GENIUS

[Chapter-3 and 4]

Relational Algebra and Relational database design.

2011 Fall

2a) By using following schemas write relational algebraic expression and SQL.

Employee (Emp-no , Name , Address)
project (pno , pName)
WorkON (EmpNO , pNO)
Part (partNO , partName , QTY_on_Hand)
use (EmpNO , PNO , partNO , Number)

i) List all the employee details who are not working yet.
ii) List partName and Quantity on hand those were used in DBMS project.
iii) List the Name of the projects that are used by employee from 'Kathmandu'.

i) $\pi_{emp\_no, Name, Address}(employee) - \pi_{empno, Name, Address}(employee \bowtie works\_on)$

ii) $\pi_{partName, QTY\_on\_Hand}(\sigma_{pName = 'DBMS'}((project) \bowtie use) \bowtie part)$

iii) $\pi_{pName}(\sigma_{Address = "Kathmandu"}(((Employee) \bowtie Workson) \bowtie project))$

Page No.

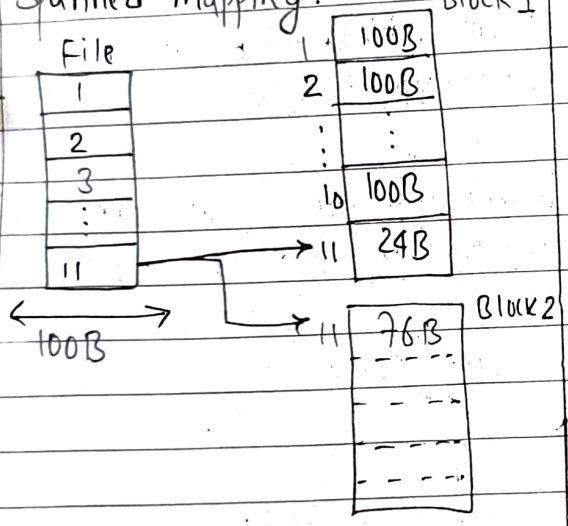
2011- Fall
Short notes on

GENIUS

7)

a) Spanned mapping of record

- In spanned mapping, the record of a file is stored inside the block even if it can only be stored partially and hence, the record is spanned over two blocks giving it the name spanned mapping.



- eg:-

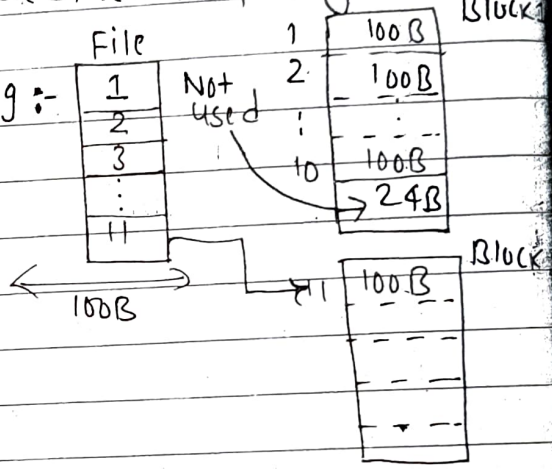
- No wastage of memory (no internal fragmentation).

- The record which has been spanned, while accessing it we would be required to access two blocks and searching time of a record inside a block as the number of blocks on the disk is too large.

- Record do not span across blockboundries.

b) Unspanned mapping of record

- In unspanned mapping unlike spanned strategy, the record of a file is stored inside the block only if it can be stored completely inside it.



- eg:-

- wastage of memory is more internal fragmentation.

- Access time of a record is less. for a single record we need to access only a single block every time and hence, it is faster.

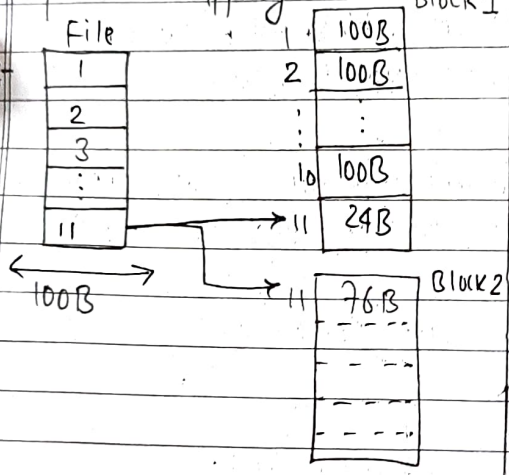
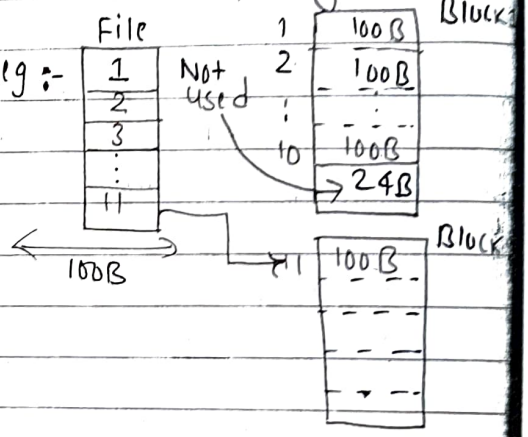
- Records are allowed to span across block boundries.

- Used with variable length

2011 - Fall
Short notes on

GENIUS

7)

a) Spanned mapping record	unspanned mapping record
• In spanned mapping, the record of a file is stored inside the block even if it can only be stored partially and hence, the record is spanned over two blocks giving it the name spanned mapping.	• In unspanned mapping unlike spanned strategy, the record of a file is stored inside the block only if it can be stored completely inside it.
• eg:- 	• eg:- 
• No wastage of memory (no internal fragmentation).	• wastage of memory is more internal fragmentation.
• The record which has been spanned, while accessing it we would be required to access two blocks and searching time of a record inside a block as the number of blocks on the disk is too large.	• Access time of a record is less. for a single record we need to access only a single block every time and hence, it is faster.
• Record do not span across block boundries.	• Records are allowed to span across block boundries. Used with variable length

GENIUS

3a) Write SQL statement for the following queries in reference to relation Emp-time provided.

Eid#	Name	Start_time	End-time
E101	Mangale	10:30	18:30
E102	Malati	8:30	14:30
E103	Fulmaya	9:00	18:00

i) Create the table Eid# as a primary key and insert the values provided.

ii) Display the name of the employee whose name start from letter 'M' and who work for more than seven hours.

iii) Delete the entire contents of the table so that new data can be inserted.

