25CS305
Database Management Systems

M. NARMADHA,
Assistant Professor,
Computer Science and Engineering

NARSIMHA REDDY ENGINEERING COLLEGE
UGC AUTONOMOUS INSTITUTION
Maisammaguda (V), Kompally - 500100, Secunderabad, Telangana State, India
Accredited by NBA & NAAC with 'A' Grade
Approved by AICTE
Permanently affiliated to JNTUH

UNIT-III

RELATIONAL MODEL

Syllabus

- Relational data model concepts,
- Constraints,
- Relational algebra,
- Relational calculus,
- SQL: DDL, DML, DCL,
- View,
- Index
- Cursors and
- Triggers

What is data model?

A data model is a collection of concepts that can be used to describe the structure of a database.

Introduction

The entire database and more particularly the modern database uses a very important data model called as the Relational Model.

This model was proposed by Codd in the year 1970.

During that time the popular models were only *Network model* and *Hierarchical model*.

CODD's rules for RDBMS

1.Information Rule
2. Guaranteed Access Rule
3. Systematic Treatment of Null Values
4. Dynamic On-line Catalog Based on the Relational Model
5. Comprehensive Data Sublanguage Rule
6. View Updating Rule
7. High-level Insert, Update, and Delete
8. Physical Data Independence
9. Logical Data Independence
10. Integrity Independence
11. Distribution Independence
12. Non subversion Rule

Relation

The relational model represents the database as a collection of relations.

Each relation resembles a simple table, having a set of rows and set of columns.

Each relation consists of a relational schema and relational instance.

Relational Model Concepts

In the formal language terminology,

- **Row is called as- tuple**
- **Column header is called as-attribute**
- **Table is called as-relation**

The data type describing the types of values that can appear in each column is called a domain.

Relational Model Concepts

The number of attributes in a relation is the **degree** of a relation.

The number of tuples in a relation is called the cardinality of that relation.

Relation schema(R) is the description about the structure of any relation.

Relation state(r) is the data in the relation at any moment of time.

INFORMAL DEFINITIONS

RELATION: A table of values

A relation may be thought of as a **set of rows**.

A relation may be thought of as a **set of columns**.

Each row represents a fact that corresponds to a real-world **entity** or **relationship**.

Each row has a value of an item or set of items that uniquely identifies that row in the table.

Each column typically is called by its **column name** or **column header** or **attribute name**.

FORMAL DEFINITIONS

A **Relation** may be defined in multiple ways.

The **Schema** of a Relation: *R* (A1, A2, ...An)

- Relation schema *R* is defined over **attributes** A1, A2, ...An

For Example -

- CUSTOMER (Cust-id, Cust-name, Address, Phone#)

FORMAL DEFINITIONS

A relation may be regarded as a ***set of tuples*** (rows).

A **tuple** is an ordered set of values

Each value is derived from an appropriate domain.

Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four values.

<632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
is a tuple belonging to the CUSTOMER relation.

FORMAL DEFINITIONS

A domain D is a set of atomic values. By atomic we mean that each value in the domain is indivisible as far as the relational model is concerned.

"USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.

"Local_phone_numbers" are the set of 7 digit phone numbers.

Employe_ages: possible ages of employees of a company; each must be a value between 15 and 80 years old.

Academic_department_names: the set of Academic department names such as, computer science, economics, and physics, in a university.

FORMAL DEFINITIONS

A relation R is a subset of the Cartesian product of the domains that define R

- $R \subseteq (dom(A1)) \times (dom(A2)) \times \ldots (dom(An))$
- This actually gives the total number of tuples in the Cartesian product. R is represented as, R (A1, A2, . . ., An).

