Digital Career Institute

Python Course - Database Consistency

Digital Career Institute

Goal of the Submodule

The goal of this submodule is to introduce the learners to database consistency and concurrency. By the end of this submodule, the learners will be able to:

- Tell the difference between transactions and operations.
- Name and understand the standard properties of transactions in relational DBMS.
- Know how a RBDMS handles concurrent transactions.
- How does PostgreSQL lock the most common commands and how to define custom locks.

Topics

- Transactions and operations
- Transaction properties
    - Atomic
    - Consistent
    - Isolated
    - Durable
- Transactions in NoSQL
- Using transactions in PostgreSQL
- Concurrency
- Database Locking
    - Lock Modes

Database Consistency & Transaction Properties

Digital Career Institute

Database Consistency

Consistency is the property of a transaction that guarantees that the changes done will be following all the defined **rules**.

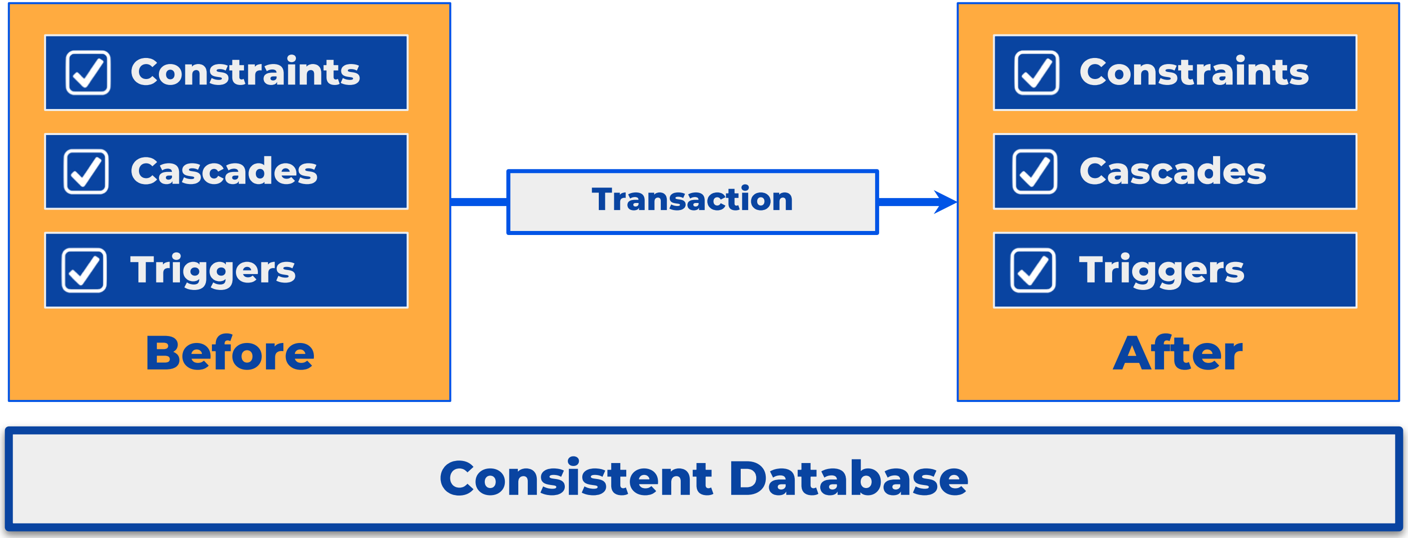
Constraints | **Cascades** | **Triggers**

Rules

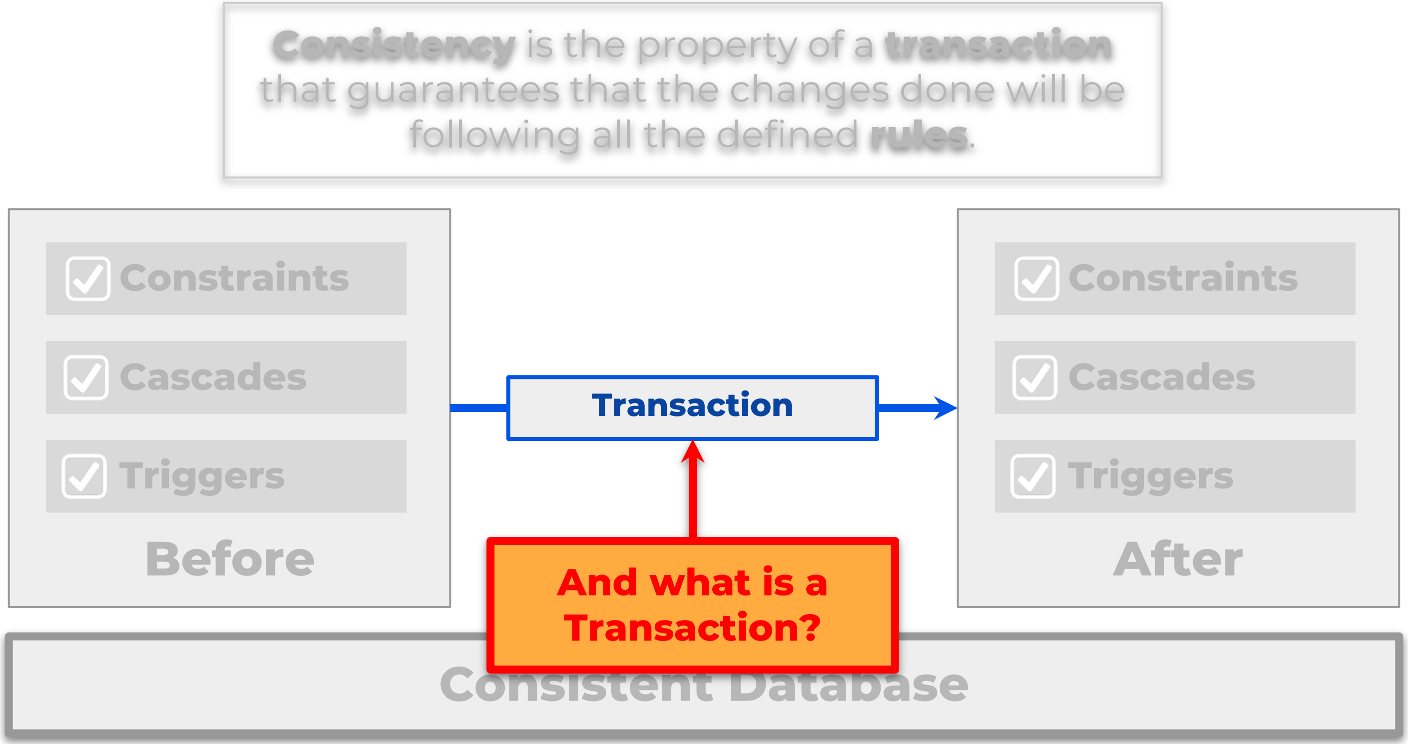
Database Consistency

Digital Career Institute
DCI

Consistency is the property of a **transaction** that guarantees that the changes done will be following all the defined **rules**.



Database Consistency



Database Transactions

A **database transaction** is
a “single change” in the database.

