

A decorative graphic on the left side of the cover, consisting of a network of yellow and white lines and circles, resembling a circuit board or a stylized tree structure.

# TRANSACCIONES

SANTOS MARTÍNEZ FERNÁNDEZ

JUAN CRESPO FUENTES

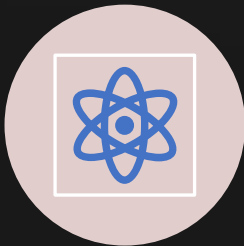
# ABSTRACT

- ¿Qué es una transacción?
- ACID
- System Change Number
- Estructura de una transacción
- Control de transacciones
- Transacciones autónomas
- Panorámica de transacciones distribuidas
- Bibliografía

# ¿QUÉ ES UNA TRANSACCIÓN?

- Una transacción es una o más operaciones que se ejecutan de manera secuencial y fiable en una base de datos.
- Para garantizar la seguridad de las transacciones, la base de datos deberá cumplir las propiedades **ACID**

# ACID



Atomicity: Para garantizar una transacción exitosa todas las operaciones de una transacción se realizarán (commit), sino es así no se realizará ninguna (roll back).



Ejemplo: Enviar dinero a una cuenta que está inactiva, primero se hace una resta a tu cuenta, tras esto se intenta hacer una suma en la cuenta receptora pero al comprobar que está inactiva se deshacen todos los cambios realizados.

# ACID



Consistency: Una base de datos pasará de un estado consistente a otro igualmente consistente después de una transacción.

# ACID



Isolation: Los efectos de una transacción no serán visibles por otra transacción hasta que se haya realizado por completo (commit).

## Ejemplo

Tienes 20€

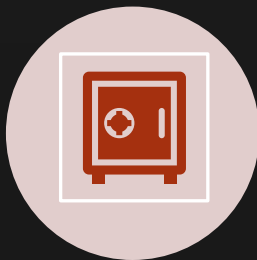
Netflix  
-12€

Lectura  
(20€)

Commit de  
Netflix

Lectura(8€)

# ACID



Durability: Para garantizar una transacciones exitosa todas las operaciones de una transaccion se realizarán (commit), sino es así no se realizará ninguna (roll back).

# SYSTEM CHANGE NUMBER



Para el correcto funcionamiento de las propiedades ACID, Oracle utiliza una marca temporal interna para controlar todos los cambios en la base de datos. Cada transacción tiene un SCN

**Ejemplo:** Si una transacción actualiza una tupla la base de datos guardará el momento en el que se actualice. Si la transacción provoca otras modificaciones estas tendrán el mismo SCN



# ESTRUCTURA - COMIENZO

DDL

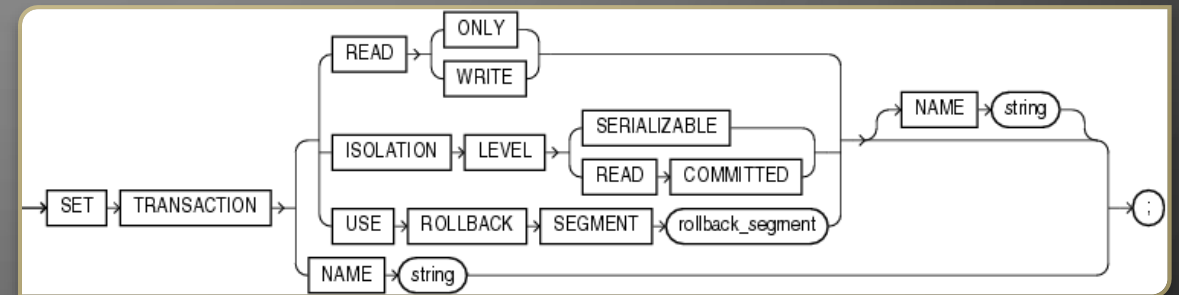
- INSERT
- DELETE
- UPDATE

DML

- CREATE
- DROP
- ALTER

# SET TRANSACTION

```
SET TRANSACTION {  
    { READ { ONLY | WRITE }  
    | ISOLATION LEVEL {  
      SERIALIZABLE | READ  
      COMMITTED}  
    | USE ROLLBACK SEGMENT  
      rollback_segment }  
    [ NAME string ]  
    | NAME string } ;
```