R: schema of the relation

Relational Integrity Constraints

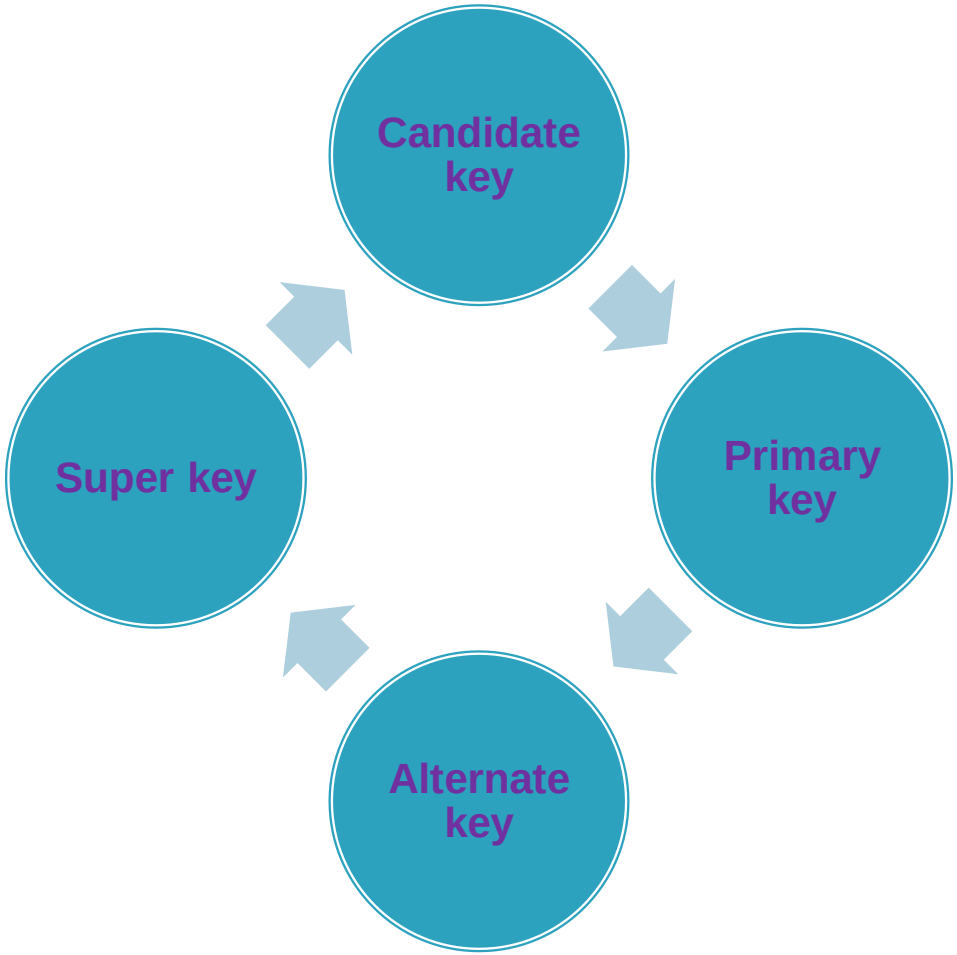
Constraints are the rules applied on the attribute.

Constraints on the relation are the constraints on the attributes of that relation.

There are three main types of constraints:

- **Key constraints**
- **Entity integrity constraints**
- **Referential integrity constraints**

Key Constraints



Alternate & Primary key

Alternate & Primary key is related with candidate key.

In a relation, primary key is a candidate key but only one key is the primary key & the left candidate keys are called alternate key.

AK=CK-PK

Super key

A super key is the superset of any candidate key.

Entity Integrity

The entity integrity constraint states that no primary key value can be null.

This is because the primary key value is used to identify individual tuples in a relation; having null values for the primary key implies that we cannot identify some tuples.

For example, if two or more tuples had null for their primary keys, we might not be able to distinguish them.

Entity Integrity: The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R). This is because primary key values are used to identify the individual tuples.

$t[PK] \neq null$ for any tuple t in r(R)

Referential Integrity

Referential integrity is concerned with foreign keys. Primary key and foreign key relates two tables.

Referential integrity is a property, when satisfied, requires every value of one attribute of a relation to exist as a value of another attribute in a different relation.