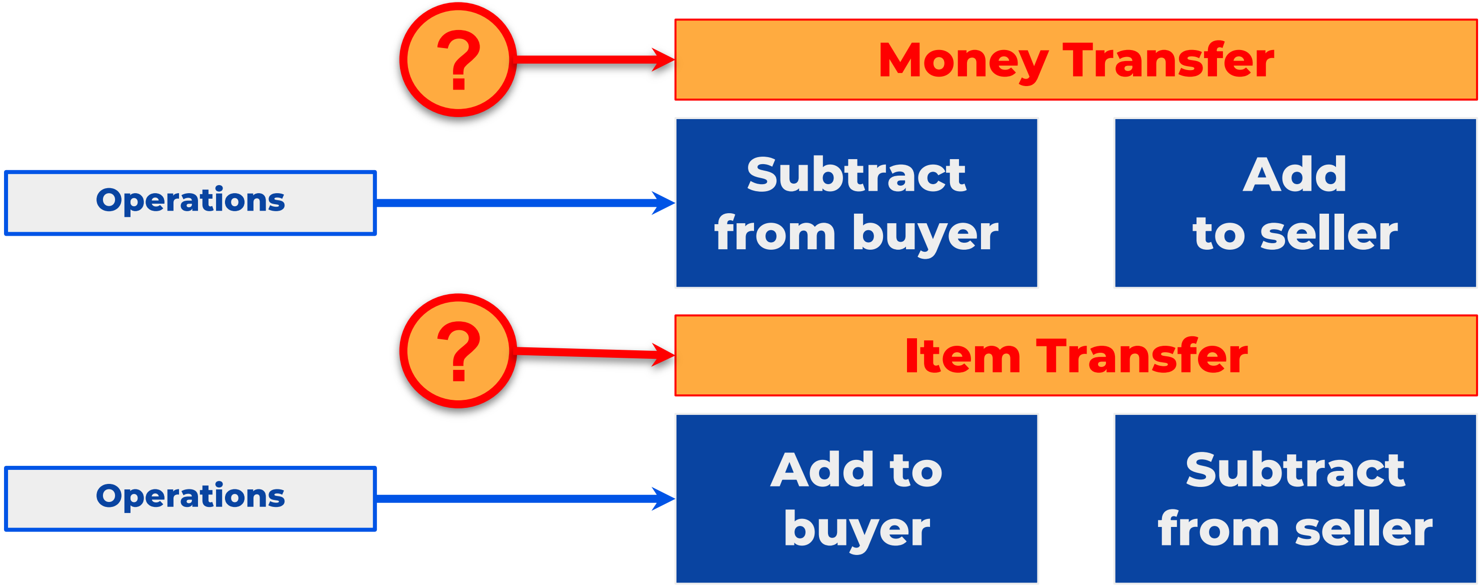
But a transaction may include
more than one **operation**.

Transaction → **Money Transfer**

Operations → **Subtract from buyer** | **Add to seller**

Database Transactions

What if we are not just
transferring money?



Database Transactions

What is a transaction and what is an operation depends on the nature of the application.

Transaction Properties: Atomicity

A database transaction must be done as a whole or not done at all.

This property of the database transactions is called **Atomicity**.

Transaction Properties

Digital Career Institute

Database transactions must be **atomic** and **consistent**.

Transaction

Atomic

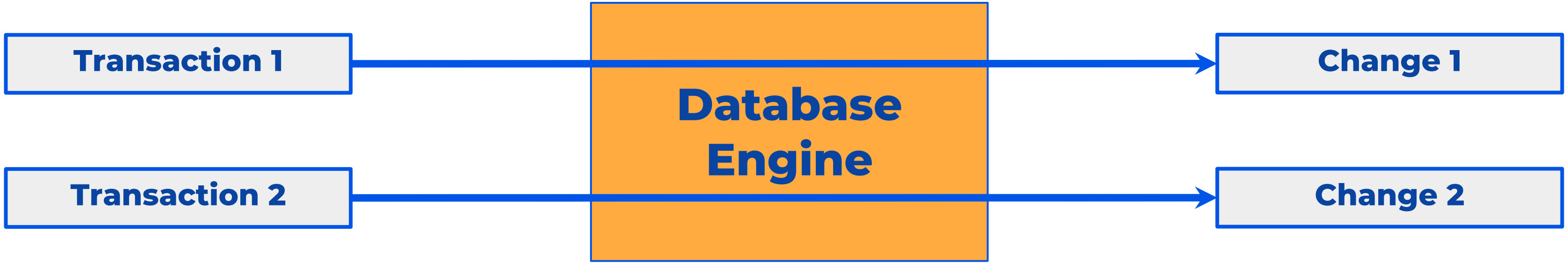
Consistent

But they must have additional properties ...

Transaction Properties

A database transaction must also be **isolated**.

A higher level of isolation has higher chances of one transaction blocking another.



A lower level of isolation has higher chances of producing **concurrency** undesired effects.

Transaction Properties

Digital Career Institute

A database transaction must also be **durable**.

The changes should be **persistent**.

Once a change is done, it is written on the file system, so that only another change will remove it.

Transaction Properties

Digital Career Institute

The properties *atomic*, *consistent*, *isolated* and *durable* are commonly known as the **ACID** properties of database transactions.

ACID Properties

Atomic	**Consistent**
Isolated	**Durable**

ACID Compliance

PostgreSQL

MySQL

SQLite

All Relational DBMS are, or can be, ACID compliant.

ACID Compliance

Digital Career Institute

MongoDB

CouchDB

Db2

Some NoSQL DBMS can be ACID compliant.

But ACID is a relational concept and most NoSQL use the **BASE properties**, which focus on availability rather than consistency.

BASE Transaction Properties

Digital Career Institute

Most **NoSQL** databases use transactions that are *basically available*, *soft state* and *eventually consistent*.

Basically Available	The data is spread in different systems to increase availability. Having some data easily available is more important than having a current version of the data.
Soft State	The database does not ensure consistency, so it is the application developer who must ensure it from its end.
Eventually Consistent	The database will eventually be consistent. The data will be updated at some point, but meanwhile it is still possible to access the data.

Transactions in PostgreSQL

Transactions in PostgreSQL

Digital Career Institute

Money Transfer

Transaction

```
BEGIN;
    UPDATE accounts
        SET balance = balance - 100
        WHERE name = "buyer";
    UPDATE accounts
        SET balance = balance + 100
        WHERE name = "seller";
COMMIT;
```

Operations

To **commit** a transaction means to make the changes on the data persistent.

Transactions in PostgreSQL

Digital Career Institute

Money Transfer

```
UPDATE accounts
     SET balance = balance - 100
     WHERE name = "buyer";
UPDATE accounts
     SET balance = balance + 100
     WHERE name = "seller";
```

Transactions

PostgreSQL will **autocommit** each statement if no BEGIN/COMMIT statements are provided.

Transactions in PostgreSQL

Money Transfer

Execution

```
BEGIN;
    UPDATE accounts
        SET balance = balance - 100
        WHERE name = "buyer";
    UPDATE wrong_table_name
        SET balance = balance + 100
        WHERE name = "seller";
COMMIT;
```

Roll back

If the system encounters an error,
it will **roll back** the previous changes.

Transactions in PostgreSQL

Digital Career Institute

Money Transfer

```
BEGIN;
    UPDATE accounts
        …
    SAVEPOINT my_point;
    UPDATE accounts
        ...
    ROLLBACK TO my_point;
COMMIT;
```

Execution

Roll back

The **SAVEPOINT** statement can be used to roll back **TO** a specific point in the transaction.

We learned ...

- What is a transaction and the difference with an operation.
- What are the standard properties of a relational database transaction: atomicity, consistency, isolation and durability.
- That all RDBMS provide some degree of ACID compliance.
- That some NoSQL also provide ACID transactions, but many use the BASE properties.
- How to define transactions in PostgreSQL.

