BIT 6113 Database Management System

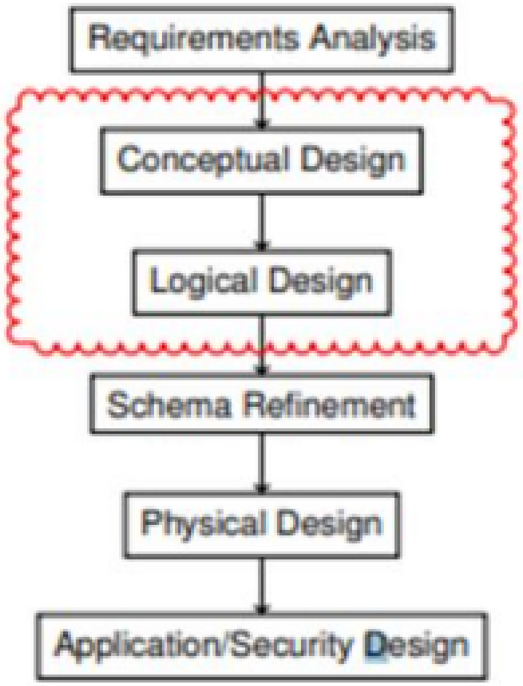
Saroj Ghimire
saroj.ghimire2@texascollege.edu.np

Lincoln University College, CN 1221

Conceptual Design

Saroj Ghimire

Database Design


Requirements Analysis
Conceptual Design
Logical Design
Schema Refinement
Physical Design
Application/Security Design

Conceptual Design

- Conceptual design is the first stage in the database design process. The goal at this stage is to design a database that is independent of database software and physical details. The output of this process is a conceptual data model that describes the main data entities, attributes, relationships, and constraints of a given problem domain.
- "All that is needed is there, and all that is there is needed". • In other words, make sure that all data needed are in the model and that all data in the model are needed. All data elements required by the database transactions must be defined in the model, and all data elements defined in the model must be used by at least one database transaction.

Conceptual Design

- The conceptual design has four steps,

1. Data analysis and requirements
2. Entity relationship modelling and normalization
3. Data model verification
4. Distributed database design

Conceptual Design

- **Data Analysis and Requirements:**

The first step in conceptual design is to discover the characteristics of the data elements. Appropriate data element characteristics are those that can be transformed into appropriate information. Therefore, the designer's efforts are focused on:

- Information needs. What kind of information is needed—that is, what output (reports and queries) must be generated by the system, what information does the current system generate, and to what extent is that information adequate?
- Information users. Who will use the information? How is the information to be used? What are the various end-user data views?
- Information sources. Where is the information to be found? How is the information to be extracted once it is found?
- Information constitution. What data elements are needed to produce the information? What are the data attributes? What relationships exist among the data? What is the data volume? How frequently are the data used? What data transformations are to be used to generate the required information? The designer obtains the answers to those questions from a variety of sources in order to compile the necessary information. Note these sources:

Conceptual Design

- Developing and gathering end-user data views. The database designer and the end user(s) interact to jointly develop a precise description of end-user data views. In turn, the end-user data views will be used to help identify the database’s main data elements.
- Directly observing the current system: existing and desired output. The end user usually has an existing system in place, whether it’s manual or computer-based. The designer reviews the existing system to identify the data and their characteristics.
- Interfacing with the systems design group. The database design process is part of the Systems Development Life Cycle (SDLC). In some cases, the systems analyst in charge of designing the new system will also develop the conceptual database model.

Conceptual Design

- **Entity Relationship Modeling and Normalization:**
- Before creating the ER model, the designer must communicate and enforce appropriate standards to be used in the documentation of the design. The process of defining business rules and developing the conceptual model using ER diagrams can be described using the following steps.

1. Identify, analyse, and refine the business rules

2. Identify the main entities, using the results of Step 1.

3. Define the relationships among the entities, using the results of Steps 1 and 2.

4. Define the attributes, primary keys, and foreign keys for each of the entities.

5. Normalize the entities. (Remember that entities are implemented as tables in an RDBMS.) 6.

Complete the initial ER diagram

7. Validate the ER model against the end users' information and processing requirements. 8.

Modify the ER model, using the results of Step 7.

Conceptual Design

- **Data Model Verification:**

The data model verification step is one of the last steps in the conceptual design stage, and it is also one of the most critical ones. In this step, the ER model must be verified against the proposed system processes in order to corroborate that the intended processes can be supported by the database model. Verification requires that the model be run through a series of tests against:

1. End-user data views.
2. All required transactions: SELECT, INSERT, UPDATE, and DELETE operations.
3. Access rights and security.
4. Business-imposed data requirements and constraints.

