University Of Portsmouth
BSc (Hons) Computer Science
First Year

Database Systems Development
M30232
September 2022 - May 2023
20 Credits

Thomas Boxall
up2108121@myport.ac.uk

Contents

S.1. Introduction to module

| 📅 29-09-22 | 🕐 13:00 | 🎓 Mark | 📍 RB LT1 |

The Module Coordinator for this module is Mark (based in BK 3.09), he's assisted by Valentin and Roy in some sessions alongside others too.

Mark is using a new piece of software to make his presentations with, this is currently in the test phases and he may change back to PowerPoint if people don't like it. Slides are available on Moodle as HTML format, they can be printed to PDF files for offline viewing.

Module Aims

This module aims to help you understand where the database sits in modern systems. It does not train us to be database administrators. It gives us the skills to design a database and the knowledge of how to access it and do so safely.

This module will start from the ground up.

Learning Outcomes

- Demonstrate the fundamental principles of database design & development

- Use appropriate analysis techniques to identify the requirements of a database.

- Design and build a relational database, given a set of requirements.

- Understand how to apply data manipulation using SQL.

Historically, this module used to focus on the elements of Computer Science which relate to databases (for example, software development lifecycles). Now, it focuses on just databases.

Content Overview

This module provides an understanding of the theory of relational database design using tools standard to the industry. We will be taught how to design databases using Crows Foot Entity Relationship Diagrams and SQL to create the database. This module will also cover normalisation.

Teaching Overview

The module is a year long, worth 20 credits and has two different styles of teaching.

There will be one, one hour lecture per week. In this session, we will be taught the knowledge which we can put into practice in the following weeks practical session.

There will be one, one and a half hour practical session per week. In this session, we will practice the skills required for databases. *(N.B. This session is timetabled for two hours on the timetable, generally the lecturers will leave after an hour and a half however students can remain in the room until the end of the two hours.)*

If you are unable to make it to a lecture, you need to read the content provided on Moodle. If you are unable to make it to a practical, you need to read and do (most importantly, do) the content on Moodle; this is so you are able to complete the following practical as they all build on each other.

Resources

There are a number of resources talked through:

- Moodle - the universities Virtual Learning Environment. Notes from lectures and from practicals will be uploaded here along with quizzes and other resources.

- Google Virtual Machine - the virtual machine in which our database lives. You do not need the university VPN to access it, as it requires a SSH connection. The data is hosted by Google, the module staff have some control over the machines. More detail on this will be provided in the first practical session.

- Google Workspace

- Microsoft Office. This is available free from the university. At some point, this will include Microsoft Visio, which is useful for coursework.

Expectations

Lecturers Expectations of Students

- Turn up for lectures (from next week, the content taught in the lectures will be used in the following weeks practical sessions)

- Arrive on time (there is usually useful information given out at the beginning of sessions)

- Participate and take notes in sessions

- Catch up on sessions if you miss them

- Finish the practical work before the following weeks practical sessions

- Study for about 4 hours a week total

These things are proven to increase the likelihood that a student gets a better mark at the end of the year.

Students Expectations of Lecturers

They are nice to students; start and end sessions on time; provide students with support and feedback on work throughout the module; and to return feedback and marks on work as quickly as they can (this usually should be within two weeks).

Assessments

There are two forms of assessment in this module.

Coursework

This will be worth 50% of the overall module mark. It will be released in the next few weeks and will be due at the end of the first week after the January assessment period (probably the Friday of that week at 11pm). The content assessed will all be from the first teaching block. We will get extra marks if we include content which hasn't been taught yet.

Exam

This will be worth 50% of the overall module mark. It will take place in the May/June assessment period and be computer based. It can include anything from the entire year however we won't have to write code (probably will have to look at code and say whats wrong). It will be multiple choice questions. There will be quizzes available on Moodle which will be similar to this where we can practice.

Brief Introduction to Databases

> **Database**
>
> "A single, possiblely large, repository of data that can be used simultaneuously by many departments and users" *(Database Solutions: A Step by Step Guide to Databases - T Connolly & C Begg)*

Spreadsheets

Spreadsheets are not databases. This is because a spreadsheet cannot hold the amount of data which a database can and eventhough though using some software, a database could be shared with multiple people, it cannot be edited by multiple people simultaneously.
This also applies to Microsoft Access.

Database Management System (DBMS)

> **DBMS**
>
> "The software which interacts with the users' application programs and the database" *(Database Solutions: A Step by Step Guide to Databases - T Connolly & C Begg)*

Examples of a DBMS include PostgreSQL, MySQL, SQL Server, Oracle and Mongo DB.

Why Use a Database

An alternative to databases are file based systems.
File based systems: are old fashioned; are not necessarily digital; they often contain duplicate data; are difficult to search; are very difficult to update; have the possibility to contain different file types which may not be compatible together; are inaccessible; and security may be an issue.
A database is: a modern approach; digital; duplicates can be removed; easy to search; easy to update; comprised of only one file type; capable of having multiple levels of access control; able to limit user access.
There are times at which a Database is not suitable for the setting. In this case, it may be more suitable to use a spreadsheet.

Integrated Database Environment

In an integrated database environment, the DBMS sites as a communication hub between all nodes. The DBMS is the server on which the database is hosted.
When the database is setup correctly, you can get more information out of it than you put in.

S.2. Practical 1

📅 29-09-22   🕐 14:00   🎓 Mark & team   📍 FTC 3rd floor

Introduction to Practical sessions

Practical documents are available on Moodle, make a copy of these and store within your university Google Drive so you can edit them during the sessions and make notes.

Access Levels

In PostgreSQL, the first level of security is that a user cannot login unless they have been given access or there is a database with the same name as their username.
We don't have `sudo` access to linux, however we have full administrative access to PostgreSQL. Don't drop the database called `upxxxxxxx` (where `xxxxxxx` is replaced with student number) or anything that is owned by `postgres` as this breaks things.

