Welcome

Course Title: Relational Database Design
Email: service-desk@ucs.cam.ac.uk
Topics covered:

- Entity-Relationship Diagrams
  - Tables and Fields
  - Keys and Joins
  - A practical understanding of normalisation
Lesson Objectives

By the end of this session you will be able to:

✓ Define: database, table, field
✓ Explain join types – 1:1, 1:n, m:n
✓ Apply Primary and Foreign keys to tables
✓ Draw an ERD for a given scenario

Ask questions at any time please!
A database is a **centralised** and **structured set of data** stored on a computer system.

It provides facilities for **adding**, **modifying** and **deleting** the data when required.

It also provides facilities for transforming **queried data** into useful **information**.
RDB Development Overview

1 Understand the requirements:
   What data do you want to store?
   What information do you want to retrieve?

2 Produce an entity relationship diagram

3 Implement in software, e.g:
   Access (Up to 2 GB max size)
   MySQL (2TB max)
   PostGres (unlimited size)
   Oracle (unlimited size)
   etc.
Entity Relationship Design Process

Step 1 – Identify the data
Step 2 – Group the data into tables and allocate a primary key for each table
Step 3 – Join the tables using PK to FK
Tables (entities) hold groups of fields (attributes)

- There is a Primary Key in each table
- Each field is just one piece of data (atomic)
- The tables are joined through the PK / FK
What are Primary and Foreign Keys?

A **Primary Key** is one (or more) fields that uniquely identify a record. Your staff or student ID is unique to you, this is the PK field in your record at the University.

A **Foreign Key** is a field that holds the value of a primary key of another table.

**Common fields** are used to join tables. Joins are either PK to FK (1:n) or PK to PK (1:1).
How Does the ERD relate to the Implementation?
... or in MySQL

```sql
CREATE TABLE customer (  
idCustomer INT (3)  
PRIMARY KEY,  
First_Name varchar(45),  
Last_Name varchar(45),  
Address varchar(45),  
Town varchar(45),  
Country varchar(45),  
PostCode varchar(45),  
TelNo varchar(45),  
Email varchar(45)  
);
```
Going back to Join Types

Relationships are drawn with a single end (1) or a crow’s foot (many):

- **Student**
  - **Loan** (1 to 1)

- **Consultant**
  - **Patient** (1 to many)

- **Patient**
  - **Drug** (many to many)

Each Student has one Loan
Each Loan is allocated to one Student

Each Consultant has many Patients
Each Patient is allocated to one Consultant

Each Patient may be prescribed many Drugs
Each Drug may be prescribed to many Patients
One to one (1:1)

Each Student has one Loan

Each Loan is allocated to one Student

The name of the PK field does not need to match

Each PK is unique in each table and the PK values must match in both tables.
A ‘Case Report’ patient takes each survey once over a period of time
One to Many (1:n)

Consultant

Patient

Each Consultant has many Patients

Each Patient is allocated to one Consultant

Each Consultant appears once in Consultant but many times in Patient
Many to Many (m:n)

Each Patient may be prescribed many Drugs

BAD design: Consider insert/update/delete transactions

Each Drug may be prescribed to many Patients
Many to Many (m:n)

Always de-compose m:n relationships into 1:n relationships because:

It is impossible to enforce referential integrity on m:n joins
They can cause duplicated data (redundancy)
Insert, update and delete operations are very tricky!
They can cause more than one data item in each field – called a repeating group

Patient 1 is prescribed 2 drugs
Aspirin is prescribed to Patients 1, 2, 3, 6
An example of data redundancy:

The **repeating group** has been resolved but at a cost!

Also consider implications for the PK
A simple rule for de-composing Many to Many Joins

Compound Keys are unique PKs using more than one field
A simple rule for de-composing Many to Many joins

A simple ID number is an alternative for the PK of Prescription
Now Patient_ID and Drug_Name are Fks

Task: Use the worksheet to de-compose the m:n joins
Illustrate both methods
Compound Keys work like this . . .

The values of Patient_ID ‘1’ and Drug_Name ‘Ferrous Sulphate’ cannot be entered in another row in this table.