Self Study

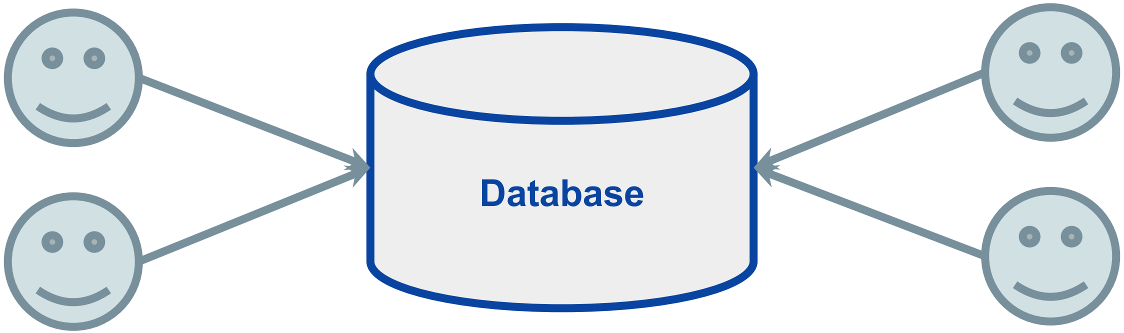
- ACID vs. BASE
- Isolation levels
- Database concurrency

Database Concurrency

Database Concurrency

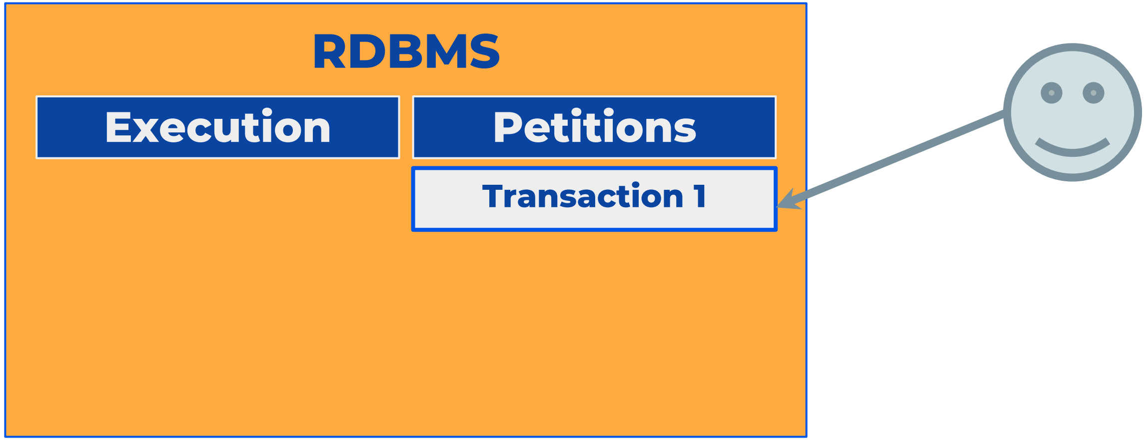
The ACID properties are specially relevant when we have a high level of concurrency.

Database concurrency is the ability to execute two or more transactions at the same time.



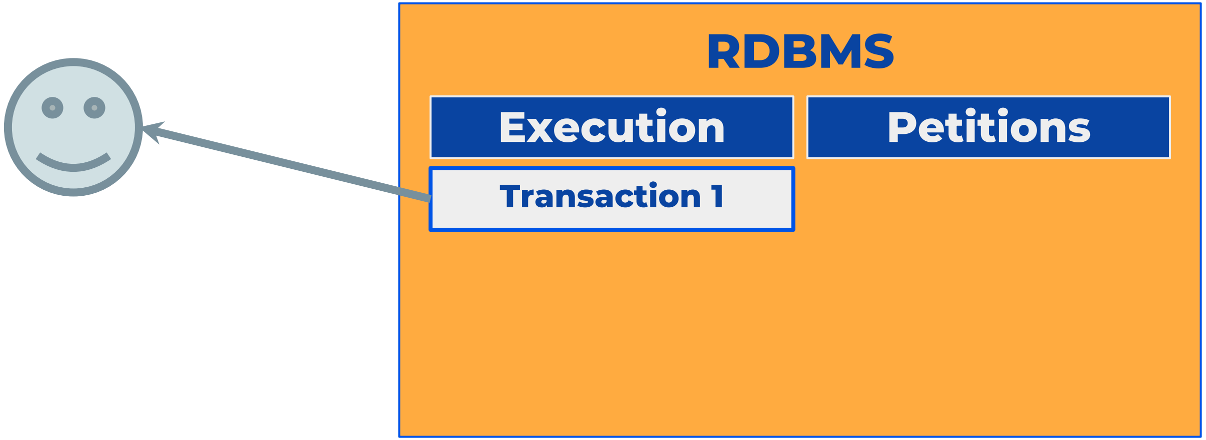
Non-concurrent Transactions

A user requests the RDBMS to execute a transaction.



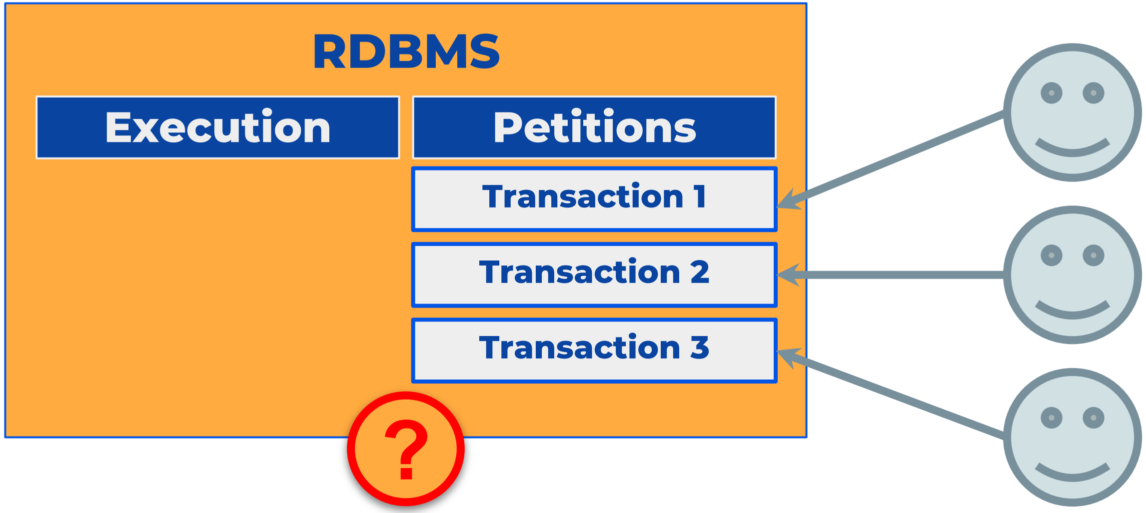
Non-concurrent Transactions

The transaction is executed and the result, if any, is returned to the user.



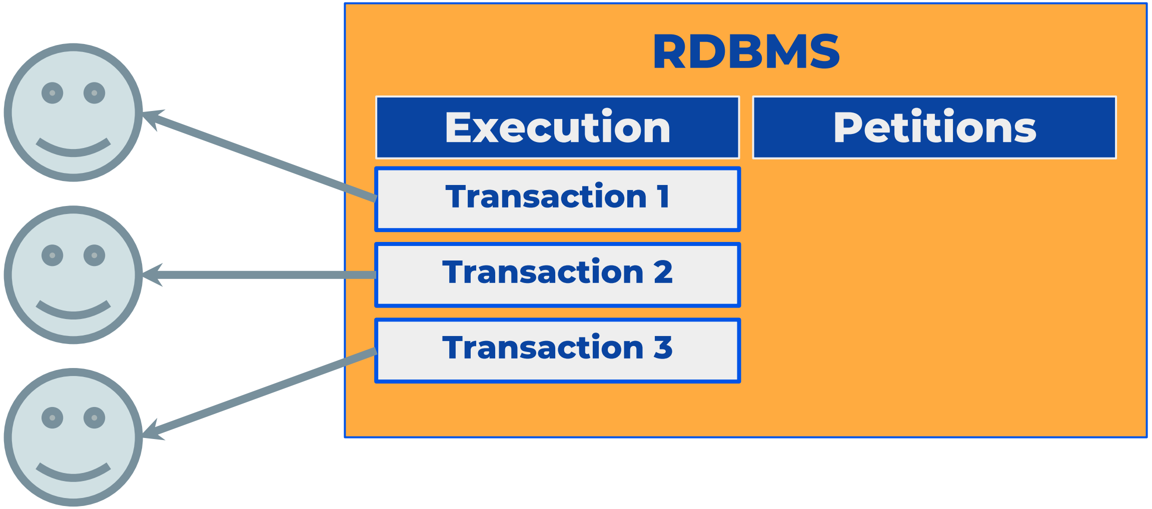
Concurrent Transactions

What happens if multiple requests
are received at the same time?



