Fundamentals of database management system

BCAC0005

Module-2

Fundamental of Database Management System BCAC0005

Lecture - 11

Presented by:
Atul Kumar Uttam
Assistant Professor
Computer Engineering & Applications Department,
GLA University, Mathura
atul.uttam@gla.ac.in, +91-8979593001

Module –II Syllabus

File Organization Techniques	Sequential file organization
	Index File Organization, Random file organization
Normalization	Functional dependencies, Normal forms based on primary keys (1NF, 2NF, 3NF & BCNF), De-normalization, Lossless Join & Dependency Preserving Decomposition

Module –II Syllabus

Relational Algebra	Relational data model concept Relational Algebra(select operation, Project Operation)
	Union operation, set difference, Cartesian product
	Joins(Natural Join, outer Join(Left, Right, Full))
SQL	Data definition in SQL(CREATE, ALTER, DROP, TRUNCATE, RENAME)
	DML Queries(SELECT, INSERT UPDATE, DELETE)
	Views in SQL
	Specifying Constraints(Primary key, Unique, Foreign key, Null)
	Group By and Having clause
	Index in SQL

File Organization Techniques

- Storing the files in certain order is called file organization.
- The main objective of file organization is
  - Optimal selection of records i.e.; records should be accessed as fast as possible.
  - Any insert, update or delete transaction on records should be easy, quick and should not harm other records.
  - No duplicate records should be induced as a result of insert, update or delete
  - Records should be stored efficiently so that cost of storage is minimal.

Types of file organization

– Sequential File Organization
– Indexed Sequential Access Method
– Heap(random) File Organization
– Hash/Direct File Organization
– B+ Tree File Organization
– Cluster File Organization

1. Sequential File Organization

- Here each **file/records are stored one after the other** in a sequential manner.
- This can be achieved in two ways:
- In the first method:
  - Records are stored one after the other as they are inserted into the tables.
  - When a **new record is inserted**, it is placed **at the end** of the file.
  - In the case of any modification or deletion of record, the record will be searched in the memory blocks.
  - Once it is found, it will be marked for deleting and new block of record is entered.

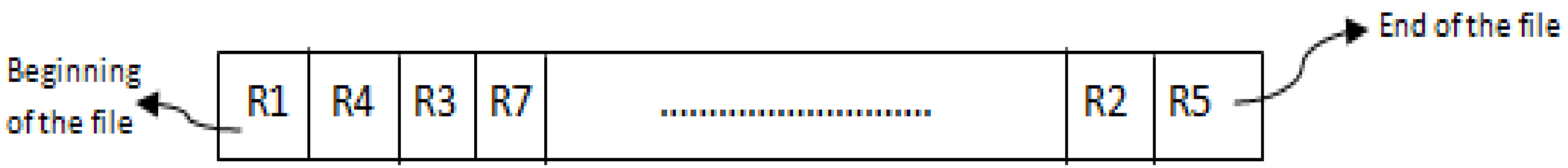
Sequential File Organization

- In the diagram above, R1, R2, R3 etc are the records.

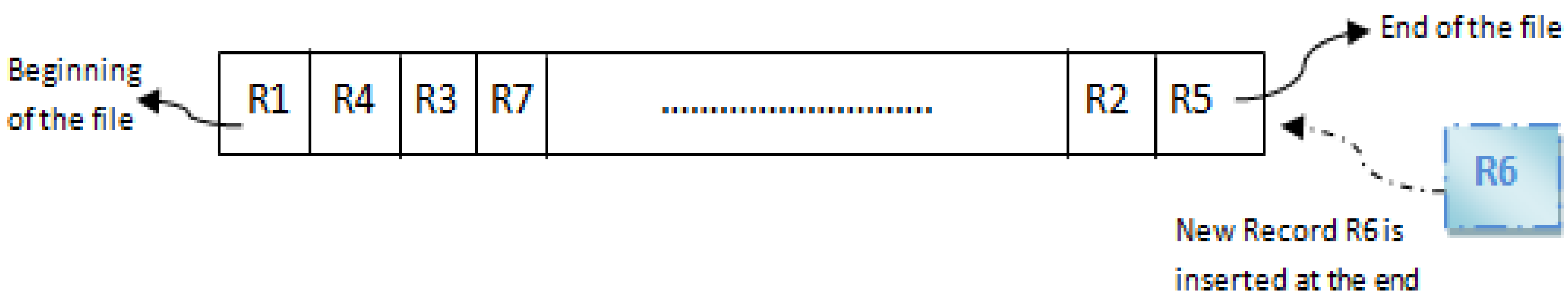
- They contain all the attribute of a row. i.e.; when we say student record, it will have his id, name, address, course, DOB etc.

- Similarly R1, R2, R3 etc can be considered as one full set of attributes.

Sequential File Organization

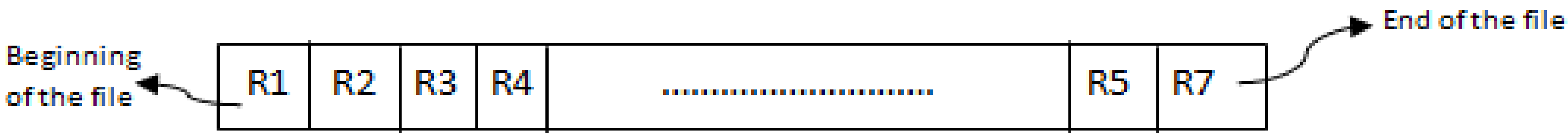


Inserting a new record

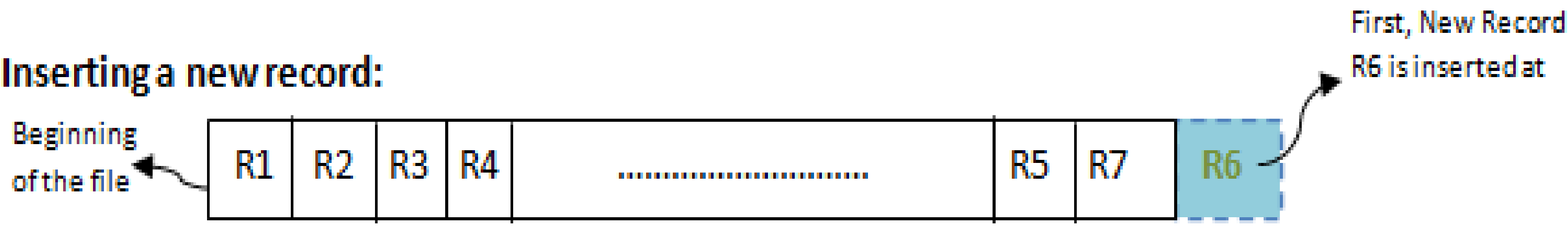


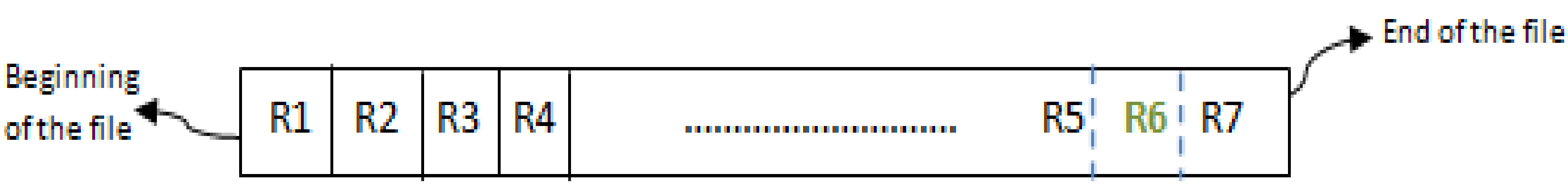
Sequential File Organization

- In the second method,
  - records are sorted (either ascending or descending) each time they are inserted into the system.
  - This method is called **sorted file method**.
  - Sorting of records may be based on the primary key or on any other columns.
  - Whenever a new record is inserted, it will be inserted at the end of the file and then it will sort – ascending or descending based on key value and placed at the correct position.
  - In the case of update, it will update the record and then sort the file to place the updated record in the right place.
  - Same is the case with delete.



Inserting a new record:

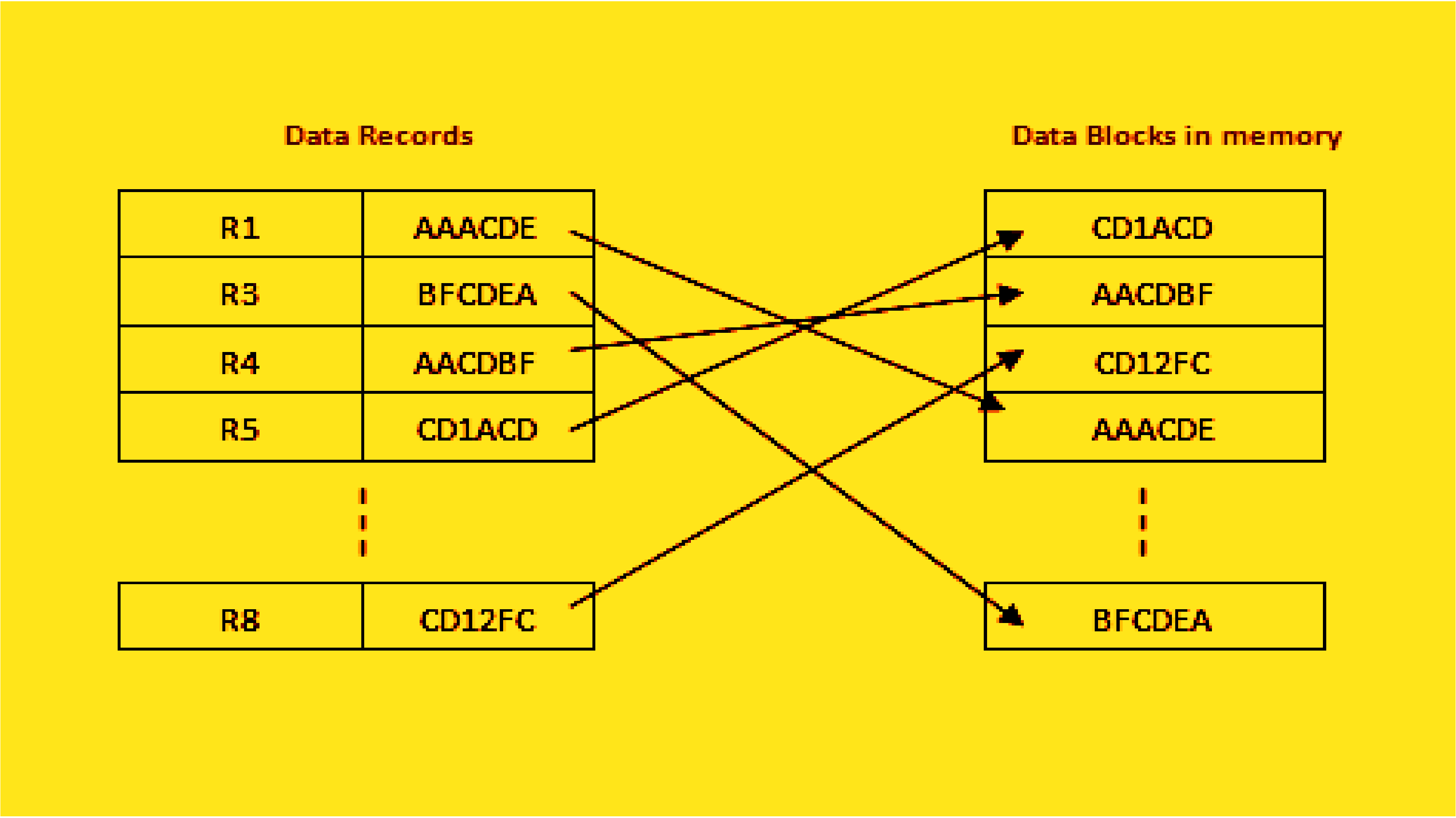




2. Indexed Sequential Access Method (ISAM)

- This is an **advanced sequential file organization** method.
- Here **records are stored in order of primary key** in the file.
- **For each primary key, an index value is generated** and mapped with the record.
- This **index is** nothing but the **address of record in the file**.
- In this method, if any record has to be retrieved, based on its index value, the data block address is fetched and the record is retrieved from memory.

Indexed Sequential Access Method (ISAM)



Indexed Sequential Access Method (ISAM)

Advantages of ISAM

- Since each record has its data block address, **searching for a record in larger database is easy and quick**.
- This method gives flexibility of **using any column as key field** and index will be generated based on that. In addition to the primary key and its index, we can have **index generated for other fields too**.
- It supports range retrieval, partial retrieval of records.
- Since the index is based on the key value, we can retrieve the data for the given range of values.

