Chapter 23
Introduction to Databases

23.1 Introduction

There are several different types of database system in common use today including Object databases, NoSQL databases and (probably the most common) Relational Databases. This chapter focusses on Relational Databases as typified by database systems such as Oracle, Microsoft SQL Server and MySQL. The database we will use in this book is MySQL.

23.2 What Is a Database?

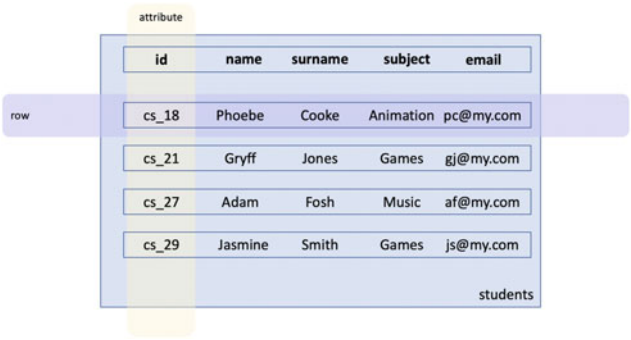
A database is essentially a way to store and retrieve data.

Typically, there is some form of query language used with the database to help select the information to retrieve such as SQL or Structured Query Language.

In most cases there is a structure defined that is used to hold the data (although this is not true of the newer NoSQL or non-relational unstructured databases such as CouchDB or MongoDB).

In a Relational Database the data is held in tables, where the *columns* define the properties or attributes of the data and each *row* defines the actual values being held, for example:

J. Hunt, *Advanced Guide to Python 3 Programming*,
Undergraduate Topics in Computer Science,
https://doi.org/10.1007/978-3-030-25943-3_23



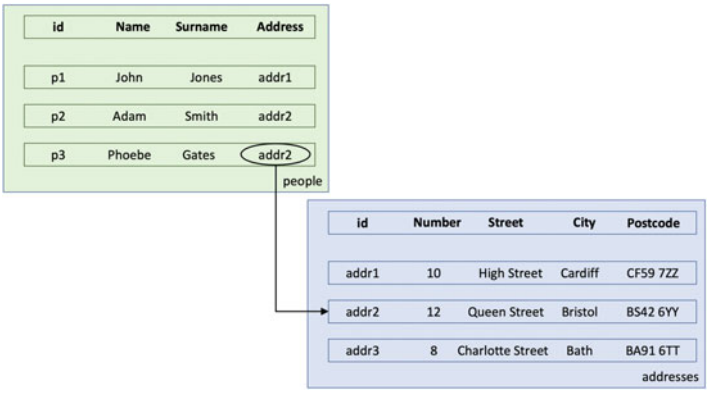
In this diagram there is a table called `students`; it is being used to hold information about students attending a meeting. The table has 5 attributes (or columns) defined for `id`, `name`, `surname`, `subject` and `email`.

In this case, the `id` is probably what is known as a *primary key*. The primary key is a property that is used to uniquely identify the student row; it cannot be omitted and must be unique (within the table). Obviously names and subjects may well be duplicated as there may be more than one student studying *Animation* or *Games* and students may have the same first name or surname. It is probable that the email column is also unique as students probably don't share an email address but again this may not necessarily be the case.

You might at this point wonder why the data in a Relational Database is called *relational* and not tables or tabular? The reason is because of a topic known as relational algebra that underpins Relational Database theory. Relational Algebra takes its name from the mathematical concept known as a *relation*. However, for the purposes of this chapter you don't need to worry about this and just need to remember that data is held in tables.

23.2.1 Data Relationships

When the data held in one table has a link or relationship to data held in another table then an index or key is used to link the values in one table to another. This is illustrated below for a table of *addresses* and a table of *people* who live in that address. This shows for example, that 'Phoebe Gates' lives at address 'addr2' which is 12 Queen Street, Bristol, BS42 6YY.



This is an example of a many to one (often written as many:1) relationship; that is there are many people who can live at one address (in the above Adam Smith also lives at address 'addr2'). In Relational Databases there can be several different types of relationship such as:

- **one:one** where only one row in one table references one and only one row in another table. An example of a one to one relationship might be from a person to an order for a unique piece of jewellery.
- **one:many** this is the same as the above address example, however in this case the direction of the relationship is reversed (that is to say that one address in the *addresses* table can reference multiple persons in the *people* table).
- **many:many** This is where many rows in one table may reference many rows in a second table. For example, many students may take a particular class and a student may take many classes. This relationship usually involves an intermediate (join) table to hold the associations between the rows.

