

CHAPTER 8

DATABASES

Syllabus: ER model, relational model (relational algebra, tuple calculus), database design (integrity constraints, normal forms), query languages (SQL), file structures (sequential files, indexing, B and B+ trees), transactions and concurrency control.

8.1 INTRODUCTION

Database is a collection of organized information so that it can easily be searched, retrieved, managed and updated. A database management system (DBMS) is a set of programs designed to manage a database. It enables users to store, retrieve and modify information in a database with utmost efficiency along with security features. DBMS is applicable to various day-to-day fields such as transactions in banking; airline/railway/hotel reservations; maintenance of student information in schools/universities; online retails; marketing and sales etc. It also allows its users to create their own databases. Different types of DBMS are available such as hierarchical, network, relational and object oriented.

8.1.1 Traditional File Processing Approach

File processing approach is generally more accurate and faster than the manual database system. Each user is responsible for the defining and implementation of the files required for the specific application. The implementation of required files sometimes creates redundancy of data, for example, one user keeps records for the savings account of the customer and another user may create the loan account of the same customer. This causes the duplicity of the records of the same customer. So, this practice is not feasible for real time applications. There is a need of centralized management of data. The data is created once and then accessed by different users. The data should be shared for different transactions. It should be self-describing in nature, which means the

database system contains not only data but it describes the description of the database structure.

8.1.2 Database Management System

A database is a collection of related data. Data is a collection of raw facts or figures, processed to form information. Database management system is a collection of programs for the creation and maintenance of database. It is an efficient and reliable approach to retrieve data for many users. It provides various functions such as:

1. **Redundancy control:** It provides redundancy by removing duplicity of data by following rules of normalization.
2. **Data independence:** It provides independence to application programs from details of data representation and storage. It also provides an abstract view of the data to insulate application code from such details.
3. **Data integrity:** It promotes and enforces some integrity rules for reducing data redundancy and increasing data consistency.
4. **Concurrency control:** It supports sharing of data, so, it has to provide an approach for managing concurrent access of the database. Hence, preserving the inconsistent state and integrity of the data.
5. **Transaction management:** It provides an approach to ensure that either all the updates for a given transaction will execute or that none of them would execute.
6. **Backup and recovery:** It provides mechanisms for backing up data periodically and recovering from different types of failures, thus, preventing loss of data.
7. **Non-Procedural query language:** It provides with query language for retrieval and manipulation of data.
8. **Security:** It protects unauthorized access in the database. It ensures the access to authorized users.

8.2 COMPONENTS OF DATABASE SYSTEMS

DBMS consists of several components, namely software, hardware, data, procedures and data access language. These components are responsible for the definition, collection, management and use of data within the environment. Figure 8.1 shows the components of database system. The description of each component is as follows:

1. **Software:** It is the collection of programs used by the computers within the database system. It is used to handle, control and manage the database. It includes the following software:

- Operating system software like Microsoft Windows, Linux OS, Mac OS.
- DBMS software such as Oracle 8I, MySQL, Access (Jet, MSDE), SQL Server etc.
- Network softwares are used for sharing share the data of database among multiple users.
- Application programs are developed like C++, VB, dotnet etc. are used to access database in dbms. These are used to access and manipulate the data in the database.

2. **Hardware:** It consists of all system's physical devices such as computers, storage devices, I/O channels, electromechanical devices etc. It also includes peripherals, such as, keyboard, mouse, modems, printers, etc.
3. **Data:** It is the collection of facts. The database contains the data and the metadata.
4. **Procedures:** There are the instructions and rules to design and use the database system. These includes the following:

- Steps for the installation of DBMS
- Steps to use the DBMS or application program
- Steps for the backup of DBMS
- Steps to change the structure of DBMS
- Steps for the generation of reports.

5. **Data access language:** The users can use it to access the data to and from the database. The function of data access language is the entry of new data, manipulation of the existing data and the retrieval of the existing data in the database. The most popular database access language is SQL (Structured Query Language). Users can perform these functions with the help of commands. The role of administrator is to access, to create and to maintain the database.
6. **People:** Persons involved to access, to create and to maintain the database are called users. These are of various types according to the role performed by them (Fig. 8.1). These are as follows:

- **System Administrator:** The role of system administrator is to supervise the general operations of DBMS.
- **Database Administrator:** The role of database administrator (DBA) is to manage the DBMS.
- **Database Designer:** The role of database designer is to design the structure of the database.
- **Application Programmer:** The role of application programmer is to create the data entry forms, reports and procedures.
- **End User:** The role of end user is to use the application programs by entering new data and manipulating and accessing existing data.

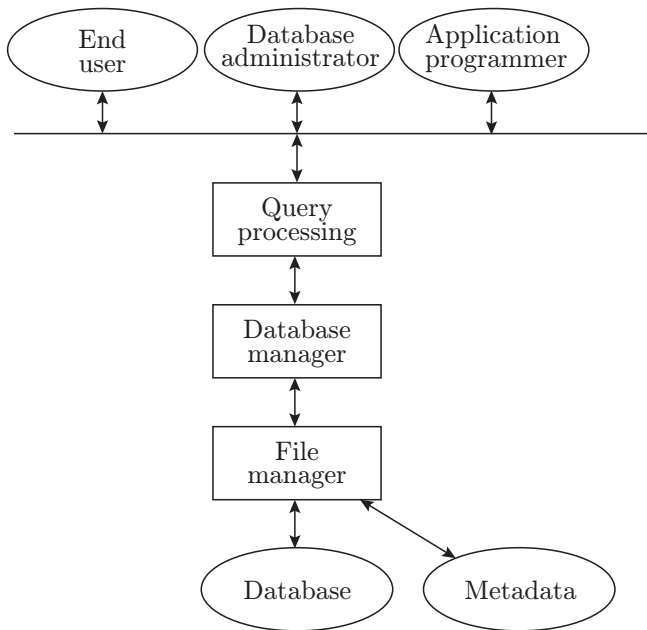


Figure 8.1 | Components of database systems.

8.3 DBMS ARCHITECTURE

It is an approach to outlook the database by users. It is means for the representation of data in an understandable way to the users. DBMS architecture can be used to divide the whole system to related and independent modules. It can be of 1-tier, 2-tier 3-tier or n -tier.

8.3.1 3-Tier Architecture

DBMS can be most widely used as 3-tier architecture. In this architecture, the database is divided into three tiers depending upon the kind of users (see Fig. 8.2).

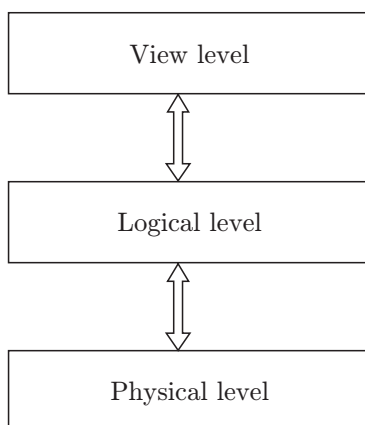


Figure 8.2 | The 3-tier architecture of DBMS.

- 1. Internal schema or physical level:** It is also called Database Tier. Database exists in this tier. It also includes query processing languages, all relations and their corresponding constraints. It describes the physical storage structure of the database.
- 2. Conceptual schema or logical level:** It is also called Application Tier. Server and program exists in this tier. It also describes the structure of the database.
- 3. External schema or view level:** It is also called Presentation Tier. End user exists in this tier. An end user is capable of multiple views of database. It also includes all views generated by applications.

8.3.2 Data Independence

Data independence is defined, as the change at one level does not affect the higher level. It is of two types:

- 1. Logical data independence:** It is defined, as the change in conceptual schema does not affect the external schema. For example, if the format of the table changes, the data lies in the table should not be changed.
- 2. Physical data independence:** It is defined, as the change in internal schema does not affect the conceptual schema. Thus, does not affect the external schema. For example, if the storage system has been changed, then it does not affect the logical structure of the database.

8.4 DATA MODELS

Data model is a collection of tools, which describes data, relationships, constraints and semantics. It gives the logical structure of the database. It describes the relationship of data in the database. Data models are of various types:

- 1. Relational Model:** It is a collection of relations (tables). In relational model, each table is stored as a separate file.
- 2. Entity–Relationship data model:** This model is based on the notion of real-world entities and relationship among them.
- 3. Object-Based data model:** It defines the database as objects, its properties and its operations. In this model, objects with the similar structures comprise a class. The classes are organized into hierarchies. The operations on these classes are performed through methods.
- 4. Semi-Structured data model:** It is also known as XML model. This model is used to exchange data over the web. It uses hierarchical tree structures. In this model, data can be represented as elements by using tags.

5. **Network model:** In this model, data is represented as record types. The data in this model has many-to-many relationship.
6. **Hierarchical model:** In this model, the data is represented as a hierarchical tree structure.

8.4.1 Relational Model

The database in relational model is represented as a collection of relations (tables). A relation is a kind of set. It is also a subset of a Cartesian product of an unordered set of ordered tuples. Relational model was proposed by E. F. Codd, which stores data in a tabular form. It consists of a table where rows represent records and columns represent the attributes. It has various terminologies as follows:

1. **Tuple:** It represents a single row of a table, which contains a single record for that relation.
2. **Relation instance:** It represents a finite set of tuples in the relational database system.
3. **Relation schema:** It represents the relation name, that is, table name, attributes and their names.
4. **Relation key:** It represents the unique key for the relation or table. Each row has one or more attributes, which can identify the row in the table uniquely.
5. **Attribute domain:** It represents the predefined value scope of each attribute.