PostgreSQL

PostgreSQL is ready to accept code when the prompt ends in `=#`. If you enter part of a command and press enter, the prompt will change to `-#`, this indicates that Postgres is waiting for you to finish the command.
PostgreSQL gives some useful error messages, SQL does not.

Code Editors

A code editor should be used to write SQL into, then the SQL should be copied and pasted into the Linux machine. The only thing that should be directly entered into the shell is to connect to a different database.
This is so that a. we have a copy of what we have done and b. so that if the VM is deleted, we are able to re-build our VM with less pain than if we didn't save all the code.
A recommended setup is to use VS code, with a SQL syntax extension. VS Code comes with integrated Powershell, allowing you to ssh to the VM from the same window.

SQL

SQL works like a procedural programming language, in that it reads the code inputted line by line. This also means that long and complex lines of code can be split across many lines, making it easier to read them.

Installing The First Database

Due to an issue with the image used to build the Virtual Machines, we have to create the database which we will use for the first few sessions. The code to do this was available on Moodle, copy and paste into the code editor then copy and paste again, this time into the Postgres prompt of the linux machine. This executes and creates the database, pre-populated with some sample data.

Tasks

1. List the databases in your server

```
LANGUAGE: SQL
```

```
1  \l
```

LANGUAGE: Unknown

```
1                                List of databases
2      Name       |     Owner      | Encoding | Collate |  Ctype  |   Access privileges
3  ----------------+----------------+----------+---------+---------+-----------------------
4  dsd_22         | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
5  mongo -2021- fix | mongo -2021- fix | UTF8     | C.UTF-8 | C.UTF-8 |
6  postgres       | postgres       | UTF8     | C.UTF-8 | C.UTF-8 |
7  template0      | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
8                 |                |          |         |         | postgres=CTc/postgres
9  template1      | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
10                |                |          |         |         | postgres=CTc/postgres
11 up2108121      | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
12 (6 rows)
```

2. Connect to the database

LANGUAGE: SQL

```
1  \c dsd_22
```

LANGUAGE: Unknown

```
1  You are now connected to database "dsd_22" as user "up2108121".
```

3. List everything in this database

LANGUAGE: SQL

```
1  \d
```

LANGUAGE: Unknown

```
1                  List of relations
2  Schema |             Name            |   Type   |   Owner
3  --------+-----------------------------+----------+-----------
4  public | category                    | table    | up2108121
5  public | category_cat_id_seq         | sequence | up2108121
6  public | cust_order                  | table    | up2108121
7  public | cust_order_cust_ord_id_seq  | sequence | up2108121
8  public | customer                    | table    | up2108121
9  public | customer_cust_id_seq        | sequence | up2108121
10 public | manifest                    | table    | up2108121
11 public | manifest_manifest_id_seq    | sequence | up2108121
12 public | product                     | table    | up2108121
13 public | product_prod_id_seq         | sequence | up2108121
14 public | role                        | table    | up2108121
15 public | role_role_id_seq            | sequence | up2108121
16 public | staff                       | table    | up2108121
17 public | staff_staff_id_seq          | sequence | up2108121
18 (14 rows)
```

4. List just the tables

LANGUAGE: SQL

```
1  \dt
```

```
           List of relations
Schema |      Name     | Type  |    Owner
--------+------------+-------+-----------
public | category    | table | up2108121
public | cust_order  | table | up2108121
public | customer    | table | up2108121
public | manifest    | table | up2108121
public | product     | table | up2108121
public | role        | table | up2108121
public | staff       | table | up2108121
(7 rows)
```

5. Get a list of all the customers in the `customer` table

LANGUAGE: SQL

```sql
SELECT * FROM customer;
```

LANGUAGE: Unknown

```
cust_id |   cust_fname    |    cust_lname     |          addr1          |     addr2     |
    ↪ town       | postcode  |              email
---------+----------------+-----------------+----------------------+-------------+
    ↪ ----------------+-----------+-------------------------------
      1 | Jobey          | Boeter          | 6 Claremont Park     | Truax       | La
    ↪ Mohammedia  | CV42 3EF  | jboeter0@mail.ru
      2 | York           | O'Deegan        | 882 Hooker Trail     |             |
    ↪ Chemnitz     | YA92 2OJ  | yodeegan1@nydailynews.com
      3 | Penelope       | Hexter          | 25 Jackson Lane      |             |
    ↪ Pingshan     | LY32 8LN  | phexter2@cbslocal.com
      4 | Chadd          | Franz-Schoninger | 7 Division Point    | Texas       |
    ↪ Baojia       | XA22 0UR  | cfranzschoninger3@google.com.hk
      5 | Vikky          | Eke             | 293 Colorado Drive   | Browning    |
    ↪ Kamenny Privoz | WQ12 3SF | veke4@elegantthemes.com
      6 | Marie-francoise | Currier        | 032 Eagan Junction   | Duke        |
    ↪ Waekolong    | NB52 4MV  | acurrier0@economist.com
      7 | Benedicte      | Dozdill         | 579 Dryden Terrace   |             |
    ↪ Dawuhan      | GY32 6GQ  | cdozdill1@amazon.de
      8 | Gorel          | Douthwaite      | 2946 Bluejay Parkway | Heath       |
    ↪ Sunbu        | PH02 3ZX  | edouthwaite2@feedburner.com
      9 | Berengere      | Menendez        | 06154 Jackson Way    | Doe Crossing |
    ↪ Tsagaanders  | HO82 5XL  | amenendez3@dell.com
      10 | Pelagie       | Hachard         | 1777 Hauk Center     |             |
    ↪ Jiantou      | NA52 4LM  | fhachard4@blinklist.com
      11 | Adaobi        | Musa            | 6 Clariss Ave        |             | La
    ↪  Mohammedia  | CV4 3F    | amusa9@mail.ca
(11 rows)
```

6. Choose a different table from the output of `\dt` and get a list of all the records in that table.

LANGUAGE: SQL

```sql
SELECT * FROM role;
```

LANGUAGE: Unknown

```
role_id |    role_name
---------+-----------------
    1 | Order Picker
    2 | Final Packer
    3 | Post Sales
    4 | Customer Retain
    5 | Misc
(5 rows)
```

S.3. The Database Environment

📅 06-10-22   🕐 13:00   🎓 Mark   📍 RB LT1

Data or Information

When we think about real world things, we will generally think of these in terms of information, not data. Everyone and everything has information. We have to break information down into data to be able to store it.

