

# Paper

Deriving Entity-Relationship and Relational Model Diagrams

# Contents

<b>THE E-R MODEL AND RELATIONAL DATABASE DESIGN .....</b>	<b>3</b>
DERIVING E-R MODEL.....	3
Deriving an E-R Diagram .....	4
IMPLEMENTING THE E-R MODEL – THE RELATIONAL MODEL.....	6
Transforming the E-R Model into a Relational Database Design .....	7
EVALUATION OF THIS TECHNIQUE .....	8
REFERENCES .....	10

# The E-R Model and Relational Database Design

Data modelling is the process of creating a logical representation of the structure of a database. (Kroenke D, 1999, p. 47) It is the most important task performed by the developer – the entire database and all applications that access this database are based on it, and if the data is modelled incorrectly, poor and difficult to user applications will result.

A data model is a model of the user's model of their business. It must identify the things to be stored in the database and define their structure and the relationships among them. (Kroenke D, 1999, p. 41) Eventually the data model leads to the development and implementation of a database. Whether or not this database will meet, and possibly exceed, the user's requirements depends on how closely the data model resembles the user's model of their business.

Database models fall into two main categories: conceptual models and implementation models. Conceptual models are concerned with the logical nature of the data and what is being represented, whereas implementation models are concerned with the physical nature of the database and with how the data will be represented.

Two very important models used in database design are the Entity-Relationship (E-R) Model and the Relational Model. In the E-R model, data is represented using entities, and relationships are defined between these entities. However, with the relational model, the entities and their relationships follow strict guidelines. Usually, an E-R model is first developed, and then it is transformed into a relational model.

## Deriving E-R Model

The E-R model was derived by Peter Chen in 1976, and in it “those things that users want to track are represented by entities, and the relationships among the entities are represented by explicitly defined relationships.” (Kroenke D, 1999, p. 144) Key elements of this model are entities, attributes, identifiers and relationships.

- Entities** – An entity is a tangible object of interest that exists in the user's domain. It is usually something of interest to the user and they (users) keep track of it. Collective nouns, or nouns, are usually used to name (describe) entities: Employee, Position, Book, Member.
- Attributes** – An attribute is one of the various properties that describe the entity's characteristics. These properties usually present a single fact – they are atomic. (University of Texas at Austin, 2005). For example, the Employee entity could have the following attributes: LastName, FirstName, SSNumber.
- Identifiers** – An identifier is an attribute that identifies entity instances. They can be unique, non-unique, or composite. Unique identifiers identify one and only one entity instance, whereas non-unique identifiers can identify more than one entity instance. Composite identifiers consist of more than one attribute.
- Relationships** – This is what exists between entities – entities in a database model are related to each other and this constitutes relationships. Verbs are usually used to describe relationships: Students take Courses – Students and Courses are entities, and take is the relationship. Relationships are one-to-one, one-to-many or many-to-many.

## Deriving an E-R Diagram

An E-R diagram is used to represent the E-R model. It contains all known entities for the given scenario, their attributes, identifiers, and the relationships that exist among the entities. Deriving the entities, their attributes, identifiers, and the relationships among them is one of the most important activities carried out during database development, so it is imperative that it is done accurately.

From developing a number of databases, I have come up with a technique that allows for rapid database design and development. This technique was initially taught to me by consultants from the CRC SOGEMA INC. group in 1998, and a similar technique is described in the IBM Informix Database Design and Implementation Guide (2005).