8.4.1.1 Constraints in Relational Model

Constraints are the restrictions that one wishes to apply on database. The following constraints are applied on relational model.

1. **Key constraints:** Each relation has at least one minimal subset of attributes, which can identify a tuple uniquely.
 - No two tuples have identical value for key attributes.
 - Key attribute does not have NULL value.
2. **Domain constraints:** Attributes have specific domain values in real world. For example, value of age can only be positive.
3. **Referential integrity constraints:** If a relation refers to a key attribute of a different relation then that key element must exist.

8.4.1.2 Relational Algebra

It is a procedural query language. It takes instances of relations as input and returns instances of relations as output. Operators are used to perform queries.

Table 8.1 | Notations of fundamental operations of relational algebra

Fundamental Operation	Symbol
Select	σ
Project	π
Union	\cup
Intersection	\cap
Set difference	$-$
Cartesian product	\times
Rename	ρ
Natural join	\bowtie

Fundamental operations of relational algebra are as follows (see Table 8.1):

1. **Select:** It is used to select rows from a relation. It is denoted by σ .
Syntax of select $\sigma_p(r)$, r is relation and p is propositional logic.
 p uses connectors and operators $\wedge, \vee, =, \neq, <, >, \leq, \geq$.
For example, $\sigma_{\text{empname} = \text{"John"}}(\text{emp})$.
2. **Project:** It is used to project columns in a relation. It is denoted by π . The duplicate tuples are automatically eliminated.
Syntax of projects $\pi_A(r)$, r is relation and A is the attribute name in a relation.
For example, $\pi_{\text{empname}, \text{sal}}(\text{emp})$.
3. **Union:** It returns a relation instance, which contains all tuples occurring in the first relation or in the second relation. It is denoted as $R \cup S$, where R and S are two relations. The duplicate tuples are automatically eliminated.
This operation is valid for the following:
 - Both relations must have the same number of attributes.
 - Attribute domains must be compatible.
4. **Intersection:** It returns a relation instance, which contains all tuples occurring in both relations. It is denoted as $R \cap S$, where R and S are two relations.
5. **Set difference:** It returns a relation instance, which contains all tuples that occur in the first relation but not in the second relation. It is denoted as $R - S$, where R and S are two relations.
6. **Cartesian product:** It returns a relation instance, which contains all the fields of the first relation followed by all the fields of the second relation. It is denoted as $R \times S$, where R and S are two relations.

- 7. Rename:** It returns a relation but without any name. It is used to rename the output relation. It is denoted as ρ .
- 8. Joins:** It returns combined information from two or more relations.
- *Condition joins:* It accepts a join condition c and a pair of relation instances as arguments, and returns a relation instance. It is denoted as $\sigma_c(R \times S)$.
 - *uijoin:* It is a special case of condition joins where the condition c contains equalities.
 - *Natural join:* It is a Cartesian product of two relations. It is denoted by \bowtie .

8.4.1.3 Tuple Calculus

It is a non-procedural query language. In this, number of tuple variables is specified. It is represented as

$\{t \mid \text{Condition}\}$, where t is a tuple variable and Condition is a conditional expression.






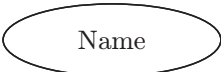

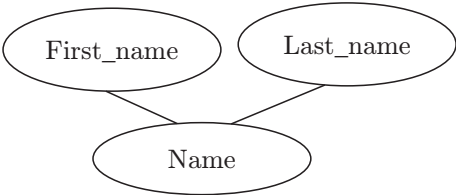

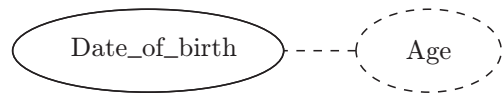
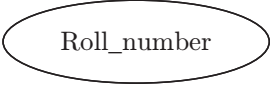

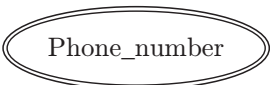
Preliminaries of tuple calculus are as follows:

1. Constants
2. Predicates
3. Boolean and, or, not
4. \exists there exists
5. \forall for all

8.4.2 ER Model

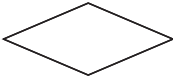
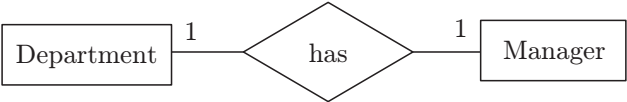

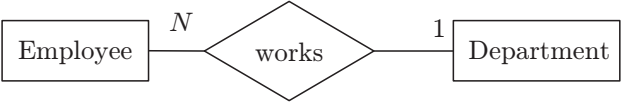

ER model represents the conceptual view of a database. It describes the relation of data to each other (Table 8.2). It was developed by Peter Chen in 1976. It views real-world data as systems of entities and relationships. ER model has three basic elements: entity, attribute and relationship. These are discussed in the following sections.

Table 8.2 | Entity-relationship objects

Element	Symbol	Example
Entity		
Weak entity		
Attribute		
Simple attribute		
Composite attribute		
Derived attribute		
Single-valued attribute		
Multi-value attribute		

(Continued)

Table 8.2 | Continued

Element	Symbol	Example
Relationship		
One-to-One relationship		
One-to-Many relationship		
Many-to-One relationship		
Many-to-Many relationship		

8.4.2.1 Entity

Entities represent the real-world things. These are data objects which maintain different relationships with each other, for example, Employee, Department, etc. These are represented by means of rectangles.



For example,

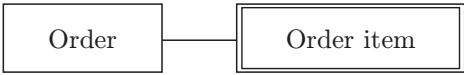


Entity set is a collection of similar types of entities. For example, all Employees set may contain all employees of all departments.

Weak entity depends on the existence of another entity. A weak entity cannot be identified by its own attributes. It uses a foreign key combined with its attributes to form the primary key. It is represented by means of double rectangles.

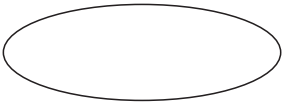


For example, Details of Employee’s spouse, order item, etc.

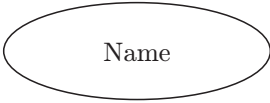


8.4.2.2 Attributes

Attribute is a property, trait or characteristic of an entity, relationship or another attribute. All attributes have values. A domain or range of values can be assigned to attributes. For example, name, class, age are attributes of the entity student. These are represented by ovals.



For example,



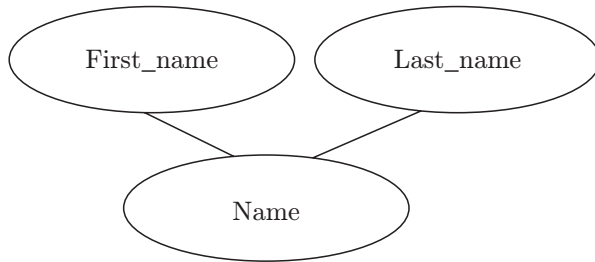
There are different types of attributes, listed as follows:

- 1. **Simple attribute:** Simple attributes contains atomic values. Atomic values cannot be divided

into sub parts. Examples are mobile number, roll number.



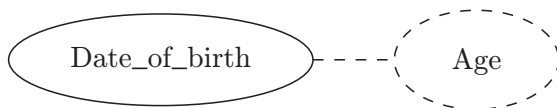
- 2. Composite attribute:** Composite attributes are composition of many simple attributes. For example address can be divided into house number, street number, locality and city.



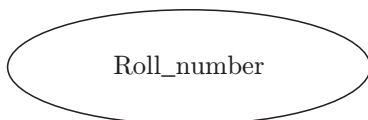
- 3. Derived attribute:** Derived attributes are those whose value is derived from some other attribute in the database. For example, age of person can be calculated from date of birth. Dotted oval is used to represent derived attributes.



For example,



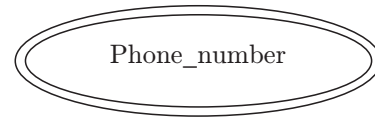
- 4. Single-valued attribute:** Single valued attributes contain only one value for that attribute. For example age for person, blood group.



- 5. Multi-value attribute:** In this, an attribute may contain more than one value. Multiple values are represented by double ovals.

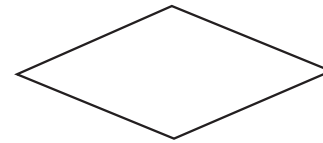


For example contact number, email ids.



8.4.2.3 Relationships

It represents the association among entities in a specified way. For example, employee entity has relation works at with department entity. Relationships are represented by diamond-shaped boxes.

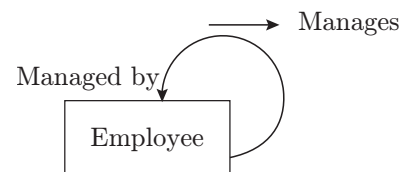


Some basic terminologies related to relationship are given below.

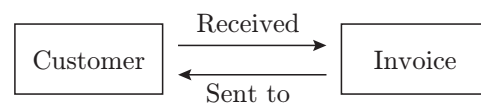
As we have discussed, relations are the core components in RDBMS, these relations are defined by two major characteristics—relationship set and the degree of relationship, defined in the following text.