> **Data**
>
> Facts and statistics collected together for reference or analysis
> *(https://en.oxforddictionaries.com/definition/data)*

> **Information**
>
> The result of applying data processing to data, giving it context and meaning. Information can then be further processed to yield knowledge *(http://foldoc.org/information)*

When we need to store information in a database, we first have to break it down into data items. These can be entered into the database then pulled out again in different states. When done right, these different states should be able to tell us more information than we put in.
We also have knowledge, this is the ability to find things.

Processing Data

If we are given random data items, we can assume what they mean. For example, if we are given `1.99`; `cheeseburger`; and `Bob's Midnight Burgers`, you could assume that you could purchase a cheeseburger from an establishment called Bob's Midnight Burger for £1.99. However, this might be completely wrong! It could in fact be three un-related pieces of information or we may have mis-interpreted the information completely. This shows that it is imperative we look at the context which surrounds data, before drawing information from it.

Database Management System

The Database Management System (DBMS) is the core of the database system. Every communication to the database is done through the DBMS, this includes queries, data in and data out. The DBMS also controls access to the data and schema (which is stored within the database itself).

> **Schema**
>
> The 'blueprint' of the database.

An advantage of using a DBMS is that different users can be restricted as to what they can access; the data can easily be managed and the DBMS provides an integrated view of an enterprise's operations. The DBMS also removes the risk of inconsistent data and improves the ease with security can be controlled.

Database Languages

There are two different types of database languages (DDL and DML), each have a different purpose. SQL is both.

Before we look at DDL and DML in more detail, we first need to understand what the term 'Query' means.

> **Queries**
>
> A query is the code which interacts with the database.
> This can be to read the contents of the database, you can 'query the database'. However it is also the code that puts the data into the database and the code which is used to build the database in the first place.

DDL

Data Definition Language (DDL) allows the DBA or users to describe and name entities, attributes and relationships required for the applications that access it and associated integrity and security constraints. It is the set of commands which are used to define the structure of the database. These are the commands used to create, modify or remove database objects (e.g., tables, users and indexes). Listed below are a number of the most commonly used DDL commands.

```sql
LANGUAGE: SQL

CREATE DATABASE
CREATE TABLE
ALTER TABLE
DROP DATABASE
DROP TABLE
RENAME TABLE
```

The following is an example of SQL code which creates a new table and as part of that defines the attributes within it.

```sql
LANGUAGE: SQL

create table property_for_rent (
    Property_id varchar(4) PRIMARY KEY,
    Street varchar(14) not null,
    City varchar(10) not null,
    Postcode varchar(10) not null,
    Type varchar(6) not null,
    Rooms integer not null,
    Rent decimal(6,2) not null,
    Owner_id varchar(4) not null REFERENCES private_owner(owner_id),
    Staff_id varchar(4) REFERENCES staff(Staff_id),
    branch_id varchar(4) REFERENCES branch(Branch_id)
);
```

DML

Data Manipulation Language (DML) provides the ability to manipulate data within the database. Its commands are used to select, insert, update and delete data items within a database. Listed below are a number of the most commonly used DML commands.

When selecting attributes to display, do not use `SELECT * FROM ...` as this selects everything. Instead, use `SELECT attribute, anotherAttribute, yetAnotherAttribute FROM ...`.

Take care when entering commands, for the configuration of our Virtual Machines, we are super users within PostgreSQL. Whatever we enter will be executed without question by the machine, this includes dropping data.

```sql
LANGUAGE: SQL

DELETE
INSERT
REPLACE
SELECT
UPDATE
```

The following is an example of SQL code which queries a table based on an attribute.

```sql
LANGUAGE: SQL
1  select property_id,
2  street,
3  city,
4  postcode,
5  owner_id from property_for_rent
6  where city = 'Glasgow';
```

S.4. PRACTICAL 2

📅 06-10-22   🕑 14:00   🎓 Mark & team   📍 FTC Floor 3

Introductory Tasks

1. After getting into PostgreSQL client, list the databases.

LANGUAGE: SQL

```
\l
```

LANGUAGE: Unknown

```
     List of databases
     Name       |     Owner      | Encoding | Collate  |  Ctype  |  Access privileges
----------------+----------------+----------+---------+---------+-----------------------
 dsd_22         | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
 mongo-2021-fix | mongo-2021-fix | UTF8     | C.UTF-8 | C.UTF-8 |
 postgres       | postgres       | UTF8     | C.UTF-8 | C.UTF-8 |
 template0      | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
                |                |          |         |         | postgres=CTc/postgres
 template1      | postgres       | UTF8     | C.UTF-8 | C.UTF-8 | =c/postgres          +
                |                |          |         |         | postgres=CTc/postgres
 up2108121      | up2108121      | UTF8     | C.UTF-8 | C.UTF-8 |
(6 rows)
```

2. Connect to the `dsd_22` database.

LANGUAGE: SQL

```
\c dsd_22
```

LANGUAGE: Unknown

```
You are now connected to database "dsd_22" as user "up2108121".
```

