SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I - Database Management Systems – SCSA1301

I. Introduction to Databases

Databases and Database users – Database system concepts and architecture – Data modeling using entity Relationship(ER) model – Enhanced ER model- Relational Model - The Relational Data Model and Relational Database Constraints - The Relational Algebra and Relational Calculus.

Data means known facts or raw facts. E.g. names, telephone numbers. Information means processed Data.Database is a collection of related Data. E.g. student table consists of name, regno, marks.Database management system (DBMS) is collection of programs that enables user to create and maintain a Database. A general-purpose software system facilitates process of defining, constructing, and manipulating DB for various applications. Database system includes Database and DBMS software.

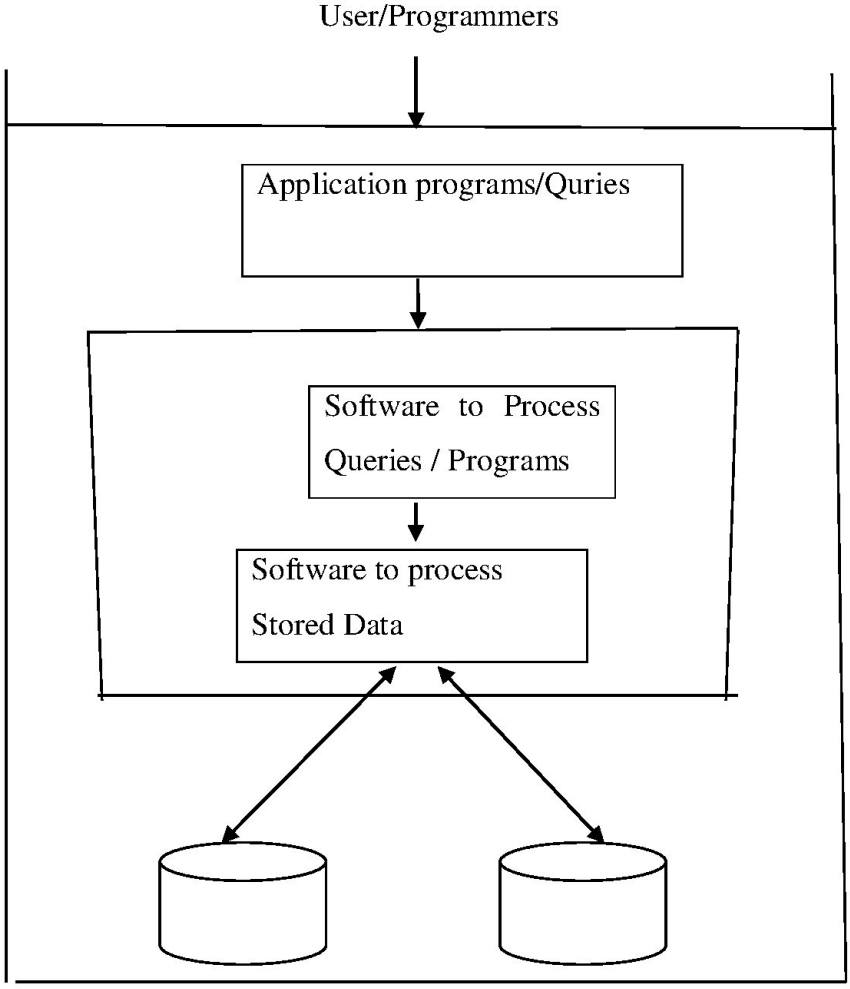


Fig. 1.1 A Simplified Database Environment

Characteristics of DBMS

1. **Self-Describing nature of DB:**

The Database system contains Data and definition of the Database. The Data definition is stored on the system catalog, which contains the structure of the files, Data type for each Data item and various constraints on the Data. The information stored in the catalog is called MetaData.

2. **Insulation between Program, Data, and Data Abstraction:**

In the DBMS system, the structure of the file should be stored separately from the access program so, whenever we modify anything in the DB or access program this will not affect the original structure. We call this property as program-Data-independence.

In object, oriented DB system the operation becomes a part of DB system. This operation consists of two parts called interface and implementation. The interface includes operation name and Data type and implementation represents method of the operation. Thus, the method or the implementation should be change without affecting the interface is called as program-operation-independence.

3. **Support of multiple views of Data:**

The multi-user DBMS can provide a facility for defining a multiple views. The view may be a subset of the db or it may contain the virtual Data, that it is derived from the original db file .so, depends upon the user specification the DBMS will display a various types of views. Example: consider the student table,

Table 1.1. Student Table

NAME	REGNO	ADDRESS	PHONENO	PERCENTAGE

VIEW1

NAME	ADDRESS

VIEW2

REGNO	PERCENTAGE

4. Sharing of Data and multi-user transaction processing:

The DBMS allow the multi-user to access the db at the same time. It must support the concurrency control software that several users trying to update the same Data, the result of the updates should be correct. E.g.: airline ticketing, on line banking, railway reservation.

Actors on the Scene

The person those who are involved on the project and those who are using the Database are called actors.

I. Data Base Administrators (DBA):

- The job of DBA is to manage the db resources.
- DBA is responsible for co-ordinating, monitoring and authorizing access to the db, and to provide whatever hardware, software needed . . Everything should be monitored by DBA.

II DB designer:

- Db designers are responsible for identifying the Data to be stored in the db, as well as they choose the appropriate structure to represent this Data.
- The db designer should communicate with all the users for understanding their requirements.

III End Users:

- The end users are the people whose job requires accessing the db for querying, updating and generating reports.

Types of End-users:

a) Casual End users:

- They occasionally access the db, but they may need different information each time.
- The user sophisticated db query language to specify their requirements.

- Ex: queries like "list the trains from Chennai to Delhi?"

b) Parametric or Naive End users:

- Their job is constantly querying and updating the db using standard types of queries called canned transaction.
- Example: Bank teller, Reservation clerk, etc.

c) Sophisticated End users:

- They have the thorough knowledge about the db.
- Example: Engineers, Scientist, Business analyst etc. They have thorough knowledge about the DBMS.
- They will implement their applications and meet their complex requirements very easily.

d) Stand-alone End-users:

- Maintain the personal db by using the readymade program packages.
- This program packages will provide easy-to-use, menu or graphic based interfaces.

IV System analyst and Application programmers:

- System analyst determines the requirements of end users especially parametric end users.
- They develop a specification for canned transactions that meet their requirements.
- Application programmers translate these requirements into programs then they test, debug, document and maintain these canned transaction. Such programmer's are called s/w engineers.

Workers behind the scene:

- DBMS system designer and implementors.
- Tool developers
- Operators and maintain personnel.

Advantages of DBMS

1. Controlling Redundancy:-

- Redundancy is storing the same Data multiple times The storage space is wasted and makes the db file becomes inconsistent.
- In file processing system, the Data files are stored along with the program files. When a user is wants to create an application, he has to create and maintain separate Data files along with the program files. Because of this, much of the Data is stored more than once. However, in the Database system a single Database is created and stored once and which can be used by different users.

2) Restricting unauthorized access:

- When multiple users share a Database, some users will not be authorized to access all information in the db like some users to read the Data only and some users they permitted to modify the Data also.
- Example: Financial Data base like banking Database, military Dataetc are accessed only by authorized person.
- These securities must be provided by Security and authorization subsystem in DBMS.

3) Providing persistent storage for program objects and Data structure:

Databases can be used to provide permanent storage for program objects and Data structures.

4) Providing multiple user interfaces:

- Different users have the different knowledge to use a db so, the DBMS should provide a variety of interfaces such as,
    - Query language for casual end users
    - Programming language for application programmers
    - Forms and commands for parametric end-users
    - Menu-driven interfaces for stand-alone end-users.

The forms-commands and menu-driven interfaces are called as *Graphical userinterfaces (GUI).*

5) Permitting inferencing and actions using rules:

Some Data base systems provide capabilities for defining deduction rules for finding new information from stored Database. Such systems are called deductive Databases.

6) Representing complex relationship among Data:

- The Database may include varieties of Data that are related to each other.
- The DBMS has the capability to represent the relationships among these different Data's.

7) Enforcing Integrity Constraint:

- DBMS should specify a set of rules or restrictions for defining the Data in the db.
- Example:

  Name must be a string of no more than 30 characters.

  The key field should not be null.

8) Providing backup and recovery:

- DBMS must provide facilities for recovering from h/w or s/w failures.
- The backup and recovery subsystem of the DBMS is responsible for recovery process.
- Example for updating the complex Data, at the middle computer system fails then the recovery system is responsible for restoring a state and starts the point at which it was interrupted.

Database System Concepts and Architecture

Data models:Data model is a collection of concepts that can be used to describe the structure of Database.

Categories of Data models:

1) High level or conceptual Data model:-

- Conceptual DM provides concepts that explains the different ways to perceive Data and uses the concepts such as entities, attributes and relationships.
- Entity represents the real world object, for example employee or project.
- Attributes represents the properties or the further description of entity. For example employee name or salary.
- Relationship represents the interaction among the entities. For example works-on relationship between employee and project.
- Ex: entity relationship model

2) Low level or Physical DM:-

- This will provide the concept of how the Data is stored in the computer
- The storage format is also specified in this Data Model such as, record format, record ordering and access path.

3) Representational or Implementation DM:-

- This is the intermediate DM between high level and low level.
- It provides the concepts that may be understood by end users but that are not too far removed from the way Data is stored in the computer.
- Ex: relational model, network model, hierarchical model.

Schema or intension:

The description of a Database is called the schema or intension.

Instance or occurrences:

Each row in the Database i.e. a set of related Data's.

Extension or Database state or snapshot:

The Data in the Database at a particular moment is called Database state or extension, which is the current set of instances. At initial state of the Database, the Database state is said to be empty.

Three Schema Architecture

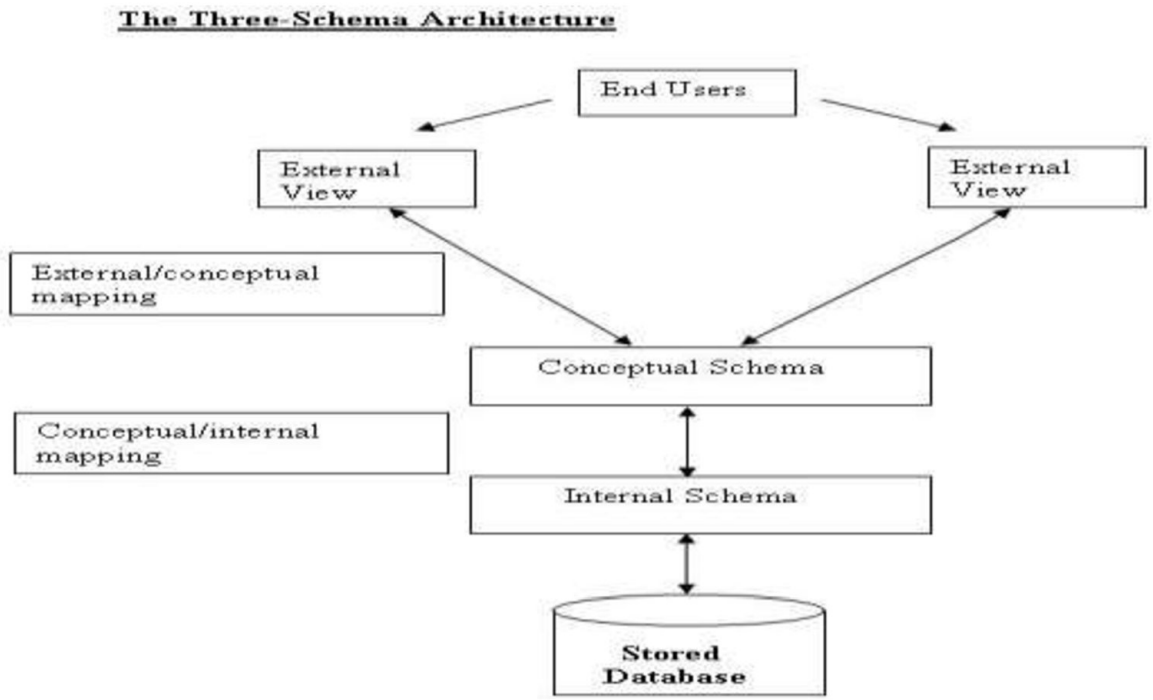


Fig.1.2. Three Schema Architecture

The schemas are defined at three levels

1. Internal level or Physical level or Low level
2. Conceptual level or High level
3. External level or View level

Internal level:

- It has an internal scheme, which describes the physical storage structure of the database by means of different data structures link list, queue, stack etc.
- It uses a Physical data model.
- It is useful for computer scientist.

Conceptual Level:

- It has a conceptual schema, which describes the structure of the whole database.
- It describes data as entities, attributes, & relationships.
- It hides the details of physical storage structures.
- It uses high-level data model or implementation data model.
- It can be understood by end users.

External Level:

- It includes a number of external schemas
- It describes the part of the data base that a particular user group is interested in and hides the rest of the data base from that group
- It uses high-level data model or implementation data model.
- It can be understood by end users.

Most DBMS do not separate the 3 levels completely but support three schema architecture to some extent. Some DBMS may include internal schema details in the conceptual schema.

Mappings:

The three schemas are only descriptions of database. The data is actually stored in the database. If a particular user wants to retrieve a data, he has to place a request in the external level. The DBMS must transform this request specified on the external schema into a request against the conceptual schema and then into a request on the internal schema for processing over the stored database. Hence, the retrieved data is reformatted and sent back to the user through the

external view. Thus, the processes of transforming requests and results between levels are called Mappings.

Data Independence:

It is the capacity to change the schema at one level of a database system without having to change at the next higher level. There are two types.

1. Logical data independence

2. Physical data independence

Logical data independence:

It is the capacity to change the conceptual schema without having to change the external schema. Only the mappings between conceptual and external schema need to be changed.

Physical data independence:

It is the capacity to change the internal schema without having to change the conceptual schema. Only the mappings between conceptual and internal schema need to be changed.

Entity- Relationship Model (E-R Model)

Entity relationship model is a high-level conceptual model, which is useful for end users. An ER model describes data as

- Entities
- Attributes
- Relationships

Entities:

An entity is defined as the real world object or thing that is described in the database. Examples: employee, student, department, project.

Attributes:

Attributes are the properties that describe an entity .For example an employee entity is described by the employee's name , age, address, salary.

A particular entity will have a value for each of its attributes.

Types of attributes:

1. composite versus simple:
2. multi-valued versus single valued
3. Stored versus derived.

Composite attributes can be divided into smaller sub parts which represent more basic attributes with independent meanings.Examples: address, name of an employee

Attributes that are not divisible are called *simple or atomic attributes*. Ex: age, sex

Multi-valued attributes have set of values for the same entity.Example: college degrees attribute for a person, phone numbers.

Single valued attributes have single value for a particular entity. Ex: age.

Derived attributes are derived from related entities (stored attribute). Ex: age attribute is derived from birth date attribute. Age attribute is a derived attribute. And birth date is a *stored attribute.*

Complex attributes are combination of composite attributes and multi-valued attributes. For representing use ( ) for composite and { } for multi-valued.

Example: address

{Address (street address, city, state, pin code)}. Assume that a person can have more than one residence.

Key attributes:

An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute and its values can be used to identify each entity uniquely.For example:

Ssn of an employee entity, regno of a student entity, rollno of a student, dno of a department entity.

An entity types can have more than one key attribute . For student entity regno, rollno both are key attributes that uniquely identifies a student.

Weak entity:

Entity types that do not have key attributes of their own is called weak entity. Example:

Consider the entity type dependent, which is used to keep track of dependents of each employee. The attributes of dependent are name, birth date, sex and relationship. Two dependents of two distinct employees may by chance have the same values for name, birth date, sex, and relationship. Hence, it is difficult to identify a dependent. so weak entities are always related to specific entities called as parent entity type . Dependent entity is always related to employee entity.

Partial key:

A weak entity normally has a partial key, which is the set of attributes that can uniquely identify weak entities. In our example if we assume that no two dependents of the same employee ever have the same name the name attribute is the partial key.

Strong entity:

Entities that do have a key attribute is called strong entity. Example: employee, student, department, project.

Relationships:

Whenever an attribute of one entity type refers to another entity type, some relationship exists between entities.

Degree of relationship:

The degree of a relationship type is the number of participating entity types. In the works_for, relationship that associates the employee and department entity the degree of relationship is two. If the degree is, two it is called as binary relationship and one of degree three is called ternary.

Constraints on relationships:

Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set.

There are two main types of relationship constraints:

1. Cardinality ratio:

The cardinality ratio for a binary relationship specifies the number of relationship instances that an entity can participate in. For examplein the WORKS_FOR binary relationship type, department: employee is of cardinality ratio 1:N. (N stands for any number of related entities) means that each department can be related to numerous employees.

The possible cardinality ratios are 1: N, 1:1, M:N.

2. Participation.

There are two types.

1. Total participation: The participation of employee in WORKS_FOR is called total participation meaning that every entity in the total set of employees must be related to a department entity via WORKS_FOR relationship.
2. Partial participation:

The participation of employee in manages relationship is called partial participation meaning that the company do not expect each and every employee must be related to department entity. Only some or part of the set of employees are related to department via manages relationship.

Attributes of relationships:

Relationships can have attributes: example: the WORKS_ON relationship, which relates employee and project, can have hours attribute to record the number of hours per week that an employee works on a particular project.

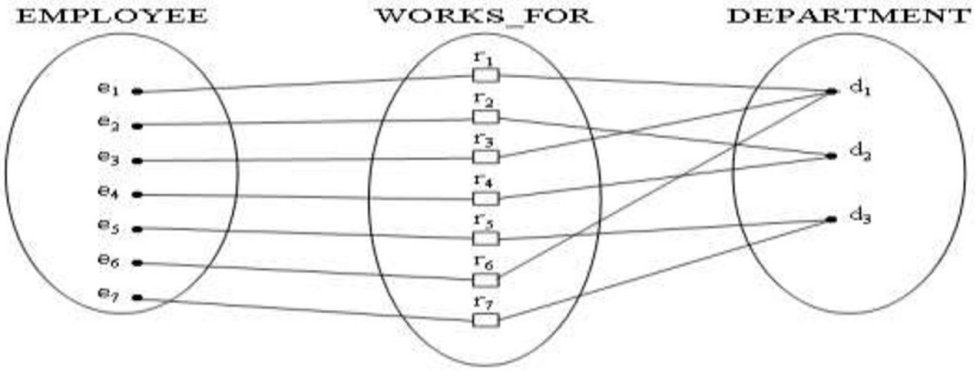
In our company database example, we specify the following relationship types:

1. Manages:

A 1:1 relationship between employee and department. Employee participation is partial.

2. Works_for:

A 1: N relationship between department and employee. Both participations are total.



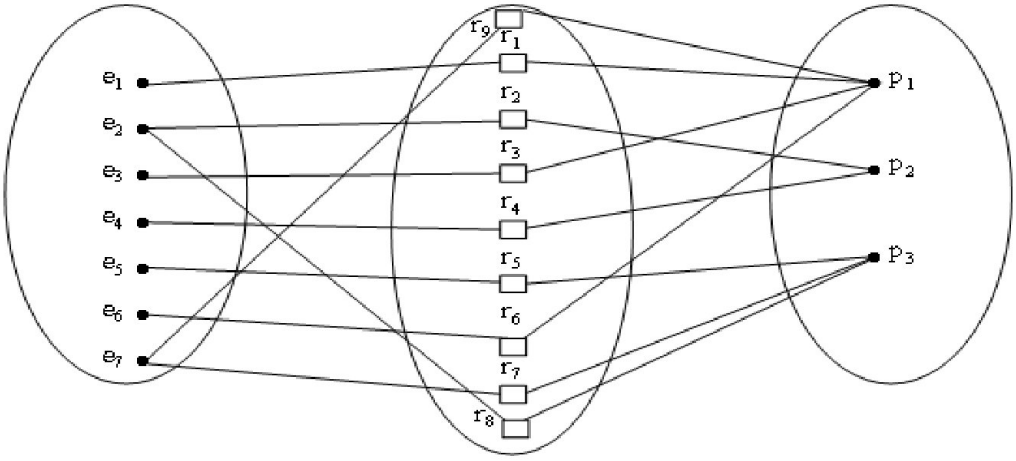
3. Controls:

A 1:N relationship between department and project.

4. Works_on:

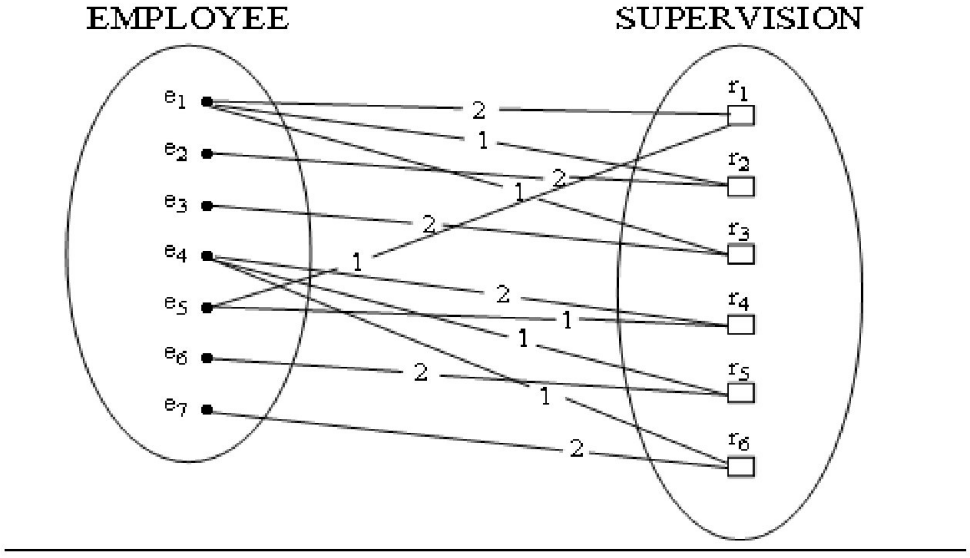
A M:N relationship between employee and project. Employee participation is partial.

EMPLOYEE WORKS_ON PROJECT



5. Supervision:

A 1:1 relationship between employee and employee.it is recursive relationship.

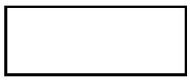


6. Dependents_of:

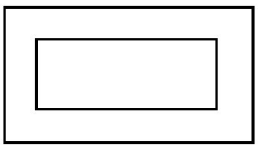
A 1:1 relationship between employee and dependent.

Notations Used In E-R Diagram

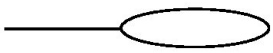
ENTITY



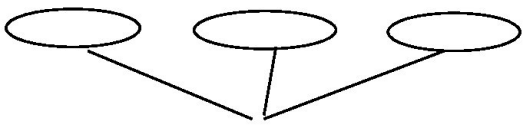
WEAK ENTITY



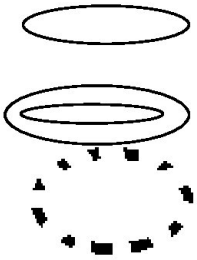
ATTRIBUTE



COMPOSITE ATTRIBUTE



MULTIVALUED ATTRIBUTE

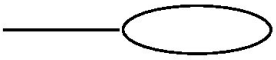


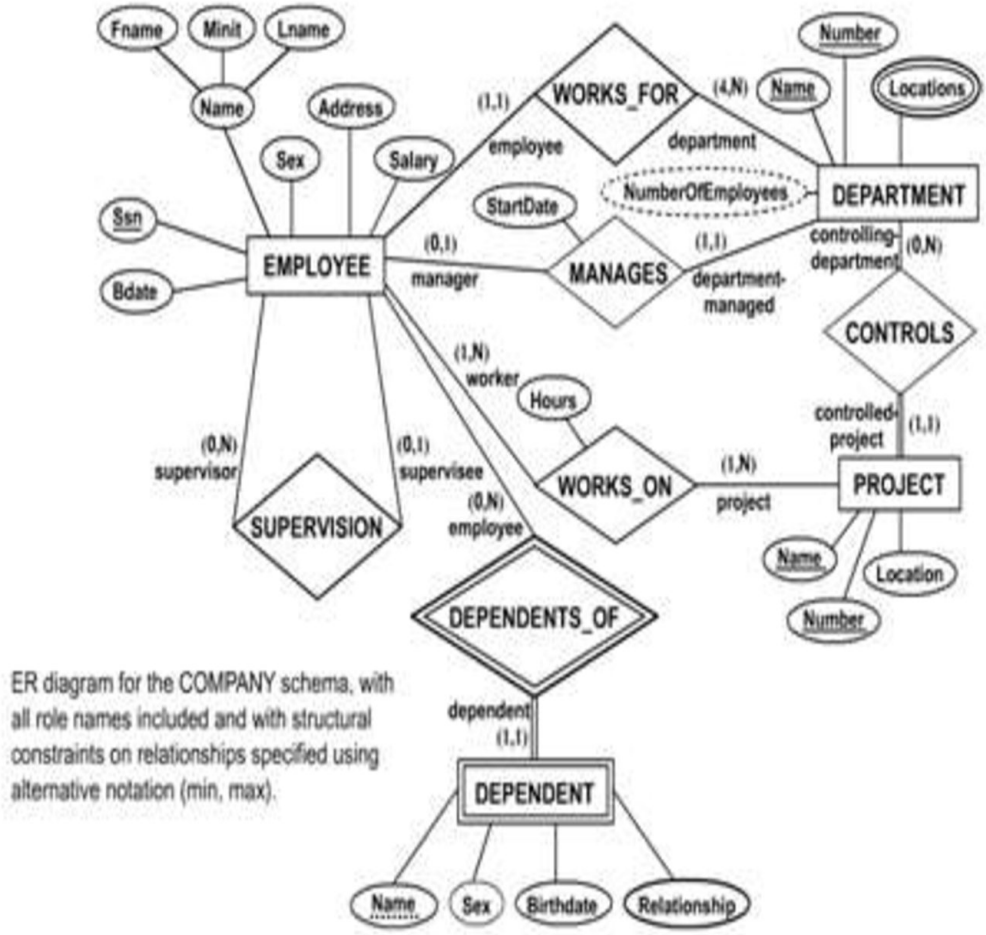
DERIVED ATTRIBUTE

RELATIONSHIP

KEY ATTRIBUTE

PARTIAL KEY ATTRIBUTE





ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

Fig.1.3. E – R diagram for the Company Database

Relational Database Design Using ER-To-Relational Mapping

ER to relational mapping provides an algorithm that can map an entity relationship (ER) schema to the corresponding relational database schema. We will use the company database to illustrate the steps for ER to relational mapping.

