The Entity Relationship Model (ERM)

The Entity Relationship Model is a representation of the conceptual database as viewed from the end user perspective.

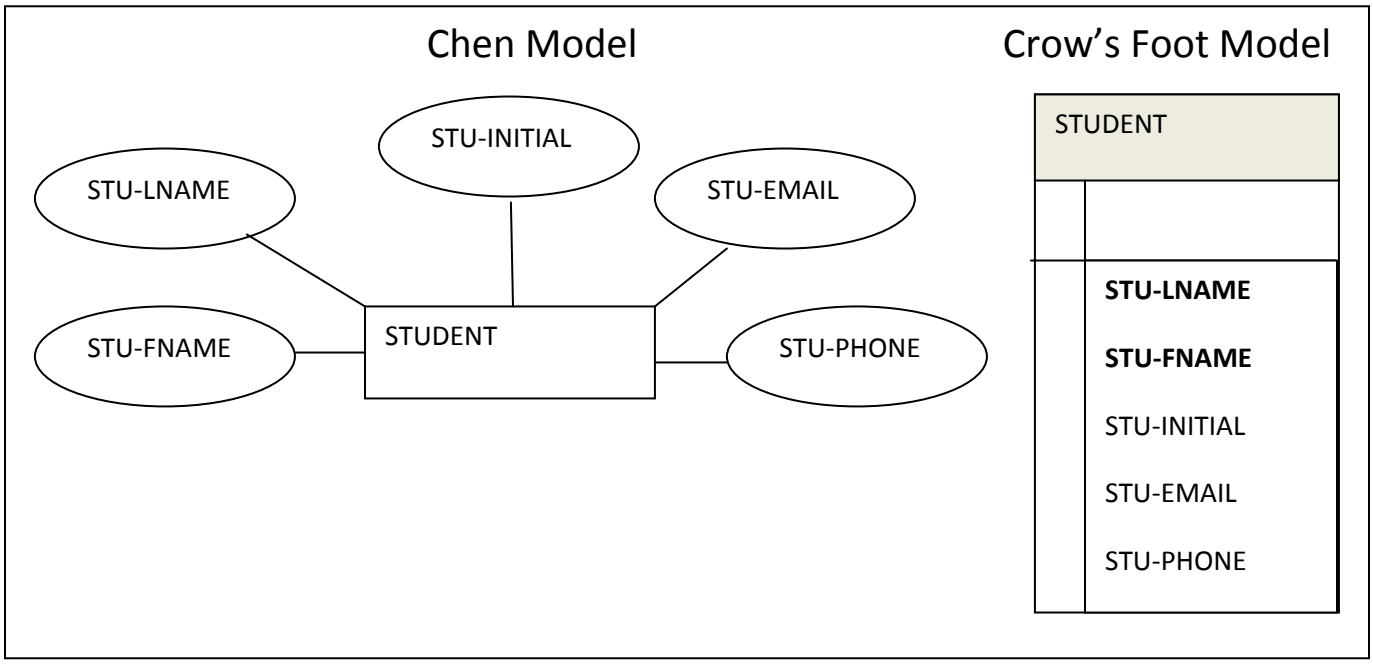
The various notations used are the Chen notation and the Crow's Foot and the UML notations.

Entities:

An entity is an object of interest to the end user. An entity corresponds to a table. In all notations, an entity is represented by a rectangle that contains the entity's name written in capital letters.

Attributes:

Attributes are characteristics of entities and are usually represented as fields of the tables at the implementation level. In the Chen model, attributes are represented as ovals, each containing the name of the attribute it represents. In the Crow's Foot notation, the attributes are written in the attribute box below the entity rectangle.



Required and Optional Attributes:

A required attribute is an attribute that must have a value, and it cannot be left empty. The two boldfaced attributes in the Crow's Foot notation indicate that the data entry is required.

An optional attribute is an attribute that does not require a value, it can be left empty.

Domains:

A domain is the set of possible values for a given attribute. Attribute may share a domain. The same attribute name is used for different entities, and then they share the same domain.