Less formally: For referential integrity to hold, any field in a table that is declared a foreign key can contain only values from a parent table's primary key.

Student

Primary key

Roll_no	Enroll	Name	Addr
1			

MARKS

Foreign key

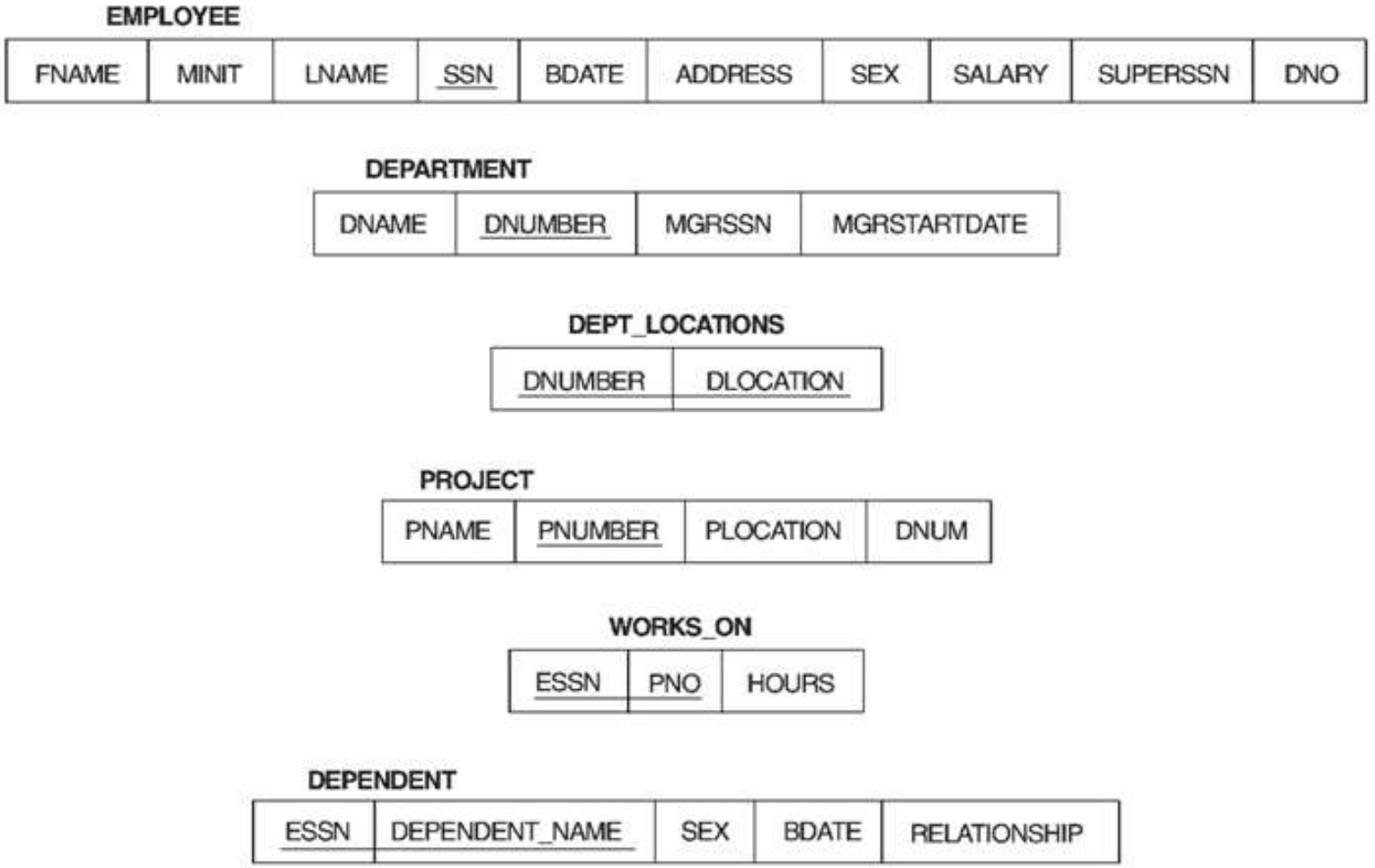
Roll_no	Marks
1	34
1	33
1	56
1	12

Referential Integrity Constraint

The value in the foreign key column FK of the referencing relation R1 can be either:

- **(1) a value of an existing primary key value of the corresponding primary key PK in the referenced relation R2,, or..**
- **(2) a null.**

Schema diagram for the COMPANY relational database schema; the primary keys are underlined.