3. List the contents of the database

LANGUAGE: SQL

```
\d
```

LANGUAGE: Unknown

```
                  List of relations
 Schema |             Name           |   Type   |   Owner
--------+----------------------------+----------+-----------
 public | category                   | table    | up2108121
 public | category_cat_id_seq        | sequence | up2108121
 public | cust_order                 | table    | up2108121
 public | cust_order_cust_ord_id_seq | sequence | up2108121
 public | customer                   | table    | up2108121
 public | customer_cust_id_seq       | sequence | up2108121
 public | manifest                   | table    | up2108121
 public | manifest_manifest_id_seq   | sequence | up2108121
 public | product                    | table    | up2108121
 public | product_prod_id_seq        | sequence | up2108121
 public | role                       | table    | up2108121
 public | role_role_id_seq           | sequence | up2108121
```

```
6 public | staff                        | table    | up2108121
7 public | staff_staff_id_seq           | sequence | up2108121
8 (14 rows)
```

5. List just the tables

LANGUAGE: SQL

```
1 \dt
```

LANGUAGE: Unknown

```
1            List of relations
2 Schema |     Name      | Type  |   Owner
3 --------+-----------+-------+-----------
4 public | category    | table | up2108121
5 public | cust_order  | table | up2108121
6 public | customer    | table | up2108121
7 public | manifest    | table | up2108121
8 public | product     | table | up2108121
9 public | role        | table | up2108121
10 public | staff       | table | up2108121
11 (7 rows)
```

The \dt command removes the sequences (which will be discussed further in a couple of weeks time).
6. Look at the structure of the role table.

LANGUAGE: SQL

```
1 \d role
```

LANGUAGE: Unknown

```
1 Table "public.role"
2 Column   |         Type         | Collation | Nullable |                  Default
3 ----------+----------------------+-----------+----------+----------------------------------------
      ↪
4 role_id  | integer              |           | not null | nextval('role_role_id_seq'::
      ↪ regclass)
5 role_name | character varying(20) |          |          |
6 Indexes:
7     "role_pkey" PRIMARY KEY, btree (role_id)
8 Referenced by:
9     TABLE "staff" CONSTRAINT "staff_role_fkey" FOREIGN KEY (role) REFERENCES role(role_id)
```

Using SQL To Access Data

Most of the commands used so far are PostgreSQL specific commands (these are the ones which begin with \).

If the output from a command is too long, PostgreSQL will show a colon (:) at the bottom of the screen. To show the next screen, press the space bar. Once all the records have been seen, the screen will show (END). At this point, hit q to exit back to the prompt. q can also be pressed at the colon to exit back to the prompt from there too.

1. Read all the records in the dsd_22 table category.

LANGUAGE: SQL

```
1 SELECT * FROM CATEGORY;
```

```
LANGUAGE: Unknown
1 cat_id |   cat_name
2 --------+-------------
3      1 | Men's Wear
4      2 | Ladies Wear
5      3 | Kid's Wear
6      4 | Outdoor
7      5 | Sport
8      6 | Health
9 (6 rows)
```

2. Run the following command and see if the output is different.

```
LANGUAGE: SQL
1 select * from category;
```

```
LANGUAGE: Unknown
1 cat_id |   cat_name
2 --------+-------------
3      1 | Men's Wear
4      2 | Ladies Wear
5      3 | Kid's Wear
6      4 | Outdoor
7      5 | Sport
8      6 | Health
9 (6 rows)
```

3. Run the following command, and see if the output is different.

```
LANGUAGE: SQL
1 select * from 'Category';
```

```
LANGUAGE: Unknown
1 ERROR:  syntax error at or near "'Category'"
2 LINE 1: select * from 'Category';
3                       ^
```

4. Run the following command and see if the output is different.

```
LANGUAGE: SQL
1 select * from "Category";
```

```
LANGUAGE: Unknown
1 ERROR:  relation "Category" does not exist
2 LINE 1: select * from "Category";
3                       ^
```

5. Run the following command and see if the output is different.

```
LANGUAGE: SQL
1 select * from 'category';
```

LANGUAGE: Unknown
```
1 ERROR:  syntax error at or near "'category'"
2 LINE 1: select * from 'category';
3                            ^
```

6. Run the following command and see if the output is different.

LANGUAGE: SQL
```
1 select * from "category";
```

LANGUAGE: Unknown
```
1 cat_id |  cat_name
2 --------+-------------
3      1 | Men's Wear
4      2 | Ladies Wear
5      3 | Kid's Wear
6      4 | Outdoor
7      5 | Sport
8      6 | Health
9 (6 rows)
```

7. Run the `\dt` command again, look at the case of the table names.
8. Run the following command (nb, this is supposed to contain non-standard quote marks as copied from the Google Doc).

LANGUAGE: SQL
```
1 SELECT * FROM "category";
```

LANGUAGE: Unknown
```
1 ERROR:  relation ""category"" does not exist
2 LINE 1: SELECT * FROM "category";
```

From these exercises, it is clear that case doesn't matter when the table name is not in quotes; and that the type of quotes used matter (there are extensions available for Google Docs which allow code to be stored in them and for it to keep its formatting).

Table Structure

To see how tables are linked together, it is possible to view the table structures. This information tells you how the attributes are linked together and what the data types and sizes of said data types are (where this is applicable).
1. Run the following command.

LANGUAGE: SQL
```
1 \d customer
```

LANGUAGE: Unknown
```
1      Table "public.customer"
2    Column    |         Type          | Collation | Nullable |                   Default
3
 ↪ ------------+-----------------------+-----------+----------+------------------------------------
```

```
  ↪
4  cust_id    | integer                    |              | not null | nextval('
     ↪ customer_cust_id_seq'::regclass)
5  cust_fname | character varying(25)  |              | not null |
6  cust_lname | character varying(35)  |              | not null |
7  addr1      | character varying(50)  |              | not null |
8  addr2      | character varying(50)  |              |          |
9  town       | character varying(60)  |              | not null |
10 postcode   | character(9)           |              | not null |
11 email      | character varying(255) |              | not null |
12 Indexes:
13     "customer_pkey" PRIMARY KEY, btree (cust_id)
14 Referenced by:
15     TABLE "cust_order" CONSTRAINT "cust_order_cust_id_fkey" FOREIGN KEY (cust_id)
     ↪ REFERENCES customer(cust_id)
```

From the output, we can see that the data type of `cust_id` is integer and the data type of `postcode` is a fixed 9 length character.

