Física



I WILL LIKE TO CALL AN EXCEL



**SPREEDSHEET A
"DATABASE"**

memes.com

I WILL LIKE TO CALL AN EXCEL

**SPREEDSHEET A
"DATABASE"**

memes.com

NO!

Por que no es ACID



Buenas Practicas



Buenas Practicas

Naming ... el arte de poner nombres a las cosas.

- Convenciones
 - Buenas prácticas de DBAs con experiencia
- Sentido común
- Consistencia (con uno mismo)

Buenas Practicas

Naming ... el arte de poner nombres a las cosas.

- Nombres de tablas en **Singular** o Plural
- Uso de mayúsculas y **minúsculas**.
- Uso de camelCase vs **underscore** vs
- **Palabra completa** o abreviaciones
- **Ingles** o Español
- Evitar palabras reservadas
 - Tipos de datos (datetime, char,)



Buenas Practicas

Naming ... el arte de poner nombres a las cosas.

- Uso de prefijos y sufijos.
 - id_, _id
- No usar caracteres especiales (todo en ASCII)
- Nombres descriptivos del uso

....

Buenas Practicas

Manejo de password / Claves

1. salt+password = password-salted
2. sha-256(password-salted)

Ejemplo:

- Password: farm1990M0O
- **Salt:** f1nd1ngn3m0
- Password-salted: f1nd1ngn3m0farm1990M0O
- **hash**= 7528ed35c6ebf7e4661a02fd98ab88d92ccf4e48a4b27338fcc194b90ae8855c

Buenas Practicas

Manejo de password / Claves

1. salt+password = password-salted
2. sha-256(password-salted)

Ejemplo:

- Password: farm1990M0O
- **Salt:** f1nd1ngn3m0 [atributo en la tabla](Texto claro o base64)
- Password-salted: f1nd1ngn3m0farm1990M0O
- **hash**= 7528ed35c6ebf7e4661a02fd98ab88d92ccf4e48a4b27338fcc194b90ae8855c [atributo en la tabla]

Buenas Practicas

Manejo de password / Claves en Kubernetes

¿donde guardo las credenciales para que un microservicio o app se conecte con la DB?

En un Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  dbusername: YWRtaW4=
  dbpassword: hash o password en base64
  dbhost: localhost
  dbport: 1523
  dbname: midb
```

Buenas Practicas

Manejo de password / Claves en Kubernetes

¿donde guardo las credenciales para que un microservicio o app se conecte con la DB?

En un Secret

Aplicar el YAML con la configuración para crear el secret

```
kubectl apply -f ./secret.yaml
```

Algunos comandos que puede ser utiles y ni tanto



Manejo de textos en PostgreSQL

Funciones de Strings

- Cambiar el case : **upper**(string) y **lower**(string)

```
minsal2=# select nombre from comunas;
 nombre
-----
 Maipu
 Ñuñoa
 Providencia
 Las Condes
 Santiago
(5 rows)
```

```
minsal2=# select upper(nombre) from comunas;
 upper
-----
 MAIPU
 ÑUÑOA
 PROVIDENCIA
 LAS CONDES
 SANTIAGO
(5 rows)
```

```
minsal2=# select * from comunas;
 id | nombre | region | fase
----+-----+-----+-----
  2 | Maipu  | Metropolitana | 3
  3 | Ñuñoa  | Metropolitana | 3
  1 | Providencia | Metropolitana | 2
  4 | Las Condes | Metropolitana | 2
  5 | Santiago | Metropolitana | 2
(5 rows)
```

```
minsal2=# select lower(nombre) from comunas;
 lower
-----
 maipu
 ñuñoa
 providencia
 las condes
 santiago
(5 rows)
```

Manejo de textos en PostgreSQL

Funciones de Strings

- Extraer texto : **substr**(string , desde, n)

```
minsal2=# select region from comunas;
        region
-----
Metropolitana
Metropolitana
Metropolitana
Metropolitana
Metropolitana
(5 rows)

minsal2=# select substr(region, 0, 6) from comunas;
        substr
-----
Metro
Metro
Metro
Metro
Metro
(5 rows)
```


Manejo de textos en PostgreSQL

Funciones de Strings

- opciones **left**(string, n) y **right**(string,n)

```
test2=# select titulo from video;
          titulo
-----
Arch Enemy - Sunset Over The Empire
Arch Enemy - Sunset Over The Empire
Hans Zimmer performs INCEPTION "Time"
(3 rows)
```

```
test2=# select left(titulo, 10) from video;
      left
-----
Arch Enemy
Arch Enemy
Hans Zimme
(3 rows)
```

```
test2=# select right(titulo, 10) from video;
      right
-----
The Empire
The Empire
ION "Time"
(3 rows)
```

Manejo de textos en PostgreSQL

Funciones de Strings

- Largo del string: **LENGTH**(string)