# SET TRANSACTION - READ

```
SET TRANSACTION {  
  { READ { ONLY | WRITE }  
  | ISOLATION LEVEL {  
    SERIALIZABLE | READ  
    COMMITTED}  
  | USE ROLLBACK SEGMENT  
    rollback_segment }  
  [ NAME string ]  
  | NAME string } ;
```

READ ONLY: Establece que la transacción actual es solo lectura.

READ WRITE: Establece que la transacción actual es de lectura/escritura

# SET TRANSACTION – ISOLATION LEVEL

```
SET TRANSACTION {  
    { READ { ONLY | WRITE }  
    | ISOLATION LEVEL {  
      SERIALIZABLE | READ  
      COMMITTED}  
    | USE ROLLBACK SEGMENT  
      rollback_segment }  
    [ NAME string ]  
    | NAME string } ;
```

- SERIALIZABLE: El objetivo es que se permita ejecutar transacciones concurrentemente sin que estas interfieran entre sí. (por defecto)
- READ COMMITTED: Evita la lectura sucia, ya que la transacción solo podrá leer datos que han sido reafirmados por el **commit** de otra transacción.

# SET TRANSACTION – USE ROLLBACK SEGMENT

```
SET TRANSACTION {  
    { READ { ONLY | WRITE }  
    | ISOLATION LEVEL {  
    SERIALIZABLE | READ  
    COMMITTED}  
    | USE ROLLBACK SEGMENT  
    rollback_segment }  
    [ NAME string ]  
    | NAME string } ;
```

- Sirve para especificar a qué segmento de rollback (rollback\_segment) queremos asociar nuestra transacción.
- Un segmento de rollback contiene los datos necesarios para realizar la operación rollback

# SET TRANSACTION - EJEMPLO

```
4
5 BEGIN TRAN;
6 UPDATE BankDetailTbl
7 SET
8     Balance=Balance-45
9 WHERE AccountNumber = '7Y290394';
10 WAITFOR DELAY '00:00:20';
11 ROLLBACK TRAN;
12
13 SELECT *
14 FROM BankDetailTbl
15 WHERE AccountNumber = '7Y290394';
```