Disadvantages of ISAM

- An **extra cost to maintain index** has to be afforded. i.e.; we need to have **extra space in the disk to store this index value**. When there is multiple key-index combinations, the disk space will also increase.
- As the **new records are inserted**, these files have to be **restructured to maintain the sequence**.
- Similarly, when the record is deleted, the space used by it needs to be released. Else, the performance of the database will slow down.

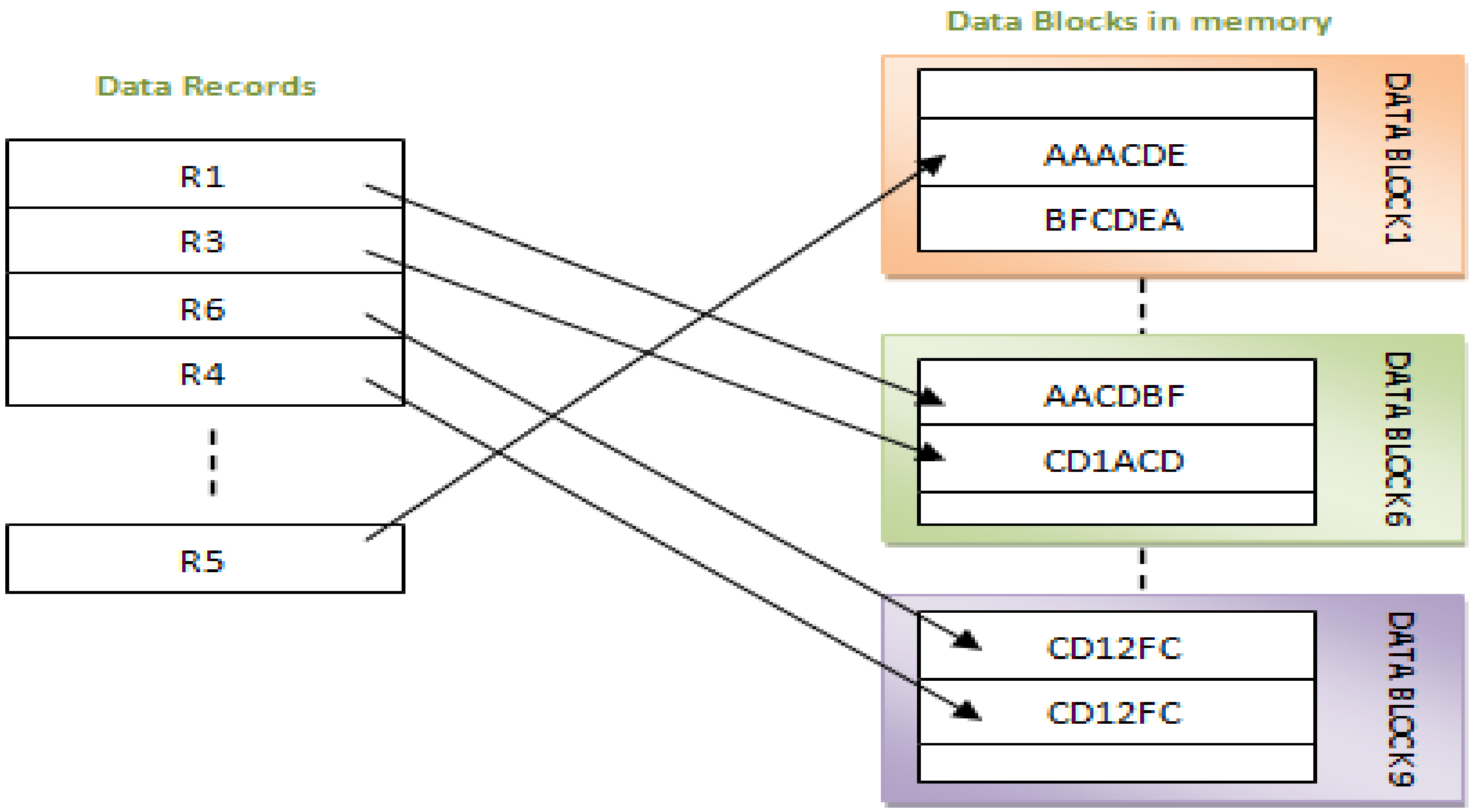
3. Random(Heap) file organization technique

- Here **records are inserted at the end of the file** as and when they are inserted.
- There is **no sorting or ordering** of the records.
- Once the **data block is full, the next record is stored in the new block**.
- This **new block need not be the very next block**.
- This method can **select any block in the memory to store** the new records.

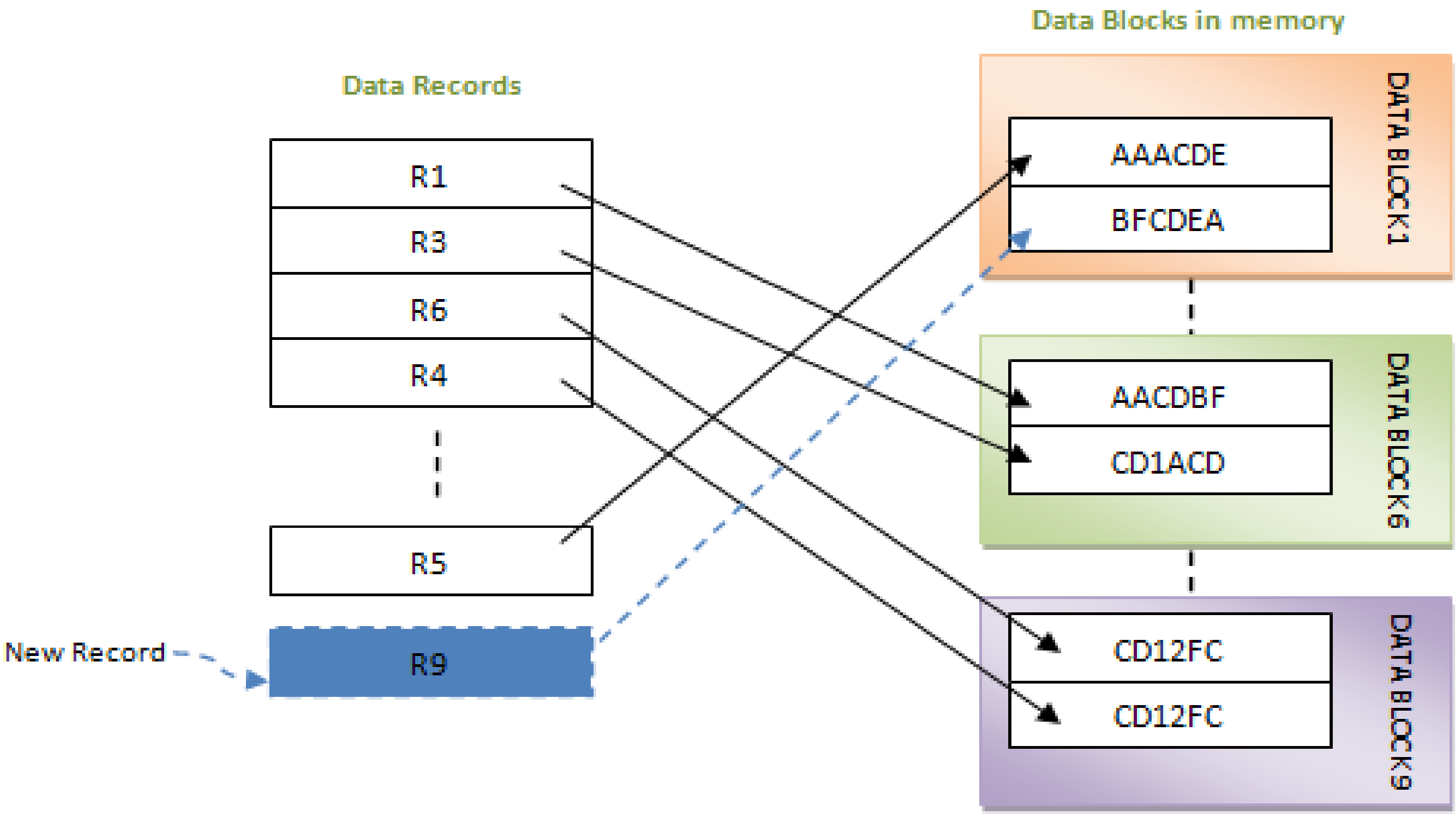
Random(Heap) file organization technique

- It **is similar to pile file in the sequential method, but here data blocks are not selected sequentially**.
- They can be any data blocks in the memory.
- It is the responsibility of the DBMS to store the records and manage them.

Random(Heap) file organization technique



Random(Heap) file organization technique



Random(Heap) file organization technique

- When a record has to be retrieved from the database, in this method, we need to traverse from the beginning of the file till we get the requested record.
- Hence fetching the records in very huge tables, it is time consuming.
- To delete or update a record, first we need to search for the record.
- Again, searching a record is similar to retrieving it- start from the beginning of the file till the record is fetched.
- If it is a small file, it can be fetched quickly. But larger the file, greater amount of time needs to be spent in fetching.

Random(Heap) file organization technique

- In addition, while deleting a record, the record will be deleted from the data block.
- But it will not be freed and it cannot be re-used.
- Hence as the number of record increases, the memory size also increases and hence the efficiency decreases.
- For the database to perform better, DBA has to free this unused memory periodically.

Random(Heap) file organization technique

Advantages of Heap File Organization

- It is **suited for very small files** as the fetching of records is faster in them. As the file size grows, linear search for the record becomes time consuming.

Disadvantages of Heap File Organization

- This method is **inefficient for larger databases** as it takes time to search/modify the record.
- Proper **memory management is required** to boost the performance. Otherwise there would be **lots of unused memory blocks** lying and memory size will simply be growing.

Fundamental of Database Management System BCAC0005

Lecture - 12

Presented by:
Atul Kumar Uttam
Assistant Professor
Computer Engineering & Applications Department,
GLA University, Mathura
atul.uttam@gla.ac.in, +91-8979593001

Relational model concepts

- Relational data model is the primary data model, which is used widely around the world for data storage and processing.
- This model is simple and it has all the properties and capabilities required to process data with storage efficiency

Basic concepts of relational data model

- **Tables** − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.
- **Tuple** − A single row of a table, which contains a single record for that relation is called a tuple.
- **Relation instance** − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
- **Relation schema** − A relation schema describes the relation name (table name), attributes, and their names.
- **Relation key** − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.
- **Attribute domain** − Every attribute has some pre-defined value scope, known as attribute domain.

Table also called Relation

Primary Key

Domain
Ex: NOT NULL

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Tuple OR Row

Total # of rows is Cardinality

Column OR Attributes

Total # of column is Degree

Fundamental of Database Management System BCAC0005

Lecture - 13

Presented by:

Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,
GLA University, Mathura

atul.uttam@gla.ac.in, +91-8979593001

Constraints in DBMS

- Relational constraints are the restrictions imposed on the database contents and operations.
- They ensure the correctness of data in the database.

Types of Constraints in DBMS

- Relational Constraints
  - Domain Constraint
  - Tuple Uniqueness Constraint
  - Key Constraint
  - Entity Integrity Constraint
  - Referential Integrity Constraint

1. Domain Constraint

- Domain constraint defines the domain or set of values for an attribute.
- It specifies that the value taken by the attribute must be the **atomic value from its domain**.

Example

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
S004	Rahul	A

Here, value **'A'** is not allowed since only integer values can be taken by the age attribute.

2. Tuple Uniqueness Constraint

- Tuple Uniqueness constraint specifies that all the tuples must be necessarily unique in any relation.

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
S004	Rahul	20

STU_ID	Name	Age
S001	**Akshay**	**20**
S001	**Akshay**	**20**
S003	Shashank	20
S004	Rahul	20

12-B Status from UGC

3. Key Constraint

- Key constraint specifies that in any relation-
  - All the values of primary key must be unique.
  - The value of primary key must not be null.

STU_ID	Name	Age
S001	Akshay	20
S001	Abhishek	21
S003	Shashank	20
S004	Rahul	20

4. Entity Integrity Constraint

- Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation.
- This is because the presence of null value in the primary key violates the uniqueness property.

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
	Rahul	20

5. Referential Integrity Constraint

- This constraint is enforced when a foreign key references the primary key of a relation.
- It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.

