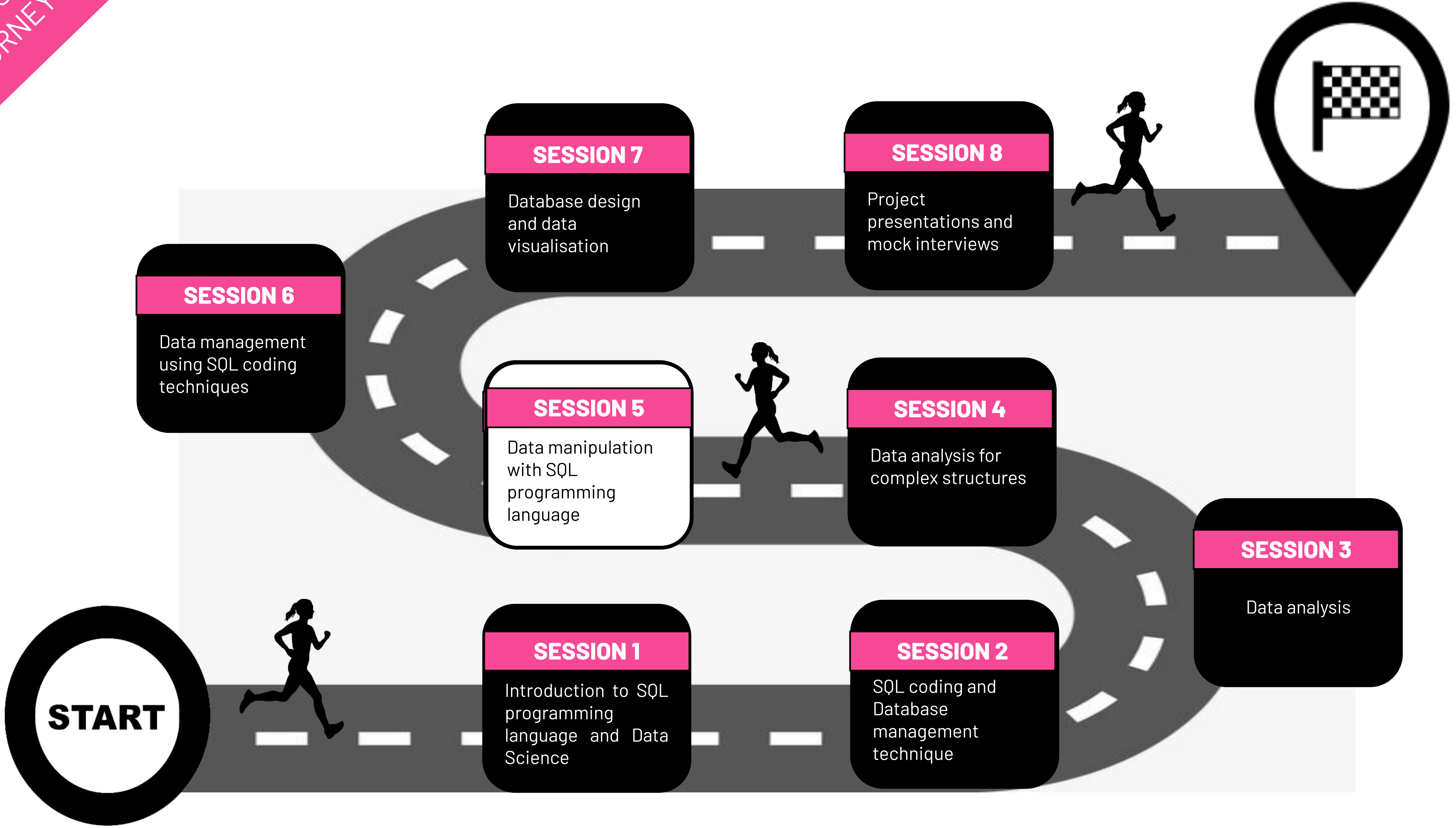
WELCOME TO CFG
YOUR INTRODUCTION TO DATABASES & SQL PROGRAMMING LANGUAGE

CODE FIRST GIRLS

TECH OPENS UP LIMITLESS OPPORTUNITIES FOR GIRLS

COURSE JOURNEY



SESSION 1
Introduction to SQL programming language and Data Science

SESSION 2
SQL coding and Database management technique

SESSION 3
Data analysis

SESSION 4
Data analysis for complex structures

SESSION 5
Data manipulation with SQL programming language

SESSION 6
Data management using SQL coding techniques

SESSION 7
Database design and data visualisation

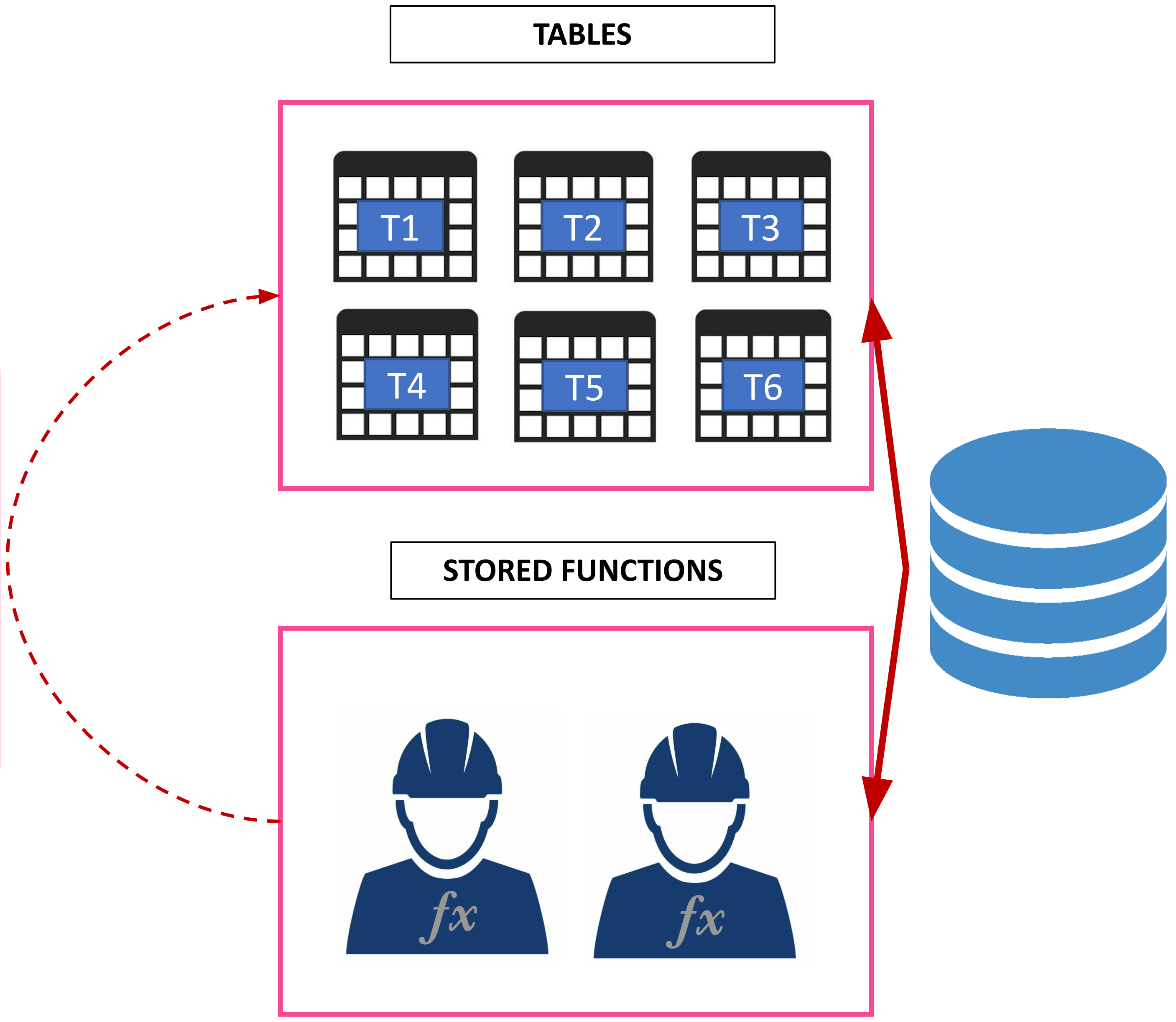
SESSION 8
Project presentations and mock interviews

- MySQL inbuilt functions
- Transaction management
- Course Project

FUNCTIONS

- Functions are encapsulated pieces of code that perform some operations and then return a result.
- Think of functions as little ‘workers’. Each of them has one particular job or task to do and they are really good at doing it.
- Every time we need a particular job to be done, we ask a dedicated ‘worker’ to perform that action.
- There are two main types of functions: **built-in** and **user-defined** functions.
- Built-ins have already been implemented for us and are readily available to be used.
- User-defined functions need to be created first and then they can be used whenever necessary.