Query-1

100 %

Id	AccountNumber	ClientName	Balance
1	7Y290394	Betty H. Bonds	78.00

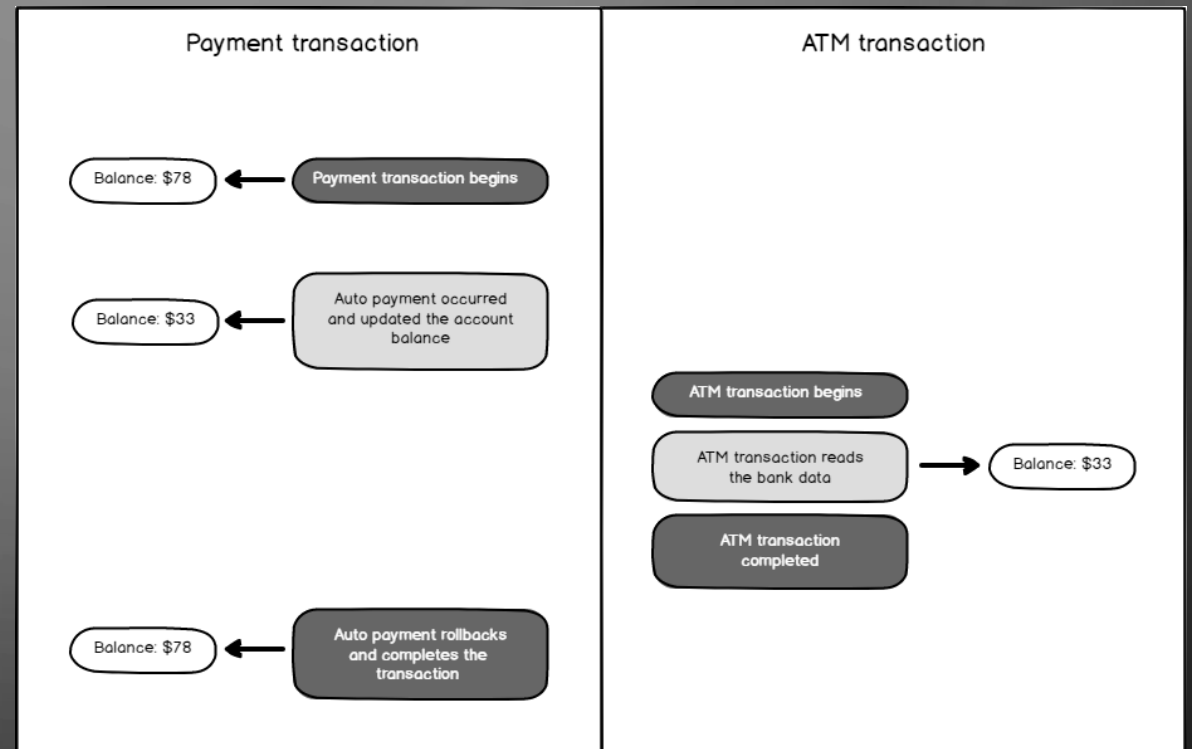
```
4
5 SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
6 BEGIN TRAN;
7 SELECT *
8 FROM BankDetailTbl
9 WHERE AccountNumber = '7Y290394';
10 COMMIT TRAN;
11
```

Query-2

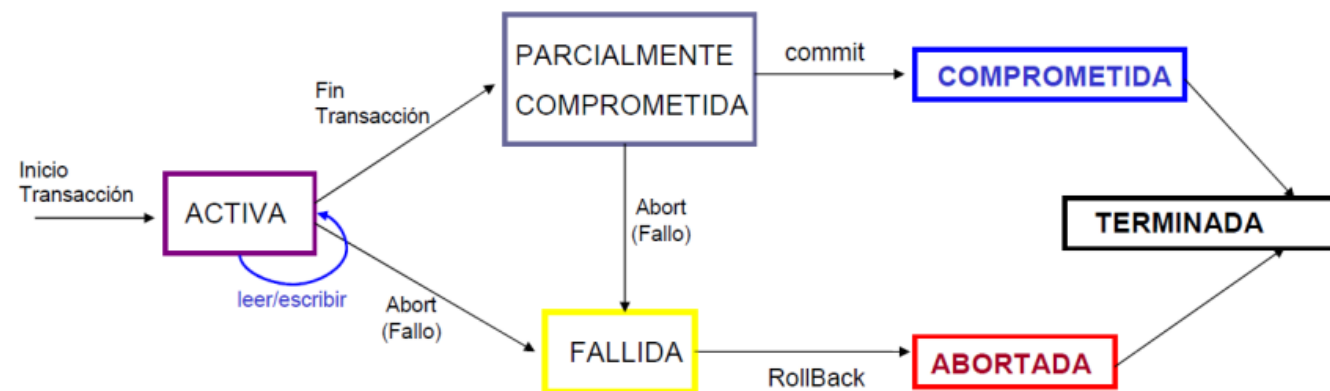
100 %

Id	AccountNumber	ClientName	Balance
1	7Y290394	Betty H. Bonds	33.00

Dirty data



# ESTADOS DE UNA TRANSACCIÓN



# ESTRUCTURA - FINALIZACIÓN

Un usuario cierra el entorno de trabajo durante una transacción (COMMIT implícito)

Un usuario realiza un COMMIT o un ROLLBACK sin usar la cláusula SAVEPOINT

Un usuario realiza la finalización abrupta de una transacción (ROLLBACK)