VVVVV Imp ~ 05 questions

Why access to db from a general purpose programming language is required? Explain.

=> SQL provides a powerful declarative query language. However, access to a db from a general-purpose Programming language is required because,

GENIUS

→ Relational instance :- A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

→ 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.

eg:-

Student relation

Name	Roll-no	Phone-no	Address	Age
Ram	14795	9863444041	Noida	24
Shyam	12839	9823048543	Delhi	35
Laxman	33289	9801099821	Ghani	20
mahesh	27857	9861977612	Agra	27
Ganesh	17282	9803540511	Delhi	40

- In the given table, Name, Roll-no, phone-no, address and age are the attributes.
- The instanse of schema student has 5 tuples.
- t3= <Laxman, 33289, 9801099821, Ghani, 20>

Page No.

- SQL in not as powerful as a general-purpose programming language. There are queries that cannot be expressed in SQL, but can be programmed in C, Fortran, pascal, cobol etc SQL, but can be programmed in C, Fortran, pascal, cobol etc
- Non-declarative actions - such as printing a report, interacting with a user, or sending the result to a GUI -- cannot be done from within SQL.

3b) Explain in brief about relational model. write the 12-E.F codd rules formulated for a pure RDBMS.

Relational Model

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

→ Tables- A table has rows and colums, where rows represents records and columns represent the attributes.

→ Tuples :- A single row of a table, which contains a single record for that relation is called a tuple.

→ Relational Schema :- A relational schema describes the relati name (table name), attributes, and their names.

GENIUS

Advantages of Relational Model.

- Simplicity; structural independence.
- Easy to use, query capability.
- Data independence, scalable.

Disadvantages of relational Model.

- Few relational databases have limits on field lengths which can't be exceeded.
- Relational databases can sometimes become complex as the amount of data grows, and the relationships between pieces of data become more complicated.
- Complex relational model systems leads to isolated databases where the information cannot be shared from one system to another.

12-E.F Rule's.

- This rules were developed by Dr. Edgar F. codd (E.F. codd) in 1985, who has vast research knowledge on the Relational Model of database systems. It presents 13 rules for a DB to test the concept of DBMS against his relational model, and if a database follows the rule, it is called a true relational database (RDBMS).

The 13 rules are :-

Page No.

Rule 0 :- The Foundation Rule.

- The database must be in relational form. So that the system can handle the database through its relational capabilities.

Rule 1 :- Information Rule

- A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Rule 2 :- Guaranteed Access Rule.

- Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of PK value, table name and column name.

- Rule 3 :- Systematic Treatment of NULL values.
This rule defines the systematic treatment of null values in a db records. The null value has various meanings in the database like missing the data, no value in a cell, inappropriate information, unknown data and the PK should not be null.

Rule 4 :- Active/Dynamic online catalog based on the relational model.

- It represents the entire logical structure of the descriptive db that must be stored online and is known as a db dictionary. It authorizes users to access the db and implement a similiar query language to access the db.

Rule 5 :- Comprehensive Data SubLanguage Rule

- The RDBMS supports various languages, and if we want to access the db, the language must be the explicit, linear or

Page No.

well-defined syntax, character strings and supports the comprehensive: data defination, view defination, data manipulati integrity constraints, and limit transaction management ope- rations. If the db allows access to the data without an language, it is considered a violation of the db.

Rule 6 :- View updating Rule

- All views table can be theoretically updated and must be practically updated by the db systems.

Rule 7 :- Relational level operation (High-level insert, update and delete) Rule.

- A db system should follow high-level relational operations such as insert, delete, update in each level or a single row. It also supports union, intersection and minus operation in the db system.

Rule 8 :- physical Data independence Rule.

- All stored data in a db or an application must be physically independent to access the db. each data should not depend on other data or an application. If data is updated or the physical structure of the db is changed, it will not show any effect on external applications, it will not show any effect on external applications that are accessing the data from the db.

Rule 9 :- logical data independence rule.

- It is similiar to physical data independence. It means, any changes occurred to the logical level (table db structure), it should not affect the user's view (application). For eg, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application.

GENIUS

Rule 10 :- Integrity independence Rule.

- A db must maintain integrity independence when inserting data into table's cell using the sql query language. All entered values should not be changed or rely on any external factor or application to maintain integrity. It is also helpful in making the database-independent for each front-end application.

Rule 11 :- Distribution independence rule

- The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the db, and these access data should be independent for every user to perform the SQL queries.

Rule 12 :- Non Subversion Rule.

- The non-submersion rule defines RDBMS as a SQL Langage to store and manipulate the data in the database.

- If a system has a low-level or separate language other than SQL to access the database system,

- It should not subvert or bypass integrity to transform data.

Page No.

GENIUS

2011 Spring

2a) What does QBE stands for? What are the different parts of SQL? Enlist different data types used in SQL.

7c) ## QBE 181546 90

- QBE is a graphical query language where we get a user interface and then we fill some required fields to get our proper result.

- It stands for Query By Example and was developed in 1970 by Moshe Zloof at IBM.

- In QBE we don't write queries like sql or other db languages it comes with some blank so we need to just fill that blanks and we get our required result.

- In SQL we will get an error if the query is not correct but in the case of QBE if the query is wrong either we get a wrong query will not be going to execute but we will never get any error.

- eg:- Consider an example where a table 'SAC' presents in the db with Name, Phone_NO and Branch fields. And we want to get the name of SAC-Representative name who belongs to the MCA Branch. If we write the query for this in SQL we have to write it like.

Select Name from SAC where Branch = ('MCA')

- And definitely we will get our correct result. But in the case of QBE, it may be done as like there is a field present and we just need to fill it with "MCA" and then click on SEARCH button (-74) will get our required result.

181546 91

GENIUS

Points about QBE:-

- Supported by most of the db programs.
- It is a Graphical Query Language.
- Created in parallel to SQL developement.

