Data Model:

- A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey.

Categories Of Data Model:

- **Conceptual (high-level, semantic) data models:**
  - Provide concepts that are close to the way many users perceive data.
    - (Also called ***entity-based*** or ***object-based*** data models.)
- **Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
  - Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Entity-Relationship Model

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.

2.1 Entity and Entity Sets

An entity is a "thing" or "object" in the real world that is distinguishable from another object. For example:

- Specific customer , Particular course in university

Entities can be described by a set of properties called *attributes*. For example: customer_id, customer_name, customer_address are attributes for entity customer. Similarly, course_id, course_name are attributes for entity course.

An entity set is a set of entities of the same type that share the same properties. Entities of an entity set has same set of attributes.
For example

- Set of all customer
- Set of all courses in an university

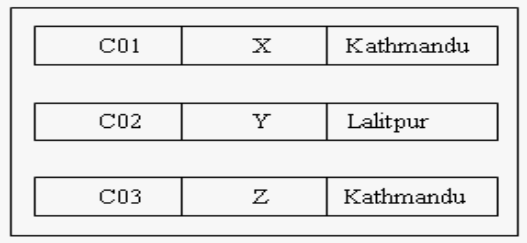


Figure 2.1.1 instance of customer entity set

2.2 Attributes

In simple, attribute is descriptive property of entity set. Set of attributes describes entity set.
For example

customer = (customer-id, customer-name, customer-city) account=(
account_number, balance)
loan = (loan_number, amount)

The set of permitted values for an attribute called domain of that attribute. For example

- set of all text strings of certain length for customer_name is domain of that attribute.
- set of all strings like "A-n" where n is a positive integer for account_number is domain of that attribute.

Formally, an attribute is a function which maps an entity set into a domain. Every entity is described by a set of (attribute, data value) pairs. There is one pair for each attribute of the entity set. For example: particular customer entity in customer entity set can describe by the set of pairs (customer_id, c01), (customer_name, x) and, (customer_city,Kathmandu).

2.2.1 Types of Attributes

Simple and Composite attribute

Attribute which can not be divide into subparts (i.e. into other attributes) called simple attribute. For example, customer_id in customer entity set is simple attribute, since it can not divide into sub attributes.

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.

Single-valued and Multivalued attributes

Attribute that can take only one value in every entry called singled-valued attribute. For example, attribute customer_name in customer entity set is single-valued attribute since it can not contain more than one customer name in any entry. An attribute that can take more than one values in any entry called multivalued attribute. For example, in a customer entity set attribute customer_phonenumber is multivalued attribute since customer may have zero or one or several phone number.

Stored and Derived attribute

Attribute whose values can be derived from the values of other related attributes or entities called derived attribute. For example, in customer entity set, attribute age is derived attribute if customer entity set has attribute date_of_birth. We can derive age of customer from date_of_birth and current_date. Here the attribute date_of_birth is stored attribute and the attribute age is derived attribute. The value of derived attribute is not stored, it is computed when required.

2.3 Relationships and Relationship Sets

A relationship is an association among two or more entities. A relationship set is a set of relationship of same type.

Formally, if $E_1, E_2, \ldots, E_n$ $(n \geq 2)$ are entity sets then a relationship set R is a subset of

$$\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots e_n \in E_n\}$$ where $(e_1, e_2, \ldots, e_n)$ is relationship.

Example: For two entity sets customer and account, we can define relationship set depositor which associates each customer to their corresponding account he/she has.

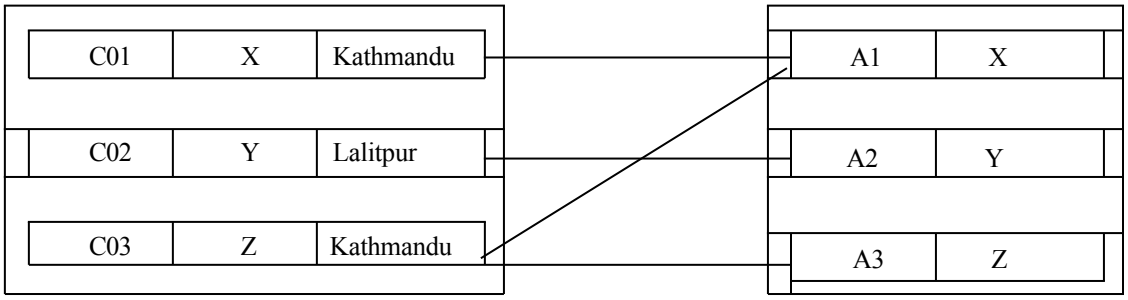


Figure: instance of depositor relationship set

The relationship set may have descriptive attributes, which generally used to record information about the relationship. For example: in depositor relationship with entity sets customer and account we can associate attribute access_date to the relationship set depositor to specify the most recent date on which a customer accessed an account.