<u>Important Results</u>

- We can not insert a record into a referencing relation if the corresponding record does not exist in the referenced relation.
- We can not delete or update a record of the referenced relation if the corresponding record exists in the referencing relation.

- Here, relation ‘Student’ references the relation ‘Department’.

Student (<u>Stu_ID</u> , Name , Dept_no)   Department (<u>Dept_no</u> , Dept_name)

(Referencing Relation)   (Referenced Relation)

Foriegn Key

<u>STU_ID</u>	Name	Dept_no
S001	Akshay	D10
S002	Abhishek	D10
S003	Shashank	D11
S004	Rahul	**D14**

<u>Dept_no</u>	Dept_name
D10	ASET
D11	ALS
D12	ASFL
D13	ASHS

Referential Integrity Constraint Violation

- There are following three possible causes of violation of referential integrity constraint-

- **Cause-01:** Insertion in a referencing relation
- **Cause-02:** Deletion from a referenced relation
- **Cause-03:** Updation in a referenced relation

Cause-01: Insertion in a Referencing Relation

- It is allowed to insert only those values in the referencing attribute which are already present in the value of the referenced attribute.

Student (Roll_no , Name , Age , Branch_Code)

(Referencing Relation)

Branch (Branch_Code , Branch_Name)

(Referenced Relation)

Foreign Key

Roll_no	Name	Age	Branch_Code
1	Rahul	22	CS
2	Anjali	21	CS
3	Teena	20	IT

Branch_Code	Branch_Name
CS	Computer Science
EE	Electronics Engineering
IT	Information Technology
CE	Civil Engineering

Cause-01: Insertion in a Referencing Relation

Student (Roll_no , Name , Age , Branch_Code)

(Referencing Relation)

Foreign Key

Branch (Branch_Code , Branch_Name)

(Referenced Relation)

Roll_no	Name	Age	Branch_Code
1	Rahul	22	CS
2	Anjali	21	CS
3	Teena	20	IT
4	James	23	ME

Branch_Code	Branch_Name
CS	Computer Science
EE	Electronics Engineering
IT	Information Technology
CE	Civil Engineering

Cause-02: Deletion from a Referenced Relation

- It is not allowed to delete a row from the referenced relation if the referencing attribute uses the value of the referenced attribute of that row.

Student (Roll_no , Name , Age , Branch_Code)

(Referencing Relation)

Foreign Key

Branch (Branch_Code , Branch_Name)

(Referenced Relation)

Roll_no	Name	Age	Branch_Code
1	Rahul	22	CS
2	Anjali	21	CS
3	Teena	20	IT

Branch_Code	Branch_Name
CS	Computer Science
EE	Electronics Engineering
IT	Information Technology
CE	Civil Engineering

- To handle this we can simultaneously delete those tuples from the referencing relation where the referencing attribute uses the value of referenced attribute being deleted.
- This method of handling the violation is called as **On Delete Cascade**.

OR

- This method involves aborting or deleting the request for a deletion from the referenced relation if the value is used by the referencing relation.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Cause-03: Updation in a Referenced Relation

- It is not allowed to update a row of the referenced relation if the referencing attribute uses the value of the referenced

Student (Roll_no , Name , Age , Branch_Code)

(Referencing Relation)

Branch (Branch_Code , Branch_Name)

(Referenced Relation)

Foreign Key

Roll_no	Name	Age	Branch_Code
1	Rahul	22	CS
2	Anjali	21	CS
3	Teena	20	IT

Branch_Code	Branch_Name
CSE	Computer Science
EE	Electronics Engineering
IT	Information Technology
CE	Civil Engineering

- We can simultaneously updating those tuples of the referencing relation where the referencing attribute uses the referenced attribute value being updated.

OR

- This method involves aborting or deleting the request for an updation of the referenced relation if the value is used by the referencing relation.

Fundamental of
Database Management System
BCAC0005

Lecture - 14

Presented by:

Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,
GLA University, Mathura

atul.uttam@gla.ac.in, +91-8979593001

Relational Algebra

- Relational algebra
  - Basic set of operations for the relational model
- Relational algebra expression
  - Sequence of relational algebra operations

- Each relation is defined to be a set of tuples in the formal relational mode

The SELECT Operation

Subset of the tuples from a relation that satisfies a selection condition:

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

Boolean expression <selection condition> contains clauses of the form:

<attribute name> <comparison op> <constant value>

Or

<attribute name> <comparison op> <attribute name>

The SELECT Operation

- **<attribute name>** is the name of an attribute of R,

- **<comparison op>** is normally one of the operators {=, <, ≤,>, ≥, ≠}, and

- **<constant value>** is a constant value from the attribute domain

The SELECT Operation

- **<selection condition>** applied independently to each individual tuple t in R
  - If condition evaluates to TRUE, tuple selected
- Boolean conditions AND, OR, and NOT
- Unary
  - Applied to a single relation

The SELECT Operation

Example

- To select the EMPLOYEE tuples whose department is 4,

$$\sigma_{Dno=4}(EMPLOYEE)$$

- To select the EMPLOYEE whose salary is greater than $30,000

$$\sigma_{Salary>30000}(EMPLOYEE)$$

The SELECT Operation

Example

- select all employees who either work in department 4 and make over $25,000 per year, or work in department 5 and make over $30,000

$$\sigma_{(\text{Dno}=4 \text{ AND } \text{Salary}>25000) \text{ OR } (\text{Dno}=5 \text{ AND } \text{Salary}>30000)}(\text{EMPLOYEE})$$

The SELECT Operation

- The **degree of the relation** resulting from a SELECT operation—**its number of attributes**—is the same as the degree of R.

- The number of tuples in the resulting relation is always less than or equal to the number of tuples in R.

The SELECT Operation

- Notice that the **SELECT operation is commutative**; that is,

$$\sigma_{<cond1>}(\sigma_{<cond2>}(R)) = \sigma_{<cond2>}(\sigma_{<cond1>}(R))$$

- Hence, a sequence of SELECTs can be applied in any order.

The SELECT Operation

- In addition, we can always combine a cascade (or sequence) of SELECT operations into a single SELECT operation with a conjunctive (AND) condition; that is,

$$\sigma_{<cond1>}(\sigma_{<cond2>}(\ldots(\sigma_{<condn>}(R)) \ldots)) =$$

$$\sigma_{<cond1>\ \mathbf{AND} <cond2>\ \mathbf{AND} \ldots \mathbf{AND}\ <condn>}(R)$$

The SELECT Operation

- In SQL, the SELECT condition is typically specified in the WHERE clause of a query.

$$\sigma_{\text{Dno=4 AND Salary>25000}}\ \textbf{(EMPLOYEE)}$$

- would correspond to the following SQL query:

SELECT *

FROM EMPLOYEE

WHERE Dno=4 **AND** Salary>25000**;**

Selection

$$\sigma_{rating>8}(S2)$$

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

QUIZ……??

Guess the output….

$$\sigma_{A=3}\left(\begin{array}{|c|c|}\hline A & B \\ \hline 1 & 3 \\ 2 & 4 \\ \hline\end{array}\right) = \emptyset$$

$$\sigma_{C=1}\left(\begin{array}{|c|c|}\hline A & B \\ \hline 1 & 3 \\ 2 & 4 \\ \hline\end{array}\right) = \text{Error}$$

The PROJECT Operation

- Selects columns from table and discards the other columns:

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

- Duplicate elimination
  - Result of PROJECT operation is a set of distinct tuples

Projection

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$$\pi_{sname,rating}(S2)$$

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

age
35.0
55.5

$$\pi_{age}(S2)$$

Selection & Projection

$S2$

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

sname	rating
yuppy	9
rusty	10

Selection & Projection

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

sname	rating
yuppy	9
rusty	10

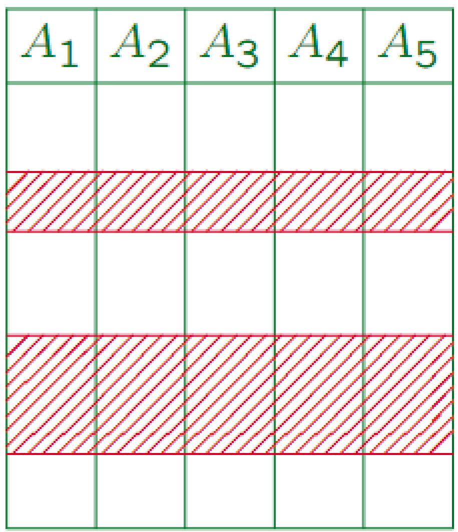
QUIZ…….??

- Guess the output…

$$\pi_B\left(\begin{array}{|c|c|}\hline A & B \\ \hline 1 & 4 \\ 2 & 5 \\ 3 & 4 \\ \hline \end{array}\right) = \begin{array}{|c|}\hline B \\ \hline 4 \\ 5 \\ \hline \end{array}$$

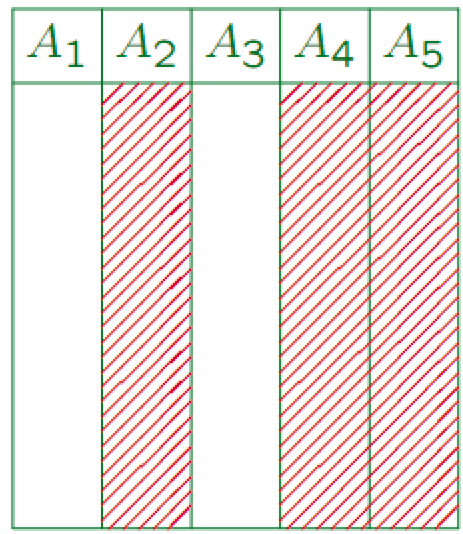
Summary

Selection σ



(Filters some rows)

Projection π



(Maps each row)

- Which of the following relational algebra expressions are syntactically correct? What do they mean?

1. STUDENTS.
2. $\sigma_{MAXPT=10}(EXERCISES)$.
3. $\pi_{FIRST}(\pi_{LAST}(STUDENTS))$.
4. $\sigma_{POINTS \leq 5}(\sigma_{POINTS \geq 1}(RESULTS))$.
5. $\sigma_{POINTS}(\pi_{POINTS=10}(RESULTS))$.

- Which of the following relational algebra expressions are syntactically correct? What do they mean?

1. STUDENTS.
2. $\sigma_{\text{MAXPT=10}}(\text{EXERCISES})$.
3. **$\pi_{\text{FIRST}}(\pi_{\text{LAST}}(\text{STUDENTS}))$.** **Wrong**
4. $\sigma_{\text{POINTS} \leq 5}(\sigma_{\text{POINTS} \geq 1}(\text{RESULTS}))$.
5. **$\sigma_{\text{POINTS}}(\pi_{\text{POINTS=10}}(\text{RESULTS}))$.**

Fundamental of
Database Management System
BCAC0005

Lecture - 15

Presented by:

Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,
GLA University, Mathura

atul.uttam@gla.ac.in, +91-8979593001

Union

S1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$S1 \cup S2$$

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

Intersection

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

$$S1 \cap S2$$

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

Set-Difference

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$S1 - S2$$

sid	sname	rating	age
22	dustin	7	45.0

Cross-Product (Cartesian Product) X

- It is also called “cross product”
- $R \times S$ concatenates each tuple from R with each tuple from S.
- If the relation R contains n tuples, and the relation S contains m tuples, then $R \times S$ contains $n * m$ tuples.
- $R \times S$ is written in SQL as

```
SELECT *
FROM R, S
```

Cross-Product (Cartesian Product)

- Each row of S1 is paired with each row of R1.
- *Result schema* has one field per field of S1 and R1, with field names `inherited' if possible.
  - *Conflict*: Both S1 and R1 have a field called *sid*.
  - in practice the Cartesian product is rarely used.

Cross-Product (Cartesian Product)

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

$(S1 \times R1)$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Pname	Price
Laptop	1500
Car	20000
Airplane	3000000

Pname	Cname	Cost
Laptop	CPU	500
Laptop	HDD	300
Laptop	CASE	700
Car	Wheels	1000

Pname	Price	Pname	Cname	Cost
Laptop	1500	Laptop	CPU	500
Laptop	1500	Laptop	HDD	300
Laptop	1500	Laptop	CASE	700
Laptop	1500	Car	Wheels	1000
Car	20000	Laptop	CPU	500
Car	20000	Laptop	HDD	300
Car	20000	Laptop	CASE	700
Car	20000	Car	Wheels	1000
Airplane	3000000	Laptop	CPU	500
Airplane	3000000	Laptop	HDD	300
Airplane	3000000	Laptop	CASE	700
Airplane	3000000	Car	Wheels	1000

Renaming

- An operator $\rho_R(S)$ that pretends "R." to all attribute names is sometimes useful:

$$\rho_R\left(\begin{array}{|c|c|}\hline A & B \\ \hline 1 & 2 \\ 3 & 4 \\ \hline\end{array}\right) = \begin{array}{|r|r|}\hline R.A & R.B \\ \hline 1 & 2 \\ 3 & 4 \\ \hline\end{array}$$

- This is only an abbreviation for an application of the projection:

$$\pi_{R.A \leftarrow A,\ R.B \leftarrow B}(S).$$

- Otherwise, attribute names in relational algebra do not automatically contain the relation name.