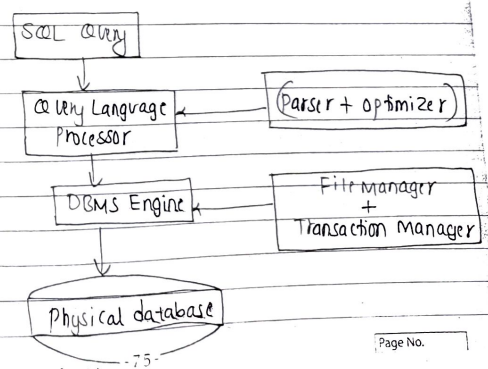
SQL process parts

- When we are executing an SQL command for any RDBMS the system determines the best way to carry out our request and SQL engine figures out how to interpret the task.

- The SQL processing includes.
  - Query dispatcher.
  - Optimization Engines
  - Classic Query Engine
  - SQL Query Engine etc.

- A classic query Engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

Following is a simple diagram the SQL Architecture -



GENIUS

Data types in sql

i) Exact Numeric Data types.

bigint, int, smallint, tinyint, bit, decimal, numeric
money smallmoney

ii) Approximate Numeric Data types
float, real

iii) Date and time Data Types.
datatime, smalldatetime, date, time

iv) character strings Data Types.
Char, varchar, varchar(max), text

v) Unicode character strings Data types.
nchar, nvarchar, nvarchar(max), ntext

vi) Binary data types

binary, varbinary, varbinary(max), image

vii) Misc data types.

sql_varient, time stamp

Uniqueidentifier, cursor

xml,

table.

Page No.

GENIUS

2b) consider a following schema:
customer (cus_id, cus_name, cus_phono)
employee (cus_id, emp_id, emp_name, emp_add)
works (branch_id, salary, cus_id)
branch (branch_id, branch_name.)
write RA

i) select names of all employee

Π_{emp_name} (employee)

ii) Give a salary rise of 5% to all the employees.

works ← $\Pi_{branch\_id, salary * 1.05, cus\_id}$ (works)

iii) Select names of all employees working for "Manang" branch.

Π_{emp_name} (($\sigma_{branch\_name = 'Manang'}$ (branch)) ⋈ works) ⋈ employee)

iv) List all branch names.

Π_{branch_name} (branch)

v) Delete all records from work table.

works ← works − $\Pi_{branch\_id, salary, cus\_id}$ (works)

Page No.

vi) List names and phono of all customers.

$\Pi_{cus\_name, phono}(customer)$

vii) select names of all employees dealing with customer having id "201"

$\Pi_{cus\_name}(\sigma_{cus\_id = "201"}(customer))$

2012-Fall

1c) Explain the distinction among the terms primary key, candidate key, super key and Foreign key with example.

Keys on DBMS

- Keys plays an important role in the RDBMS.
- It is used to uniquely identify any records or row of data from the table. It is also used to establish and identify relationships between tuples.
- Foreg:- In student table, ID is used as a key because it is unique for each student. In person table, passport_no, liscence_no are keys since they are unique for each person.

Student
ID
Name
Address

Person
Name
DOB
passport_no
license_no

Page No.

Types of keys.

1) primary key (PK)

- It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in person table. The key which is most suitable from those lists.
- For each enitity, selection of the PK, is based on requirements and developers.

eg:-

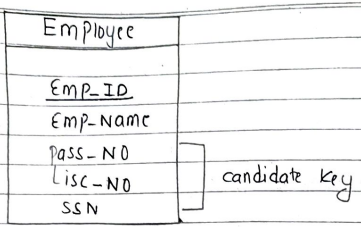
Employee
Emp_ID → PK (primary key)
Emp_name
Lisc_NO
Pass_NO
SSN
Address

2) Candidate Key

- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- The remaining attributes except for PK are considered as a candidate key. The candidate keys are as strong as the primary key.
- Foreg:- In the employee table, id is best suited for the PK, Rest of the attributes like SSN, pass-No, and Lisc_NO etc. are considered as a candidate key.

Page No.

eg:-



3) Super Key

- Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

eg:- In above employee table, for (emp_ID, emp_Name) the name of two employees can be same, but their emp_ID can't be the same. Hence, this combination can also be a key.

- The super key would be emp_ID, (emp_ID, emp_name) etc.

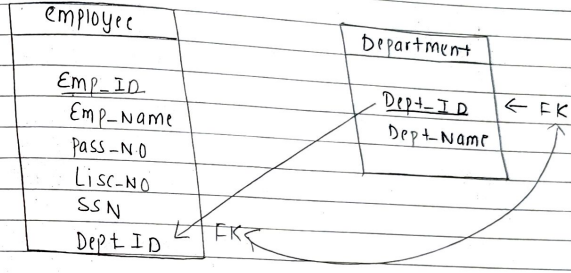
4) Foreign Key (FK)

- Foreign keys are the column of the table which is used to point the PK of another table.

- In company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.

- We add the PK of the Department table, Department ID as a new attribute in the employee table.

- Now, in the employee table, Department_ID is the FK, and both the tables are related.



2a) Consider a following db:

Student (sid, name, age)
Has (sid, cid)
college (cid, cname)

Write RA

i) Find average age of student

$$\mathcal{G}_{avg(age)}(student)$$

ii) Display name of student who studies in "QWERT" college.

$$\Pi_{name}\left(\sigma_{cname = "QWERT"}((college) \bowtie Has) \bowtie student)\right)$$

iii) Insert a new student

Student ← Student ∪ {"104", "Thomas", "20"}

iv) Delete record of "ASDFG" college from college relation.

College ← college − ($\sigma_{cname = "ASDFG"}$ (college))

v) Display name of students whose name begin from "S".

2b) Employee (empname, street, city)
Works (empname, cmpname, salary)
Company (cmpname, city)
Manages (empname, cmpname)

Write SQL statement

i) Modify the database so that Amrit now lives in Naxal.

Update employee set city = 'Naxal'
where employee name = 'Amrit'

ii) Delete all tuples in the works relation for employee of xyz corporation

Delete works where cmpname = "xyz corporation"

iii) update works set salary = salary * 1.1 where cmpname = 'ABC'

Increase salary of all employee of ABC company by 10%

iv) Display all company name located at city pokahara or kathmandu from the company tables.

Select cmpname from company where
city = 'pokahara' or city = 'kathmandu'

v) Display all employee name who have salary greater than 5000 from work table.

Select empname from works where salary > 5000.

