# DBMS - 1

18-08-'14

# DBMS

## Introduction

### Data

- The word data is used to refer to a fact or facts about a person, place, object, event or concept.
- Data is a collection of facts and raw materials that reflects the objects seen around us.
- Data is a raw material to generate information.

### Information

- Data arranged in certain order and form which is useful to us is called an information.
- Data is a raw material to generate information, i.e. data is to be processed to produce information. Thus, information is the output of data processing operation.

For eg : The weights of students are 16 kg, 17 kg, 57 kg, 41kg, 36 kg, 44 kg.....
The above numbers are raw or little meaning, but let us process the data to answer the following questions.

- What is the average weight?
- What is the weight of the heaviest?
- What is the weight of the lightest?

## Database

A database consists of a collection of interrelated data stored in a standarised format and a set of programs to access those data so that multiple users can share them. The collection of data, usually referred to as database contains information about one particular enterprise.

## DBMS

DBMS is a software that defines a database, stores data, supports a query language, produce reports and creates data entry screen. The primary goal of DBMS is

to provide an environment that is both convinient and efficient to use in retrieving and storing database information. Database systems are designed to manage large bodies of information.

Some popular DBMS software are. Oracle, SQL server, IBM-DV2, MySQL, MSQL, MS-Access.

Applications

- Banking : customer and their account info
- Airlines : reservations and setting info
- Universities : students info, grades, etc
- Telecommunication : records of all calls made
- Finance : for storing information about holding, sales and purchases.
- Sales : for customer, product and purchase information
- Manufacture : for managing supply obtain
- Human resource : for information of employee

19-08-'14

Purpose of DBMS / Why DBMS is required?

Traditionally, file processing system was used to manage information. It stores data in various files of different application programs to extract or insert data to appropriate file.

File processing system has several drawbacks due to which DBMS is required. DBMS removes problems found in file processing system.

Some major problems of file processing systems are:

i. Data redundancy and inconsistency
ii. Difficult in accessing data
iii. Data isolation
iv. Integrity problem
v. Atomicity problem
vi. Concurrent access problem
vii. Security problem

21-08-14

Objective of DBMS / Properties of DBMS

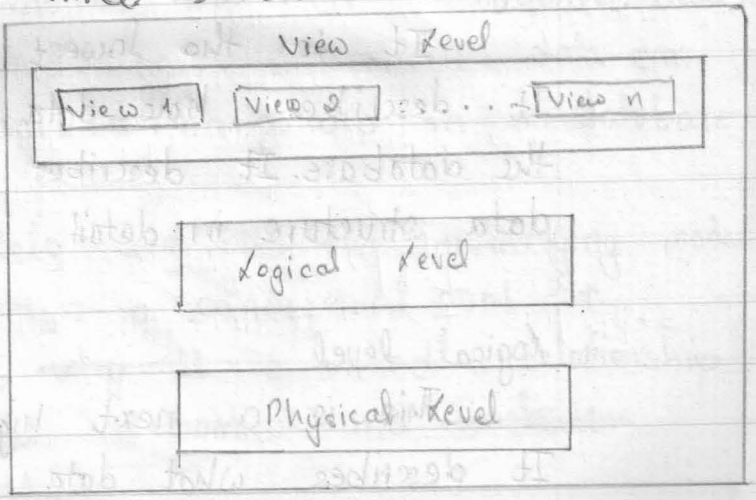
i. Provide for mass storage of relevant data
ii. Making easy access to data for authorized user.
iii. Combining interrelated data to generate report.
iv. Protect the data from physical harm and unauthorized access.
v. Allow multiple users to be active at one time

Data Abstraction

Data abstraction in database system is a mechanism to hide complexity of database. It allow database system to provide abstract view to database user. It hides how data are actually stored and maintained in data base.

There are three levels of abstraction:

i. Physical level
ii. Logical level
iii. View level



i. Physical level
- lowest level of data abstraction.
- how data is actually stored in the database.

ii. Logical level
- second lowest level of data abstraction.
- describes what type of data are stored in the database and what is the relation between them

iii. View level
- highest level of data abstraction
- describes the interaction between the users and system.

i. Physical level

It is the lowest level of abstraction. It describes how data are actually stored in the database. It describes complex lower level data structure in detail.

ii. Logical level

This is a next highest level of abstraction. It describes what data are stored in database

and what relationship exist among them. It describes entire database relatively in a simple structure.

iii. View level

This is the highest level of abstraction. It describes only part of a entire database. It simplify interaction with the system. It allows database system to provide many views for the same database, i.e. it allows each user/application to get different perspective of the database.

## Data Models

Database models are used to organize data. It determines the manner in which data can be stored, organized and manipulated in a database system.

Data models describe the underlying model of database. It is a conceptual tool for describing data, ~~why it is called a~~ relationship among data, data semantics and ~~consistency~~ consistency constraints.

There are several data models which can be put into different categories:

i. Object - based logical models
ii. Record - based logical models
iii. Physical Data models.

29-08-14 Assignment # 2

Prepare a detail report on "History of AI"

24-08-14

i. Object-based logical Models

Object-based logical model describe data at the logical and view levels. It has flexible structuring capabilities. It allows to specify data constraints explicitly. ~~under~~

Under object-based logical model, there are two data models : ~~entity~~

a. Entity-relationship model
b. Object-oriented model

a. Entity-relationship model

E-R model describes the design of database in terms of entities and relationship among them. An entity is a 'thing' or 'object' in real world that are distinguishable from other objects. An entity is described by a set of attributes.

E.g: Attributes: customer_id, customer_name, customer-city may describe entity customer.

Attributes: acc_no, balance may describe entity account.

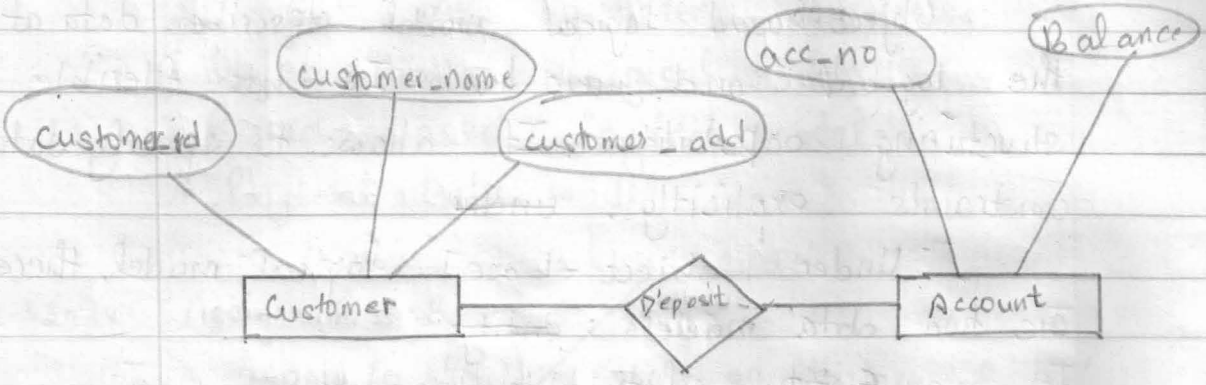



Fig. Sample of E-R diagram

- A relationship is an association among several entities. For eg: A depositor relationship associates ~~with~~ a customer with each account s/he has.
- The set of all entities of same type called entity set and similarly, set of all relationships of same ~~type~~ is called relationship set.
- Attribute is a property or the characteristics of an entity which is represented by oval or ellipse.

b. Object-oriented model

- The object oriented model is based on a collection of objects, like the E-R model. Object contains values and bodies of codes, which are called methods. An object can contain another object to an arbitrarily deep level of nesting.
- In object-oriented model, data is represented in the form of object.
- Another feature of object-oriented model is inheritance, i.e. new class can be derived from the existing class.

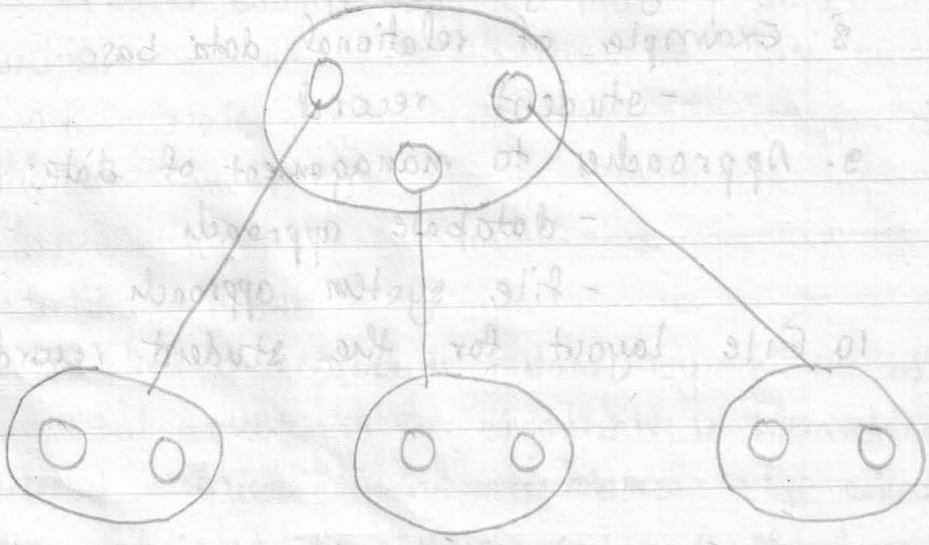


Fig. Object-oriented model

25-08-14

Assignment # 1

1. Why data management is important?
2. When does data become information?
3. DBMS :
   - definition
   - primary goal of DBMS
   - related data
   - integrated data
4. DBMS and DBS
5. Example : uses of DBMS (application)
6. History
7. simplify system organization
8. Example of relational data base
   - student record
9. Approaches to management of data:
   - database approach
   - file system approach
10. File layout for the student records

26-08-14

ii. Record-based logical model

Record-based logical model also describes data at logical and view level but it describes logical structure of database in more detail for implementation point of view. It describes data structure in terms of fixed format records of different types. Each table contains records of a particular type. And each record type defines fixed number of fields or attributes. Each field is usually of a fixed length

The three most widely accepted models under record based logical models are:

- hierarchical
- network model
- relational model

a. Hierarchical Model

It also represents data by a set of records but records are organized in hierarchical or ordered structure and database is a collection of such disjoint trees. The nodes of the tree represents record types. Hierarchical trees consist one root record type along with 0 more

occurance of its dependent subtree. And each dependent subtree is again hierarchical No dependent record can occur without its parent record.

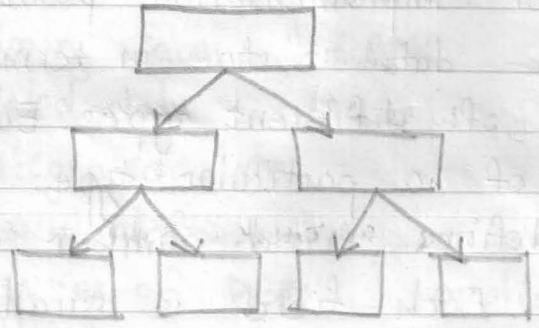


Fig. Hierarchical model

b. Network model

- In network model, data are represented by the set of records and relationship among data are represented by links.
- In network database model, network of connection between data elements are established. This model overcomes the problems faced in hierarchical model of searching data from bottom or middle.
- Networked data base have large number of sets of record, each containing a small amount of information and a large number of pointers to other sets of records.

29-08-'19

- So network database are difficult to use but more flexible than hierarchical database.

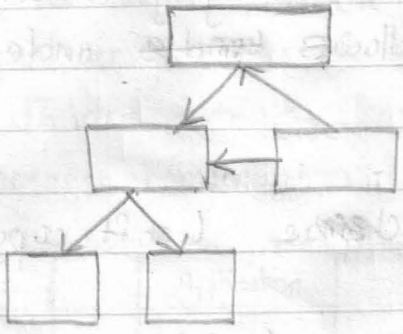


Fig. Network Model

c. Relational Model

- It describes database design by a collection of tables (relation). It represents both data and their relationships among those data. Each table consists number of columns (attributes) with unique names. It is a most widely used data model. Relational model is lower level abstraction than ER model. Database model are often carried out in ER model and then translated into relational model.

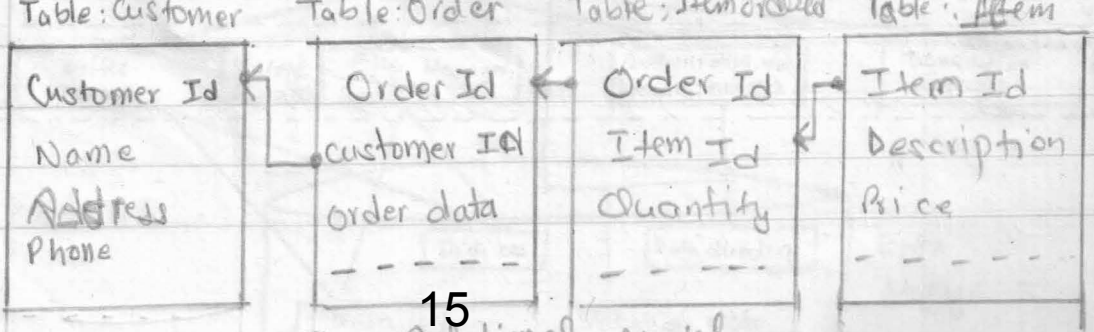



Fig. Relational Model

iii. Physical Data Model

Physical data models are used to describe data at the lowest level. Unifying model and frame memory are follows under physical data model.