Creating new Tables in SQL

The syntax for creating a table (or relation, if we're being proper) is shown below.

```
LANGUAGE: SQL

1 CREATE TABLE tableName(
2 attributeName dataType (options),
3 attributeName dataType (options),
4 ...
5 );
```

Task

1. Create a new database with a name of your choice.

2. Connect to the database.

3. Create a new table with two attributes (one of data type INT, that is also the primary key and one that has a data type of your own choosing).

```
LANGUAGE: SQL

1 CREATE DATABASE week02;
2
3 CREATE TABLE NEWTABLE(
4 IAMNUMBER INT PRIMARY KEY,
5 IAMSTRING VARCHAR(10)
6 );
```

Now, insert a record into the table.

```
LANGUAGE: SQL

1 INSERT INTO NEWTABLE (IAMNUMBER, IAMSTRING) VALUES(12, 'cheese');
```

Now, insert another new record into the table, using the same INT value as the first record. Take note of the message which is displayed.

```
LANGUAGE: SQL

1 INSERT INTO NEWTABLE (IAMNUMBER, IAMSTRING) VALUES(12, 'ham');
```

LANGUAGE: Unknown

```
1 ERROR:  duplicate key value violates unique constraint "newtable_pkey"
2 DETAIL:  Key (iamnumber)=(12) already exists.
```

LANGUAGE: Unknown

```
1 ERROR:  duplicate key value violates unique constraint "newtable_pkey"
2 DETAIL:  Key (iamnumber)=(12) already exists.
```

S.5. Database Concepts

📅 13-10-22   🕐 13:00   🎓 Mark   📍 RB LT1

Despite the fact that the relational database model was designed by Codd in the 1970s, it is a valid system and used widely.

Key Terms

Database Term	Description
Entity	An object or a 'thing' about which data is stored.
Attributes	Some quality associated with the entity (eg ID number, user-name, size). These have data types (eg number, string etc) and maximum sizes. Other terms are elements and properties.
Relation	A two dimensional representation (table) of entities and/ or relationships. Other terms used are relation table or table.
Entity Set	A set of entities of the same type.
Relationship	How two relations (tables) are related to each other. Relationships are represented in relations.
Tuple	Corresponds to rows of the table or records of a relation. Other terms used are record and row.
Domain	A pool of all legal values from which actual attribute values are drawn.
Primary Key	An attribute or combination of attributes for which values uniquely identify tuples in the relation. The primary key is chosen from a set of candidate keys. If you have a numeric value which the system can generate, let it do it for you.
Candidate Key	There may be more than one potential primary keys for a relation. Each is called a candidate key or super-key.
Alternate Key	An alternate access path to data that is not via the primary key.
Composite Key	A combination of attributes that act as a candidate key in a relation. Each participating attribute in the composite key (also known as candidate key) is called a simple key.
Foreign Key	An attribute (or combination of attributes) that is a primary key in another relation. They can appear many times.
Degree	Number of attributes in a relation; also called the arity.

When designing a database, the first thing you need to think about is what entities do you need to store information about. Then think about the attributes which you need to store about each entity. Then create relations. At this point, think about the domain for any of the attributes (for example, month 1-12 or day 0-6 (Sunday to Saturday) or hours 0-23). Now think about keys.

Entity

An entity is a thing, it could be a person or a specific type of person.
To identify entities, look at the information given to you and identify the nouns. The nouns give an idea of what the entities look like but they require fine tuning.
There can be as many entities as needed.
We can describe entities using their attributes.

We now think about keys.

Primary Key

To identify what will be a primary key, we look for something that is unique. This should be something which cannot be changed. If there is nothing suitable, create your own primary key.

Foreign key

Does not have to be primary key in other table, however it has to be unique within the other table.

S.6.   PRACTICAL 03

📅 13-10-22   🕐 14:00   🎓 Mark and team   📍 FTC Floor 3

Q1. using the `count()` function demonstrated by your tutor, how many records are there in each of the tables in the dsd_22 database. (Remember to use `\dt` to give you a list of tables in the database.) Copy the outputs below.

```
LANGUAGE: Unknown

dsd_22=# select count(*) from category;
count
-------
    6
(1 row)
dsd_22=# select count(*) from cust_order;
count
-------
150
(1 row)
dsd_22=# select count(*) from customer;
count
-------
11
(1 row)
dsd_22=# select count(*) from manifest;
count
-------
150
(1 row)

dsd_22=# select count(*) from product;
count
-------
100
(1 row)

dsd_22=# select count(*) from role;
count
-------
    5
(1 row)

dsd_22=# select count(*) from staff;
count
-------
10
(1 row)
```

Q2. Use the max() function to find the highest value of the role_id attribute in the role table. Copy the output below

```
LANGUAGE: SQL

select max(role_id) from role;
```

```
LANGUAGE: Unknown

max
-----
    5
(1 row)
```

Q3. Insert a new row of data into the role table with

```
LANGUAGE: SQL
1 INSERT INTO ROLE (ROLE_NAME) VALUES ('Pre Sales');
```

Q4. How many rows of data are now in the role table? Copy it below.

```
LANGUAGE: SQL
1 select count(*) from role;
```

```
LANGUAGE: Unknown
1 count
2 -------
3      6
4 (1 row)
```

