COURSEWARE

Professional Skills

Agile Fundamentals

Jira

Git

Databases Introduction

- Installing MySQL on Windows
- Provision a MySQL Server (Google Cloud Platform)
- Introduction to Relational Databases
- Data Design
- Data Definition Language (DDL)
- Entity-Relationship Diagrams
- Data Manipulation Language (DML)
- Data Query Language using SELECT
- Aggregate Functions
- Nested Queries
- Joins
- Data Normalisation

Java Beginner

Maven

Testing (Foundation)

Java Intermediate

HTML

CSS

Javascript

Spring Boot

Selenium

Sonarqube

Advanced Testing (Theory)

Cucumber

MongoDB

Express

NodeJS

Introduction to Relational Databases

Contents

- Overview
- Databases
  - Data
  - Schema
  - Queries
- Database Types
- Database Objects
- Relationships
- Tutorial
- Exercises

Overview

This module serves as a general introduction to relational databases and the *Relational Database Management Systems* (RDBMS) used to access them.

Databases

A *database* is an organised collection of data. All web applications need some kind of database for data storage and retrieval.

Consider a website like Amazon, which serves millions of customers; it's impossible to store all of that information in the website or code - we need a database to do this!

If we think of the actual Web site we see as the *front-end*, and the bit that does the heavy lifting in the background as the *back-end*, a database hooks into the back-end code for data. Requested data is sent from the back-end code to the database, and the database sends the appropriate data back.

Databases are generally made up of two distinct areas:

- the *data* - the actual information it stores
- the *schema* - the way that information is stored (the layout the data takes)

Data

Data stored in a database is organised into *tables* (also known as *entities*), which are modelled on real-life objects that we would store information on, such as *customers*, *products* or *orders*.

These tables then contain one or more *records* - each record is an instance, or set, of complete data.

For example, a record in the customer table would be 1 set of completed data, with all information we need to be filled for a customer.

Schema

A database *schema* defines the structure and relationships between data.

For example, the *customer* table would most likely link, or relate, in some way to the *order* table, as customers typically place orders in real life.

Therefore, we can model this relationship within our database through the schema!

React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
AWS Intermediate
Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

Queries

Varying types of *statements* are used to access the data stored in databases. The type of statement used will differ, depending on the task you are trying to achieve. For instance, to select specific records within a database, you'd use a *query*.

From there, the statement is run against the database and its data, and the database will respond to the query with the information you require.

For example, we could write a query asking to see all of the customers over the age of 35 - the database would take this query, look at the relevant tables and data, and present said information to the user.

Database Types

Typically, databases fall into two distinct types: *relational* and *non-relational*.

Relational databases focus on modelling a business structure, as well as implementing strong business rules and constraints on data and the type of information captured. Examples include MySQL, Oracle SQL, PostgreSQL, and Microsoft SQL Server.

Non-relational databases focus on the read/write capacity and capability of data, emphasising the speed in which we access data. Examples include MongoDB, Amazon's DynamoDB, and Cassandra.

Of the two, a relational database will be easier to model, while a non-relational database is easier to read from and write information to.

Database Objects

Databases need to transform data into information by adding context.

Raw data means nothing; a list of numbers, keywords, or characters may not add any value without interpretation.

```
Jeff 23
Cyrus 24
```

Databases allow for this, by categorising data into tables and adding fields to these data tables for further organising. For instance, this might be inside the *customers* table:

Name	Age
Jeff	23
Cyrus	24

- *Database* - a full data set, consisting of a collection of tables
- *Table* - an organised set of data objects
- *Field* - a way to categorise stored data
- *Record* - an instance of fields in a table

Relationships

Relational databases run on *relationships* - a way of linking tables together.

Consider a retail-based business; several different tables could be modelled, such as:

- a *customers* table to store user and account information
- a *orders* table to keep a log of purchases
- a *product* table to store data on all of the items available

Naturally, some of these tables will need to *link* in some way.

Linking tables will allow one record in a table to access another; for example, linking the customer and order table will allow us to check what customers made an order without having to duplicate data across multiple tables.

Tutorial

There is no tutorial for this module.

Exercises

Think about the type of data that a games shop would want to store in their database.

From here, think about the tables that you might find in the games shop's database. What kinds of information will they store?

▶ Solution