### Steps

1. Discover the entities and the basic attributes for each entity. This process usually involves performing a careful analysis of the processes and documentation used by the users in the problem domain. Make a list of each of the entities discovered along with the basic attributes of each entity.
2. Identify the relationships among the entities. Using an Entity-Entity Matrix (E-E Matrix) is one of the fastest methods for deriving all the relationships among all the entities. This matrix consists of an equal number of rows and columns, with each entity discovered heading a row and a column. The intersection of the rows and columns represents relationships that may exist between the entities – it is possible that no relationship will exist between entities, that more than one relationship will exist between entities, and that an entity will be related to itself.
3. Use the information from the E-E matrix to construct a simple initial E-R diagram. This diagram will contain the entities and their basic attributes (and possible identifiers), and the relationships indicated in the E-E matrix.
4. Derive optionalities and cardinalities for each relationship in the initial E-R diagram and write assertions for each relationship. This records the assertions that can be made about the data (relationships) in the user's domain. Optionality says what can and must happen in a relationship, and cardinality indicates the number of entity occurrences in a relationship:
  - Optionality "0" – Can (optional)
  - Optionality "1" – Must (obligatory)
  - Cardinality "1" – Only one
  - Cardinality "N" – Many/At least one
5. Construct a detailed E-R diagram that includes all the derived information. This diagram will include optionalities, cardinalities, attributes, identifiers, and relationships. Unique identifiers are indicated with an asterisk (\*), and non-unique identifiers with a plus (+).

Following these steps will produce a detailed E-R diagram. This diagram is a conceptual model and represents the logical nature of the data that exists in the user's domain. This model will be later used to create a relational model, which can then be implemented on any relational database management system (RDBMS).

### Example

Consider as an example deriving a simple model for students taking courses at a post-secondary college, and the grades they get for assessments in these courses.

### Step 1

Student {StudentId, LastName, FirstName, Sex, Email, HTel, WTel}

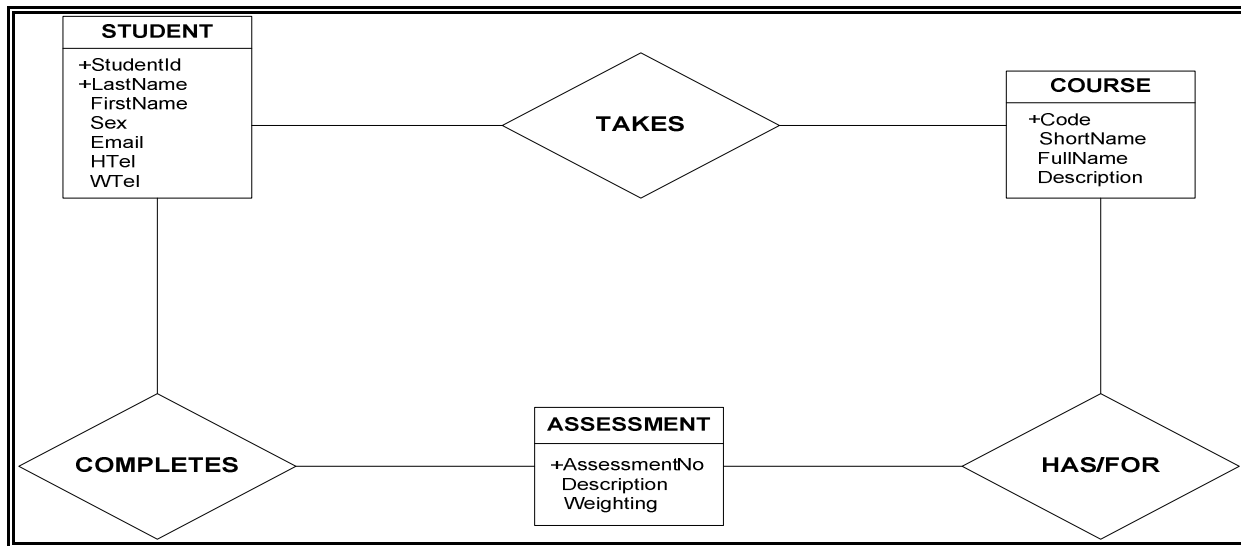
Course {Code, ShortName, FullName, Description}