Recursive relationship set

The function that an entity plays in relationship called that entity's role. In relationship set, usually roles of entity's are not specified but it is useful if meaning of relationship need to clear. In such case, same entity may participate in relationship set more than once with different roles. This type of relationship set called recursive relationship set.
Example:
Let us consider entity set employee that records information about all employees of bank. We may have relationship set "work-for" as recursive relationship set because same employee may plays worker as well as manager.

Binary and ternary relationship set

Relationship set that involves only two entity sets known as binary-relationship set. For example: depositor relationship set is a binary relationship set where relationship set involves only two entity set "customer" and "account".

Most relationship sets in database system are binary. However relationship set may involves in more than two entity sets. Relationship set that involves three entity sets known as ternary relationship. For example: the relationship set "work-on" among employee, branch and job is example of ternary relationship.

The no. of entity sets that participate in relationship set refers degree of relationship set. Here degree of ternary relationship is 3.

2.4 Constraints in E-R Model

E-R model has a capability to enforce constraints. Two most important type of constraints in E- R 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.

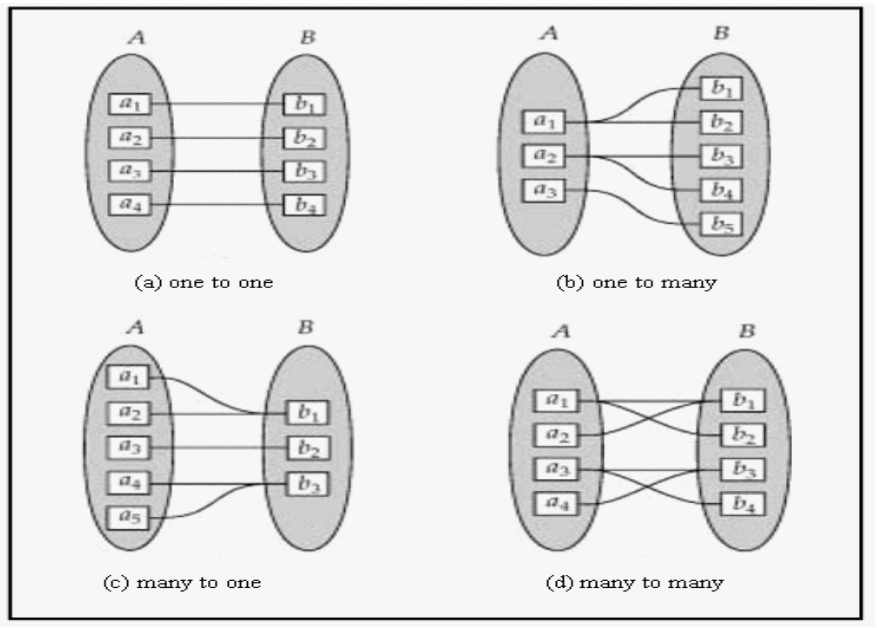


Figure: Mapping Cardinalities

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. We will discuss it in later section.

2.5 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.

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.

The above example of supekey shows that superkey may contains extraneous attributes. That is, if K is superkey then any superset of K is superkey.

Candidate key

The minimal superkey called candidate key. That is, candidate key is a superkey but its proper subset is not superkey. For example: customer_id is a candidate key in customer relation. Similarly account_id is a candidate key in account relation.

Primary key

In a relation, it is possible that we can choose distinct set of attributes as a candidate key. For example: in customer we can choose single attribute {custome_id} or set attributes {customer_name, customer_city} as candidate key. Candidate key chosen by database designer for particular relation known as primary key.