Cuando se utiliza una comando DDL o DML (COMMIT implícito)



# CONTROL DE TRANSACCIONES

## COMMIT

- Termina la transacción actual y convierte todos los cambios ejecutados por la transacción en permanente. Además, borra todos los SAVEPOINT

## SAVEPOINT

- Marca un punto de control en la transacción en la que después se podrá hacer ROLLBACK

## ROLLBACK

- Revierte todos los cambios de la transacción actual

## ROLLBACK TO SAVEPOINT

- Revierte los cambios hasta el último SAVEPOINT, pero no la transacción entera

# EJEMPLO


1: SET TRANSACTION NAME 'NOTAS\_UPDATE';

2: UPDATE abd  
SET nota = 6  
WHERE name = 'Antonio'

3: SAVEPOINT nota\_guardada;

4: UPDATE abd  
SET nota = 8  
WHERE name = 'Lucía';

5: ROLLBACK TO SAVEPOINT nota\_guardada;



```
6: UPDATE abd  
SET nota = 9  
WHERE name = 'Lucía'
```



```
7: COMMIT;
```

# TRANSACCIONES AUTÓNOMAS

Son transacciones que se ejecutan de manera independiente de la transacción principal. Cuando se llama a una transacción autónoma se detiene la principal.

COMMIT/ROLLBACK  
solo afectan a las  
modificaciones dentro  
de esta transacción

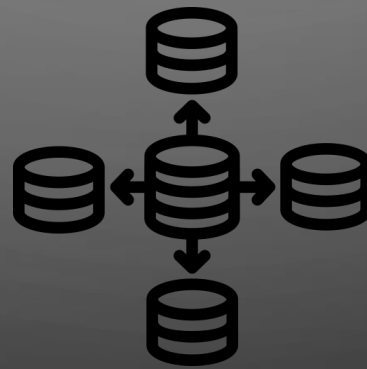
Con EXECUTE  
IMMEDIATE podemos  
usar sentencias DDL

Serán visibles por  
otras transacciones si  
usan READ  
COMMITTED

# TRANSACCIONES DISTRIBUIDAS

En una **base de datos distribuida**, los datos se encuentran en diferentes sitios o nodos, lo que hace necesario coordinar la actualización o consulta de datos a través de la red.

Las **transacciones distribuidas** se deben coordinar mediante protocolos para garantizar la integridad y consistencia de datos en toda la red.





---

Cada nodo involucrado en la transacción puede tener su propia transacción local para actualizar los datos locales.

---

En las transacciones globales, cada nodo involucrado en la transacción informa al coordinador central sobre si puede realizar la transacción localmente. Si todos los nodos están de acuerdo, se avanza a la fase de confirmación. En la fase de confirmación, se confirma la transacción global en cada nodo y se garantiza que todos los nodos estén en un estado coherente.

---

# BIBLIOGRAFÍA

- <https://diego.com.es/transacciones-en-sql>
- [https://en.wikipedia.org/wiki/Data\\_definition\\_language](https://en.wikipedia.org/wiki/Data_definition_language)
- <https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/SET-TRANSACTION.html#GUID-F11E1E30-5871-48D1-8266-F80A1DF126A1>
- <https://www.sqlservercentral.com/articles/isolation-levels-in-sql-server>
- <https://www.magicplsql.com/pl-sql/item/84-los-pragmas-autonomous-transaction-y-exception-init>
- <https://www.sqlshack.com/dirty-reads-and-the-read-uncommitted-isolation-level/>
- Oracle11.2 - Transactions.pdf
- Tema4\_ABD\_13\_14\_Parte1.pdf
- ChatGPT
- Apuntes de la asignatura 'Bases de Datos'

The slide features a dark gray background with a subtle pattern of concentric circles. In each of the four corners, there are decorative yellow circuit-like lines. These lines consist of straight segments connected by small circles, resembling a stylized electronic circuit board. The lines vary in length and orientation, creating a symmetrical, geometric pattern around the central text.

¿ALGUNA PREGUNTA?