2012-spring

2a) Consider a db:-
Customer (cust#, cname, city)
order (order#, odate, cust#, ord_amt)
order_item (order#, Item#, qty)
Item (Item# unit_price)
Shipment (order#, warehouse#, ship_date)
Warehouse (warehouse#, city

Write RA.

GENIUS

i) List the order# and ship date for all orders shipped from warehouse number "W2".

$\pi_{order\#, shipdate}(\sigma_{warehouse\# = 'W2'}(Shipment))$

ii) List the warehouse information for which the customer named 'Jose Lopez' was supplied his orders.

$\pi_{warehouse\#, city}(\sigma_{cname = 'Jose\ Lopez'}(customer) \bowtie order \bowtie shipment \bowtie warehouse)$

iii) List the orders that were not shipped within 30 days.

iv) List the order # for orders that were shipped from all warehouses that the company has in network

$\Rightarrow$

$\pi_{order\#}(\sigma_{city = 'newyork'}(warehase) \bowtie shipment)$

iii) $\pi_{order\#}($

Best ans

iii) Timely_shipped $\leftarrow$ ship_date $\leq$ odate + 30 (order * shipment) Result $\leftarrow \pi_{order\#}$ (order $- \pi_{order\#}$ (Timely_shipped))

Page No.

GENIUS

Best Answer for this is below:-

i) SHIP-W2 $\leftarrow$ warehouse# = 'W2' (SHIPMENT)

RESULT $\leftarrow \pi_{order\#, ship\_date}$ (SHIP-W2)

ii) ORDERS_JL_ORDER* $\leftarrow$ cname = 'Jose Lopez' (customer)

Result $\leftarrow \pi_{order\#, warehouse\#}$ (Orders_JL* SHIPMENT)

iii) Timely_shipped $\leftarrow$ ship_data $\leq$ odate + 30 (order * Shipment)

Resut $\leftarrow \pi_{order\#}$ (order $- \pi_{order\#}$ (Timely_shipped))

iv) New_York $\leftarrow \pi_{warehouse\#}(\sigma_{city = 'new\ york'}(warehase))$

Result $\leftarrow \pi_{order\#, warehouse\#}(shipment) \div$ New_YORK.

Page No.

GENIUS

3a) consider a relations:

Employee (empID, FirstName, LastName, address, DOB, Sex, position, dept NO)

Departement (dept NO, deptName, ~~dept NO~~), mgr, emailID

Project (projNO, projName, dept NO)

Work on (empID, projNO, hours worked)

Write SQL

i) List the name and address of all employees who work for the IT department.

Select FirstName, LastName, address From employee ~~where~~ innerjoin Departement on employee.dept NO = Departement.dept NO

where Departement.deptName = "IT".

ii) List the total hours worked by each employee. arranged in order of departement number and within departement, alphabetically by employee surname

VVVI

Page No.

From Here.

2013 - Fall (only Theory)

GENIUS

1b) Explain the structure of RDBMS [5 marks]

- Relational database system has a simple logical structure with sound theoretical foundation. The relational model is based on the core concept of relation.
- In the relational model, all data is logically structured within relations (also called table). Informally a relation may be viewed as a named two-dimensional table representing an entity set.
- A relation has a fixed number of named columns (or attributes) and a variable number of rows (or tuples).
- Each tuples represents an instance of the entity set and each attributes contains a single value of some recorded property for the particular instance.
- All members of the entity set have the same attributes. The number of tuples is called cardinality.
- This allows a high degree of data independence.

Advantages of RDBMS

- It is easy to use, secured in nature, better data integrity
- It limits redundancy and replication of data, provides multiple interfaces.
- Data Manipulation can be done, provides backup and recovery procedures.
- Multiple users can access the database which is not possible in DBMS and provides physical data independence.

Page No.

Disadvantages of RDBMS.

- Software is expensive and it is difficult to recover the lost data.
- complex software refers to expensive hardware and hence increases overall cost to avail the RDBMS service.
- It requires human resources to implement. certain applications are slow in processing.

2a) What do you mean by relational algebra and relational algebric operators? Explain the basic operators with examples. Compare and contrast relational algebra and relational calculus. Explain DDL and DML with eg. What is query language also explain the importance in DBMS. Explain the need of stored procedure and it's application. What is joining in DBMS? Explain different types of join in SQL. Explain TRC and DRC. Differences between join and subquery.

- Define stored procedure. How it is created?
- How does "GROUP BY" claus work? what is differences between WHERE and Having clause? Explain each with examples.
- Describe the basic structure of SQL queries. considering at least two relations, write SQL for illustrating different types of set operation.
- Differences between SQL and MySQL. why access to db from a general purpose programming language is required? Explain.

Page No.

181546 14

Relational Algebra

2a)

- Relational algebra is a procedural query language.
- It gives a step by step process to obtain the result of the query.
- It uses operators to perform queries.

Relational algebric operators

- Relational set operators uses relational algebra to manipulate contents in a database.
- It helps to test the relationship between two entities.
- It is the mathematical symbol which accept relations as input and yield relations as their output.
- Operators can be either binary or unary.

Basic operators in relational algebra.

1) Selection operations:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by (σ) and Notation as: $\sigma_p(r)$ where 'σ' is used for selection prediction, 'r' is used for relation 'p' is used as a propositional logic formula which may use connectors like: And or and Not. These relations can use relational operators like $=, \neq, \geq, <, >, \leq$.

Page No.

GENIUS

eg:- $\sigma_{subject = "database"}(BOOKS)$

output:- It selects tuples from books where subject is "database".

ii) Projection operations.

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by π. and Noted as $\pi_{A_1, A_2, A_n}(r)$ where, A_1, A_2, A_n is the attribute name of relation r.

eg:- $\pi_{sub, author}(BOOKS)$

output:- It selects and projects columns named as subject & author from the relation books.

iii) Union operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R&S.
- It eliminates the duplicate tuples. It is denoted by $\cup$.

Notation: $R \cup S$

It holds the following conditions:

- R and S must have the attribute of same number.
- Duplicate tuples are eliminated automatically.

eg:- $\pi_{author}(BOOKS) \cup \pi_{author}(ARTICLES)$

output:- Project the names of the authors who have either written a book or an article or both.

Page No.

GENIUS

iv) Set Intersection.

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection(∩). Notation as R ∩ S.
- eg:- using the above Depositor table and BORRow table.