Querying Relational Data Database Languages

Data definition language (DDL)

Storage definition language (SDL)

View definition language (VDL)

Data manipulation language (DML)

- High-level or nonprocedural DML or **set-at-a-time or set-oriented DMLs**
- Low-level or procedural DML or **record-at-a-time DMLs**

DDL

Once the design of a database is completed and a DBMS is chosen to implement the database, the first order of the day is to specify conceptual and internal schemas for the database and any mappings between the two.

In many DBMSs where no strict separation of levels is maintained, one language, called the **data definition language (DDL), is used by the DBA and by database designers to define** both schemas.

SDL

In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only.

Another language, the **storage definition language (SDL), is used to specify the internal schema.**

The mappings between the two schemas may be specified in either one of these languages.

VDL

For a true three-schema architecture, we would need a third language, the **view definition language (VDL), to specify user views and their mappings to the** conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.

DML

Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database.

Typical manipulations include retrieval, insertion, deletion, and modification of the data.

The DBMS provides a **data manipulation language (DML) for these** purposes.

Types of DML

A **high-level or nonprocedural DML can be used on its own to** specify complex database operations in a concise manner. High-level DMLs, such as SQL, can specify and retrieve many records in a single DML statement and are hence called **set-at-a-time or set-oriented DMLs.**

A **low-level** or **procedural DML** typically retrieves individual records or objects from the database and processes each separately. Low-level DMLs are also called **record-at-a-time DMLs because of this** property.

Comprehensive integrated language

In current DBMSs, the preceding types of languages are usually *not considered distinct languages;* rather, a comprehensive integrated language is used that includes constructs for conceptual schema definition, view definition, and data manipulation.

A typical example of a comprehensive database language is the SQL relational database language, which represents a combination of DDL, VDL, and DML, as well as statements for constraint specification and schema evolution.

VIEWS

Introduction

After a table is created and populated with data, it may become necessary to prevent all users from accessing all columns of a table, for data security reasons.

Consider the employee table

Employee(fname, lname,SSN, Bdate address,dno, salary, supSSN)

The company does not want to show the salary of employee to others.

For that reason, company will create a view for the others that view will contain all data except the salary of employee.

Introduction

Views are logical tables of data extracted from existing tables.

Views do not store any data.

It can be queried just like a table, but does not require disk space .

View stores it ‘s definition in Oracle system catalog.

A view contains no data of its own but it like a window through which data from tables can be viewed or changed.

When a reference is made to a view, its definition is scanned, the base table is opened and the view created on the top of the base table.

It can be used to hide sensitive columns. To do this the owner of the view must grant the select privilege on the view and revoke the privileges on the underlying tables.