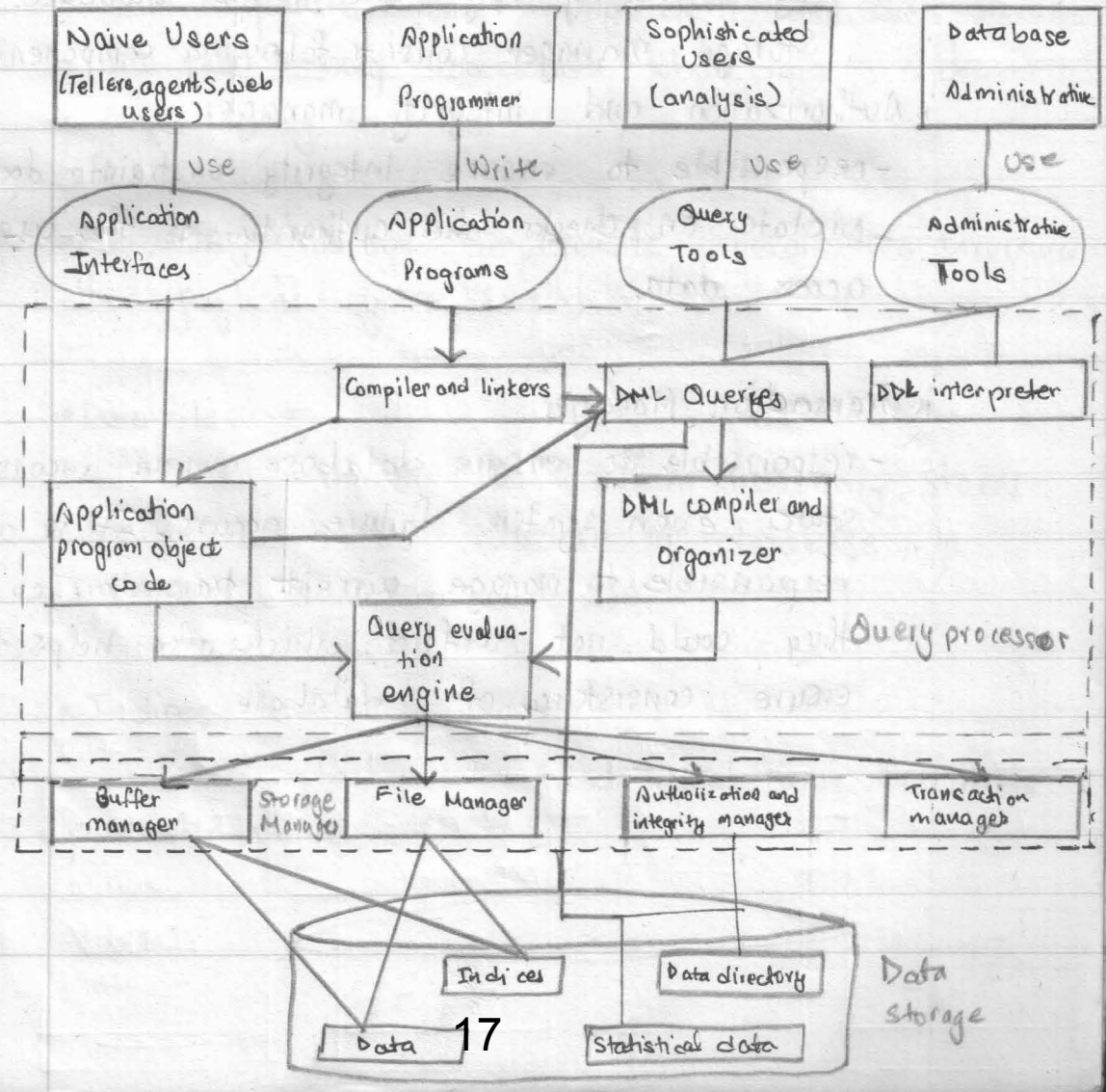
* Instance and schemas (soft copy)

* Data Independence (soft copy)
* Database language
* Database Manager
* Database Administrator

31-08-'14

## Database System Structure (Overall System Structure)

A database system is partitioned in models that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage management and the query processor component.



* Storage Manager

Storage manager is a program module that provides interface between the low level data stored in the database and, the applications programs and the queries submitted to the system. It is responsible for the interaction between the file manager and storing, retrieving and updating data in the database.

Storage manager consists following components.

i. Authorization and integrity manager
- responsible to ensure integrity constraints does not violate an checks the authority of the users to access data.

ii. Transaction Manager
- responsible to ensure database remain inconsistent state even system failure occurs. It is also responsible to manage current transaction so that they could not conflict, which also helps to ensure consistency of database.

01-08-14

iii. File Manager

Responsible to manage the allocation of space on disk storage and data structure used to represent information stored on disk.

iv. Buffer Manager

Responsible for fetching data from disk storage into main memory and decides what data to cache in main memory. ~~The stor~~

The storage manager implements several data structure for physical system implementation.

- Data files

Stores database itself. Data dictionary, stores meta data about structure of database, in particular schema of database.

- Indices

Provides fast access to data ~~val~~ items that holds particular values.

* Query Processor

The query processor is responsible to simplify and facilitate access data. It is responsible to translate updates and queries retain in non procedural language at the logical level into an efficient sequence of operations at the physical level.

The query processor components include the following components:

i. DDL Interpreter
- Responsible to interpret DDL statements and records the definition in the data dictionary.

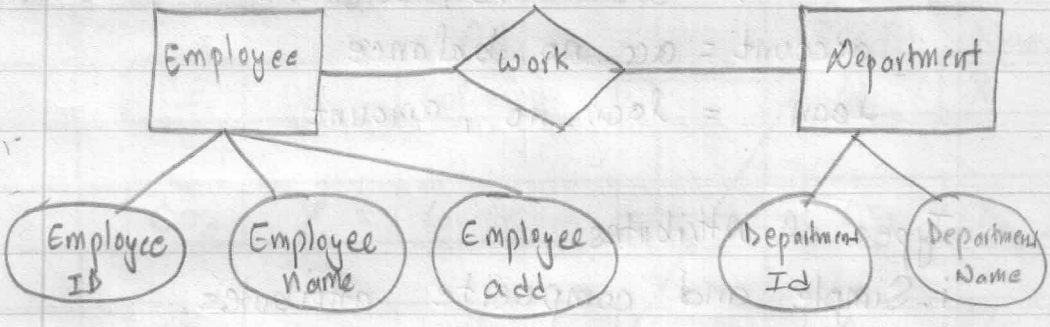
ii. DML Compiler
- Responsible to translate DML statements in a query language into low level instructions that query evaluation engine understands query is generally translated into members of alternative evaluation plans that produce the same results. It is often also responsible for query optimization. It requires to select the lowest cost evaluation plan among the alternatives.

iii. Query Evaluation ~~Model~~
- Responsible to execute laser level instruction genesated by DML compiler.

iii. Query Evaluation
- Responsible to execute lower level instruction generated by DML compiler.

Entity Relationship Model

Make an ER diagram to show the relationship between employee and department




Entity relationship model describes data involves in real world in terms of object and their relationship. It is widely used for initial database design. It describes overall structure of database. E-R model is infact, semantic data model which describes the meaning of data. It has a capability to map the meaning and interaction of real world objects on the conceptual schema.

## Entity and Entity Sets

Entity is a object or a things.

### Attribute

In simple, attributes is ~~discipline~~ descriptive properties of entity set. Set of attributes describes entity set.

For eg:

customer = customer_id, customer_name, customer_city

account = acc_no, balance

loan = loan_no, amount

### Types of Attributes

i. Simple and composite attributes
ii. Single valued and multivated attributes.
iii. Stored and derived attributes.

### Relationships to relationships set

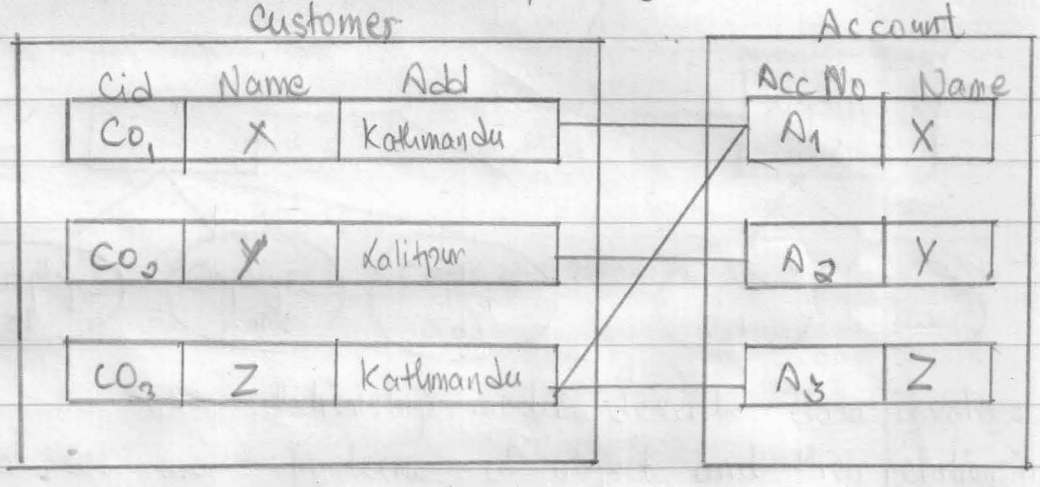
A relationship is an association among two or more entities. A relationship set is a set of relationship of same type.

Formally, if $E_1, E_2 \ldots E_n$ $(n \geq 2)$ are entity sets then a relationship set ~~of relationship of same type~~ 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.

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



02-09-14

## Constraints in E-R Model

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

i. Mapping cardinalities (cardinality ratio)
ii. Participation constraints

### i. Mapping cardinalities

Mapping cardinalities describe number 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 the relationship sets that involves more than two entity sets. For binary relationship set between entity sets A and B, mapping cardinality must be one of the following:

a. one-to-one (1:1)
b. one-to-many (1:M)
c. many-to-one (M:1)
d. many-to-many (M:N)

a. One-to-one

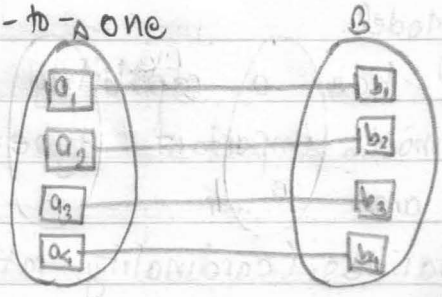



Fig One-to-one

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.

b. One to many

An entity in A is associated with 0 or more entities in B but entity in B can be associated with at most one entity in A.

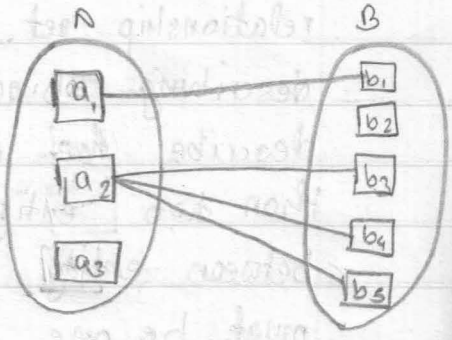



Fig One-to-many

c. Many to one

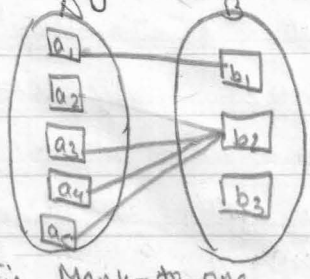



Fig. Many-to-one

An entity in A is associated with at most one entity in B but an entity in B can be associated with 0 or more entities in A.

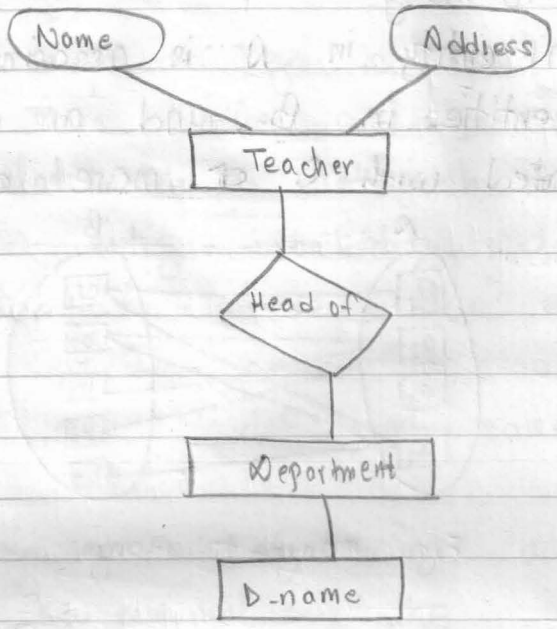



Fig. E-R diagram of one-to one relationship

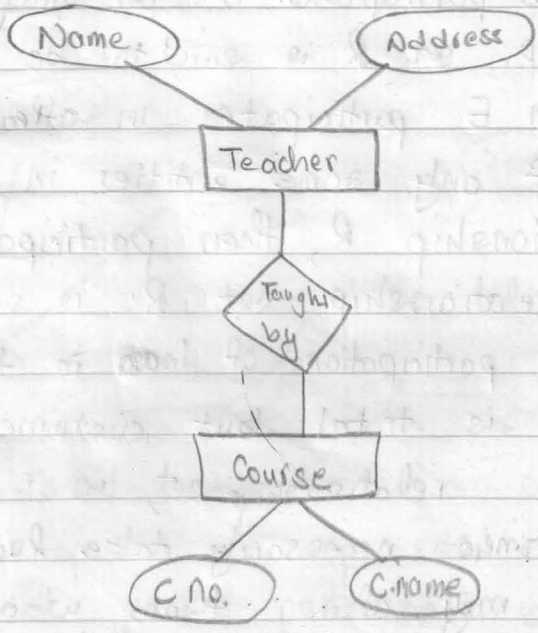



Fig. E-R diagram of one to many relationship

d. Many-to-many

An entity in A is associated with 0 or more entities in B, and an entity in B is associated with 0 or more entities in A

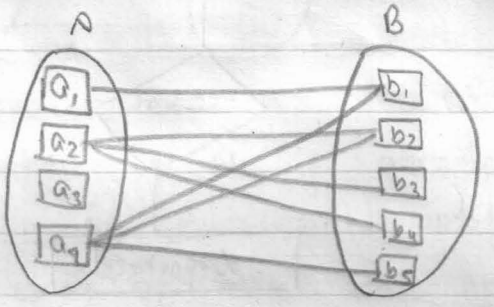



Fig. many-to many

ii. Participation Constraints

The participation of an entity set A in a relationship set R is said to be total if every entity in E participates in atleast one relationship in R. If only some entities in E participate in relationship R, then participation of entity set E in relationship set R is said to be partial.

The participations 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 holders. Such participation constraint can by expressed by E-R model.

4-08-14

## Keys

The concept of key is important to distinguish one entity from another and one relationship from another relationship. To distinguish one entity from another ~~in entity~~ in an entity set, there must exist attribute/s whose values must not duplicate in entity set. It ensure no two entities in an entity set can exist with same values for all attributes.

- Super key combination of 1 or more attributes
- Candidate key minimal super key
- primary key candidate key

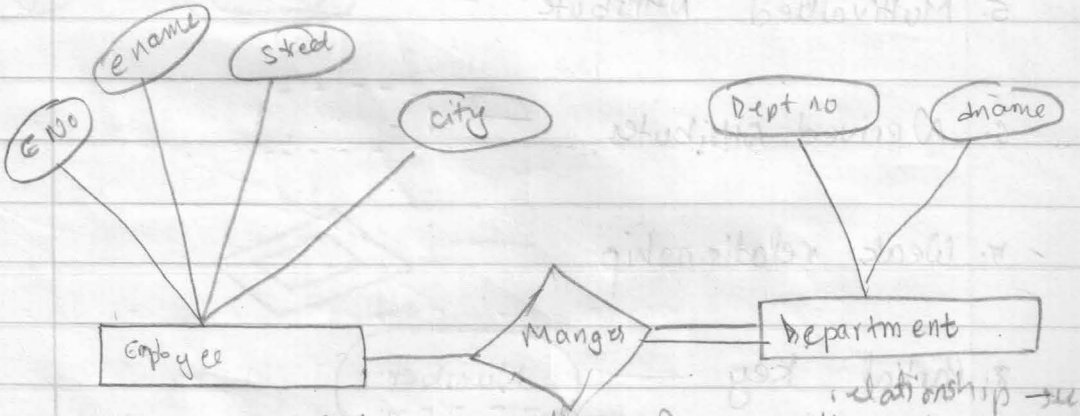
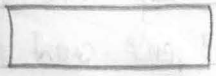



Fig. Total participation of an entity set in 1 relationship

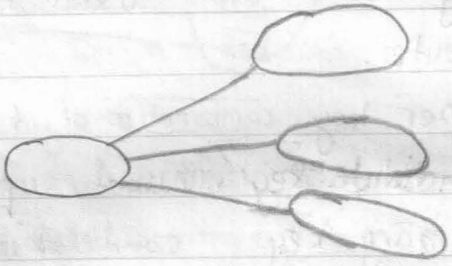
- Total participation is represented by a double line and partial participation is represented by a single line.