Assessment {AssessmentNo, Description, Weighting}

### Step 2

	STUDENT	COURSE	ASSESSMENT
STUDENT			
COURSE	takes		
ASSESSMENT	completes	has/for	

### Step 3



### Step 4

A Student can take many Courses

A Course can be taken by many Students

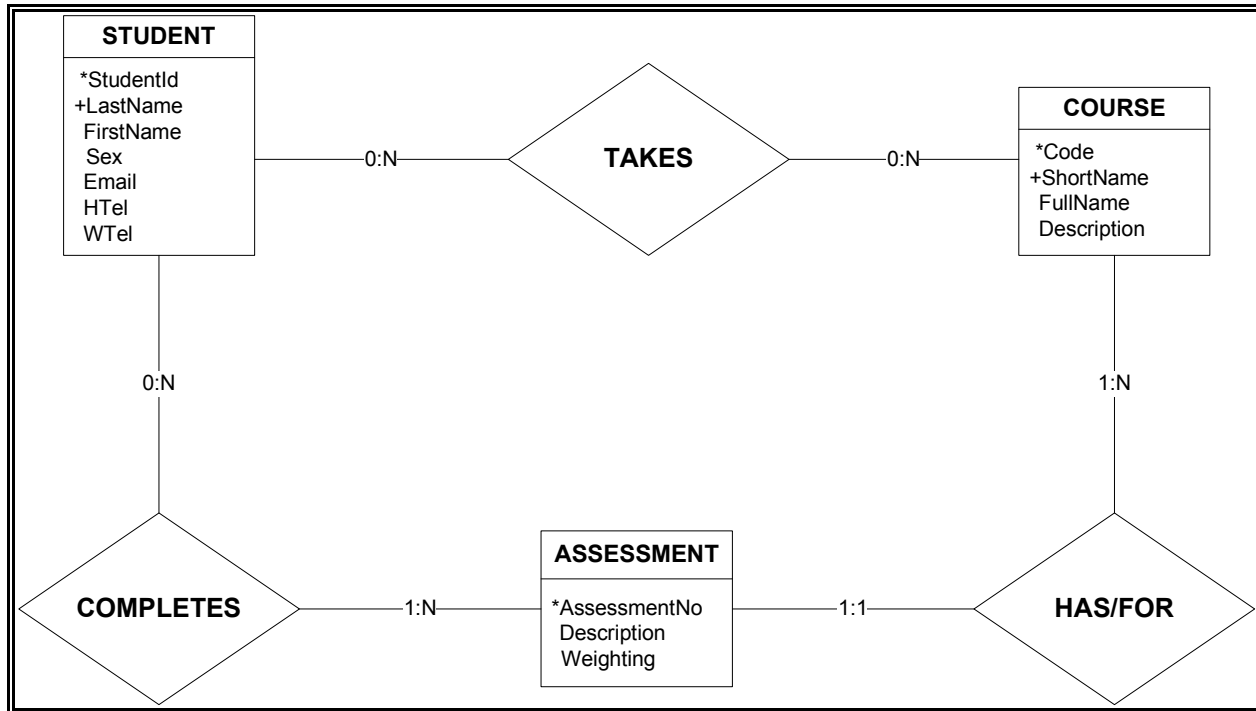
A Student can complete many Assessments

An Assessment must be completed by at least one Student

A Course must have at least one Assessment

An Assessment is for only one Course

## Step 5



## Implementing the E-R Model – The Relational Model

While an E-R diagram displays the logical nature of the data that exists in the user's domain, the relational model shows how this data will be represented in a Relational Database Management System (RDBMS). This model is important because the most common and popular database management systems in use today are relational, and relational model diagrams can be directly implemented on any RDBMS.

The relational model is similar to the E-R model but based on a branch of mathematics called relational algebra, and as such, there are strict definitions and rules regarding the elements of this model. Here are some of these definitions and rules:

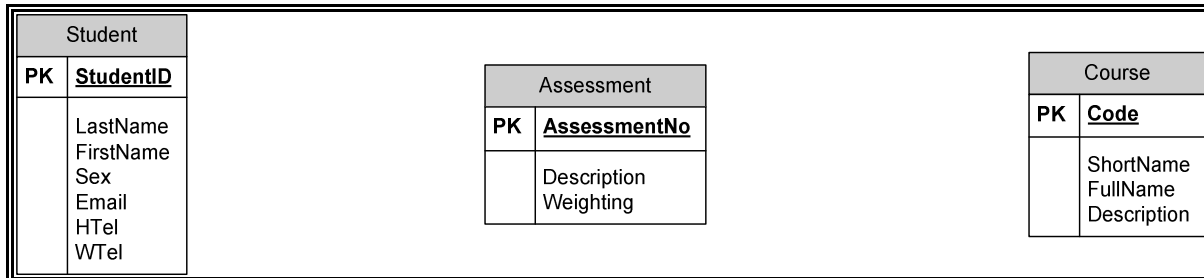
- A relation is defined as a two dimensional table that contains rows and columns. Rows are instances of the entity and columns are attributes of the entity.
- No two rows of the table must be identical – each row must be unique (the contents of the row that is), and the order of the rows is not important.
- Each column must have a unique name, and the order of the columns is not important. They must also contain single values – multiple values and repeating groups are not allowed.
- A key is a group of one or more attributes that uniquely identifies a row – each table must have a key. Since each row in the table is unique, the key is what is used to distinguish one row from another. It can be comprised of a single attribute or it can be composite, comprising of more than one attribute. (Kroenke D, 1999, p. 113-117)
- Every column (attribute) in a table must depend solely on the primary key. It should not depend on any other column in the table, or on part of the primary key if it is a composite key.

## Transforming the E-R Model into a Relational Database Design

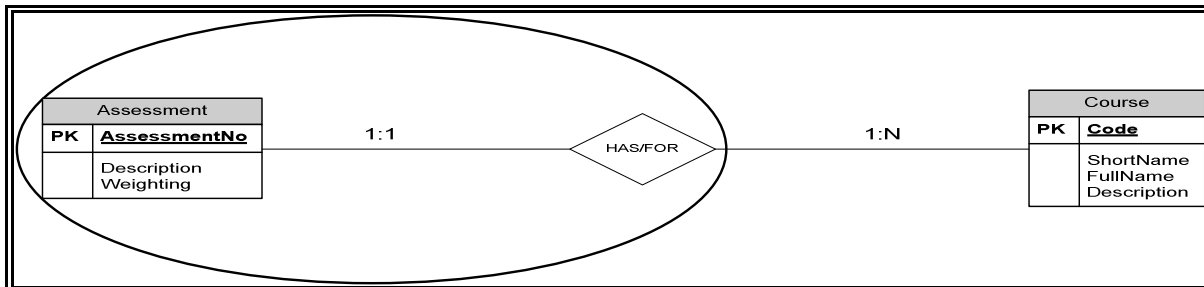
The technique that I use to transform the E-R model into a relational model was also taught to me by a consultant from the CRC SOGEMA INC. group in 1998. It is a very simple and straightforward technique that uses the existing E-R diagram to develop a Crow's Foot Relational Model diagram.

### Steps

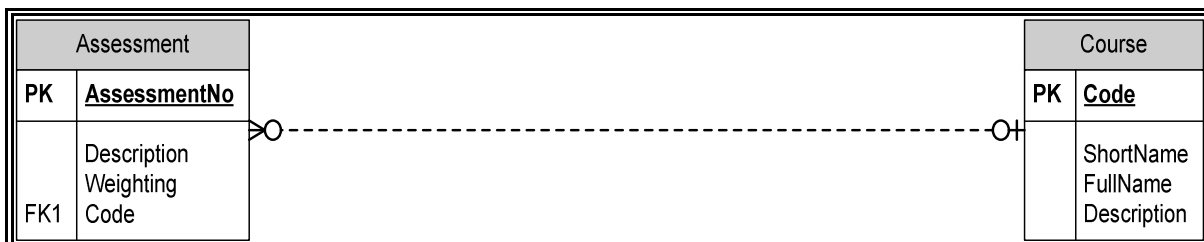
1. Clearly identify the primary key and attributes for each entity defined in the E-R model, and ensure that it is in accordance with the rules of the relational model as previously discussed.