- 1. Relationship set:** Relationship of similar type is called relationship set. It has attributes. These attributes are called descriptive attributes.
- 2. Degree of relationship:** It defines the number of participating entities in a relationship. They are of the following types:

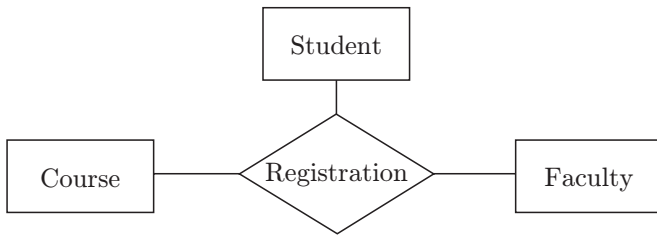
- *Unary relationship (Degree 1):* One entity participates. For example,



- *Binary relationship (Degree 2):* Two entities participate. For example,



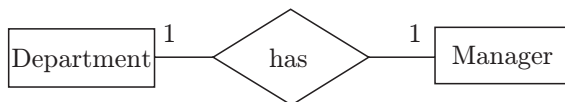
- *Ternary relationship (Degree 3):* Three entities participate. For example,



- *n-ary relationship (Degree n)*: n entities participate.

3. Cardinality: It defines the number of instances of an entity, which can be associated to the number of instances of other entity via relationship set.

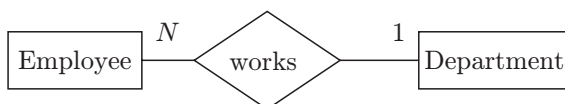
- *One to one*: One instance of an entity is associated with at most one instance of another entity with the relationship. It is represented as '1-1'. For example,



- *One to many*: One instance of an entity is associated with more than one instance of another entity with the relationship. It is represented as '1- N '. For example,



- *Many to one*: More than one instance of an entity is associated with another entity with the relationship. It is represented as ' N -1'. For example,



- *Many to many*: More than one instance of an entity is associated with more than one instance of another entity with the relationship. It is represented as ' N - N '. For example,



4. Generalization: It is a collection of entity sets, having similar characteristics, brought together into one generalized entity. For example, salaried and contract employees are generalized as employee.

5. Specialization: It is the process of identifying subsets of an entity set. It is a reverse process

of generalization. For example, employees are specialized as salaried and contract as shown in Fig 8.3.

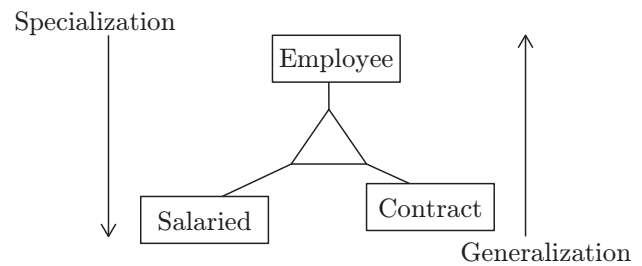


Figure 8.3 | Specialization and Generalization.

6. Aggregation: It allows a relationship set participate in another relationship set (see Fig. 8.4).

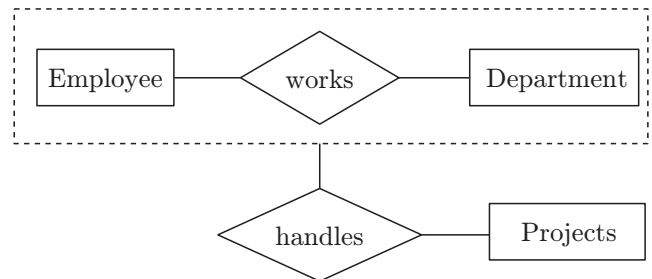


Figure 8.4 | Aggregation.

8.5 DATABASE DESIGN

The design of a database consists of the following steps:

1. Identifying entities
2. Identifying relationships
3. Identifying attributes
4. Presenting entities and relationships: Entity relationship diagram (ERD)
5. Assigning keys
6. Defining the attribute's data type
7. Normalization

8.5.1 Integrity Constraints

Integrity constraints are applied to maintain the consistency in a database. This helps in providing the unique answer to a given query on the database. For example, if the answer to particular query is 'x' then it should be 'x' if such a query is carried out again (without adding/deleting/modifying) on the same table.

- 1. Domain integrity:** It defines a valid set of values for an attribute, for example, length or size, data type, etc.

2. **Entity integrity constraint:** It defines that the primary keys cannot be null. There must be a proper value in the primary key field.
3. **Referential integrity constraint:** It is specified between two tables. It is used to maintain the consistency among rows between the two tables.
4. **Foreign key integrity constraint:** There are two types of foreign key integrity constraints:
 - *Cascade update related fields:* Whenever the primary key of a row in the primary table is changed, the foreign key values are updated in the matching rows in the related table.
 - *Cascade delete related rows:* Whenever a row in the primary table has been deleted, the matching rows are automatically deleted in the related table.

8.5.2 Normal Forms

Normalization is a technique of organizing the database tables, such that they have minimum redundancy. Following are the normal forms that are to be achieved for the process of normalization. The underlying concept is that, if we say a table to be satisfying an 'x' level normal form, then it is understood that it has also satisfied the 'x-1' level normal form.

1. **First normal form:** It defines that all the attributes in a relation must have atomic domains.
2. **Second normal form:** It defines that every non-prime attribute should be fully functionally dependent on prime key attribute. A prime attribute is a part of prime key. The relation must be in the First normal form.
3. **Third normal form:** It defines that no non-prime attribute is transitively dependent on prime key attribute. The relation must be in the Second normal form.
4. **Boyce-Codd normal form:** It defines that for any non-trivial FD, $X \rightarrow A$, then X must be a superkey. It is an extension of the Third normal form.
5. **Fourth normal form:** It defines that for every multivalued dependency $X \twoheadrightarrow Y$ that holds over R , one of the following statements is true: (a) Y is the subset of X and (b) X is a superkey. Also, sometimes called, Multi-valued Dependency Normal Form (MDNF).
6. **Fifth normal form:** It denies that for every join dependency $\bowtie \{R_1, \dots, R_n\}$ that holds over R , one of the following statements is true: (a) $R_i = R$ and (b) the join dependency is implied by the set of those FDs over R in which the left side is a key.

Also, sometimes called, Project-Join Normal Form (PJNF).

8.5.3 Attribute Closure

Set of all attributes functionally determined by X is called closure of X . Closure set of X is denoted by X^+ .

Problem 8.1: Functional dependencies $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$ is given, find closure of A , B , C and D .

Solution:

Closure of $A = A^+ = (A, B, C, D)$

Closure of $B = B^+ = (B, C, D)$

Closure of $C = C^+ = (C, D)$

Closure of $D = D^+ = (D)$

8.5.4 Key

It is a minimum set of attributes used to differentiate all the tuples of the table.

8.5.4.1 Superkey

It is a set of attributes that uniquely identifies each record in a table. It is a superset of candidate key. In other words, let R be the schema and X be the set of attributes over R . If closure of X (X^+) determines all the attributes of R , then X is called superkey.

Problem 8.2: Consider a schema $R(ABCDE)$ and FDs $\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$. Find superkey.

Solution:

Find closure set of (A, B, C, AB) .

Closure of $A = A^+ = \{A\}$

Closure of $B = B^+ = \{B, E\}$

Closure of $C = C^+ = \{C, D\}$

Closure of $AB = AB^+ = \{A, B, C, D, E\}$

So, AB is the superkey.

8.5.4.2 Candidate Key

It is an attribute or set of attribute that can act as a primary key of a table that uniquely identifies each record

in a table. Every candidate key is a superkey but not vice versa.

In other words, candidate key is the minimal superkey. If X is a superkey and none of the proper subset of X is a superkey, then X is called the minimal superkey or candidate key.

Problem 8.3: Consider a schema $R(ABCDE)$ and FDs $\{AB \rightarrow C, C \rightarrow D, B \rightarrow EA\}$. Find the superkey.

Solution:

Find closure set of (A, B, C, AB)

Closure of $A = A^+ = \{A\}$

Closure of $B = B^+ = \{B, E, A, C, D\}$

Closure of $C = C^+ = \{C, D\}$

Closure of $AB = AB^+ = \{A, B, C, D, E\}$

So, AB and B are superkeys. But only B is the candidate key.

8.5.4.3 Primary Key

It is one or more data attributes that uniquely identify an entity. It does not allow null values.

1. **Alternate key:** The candidate key, which is not selected as a primary key.
2. **Composite key:** It consists of two or more attributes.
3. **Foreign key:** It is an entity that is the reference to the primary key of another entity.

8.5.5 Decomposition

It is required to eliminate redundancy from the schema. If a relational schema R has redundancy in the data, then decompose R into two R_1 and R_2 schema. There are two properties, which should be maintained when we perform decomposition, it should be a lossless join as well as dependency preserving.

8.5.5.1 Lossless Join

Let R be a relation schema and let F be a set of functional dependency (FD) over R . R is decomposed into R_1 and R_2 . R_1 and R_2 are called lossless-join decomposition if $R = R_1 \bowtie R_2$ or if we can recover original relation from the decomposed relation.

