1. Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.

2. Design an E-R diagram for keeping track of the exploits of your favorite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modeled as derived attributes. 5. Extend the E-R diagram of the previous question to track the same information for all teams in a league.

3. A large bank operates several divisions. Information Technology (IT) is operated as one of these divisions. Within the IT division are many departments that are managed by one manager, and all IT employees belong to one of these departments. The IT division assigns its employees to one or more ongoing projects in the bank. A project may be planned, but not have any employees assigned to it for several months. Each project will have a single employee assigned who acts as a project leader.

4. A hardware store sells several home workshop products to the public (such as power saws and sanders). Each product has several different manufacturers who manufacture it and prices are different for products made by different manufacturers. Each time one or more products are sold to a customer, an invoice is created which lists the date, items purchased and their prices, and then the total purchase and tax amounts.

5. The Ministry of Transportation (MOT) supplies department keeps track of all the items (furniture and equipment such as a chair or printer) in the Ministry offices. There are several MOT buildings and each one is given a different name to identify it. Each item is assigned a unique ID when it is purchased. This ID is used to keep track of the item, which is assigned to a room within a building. Each room within a building is assigned to a department, and each department has a single employee as its manager.

6. A cooking club organizes several dinners for its members. The purpose of the club is to allow several members to get together and prepare a dinner for the other members. The club president maintains a database that plans each meal and tracks which members attends each dinner, and also keeps track of which members creates each dinner. Each dinner serves many members and any member is allowed to attend. Each dinner has an invitation. This invitation is mailed to each member. The invitation includes the date of the dinner and location. Each dinner is based on a single entrée and a single dessert. This entrée and dessert can be used again for other dinners.

7. ABC Consulting is a small-sized consulting firm in the IT industry. ABC's business is managing several Systems Development projects by assigning staff consultants to these projects as their skills are needed. Each employee is designated to have one primary skill, but there may be other employees with the same primary skill. A consultant may work on one or more projects, or may not yet be assigned to a project. The company charges for each project by billing each consultant's hours worked by the billing rate. The hourly billing rate is dependent on the employee's primary job skill.

Riyad

Introduction

In this tutorial you are required to design an Entity Relationship Model for a database to be used by the organisers of a poster exhibition in order to keep track of three phases in the exhibition: submission, selection and presentation. You should read the scenario carefully, making a note of all the candidate entities you think are involved, and of their attributes. Think about the relationships between the entities as you do this. The tutorial proceeds with a series of questions about your design, finishing with an ER diagram.

Scenario:

A Poster Exhibition The setting is that you are one organiser of a poster exhibition on "Global Problems of the 21st Century", and you must design a database to keep track of the administration of the exhibition. Three main phases are recognised for the exhibition: the submission, the selection and the presentation phase. These are further explained below.

Submission Phase: Graphic designers create posters for the exhibition to illustrate one of the chosen global problems. Relevant information on designers includes their name and their affiliation, i.e. the organization they work for. A poster has a title and is assigned an identification number, and it may be created by several graphic designers; although each individual designer may only be involved with one poster. Where a group of graphic designers create a poster, we distinguish between the main designer and the co-designers. In case of a single graphic designer, that person is considered to be the main designer of the poster. The main designer is always the point of contact, so should provide an email address.

Selection Phase: All posters created for this exhibition are judged by members of a jury. A judge is a graphic design expert with experience in communication for raising public awareness and for public benefit. Judge information that is of relevance to the organising committee includes the judge's name, their affiliation and email. Each poster is judged by three different judges. When judging a poster, a judge gives a decision: accept or reject. A poster is selected for the exhibition only if all three judges give an "accept" decision. Note that judges are not allowed to compete in "Global Problems of the 21st Century" themselves.

Presentation Phase: All selected posters are then presented in the exhibition by their main graphic designers. The poster presentation is allocated a stand and an exhibition session. Each exhibition session takes place at a specific date, and 4 session topics have been announced: human rights, environmental pollution, poverty, and war.

1. A college runs many classes. Each class may be taught by several teachers, and a teacher may teach several classes. A particular class always uses the same room. Because classes may meet at different times or on different evenings, it is possible for different classes to use the same room.
2. Each employee in an engineering company has at most one recognized skill, but a given skill may be possessed by several employees. An employee is able to operate a given machine-type (e.g., lathe, grinder) if he has one of several skills, but each skill is associated with the operation of only one machine type. Possession of a given skill (e.g., mechanic, electrician) allows an employee to maintain several machine-types, although maintenance of any given machine-type requires a specific skill (e.g., a lathe must be maintained by a mechanic).
3. Employees *work in departments; each department is managed by an* employee; a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.

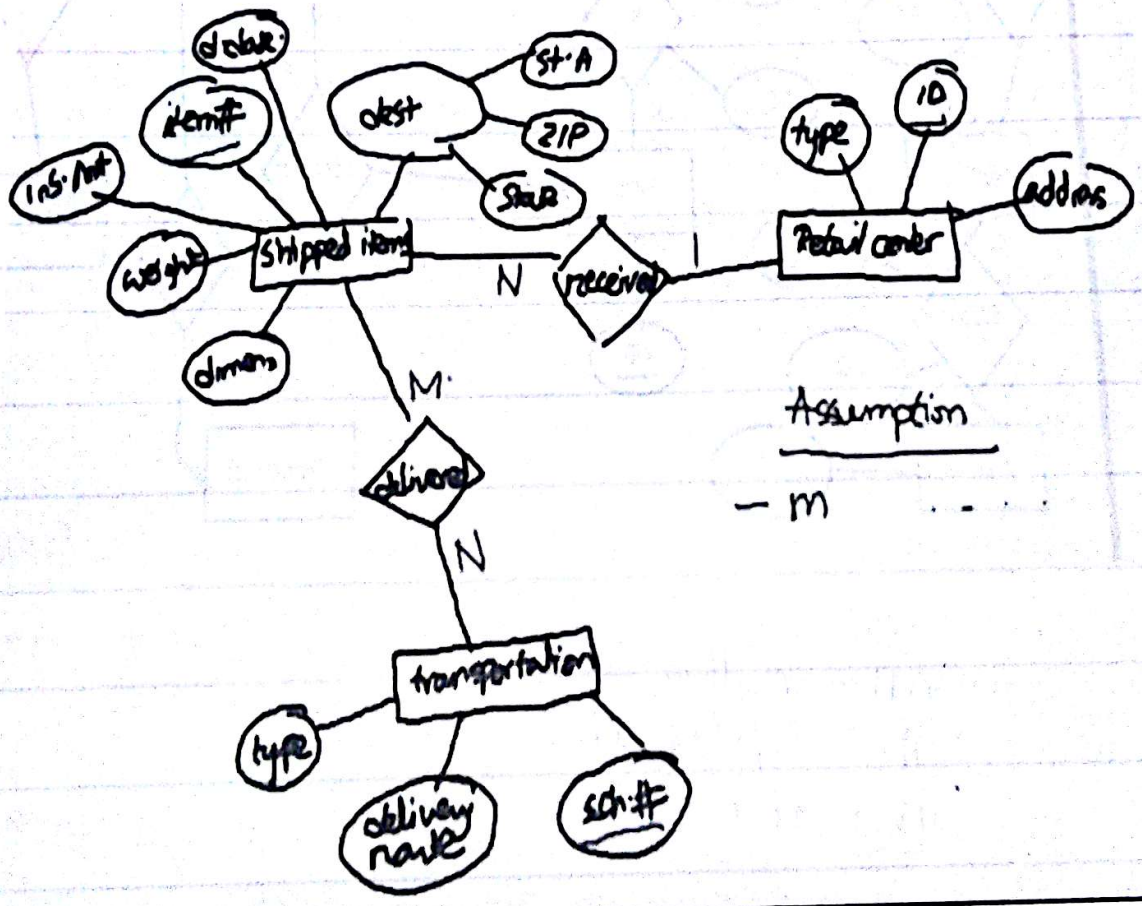
3. A Bioinformatics Application

- Patient: has a unique MSP number, a Patient name, a Date of Birth, a Tissue Type and an indicator denoting whether the tissue is cancerous or normal.
- A patient library associates a patient with multiple tags
- Each tag has a unique tag number and a unique nucleotide sequence.
- For each tag in the patient library, a count is given to record the number of times the tag occurs in the library. In general, the same tag can be associated with any number of patients.
- A tag may be mapped to a gene. Each gene has a unique gene name and a type.
- In general, multiple tags may be mapped to the same gene. However, two different genes cannot be mapped to the same tag.
- Finally, an article is identified by a unique article number and a journal name. An article may analyze multiple genes and a gene may be analyzed by multiple articles.

Let us construct an ER model for the above application.