A patient has a row for each drug they are prescribed.
When implemented, the resolved m:n looks like this . . .

Each Patient may be prescribed many drugs

Each Drug may be prescribed to many patients

Each row has a unique compound PK
Normalisation

1st Normal Form:
Each field must hold only one piece of data (relevant to the PK)

2nd Normal Form:
Each non PK field is relevant to the whole PK (when PK is compound)

3rd normal Form:
Check for potential PKs in the non PK fields (avoid dependencies)
## Dependencies

<table>
<thead>
<tr>
<th>STORE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># Store ID</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td>Address Line</td>
</tr>
<tr>
<td></td>
<td>City</td>
</tr>
<tr>
<td></td>
<td>County</td>
</tr>
<tr>
<td></td>
<td>Post code</td>
</tr>
<tr>
<td></td>
<td>Country</td>
</tr>
<tr>
<td></td>
<td>Phone number</td>
</tr>
<tr>
<td></td>
<td>Comments</td>
</tr>
</tbody>
</table>

- If I know the # StoreID then I know the Name, Address Line, City, County, Post Code, Country, Phone no, Comments
  - Name is dependent on Store ID
  - Address line is dependent on Store ID
  - City is dependent on Store ID
  - and so on . . .

### Remember:

1NF: Each non key field must be one data item related (i.e. ‘dependent’) on the PK
1st Normal Form:
Each field must hold only one piece of data (relevant to the PK)

Table design for the business rule ‘Stores are our customers’

<table>
<thead>
<tr>
<th>STORE</th>
<th>STORE</th>
</tr>
</thead>
<tbody>
<tr>
<td># Store ID</td>
<td># Store ID</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Address</td>
<td>Address Line</td>
</tr>
<tr>
<td>Phone number</td>
<td>City</td>
</tr>
<tr>
<td>Comments</td>
<td>County</td>
</tr>
<tr>
<td></td>
<td>Post code</td>
</tr>
<tr>
<td></td>
<td>Country</td>
</tr>
<tr>
<td></td>
<td>Phone number</td>
</tr>
<tr>
<td></td>
<td>Comments</td>
</tr>
</tbody>
</table>

Not good
‘Address’ holds more than one piece of data

Good
Another example of the violation of 1NF

<table>
<thead>
<tr>
<th>Drug_Name</th>
<th>Indications</th>
<th>Contraindications</th>
<th>Price per unit</th>
<th>Side Effects</th>
<th>Patient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aspirin</td>
<td>Fever</td>
<td>Allergic to Ibuprofen</td>
<td>£0.01</td>
<td>Bleeding</td>
<td>1, 2, 3, 6</td>
</tr>
<tr>
<td>Ferrous Sulphate</td>
<td>Anaemia</td>
<td>Hypersensitivity</td>
<td>£0.02</td>
<td>Nausea</td>
<td>1, 3, 4, 5</td>
</tr>
</tbody>
</table>

This is called a ‘repeating group’
The cell is storing more than one fact

1st Normal Form:
Each field must hold only **one piece of data** (relevant to the PK)
2nd Normal Form:
Each non PK field is relevant to the whole PK (when PK is compound)

Only consider 2NF when you have a compound key

Enrolment(# Student_ID, # Course_ID, DateOfEnrolment, StudentName, CourseName)

Which non key fields are NOT fully dependent on the compound key?
Third Normal Form

3NF: Check for potential PKs in the non PK fields (avoid dependencies)

Suppose we wanted to track Staff who work in Departments . . .

Can you see any problems?
Violation of 3NF

The PK is identifying each row uniquely (good) but is not identifying each Department uniquely (bad).

Data redundancy is occurring (bad)

Each staff entry is forcing the data onto many rows.

