**E-R model and E-R diagram**

Conceptual modeling is an important phase in designing a successful database application. Generally, the term database application refers to a particular database—for example, a BANK database that keeps track of customer accounts—and the associated programs that implement the database queries and updates—for example, programs that implement database updates corresponding to customers making deposits and withdrawals. These programs often provide user-friendly graphical user interfaces (GUIs) utilizing forms and menus. Hence, part of the database application will require the design, implementation, and testing of these application programs**.**

**Entity-Relationship (ER) model**, which is a popular high-level conceptual data model. This model and its variations are frequently used for the conceptual design of database applications, and many database design tools employ its concepts Entity-relationship model describes data involves in real world in terms of object and their relationships. It is widely used for initial database design. It describes overall structure of database. E-R model is in fact, semantic data model which describes the meaning of data. It has a capability to map the meanings and interactions of real world objects on to the conceptual schema.

**Entity Types :-** there are two types of entities namely, dependent entities (also called weak entities) and independent entities(also called strong entities).

* 1. The **dependent(Weak) entity** is the one whose existence depends on another entity. An entity set is called a weak entity set if its existence depends on the existence of another weak entity or any other strong entity. A weak entity set does not have sufficient attributes to form a primary key.

Weak entity

* + 1. Strong entity set:- the entities which are not dependant on any other entities or the entities which have own unique primary key is known as strong entity. The rectangle symbol is use to represent strong entity.

Strong entity set

**Attributes :-** Each column in a relational table represents a field, also referred to as an attribute. Attributes are characteristics of an entity.

Consider the entity set employee with attributes employee-name and telephone-number. It can easily be argued that a telephone is an entity in its own right with attributes telephone-number and location.

**Types of Attribute:**

1. **Simple Value Attribute (Name) and composite attributes :** the attribute which is not divided in subparts is called simple attribute. like as symbol number, age etc

Attribute that can further divide into subparts called composite attribute. For example,

customer\_name in customer entity set is composite attribute since it can be divided into sub

attributes: customer\_fname, customer\_mname and customer\_lname. Composite attributes

helps to group related attributes, which makes modeling clearer.

1. **Multi Value Attribute(Phone, add)** the attributes which are divided in several parts is called multi value attribute or composite attribute.
2. **Inherited Attribute.(Number of students)** the attribute whose values is always depend to another attribute is called inherited attribute.
3. **Null Values**

In some cases a particular entity may not have an applicable value for an attribute. For example, the ApartmentNumber attribute of an address applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes. Similarly, a CollegeDegrees attribute applies only to persons with college degrees. For such situations, a special value called **null** is created. An address of a single-family home would have null for its ApartmentNumber attribute, and a person with no college degree would have null for CollegeDegrees. Null can also be used if we do not know the value of an attribute for a particular entity—for example, if we do not know the home phone of "John Smith"

**Degree of Relationship :**

Degree of a Relationship Type. The degree of a relationship type is the number of participating entity types. Hence, the WORKS\_FOR relationship is of degree two. A relationship type of degree two is called binary, and one of degree three is called ternary. An example of a ternary relationship is SUPPLY, shown in Figure 7.10, where each relationship instance ri associates three entities—a supplier s, a part p, and a project j—whenever s supplies part p to project j. Relationships can generally be of any degree, but the ones most common are binary relationships. Higherdegree relationships are generally more complex than binary relationships.

|  |
| --- |
|  |

**Constraint on E-R model :-**

E-R model has a capability to enforce constraints. Two most important type of constraints in ER model are- Mapping Cardinalities (Cardinality ratio), Participation Constraints

***Mapping Cardinalities***

Mapping Cardinalities describes no. of entities to which another entity can be associated via

relationship set. Mapping cardinalities are most useful in describing binary relationship sets but

it can also describe relationship sets that involve more than two entity sets. For binary

relationship set between entity set A and B mapping cardinality must one of the following.

**One to one**: An entity in A is associated with at most one entity in B and entity in B is

associated with at most one entity in A.

**One to many**: An entity in A is associated with zero or more entities in B but entity in B

can be associated with at most one entity in A.

**Many to one**: An entity in A is associated with at most one entity in B but an entity in B

can be associated with zero or more entities in A.