Major components of E-R diagram are:

1. Entity —


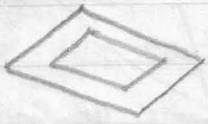
2. Weak entity —

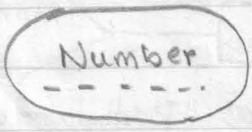

3. Relationship —

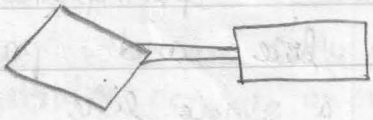

4. Attributes —


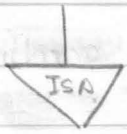
5. Multivalued Attribute —


6. Derived Attributes —


7. Weak relationship —


8. Partial key —


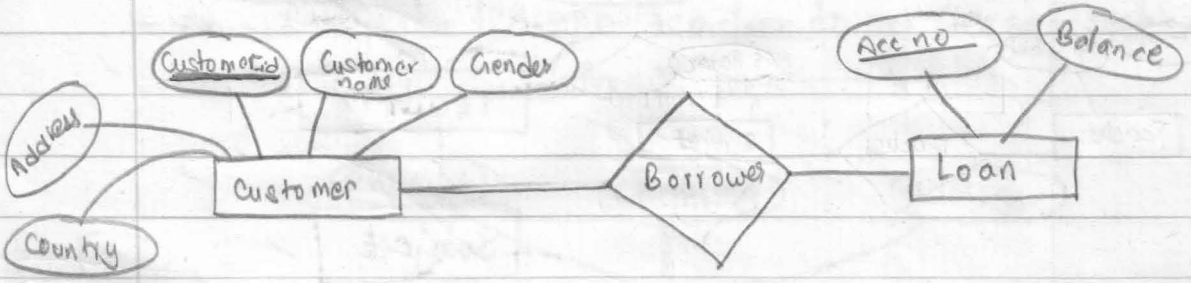

9. Total Participation —


10. ISA - 


11. Primary key — 

Representation of Relationship set in E-R diagram

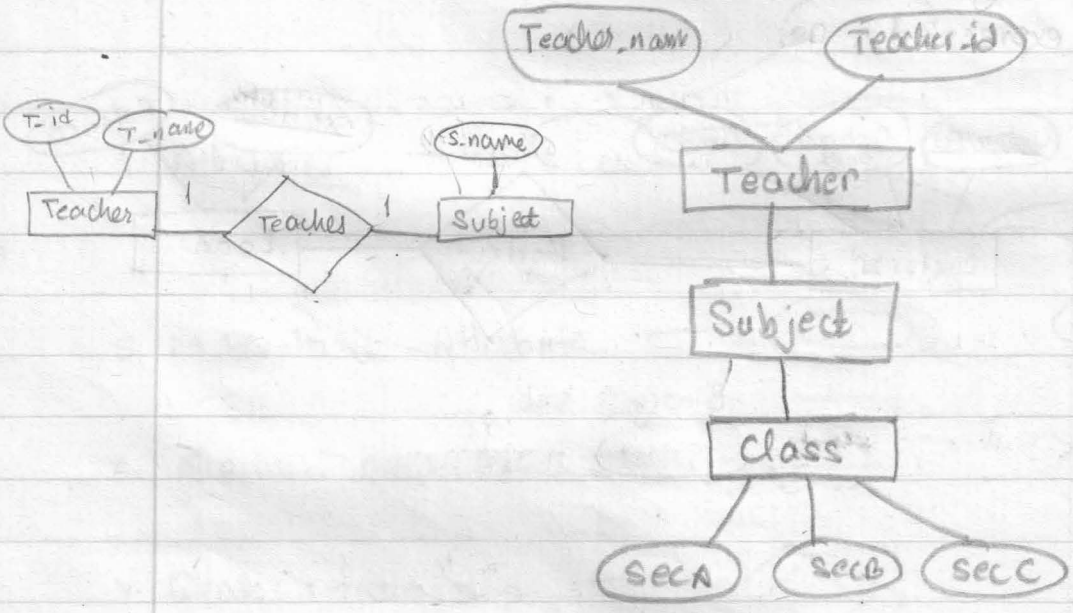
The relationship set borrower having two entity sets customers and loans can be expressed as




Binary set

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

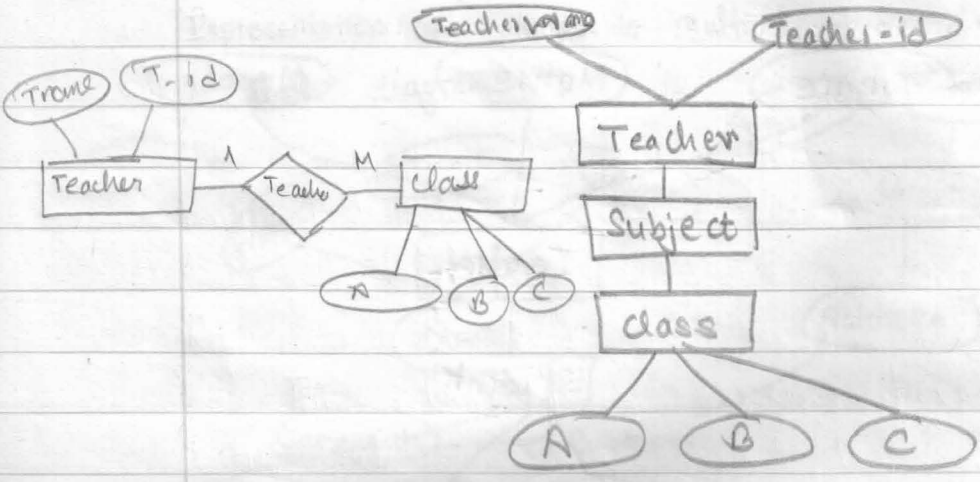
The directed lines in E-R diagram are used to specify mapping cardinalities of relationship sets. The directed line (⟶) tells "one" and undirected line (—) tells "many" between the relationship sets and entity sets.




E-R diagram of one-to-one

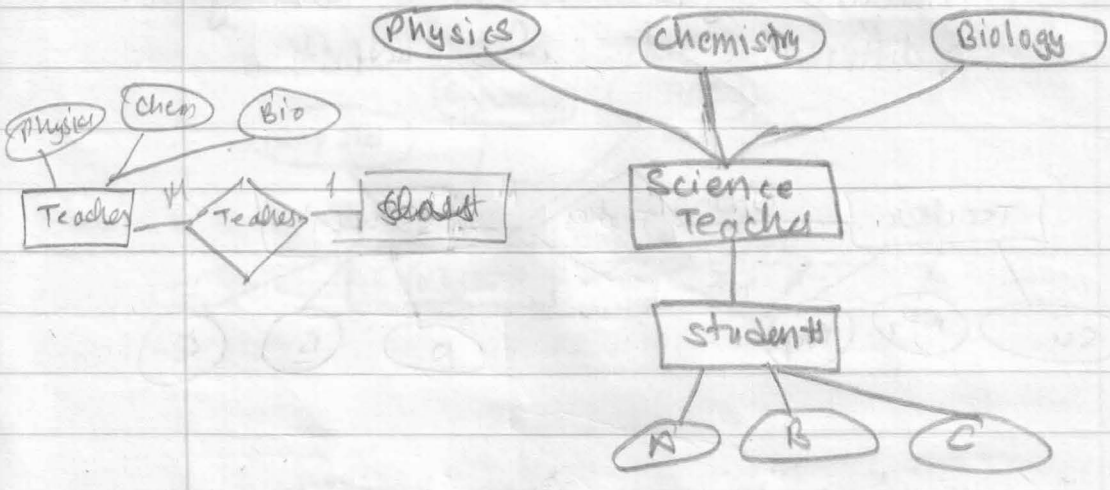
One teacher teaching one particular subject to the students

Homework




E-R diagram of one-to-many

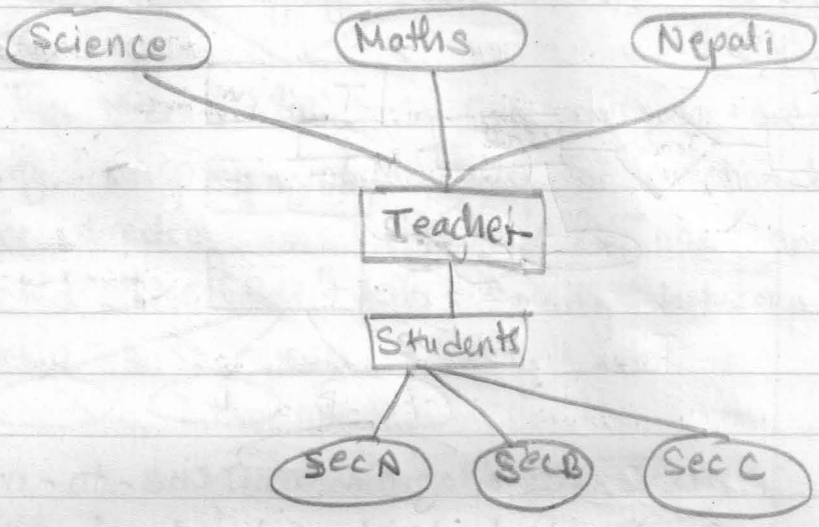
A subject teacher teacher to different section of a class



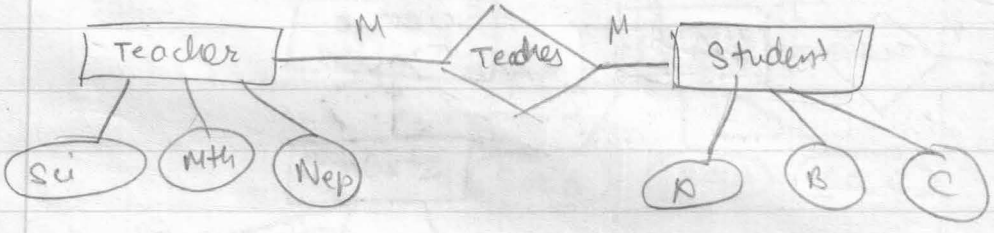

E-R diagram of many-to-one

Different science teachers teaching the same set of students

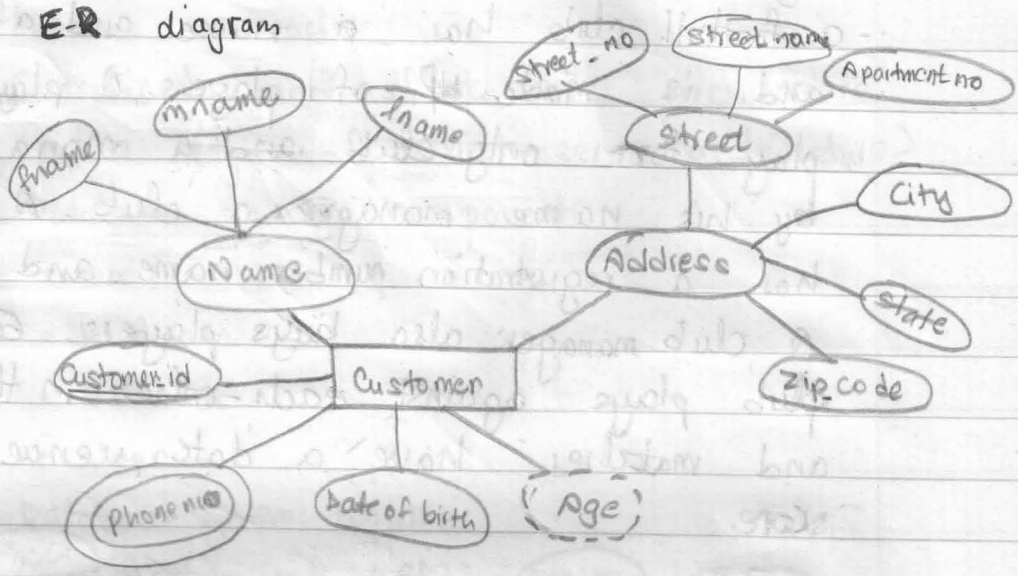
Homework



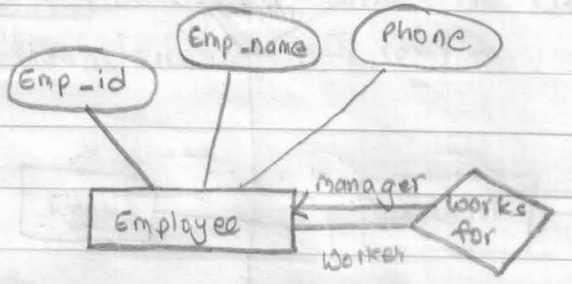

E-R diagram of many-to-many
Different subject teachers teaching to different sets of students