Views

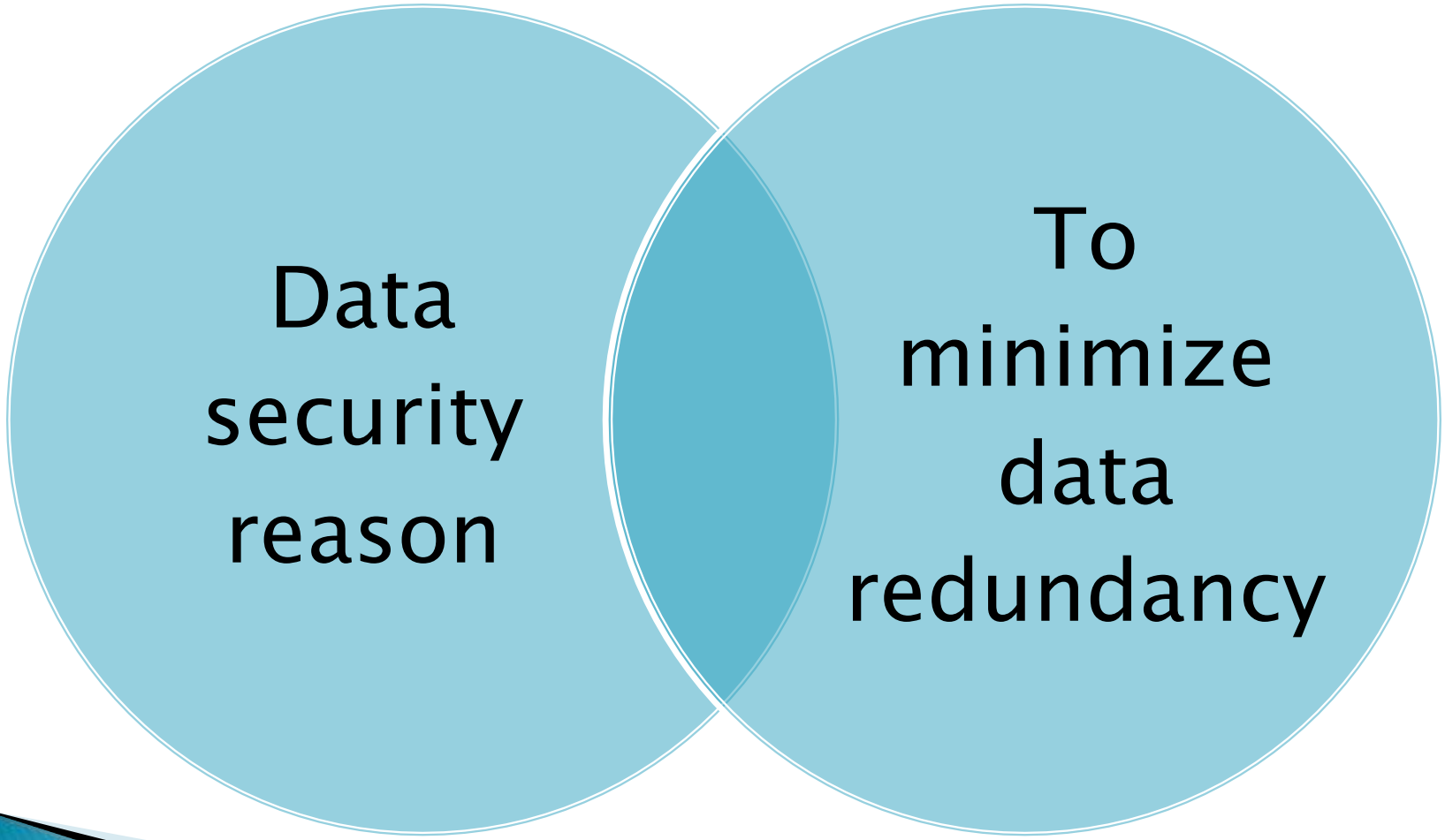
A **view** provides a mechanism to hide certain data from the view of certain users.

A view is a **virtual** table.

Some views are used for looking at table data, called as read-only view.

A view that is used to look at table data as well as insert, update and delete table data is called an updatable view.

Reasons why views are created



Advantage of View

Data independence

Improved security

Reduced complexity: Instead of forcing your users to learn the SQL JOIN syntax you might wish to provide a view that runs a commonly requested SQL statement.
Convenience

Customization: If you wish to display some computed values or column names formatted differently than the base table columns, you can do so by creating views.

Disadvantage of View

1. **Update restriction**
2. **Structure restriction** (new attribute in the base table)
3. **Performance :** Even though views can be a great tool for securing and customizing data, they can be slow.