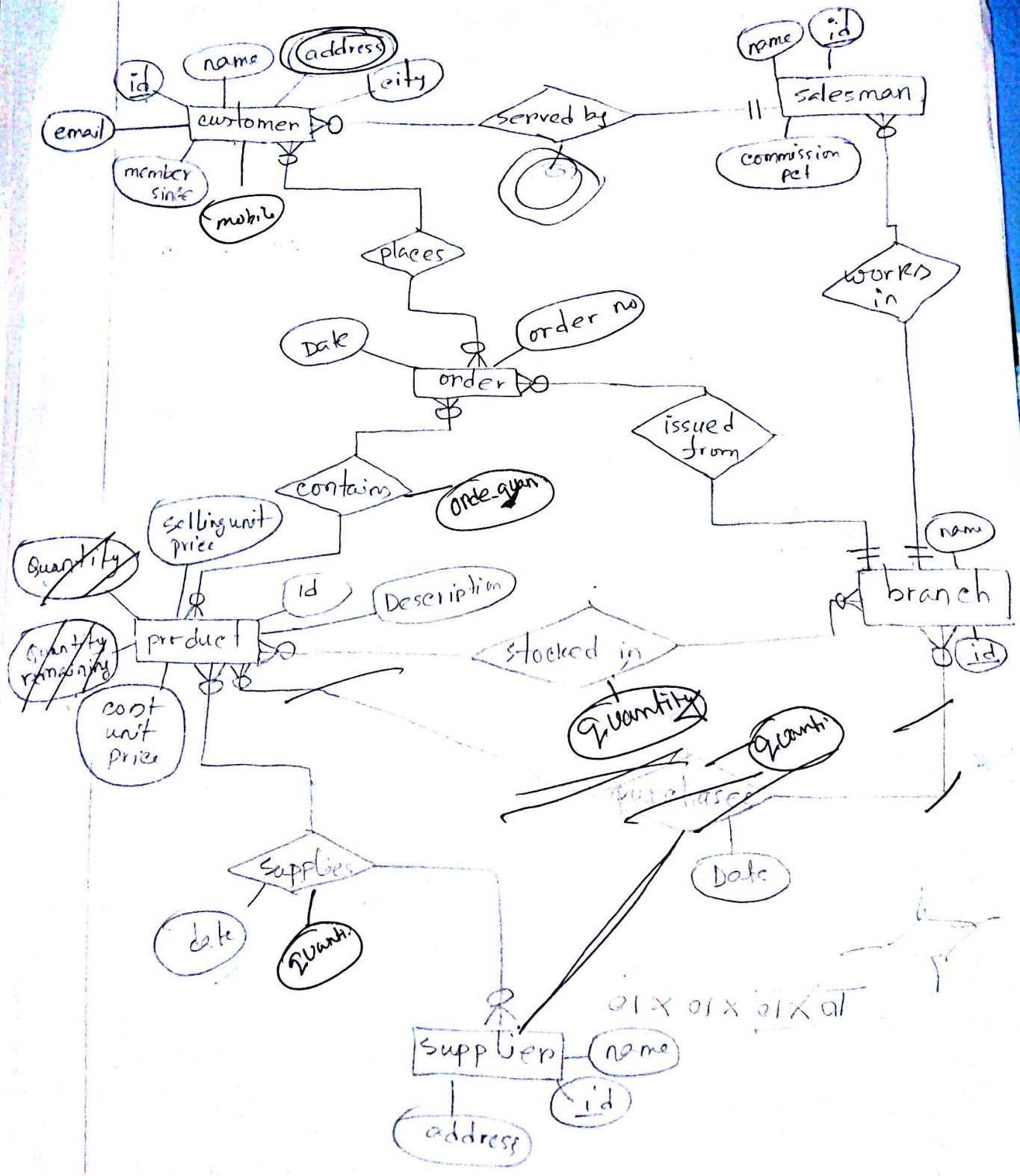
UPS prides itself on having up-to-date information on the processing and current location of each shipped item. To do this, UPS relies on a company-wide information system.

Shipped items are the heart of the UPS product tracking information system. Shipped items can be characterized by item number (unique), weight, dimensions, insurance amount, destination with it's street address, zip code and state, and final delivery date. Shipped items are received into the UPS system at a single retail center. Retail centers are characterized by their type, uniqueID, and address. Shipped items make their way to their destination via one or more standard UPS transportation events (i.e., flights, truck deliveries). These transportation events are characterized by a unique scheduleNumber, a type (e.g, flight, truck), and a deliveryRoute.



Riyad, Md. Ahsan Ferdous
15-29804-2

Saied, Ahmed Foyez
16-31496-1



Normalization Exercise 2

INVOICE

HILLTOP ANIMAL HOSPITAL
INVOICE # 987

DATE: JAN 13/2002

MR. RICHARD COOK
123 THIS STREET
MY CITY, ONTARIO
Z5Z 6G6

PET	PROCEDURE	AMOUNT
ROVER	RABIES VACCINATION	30.00
MORRIS	RABIES VACCINATION	24.00
	TOTAL	54.00
	TAX (8%)	4.32
	AMOUNT OWING	58.32

UNF:

1NF:

2NF:

3NF:

1 to M

1. **Normalize up to 3rd normal forms and give suitable name to each of the new relations.**

PROD_SUPP_LIST (Product_no, Product_name, Product_typ_code, Product_typ_name, Supplier_no, Supplier_name, product price, description)

2. Fields in the original data table will be as follows:
SalesOrderNo, Date, CustomerNo, CustomerName, CustomerAdd, ClerkNo, ClerkName, ItemNo, Description, Qty, UnitPrice → multivalued ? how ?
Think of this as the baseline – one large table. Normalize up to 3rd normal forms and give suitable name to each of the new relations.

Normalization

Normalization is a method for organizing data elements in a database into tables.

Normalization Avoids

- Duplication of Data – The same data is listed in multiple lines of the database
- Insert Anomaly – A record about an entity cannot be inserted into the table without first inserting information about another entity – Cannot enter a customer without a sales order
- Delete Anomaly – A record cannot be deleted without deleting a record about a related entity. Cannot delete a sales order without deleting all of the customer's information.
- Update Anomaly – Cannot update information without changing information in many places. To update customer information, it must be updated for each sales order the customer has placed

Normalization is a three stage process – After the first stage, the data is said to be in first normal form, after the second, it is in second normal form, after the third, it is in third normal form

Before Normalization

1. Begin with a list of all of the fields that must appear in the database. Think of this as one big table.
2. Do not include computed fields
3. One place to begin getting this information is from a printed document used by the system.
4. Additional attributes besides those for the entities described on the document can be added to the database.

Before Normalization – Example

See Sales Order from below:

Sales Order

Fiction Company
202 N. Main
Mahattan, KS 66502

CustomerNumber:	*1001*	**Sales Order Number:**	405
Customer Name:	*ABC Company*	**Sales Order Date:**	2/1/2000
Customer Address:	*100 Points*	**Clerk Number:**	*210*
	Manhattan, KS 66502	**Clerk Name:**	*Martin Lawrence*

Item Ordered	Description	Quantity	Unit Price	Total
800	*widgit small*	*40*	*60.00*	*2,400.00*
801	*tingimajigger*	*20*	*20.00*	*400.00*
805	*thingibob*	*10*	*100.00*	*1,000.00*
			Order Total	*3,800.00*

Fields in the original data table will be as follows:

SalesOrderNo, Date, CustomerNo, CustomerName, CustomerAdd, ClerkNo, ClerkName, ItemNo, Description, Qty, UnitPrice

Think of this as the baseline – one large table

Normalization: First Normal Form

- Separate Repeating Groups into New Tables.
- ***Repeating Groups*** Fields that may be repeated several times for one document/entity
- Create a new table containing the repeating data
- The primary key of the new table (repeating group) is always a composite key. Usually document number and a field uniquely describing the repeating line, like an item number.

First Normal Form Example

The new table is as follows:

SalesOrderNo, ItemNo, Description, Qty, UnitPrice

The repeating fields will be removed from the original data table, leaving the following.

SalesOrderNo, Date, CustomerNo, CustomerName, CustomerAdd, ClerkNo, ClerkName

These two tables are a database in first normal form

What if we did not Normalize the Database to First Normal Form?

Repetition of Data – SO Header data repeated for every line in sales order.

Normalization: Second Normal Form

- Remove Partial Dependencies.
- ***Functional Dependency*** The value of one attribute in a table is determined entirely by the value of another.
- ***Partial Dependency*** A type of functional dependency where an attribute is functionally dependent on only part of the primary key (primary key must be a composite key).
- Create separate table with the functionally dependent data and the part of the key on which it depends. Tables created at this step will usually contain descriptions of resources.

Second Normal Form Example

The new table will contain the following fields:

ItemNo, Description

All of these fields except the primary key will be removed from the original table. The primary key will be left in the original table to allow linking of data:

SalesOrderNo, ItemNo, Qty, UnitPrice

Never treat price as dependent on item. Price may be different for different sales orders (discounts, special customers, etc.)

Along with the unchanged table below, these tables make up a database in second normal form:

SalesOrderNo, Date, CustomerNo, CustomerName, CustomerAdd, ClerkNo, ClerkName

What if we did not Normalize the Database to Second Normal Form?

