

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

شروع اللہ کے پاک نام سے جو بڑا مہربان نہایت رحم والا ہے



Dr. Abid Sohail Bhutta

abidbhutta@cuilahore.edu.pk

**Department of Computer
Science,**

**COMSATS University , Lahore
Campus**

Database Systems

Lecture 11

Creation of Subschemas using Views (SQL Views) and Indexes (SQL Indexes)



Recall Lecture 10

- SQL Constraints
 - ❑ Foreign key,
 - ❑ Unique,
 - ❑ Not Null
 - ❑ Check constraints in SQL

Schema and Subschemas


 **Internal Level**
Physical Database


 **DBMS** DBMS Software


Some end-user applications can
be supported by *views*

 **Schema** **Conceptual Level**

**External
Level**

 **Subschema**

 **Subschema**

 **Subschema**

 **User**

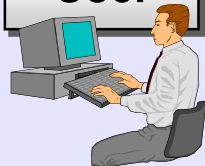
 **User**

 **User**

 **User**

 **User**

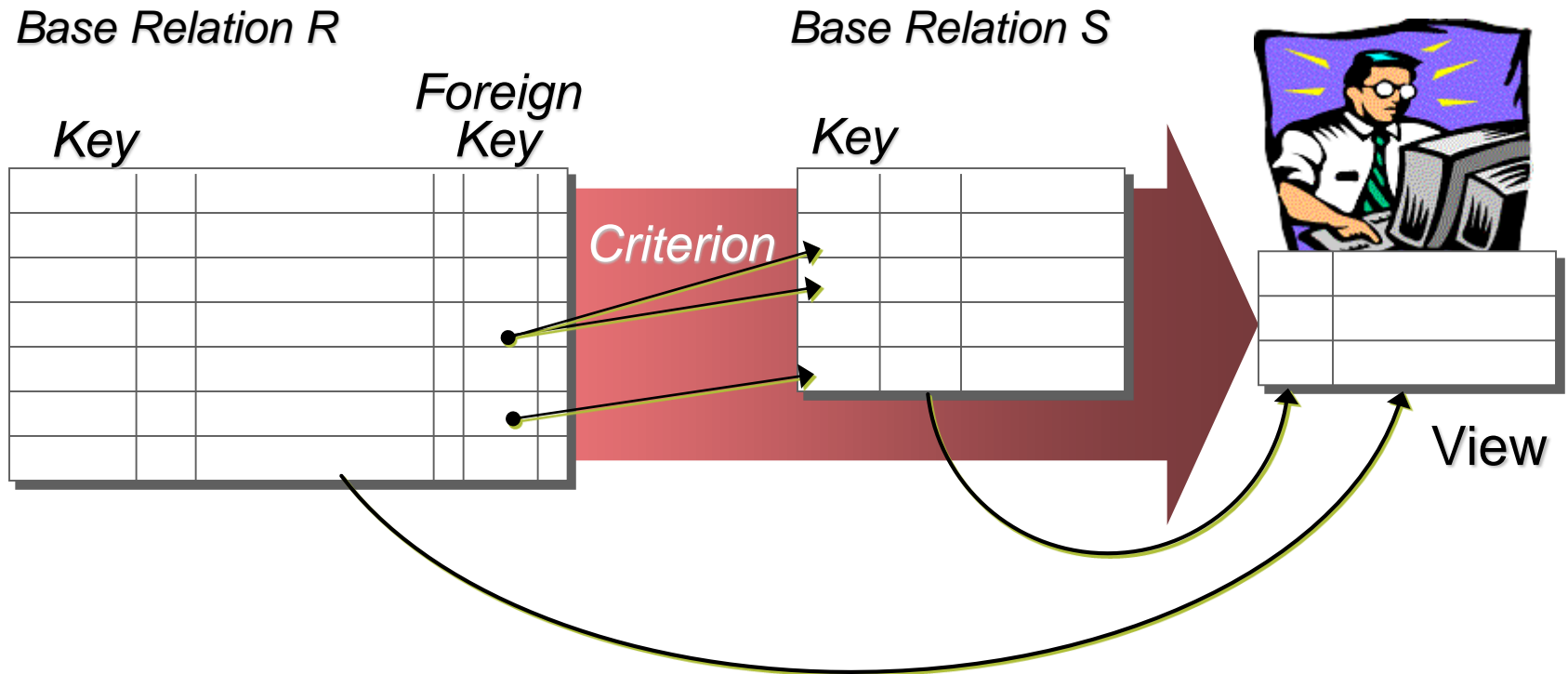
 **User**



Views

- A view is a *virtual relation* or one that does not actually exist, but *dynamically derived*
 - Can be constructed by performing operations (i.e., select, project, join, etc.) on values of existing base relations
 - Base relation - a named relation, corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database
 - View - a dynamic result of one or more relational operations operating on the base relations to produce another