Representation of composite, multivalued and derived attributes in E-R diagram

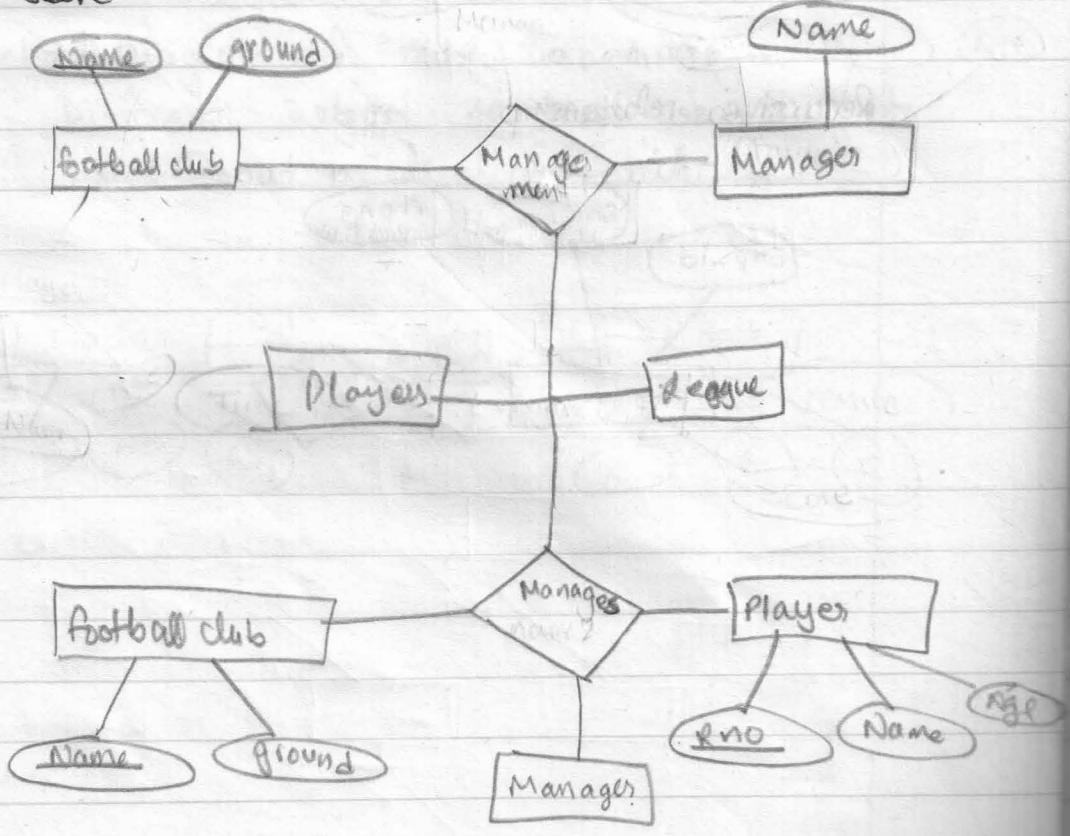



Recursive relationship



Design choice for conceptual database

- a football club has a name and a ground and is made up of players. A player can play for only club and a manager, represented by his name manages a club. A footballer has a registration number, name and age. A club manager also buys players. Each club plays against each-other in the league and matches have a date, venue and score



07-09-14

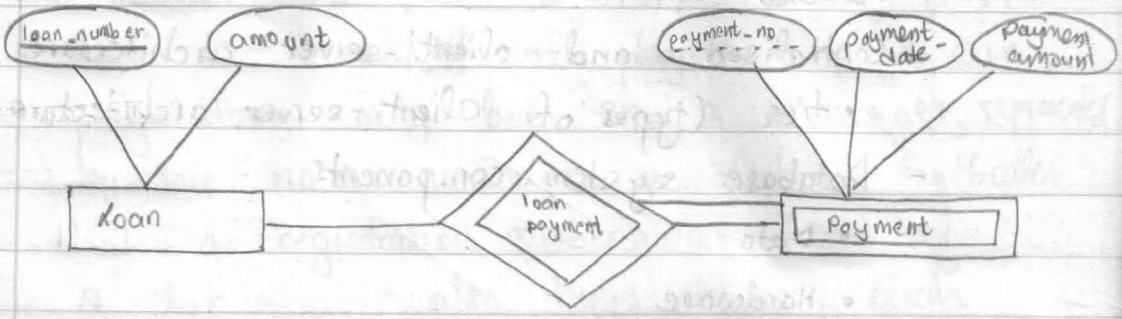
Assignment #2

- Database approach
- Centralised and client-server architecture
  - tier (types of client-server architecture)
- Database system components
  - Data
  - Hardware
  - Software
  - User
- Database system utilities
- Classification of DBMS
- Database system lifecycle
- E-R diagrams with one case study
- Characteristics of relation.

Specialization - top down design approach of DBMS

09-09-14

## Weak entity sets and their representation in ER diagram




E-R diagram of weak entity set

**Specialization**

→ top-down design approach of DBMS

**Generalization**

→ bottom-up design approach of DBMS

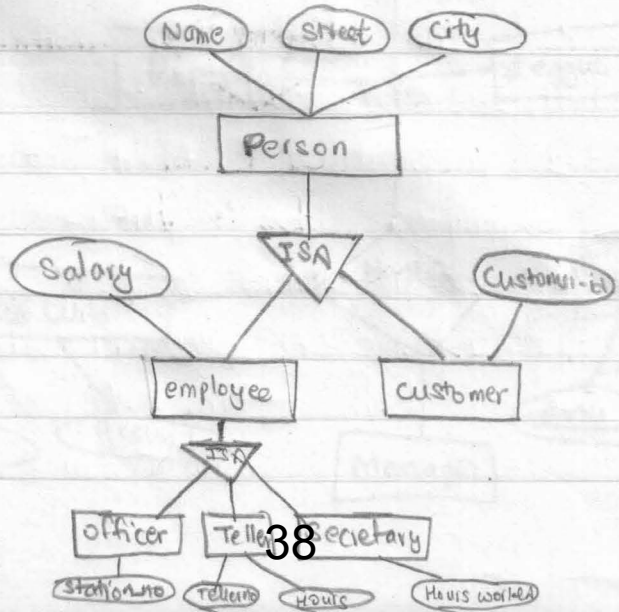



Fig Specialization and generalization

12-09-14

## Aggregation

E-R model cannot 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 manage particular job/caste performed by particular employee at particular branch.

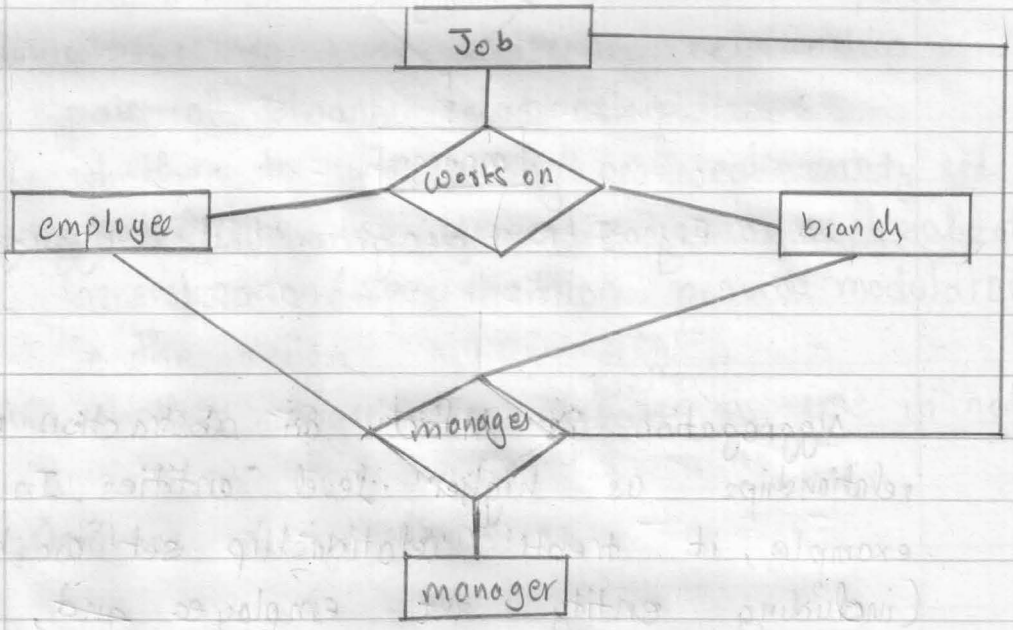



Fig. E-R diagram with redundant relationship

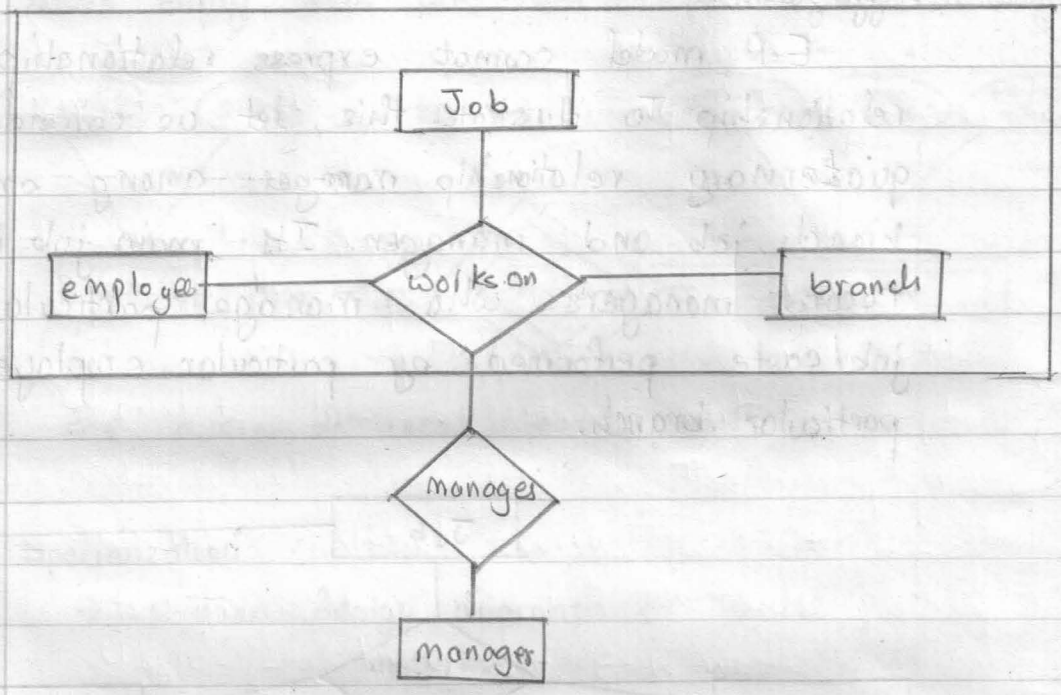



Fig. E-R diagram with aggregation

Aggregation is infact an abstraction it treats relationships as higher level entities. In our example, it treats relationship set "works-on" (including entity set employee ~~and~~, branch and job) as a entity set. So, now we can create binary relationship set "manages" between "work-on" and managers. This reduces redundant information.

## 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 take following decisions:

i. whether to use an ~~entity~~ attribute or entity set to represent an object.
ii. whether a real world concept is based to express an entity set or relationship set.
iii. whether to use a ternary relationship or pair of binary relationship.
iv. whether to use strong or weak entity set.
v. whether to use generalization or specialization. Generalization / specialization provide modulatory in the design
vi. whether aggregation ~~greater~~ better to use or not.

## Database Design Phase

a. User specification requirements

In this 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 these tasks.

b. Conceptual Design

In this phase, database designer need to choose appropriate data models to translate the requirement into conceptual schema of the database. The conceptual design describes detail overview of enterprise. E-R model can be used 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 describe kinds of operations that can be performed on data.

c. Logical design phase

In this phase, database designer need to map the high level of 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.

Database design for a company database

Here, we consider only a few aspects of the company in order to illustrate the process of database design.

1. Data requirements/ analysis

Major characteristics of the company database:

The company database keeps track of a company's employees, departments and projects. Suppose that after the requirement collection and analysis phase, the database designers provide the

following descriptions of the part of the company to be represented by the database.

i. The company is organized into departments. Each department has a unique name, a unique number and a particular manager who manages the department. A department may have different locations.

ii. A department controls a number of projects, each of which has a unique name, a unique number and a single location.

iii. We store each employee's name, id_no, address, salary, sex and date of birth. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of direct supervisor of each employee.

iv. We want to keep track of the department of each employee for the insurance purpose: we keep each ~~departments'~~ dependents' first name, sex, birth date and relationship to the employee

~~employee~~

2. Entity set designation / Conceptual database design

i. An entity type department with attributes name, number, location, manager and managers_start_date, location is a multi-valued attribute. Name and number both are key attributes.
Projects with attribute name, number, location, controlling department, both name and number can be key attributes.

ii. Employee with attributes ~~SSN~~ SSN, name, sex, address, salary, dob, department and supervisor, name and address can be composite attributes.

iii. Dependent with attributes employee, dependentname, sex, ~~DOB~~ DOB and relationship

| Department | Projects | Employee | Dependent |
|---|---|---|---|
| name | Name | SSN | Employee |
| number | number | name | dependentname |
| location | location | sex | sex |
| manager | controllingdepartment | address | dob |
| Managerstartdate | | salary | relationship |
| | | dob | |
| | | department | |

3. Relationships

i. 'Manages', ~~manages~~ a 1:1 relationship type between employee and department. Employee participation is partial, department participation is not clear from the requirements. The attribute start_date is assigned to the relationship type.

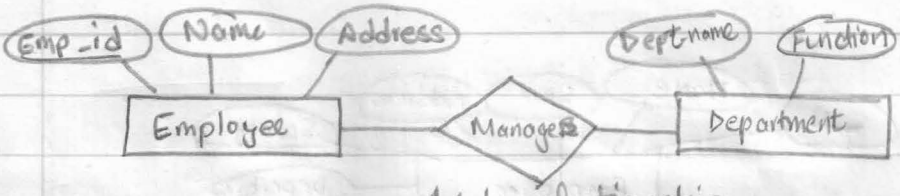
ii. 'works for', a 1:N relationship department, ~~emp~~ and employee, both participation being total.

iii. 'controls', a 1:N relationship between department and project (department: project). Participation of project is total, department is partial.

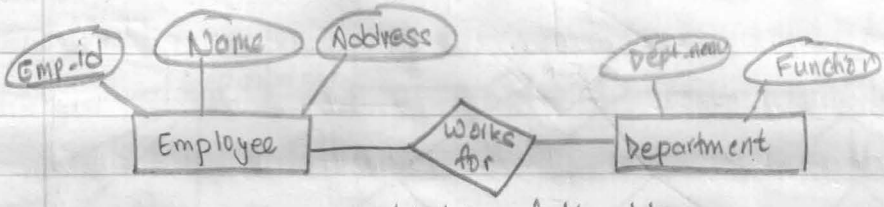
iv. 'supervision', a 1:N relationship between employee (in supervisor role) and employee (in supervisee role). Both participations are partial.

v. 'works on', a M:N relation~~ship~~ with attributes hours between employee: project, both total.

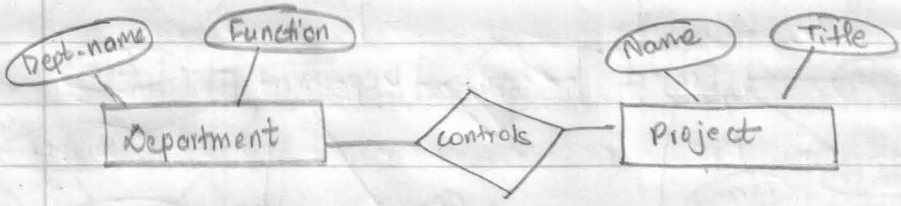
vi. 'dependence_of', a 1:N relationship between employee and department, employee participation is partial, dependent participation is total.