Fundamental of Database Management System BCAC0005

Lecture - 16

Presented by:
Atul Kumar Uttam
Assistant Professor
Computer Engineering & Applications Department,
GLA University, Mathura
atul.uttam@gla.ac.in, +91-8979593001

Joins: used to combine relations

- ***Condition Join***: $R \bowtie_c S = \sigma_c (R \times S)$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$S1 \bowtie_{S1.sid < R1.sid} R1$$

- ***Result schema*** **same as that of cross-product.**
- **Fewer tuples than cross-product, might be able to compute more efficiently**
- Sometimes called a ***theta-join***.

Join

- ***Equi-Join***: A special case of condition join where the condition *c* contains only ***equalities***.

S1.sid	sname	rating	age	R1.sid	bid	day
22	dustin	7	45.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

$$S1 \bowtie_{S1.sid = R1.sid} R1$$

- ***Natural Join***: **Equijoin on *all* common fields,** but only one copy of fields for which equality is specified.

- Theta (θ) Join, equi join and natural joins are called Inner join.

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Note

- A *theta join* allows for arbitrary comparison relationships (such as $\geq$).
- An *equijoin* is a theta join using the equality operator.
- A *natural join* is an equijoin on attributes that have the same name in each relationship.
- Additionally, a natural join removes the duplicate columns involved in the equality comparison so only 1 of each compared column remains.

Outer Joins

- An inner join includes only those tuples with matching attributes and the rest are discarded in the resulting relation.
- Therefore, we need to use outer joins to include all the tuples from the participating relations in the resulting relation.
- There are three kinds of outer joins – left outer join, right outer join, and full outer join.

Left Outer Join(R ⟕ S)

Courses	
A	**B**
100	Database
101	Mechanics
102	Electronics

⟕

HoD	
A	**B**
100	Alex
102	Maya
104	Mira

Courses ⟕ HoD			
A	**B**	**C**	**D**
100	Database	100	Alex
101	Mechanics	---	---
102	Electronics	102	Maya

⟖

Right Outer Join: (R ⟖ S)

Courses ⟖ HoD			
A	**B**	**C**	**D**
100	**Database**	**100**	**Alex**
102	**Electronics**	**102**	**Maya**
---	**---**	**104**	**Mira**

⟗

Full Outer Join: (R ⟗ S)

Courses ⟗ HoD			
A	**B**	**C**	**D**
100	**Database**	**100**	**Alex**
101	**Mechanics**	**---**	**---**
102	**Electronics**	**102**	**Maya**
---	**---**	**104**	**Mira**

Division

- Not supported as a primitive operator, but useful for expressing queries like: *Find sailors who have reserved <u>all</u> boats*.
- Let *A* have 2 fields, *x* and *y*; *B* have only field *y*:
  - ***A/B*** = $\left\{ \langle x \rangle \mid \exists \langle x, y \rangle \in A \ \forall \langle y \rangle \in B \right\}$
  - i.e., ***A/B* contains all *x* tuples (sailors) such that *for <u>every</u> y tuple (boat) in B, there is an xy tuple in A.***

A/B **contains all *x* tuples such that *for <u>every</u> y tuple in B, there is an xy tuple in A.***

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

A

pno
p2

B1

pno
p2
p4

B2

pno
p1
p2
p4

B3

sno
s1
s2
s3
s4

A/B1

sno
s1
s4

A/B2

sno
s1

A/B3

Division Operation – Example

- Relations *r, s*:

A	*B*
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
$\in$	6
$\in$	1
β	2

r

B
1
2

s

- $r \div s$:

A
α
β

Another Division Example

- Relations *r, s*:

A	*B*	*C*	*D*	*E*
α	a	α	a	*1*
α	a	γ	a	*1*
α	a	γ	b	*1*
β	a	γ	a	*1*
β	a	γ	b	*3*
γ	a	γ	a	*1*
γ	a	γ	b	*1*
γ	a	β	b	*1*

r

D	*E*
a	*1*
b	*1*

s

- $r \div s$:

A	*B*	*C*
α	a	γ
γ	a	γ

Example of Division

- **Find all customers who have an account at all branches located in Chicago**
  - Branch (bname, assets, bcity)
  - Account (bname, acct#, cname, balance)

Example of Division

r1: Find all branches in Chicago

$$r1 = \pi_{bname}(\sigma_{bcity='Chicago'}Branch)$$

r2: Find (bname, cname) pair from Account

$$r2 = \pi_{bname,cname}(Account)$$

r3: Customers in r2 with every branch name in r1

$$r3 = r2 \div r1$$

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation *R*.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of *R*, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$

EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{\text{(<join attributes 1>),}\ \text{(<join attributes 2>)}} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{\text{(<join attributes 1>),}\ \text{(<join attributes 2>)}} R_2$ OR $R_1 * R_2$

UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$

CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Exercise 1

Given relational schema:

Sailors (sid, sname, rating, age)

Reservation (sid, bid,_date)

Boats (bid, bname, color)

1) Find names of sailors who’ve reserved boat #103
2) Find names of sailors who’ve reserved a red boat
3) Find sailors who’ve reserved a red or a green boat
4) Find sailors who’ve reserved a red <u>and</u> a green boat
5) Find the names of sailors who’ve reserved all boats

1) Find names of sailors who've reserved boat #103

- Solution 1: $\pi_{sname}((\sigma_{bid=103} \text{Reserves}) \bowtie Sailors)$

❖ Solution 2: $\rho\ (Temp1,\ \sigma_{bid=103} \text{Reserves})$

$\rho\ (Temp2,\ Temp1 \bowtie Sailors)$

$\pi_{sname}(Temp2)$

❖ Solution 3: $\pi_{sname}(\sigma_{bid=103}(\text{Reserves} \bowtie Sailors))$

2) Find names of sailors who've reserved a red boat

- Boats (bid, bname, color)
- Information about boat color only available in Boats; so need an extra join:

$$\pi_{sname}((\sigma_{color='red'} Boats) \bowtie \text{Reserves} \bowtie Sailors)$$

❖ A more efficient solution -- why more efficient?

$$\pi_{sname}(\pi_{sid}((\pi_{bid}\sigma_{color='red'} Boats) \bowtie \text{Res}) \bowtie Sailors)$$

A query optimizer can find this, given the first solution!

3) Find sailors who’ve reserved a red or a green boat

- Can identify **all red or green boats**, then find sailors who’ve reserved one of these boats:

$$\rho(Tempboats, (\sigma_{color='red' \vee color='green'} Boats))$$

$$\pi_{sname}(Tempboats \bowtie Reserves \bowtie Sailors)$$

4) Find sailors who've reserved a red <u>and</u> a green boat

- Previous approach won't work! Why?
- Must identify sailors who've reserved red boats, sailors who've reserved green boats, then find the intersection (note that *sid* is a key for Sailors):

4) Find sailors who've reserved a red <u>and</u> a green boat

- Previous approach won't work! Why?
- Must identify sailors who've reserved red boats, sailors who've reserved green boats, then find the intersection (note that *sid* is a key for Sailors):

$$\rho(Tempred, \pi_{sid}((\sigma_{color='red'} Boats) \bowtie \text{Reserves}))$$

$$\rho\ (Tempgreen,\ \pi_{sid}((\sigma_{color='green'}\ Boats) \bowtie \text{Reserves}))$$

$$\pi_{sname}((Tempred \cap Tempgreen) \bowtie Sailors)$$

5) Find the names of sailors who've reserved all boats

- Uses division; schemas of the input relations to division (/) must be carefully chosen:

$$\rho\ (Tempsids, (\pi_{sid,bid}\mathrm{Re}serves) \ /\ (\pi_{bid}\, Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

❖ To find sailors who've reserved all 'Interlake' boats:

$$\ldots\ldots \quad /\ \pi_{bid}(\sigma_{bname='Interlake'}\ Boats)$$

Exercise 2

- **Student(sID, surName, firstName, campus, email, cgpa)**
- **Course(dept, cNum, name, breadth)**
- **Offering(oID, dept, cNum, term, instructor)**
- **Took(sID, oID, grade)**

- Student number of all students who have taken cNum= 343 from dept = csc.

$$\Pi_{sID}\sigma_{dept=\text{“}csc\text{”} \wedge cNum=343}(Took \bowtie Offering)$$

- Student number of all students who have taken csc343 and earned an A+ in it.

$$\Pi_{sID}\sigma_{dept=\text{“csc”} \wedge cNum=343 \wedge grade \geq 90}(Took \bowtie Offering)$$

Exercise 3

- employee (<u>person-name</u>, street, city)
- works (<u>person-name</u>, company-name, salary)
- company (<u>company-name</u>, city)
- manages (<u>person-name</u>, manager-name)

a. Find the names of all employees who work for First Bank Corporation.

$\Pi_{\text{person-name}}\ (\sigma_{\text{company-name = “First Bank Corporation”}}\ (\text{works}))$

Find the names and cities of residence of all employees who work for First Bank Corporation.

$$\Pi_{\text{person-name, city}} (\text{employee} \bowtie (\sigma_{\text{company-name} = \text{“First Bank Corporation”}} (\text{works})))$$

Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than \$10,000 per annum.

$\Pi_{\text{person-name, street, city}}(\sigma_{\text{company-name = “First Bank Corporation”} \wedge \text{salary >10000}})\text{works} \bowtie \text{employee})$

Find the names of all employees in this database who live in the same city as the company for which they work.

$$\Pi_{\text{person-name}} \text{ (employee } \bowtie \text{ works } \bowtie \text{ company)}$$

Assume the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located

$$\Pi_{\text{company-name}}\ (\text{company} \div (\Pi_{\text{city}}\ (\sigma_{\text{company-name = “Small Bank Corporation”}}\ (\text{company}))))$$

Exercise 4

- Suppliers(<u>sid: integer</u>, sname: string, address: string)
- Parts(<u>pid: integer</u>, pname: string, color: string)
- Catalog(<u>sid: integer</u>, pid: integer, cost: real)

Find the names of suppliers who supply some red part.

$\pi_{sname}(\pi_{sid}((\pi_{pid}\sigma_{color=red}\ \mathbf{Parts}) \bowtie \text{Catalog}) \bowtie \text{Suppliers})$

Find the sids of suppliers who supply some red or green part.

$\pi_{sid}(\pi_{pid}(\sigma_{color=red \vee color=green}\ Parts) \bowtie\ catalog)$

Find the sids of suppliers who supply some red part or are at 221 Packer Street.

$$\rho(R1, \pi_{sid}((\pi_{pid}\sigma_{color=red}\ \text{Parts}) \bowtie \text{Catalog}))$$

$$\rho(R2, \pi_{sid}(\sigma_{address=221PackerStreet}\ \text{Suppliers}))$$

$R1 \cup R2$

Find the sids of suppliers who supply some red part and some green part.

$\rho(R1, \pi_{sid}((\pi_{pid}\sigma_{color=red}\ Parts) \bowtie Catalog))$

$\rho(R2, \pi_{sid}((\pi_{pid}\sigma_{color=green}\ Parts) \bowtie Catalog))$

$R1 \cap R2$

Find the sids of suppliers who supply every part.

$(\pi_{sid,pid}\ Catalog)/(\pi_{pid}\ Parts)$

Find the sids of suppliers who supply every red part.

$(\pi_{sid,pid} \text{ Catalog})/(\pi_{pid}\sigma_{color=red} \text{ Parts})$

Find the sids of suppliers who supply every red or green part

$(\pi_{sid,pid} Catalog) / (\pi_{pid} \sigma_{color=red \vee color=green} Parts)$

Fundamental of
Database Management System
BCAC0005

Lecture - 17

Presented by:

Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,
GLA University, Mathura

atul.uttam@gla.ac.in, +91-8979593001

Functional Dependency in DBMS

- In any relation, a functional dependency

  $\alpha \rightarrow \beta$ holds if
- Two tuples having same value of attribute α also have same value for attribute β.