2.6 Primary Keys for Relationship Sets

Suppose R is a relationship set involving entity sets $E_1, E_2, \ldots E_n$. Lets consider primary-key(E_i) is a set of attributes that form primary key in each entity sets E_i. (i=0,1, ..n). Then set of attributes

$$\text{Primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \ldots \cup \text{primary-key}(E_n)$$

describes individual relationship in relationship set R.

Since relationship set may also consist another attributes (e.g. descriptive attributes) so assume relationship set consist attributes $\{a_1,a_2, \ldots a_n\}$ then set of attributes

$$\text{Primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \ldots \cup \text{primary-key}(E_n) \cup \{a_1,a_2, \ldots a_n\}$$

also describes individual relationship in relationship set R.

In both of the above case, the set of attributes

$$\text{Primary-key}(E_1) \cup \text{primary-key}(E_2) \cup \ldots \cup \text{primary-key}(E_n)$$

form super key in relationship set R.

This indicates that primary keys of entity sets those involves in relationship set form super key in relationship set. For example: in relationship set depositor, {customer_id, account_number} are attributes and each attribute customer_id and account_number are primary key in relation customer and account respectively. If relationship set consist declarative attribute say access_date then it can also consider as attributes of relationship set depositor.

The selection of primary key in relationship set depends up on mapping cardinality of that relationship set. To illustrate this, let us consider entity sets customer and account, and relationship set depositor with declarative attribute access_date. Suppose that relationship set is many to many. Then primary key of depositor relationship set is combination of primary keys of customer and account. If customer can have only one account, that is, if relationship set is many to one from customer to account the primary key of depositor is simply primary key of customer (i.e customer_id). Similarly, if each account is own by at most one customer, that is, relationship set is many to one from account to customer then primary key of depositor is simply the primary key of account (i.e. account_id). If relationship set is one to one then we can choose either customer_id or account_id as primary key.

For non-binary relationship set where no cardinality constraints are define then superkey is only one possible candidate key. So it needs to be chosen as a primary key.

2.7 Entity Relationship Diagram (ERD)

We already discuss that E-R Diagram has a capability to describes overall logical structure of database graphically in brief. In this section we discuss E-R diagram capabilities in detail

Major components of E-R diagram are:

Rectangles: representing entity sets.
Ellipses: representing attributes.
Diamonds: representing relationship sets.
Lines: linking attributes to entity sets and entity sets to relationship sets.
Double ellipses: represents multivalued attributes.
Dashed ellipses: represents derived attributes
Double lines: represents total participation of entity in relationship sets **Double rectangles**: represents weak entity sets (discuss in later section) **Underline**: indicates primary key attributes

Representation of relationship set in E-R Diagram

The relationship set borrower having two entity sets customer and loan can be express as

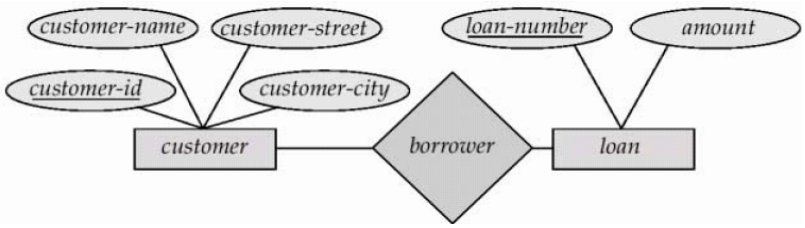


Figure: borrower relationship set in E-R Diagram

Representation of mapping cardinalities of relationship set in E-R Diagram

The directed lines in E-R diagram are used to specify mapping cardinalities of relationship sets. The directed line (→) tells "one," and an undirected line (—) tells "many" between the relationship set and the entity set.

One-to-one relationship representation

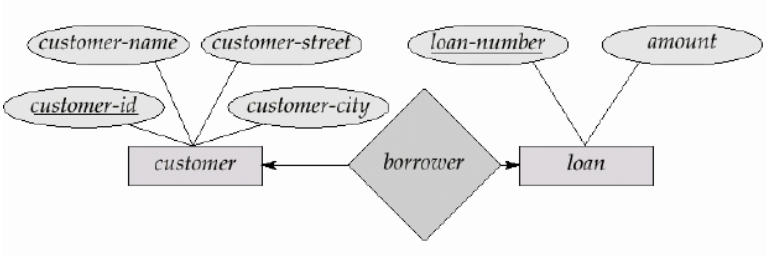


Figure: one to one relationship set in E-R Diagram

This indicates a customer is associated with at most one loan via the relationship borrower and a loan is associated with at most one customer via borrower.