**Many to many**: An entity in A is associated with zero or more entities in B, and an

entity in b is associated with zero or more entities in A.

|  |
| --- |
|  |

The appropriate mapping cardinality for a particular relationship set depend upon real world

situation that the relationship set is going to modeling. For example: in borrower relationship

set, if loan can belong to any one customer and customer can have several loan then

relationship set from customer to loan is one to many. If a loan can belong to several customer

(loans taken jointly by several customers) then relationship set is many to many.

***Participation Constraints***

The participation of an entity set E in a relationship set R is said to be *total* if every entities in E

participates in at least one relationship in R. If only some entities in E participate in relationship

in R, then participation of entity set E in relationship set R is said to be *partial*. The participation

of loan in the relationship set borrower is total but customer entity set in borrower relationship

set is partial since not all customers necessarily take loan from bank, customer may also those

who are only account holder. Such participation constraint can be express by E-R model.

**Keys**

The concept of key is important to distinguish one entity from another and one relationship

from another relationship. In fact, values of attributes distinguish one entity from another

entity. To distinguish one entity from another entity in entity set there must exist attribute/s

whose values must not duplicate in entity set. It ensures no two entities in an entity set can

exist with same values for all attributes.

1. **Super key**

A super key is a set of one or more attributes which uniquely identifies an entity in entity set.

For example: in customer relation single attribute customer\_id is sufficient to uniquely identify one customer entity to another. So customer\_id is a superkey in a customer relation. Since combination of customer\_id and customer\_name can also uniquely identifies one customer entity to another. So combination of attributes {customer\_id,customer\_name} is also superkey in relation customer. But single attribute customer\_name can not superkey in relation customer because customer name only can not uniquely identify one customer entity to another, there would be number of customers having same name.

1. **Candidate key:**  A candidate key is an attribute or set of attributes that uniquely identifies a record. These attributes or combinations of attributes are called candidate keys. In such a case one of the candidate key is chosen to be a primary key. The remaining candidate keys are called alternate keys.
2. **Primary key :**  the primary key uniquely identifies each record in a table and must never be the same for two records. For example, emp-code can be primary key for the entity set employees.
3. **Composite primary key :-** in many cases, as we design a database, we will have tables that will use more than one column as part of the primary key. These called composite keys or concatenated keys. For example Item\_no+ Order\_no
4. **Domain Key :-**  when in a column, only set of predefine values allowed to be inserted is called Domain key. For example, city can be one of them(Chitwan, Pokhara, Kathmandu).
5. **Referential key (Foreign key)** : when one table primary key is exist into another table in second table it is called foreign key.

**ER diagram consists of the following major components**

|  |  |
| --- | --- |
| **Rectangles,** Which represent entity set. |  |
| **Ellipses,** which represent attributes |  |
| **Diamonds**, which represent relationship sets. |  |
| **Lines**, which link attributes to entity sets and entity sets to relationship set |  |
| **Double ellipses,** which represent **multi valued** attributes. |  |
| **Dashed Ellipses,** which denote derived attributes |  |
| **Double lines**, which indicate total participation of an entity in a relationship set. |  |
| **Double rectangles**, which represent weak entity sets. |  |
| **Double line Diamond,** which represents the relationship set for weak entity set. |  |
| **Ellipses with solid line inside**, represents primary key |  |
| **Ellipses with dashed line inside**, represents discriminating attribute of weak entity set |  |
| Many to Many Relationship |  |
| One to one relationship |  |
| ● Many to One Relationship |  |
| one to many relationship |  |
| ISA (Specialization) | ISA |

**Weak Entity Sets and their representation in E-R Diagram**

Entity set that does not have primary key known as weak entity set and entity set that has a

primary key known as strong entity set.

Let us consider entity set

Payment= (payment\_number,payment\_date,payment\_amount)

Here, payment numbers are typically sequence of numbers, starting from 1 and generated for

each loan. Thus although each payment entry is distinct, payments for different loan may share

the same payment number. Thus, this entity set does not have a primary key; it is a weak

entity set

For weak entity set to be meaningful, it must be associated with another entity set called

identifying or owner entity set. The relationship between weak entity set and identifying set

known as identifying relationship. The identifying relationship is many to one from the weak

entity set to identifying entity set, and participation of weak entity set in relationship is total.



Figure: E-R diagram with weak entity set.