Q5. What is the maximum value of the role_id now? Copy it below.

```
LANGUAGE: SQL
1 select max(role_id) from role;
```

```
LANGUAGE: Unknown
1 max
2 -----
3      6
4 (1 row)
```

Q6. Delete this new row with

```
LANGUAGE: SQL
1 DELETE FROM ROLE WHERE ROLE_NAME = 'Pre Sales';
```

```
LANGUAGE: Unknown
1 DELETE 1
```

Q7. How many rows of data are now in the role table? Copy it below.

```
LANGUAGE: Unknown
1 count
2 -------
3      5
4 (1 row)
```

Q8. What is the maximum value of the role_id now? Copy it below.

```
LANGUAGE: Unknown
1 max
2 -----
3      5
4 (1 row)
```

Q9. Reinsert the row of data into the role table again with

```
LANGUAGE: SQL

1 INSERT INTO ROLE (ROLE_NAME) VALUES ('Cleaning Team');
```

```
LANGUAGE: Unknown

1 INSERT 0 1
```

Q10. How many rows of data are now in the role table? Copy it below.

```
LANGUAGE: Unknown

1 count
2 -------
3       6
4 (1 row)
```

Q11. What is the maximum value of the role_id now? Copy it below.

```
LANGUAGE: Unknown

1 max
2 -----
3     6
4 (1 row)
```

Q12. Create a random value using the random function. Copy the value below

```
LANGUAGE: SQL

1 SELECT RANDOM();
```

```
LANGUAGE: Unknown

1        random
2 ------------------
3  0.175315219908953
4 (1 row)
```

Q12a. Create another random number. Copy the value below

```
LANGUAGE: SQL

1 SELECT RANDOM();
```

```
LANGUAGE: Unknown

1        random
2 ------------------
3  0.272884896956384
4 (1 row)
```

Q13. Create one more random value but now multiply it by 11. Remember that to multiply you do not use x but use the * symbol. Run this code 5 times and copy the values below.

```
LANGUAGE: Unknown

1 dsd_22=# select random()*11;
2      ?column?
3 -----------------
```

```
4    9.60335403773934
5   (1 row)
6   dsd_22=# select random()*11;
7          ?column?
8   -------------------
9    0.160588529892266
10  (1 row)
11
12  dsd_22=# select random()*11;
13         ?column?
14  ------------------
15   5.25661591161042
16  (1 row)
17
18  dsd_22=# select random()*11;
19         ?column?
20  ------------------
21   7.78145408304408
22  (1 row)
23  dsd_22=# select random()*11;
24         ?column?
25  ------------------
26   10.1819118564017
27  (1 row)
```

Q14. Connect to your home database, upxxxxxxx and run the following code to create a new table and insert some random numbers into it.

```
LANGUAGE: SQL
1  create table numb1(numb_id int primary key, ran_val decimal(17,15));
2
3  insert into numb1(numb_id, ran_val) values
4  (1,random()),(2,random()),(3,random()),(4,random()),(5,random()),(6,random()),(7,random())
     ↪ ,(8,random()),(9,random()),(10,random());
```

```
LANGUAGE: Unknown
1      INSERT 0 10
```

Q14a. Check that there are 10 rows of data with `SELECT COUNT(*) FROM NUMB1;` If not, check your output for any error messages. You should get responses below except the prompt will be your student id number.

```
LANGUAGE: Unknown
1    count
2   -------
3      10
4   (1 row)
```

Q15. Run a `SELECT * FROM NUMB1;` Copy the output below.

```
LANGUAGE: Unknown
1    numb_id |       ran_val
2   ---------+-------------------
3          1 | 0.481754711363465
4          2 | 0.020102311857045
5          3 | 0.541421711910516
6          4 | 0.046512784436345
7          5 | 0.842869907151908
8          6 | 0.137599688488990
9          7 | 0.925696460530162
10         8 | 0.765472991392016
11         9 | 0.712954005226493
12        10 | 0.161490791942924
```

```
3 (10 rows)
```

Q15a. Compare the values that you get with the values below. They should be different. This is because the code used inserts a fixed value, the numb_id and a completely random value into the ran_val attribute for each row.

LANGUAGE: Unknown

```
1  test_num=# SELECT * FROM NUMB1;
2   numb_id |      ran_val
3  ---------+-------------------
4       1 | 0.477631121408194
5       2 | 0.978080025874078
6       3 | 0.516494689509273
7       4 | 0.849129045847803
8       5 | 0.484937957022339
9       6 | 0.895700289402157
10      7 | 0.852438564877957
11      8 | 0.727535046637058
12      9 | 0.062769805546850
13     10 | 0.594313766807318
14 (10 rows)
```

Q16. Find the highest value of ran_val using the max() function. Copy it below.

LANGUAGE: SQL

```
1 select max(ran_val) from numb1;
```

LANGUAGE: Unknown

```
1         max
2  -------------------
3   0.925696460530162
4  (1 row)
```

Q17. Find the lowest value of ran_val using the min() function. Copy it below.

LANGUAGE: SQL

```
1 select min(ran_val) from numb1;
```

LANGUAGE: Unknown

```
1         min
2  -------------------
3   0.020102311857045
4  (1 row)
```

Q18. What is the average value of ran_val. Reminder: look at the basic functions document for ideas.

LANGUAGE: SQL

```
1 select avg(ran_val) from numb1;
```

LANGUAGE: Unknown

```
1           avg
2  ------------------------
```

```
3  0.46358753642998640000
4 (1 row)
```

Q19. What is the current timestamp on your server? Copy it below

```
LANGUAGE: SQL

1 select now();
```

```
LANGUAGE: Unknown

1                 now
2 ------------------------------
3  2022-10-13 13:43:49.196518+00
4 (1 row)
```

Q20. What is the first name of the customer with the ID number of 3?

```
LANGUAGE: SQL

1 select cust_fname from customer where cust_id=3;
```

```
LANGUAGE: Unknown

1  cust_fname
2 ------------
3  Penelope
4 (1 row)
```

Q21. What is the category id number of the outdoor category? Copy below.

```
LANGUAGE: SQL

1 select cat_id from category where cat_name='Outdoor';
```

```
LANGUAGE: Unknown

1  cat_id
2 --------
3       4
4 (1 row)
```

Q22. How many orders in the cust_order table are for cust_id 15? Copy below.

```
LANGUAGE: SQL

1 select count(*) from cust_order where cust_id=15;
```

```
LANGUAGE: Unknown

1  count
2 -------
3      0
4 (1 row)
```

Q23. List the first and last names of the staff members who live in Portsmouth. Copy below.