(a) *Algorithm for finding decomposition is lossless:*

Step 1: Union of all decomposed sub-relation should be equal to relation R .

$$R_1 \cup R_2 \cup R_3 \cup \dots \cup R_n = R$$

Step 2: Any two sub-relations R_i and R_j can be merged into R_{ij} with $R_1 \cup R_2$ only if

i. $R_i \cap R_j \neq \emptyset$ (null)

ii. $R_i \cap R_j = X$ then closure of $X(X^+) \supseteq R_i$
or

$R_i \cap R_j = X$ then closure of $X(X^+) \supseteq R_j$

Step 3: Repeat step 2 until ' N ' relations become one relation. If ' N ' relations become single relation, then composition is called lossless, otherwise not.

Problem 8.4: Consider a schema $R(ABCDEFGH IJ)$ and functional dependencies $\{FDs = (AB \rightarrow C, A \rightarrow D, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ)\}$ and decompositions

- (a) $\{D = (ABCDE, BFGH, DIJ)\}$
- (b) $\{D = (ABCD, DE, BF, FGH, DIJ)\}$

Check whether the decomposition is lossless or not.

Solution:

(a) Given

$$R_1 = (ABCDE)$$

$$R_2 = (BFGH)$$

$$R_3 = (DIJ)$$

Apply algorithm for lossless decomposition:

Step 1:

$$R_1 \cup R_2 \cup R_3 = \{(ABCDE) \cup (BFGH) \cup (DIJ)\} \\ = (ABCDEFGH IJ) = R$$

Step 1 satisfies the given condition, so it is true.

Step 2:

For R_1 and R_2 :

$$R_1 \cap R_2 = (ABCDE) \cap (BFGH) = B$$

Find closure of $B = B^+ = \{B, F, G, H\}$.

Condition (ii) is satisfied, so R_1 and R_2 can be merged together. After merging R_1 and R_2 ,

$$R_{12} = (ABCDEFGH)$$

Now merge R_{12} and R_3 .

$$R_{12} \cap R_3 = (ABCDEFGH) \cap (DIJ) = D$$

Find closure of $D = D^+ = \{D, I, J\}$.

Condition (ii) is satisfied, so R_{12} and R_3 can be merged together. After merging R_{12} and R_3

$$R_{123} = (ABCDEFGH IJ)$$

So, $R_{123} = R$, therefore, it is lossless decomposition.

(b) Given that

$$R_1 = (ABCD)$$

$$R_2 = (DE)$$

$$R_3 = (BF)$$

$$R_4 = (FGH)$$

$$R_5 = (DIJ)$$

Apply algorithm for lossless decomposition.

Step 1:

$$R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5 = \{(ABCD) \cup (DE) \cup (BF) \cup (FGH) \cup (DIJ)\} = (ABCDEFGH IJ) = R$$

Step 1 satisfies the given condition, so it is true.

Step 2:

For R_1 and R_2

$$R_1 \cap R_2 = (ABCD) \cap (DE) = D$$

Find closure of $D = D^+ = \{D, I, J\}$.

Condition (ii) of step 2 does not satisfy, so R_1 and R_2 cannot be merged together.

For R_1 and R_3

$$R_1 \cap R_3 = (ABCD) \cap (BF) = B$$

Find closure of $B = B^+ = \{B, F, G, H\}$.

Condition (ii) of step 2 is satisfied, so R_1 and R_3 can be merged together. After merging R_1 and R_3

$$R_{13} = (ABCDF)$$

For R_{13} and R_4

$$R_{13} \cap R_4 = (ABCDF) \cap (FGH) = F$$

Find closure of $F = F^+ = \{F, G, H\}$

Condition (ii) of step 2 is satisfied, so R_{13} and R_4 can be merged together. After merging R_{13} and R_4

$$R_{134} = (ABCD FGH)$$

For R_{134} and R_5

$$R_{134} \cap R_5 = (ABCD FGH) \cap (DIJ) = D$$

Find closure of $D = D^+ = \{D, I, J\}$

Condition (ii) of step 2 is satisfied, so R_{134} and R_5 can be merged together. After merging R_{134} and R_5

$$R_{1345} = (ABCD FGH IJ)$$

For R_{1345} and R_2

$$R_{1345} \cap R_2 = (ABCD FGH IJ) \cap (DE) = D$$

Find closure of $D = D^+ = \{D, I, J\}$.

Condition (ii) of step 2 does not satisfy, so R_{1345} and R_2 cannot be merged together.

So, finally we left with two decompositions which cannot be merged into a single relation. So condition given in step 3 does not satisfy, therefore, it is not lossless decomposition.

8.5.5.2 Dependency Preserving

All the dependency should be preserved after the decomposition of schema R .

Let R be the relational schema with FD set F decomposed into $R_1, R_2, R_3, \dots, R_n$ with FD sets $F_1, F_2, F_3, \dots, F_n$, respectively. The decomposition is said to be dependency preserved if

$$F_1 \cup F_2 \cup F_3 \cup \dots \cup F_n = F$$

And composition is called non-dependency preserved if

$$F_1 \cup F_2 \cup F_3 \cup \dots \cup F_n \subset F$$

(a) *Algorithm to check dependency is preserved or not:*

Step 1: Find all the FDs of sub-relations.

Step 2: Check that all the FDs of relation F is covered by FDs of sub-relation in any form directly or indirectly.

Problem 8.5: Consider a schema $R(ABCD)$ and FDs set $F = (A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A)$ and decompositions $D = (AB, BC, CD)$. Check decomposition is dependency preserving or not.

Solution:

Step 1: Find all the FDs of sub-relations.

	$R_1 = (AB)$	$R_2 = (BC)$	$R_3 = (CD)$
Direct Dependency	$A \rightarrow B$	$B \rightarrow C$	$C \rightarrow D$
Indirect Dependency	$B \rightarrow A$	$C \rightarrow B$	$D \rightarrow C$

To calculate indirect dependency, use closure property. In relation R_1 , we have indirect dependency $B \rightarrow A$. We have to first find closure of all attributes in that decomposition (like in R_1 we have A and B) by using FD set F and then check that we are getting attribute A in closure of B . So closure of $B = B^+ = ABCD$. So we can write that indirectly as we have three dependencies $\{B \rightarrow A, B \rightarrow C \text{ and } B \rightarrow D\}$ but only $B \rightarrow A$ is valid for R_1 as we have only two attributes (A and B) in relation R_1 .

Step 2: Check that all the FDs of relation F is covered by FDs of sub-relation in any form directly or indirectly.

- $A \rightarrow B$ (Directly covered by sub-relation FD)
- $B \rightarrow C$ (Directly covered by sub-relation FD)
- $C \rightarrow D$ (Directly covered by sub-relation FD)
- $D \rightarrow A$ (Indirectly covered by sub-relation FD by using $\{D \rightarrow C \text{ then } C \rightarrow B \text{ and then } B \rightarrow A\}$).

All the dependency is preserved, so it is dependency-preserved decomposition.

8.5.5.3 Relation between Two Functional Dependency (FD) Sets

Let F and G be two FD sets:

- Set F and G are equal if and only if closure of $F(F^+) = \text{Closure of } G(G^+)$
- Set F and G are equal only if both of the below conditions satisfy:
 - F covers G : All FD in G is logically implied by F .
 - G covers F : All FD in F is logically implied by G .
- If all FD in G is logically implied by F but all FD in F is not logically implied by G then $G \subset F$.

Problem 8.6: Let there be two FD sets F and G , given as follows:

$$F = \{A \rightarrow B, B \rightarrow C\}$$

$$G = \{A \rightarrow B, AB \rightarrow C, B \rightarrow C, A \rightarrow C\}$$

Find the relation between both FDs.

Solution:

- F covers G :

Functional Dependency	Covered by F
$A \rightarrow B$	Direct covered
$AB \rightarrow C$	Indirect covered ($AB^+ = ABC$), so AB can determine C .

Functional Dependency	Covered by F
$B \rightarrow C$	Direct covered
$A \rightarrow C$	Indirect covered ($A^+ = ABC$), so A can determine B and C .

(b) G covers F

Functional Dependency	Covered by G
$A \rightarrow B$	Direct covered
$B \rightarrow C$	Direct covered

In our example, F covers G and vice versa. So $F = G$.

8.6 QUERY LANGUAGES (SQL)

SQL stands for Structured Query Languages, which is a standard computer language for relational database management and data manipulation. It is used to query, insert, update and modify data in the table. Some common RDBMS that use SQL are Oracle, Microsoft SQL Server, Access, Ingres, Sybase, etc. Raymond Boyce and Donald Chamberlin developed it in the early 1970s at IBM, but commercially released by Relational Software Inc. (now, Oracle Corporation) in 1979. Looking into the history it was the software named- VULCAN, that was procured by Ashton Tate and then by FoxPro and later was purchased by Microsoft. Other popular softwares were/are – Clipper and Gupta Technologies.

8.6.1 SQL Commands

The SQL commands are used to interact with relational databases. These commands can be classified into the following groups (see Tables 8.3 and 8.4):

- Data Definition Language (DDL):** It contains metadata, that is, data about data. All the integrity constraints and data base schemas are defined through DDL. These commands are used to create, modify and delete database objects. CREATE, ALTER, TRUNCATE and DROP are part of DDL. TRUNCATE command is used to delete complete data from an existing table, while DROP command is used to remove a table definition and all data, indexes, triggers and constraints for a table.