Concurrent Transactions

Transactions are executed simultaneously.



Concurrent Transactions

Digital Career Institute

But this is not exactly true.

There are some types of transactions that cannot be executed simultaneously with another transaction.

Reading vs. Writing

The nature of the transactions determines how the concurrent transactions are managed.

Reading transactions can be executed simultaneously.

Types of Transactions

Reading	Writing
SELECT	UPDATE
	DELETE
	INSERT

Writing transactions very often cannot be executed simultaneously.

Concurrency Issues

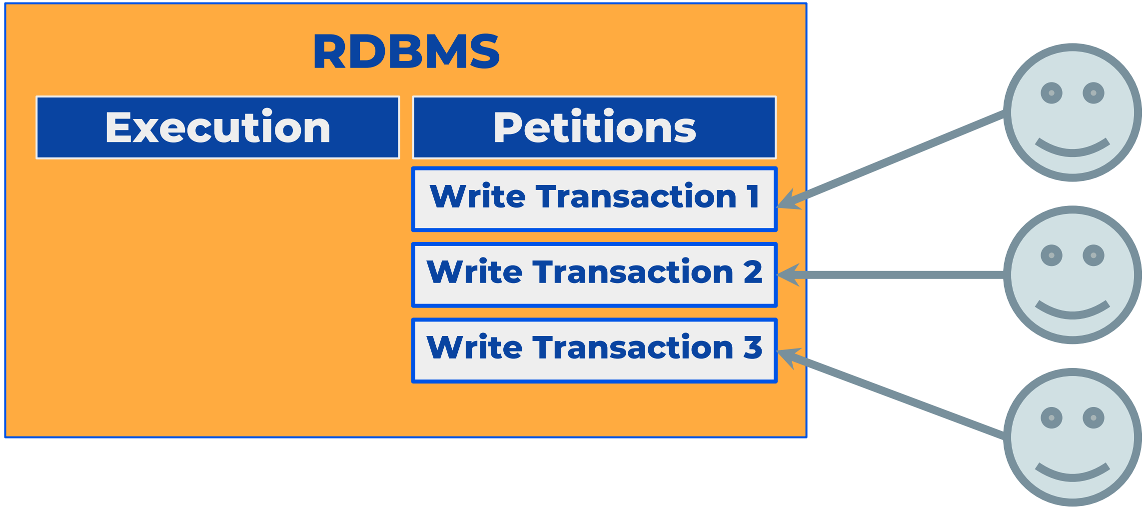
The **isolation** property prevents the occurrence of some undesired behavior due to concurrent writing transactions.

Dirty reads	A transaction reads data written by concurrent uncommitted transactions.
Non-repeatable reads	A transaction re-reads data it has previously read and finds that data has been modified by another transaction (that committed since the initial read).
Phantom reads	A transaction re-executes a query returning a set of rows that satisfy a search condition and finds that the result changes if executed again, due to another recently-committed transaction.

Concurrent Writing Transactions

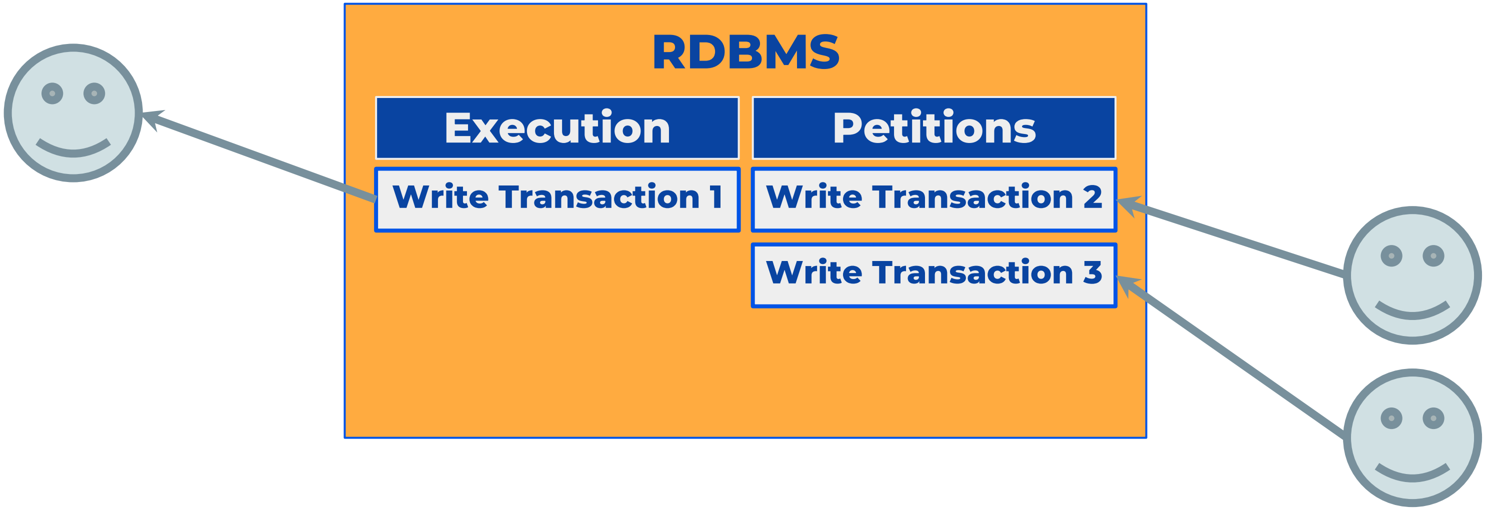
Digital Career Institute

Concurrent writing transactions are executed sequentially to prevent isolation issues.



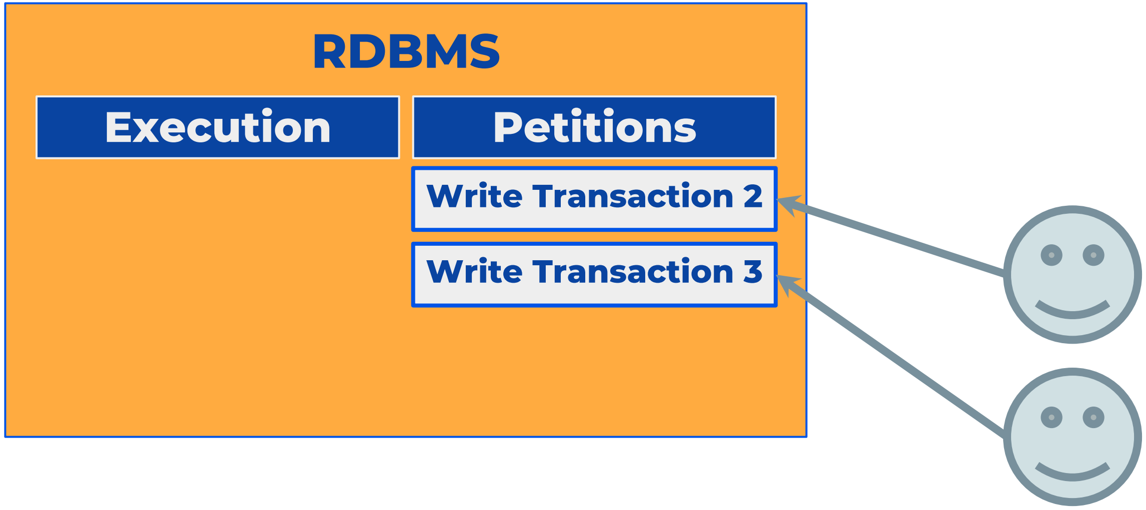
Concurrent Writing Transactions

While the system executes the first transaction the rest stays “on hold”, waiting for the first to finish.