LANGUAGE: SQL

```sql
select staff_fname, staff_lname from staff where town='Portsmouth';
```

LANGUAGE: Unknown

```
 staff_fname | staff_lname
-------------+-------------
 Niel        | Welsby
 Janeva      | Gillicuddy
(2 rows)
```

Q24. What values does addr1 and addr2 have for the staff member whose id = 4? Copy below.

LANGUAGE: SQL

```sql
    select addr1 , addr2 from staff where staff_id=4;
```

LANGUAGE: Unknown

```
      addr1       | addr2
------------------+-------
 959 Algoma Plaza |
(1 row)
```

Q25. How many members of staff have the role value of 3? Copy below.

LANGUAGE: SQL

```sql
select count(*) from staff where role=3;
```

LANGUAGE: Unknown

```
 count
-------
     3
(1 row)
```

Q26. How many products are in the product category = 2?

LANGUAGE: SQL

```sql
select count(*) from product where prod_id=2;
```

LANGUAGE: Unknown

```
 count
-------
     1
(1 row)
```

S.7. Coursework & Entity Relationship Diagrams

📅 20-10-22   🕐 13:00   🎓 Mark   📍 RB LT1

Coursework

> **Coursework**
>
> The coursework is now available on Moodle, within Assessment and Support Materials.

The deadline for the coursework isn't until February.

It is recommended to submit the files to Moodle well in advance of the deadline because there is a chance there will be a technical issue with Moodle when the deadline is, no extenuating circumstances will be given if this is the case.

The Entity Relationship Diagram submitted must be produced digitally, hand drawn diagrams will gain 0 credits.

Mark uses Mocakroo and Lucid Charts for generating dummy data and drawing ERDs respectively. This is what works well for him, there are other platforms available for both, with more information in the Coursework document.

Entity Relationship Diagrams

Entity Relationship Diagrams (ERDs) are diagrams which show how entities are related, down to the detail of what the attributes are and how they relate to each other as well.

Business Rules

When designing databases, business rules will be taken into consideration.

> **Business Ruls**
>
> A statement that defines how a company does stuff or how stuff works within a company.

We can use business rules to help guide us on how to design databases.

Relationship Links

We will be using Crows Foot Notation, there are a number of other types of notation however we won't look at any of these.
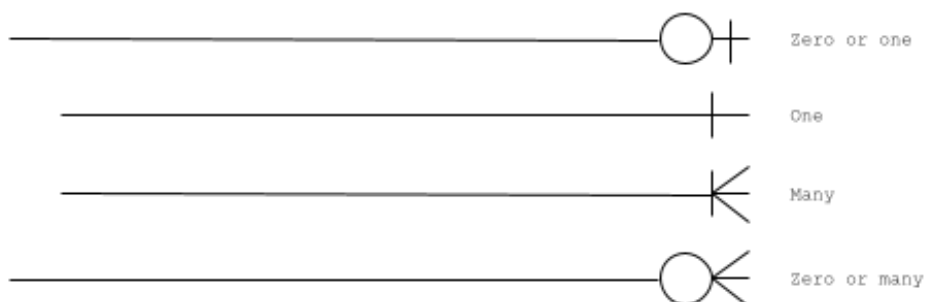
Figure 1: Crow's feet notation

When designing entities, it is important to name them in the singular, for example `pig` not `pigs`, and to use underscore notation where multiple words comprise the entity name, not camel notation. Many-to-many relationships are not permitted. We will return to this in a future lecture.

Constraints

> **Constraint**
>
> A rule that protects your data or enforces certain behaviour.

For example, a constraint may be set to be `NOT NULL`, this would ensure that whenever a row of data is inserted into a table, that attribute would have to contain data.

Keys are constraints. The primary key is automatically set to be `NOT NULL`, we do not have to specify that when creating a table. We could use a default constraint, to specify the the time that a record was entered into a table.

Check constraints can be used to validate data as it is entered, for example a price must contain two decimal places. Check may be needed as part of the coursework.

S.8. Practical: SQL and Entities

📅 20-10-22   🕑 14:00   🎓 Mark & Team   📍 FTC 3rd Floor

Task 1: Run the provided code and observe the outputs.

Run the following DDL code.

```sql
LANGUAGE: SQL
1  CREATE DATABASE customer_db;
2
3  \c customer_db
4
5  CREATE TABLE customer1 (cust_id SERIAL PRIMARY KEY, cust_fname VARCHAR(20) NOT NULL,
       ↪ cust_lname VARCHAR(20) NOT NULL);
6
7  \d customer1
8
9  ALTER TABLE customer1 ADD COLUMN cust_email varchar(100) NOT NULL UNIQUE;
10
11 \d customer1
12
13 DROP TABLE customer1;
14
15 -- check that the table is gone now
16
17 -- anything in the line after the two dashes is a comment by the way
18
19 \l
20 \d customer1
```

Run the following DML code.
Creating a new table and populating it with some dummy data.

```sql
LANGUAGE: SQL
1  CREATE TABLE customer (cust_id SERIAL PRIMARY KEY, cust_fname VARCHAR(20) NOT NULL,
       ↪ cust_lname VARCHAR(20) NOT NULL, cust_email varchar(60) NOT NULL);
2
3  INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (22,'Kamil', '
       ↪ Novak','kamnovak@gmail.com');
4
5  INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (66,'Aarav', '
       ↪ Anand','aanand98@gmail.com');
6
7  INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (67,'Alia', '
       ↪ Anand','aanand98@gmail.com');
```