Conceptual Design

- **Distributed Database Design:**

Although not a requirement for most databases, sometimes a database may need to be distributed among multiple geographically disperse locations. Processes that access the database may also vary from one location to another. For example, a retail process and a warehouse storage process are likely to be found in different physical locations. If the database data and processes are to be distributed across the system, portions of a database, known as database fragments, may reside in several physical locations.

A database fragment is a subset of a database that is stored at a given location. The database fragment may be composed of a subset of rows or columns from one or multiple tables.

Requirement 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.
- Customer communication!
- Identify essential "real world" information
- Remove redundant, unimportant details C
- larify unclear natural language statements
- Fill remaining gaps in discussions

- Distinguish data and operations
- Requirement analysis & Conceptual Design aims at focusing thoughts and discussions !

Identify the Relation

Entities and attributes

- *Entities* are basically people, places, or things you want to keep information about. For example, a library system may have the *book*, *library* and *borrower* entities. Learning to identify what should be an entity, what should be a number of entities, and what should be an *attribute* of an entity takes practice, but there are some good rules of thumb. The following questions can help to identify whether something is an entity:

- Can it vary in number independently of other entities? For example, *person height* is probably not an entity, as it cannot vary in number independently of *person*. It is not fundamental, so it cannot be an entity in this case.

  - Is it important enough to warrant the effort of maintaining. For example *customer* may not be important for a small grocery store and will not be an entity in that case, but it will be important for a video store, and will be an entity in that case.

- Is it its own thing that cannot be separated into subcategories? For example, a car-rental agency may have different criteria and storage requirements for different kinds of vehicles. *Vehicle* may not be an entity, as it can be broken up into *car* and *boat*, which are the entities.

- Does it list a type of thing, not an instance? The video game *blow-em-up 6* is not an entity, rather an instance of the *game* entity.
- Does it have many associated facts? If it only contains one attribute, it is unlikely to be an entity. For example, *city* may be an entity in some cases, but if it contains only one attribute, *city name*, it is more likely to be an attribute of another entity, such as *customer*

Identify the Relation

The following are examples of entities involving a university with possible attributes in parentheses.

- **Course** (name, code, course prerequisites)
- **Student** (first_name, surname, address, age)
- **Book** (title, ISBN, price, quantity in stock)

An instance of an entity is one particular occurrence of that entity. For example, the student Rudolf Sono is one instance of the student entity. There will probably be many instances. If there is only one instance, consider whether the entity is warranted. The top level usually does not warrant an entity. For example, if the system is being developed for a particular university, *university* will not be an entity because the whole system is for that one university. However, if the system was developed to track legislation at all universities in the country, then *university* would be a valid entity.

Identify the Relation

Relationships

Entities are related in certain ways. For example, a borrower may belong to a library and can take out books. A book can be found in a particular library. Understanding what you are storing data about, and how the data relate, leads you a large part of the way to a physical implementation in the database.

There are a number of possible relationships:

- One-to-one (1:1)

This is where for each instance of entity A, there exists one instance of entity B, and vice-versa. If the relationship is optional, there can exist zero or one instances, and if the relationship is mandatory, there exists one and only one instance of the associated entity.

- One-to-many (1:M)

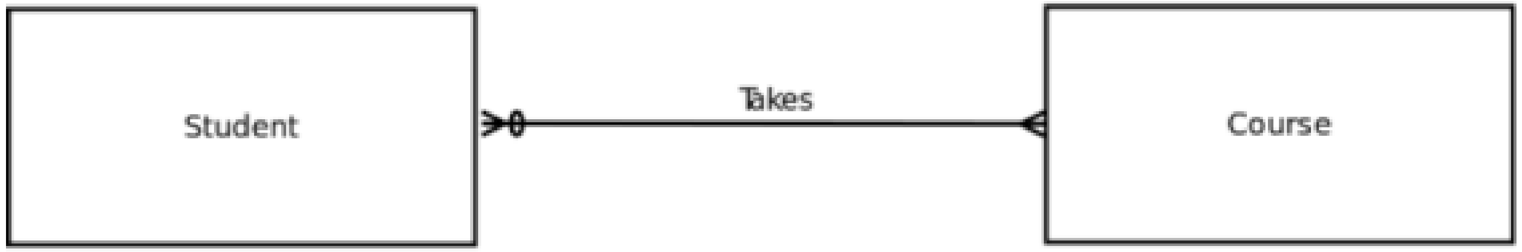
For each instance of entity A, many instances of entity B can exist, which for each instance of entity B, only one instance of entity A exists. Again, these can be optional or mandatory relationships.