Identifiers or Primary keys:

In ERM, an identifier is one or more attributes that uniquely identify each instance or tuple. In the relational model, entities are mapped to tables and the entity identifier is mapped as the table's primary key (PK). Identifiers and primary keys are underlined in the ERD.

Key attributes are also underlined in the relational schema:
TABLE_NAME (**KEY_ATTRIBUTE1**, ATTRIBUTE2, ……, ATTRIBUTE K)

Composite Identifiers:

Ideally, an entity identifier is only composed of only a single attribute. However, it is possible to have a composite identifier, which is a primary key that is composed of more than one attribute.

Composite and Simple Attributes:

Attributes are classified as simple or composite. A composite attribute is an attribute that can be further subdivided to yield additional attributes. For example, the attribute PHONE_NUMBER can be subdivided into area code and exchange number.

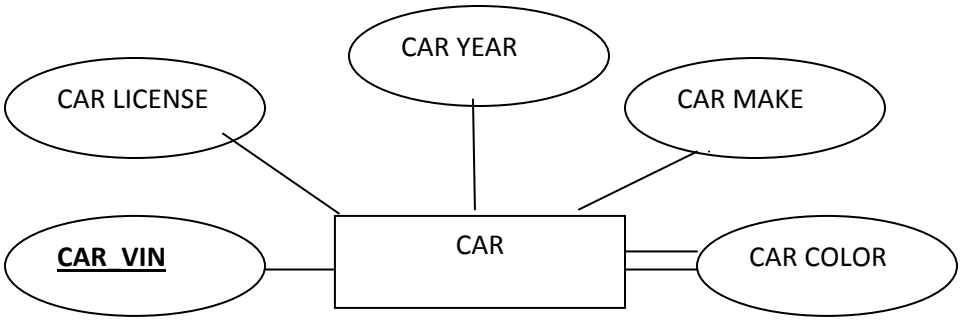
Single_valued attribute:

A single-valued attribute is an attribute that can have only a single value. For example, a person can have only one SSN.

A single valued attribute is not necessarily a simple attribute. For instance, a part's serial number such as SE-08-02-16745 is single-valued but is a composite attribute that can be divided into the production region SE, the plant within the region: 08, the shift: 02 and the part number.

Multi-valued Attributes:

Multi-valued attributes are attributes that can have many values. For example, a person may have several college degrees , and a household may have several phone numbers. In the Chen ERM, multivalued attributes are shown by a double line connecting the attribute to the entity. The Crow's Foot notation does not identify multivalued attributes.



Implementing Multi-valued Attributes:

Although conceptually, the model can handle M:N relationships and mutivalued attributes, they cannot be implemented in the

RDBMS. So, if multivalued attributes exist, the designer must decide two possible course of action:

1. Within the original entity, create several new attributes, one for each component of the original mutivalued attribute. For example, the CAR entity's attribute CAR_COLOR can be split to create new attributes CAR_TOPCOLOR, CAR_BODYCOLOR and CAR_TRIMCOLOR which are then assigned to the CAR entity. The problem with this approach is that if there are many options for some tuples and not for all tuples, then the values for most of the new attributes is going to be NULL.
2. Create a new entity composed by the original multivalued attribute's components. The new entity allows the designer to define different values for the attribute. Doing it this way, allows us to define as many colors for the car as needed without having to change the table structure. This is the preferred way to deal with multivalued attributes. Creating a new entity in a 1:M relationship with the original entity yields several benefits: it is more flexible, expandable solution, and it is compatible with the relational model.

Derived Attributes:

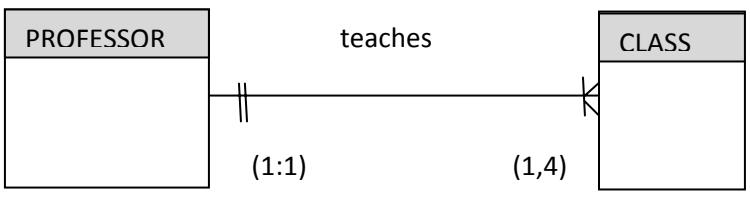
A derived attribute or a computed attribute is an attribute whose value is calculated (derived) from other attributes. The derived attribute need not be physically stored in the database.