STEP1:

✓ For each strong entity, create a relation that includes all the simple attributes of strong entity.

- ✓ For composite attributes include only single component.
- ✓ Choose one of the key attributes of the entity as primary key of the relation.
- ✓ Example: consider the ER model for company database(refer I unit notes)

Strong entities	attributes	Composite Attributes	Key attributes
Employee	Ssn, sex, bdate, salary, address	Name	Ssn
department	Name, number	-	Name, Number
Project	Name, number, location	-	Name, Number

- ✓ Relational model for company database after step1:

EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY

DEPARTMENT

DNAME	DNUMBER

PROJECT

PNAME	PNUMBER	PLOCATION

STEP2:

- ✓ For each weak entity, create a relation that includes all the simple attributes of strong entity.
- ✓ For composite attributes include only single component.

- ✓ Always a weak entity is associated with an owner entity. Include the primary key of the owner entity as the foreign key of weak entity
- ✓ The primary key of the weak entity is the combination of the partial key of the weak entity and the foreign key.
- ✓ In our example:

  Weak entity: dependent; attributes: dependent _name, sex, bdate, relationship

  Owner entity: employee

  Primary key of employee is SSN, which is the foreign key of dependent.

  Rename it as ESSN to avoid confusions.

  Partial key of dependent: dependent _name

  Primary key of dependent: dependent _name + ESSN

- ✓ Relational model for company database after step2:

  EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY

DEPARTMENT

DNAME	DNUMBER

PROJECT

PNAME	PNUMBER	PLOCATION

DEPENDENT

ESSN	DEPE_NAME	SEX	BDATE	RELATIONSHIP

STEP3:

- ✓ For each

1:1 relationship, identify the entities (S, T) participating in that relationship.

- ✓ Choose one entity (assume T), which has total participation in that relationship.
- ✓ In the T relation include the primary key of S relation as the foreign key.
- ✓ If the relationship has any attributes include that in the T relation.
- ✓ In our example:

  MANAGES is a 1:1 relationship. Employee and department are the participating entities. Department entity has the total participation constraint. Because every department has a manager. But every employee is not a manager. Hence the employee entity has a partial participation constraint in that relationship. Include the primary key SSN of employee as the foreign key of department and rename it as MGRSSN. The relationship has an attribute MGRSTARTDATE include that in the department relation.

- ✓ Relational model for company database after step3:

EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION

DEPENDENT

ESSN	DEPE_NAME	SEX	BDATE	RELATIONSHIP

STEP4:

- ✓ For each 1:N relationship, identify the entities (S, T) participating in that relationship.
- ✓ Choose N – side of the relationship (assume T).
- ✓ In the T relation, include the primary key of S relation as the foreign key.
- ✓ If the relationship has any attributes include that in the T relation.
- ✓ In our example:

I:N Relationship	Participating entities	N-side	Foreign key in N side (Primary keys in 1-side are renamed)
WORKS_FOR	Department, employee	employee	Dno (primary key of department)
CONTROLS	Department, project	project	Dnum (primary key of department)
SUPERVISION	Employee, employee	Employee	Superssn (primary key of employee)

- ✓ Relational model for company database after step4:

EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SUPERSSN	SALARY	*DNO*

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

DEPENDENT

ESSN	DEPE_NAME	SEX	BDATE	RELATIONSHIP

STEP5:

- ✓ For each M: N relationship, create a new relation.
- ✓ Both the primary keys of the participating entities are included as foreign key in the new relation.
- ✓ Primary key of the new relation is the combination of these two foreign keys.
- ✓ If the relationship has any attributes include that in the new relation.
- ✓ In our example:

  WORKS _ON is a M: N relationship. Project and employee are the participating entities. Include SSN, PNUMBER as the foreign keys in the new relation.the combination of SSN, The relationship has an attribute HOURS include that in the WORKS _ON relation.

STEP6:

- ✓ For each multi valued attribute (MA), create a new relation.
- ✓ This relation includes an attribute corresponding to MA, plus the primary key of the relation that has this multi valued attribute, as foreign key.
- ✓ Primary key of the new relation is the combination of foreign key, attribute.
- ✓ Relational model for company database after steps 5& 6:

EMPLOYEE

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION

WORKS_ON

ESSN	PNO	HOURS

DEPENDENT

ESSN	DEPE_NAME	SEX	BDATE	RELATIONSHIP

STEP7:

- ✓ For each n-ary relationship (i.e.degree of relationship > 2), create a new relation.
- ✓ The primary keys of the participating entities are included as foreign key in the new relation.
- ✓ Primary key of the new relation is the combination of these foreign keys.
- ✓ If the relationship has any attributes include that in the new relation.
- ✓ For example consider the relationship supply. The degree of relationship is 3

Supplier

sname	….

project

Projname	…

part

partno	…

Supply:

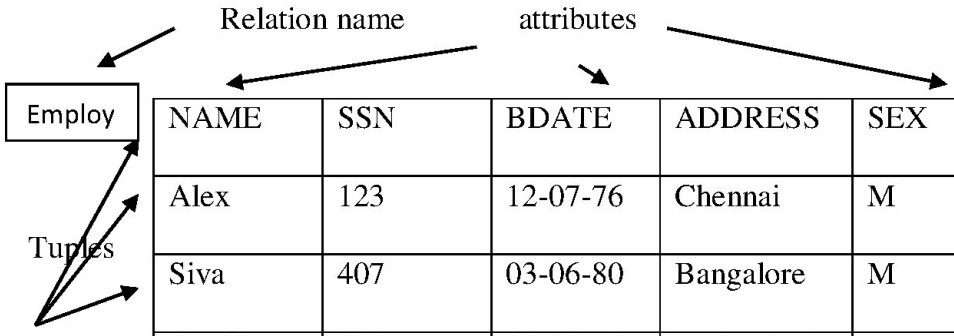
Sname	Projname	Partno	Quantity

Relational Model

Relational model is an example of implementation model or representational model. An implementation model provides concepts that may be understood by end users but that are not too far removed from the way data is organized within the computer.

The relational model represents database as a collection of relations. A relation is a table of values, and each row in the table represents a collection of related data values. In the relational model terminology a row is called a tuple, a column header is called an attribute. The data type describing the types of values that can appear in each column is called a domain. Example

Consider the RELATION: EMPLOYEE



NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO
Alex	123	12-07-76	Chennai	M	20000	1
Siva	407	03-06-80	Bangalore	M	15000	2
Sruthi	905	01-12-78	Hyderabad	F	17000	3
Gayathri	406	27-01-79	Chennai	F	14000	1
Mani	384	29-12-78	Chennai	M	17000	4

NOTATIONS USED:

A relation schema R, denoted by R (A1, A2, An), where R is the relation name.

{A1, A2, An} the list of attributes of relation R.

Dom. (Ai) denotes domain of Ai, represents the number of values in the attribute Ai.

Degree of the relation is the number of attributes in the relation and denoted by n.

Each tuple t is a list of values denoted by t = <v1, v2, vn> where each value vi is an element of dom(Ai) or is a special value null.

Null values represent attributes whose values are “unknown” or “value exists but not available” or “do not exist”.

In our example, relation employee is represented as

Employee (name, ssn, bdate, address, sex, salary, dno)

Degree = 7

A1 = name

Dom(A1) = character

First tuple {alex, 123, 12-07-76, M, 20000, 1}

T (A1)=alex

Relational Model Constraints:

Constraints are various restrictions or conditions on data that can be specified on a relational database. These include

1. Domain constraints
2. Key constraints
3. Entity integrity constraints
4. Referential integrity constraints.

Domain Constraints:

Domain constraints specify that the value of each attribute A must be an atomic value from the domain dom(A). Atomic value means that it is not divisible into components. Hence the composite and multi valued attributes are not allowed. Multi valued attributes must be represented by separate relations, and composite attributes are represented only by their simple component attributes.

Key Constraints:

No two tuples can have the same combination of values for all their attributes. There must be at least one attribute, which identifies each tuple uniquely. That attribute is called

as key attribute. A relation may have more than one key and each of the keys is called a candidate key and any one of the candidate keys is designated as a primary key.

Entity Integrity Constraints:

The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation; having null values for the primary key implies that we cannot identify some tuples. For example, regno field in the student relation cannot be null.

Referential Integrity Constraint:

The Referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the relations. To define referential integrity more formally, we first define the concept of a foreign key. The foreign key specifies referential integrity constraints between the two relation schemas R1 and R2.

Foreign key:

A set of attributes FK in relation schema R1 is a foreign key of R1 that references relation R2 if it satisfies the following two rules:

1. The attributes of FK of R1, PK of R2 should have the same domain.
2. A value of FK in a tuple t1 of R1 should exists as a value of PK for some tuple t2 of R2

R1 is called a referencing relation and R2 is called as referenced relation.

In our example, SSN is the primary key of employee relation and DNO is the primary key of department relation. The employee relation needs to refer the department relation hence we designate DNO as foreign key of employee.

The relational model for employee database is given below in that the primary keys are underlined and referential integrity constraints are represented by, drawing directed arc from each foreign key to the relation it references. The arrowhead points to the primary key of referenced relation.

Relational Model for Company Database

Employee

FNAME	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

Department

DNAME	DNUMBER	DMGRSSN	MGRSTARTDATE

Dept_Locations

DNUMBER	DLOCATION

Project

PNAME	PNUMBER	PLOCATION	DNUM

Works_On

ESSN	PNO	HOURS

Dependent

ESSN	DEPE_NAME	SEX	BDATE	RELATIONSHIP

Insert Operation:

The insert operation provides a list of attribute values for a new tuple t that is to be inserted into a relation R. insert can violate any of the four types of constraints.

Example: Insert <*'ProductX', 501, 'Stafford', 4*> Into *Project.*

- If there is a null in the primary key field then that insertion violates entity integrity constraints.
- This insertion may violate the key constraint if there is another tuple with a same project no.

➢ This insertion may violate the referential integrity constraint if there is no information regarding DNO = 4, which is the foreign key.

Delete Operation:

The delete operation is used to delete existing tuples, which satisfy the specified condition. The deletion operation violates referential integrity consraints.

Example: Delete The*Employee* Tuple With*Name='John'*

Update Operation:

The update operation is used to change the values of one or more attributes in a tuple of some relation R. it is necessary to specify a condition on the attributes of the relation to select the tuple to be modified.

Example: Update The*Salary*Of The *Employee* Tuple With *SSN= '99988777'* To *28000.*

Basic Relational Algebra Operations:

A basic set of relational model operations constitutes the relational algebra. These operations enable the user to specify basic retrieval requests. A sequence of relational algebra operations forms a relational algebra expression.

Relational algebra includes operations like

1. SELECT
2. PROJECT
3. RENAME
4. SET OPERATIONS LIKE
   a) UNION
   b) NTERSECTION
   c) SET DIFFERENCE
5. JOIN
6. DIVISION

1. Select Operation:

The SELECT operation is used to select a subset of the tuples from a relation that satisfy a selection condition. I.e. it selects some of the rows from the table.

Syntax: $\sigma_{<\text{selection condition}>}(R)$

Where σ (sigma) symbol is used to specify the SELECT operator and the selection condition is a Boolean expression specified on the attributes of relation R. the result is also a relation with the same attributes of R. the selection condition is of the form

<Attribute name><comparison op><attribute name/constant>

Comparison operator includes {=, <, >,≥, ≠, ≤}.

Example 1: Select the employees with dno 4

Ans.: $\sigma_{dno=4}$ (EMPOYEE)

Result:

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO
Mani	384	29-12-78	Chennai	M	17000	4

Example 2: List the employees whose salary is greater than 18000

Ans.: $\sigma_{SALARY>18000}$ (EMPOYEE)

Result:

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO
Alex	123	12-07-76	Chennai	M	20000	1

More than one conditions can be connected by Boolean operators AND, OR, NOT to form a general selection condition. Example: consider the following query.

Example 3: Select the employees who either work in dno 3 and receives more than 30000 or work in dno 4 and receives more than 15000.

Ans.: $\sigma_{(dno=3\ AND\ salary>30000)\ OR\ (dno=4\ AND\ salary\ >15000)}$ (EMPOYEE)

Result:

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO
Mani	384	29-12-78	Chennai	M	17000	4

2. Project Operation:

The PROJECT operation is used to select certain set of attributes from a relation. I.e. it selects some of the columns from the table.

Syntax: $\pi_{<attribute\ list>}$ (R)

Where π (pi) symbol is used to specify the PROJECT operator and the attribute list is the list of attributes of relation R. the result is also a relation with the selected set of attributes of R.

Example 4: List each employee's name and salary.

Ans.: $\pi_{name,\ salary}$ (EMPOYEE)

Result:

NAME	SALARY
Alex	20000
Siva	15000
Sruthi	17000
Gayathri	14000
Mani	17000

Nesting of SELECT, PROJECT operators are allowed in a relational algebra expression.

Example 5: retrieve the name and salary of all employees who works in department number 1.

Ans.: $\pi_{name, salary}$ ($\sigma_{dno=1}$ (EMPOYEE))

NAME	SALARY
Alex	20000
Gayathri	14000

It is often simpler to break down complex sequence of operations by specifying intermediate result relations than to write a single relational algebra expression. RENAME operator allows storing the intermediate results in a new relation.

Example 5 can be expressed as following

$$Dep1\_emp \leftarrow \sigma_{dno=1} (EMPOYEE)$$

Result: relation: dep1_emp

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO
Alex	123	12-07-76	Chennai	M	20000	1
Gayathri	406	27-01-79	Chennai	F	14000	1

Result $\leftarrow \pi_{name, salary}$(dep1_emp)

NAME	SALARY
Alex	20000
Gayathri	14000

Dep1_emp, Result are temporary relations created automatically.

3. Rename Operation:

The RENAME operation can rename either the relation name or the attribute names or both.

Syntax: $\rho_{S\,(B1,\,B2\ldots.\,B\,n)}\,(R)$

Where the symbol ρ denote the RENAME operator, S denote the new relation, B1, B2 are new attribute names.

Example 6: change the name of the EMPLOYEE relation as STAFF LIST

Ans.: $\rho_{STAFFLIST}$ (EMPLOYEE)

Result: RELATION: STAFFLIST

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO
Alex	123	12-07-76	Chennai	M	20000	1
Siva	407	03-06-80	Bangalore	M	15000	2
Sruthi	905	01-12-78	Hyderabad	F	17000	3
Gayathri	406	27-01-79	Chennai	F	14000	1
Mani	384	29-12-78	Chennai	M	17000	4

Example 7: change the name of the column header SSN as EMPCODE

Ans.: $\rho_{(NAME,\,EMPCODE,\,BDATE,\,ADDRESS,\,SEX,\,SALARY,\,DNO)}$ (EMPLOYEE)

Result: RELATION: EMPLOYEE

NAME	EMPCODE	BDATE	ADDRESS	SEX	SALARY	DNO
Alex	123	12-07-76	Chennai	M	20000	1

Siva	407	03-06-80	Bangalore	M	15000	2
Sruthi	905	01-12-78	Hyderabad	F	17000	3
Gayathri	406	27-01-79	Chennai	F	14000	1
Mani	384	29-12-78	Chennai	M	17000	4

4. Set Operations:

Set operations are used to merge the elements of two sets in various ways including UNION, INTERSECTION, and SET DIFFERENCE. The three operations UNION, INTERSECTION, and SET DIFFERENCE can be applied on two union compatible relations. What is a union compatible relation? Two relations are said to be union compatible relation if they have same degree n (same no of attributes) and that each pair of corresponding attributes have the same domain.

Consider two union compatible relations.

STUDENT	NAME
	SUSAN
	RAMESH
	JOHNNY
	JIMMY
	BARBARA
	FRANCIS
	ERNEST

TEACHER	NAME
	SUSAN
	JIMMY
	MAYA
	PRIYA
	PATRICK
	RAMESH

Union:

R $\cup$ S where R, S are two relations. The result is also a relation that includes all tuples that are either in R or in S or in both R and S.

Intersection:

R ∩ S where R, S are two relations. The result is also a relation that includes all tuples that are in both R and S.

Set Difference:

R – S The result is also a relation that includes all tuples that are in R but not in S.

Results:

R ∪ S	Name
	SUSAN
	RAMESH
	JOHNNY
	JIMMY
	BARBARA
	FRANCIS
	ERNEST
	MAYA
	PRIYA
	PATRICK

R ∩ S	Name
	SUSAN
	RAMESH
	JIMMY

R - S	Name
	ERNEST
	FRANCIS
	JOHNNY
	BARBARA

4. Cartisian Product:

It is also known as CROSS JOIN or CROSS PRODUCT and denoted by X. like set operations cross product is also used merge two relations but the relations on which it is applied need not be union compatible.

Let us consider the relational algebra expression. Q ← R X S.

Where R is a relation with n attributes (A1, A2.An) and S is a relation with m attributes (B1, B2.Bn) and the resulting relation Q with n+m attributes (A1, A2, An, B1, B2, Bn) in that order. The resulting relation Q has one tuple for each combination of tuples one from R and one from S. hence if R has t1 tuples and S has t2 tuples then R X S t1* t2 tuples.

Consider the RELATION: EMPLOYEE

NAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO
Alex	123	12-07-76	Chennai	M	20000	1
Siva	407	03-06-80	Bangalore	M	15000	3
Sruthi	905	01-12-78	Hyderabad	F	17000	3
Gayathri	406	27-01-79	Chennai	F	14000	1
Mani	384	29-12-78	Chennai	M	17000	4

And the RELATION: DEPARTMENT

DNAME	DNO	MGRSSN
Research	1	123
Accounts	3	905
administration	4	384

Example 8: Retrieve the manager name for each department

The MGRSSN (manager's social security number) is present in the department relation and name of the manager is present in the employee relation. MGRSSN is the foreign key of department relation and SSN is the primary key of employee relation.

Ans.:

Temp1 ← employee X department

Relation: Temp 1

NAME	SSN	…	…	…	…	..	DNAME	DNO	MGRSSN
Alex	123						Research	1	123
Alex	123						Accounts	3	905
Alex	123						administration	4	384
Siva	407						Research	1	123
Siva	407						Accounts	3	905
Siva	407						administration	4	384
Sruthi	905						Research	1	123
Sruthi	905						Accounts	3	905
Sruthi	905						administration	4	384
Gayathri	406						Research	1	123
Gayathri	406						Accounts	3	905
Gayathri	406						administration	4	384
Mani	384						Research	1	123
Mani	384						Accounts	3	905
Mani	384						administration	4	384

The CARTESIAN product creates tuples with the combined attributes of two relations the operation applied by itself is generally meaning less. It is useful when it is followed by a SELECT operation that selects only related tuples from two relations according to the selection condition.

Temp2 $\leftarrow \sigma_{ssn = mgrssn}(temp1)$

RELATION: TEMP 2

NAME	SSN	…	…	…	…	..	DNAME	DNO	MGRSSN

Alex	123						Research	1	123
Sruthi	905						Accounts	3	905
Mani	384						administration	4	384

Managerlist←$\Pi_{dname,\ name}$ (temp2)

Relation: Managerlist

DNAME	NAME
Research	Alex
Accounts	Sruthi
administration	Mani

5. Join Operation:

The CARTESIAN product followed by SELECT is commonly used to select related tuples from two relations. Hence a special operation called JOIN was created to specify this sequence as a single operation. Join operation is denoted by

Q ← R <join condition> S

Where R (A1, A2, .An), S (B1, B2, Bn) are two relations with degree m, n respectively and Q is the resulting relation with m + n attributes, has one tuple for each combination of tuples one from R and one from S –whenever the combination satisfies the join condition. This is the main difference between CARTESIAN PRODUCT and JOIN: in JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result.

The Join condition is of the form A $_{I}$ θBj, where θ (theta) is one of the comparison operators {<, >, ≠, ≥, ≤, =}. The JOIN operation with such a general join condition is called THETA JOIN. If the comparison operator used is = then that join is called EQUI JOIN.

Example 8 can be performed using join operation as following

Temp 1 ← EMPLOYEE SSN=DMGRSSN DEPARTMENT

Managerlist$\leftarrow \Pi_{dname,\ name}$ (temp1)

Natural Join:

Natural join requires that the two join attributes have the same name in both the relations. If this is not the case, a renaming operation is applied first

$Q \leftarrow R * S$

Where * denotes the natural join operation, the relations R and S must have an attribute with same name. If there are more than one attribute pair with same name the NATURAL JOIN operation is performed by equating all attribute pairs that have the same name in both relations.

Example 9: Retrieve the name of the department that GAYATHRI works:

GAYATHRI belongs to DNO 1 and the DNAME information is present in DEPARTMENT relation. The foreign key DNO is used to find out these related details. Here both primary key and foreign key are having same name so you can apply natural join

Temp1 $\leftarrow$ DEPARTMENT * EMPLOYEE

This is equivalent of applying

Temp 1 $\leftarrow$ EMPLOYEE $_{employee.dno=department.dno}$ DEPARTMENT

Gayathri_dept$\leftarrow \Pi_{dname}(\sigma_{NAME = "GAYATHRI"}$ (Temp 1))

Left Outer Join

Use this when you only want to return rows that have matching data in the left table, even if there's no matching rows in the right table.

Example SQL statement

```
SELECT * FROM Individual AS Ind,Publisher AS Pub
WHERE Ind.IndividualId(+)= Pub.IndividualId
```

Source Tables

Left Table-Individual

Id	First Name	Last Name	User Name
1	Fred	Flinstone	freddo
2	Homer	Simpson	homey
3	Homer	Brown	notsofamous
4	Ozzy	Ozzbourne	sabbath
5	Homer	Gain	noplacelike

Right Table-Publisher

Individual Id	Access Level
1	Administrator
2	Contributor
3	Contributor
4	Contributor
10	Administrator

Result

IndividualId	FirstName	LastName	UserName	IndividualId	AccessLevel
1	Fred	Flinstone	freddo	1	Administrator
2	Homer	Simpson	homey	2	Contributor
3	Homer	Brown	notsofamous	3	Contributor
4	Ozzy	Osbourne	sabbath	4	Contributor

5 Homer Gain noplacelike NULL NULL

Right Outer Join

Use this when you only want to return rows that have matching data in the right table, even if there's no matching rows in the left table.

Example SQL statement

```
SELECT * FROM Individual AS Ind, Publisher AS Pub
WHERE Ind.IndividualId = Pub.IndividualId(+)
```

Source Tables

Left Table- Individual

Id	FirstName	LastName	UserName
1	Fred	Flinstone	freddo
2	Homer	Simpson	homey
3	Homer	Brown	notsofamous
4	Ozzy	Ozzbourne	sabbath
5	Homer	Gain	noplacelike

Right Table - Publisher

IndividualId	AccessLevel
1	Administrator
2	Contributor
3	Contributor

4	Contributor
10	Administrator

Result

IndividualId	FirstName	LastName	UserName	IndividualId	AccessLevel
1	Fred	Flinstone	freddo	1	Administrator
2	Homer	Simpson	homey	2	Contributor
3	Homer	Brown	notsofamous	3	Contributor
4	Ozzy	Osbourne	sabbath	4	Contributor
NULL	NULL	NULL	NULL	10	Administrator

Full Outer Join

Use this when you want to all rows, even if there's no matching rows in the right table.

Example SQL statement

```
SELECT * FROM Individual AS Ind,Publisher AS Pub WHERE
Ind.IndividualId(+) = Pub.IndividualId(+)
```

Source Tables

Left Table- Individual

Id	FirstName	LastName	UserName
1	Fred	Flinstone	freddo
2	Homer	Simpson	homey
3	Homer	Brown	notsofamous

4	Ozzy	Ozzbourne	sabbath
5	Homer	Gain	noplacelike

Right Table- Publisher

IndividualId	AccessLevel
1	Administrator
2	Contributor
3	Contributor
4	Contributor
10	Administrator

Result

IndividualId	FirstName	LastName	UserName	IndividualId	AccessLevel
1	Fred	Flinstone	freddo	1	Administrator
2	Homer	Simpson	homey	2	Contributor
3	Homer	Brown	notsofamous	3	Contributor
4	Ozzy	Osbourne	sabbath	4	Contributor
5	Homer	Gain	noplacelike	NULL	NULL
NULL	NULL	NULL	NULL	10	Administrator

7. Aggregate Functions And Grouping:

Aggregate functions are applied to collection of numeric values include SUM, AVERAGE, MINIMUM, and MAXIMUM. The COUNT function is used for counting tuples or values. Grouping is used to group the tuples in a relation by the value of some of their attributes.

Views InSql

- A view in SQL is a single table that is derived from other tables.
- These other tables could be base tables or previously defined views
- A view does not exit in a physical form.
- It is considered as Virtual table.
- View is a way of specifying a table that we need to reference frequently.

For example:

We may frequently issue queries that retrieve the employee name and project names that the employee works on. Rather than having to specify the join of the employee, works on & project tables every time we issue that query, we can define a view that is a result of these joins. We then issue queries on the view.

Specification of Views in SQL:

The command to specify view is:

Syntax:

CREATE VIEW <View name> AS SELECT <Attribute list> FROM <Table list> WHERE <condition>;

The view is given a table name (view name), a list of attribute name, and a query to specify the contents of the view.

Example:

CREATE VIEW EMP_PROJ

AS SELECT FNAME, LNAME, PNAME, HOURS

FROM EMPLOYEE, PROJECT, WORKS-ON

WHERE SSN=ESSN AND PNO=PNUMBER;

In this case EMP_PROJ inherits the names of the view attributes from the defining tables EMPLOYEE, PROJECT, WORKS-ON

EMP_PROJ

FNAME	LNAME	PNAME	HOURS

Retrieve the first name of all employees who work on 'Product-X '

Q: SELECT FNAME, LNAME FROM EMP_PROJ WHERE PNAME='Product-X';

Advantage: It is simplify the specification of certain queries. It is also used as a security and authorization mechanism.

- View is always *up to date*; if we modify the tuple in the base tables on which the view is defined, the view must automatically reflect these changes.
- If we do not need a view any more we can use the DROP VIEW command to dispose of it.

DROP VIEW EMP_PROJ;

View Implementation and View Update:

- Updating of views is complicated.
- An update on a view defined on a single table without any aggregate functions can be mapped to an update on the underlying base table.
- Update the PNAME attribute of 'John Smith' from 'ProductX' to 'ProductY'.

UPDATE EMP_PROJ

SET PNAME='ProductY' WHERE FNAME='John' AND LNAME='Smith' AND PNAME='ProductX';

- A view update is feasible when only one update on the base relations can accomplish the desired update effect on the view.
- Whenever an update on the view can be mapped to more than one tuple on the underlying base relations, we must have a certain procedure to choose the desired update.
- For choosing the most likely update, A view with a single defining table is updatable if the view attributes contain the primary key of the base relation, because this maps each view tuple to a single base relation.