One to many relationship representation

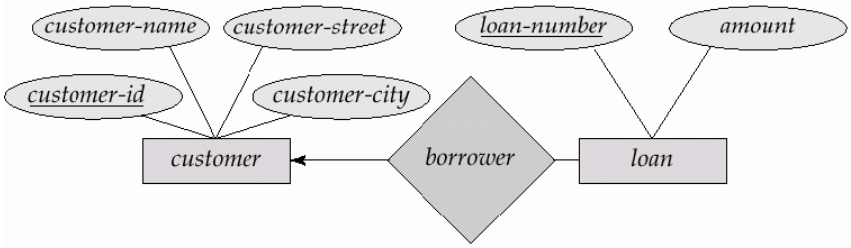


Figure: one to many relationship set in E-R Diagram

This indicates a loan is associated with at most one customer via borrower and a customer is associated with several (including 0) loans via borrower.

Many to one representation

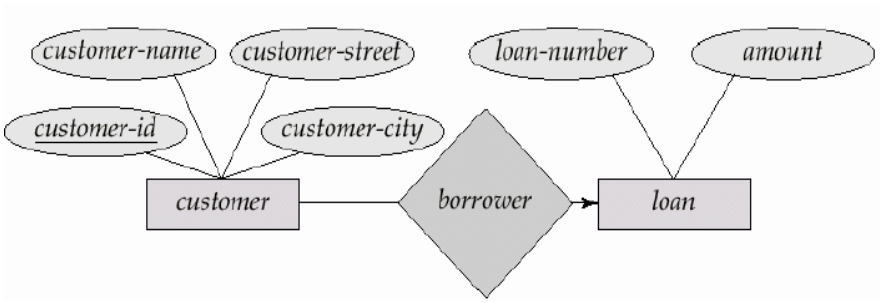


Figure: many to one relationship set in E-R Diagram

This indicates that a loan is associated with several (including 0) customers via borrower but a customer is associated with at most one loan via borrower.

Many to many relationship representation

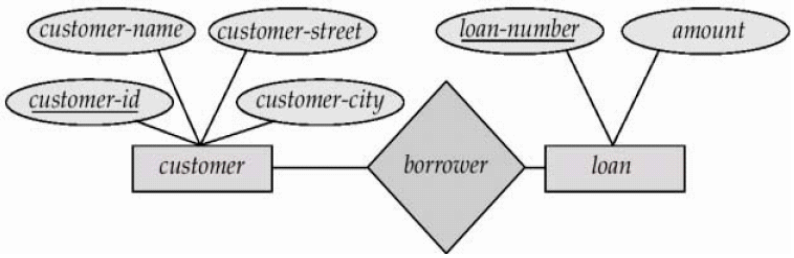


Figure: many to many relationship set in E-R Diagram

This indicates a customer is associated with several (including 0) loans via borrower and a loan is associated with several (including 0) customers via borrower

Representation of Composite, Multivalued, and Derived Attributes in E-R Diagram

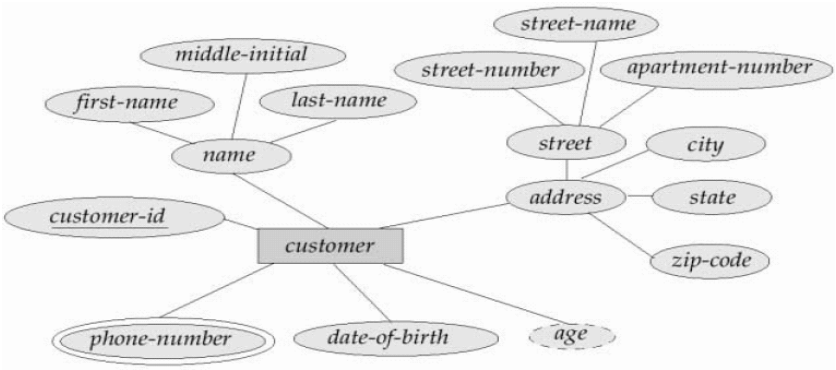


Figure: E-R Diagram with Composite, Multivalued, and Derived Attributes

Here attribute "name" is composite attribute, "phone_number" is multivalued attribute and "age" is derived attribute.