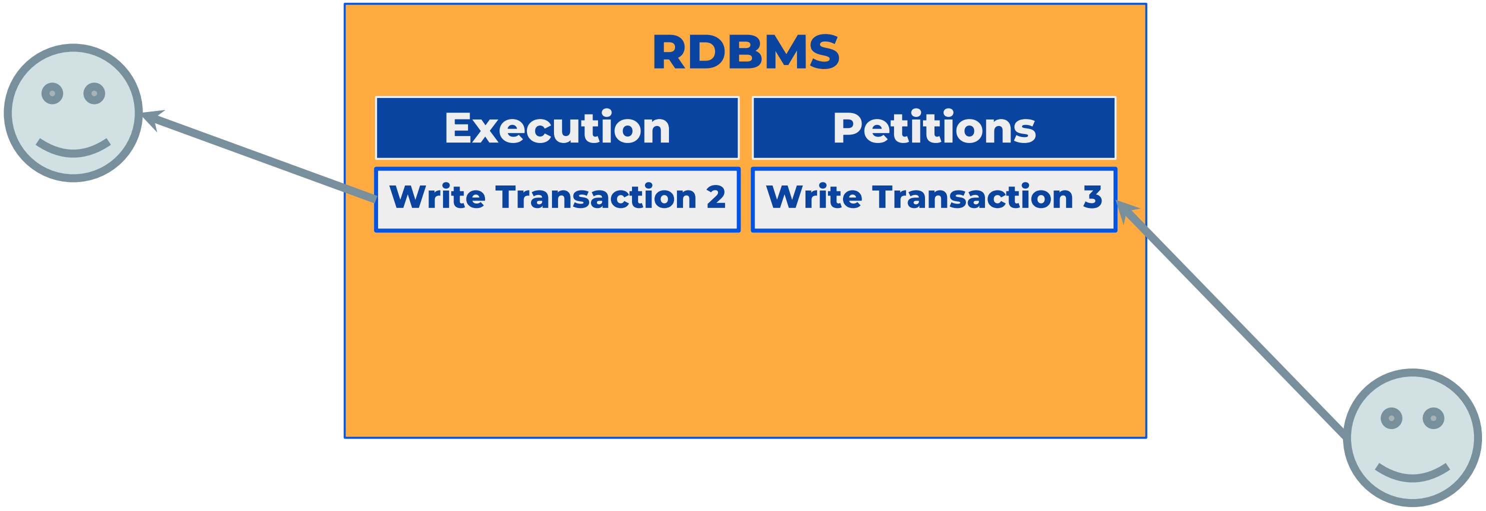


Concurrent Writing Transactions

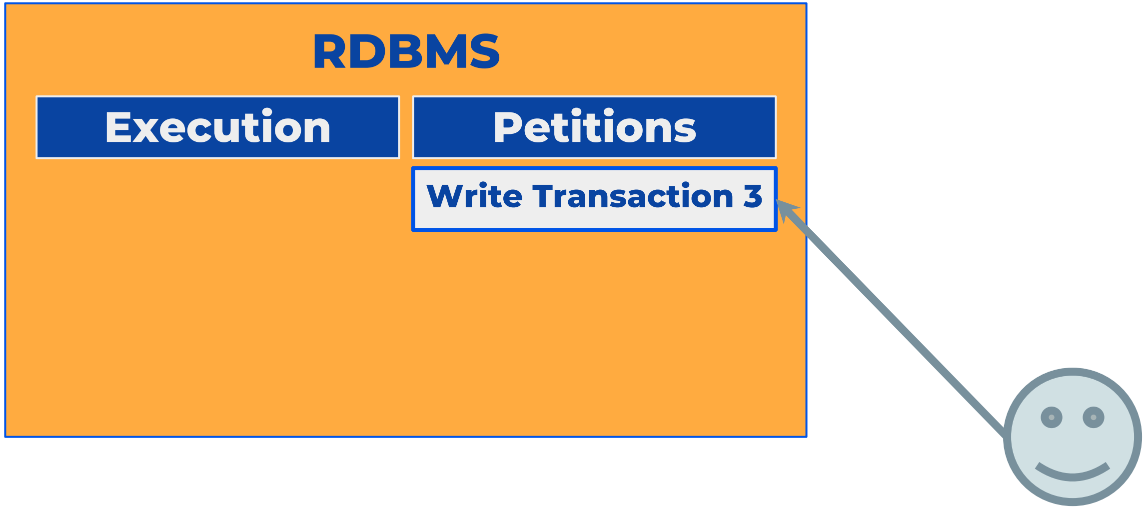
When the first transaction has finished, the next one in the queue will be executed.