23.2.2 The Database Schema

The structure of a Relational Database is defined using a Data Definition Language or Data Description Language (a DDL).

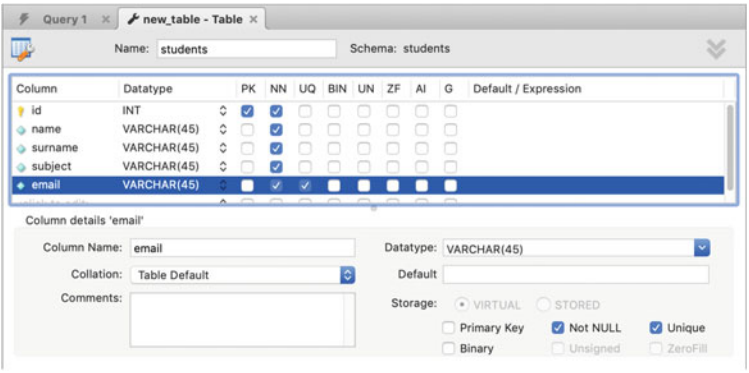
Typically, the syntax of such a language is limited to the semantics (meaning) required to define the structure of the tables. This structure is known as the database schema. Typically, the DDL has commands such as `CREATE TABLE`, `DROP TABLE` (to delete a table) and `ALTER TABLE` (to modify the structure of an existing table).

Many tools provided with a database allow you to define the structure of the database without getting too bound up in the syntax of the DDL; however, it is useful to be aware of it and to understand that the database can be created in this way. For example, we will use the MySQL database in this chapter. The MySQL

Workbench is a tool that allows you to work with MySQL databases to manage and query the data held within a particular database instance. For references for mySQL and the MySQL Workbench see the links at the end of this chapter.

As an example, within the MySQL Workbench we can create a new table using a menu option on a database:

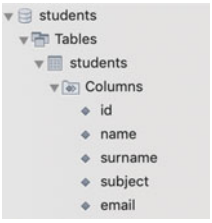
Using this we can interactively define the columns that will comprise the table:



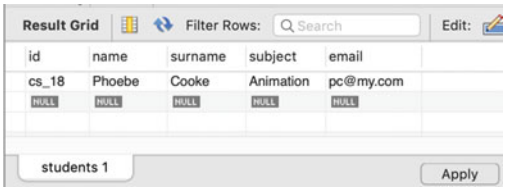
Here each column name, its type and whether it is the primary key (PK), not empty (or Not Null NN) or unique (UQ) have been specified. When the changes are applied, the tool also shows you the DDL that will be used to create the database:

```
1  CREATE TABLE `students`.`students` (
2    `id` INT NOT NULL,
3    `name` VARCHAR(45) NOT NULL,
4    `surname` VARCHAR(45) NOT NULL,
5    `subject` VARCHAR(45) NOT NULL,
6    `email` VARCHAR(45) NOT NULL,
7    PRIMARY KEY (`id`),
8    UNIQUE INDEX `email_UNIQUE` (`email` ASC));
9
```

When this is applied a new table is created in the database as shown below:



The tool also allows us to populate data into the table; this is done by entering data into a grid and hitting apply as shown below:



23.3 SQL and Databases

We can now use query languages to identify and return data held in the database often using specific criteria.

For example, let us say we want to return all the people who have the surname Jones from the following table:

id	name	surname	subject	email
cs_18	Phoebe	Cooke	Animation	pc@my.com
cs_21	Gryff	Jones	Games	gj@my.com
cs_27	Adam	Fosh	Music	af@my.com
cs_29	Jasmine	Smith	Games	js@my.com
cs_31	Tom	Jones	Music	tj@my.com

student_table

We can do this by specifying that data should be returned where the surname equals 'Jones'; in SQL this would look like:

```
SELECT * FROM students where surname='Jones';
```

The above `SELECT` statement states that all the properties (columns or attributes) in a row in the table students are to be returned where the surname equals 'Jones'. The result is that two rows are returned:

id	name	surname	subject	email
2	Gryff	Jones	Games	gj@my.com
5	Tom	Jones	Music	tj@my.com

Note we need to specify the table we are interested in and what data we want to return (the '*' after the select indicated we want all the data). If we were only interested in their first names then we could use:

```
SELECT name FROM students where surname='Jones';
```

This would return only the names of the students:

name
Gryff
Tom

23.4 Data Manipulation Language

Data can also be inserted into a table or existing data in a table can be updated. This is done using the Data Manipulation Language (DML).