- Repetition of Data – Description would appear every time we had an order for the item
- Delete Anomalies – All information about inventory items is stored in the SalesOrderDetail table. Delete a sales order, delete the item.

- Insert Anomalies – To insert an inventory item, must insert sales order.
- Update Anomalies – To change the description, must change it on every SO.

Normalization: Third Normal Form

- Remove transitive dependencies.
- ***Transitive Dependency*** A type of functional dependency where an attribute is functionally dependent on an attribute other than the primary key. Thus its value is only indirectly determined by the primary key.
- Create a separate table containing the attribute and the fields that are functionally dependent on it. Tables created at this step will usually contain descriptions of either resources or agents. Keep a copy of the key attribute in the original file.

Third Normal Form Example

The new tables would be:

CustomerNo, CustomerName, CustomerAdd
ClerkNo, ClerkName

All of these fields except the primary key will be removed from the original table. The primary key will be left in the original table to allow linking of data as follows:

SalesOrderNo, Date, CustomerNo, ClerkNo

Together with the unchanged tables below, these tables make up the database in third normal form.

ItemNo, Description
SalesOrderNo, ItemNo, Qty, UnitPrice } keeping unchanged?

What if we did not Normalize the Database to Third Normal Form?

- Repetition of Data – Detail for Cust/Clerk would appear on every SO
- Delete Anomalies – Delete a sales order, delete the customer/clerk
- Insert Anomalies – To insert a customer/clerk, must insert sales order.
- Update Anomalies – To change the name/address, etc, must change it on every SO.

Completed Tables in Third Normal Form

Customers: CustomerNo, CustomerName, CustomerAdd ✓
Clerks: ClerkNo, ClerkName ✓
Inventory Items: *ItemNo*, Description ✓
Sales Orders: SalesOrderNo, Date, CustomerNo, ClerkNo ✓ composit P-key
SalesOrderDetail: SalesOrderNo, ItemNo, Qty, UnitPrice ... en 3NF? possible?

insert into table_name (colnm name ... ..)

values (value1, value 2).

insert into table_nam values (val1, v2. ..)

update column values

update table_name.

Set colname = "new value"

where conditim

Delete — it delete row.
if no condition given
then delete all records
or rows.

Delete ~~table_name~~ from table_nam
~~where~~ where conditon.

command

union
intersect
minus

Add columns in a table

```
ALTER TABLE table_name
    ADD colmn_name colm_definition.
```

Modify column in a table

```
ALTER TABLE table_name
    MODIFY colm_name colm_type
```

Drop column in table

```
ALTER TABLE table_name
    DROP COLUMN colmn_name.
```

Rename column name

```
ALTER TABLE table_name
    Rename COLUMN old_colname
        TO new_colnam
```

Rename table

~~RENAME~~

```
ALTER TABLE table_name
    Rename to New tabb name
```

Alter foreign key

```
ALTER TABLE
    ADD coln definition

ALTER TABLE riyad
    ADD ~~coln~~ f_id number(10)
        constraint ~~fk~~ riyad_f_id
            references f ahsan(id)
```

1. Consider a database with the following schema:

Person (name, age, gender)	name is a key
Frequents (name, pizzeria)	(name, pizzeria) is a key
Eats (name, pizza)	(name, pizza) is a key
Serves (pizzeria, pizza, price)	(pizzeria, pizza) is a key

Write relational algebra expressions for the following nine queries. (Warning: some of the later queries are a bit challenging.)

a. Find all pizzerias frequented by at least one person under the age of 18.

b. Find the names of all females who eat either mushroom or pepperoni pizza (or both).

c. Find the names of all females who eat both mushroom and pepperoni pizza.

d. Find all pizzerias that serve at least one pizza that Amy eats for less than $10.00.

e. Find all pizzerias that are frequented by only females or only males.

f. For each person, find all pizzas the person eats that are not served by any pizzeria the person frequents. Return all such person (name) / pizza pairs.

g. Find the names of all people who frequent only pizzerias serving at least one pizza they eat.

h. Find the names of all people who frequent every pizzeria serving at least one pizza they eat.

i. Find the pizzeria serving the cheapest pepperoni pizza. In the case of ties, return all of the cheapest-pepperoni pizzerias.

Chapter 2: Relational Model

Database System Concepts, 5th Ed.

Relational Algebra

- Procedural language
- Six basic operators
  - select: σ
  - project: Π
  - union: ∪
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

tuples = columns.

Select Operation – Example

- Relation *r*

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Select Operation

- Notation: $\sigma_p(r)$
- *p* is called the selection predicate
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

Where *p* is a formula in propositional calculus consisting of terms connected by : $\wedge$ **(and)**, $\vee$ **(or)**, $\neg$ **(not)**
Each term is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: $=, \neq, >, \geq, <, \leq$

- Example of selection:

$$\sigma_{branch\_name="Perryridge"}(account)$$

Figure 2.9

Result of $\sigma_{branch\_name = "Perryridge"}(loan)$

loan_number	branch_name	amount
L-15	Perryridge	1500
L-16	Perryridge	1300

Project Operation – Exam

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{AC}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Project Operation

- Notation:

$$\Pi_{A_1, A_2, K, A_k}(r)$$

where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *branch_name* attribute of *account*

$$\Pi_{account\_number,\ balance}(account)$$

Figure 2.10:
Loan number and the amount of the loan

loan_number	amount
L-11	900
L-14	1500
L-15	1500
L-16	1300
L-17	1000
L-23	2000
L-93	500

Composition of Relational Operations

- Find the customer who live in Harrison
  - $\Pi_{customer\_name}(\sigma_{customer\_city="Harrison"}(customer))$
- Notice that instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation

Union Operation – Example

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cup s$:

A	B
α	1
α	2
β	1
β	3

Union Operation

- Notation: $r \cup s$
- Defined as:

$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- For $r \cup s$ to be valid.
  1. *r*, *s* must have the *same* arity (same number of attributes)
  2. The attribute domains must be compatible (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)
- Example: to find all customers with either an account or a loan

$\Pi_{customer\_name}(depositor) \cup \Pi_{customer\_name}(borrower)$

Set Difference Operation – Example

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r - s$:

A	B
α	1
β	1

Set Difference Operation

- Notation $r - s$
- Defined as:

$$r - s = \{t \mid t \in r \text{ and } t \notin s\}$$

- Set differences must be taken between compatible relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible

Cartesian-Product Operation – Ex

- Relations r, s:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- $r \times s$:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

$$r \times s = \{t\, q \mid t \in r \text{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Rename Operation

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_x(E)$$

returns the expression E under the name X

- If a relational-algebra expression E has arity n, then

$$\rho_{x(A_1,A_2,\ldots,A_n)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

Banking Example

branch (branch_name, branch_city, assets)

customer (customer_name, customer_street, customer_city)

account (account_number, branch_name, balance)

loan (loan_number, branch_name, amount)

depositor (customer_name, account_number)

borrower (customer_name, loan_number)

Example Queries

- Find all loans of over $1200

$$\sigma_{amount > 1200}(loan)$$

- Find the loan number for each loan of an amount greater than $1200

$$\Pi_{loan\_number}(\sigma_{amount > 1200}(loan))$$

- Find the names of all customers who have a loan, an account, or both, from the bank

$$\Pi_{customer\_name}(borrower) \cup \Pi_{customer\_name}(depositor)$$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.

$$\Pi_{customer\_name}(\sigma_{branch\_name="Perryridge"}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

- Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

$$\Pi_{customer\_name}(\sigma_{branch\_name = "Perryridge"}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan))) - \Pi_{customer\_name}(depositor)$$

Example Queries

- Find the names of all customers who have a loan at the Perryridge branch.
  - Query 1

$$\Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(\sigma_{borrower.loan\_number = loan.loan\_number}(borrower \times loan)))$$

  - Query 2

$$\Pi_{customer\_name}(\sigma_{loan.loan\_number = borrower.loan\_number}((\sigma_{branch\_name = \text{“Perryridge”}}(loan)) \times borrower))$$

Example Queries

- Find the largest account balance
  - Strategy:
    - Find those balances that are *not* the largest
      - Rename *account* relation as *d* so that we can compare each account balance with all others
    - Use set difference to find those account balances that were *not* found in the earlier step.
  - The query is:

$$\Pi_{balance}(account) - \Pi_{account.balance}(\sigma_{account.balance < d.balance}(account \times \rho_{d}(account)))$$

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p(E_1)$, P is a predicate on attributes in E_1
  - $\Pi_s(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x(E_1)$, x is the new name for the result of E_1

Additional Operations

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

- Set intersection
- Natural join
- Division
- Assignment

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

Set-Intersection Operation – Example

- Relation r, s:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Natural-Join Operation

- Notation: $r \bowtie s$
- Let r and s be relations on schemas R and S respectively.
  Then, $r \bowtie s$ is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s
- Example:
  $R = (A, B, C, D)$
  $S = (E, B, D)$
  - Result schema = (A, B, C, D, E)
  - $r \bowtie s$ is defined as:
    $\Pi_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \wedge r.D = s.D} (r \times s))$