$\pi_{customer\_name}(Borrow) \cap \pi_{customer\_name}(Depositor)$

- output:- Project the name of customer who have both Borrow and Depositor.

v) Set Difference:-

- Suppose there are two tuples R and S. It contains all tuples that are in R but not in S.
- It is denoted by intersection minus(−). Notation as R − S.

eg:-

$\pi_{author}(BOOKS) - \pi_{author}(articles)$

output:- Provides the name of authors who have written books but not articles.

vi) The cartesian product.

- The cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X. Notation as:- $E \times D = \{qt \mid q \in E \text{ and } t \in D\}$

eg:-

$\sigma_{author = 'JK'}(BOOKS \times Articles)$

output:- yields a relation, which shows all the books and articles written by 'JK'.

Page No.

vii) Rename operations

- The rename operation is used to rename the output relation.
- It is denoted by rho(ρ)

eg:-

ρ(student1, student)

We can rename operator to student to student1.

Relational Calculus.

- It is a non-procedural query language, that is, it tells what to do but never explains how to do it.

- It exists in two forms -

i) Tuple Relational calculus (TRC)

- Filtering variable ranges over tuples.

Notation - {T | condition}

Returns all tuples T that satisfies a condition.

For eg:-

{T.name | Author(T) AND T.article = 'db'}

output- Returns tuples with 'name' from Author who have written article on 'db'.

- TRC can be quantified. we can use Existential (∃) and universal quantifiers (∀).

For eg:- {R | ∃T ∈ Authors (T.article = 'db' AND R.name = T.name)}

output- The above query will yield the same result as the previous one.

Page No.

ii) Domain Relational Calculus (DRC).

- In DRC, the filtering variable uses the domain of attributes instead of entire tuples values (as done in TRC, mentioned above). Notation as:- {⟨a1, a2, a3, ... an⟩ | p(a1, a2, a3, ... an)}

where a1, a2 are attributes and 'p' stands for formulae built by inner attributes.

For eg:-

{⟨article, page, subject⟩ | ∈ Tutorial ∧ subject = 'db'}

output- Yields Article, page and subject from the relation Tutorial, where subject is db.

- Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

- The expression power of Tuple Relation Calculus and Domain Relation calculus is equivalent to Relational algebra.

Relational algebra VS Relational calculus.

Relational Algebra	Relational calculus
• Relational Algebra is a procedural query language.	• Relational calculus is Declarative query language.

Page No.

• Relational Algebra states how to obtain the result.	• Relational calculus states what result we have to obtain.
• Relational Algebra describes the order in which operations have to be performed.	• Relational calculus does not specify the order of operations.
• Relational Algebra is not domain dependent.	• Relational calculus can be domain dependent.
• It is close to a programming language.	• It is close to the natural language.
• We specify the sequence of operators operations to perform a particular request.	• We specify the only what is required without bothering about the sequence of operations to perform the request.
• It is prescriptive or rigid in nature ie. It describes steps to perform a given task.	• It is descriptive or straightforward in nature ie. describe desired result.
• It is used as a vehicle for implementation of Relational Calculus.	• Relational calculus queries are converted into equivalent required algebra format by starting codd's reduction algorithm and then it is implemented with the help of relational algebra operators.

Page No.

DDL vs DML

⇒ DDL

- DDL is Data Defination Language which is used to define data structures.
- It is used to create db schema and can be used to define some constraints as well.
- It basically defines the column (Attributes) of the table.
- It does not have any further classification.
- DDL does not use WHERE clause in its statements.
- Basic comands present in DDL are Create, Drop, Rename, Alter etc.

→ The following reasons to use DDL commands.

- It allows us to store shared data in a db.
- It improved integrity due to the data independence feature.
- It will enable multiple users to work on the same db.
- It improved security efficient data access.

eg:- To create table.

Create table student (roll int, name varchar(20)

Page No.

eg:- Alter is used to add, delete or modify columns in a table.

```
Alter Table student
Drop column name
```

To add :-
```
Alter table student
Add email varchar(50)
```

To change data type of column in a table.
```
Alter table student
Alter column name char(10)
```

eg:- Drop table.
```
DRop table student
```

eg:- Truncate table
```
Truncate table student.
```

⇒ DML

- DML is Data Manipulation Language which is used to manipulate data itself.
- It is used to add, retrieve or update the data.
- It adds or update the row of the table. These rows are called tuples.
- It is further classified into procedural and non-procedural DML.
- It while used WHERE clause in its statement.

Page No.

- Basic commands present in DML are insert, update, delete instruction in SQL.

The following are the reasons to use the DML commands.
- It helps user to change the data in a db table.
- It helps users to specify what data is needed.
- It facilitates human interaction with the system.

eg:- Inserting

```
Insert into student values (1, 'ram' 6.1)

Insert into student values (1, 'Thomas'), (2, 'gita')
```

eg:- Select

```
Select * from student where e_add = 'Kathmandu'
```

eg:- update.

```
Update employee set e_add = 'Pharping' where e_id=1
```

eg:- Delete

```
Delete from employee where e_id = 1 .
```

DCL

- It stands for data control language.
- They are used to grant and take back authority from any database user.

Page No.

GENIUS

some commands are :-

a) Grant :- It is used to give user access privileges to a db.
eg :-

```
Grant select, update on my_table To some_user,
        Another_user;
```

b) Revoke :- It is used to take back permissions from the user.

eg :- Revoke select, update on my_table from user1, user2;

TCL

- It stands for Transaction control Language.
- It can be use with DML commands.
- These operations are automatically committed in the db that's why they cannot be used while creating tables or dropping them.

some commands are

a) commit :- commit is use to save all transaction to db.
eg :- Delete from student where AGE = 10; commit;

b) Rollback :- It is used to undo transactions that have not already been saved to the db.
eg :-
```
Delete from student
    where age = 10;
Rollback.
```

c) savepoint :- It is used to roll the transaction back to a certain point without rolling back the entire transaction.
eg :- SYNTAX :- savepoint savepoint_Name;

Page No.

GENIUS

every language with it's importance in DBMS.

- every language is a standard db language which is used to create maintain and retrieve the relational db.

Importance.