- Syntax of create command

```
CREATE TABLE table_name( column1
datatype, column2 datatype, column3
```

```
datatype, ..... columnN datatype,
PRIMARY KEY( one or more columns ) );
For example, create table instructor
(INST_ID char(5), name varchar(20),
dept_name varchar(20), salary
numeric(8,2));
```

- Syntax of alter command

```
ALTER TABLE table_name ADD column_
name datatype;
ALTER TABLE table_name DROP COLUMN
column_name;
ALTER TABLE table_name MODIFY COLUMN
column_name datatype;
ALTER TABLE table_name ADD CONSTRAINT
Constraint UNI (column1, column2...);
ALTER TABLE table_name ADD CONSTRAINT
Constraint CHECK (CONDITION);
ALTER TABLE table_name ADD CONSTRAINT
PrimaryKey PRIMARY KEY (column1,
column2...);
For example, ALTER TABLE CUSTOMERS ADD
GRADE char(1);
```

- Syntax of truncate command

```
TRUNCATE TABLE table_name;
For example, TRUNCATE TABLE emp;
```

- Syntax of drop command

```
DROP TABLE table_name;
For example, DROP TABLE emp;
```

Table 8.3 | SQL commands

Command	Description
CREATE	Creates a new table, a view of a table or other objects in the database
ALTER	Modifies an existing database (table)
DROP	Deletes a table, a view of a table or other objects in the database
SELECT	Retrieves data from one or more tables (database)
INSERT	Inserts new data record in the database
UPDATE	Modifies data in the database
DELETE	Deletes data from the database
GRANT	Gives a privilege to the user
REVOKE	Takes back privileges granted from the user

Table 8.4 | Clauses in SQL

Clause	Description
From	Equals to cross product
Where	Selects the tuples which satisfies the condition
Group by	Groups the table based on specified attribute
Having	Used to select groups based on condition

2. Data Manipulation Language (DML):

DML is used to manipulate the data in database. It allows to insert, update and delete data items. SELECT, INSERT, DELETE and UPDATE are part of DML. Control statements BEGIN TRANSACTION, SAVEPOINT, COMMIT and ROLLBACK are also part of DML.

- Syntax of select command

```
SELECT * FROM table_name;
SELECT column1, column2, columnN
FROM table_name;
```

For example, SELECT empno, ename FROM emp;

- Syntax of insert command

```
INSERT INTO TABLE_NAME (column1,
column2, column3,...columnN) VALUES
(value1, value2, value3,...valueN);
INSERT INTO TABLE_NAME VALUES
(value1,value2,value3,...valueN);
```

For example, INSERT into emp (empno, ename, sal, dept) VALUES (100, ABC, 10000, Accounts);

- Syntax of delete command

```
DELETE FROM table_name WHERE [condi-
tion];
```

For example, DELETE FROM emp WHERE empno = 8;

- Syntax of update command

```
UPDATE table_name SET column1 =
value1, column2 = value2...,
columnN = valueN WHERE [condition];
```

For example, UPDATE mp SET sal = 12000 WHERE empid = 8;

3. Data Control Language (DCL): To assign or revoke access rights data control language is used. GRANT and REVOKE are used for DCL.

- Syntax of grant command

```
GRANT privilege_name ON object_name
TO {user_name | PUBLIC | role_name}
[with GRANT option];
```

For example, GRANT SELECT ON emp TO user1;

- Syntax of revoke command

```
REVOKE privilege_name ON object_name
FROM {User_name | PUBLIC | Role_name};
```

For example, REVOKE SELECT ON emp TO user1;

8.7 FILE STRUCTURES

File structure mainly deals with how files are stored on the disk. Various file organisations are described below.

8.7.1 Sequential Files

It is a file organisation system where every file record contains an attribute to uniquely identify a particular record. Records are placed in a sequential order with some unique key.

8.7.2 Indexing

Indexing is a data structure mechanism to efficiently retrieve records from the database, for example, book index. It is defined based on its indexing attributes. Indexing is of three types:

1. **Primary index:** Indexing is based on ordering key field of file.
2. **Secondary index:** Indexing is based on non-ordering field of file.
3. **Clustering index:** Indexing is based on ordering non-key field of file.

Ordering field is the field on which the records of file are ordered. Ordered indexing is of two types: dense index and sparse index.

1. **Dense index:** For every search key value in the database, there is an index record. Index record has two parts: search key value and the pointer. The pointer is pointed to the actual record.
2. **Sparse index:** In this no index record is created for every search key.

8.7.3 B Tree

A B tree is a data structure that stores data in such a manner that search, insertions and deletions can be done

in logarithmic time. B trees are a general form of binary trees where a node can have more than one child. The B-trees are efficient for those systems that read and write large blocks of data, that is, databases and file systems.

B trees are self-balancing trees. All the leaf nodes are at the same level. A B tree with order p has:

1. Root node may contain minimum 1 key
2. Minimum number of child = $\left(\frac{p}{2}\right) - 1$
3. Maximum number of children = p
4. Maximum keys = $p - 1$

The order of B-tree can be found as follows:

$$p \times P + (p - 1)(K + P_r) \leq \text{Block size}$$

where p is order of the tree, P is the block pointer, P_r is the record pointer and K is the key pointer.

Problem 8.7: The order of B-tree index is the maximum number of children it can have. Suppose that a block pointer takes 6 bytes, the search field value takes 9 bytes, record pointer takes 7 bytes and the block size is 512 bytes. What is the order of the B tree?

Solution:

Given that block size = 512 B; record pointer (P_r) = 7 B; block pointer (P_B) = 6 B; key pointer (K) = 9 B. So, we have

$$\begin{aligned} p \times P + (p - 1)(K + P_r) &\leq \text{Block size} \\ 6p + (p - 1)(9 + 7) &= 512 \\ p &= 24 \end{aligned}$$

8.7.4 B+ Trees

It supports multilevel indexing. The leaf nodes of B+ tree represent actual data pointers. It ensures all leaf nodes are balanced, that is, at the same height. Leaf nodes are linked using link list.

B+ tree structure is such that B+ tree is of order n and it is fixed for every B+ tree.

The internal nodes contain at least $\lceil n/2 \rceil$ pointers, except the root node and at most n pointers. Leaf nodes contain at least $\lceil n/2 \rceil$ record pointers, at least $\lceil n/2 \rceil$ key values, at most n record pointers, at most n key values, and every leaf node contains one block pointer P to point to next leaf node and forms a linked list.

Order of internal nodes can be calculated as:

$$p \times P_B + (p - 1) \times k \leq \text{Block size}$$

Order of leaf nodes:

$$P_{\text{leaf}} \times [k + P_r] + P_B \leq \text{Block size}$$

where p is the order of internal nodes, P_r is record pointer, P_B is block pointer, k is key pointer, P_{leaf} is the order of leaf nodes.

Problem 8.8: The order of an internal node in a B+ tree index is the maximum number of children it can have. Suppose that a block pointer takes 6 bytes, the search field value takes 9 bytes, record pointer take 7 bytes and the block size is 512 bytes. What is the order of the internal node and leaf node?

Solution:

Given that block size = 512 B, record pointer (P_r) = 7 B, block pointer (P_B) = 6 B and key pointer (k) = 9 B.

Order of internal nodes:

$$\begin{aligned} p \times P_B + (p - 1)k &\leq \text{Block size} \\ 6p + (p - 1)9 &= 512 \\ p &= 34 \end{aligned}$$

Order of leaf nodes:

$$\begin{aligned} P_{\text{leaf}} \times [k + P_r] + P_B &\leq \text{Block size} \\ P_{\text{leaf}} [9 + 7] + 6 &= 512 \\ P_{\text{leaf}} &= 31 \end{aligned}$$

8.8 TRANSACTIONS AND CONCURRENCY CONTROL

Transactions are series of read and write operations on database. When many users access same database at the same time, some control mechanism is required by databases to stay in consistent state. Following sections will provide description about transactions and their control mechanism.

8.8.1 Transactions

It is a series of reads and writes of database objects. It maintains the integrity of a database while running multiple concurrent operations. Transactions have four properties that a DBMS must ensure to maintain data. These are known as ACID properties:

1. **Atomicity:** Either all actions are carried out or none (no partial transaction).
2. **Consistency:** After the execution of transaction, the database must be in a consistent state (no concurrent execution of other transactions).
3. **Isolation:** All the transactions are carried out and executed as the only transaction in the system

(no transaction will affect the existence of any other transaction).

4. **Durability:** Persistence of data even if the system fails and restarts.

8.8.2 Schedule

A chronological execution sequence of transactions is called schedule. It is a list of actions reading, writing, aborting or committing from a set of transactions. A schedule can have many transactions. Schedule can be further divided into two types—serial schedule and concurrent schedule.

8.8.2.1 Serial Schedule

A schedule is called serial schedule if all transactions are executed in a non-interleaving manner. Let there are two transactions T_1 and T_2 schedule S then schedule S is called serial schedule if one transaction executes after another shown in Fig. 8.5.

T_1	T_2	T_1	T_2
Read (X)			Read (X)
$X = X - Z$			$X = X + W$
Write (X)			Write (X)
Read (Y)		Read (X)	
$Y = Y + Z$		$X = X - Z$	
Write (Y)		Write (X)	
	Read (X)	Read (Y)	
	$X = X + W$	$Y = Y + Z$	
	Write (X)	Write (Y)	

Figure 8.5 | Serial schedule.

8.8.2.2 Concurrent Schedule

A schedule is called concurrent schedule if transactions are executed in an interleaving manner or simultaneous execution of two or more transactions. Let there be two transactions T_1 and T_2 in a schedule S , then schedule S is called concurrent schedule if both the transactions execute parallel. One of the scenarios is shown in Fig. 8.6.