- Each entity, with its clearly identified primary key (indicated by PK), and attributes satisfying the previously discussed rules, becomes a table in the relational model.
2. Group together tables (formerly entities) and their relationships that have a cardinality of one – relationships with 0:1 or 1:1 for their opt:card. That is, absorb relationships where the cardinality is one into the corresponding tables.
    - Maintain the initial structure for the absorbing table – do not change its primary key or any of its attributes.

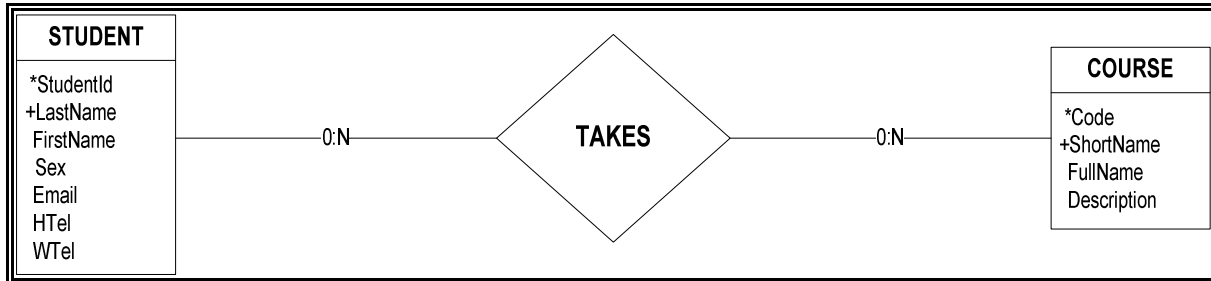


- The primary key of the other table in the relationship becomes a foreign key in the absorbing table. This is indicated by an FK in the absorbing table.
- If the cardinality between the absorbed relationship and the other table in the relationship was N (0:N or 1:N), then the modified table becomes the many part of a one-to-many relationship in the new relational model.

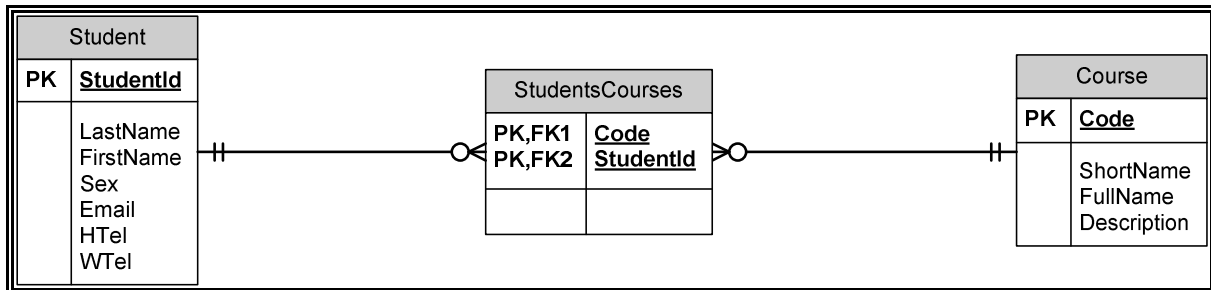


- Otherwise, if the cardinality between the absorbed relationship and the other table in the relationship was 1 (0:1 or 1:1), then the new relationship becomes a one-to-one relationship in the new relational model (not shown).

3. The remaining relationships whose cardinalities are N (1:N or 0:N) on both sides become new tables in the new relational model.