- High speed :- using queries, the user can quickly retrieve a large amount of records from a db.
- No coding Needed :- It doesn't require a substantial amount of code to manage the db system.
- well defined standards :- long established are used by the SQL db that are being used by Iso and ANSI.
- Portability :- It can be used in laptop, pcs, even in some mobile phones.
- Interactive language :- SQL is a domain language used to communicate with the db. It is also used to receive answers to the complex questions in seconds.
- Multiple data view :- using this language, the users can make different views of db structure.

Stored procedure

- It is created to perform one or more DML operation on db.

Page No.

- It is a group of sql statements that accepts some input in the form of parameters and performs some task and may or may not returns a value.

SYNTAX:- Creating a procedure

```
create or Replace procedure name (parameters)
  Is variables; Begin
   // statements ; End ;
```

The most important part is parameters. parameters are used to passs values to the procedure the three ways to follows

- IN:- This is the default parameter for the procedure. It always receives the values from calling program.
- OUT:- This parameter always sends the values to the calling program.
- In out :- This parameter performs both the operations. It receives value from the as well as sends values to the calling program.

eg:- Imagine a table named with emp-table stored in db. we are writing a procedure to update a salary of Employee with 100.

```
Create or Replace procedure Inc-SAL (eno In Number,
up-sal out Number)
  Is
```


Page No.

```
Begin
update emp-table set salary = salary +1000 where
  emp-no = eno ;

  Commit ;
salary sal INTO up-sal From emp-table where emp-no =
  eno;
  End ;
```

- Declare a variable to store the value coming out from procdure:-

```
variable v Number;
```

- Execution of the procedure:

```
Execute INC_SAL (1002, :v);
```

- To check the updated salary use select statement:

```
select * from emp-table where emp-no = 1002;
```

- or use print statement

```
print :v
```

→ Advantages:-

- It is faster.
- It is pre-compiled.
- It reduces network traffic.
- It is reusable, better performance.
- It's security is high.

Page No.

→ Disadvantages

- It is difficult to debug.
- Needs expert developer, since difficult to write code.
- It is database dependent.
- It is non-portable.
- It is expensive.

Joins in SQL.

- Joins in DBMS is used to combine rows from two or more tables, based on a related column between them.

- Joins in DBMS is a binary operation which allows us to combine join product and selection in one single statement.

- Types of joins are:-

i) Inner join
   a) Theta join
   b) Equi join
   c) Natural join

ii) Outer join
   a) Left outer join
   b) Right outer join
   c) Full outer join

i) Inner join

- Inner join is used to return rows from both tables which satisfies the given conditions. It is the most widely used join operation and can be considered as a default join-type.

- It returns records that have matching values in both tables.

Page No.

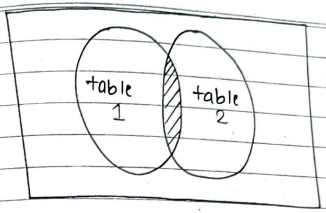


Fig:- Inner join.

Inner join further divided into three subtypes:-

a) Theta join

- It allows you to merge two tables based on the condition represented by theta.

- Theta joins work for all comparison operators. It is denoted by symbol θ. The general case of JOIN operation is called a Theta join.

SYNTAX:- $A \bowtie_\theta B$

eg:- consider the following tables.

Table A

column1	column2
1	1
1	2

Table B

column1	column2
1	1
1	3

$A \bowtie_{A.column2 > B.column2} (B)$ will be

column1	column2
1	2

Page No.

GENIUS

b) Equi join

- Equi join is done when a Theta join uses only the equivalence condition.
- Equi join is the most difficult operation to implement efficiently in an RDBMS, and one reason why RDBMS have essential performance problems.

From example table above :-

$A \bowtie_{A.column2 = B.column2} (B)$ will be,

column1	column2
1	1

c) Natural join

- It does not utilize any of the comparision operators.
- It performs selection forming equality on those attributes which appears in both relations and eliminates the duplicate attributes.
- Here attribute should have the same name and domain.

eg :- consider the following two tables.

Table C

Num	square
2	4
3	9

Table D

Num	cube
2	8
3	18

Now, C ⋈ D then

Num	square	cube
2	4	8
3	9	18

Page No.

GENIUS

ii) outer joins

- An outer joins doesn't require each record tables to have a matching records. In this type of join, the table retains each record even if no other matching record exists.

There are three types of outer joins they are :-

a) Left outer join.

- It Returns all records from the left table, and the matched records from the right table.
- when no matching record found in the table in the table on the right, NULL is returned.

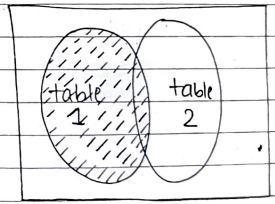


Fig :- Left join

eg :- Table A

Num	square
2	4
3	9
4	16

Table B

Num	cube
2	8
3	18
5	75

then, A ⋈ B will be,

Num	square	cube
2	4	8
3	9	18
4	16	-

Page No.

b) Right outer join

- It returns all records from the right table, and the matched records from the left table.
- when no match found in the table on left, NULL is returned.
- It is just opposite to left join.

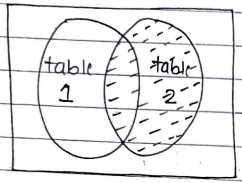


Fig :- Right join

From above table A ⋈ B will be,

Num	cube	square
2	8	4
3	18	9
5	75	—

c) Full outer join.

- It returns all the records when there is a match in either left or right table.

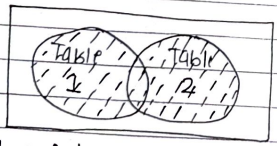


Fig :- Full outer join

From abon eg :- A ⋈ will be

Num	cube	square
2	4	8
3	9	18
4	16	—
5	—	75

Page No.

GENIUS

join vs subquery.

join	subquery
• A join is a query that combines records from an two or more tables.	• It is inner query or nested query embedded within the WHERE clause.
• The select list of the query can select any columns from any of the tables.	• A subquery is a select statement that is embedded in clause of another SQL statement
• Most join queries contains at least one join condition, either in the FROM clause or in the where clause.	• The subquery can be placed in the following sql clauses they are where clause, having clause and From clause.
• eg :- Select cName, cAddress, Ccity, cstate, Cpostalcode, From Customer_T, order_T Where Customer_T. customerID = order_T. customerID AND orderID = 1008;	eg :- select cName, caddress, cstate, Cpostalcode, Ccity From Customer_T where Customer_T. customerID = (Select order_T. CustomerID From order_T where OrderID = 1008);
• They are not easy to read as subqueries.	• They are very easier to read than joins.
• The retrival time of query using joins will be faster than that of subquery.	• The retrival time is slower than the joins.