When we write queries we can create and use new functions to retrieve and analyse data from the tables or use any in-built 'workers' to perform tasks and actions.



SYNTAX

EXAMPLE

List of Functions + Examples [here]()

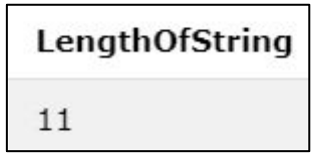
SELECT CHAR_LENGTH(<string>) AS <assigned_name>;

SELECT LOWER(<STRING>) AS <assigned_name>;

SELECT SUBSTRING(<string>, 5, 3) AS <assigned_name>;

SELECT CHAR_LENGTH("Hello World")
AS LengthOfString;

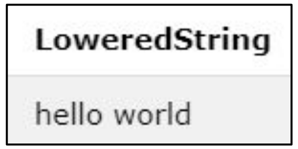
$\rightarrow$



LengthOfString
11

SELECT LOWER("HELLO WORLD")
AS LoweredString;

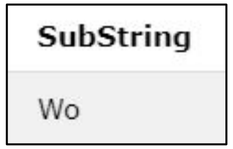
$\rightarrow$



LoweredString
hello world

SELECT SUBSTRING("Hello World", 5, 3)
AS SubString;

$\rightarrow$



SubString
Wo

PRACTICE

IN-BUILT FUNCTIONS

These functions come with MySQL server and they allow us to perform different types of manipulations on the data.

There are many different categories of in-built functions available, but the most commonly used are:

- **Strings functions** – deal with 'string' data type, in other words 'text data'
- **Numeric functions** – operate on numeric types, e.g. INT, FLOAT and others
- **Date functions** – deal with date data types

Let's practice writing queries using different types of functions together! We can see what kind of 'power' does each in-built have.

Let's try the **REPLACE** string function, the **POW** numeric function and the **ADDDATE** date function.

TRANSACTIONS

- A **transaction** is a unit of work or a sequence of operations that is performed against a database in a logical way.

- After a transaction has been run, it can be either:
  - COMMITTED – action is applied to the database
  - ROLLED BACK — action is undone from the database

- The main objective of the transaction management is to ensure that the database never contains the result of partial operations:
  - If one of the operations fails, the rollback occurs to restore the database to its original state.
  - If no error occurs, the entire set of statements is committed to the database.

- By default, MySQL automatically commits the changes permanently to the database. To force MySQL not to commit changes automatically we need to SET **autocommit** to ON or OFF state.