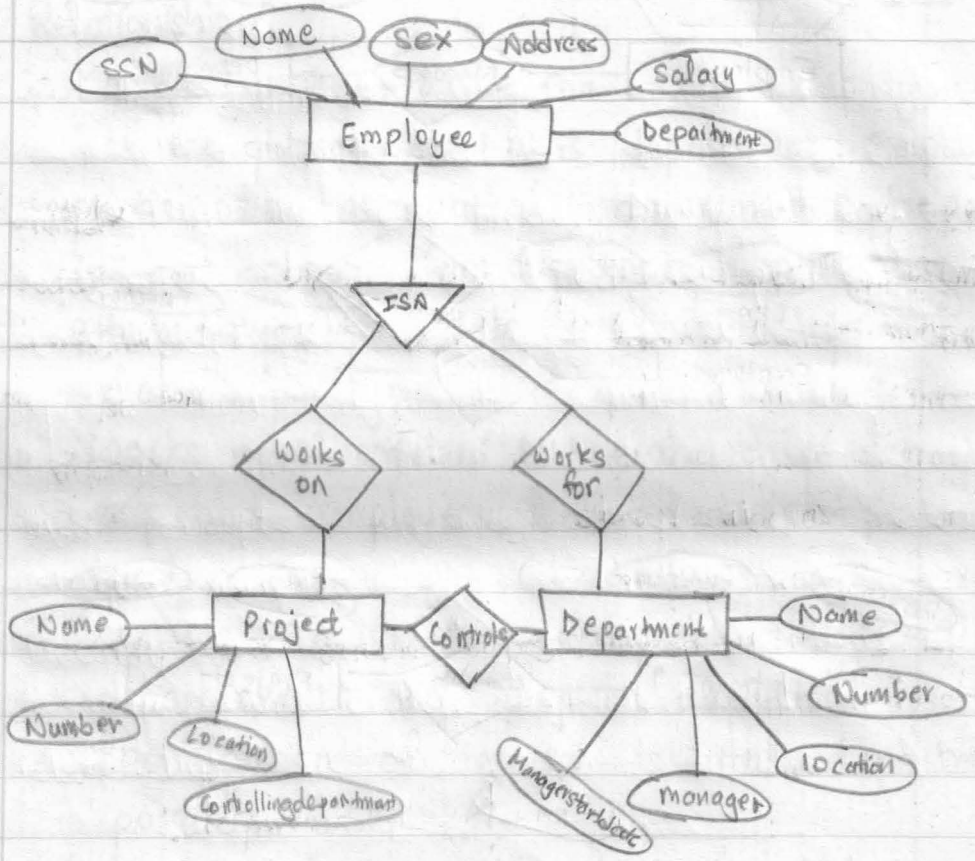

1 : 1 relationship

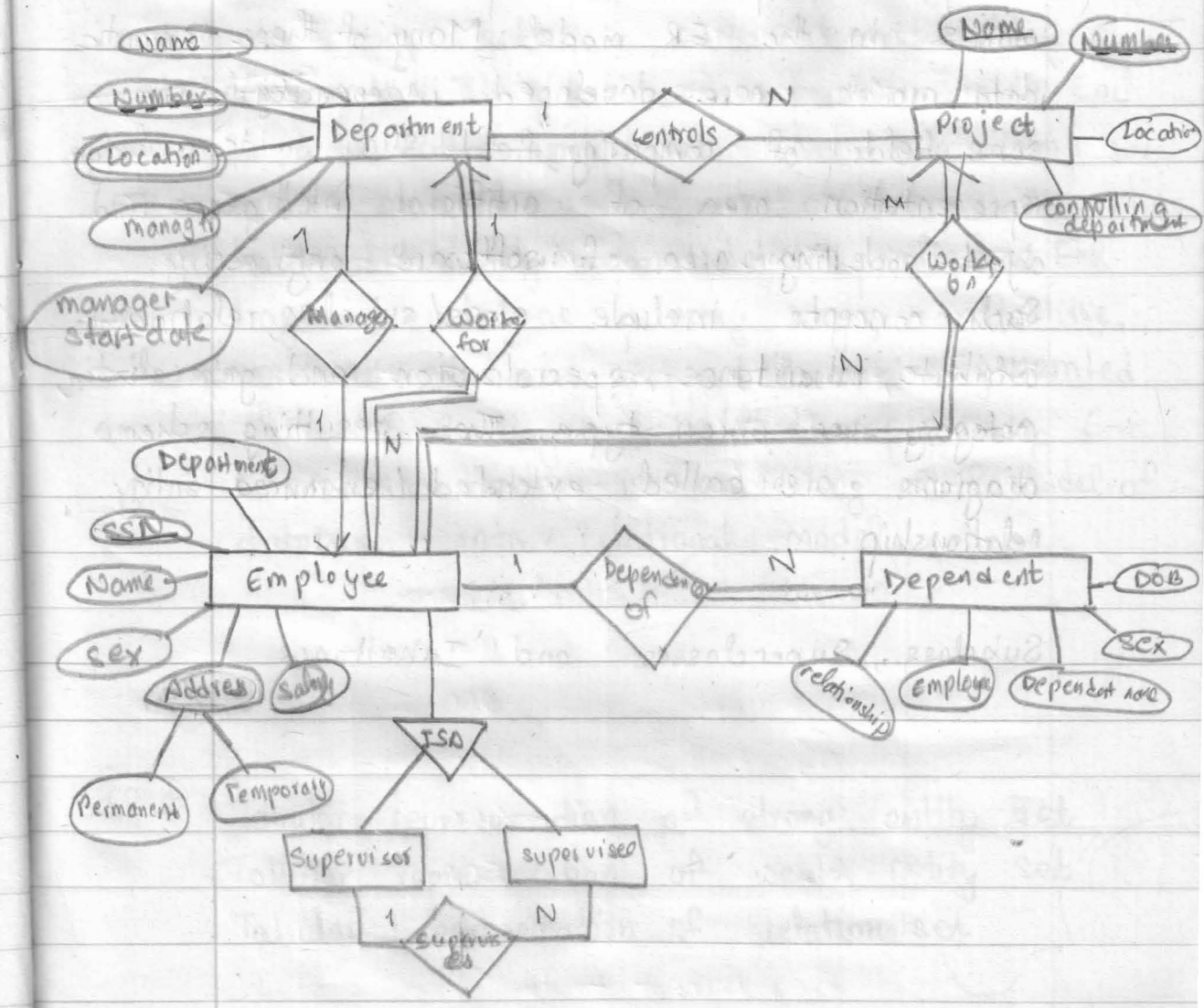



1 : N relationship




1 : N relationship


SSN
Name
Sex
Address
Salary
Employee
department
ISA
Works on
Works for
Name
Project
Controls
Department
Name
Number
Location
Controllingdepartment
Number
Location
Managerstartdate
manager


Name
Number
Location
Department
1
controls
N
Name
Number
Location
Project
manager
manager start date
1
1
Manages
Works for
M
Works on
N
1
N
Department
SSN
Name
Sex
Address
Salary
Employee
1
Depends on
N
Dependent
DOB
Sex
relationship
Employee
Dependent name
ISA
Permanent
Temporary
Supervisor
supervised
1
N
Supervises

14-09-'14

## Enhanced / Extended Entity Relationship

The enhanced entity relationship (EER) model attempts to input the concept of semantic data models in the ER model. Many of these semantic data models were developed independtyly in some fields of computing, ~~field~~ such as knowledge representation area of artificial intelligence and object-modelling area of software engineering. Such concepts include class/ subclass, relationship, attribute, inheritance, specialization and generalization, category or union type. These resulting schema diagrams are called extended / enhanced entity relationship.

## Subclass, Superclasses and Inheritance

## Reduction of an E-R schema to tables

We can represent the E-R database schema by a set of tables. Each entity set and eat entity sets in E-R schema can be represented by their corresponding tables. Each attributes of entity set and relationship set are mapped 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.

Tabular representation of strong entity set
Tabular representation of weak entity set
Tabular representation of relation set

read and write yourself

Tabular Representation of strong Entity Sets

Strong entity set represented by a table with all attributes of it. Let E be a strong entity set with attributes $a_1, a_2, \ldots, a_n$. Then, it is represented by a table E of ~~n distance~~ a n distinct columns, each of which corresponds to one of the strong entity set F. Each row of this table corresponds to entity of entity set F.

For example, entity set account with account number and balance can be represented by table "account" as

| acc no | balance |
|---|---|
| $a_1$ | 100 |
| $a_2$ | 200 |
| $a_3$ | 300 |
| $a_4$ | 400 |
| $a_5$ | 500 |

Fig. Account table

Tabular Representation of weak entity sets

Let A be a weak entity set with attributes $\{a_1, a_2 \ldots a_n\}$. Let B be a strong entity set on which A depends on. Let primary key of B consists of attributes $\{b_1, b_2 \ldots b_n\}$. Now, weak entity set A can be represented by table A with attributes $\{a_1, a_2, \ldots a_m\} \cup \{b_1, b_2 \ldots b_n\}$

For eg: Let us consider weak entity set 'payment' with attributes : payment_number, payment_date and payment_amount. Consider bank as a strong entity set on which weak entity set payment depends upon and suppose that loan_number is a primary key of loan entity set. Now, weak entity set payment can be represented with attributes:

loan_no, pay_no, pay_date and pay_amt

| Loan_no | Pay_no | Pay_date | Pay_amt |
|---|---|---|---|
| 118 | 4 | 25-07-2010 | 2000 |
| 210 | 12 | 05-03-2011 | 5000 |
| 314 | 20 | 30-05-2012 | 1000 |
| 180 | 30 | 15-10-2013 | 4000 |

Fig. The payment table

Tabular Representation of Relation sets

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

Example: Consider a relationship set "borrower" that involves entity sets:

i. customer with primary key customer_id

ii. loan with primary key loan_no

Since relationship set does not consist of any descriptive attribute, relationship set, borrower can be represented by table borrower with attributes customer_id and loan_no.

| customer_id | loan_no |
|---|---|
| $c_1$ | $l_1$ |
| $c_2$ | $l_2$ |
| $c_3$ | $l_3$ |
| $c_4$ | $l_4$ |

15-09-14

Lab #2

1. Insert Statement

Syntax: Insert into <table_name>

E.g. Insert into person
values (001, 'Ram', 'Patan')

2. Column into value

Syntax: Insert into person (id, fname, phone)
values (002, 'Shyam', 5558881)

19-09-14 3. Update statement

Syntax: update table_name
set column1 = value, column2 = value...
where some_column = some_value

E.g: Update teachers
set mname = "krishna"
where fname = "Bal krishna" and id = 1
& id = 1

16-09-'14

# Relational Model RELATIONAL MODEL

Introduction:

- The relational data model represents the database as a collection of table
- There is a direct correspondent between a concept of a table and the mathematical collection of a relation.
- Database in a relational model is simply a collection of one or more relations, where each relation is represented by table with rows and columns. It allows simple high level language to query data.

Basic Structure

A relational database consist of a collection of tables, i.e. in this model, entity sets and relationships all are represented by tables. A row in a table represents a relationship among a set of values. Since a table is a collection of such relationships there is a close correspondence between the concept of a table and the

mathematical concept of a relation from which the relational data model takes its name.

1. Consider the following elements of the sets Name, SSN, Address, Age, GPA

Name = { Dick Davidson, Barbara Benson, Charles Cooper }

SSN = { 422-11-2320, 523-69-1238, 489-22-1100 }

Address = { 3452 Elgin Rd, 265 Lark Lane }

GPA = { 3.53, 3.55, 2.89 }

Age = { 25, 19, 28 }

cardinality ↓

| Name | SSN | Address | Age | GPA |
|---|---|---|---|---|
| Dick Davidson | 422-11-2320 | 3452 Elgin Rd | 25 | 3.53 |
| Barbara Benson | 523-69-1238 | [NULL] | 19 | 3.55 |
| Charles Cooper | 489-22-1100 | 265 Lark Lane | 28 | 2.89 |

← attributes degree →

Row representation of data in table = tuples

Column representation of data in table = cardinality

Name of the table = relation name

Name of the field = domain name

The values of the field = domain

21-09-14

## Database Schema

We must differentiate between the database schema which is the logical design of the database and a database instance, which is a snapshot of the data in the database at a given instant in time.

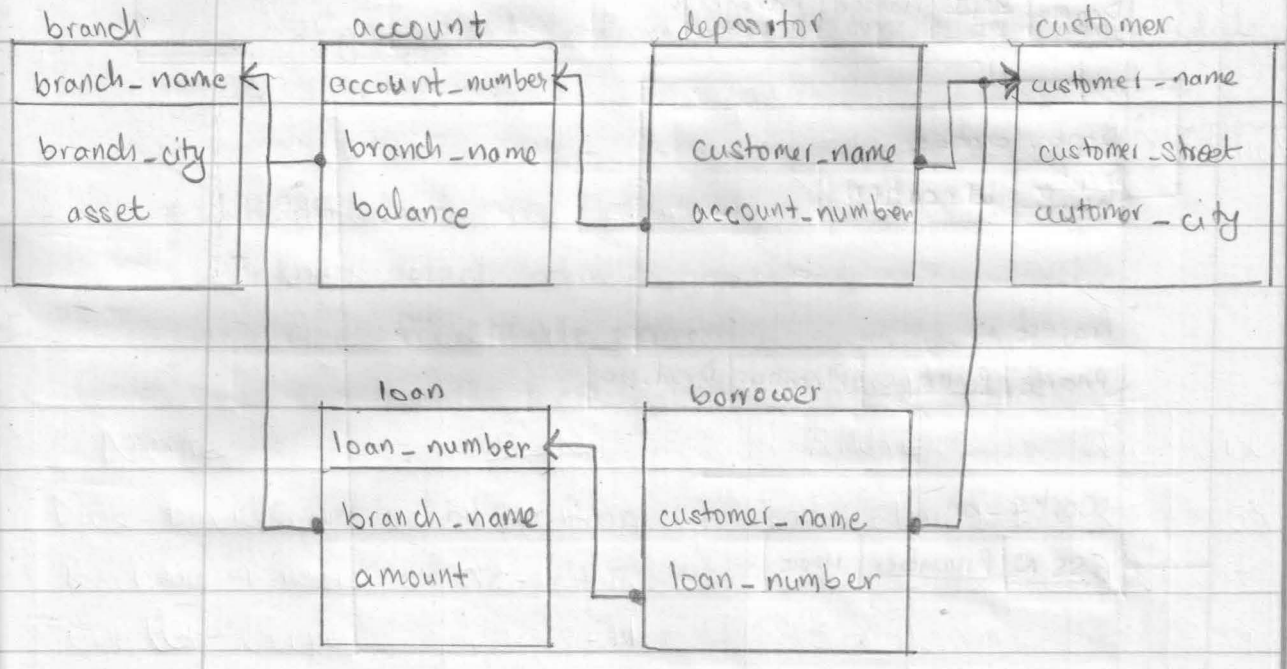
The concept of ~~schem~~ relation corresponds to the programming language notation of variable, most precisely a structure variable, and a concept of relational schema corresponds to the ~~typ~~ notation of type definition or structure definition.