Representation of roles (recursive relationship set) in E-R Diagram

Roles in E-R diagrams are indicated by labeling the lines that connect diamonds to rectangles. Role labels are optional, and are used to clarify semantics of the relationship

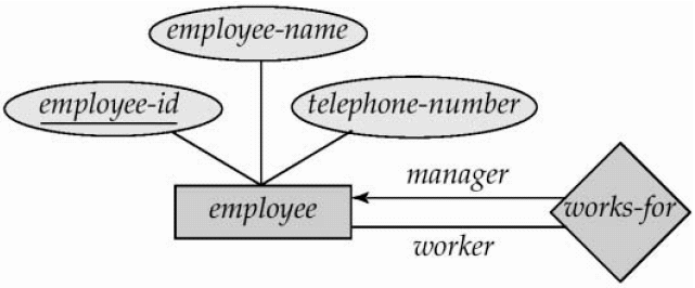


Figure: Roles in E-R Diagram

Here relationship set "work-for" is recursive relationship set and the labels "manager" and "worker" are roles.

Representation of non binary relationship Set in E-R Diagram

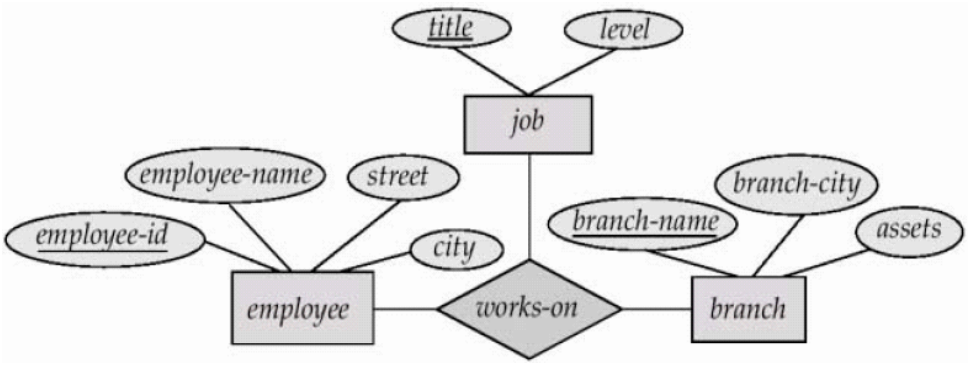


Figure: ternary relationship set in E-R Diagram

In ternary (or higher degree) relationship set there are one arrow to indicate a cardinality constraint. For example: an arrow from works-on to job indicates each employee works on at most one job at any branch. If there is more than one arrow, there are two ways of defining the meaning. Suppose a ternary relationship R between A, B and C with arrows to B and C this can be interoperate as

- each entity in A is associated with a unique entity in B and C or
- each pair of entities (A, B) is associated with a unique entity in C, and each pair of entities (A, C) is associated with a unique entity in B.

Representation of participation constraints of entity set in relationship set

In E-R diagram total participation of entity set in relationship set indicated by double line between that relationship set and entity set, single line indicates partial participation.

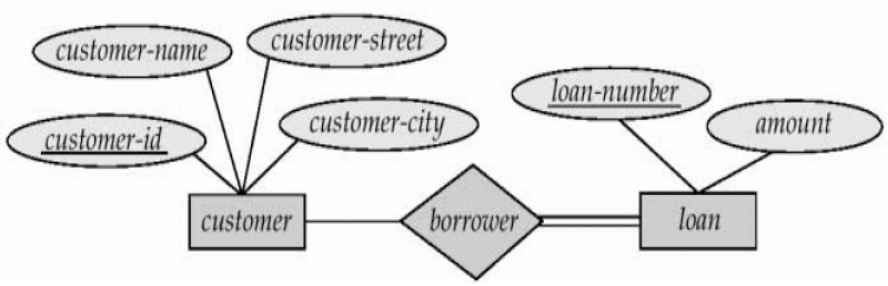


Figure: participation constraints in E-R Diagram

This indicates participation of *loan* in *borrower relationship is* total and participation of *customer* in *borrower relationship* is partial. That is, every loan must have a customer associated to it via borrower but all customers may not associate to loan.