Views



Purpose of Views

- Provides a powerful and flexible security mechanism by hiding parts of the database from certain users
- Permits user access in a way that is customized to their needs
- Simplify complex operations on the base relations
- Designed to support the external model
- Provides logical independence

SQL Code Example

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

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

SQL Code Example (Continue ..)

- `SELECT * FROM [Brazil Customers];`

Number of Records: 9

CustomerName	ContactName
Comércio Mineiro	Pedro Afonso
Familia Arquibaldo	Aria Cruz
Gourmet Lanchonetes	André Fonseca
Hanari Carnes	Mario Pontes

Types of Views

View	Description
Simple View	A view based on the only a single table, which doesn't contain GROUP BY clause and any functions.
Complex View	A view based on multiple tables, which contain GROUP BY clause and functions
Inline view	A view based on a subquery in FROM Clause, that subquery creates a temporary table and simplifies the complex query.
Materialized view	A view that stores the definition as well as data. It creates replicas of data by storing it physically.

Simple View

Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
CREATE VIEW emp_view AS  
SELECT EmployeeID, Ename  
FROM Employee  
WHERE DeptID=2;
```

Creating View
by filtering
records using
WHERE clause

emp_view

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1004	David	2	5000
1005	Mark	2	3000

Complex View

Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
CREATE VIEW emp_view AS  
SELECT DeptID, AVG(Salary)  
FROM Employee  
GROUP BY DeptID;
```

Create View of
grouped records
on Employee
table

emp_view

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

Simple View vs Complex View

Features	Simple Views	Complex Views
No. of tables	One	One or More
Containing Functions	No	Yes
Contain Group of data	No	Yes
DML through view	Yes	Not Allowed
Features	Simple Views	Complex Views

Inline View

- An inline view is a SELECT statement in the FROM-clause of another SELECT statement (Sub - Query) to create a **temporary table** that could be referenced by the SELECT statement.

```
SELECT column_names, ...  
FROM (subquery)  
WHERE ROWNUM<= N;
```


Materialized View

- Materialized view replicates the retrieved data physically.
 - This replicated data can be reused without executing the view again.
- This type of view is also known as "SNAPSHOTS". Materialized view reduce the processing time to regenerate the whole data.
 - It helps remote users to replicate data locally and improve query performance.
- The challenging part here is to synchronize the changes in materialized views underlying tables.