Natural Join Operation – Example

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ϵ

s

- $r \bowtie s$

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

Division Operation

- Notation: $r \div s$
- Suited to queries that include the phrase "for all".
- Let r and s be relations on schemas R and S respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of $r \div s$ is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

  $$r \div s = \{ t \mid t \in \Pi_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$

  Where tu means the concatenation of tuples t and u to produce a single tuple

Examples of Division A/B

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

A

pno
p2

B1

sno
s1
s2
s3
s4

A/B1

pno
p2
p4

B2

sno
s1
s4

A/B2

pno
p1
p2
p4

B3

sno
s1

A/B3

Bank Example Queries

- Find the names of all customers who have a loan and an account at bank.

  $$\Pi_{customer\_name}(borrower) \cap \Pi_{customer\_name}(depositor)$$

- Find the name of all customers who have a loan at the bank and the loan amount

  $$\Pi_{customer\_name,\ loan\_number,\ amount}(borrower \bowtie loan)$$

Bank Example Queries

- Find all customers who have an account from at least the "Downtown" and the Uptown" branches.
  - Query 1

  $$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Downtown"}}(depositor \bowtie account)) \cap$$

  $$\Pi_{customer\_name}(\sigma_{branch\_name = \text{"Uptown"}}(depositor \bowtie account))$$

Extended Relational-Algebra-Operations

- Generalized Projection
- Aggregate Functions
- Outer Join

Generalized Projection

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, ..., F_n are are arithmetic expressions involving constants and attributes in the schema of E.
- Given relation *credit_info(customer_name, limit, credit_balance)*, find how much more each person can spend:

$$\Pi_{customer\_name,\ limit - credit\_balance}(credit\_info)$$

Aggregate Functions and Operations

- **Aggregation function** takes a collection of values and returns a single value as a result.

  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$G_1, G_2, \ldots, G_n \mathcal{G}_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)}(E)$$

E is any relational-algebra expression

- $G_1, G_2 \ldots, G_n$ is a list of attributes on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name

Aggregate Operation – Example

- Relation r:

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{sum(c)}(r)$

sum(c)
27

Aggregate Operation – Example

- Relation *account* grouped by *branch-name*:

branch_name	account_number	balance
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$branch\_name\ \mathcal{G}\ \mathbf{sum}(balance)\ (account)$

branch_name	sum(balance)
Perryridge	1300
Brighton	1500
Redwood	700

Aggregate Functions (Cont.)

- Result of aggregation does not have a name
  - Can use rename operation to give it a name
  - For convenience, we permit renaming as part of aggregate operation

$branch\_name\ \mathcal{G}\ \mathbf{sum}(balance)\ \mathbf{as}\ sum\_balance\ (account)$

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.
    - We shall study precise meaning of comparisons with nulls later

Outer Join – Example

- Relation *loan*

loan_number	branch_name	amount
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

customer_name	loan_number
Jones	L-170
Smith	L-230
Hayes	L-155

Outer Join – Example

- Join

  loan ⋈ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- Left Outer Join

  loan ⟕ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

Outer Join – Example

- Right Outer Join

  loan ⟖ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

- Full Outer Join

  loan ⟗ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

Null Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null*.
- Aggregate functions simply ignore null values (as in SQL)
- For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

Null Values

- Comparisons with null values return the special truth value: *unknown*
  - If *false* was used instead of *unknown*, then *not* (A < 5) would not be equivalent to A >= 5
- Three-valued logic using the truth value *unknown*:
  - OR: (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown*) = *unknown*
  - AND: (*true* **and** *unknown*) = *unknown*,
    (*false* **and** *unknown*) = *false*,
    (*unknown* **and** *unknown*) = *unknown*
  - NOT: (**not** *unknown*) = *unknown*
  - In SQL "*P* **is unknown**" evaluates to true if predicate *P* evaluates to *unknown*
- Result of select predicate is treated as *false* if it evaluates to *unknown*

CHAPTER

6 E-R Diagrams

An **entity-relationship diagram** (E-R diagram) is a graphical representation of data for an organisation at conceptual level.

Keypoints

An E-R diagram is a graphical representation of data for an organisation at conceptual level.

E-R diagrams are frequently used by database designers to build *data models* in the process of database development (see Conceptual Data Modeling in 5.2). A well-drawn E-R diagram can be transformed into relational database tables easily: *Entities* will be mapped into *tables* and *attributes* will be mapped into *fields*.

There are some software tools that help people produce E-R diagrams and generate *database schema* (see Chapter 7) automatically.

The basic constructs of an E-R diagram are *entities*, *attributes* and *relationship*, with symbols shown in Fig.6.1.

Meaning	Symbol
Entity	Entity
Attribute	Attribute
Key Attribute	Key Attribute
Relationship	Relationship

Fig.6.1 Symbols used in E-R diagrams

6.1 RELATIONSHIPS

Recall that an *entity* is a person, place, device, event or concept and an *attribute* is a property of an entity class. The requirements for an entity are

1. one or more attributes
2. many possible distinct instances.

A relationship is an association between two entities based on a *key attribute*.

Relationship is a glue that holds together the various components of an E-R diagram.

Keypoints

An entity has one or more attributes, andmany possible distinct instances.

A relationship is an association between two entities based on a key attribute.

Note

Do not confuse *relationship* with *relation*: A relation is a table. A relationship links up two tables.

A. Identifying Entities and Relationships

In this book, an entity will be represented by single *noun* (e.g. STUDENT, CLUB, CERTIFICATE, etc.) and a relationship will be represented by a *verb* in present tense (e.g. Buys, Attends, Borrows etc.). Fig.6.2 shows the entities and relationships identified for some daily life examples. In these examples, some entities and relationships are not obvious, and you need to filter out unwanted information.

Example	Entities	Relationship
1. An employee works in a department.	EMPLOYEE, DEPARTMENT	Works_in
2. A student studies English, Chinese, Mathematics, Computer or Economics.	STUDENT, SUBJECT	Studies
3. An employee has three children.	EMPLOYEE, CHILDREN	Has
4. Some teachers are assigned a car park.	TEACHER, CAR_PARK	Is_assigned
5. An event in the sports day of a school will be cancelled if there are no participants.	STUDENT, EVENT	Participates
6. A company sells ten products. Customers can order products by phone.	CUSTOMER, PRODUCT	Orders
7. A teacher calculates the average mark of each student. The marks are recorded in report cards which are sent to the parents.	STUDENT, REPORT_CARD	Has

Fig.6.2 Some daily-life examples of entities and relationship

As you can see, identifying entities and relationships for a data model is a process of filtering and integrating. It requires certain ***common sense.***

Quick Check 6.2

1. An entity should have one or more _____ and many possible _____.
2. A relationship is an _____ between two _____ based on a _____.

1. Guides to Identifying Entities

The following hints may be helpful to you in identifying entities:

1. *Do not include specific data instances*, like English, Chinese, etc. in the 2nd example. Use an entity class instead.
2. *Defer dealing with numbers*, like "three children" in the 3rd example, "some teachers" in the 4th example and so on.
3. *Ignore those data that do not have instances.* In the 5th example, you will not record the sports day, but you will record events in the sports day.
4. *Ignore facts that can be derived.* In the 6th example, "ten products " can be derived using function COUNT. But, "product" should be recorded.
5. *Ignore calculated (derived) result.* The average marks in the 7th example will be calculated by the computer.
6. *Ignore equipment needed to carry out a process.* The "phone" in the 6th example can be ignored.
7. Bear in mind that *a true entity will have many possible instances*, each with a distinguishing characteristic.

2. Guides to Naming Relationships

Defining a name for a relationship is the perhaps the most challenging task for students. The following are some guidelines:

1. *A relationship represents an action being taken.* Therefore, a relationship name is a verb phrase (see Fig.6.2) in present tense.
2. *Relationship in E-R diagram should be read from left to right or from up to down.* Use passive voice if the action is not carried out by the entity on the left. e.g. If SUBJECT is on the left and STUDENT on the right, the name of the relationship should be *Is_studied_by*, or simply *Studied_by*.
3. *A relationship name states the action taken, not the result of the action.* e.g. use *Works_in*, not *Staff_of*; use *Studies*, not *Student_of*; use *Is_Assigned*, not *Assignment*; use *Participates* not *A_participant_of*; use *Orders* not *business* etc.
4. *Avoid using vague terms*, like *Is_related_to*.
5. *Ignore calculating processes.* In the 7th example, calculating average marks will be done by a computer.
6. *Ignore business activities like transfer, send, summarize.* In the 7th example, "sending report cards to parents" should be ignored, because this activity is irrelevant to the design of a database system. Remember that you should choose a *relationship which is a true association* between two entities.

Quick Check 6.3

In identifying entities, ignore data without _____, data that can be _____ and equipment used to carry out a _____.

Quick Check 6.4

In identifying relationships, ignore process which involves _____, and business activities, like _____, _____ and _____.