Page No.

GENIUS

How does Groupby clause work? [2 Marks]

- The GroupBy clause in sql is used to arrange identical data into groups with the help of some functions. ie. If a particular column has same values in different rows then it will arrange these rows in a group.

- Groupby clause is used with the select statement.
- In the query, Groupby clause is placed after the where clause.
- In the query, Groupby clause is placed before order by clause if used any.
- In GroupBy A,B,C and GroupBy C,A,B Both returns same results.

Syntax :-

```
Select column1, function_name (column2)
from table_name
where condition
Group By column1, column2
order By column1, column2;
```

function_name: Name of the function used for example, sum(), AVG().
table_name : Name of the table.
condition: condition used.

Let us create a table

Named: product_Mast

Page No.

GENIUS

Product	Company	Qty	Rate	Cost
Item1	com1	2	10	20
Item2	com2	3	25	75
Item3	com1	2	30	60
Item4	com3	5	10	50
Item5	com2	2	20	40
Item6	com1	3	25	75
Item7	com1	5	30	150
Item8	com1	3	10	30
Item9	com2	2	25	50
Item10	com3	4	30	120

eg:-

```
Select company, count(*)
From product_Mast
Group By company;
```

output :-

com1	5
com2	3
com3	2

- A GroupBy clause can have multiple columns such as :-

```
Select company, qty, count(*)
From product_mast
Group By company, qty;
```

Page No.

ii) Having Clause.

- Having clause is used to specify a search condition for a group or an aggregate.
- Having is used in a Group by clause. If we are not using Group by clause then we can use Having function like Where clause.
- This is where having clause is used to place conditions to decide which group will be the part of final result-set.
- The Having clause selects selects rows after grouping.
- It can only be used with the select statement.
- SYNTAX :-

```
Select  column1, column2
From  table_name
Where conditions
Group By column1, column2
 Having conditions
 order by column1, column2;
```

- Having clause is used with multiple row functions like Sum, count etc.
- It can have aggregate functions.

eg:- from above table; select company, count(*)
From product_Mast
Group By company
Having count(*) >2;

Page No.

- output: com1 5
com2 3

iii) Where Clause

- It is used to filter the records from the table based on the specified condition.
- It can be used without Group By clause.
- Where clause implements in row operations.
- Where clause cannot contain aggregate function. Where clause can be used with select, update, delete statement.
- Where clause is used before Group by clause.
- Where clause is used with single row function like upper, lower etc.
- The where clause selects rows before grouping.
- eg:- Let's consider a table emp_details

name_employee	Address	ID	Phone no
Ram	Ktm	1	9863444041
Hari	BK+	2	9823048543
Sita	Lalitpur	3	9803540511

Page No.

Now, we can use select, update Delete etc. then,

- We write a query to select name and address of employee whose Address is K+m then,

```
select name_employee, Address
    from emp_details
    where Address = 'K+m'
```

Output :-

name_employee	Address
Ram	K+m

iv) order by clause

- The SQL order by clause is used for sorting data in ascending and descending order based on one or more columns.
- controlling the presentation of the columns for the results of the select statement.
- order by clause is not allowed in the Create View statement.
- The order by clause is always placed after the Group by clause in select statement.
- As we now that order by sorts the data either in ascending order or in descending order as specified in the query. so here, sorting the data is an overhead.

Page No.

- syntax :-

```
Select expressions
From tables
Where conditions
Order by expression [ASC|DESC];
```

- If order By A,BC and order by C,A,B than Both returns the same number of records. But now order will be different

- eg :- consider customer table.

ID	Name	Age
1	Ram	32
2	Jsita	25
3	priti	23
4	Hari	25
5	Shyam	27

The following code block has an example, in descending order by Name

SQL
```
Select * from customer
ORder by Name DESC;
```

ID	Name	Age
5	Shyam	27
1	Ram	32
3	priti	23
2	Jsita	25
4	Hari	25

Page No.

GENIUS

Basic structure of SQL

- SQL is based on set and relational operations with certain modifications and enhancements.
- The basic structure of SQL expression consists of three clause. they are :-
- The select clause corresponds to the projection operation of the relational algebra. It is used to list the attributes desired in the result of a query.
- The from clause corresponds to the cartesian - product operations of the relational algebra. It lists the relations to be scanned in the evaluation of the expression.
- The where clause corresponds to the selection predicate of the relational algebra. It consists of a predicate involving attributes of the relations that appear in the from of clause
- A typical SQL query has the form:

$$\text{select } A_1, A_2, \ldots A_n \quad \text{from } r_1, r_2, \ldots r_n \quad \text{where } P$$

Here, A_1, A_2, A_3 represents an attribute
r_1, r_2, r_n is a relation
and P represents predicate.

Page No.

GENIUS

SQL vs MySQL

SQL	MySQL
• It is a programming language used to issue instructions to a relational db management system.	• It is an open source relational db management system.
• We need to learn the language to use it.	• Readily available through download and installations.
• It is fixed and not updatable.	• Regularly updated.
• It is a query Language and it has no support for it's connector.	• It is a relational db that uses SQL and has MySQL Workbench integrated tool.
• The server remains independent of the db.	• The server blocks the db during a data backup session.
• It supports a single storage engine.	• supports multiple and pluggable storage engine.
• It performs operation on data in a db.	• It stores the existing data in an organized manner in its db
• It writes queries for dbs.	It stores, modifies and manages data in a tabular format.
• SQL code and commands are used in various DBMS and RDBMS systems including MySQL.	• MySQL is used as an RDBMS database.

Page No.

GENIUS

2011 - spring

7a) Multiple granularity (Short notes)

- Granularity is the size of data item allowed to lock. It can be defined as hierarchically breaking up the db into blocks which can be locked.
- The multiple Granularity protocol enhances concurrency and reduces lock overhead.
- It maintains the track of what to lock and how to lock.
- It makes easy to decide to lock a data item or to unlock a data item. This type of hierarchy can be graphically represented as a tree.
- There are three additional lock modes with multiple granularity Intention Mode lock.

i) Intention-shared (Is) :- It contains explicit locking at a lower level of the tree but only with shared locks.
ii) Intention-Exclusive (Ix) :- It contains explicit locking at a lower level with exclusive or shared lock.
iii) Shared & Intension-Exclusive :- (SIx) :- In this lock, the node is locked in shared mode, and some node is locked in exclusive mode by the same transaction.