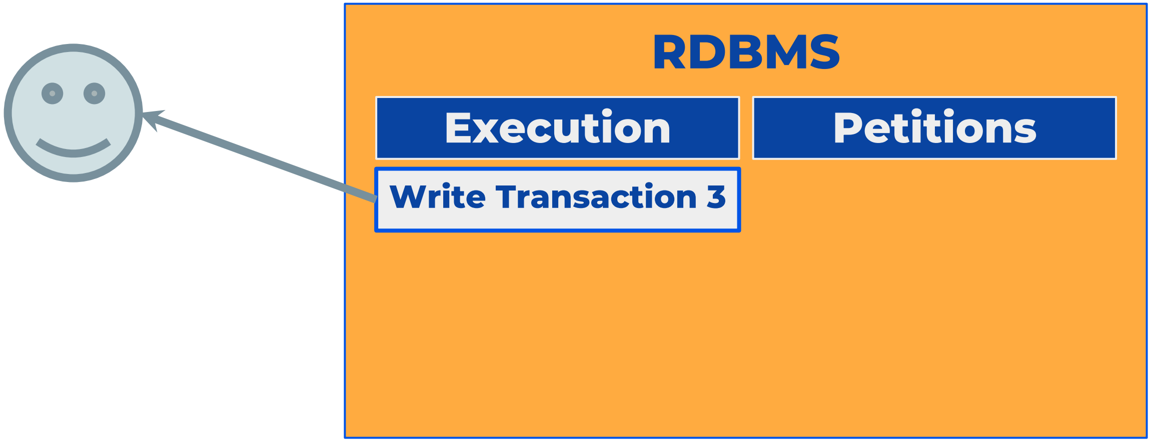
Concurrent Writing Transactions



Concurrent Writing Transactions



Concurrent Writing Transactions



Concurrent Writing Transactions

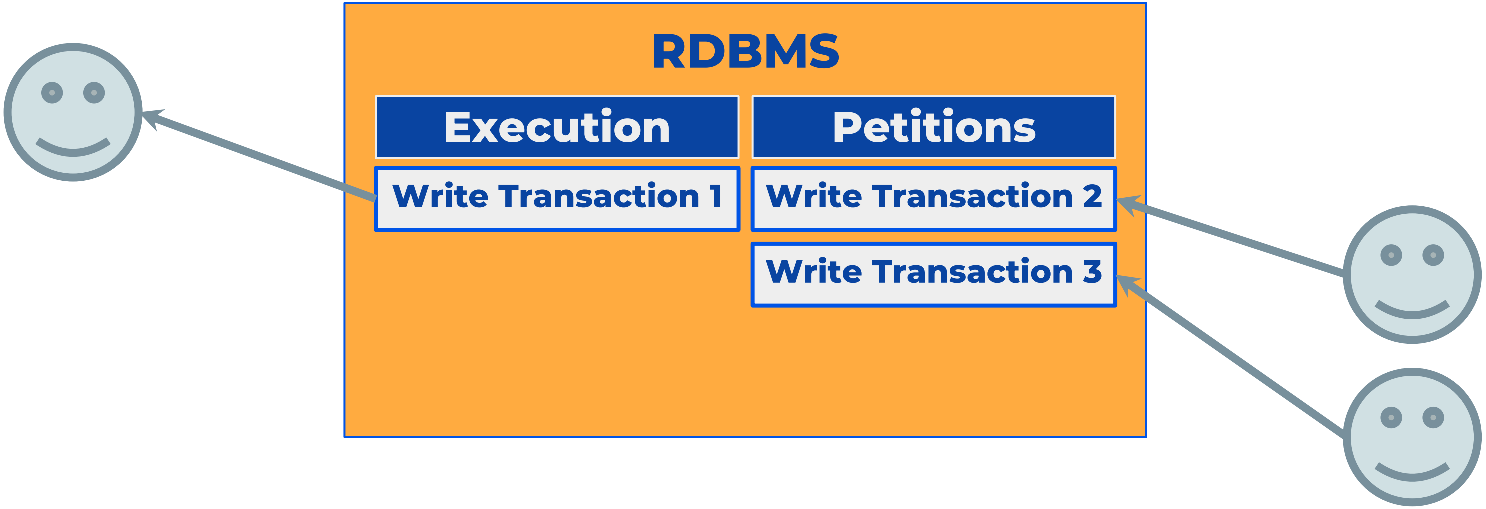
Digital Career Institute

But, again, this is not exactly true.

A transaction does not know if there is another transaction running.

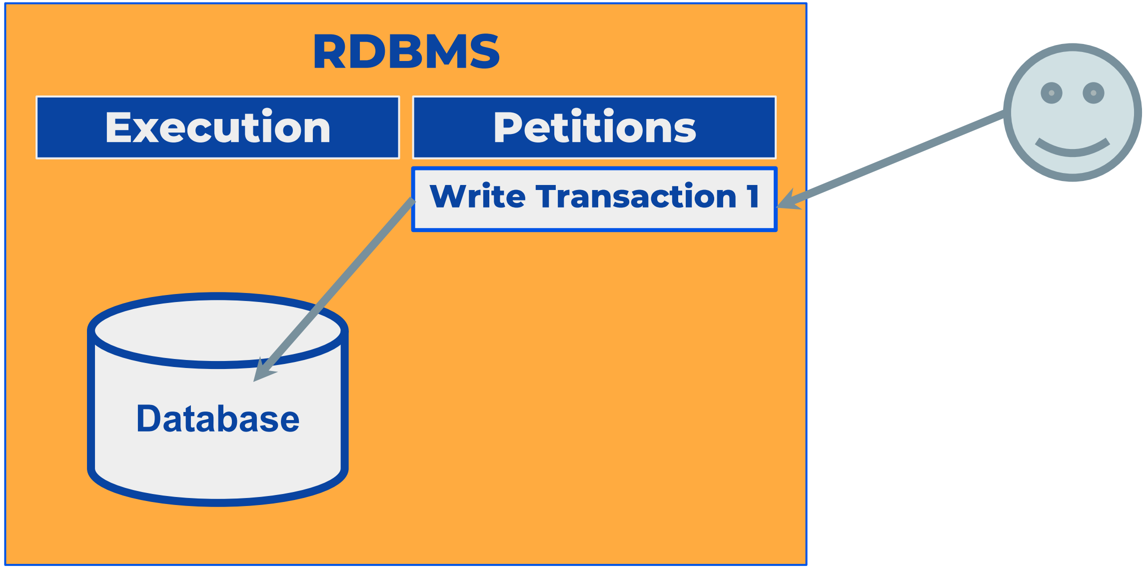
Database Locking

So how do **Write Transaction 2 & 3** know they need to wait?



Database Locking

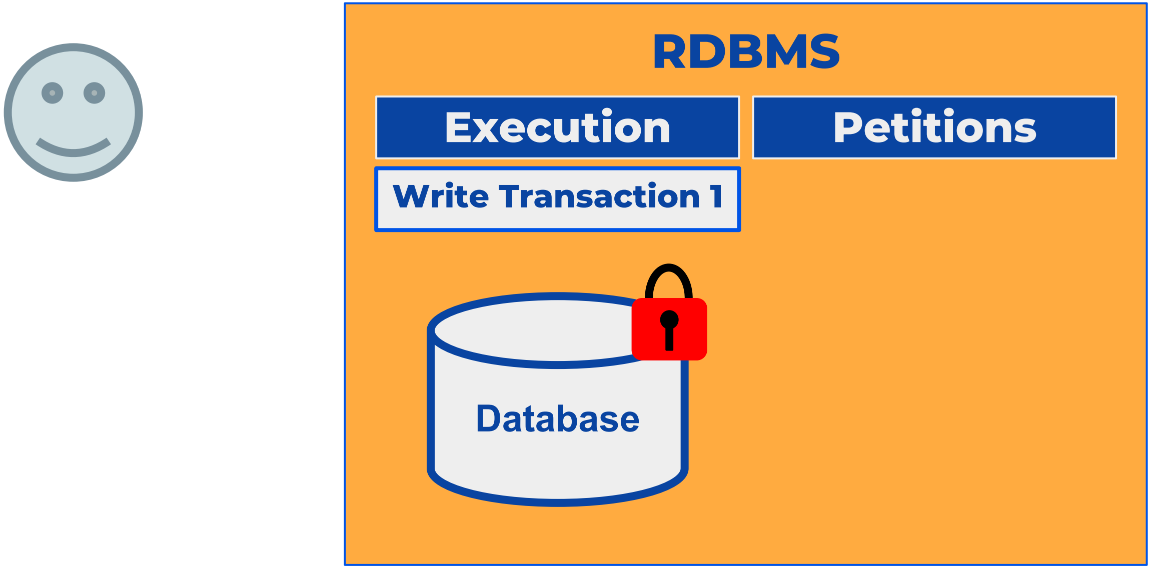
A writing transaction request arrives. It checks if it has access to the data. It does, so it executes.



Database Locking

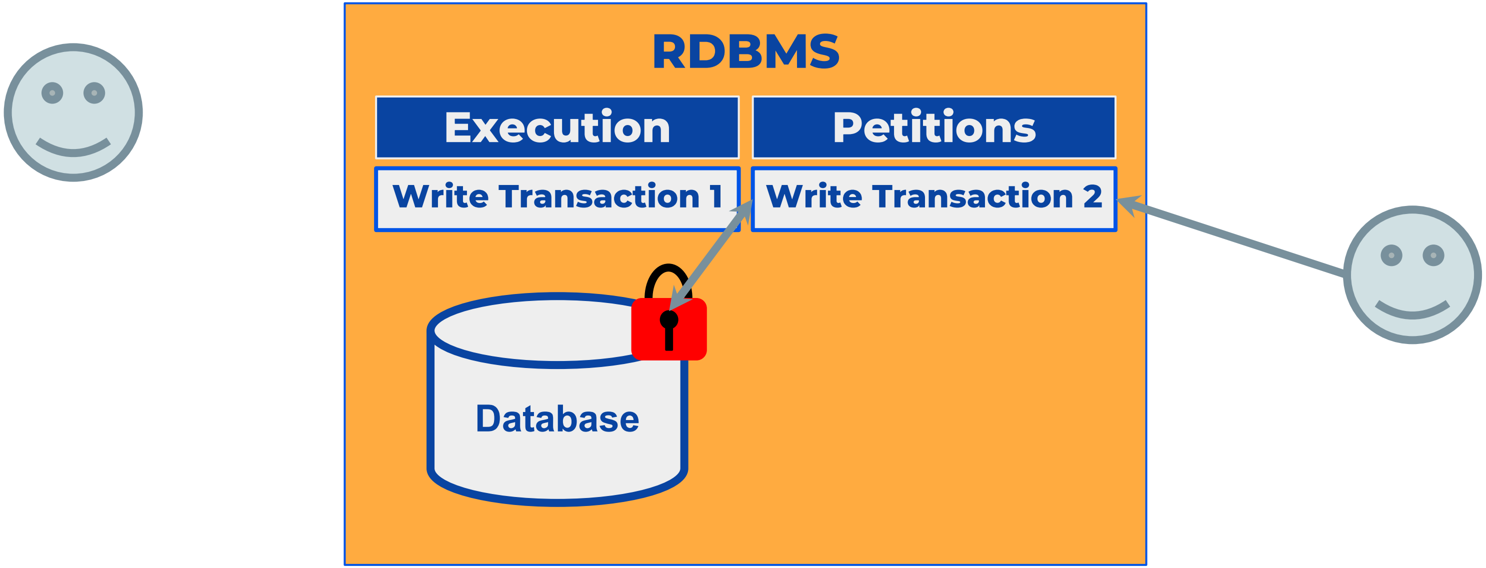
Digital Career Institute

When the transaction is executed,
a **lock** is added to the database.



