Data Modelling and E-R Diagrams

So far we have considered some of the basic ideas behind relational theory, and we will continue with this in subsequent sections. In this section we look at the processes involved in capturing the information required to design and create a database.

Data Analysis and Database Design

We can subdivide the process of designing a database into three separate phases: Data Analysis, System Design, and Technical Design.

User Environment: The ‘real world’ environment about which the organisation wishes to store data for the purposes of some application (the application might be, for example, a computerised stock control and accounting system).

Data Analysis: In this phase the ‘real world’ environment, in the context of the required application, is modelled to produce (i) an abstract data model which focuses on the data that needs to be stored and the relationships between the data entities and (ii) the functional model which deals with how the data will be processed (e.g. the design of queries to the database). The abstract data and functional models produced are independent of the specific DBMS that will eventually be used to implement the database. Note that this phase is also sometimes called conceptual design.
**System Design:** The data and functional models feed into the system design phase which produces the *database specification*. The database specification details how the abstract data model from the previous design phase will map onto the specific data model of the chosen DBMS (this activity is sometimes called *logical design*). The database specification also describes information such as statistics on how frequently data will be accessed and access patterns across the datasets etc. The aim of these specifications is to ensure that the implementation will be efficient and that sufficient capacity (storage, memory, processing) is provisioned.

**Technical Design:** Taking the database specification as input, the technical design phase specifies exactly how the database will be physically implemented using a DMBS, so that a certain set of criteria are met. This phase is sometimes called *physical design*. Examples of the criteria are:

- Data availability: ensure that the data and relationships required by the application are stored in the database
- Data reliability: the data will not be lost or corrupted
- Data currency: the data value in the database is the latest available
- Data consistency: the same data values will be obtained from different queries that are executed at the same time
- Data efficiency: the data are stored and retrieved at minimum cost (in terms of storage, memory or processing power required)

**Database:** Finally the database is implemented according to the technical design requirements using a chosen DBMS.

In this course, we are most interested in the high level design process of Conceptual Design (Data Analysis) and the relational theories on which it builds.
Data Models

As mentioned above, the data model is an abstraction of the data structures required by a database. By ‘abstraction’, we mean that data model represents the data as a user would observe it in the ‘real world’ rather than how it is stored on disk or how software manipulates it. This means that the data model is independent of the particular DBMS that will eventually be used to implement the database, a useful property.

The data model is composed of data entities, relationships between different data entities and also rules which govern operations on the data. Note that the data model does not specify the operations themselves that will be performed on the data (these are specified in the functional model), but rather just the restrictions that apply when any operation is performed on the data structures.

Entity-Relationship (E-R) Models

An E-R model is a particular type of data model suited to designing relational databases. The main component of the model is the Entity-Relationship Diagram. The E-R diagram is a simple way of representing the data entities being modelled and the relationships between these data entities. It is easy to transform E-R diagrams to the Relational Model (data entities correspond to relations and relationships correspond to the implied associations created by keys and foreign keys of relations).

Entities

Entities are analogous to relations in the relational model. They represent the principle data objects about which information is to be collected. Entities represent concepts or concrete or abstract objects this such as person, place, physical things, events, for example STUDENT, PROJECT, INVOICE, PURCHASE. In an E-R diagram, an entity is represented as a named rectangular shape, which may include a list of attributes. For clarity, normally only attributes that are involved in relationships between entities are included, i.e. primary key (PK) and foreign keys (FK). This maintains an uncluttered diagram. Recall our student relation from earlier:

STUDENT (ID, Surname, Forename, Programme, Date_of_Birth)

We can denote our STUDENT entity in an E-R diagram as shown below:
**Relationships**

A relationship represents some association between two or more entities. In an E-R diagram, a relationship is represented as a diamond shape, containing the name of the relationship between the entities. Returning to our final year project database relations from earlier,

\[
\begin{align*}
\text{STUDENT} & \quad (\text{ID, Surname, Forename, Programme, Date\_of\_Birth}) \\
\text{PROJECT} & \quad (\text{Proj-ID, Project\_Title, Student-ID, Year})
\end{align*}
\]

we can represent the relationship between these relations as follows:

![Diagram showing one-to-one relationship]

Note that we can show the **connectivity** of the relationship, which in this case is a **one-to-one (1:1)** relationship. The one-to-one relationship shown reflects the fact that each student is assigned (at most) one project and that each project is assigned to (at most) one student.

**One-to-many (1:N) Relationships**

Introducing a new relation SUPERVISOR to our schema, an example of an E-R diagram of a one-to-many relationship is shown below. This relationship models the fact that every final year project supervisor supervises a number of projects but that each project has only one supervisor. (Note that we have added the Supervisor_ID attribute to the relation PROJECT to form a foreign key, reflecting the primary key of SUPERVISOR).

![Diagram showing one-to-many relationship]
Many-to-many (N:M) Relationships

Introducing another relation PROGRAMME to our schema, an example of a many-to-many relationship is shown below. This relationship models the fact that a project could suit students on different programmes (e.g., a project involving database systems would suit either DME or ICE) and also that a given programme will have multiple projects that suit it. (Note that we have added the foreign key Program_ID to relation PROJECT to form the relationship).

Resolving many-to-many relationships

Many-to-many relationships in E-R diagrams (as above) cannot be expressed directly in the relational model and must be resolved at the modelling stage. To illustrate the problem, as an exercise complete a table showing an instance of PROJECT which shows project number 4 in 2006 being suitable for both the ICE and DME programmes:

**RELATION: PROJECT**

<table>
<thead>
<tr>
<th>Proj_ID</th>
<th>Project Title</th>
<th>Student_ID</th>
<th>Year</th>
<th>Supervisor_ID</th>
<th>Program_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To resolve this conflict, a many-to-many relationship must be replaced with an association entity which relates the entities by two one-to-many relationships:
The general pattern to forming the association entity is to migrate the primary keys of the two entities to the association entity. The combination of these two foreign keys becomes the primary key of the association entity. This new structure allows us to express a many-to-many relationship between the original two entities whilst obeying the rules of relations. Try the above exercise again on the new schema: show project number 4 in 2006 being suitable for both the ICE and DME programmes.

The complete schema for the final year project database (neglecting attributes other than PKs and FKS) is shown below.
We have shown (some) of the notation of E-R diagrams and how data objects and the relationships between them may be captured using data modelling. As mentioned previously, the other aspect of conceptual design is functional modelling, which entails specify the operations that will be performed on the database. In the next section we look at an abstract language (independent of the particular DBMS that will be used) that allows us to specify these operations.

**Exercise 3**

Consider the group of four relations listed below:

- **STUDENT** (Student_ID, First_Name, Last_Name)
- **REGISTER** (Student_ID, Module_ID, Semester-Start-Date)
- **LECTURER** (Lecturer_ID, First_Name, Last_Name)
- **MODULE** (Module_ID, Module_Name, Lecturer_ID)

The following additional information has been determined by the database designer:

- The primary key of STUDENT is \{Student_ID\}
- A student may register for a module more than once (e.g. to repeat it), but only once in any given semester. That is, the primary key of REGISTER is \{Student_ID, Module_ID, Semester-Start-Date\}
- The primary key of LECTURER is \{Lecturer_ID\}
- The primary key of MODULE is \{Module_ID\}

Draw an E-R diagram showing the entities and relationships in this database schema. Show the primary and foreign keys and the connectivity (i.e. one-to-one, many-to-one, etc.) between entities.