## Schema Diagram

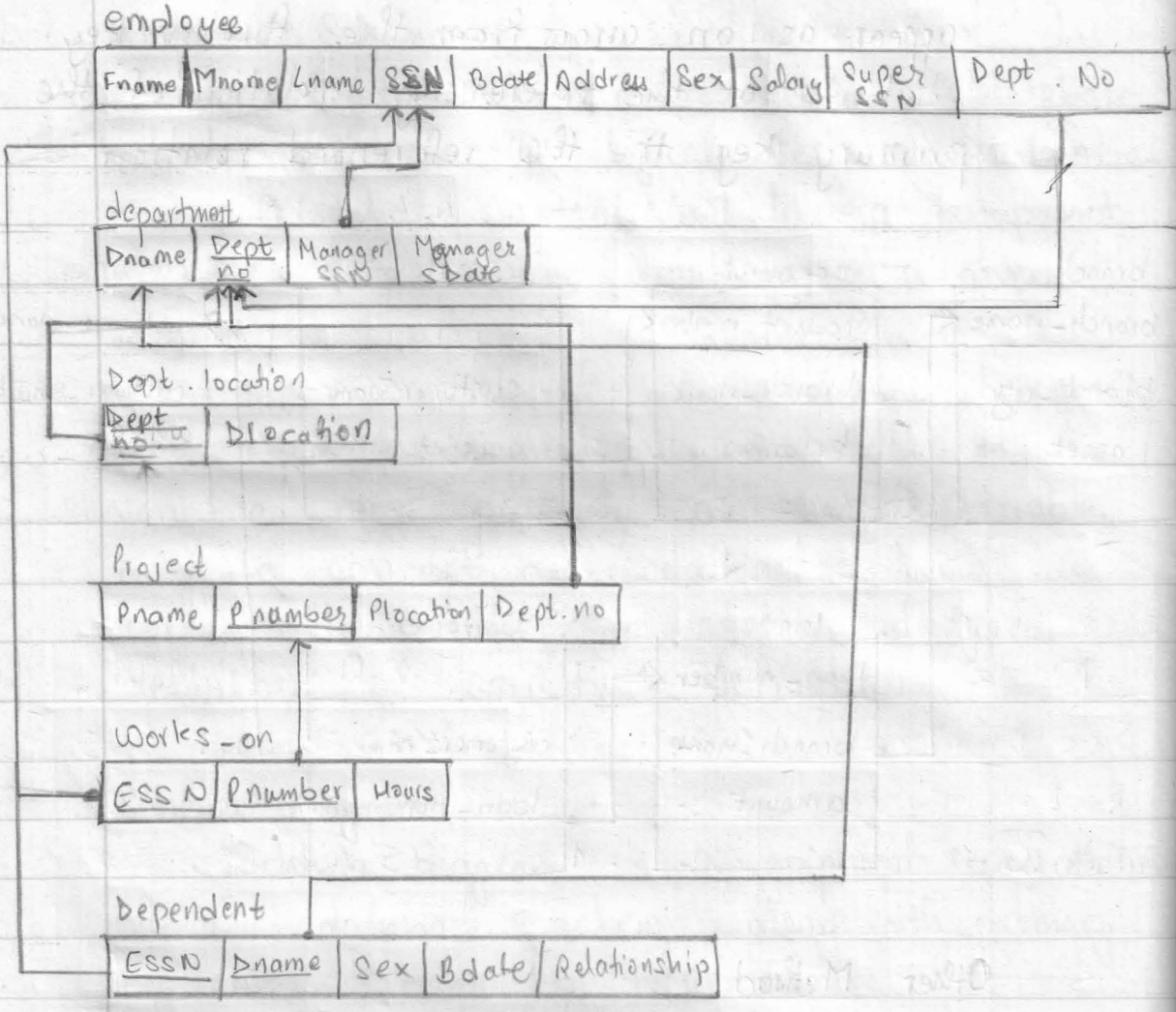
Schema diagram is a graphical representation of the database schema along with primary key and foreign key dependencies.

In schema diagram, each relation is represented by box where attribute are listed inside box and relation ~~name~~ name is specified above it. Primary key in relation is placed above the horizontal line that ~~is~~ crosses the the box. Foreign key in schema diagram

appear as an arrow from the foreign key attributes of the referencing relation of the primary key to the referenced relation




Other Method


employee
Fname
Mname
Lname
SSN
Bdate
Address
Sex
Salary
Super SSN
Dept. No
department
Dname
Dept no
Manager SSN
Manager sDate
Dept. location
Dept no.
Dlocation
Project
Pname
Pnumber
Plocation
Dept. no
Works_on
ESSN
Pnumber
Hours
Dependent
ESSN
Dname
Sex
Bdate
Relationship

22-09-'14

Lab # 4

* SELECT Statement

Syntax: Select * from <table_name>

Eg: select * from person

to select the whole table information

* WHERE clause

Syntax: select column_name/s
from table_name
where column_name operator value

Eg: select * from person
where city = 'Ktm'

* AND, OR Operator

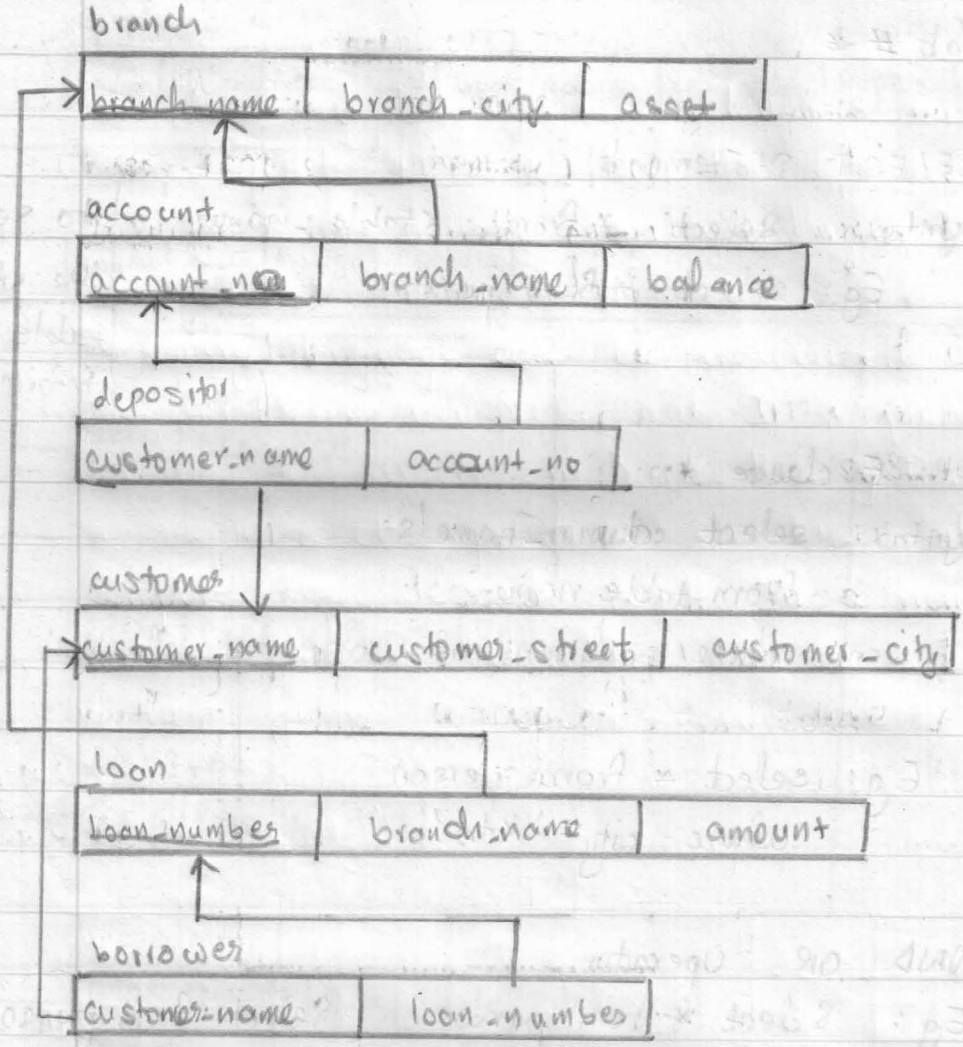
Eg: Select * from person
where fname = 'Ram'
AND Lname = 'Chadka'

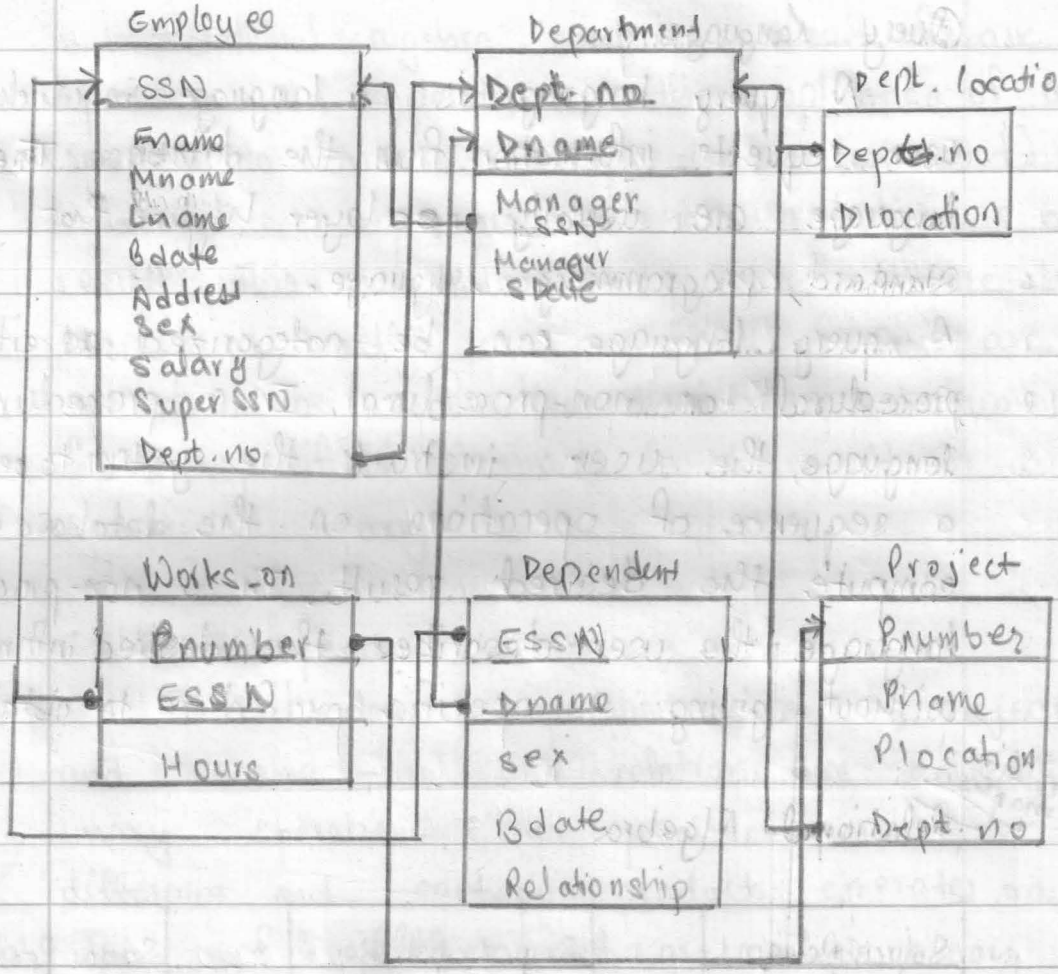
Select from person
where fname = 'Ram'

Select * from person
where fname = 'Ram'
OR lname = 'shyam'

select fname, lname, sex
from person
where address = 'Ktm'

25-09-24


branch
branch_name
branch_city
asset
account
account_no
branch_name
balance
depositor
customer_name
account_no
customer
customer_name
customer_street
customer_city
loan
loan_number
branch_name
amount
borrower
customer_name
loan_number


Employee
SSN
Fname
Mname
Lname
Bdate
Address
Sex
Salary
Super SSN
Dept. no
Department
Dept. no
Dname
Manager SSN
Manager Sdate
Dept. locations
Dept. no
Dlocation
Works_on
Pnumber
ESSN
Hours
Dependent
ESSN
Dname
sex
Bdate
Relationship
Project
Pnumber
Pname
Plocation
Dept. no

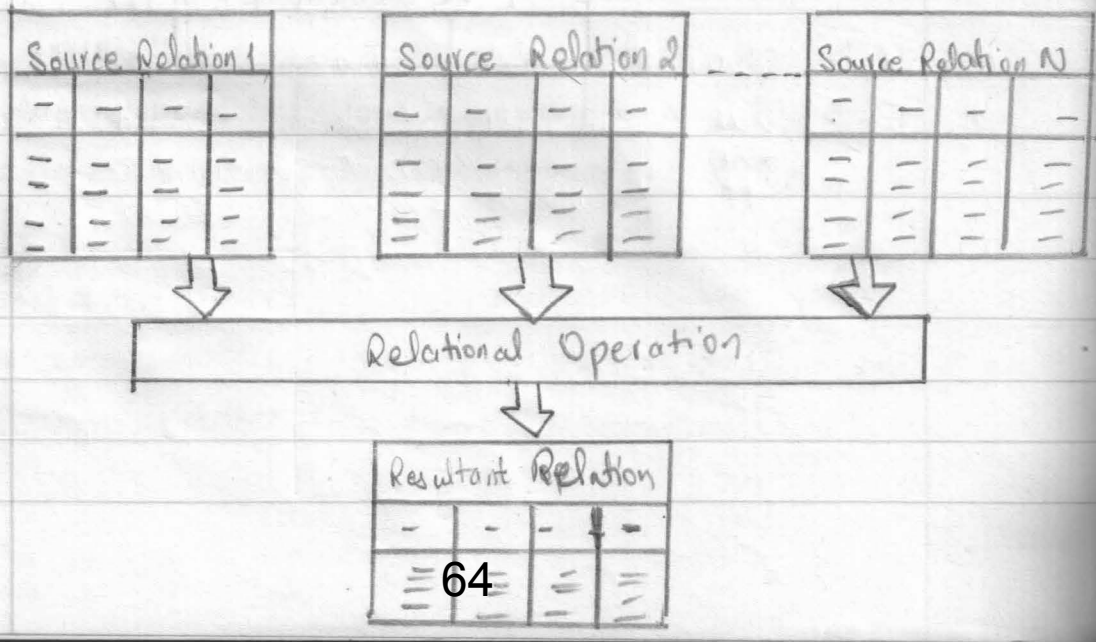
## Query Languages

A query language is a language in which a user requests information from the database. These language are usually on a layer higher that of a standard programming language.

A query language can be categorized as either procedural or non-procedural. In a procedural language, the user instruct the system to perform a sequence of operations on the database to compute the desired result. In a non-procedural language, the user describes the desired information without giving a specific procedure to obtain it.

10-15 marks

## Relational Algebra



Relational algebra comprises a set of basic operations. An operation is the application of an operator to one or more source (or input) relations to produce a new relation as a result. This is illustrated in the figure above. More abstractly, we can think of such an operation as a function that maps arguments from specified domains to a result in a specified range.

08-10-'14 Fundamental Operations

The fundamental operations selection, projection and rename on one relation are called unary operations. Other operations union, set difference and cartesian product operates on pairs of relations and are called binary operations.