Viewing what is in the table

```sql
LANGUAGE: SQL
1  SELECT * FROM customer;
2
3  SELECT cust_fname, cust_email from customer;
```

Selecting only the attributes which we need, so we don't have to retrieve all of the data from a table.

```sql
LANGUAGE: SQL
1  SELECT cust_email, cust_id, cust_fname, cust_lname from customer;
```

Insert more records, some of these return errors.

```sql
LANGUAGE: SQL
1  UPDATE customer SET cust_email = 'i_love_elephants_1@gmail.com' where cust_id = 66;
2  SELECT * FROM customer;
3
4  INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (67,'Alia', '
       ↪ Anand','aanand98@gmail.com');
5
6  INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (70,'Connor', '
       ↪ Murphy','connormurphy99@gmail.com');
7
8  INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (71,'Connor', '
       ↪ Murphy','connormurphy199@gmail.com');
9
10 INSERT INTO customer (cust_id, cust_fname, cust_lname, cust_email) VALUES (1,'Tim', 'Nice-
       ↪ but-Dimm','alongemailaddresswillfit@quitealongdomainnametoo.com');
11
12 INSERT INTO customer ( cust_fname, cust_lname, cust_email) VALUES ('Tim', 'Nice-but-Dimm',
       ↪ '
       ↪ averylongemailaddresswillnotfit1111111111111111111111111111111111111111a@quitealongdomainnametoo
       ↪ .com');
```

Task 2: Write SQL code for the following questions.

1. Create a new database called `code_test`

```sql
LANGUAGE: SQL
1  CREATE DATABASE code_test;
```

2. Connect to this new database

```sql
LANGUAGE: SQL
1  \c code_test
```

3. Create a new table called `table_one`, which has the following attributes

   (a) `Record_id` an integer
   (b) `Att_1` a varchar that will hold upto 30 characters
   (c) `Att_2` a char that will hold 10 characters
   (d) `Att_3` a decimal that can hold the value of 9.99.

```sql
LANGUAGE: SQL
1  CREATE TABLE table_one(Record_id INT PRIMARY KEY, Att_1 VARCHAR(30), Att_2 CHAR(10),
       ↪   Att_3 DECIMAL(3,2));
```

4. Look at the structure of this table once you have created it. Show the output below.

```sql
LANGUAGE: SQL
1  \d table_one
```

```
LANGUAGE: Unknown
1                  Table "public.table_one"
2    Column   |         Type          | Collation | Nullable | Default
3  -----------+-----------------------+-----------+----------+---------
4   record_id | integer               |           | not null |
5   att_1     | character varying(30) |           |          |
6   att_2     | character(10)         |           |          |
```

```
7  att_3       | numeric(3,2)            |              |             |
8 Indexes:
9     "table_one_pkey" PRIMARY KEY, btree (record_id)
```

5. Alter the table by adding a new column called `Att_4` that will hold another integer.

LANGUAGE: SQL

```sql
1 ALTER TABLE table_one ADD COLUMN Att_4 INT;
```

6. Look at the structure of this table again once you have added this new column. Show the output below.

LANGUAGE: SQL

```sql
1 \d table_one
```

LANGUAGE: Unknown

```
1                    Table "public.table_one"
2   Column   |          Type          | Collation | Nullable | Default
3 -----------+------------------------+-----------+----------+---------
4  record_id | integer                |           | not null |
5  att_1     | character varying(30)  |           |          |
6  att_2     | character(10)          |           |          |
7  att_3     | numeric(3,2)           |           |          |
8  att_4     | integer                |           |          |
9 Indexes:
10    "table_one_pkey" PRIMARY KEY, btree (record_id)
```

7. Insert two records into the table called `table_one`

   (a) Record_id = 1, Att_1 = continent , Att2 = 0olP$fguj , Att_3 = 9.99 , Att_4 = 42

   (b) Record_id = 2, Att_1 = Portsmouth University , Att2 = Violet , Att_3 = 9.99 , Att_4 = 99999

LANGUAGE: SQL

```sql
1 INSERT INTO table_one (Record_id, Att_1, Att_2, Att_3, Att_4) VALUES (1, 'continent'
    ↪ , '0olP[dollarSign]fguj', 9.99, 42);
2 INSERT INTO table_one (Record_id, Att_1, Att_2, Att_3, Att_4) VALUES (2, 'Portsmouth
    ↪  University', 'Violet', 9.99, 9999);
```

8. Get all fo the data from the table

LANGUAGE: SQL

```sql
1 SELECT * FROM table_one;
```

9. Get a screenshot of the data

LANGUAGE: Unknown

```
1  record_id |          att_1          |    att_2     | att_3 | att_4
2 -----------+-------------------------+--------------+-------+-------
3          1 | continent               | 0olP[dollarSign]fguj |  9.99 |    42
4          2 | Portsmouth University   | Violet       |  9.99 |  9999
5 (2 rows)
```

10. Change the value of `Att_4` in record 1 from `44` to `66`

LANGUAGE: SQL

```sql
1  UPDATE table_one SET Att_4 = 66 WHERE record_id = 1;
```

11. Get the data from the table for only record 1

LANGUAGE: SQL

```sql
1  SELECT * FROM table_one WHERE record_id = 1;
```

12. Get a screenshot of the results.

LANGUAGE: Unknown

```
1   record_id |   att_1    |    att_2     | att_3 | att_4
2  -----------+-----------+------------+-------+-------
3           1 | continent | 0olP[dollarSign]fguj  |  9.99 |    66
4  (1 row)
```