Databases - Entity-Relationship Modelling
Database Design Process

Ramakrishnan & Gehrke identify six main steps in designing a database

- Requirements Analysis
- Conceptual Design
- Logical Design
- Schema Refinement
- Physical Design
- Application & Security Design

(GF Royle, N Spadaccini 2006-2010)
Database Design

Requirements Analysis

Conceptual Design

Logical Design

Schema Refinement

Physical Design

Application/Security Design

ER-Modelling
Requirements Analysis

Requirements Analysis is the process of determining what the database is to be used for.

It involves interviews with user groups and other stakeholders to identify what functionality they require from the database, what kinds of data they wish to process and the most frequently performed operations.

This discussion is at a non-technical level and enables the database designers to understand the business logic behind the desired database.
Using the ER data model, the conceptual design stage involves identifying the relevant \textit{entities} and the \textit{relationships} between them and producing an \textit{entity-relationship} diagram.

The logical design stage involves translating the ER diagram into actual relational database schema.

Once a suitable ER model has been constructed, then this can be translated into a relational database fairly easily.

However ER modelling is as much an art as a science, as there are usually many choices to be made and the consequences of each choice sometimes does not become apparent until problems arise later.
After the modelling

We will cover the remaining steps of the process later as they involve the effective *use* of the database after it’s fundamental structure has been determined. This involves

- Mathematical analysis and refinement of the schema
- Performance-based decisions on indexing, machine capacities, required performance etc.
- Interfacing with client applications and security
Iterative Process

As with all software engineering processes, information uncovered during later phases of the project may alter some design decisions so the nice neat diagram of the 6 phases is really an iterative process where each stage feeds back to the previous stages.

One of the major causes of design alterations is incomplete requirements analysis — this is frequently attributed to users not being aware of the possibilities until they start using the system, or at least a prototype.
Running Example

We use a modified version of Exercise 2.3 in the text as an example. This question asks the user to produce an ER model from the following requirements:

- A lecturer has a staff number, a name and a rank
- Research projects have a project id, a sponsoring organization and a budget
- Each project has one lecturer as a principal investigator
- Each project may have other lecturers as co-investigators
- Each lecturer can be principal or co-investigator on multiple projects
Entity Sets

We have already introduced the concept of an *entity-set* and explained how that can be diagrammed and then translated directly into a relational database.

One detail that we omitted is that a relation should be a *set* and therefore cannot contain any duplicates elements — the corresponding database table should not have any duplicate rows.

A *key* for an entity set is an attributed, or combination of attributes that is guaranteed to distinguish the elements.
Example key

A Student is always uniquely identified by their student-id so this attribute is a suitable key for this relation.

In an ER diagram, an entity set’s key is designated by underlining the attribute or attributes that form the key.
Why are keys important?

It is important to identify the key for an entity set and indeed — as we will see later — for any relation in a database for several reasons:

- Explicitly identify a key ensures that the data model is logically consistent.
- When implemented, the DBMS can enforce key constraints that ensure that the nominated key does indeed uniquely identify each row.
- The DBMS can index the table using the key values and manipulate that relation very efficiently.
- The DBMS can enforce referential integrity by ensuring that tables that refer to other tables remain in a consistent state — this comes later!
Example – Lecturer Entity

Modelling of the *entities* in the example is fairly easy:

- staff-id
- name
- rank

So the actual entities would be things like

<table>
<thead>
<tr>
<th>staff-id</th>
<th>name</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>00144238</td>
<td>&quot;J Smith&quot;</td>
<td>&quot;Associate Professor&quot;</td>
</tr>
</tbody>
</table>

(GF Royle, N Spadaccini 2006-2010)
Example – Project Entity

So the actual entities would be things like

<table>
<thead>
<tr>
<th>proj-id</th>
<th>sponsor</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>DH10304</td>
<td>ARC</td>
<td>$120000</td>
</tr>
</tbody>
</table>

(GF Royle, N Spadaccini 2006-2010)
Hmmm, what about the key

Although it is unlikely that two sponsoring organizations use the same format for project id numbers, it is possible. So maybe the key should consist of the two attributes \( \text{proj-id, sponsor} \)?

In this case, the attributes that form the key are all underlined.
Relationship Sets

The power of relational databases comes from the ability to model and query *relationships between entity sets*.  

If $E_1, E_2, \ldots, E_n$ are entity sets, then a *relationship set* is a subset

$$R \subseteq E_1 \times E_2 \times \cdots \times E_n$$

In other words $R$ is an $n$-ary relation whose elements are *entities*.  

As entity sets are *also* relations, this means that we are using relations/tables to model both entity sets and relationship sets!
Diagramming Relationships

A *relationship* is diagrammed by a named *diamond* shape that is connected by lines to the related entity sets.

Here we are modelling the relationship `Principal` which relates projects to their principal investigators.
Elements of the relationship set

The actual elements of the relationship set are the pairs that specify which lecturer manages which project.

Thus if Associate Professor Smith manages the project DH10304 then that pair of entities would be an element of the Principal relationship set.

Notice that we can unambiguously specify this pair just by storing the keys for this pair:

\[(00144238, DH10304) \in \text{Principal}\]

This of course is exactly how an RDBMS stores a relationship set.
A sample relationship set

Each black circle represents one element of the Principal relation.
Participation Constraints

Lecturer
Partial participation

Principal
One-to-Many

Project
Total participation

(GF Royle, N Spadaccini 2006-2010)
Participation Constraints

If the participation is a *constraint* based on the business logic (rather than just an accident of this particular set of data), then it can be encoded into the ER diagram (and subsequently enforced by the DBMS).

*Every* project must have a principal investigator and so there is *total participation* of the entity set *Projects* in the relationship set *Principal*.

This is indicated in the ER diagram by a thick black line connecting the entity set with the relationship set.
ER diagram with participation constraints
Key constraints

The relationship **Principal** is one-to-many because each project has just *one* principal investigator.

This is called a *key constraint* because it means that in any allowable instance of **Principal** each entity from **Project** appears at most once.

In particular, this means that the key for **Project** can be used as a key for **Principal**.

This is indicated in the ER diagram by an arrow-head on the line connecting the entity set with the relationship set.
ER diagram with participation and key constraints
Other conventions

There are lots of other conventions for ER diagrams that include other symbols, or possibly little numbers indicating the exact form of the relationship etc., thus making it more detailed and more expressive.

We deliberately use this very simple form of ER diagramming because the constraints that are used in this model can all be implemented in standard SQL, and thus the database model corresponds precisely to the ER diagram.