B. Relationship Cardinality

Cardinality specifies the *maximum* and *minimum* numbers of instances of an entity participating in an association with another entity.

1. Maximum Cardinality

There are three basic types of relationships with the *maximum* number of instances specified: *one-to-one (1:1), one-to-many (1:M)* and *many-to-many (M:N)*.

The typical example of ***one-to-one relationship*** is the relationship between HKID (HKID card) and PERSON (permanent resident in Hong Kong). Each HKID card is issued to one and only one permanent resident. Each permanent resident is issued one and only one HKID card.

The typical example of ***one-to-many relationship*** is the relationship between PUBLISHER and BOOK. A publisher may publish more than one book. But, each book must be published by exactly one publisher.

The typical example of ***many-to-many relationship*** is the relationship between CUSTOMER and PRODUCT. A customer may purchase more than one type of product. A type of product may be purchased by more than one customer.

Fig.6.3 shows the symbols used for specifying maximum cardinality. *Fig.6.4* shows some more examples.

Meaning	Symbol for Relationship
One-to-one	1 — Relationship — 1 or Relationship
One-to-many	1 — Relationship — M or Relationship
Many-to-one	M — Relationship — 1 or Relationship
Many-to-many	M — Relationship — M or Relationship

Fig.6.3 Symbols for maximum cardinality

Keypoints

Cardinality specifies the maximum and minimum numbers of instances of an entity participating in an association with another entity.

Note

A *many-to-one* relationship is an *one-to-many* relationship in opposite direction

Quick Check 6.5

1. AUTHOR and BOOK is an _____ relationship.
2. BOOK and ISBN is an _____ relationship.
3. BOOK and ORDER is an _____ relationship.

Relationship	Examples	Relationships illustrated
One-to-one	1. A student may buy a lunch box. A lunch box may be bought by one student.	Student : Lunch Box = 1:1
	2. A teacher may be in charge of a class. Each class must be in charge of by one teacher.	Teacher : Class = 1:1
	3. A student has one and only one student card. Each student card is owned by one and one only student.	Student : Student Card = 1:1
One-to-many	1. A student may borrow some books from the library. A book in the library may be borrowed by a student.	Student : Book = 1:M
	2. A summer course is attended by one or more students. Each student may apply for at most one summer course.	Summary Course : Student = 1:M
	3. A student may be assigned to some functional posts (monitor, perfect or club chairman). Each functional post is assigned to one and only one student.	Student : Functional Post = 1:M
	4. A class is formed by a group of at least one student. A student is allocated to one and only one class.	Class : Student = 1:M
Many-to-many	1. A student may apply for some scholarships. Each scholarship may be applied for by some students.	Student : Scholarship = M:N
	2. An event in the sports day is participated by at least one student (Otherwise the event will be cancelled). Each student may participate in some event.	Event : Student = M:N
	3. A student takes at least at least one course. A course is taken by at least one student.	Student : Course = M:N

Fig.6.4 Examples of three basic cardinality

Besides the maximum number of instances in a relationship, cardinality also specifies the minimum number of instances, using terms: *mandatory* and *optional.*

2. Mandatory Cardinality

An instance is **mandatory** (symbol —|—) if there exists at least one instance in the relationship. Terms like *must, at least one* and *one and only one* are used to describe mandatory cardinality.

For example, a class must have a class teacher. Therefore, TEACHER is mandatory in the relationship between CLASS and TEACHER. However, CLASS is not mandatory because some teachers may not be a class teacher.

Another example is the relationship between HKID (HKID card) and PERSON (permanent resident in HK). The relationship is mandatory *in both directions* because every HKID card must have an owner and a permanent resident must have an HKID card.

The above two examples illustrate mandatory cardinality in one-to-one relationships. The relationship between CUSTOMER and ORDER is a mandatory one-to-many relationship. CUSTOMER is mandatory because a sale order must be submitted by a customer. However, ORDER may or may not be mandatory, depending on the company policy, which is called **business rule**. Some companies treat potential customers who have asked for price quotation as customers. But, some do not.

Keypoints

There are four possible cardinalities: mandatory one, mandatory many, optional one and optional many.

3. Optional Cardinality

An instance is **optional** (symbol —⊖—) if the instance may or may not exist. Terms like *may, or none* and *at most one* are used to describe optional cardinality.

Consider a school music concert in which students may buy any number of tickets. The relationship between STUDENT and TICKET in both directions are optional. A student may not buy any ticket. Also, a ticket may not be sold out at all.

In conclusion, there are four possible cardinalities: *mandatory one, mandatory many, optional one* and *optional many*, as shown in Fig.6.5. See more examples in Fig.6.6.

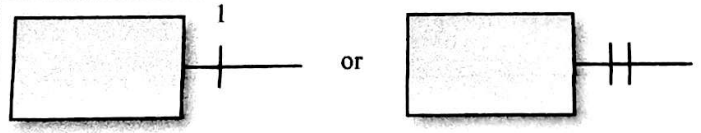
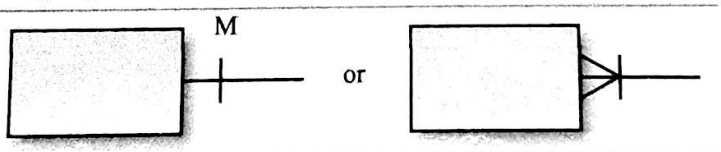
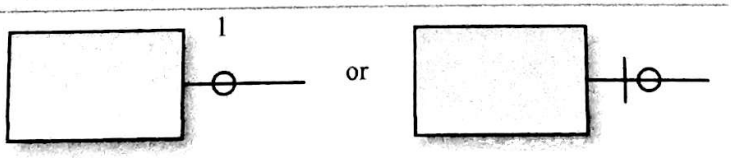
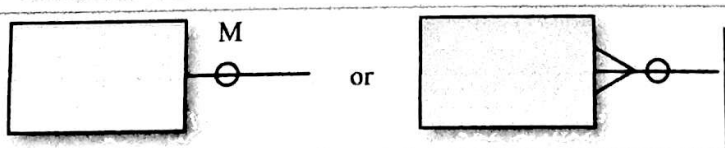
Meaning	Symbol for Relationship Cardinality
Mandatory One *One and only one*	
Mandatory Many *One or many*	
Optional One *Zero, or one*	
Optional Many 0, 1, 2, …	

Fig.6.5 All possible cardinalities

Quick Check 6.6

1. In the relationship between AUTHOR and BOOK, both sides are _____.
2. In the relationship between BOOK and ORDER, the mandatory side is _____ and the optional side is _____.

C. Sample relationships

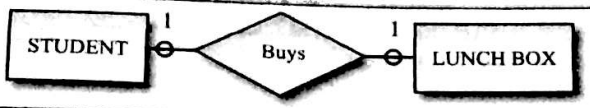
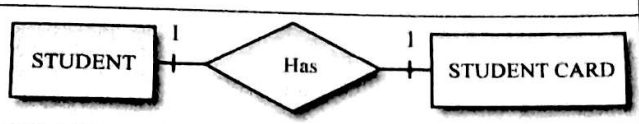
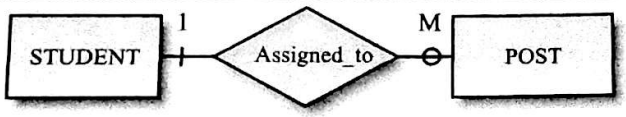
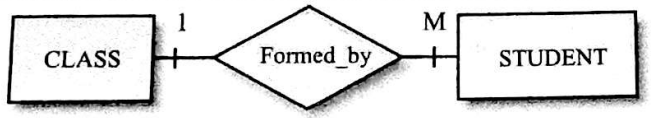
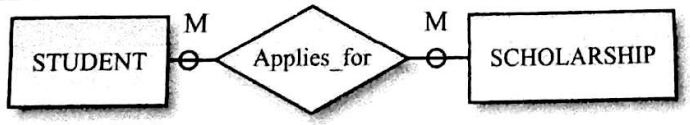
E-R diagram	Remark
1. 	*One-to-one relationship; Optional instances on both sides.* Student may buy at most one lunch box. Each lunch box may or may not be sold to a student.
2. 	*One-to-one relationship; Optional instance on one side, mandatory instance on the other side.* A teacher may or may not be in charge of a class. But, each class must be in charge of by one teacher.
3. 	*One-to-one relationship; Mandatory instances on both sides.* Each student must have a student card and each student card belongs to exactly one student.
4. 	*One-to-many relationship; Optional instances on both sides.* A student may borrow some books from the library. A book in the library may be borrowed by at most a student.
5. 	*One-to-many relationship; Optional instance on one side.* There must be at least one student for a summer course to exist. Enrollment by students is optional. But, a student may apply for at most one summer course.
6. 	*One-to-many relationship; Optional instance on one side.* A student may be assigned to some posts, like monitor, perfect or club chairman. Each post must be responsible by one student.
7. 	*One-to-many relationship; Mandatory instances on both sides.* A class is formed by a group of at least one student. Each student is allocated to one and only one class.
8. 	*Many-to-many relationship; Optional instances on both sides.* A student may apply for more than one scholarship. Each scholarship may receive some applications from students, or none.
9. 	*Many-to-many relationship; Optional instance on one side.* An event in the sports day is participated by at least one student (Otherwise the event will be cancelled). Each student may participate in some event.
10. 	*Many-to-many relationship; Optional instance on one side.* A student takes at least one course. A course is taken by at least one student.