- Views defined on multiple tables using joins are generally not updateable.
- Views defined using grouping and aggregate functions are not updatable.

Tuple Relational Calculus

- Relational calculus is a formal query language, where to specify a retrieval request.
- A calculus expression specifies what is to be retrieved rather than how to retrieved it.
- There fore, the relational calculus is considered to be a non procedural language.
- This differs from relational algebra, where we must write a sequence of operations to specify a retrieval request.
- Relational calculus is considered as a procedural way of stating a query.
- Expressive power of the 2 languages is identical.

Tuple variable & Range Relation:

- Tuple Relational calculus is based on specifying a number of tuple variables.
- Each tuple variable usually ranges over a particular database relation.
- That is the variable may take as its value any individual tuple from that relation.

A simple tuple relational calculus query is of the form:

{ t | COND (t) }

where, t – is a tuple variable & COND(t)- includes Range relation of tuple and selective condition.

The result of such a query is the set of all tuples t that satisfy COND(t)

Example:

Q1. To find all employees whose salary is above 50,000.

Tuplerelational Expression:

{ t | Employee(t) and t. salary > 50000}

Q2. To retrieve only some of the attributes:

{ t. fname, t. address | Employee(t) and t. salary > 50000}

Information specified in a tuple calculus expression:

- For each tuple variable t, the range relation R of t. this value is specified by a condition of the form R(t).
- A COND to select particular combinations of tuples.
- A set of attributes to be retrieved, the requested attributes.

Expressions & Formulas in Tuple Relational Calculus:

A general Expression of the Tuple relational calculus

{ t1.A1, t2. A2, …..,t_n. A_n | COND(t1,t2,…..,t_n, t_{n+1},…..,t_{n+m}) }

t1.A1, t2. A2, ….., t_n. A_n are tuple variables, each A_i is an attribute of the relation on which t_i ranges.

- COND is made up of atoms is of the form t_i. A_iOPt_j. $B_j$ort_i. A_i OP C or
  C OP t_j. B_j
- Where OP is one of the comparison operators is the set of { =,<,<=,>,>=,!=}and C is constant.

A Formula is made up of one or more atoms connected via the logical operators and, or ,not.

Example formulas: F1 AND F2, F1 OR F2 , NOT F1

The Existential & Universal Quantifiers:

- In a formula two additional symbols called Quantifiers: Existential( ), universal ( )
- A tuple variable t is BOUND if it is quantified or it is FREE.
- Example:

F1: d. dname ='Research'

F2: (d)(d.MGESSN = 987654321)

F3: (t) (t.dname = t.dno)

The tuple variable d is FREE in F1 & F3, d is BOUND in F2. The variable t is BOUND in F3.

- If F is a formula, then so is (t) (F), where t is a tuple variable. The formula (t) (F) is true if the formula F evaluates to TRUE for some (at least one) tuple assigned of t in F, otherwise (t) (F) is false.
- If F is a formula, then so is (t) (F), where t is a tuple variable. The formula (t) (F) is true if the formula F evaluates to TRUE for ever tuple assigned of t in F, otherwise (t) (F) is false.

Q3: Retrieve the name , address of all employees who work for the research department.

{ t. fname, t. address | Employee(t) and (d)(Department (d) and d.dname =' Research' and d.Dnumber = t. DNO) }

Safe Expression:

- The expression in relational calculus is the one which produce finite tuples as its result, then that expression is known as Safe Expression or it is known as Unsafe Expression.
- An expression is safe if all values in its result are from the domain of the expression.
- Example: { t | not (Employee(t) }

It is unsafe, it produce all tuples in the universe that are not employee tuples, which are infinite.

Domain Relational Calculus

- The domain relational calculus differs from the tuple relational calculus in the type of variables used in formula.
- Rather than having domain variables range over tuples, the domain variables ranges over the domain of attributes.

An expression of the Domain calculus is of the form:

{ x1,x2,.....x_n | COND (x1, x2, ...x_n, x_{n+1}, ...x_{n+m})}

x1,x2,.....$x_n$are domain variables that range over domain(attributes)

Q1: Retrieve the birthdate address of all employees.

{ tu | employee(pqrstuvwxy)}

Ten variables for the employee relation, one to range over the domain of each attribute in order.

Q2: Retrieve the name of all employees whose salary is >35 000.

{ p | employee(pqrstuvwxy) and w>35000 }

Q3: Retrieve the all employees ssn who are all working in project number.

{ a | project (abc) and b=1}

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – II - Database Management Systems – SCSA1301

II. Database Design

Overview of the hierarchal Data model- Overview of the Network Data model-Relational Database Design- Mapping ER Model-QBE-Functional Dependency-Normalization

Data Models

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organized rather than what operations will be performed on the data. To use a common analogy, the data model is equivalent to an architect's building plans.

Data models:

A collection of tools for describing Data

Relationship among data

Their semantics and

Constraints

Types of data models:

1. Object based logical data models

2. Record based logical data models.

3. Physical data models.

4. Semi structured data models (XML)

1. Object based logical data models

- Object oriented logical data model
- Object relational data models Ex. E.R. model

2. Record based logical data models.

- Network data model
- Hierarchical data model
- Relational data model.

3. Physical data models.: Used for internal shcema definition.

Record based data models

1. **Hierarchical data models:**

- Data are organized in a ordered tree manner.
- Relationship between records is parent child type.
- The data base in this model is a collection of disjoint trees.
- It is a simple and straight forward model & easy to understand.
- It is used when there is question of hierarchical relationship
- In order to represent link among records pointers are used.
- The relationship among records are physical
- Searching for a record is not easy.
- Represents one to many relationship
- Deletion of an internal node deletion of its leaves.

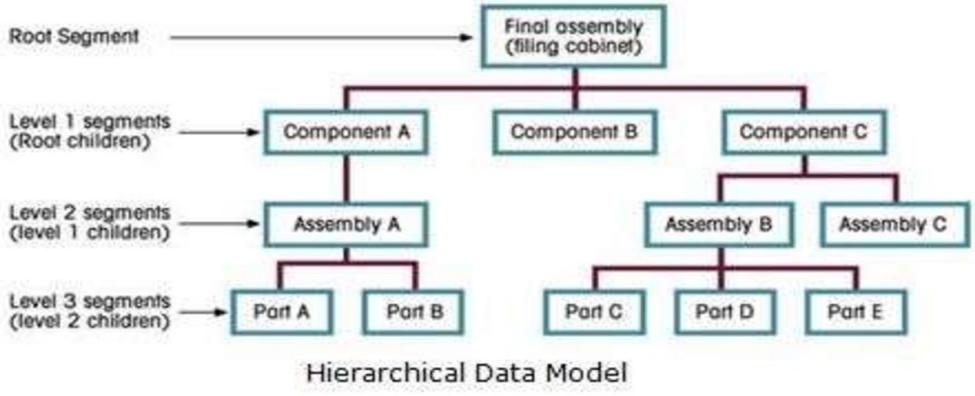


Fig. 2.1 Hierarchical Data Model

Advantages.

- Simple and easy to use

- Data with hierarchical relationship can be mapped on this model.
- Suitable for application such as: Employee Dept.

Disadvantages:

- Search for (an element or) a record is difficult.
- Insertion, deletion and updation are difficult.
- Many – Many relationshipscan not be established.
- Data with non-hierarchical relationship cannot be mapped.
- The hierarchical relationship is maintained using points which require extra storage.
- Changes in relationship require changes in entire structure of the database.
- Processing is sequential among branches of the tree so access time is high.

Network data model:

- It was formalized by conference on data system language (codasyl)
- It is an improvement over hierarchical data model.
- It represents many-to-many relationships.
- A child can have multiple parents.
- Relationship between records is maintained by pointers.
- Each record has a pointer field for the record with which it is associated.

Advantages:

- Many to many relationship among records can be implemented.
- Useful for representing such records which are represented in many to many relationships.
- Searching a record is easy since there are multiple access paths.

- No problem of consistency in case of addition deletion of records.

Disadvantages:

- Implementations of records and relationship are complex.
- Storage space requirement is high because of so many pointers.

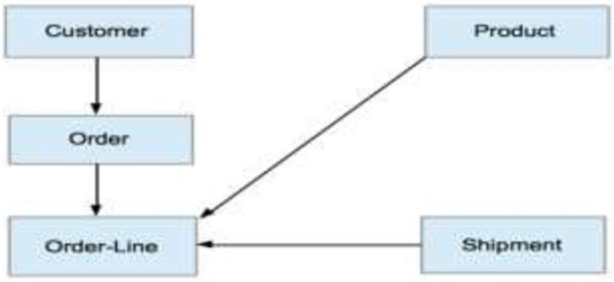


Fig 2.2. Network Model

Relational data model:

1. In relational data model the relation or a named table is the only construct required to represent an association among the attributes of an entity set and the relationships among different entities.
2. Relationship between record is represented by a relation that contains a key for each record involved in the relation.
3. Many to many relationship can easily be implemented.
4. Relationship implementation is very easy through the use of key and composite key fields.
5. Relational model is useful for representing most of the real world entities and relationships among them.
6. Relational model does not maintain physical connection among records.
7. Data is organized logically in the form of rows and columns.
8. A unique indexed key field is used to search for data element.

9. Data integrity is maintained by the process like normalization .
10. Description of data in terms of this model is called a schema.
11. Schema for relation specifies, its name, name of each field. Ex. Student (sid: Integer, name: string, login: string etc.)

Advantages:

1. Tabular structure is easy to understand simple.
2. Data manipulation is easy.
3. We can apply mathematical operation on tables.
4. Built in query language support such as SQL.
5. Very flexible data organization.

Disadvantages:

1. Size of the data base becomes large.

Example:Oracle, Ingress, Sybase, Unity etc.

- The data represented in this model is in the form of two dimensional table called relation.
- An entity is represented by a tuple in a row.
- The rows of a table are distinct.
- Ordering of rows is immaterial.
- Each column of the table is assigned distinct heading called name of the attribute.
- In each column data item are of similar type.
- The ordering of the columns is immaterial.
- If there are M. columns, it is said to be of degreeim.

- If there are N rows, it is called a N tupple table or cardinality of the table is N.
- Both the rows and columns can be viewed in any sequence at any time without
- affecting the information.

Properties of Relational Data Base Management System (RDBMS)

- A relational database management is represented by relational data models.
- It uses a collection of tables to represent the data and the relationship among them.
- Each table has multiple no of rows and columns
- Supports the concept of null values.
- Does not require the user to understand its physical implementation.
- Provides information about its contents and structure.

Domain and Integrity Constraints:

Domain Constraints:

- Limit the range of domain an attribute values on.
- Specify uniqueness and null ness of attributes.
- Specify default value of an attribute.

Integrity constraints:

Entity integrity: Every tuple is uniquely identified by a unique non null attribute, primary key i.e. the primary key values can not be null.

Referential integrity: Rows in different tables are correctly related by valid key values (Foreign keys refers to primary keys\

Basic terminologies of relational data model: (RDBMS)

1. **Entity:** A real world object with some properties which can be easily identified iscalled an entity.
2. **Attributes:** An attribute is a descriptive property or a characteristics of an entity.Ex name, roll marks, etc.
3. **Degree:** The no of columns or attributes associated with a table (or a relationamong them) is called degree of the relation.
4. **Cardinality:** The no of rows in a table is called cordinality.
5. **Tuples:** A relation consisting of a number of records represented in row wiseinformation called tuples.
6. **Domain:** The set of possible values that can allotted to an attribute is calleddomain of that attribute. Domain d is a set of atomic values of an attribute.
7. **Entity set:** A set of entities representing a real world object. Ex. Student,teacher, depositor, player etc.
8. **Weak entity:** An entity set which may not have sufficient attribute to form aprimary key then it is called an weak entity.
9. **Strong entity:** An entity set which have a key attribute.
10. **Key:** An attribute that allows us to uniquely identity a record or an entity type.
11. **Super key:** A set of attributes that collectively allows us to identify an entity is anentity set.
12. **Candidate key:** A candidate key is the minimal super key. The super key forwhich no proper subset is a super key.
13. **Primary key:** Primary key is the one of the candidate key to identify the entityuniquely. The primary key is the principal means of identifying an entity.
14. **Entity type:** The occurrences of an entity set are called entities

15. **Composite key:** A composite key is a candidate key that consists of two or moreattributes.

16. **Foreign key:** A foreign key is an attribute (a group of attributes) that is primarykey to another relation. A ford on key represents a relationship between two tables.

Note: Primary key of the strong entity set is not explicitly stored with the weak entity set since it is implicit in the identifying relationship.

- **Single valued attribute:** An attribute that holds a single value for each occurrence of an entity type Ex. Empid, emphasis ok.
- **Multivalued attribute:** A multivalue attribute is an attribute that holds multiplevalues for each occurance of an entity type Ex. Phone numbers, skill of an employee.
- **Derived attribute:** Can be computed from other attributes . Ex. Age, for givendate of birth, year of service, given a date of joining.
- **Identifier attribute:** An attribute that uniquely identifies a particular entity from anentity set.
  Ex. Empid, personid ,regn. no etc.

Banking E-R Diagram

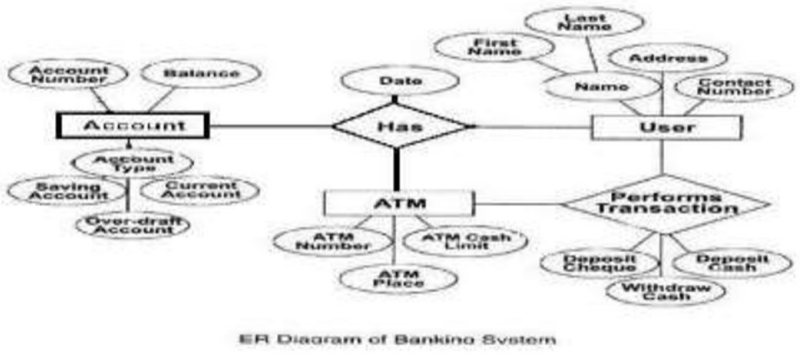


Fig. 2.3 E R Diagram of Banking System

Schema:

The database schema of a database is its structure described in a formal language supported by the database management system (DBMS). The term "schema" refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).

Schema is of three types: Physical schema, logical schema and view schema

1. **Logical Database Schema:** This schema defines all the logical constraints that require tobe applied on the information stored. It defines tables, views, and integrity constraints.
2. **Physical Database Schema:** This schema pertains to the particular storage of informationand it's kind of storage like files, indices, etc. It defines how the information will be stored in an exceedingly auxiliary storage.
3. **View schema:** View schema is outlined because of the style of a database at view levelthat usually describes end-user interaction with database systems.

Normalaization Techniques

Normalization is a process of organizing the data in database to avoid data redundancy,insertion anomaly, update anomaly & deletion anomaly.Let's discuss about anomalies first then we will discuss normal forms with examples.

Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized.

1. **Insertion Anomaly**
2. **Update Anomaly**
3. **Deletion Anomaly**

Example: Suppose a manufacturing company stores the employee details in a table named

Table : Employee

Attributes: emp_id, emp_name, emp_address, emp_dept

At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

The above table is not normalized. We will see the problems that we face when a table is notnormalized.

Update anomaly: In the above table we have two rows for employee Rick as he belongs to twodepartments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly: Suppose a new employee joins the company, who is under training andcurrently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Delete anomaly: Suppose, if at a point of time the company closes the department D890 thendeleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

Normalization

Here are the most commonly used normal forms:

- **First normal form(1NF)**
- **Second normal form(2NF)**
- **Third normal form(3NF)**
- **Boyce &Codd normal form (BCNF)**

First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values.

It should hold only atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. Itcreates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222

103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above. This table is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the emp_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212

104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach.Theycreate a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40

333	Chemistry	40

Candidate Keys: {teacher_id, subject}

Non prime attribute: teacher_age

The table is in 1 NF because each attribute has atomic values.

However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key.

This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

To make the table complies with 2NF we can break it in two tables like this:

teacher_details table:

Teacher id	**Teacher age**
111	**38**
222	**38**
333	**40**

teacher_subject table:

teacher_id	subject
111	Maths

111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this:

A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Suppose a company wants to store the complete address of each employee, theycreate a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	DayalBagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on

Candidate Keys: {emp_id}

Non-prime attributes: all attributes except emp_id are non-prime as they are not part of anycandidate keys.Here, emp_state, emp_city&emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city&emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency.

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	DayalBagh
222008	TN	Chennai	M-City

282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Boyce Codd normal form (BCNF)

Boyce-Codd Normal Form (BCNF) is one of the forms of database normalization. A databasetable is in BCNF if and only if there are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. BCNF is also sometimes referred to as 3.5NF, or 3.5 Normal Form. It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in **more than one department**.

They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100

1002	American	Purchasing department	D134	600

Functional dependencies in the table above:

emp_id ->emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}

The table is not in BCNF as neither emp_id nor emp_dept alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250

design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:

emp_id ->emp_nationality

emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:

For first table: emp_id

For second table: emp_dept

For third table: {emp_id, emp_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

What are the Advantages of normalization?

1. **Data consistency**

   Data consistency means that the data is always real and it is not ambiguous.

2. **Data becomes non redundant**

   Non-redundant means that only copy original copy of data is available for each user and for every time. There are no multiple copies of the same data for different persons. So when data is changed in one file and stay in one file. Then of course data is consistent and non-redundant. Here redundant is not the same as a backup of data, both are different things.

3. **Reduce insertion, deletion and updating anomalies**

   Insertion anomaly is an anomaly that occurs when we want to insert data intothe database but the data is not completely or correctly inserted in the target attributes. If completely inserted in the database then not correctly entered.

   Deletion anomaly is an anomaly that occurs when we want to delete data in thedatabase but the data is not completely or correctly deleted in the target attributes.

   Updation anomaly is an anomaly that occurs when we want to update data in thedatabase but the data is not completely or correctly updated in the target attributes.

4. **Database table compaction**

   When we normalize the database, we convert the large table into a smaller table that leads to data and table compaction. Compaction means to have the least and required size.

5. **Better performance**

6. **Fast queries**

What are the disadvantages of normalization?

1. Required experienced database designer
2. Difficult and expansive
3. Requires detailed database design
4. As the normal form type progresses, the performance becomes slower and slower.
5. Proper knowledge is required on the various normal forms to execute the normalization process efficiently. Careless use may lead to terrible design filled with major anomalies and data inconsistency

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – III - Database Management Systems – SCSA1301

III. Query Processing

SQL Queries -Embedded SQL -My SQL: Basics, Queries in MySQL and Algorithms for Query Processing and Optimization - Introduction to Transaction Processing Concepts and Theory - Concurrency control techniques.

SQL- Structured Query Language

Introduction

- SQL is an ANSI standard computer language for accessing and manipulating databases.
- The databases covered are: Oracle, DB2, Sybase, Informix, Microsoft SQL Server, Microsoft Access, and other database systems.
- Like QBE, SQL provides users with a way of querying relational databases.
- SQL was developed under the name SEQUEL at the IBM San Jose research center in the mid 1970s.
- In 1980, it was renamed SQL (still pronounced as "Sequel") to avoid confusion with an unrelated hardware product also called SEQUEL
- SQL stands for Structured Query Language
- SQL allows you to access a database
- SQL is an ANSI standard computer language
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert new records in a database
- SQL can delete records from a database
- SQL can update records in a database
- SQL is easy to learn

SQL - Languages

Data Definition Language (DDL) – Consists of commands which are used to define the database.

Data Manipulation Language (DML) – Consists of commands which are used to manipulate the data present in the database.

Data Control Language(DCL) – Consists of commands which deal with the user permissions and controls of the database system.

Transaction Control Language(TCL) – Consist of commands which deal with the transaction of the database.

SQL Commands:

Data Definition Language Commands (DDL)

- CREATE
- DROP
- TRUNCATE
- ALTER
- BACKUP DATABASE

CREATE Command

To Create User

CREATE USER username

Example

CREATE USER maran;

Grant Permission To The User

GRANT ALL TO maran;

(From root granting DDL,DML commands to user)

CREATE TABLE

This statement is used to create a table.

Syntax

CREATE TABLE TableName(Column1datatype, Column2 datatype, Column3 datatype, ... ColumnNdatatype);

Example

```
CREATE TABLE Employee_Info (EmployeeIDint,
EmployeeNamevarchar(255),
EmergencyContactNamevarchar(255), PhoneNumberint,
Address  varchar(255), City  varchar(255), Country  varchar(255));
```

CREATE TABLE using another TABLE

Example

```
CREATE TABLE ExampleTable AS
SELECT EmployeeName, PhoneNumber
FROM Employee_Info;
```

DROP Command

DROP

This statement is used to drop (delete) an existing table or a database with data.

Syntax

```
DROP DATABASE Database Name;
```

Example

```
DROP DATABASE Employee;
```

Syntax

```
DROP TABLE Table Name;
```

Example

```
DROP Table Employee_Info
```

TRUNCATE Command

This command is used to delete the information(records) present in the table but does not delete the table. So, once you use this command, your information will be lost, but not the table.

Syntax

TRUNCATE TABLE TableName;

Example

TRUNCATE Table Employee_Info

ALTER_Command

ALTER

This command is used to delete, modify or add constraints or columns in an existing table.

The 'ALTER TABLE' Statement with ADD/DROP COLUMN

Syntax

ALTER TABLE TableName ADD ColumnNameDatatype;

ALTER TABLE TableName DROP COLUMN ColumnName;

Example

ADD Column BloodGroup:

ALTER TABLE Employee_Info ADD BloodGroupvarchar(255);

DROP Column BloodGroup:

ALTER TABLE Employee_Info DROP COLUMN BloodGroup ;

The 'ALTER TABLE' Statement with ALTER/MODIFY COLUMN

This statement is used to change the datatype of an existing column in a table.

Syntax

ALTER TABLE TableName ALTER COLUMN ColumnNameDatatype;

Example

Add a column DOB and change the data type to Date.

ALTER TABLE Employee_Info ADD DOB year;

ALTER TABLE Employee_Info ALTER DOB date;

KEYS in the DATABASE

There are mainly 7 types of Keys, that can be considered in a database.

1. **Candidate Key** – A set of attributes which can uniquely identify a table can be termed asa Candidate Key. A table can have more than one candidate key, and out of the chosen candidate keys, one key can be chosen as a Primary Key. In the above example, since EmployeeID, InsuranceNumber and PanNumber can uniquely identify every tuple, they would be considered as a Candidate Key.
2. **Super Key** – The set of attributes which can uniquely identify a tuple is known as SuperKey. So, a candidate key, primary key, and a unique key is a superkey, but vice-versa isn't true.
3. **Primary Key** – A set of attributes which are used to uniquely identify every tuple is alsoa primary key. In the above example, since EmployeeID, InsuranceNumber and PanNumberare candidate keys, any one of them can be chosen as a Primary Key. Here EmployeeID is chosen as the primary key.
4. **Alternate Key** – Alternate Keys are the candidate keys, which are not chosen as aPrimary key. From the above example, the alternate keys are PanNumber and Insurance Number.
5. **Unique Key** – The unique key is similar to the primary key, but allows one NULL valuein the column. Here the Insurance Number and the Pan Number can be considered as unique keys.
6. **Foreign Key** – An attribute that can only take the values present as the values of someother attribute, is the foreign key to the attribute to which it refers. in the above example, the Employee_ID from the Employee_Information Table is referred to the Employee_ID from the Employee_Salary Table.
7. **Composite Key** – A composite key is a combination of two or more columns that identify each tuple uniquely. Here, the Employee_ID and Month-Year_Of_Salary can be grouped together to uniquely identify every tuple in the table.

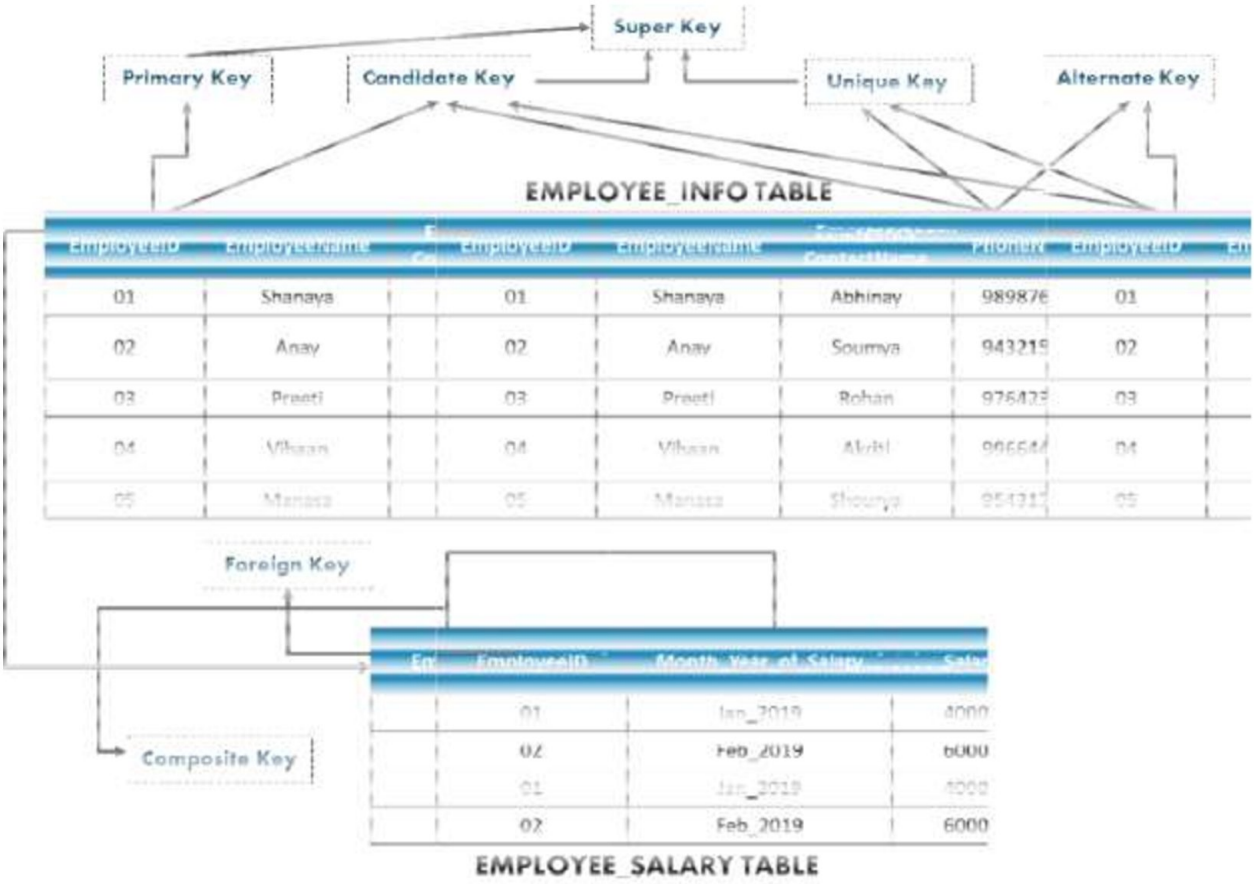


Fig. 3.1 Keys in the Database

SQL Commands: Constraints Used In Database

Constraints are used in a database to specify the rules for data in a table. The fol lowing are the different types of constraints:

- NOT NULL
- PRIMARY KEY
- FOREIGN KEY
- UNIQUE
- CHECK
- DEFAULT
- INDEX

NOT NULL Constraint

This constraint ensures that a column cannot have a NULL value.