Mathematically,

- If α and β are the two sets of attributes in a relational table R where:

$$\alpha \subseteq R$$

$$\beta \subseteq R$$

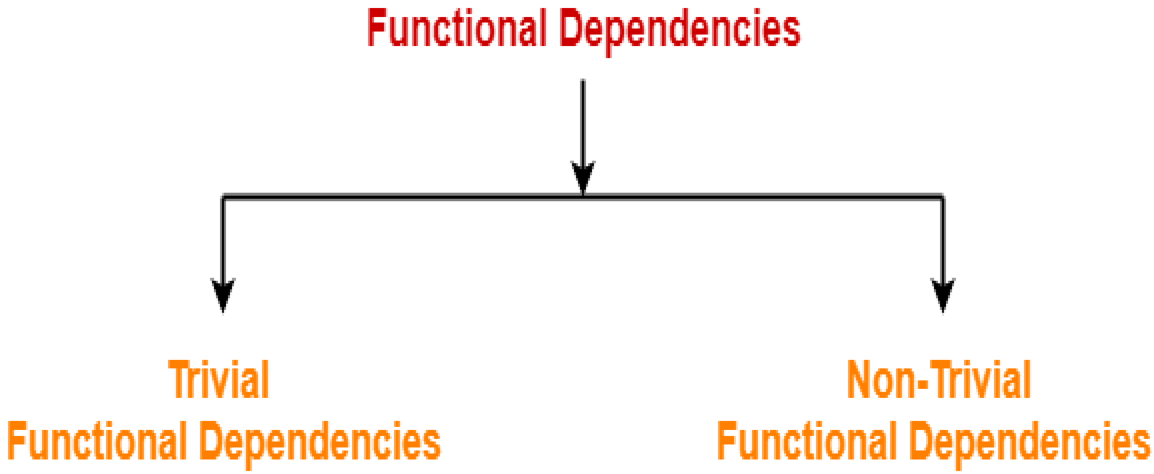
- Then, for a functional dependency to exist from α to β, If t1[α] = t2[α], then t1[β] = t2[β]

- $\mathbf{f_d : \alpha \rightarrow \beta}$

α	**β**
t1[α]	t1[β]
t2[α]	t2[β]
…….	…….

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Types Of Functional Dependencies



1. Trivial Functional Dependencies

- A functional dependency X → Y is said to be trivial if and only if Y ⊆ X.

- Thus, if RHS of a functional dependency is a subset of LHS, then it is called as a trivial functional dependency.

Examples

- AB → A
- AB → B
- AB → AB

2. Non-Trivial Functional Dependencies

- A functional dependency X → Y is said to be non-trivial if and only if Y ⊄ X.
- Thus, if there exists at least one attribute in the RHS of a functional dependency that is not a part of LHS, then it is called as a non-trivial functional dependency.

Examples

- AB → BC
- AB → CD

Inference Rules

Reflexivity

- If B is a subset of A, then A → B always holds.

Transitivity

- If A → B and B → C, then A → C always holds.

Augmentation

- If A → B, then AC → BC always holds.

Decomposition

- If A → BC, then A → B and A → C always holds.

Composition

- If A → B and C → D, then AC → BD always holds.

Additive

- If A → B and A → C, then A → BC always holds.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Rules for Functional Dependency

Rule-01:

- A functional dependency X → Y will always hold if all the values of X are unique (different) irrespective of the values of Y.

A	B	C	D	E
5	4	3	2	2
8	5	3	2	1
1	9	3	3	5
4	7	3	3	8

A → B
A → BC
A → CD
A → BCD
A → DE
A → BCDE

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Rule-02:

- A functional dependency X → Y will always hold if all the values of Y are same irrespective of the values of X.

A	B	C	D	E
5	4	3	2	2
8	5	3	2	1
1	9	3	3	5
4	7	3	3	8

A → C
AB → C
ABDE → C
DE → C
AE → C

Closure of an Attribute Set

- The set of all those attributes which can be functionally determined from an attribute set is called as a closure of that attribute set.
- Closure of attribute set $\{X\}$ is denoted as $\{X\}^+$

Steps to Find Closure of an Attribute Set

Step-01:

- Add the attributes contained in the attribute set for which closure is being calculated to the result set.

Step-02

- Recursively add the attributes to the result set which can be functionally determined from the attributes already contained in the result set.

Example

- Consider a relation R (A , B , C , D , E , F , G) with the functional dependencies-

A → BC

BC → DE

D → F

CF → G

- Now, let us find the closure of some attributes and attribute sets

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Closure of attribute A

$A^+ = \{ A \}$

$= \{ A , B , C \}$ (Using $A \rightarrow BC$)

$= \{ A , B , C , D , E \}$ (Using $BC \rightarrow DE$)

$= \{ A , B , C , D , E , F \}$ (Using $D \rightarrow F$)

$= \{ A , B , C , D , E , F , G \}$ (Using $CF \rightarrow G$)

Thus,

- **$A^+ = \{ A , B , C , D , E , F , G \}$**

Closure of attribute D

$D^+ = \{ D \}$

$= \{ D , F \}$ (Using $D \rightarrow F$)

- We can not determine any other attribute using attributes D and F contained in the result set. Thus,

$$\mathbf{D^+ = \{ D , F \}}$$

Closure of attribute set {B, C}

$\{ B , C \}^+ = \{ B , C \}$

$= \{ B , C , D , E \}$ (Using $BC \rightarrow DE$)

$= \{ B , C , D , E , F \}$ (Using $D \rightarrow F$)

$= \{ B , C , D , E , F , G \}$ (Using $CF \rightarrow G$)

- Thus,

$\{ B , C \}^+ = \{ B , C , D , E , F , G \}$

Finding the Keys Using Closure

Super Key

- If the closure result of an attribute set contains all the attributes of the relation, then that attribute set is called as a super key of that relation.
- Thus, we can say-
- **“The closure of a super key is the entire relation schema.”**

Candidate Key

- If there exists no subset of an attribute set whose closure contains all the attributes of the relation, then that attribute set is called as a candidate key of that relation.

Problem

Consider the given functional dependencies-

- AB → CD
- AF → D
- DE → F
- C → G
- F → E
- G → A

Which of the following options is false?

A. $\{ CF \}^+ = \{ A , C , D , E , F , G \}$

B. $\{ BG \}^+ = \{ A , B , C , D , G \}$

C. $\{ AF \}^+ = \{ A , C , D , E , F , G \}$

D. $\{ AB \}^+ = \{ A , C , D , F ,G \}$

Answer

- **Option (C) and Option (D)**

Fundamental of
Database Management System
BCAC0005

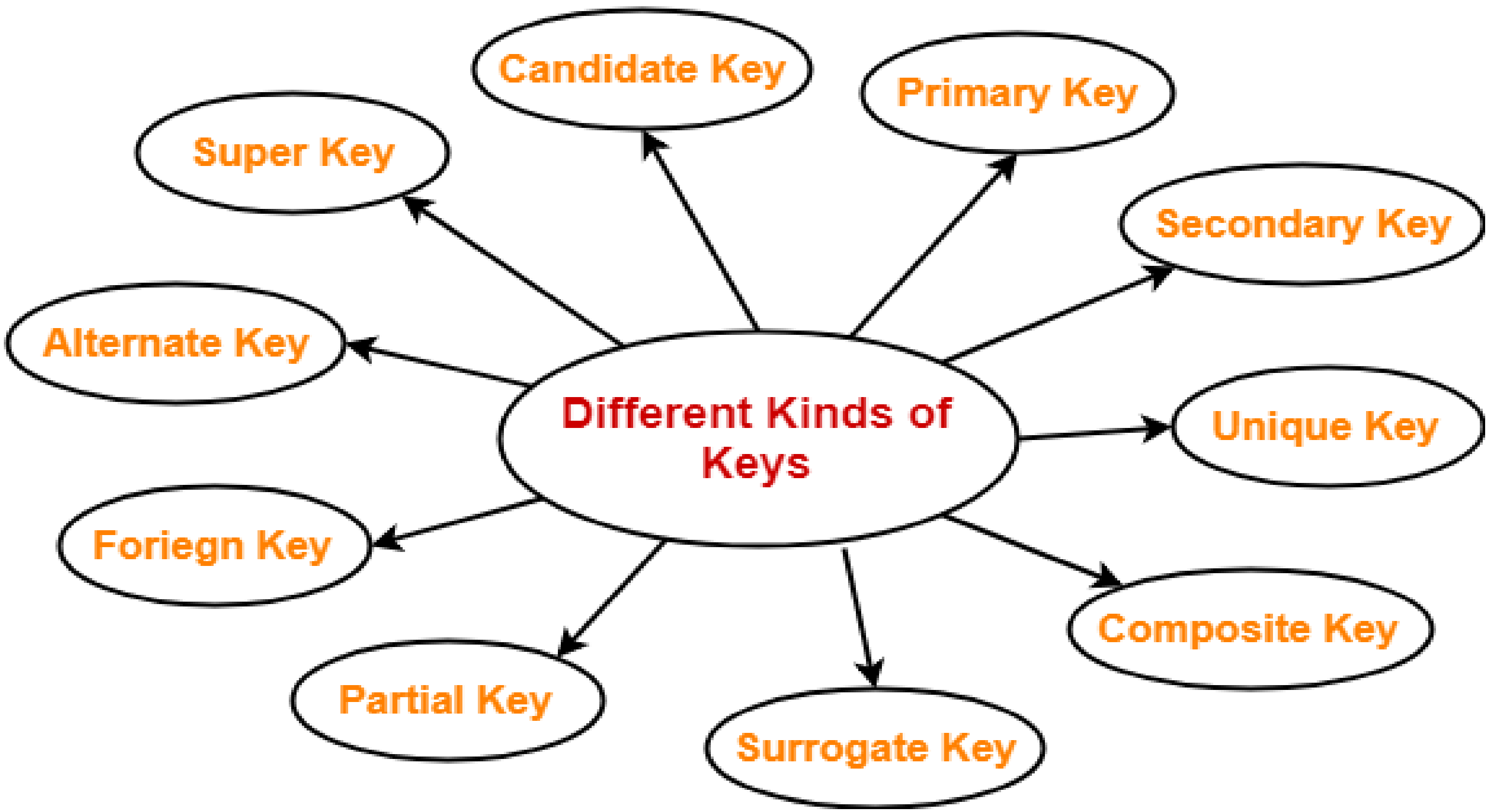
Lecture - 18

Presented by:
Atul Kumar Uttam
Assistant Professor
Computer Engineering & Applications Department,
GLA University, Mathura
atul.uttam@gla.ac.in, +91-8979593001

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Keys in DBMS

- A key is a set of attributes that can identify each tuple uniquely in the given relation.



1. Super Key

- A super key is a set of attributes that can identify each tuple uniquely in the given relation.
- A super key is not restricted to have any specific number of attributes.
- Thus, a super key may consist of any number of attributes.

1. Super Key

EXAMPLE:

Student (class_roll , name , age , address , course , section)

- Given below are the examples of super keys since each set can uniquely identify each student in the Student table-

(class_roll , name , age , address , course , section)

(course , section , class_roll)

(name , address)

1. Super Key

NOTE-

- All the attributes in a super key are definitely sufficient to identify each tuple uniquely in the given relation but all of them may not be necessary.

GLA UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

2. Candidate Key

- A set of minimal attribute(s) that can identify each tuple uniquely in the given relation is called as a candidate key.

2. Candidate Key

Example

Student (class_roll , name , age , address , course , section)

- Given below are the examples of candidate keys since each set consists of minimal attributes required to identify each student uniquely in the Student table-

(course , section , class_roll)

(name , address)

2. Candidate Key

NOTES

- All the attributes in a candidate key are sufficient as well as necessary to identify each tuple uniquely.
- Removing any attribute from the candidate key fails in identifying each tuple uniquely.
- The value of candidate key must always be unique.
- The value of candidate key can never be NULL.
- It is possible to have multiple candidate keys in a relation.
- Those attributes which appears in some candidate key are called as **prime attributes**.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

3. Primary Key

- A primary key is a candidate key that the database designer selects while designing the database.