Fig.6.6 Ten examples of relationships

Example 1 *In a school, students are allocated to different classes. Each student must be allocated to exactly one class, and a class is formed by at least 30 students. Each class must be managed by several different students, namely, prefect, 1st monitor, 2nd monitor and 3rd monitor. Draw an E-R diagram for the school, indicating cardinality.*

Solution The first step is to classify the prefect and monitors into an entity class, CLASS_POST. The relationship between CLASS and CLASS_POST is mandatory in both directions because each class must have several posts and these posts cannot exist without a class. The relationship between STUDENT and CLASS is also mandatory in both directions, because student must be allocated to one class and a class must consist of at least one student. A student may or may not be assigned to a post. Therefore, it is optional on the side of CLASS_POST. A post must be assigned to a student. Therefore, it is mandatory on the side of STUDENT.

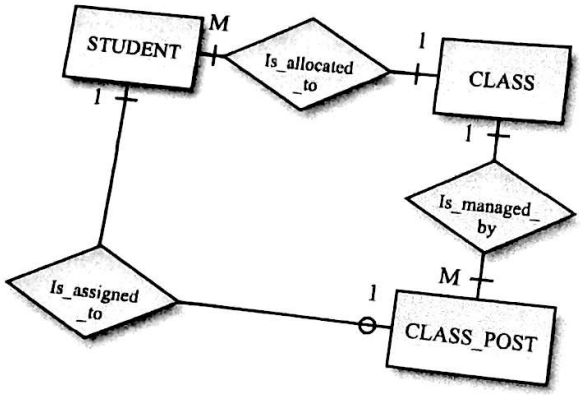


Fig.6.7

Example 2 *Wing Lee Inc. is a construction company with over 1000 employees. A customer can hire the company for more than one project, and employees sometimes work on more than one project at a time. Equipment is assigned to only one project. Draw an E-R diagram for Wing Lee Inc., indicating cardinality.*

Solution It is reasonable to assume that a construction project involves more than one employee and requires more than one equipment.

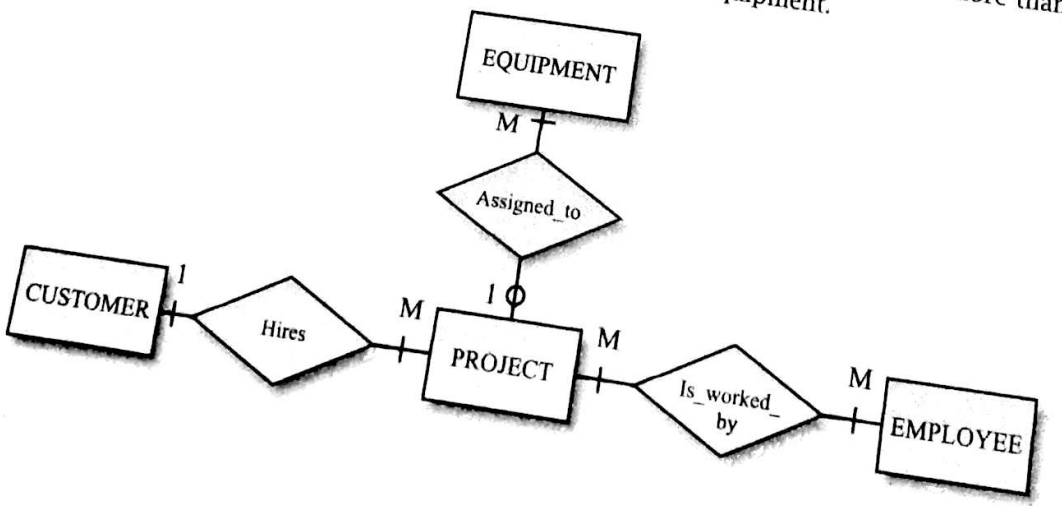


Fig.6.8

Example 3 A library is going to use a computer system to handle the circulation. Each user may borrow several books at each time. Each book is recorded separately by a librarian into a check-out record, which contains data about the book, borrower, librarian, and dates (borrow and return). There are three librarians working on three shifts.

(a) Draw an E-R diagram to show the relationships between BOOK and USER, indicating cardinality.

(b) Identify two other entities for the library information system and draw another E-R diagram to represent the relationships between these entities, showing the cardinality of each relationship.

Solution (a)

Fig.6.9a

The relationship between BOOK and USER is optional because a book may never be borrowed out, and a user may never borrow any book.

(b) Since each check-out has attributes, like date, borrower, etc., check-out is an entity. Another entity is librarian.

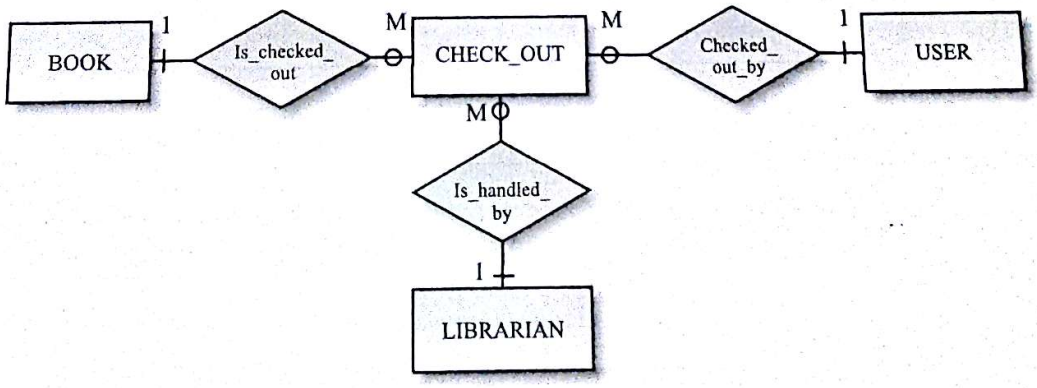


Fig.6.9b

The direction from LIBRARIAN to CHECK_OUT is optional because a librarian may never handle any check-out (e.g. newly employed or the system is just started). However, the direction from CHECK_OUT to LIBRARIAN is mandatory because each check-out must involve one librarian. Similarly, each check-out must involve one book and one user.

D. Degree of Relationship

Keypoints

The degree of a relationship is the number of entity classes participating in that relationship.

Binary relationships involve two entities and are the most common. Unary relationships involve one entity.

The **degree** of a relationship is the number of entities participating in that relationship. All the relationships mentioned above involve two different entities. Therefore, they are of degree 2 and called **binary relationships**. Binary relationships are the most common.

Unary (degree 1) relationship involves one entity. The following are two examples of unary relationship:

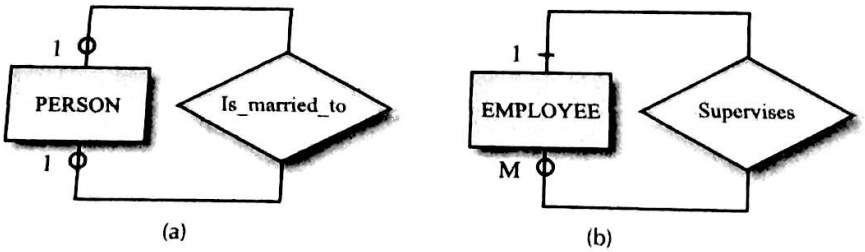


Fig.6.10 Unary relationship

In Fig.6.10a, a person may (*or may not*) be married to another person. If they are married, they have exactly one spouse. Therefore, the relationship is *one-to-one* and *optional* in both directions.

In Fig.6.10b (Read the relationship from top to bottom through the relationship name), an employee may supervise more than one staff. The term "may" means that an employee may or may not supervise other employee at all. Therefore, optional-many is used below EMPLOYEE. When the statement is read in the reverse order, an employee must be supervised by another employee. In our example, each staff belongs to one department each of which must have a department head. Therefore, mandatory-one is used above EMPLOYEE.

Ternary (degree 3) relationship and higher degree relationships are beyond the scope of this book. Fig.6.11 shows the typical shape of ternary relationship.

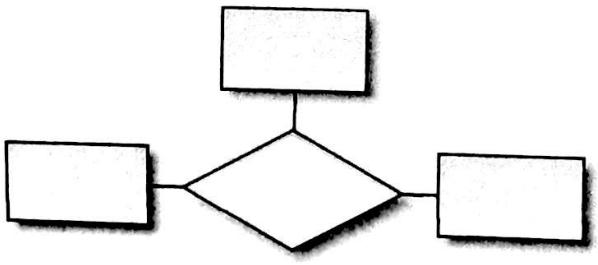


Fig.6.11 Ternary relationship

Quick Check 6.7

The numbers of entities involved in unary, binary and ternary relationships are ____, ____ and ____ respectively.