**Specialization**

Specialization follows top down design approach. Entity sets are subgroups in distinct entity sets. For example entity set person with attributes name, street and city can further subgroup into two entities sets customer and employee. Each of these person types can describes by set of attributes that includes all the attributes of entity set person plus all possible attributes of itself. For example, customer entity set can further described by set of attributes: customer\_id,enroll\_date etc. Similarly entity attributes can further describes by set of attributes: employee\_id, salary etc. The process of sub groupings within an entity set is called specialization. We can apply specialization repeatedly to refine a design schema. For instance bank employees may be further classified into officer, teller or secretary.

In E-R diagram, specialization can be represented by a triangle component labeled ISA. The label ISA stands for “is a “. For example customer is a person, officer is an employee etc. The ISA relationship also called super class-subclass relationship

**Generalization**

Generalization follows bottom-up approach in which multiple entity sets are synthesized into higher-level entity set on the basis of common features. For example, the database designer may have first identified a customer entity set with the attributes: name,street, city and customer\_id and employee entity set with the attributes name, street, city, employee\_id and salary. In both entities some attributes are common. These similarities between these two entities can be express by generalization.

|  |
| --- |
|  |

Figure**: Specialization and generalization.**

During the course of database design or E-R schema for enterprise database designer may use both specialization and generalization process. Specialization and generalization in E-R diagram represent by a same way. The terms specialization and generalization are used interchangeably.

**Constraints on Generalization and Specialization**

Database designer can enforce certain constraints in generalization and specialization. These

**constraints are as follows**

**(a)** Constraint on which entities can be member of a given lower level entity set

Membership of lower-level entity can be express one of the following way

**Condition defined:**

Database designer can define condition to lower level entity set that must follows entities in higher level for member of lower level entity set. Suppose account is a higher level entity set with attribute account\_type. Assume that saving\_account and checking\_account are two lower level entity set. To specify which entities in account belongs which lower level entity set database designer can specify condition for each lower level entity set. For saving\_account entity set, database designer can enforce membership condition account\_type=’saving account’ and for checking\_account entity set, database designer can enforce membership condition account\_type=’checking account’. This type of specialization/generalization known as *attribute defined*.

**User defined:**

It does not enforce any membership condition in lower-level entity set. Database user itself assigns entities into other entity set. For instance, we can assume that after 3 month of employment, bank employees are assign in one of the available workgroups.

**(b)** Constraint on whether or not entities may belong to more than one lower-level entity set

within a single generalization.

To indicate this constraint lower level entity set may one of the following

**Disjoint:**

Disjointness constraints enforce an entity can belong to only one lower level entity set. In previous account example, entities in account can either belong to saving\_account or checking\_account, can not both. In E-R diagram it can be express by writing disjoint next to the ISA triangle.

**Overlapping**

In overlapping generalization/specialization, same entity may belong to more than one lower-level entity set. For example: employee may involve in more than one workgroups. Same people may customer as well as employee. Lower level entity overlap is default case.

**(c)** Completeness constraint: specifies whether or not an entity in the higher-level entity set

must belong to at least one of the lower-level entity sets within a generalization/

specialization. This constraint may specify as follows

**Total generalization/specialization**: tells each higher level entities must belong to one of

the lower-level entity sets. In E-R diagram total generalization/specialization specifies by using a double line to connect the box representing higher level entity set to the triangle symbol.

**Partial generalization/specialization**: tell not all entities in higher level need to belong to

one of the lower-level entity sets. Partial generalization is default.

**Aggregation**

E-R model can not express relationship among relationship. To illustrate this, let us consider

quaternary relationship manages among employee, branch, job and manager. Its main job is to record managers who manages particular job/task perform by particular employee at particular branch.

E-R diagram with redundant relationships.

This quaternary relationship is required since binary relationship between manager and

employee can not represent required information. This E-R diagram is able to represent the

required information but information are redundant since every employee, branch and job exist both relationship set “work-on” and “manages”. Here aggregation is better to represent such information.

Aggregation is in fact an abstraction it treats relationships as higher level entities. In our

example, it treats relationship set work-on (including entity set employee, branch and job) as

entity set. So now we can create binary relationship set “manages” between work-on and

manager. This removes redundant information.



E-R Diagram with aggregation.