The relation employee

| ENo | Ename | Address | Salary | Job. Status | Age | Spr. No. | Dept No. |
|---|---|---|---|---|---|---|---|
| 101 | Sagar | Bhaktapur | 3500 | Research Fellow | 28 | 102 | 10 |
| 107 | Rashmi | Kathmandu | 4200 | Office Ass. | 24 | 102 | 20 |
| 102 | Namrata | Bhaktapur | 5000 | Secretary | 23 | 112 | 30 |
| 112 | Sachin | Kathmandu | 9000 | Administrator | 25 | 107 | 20 |
| 109 | Supraj | Pokhara | 3700 | Office Ass. | 30 | 112 | 10 |
| 115 | K. Singh | Lalitpur | 8500 | Professor | 45 | 113 | 20 |
| 111 | Sharad | Pokhara | 12000 | Director | 35 | 115 | 40 |
| 113 | Bidur | Chitwan | 4500 | Research fellow | 28 | 115 | 40 |
| 116 | Narendra | Biratnagar | 9000 | Professor | 49 | 107 | 10 |
| 103 | Pooja | Pokhara | 4500 | Office Ass. | 24 | 107 | NULL |

The relation Projects

| PNo | PName | SDate | Location | Pmgr | Dept. No |
|---|---|---|---|---|---|
| M110 | Mars | 03-Dec-2007 | Kathmandu | 102 | 10 |
| J220 | Jupiter | 03-Jan-2008 | Pokhara | 107 | 30 |
| V208 | Venus | 15-Oct-2007 | Hetauda | 112 | 40 |
| E005 | Earth | 01-Feb-2008 | Butwal | 252 | 50 |

The relation Work-In

| ENo | PNO | P-Job | Hours |
|---|---|---|---|
| 101 | M110 | Coordinator | 3 |
| 107 | V208 | Engineer | 4 |
| 102 | M110 | Typist | 5 |
| 112 | E005 | Accountant | 9 |
| 107 | M110 | Scientist | 3 |
| 112 | V208 | Coordinator | 8 |
| 111 | M110 | Engineer | 2 |
| 107 | E005 | Scientist | 4 |
| 102 | V208 | Scientist | 9 |
| 107 | J220 | Scientist | 2 |

The relation Department

| DeptNo | Dname | Location | DMgr | Mgr Sdate |
|---|---|---|---|---|
| 10 | Store | Kathmandu | 101 | 03-Dec-2006 |
| 20 | Research | Bhaktapur | 112 | 07-Feb-2007 |
| 30 | Sales | Lalitpur | 102 | 05-Jan-2008 |
| 40 | Marketing | Pokhara | 111 | 26-Dec-2007 |
| 50 | Administration | Lalitpur | 222 | 01-Apr-2005 |

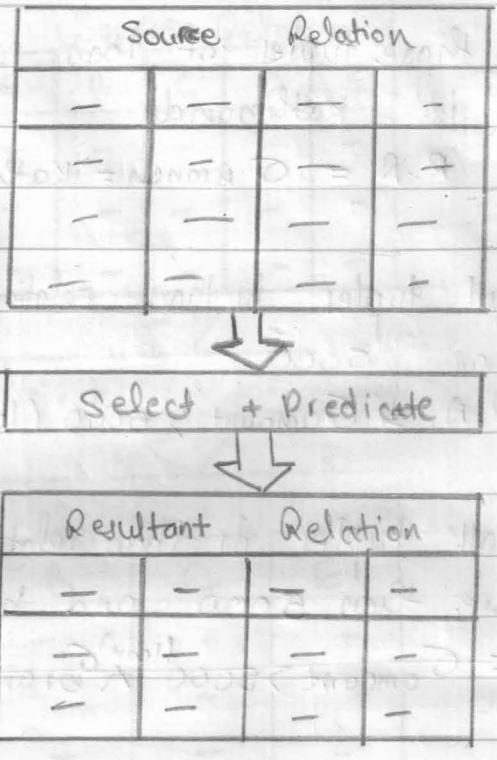
* Selection ($\sigma$) → Sigma

Selection is used to extract tuples from a relation. A tuple from a source relation is selected based on whether it satisfy a specified predicate (condition). A predicate is a truth value expression involving tuple component values and their relationships. All tuples satisfying the predicate are then collected into the resultant relation.

Selection is denoted by a lower case greek letter sigma ($\sigma$).

Let R be a relation of an cardinality and arity.

The selection of a relation R, denoted by $\sigma_F(R)$, is the set of specified tuples of the relation R. 'F' is the predicate (condition) to extract the required tuples. The result of a selection operation is a horizontal subset of R.




List all the employees who live in the city Pokhara

$$R.A = \sigma_{Address = \text{"Pokhara"}}(employee)$$

| ENo | EName | Address | Salary | Job. Status | Age | Spr. No | Dept No |
|---|---|---|---|---|---|---|---|
| 109 | Supraj | Pokhara | 3700 | Office Ass. | 30 | 110 | 10 |
| 111 | Sharad | Pokhara | 12000 | Director | 35 | 115 | 40 |
| 103 | Pooja | Pokhara | 4500 | Office Ass. | 24 | 107 | NULL |

<> Not equal to ! equal to

$\wedge$ AND

$\vee$ OR

NOT $\neg$

Q. Select those tuples of loan relation where the branch is 'Kathmandu'

$$R.A = \sigma_{Branch = "Kathmandu"}(Loan)$$

Q Find all tuples in loan relation in which amount is more than 5000

$$R.A = \sigma_{amount > 5000}(loan)$$

Q. Find all tuples in loan relation where amount is more than 5000 and branch is kathmandu

~~$R.A. = \sigma_{amount > 5000}^{(loan)} \wedge \sigma_{branch = "kathmandu"}(loan)$~~

$$R.A. = \sigma_{amount > 5000 \ \wedge \ branch = "kathmandu"}(loan)$$

* Projection ($\pi$) → Pi

A projection operation extracts specified columns of a relation. The desired columns are simply specified by name. With the help of this operation, any number of columns can be omitted from a table or columns of a table can be rearranged.

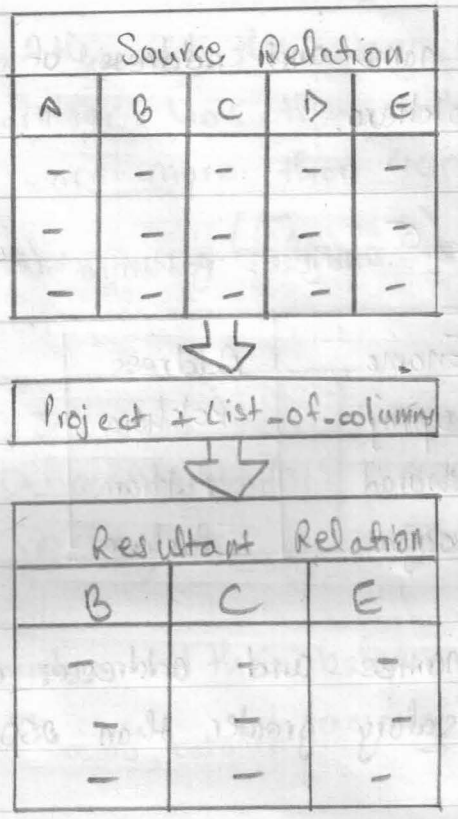



Fig. Projection Operation

Let R be a relation of any arity and cardinality.

DEF: Projection

The projection of R, denoted by $\Pi_{A,B...}(R)$ is the set of specified attributes A, B... of the relation R, i.e., a vertical subset of R.

11-10-'14

Print the name and address of all employees who live in city Pokhara.

$\Pi_{Ename, address}(\sigma_{address = "Pokhara"}(employee))$

| Ename | Address |
|---|---|
| Supraj | Pokhara |
| Sharad | Pokhara |
| Pooja | Pokhara |

List all names and addresses of all employees who has salary greater than 2500.

$R.A = \Pi_{Ename, address}(\sigma_{salary > 2500}(employee))$

| Ename | Address |
|---|---|
| Sagar | Bhaktapur |
| Rashmi | Kathmandu |
| Namrata | Bhaktapur |
| Sachin | Kathmandu |
| Supraj | Pokhara |
| K. Singh | Lalitpur |
| Sharad | Pokhara |
| Bidur | Chitwan |
| Narendra | Biratnagar |
| Pooja | Pokhara |

List the job title and the salary of those employees whose supervisor has the number 102 or 112 and who earn more than 4000.

$$R.A = \Pi_{Jobstatus, salary}(\sigma_{(spr.no = 102 \vee spr.no = 112) \wedge (salary > 4000)}(employee))$$

| Job status | Salary |
|---|---|
| Office Assistant | 4200 |
| Seretary | 5000 |

Find account number and their balance from account relation

$$R.A = \Pi_{acc\_no, balance}(account)$$

Find the customers who stay in Kathmandu.

$$R.A = \Pi_{cus\_name}(\sigma_{address = "kathmandu"}(customer))$$

* Set Union

The union operation requires that involved relation must have the same attributes, that is, the relations are union compatible. Note that it does not restrict that the column headings of two tables has to be identical but the columns ~~the~~ must share the same domain.

The union of relation $R_1$ and $R_2$, denoted: $R_1 \cup R_2$, is a set of tuples, with arity n, that are in $R_1$ or $R_2$ or both.

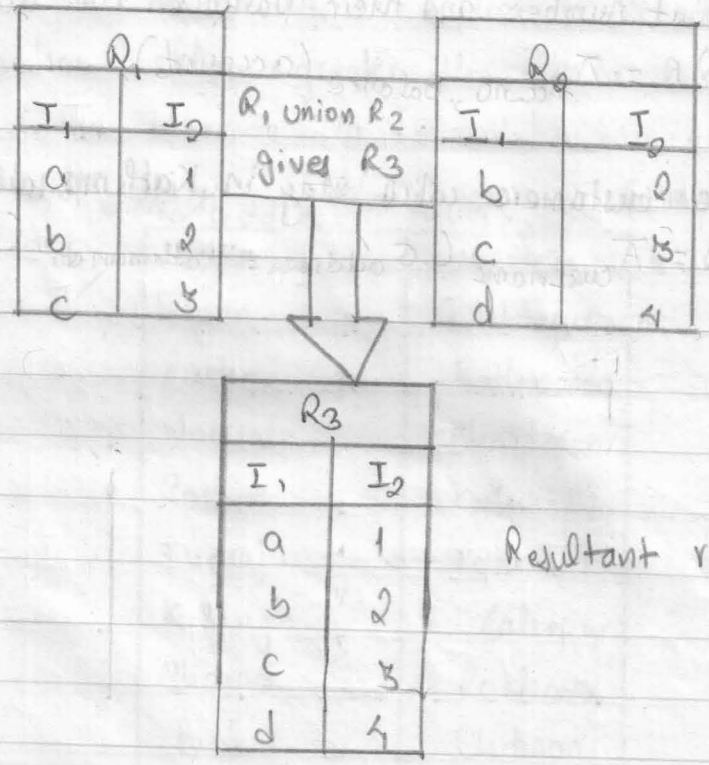



Fig. Union operation

List name and address of the employee who are working on department 10 or live in Pokhara or both.

R.A. = $\Pi_{ename, address}(\sigma_{deptno=10 \cup address="Pokhara"}(employee))$

| Ename. | Address |
|---|---|
| Sagar | Bhaktapur |
| Supraj | Pokhara |
| Sharad | Pokhara |
| Narendra | Biratnagar |
| Pooja | Pokhara |

List name and address of all employee who are working in department 10 or has age greater than 30.

R.A. = $\Pi_{ename, address}(\sigma_{deptno=10 \cup age>30}(employee))$

| Ename. | Address |
|---|---|
| Sagar | Bhaktapur |
| Supraj | Pokhara |
| K. Singh | Lalitpur |
| Sharad | Pokhara |
| Narendra | Biratnagar |

List all employee number who works in project with project number V208 or E005 or both

$$R.A. = \Pi_{Eno}(\sigma(Pno = "V208" \cup Pno = "E005")(Works\_in))$$

| Eno |
|---|
| 107 |
| 112 |
| 100 |
| 10[illegible] |
| 102 |

* Intersection

The intersection operation also requires that the two involved relation must be union compatible. Let $R_1$ and $R_2$ be two relations which are union compatible.

The intersection of relations $R_1$ and $R_2$, denoted $R_1 \cap R_2$, is the set of tuples that are in both $R_1$ and $R_2$

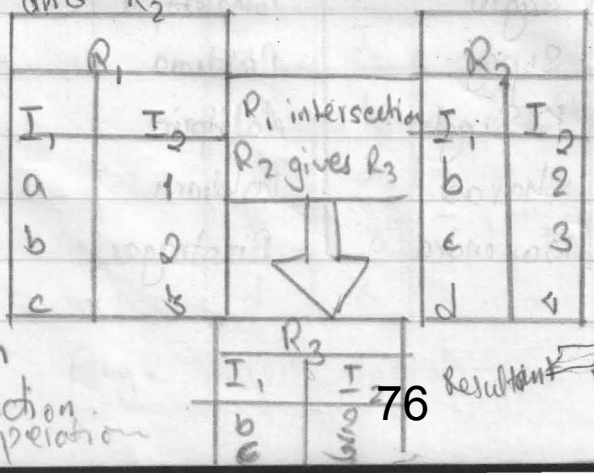



Fig. Intersection operation

Relation

Resultant relation

List name and address of employee who are working on department 10 and live in Pokhara.

$R.A. = \Pi_{ename, address}(\sigma_{(dept\ no = 10 \wedge address = "Poldhara")}(employee))$

| EName | Address |
|---|---|
| Supraj | Pokhara |

List name and address of all employee who are working in department 10 and has age > 30

$R.A. = \Pi_{ename, address}(\sigma_{(dept\ no = 10 \wedge age > 30)}(employee))$

| Ename | Address |
|---|---|
| Narendra | Biratnagar |

* Set Difference

The difference operations also require that the two involved relation must be union compatible. Let $R_1$ and $R_2$ be two relations which are union compatible.

The difference of relations $R_1$ and $R_2$, denoted $R_1 - R_2$, is the set of tuples that are in $R_1$ but not in $R_2$.