- compatibility matrix with Intention Lock Mode:
The below table describes the compatibility matrix for these lock modes :-

Page No.

GENIUS

	IS	Ix	S	SIx	X
IS	✓	✓	✓	✓	x
Ix	✓	✓	x	x	x
S	✓	x	✓	x	x
SIx	✓	x	x	x	x
X	x	x	x	x	x

Fig:- The multiple Granularity tree Hierarchy

Here, Is = Intention shared X = Exclusive
Ix = Intention Exclusive S = Shared
SIx = Shared & Intention Exclusive

- It Helps locking protocol using the intention lock modes to ensure serializability.
- For eg :- consider a tree which has four levels of codes (nodes).

- The first level or higher level shows the entire database.
- The second level represents a node of type area. The higher level db consists of exactly these areas.
- The area consists of children nodes which are known as files. No file can be present in more than one area.
- Finally, each file contains child nodes known as records. The file has exactly those records that are its child nodes. No records represent in more than one file.
- Hence, the levels of the tree starting from the top level as they are as follows:
i) Db ii) Area iii) File iv) Record.

Page No.

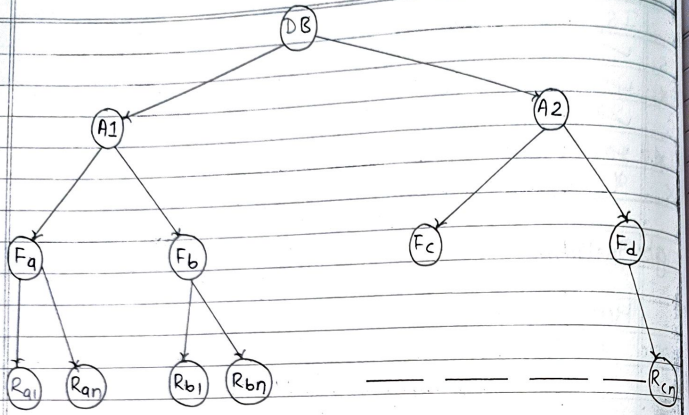


Fig:- Multi Granularity tree Hierarchy.

- In this above example, the highest level shows the entire db. The levels below are file, record, and fields.

Buffer Management in DBMS

- The Buffer Manager is the software layer that is responsible for bringing pages from physical disk to main memory as needed.
- The buffer manages the available main memory by dividing the main memory into the collection of pages, which we called as buffer pool.
- The main memory pages in the buffer pool are also known as frames.

Page No.

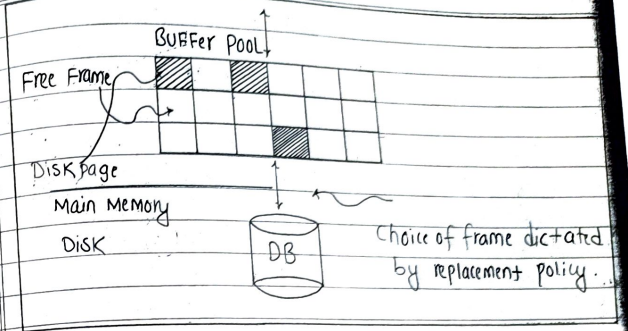


Fig:- Buffer Management in a DBMS

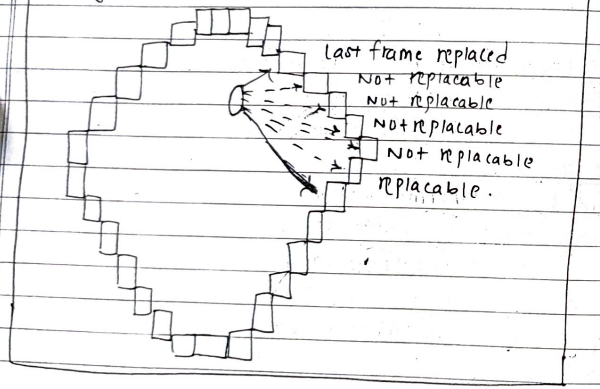
- Data must be in RAM for DBMS to operate on it!
- Buffer Manager hides the fact that not all data is in RAM.
- The goal of buffer manager is to ensure that the data requests made by programs are satisfied by copying data from secondary storage device into buffer.
- If an input operation does not find the requested page in the buffer pool, the the buffer manager (software) will have to do a (physical) transfer the page from the secondary memory (disk) to a free block in buffer pool and then make the requested page placed in the buffer pool, that is available to the program requesting the original input operation.
- In addition: to the buffer pool itself, the buffer manager maintains same block keeping information and two variables for each time frame in the pool; pin-count and dirty.

Page No.

- The number of times the page is requested for each frame - each time the pin-count variable is incremented for that frame (because that page is in this frame).

- For satisfying each request of the user; the pin-counter variable is decremented each time for that Frame. Thus, if a page is requested the pin-count is incremented; if it fulfills the request the pin-count is decremented. In addition to this; if the page has been modified the Boolean variable; dirty is set as 'on'. otherwise off.

Fig:- Buffer pool Frames



Denormalization

- It is the process of attempting to optimize the performance of a db by adding redundant data or by grouping data.

- It is needed when multiple joins in some query can have negative impact on performance.

Page No.

- It is the process of adding pre-computed redudant data to an otherwise normalized relational db to improve read performance of the db.

- It should take place after a satisfactory level of normalization has been taken and that any required constraints and/or rules have been created to deal with the inherent anamolies in design.

- It is used to combine multiple tables into one so that it can be queried quickly.

* pros of Denormalization:-
- Retrieving data is faster since we do fewer joins.
- Queries to retrieve can be simpler (and therefore less likely to have bugs), since we need to look at fewer tables.

* Cons of Denormalization.
- updates and inserts are more expensive.
- It can make update and insert code harder to write.
- Data may be incosistent. Which is the "correct" value for a piece of data.
- Data redundancy necessitates more storage.

- A denormalized db should never be confused by a db that has never been normalized.

- The denormalization is done after normalization for improving the performance of the db.

- This is done to speed up db access speed.

Page No.