E-R diagram can also specify more complex constraints. It can specify cardinality limits. That is, it can specify how many no. of times each entity may participates in relationships in relationship set. An edge between an entity set and binary relationship can have an associated minimum and maximum cardinality in the form l..h, where l is the minimum and h is the maximum cardinality. A minimum value 1 indicates total participation of the entity set in the relationship set. A maximum value 1 indicates that the entity participates in at most one relationship, while maximum value * indicates no limit. That is label 1..* on an edge is equivalent to double line.

Example:

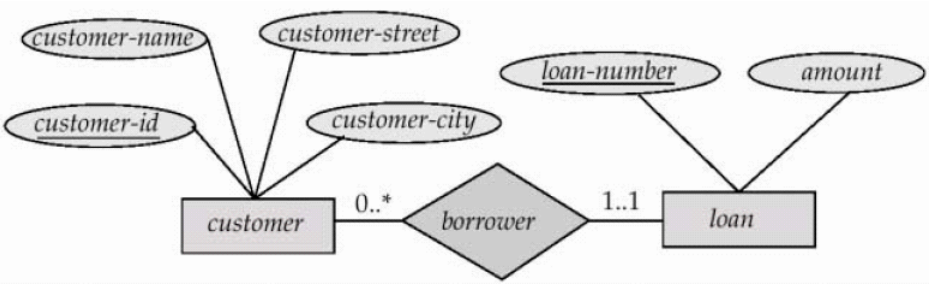


Figure: Cardinality limits on relationship sets

Here, edge between loan and borrower has cardinality limit 1..1. This indicates loan must have exactly one associated customer. The cardinality limit 0..* on the edge from customer to borrower indicates that customer can have zero or more loans. Thus the relationship borrower is one to many from customer to loan and further the participation of loan in borrower is total.

2.8. 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.

In our example, identifying entity set for weak entity set payment is loan, and a relationship loan-payment associates payment entities with their corresponding loan entities in identifying relationship.

Although, a weak entity set does not have a primary key, it contains attribute or set of attributes that distinguishes all entities of a weak entity set called *discriminator of weak entity*

set or *partial key*. In payment weak entity set payment_number is a discriminator since for each loan, payment number uniquely identifies one single payment for that loan.

The primary key of weak entity set can be compose by combining primary key of identifying entity set and the weak entity set's discriminator. For weak entity set payment primary key is {loan_number,paymenr_number}, where loan_number is primary key of identifying entity set loan, and payment_number is discriminator of weak entity set payment.

In E-R diagram, weak entity set represented by double rectangles. The discriminator of a weak entity set is represent by underline attribute with a dashed line and double outlined diamond represents identifying relationship.

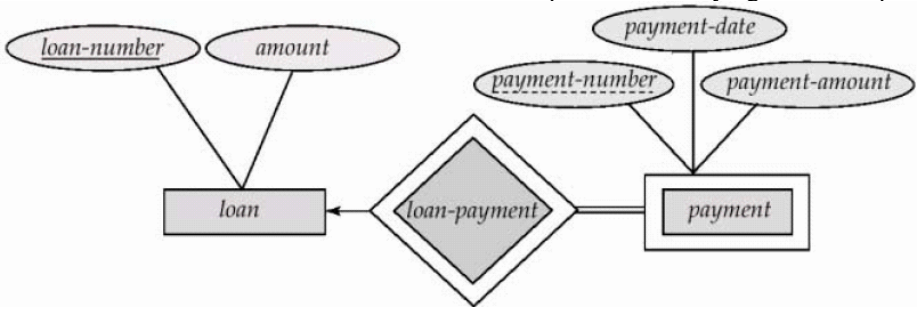


Figure: E-R diagram with weak entity set.

2.9 Extended E-R Features

2.9.1 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: emplouee_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.

2.9.2 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.

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.