```
test2=# select length(titulo) from video;
length
-----
      35
      35
      37
(3 rows)
```

Manejo de textos en PostgreSQL

Funciones de Strings

- Reemplazar un texto : **replace**(string text, from text, to text)

```
test2=# select titulo from video;  
          titulo
```

```
-----  
Arch Enemy - Sunset Over The Empire  
Arch Enemy - Sunset Over The Empire  
Hans Zimmer performs INCEPTION "Time"  
(3 rows)
```

```
test2=# select replace(titulo, 'Enemy', 'Linux') from video;  
          replace
```

```
-----  
Arch Linux - Sunset Over The Empire  
Arch Linux - Sunset Over The Empire  
Hans Zimmer performs INCEPTION "Time"  
(3 rows)
```

Manejo de textos en PostgreSQL

Funciones de Strings

- Retornar posición en texto: **position**(substring in texto)

```
test2=# select position('Enemy' in titulo) from video;
 position
-----
        6
        6
        0
(3 rows)
```

Tecnicas para crear querys SQL



Tecnica para crear Querys SQL

Formula para crear Querys de forma mecánica



Teorema de Mishi:

0. Anotar primera linea **SELECT**, segunda linea **FROM**, tercera linea **WHERE** 1=1

1. ¿Que entidades están interactuando?, anotarlas en el **FROM**

2. Agregar las relaciones, ¿Cuales son los atributos que están interactuando?, después del **WHERE** agregarlas con un **AND**, buscar coincidencias

3. Agregar condiciones con un **AND**

4. en la linea del **SELECT** colocar los atributos solicitados



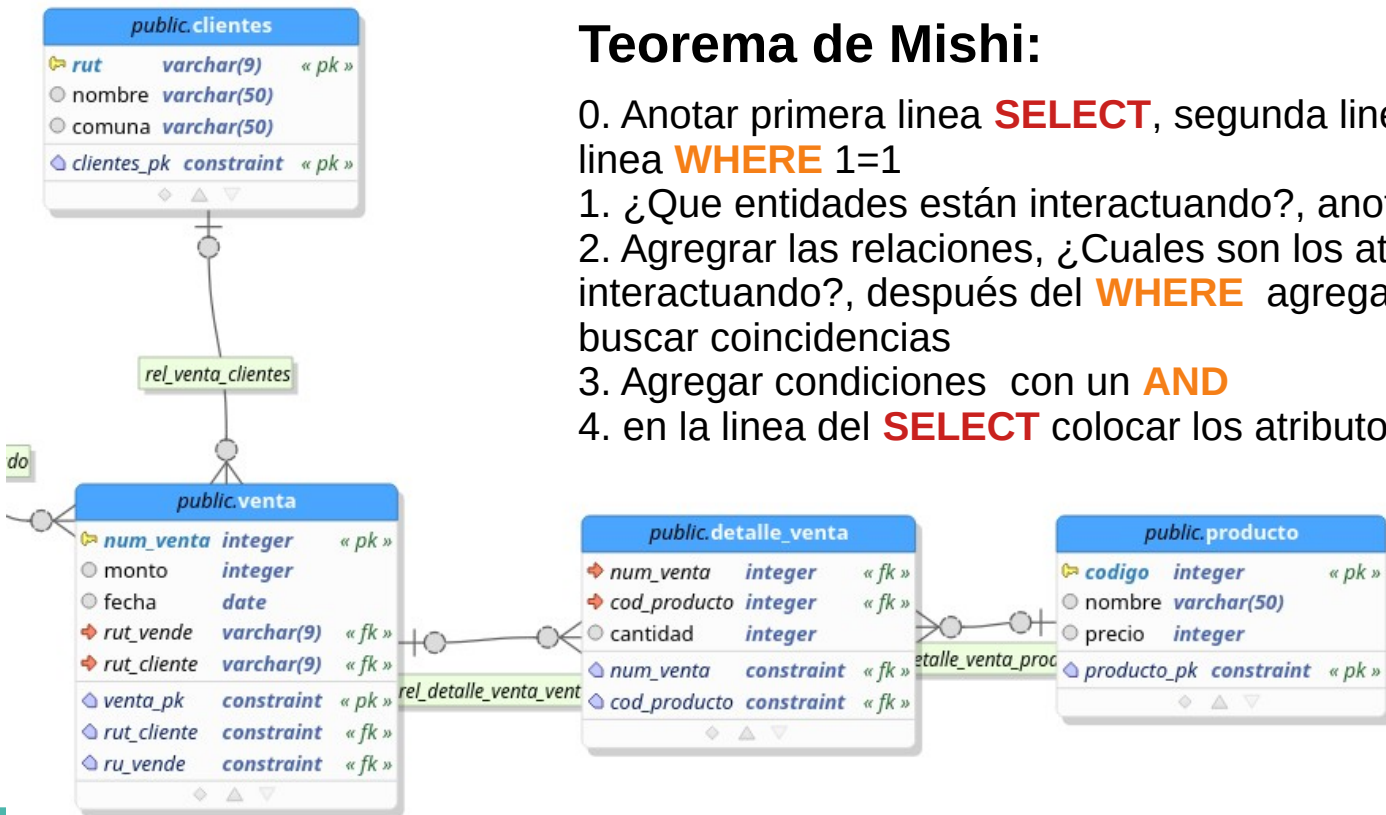
Ejercicio - Tienda



9. Nombre de los productos comprados por los habitantes de Las Condes

Teorema de Mishi:

0. Anotar primera línea **SELECT**, segunda línea **FROM**, tercera línea **WHERE** 1=1
1. ¿Que entidades están interactuando?, anotarlas en el **FROM**
2. Agregar las relaciones, ¿Cuales son los atributos que están interactuando?, después del **WHERE** agregarlas con un **AND**, buscar coincidencias
3. Agregar condiciones con un **AND**
4. en la línea del **SELECT** colocar los atributos solicitados



SQL Selección - Ejercicios



SQL de:

9. Nombre de los productos comprados por los habitantes de Las Condes