Database Locking

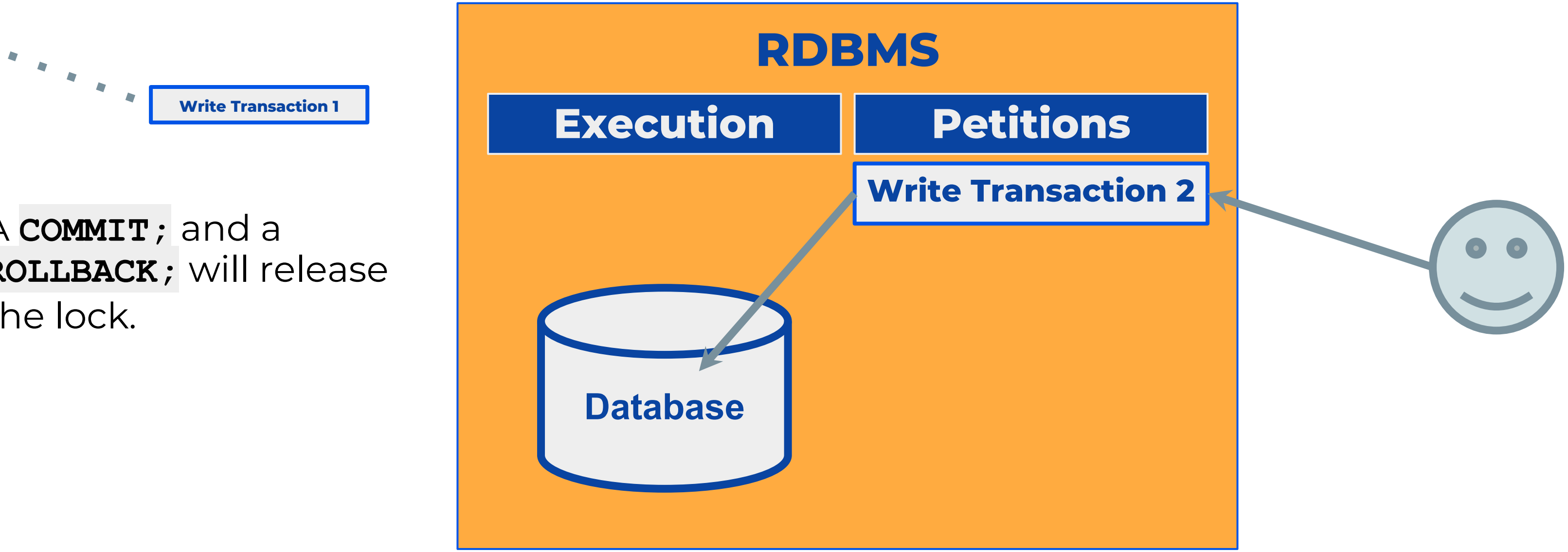
A second transaction request arrives. It checks if it has access to the data. It finds the lock, so it waits.



Database Locking

Digital Career Institute

When the first transaction finishes, the lock is released and the next transaction checks again.



A **`COMMIT;`** and a **`ROLLBACK;`** will release the lock.

Concurrent Writing Transactions

Digital Career Institute

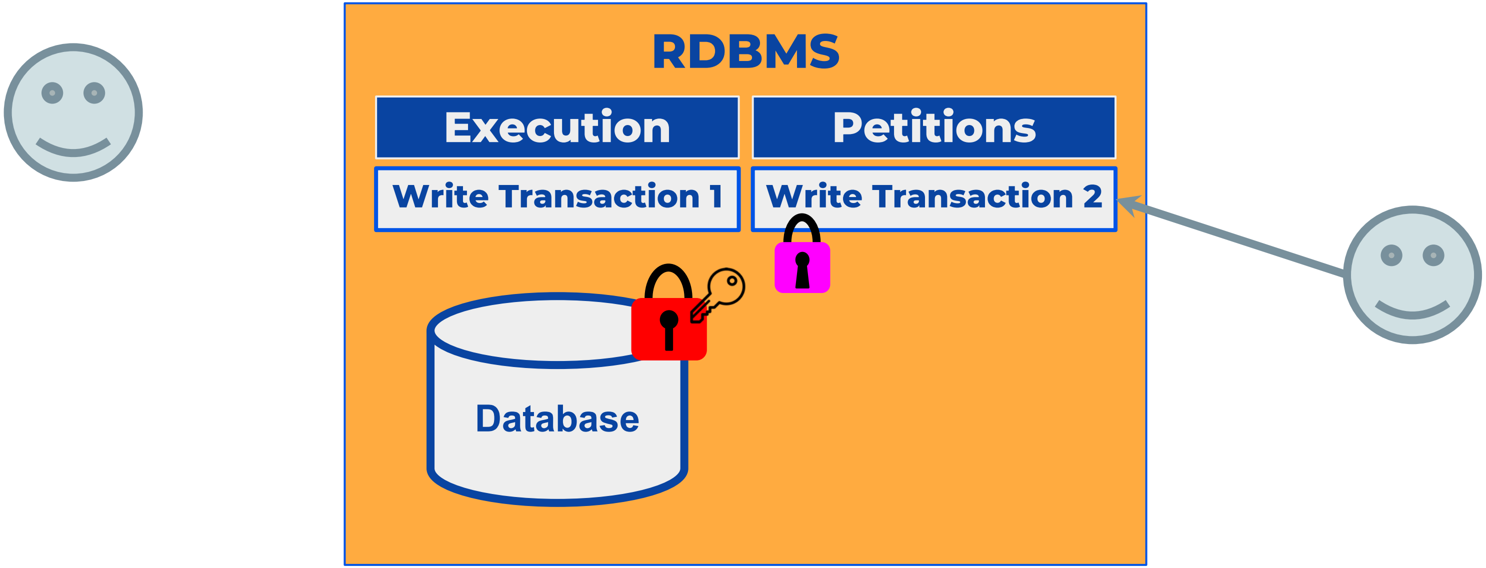
But, again, this is not exactly true.

Some writing transactions can actually be done simultaneously.

Reading transactions can often be done while a writing transaction is underway.