SYNTAX

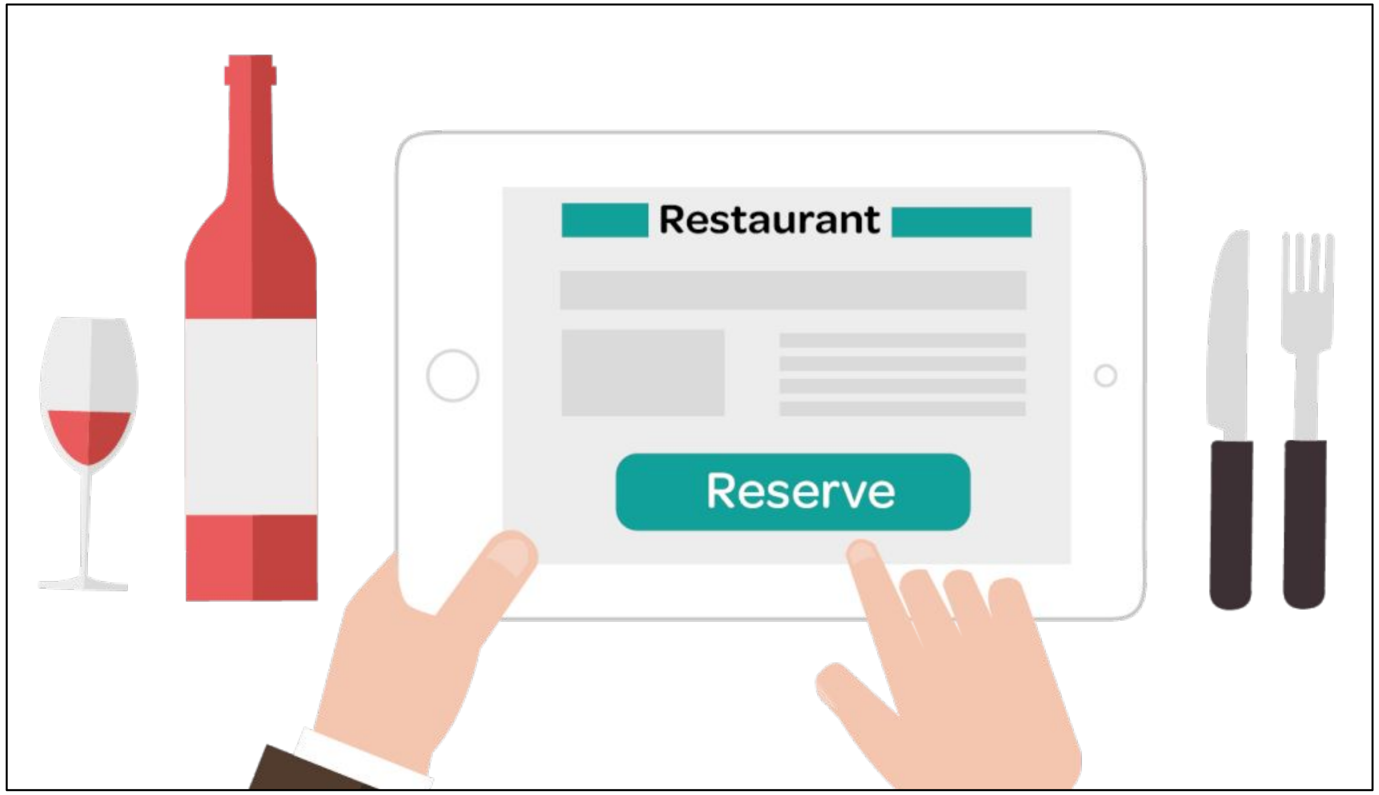
EXAMPLE

Instead of 'START TRANSACTION' you may also use such commands as 'BEGIN' or 'BEGIN WORK'. These commands are aliases and perform the same action.

```
START TRANSACTION;

<
get latest booking number,
increase it by one,
insert a new booking into a table,
check that it is available,
insert booking info into another table,
>

COMMIT;
```



PRACTICE

A good example to demonstrate how transactions work would be a real life banking transaction.

TASK

- Let's write a transaction statement imitating a transfer of £50 between two accounts.

- In order to deposit money into one account, we must first take money from another account.

- Our SQL transaction needs to do the following:
  - check that the balance of the first account is greater than £50.
  - Deduct £50 from the first account.
  - Add £50 to the second account.

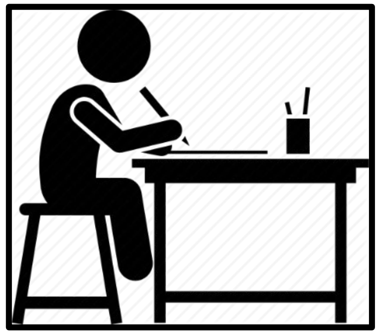
PROJECT

The project is an essential part of his course. In order to 'graduate' and get a certificate of completion you need to create, deliver and present your own database project. We are halfway through the course and have already covered the basics of SQL and relational Databases. We still have more useful topics to learn about , but we need to start thinking of the project theme.

PROJECT – TO DO LIST

- ✔ Come up with an idea for your projects (it is the most difficult part). Would it be an airline database or perhaps a travel agency or maybe a bank or a shop?
- ✔ Draft few core tables and relationships between them (remember you can ALTER and UPDATE your tables later, but it is important to make a start)
- ✔ Tables in your DB need to be normalised, so think about what kind of columns would each table have, what information are they going to store?
- ✔ Are you going to do an individual project or pair up with a fellow student to work on a DB together?