Example

CREATE TABLE Employee_Info(

EmployeeIDint NOT NULL, EmployeeNamevarchar(255) NOT NULL,

Emergency ContactNamevarchar(255), PhoneNumberint NOT NULL,

Address varchar(255), City varchar(255), Country varchar(255));

NOT NULL on ALTER TABLE

ALTER TABLE Employee_Info MODIFY PhoneNumber int NOT NULL;

UNIQUE Constraint

CREATE TABLE Employee_Info (

EmployeeIDint NOT NULL UNIQUE,

City varchar(255), Country varchar(255));

UNIQUE on ALTER TABLE

ALTER TABLE Employee_Info ADD UNIQUE (Employee_ID);

To drop a UNIQUE constraint

ALTER TABLE Employee_Info DROP CONSTRAINT UC_Employee_Info;

PRIMAY KEY constraint

The Primary key constraint uniquely identifies each record in a table, Primary keys must contain UNIQUE values, and cannot contail NULL values. A table can have only one primary key, and in the table, this primary key can consist of single or multiple columns (fields). The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created.

CREATE TABLE Persons (ID int NOT NULL PRIMARY KEY, LastNamevarchar(255) NOT NULL, FirstNamevarchar(255),Age int);

CREATE TABLE Persons (ID int NOT NULL, LastNamevarchar(255) NOT NULL, FirstNamevarchar(255), Age int, CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);

ALTER TABLE Persons
ADD PRIMARY KEY (ID);

ALTER TABLE Persons
DROP PRIMARY KEY;

ALTER TABLE Persons
DROP CONSTRAINT PK_Person;

REFERENCES on Primary key and Foreign Key

CREATE TABLE Orders (OrderIDint NOT NULL, OrderNumberint NOT NULL,

PersonIDint, PRIMARY KEY (OrderID), CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)

ALTER TABLE Orders ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
ALTER TABLE Orders ADD CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
ALTER TABLE Orders DROP FOREIGN KEY FK_PersonOrder;
ALTER TABLE Orders DROP CONSTRAINT FK_PersonOrder;

SQL Date Data Types

SQL comes with the following data types for storing a date or a date/time value in the database:

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

SQL View

SQL CREATE VIEW Examples

CREATE VIEW [Brazil Customers] AS SELECT CustomerName, ContactName FROM Customers WHERE Country = 'Brazil';

CREATE VIEW [Products Above AveragePrice] AS SELECT ProductName, Price FROM Products WHERE Price > (SELECT AVG(Price) FROM Products);

CREATE OR REPLACE VIEW *VIEW_NAME* AS SELECT *COLUMN1*, *COLUMN2*, ... FROM *TABLE_NAME* WHERE *CONDITION*;

CREATE OR REPLACE VIEW [Brazil Customers] AS SELECT CustomerName, ContactName, City FROM Customers WHERE Country = 'Brazil';

DROP VIEW *VIEW_NAME*;

Embedded SQL

Applications are developed using some general purpose languages like C, C++, JAVA, etc. These languages are used to get UIs, forms etc. SQLs inside the application language like C, C++, Java etc, and make these applications to communicate with DB.

Structure of Embedded SQL

Structure of embedded SQL defines step by step process of establishing a connection with DB and executing the code in the DB within the high level language

Connection to DB

- This is the first step while writing a query in high level languages.
- First connection to the DB that we are accessing needs to be established. This can be done using the keyword CONNECT. But it has to precede with
- 'EXEC SQL' to indicate that it is a SQL statement.
- EXEC SQL CONNECT db_name;
- EXEC SQL CONNECT HR_USER; //connects to DB HR_USER

Declaration Section

- Once connection is established with DB, we can perform DB transactions. Since these DB transactions are dependent on the values and variables of the host language.
- Depending on their values, query will be written and executed.
- Similarly, results of DB query will be returned to the host language which will be captured by the variables of host language.

- Hence we need to declare the variables to pass the value to the query and get the values from query. There are two types of variables used in the host language

Host Variable :

- These are the variables of host language used to pass the value to the query as well as to capture the values returned by the query.
- Since SQL is dependent on host language
- We have to use variables of host language and such variables are known as host variable. But these host variables should be declared within the SQL area or within SQL code.

(i) That means compiler should be able to differentiate it from normal C variables.

(ii) Hence we have to declare host variables within

(iii) BEGIN DECLARE and END DECLARE section.

(iv) Again, these declare block should be enclosed within EXEC SQL and ';'.

```
EXEC SQL BEGIN DECLARE SECTION;
    int STD_ID;
    char STD_NAME [15];
    char ADDRESS[20];
EXEC SQL END DECLARE SECTION
```

(i) We can note here that variables are written inside begin and end block of the SQL, but they are declared using C code.

(ii) It does not use SQL code to declare the variables. Why?

(iii) This is because they are host variables C language.

(iv) Hence we cannot use SQL syntax to declare them. Host language supports almost all the data types from int, char, long, float, double, pointer, array, string, structures etc

(v) When host variables are used in a SQL query, it should be preceded by colon – ':' to indicate that it is a host variable.

(vi) Hence when pre-compiler compiles SQL code, it substitutes the value of host variable and compiles.

(vii) EXEC SQL SELECT * FROM STUDENT WHERE STUDENT_ID =:STD_ID;

(viii) In above code, :STD_ID will be replaced by its value when pre-compiler compiles it.

(ix) Suppose we do not know what should be the datatype of host variables or what is the datatype in oracle for few of the columns.

(x) In such case we can allow the compiler to fetch the datatype of column and assign it to the host variable. It is done using 'BASED ON' clause. But format of declaration will be in host language.

```
EXEC SQL BEGIN DECLARE SECTION;

    BASED ON STUDENT.STD_ID sid;

    BASED ON STUDENT.STD_NAME sname;

    BASED ON STUDENT.ADDRESS saddress;

EXEC SQL END DECLARE SECTION;
```

Execution Section This is the execution section, and it contains all theSQL queries and statements prefixed by 'EXEC SQL'

```
EXEC SQL SELECT * FROM STUDENT WHERE STUDENT_ID
=:STD_ID; EXEC SQL SELECT STD_NAME INTO :SNAME :IND_SNAME

   FROM STUDENT WHERE STUDENT_ID =:STD_ID;

INSERT INTO STUDENT (STD_ID, STD_NAME)  VALUES (:SID, :SNAME);

UPDATE STUDENT SET ADDRESS = :STD_ADDR
  WHERE STD_ID = :SID;
```

Consider a simple Pro*C program to illustrate embedded SQL. This program below accepts student name from the user and queries DB for his student i

```
#include <stdio.h>

#include <sqlca.h>

int main(){

EXEC SQL INCLUDE SQLCA;
```

```
EXEC SQL BEGIN DECLARE SECTION;

BASED ON STUDENT.STD_ID SID; // host variable to store the value returned by
query char *STD_NAME; // host variable to pass the value to the query short ind_sid;//
indicator variable

EXEC SQL END DECLARE SECTION;

//Error handling

EXEC WHENEVER NOT FOUND GOTO error_msg1;
EXEC WHENEVER SQLERROR GOTO error_msg2;
printf("Enter the Student name:");

scanf("%s", STD_Name)

// Executes the query

EXEC SQL SELECT STD_ID INTO : SID INDICATOR ind_sid FROM STUDENT

WHERE STD_NAME = : STD_NAME;

printf("STUDENT ID:%d", STD_ID); // prints the result from
DB exit(0);
```

- Error handling

```
labels error_msg1:
printf("Student Id %d is not found",
STD_ID); printf("ERROR:%ld", sqlca-
>sqlcode); printf("ERROR State:%s", sqlca-
>sqlstate); exit(0);

error_msg2:

printf("Error has occurred!");
printf("ERROR:%ld", sqlca->sqlcode);
printf("ERROR State:%s", sqlca-
>sqlstate); exit(0);
```

- MySQL is free.
- Very widely used.
- Implements SQL database mangement.
- Linux Red Hat already in cludes MySQL.
- Facebook uses MySQL.

No	Dev./License	MySQL	Oracle	SQL Server
1	Developer / Maintainer	Oracle Corp. (since 2010) Sun Microsystems (2008-2010) MySQL AB (before 2008)	Oracle Corp. (Since 1977)	Microsoft Corp.
2	License	General Public License, and Proprietary	Proprietary	Proprietary

SQL	MySQL
Is a programming language used to issue instructions to a relational database management system	Is an open source relational database management system
Used to issue instructions about how data is to be inserted, manipulated, removed or accessed within the RDBMS	Used to store data for manipulation and retrieval in the future
You need to learn the language to use it	Readily available through download and installation
Is fixed and not updatable	Regularly updated

(i) MySQL is currently the most popular database management system soft ware used for managing the relational d atabase.

(ii) It is open-source databas e software, which is supported by Oracle Comp any.

(ii) It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database.

(iii) It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

(iv) MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

(v) MySQL follows the working of Client-Server Architecture.

(vi) This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services.

(vii) Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched.

(viii) The process of MySQL environment is the same as the client-server model.

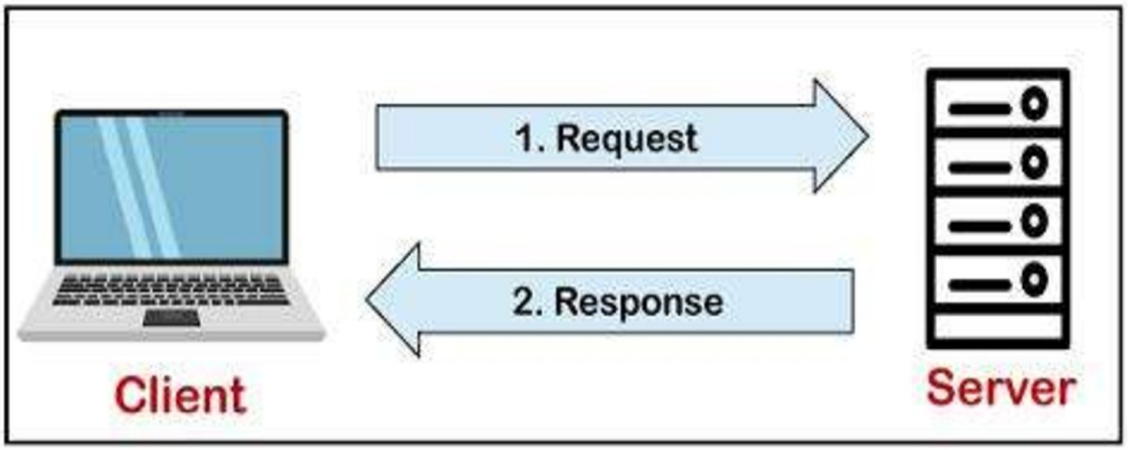


Fig 3.2 Client Server Communication

MySQL Features

- Relational Database Management System (RDBMS)
- Easy to use
- It is secure
- Free to download
- It is scalable. It can handle almost any amount of data, up to as much as 50 million rowsor more.
- Speed

- High Flexibility. MySQL supports a large number of embedded applications, whichmakes MySQL very flexible.
- Compatible on many operating systems
- Memory efficiency
- GUI Support

Disadvantages/Drawback of MySQL

- ROLE, COMMIT, and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently, and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

Grant Privileges to the MySQL New User

MySQL server provides multiple types of privileges to a new user account. Some of the most commonly used privileges are given below:

- **ALL PRIVILEGES:** It permits all privileges to a new user account.
- **CREATE:** It enables the user account to create databases and tables.
- **DROP:** It enables the user account to drop databases and tables.
- **DELETE:** It enables the user account to delete rows from a specific table.
- **INSERT:** It enables the user account to insert rows into a specific table.
- **SELECT:** It enables the user account to read a database.
- **UPDATE:** It enables the user account to update table rows.

Algorithms for Query Processing and Optimization - Introduction to Transaction Processing Concepts and Theory - Concurrency control techniques.

QUERY PROCESSING

Algorithms for Query Processing and Optimization

Query Processing refers to the range of activities involved in extracting data from adatabase. The activities include translation of queries in high-level database languages into expressions that can be used at the physical level of the file system, a variety of query-optimizing transformations, and actual evaluation of queries.

The steps involved in processing a query are:

- Parsing and translation.

- Optimization.
- Evaluation.

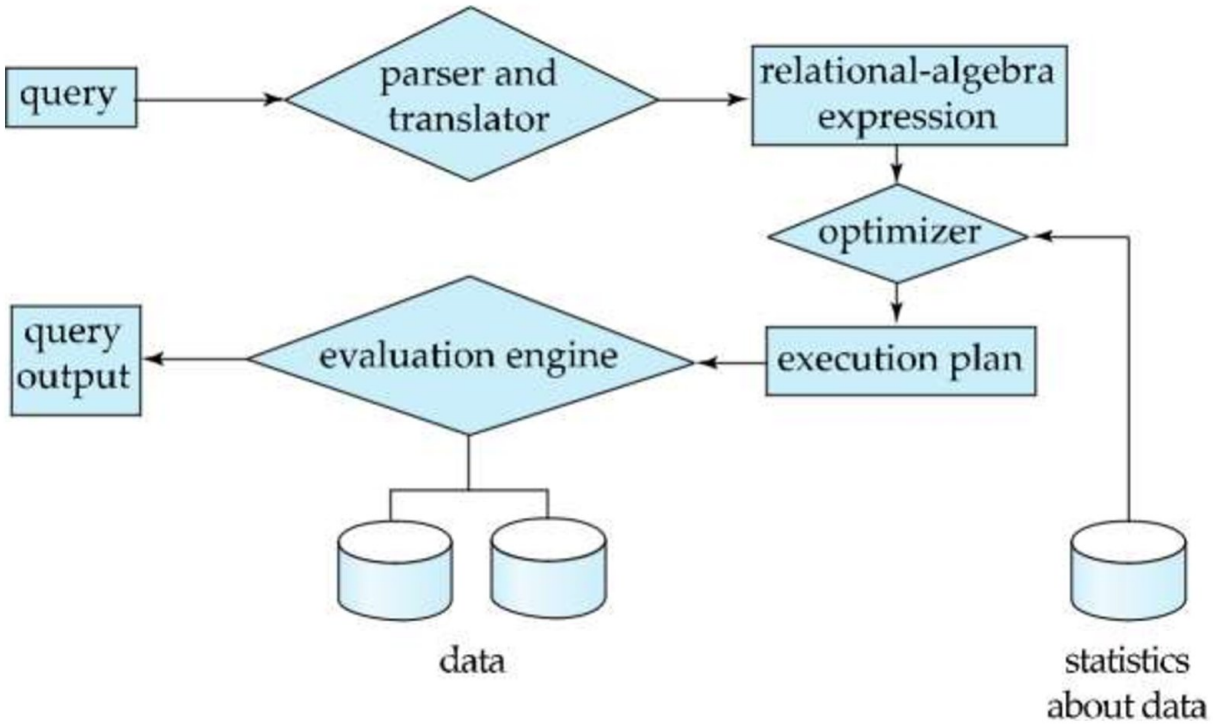


Fig. 3.3 Typical steps when processing a high level query.

- **Parsing and translation**

Translate the query into its internal form. This is then translated into relational algebra.Parser checks syntax, verifies relation.

- **Optimization**

SQL is a very high level language. The users specify what to search for - not how the search is actually done. The algorithms are chosen automatically by the DBMS. For a given SQL query there may be many possible execution plans. Amongst all equivalent plans choose the one with lowest cost. Cost is estimated using statistical information from the database catalog.

3. Evaluation

The query evaluation engine takes a query evaluation plan, executes that plan and returns the answer to that query. As an illustration, consider the query:

select salary from instructor where salary < 75000;

This query can be translated into either of the following relational-algebra expressions:

$$\sigma_{salary\ <75000}\ (\Pi_{salary}\ (instructor))$$

(or)

$$\Pi_{salary}\ (\sigma_{salary<75000}\ (instructor))$$

Further, each relational-algebra operation can be executed by one of several different algorithms. For example, to implement the preceding selection, we can search every tuple in instructor to find tuples with salary less than 75000. If a B^+ tree index is available on the attribute salary, we can use the index instead to locate the tuples. To specify fully how to evaluate a query, providing the relational-algebra expression will not be anought, but also to annotate it with instructions specifying how to evaluate each operation.

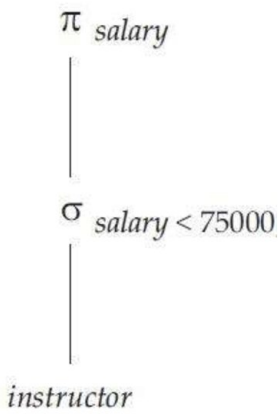


Figure 3.1.2 A query-evaluation plan

A sequence of primitive operations that can be used to evaluate a query is a query-execution plan or query-evaluation plan. Figure 3.1.2 illustrates an evaluation plan for our example query. The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query. The different evaluation plans for a given query can have different costs. It is the responsibility of the system to construct a query evaluation plan that minimizes the cost of query evaluation; this task is called query optimization. Once the query plan is chosen, the query is evaluated with that plan, and the result of the query is output. In order to optimize a query, a query optimizer must know the cost of each operation.

Measures of Query Cost

The cost of query evaluation can be measured in terms of a number of different resources, including disk accesses, CPU time to execute a query and in a distributed or parallel database system, the cost of communication.

The response time for a query-evaluation plan could be used as a good measure of the cost of the plan. In large database systems, however, disk accesses are usually the most important cost, since disk accesses are slow compared to in-memory operations.

Most people consider the disk accesses cost a reasonable measure of the cost of a query-evaluation plan.

The number of block transfers from disk is also used as a measure of the actual cost. It takes more time to write a block to disk than to read a block from disk. For more accurate measure, find:

- The number of seek operations performed
- The number of blocks read
- The number of blocks written

and then add up these numbers after multiplying them by average seek time, average transfer time for reading a block and average transfer time for writing a block respectively.

Using Heuristics in Query Optimization

Query Optimization - Heuristics

Query Trees and Graphs

selectP.Pnumber, P.Dnum, E.Lname, E.Address, E.Bdate from Project as P, Department as D, Employee as E where P.Dnum=D.Dnumber and .Mgr_ssn=E.Ssn and P.Plocation='Stafford';

πPnumber, Dnum, Lname, Address, Bdate[[(σPlocation='Stafford'(Project)) ⋈Dnum=Dnumber (Department)] ⋈Mgr_ssn=Ssn(Employee)]

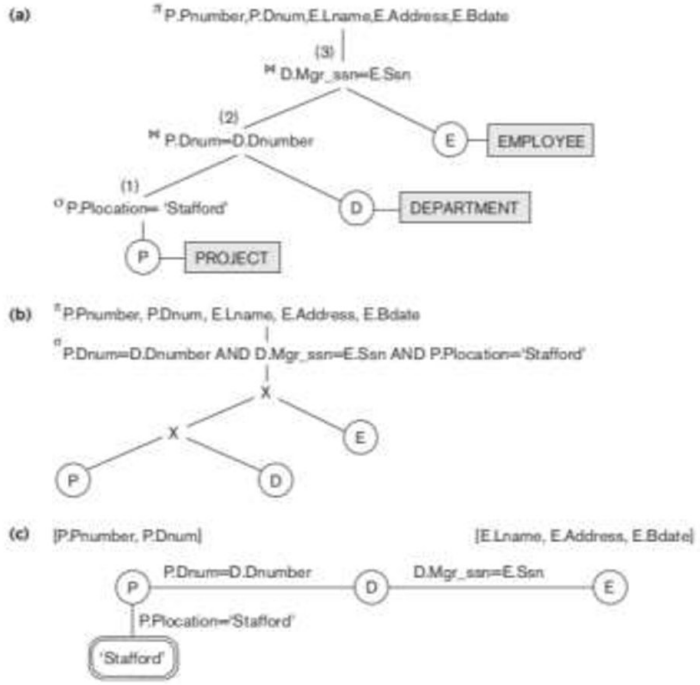


Figure 19.4
Two query trees for the query Q2. (a) Query tree corresponding to the relational algebra expression for Q2. (b) Initial (canonical) query tree for SQL query Q2. (c) Query graph for Q2.

Optimizing Them

In actuality the initial translation is simplistic:

Where σ

Select π

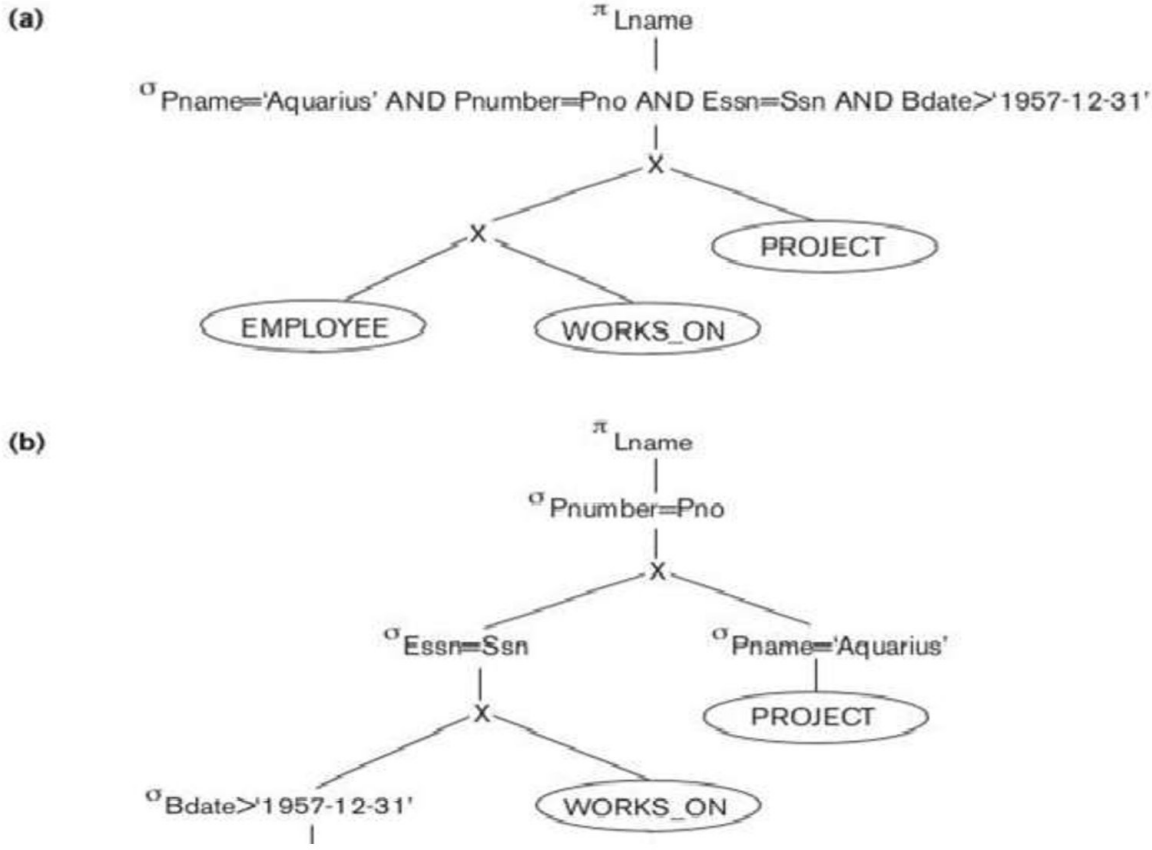
Then optimizations transform the query tree into something more efficient based on reordering and transforming the extended relational operations. After that the tree is converted to a *query execution plan* that chooses the best algorithms to implement the portions of the tree. πPnumber, Dnum, Lname, Address, Bdate(σP.Dnum=D.DnumberΛD.Mgr_ssn=E.SsnΛ

P.Plocation='Stafford'((Project × Department) × Employee))

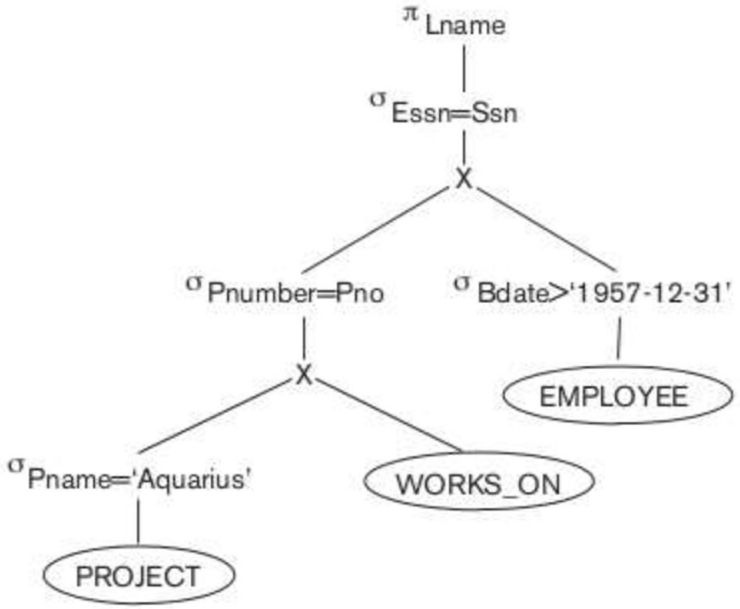
Find the last names of employees born after 1957 who work on a project named ‘Aquarius’.

select Lname fromEmployee, Works_On, Project where Pname='Aquarius' and Pnumber=Pno and Essn=Ssn and Bdate> '1957-12-31';

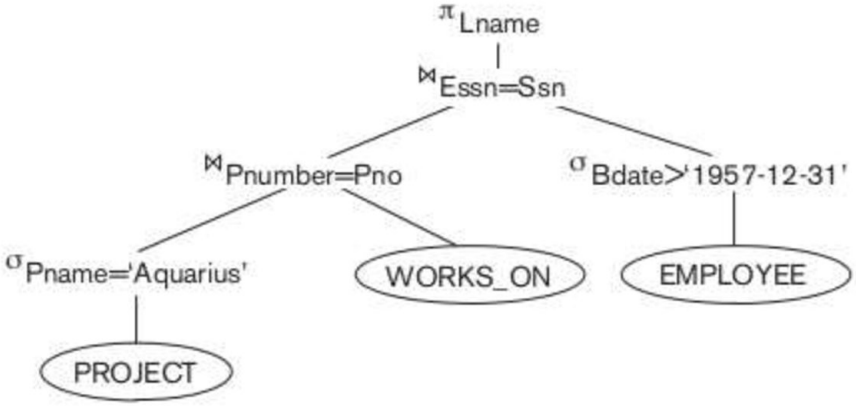
Steps in converting a query tree during heuristic optimization.
(a) Initial (canonical) query tree for SQL query Q.
(b) Moving SELECT operations down the query tree.
(c) Applying the more restrictive SELECT operation first.
(d) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.
(e) Moving PROJECT operations down the query tree.