OR

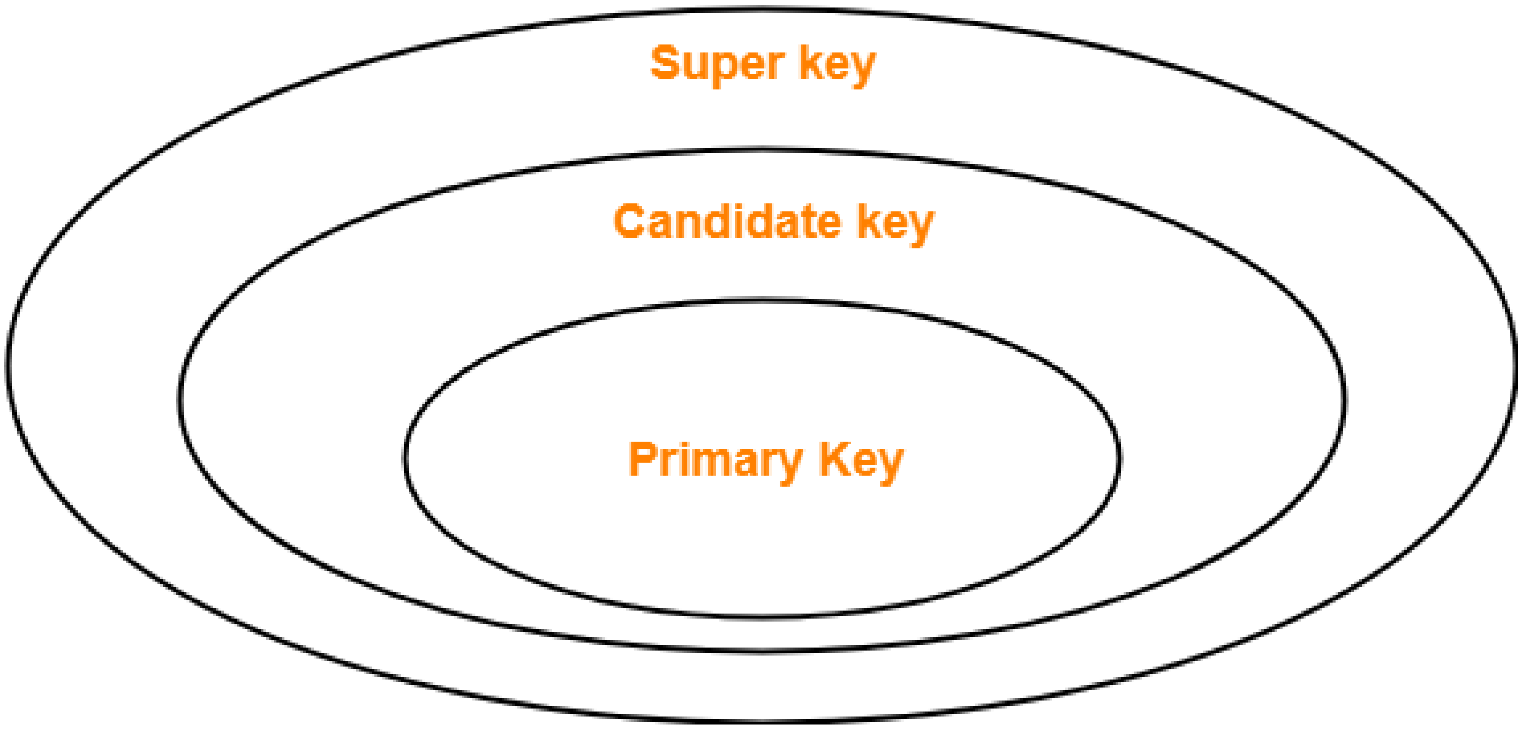
- Candidate key that the database designer implements is called as a primary key.

3. Primary Key

NOTES

- The value of primary key can never be NULL.
- The value of primary key must always be unique.
- The values of primary key can never be changed i.e. no updation is possible.
- The value of primary key must be assigned when inserting a record.
- A relation is allowed to have only one primary key.

GLA UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC



4. Alternate Key

- Candidate keys that are left unimplemented or unused after implementing the primary key are called as alternate keys.

OR

- Unimplemented candidate keys are called as alternate keys.

5. Foreign Key

- An attribute ‘X’ is called as a foreign key to some other attribute ‘Y’ when its values are dependent on the values of attribute ‘Y’.
- The attribute ‘X’ can assume only those values which are assumed by the attribute ‘Y’.
- Here, the relation in which attribute ‘Y’ is present is called as the **referenced relation**.
- The relation in which attribute ‘X’ is present is called as the **referencing relation**.
- The attribute ‘Y’ might be present in the same table or in some other table.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

5. Foreign Key

Teacher (t_no , t_name , t_age , t_dept)

(Referencing Relation)

Foreign Key

Department (dept_no , dept_name)

(Referenced Relation)

Here, t_dept can take only those values which are present in dept_no in Department table since only those departments actually exist.

5. Foreign Key

NOTES

- Foreign key references the primary key of the table.
- Foreign key can take only those values which are present in the primary key of the referenced relation.
- Foreign key may have a name other than that of a primary key.
- Foreign key can take the NULL value.
- There is no restriction on a foreign key to be unique.
- In fact, foreign key is not unique most of the time.
- Referenced relation may also be called as the master table or primary table.
- Referencing relation may also be called as the foreign table.

6. Partial Key

- Partial key is a key using which all the records of the table can not be identified uniquely.
- However, a bunch of related tuples can be selected from the table using the partial key.

6. Partial Key

Dependent (Emp_no, Dependent_name , Relation)

Here, using partial key Emp_no, we can not identify a tuple uniquely but we can select a bunch of tuples from the table.

Emp_no	Dependent_name	Relation
E1	Suman	Mother
E1	Ajay	Father
E2	Vijay	Father
E2	Ankush	Son

7. Composite Key

- A primary key comprising of multiple attributes and not just a single attribute is called as a composite key.

GLA UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

8. Unique Key

- Unique key is a key with the following properties-
  - It is unique for all the records of the table.
  - Once assigned, its value can not be changed i.e. it is non-updatable.
  - It may have a NULL value.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

8. Unique Key

Example

- The best example of unique key is **Adhaar Card Numbers**
- The Adhaar Card Number is unique for all the citizens (tuples) of India (table).
- If it gets lost and another duplicate copy is issued, then the duplicate copy always has the same number as before.
- Thus, it is non-updatable.
- Few citizens may not have got their Adhaar cards, so for them its value is NULL.

9. Surrogate Key

- Surrogate key is a key with the following properties-
- It is unique for all the records of the table.
- It is updatable.
- It can not be NULL i.e. it must have some value.

Example

- Mobile Number of students in a class where every student owns a mobile phone.

10. Secondary Key

- Secondary key is required for the indexing purpose for better and faster searching.

Finding Candidate Keys

- A set of minimal attribute(s) that can identify each tuple uniquely in the given relation is called as a candidate key.

OR

- A minimal super key is called as a candidate key.

Finding Candidate Key

Step-01

- Determine all essential attributes of the given relation.

- Essential attributes are those attributes which are not present on RHS of any functional dependency.
- Essential attributes are always a part of every candidate key.
- This is because they can not be determined by other attributes.

Finding Candidate Key

Example

- Let R(A, B, C, D, E, F) be a relation scheme with the following functional dependencies

$A \rightarrow B$

$C \rightarrow D$

$D \rightarrow E$

- Here, the attributes which are not present on RHS of any functional dependency are A, C and F.
- So, essential attributes are- **A, C and F**.

Finding Candidate Key

Step-02

- The remaining attributes of the relation are non-essential attributes.
- This is because they can be determined by using essential attributes.

- Now, following two cases are possible

Case-01

- If all essential attributes together can determine all remaining non-essential attributes, then-
  - The combination of essential attributes is the candidate key.
  - It is the only possible candidate key.

Case-02

- If all essential attributes together can not determine all remaining non-essential attributes, then-
  - The set of essential attributes and some non-essential attributes will be the candidate key(s).
  - In this case, multiple candidate keys are possible.
  - To find the candidate keys, we check different combinations of essential and non-essential attributes.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

FINDING CANDIDATE KEYS
Problem-01

- Let R = (A, B, C, D, E, F) be a relation scheme with the following dependencies-

C → F

E → A

EC → D

A → B

Which of the following is a key for R?

A. CD

B. EC

C. AE

D. AC

Solution

$\{ CE \}^+$

$= \{ C , E \}$

$= \{ C , E , F \}$ (Using $C \rightarrow F$)

$= \{ A , C , E , F \}$ (Using $E \rightarrow A$)

$= \{ A , C , D , E , F \}$ (Using $EC \rightarrow D$)

$= \{ A , B , C , D , E , F \}$ (Using $A \rightarrow B$)

- We conclude that CE can determine all the attributes of the given relation.
- So, CE is the only possible candidate key of the relation.

Problem-02

- Let R = (A, B, C, D, E) be a relation scheme with the following dependencies-

$AB \rightarrow C$

$C \rightarrow D$

$B \rightarrow E$

- Find the candidate keys and super keys.

Solution

$\{ AB \}^+$

$= \{ A , B \}$

$= \{ A , B , C \}$ (Using $AB \rightarrow C$)

$= \{ A , B , C , D \}$ (Using $C \rightarrow D$)

$= \{ A , B , C , D , E \}$ (Using $B \rightarrow E$)

Hence AB is the candidate key.

Any combination along with AB will be superkey.

Problem-03

- Consider the relation scheme R(E, F, G, H, I, J, K, L, M, N) and the set of functional dependencies-

{ E, F } → { G }

{ F } → { I , J }

{ E, H } → { K, L }

{ K } → { M }

{ L } → { N }

What is the key for R?

A. { E, F }

B. { E, F, H }

C. { E, F, H, K, L }

D. { E }

Solution

$\{ EFH \}^+$

$= \{ E , F , H \}$

$= \{ E , F , G , H \}$ (Using $EF \rightarrow G$)

$= \{ E , F , G , H , I , J \}$ (Using $F \rightarrow IJ$)

$= \{ E , F , G , H , I , J , K , L \}$ (Using $EH \rightarrow KL$)

$= \{ E , F , G , H , I , J , K , L , M \}$ (Using $K \rightarrow M$)

$= \{ E , F , G , H , I , J , K , L , M , N \}$ (Using $L \rightarrow N$)

Fundamental of Database Management System BCAC0005

Lecture - 19

Presented by:

Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,
GLA University, Mathura

atul.uttam@gla.ac.in, +91-8979593001

Decomposition of a Relation

- The process dividing a single relation into two or more sub relations is called as decomposition of a relation.

Properties of Decomposition

1. **Lossless decomposition**
2. **Dependency preserving decomposition**

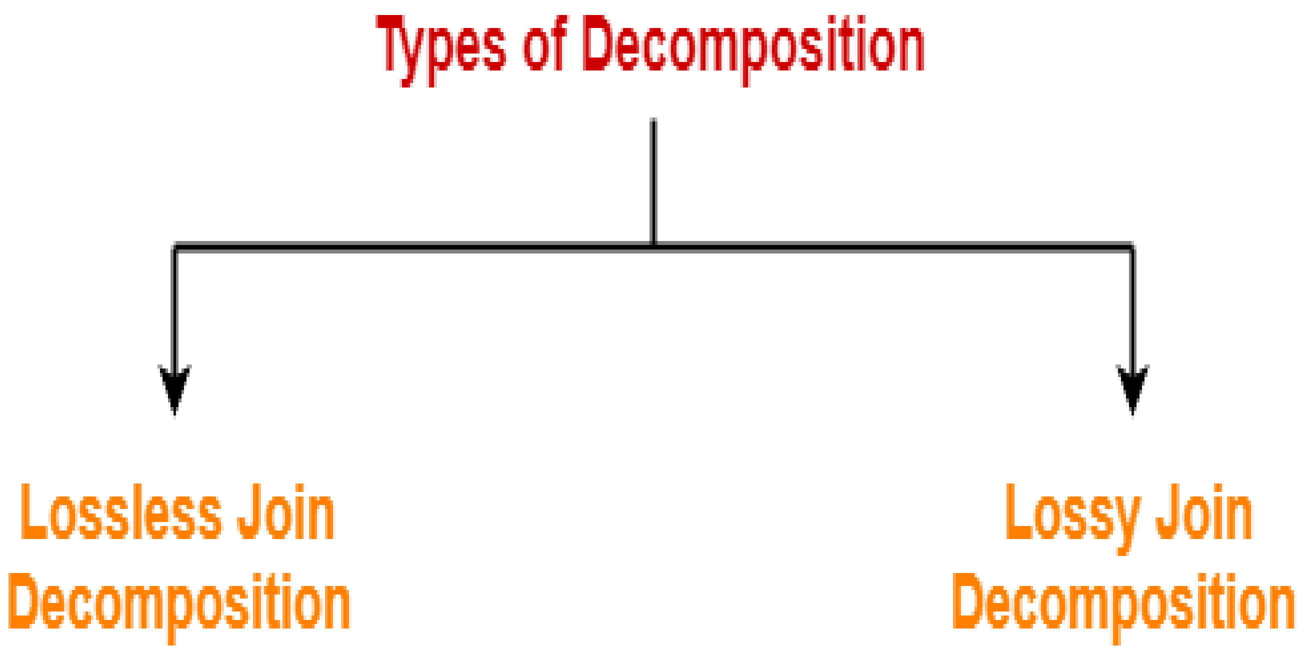
1. Lossless decomposition

- Lossless decomposition ensures
  - No information is lost from the original relation during decomposition.
  - When the sub relations are joined back, the same relation is obtained that was decomposed.
  - Every decomposition must always be lossless.