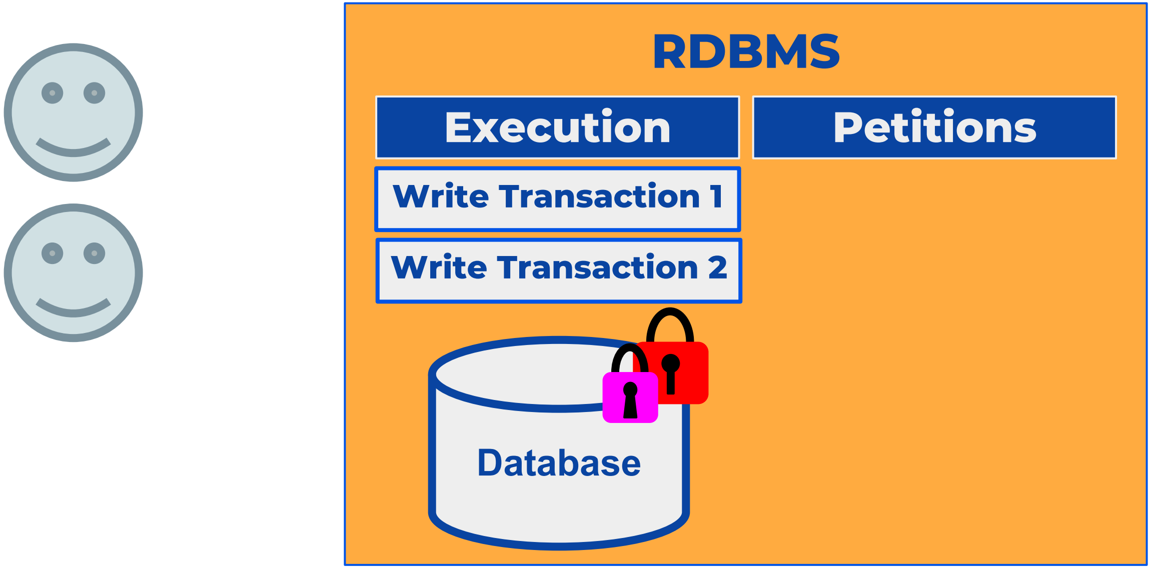
Database Locking

Each new transaction has a lock and a key.
Checks if the key fits in the current lock.



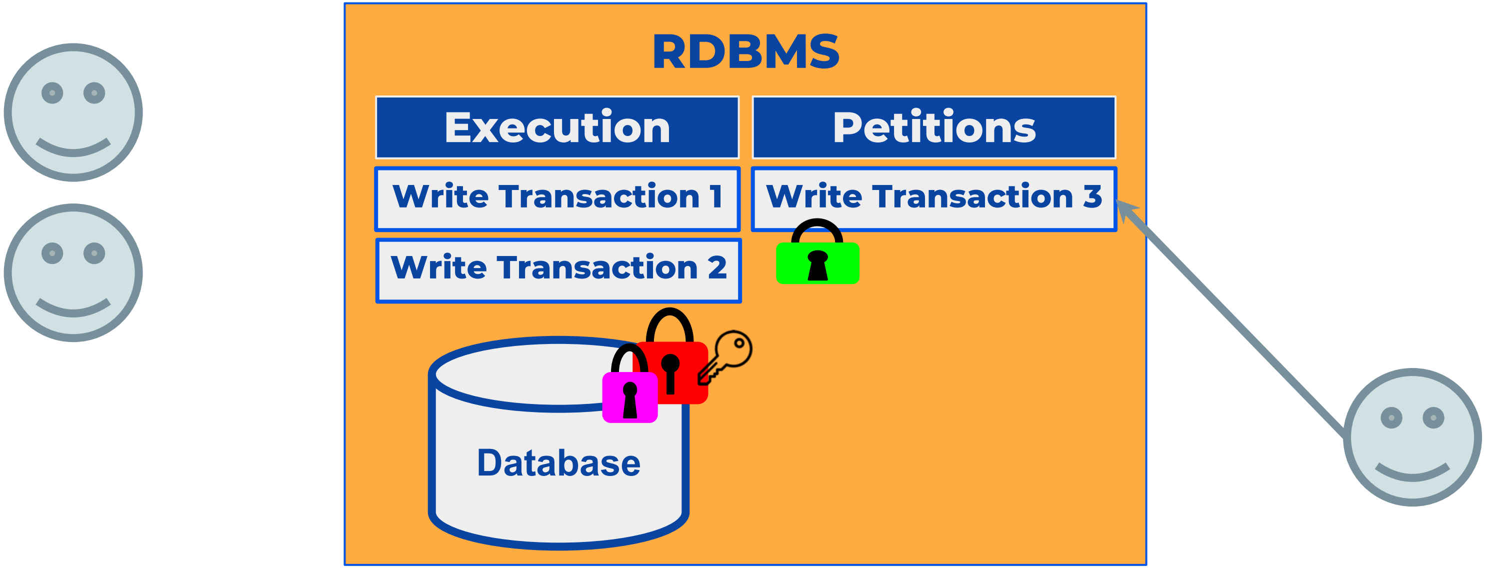
Database Locking

If the key fits, adds its own lock and executes.



Database Locking

A third transaction will try to open every lock it finds on its way to the data. If it succeeds, adds a new lock and executes.



Lock Modes in PostgreSQL

Digital Career Institute

Locks and keys are managed with **Lock Modes**.

Each operation triggers a lock of a particular mode.
Each mode is compatible with only some other modes.

ACCESS SHARE

The **ACCESS SHARE** lock mode is used by `SELECT` operations. It accesses the contents and is compatible with all lock modes, except **ACCESS EXCLUSIVE**.

ACCESS EXCLUSIVE

The **ACCESS EXCLUSIVE** lock mode guarantees that the current transaction is the only one with any kind of access. Used by `DROP TABLE` and other commands.

EXCLUSIVE

The **EXCLUSIVE** lock mode allows only reading access to the table. It is used by `REFRESH MATERIALIZED VIEW CONCURRENTLY`.

Lock Modes in PostgreSQL

Digital Career Institute

ROW EXCLUSIVE

The **ROW EXCLUSIVE** lock mode is used by `UPDATE`, `DELETE` and `INSERT` operations. It conflicts with **SHARE**, **SHARE ROW EXCLUSIVE**, **EXCLUSIVE**, and **ACCESS EXCLUSIVE** lock modes.

SHARE

The **SHARE** lock mode is triggered by `CREATE INDEX`. Conflicts with **ROW EXCLUSIVE** and all lock modes used by DDL commands (`ALTER TABLE`, `DROP TABLE`,...).

...

There are a variety of other lock modes triggered by different commands.

Locking with PostgreSQL

Locks and keys are managed with **Lock Modes**.
The most common ones are used on tables.

SELECT

The **`SELECT`** command only blocks some **`ALTER TABLE`** and all **`DROP TABLE`** and **`VACUUM`** commands.

UPDATE, INSERT, DELETE

The **`UPDATE`**, **`INSERT`** and **`DELETE`** commands block the same as the **`SELECT`**, plus **`CREATE INDEX`**.

ALTER TABLE

The **`ALTER TABLE`** command may use different locks depending on what it does. Some of them use the **ACCESS EXCLUSIVE** lock mode.

Locking with PostgreSQL

Locks can be used explicitly in a transaction.

```
BEGIN;
    LOCK TABLE accounts
        IN ROW EXCLUSIVE MODE;
    UPDATE accounts
        …
    LOCK TABLE accounts;
    UPDATE accounts
        . . .
COMMIT;
```

If no mode is indicated, it will lock the table in **ACCESS EXCLUSIVE** lock mode.

We learned ...

- The most common concurrency issues we may run into.
- That different SQL commands use different lock modes.
- That lock modes are incompatible with other lock modes.
- How locks can be used to manage concurrent conflicting transactions.
- How to define explicit locks in our transactions.

Documentation

Documentation

Transactions

- https://en.wikipedia.org/wiki/Database_transaction
- https://www.postgresql.org/docs/8.3/tutorial-transactions.html

Transaction Models: ACID & BASE

- https://en.wikipedia.org/wiki/ACID
- https://database.guide/what-is-acid-in-databases/
- https://www.section.io/engineering-education/ensuring-acid-compliance-in-database-transactions/
- https://phoenixnap.com/kb/acid-vs-base
- https://neo4j.com/blog/acid-vs-base-consistency-models-explained/

Database Concurrency

- https://docs.oracle.com/cd/B19306_01/server.102/b14220/consist.htm
- https://www.geeksforgeeks.org/concurrency-control-in-dbms/
- https://www.codemag.com/article/0607081/Database-Concurrency-Conflicts-in-the-Real-World

THANK YOU

Contact Details
DCI Digital Career Institute gGmbH