T_1	T_2
Read (X)	
$X = X - Z$	
	Read (X)
	$X = X + W$
Write (X)	
	Write (X)
Read (Y)	
$Y = Y + Z$	
Write (Y)	

Figure 8.6 | Concurrent schedule.

8.8.2.3 Comparison between Serial and Concurrent Schedule

Table 8.5 shows the comparison between serial and concurrent schedule.

Table 8.5 | Serial schedule vs. concurrent schedule

Serial Schedule	Concurrent Schedule
All serial schedules are consistent schedules	All serial schedules are not consistent schedules
Less throughput	More throughput
Poor resource utilization	Good resource utilization
More response time	Less response time
For the given transactions, the number of serial schedule is very much less than the number of concurrent schedule	For the given transactions, the number of concurrent schedule is more than the number of serial schedule

8.8.2.4 Problems Occurring due to Concurrent Schedule

In concurrent schedule, more than one transaction is executed simultaneously; and due to this some problem arises with concurrent schedule given as follows:

- 1. WR problem (Read after Write):** WR problem is also known as dirty read problem or uncommitted read problem. Let there be two transactions T_i and T_j of schedule S. If transaction T_j reads a data item which is written by T_i , but till that time transaction T_i is not committed, then WR problem can occur. If in transaction T_i rollback or failure occurs, then database will be inconsistent due to uncommitted read by T_j transaction (Fig. 8.7).

Transaction T_i	Transaction T_j
Read (A)	
$A = A + X$	
Write (A)	
	Read (A)
	Write (B)
	$A = A + Y$
Read (B)	
.....	
.....	
Failure or rollback	
	Write (A)

Figure 8.7 | Uncommitted read problem (WR problem).

Example: Let value of data item $A = 1000$, $X = 100$ and $Y = 200$, then transaction T_i will update the database with value 1100. Transaction T_j will read data item value as 1100. Transaction T_j will update database with value 1300. As transaction T_i fails, the database value should be 1200. Therefore, this problem is known as uncommitted read problem or dirty read problem.

- 2. RW problem (Write after Read):** RW problem is also known incorrect summary problem or unrepeatable read problem (Fig. 8.8). Let there be two transactions T_i and T_j of schedule S. Transaction T_i read a data item and T_j also read similar data item. Transactions T_i and T_j both have write operation on that data item. If in both transactions, read operation occurs before commit of the other transaction and before writing back that data item in database by other transaction then RW problem will arise.

Transaction T_i	Transaction T_j
Read (A)	
If ($A > 0$)	
$A = A + 1$	
	Read (A)
	If ($A > 0$)
	$A = A + 1$
Write (A)	
Commit	Write (A)
	Commit

Figure 8.8 | Incorrect summary problem (RW problem)

Example: Let value of data item $A = 100$. Transaction T_i will read data item with value 100 and update database with value 101. Transaction T_j will read data item value as 100 because it will read before updating value of data item A by transaction T_i . Transaction T_j will update database with value 101. But data item 'A' value should be 102 because both the transaction have commit successfully. Therefore, this problem is known as unrepeatable read problem.

- 3. WW Problem (Write after Write):** WW problem in concurrent schedule is also known as lost update problem (Fig. 8.9). Let there are two transactions T_i and T_j of schedule S. Transaction T_i reads a data item and updates it. Now, transaction T_j also writes similar data item with some other value and transaction T_j commits successfully. After commit of transaction T_j , transaction T_i will rollback. Thus, updated value of data item by transaction T_j will be lost.

Transaction T_i	Transaction T_j
Read (A)	
Write (A)	
	Write (A)
	Commit
Failure and Rollback	

Figure 8.9 | Lost update problem (WW problem).

Example: Let the value of data item $A=100$. Transaction T_i read data item and update database with value 200. Let transaction T_j updated database with value 250 and commit successfully. After committing transaction T_j , transaction T_i rollback which set data item value to its initial value 100. So, this problem is known as lost update problem.

8.8.3 Classification of Schedule Based on Recoverability

8.8.3.1 Irrecoverable Schedule

Let there be two transactions T_i and T_j of schedule S . If transaction T_j reads a data item which is updated by transaction T_i and transaction T_j commits before the commit (or rollback) of transaction T_i , then the given schedule S is called irrecoverable schedule.

8.8.3.2 Recoverable Schedule

Let a schedule S have two transactions T_i and T_j . If transaction T_j reads a data item which is updated by transaction T_i and transaction T_j is not allowed to commit (or rollback) before the commit (or rollback) of transaction T_i , then the given schedule S is called recoverable schedule. Recoverable schedule may suffer from uncommitted read, lost update and incorrect summary problem.

8.8.3.3 Cascading Rollback Recoverable Schedule

Let there be four transactions (T_1, T_2, T_3, T_4) in a given schedule S . In schedule S , if rollback of a transaction (T_1) results in the rollback of the other transactions (T_2, T_3, T_4) (because of their dependency on each other), then this is called cascading rollback.

If a schedule is recoverable and has no cascading rollback, then it is called cascading rollback recoverable schedule. Incorrect summary and lost update problems

may exist in cascading rollback recoverable schedule. Problems of no recoverability, Uncommitted Read problem (WR problem) and cascading rollback problem do not occur in such schedule.

8.8.3.4 Strict Recoverable Schedule

Let a schedule S have two transactions T_i and T_j . If S is called a strict recoverable schedule then it satisfies these two conditions:

1. Schedule S should be cascading rollback recoverable schedule.
2. If one transaction T_i writes a data item 'A', then the other transaction T_j is not allowed to write on that data item.

Only incorrect summary problem (RW problem) may arise in strict recoverable schedule. Irrecoverable, Uncommitted Read problem (WR problem), lost update and cascading rollback problems do not occur in such schedule.

8.8.3.5 Summary of Schedules Based on Recoverability

Schedule	Problem Exists	Problem Remove
Irrecoverable	All	None
Recoverable	Incorrect summary problem (RW), Uncommitted Read problem (WR problem), lost update (WW) and cascading rollback problem	Irrecoverable
Cascading rollback recoverable	Incorrect summary problem (RW) and lost update (WW).	Irrecoverable, Uncommitted Read problem (WR problem) and cascading rollback problem.
Strict recoverable	Incorrect summary problem (RW)	Irrecoverable, Uncommitted Read problem (WR problem), lost update (WW) and cascading rollback problem.

Problem 8.9: A concurrent schedule S has three transactions T_1, T_2, T_3 . Transactions execute read/write operation in the following sequence:

Read₁(x), Read₂(z), Read₃(x), Read₁(z), Read₂(y),
Read₃(y), Write₁(x), Commit₁, Write₂(z), Write₃(y),
Write₂(y), Commit₃, Commit₂.

Find out the type of recoverable schedule?

Solution:

Transaction T_1	Transaction T_2	Transaction T_3
Read ₁ (x)	----	----
----	Read ₂ (z)	----
----	----	Read ₃ (x)
Read ₁ (z)	----	----
----	Read ₂ (y)	----
----	----	Read ₃ (y)
Write ₁ (x)	----	----
Commit ₁	----	----
----	Write ₂ (z)	----
----	----	Write ₃ (y)
----	Write ₂ (y)	----
----	----	Commit ₃
----	Commit ₂	----

Step 1: Check for recoverable. By using the definition of recoverable schedule we can find that given schedule is recoverable or not. (In this problem, no transaction has read operation on a data item after that data item is written by another transaction, so the given schedule is recoverable.)

Step 2: Check for cascading recoverable. In cascading recoverable schedule, no uncommitted read is allowed. In the given problem, there is no uncommitted read so the given schedule is cascading schedule.

Step 3: Check for strict recoverable. There should be commit operation between write operations on similar data items by two different transactions. In the given problem, transaction T_2 and transaction T_3 have write operations on data item (y), but no commit operation is performed by transaction T_3 before data item is updated by transaction T_2 . Hence, the given schedule is not strict recoverable.

8.8.4 Classification of Schedule Based on Serializability

A concurrent transaction schedule is said to be a serializable schedule if its outcome (resulting database state) is equal to the outcome of the serial execution of transactions in the given concurrent schedule. Serializable schedule can be divided into two types: conflict serializability and view serializability.

8.8.4.1 Conflict Serializability

A concurrent schedule S is said to be conflict serializable schedule if S is conflict equivalent to a serial schedule.

- 1. Conflict equivalent schedule:** Let there be two schedules S_1 and S_2 . If after swapping consecutive non-conflict pairs of operation of transactions in schedule S_1 results in schedule S_2 , then S_1 and S_2 schedules are called conflict equivalent schedules.
- 2. Conflict and non-conflict pairs:** A pair of operations in a schedule is called conflict schedule if they have all three conditions:
 - At least one of the operations should be write operation.
 - Both the operations should be performed on the same data item.
 - Both operations should be performed by different transactions.

If a pair of operations do not fulfill the above three conditions, then the pair is called a non-conflict pair.

Problem 8.10: A schedule S_1 with two transactions T_1 and T_2 is given as:

Read₁(x), Write₁(x), Read₂(x), Read₁(y), Write₁(y),
Read₂(y),....

Find conflict equivalent schedule to S_1 .