2. Dependency Preservation

- Dependency preservation ensures-
  - None of the functional dependencies that holds on the original relation are lost.
  - The sub relations still hold or satisfy the functional dependencies of the original relation.

Types of Decomposition



1. Lossless Join Decomposition

- Consider there is a relation R which is decomposed into sub relations R_1 , R_2 , …. , $R_{n.}$
- This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.
- For lossless join decomposition, we always have

$$\mathbf{R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots \bowtie R_n = R}$$

where ⋈ is a natural join operator

Example

R(A, B, C) is decomposed into:

R_1(A , B)
and
R_2(B , C)

Now, let us check whether this decomposition is lossless or not.

A	B	C
1	2	1
2	5	3
3	3	3

R (A , B , C)

R1 (A , B)

A	B
1	2
2	5
3	3

R2 (B , C)

B	C
2	1
5	3
3	3

For lossless decomposition, we must have:

$R_1 \bowtie R_2 = R$

Now, if we perform the natural join ($\bowtie$) of the sub relations R_1 and R_2 , we get

A	B	C
1	2	1
2	5	3
3	3	3

This relation is same as the original relation R.
Thus, we conclude that the above decomposition is lossless join decomposition.

2. Lossy Join Decomposition

- Consider there is a relation R which is decomposed into sub relations R_1 , R_2 , …. , R_n.
- This decomposition is called lossy join decomposition when the join of the sub relations does not result in the same relation R that was decomposed.
- The natural join of the sub relations is always found to have some extraneous tuples.
- For lossy join decomposition, we always have

$$\mathbf{R_1 \bowtie R_2 \bowtie R_3 \ldots\ldots. \bowtie R_n \supset R}$$

where ⋈ is a natural join operator

$R_1(A, B)$

$R_2(B, C)$

$R_1 \bowtie R_2 \supset R$

R(A , B , C)

A	B	C
1	2	1
2	5	3
3	3	3

R (A , B , C)

R1 (A , C)

A	C
1	1
2	3
3	3

R2 (B , C)

B	C
2	1
5	3
3	3

Now, if we perform the natural join ($\bowtie$) of the sub relations R_1 and R_2 we get-

$R_1 \bowtie R_2$

A	B	C
1	2	1
2	5	3
2	3	3
3	5	3
3	3	3

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Determining Whether Decomposition Is Lossless Or Lossy

Condition-01

- Union of both the sub relations must contain all the attributes that are present in the original relation R.

$$\mathbf{R_1 \cup R_2 = R}$$

Condition-02

- Intersection of both the sub relations must not be null.
- In other words, there must be some common attribute which is present in both the sub relations.

$$\mathbf{R1 \cap R2 \neq \emptyset}$$

Condition-03

- Intersection of both the sub relations must be a super key of either R_1 or R_2 or both.

$$\mathbf{R_1 \cap R_2 = \text{Super key of } R_1 \text{ or } R_2}$$

If any of these conditions fail, then the decomposition is lossy.

Problem-01

- Consider a relation schema R (A , B , C , D) with the functional dependencies $A \rightarrow B$ and $C \rightarrow D$. Determine whether the decomposition of R into R_1 (A , B) and R_2 (C , D) is lossless or lossy.

12-B Status from UGC

Solution

Condition-01

- According to condition-01, union of both the sub relations must contain all the attributes of relation R.

$$R_1 (A , B) \cup R_2 (C , D) = R (A , B , C , D)$$

- Clearly, union of the sub relations contain all the attributes of relation R.
- Thus, condition-01 satisfies.

Condition-02

- According to condition-02, intersection of both the sub relations must not be null.

$$R_1 (A , B) \cap R_2 (C , D) = \Phi$$

- Clearly, intersection of the sub relations is null.
- So, condition-02 fails.
- Thus, we conclude that the decomposition is lossy.

Problem-02

- Consider a relation schema R (A , B , C , D) with the following functional dependencies-

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow B$

- Determine whether the decomposition of R into R_1 (A , B) , R_2 (B , C) and R_3 (B , D) is lossless or lossy.

Solution

Condition 1

- R_1 (A , B) ∪ R_2 (B , C) ∪ R_3 (B , D) = R(A,B,C,D)

Condition 2

- **R1 ∩ R2 ≠ Ø** True
- **(R1 U R2) ∩ R3 ≠ Ø** True

Condition 3

- **$R_1 \cap R_2 = \{B\}^+ = \{BCD\}$ super key of R2**
- **$(R_1 \cup R_2) \cap R_3 = \{B\}^+ = \{BCD\}$ super key of R3**
- **Hence lossless decomposition**

Fundamental of
Database Management System
BCAC0005

Lecture - 20

Presented by:
Atul Kumar Uttam
Assistant Professor
Computer Engineering & Applications Department,
GLA University, Mathura
atul.uttam@gla.ac.in, +91-8979593001

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Normalization in DBMS

- Reducing the redundancies
- Ensuring the integrity of data through lossless decomposition
- Normalization is done through normal forms.

Normal Forms in DBMS
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (2NF)
- Boyce-Codd Normal Form (BCNF)

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

First Normal Form

- A given relation is called in First Normal Form (1NF)
  - if each cell of the table contains only an atomic value.

OR

  - if the attribute of every tuple is either single valued or a null value.

First Normal Form

Example

Student_id	Name	Subjects
100	Akshay	Computer Networks, Designing
101	Aman	Database Management System
102	Anjali	Automata, Compiler Design

Relation is not in 1NF

First Normal Form

Relation is in 1NF

Student_id	Name	Subjects
100	Akshay	Computer Networks
100	Akshay	Designing
101	Aman	Database Management System
102	Anjali	Automata
102	Anjali	Compiler Design

First Normal Form

NOTE

- By default, every relation is in 1NF.
- This is because formal definition of a relation states that value of all the attributes must be atomic.

Second Normal Form

- A given relation is called in Second Normal Form (2NF) if and only if-
  - Relation already exists in 1NF.
  - No partial dependency exists in the relation.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Second Normal Form

<u>Partial Dependency</u>

- A partial dependency is a dependency where a part of the candidate key determines non-prime attribute(s).

- In other words,

A → B is called a partial dependency if and only if-
  - A is a subset of some candidate key
  - B is a non-prime attribute.
- If any one condition fails, then it will not be a partial dependency.

Second Normal Form

Example

- Consider a relation- **R (V , W , X , Y , Z)** with functional dependencies-

  VW → XY

  Y → V

  WX → YZ

- The possible candidate keys for this relation are- **VW , WX , WY**
- Prime attributes = { V , W , X , Y }
- Non-prime attributes = { Z }
- Now, if we observe the given dependencies-
- There is no partial dependency.
- Thus, we conclude that the given relation is in 2NF.

Consider a relation- **R (V , W , X , Y , Z)** with functional dependencies-

$$\mathbf{VW \rightarrow XY}$$

$$\mathbf{Y \rightarrow V}$$

$$\mathbf{WX \rightarrow YZ}$$

Third Normal Form

- A given relation is called in Third Normal Form (3NF) if and only if-
  - Relation already exists in 2NF.
  - No transitive dependency exists for non-prime attributes.

If A->B and B->C are two FDs then A->C is called transitive dependency.

Where A is a prime attribute, B & C are Non Prime attribute

Third Normal Form

- For every non-trivial function dependency X –> Y:
  - X is a super key.
  - Y is a prime attribute (each element of Y is part of some candidate key).

Third Normal Form

Example

- Consider a relation- R (A , B , C , D , E) with functional dependencies-

A → BC

CD → E

B → D

E → A

- The possible candidate keys for this relation are-

  A , E , CD , BC

- Prime attributes = { A , B , C , D , E }
- There are no non-prime attributes
- It is clear that there are no non-prime attributes in the relation.
- Thus, we conclude that the given relation is in 3NF.

Boyce-Codd Normal Form

- A given relation is called in BCNF if and only if-
  - Relation already exists in 3NF.
  - For each non-trivial functional dependency A → B, A is a super key of the relation.

Boyce-Codd Normal Form

Example

- Consider a relation- R (A , B , C) with the functional dependencies-

A → B

B → C

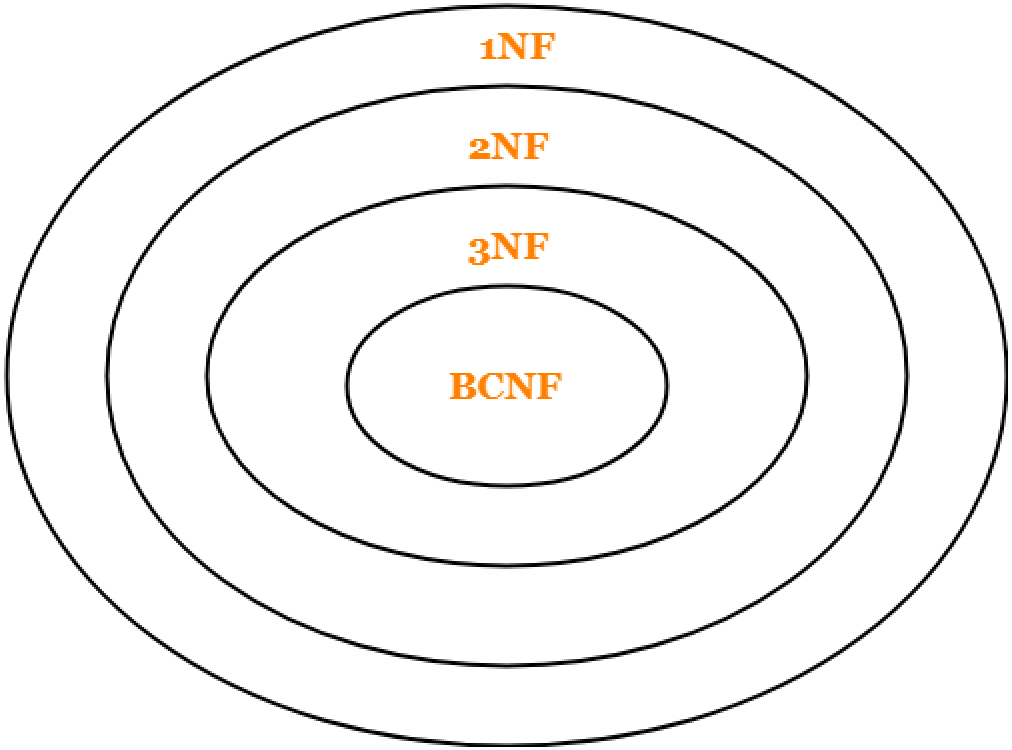
C → A

- The possible candidate keys for this relation are-

A , B , C

- All RHS are superkey hence relation R is in BCNF.

Normal Form Summary



Question 1

- Given a relation R(A, B, C, D) and Functional Dependency set FD = { AB → CD, B → C }, determine whether the given R is in 2NF? If not convert it into 2 NF.