Tables that are not in 3NF cause insert, update and delete anomalies:

**Insert:** What if a new department is created but no staff are allocated?

**Update:** What if the accounts department changed its Department_ID?

**Delete:** What if a department was deleted?
To resolve into correct 3NF, split the table
Leave a copy of the Department_ID in Staff to make the join

This may look like redundancy but it is not. This is a FK field and makes the link between Department and Staff tables.
A worked example
Developing an ERD from a business scenario

Step 1 – Identify the data
Step 2 – Group the data into tables and allocate a primary key for each table
Step 3 – Join the tables using PK to FK
From this narrative description identify the data to be stored:

“I’m a manager of the Sporting-Goods Wholesale Company that operates worldwide to fill orders from retail sporting-goods stores. The stores are our customers. For each customer, we must track an ID and a name. We may track an address including the city, county, post code, country and phone number.

We need to record information about our stock including ID, description, price, amount in stock. We hold stock in warehouses to best fill the orders of our customers. For each order, we must track an ID. We track the date ordered, date shipped, and payment type when the information is available.

Each warehouse must have an ID for which we track an address including the city, county, post code, country and phone number.

Departments are responsible for placing and tracking the orders when our customers call. For each department, we must track the ID and name. We may also record general comments about a customer.”
“I’m a manager of the **Sporting-Goods Wholesale Company** that operates worldwide to fill orders from retail sporting-goods stores. The **stores** are our customers. For each customer, we must track an **ID** and a **name**. We may track an **address** including the **city**, **county**, **post code**, **country** and **phone number**.

We need to record information about our **stock** including **ID**, **description**, **price**, **amount in stock**. We hold stock in **warehouses** to best fill the **orders** of our customers. For each order, we must track an **ID**. We track the **date ordered**, **date shipped**, and **payment type** when the information is available.

Each warehouse must have an **ID** for which we track an **address** including the **city**, **county**, **post code**, **country** and **phone number**.

**Departments** are responsible for placing and tracking the orders when our customers call. For each department, we must track the **ID** and **name**. We may also record general **comments** about a customer.”
Step 2a – Group the fields into tables
ensure data is split into single data items

<table>
<thead>
<tr>
<th>STORE</th>
<th>ORDER</th>
<th>STOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store ID</td>
<td>Order ID</td>
<td>Stock ID</td>
</tr>
<tr>
<td>Name</td>
<td>Date ordered</td>
<td>Description</td>
</tr>
<tr>
<td>Address Line</td>
<td>Date shipped</td>
<td>Price</td>
</tr>
<tr>
<td>City</td>
<td>Payment type</td>
<td>Amount in stock</td>
</tr>
<tr>
<td>County</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Post code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comments</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| WAREHOUSE      |                        |                           |
| Warehouse ID   | Address Line           |                           |
| City           | Date ordered           |                           |
| County         | Date shipped           |                           |
| Post code      | Payment type           |                           |
| Country        |                        |                           |
| Phone number   |                        |                           |

| DEPARTMENT     |                        |                           |
| Department ID  | Address Line           |                           |
| Name           | Date ordered           |                           |
|                | Date shipped           |                           |
|                | Payment type           |                           |

Each field is listed **once** (unless involved in a PK to FK join) **and** is in the correct table
Step 2b – allocate a PK for each table

<table>
<thead>
<tr>
<th>STORE</th>
<th>ORDER</th>
<th>STOCK</th>
<th>WAREHOUSE</th>
<th>DEPARTMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td># Store ID</td>
<td># Order ID</td>
<td># Stock ID</td>
<td># Warehouse ID</td>
<td># Department ID</td>
</tr>
<tr>
<td>Name</td>
<td>Date ordered</td>
<td>Description</td>
<td>Address</td>
<td>Name</td>
</tr>
<tr>
<td>Address</td>
<td>Date shipped</td>
<td>Price</td>
<td>City</td>
<td></td>
</tr>
<tr>
<td>City</td>
<td>Payment type</td>
<td>Amount in stock</td>
<td>County</td>
<td></td>
</tr>
<tr>
<td>County</td>
<td></td>
<td></td>
<td>Post code</td>
<td></td>
</tr>
<tr>
<td>Post code</td>
<td></td>
<td></td>
<td>Country</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td></td>
<td></td>
<td>Phone number</td>
<td></td>
</tr>
<tr>
<td>Phone number</td>
<td></td>
<td></td>
<td>Comments</td>
<td></td>
</tr>
</tbody>
</table>
“I’m a manager of the **Sporting-Goods Wholesale Company** that operates worldwide to fill orders from retail sporting-goods stores. The stores are our customers. For each customer, we must track an ID and a name. We may track an address including the city, county, post code, country and phone number.

