Conceptual design

The purpose of the conceptual design phase is to build a conceptual model based upon the previously identified requirements, but closer to the final physical model. A commonly-used conceptual model is called an *entity-relationship* model.

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 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)

### 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.

#### 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.

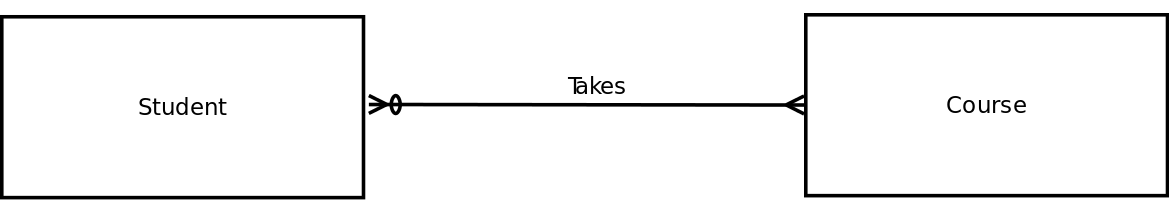
#### One-to-many (1:M)

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.

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.



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).

# Logical and Physical Design

Once the conceptual design is finalized, it's time to convert this to the logical and physical design

Logical design is the second stage in the database design process. The logical design goal is to design an enterprise-wide database based on a specific data model but independent of physical-level details. Logical design requires that all objects in the conceptual model be mapped to the specific constructs used by the selected database model. For example, the logical design for a relational DBMS includes the specifications for the relations (tables), relationships, and constraints (i.e., domain definitions, data validations, and security views).4. Validate logical model against user requirements

## 1. Map the Conceptual Model to the Logical Model:

The first step in creating the logical design is to map the conceptual model to the chosen database constructs. Logical design generally involves translating the ER model into a set of relations (tables), columns, and constraints definitions. The process of translating the conceptual model into a set of relations is depicted as follows.

1.map strong entities  
2. Map supertype/subtype relationships  
3. Map weak entities  
4. Map binary relationships  
5. Map higher degree relationships

## 

## 2.Validate the Logical Model Using Normalization:

The logical design should contain only properly normalized tables. The process of mapping the conceptual model to the logical model may unveil some new attributes or the discovery of new multivalued or composite attributes. Therefore, it’s very likely that new attributes may be added to tables or entire new tables added to the logical model. For each  identified table (old and new), you must ensure that all attributes are fully dependent on the identified primary key and that the tables are in at least third normal form (3NF).

## 3. Validate Logical Model Integrity Constraints:

The translation of the conceptual model into a logical model also requires the definition of the attribute domains and appropriate constraints. For example, the domain definitions for the CLASS\_CODE, CLASS\_DAYS, and CLASS\_TIME attributes of the CLASS entity are written this way:

CLASS\_CODE is a valid class code.  
         
            Type: numeric  
            Range: low value = 1000 high value = 9999  
            Display format: 9999  
            Length: 4

CLASS\_DAYS is a valid day code.  
  
            Type: character  
            Display format: XXX  
            Valid entries: MWF, TTh, M, T, W, Th, F, S  
            Length: 3

CLASS\_TIME is a valid time.  
             
            Type: character  
            Display format: 99:99 (24-hour clock)  
            Display range: 06:00 to 22:00  
            Length: 5

## 4. Validate the Logical Model against User Requirements:

The logical design translates the software-independent conceptual model into a software-dependent model. The final step in the logical design process is to validate all logical model definitions against all end-user data, transaction, and security requirements. The stage is now set to define the physical requirements that allow the system to function within the selected DBMS/hardware environment.

# Physical Database Design:

Physical database design translates the logical data model into a set of SQL statements that define the database. For relational database systems, it is relatively easy to translate from a logical data model into a physical database.

* Entities become tables in the physical database.
* Attributes become columns in the physical database. Choose an appropriate data type for each of the columns.
* Unique identifiers become columns that are not allowed to have NULL values. These are referred to as primary keys in the physical database. Consider creating a unique index on the identifiers to enforce uniqueness.
* Relationships are modeled as foreign keys.

.