R(A, B, C, D)

FD = { AB → CD, B → C }

R (A, B, C, D)

$\{AB\}^+ = \{ABCD\}$

hence **AB is Candidate Key**

Prime Attribute: A,B

Non Prime Attribute: C,D

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key

B → C is Partial dependency, hence relation R is not in 2NF

Convert the table R(A, B, C, D) in 2NF:

- Since **FD: B → C,** our table was not in 2NF, let's decompose the table

R1(B, C)

- Since the key is AB, and from FD {AB → CD}, we can create R2(A, B, C, D) but this will again have a problem of partial dependency B → C, hence R2(A, B, D).
- Finally, the decomposed table which is in 2NF

a) R1(B, C)

b) R2(A, B, D)

Question 2

- **Given a relation R(P, Q, R, S, T) and Functional Dependency set FD = { PQ → R, S → T }, determine whether the given R is in 2NF? If not convert it into 2 NF.**

R(P, Q, R, S, T)

{ PQ → R, S → T }

R (P, Q, R, S, T)

$\{PQS\}^+ = \{PQRST\}$

PQ → R and S → T, Partial functional Dependency
hence **R(P, Q, R, S, T) is not in 2NF**

Convert the table R(P, Q, R, S, T) in 2NF:

- Since due to FD: PQ → R and S → T, our table was not in 2NF, let's decompose the table
- **R1(P, Q, R)** (Now in table R1 FD: PQ → R is Full F D, hence R1 is in 2NF)
- **R2(S, T)** (Now in table R2 FD: S → T is Full F D, hence R2 is in 2NF)
- And create one table for the key, since the key is PQS.
- **R3(P, Q, S)**

- **Finally, the decomposed tables which is in 2NF are:**

a) R1(P, Q, R)

b) R2(S, T)

c) R3(P, Q, S)

Question 3

- **Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, PS → VW, QS → TU, P → X, W → Y }, determine whether the given R is in 2NF? If not convert it into 2 NF.**

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

R(P, Q, R, S, T, U, V, W, X, Y)

Functional Dependency set FD =

{ PQ → R, PS → VW, QS → TU, P → X, W → Y }

R (P, Q, R, S, T, U, V, W, X, Y)

$\{PQS\}^+$ = {PQRSTUVWXY}

prime attribute(part of candidate key) are {P, Q, S}

non-prime attribute are {R, T, U, V, W, X ,Y}

PQ → R, PS → VW, QS → TU, P → X are Partial FD

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Convert the table R(P, Q, R, S, T, U, V, W, X, Y) in 2NF:

- Since due to FD: PQ → R, PS → VW, QS → TU, P → X our table was not in 2NF, let's decompose the table
- **R1 (P, Q, R)** (Now in table R1 FD: PQ → R is Full F D, hence R1 is in 2NF)
- **R2 (P, S, V, W)** (Now in table R2 FD: PS → VW is Full F D, hence R2 is in 2NF)
- **R3 (Q, S, T, U)** (Now in table R3 FD: QS → TU is Full F D, hence R3 is in 2NF)
- **R4 (P, X)** (Now in table R4 FD : P → X is Full F D, hence R4 is in 2NF)
- **R5 (W, Y)** (Now in table R5 FD: W → Y is Full F D, hence R2 is in 2NF)
- And create one table for the key, since the key is PQS.
- **R6 (P, Q, S)**

- Finally, the decomposed tables which is in 2NF are:

R1(P, Q, R)

R2(P, S, V, W)

R3(Q, S, T, U)

R4(P, X)

R5(W, Y)

R6(P, Q, S)

Question 4

- Given a relation R(A, B, C, D, E) and Functional Dependency set FD = { A → B, B → E, C → D}, determine whether the given R is in 2NF? If not convert it into 2 NF.

R(A, B, C, D, E)

FD = { A → B, B → E, C → D}

{AC}$^+$ = {ABCDE}

Prime attribute = A, C
Non-prime attribute = B D E

FD: **A → B, C → D** does not satisfy the definition of 2NF,
Hence because of FD A → B and C → D, the above table R(A, B, C, D, E) is not in 2NF

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

Convert the table R(A, B, C, D, E) in 2NF:

- Since due to FD: A →B and C → D our table was not in 2NF, let's decompose the table
- **R1(A, B, E)** (from FD: A → B and B → E and both are violating 2 NF definition)
- **R2(C, D)** (Now in table R2 FD: C → D is Full F D, hence R2 is in 2NF)
- And create one table for candidate key AC
- **R3 (A, C)**

Finally, the decomposed tables which are in 2NF:

R1(A, B, E)

R2(C, D)

R3(A, C)

- **Question 1:** Given a relation R(X, Y, Z) and Functional Dependency set FD = { X → Y and Y → Z }, determine whether the given R is in 3NF? If not convert it into 3 NF.

R(X, Y, Z)

FD = { X → Y and Y → Z }

R(X, Y, Z)

$\{X\}^+$ = {X, Y, Z}

X is Candidate Key

FD are X → Y and Y → Z

So, we can write X → Z

X → Y → Z

Prime Non Prime Non Prime

Hence the relation is not in 3 NF

- **Now check the above table is in 2 NF.**
- FD: X → Y is in 2NF (as Key is not breaking and its Fully functional dependent)
- FD: Y → Z is also in 2NF(as it does not violate the definition of 2NF)
- **Hence above table R(X, Y, Z) is in 2NF but not in 3NF.**

Convert the table R(X, Y, Z) into 3NF:

- Since due to FD: Y → Z, our table was not in 3NF, let's decompose the table
- FD: Y → Z was creating issue, hence one table R1(Y, Z)
- Create one Table for key X, R2(X, Y), since X → Y
- Hence decomposed tables which are in 3NF are:

R1(X, Y)

R2(Y, Z)

- **Question 2:** Given a relation R(X, Y, Z, W, P) and Functional Dependency set FD = { X → Y, Y → P, and Z → W}, determine whether the given R is in 3NF? If not convert it into 3 NF.

R(X, Y, Z, W, P) and FD = { **X → Y, Y → P**, and **Z → W**}

R(X, Y, Z, W, P)

$\{XZ\}^+ = XZYPW$

XZ is **Candidate Key**

{ **X → Y, Y → P**, and **Z → W**}

X → Y → P

Prime Non Prime Non Prime

Hence the relation is not in 3 NF

Transaction

- Transaction is a set of operations which are all logically related.

OR

- Transaction is a single logical unit of work formed by a set of operations.

Operations in Transaction

1. Read Operation

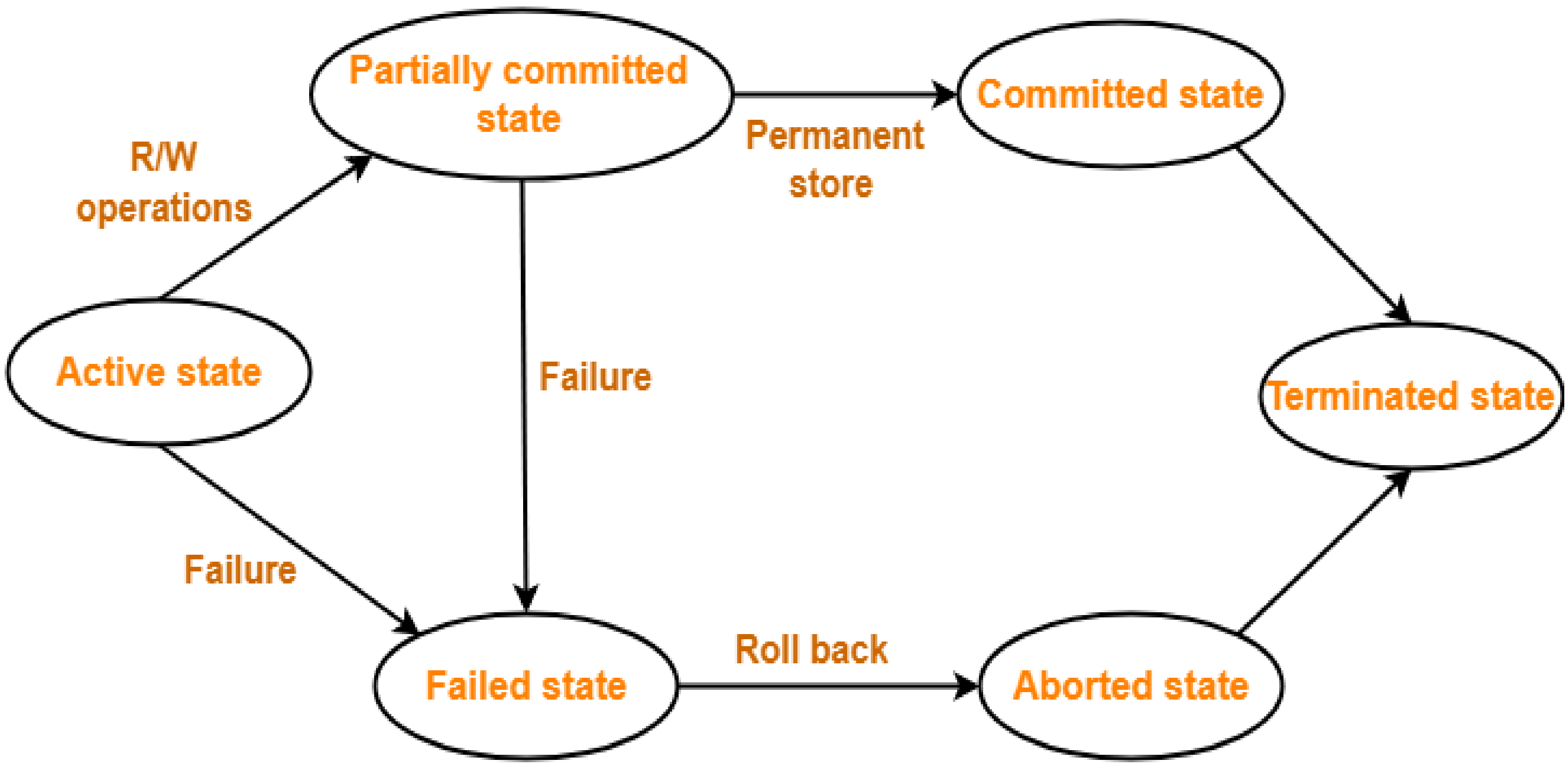
- Read operation reads the data from the database and then stores it in the buffer in main memory.
- For example- **Read(A)** instruction will read the value of A from the database and will store it in the buffer in main memory.

2. Write Operation

- Write operation writes the updated data value back to the database from the buffer.
- For example- **Write(A)** will write the updated value of A from the buffer to the database.

Transaction States

- A transaction goes through many different states throughout its life cycle.
- Transaction states are as follows-
  - Active state
  - Partially committed state
  - Committed state
  - Failed state
  - Aborted state
  - Terminated state



Transaction States in DBMS

1. Active State

- This is the first state in the life cycle of a transaction.
- A transaction is called in an **active state** as long as its instructions are getting executed.
- All the changes made by the transaction now are stored in the buffer in main memory.

2. Partially Committed State

- After the last instruction of transaction has executed, it enters into a **partially committed state**.
- After entering this state, the transaction is considered to be partially committed.
- It is not considered fully committed because all the changes made by the transaction are still stored in the buffer in main memory.

3. Committed State

- After all the changes made by the transaction have been successfully stored into the database, it enters into a **committed state**.
- Now, the transaction is considered to be fully committed.

4. Failed State

- When a transaction is getting executed in the active state or partially committed state and some failure occurs due to which it becomes impossible to continue the execution, it enters into a **failed state**.

5. Aborted State

- After the transaction has failed and entered into a failed state, all the changes made by it have to be undone.
- To undo the changes made by the transaction, it becomes necessary to roll back the transaction.
- After the transaction has rolled back completely, it enters into an **aborted state**.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

6. Terminated State

- This is the last state in the life cycle of a transaction.
- After entering the committed state or aborted state, the transaction finally enters into a **terminated state** where its life cycle finally comes to an end.

ACID Properties OF Transaction

- It is important to ensure that the database remains consistent before and after the transaction.
- To ensure the consistency of database, certain properties are followed by all the transactions occurring in the system.
- These properties are called as **ACID Properties** of a transaction.

A = Atomicity

C = Consistency

I = Isolation

D = Durability

Atomicity

- This property ensures that either the transaction occurs completely or it does not occur at all.
- In other words, it ensures that no transaction occurs partially.
- That is why, it is also referred to as “**All or nothing rule**“.
- It is the responsibility of Transaction Control Manager to ensure atomicity of the transactions.

2. Consistency

- This property ensures that integrity constraints are maintained.
- In other words, it ensures that the database remains consistent before and after the transaction.
- It is the responsibility of DBMS and application programmer to ensure consistency of the database.

3. Isolation

- Transactions can occur simultaneously without causing any inconsistency.
- During execution, each transaction feels as if it is getting executed alone in the system.
- A transaction does not realize that there are other transactions as well getting executed in parallel.
- Changes made by a transaction becomes visible to other transactions only after they are written in the memory.
- The resultant state of the system after executing all the transactions is same as the state that would be achieved if the transactions were executed serially one after the other.
- It is the responsibility of concurrency control manager to ensure isolation for all the transactions.

GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

4. Durability

- This property ensures that all the changes made by a transaction after its successful execution are written successfully to the disk.
- It also ensures that these changes exist permanently and are never lost even if there occurs a failure of any kind.
- It is the responsibility of recovery manager to ensure durability in the database.