E. Multiple Relationships

In some situations, there are more than one relationship between two entities. Consider the following example:

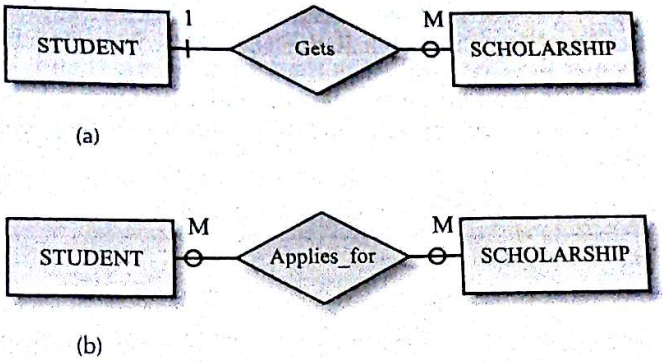


Fig.6.11 Two relationships between two entities

Refer to Fig.6.11a. A student may get more than one scholarship (or may not get any scholarship). A scholarship must be awarded to exactly one student. Refer to Fig.6.11b. A student may apply for more than one scholarship. A scholarship may be applied for by more than one student.

The two relationships can be drawn on the same E-R diagram as shown in Fig.6.12.

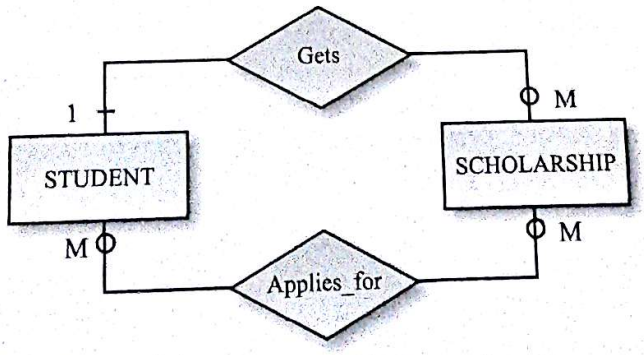


Fig.6.12 Multiple relationships

6.2 E-R DIAGRAMS WITH ATTRIBUTES DRAWN

Keypoints

- Foreign keys are usually not drawn in E-R diagrams.

An attribute is shown using an oval connected to an entity. Key attributes are underlined. When an entity is transformed into a database table, the key attribute will be the primary key of the table.

A relationship is formed by linking the foreign key of a table to the primary key of another table. In general, **if a relationship has been drawn to link two entities and the first entity has the key attribute, then it is not necessary to draw the foreign key for the second entity**. In other words, the second entity is assumed to have an attribute with values matching the key attribute of the first entity.

The following are some guidelines in drawing attributes:

1. Use *noun* to represent attributes.
2. Attributes must be *unique*.
3. Every *basic entity* must have a *key attribute*. **Basic entities** are entities originally identified and represent fundamental objects in an organisation, like STUDENT, BOOK, LIBRARIAN etc.
4. An attribute that represents the *foreign keys* of a relationship need *not* be drawn (see Example 4 below).
5. Attributes derivable from other attributes need *not* be drawn. For example, if `Date_birth` has been recorded, `Age` or `Year_birth` are derived attributes that can be ignored (see Example 4 below).
6. Clearly indicate *multi-valued attributes*. For example, `Qualification` is a multi-valued attribute for a teacher because a teacher may obtain several qualifications. Use double-oval to indicate multi-valued attributes (See Example 5 below).

Example 4 In a school, a student may be assigned with one or more functional posts, like prefect, monitor or chairman. A post must be assigned to exactly one student. Draw a complete E-R diagram to represent the relationship between STUDENT and POST, indicating cardinality.

Solution

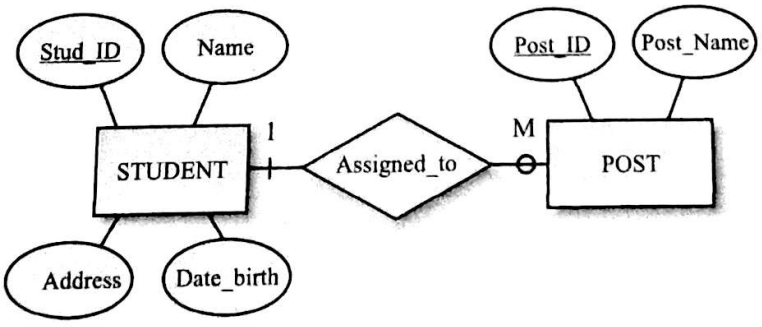


Fig.6.13

The entity POST has foreign key `Stud_ID`. But, we don't need to draw foreign key in E-R diagram, because the relationship has associated `Stud_ID` with `Post_ID`. Also, we don't need to include the *derived* attributes `Year_birth` in STUDENT (see Fig.6.14).

Quick Check 6.8

1. Foreign keys need/need not* be drawn in E-R diagrams.
2. Derived attributes need/need not* be drawn in E-R diagrams.
3. MULTIVALUED attributes are represented by double-oval.
4. BASIC entities must have a key attribute.

*delete inappropriate

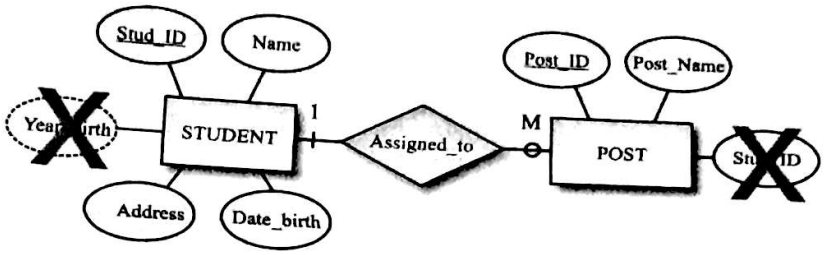


Fig.6.14 Stud_ID in POST is a foreign key; Year_Birth in STUDENT is a derived attribute

Example 5 In a school, a teacher teaches in one or more classes, each of which is taught by one or more teachers. A teacher may have more than one qualification. On the other hand, each class must be responsible (lead) by one class teacher. Sketch an E-R diagram to show the relationship between TEACHER and CLASS, indicating cardinality.

Solution

Alternative 1 Single relationship

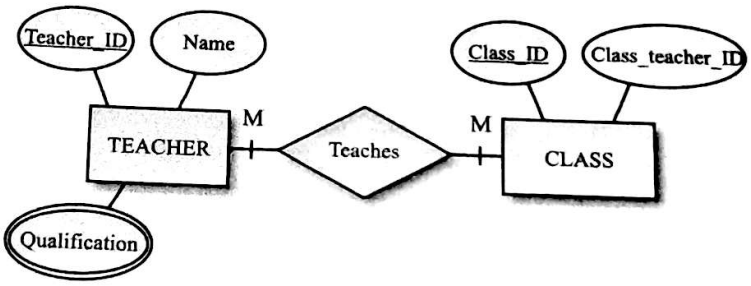


Fig.6.15a

Note

Double-oval is used to indicate multi-valued attribute.

Alternative 2 Multiple relationships

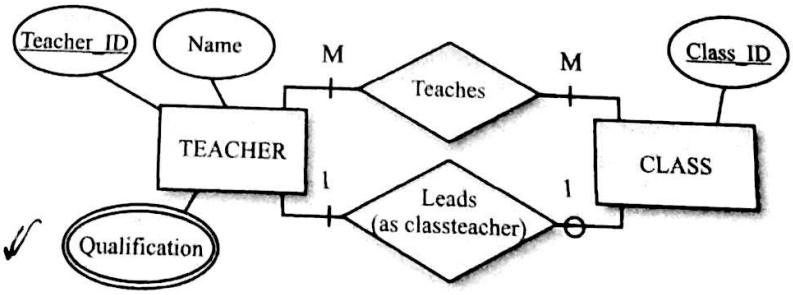


Fig.6.15b

The difference between the above diagrams is that `Class_teacher_ID` is added to CLASS in Fig.6.15a. This is necessary because the relationship "Teaches" does not represent the class teacher.

In Fig.6.15b, `Class_teacher_ID` needs not be drawn explicitly because a new relationship is added to associate `Class_ID` with `Teacher_ID` primarily to describe class teacher.