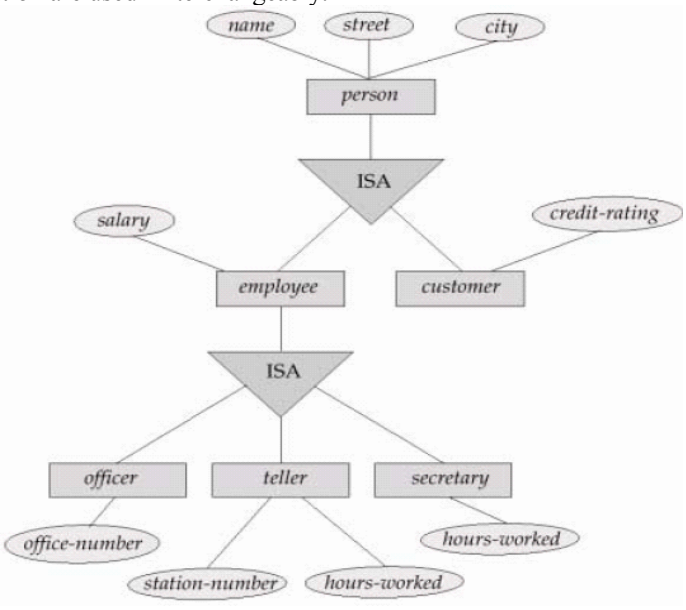


Figure: Specialization and generalization.

2.9.3 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.

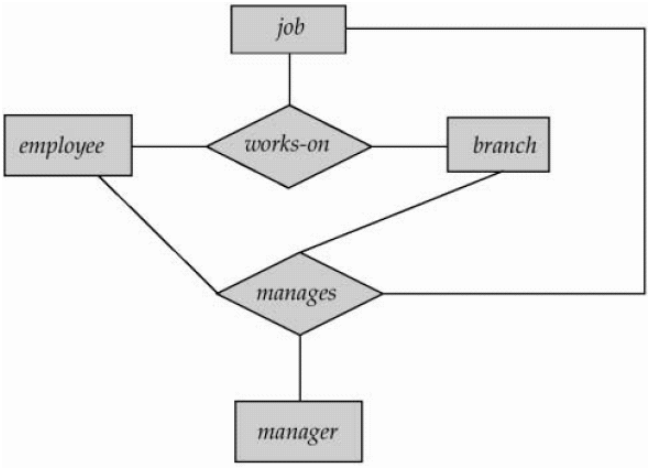
This constraint may specify as follows

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.

2.10 Aggregation

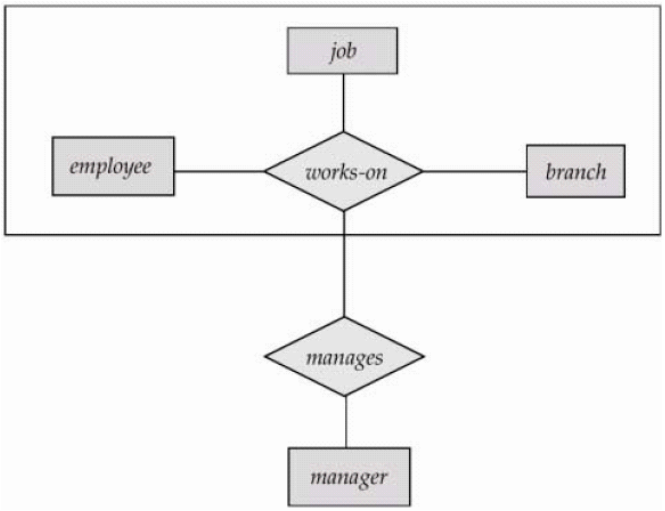
E-R model can not express relationship among relationship. To illustrate this, let us consider quaternary relationship manages among employee, branch, job and manager. Its main job is to record managers who manages particular job/task perform by particular employee at particular branch.



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

2.11 Design of an E-R Database Schema

E-R data model provides much flexibility in designing database schema. Database designer can select wide range of alternatives. Database designer must make following decisions:

- Whether to use an attribute or entity set to represent an object.
- Whether a real-world concept is best to express by an entity set or a relationship set
- Whether to use a ternary relationship or pair of binary relationship.
- Whether to use strong or weak entity set.
- Whether to use generalization or specialization. Generalization/Specialization provides modularity in the design.
- Whether aggregation is better to use or not.