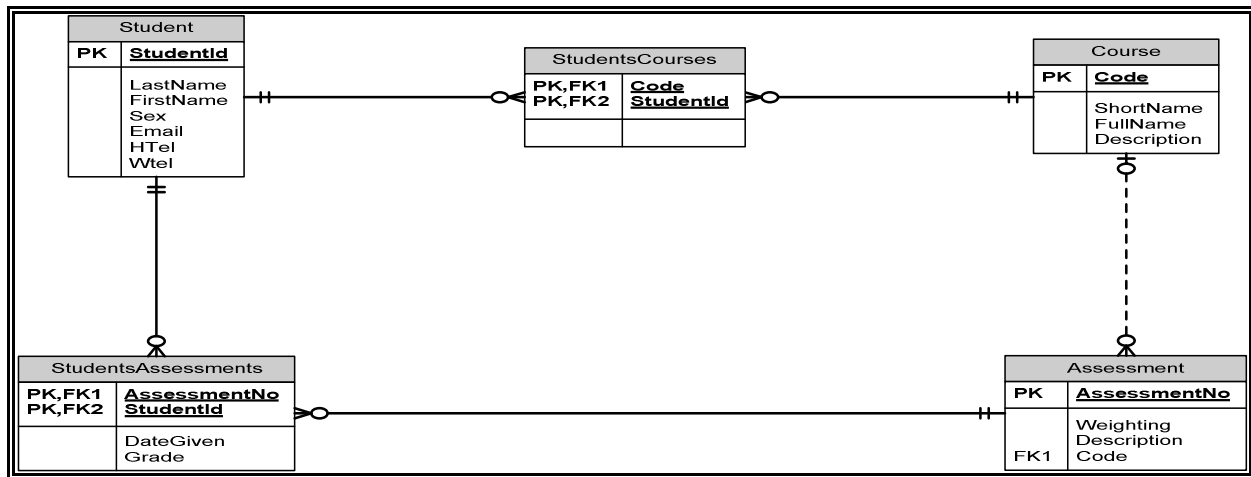
- The primary keys from the two tables involved in the relationship become a composite primary key in the new table, and the new table usually has a name that is a combined form of the two original table names.



- The newly created table becomes the many part of the relationship between both tables, and thus creates a many-to-many relationship between the two pre-existing tables. In some instances, the newly created may have its own attributes, but this is rare.

### Example

As an example, the E-R diagram from the example above is transformed using the technique described above. The resulting diagram is given below.



### Evaluation of this Technique

The technique outlined above give the most basic steps necessary to quickly create a complete E-R and relational model diagram for any system being modelled. The relational model can then be implemented on any RDBMS. Although scenarios that are more complicated than the example given will be encountered, this method is ideal for small systems and for quickly creating prototypes. I have not had the opportunity to test this method with more than about 15 to 20 entities, which results in

about 30 tables in the relational model, however, I am confident that with careful attention to detail this technique will work.

It is easy to obtain all the relationships that exist between all the entities using an E-E matrix using the technique outlined above. It allows you to obtain more than one relationship for the same pair of entities, and allows you to obtain recursive relationships. However, all of the relationships obtained are binary relationships, which might not model the true nature of the system's data. In some complex, or even simple, systems, ternary or higher relationships might exist, and using an E-E matrix will not capture these relationships.

While a true E-R model of a system's data might include ternary or higher relationships, in most cases these relationships can be broken down into binary relationships and the system can be accurately modelled using only these binary relationships. Therefore, while an E-E matrix might not model the *true* nature of the system's data, it will still give an *accurate* model of it.

In transforming the E-R model into a relational model, there are certain rules that must be adhered to – the rules of normalization. Consequently, the resulting relational model and database are highly normalized. While this may be a good thing in theory, it will not always be desirable in practice. In such instances, the resulting relational model will have to be de-normalized to accommodate reality. Transaction based systems and systems where many detailed and complicated reports are required usually need to be de-normalized. They will also have tables and relationships that cannot be derived using the technique described above. In essence, the technique described above will not be able to derive all the entities and relationships in systems where most of the data is de-normalized.

Normalization produces many narrow tables with complex relationships. This means that many joins are necessary when querying data, and this can be slow. Although newer and faster RDBMSs will allow a large number of, and complicated, joins, there is still a high penalty for running such queries, which becomes evident in larger and more complicated databases. My recommendation is to use the method described above to create the initial relational model, then de-normalise as required.

While the technique described above produces a highly normalized database with only binary relationships, it is a fast and efficient way to model and create a relational database. In an era where agile methods are gaining popularity, this technique fits perfectly into the agile methodology – start with a simple working system and refine it.

## References

Kroenke D. (1999). *Database Processing: Fundamentals, Design, and Implementation*. (7<sup>th</sup> Edition)  
Pearson Publications.