Solution:

Schedule S_1 :

Transaction T_1	Transaction T_2
Read ₁ (x)
Write ₁ (x)
.....	Read ₂ (x)
Read ₁ (y)
Write ₁ (y)
	Read ₂ (y)

Conflict and non-conflict pairs in S_1 .

After swapping non-conflict pair in schedule S_1 we get another schedule called S_2 . The schedules S_1 and S_2 are conflict equivalent schedules.

Schedule S_2 :

Transaction T_1	Transaction T_2
Read ₁ (x)	-----
Write ₁ (x)	-----
Read ₁ (y)	-----
-----	Read ₂ (x)
Write ₁ (y)	-----
-----	Read ₂ (y)

Conflict equivalent schedules.

3. Testing method of conflict serializable schedule:

To check that a given schedule is conflict serializable or not, we use precedence graph. If precedence drawn from a schedule does not contain a cycle then only it is a conflict serializable schedule. If precedence graph contains a cycle then it is not a conflict serial schedule. A precedence graph G contains:

- Vertex set V : denotes the transactions of the schedule.
- Edge between two vertex from V_i to V_j will be draw where $i \neq j$ and V_i 's operation occur before V_j 's operation, if V_i and V_j have following cases:

- V_i : Read(x) operation and V_j : Write(x) operation
- V_i : Write(x) operation and V_j : Read(x) operation
- V_i : Write(x) operation and V_j : Write(x) operation

Problem 8.11: Consider the given schedule S and draw precedence graph for S .

Transaction T_1	Transaction T_2
-----	Read ₁ (x)
Write ₁ (x)	-----
-----	Read ₂ (x)
Write ₁ (y)	-----
-----	Read ₂ (y)

Solution:

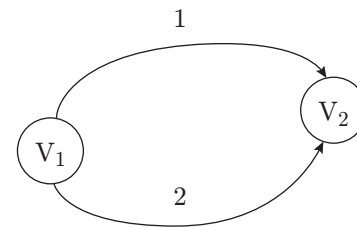
Vertices V_1 and V_2 represent transactions T_1 and T_2 .

Edge 1 for

V_2 : Read(x) operation and V_1 : Write(x) operation

Edge 2 for

V_1 : Write(y) operation and V_2 : Read(y) operation



Note: If two schedules S_1 and S_2 are given then they will be conflict equivalent if they satisfy following conditions:

- Precedence graph of S_1 and S_2 should be equal.
- Each transaction operation should be equal in S_1 and S_2 . (The two transaction will be equal if they have same number of operations and type of operation are also same.)

Problem 8.12: Consider the given schedule S and identify that schedule S is conflict serial schedule or not by the use of precedence graph.

Transaction T_1	Transaction T_2	Transaction T_3
Read (x)	-----	-----
Write (x)	-----	-----
-----	Read (x)	-----
-----	-----	Write (x)
-----	Write (y)	-----
Write (y)	-----	-----
-----	-----	Read (y)

Solution:

Vertices V_1 , V_2 and V_3 represent transactions T_1 , T_2 and T_3 .

Edge 1 for

V_1 : Write(x) operation and V_2 : Read(x) operation

Edge 2 for

V_2 : Read(x) operation and V_3 : Write(x) operation

Edge 3 for

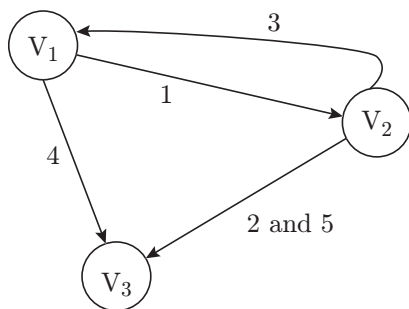
V_2 : Write(y) operation and V_1 : Write(y) operation

Edge 4 for

V_1 : Write(y) operation and V_3 : Read(y) operation

Edge 5 for

V_2 : Write(y) operation and V_3 : Read(y) operation



Precedence graph drawn from the given schedule contains cycle with vertices V_1 and V_2 . So, the given schedule is not a conflict serializable schedule.

8.8.4.2 View Serializability

If a given concurrent schedule is view equivalent to one of the possible serial schedule, then it is called a view serializable schedule. Let there be two schedules S_1 and S_2 with the similar set of transactions. Schedules S_1 and S_2 will be view equivalent if they satisfy following three conditions:

1. If transaction T_i reads the initial value of data item (x) in schedule S_1 , then transaction T_i must read the initial value of (x) in schedule S_2 .
2. In schedule S_1 , if transaction T_i reads the value of data item (x) produced by transaction T_j then T_i must reads the value of (x) that produced by transaction T_j in the schedule S_2 .
3. If transaction T_i write the final value of data item (x) in schedule S_1 , then transaction T_i must write the final value of (x) in schedule S_2 .

Problem 8.13: Identify whether the given concurrent schedule is a view serializable schedule or not?

T_1	T_2	T_3
-----	-----	Read (x)
-----	Read (x)	-----
-----	-----	Write (x)
Read (x)	-----	-----
Write (x)	-----	-----

Solution: We have a concurrent schedule S_1 with three transactions, so the number of possible serializable schedules with three transactions is $8(2^3 = 8)$. So, we will pick one serializable schedule S_2 and if the given concurrent schedule is be view equivalent to that schedule, only then would the given schedule will be view serializable.

T_1	T_2	T_3
-----	-----	Read (x)
-----	Read (x)	-----
-----	-----	Write (x)
Read (x)	-----	-----
Write (x)	-----	-----

Schedule S_1

T_2	T_3	T_1
Read (x)		
	Read (x)	
	Write (x)	
		Read (x)
		Write (x)

Schedule S_2

Step 1: Initial Reads. Transactions T_3 and T_2 read initial value of data item (x) in schedule S_1 (will not consider T_1 because it reads updated value of x by T_3). T_3 and T_2 also read the initial value of (x) in serial schedule S_2 . So, the first condition is true.

Step 2: W-R Conflicts. In the given schedule S_1 , there is only one W-R conflict (between T_3 and T_1). Transaction T_1 reads the value of x produced by transaction T_3 in schedule. In schedule S_2 , transaction T_1 also reads the value of x produced by transaction T_3 in schedule. So, the second condition is also true.

Step 3: Final Write. Transaction T_1 writes the final value of data item (x) in schedules S_1 and S_2 both. So, the third condition is also satisfied.

All the three conditions are satisfied, so the given concurrent schedule S_1 is view equivalent to one of the possible serial schedule S_2 . Therefore, the given concurrent schedule is a view serializable schedule.

8.8.5 Concurrency Control Protocol

Concurrency control protocols are used to ensure serializability of transactions in a concurrent transaction execution environment. They can be lock based protocols and Time stamp ordering protocols.

8.8.5.1 Lock Based Protocol

Lock based protocols uses a mechanism through which a transaction has to obtain a read lock (for read operation) and write lock (for write operation) on a data item (x) without obtaining a lock transaction not allowed to perform any operation. There are two types of lock mechanisms supported by lock based protocol.

1. **Shared lock:** If a transaction T has shared lock on a data item (x), then it can only perform Read operation on that data item.
2. **Exclusive lock:** If a transaction T has exclusive lock on a data item (x), then it can perform Read or Write any operation on that data item.

Lock Compatibility Matrix is given in Table 8.6, which shows that if a transaction has shared/exclusive lock then for which lock it can make a request.

Table 8.6 | Lock compatibility matrix

Lock Request \ Lock Hold	Shared Lock	Exclusive Lock
Shared Lock	Yes	No
Exclusive Lock	No	No

1. **Two Phase Locking Protocol (2PL):** Two phase locking protocol is used by the transaction to lock a data item before reading and writing to maintained consistency in the database. The protocol uses two phases to apply the locking scheme as:

- *Expanding phase:* Transaction acquired locks in this phase and no locks are released.
- *Shrinking phase:* Transaction releases locks in this phase and no locks are acquired.

Note: Two phase locking protocol ensure conflict serializability, but may not be free from irrecoverable, deadlock and starvation.

2. **Two Phase Locking Protocol with Lock Up-gradation:** This technique allows transaction holding a shared lock on a data item can upgrade shared lock into the exclusive lock on the data item without performing unlock for shared lock.
3. **Strict Two Phase Locking Protocol:** In this protocol, if a transaction has exclusive lock on a data item, then it cannot release the lock until commit or rollback is performed by that transaction.
4. **Rigorous Two Phase Locking Protocol:** In this protocol, if a transaction has any lock (exclusive or shared) on a data item, then it cannot release the lock until commit or rollback performed by that transaction.

8.8.5.2 Time Stamp Ordering Protocol

The time stamp ordering protocol uses time stamp values (unique) of the transactions assigned by database to ensure serializability among transactions. A time stamp is a unique value assigned by the database in ascending order. Suppose we have three transactions T_1 , T_2 and T_3 arriving in database, then assigned time stamp values will be as $T_1 < T_2 < T_3$. A data item has two types of time stamp.

1. **Read Time Stamp (x):** It is the highest transaction time stamp that has performed Read operation successfully on a data item (x). It is also denoted by RTS(x). Initial value of RTS(x) is zero.
2. **Write Time Stamp (x):** It is the highest transaction time stamp that has performed Write operation successfully on a data item (x). It is also denoted by WTS(x). Initial value of WTS(x) is zero.