2.11.1 Database Design Phase

(a) user specification requirements

In the initial phase of database design, database designer need to characterize what data needs for database users and how the database is structured to fulfill these requirements. Database designer needs to interact with domain experts and users to carry out this task.

(b) Conceptual design

In this phase, database designer need to choose appropriate data model to translate the requirement into conceptual schema of the database. The conceptual design describes detail overview of enterprise. E-R model can be use to develop conceptual schema. In terms of E- R model, conceptual schema specifies all entity sets, relationship sets, attributes, and mapping constraints. Conceptual schema is also able to describe functional requirements of the enterprise. In functional requirements user can describes kind of operations that will be perform on data.

(c) Logical design phase:

In this phase database designer need to maps the high level conceptual schema onto the implementation data model of the database system.

(d) Physical design phase

In this phase, database designer specifies physical features of the database. These features include form of file organization and storage structure.

2.12 Reduction of an E-R Schema to Tables

We can represent the E-R database schema by a set of tables. Each entity sets and each relationship sets in E-R schema can be represents by their corresponding tables. Each attributes of entity sets and relationship sets are map as columns of their corresponding tables. Similarly constraints specified in E-R diagram such as primary key, cardinality constraints etc are mapped to tables generated from E-R diagram. In fact, representing E-R schema into tables is converting E-R model of database into relational model.

2.12.1 Tabular representation of Strong Entity Sets

Strong entity set represent by a table with all attributes of its. Let E be a strong entity set with attributes $a_1, a_2, \ldots, a_n$ then it represent by a table E with n distinct columns, each of which correspond to one of the attributes of strong entity set E. Each row in this table corresponds to entity of entity set E.

For example: entity set account with account_number and balance can be represented by table "account" as

account_number	balance
A1	200
A2	300
A3	500
A4	500

Figure: The account table

Suppose D_1 denotes the set of all account numbers, and D_2 denotes all balances. Each row of the account table contains 2-tuple (v_1,v_2), where v_1 is loan number (i.e. v_1 is in set D_1) and v_2 is balance (i.e. v_2 is in set D_2). In general, we can say that account table contains only a subset of all possible rows. We can refer set of all possible rows of account table as Cartesian product of D_1 and D_2, denoted by $D_1 \quad D_2$.

In general, table contains n columns, then set of all possible rows in that table can express by Cartesian product of $D_1,D_2, \ldots,D_n$, denoted by $D_1 \quad D_2 \quad \ldots \quad D_{n-1} \quad D_n$.

2.12.2 Tabular representation of Weak Entity Sets

Let A be a weak entity set with attributes $a_1,a_2, \ldots a_m$. Let B be a strong entity set on which A depend on. Let primary key of B consist attributes $b_1,b_2, \ldots,b_n$. Now weak entity set A can be represent by table A with attributes $\{a1, a2, \ldots a_m\} \cup \{b_1,b_2, \ldots,b_n\}$.

Example: Let us consider weak entity set payment with attributes; payment_number, payment_date, and payment_amount. Consider loan is a strong entity set on which weak entity set payment depends on and suppose that loan_number is a primary key of loan entity set. Now weak entity set payment can be represent with attributes: loan_number, payment_number,payment_date, and payment_amount.

loan_number	Payment_number	Payment_date	balance
L12	5	02-04-2005	100
L15	7	12-04-2005	250
L15	8	15-04-2005	150
L19	9	03-05-2005	500

Figure: The payment table

2.12.3 Tabular representation of Relationship Sets

Let R be a relationship set with set of attributes $a_1,a_2, \ldots a_m$ formed by union of primary keys of each entity sets that participates in R. Assume that R consist descriptive attributes $b_1,b_2, \ldots,b_n$. Now, relationship set R represent by table R with attributes $\{ a_1, a_2, \ldots a_m \} \cup \{b_1,b_2, \ldots,b_n\}$.

Example; consider a relationship set borrower that involves two entity sets

- customer with primary key customer_id
- loan with primary key loan_number

Since relationship set does not consist any descriptive attribute, relationship set borrower can represent by table borrower with attributes customer_id and loan_number.

customer_id	**loan_ number**
C1	L1
C2	L2
C3	L3
C4	L4

Figure: The borrower relationship set