Relationships:

A relationship is an association between entities. The entities that participate in a relationship are also called participants. A relationship between entities always operates in both directions. To define the relationship between the entities name, entity relationships may be classified as one-to-one, one-to-many, or many-to-many.

Connectivity and Cardinality:

Cardinality expresses the minimum and maximum number of tuples associated with one tuple of the related entity. In the ERD cardinality is indicated by placing the appropriate number besides the entities, using the format (x,y). The first value represents the minimum number of associated tuples, while the second value represents the maximum number of associated tuples.



(1:4) means that each professor teaches up to 4 classes. This means that a PROFESSOR' tuple primary key occurs at least once and at most four times as a foreign key values in the CLASS table.

(1:1) indicates that each class is taught by one and only one professor. That is each tuple from the CLASS table is associated with one and only one tuple from the PROFESSOR's table.

Existence Dependence:

An entity is said to be existence-dependent if it can exist in the database only when it is associated with another related entity.

In implementation terms, an entity is existence-dependent if it has a mandatory foreign key, which cannot be NULL. Example: an employee dependents entity can only exists if there is a tuple for a specific employee in the EMPLOYEE table.

Strong Entities: If an entity can exist apart from all its related entities, then it is existence-independent and it is referred to as strong entity or regular entity.

Relationship Strengths: (Strong Relationships)

The concept of relationship strength is related on how primary keys (PK) are defined among entities. To implement a relationship, the primary key of one entity (the parent entity , normally on the “one” side of the one-to-many relationship) appears as a foreign key (FK)in the related entity (the “many” side on the one-to-many relationship). Sometimes, the foreign key is also a primary key in the related entity.

Weak or Non_identifying Relationships:

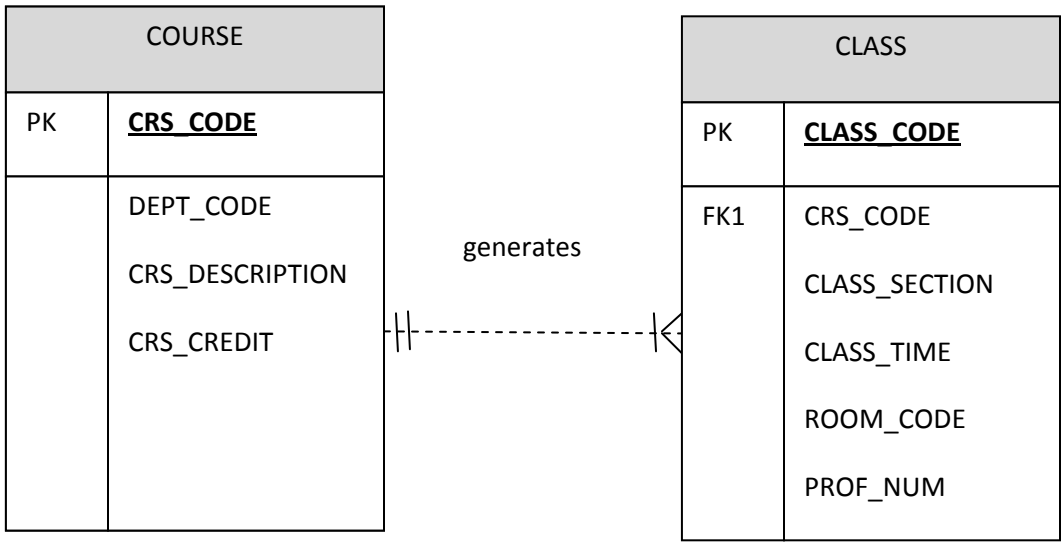
A weak relationship exists if the primary key of the related entity does not contain a primary key component of the parent entity. By default, relationships are established by having the primary key of the parent entity appear as a foreign key on the related entity(child entity)

Example, suppose we have a 1:M relationship between COURSE and CLASS as:
COURSE(**CRS_CODE,**DEPT_CODE,CRS_DESCRIPTION,CRS_CREDIT)

CLASS(**CLASS_CODE,** CRS_CODE, CLASS-SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)

In this case, a weak relationship exists between COURSE and CLASS, because the PK of COURSE is a FK for CLASS.