**Subclass / Superclass :-**

We call the relationship between a superclass and any one of its subclasses a **superclass/subclass** or **supertype/subtype** or simply **class/subclass relationship**. In our example, EMPLOYEE/SECRETARY and EMPLOYEE/TECHNICIAN are two class/subclass relationships. Notice that a member entity of the subclass represents the *same real-world entity* as some member of the superclass; for example, a SECRETARY entity ‘Joan Logano’ is also the EMPLOYEE ‘Joan Logano.’ Hence, the subclass member is the same as the entity in the superclass, but in a distinct *specific role.*When we implement a superclass/subclass relationship in the database system, however, we may represent a member of the subclass as a distinct database object— say, a distinct record that is related via the key attribute to its superclass entity.

|  |
| --- |
|  |
|  |

**Example of E-R diagrams**

|  |
| --- |
|  |

|  |
| --- |
|  |

**Relational Model :**

**Relational Model**

Relational database model is a primary data model for commercial data-processing applications. It is popular because of its simplicity; it provides simple but powerful way of

representing data. It also supports complex query. Database in a relational model is simply

a collection of one or more relations, where each relation is represented by table with rows

and columns. It allows simple high level languages to query data.

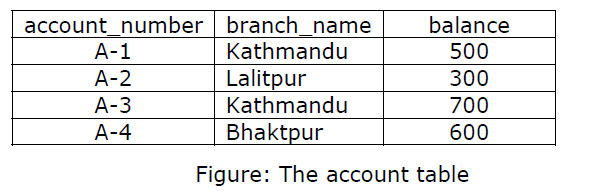
**Structure of Relational Databases**

In relational model, table is a major construct for representing data. Relational database

consist set of tables. A row in a table represents a relationship among set of values. So

table can be refers as a collection of such relationship.

To illustrate the basic structure of database, let us consider a table “account”



The columns of table “account” are: account\_number, brance\_name and balance refer attributes. The set of permitted values for each attribute known as domain of that attribute. For example: set of all accounts numbers is a domain of attribute account\_number.

Let D1 denotes set of all account number, D2 denotes set of all branch names and D3 denotes set of all balances. Any row of table “account” must consist 3-tuple (v1, v2, v3), where v1 is account number (i.e. v1 is a domain of D1), v2 is branch name (i.e. v2 is a domain of D2), and v3 is account balance (i.e. v3 is a domain of D3). In general, we can express table “account” will contain only subset of all possible rows. That is, “ account” is a subset of

D1×D2×D3

In general, a table of n attributes must be subset of

D1×D2×D3×. . ×Dn-1×Dn

For each relations r, domains of all attributes of r must be atomic. The domain is said to be ***atomic*** if elements of domain is indivisible unit. Domains of multivalued and composite attributes are **nonatomic**.

Several attributes may have same domain. Suppose we have two relation customer with customer\_name as one of its attribute and employee is another relation with one of its attribute as employee\_name. It is possible that attribute customer\_name and employee\_name may have same domain. If we look attributes: customer\_name and branch\_name of relations customer and account respectively, at physical level, their domain may be same, both are defined by set of character string. But at logical level, customer\_name and branch\_name must have distinct domain. Null value is a member of any possible domain. It signifies value is unknown or does not exist. Domain for

customer\_phoneno of customer entity set may null, meaning is that particular customer does not have any phone number or phone no. is not available.

**Database Schema**

A relation schema is a list of attributes and their corresponding domains. For example, the relation schema for relation customer is express as

Customer-schema = (customer\_id, customer\_name, customer\_city)

We may also specify domains of attributes as

Customer-schema = (customer\_id: integer, customer\_name: string,

customer\_city:string)

We may state customer is a relation on Customer-schema by

customer(Customer-schema)

Values or data contain in relation change when it is updated. Relation instance is a snapshot of data in relation at particular time. But, in general, we simply say relation even it is actually relation instance.

Database schemas for banking enterprise

Branch-schema = (branch\_name,branch\_city,assets)

Account\_schema = (account\_number,branch\_name,balance)

Customer-schema = (customer\_id,customer\_name,customer\_street,customer\_city)

Depositor-schema = (customer\_id,account\_number)

Loan-schema = (loan\_number,branch\_name,amount)

Borrower-schema = (customer\_id,loan\_no)

**Schema Diagram**

Schema diagram is a graphical representation of database schema along with primary key

and foreign key dependencies. In schema diagram, each relation is represented by box where attributes are listed inside box and relation name is specified above it. Primary key in relation is place above the horizontal line that crosses the box. Foreign key in schema diagram appear as arrow from the foreign key attributes of the referencing relation to the primary key of the referenced relation.

**Figure: Schema**