- Many-to-many (M:N)

For each instance of entity A, many instances of entity B can exist, and vice versa. These can be optional or mandatory relationships.

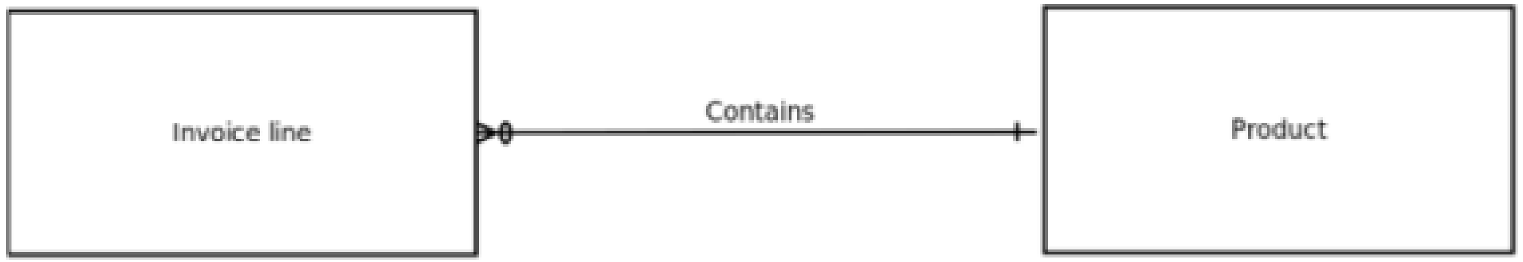
Identify the Relation

- There are numerous ways of showing these relationships. The image below shows *student* and *course* entities. In this case, each student must have registered for at least one course, but a course does not necessarily have to have students registered. The student-to-course relationship is mandatory, and the course-to-student relationship is optional.



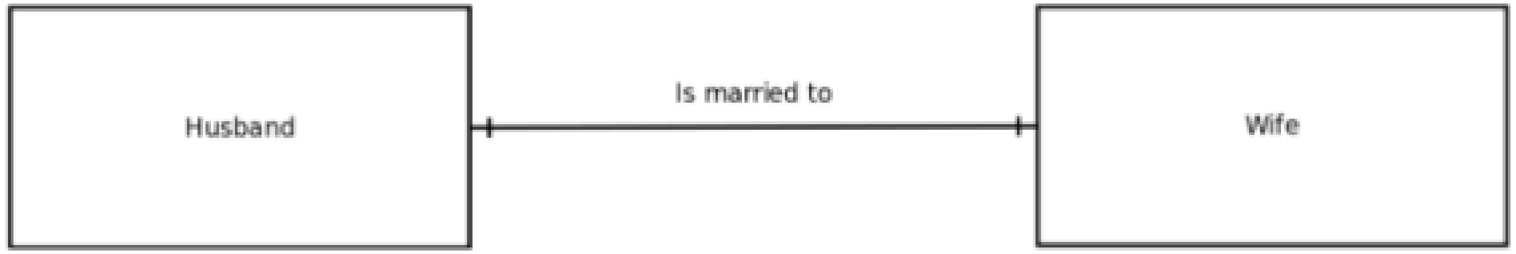
Identify the Relation

- The image below shows *invoice_line* and *product* entities. Each invoice line must have at least one product (but no more than one); however each product can appear on many invoice lines, or none at all. The *invoice_line-to-product* relationship is mandatory, while the *product-to-invoice_line* relationship is optional.



Identify the Relation

- The figure below shows husband and wife entities. In this system (others are of course possible), each husband must have one and only one wife, and each wife must have one, and only one, husband. Both relationships are mandatory.



Identify the Relation

- An entity can also have a relationship with itself. Such an entity is called a *recursive entity*. Take a *person* entity. If you're interested in storing data about which people are brothers, you wlll have an "is brother to" relationship. In this case, the relationship is an

M:N relationship.

• Conversely, a *weak entity* is an entity that cannot exist without another entity. For example, in a school, the *scholar* entity is related to the weak entity *parent/guardian*. Without the scholar, the parent or guardian cannot exist in the system. Weak entities usually derive their primary key, in part or in totality, from the associated entity. *parent/guardian* could take the primary key from the scholar table as part of its primary key (or the entire key if the system only stored one parent/guardian per scholar). • The term *connectivity* refers to the relationship classification.

• The term *cardinality* refers to the specific number of instances possible for a relationship. *Cardinality limits* list the minimum and maximum possible occurrences of the associated entity. In the husband and wife example, the cardinality limit is (1,1), and in the case of a student who can take between one and eight courses, the cardinality limits would be represented as (1,8).