Creation of views

- **Create** view view-name AS
- **SELECT** column-name, column-name
- **FROM** table-name
- **WHERE** column_name =expression list;

Example

- Create view v_emp AS
- Select fname, lname,SSN, Bdate, address,dno, supSSN from employee;

- Create view v_emp AS
- Select * from employee;

To display the view

- **Select *from emp_view;**

To view the columns in a view

- **Desc view_name**

Renaming the columns of a view

- Create view v_emp AS
- Select fname, lname,SSN, Bdate, address add1 ,dno dnumber, supSSN from employee;

Selecting a data set from a view

- Once a view has been created, it can be queried exactly like a base table.

```
select column-name, column_name
FROM view_name;
```

Updateable view

Views on which data manipulation can be done are called updateable views.

Views defined from single table

If the user wants to INSERT records with the help of a view, then the PRIMARY KEY columns and all the NOT NULL columns must be included in the view.

The user can UPDATE, DELETE records with the help of a view even if the PRIMARY KEY column and NOT NULL columns are excluded from the view definition.

INSERT, UPDATE, DELETE

Insert into view_name values (……);

Update view_name set bdate=…. Where fname=…..;

Delete from view name where fname=……;

Common restrictions on updateable views

For the view to be updateable the view definition must not include:

- **Aggregate functions**
- **DISTINCT, GROUP BY or HAVING clause**
- **Sub-queries**
- **Constants, strings or value expression like price*.5**
- **UNION, INTERSECT or MINUS clause**

Destroying a view

Drop view
view_name;

Relational Algebra

Relational Algebra

Relational algebra is a procedural query language.

A query language is a language in which a user requests information from the database.

The basic set of operations for the relational model is known as the relational algebra.

Unary Relational Operations

These operations are called unary operations, because they operate on one relation.

select: σ(sigma)

project: Π(pi)

rename: *ρ (rho)*

SELECT Operation

SELECT operation is used to select a subset of the tuples from a relation that satisfy a **selection condition**.

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

where the symbol **σ** (sigma) is used to denote the select operator. The selection condition appears as a subscript to *σ.*

Write the Relational Algebra expression

- Select those tuples of the loan relation where the branch is “Perryridge”
- Find tuples in which the amount lent is more than $1200.
- Find those tuples pertaining to loans of more than $1200 made by the Perryridge branch.

PROJECT Operation

This operation selects certain columns from the table and discards the other columns.

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

where π (pi) is the symbol used to represent the project operation and <attribute list> is the desired list of attributes from the attributes of relation R.

Write the Relational Algebra expression

1. List all loan numbers and the amount of the Loan.

Loan Relation

loan-number	*branch-name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Give the Relational Algebra expression

1. Find those customers who live in Harrison.

customer relation

customer-name	*customer-street*	*customer-city*
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

Rename Operation

The rename operation is represented by the lowercase Greek letter ρ (rho).

Renaming operation is very useful in the case of union and join operation.

In improves the readability and better understandability.

We rename relations as well as the attribute.

Syntax for rename:

$\rho_{S(B1, B2, \ldots, Bn)}(R)$ is a renamed relation S based on R with column names $B_1, B_2, \ldots\ldots B_n$.

$\rho_S(R)$ is a renamed relation S based on R.

$\rho_{(B1, B2, \ldots, Bn)}(R)$ is a renamed column names $B_1, B_2, \ldots\ldots B_n$ based on R.

Rename Operation

ρ_{EMP1} (EMP)

In this example we only rename the relation from EMP to EMP1.

$\rho_{EMP1(Name1,dept1)}$ (EMP)

In this example we rename EMP as EMP1 and attributes Name and dept as Name1 and dept1.

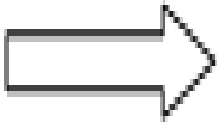
$_{(Name1,dept1)}$ (EMP)

In this example we rename the attributes only.

EMP

Name	dept
Animesh	CSE
Jyoti	CSE
Kalpna	IT
Anamika	IT

After Rename



EMP1

Name 1	Dept 1
Animesh	CSE
Jyoti	CSE
Kalpna	IT
Anamika	IT


THANK
YOU