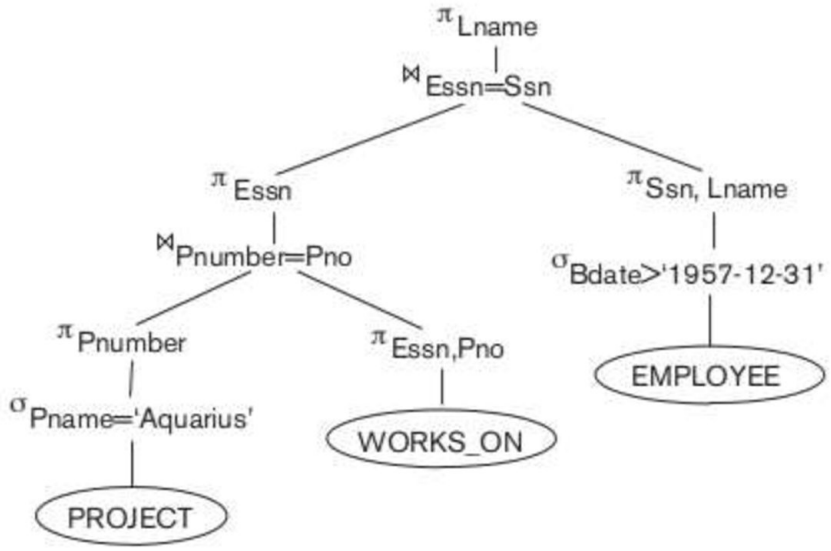
(c)


πLname
σEssn=Ssn
X
σPnumber=Pno
σBdate>'1957-12-31'
X
EMPLOYEE
σPname='Aquarius'
WORKS_ON
PROJECT

(d)


πLname
⋈Essn=Ssn
⋈Pnumber=Pno
σBdate>'1957-12-31'
σPname='Aquarius'
WORKS_ON
EMPLOYEE
PROJECT

(e)


πLname
⋈Essn=Ssn
πEssn
πSsn, Lname
⋈Pnumber=Pno
σBdate>'1957-12-31'
πPnumber
πEssn,Pno
EMPLOYEE
σPname='Aquarius'
WORKS_ON
PROJECT

General Transformation Rules

Generally, perform σ and π (which tend to reduce the size of the intermediate tables) *before* any &bowie; operations (which tend to multiply their sizes). .

1	Cascade σ	$\sigma c_1 \wedge c_2 \wedge \ldots \wedge cn \equiv \sigma c_1(\sigma c_2(\ldots \sigma cn(R)))$	
2	Commutativity of σ	$\sigma a(\sigma b(R)) \equiv \sigma b(\sigma a(R))$	
3	Cascade π	$\pi a(\pi b(\ldots(R))) \equiv \pi a(R)$	
4	Commute σ with π	$\pi a,b,\ldots(\sigma c(R)) \equiv \sigma c(\pi a,b,\ldots(R))$	Only if c only depends on the attributes of π
5	Commutativity of ⋈	$R \bowtie S \equiv S \bowtie R$	Also true of ×
6	Commuting σ with ⋈	$\sigma c(R \bowtie S) \equiv (\sigma c(R)) \bowtie S$	Only if c involves only the attributes of R. Also applies to ×.
6a	Commuting σ with ⋈, conjunction version	$\sigma c_1 \wedge c_2(R \bowtie S) \equiv (\sigma c_1(R)) \bowtie (\sigma c_2(S))$	Where c_1 and c_2 involve the attributes of R and S, respectively.

7	Commuting π with ⋈	$\pi L (R \bowtie c S) \equiv (\pi A_1,\ldots,An(R)) \bowtie c(\pi B_1,\ldots,Bm(S))$	So long as the projection attributes L = $\{A_1,\ldots,An,B_1,\ldots,Bm\}$ where A ∈Attr(R), B ∈Attr(S), and the join condition c only dependes on attributes in L.
7a	Commuting π with ⋈, extra join attributes version	$\pi L (R \bowtie c S) \equiv \pi L [(\pi A_1,\ldots,An,\ldots,An+k(R)) \bowtie c(\pi B_1,\ldots,Bm,\ldots,Bm+p(S))]$	Here the projection attributes in L are the same, but the join condition c depends on additional attributes from R and S. These attributes are shown as An+1…An+k and Bm+1…Bm+p. The original πL must be wrapped around the right-side expression to remove those extra join atttributes one they're unneeded.
8	Commutativity of set operations		∩ and ∪ are commutative but – is not
9	Associativity of ⋈, ×, ∪, and ∩	$(R\times S)\times T \equiv R\times(S\times T)$	Also true for each of the other three

General approach:

- Start with the canonical tree.
- Move σs as low as possible.
- Merge operations when possible (×,σ → ⋈ is classic).
- Move πs down, but not probably not below σs.

Example query optimization

```
selectFname, Lname, Address
from  Employee as E, Department as D
where  D.Super_ssn=E.Ssn and D.Dname like 'Plan%';
```

Converting Query Trees to Execution Plans

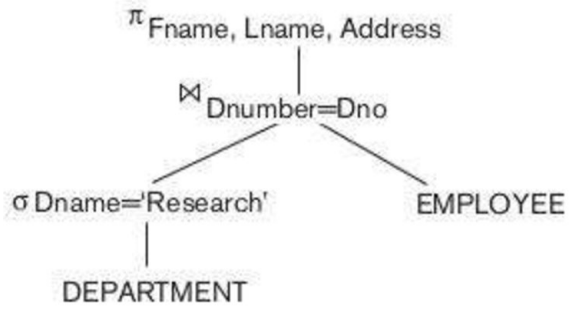


Figure 19.6
A query tree for query Q1.

- If there is an index on Department.Dname, use it for an index search to implement the σDname='Research'.
- If there is an index on Employee.Dno, implement the ⋈Dnumber=Dno by a J2 (Single-loop join).
- Lastly a simple π over those results.
- Ideally all these algorithms would be pipelined as a single step without writing intermediate files.

Using Selectivity and Cost Estimates in Query Optimization

The DBMS attempts to form a good *cost model* of various query operations as applied to the current database state, including the attribute value statistics (histogram), nature of indices, number of block buffers that can be allocated to various pipelines, selectivity of selection clauses, storage speed, network speed (for distributed databases in particular), and so on.

Access cost to secondary storage

Reading and writing blocks between storage and ram. (time)

Disk storage cost

Cost of temporary intermediate files in storage. (time/space)

Computation cost

Usually slower than storage, but sometimes not, the cpu cost of evaluating the query. (time)

Memory usage cost

The amount of ram needed by the query. (space)

Communication cost

The cost to transport query data over a network between database nodes. (time) "Typical" databases emphasize access cost, the usual limiting factor. In-memory databases minimize computation cost, while distributed databases put increasing emphasis on communication cost.

Catalog Information Used in Cost Functions

- Basic file data: number of records (r), record size (R), number of blocks (b).
- Primary file organization (heap file, ordered, ordered with index, hashed, overflow?)
- Other indices, if present.
- Number of levels (x) for multilevel indices, top-level block count (b_{I1}).
- Number of distinct values (d) of an attribute. If the values are uniformly distributed, the *selectivity* (sl) is ($^1/_d$), and the selection cardinality ($s = sl \cdot r$). If the attribute values are *skewed* this isn't a good estimate, and a more detailed histogram of sl is appropriate.

Examples of Cost Functions for Select

S1. linear search

On average half the blocks must be accessed for an equality condition on the key, all the blocks otherwise.

S2. binary search

lg b for the search, plus more if it's nonkey.

S3a. primary index for a single record

One more than the number of index levels.

S3b. hash index for a single record

Average of 1 or 2 depending on the type of hash.

S4,.ordered index for multiple records

$C_{S4}=x+{}^{b}/_{2}$ as a rough estimate.

S5. clustering index for multiple records

x for the index search to get the cluster block, then $\lceil s/bfr \rceil$ file blocks pointed to by the cluster.

S6. B^{+}-Tree

Tree height + 1 if its key, that plus the selectivity (s) if not.

S7. conjunctive selection

The sum of the costs of the subconditions, if the set intersection fits in memory.

S8,.conjunctive selection with composite index

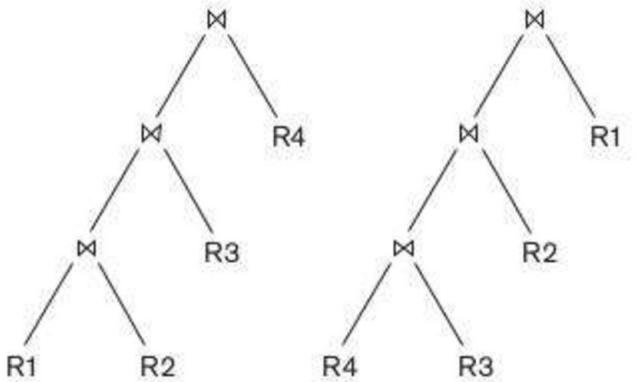
The same as above for whichever type of index.

Examples of Cost Functions for Join

A *join selectivity* (js) is the ratio of the number of tuples produced by a join to the number of tuples produced by a cross product of the underlying relations, or $js = |(R \bowtie c\ S)| / |(R \times S)| = |(R \bowtie c\ S)| / (|R| \cdot |S|)$.

Multiple Relation Queries and Join Ordering

Figure 19.7
Two left-deep (JOIN) query trees.



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – IV- Database Management Systems – SCSA1301

IV. Recovery Techniques

Database Recovery Techniques - Database Security – Debate on the distributed databases and Client- Server Architecture with reference to Indian Railway Reservation System.

Database Recovery

Purpose of Database Recovery

To bring the database into the last consistent state, which existed prior to the failure.

To preserve transaction properties (Atomicity, Consistency, Isolation and Durability).

Example:

If the system crashes before a fund transfer transaction completes its execution, then either one or both accounts may have incorrect value. Thus, the database must be restored to the state before the transaction modified any of the accounts.

Types of Failure

The database may become unavailable for use due to

Transaction failure: Transactions may fail because of incorrect input, deadlock, incorrect synchronization.

System failure: System may fail because of addressing error, application error, operating system fault, RAM failure, etc.

Media failure: Disk head crash, power disruption, etc.

Transaction Log

For recovery from any type of failure data values prior to modification (BFIM BeFore Image) and the new value after modification (AFIM – AFter Image) are required.

These values and other information is stored in a sequential file called Transaction log.

Data Update

Immediate Update: As soon as a data item is modified in cache, the disk copy is updated.

Deferred Update: All modified data items in the cache is written either after a transaction ends its execution or after a fixed number of transactions have completed their execution.

Shadow update: The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.

In-place update: The disk version of the data item is overwritten by the cache version.

Data Caching

Data items to be modified are first stored into database cache by the Cache Manager (CM) and after modification they are flushed (written) to the disk.

The flushing is controlled by Modified and Pin-Unpin bits.

Pin-Unpin: Instructs the operating system not to flush the data item.

Modified: Indicates the AFIM of the data item.'

Transaction Roll-back (Undo) and Roll-Forward (Redo)

To maintain atomicity, a transaction's operations are redone or undone.

Undo: Restore all BFIMs on to disk (Remove all AFIMs).

Redo: Restore all AFIMs on to disk.

Database recovery is achieved either by performing only Undos or only Redos or by a combination of the two. These operations are recorded in the log as they happen. When in-place update (immediate or deferred) is used then log is necessary for recovery and it must be available to recovery manager. This is achieved by WriteAhead Logging WAL) protocol. WAL states that

For Undo: Before a data item's AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).

For Redo: Before a transaction executes its commit operation, all its AFIMs must be Written to the log and the log must be Saved on a stable store.

Checkpointing

Time to time (randomly or under some criteria) the database flushes its buffer to database disk to minimize the task of recovery. The following steps defines a checkpoint operation:

Suspend execution of transactions temporarily. Force writes modified buffer data to disk.

Write a [checkpoint] record to the log, save the log to disk. Resume normal transaction execution. During recovery redo or undo is required to transactions appearing after [checkpoint] record.

Steal/No-Steal and Force/No-Force

Possible ways for flushing database cache to database disk:

Steal: Cache can be flushed before transaction commits.

No-Steal: Cache cannot be flushed before transaction commit.

Force: Cache is immediately flushed (forced) to disk.

No-Force: Cache is deferred until transaction commits

These give rise to four different ways for handling recovery:

Steal/No-Force (Undo/Redo)

Steal/Force (Undo/No-redo)

No-Steal/No-Force (Redo/No-undo)

No-Steal/Force (No-undo/No-redo)

Recovery Scheme

a. Deferred Update (No Undo/Redo)

The data update goes as follows:

A set of transactions records their updates in the log.

At commit point under WAL scheme these updates are saved on database disk.

After reboot from a failure the log is used to redo all the transactions affected by this failure. No undo is required because no AFIM is flushed to the disk before a transaction commits.

b. Deferred Update with concurrent users

This environment requires some concurrency control mechanism to guarantee isolation property of transactions. In a system recovery transactions which were recorded in the log after the last checkpoint were redone. The recovery manager may scan some of the transactions recorded before the checkpoint to get the AFIMs

Deferred Update with concurrent users.

Two tables are required for implementing this protocol:

Active table: All active transactions are entered in this table.

Commit table: Transactions to be committed are entered in this table.

During recovery, all transactions of the commit table is redone and all transactions of active tables are ignored since none of their AFIMs reached the database. It is possible that a commit table transaction may be redone twice but this does not create any inconsistency because of a redone is "idempotent", that is, one redone for an AFIM is equivalent to multiple redone for the same AFIM.

c. Recovery Techniques Based on Immediate Update

Undo/No-redo Algorithm

In this algorithm AFIMs of a transaction are flushed to the database disk under WAL before it commits. For this reason the recovery manager undoes all transactions during recovery.No

transaction is redone. It is possible that a transaction might have completed execution and ready to commit but this transaction is also undone.

Undo/Redo Algorithm (Single-user environment)

Recovery schemes of this category apply undo and also redo for recovery. In a single-user environment no concurrency control is required but a log is maintained under WAL.Note that at any time there will be one transaction in the system and it will be either in the commit table or in the active table.

The recovery manager performs:

Undo of a transaction if it is in the active table.

Redo of a transaction if it is in the commit table.

Undo/Redo Algorithm (Concurrent execution)

Recovery schemes of this category applies undo and also redo to recover the database from failure.

In concurrent execution environment a concurrency control is required and log is maintained under WAL.

Commit table records transactions to be committed and active table records active transactions. To minimize the work of the recovery manager check pointing is used.

The recovery performs:

Undo of a transaction if it is in the active table.

Redo of a transaction if it is in the commit table.

d.Shadow Paging

The AFIM does not overwrite its BFIM but recorded at another place on the disk. Thus, at any time a data item has AFIM and BFIM (Shadow copy of the data item) at two different places on the disk.

X and Y: Shadow copies of data items

X' and Y': Current copies of data items

To manage access of data items by concurrent transactions two directories (current and shadow) are used

□The directory arrangement is illustrated below. Here a page is a data item.

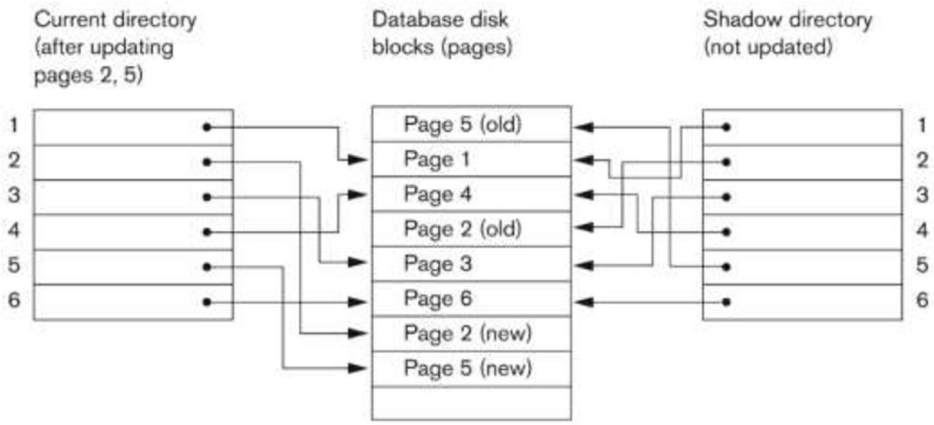


Figure 19.5
An example of shadow paging.

Fig 4.1 Shadow Paging

e. The ARIES Recovery Algorithm

The ARIES Recovery Algorithm is based on:

WAL (Write Ahead Logging) Repeating history during redo:

ARIES will retrace all actions of the database system prior to the crash to reconstruct the database state when the crash occurred.

Logging changes during undo:

It will prevent ARIES from repeating the completed undo operations if a failure occurs during recovery, which causes a restart of the recovery process.

The ARIES recovery algorithm consists of three steps:

Analysis: step identifies the dirty (updated) pages in the buffer and the set of transactions active at the time of crash. The appropriate point in the log where redo is to start is also determined.

Redo: necessary redo operations are applied.

Undo: log is scanned backwards and the operations of transactions active at the time of crash are undone in reverse order.

The Log and Log Sequence Number (LSN)

A log record is written for:

- data update
- transaction commit
- transaction abort

- undo
- transaction end

In the case of undo a compensating log record is written. The Log and Log Sequence Number (LSN) (contd.). A unique LSN is associated with every log record. LSN increases monotonically and indicates the disk address of the log record it is associated with.

In addition, each data page stores the LSN of the latest log record corresponding to a change for that page.

A log record stores

(a) the previous LSN of that transaction

(b) the transaction ID

A log record stores:

Previous LSN of that transaction: It links the log record of each transaction. It is like a back pointer points to the previous record of the same transaction , Transaction ID , Type of log record. For a write operation the following additional information is logged:

Page ID for the page that includes the item

Length of the updated item

Its offset from the beginning of the page

BFIM of the item

AFIM of the item

The Transaction table and the Dirty Page table

For efficient recovery following tables are also stored in the log during check pointing:

Transaction table: Contains an entry for each active transaction, with information such as transaction ID, transaction status and the LSN of the most recent log record for the transaction.

Dirty Page table: Contains an entry for each dirty page in the buffer, which includes the page ID and the LSN corresponding to the earliest update to that page.

e. Checkpointing

A checkpointing does the following:

Writes a begin checkpoint record in the log Writes an end checkpoint record in the log. With this record the contents of transaction table and dirty page table are appended to the end of the log.Writes the LSN of the begin_checkpoint record to a special file. This special file is accessed

during recovery to locate the last checkpoint information. To reduce the cost of checkpointing and allow the system to continue to execute transactions, ARIES uses "fuzzy checkpointing".

The following steps are performed for recovery

Analysis phase: Start at the begin_checkpoint record and proceed to the end_checkpoint record. Access transaction table and dirty page table are appended to the end of the log. Note that during this phase some other log records may be written to the log and transaction table may be modified. The analysis phase compiles the set of redo and undoes to be performed and ends.

Redo phase: Starts from the point in the log up to where all dirty pages have been flushed, and move forward to the end of the log. Any change that appears in the dirty page table is redone.

Undo phase: Starts from the end of the log and proceeds backward while performing appropriate undo. For each undo it writes a compensating record in the log. The recovery completes at the end of undo phase.

(a)

Lsn	Last_lsn	Tran_id	Type	Page_id	Other_information
1	0	T_1	update	C	...
2	0	T_2	update	B	...
3	1	T_1	commit		...
4	begin checkpoint				
5	end checkpoint				
6	0	T_3	update	A	...
7	2	T_2	update	C	...
8	7	T_2	commit		...

(b)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	2	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	1
B	2

(c)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	8	commit
T_3	6	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	1
B	2
A	6

Figure 19.6
An example of recovery in ARIES. (a) The log at point of crash. (b) The Transaction and Dirty Page Tables at time of checkpoint. (c) The Transaction and Dirty Page Tables after the analysis phase.

Fig

4.2 Aries Algorithm

f. Recovery in multidatabase system

A multi database system is a special distributed database system where one node may be running relational database system under UNIX, another may be running object-oriented system under Windows and so on.

A transaction may run in a distributed fashion at multiple nodes. In this execution scenario the transaction commits only when all these multiple nodes agree to commit individually the part of the transaction they were executing.

This commit scheme is referred to as "two-phase commit" (2PC). If any one of these nodes fails or cannot commit the part of the transaction, then the transaction is aborted.Each node recovers the transaction under its own recovery protocol.

Distributed Database:

Distributed databases bring the advantages of distributed computing to the database management domain. A distributed computing system consists of a number of processing elements, not necessarily homogeneous, that are interconnected by a computer network, and that cooperate in performing certain assigned tasks.

A distributed database (DDB) is a collection of multiple, logically interrelated databases distributed over a computer network • A distributed database management system (DDBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.

Data stored at a number of sites each site logically consists of a single processor at different sites are interconnected by a computer network (we do not consider multiprocessors in DDBMS, cf. Parallel systems). DDBS is a database, not a collection of files (cf. relational data model).Placement and query of data is impacted by the access patterns of the user DDBMS is a collections of DBMSs (not a remote file system).

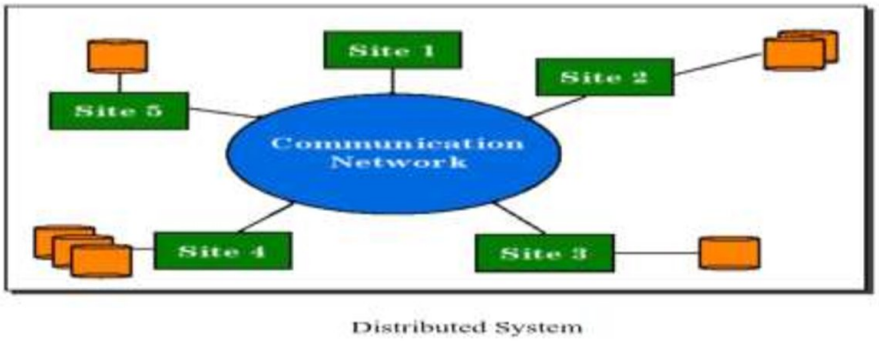


Fig 4.3 Distributed System

- **Example:** Database consists of 3 relations `employees`, `projects`, and `assignment` which are partitioned and stored at different sites (fragmentation).

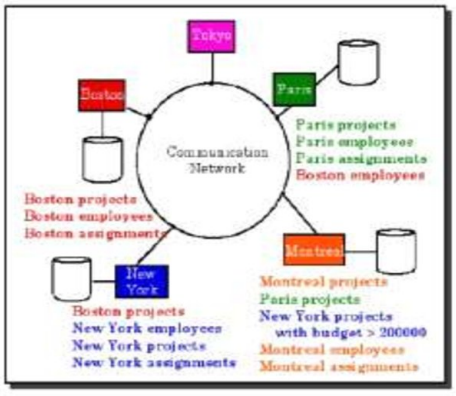


Fig 4.4 Distributed Databases

Applications of Distributed Databases:

- Manufacturing
- Military command and control
- Airlines
- Hotel Chains
- Any organization which has a decentralized organization structure

Advantages of Distributed Databases

1. Management of distributed data with different levels of transparency:

Distribution or network transparency: This refers to freedom for the user from the operational details of the network. It may be divided into location transparency and naming transparency. Location transparency refers to the fact that the command used to perform a task is independent of the location of data and the location of the system where the command was issued. Naming transparency implies that once a name is specified, the named objects can be accessed unambiguously without additional specification.

2. Replication transparency:

Copies of data may be stored at multiple sites for better availability, performance, and reliability. Replication transparency makes the user unaware of the existence of copies.

3. Fragmentation transparency:

Two types of fragmentation are possible.

Horizontal fragmentation distributes a relation into sets of tuples (rows).

Vertical fragmentation distributes a relation into subrelations where each subrelation is defined by a subset of the columns of the original relation. A global query by the user must be transformed into several fragment queries. Fragmentation transparency makes the user unaware of the existence of fragments.

4. Increased reliability and availability:

These are two of the most common potential advantages cited for distributed databases.

Reliability is broadly defined as the probability that a system is running (not down) at a certain time point, whereas availability is the probability that the 1 Page 658 of 893 system is continuously available during a time interval. When the data and DBMS software are distributed over several sites, one site may fail while other sites continue to operate. Only the data and software that exist at the failed site cannot be accessed. This improves both reliability and availability.

5. Improved performance:
6. Easier expansion: In a distributed environment, expansion of the system in terms of adding more data, increasing database sizes, or adding more processors is much easier.

Data Fragmentation

Split a relation into logically related and correct parts. A relation can be fragmented in two ways:

Horizontal Fragmentation

Vertical Fragmentation

Horizontal fragmentation:

It is a horizontal subset of a relation which contains those of tuples which satisfy selection conditions. Consider the Employee relation with selection condition (DNO = 5). All tuples satisfy this condition will create a subset which will be a horizontal fragment of Employee relation. A selection condition may be composed of several conditions connected by AND or

OR. Derived horizontal fragmentation: It is the partitioning of a primary relation to other secondary relations which are related with foreign keys.

Vertical fragmentation

It is a subset of a relation which is created by a subset of columns. Thus a vertical fragment of a relation will contain values of selected columns. There is no selection condition used in vertical fragmentation. Consider the Employee relation. A vertical fragment of can be created by keeping the values of Name, Bdate, Sex, and Address. Because there is no condition for creating a vertical fragment, each fragment

must include the primary key attribute of the parent relation Employee. In this way all vertical fragments of a relation are connected.

Fragmentation Representation

Horizontal fragmentation

1. Each horizontal fragment on a relation can be specified by a sCi(R) operation in the relational algebra.

2. Complete horizontal fragmentation

3. A set of horizontal fragments whose conditions C1, C2, …, Cn include all the tuples in R- that is, every tuple in R satisfies (C1 OR C2 OR … OR Cn).