HOMEWORK



COURSE PROJECT

- Start working on your project and come up with an idea
- Decide whether it is going to be an individual projects or a group work
- By week 6 you need to submit the following:
    - Project idea and brief summary
    - Name of your project
    - Name(s) of student(s) working on the project

CODE FIRST GIRLS

SELF-STUDY TOPICS

REVISION

TABLE LOCKING

- Table locking is one of the essential practices in the SQL transaction management.
- It is an isolation requirement that allows us to lock an object, i.e. lock a table in a database to perform some updates, maintenance or any other actions.
- Think of a lock as a flag associated with a table. Whilst running MySQL session, we can acquire a lock to prevent other sessions or users to access the same table in the DB for a specific period.
- Whilst a lock is imposed on an object, others are not allowed to make any data changes, they have to wait for the lock to be released.
- Locking is designed to ensure the integrity of the data in the database.

READ LOCK

- The session that holds the READ lock can only read data from the table, but cannot write to the table.

- READ lock for a table can be acquired by multiple sessions at the same time.

- Other sessions cannot write data to the table until the READ lock is released. The write operations from another session will be put into the waiting queues until the READ lock is released

- If for any reason the session is normally or unexpectedly terminated, then MySQL will release all locks automatically.

SYNTAX

```
LOCK TABLE <table_name> READ;

<
 write queries,
 update data,
 perform actions,
>

UNLOCK TABLES;
```

EXAMPLE

- We have a register office DB that is being used by many of our colleagues to register marriages, births, partnerships and so on.
- Our task is to register births, i.e. add new entries to the two tables : 'Parent' and 'Child'. These are two separate tables, but they have primary and secondary key connections.
- We want to insert a new row into the table 'Child'.
- Due to relationships between the tables in our DB, we need to ensure that every child row refers to a parent row in the table 'Parent'.
- Can we just safely add a new child row? - NO. What if at the same time somebody is removing the relevant parent record or updating parent's name (e.g. mum has changed surname after marriage).
- We need to get READ lock for 'Parent' (we only want to ensure that nobody modifies data in 'Parent' for a moment, but they can access data) , do our update on 'Child', then safely release all locks.

WRITE LOCK

- Think of it as an exclusive lock, which gives us full writes, but prevents others from doing anything, not even seeing the data.
- The only session that holds the lock of a table can read and write data from the table.
- Other sessions cannot read data from and write data to the table until the WRITE lock is released

SYNTAX

```
LOCK TABLE <table_name> WRITE;
<
 write queries,
 update data,
 perform actions,
>
UNLOCK TABLE;


LOCK TABLES <table_name> READ,
              <table_name> WRITE;
<
perform update,
>
UNLOCK TABLES;
```

EXAMPLE

- Continuing with our 'Parent' and 'Child' tables example. We have obtained the READ lock for 'Parent' to ensure that our referential integrity is not affected.
- We need to add a new record to the 'Child' table and we need to ensure that nobody writes new data into the same table at the same time, so that there are no clashes and our record is definitely added, so the whole database is updated.
- We need to obtain WRITE lock for 'Child' table, insert new data and then release all locks.

To summarise:

- we need to obtain READ lock for 'Parent' and WRITE lock for 'Child' at the same time, perform the action and then release locks.

THANK YOU
HAVE A GREAT WEEK!

CODE FIRST GIRLS

REFERENCE MATERIALS

REVISION

QUICK SUMMARY

- A transaction is a sequential group of operations, which is seen and is performed as if it is one single unit of work.
- A transaction will never be complete unless each part of the work unit (i.e. the group) is successful.
- If any operation within the transaction fails, the entire transaction will fail, i.e. the entire transaction is rolled back, meaning that the tables and the data inside them revert to their previous state.
- Transactions help to ensure and maintain referential data integrity and minimise the number of data errors within databases.

QUICK SUMMARY

- Read locks are “shared” locks which prevent a write lock is being acquired but not other read locks.
- Write locks are “exclusive ” locks that prevent any other lock of any kind.