We need to record information about our stock including ID, description, price, amount in stock. **We hold stock in warehouses** to best fill the **orders of our customers**. For each order, we must track an ID. We track the date ordered, date shipped, and payment type when the information is available. (**Stock is in an order**) 

Each warehouse must have an ID for which we track an address including the city, county, post code, country and phone number.

**Departments are responsible for** placing and tracking the **orders** when our customers call. For each department, we must track the ID and name. We may also record general comments about a customer.”
Step 3 – Join the tables using PK to FK

Each field is stored once (unless it is part of a join) and is in the correct table.
The ERD is not quite there, but OK for a 1\textsuperscript{st} draft

This table needs further refinement, can you see why?

**Task**: Complete the occurrence table with dummy data. Add three items into one order. Can you identify the problem?
Because the customer orders more than one stock item (repeating group) this is causing data redundancy.

Each stock item ordered requires a new row and a unique PK, the table is not in 1NF
Remove the Stock_ID field from the ORDER table
Place it into its own table
Take a copy of the ORDER_ID to make the join - now a 1:n between ORDERS and ORDER_DETAILS
Adding ‘Quantity Ordered’ is a good idea too!
Better but still not completed, ‘Payment type’ is likely to need further refinement
A horticultural research company is conducting a study of different plants grown using a new fertilizer. You are responsible for designing the database.

You need to store data on a variety of plants including scientific name, common name and family.

The environmental conditions of each greenhouse should be recorded including temperature and luminosity.

The soil substrate (type and PH) and the dilution of the new fertilizer is recorded for each tray.

A conclusion for each tray result should be recorded.

Daily observations of plant growth for each plant group must be stored with an optional comment.

Identify the data, and then model the entities in an ERD.
A possible solution

Each greenhouse has many trays with different soil conditions
Each plant may be grown in many different trays (with different conditions)
Each tray has many observations
Each conclusion is about an individual plant grown in a specific tray
Ear, Nose and Throat Research Study

Prior to **Flexible Laryngoscopy** topical nasal preparation is used to reduce patient discomfort and ease passage of the scope. The ideal nasal preparation should be comfortable for the patient, produce adequate anaesthesia and widen nasal patency.

**Null hypothesis:** There is no difference between **co-phenylcaine spray** or **lidocaine /epinephrine nasal packing** for preparing the nose prior to flexible laryngoscopy in terms of (1) patient comfort; and (2) degree of decongestion and ease of endoscope passage.

Patients rate experience on scale from 0 to 100 (zero = no pain) for each of these:
- Nasal Preparation (**NP**) bad taste, pain, anxiety and overall unpleasantness
- Flexible Laryngoscopy (**FL**) pain, anxiety, gagging and overall unpleasantness
- Patients have **one treatment** only

The surgeon used a visual analogue scale (VAS) from 0 -100 to record:
- Degree of decongestion
- Ease of endoscope passage

**Other details recorded:**
- Cottle’s grading of septal deviation of side scoped
- Time given for preparation to take effect.
This is easier than it first appears!

Each patient has one treatment using one anaesthetic and one set of observations.

A ‘flat file’ one table database could be used. A spread sheet package is an alternative application to use.
**MyPlace Rental Properties**

**Cressbrook Drive, Gt Cambourne**
A modern two bedroom house located in the Cambourne village development located to the West of Cambridge.

£650 To Let

**Grenadier Walk Hardwick, Cambridge**
Modern first floor flat in a popular village with local shop and pub and easy access to Cambridge A14 and M11.

£525 To Let

**Stonefield Bar Hill, Cambridge**
A modern home situated within the village of Bar Hill close to local shops and facilities providing good access to the A14.

£580 To Let

**Chesterton Road, Cambridge**
One single furnished room available in this five bedroom Victorian terraced house with easy access to the City Centre.

£290
MyPlace Estate Agents Case Study
A possible solution