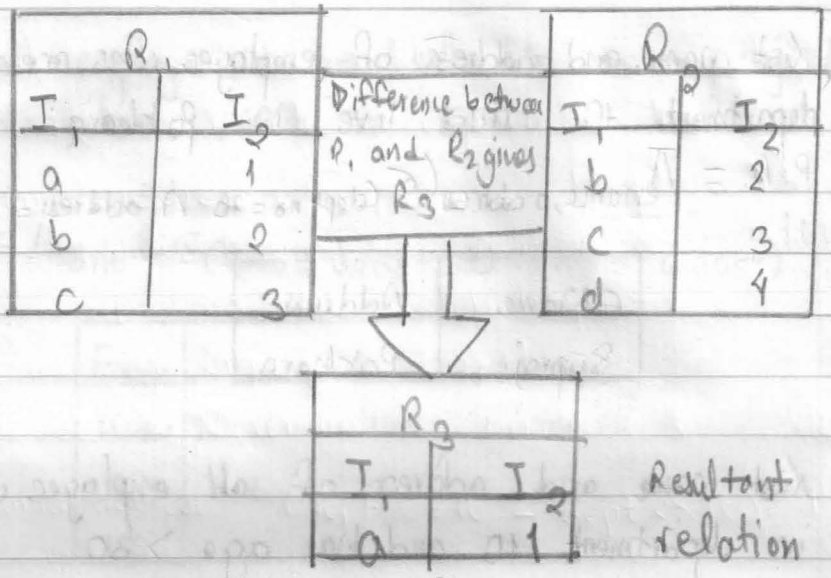



Fig. Relation difference operation

List name and address of all employee who are working on department 10

| Name | Address |
|---|---|
| Sagar | Bhaktapur |
| Supraj | Pokhara |
| Narendra | Biratnagar |

18-10-14

Lab Assignment #3

1. Constraints in SQL
   - Key constraints
   - Attribute constraints
   - Tuple constraints
   - Trigger and assertions
   - ECA rules (Event condition action)
2. Authorization and Privileges
3. Grant and revoke authorization
4. Data encryptions (basic concept only)

15-10-14 List name and address of all employees who are working on department 10 and does not live in Pokhara.

$$\pi_{ename,\ address}(\sigma_{(deptno=10\ -\ address="Pokhara")}(employee))$$

| Name | Address |
|---|---|
| Sagar | Bhaktapur |
| Narendra | Biratnagar |

List name and address of all employee who are working only in department 10

$\pi_{ename, address}(\sigma_{deptno=10}(employee))$

| eName | Address |
|---|---|
| Sagar | Bhaktapur |
| Supraj | Pokhara |
| Narendra | Biratnagar |

$\pi_{ename, address}(\sigma_{(deptno=10 - dept<>10}(employee))$

List name and address of all employees who are working in department 10 and has age greater than 30.

$\pi_{ename, address}(\sigma_{(deptno=10 \cap age>30)}(employee))$

| Name | Address |
|---|---|
| Narendra | Biratnagar |

List all employee no who works in project ~~with~~ project no V208 but not working in E005

$\pi_{Eno}(\sigma_{(pno="V208" - pno="E005")}(Work-in))$

| ENo |
|---|
| 102 |

* Cartesian Product Operation

The cartesian product operation denoted by cross "x". It allows to combine information from any two relations. Cartesian product of two relations r and s denoted by $r \times s$ returns a relation instance whose schema contains all the fields of r followed all fields of s. The result of $r \times s$ contains one-tuple $\langle r, s \rangle$ (concatenation of tuples of r and s) for each pair tuples $t \in r$, $q \in s$.

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

Eg.

| A | B |
|---|---|
| α | 1 |
| β | 2 |

Relation r

| C | D | E |
|---|---|---|
| α | 10 | a |
| β | 10 | a |
| β | 20 | b |
| γ | 10 | b |

Relations

$r \times s$:

| 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 |

Eg:

| Customer_name | Loan_number |
|---|---|
| X | L01 |
| Y | L02 |

Relation borrower

| Loan_number | Branch-name | Amount |
|---|---|---|
| L01 | $B_1$ | 5000 |
| L02 | $B_2$ | 6000 |

Relation loan

Find all customer who have taken loan from branch $B_1$

R.A. = $\Pi_{cname}(\sigma_{(bname = 'B_1')}(loan) \times borrower))$

loan x borrower

$\Pi_{customer\_name}(\sigma_{borrower.loan\_no = loan.loan\_no}(\sigma_{bname='B_1'}(borrower \times$

| Customer_name | borrower. loannumber | loan. loannumber | Branch name | Amount |
|---|---|---|---|---|
| X | L01 | L01 | $B_1$ | 5000 |
| X | L01 | L02 | $B_2$ | 6000 |
| Y | L02 | L01 | $B_1$ | 5000 |
| Y | L02 | L02 | $B_2$ | 6000 |

$\therefore$

| Customer_name |
|---|
| X |

$\sigma_{bname='B_1'}$ (borrower x loan)

| Customer name | borrower loan no | loan loan no | Branch name | Amount |
|---|---|---|---|---|
| X | L01 | L01 | $B_1$ | 5000 |
| Y | L02 | L01 | $B_1$ | 5000 |

$\sigma_{borrower.loanno = loan.loanno}(\sigma_{branchname = "B_1"}(borrower \times loan))$

| | | | | |
|---|---|---|---|---|
| X | L01 | L01 | $B_1$ | 5000 |

∴

| Customer_name |
|---|
| X |

loan)))

## * Join

The "JOIN" operation is one of the essential operations of relational algebra. It is a binary operation that allows the user to combine two relations in a specified way. Join can be further divided into :

i. theta join ii. Natural join iii. Outer join

### i. Theta-join (θ)

Generally, the join in theta (θ) join, or theta join means that it's result always obeys the general definition of the join operation. Theta is a predicate which consist of a one of the comparision operations $\{=, <, <=, >=, >, <>\}$ and specified the join condition. If join condition, θ is = the operation, the operation is called equi-join

ii Natural join ⋈

The natural join operation is the most common variant of the joins. A natural join is an equijoin, i.e. its θ consist of an =; however, in addition, we drop the duplicate attributes resulting in an arity less than that of the equi-join.

The natural join operation requires that the two operand relations must have atleast one common attribute, i.e. attributes that share the same domain.

iii. Outer join

The outer join operation is an extension of the join operations to deal with missing information. For theta join and natural join, where some of the tuples in the joined relation do not have identical values. For common attributes, those tuples will be lost in the result. To retain all the information of both relation, it is desirable to have a joiner which keeps the tuples having no corresponding values in both relations associated with NULL values. This is the external join or outer join.

There are three types of outer join:
- left outer join
- right outer join
- full outer join

Left Outer join

The left outer join takes all tuples in the left relation that did not match with any tuple in the right relation, pads the tuples with NULL values for all other attributes from the right relation and adds them to the result of natural join.

Right Outer join

The right outer join takes all tuples in the right relation that did not match with any tuple in the left relation, pads the tuples with NULL values for all other attributes from the left relation and adds them to the result of natural join.

Full Outer join

The full outer join does both of those operations, padding tuples from the left relation that did not match with any from the right relation, as well as

tuples from the right relation that did not match any from the left relation, and adding them to the result of join.

## Modification of Database

We express database modifications by assigning the operation. We make assignments to actual database relation by using the same notation as that described in the fundamental operations.

* Deletion

We can delete only whole tuple. We can not delete values on only particular attributes. In relational algebra, deletion is expressed by

$$R \leftarrow R - E$$

where R is the relation and E is the relation algebra query.

Delete all employees who are working in deptno = 20

$Employee \leftarrow Employee - (\sigma_{deptno = 20}(employees))$

Delete all departments which is located in Chitwan

$Department \leftarrow Department - (\sigma_{address = "Chitwan"}(department))$

* Insertion

To insert data into a relation, we either specify a tuple to be inserted or a query who result is a set of tuples to be inserted. The attribute values for inserted tuples must be members of the attribute domain. Similarly, tuples inserted must be of the correct arity. The relational algebra expresses an insertion by

$$R \leftarrow R \cup E$$

where R is the relation and E is the relation algebra expression.

Eg: Employee ← Employee ∪{129, "Supraj", "Kalimati", 15000, "Director", 28, 50)}

* Updating

~~Inserted~~ In certain situations, we may wish to change a value in a tuple without changing all values in the tuple. We can use the generalized projection operation to do this task.

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

where each $F_i$ is either the $i^{th}$ attribute of R. If the $i^{th}$ attribute is not updated or if the attribute is to be updated, $F_i$ is an ~~operation~~ expression involving only constants and the attributes of R that gives the new value for the attribute

Eg: All employees working in department 20 has increased their salary by 5%.

$$\text{employee} \leftarrow \Pi_{\text{eno, ename, address, salary} + \text{salary} * 0.05, \text{jobstatus, age, deptno}}(\sigma_{\text{deptno}=20}(\text{employee}))$$

All employees with age greater than 25 had 7% increase in salary and all other below 25 had salary 5% increase.

$$\text{employee} \leftarrow \Pi_{\text{eno, ename, address, salary} + \text{salary} * 0.07, \text{jobstatus, age, deptno}}(\sigma_{\text{age} > 25}(\text{employee})) \cup \Pi_{\text{eno, ename, address, salary} + \text{salary} * 0.05, \text{jobstatus, age, deptno}}(\sigma_{\text{age} \leq 25}(\text{employee}))$$

## NULL VALUE

Tuples may not have any values for some of the attributes. At that time, the attribute is said to have null value and is denoted by NULL. It simplifies an unknown value or a value does not exist. The result of any arithmetic or comparison involving NULL is NULL. There are often more than one possible ways of dealing with null values, as a result, our definition can sometimes be arbitary Therefore, arithmetic operations and comparisons on null values should be avoided if possible.

Comparisons involving nulls may occus inside Boolean expression: AND, OR & NOT operations.

## Views

For a security reason, it is not desirable for all users to see the entire logical model of database (i.e. all actual relations stored in database). We may require certain data to be hidden from users. We may wish to create a personalized collection of relations that are better matched to certain users' intuition than whole database.

Eg: An employee is in advertising department might be able to see a relation consisting of customers who have either an account or loan at the bank and the branches with which they do business.

In relational algebra it is expressed by

$$\Pi_{branch\_name,\ customer\_name}(depositor \bowtie account) \cup \Pi_{branch\_name,\ customer\_name}(borrower \bowtie loan)$$

## Tuple relational calculus

Tuple relational calculus is non-procedural query language. It describes desired information without specifying procedure for obtaining that information. The general structure of query in relational calculus is expressed as:

$$\{t \mid P(t)\}$$

It is read as set of all tuples of t such that predicate P is true for t.

20-10-14

Lab #4

1. Alter table

Syntax : ALTER table <table-name>
ADD <Column_name> datatype

Eg : ALTER table person
ADD DOB date

27-10-14 Lab # 5

1. To retrieve all the rows in table, we may use the following statement

SELECT eno, ename, address ...
FROM employee
OR
SELECT * from employee

1. Find the eno and ename of the employee whose salary is greater than 4000.
2. List all employee name with their project name.
3. List name of all employee and their salary who are working in the project Mars or Venus.
4. List name of all employee and their salary who are working in the project Earth and Venus

List name and address of all employees who works in department which is located in Pokhara

28-10-14

# INTEGRITY CONSTRAINTS

Integrity constraints are those constraints in database system which guard against invalid database operations or accidental damage to the database, By ensuring that authorized changes to the database. It does not allow loss of data consistency in database. It ensure database consistency.

In fact, integrity constraints provide a way of ensuring that changes made to the database by authorised users do not result in ~~data~~ a loss of data consistency.

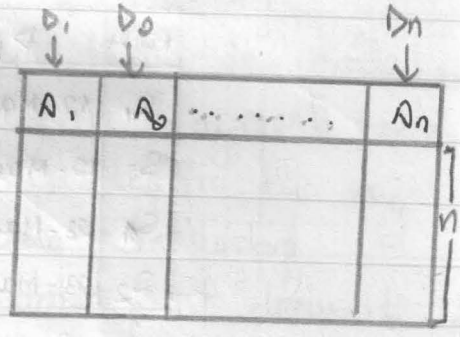
Example of integrity constraint in E-R model:

i. Key declaration: candidate key, primary key
ii. Form of relationship: mapping cardinalities, one-to-one, one-to-many, etc.

In DBMS, we can enforce any arbitary predicate as integrity constraint but it adds overhead to the database system so its cost should be evaluated as far as possible. Integrity constraint should be with minimal overhead.

**Domain Constraints (Pool)**

Set of all possible values for attributes is known as its domain. A domain is a pool of values of the same (data) type from which one or more attributes in one or more tables take their values. Above we can see that attribute $A_1$ draws value from $D_1$, $A_2$ from $D_2$, etc. If two attributes draw values from the same domain then comparisons between tuples can be made on these attributes.




Data is said to contain domain integrity when the value of a column is derived from the domain. Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take.

For eg: Column data of joining must be a valid data. All valid datas from one domain.

If the value of ~~data~~ of joining is an invalid data, then it is said to violate domain integrity.

## Referential Integrity

Payments

| Roll No | DP | Amount |
|---|---|---|
| $S_1$ | 12-May-2001 | 1000 |
| $S_2$ | 10-May-2001 | 2500 |
| $S_1$ | 23-May-2001 | 1500 |
| $S_2$ | 23-May-2001 | 1000 |

Students

| Roll no. | Name | Phone |
|---|---|---|
| $S_1$ | Louis Figo | 456333 |
| $S_2$ | Raul | 656675 |
| $S_3$ | RobertoCarlos | 546762 |
| $S_4$ | Cruti | 567345 |

Foreign Key

- In relational model, we often store data in different tables and put them together to get complete information. For example, in payments table, we have only roll no of the student. To get remaining information about the student, we have to use students table. Roll no in payments table can be used to ~~remai~~ obtain remaining information about the student.

- The relationship between entities students and payments is one-to-many. One student can make payments many times as we already have roll no. column in Roll no. column in payments table, it is possible to join with students table and get information about parent entity (student).
- Roll no. column of payment table is called as foreign key as it is used to join payments table with students table. So, foreign key is the key on the many side of the relationship.
- Roll no. column of payment table must derive its value from roll no column of students table.
- Thus, for referential integrity, a foreign key can have only two possible values - either the relevant primary key or a NULL value. No other values are allowed.
- In above payments table, if we add a tuple with roll no 5 means that we have a foreign key that doesn't reference a valid key in the parent students relation. Thus, a violation by an inserting operation. Likewise, if we were to delete 5 from the students relation, we would again have a foreign key in the base or parent relation.

- Thus, the rule for the 'referential integrity' constraint asserts that if a relation $R_2$ inside a foreign key that matches the primary key of another relation $R_1$, that every attributes value of the foreign key in $R_2$ must either
  - be equal to the value of primary key in some tuple of $R_1$ or
  - be wholly NULL

## Referential Integrity in E-R Model

Referential integrity constraints arise frequently. Every relation arising from a relationship set has referential integrity constraints.

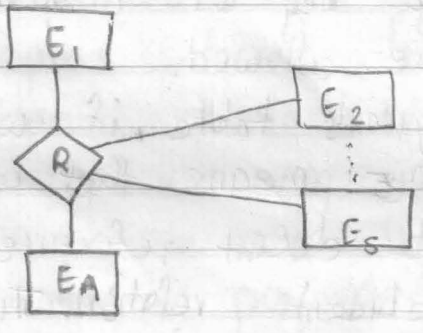



Fig An n-ary relationship

M-a The above figure shows an n-ary relationship set R relating entity sets $E_1, E_2 \dots E_n$

- Let $K_i$ denote the primary key of $E_i$
- The attributes of the logic relation schema for relationship set (R) include

$$K_1 \cup K_2 \cup K_3 \cup \dots \cup K_n$$

- Each $K_i$ in the schema for R is a foreign key that leads to a referential integrity constraints