4. Disjoint complete horizontal fragmentation: No tuple in R

satisfies (Ci AND Cj) where i ≠ j.

5. To reconstruct R from horizontal fragments a UNION is applied.

Vertical fragmentation

1. A vertical fragment on a relation can be specified by a PLi(R) operation in the relational algebra.

2. Complete vertical fragmentation

3. A set of vertical fragments whose projection lists L1, L2, …, Ln include all the attributes in R but share only the primary key of R. In this case the projection lists satisfy the following two conditions:

4. L1 » L2 » ... » Ln = ATTRS (R)

5. Li « Lj = PK(R) for any i j, where ATTRS (R) is the set of attributes of R and PK(R) is the primary key of R.

6. To reconstruct R from complete vertical fragments a OUTER UNION is applied.

Mixed (Hybrid) fragmentation

1. A combination of Vertical fragmentation and Horizontal fragmentation.
2. This is achieved by SELECT-PROJECT operations which is represented by PLi(sCi (R)).
3. If C = True (Select all tuples) and L ≠ ATTRS(R), we get a vertical fragment, and if C ≠ True and L ≠ ATTRS(R), we get a mixed fragment.
4. If C = True and L = ATTRS(R), then R can be considered a fragment

Fragmentation schema

A definition of a set of fragments (horizontal or vertical or horizontal and vertical) that includes all attributes and tuples in the database that satisfies the condition that the whole database can be reconstructed from the fragments by applying some sequence of UNION (or OUTER JOIN) and UNION operations.

Allocation schema

It describes the distribution of fragments to sites of distributed databases. It can be fully or partially replicated or can be partitioned

Data Replication

Database is replicated to all sites. In full replication the entire database is replicated and in partial replication some selected part is replicated to some of the sites. Data replication is achieved through a replication schema. Data Distribution (Data Allocation) is relevant only in the case of partial replication or partition.The selected portion of the database is distributed to the database sites.

Types of Distributed Database Systems

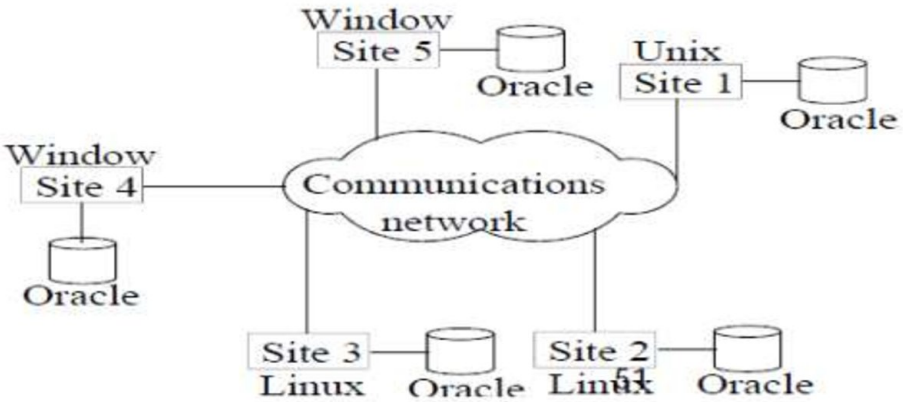
Homogeneous

1)All sites of the database system have identical setup, i.e., same database system software.

2)The underlying operating system may be different.

For example, all sites run Oracle or DB2, or Sybase or some other database system.

The underlying operating systems can be a mixture of Linux, Window, Unix, etc.



Homogeneous System

Fig 4.5. Homogeneous System

Heterogeneous System

1) Federated: Each site may run different database system but the data access is managed through a single conceptual schema. This implies that the degree of local autonomy is minimum. Each site must adhere to a centralized access policy. There may be a global schema.

2) Multidatabase: There is no one conceptual global schema. For data access a schema is constructed dynamically as needed by the application software.

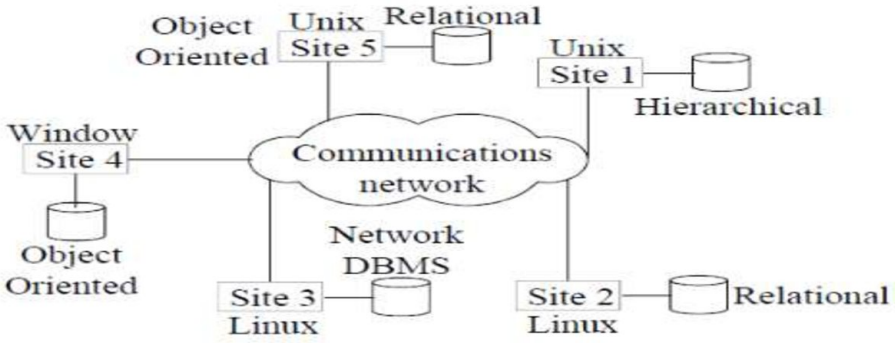


Fig 4.6 Hetrogeneous System

Query processing in Distributed Databases:

Issues:

1) Cost of transferring data (files and results) over the network. This cost is usually high so some optimization is necessary. Example relations:

Employee at site 1 and Department at Site 2

Example: Employee at site 1. 10,000 rows. Row size = 100 bytes. Table size = 106 bytes.

Fname	Minit	Lname	SSN	Bdate	Address	Sex	Salary	Superssn	Dno

Department at Site 2. 100 rows. Row size = 35 bytes. Table size = 3,500 bytes.

For each employee, retrieve employee name and department name Where the employee works.

A: PFname, Lname, Dname (Employee Dno = Dnumber Department)

The result of this query will have 10,000 tuples, assuming that every employee is related to a department. Suppose each result tuple is 40 bytes long. The query is submitted at site 3 and the result is sent to this site.

Problem: Employee and Department relations are not present at site 3.

Strategies:

1. Transfer Employee and Department to site 3. Total transfer bytes = 1,000,000 + 3500 = 1,003,500 bytes.

2. Transfer Employee to site 2, execute join at site 2 and send the result to site 3 Query result size = 40 * 10,000 = 400,000 bytes. Total transfer size = 400,000 + 1,000,000 = 1,400,000 bytes.

3. Transfer Department relation to site 1, executes the join at site 1, and sends the result to site 3.

Total bytes transferred = 400,000 + 3500 = 403,500 bytes. Optimization criteria: minimizing data transfer.

Preferred approach: strategy 3.

Concurrency Control and Recovery

Distributed Databases encounter a number of concurrency control and recovery problems which are not present in centralized databases. Some of them are listed below.

1) Dealing with multiple copies of data items

The concurrency control must maintain global consistency. Likewise the recovery mechanism must recover all copies and maintain consistency after recovery.

2) Failure of individual sites

Database availability must not be affected due to the failure of one or two sites and the recovery scheme must recover them before they are available for use

3) Communication link failure

This failure may create network partition which would affect database availability even though all database sites may be running

4) Distributed commit

A transaction may be fragmented and they may be executed by a number of sites. This require a two or three-phase commit approach for transaction commit

1) Distributed deadlock

Since transactions are processed at multiple sites, two or more sites may get involved in deadlock. This must be resolved in a distributed manner.

Concurrency Control and Recovery

Distributed Concurrency control based on a distributed copy of a data item

Primary site technique: A single site is designated as a primary site which serves as a coordinator for transaction management

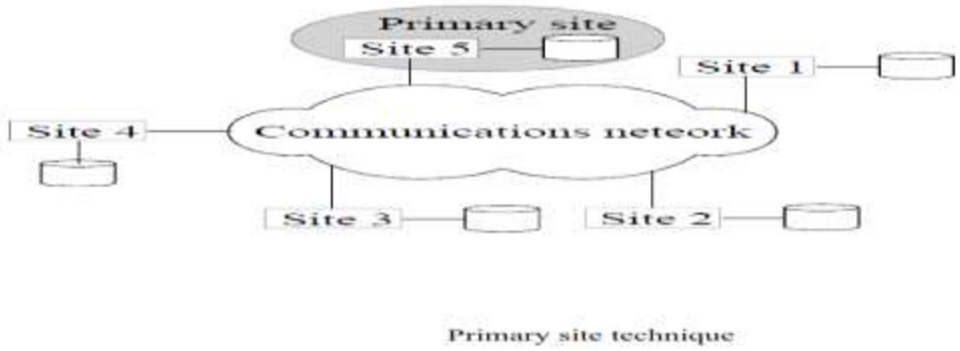


Fig 4.7 Primary Site Technique

Transaction management:

Concurrency control and commit are managed by this site.In two phase locking, this site manages locking and releasing data items. If all transactions follow two-phase policy at all sites, then serializability is guaranteed

Primary Copy Technique:

In this approach, instead of a site, a data item partition is designated as primary copy. To lock a data item just the primary copy of the data item is locked.

Advantages:

Since primary copies are distributed at various sites, a single site is not overloaded with locking and unlocking requests.

Disadvantages:

Identification of a primary copy is complex. A distributed directory must be maintained, possibly at all sites.

Recovery from a coordinator failure

In both approaches a coordinator site or copy may become unavailable. This will require the selectionof a new coordinator.

Primary site approach with no backup site

Aborts and restarts all active transactions at all sites. Elects a new coordinator and initiates transaction processing.

Primary site approach with backup site:

Suspends all active transactions, designates the backup site as the primary site and identifies a new back up site. Primary site receives all transaction management information to resume processing.

Primary and backup sites fail or no backup site:

Use election process to select a new coordinator site.

Concurrency control based on voting:

There is no primary copy of coordinator.Send lock request to sites that have data item. If majority of sites grant lock then the requesting transaction gets the data item.Locking information (grant or denied) is sent to all these sites.

To avoid unacceptably long wait, a time-out period is defined. If the requesting transaction does not get any vote information then the transaction is aborted.

Architectural Models

Some of the common architectural models are −

• Client - Server Architecture for DDBMS

• Peer - to - Peer Architecture for DDBMS

• Multi - DBMS Architecture

Client - Server Architecture for DDBMS

This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface. However, they have some functions like consistency checking and transaction management.

The two different clients - server architecture are −

• Single Server Multiple Client

• Multiple Server Multiple Client (shown in the following diagram)

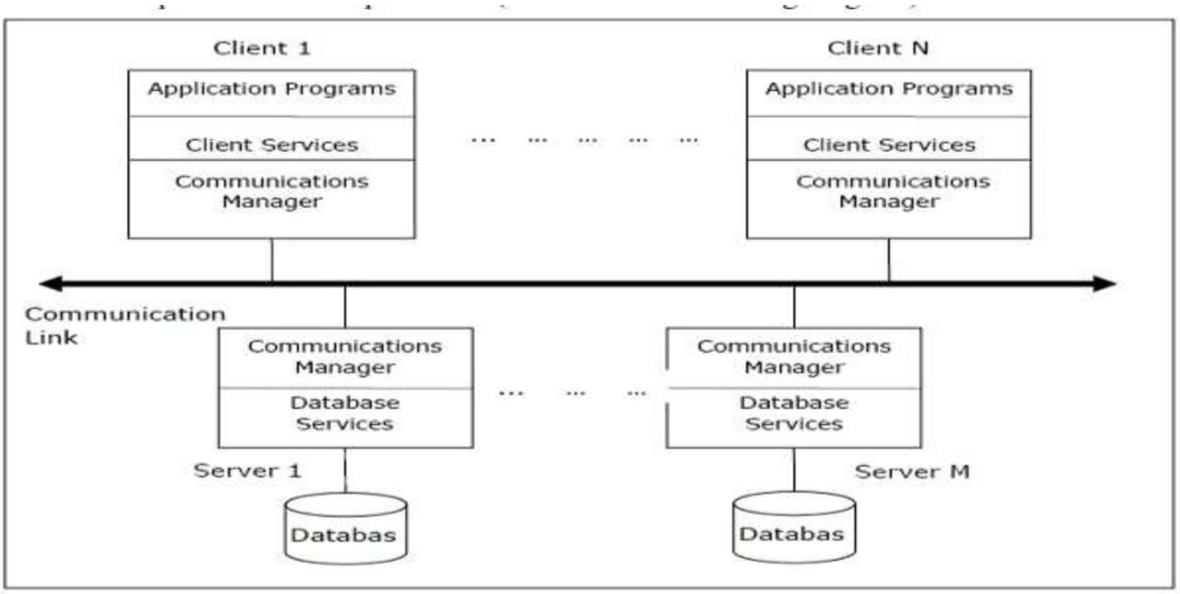


Fig 4.8 Client – Server Model

Peer- to-Peer Architecture for DDBMS

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.

This architecture generally has four levels of schemas −

• Global Conceptual Schema − Depicts the global logical view of data.

• Local Conceptual Schema − Depicts logical data organization at each site.

• Local Internal Schema − Depicts physical data organization at each site.

• External Schema − Depicts user view of data.

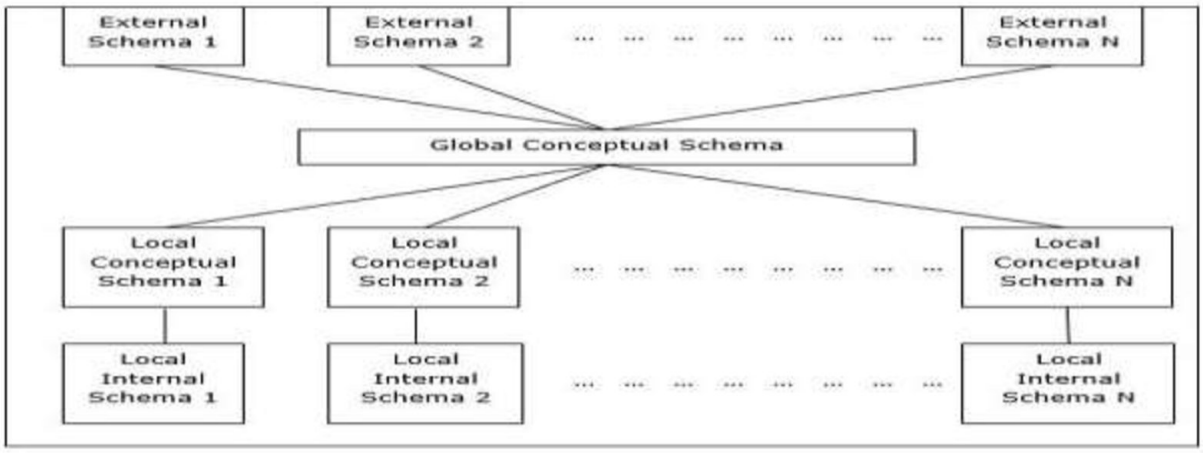


Fig 4. 9 Peer to Peer Architecture

Multi - DBMS Architectures

This is an integrated database system formed by a collection of two or more autonomous database systems.

Multi-DBMS can be expressed through six levels of schemas −

• Multi-database View Level − Depicts multiple user views comprising of subsets of the integrated distributed database.

• Multi-database Conceptual Level − Depicts integrated multi-database that comprises of global logical multi-database structure definitions.

• Multi-database Internal Level − Depicts the data distribution across different sites and multi-database to local data mapping.

• Local database View Level − Depicts public view of local data.

• Local database Conceptual Level − Depicts local data organization at each site.

• Local database Internal Level − Depicts physical data organization at each site.

There are two design alternatives for multi-DBMS −

• Model with multi-database conceptual level.

• Model without multi-database conceptual level.

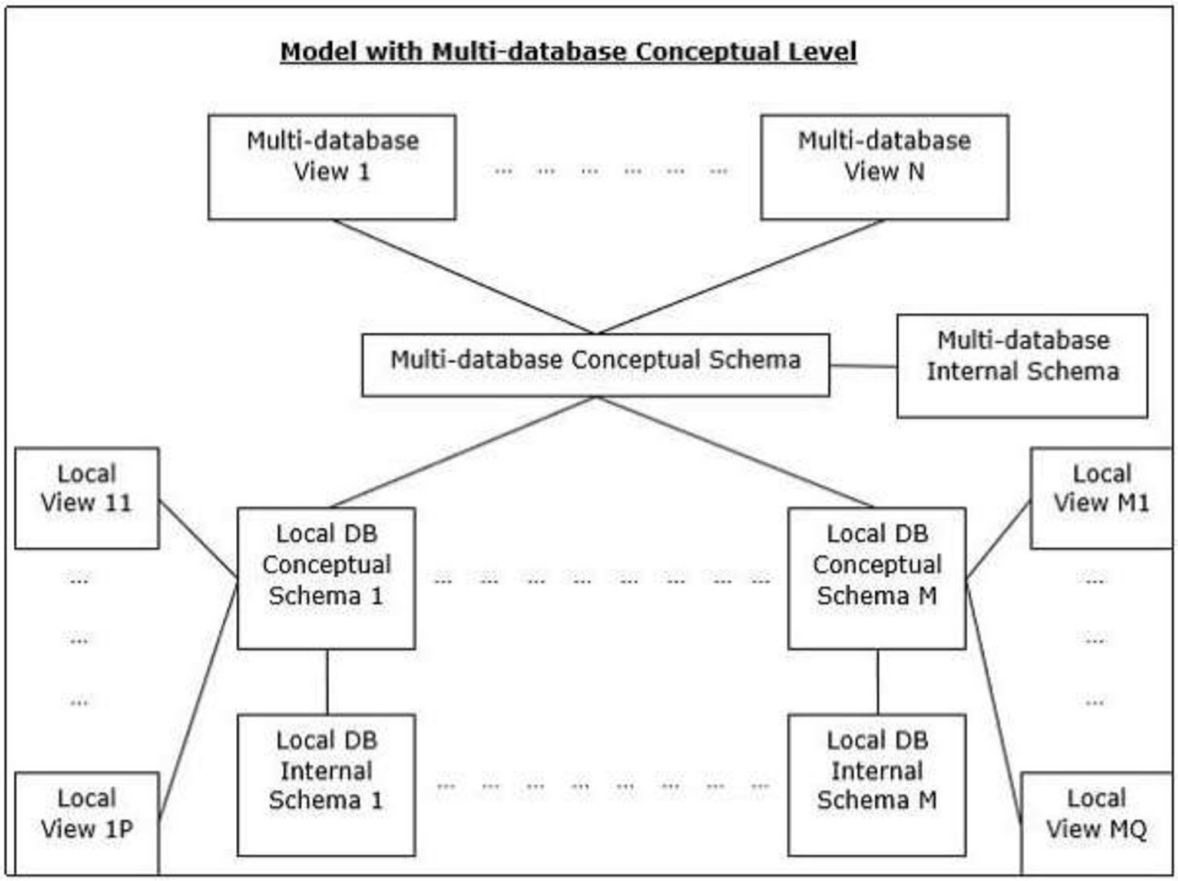


Fig 4.10 Multi DBMS Architecture

Database Security

Types of Security

– Legal and ethical issues

– Policy issues

– System-related issues

The need to identify multiple security levels

A DBMS typically includes a database security and authorization subsystem that is responsible for ensuring the security portions of a database against unauthorized access.

Two types of database security mechanisms:

Discretionary security mechanisms

The typical method of enforcing discretionary access control in a database system is based on the granting and revoking privileges.

Mandatory security mechanisms

In many applications, and additional security policy is needed that classifies data and users based on security classes. This approach as mandatory access control, would typically be combined with the discretionary access control mechanisms.

Security Issues in Databases

• The security mechanism of a DBMS must include provisions for restricting access to the database as a whole; this function is called access control and is handled by creating user accounts and passwords to control login process by the DBMS.

• The security problem associated with databases is that of controlling the access to a statistical database, which is used to provide statistical information or summaries of values based on various criteria.

• The counter measures to statistical database security problem is called inference control measures.

• Another security is that of flow control, which prevents information from flowing in such a way that it reaches unauthorized users.

• Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called covert channels.

• A final security issue is data encryption, which is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type communication network.

• The data is encoded using some coding algorithm. An unauthorized user who access encoded data will have difficulty deciphering it, but authorized users are given decoding or decrypting algorithms (or keys) to decipher data.

Database Security and the DBA

The database administrator (DBA) is the central authority for managing a database system. The DBA's responsibilities include granting privileges to users who need to use the system and classifying users and data in accordance with the policy of the organization. The DBA has a DBA account in the DBMS, sometimes called a

system or superuser account, which provides powerful capabilities :

1. Account creation

2. Privilege granting
3. Privilege revocation
4. Security level assignment

The DBA is responsible for the overall security of the database system. Action 1 is access control, whereas 2 and 3 are discretionary and 4 is used to control mandatory authorization.
Whenever a person or group of person s need to access a database system, the individual or group must first apply for a user account. The DBA will then create a new account number and password for the user if there is a legitimate need to access the database.
If any tampering with the database is suspected, a database audit is performed, which consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.
A database log that is used mainly for security purposes is sometimes called an audit trail.

Types of Discretionary Privileges

The account level: At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database. The privileges at the account level apply to the capabilities provided to the account itself and can include the CREATE SCHEMA or CREATE TABLE privilege, to create a schema or base relation; the ALTER privilege, To apply schema changes such adding or removing attributes from relations; the DROP privilege, to delete relations or views; the MODIFY privilege, to insert, delete, or update tuples; and the SELECT privilege, to retrieve information from the database by using a SELECT query.

The relation (or table level): At this level, the DBA can control the privilege to access each individual relation or view in the database. To control the granting and revoking of relation privileges, each relation R in a database is assigned an owner account, which is typically the account that was used when the relation was created in the first place. The owner of a relation is given all privileges on that relation. In SQL2, the DBA can assign an owner to a whole schema by creating the schema and associating the appropriate authorization identifier with that schema, using the CREATE SCHEMA command. The owner account holder can pass privileges on any of the owned relation to other users by granting privileges to their accounts.

In SQL the following types of privileges can be granted on each individual relation R:

• SELECT (retrieval or read) privilege on R: Gives the account retrieval privilege. In SQL this gives the account the privilege to use the SELECT statement to retrieve tuples from R.

• MODIFY privileges on R: This gives the account the capability to modify tuples of R. In SQL this privilege is further divided into UPDATE, DELETE, and INSERT privileges to apply the corresponding SQL command to R. In addition, both the INSERT and UPDATE privileges can specify that only certain attributes can be updated by the account.

** Note that to create a view, the account must have SELECT privilege on all relations involved in the view definition. The mechanism of views is an important discretionary authorization mechanism in its own right.

For example, if the owner A of a relation R wants another account B to be able to retrieve only some fields of R, then A can create a view V of R that includes only those attributes and then grant SELECT on V to B. The same applies to limiting B to retrieving only certain tuples of R; a view V' can be created by defining the view by means of a query that selects only those tuples from R that A wants to allow B to access.

Revoking Privileges

For example, the owner of a relation may want to grant the SELECT privilege to a user for a specific task and then revoke that privilege once the task is completed. Hence, a mechanism for revoking privileges is needed.

In SQL, a REVOKE command is included for the purpose of canceling privileges.

Example of Grant and Revoke Commands in SQL Suppose that the DBA creates four accounts -- A1, A2, A3, and A4-- and wants only A1 to be able to create base relations; then the DBA must issue the following GRANT command in SQL:

```
GRANT CREATETAB TO A1;
```

In SQL2 the same effect can be accomplished by having the DBA issue a CREATE SCHEMA command as follows:

CREATE SCHEMA EXAMPLE AUTHORIZATION A1;
GRANT privilege_name ON object_name
TO {user_name |PUBLIC |role_name} [WITH GRANT OPTION];

For eg:

GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;
GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2

The REVOKE command removes user access rights or privileges to the database objects.
The Syntax for the REVOKE command is:

REVOKE privilege_name ON object_name

FROM {user_name |PUBLIC |role_name}
REVOKE SELECT ON EMPLOYEE FROM A3

Privilege for Views

Suppose that A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege. The limitation is to retrieve only the NAME, BDATE, and ADDRESS attributes and only for the tuples with DNO=5.

A1 then create the view:

CREATE VIEW A3EMPLOYEE AS
SELECT NAME, BDATE, ADDRESS

FROM EMPLOYEE
WHERE DNO = 5;

After the view is created, A1 can grant SELECT on the view A3EMPLOYEE to A3 as follows:

GRANT SELECT ON A3EMPLOYEE TO A3 WITH GRANT OPTION;

Mandatory Access Control and Role-Based Access Control for Multilevel Security
** The discretionary access control techniques of granting and revoking privileges on relations has traditionally been the main security mechanism for relational database systems.

This is an all-or-nothing method: A user either has or does not have a certain privilege.
In many applications, and additional security policy is needed that classifies data and users based on security classes. This approach as mandatory access control, would typically be combined with the discretionary access control mechanisms.

Typical security classes are

Top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest:

$TS \geq S \geq C \geq U$

The commonly used model for multilevel security, known as the Bell-LaPadula model.
Classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications, T, S, C, or U. Clearance (classification) of a subject S as class(S) and to the classification of an object O as class(O).

Two restrictions are enforced on data access based on the subject/object classifications:

1. A subject S is not allowed read access to an object O unless class(S) $\geq$ class(O). This is known as the simple security property.
2. A subject S is not allowed to write an object O unless class(S) $\leq$ class(O). This known as the star property (or * property).
3. To incorporate multilevel security notions into the relational database model, it is common to consider attribute values and tuples as data objects. Hence, each attribute A is associated with a classification attribute C in the schema, and each attribute value in a tuple is associated with a corresponding security classification.

4. In addition, in some models, a tuple classification attribute TC is added to the relation attributes to provide a classification for each tuple as a whole. Hence, a multilevel relation schema R with n attributes would be represented as R(A1,C1,A2,C2, …, An,Cn,TC)
where each Ci represents the classification attribute associated with attribute Ai.

The value of the TC attribute in each tuple t – which is the highest of all attribute classification values within t – provides a general classification for the tuple itself, whereas each Ci provides a finer security classification for each attribute value within the tuple.

A multilevel relation will appear to contain different data to subjects (users) with different clearance levels.

In some cases, it is possible to store a single tuple in the relation at a higher classification level and produce the corresponding tuples at a lower-level classification through a process known as filtering.

In general, the entity integrity rule for multilevel relations states that all attributes that are members of the apparent key must not be null and must have the same security classification within each individual tuple.

Comparing Discretionary Access Control and Mandatory Access Control

• Discretionary Access Control (DAC) policies are characterized by a high degree of flexibility, which makes them suitable for a large variety of application domains.
• The main drawback of DAC models is their vulnerability to malicious attacks, such as Trojan horses embedded in application programs.
• By contrast, mandatory policies ensure a high degree of protection in a way, they prevent any illegal flow of information.
• Mandatory policies have the drawback of being too rigid and they are only applicable in limited environments.
• In many practical situations, discretionary policies are preferred because they offer a better trade-off between security and applicability.