3. Basic Time Ordering Protocol

- Transaction T_i issues Read(x) operation
 - (a) If (WTS(x) > Time Stamp(T_i))
Rollback T_i
 - (b) Otherwise allowed execution of Read(x) by transaction T_i
Set RTS(x) = Maximum (RTS(x), Time Stamp (T_i))

- Transaction T issues Write(x) operation
 - (a) If $(RTS(x) > \text{Time Stamp}(T_i))$

Rollback T_i

- (b) If $(WTS(x) > \text{Time Stamp}(T_i))$

Rollback T_i

Note: Basic Time Ordering protocol ensures serializability, equivalent serial schedule based on time stamp Ordering. It is deadlock free protocol, but basic time ordering protocol can have starvation and irrecoverable schedule (because it does not ensure order of commit).

- 4. Strict Time Ordering Protocol:** In this protocol, an additional condition is added in the Basic Time Ordering protocol. This condition is as follows:

Let a schedule have two transactions T_i and T_j where time stamp of (T_i) is less than time stamp of T_j . If transaction T_j issues Read(x)/ Write(x) operation with $WTS(x) < \text{Time Stamp}(T_j)$ then (T_j) has to be delayed Read(x)/ Write(x) operation until commit or rollback of transaction T_i that has performed Write(x).

Strict Time Ordering Protocol ensures serializability and deadlock free. But it may suffer from starvation.

- 5. Deadlock Prevention:** To prevent deadlock situation in a database system it is very important

to inspect all the operation performed by transactions in a schedule. There are various deadlock prevention scheme which uses time stamp ordering mechanism.

- 6. Wait-Die Protocol:** Assume $\text{TimeStamp}(T_1) < \dots < \text{TimeStamp}(T_n)$ and T_i and T_j are any two transactions and transaction T_i is older than transaction T_j . This implies

$\text{TimeStamp}(T_i) < \dots < \text{TimeStamp}(T_j)$.

- If transaction T_i required a lock which is hold by transaction T_j , then transaction T_i is allowed to wait.
- If transaction T_j required a lock which is hold by transaction T_i then transaction T_j will rollback.

- 7. Wound-Wait Protocol:** Let $\text{TimeStamp}(T_1) < \dots < \text{TimeStamp}(T_n)$ and T_i and T_j are any two transactions and transaction T_i is older than transaction T_j . This implies

$\text{TimeStamp}(T_i) < \text{TimeStamp}(T_j)$

- If transaction T_i required a lock which is hold by transaction T_j then rollback transaction T_j .
- If transaction T_j required a lock which is hold by transaction T_i then transaction T_j is allowed to wait.

IMPORTANT FORMULAS

1. All the 3NF and BCNF decomposition guarantee lossless join.
2. All 3NF and BCNF decomposition may not guarantee dependency preservation.
3. Any relation with two attributes is in BCNF.
4. Functional dependency $F: X \rightarrow Y$ implies that for any two tuples if $t_1[X] = t_2[X]$, they must have $t_1[Y] = t_2[Y]$.
5. CP: Child Pointer
6. KV: Key Value
7. Order of internal node in B+ tree is $P_1 \cdot CP + (P_1 - 1)KV \leftarrow \text{Block size}$

Inference Rules:

- 1. Reflexivity**
If $Y \subseteq X$, then $X \rightarrow Y$
- 2. Augmentation**
If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
- 3. Transitivity**
If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- 4. Union**
If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- 5. Decomposition**
If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- 6. Pseudo-transitivity**
If $A \rightarrow B$ and $BC \rightarrow D$, then $AC \rightarrow D$
7. Number of rows in cross product = $r_1 \times r_2$
8. Number of attributes in cross product = $m + n$
9. Complete set of operations are $\{\sigma, \pi, \cup, -, \times\}$

10. In clustering index, file is ordered on non-key field.
11. Secondary indexes are created on unordered file on either key or non-key field.

12. Different kind of keys – Primary Key, Secondary Key, Candidate Key, Alternate key, Foreign Key, Superkey, Compound key (or Composite key) (or concatenated key).

SOLVED EXAMPLES

1. Which of the following operations does not modify the database?

(a) Sorting (b) Insertion at the beginning
(c) Append (d) Modify

Solution: Sorting does not involving addition/insertion or modification of any element.

Ans. (a)

2. Which of the following operations is based on relational algebra?

(a) rename and union
(b) select and union
(c) only union
(d) rename, select and union

Solution: Rename, select and union are some of the operations of relational algebra.

Ans. (d)

3. ACID properties of a transaction are

(a) atomicity, constant, integrity, durability.
(b) atomicity, consistency, isolation, durability.
(c) atomicity, compact, in-built, durability.
(d) automatically, consistency, isolation, database.

Solution: ACID properties are atomicity, consistency, isolation and durability.

Ans. (b)

4. Which normal form is concerned with removing transitive dependency?

(a) 1NF (b) 2NF
(c) 3NF (d) BCNF

Solution: 3NF is also the highest normal form.

Ans. (c)

5. The concept Database Lock is used to overcome the problem of

(a) lost update and inconsistent data.
(b) uncommitted dependency and inconsistent data.
(c) inconsistent data only.
(d) lost updates, inconsistent data and uncommitted dependency.

Solution: Lost update, inconsistent data and uncommitted (WR) problems are overcome by database lock.

Ans. (d)

6. Database language may consist of the following?

(a) DDL and DML
(b) DML
(c) Query language
(d) DDL, DML, query language

Solution: DDL, DML, query language are all database languages.

Ans. (d)

7. Which of the following is not a function of DBA?

(a) Network maintenance
(b) Routine maintenance
(c) Defining the schema
(d) Data access through authorization

Solution: The role of database administrator is to manage the DBMS.

Ans. (a)

8. In a relational database the category type of information is represented in

(a) tuple. (b) field.
(c) primary key. (d) database name.

Solution: Field shows type of information in relational database.

Ans. (b)

9. Which key is used to identify a tuple uniquely?

(a) Primary key (b) Tuple key
(c) Unique key (d) Domain key

Solution: Primary key.

Ans. (a)

10. An association of the information is represented by

(a) an attribute. (b) a relationship.
(c) a normal form. (d) the records.

Solution: Relationship shows association of information.

Ans. (b)

11. In which stage of database design, all the necessary fields and their types of a database are listed?

(a) Data definition (b) Data field definition
(c) E-R diagram (d) User definition

Solution: In data definition stage, all the fields and their types are listed.

Ans. (a)

12. The maximum length of an attribute of type text is

(a) 127. (b) 255.
(c) 256. (d) It is a variable.

Solution: 255

Ans. (b)

13. When a transaction has completed all its operations and is waiting for commit or rollback action, the state of transaction is

(a) partially commit. (b) ready commit.
(c) half commit. (d) commit and rollback.

Solution: The state of that transaction is partially commit.

Ans. (a)

14. Which of the following statement is not false?

(a) Time stamp protocols avoid deadlock.
(b) Locking technique is used to avoid deadlock.
(c) 2Phase locking does not provide serializability.
(d) 2Phase deals with input and storing phase.

Solution: Time stamp protocol is a deadlock free protocol.

Ans. (a)

15. Data integrity control makes use of _____ for maintaining the integrity of database.

(a) specific alphabets (uppercase and lowercase alphabets)
(b) passwords
(c) relational algebra
(d) storing on a backup hard disk

Solution: It promotes and enforces some integrity rules for the reducing data redundancy and increasing data consistency.

Ans. (a)

16. Data warehouse provides

(a) transaction responsiveness.
(b) storage, functionality responsiveness to queries.
(c) demand and supply responsiveness.
(d) storage of transactions.

Solution: It provides data storage and responsiveness to queries.

Ans. (a)

17. If D_1, D_2, \dots, D_n are domains in a relational model, then the relation is a table, which is a subset of

(a) $D_1 \oplus D_2 \oplus \dots \oplus D_n$ (b) $D_1 \times D_2 \times \dots \times D_n$
(c) $D_1 \cup D_2 \cup \dots \cup D_n$ (d) $D_1 \cap D_2 \cap \dots \cap D_n$

Solution: $D_1 \times D_2 \times \dots \times D_n$

Ans. (a)

18. Which normal form is considered adequate for normal relational database design?

(a) 2NF (b) 5NF
(c) 4NF (d) 3NF

Solution: Decomposition in BCNF is not always lossless and dependency preserving. So, 3NF is considered to be most adequate form in relational database.

Ans. (d)

19. Let $R = (A, B, C, D, E, F)$ be a relation scheme with the following dependencies $C \rightarrow F, E \rightarrow A, EC \rightarrow D, A \rightarrow B$. Which of the following is a key for R ?

(a) CD (b) EC (c) AE (d) AC

Solution: To find key, we try to find closure.

$$(CD)^+ = \{C, D, F\}$$

$$(EC)^+ = \{A, B, C, D, E, F\}$$

$$(AE)^+ = \{A, B, E\}$$

$$(AC)^+ = \{A, B, C, F\}$$

Only EC satisfies the closure property.

Ans. (b)

20. Give the following relation instance:

x	y	z
1	4	2
1	5	3
1	6	3
3	2	2

Which of the following functional dependencies are satisfied by the instance?

(a) $XY \rightarrow Z$ and $Z \rightarrow Y$
(b) $YZ \rightarrow X$ and $Y \rightarrow Z$
(c) $YZ \rightarrow X$ and $X \rightarrow Z$
(d) $XZ \rightarrow Y$ and $Y \rightarrow X$