For example, to insert data into a table we merely need to write an INSERT SQL statement providing the values to be added and how they map to the columns in the table:

```
INSERT INTO 'students' ('id', 'name', 'surname', 'subject',
            'email') VALUES ('6', 'James', 'Andrews', 'Games',
            'ja@my.com');
```

This would add the row 6 to the table `students` with the result that the table would now have an additional row:

id	name	surname	subject	email
1	Phoebe	Cooke	Animation	pc@my.com
2	Gryff	Jones	Games	gj@my.com
3	Adam	Fosh	Music	af@my.com
4	Jasmine	Smith	Games	js@my.com
5	Tom	Jones	Music	tj@my.com
6	James	Andrews	Games	ja@my.com

Updating an existing row is a little more complicated as it is first necessary to identify the row to be updated and then the data to modify. Thus an `UPDATE` statement includes a where clause to ensure the correct row is modified:

```
UPDATE 'students' SET 'email'='grj@my.com' WHERE 'id'='2';
```

The effect of this code is that the second row in the students table is modified with the new email address:

id	name	surname	subject	email
1	Phoebe	Cooke	Animation	pc@my.com
2	Gryff	Jones	Games	grj@my.com
3	Adam	Fosh	Music	af@my.com
4	Jasmine	Smith	Games	js@my.com
5	Tom	Jones	Music	tj@my.com
6	James	Andrews	Games	ja@my.com

23.5 Transactions in Databases

Another important concept within a database is that of a Transaction. A Transaction represents a unit of work performed within a database management system (or similar system) against a database instance, and is independent of any other transaction.

Transactions in a database environment have two main purposes

- To provide a unit of work that allows recovery from failures and keeps a database consistent even in cases of system failure, when execution stops (completely or partially). This is because either all the operations within a transaction are performed or none of them are. Thus, if one operation causes an error then all the changes being made by the transaction thus far are rolled back and none of them will have been made.
- To provide isolation between programs accessing a database concurrently. This means that the work being done by one program will not interact with another programs work.

A database transaction, by definition, must be atomic, consistent, isolated and durable:

- **Atomic** This indicates that a transaction represents an atomic unit of work; that is either all the operations in the transaction are performed or none of them are performed.
- **Consistent** Once completed the transaction must leave the data in a consistent state with any data constraints met (such as a row in one table must not reference an non-existent row in another table in a one to many relationship etc.).
- **Isolated** This relates to the changes being made by concurrent transactions; these changes must be isolated from each other. That is, one transaction cannot see the changes being made by another transaction until the second transaction completes and all changes are permanently saved into the database.
- **Durable** This means that once a transaction completes then the changes it has made are permanently stored into the database (until some future transaction modifies that data).

Database practitioners often refer to these properties of database transactions using the acronym ACID (for Atomic, Consistent, Isolated, Durable).

Not all databases support transactions although all commercial, production quality databases such as Oracle, Microsoft SQL Server and MySQL, do support transactions.

23.6 Further Reading

If you want to know more about databases and database management systems here are some online resources:

- https://en.wikipedia.org/wiki/Database which is the wikipedia entry for databases and thus acts as a useful quick reference and jumping off point for other material.
- https://en.wikibooks.org/wiki/Introduction_to_Computer_Information_Systems/Database which provides a short introduction to databases.
- https://www.techopedia.com/6/28832/enterprise/databases/introduction-to-databases another useful starting point for delving deeper into databases.
- https://en.wikipedia.org/wiki/Object_database for information on Object databases.
- https://en.wikipedia.org/wiki/NoSQL for an introduction to No SQL or non relational databases.
- https://www.mysql.com/ for the MySQL Database.
- https://dev.mysql.com/downloads/workbench The MySQL Workbench home page.
- https://www.mongodb.com/ for the home page of the MongoDB site.
- http://couchdb.apache.org/ for the Apache Couch Database.

If you want to explore the subject of database design (that is design of the tables and links between tables in a database) then these references may help:

- https://en.wikipedia.org/wiki/Database_design the wikipedia entry for database design.
- https://www.udemy.com/cwdatabase-design-introduction/ which covers most of the core ideas within database design.
- http://en.tekstenuitleg.net/articles/software/database-design-tutorial/intro.html which provides another tutorial that covers most of the core elements of database design.

If you wish to explore SQL more then see:

- https://en.wikipedia.org/wiki/SQL the wikipedia site for SQL
- https://www.w3schools.com/sql/sql_intro.asp which is the W3 school material on SQL and as such an excellent resource.
- https://www.codecademy.com/learn/learn-sql which is a codecademy site for SQL.