6.3 RELATIONSHIPS WITH ATTRIBUTES

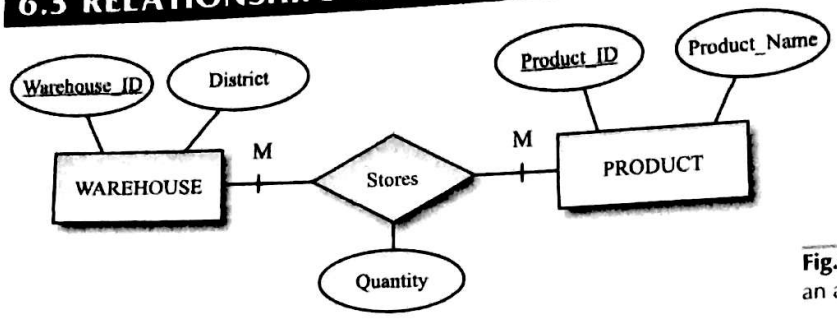


Fig.6.16 A relationship with an attribute

Some relationships may have attributes. This commonly occurs in *many-to-many* relationships. Let's consider the M:N relationship between WAREHOUSE and PRODUCT shown in Fig.6.16. A warehouse may store several kinds of products; and each kind of product may be stored in several warehouses. Attribute `Quantity` is added to the relationship to record the quantity of a kind of product in a warehouse.

As a general rule, **relationships with attributes should be converted into entities**. The newly created entity is known as **associative entity**. The entity is *associative* because it is converted from a relationship which is an *association* between two entities. It is an *entity* because it has instances each with distinguishing attribute values (see 6.1 for requirements for an entity).

A feature of an associative entity is that the primary key is formed by a combination of two foreign keys. As mentioned in 6.2, foreign keys are not shown in E-R diagrams. Therefore, associative entities are drawn *without* any key attribute.

Fig.6.17 and 6.18 show two equivalent E-R diagrams in which a new entity STORAGE is created to replace the original relationship *Stores*.

Keypoints

- Relationships with attributes should be converted into entities.
- Entity converted from a relationship is called associative entity.
- The primary key of an associative entity is formed by a combination of the foreign keys.

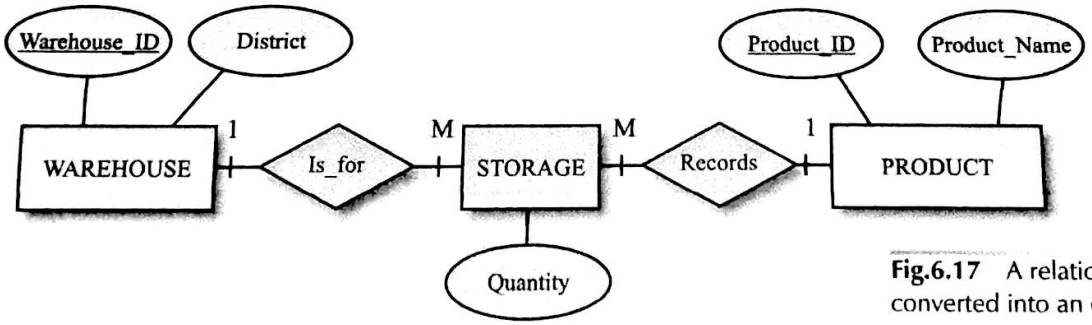


Fig.6.17 A relationship converted into an entity

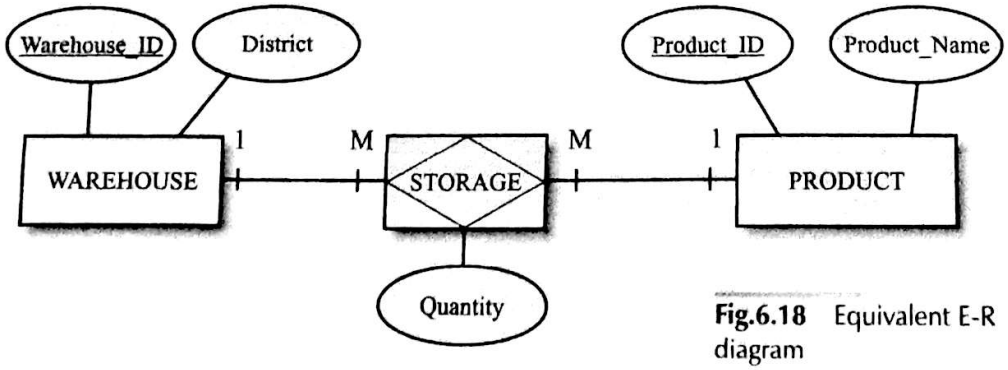


Fig.6.18 Equivalent E-R diagram

Quick Check 6.9

An associative entity is converted from a _____. The primary key of an associative entity is _____ _____.

Example 6 A joint project is hold by a school and a voluntary service organisation. The organisation offers three types of voluntary services to students in the school. Students may participate in more than one such services. But, the teachers will adjust the services chosen by students so that the number of participants in each service is more or less the same. Score will be calculated for each student based on the number of hours worked and the type of services at the end of the school year.

(a) Draw an E-R diagram to show the relationship between STUDENT and SERVICE and include the necessary attribute(s) that enable(s) the score to be calculated, indicating cardinality.

(b) Re-draw the E-R diagram with a new entity converted from the relationship in (a).

Solution (a)

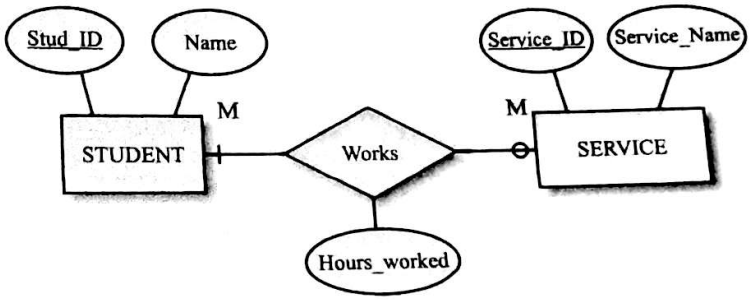


Fig.6.19

(b)

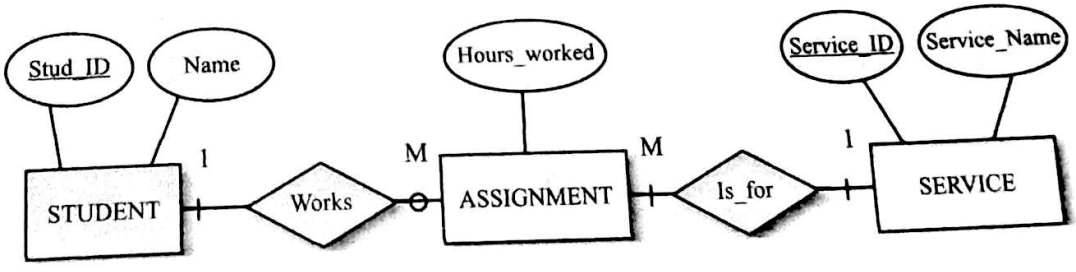


Fig.6.20

6.4 RESOLVING E-R DIAGRAM FOR RELATIONAL DB

The ultimate purpose of drawing E-R diagrams is to design a database. Some entities can be transformed into tables directly, while others need some conversions.

For instance, consider an entity with a multi-valued attribute. Since we do not know the number of attribute values in each instance (record), we have to use one row for each attribute value. As a result, an instance will have multiple rows. Or, a row will have multiple rows. This is not allowed in relational databases.

Similarly, many-to-many (M:N) relationships cannot be implemented directly without conversion. In an M:N relationship, we cannot tell the number of rows for a foreign key field in each record. This ends up with multiple rows in a row, similar to the case of multi-valued attributes above.

The process of converting an E-R diagram into a form that makes it possible to transform into a relational database is called **resolution**. You will see that all resolutions involve creating new entities. In fact, the discussion in 6.3 is a resolution that converts an M:N relationship into multiple 1:M relationships by creating an *associative entity*.

We shall discuss the resolutions of three types of relationships: *binary M:N relationship, multi-valued attribute* and *unary M:N relationship.*

Keypoints

Resolution is to convert an E-R diagram into a form transformable into a relational database.

An M:N relationship must be converted into multiple 1:M relationships, by creating a new entity.

A. Resolving Binary M:N Relationship

As a general rule, **an M:N relationship must be converted into multiple 1:M relationships**. The technique is to **create a new entity to replace the original M:N relationship**.

Fig.6.21 shows a binary M:N relationship between CUSTOMER and PRODUCT: A customer may have purchased more than one product. On the other hand, a product may be purchased by more than one customer.

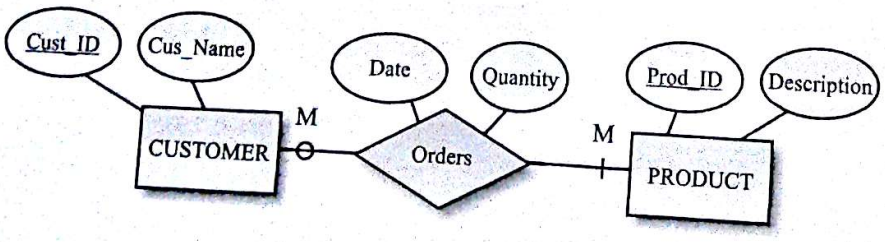


Fig.6.21 Many-to-many relationship

Quick Check 6.10

1. Multi-valued attributes cannot be represented in a _____.
2. An M:N relationship must be resolved into two or more _____ relationships.