```
tienda=> select p.nombre from productos p, ventas v, ventas_detalle vd, clientes c where v.rut_cliente = c.rut and vd.num_venta = v.num_venta and vd.cod_producto = p.codigo and c.comuna='Las Condes' group by p.nombre;
      nombre
-----
Minicomponente
dvd
televisor
mesa
Sillon
proyector
(6 rows)
```


SQL Selección – JOIN

Formula para crear Querys de forma mecánica, usando JOINS
Recomendado cuando se tienen varias tablas a relacionar

La técnica de la Fusión:

0. Identificar Entidades y sus relaciones
1. Anotar **SELECT**
2. Anotar un **JOIN** Con Cada tabla o subquery
3. Relacionar los Join (Fk con PK) usar el **ON** FK = PK o ON atributo=atributo_calculado
4. Anotar **AND** con los filtros requeridos en los JOINS de las tablas correspondientes
5. En el **SELECT** los atributos deseados a obtener y la tabla que los contiene



SQL Selección - Ejercicios

Enseña [<https://enseña.cl>] – Tienda Online

19. Nombre del vendedor que ha vendido el producto mas caro



```
select e.nombre, e.sueldo from empleados e
join ventas v
  on v.rut_vende = e.rut
join ventas_detalle vd
  on v.num_venta = vd.num_venta
join productos p
  on vd.cod_producto = p.codigo
join (select max(precio) as pmax from productos) p2
  on p.precio = p2.pmax;
```

nombre		sueldo
-----	+	-----
MARTIN		1250.00
TURNER		1500.00
(2 rows)		



WITH

Min? Max? AVG? ,With podría ser lo que simplifique la construcción de la query

Ejemplo: ¿Cual es el cliente que menos compras a realizado en la tienda?

```
WITH ventas_por_cliente AS (  
    SELECT COUNT(*), v.rut_cliente  
    FROM ventas v  
    GROUP BY v.rut_cliente  
)  
  
SELECT c.rut, c.nombre  
FROM clientes c, ventas_por_cliente vpc  
WHERE vpc.count = (SELECT MIN(count) FROM ventas_por_cliente)  
AND vpc.rut_cliente = c.rut
```

SQL Selección – GROUP BY

GROUP BY

- Se usa con el SELECT
- Agrupa filas de la tabla que tienen valores idénticos
- Se utiliza para eliminar redundancia en la salida
- Se utiliza para calcular (contar, sumar) elementos con atributos del mismo valor

```
SELECT column-list  
FROM table_name  
WHERE [ conditions ]  
GROUP BY column1, column2....columnN  
ORDER BY column1, column2....columnN
```

SQL Selección – GROUP BY

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```



Técnicas para crear SQL

How to create SQL?

Es la combinación de las técnicas anteriores

- Min, Max, AVG → se recomienda usar With crear una tabla: id, valor (count / sum), sobre eso sacar el min y asociar



Técnicas para crear SQL

How to create SQL?

Es la combinación de las técnicas anteriores

- Varias relaciones a múltiples tablas, usar la técnica de la fusión, unir tablas con JOIN y relacionarlas con el ON usando FK=PK



Técnicas para crear SQL

Combinar lo anterior

Material Disponible en:

https://gitlab.com/l30bravo/db_udp

**Muchas
gracias!**

Correo:

leonardo.bravo@mail.udp.cl