In the CROW notation, a weak relationship is represented by placing a dashed relationship line between entities



Strong (Identifying) Relationships:

A strong relationship exists when the primary key of the related entity contains a primary key component of the parent entity. For example, suppose the 1:M relationship between CLASS and COURSE is defined as:

COURSE(**CRS_CODE,** DEPT_CODE, CRS_DESCRIPTION, CRS_CREDIT)

CLASS(**CRS_CODE, CLASS_CODE,** CLASS-SECTION, CLASS_TIME, ROOM_CODE, PROF_NUM)

In this case, the CLASS entity primary key is composed of CRS_CODE and CLASS_SECTION. Therefore, a strong relationship exists between COURSE and CLASS.

Weak Entities:

In contrast to strong or regular entities mentioned earlier, a weak entity is one that meets two conditions:

1. The entity is existence-dependent, it cannot exist without the entity with which it has a relationship.
2. The entity has a primary key that is partially or totally derived from the parent entity in the relationship.

For example, a company insurance policy may cover an employee and their dependents. An EMPLOYEE may or may not have any DEPENDANTS, however a DEPENDENT must be associated with an EMPLOYEE and if the EMPLOYEE is deleted then the DEPENDENT is also deleted. DEPENDENT is a weak entity in the relationship “EMPLOYEE has DEPENDENT”.

Relationship Participation:
Participation in a relationship is either optional or mandatory.
Optional participation: means that one entity occurrence does not require a corresponding entity occurrence in the

relationship. For example, an EMPLOYEE may have no DEPENDENT. Therefore the DEPENDENT entity is optional for the EMPLOYEE entity. The Crow's Foot notation , an optional relationship is shown by drawing a small circle(O) on the side of the optional entity. The corresponding minimal cardinality is 0.

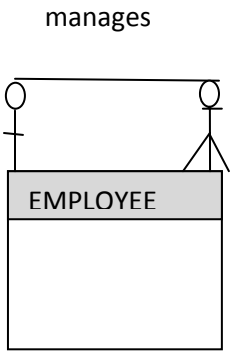
Mandatory participation: means that one entity occurrence requires a corresponding entity recurrence in a particular relationship. If no optionality symbol is depicted, the relationship is assumed to be mandatory. The existence of a mandatory participation implies a connectivity of 1 and a minimum cardinality of 1.

Relationship degrees:

A relationship degree indicates the number of entities or participants associated with the relationship.

A unary relationship exists when an association is maintained within a single entity. For example: an employee within the EMPLOYEE entity is the manager of one or more employees within the entity. In this case, the "manages" relationship means that EMPLOYEE requires that another EMPLOYEE be the manager that is EMPLOYEE has a relationship with itself .

This is known as a recursive relationship. Recursive relationships are represented like:



A binary relationship requires two entities to be related. These are the most common type of relationships and usually higher order relationships are decomposed into appropriate equivalent binary relationships.

A ternary relationship exists when three entities are related. Although, they are rare, they are sometimes necessary. A ternary relationship implies an association among three different entities. Example:

- A DOCTOR writes one or more PRESCRPTIONS
- A PATIENT may receive one or more PRESCRIPTIONS
- A DRUG may appear in one or more PRESCRIPTIONS.

A doctor can be scheduled for many appointments, but may not have any scheduled at all. Each appointment is scheduled with exactly 1 doctor. A patient can schedule 1 or more appointments. One appointment is scheduled with exactly 1 patient.

An appointment must generate exactly 1 bill, a bill is generated by only 1 appointment. One payment is applied to exactly 1 bill, and 1 bill can be paid off over time by several payments. A bill can be outstanding, having nothing yet paid on it at all. One patient can make many payments, but a single payment is made by only 1 patient. Some patients are insured by an insurance company. If they are insured, they can only carry insurance with one company. An insurance company can have many patients carry their policies. For patients that carry insurance, the insurance company will make payments, each single payment is made by exactly 1 insurance company.