Role-Based Access Control

Role-based access control (RBAC) emerged rapidly in the 1990s as a proven technology for managing and enforcing security in large-scale enterprisewide systems. Its basic notion is that permissions are associated with roles, and users are assigned to appropriate roles. Roles can be created using the CREATE ROLE and DESTROY ROLE commands.
The GRANT and REVOKE commands discussed under DAC can then be used to assign and revoke privileges from roles.

CREATE ROLE role_name [WITH ADMIN {CURRENT_USER | CURRENT_ROLE}]
With the above syntax, a role with role_name is created and immediately assigned to the current user or the currently active role is passed on to other users. The default usage is WITH ADMIN CURRENT_USER.

• RBAC appears to be a viable alternative to traditional discretionary and mandatory access controls; it ensures that only authorized users are given access to certain data or resources.
• Many DBMSs have allowed the concept of roles, where privileges can be assigned to roles.
• Role hierarchy in RBAC is a natural way of organizing roles to reflect the organization's lines of authority and responsibility.

• Another important consideration in RBAC systems is the possible temporal constraints that may exist on roles, such as time and duration of role activations, and timed triggering of a role by an activation of another role.

• Using an RBAC model is highly desirable goal for addressing the key security requirements of Web-based applications.

In contrast, discretionary access control (DAC) and mandatory access control (MAC) models lack capabilities needed to support the security requirements emerging enterprises and Web-based applications.

An Overview of Three-Tier Client/Server Architecture

Division of DBMS functionality among the three tiers can vary

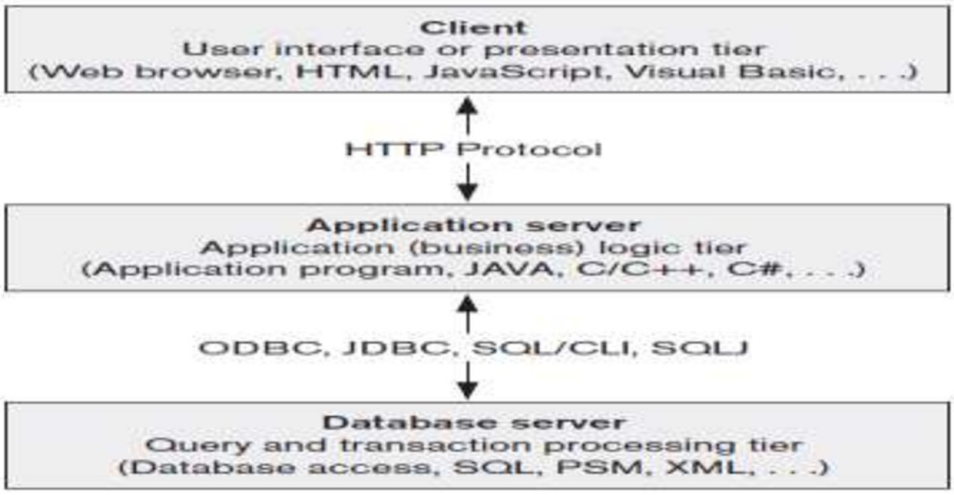


Fig 4.11 Three Tier Client Server Architecture

Case Study/Debate – Railway

Reservation System

RAILWAY RESEVATION SYSTEM

PROBLEM STATEMENT

A software has to be developed for automating the manual railway reservation system. The system should have distributed functionalities as described below:-

1. RESERVE SEAT - A passenger should be able to reserve a seat in the train specified by him if available. For this he has to fill a reservation form with the details about his journey. The clerk checks for the availability of the seat in the train and if the seat is available then he makes entries regarding train name, train number, date of journey, boarding station, destination. The passenger is the asked to pay the fair .After making payment the passenger can collect the ticket from the clerk.

2. CANCEL RESERVATION -There may arise a case when the passenger wants to cancel his reservation .For this he has to fill a cancellation form providing all the details about the ticket reserved by him. The clerk then checks for the entries from the database and cancels the reservation finally returning the ticket amount with some deduction

3. UPDATE TRAIN INFORMATION & REPORT GENERATION :- Only the Administrator has the right to make changes in train details(train name, train no. etc.).The system should also be able to generate report when needed in the form of reservation charts , train schedule charts etc.

4. LOGIN Only the user with specified login id & password can get access to the system. This provides security from unauthorized access.

5. VIEW RESERVATION STATUS & TRAIN SCHEDULE All the users should be able to see the information about the reservation status & train schedule, train name, train number etc.

:::USE CASE DIAGRAM:::

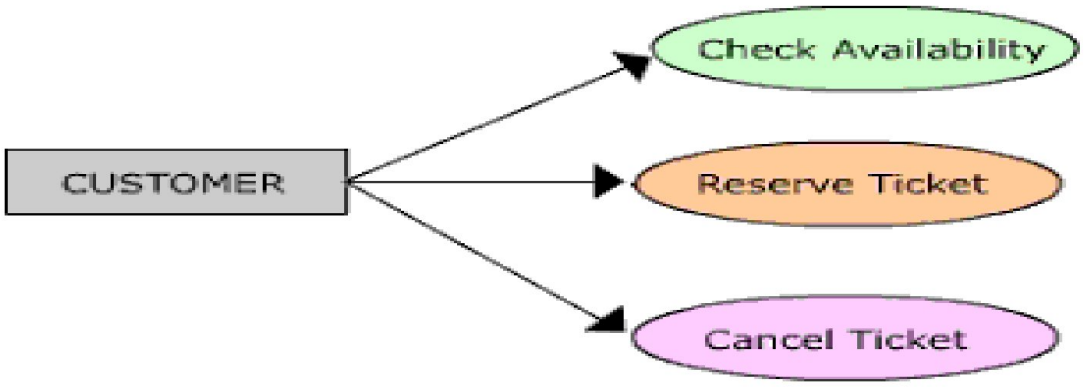


Fig 4.12 Customer Activities

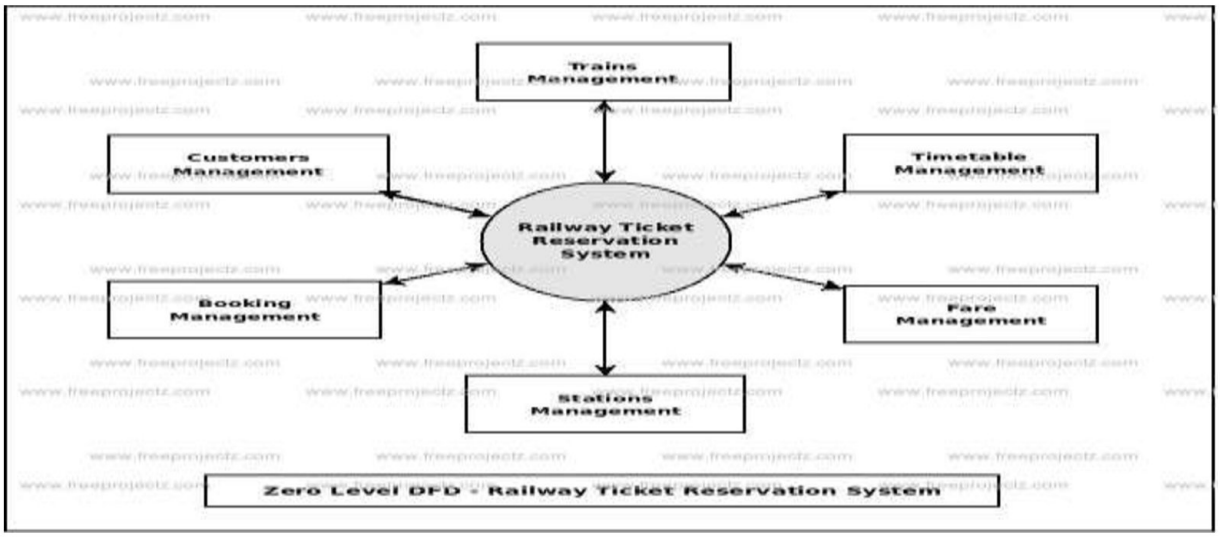


Fig. 4.13(a). DFD - Railway Reservation System

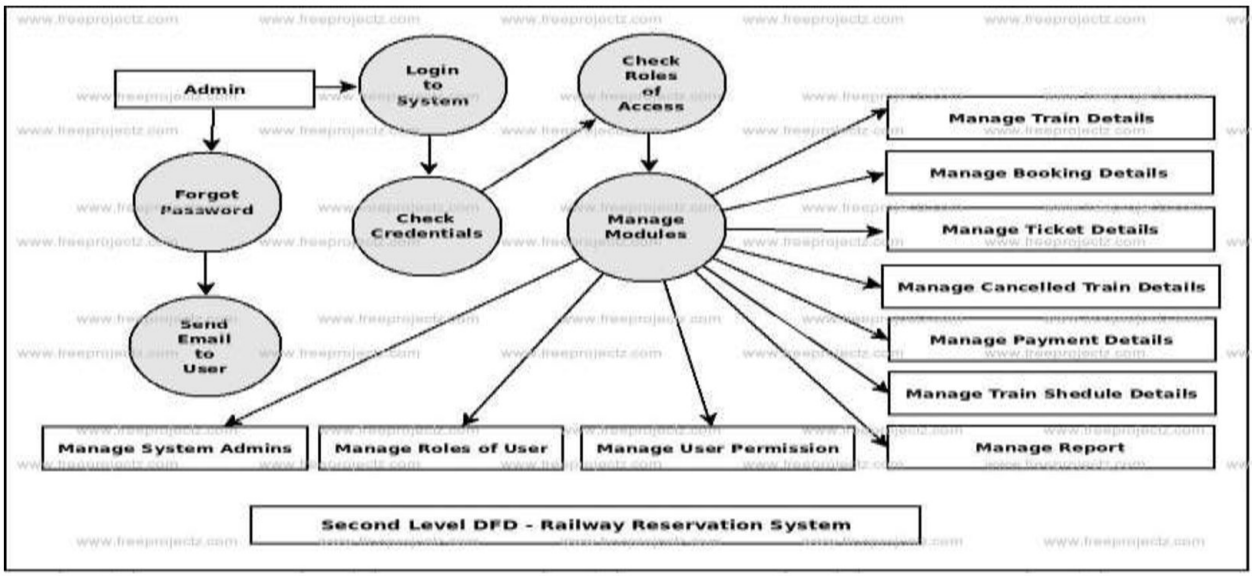


Fig. 4.13(b). DFD - Railway Reservation System

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – V- Database Management Systems – SCSA1301

IV. Object Database and Current Trends

Concepts for Object Database - Emerging Database Technologies and Application - Introduction to Data warehousing & Data mining –Applications of Data mining.

Object Oriented DBMS Concepts

Relational DBMSs support a small, fixed collection of data types (e.g. integer, dates, string, etc.) which has proven adequate for traditional application domains such as administrative and business data processing. RDBMSs support very high-level queries, query optimization, transactions, backup and crash recovery, etc.

However, many other application domains need complex kinds of data such as CAD/CAM, multimedia repositories, and document management. To support such applications, DBMSs must support complex data types.

Object-oriented strongly influenced efforts to enhance database support for complex data and led to the development of object- database systems.

Object-database systems have developed along two distinct paths:

Object-Oriented Database Systems. The approach is heavily influenced by OO programming languages and can be understood as an attempt to add DBMS functionality to a programming language environment. The Object Database Management Group (ODMG) has developed a standard Object Data Model (ODM) and Object Query Language (OQL), which are the equivalent of the SQL standard for relational database systems.

Object-Relational Database Systems. ORDB systems can be thought of as an attempt to extend relational database systems with the functionality necessary to support a broader class of application domains, provide a bridge between the relational and object-oriented paradigms. This approach attempts to get the best of both

Object and Class

A conceptual **entity** is anything that exists and can be distinctly identified.

E.g. a person, an employee, a car, a part

In an OO system, all conceptual entities are modeled as objects.

- An object has **structural** properties defined by a finite set of **attributes** and **behavioural**

properties defined by a finite set of **methods**.

- Each object is associated with a logical **non-reusable** and unique **object identifier** (**OID**).
- The OID of an object is **independent** of the values of its attributes.
- All objects with the same set of attributes and methods are grouped into a **class**, and form **instances** of that class.
- Classes are classified as **lexical classes** and **non-lexical classes**.
- A **lexical class** contains objects that can be directly represented by their values.

  E.g. integer, string.
- A **non-lexical class** contains objects, each of which is represented by a set of attributes and methods.
- Instances of a non-lexical class are referred to by their **OID**s.

  E.g. PERSON, EMPLOYEE, PART are non-lexical classes.
- In some OO systems, a class is treated as an object also, and therefore processes its own attributes and methods. These properties are called **class attributes** and **class methods**.(Similar to **static fields** or **class variables** in Java)
- A class EMPLOYEE can have class attributes called NO_of_EMPLOYEES which holds a count of the number of employee instances in the class, and NEXT_ENO which holds the employee number of the next new employee.
- The class EMPLOYEE can have a class method called **NEW** which is used to construct new instances of the class.

Attribute

The domain **of an attribute of a non-lexical class A can be one of the following:**

Case (a): a lexical class such as integer, string. An attribute with this domain is called a data-valued attribute**.**

Case (b): a non-lexical class B. An attribute with this domain is called an **entity-valued attribute**.

- ✓ Note the **recursive** nature of this definition.
- ✓ There is an implicit binary relationship between attributes A andB.
- ✓ The value of the attribute A is the OID of an instance of B, which must exist before it can be assigned to the attribute.This provides **referential integrity**.

Case (c): a **set**, set(E), where E is either a lexical class or a non-lexical class. An attribute with this domain is called a **set-valued attribute**.

- ✓ If E is lexical, values from E are stored in the set.
- ✓ If E is non-lexical, members of the set can either be an **instance of E or its subclasses**. In this case, the set comprises instances from possibly heterogeneous classes. Only OID of each instance is stored in the set.

Method

A method of an object is invoked by sending a **message** (which is normally the method name) to the object. Such a **message-passing** mechanism represents a **binary interaction** between the sender of the message and the recipient.

- ✓ A method's specification is represented by a **method signature**, which provides the method name and information on the types of the method's input parameters and its results.

- ✓ The implementation of the method is separated from the specification. This provide some degrees of **data independence.**

Methods play an important role in defining object semantics.

E.g. When an employee is fired, we need to delete the employee information from the employee file, delete the employee from the employee-project file, and insert the employee information into a history file, etc.

One methd called "**Fire-employee**" can be defined that incorporates this sequence of actions.

Class Hierarchy

Given 2 classes X and Y, **X ISA Y** means that each instance of X is also an instance of Y. We call X a **subclass** of Y and Y a **superclass** of X.

E.g. Manger isa Employee

A class hierarchy provides an **inheritance mechanism** which allows a class to inherit properties (attributes and methods) from its superclasses.

In **single inheritance** systems, a class can have **at most one** direct superclass and therefore can only inherit from that superclass.

The class hierarchy forms a tree.

In **multiple inheritance** systems, a class can have **more than one** direct superclass.

The class hierarchy is a lattice.

Note: In multiple inheritance systems, a class may inherit properties and methods from different super classes and therefore may have **inheritance conflicts.**

Extensibility

Extensibility is another important feature of the OO paradigm. It allows the creation of new data types, i.e. user-defined types, and operations on these new data types from built-in atomic data types and user defined data types using the type constructor.

A **type constructor** is a mechanism for building new domains.

A **complex object** is built using type constructors such as sets, tuples, lists and nested combinations.

A combination of an user-defined type and its associated methods is called an abstract data type (ADT).

Object Oriented Concepts

- ✓ Abstract Data Types
    - Class definition, provides extension to complex attribute types
- ✓ Encapsulation
    - Implementation of operations and object structure hidden
- ✓ Inheritance
    - Sharing of data within hierarchy scope, supports code reusability
- ✓ Polymorphism
    - Operator overloading

Encapsulation of Operations, Methods, and Persistence

- Specifying **Object Behavior** via Class Operations:

✓ The main idea is to define the **behavior** of a type of object based on the **operations** that can be externally applied to objects of that type.

✓ In general, the **implementation** of an operation can be specified in a *general-purpose programming language* that provides flexibility and power in defining the operations.

✓ For database applications, the requirement that all objects be completely encapsulated is too stringent.

Specifying Object Persistence via Naming and Reachability:

Naming Mechanism:

Assign an object a unique persistent name through which it can be retrieved by this and other programs.

Reachability Mechanism:

Make the object reachable from some persistent object.
An object B is said to be **reachable** from an object A if a sequence of references in the object graph lead from object A to object B.

Type and Class Hierarchies and Inheritance

Type (class) Hierarchy

A type in its simplest form can be defined by giving it a type name and then listing the names of its visible (**public**) functions

When specifying a type in this section, we use the following format, which does not specify arguments of functions, to simplify the discussion:

TYPE_NAME: function, function, . . . , function

Example:

PERSON: Name, Address, Birthdate, Age, SSN

Subtype:

When the designer or user must create a new type that is similar but not identical to an already defined type

Supertype:

It inherits all the functions of the subtype

Example (1):

PERSON: Name, Address, Birthdate, Age, P-ID

EMPLOYEE: Name, Address, Birthdate, Age, P-ID, Salary, HireDate, Seniority

STUDENT: Name, Address, Birthdate, Age, P-ID, Major, GPA

OR:

EMPLOYEE **subtype-of** PERSON: Salary, HireDate, Seniority

STUDENT **subtype-of** PERSON: Major, GPA

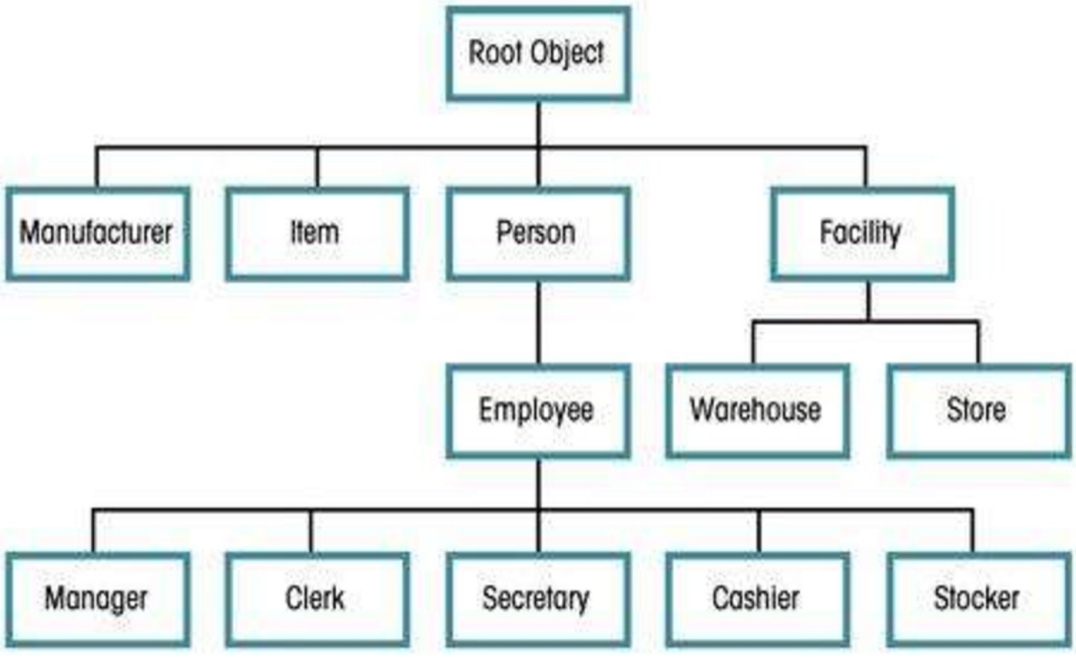


Fig. 5.1 Class Hierarchy For The EDLP Retail Corporation

Type and Class Hierarchies and Inheritance

Extents:

In most OO databases, the collection of objects in an extent has the same type or class.

However, since the majority of OO databases support types, we assume that extents are collections of objects of the same type for the remainder of this section.

Persistent Collection:

This holds a collection of objects that is stored permanently in the database and hence can be accessed and shared by multiple programs

Transient Collection:

This exists temporarily during the execution of a program but is not kept when the program terminates

Other Objected-Oriented Concepts

- **Polymorphism (Operator Overloading):**
    - This concept allows the same operator name or symbol to be bound to two or more different implementations of the operator, depending on the type of objects to which the operator is applied
    - **For example + can be:**
        - Addition in integers
        - Concatenation in strings (of characters)

Complex Objects

- **Unstructured complex object:**
- These is provided by a DBMS and permits the storage and retrieval of large objects that are needed by the database application.
- Typical examples of such objects are bitmap images and long text strings (such as documents); they are also known as binary large objects, or BLOBs for short.
- This has been the standard way by which Relational DBMSs have dealt with supporting complex objects, leaving the operations on those objects outside the RDBMS.

Structured complex object:

This differs from an unstructured complex object in that the object's structure is defined by repeated application of the type constructors provided by the OODBMS. Hence, the object

structure is defined and known to the OODBMS.The OODBMS also defines methods or operations on it.

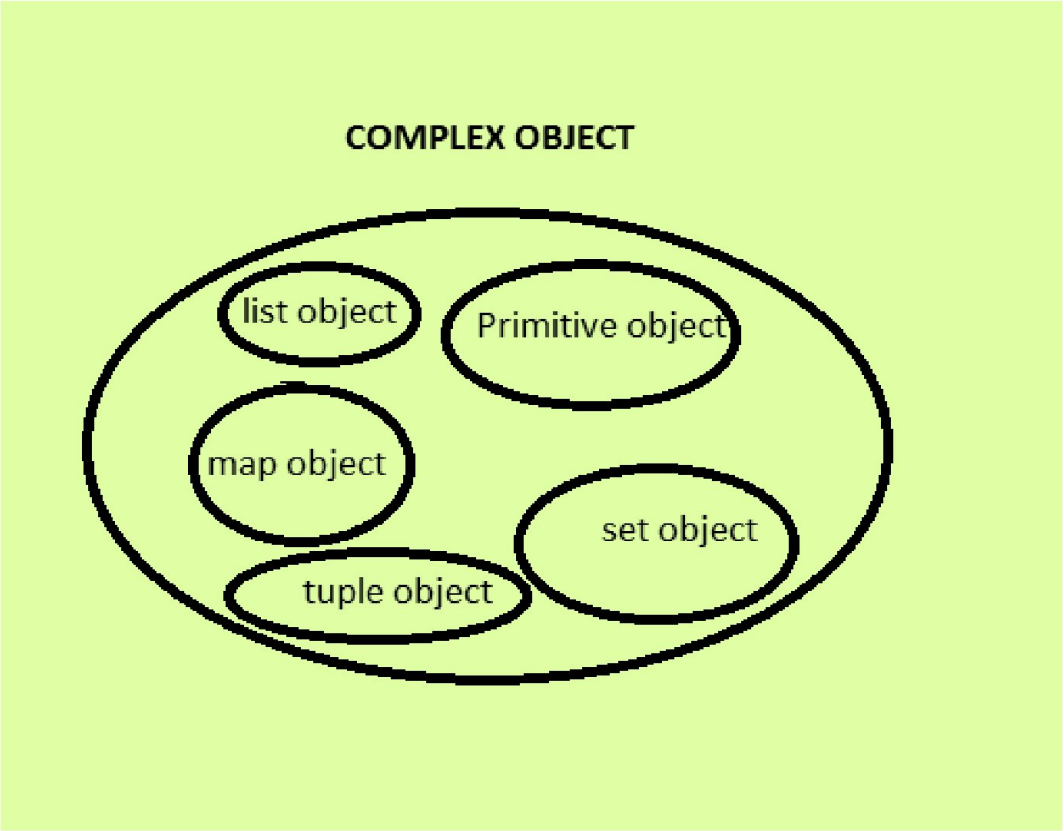


Fig. 5.2 Complex Object

What is Object Oriented Database? (OODB)

- A database system that incorporates all the important object-oriented concepts
- Some additional features
    - Unique Object identifiers
    - Persistent object handling

Advantages

- Designer can specify the structure of objects and their behavior (methods)
- Better interaction with object-oriented languages such as Java and C++
- Definition of complex and user-defined types
- Encapsulation of operations and user-defined methods

Disadvantages

- Lack of theoretical foundation.
- Lack of standard *ad hoc* query language.
- Lack of business data design and management tools.
- Lack of resources.

Object Query Language(OQL)

- Declarative query language
  - Not computationally complete
- Syntax based on SQL (select, from, where)
- Additional flexibility (queries with user defined operators and types)

The following is a sample query

"what are the names of the black product?"

Select distinct p.name

From products p

Where p.color = "black"

- Valid in both SQL and OQL, but results are different

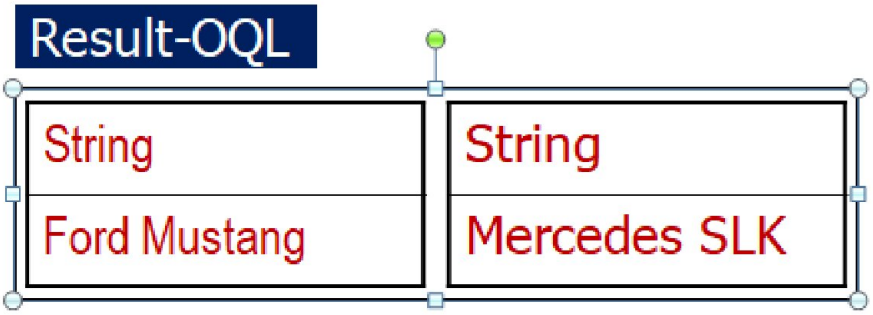
Product no	Name	Color
P1	Ford Mustang	Black
P2	Toyota Celica	Green
P3	Mercedes SLK	Black

Result of the query(SQL-Returns table with rows)

Name

Ford Mustang

Mercedes SLK



- The statement queries a object-oriented database

=> Returns a collection of objects.

OQL	SQL
Object	Tuple
Collection of objects	Table

Emerging Database Technologies and Application

- Mobile Databases
- Multimedia Databases
- Geographic Information Systems
- GENOME Data Management

1. **Mobile Databases**

✓ Recent advances in portable and wireless technology led to mobile computing, a new dimension in data communication and processing.

✓ Portable computing devices coupled with wireless communications allow clients to access data from virtually anywhere and at any time.

✓ There are a number of hardware and software problems that must be resolved before the capabilities of mobile computing can be fully utilized.

✓ Some of the software problems – which may involve data management, transaction management, and database recovery – have their origins in distributed database systems.

