Schema

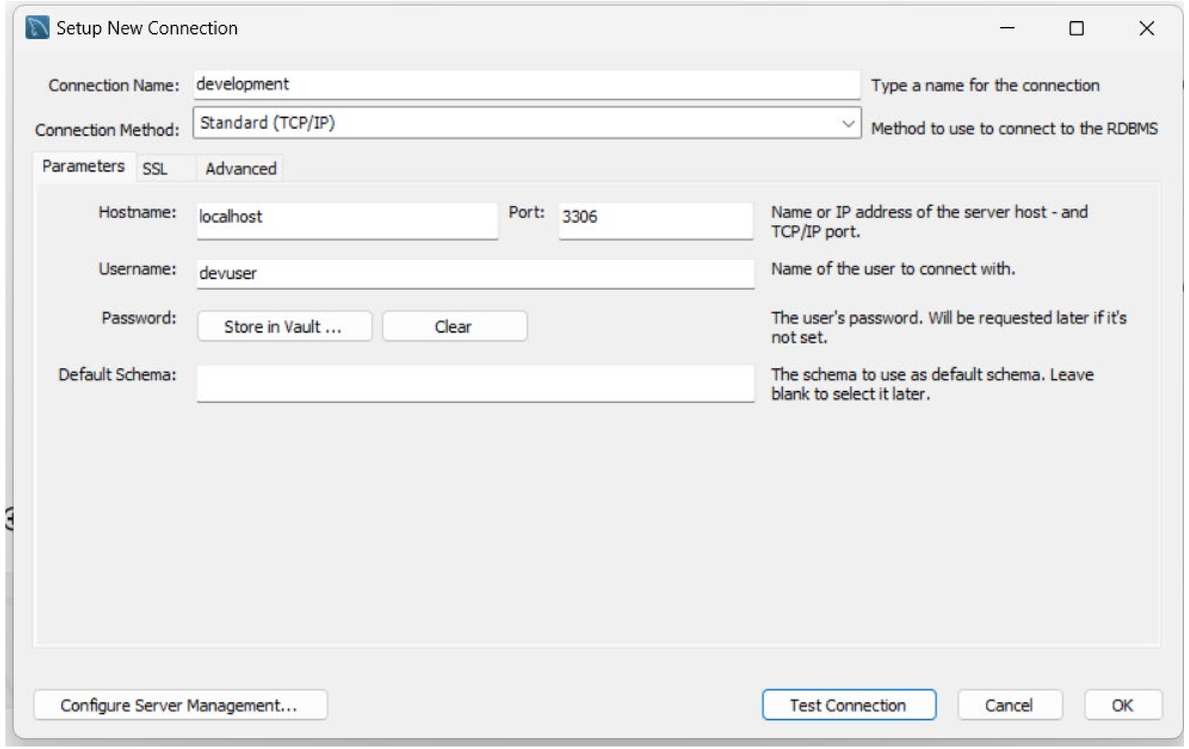
This section is for those of you who are new to databases and database terminology.

Here, I'll give you a quick tour of a Relational Database Management System, or RDBMS, and the objects we'll be working with, in case these might be unfamiliar to you.

At the highest level, data is organized into a schema.

This is used to group related database objects together.

A schema represents a logical grouping, or namespace, for database objects such as tables, views, and procedures.



Is a database a schema?

In some database systems, like PostgreSQL and Oracle, the term "schema" is used to represent the namespace within a database where objects are organized.

These systems can have multiple schemas within a single database.

In other database systems, like MySQL, the term schema is often used interchangeably with database.

In MySQL, you typically create databases, and refer to them as such, but the SQL Workbench refers to them as schemas.

Common Database Objects or Artifacts

Database Object	Description
Table	A table stores rows or records of data, in attribute fields, with values specific to that record.
Index	An index consists of a table name, a key value and a record locator field, to quickly access a record from a table. A primary key is a unique identifier for a record.
View	A view is a stored query, which can be accessed like a table, but hides the details of the table implementation from the client.
User	A user represents a package a privileges to database artifacts given to an account

Structured Query Language (SQL)

Like any software, databases also have a language you need to learn to use to work with that software.

In databases, the language lets us create objects, populate them with information, create relationships, and query data.

This language is called the Structured Query Language, or SQL.

For Relational Databases, the language has become standardized with ANSI SQL, although each database may still retain some of it's own unique clauses or commands.

SQL is divided into two categories, Data Definition Language or DDL, and Data Manipulation Language or DML, which serve two very distinct purposes.

Data Definition Language (DDL)

The Data Definition Language is used to define, create, manage, and modify the **database objects**.

DDL statements don't manipulate the data in the object, instead they manipulate data structures, that store and organize the data.

Some of the most common DDL statements are shown on this table.

SQL Command	Description
`CREATE`	Used to create database objects like tables, indexes, views, and schemas.
`ALTER`	Used to modify the structure of existing database objects.
`DROP`	Used to delete or remove database object.
`TRUNCATE`	Used to remove all rows from a table while keeping the table structure intact.
`RENAME`	Used to rename database objects.

Data Manipulation Language (DML)

The Data Manipulation Language, is used to interact with, and manipulate, the **data** stored within the database objects or artifacts.

DML statements perform operations like inserting, updating, retrieving, and deleting data in the database.

Some of the most common DML statements are shown on this table.

These commands are often represented with another acronym, you might be familiar with.

SQL Command	Description
`SELECT`	Used to retrieve data from one or more tables.
`INSERT`	Used to add new rows of data into a table.
`UPDATE`	Used to modify existing data in a table.
`DELETE`	Used to remove rows from a table.

CRUD

This acronym is CRUD, and it stands for:

Create: This operation involves creating new records or entries in a database. This is equal to an INSERT command.

Read: The read operation involves retrieving or querying data from a database. This is equal to a SELECT command.

Update: Updating refers to modifying existing data in a database. This is equal to an UPDATE command.

Delete: The delete operation involves removing records or data from a database. This is equal to a DELETE command.

CRUD represents the fundamental operations for managing data in a database system.

Later, we'll cover different Java frameworks that all aim to simplify database interactions, making it easier to perform CRUD operations in your Java projects.

Relational Database

Unlike a simple spreadsheet, data stored in an RDBMS can be stored in many different tables.

The tables can be associated with one another, through different kinds of relationships.

Relationship	Description
One to One	One row in the first table is related to only one row in a second table.
One to Many	One row in the first table is related to many rows in a second table.
Many to Many	Many rows in the first table are related to many rows in a second table.

Database Design through Normalization

Database normalization is the process of organizing data in a database, mainly to reduce data redundancy, and improve data integrity.

It's a design process which involves breaking down a large table, into smaller related tables, and defining relationships between them.

This process is guided by a set of rules, often called normal forms.

The goal of each of these forms is to progressively eliminate redundancy, meaning a data attribute should be represented in only one table, as well as dependency issues.

Once the design process is complete, you'll have a schema full of tables, with implicit or explicit relationships.

Having data in these separate silos, where no data is repeated, is great for maintenance and storage, but it complicates data retrieval.

Joins

A join is a SQL clause, that combines rows from two or more tables, based on a common field.

Joins are used to create more complex queries, and to retrieve data from multiple related tables simultaneously.

Join Type	Result
Inner join	Returns all rows from both tables where the join condition is met
Left join	Returns all rows from the left table, even if there is no matching row in the right table.
Right join	Returns all rows from the right table, even if there is no matching row in the left table.
Full join	Returns all rows from both tables, regardless of whether there is a matching row in the other table.
Cross join	Returns all possible combinations of rows from the two tables.