In mobile computing, the problems are more difficult, mainly:

✓ The limited and intermittent connectivity afforded by wireless communications.

✓ The limited life of the power supply (battery).

- ✓ The changing topology of the network.
- ✓ In addition, mobile computing introduces new architectural possibilities and challenges.

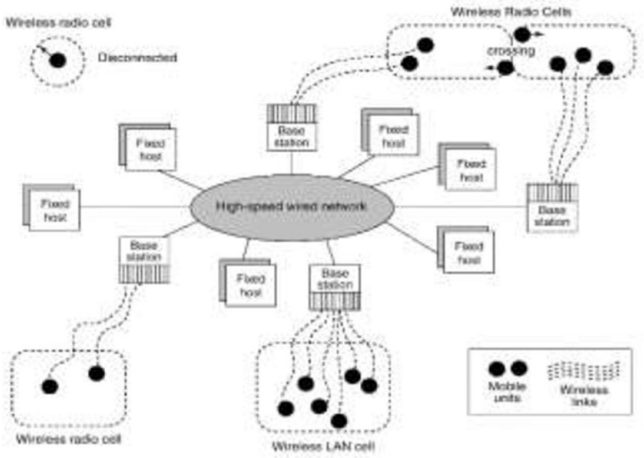


Figure 5.3 Mobile Computing Architecture

2. Multimedia Databases

In the years ahead multimedia information systems are expected to dominate our daily lives.

- ✓ Our houses will be wired for bandwidth to handle interactive multimedia applications.
- ✓ Our high-definition TV/computer workstations will have access to a large number of databases, including digital libraries, image and video databases that will distribute vast amounts of multisource multimedia content.

Types of multimedia data are available in current systems

- ✓ **Text:** May be formatted or unformatted. For ease of parsing structured documents, standards like SGML and variations such as HTML are being used.
- ✓ **Graphics:** Examples include drawings and illustrations that are encoded using some descriptive standards (e.g. CGM, PICT, postscript).
- ✓ **Images:** Includes drawings, photographs, and so forth, encoded in standard formats such as bitmap, JPEG, and MPEG. Compression is built into JPEG and MPEG.
    - o These images are not subdivided into components. Hence querying them by content (e.g., find all images containing circles) is nontrivial.
- ✓ **Animations:** Temporal sequences of image or graphic data.
- ✓ **Video:** A set of temporally sequenced photographic data for presentation at specified

rates– for example, 30 frames per second.

- ✓ **Structured audio:** A sequence of audio components comprising note, tone, duration, and so forth
- ✓ **Composite or mixed multimedia data:** A combination of multimedia data types such as audio and video which may be physically mixed to yield a new storage format or logically mixed while retaining original types and formats. Composite data also contains additional control information describing how the information should be rendered

Fig 5.4 Multimedia Data

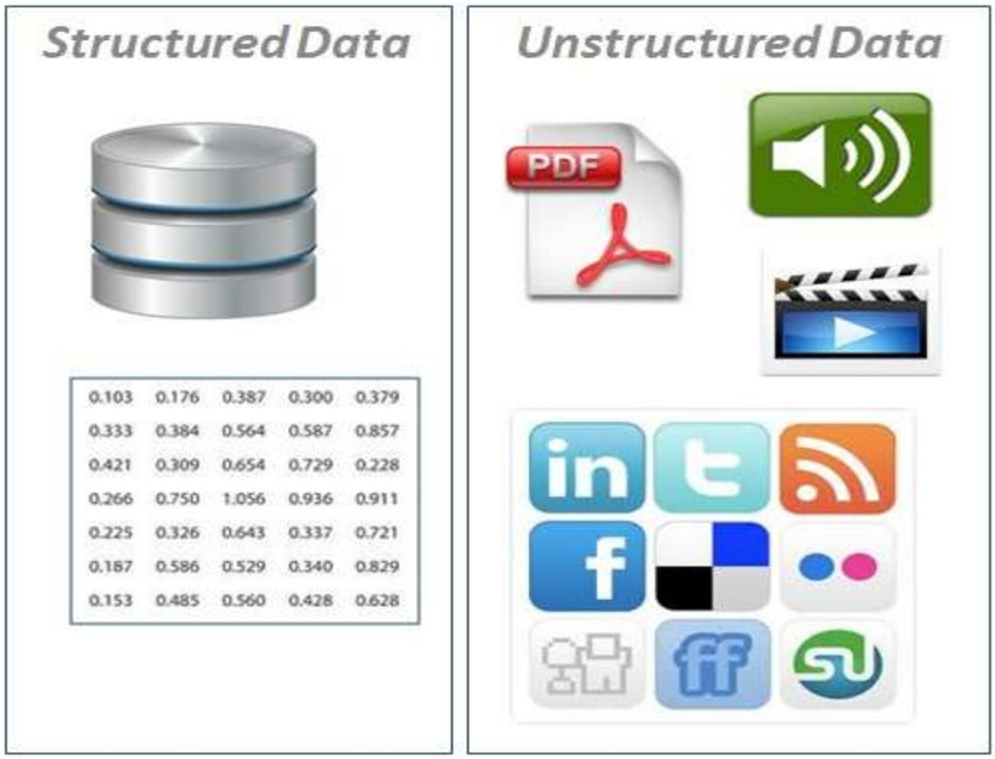


Fig 5.5 Structured and Unstructured Data

Applications based on their data management characteristics:

- ✓ **Repository applications:** A large amount of multimedia data as well as metadata is stored for retrieval purposes. Examples include repositories of satellite images, engineering drawings and designs, space photographs, and radiology scanned pictures.
- ✓ **Presentation applications:** A large amount of applications involve delivery of multimedia data subject to temporal constraints; simple multimedia viewing of video data, for example, requires a system to simulate VCR-like functionality.
- ✓ **Collaborative work using multimedia information:** This is a new category of applications in which engineers may execute a complex design task by merging drawings, fitting subjects to design constraints, and generating new documentation, change notifications, and so forth. Intelligent healthcare networks as well as telemedicine will involve doctors collaborating among themselves, analyzing multimedia patient data and information in real time as it is generated.

Multimedia Database Applications

- ✓ **Large-scale applications of multimedia databases can be expected encompasses a large number of disciplines and enhance existing capabilities.**
- ✓ **Documents and records management**
- ✓ **Knowledge dissemination**
- ✓ **Education and training**
- ✓ **Marketing, advertising, retailing, entertainment, and travel**
- ✓ **Real-time control and monitoring**

3. Geographic Information System

The scope of GIS broadly encompasses two types of data:

- ✓ Spatial data, originating from maps, digital images, administrative and political boundaries, roads, transportation networks, physical data, such as rivers, soil characteristics, climatic regions, land elevations, and
- ✓ Non-spatial data, such as socio-economic data (like census counts), economic data, and sales or marketing information. GIS is a rapidly developing domain that offers highly innovative approaches to meet

some challenging technical demands.

Fig 5.6 GIS

Categorization of GIS:

- Cartographic applications
- Digital terrain modelling applications
- Geographic objects applications

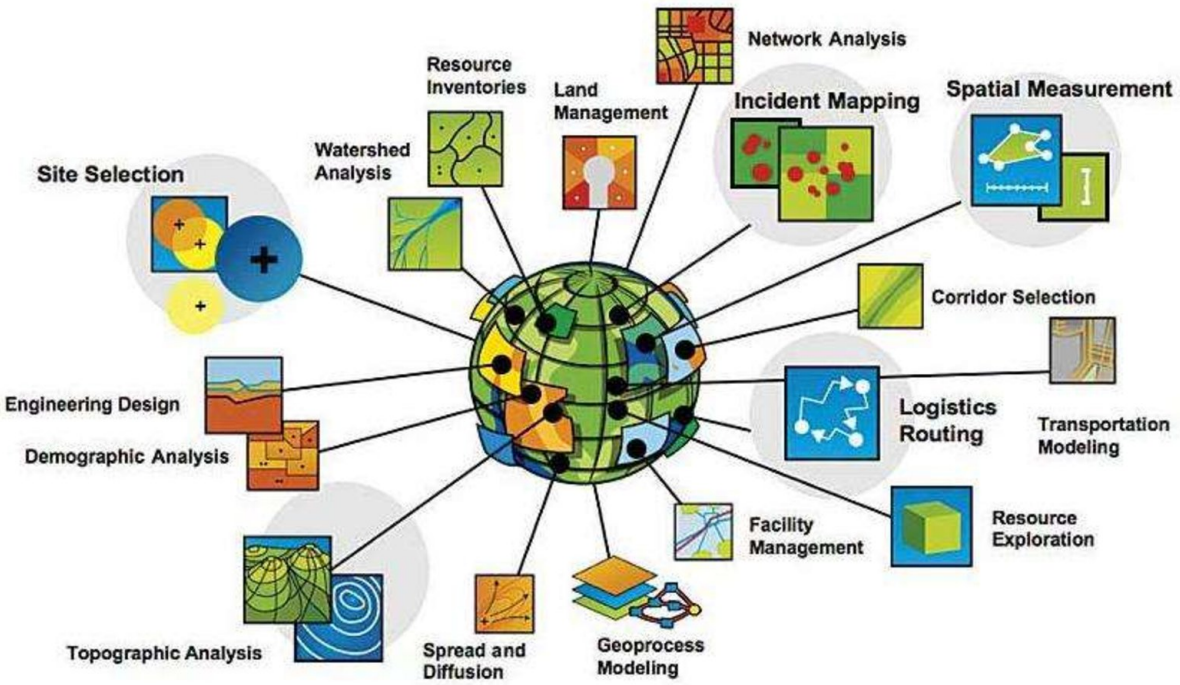
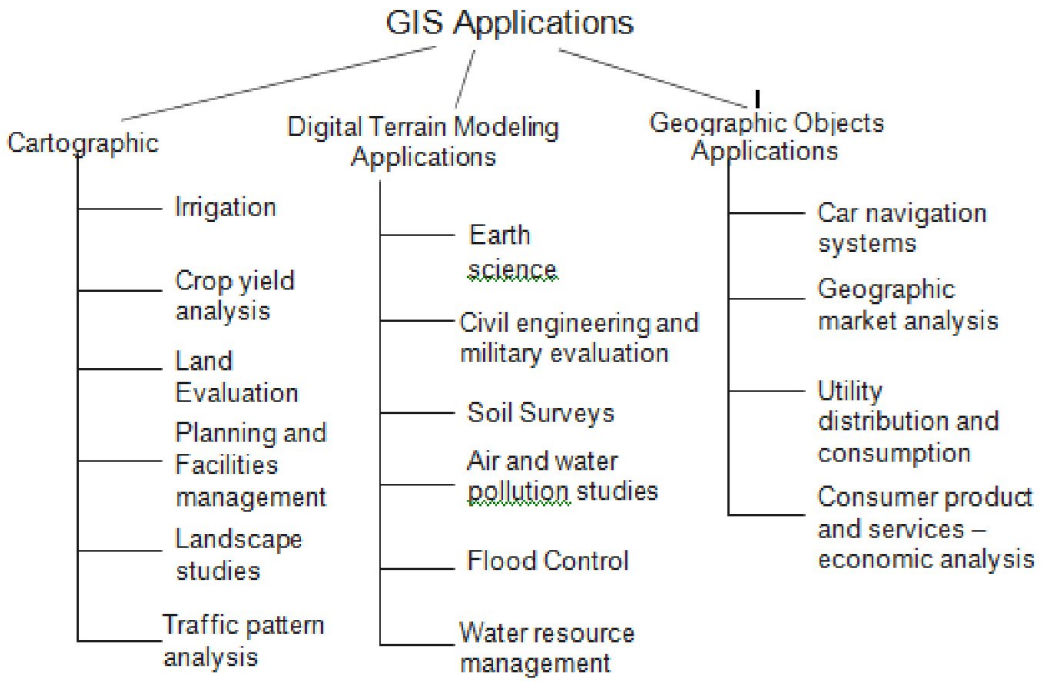


Fig 5.6 GIS Applications



Data Warehousing

Introduction

A data warehouse is constructed by integrating data from multiple heterogeneous sources. It supports analytical reporting, structured and/or ad hoc queries and decision making. This tutorial adopts a step-by-step approach to explain all the necessary concepts of data warehousing. The term "Data Warehouse" was first coined by Bill Inmon in 1990.

According to Inmon, a data warehouse is a subject oriented, integrated, time-variant, and non- volatile collection of data. This data helps analysts to take informed decisions in an organization. An operational database undergoes frequent changes on a daily basis on account of the transactions that take place. Suppose a business executive wants to analyze previous feedback on any data such as a product, a supplier, or any consumer data, then the executive will have no data available to analyze because the previous data has been updated due to transactions. A data warehouses provides us generalized and consolidated data in multidimensional view. Along with generalized and consolidated view of data, a data warehouses also provides us Online Analytical Processing (OLAP) tools. These tools help us in interactive and effective analysis of data in a multidimensional space. This analysis results in data generalization and data mining. Data mining functions such as association, clustering, classification, prediction can be integrated with OLAP operations to enhance the interactive mining of knowledge at multiple level of abstraction. That's why data warehouse has now become an important platform for data analysis and online analytical processing.

Understanding a Data Warehouse

- A data warehouse is a database, which is kept separate from the organization's operational database.
- There is no frequent updating done in a data warehouse.
- It possesses consolidated historical data, which helps the organization to analyze its business.
- A data warehouse helps executives to organize, understand, and use their data to take strategicdecisions.
- Data warehouse systems help in the integration of diversity of application systems.
- A data warehouse system helps in consolidated historical data analysis.

Why a Data Warehouse is Separated from Operational Databases

A data warehouses is kept separate from operational databases due to the following reasons:

- An operational database is constructed for well-known tasks and workloads such as searching particular records, indexing, etc. In contract, data warehouse queries are often complex and they present a general form of data.
- Operational databases support concurrent processing of multiple transactions. Concurrency control and recovery mechanisms are required for operational databases to ensure robustness and consistency of the database.
- An operational database query allows to read and modify operations, while an OLAP query needs only **read only** access of stored data.
- An operational database maintains current data. On the other hand, a data warehouse maintains historical data.

Data Warehouse Features

The key features of a data warehouse are discussed below:

- **Subject Oriented** - A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations. These subjects can be product, customers, suppliers, sales, revenue, etc. A data warehouse does not focus on the ongoing operations; rather it focuses on modelling and analysis of data for decision making.
- **Integrated** - A data warehouse is constructed by integrating data from heterogeneous sources such as relational databases, flat files, etc. This integration enhances the effective analysis of data.
- **Time Variant** - The data collected in a data warehouse is identified with a

particular time period. The data in a data warehouse provides information from the historical point of view.

- **Non-volatile** - Non-volatile means the previous data is not erased when new data is added to it. A data warehouse is kept separate from the operational database and therefore frequent changes in operational database is not reflected in the data warehouse.

  Note: A data warehouse does not require transaction processing, recovery, and concurrency controls, because it is physically stored and separate from the operational database.

  Data Warehouse Applications - As discussed before, a data warehouse helps business executives to organize, analyze, and use their data for decision making. A data warehouse serves as a sole part of a plan-execute-assess "closed-loop" feedback system for the enterprise management. Data warehouses are widely used in the following fields:

  - Financial services
  - Banking services
  - Consumer goods
  - Retail sectors
  - Controlled manufacturing

Types of Data Warehouse

Information processing, analytical processing, and data mining are the three types of data warehouse applications that are discussed below:

- **Information Processing** - A data warehouse allows to process the data stored in it. The data can be processed by means of querying, basic statistical analysis, reporting using crosstabs, tables, charts, or graphs.
- **Analytical Processing** - A data warehouse supports analytical processing of the information stored in it. The data can be analyzed by means of basic OLAP operations, including slice-and-dice, drill down, drill up, and pivoting.
- **Data Mining** - Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction. These mining results can be presented using the visualization tools.

No.	Data Warehouse (OLAP)	Operational Database(OLTP)
1	It involves historical processing of information.	It involves day-to-day processing.
2	OLAP systems are used by knowledge workers such as executives, managers, and analysts.	OLTP systems are used by clerks, DBAs, or database professionals.
3	It is used to analyze the business.	It is used to run the business.
4	It focuses on Information out.	It focuses on Data in.
5	It is based on Star Schema, Snowflake Schema, and Fact Constellation Schema.	It is based on Entity Relationship Model.
6	It focuses on Information out.	It is application oriented.
7	It contains historical data.	It contains current data.
8	It provides summarized and consolidated data.	It provides primitive and highly detailed data.
9	It provides summarized and multidimensional view of data.	It provides detailed and flat relational view of data.
10	The number of users is in hundreds.	The number of users is in thousands.
11	The number of records accessed is in millions.	The number of records accessed is in tens.
12	The database size is from 100GB to 100 TB.	The database size is from 100 MB to 100 GB.
13	These are highly flexible.	It provides high performance.

Data Mining

Data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Data Mining is defined as extracting information from huge sets of data. In other words, we can say that data mining is the procedure of mining knowledge from data. The information or knowledge extracted so can be used for any of the following applications −

- Market Analysis
- Fraud Detection
- Customer Retention
- Production Control
- Science Exploration

Data Mining Applications

Data mining is highly useful in the following domains −

- Market Analysis and Management
- Corporate Analysis &Risk Management
- FraudDetection

Apart from these, data mining can also be used in the areas of production control, customer retention, science exploration, sports, astrology, and Internet Web Surf-Aid.

Market Analysis and Management

Listed below are the various fields of market where data mining is used −

- **Customer Profiling** − Data mining helps determine what kind of people buy what kind ofproducts.
- **Identifying Customer Requirements** − Data mining helps in identifying the best products for different customers. It uses prediction to find the factors that may attract new customers.
- **Cross Market Analysis** − Data mining performs association/correlations between productsales.
- **Target Marketing** − Data mining helps to find clusters of model customers who share the same characteristics such as interests, spending habits, income,etc.
- **Determining Customer purchasing pattern** − Data mining helps in determining customer purchasingpattern.
- **Providing Summary Information** − Data mining provides us various multidimensional summaryreports.

Corporate Analysis and RiskManagement

Data mining is used in the following fields of the Corporate Sector−

- **Finance Planning and Asset Evaluation** − It involves cash flow analysis and prediction, contingent claim analysis to evaluate assets.
- **Resource Planning** − It involves summarizing and comparing the resources and spending.
- **Competition** − It involves monitoring competitors and marketdirections.

Fraud Detection

Data mining is also used in the fields of credit card services and telecommunication to detect frauds. In fraud telephone calls, it helps to find the destination of the call, duration of

the call, time of the day or week, etc. It also analyzes the patterns that deviate from expected norms.

Knowledge discovery in databases (KDD)

Knowledge discovery in databases (KDD) is the process of discovering useful knowledge from a collection of data. This widely used data mining technique is a process that includes data preparation and selection, data cleansing, incorporating prior knowledge on data sets and interpreting accurate solutions from the observed results.

Here is the list of steps involved in the knowledge discovery process −

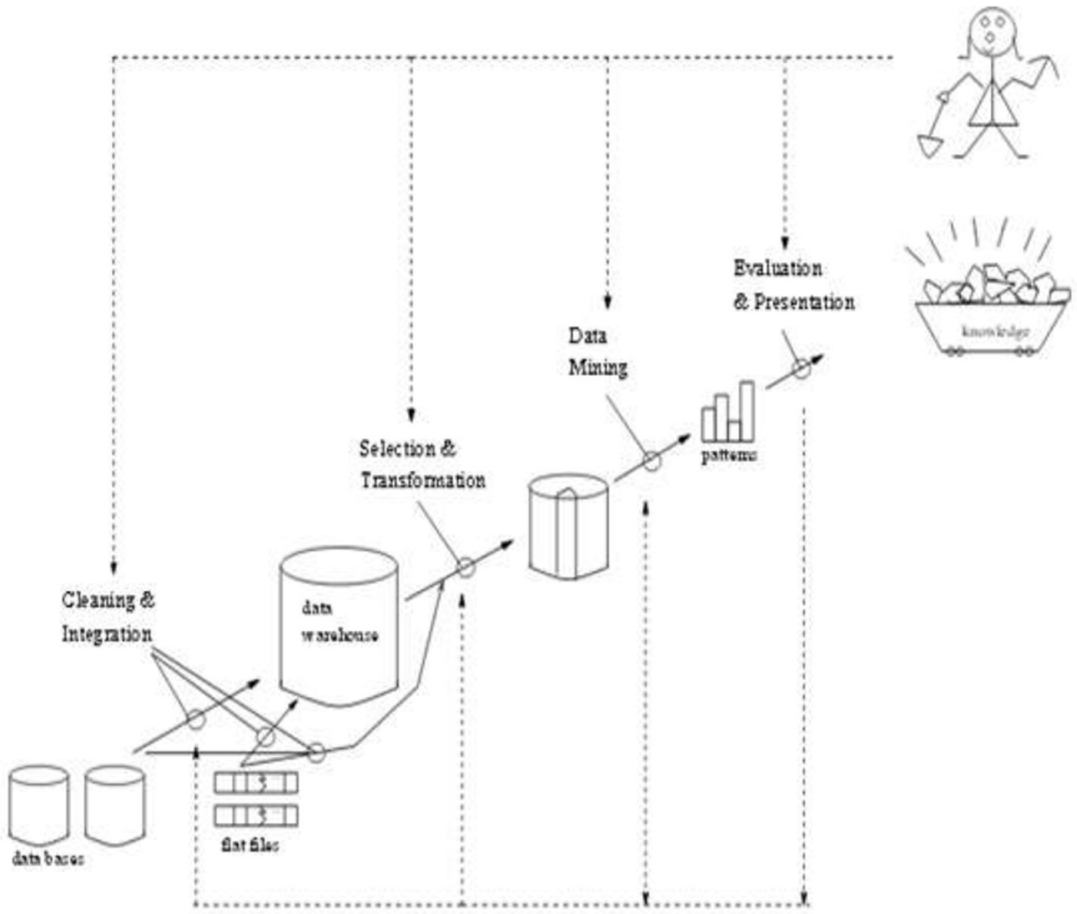


Fig. 5.7 Data Mining as a process of knowledge discovery

- **Data Cleaning** − In this step, the noise and inconsistent data isremoved.
- **Data Integration** − In this step, multiple data sources arecombined.
- **Data Selection** − In this step, data relevant to the analysis task are retrieved from the database.
- **Data Transformation** − In this step, data is transformed or consolidated intoforms appropriate for mining by performing summary or aggregationoperations.
- **Data Mining** − Inthis step, intelligent methods are applied in order to extract data patterns.

- **Pattern Evaluation** − In this step, data patterns areevaluated.
- **Knowledge Presentation** − Inthis step, knowledge isrepresented.

Data Mining Applications

Here is the list of areas where data mining is widely used −

- Financial Data Analysis
- Retail Industry
- Telecommunication Industry
- Biological Data Analysis
- Other Scientific Applications
- Intrusion Detection

Financial Data Analysis

The financial data in banking and financial industry is generally reliable and of high quality which facilitates systematic data analysis and data mining. Some of the typical cases are as follows −

- Design and construction of data warehouses for multidimensional data analysis and data mining.
- Loan payment prediction and customer credit policy analysis.
- Classification and clustering of customers for targeted marketing.
- Detection of money laundering and other financial crimes.

Retail Industry

Data Mining has its great application in Retail Industry because it collects large amount of data from on sales, customer purchasing history, goods transportation, consumption and services. It is natural that the quantity of data collected will continue to expand rapidly because of the increasing ease, availability and popularity of the web.

Data mining in retail industry helps in identifying customer buying patterns and trends that lead to improved quality of customer service and good customer retention and satisfaction. Here is the list of examples of data mining in the retail industry

- Design and Construction of data warehouses based on the benefits of data mining.
- Multidimensional analysis of sales, customers, products, time and region.
- Analysis of effectiveness of sales campaigns.

- Customer Retention.
- Product recommendation and cross-referencing of items.

Telecommunication Industry

Today the telecommunication industry is one of the most emerging industries providing various services such as fax, pager, cellular phone, internet messenger, images, e-mail, web data transmission, etc. Due to the development of new computer and communication technologies, the telecommunication industry is rapidly expanding. This is the reason why data mining is become very important to help and understand the business.

Data mining in telecommunication industry helps in identifying the telecommunication patterns, catch fraudulent activities, make better use of resource, and improve quality of service. Here is the list of examples for which data mining improves telecommunication services −

- Multidimensional Analysis of Telecommunication data.
- Fraudulent pattern analysis.
- Identification of unusual patterns.
- Multidimensional association and sequential patterns analysis.
- Mobile Telecommunication services.
- Use of visualization tools in telecommunication data analysis.

Biological Data Analysis

In recent times, we have seen a tremendous growth in the field of biology such as genomics, proteomics, functional Genomics and biomedical research. Biological data mining is a very important part of Bioinformatics. Following are the aspects in which data mining contributes for biological data analysis −

- Semantic integration of heterogeneous, distributed genomic and proteomic databases.
- Alignment, indexing, similarity search and comparative analysis multiple nucleotide sequences.
- Discovery of structural patterns and analysis of genetic networks and protein pathways.
- Association and path analysis.
- Visualization tools in genetic data analysis.

Other Scientific Applications

The applications discussed above tend to handle relatively small and homogeneous data sets for which the statistical techniques are appropriate. Huge amount of data have been collected from scientific domains such as geosciences, astronomy, etc. A large amount of data sets is being generated because of the fast numerical simulations in various fields such as climate and ecosystem modeling, chemical engineering, fluid dynamics, etc. Following are the applications of data mining in the field of Scientific Applications −

- Data Warehouses and data preprocessing.
- Graph-based mining.
- Visualization and domain specific knowledge.

Intrusion Detection

Intrusion refers to any kind of action that threatens integrity, confidentiality, or the availability of network resources. In this world of connectivity, security has become the major issue. With increased usage of internet and availability of the tools and tricks for intruding and attacking network prompted intrusion detection to become a critical component of network administration. Here is the list of areas in which data mining technology may be applied for intrusion detection −

- Development of data mining algorithm for intrusion detection.
- Association and correlation analysis, aggregation to help select and build discriminating attributes.
- Analysis of Stream data.
- Distributed data mining.
- Visualization and query tools.

Trends in Data Mining

Data mining concepts are still evolving and here are the latest trends that we get to see in this field −

- Application Exploration.
- Scalable and interactive data mining methods.
- Integration of data mining with database systems, data warehouse systems and web database systems.
- Standardization of data mining query language.

- Visual data mining.
- New methods for mining complex types of data.
- Biological data mining.
- Data mining and software engineering.
- Web mining.
- Distributed data mining.
- Real time data mining.
- Multi database data mining.
- Privacy protection and information security in data mining.