Materialized View

Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
CREATE MATERIALIZED VIEW emp_view AS  
SELECT EmployeeID, Ename  
FROM Employee  
WHERE DeptID=2;
```

Creating Materialized View
by filtering records using
WHERE clause

emp_view

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1004	David	2	5000
1005	Mark	2	3000

This view stores the retrieved data physically on memory.

Materialized View

Comparison Between View and

View	Materialized View
View is a logical structure of the table which will be used to retrieve data from one or more table.	Materialized views are also logical structure but data is physically stored in database.
Data access is slower compared to materialized views	Data access is faster compared to simpler view because data is directly accessed from physical location
Views are generally used to restrict data from database	Materialized Views are used in Data Warehousing.

Advantages of view

- Views can be utilized as a subset of actual data to perform certain operations.
- It helps us to provide an abstraction to various users or hide the complexity for users who are accessing data from the table.
 - For example, a user has permission to access particular columns of data rather than the whole table.
- It can help us to simplify complex queries into a simpler one.
- It also simplifies data access from multiple joined tables.
- It can be used as aggregated tables using group by operations.
- Views can be used for security purposes or can add extra value from the security point of view.
- It does not hold any space because it only has the definition in the data dictionary, not the copy of actual data.

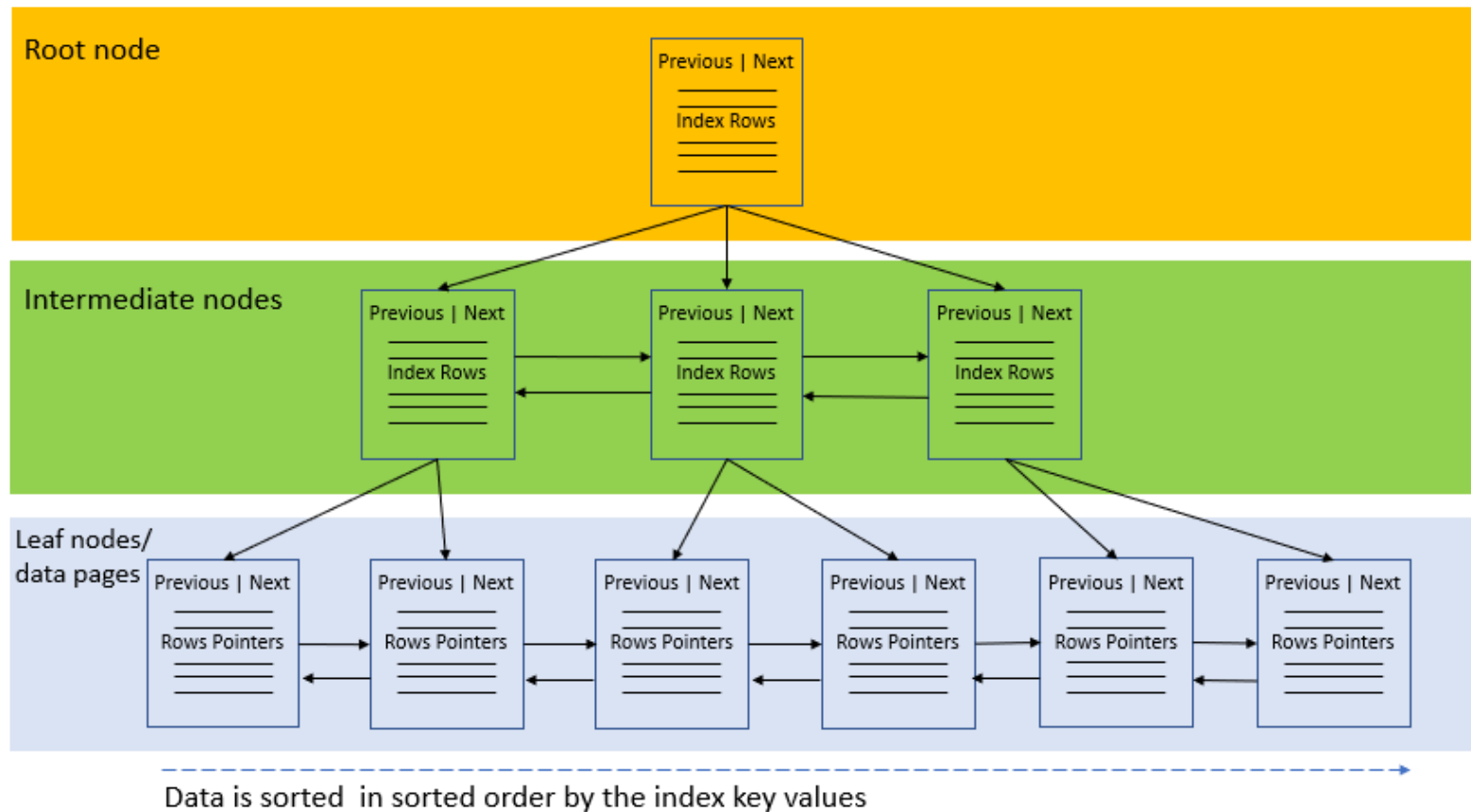
Indexes

- An **index** is a schema object. It is used by the server to **speed up** the retrieval of rows by using a pointer.
 - Reduce disk **I/O**(input/output) by using a rapid path access method to locate data quickly.
- An index helps to speed up **select** queries and **where** clauses,
 - It slows down data input, with the **update** and the **insert** statements.

Indexes (Continue)

- Indexes can be created or dropped with no effect on the data.
- For example, if you want to reference all pages in a book that discusses a certain topic, you first refer to the index, which lists all the topics alphabetically and is then referred to one or more specific page numbers.

Types of Indexes



- ☐ **Clustered Index**
- ☐ **Non-Clustered Index**

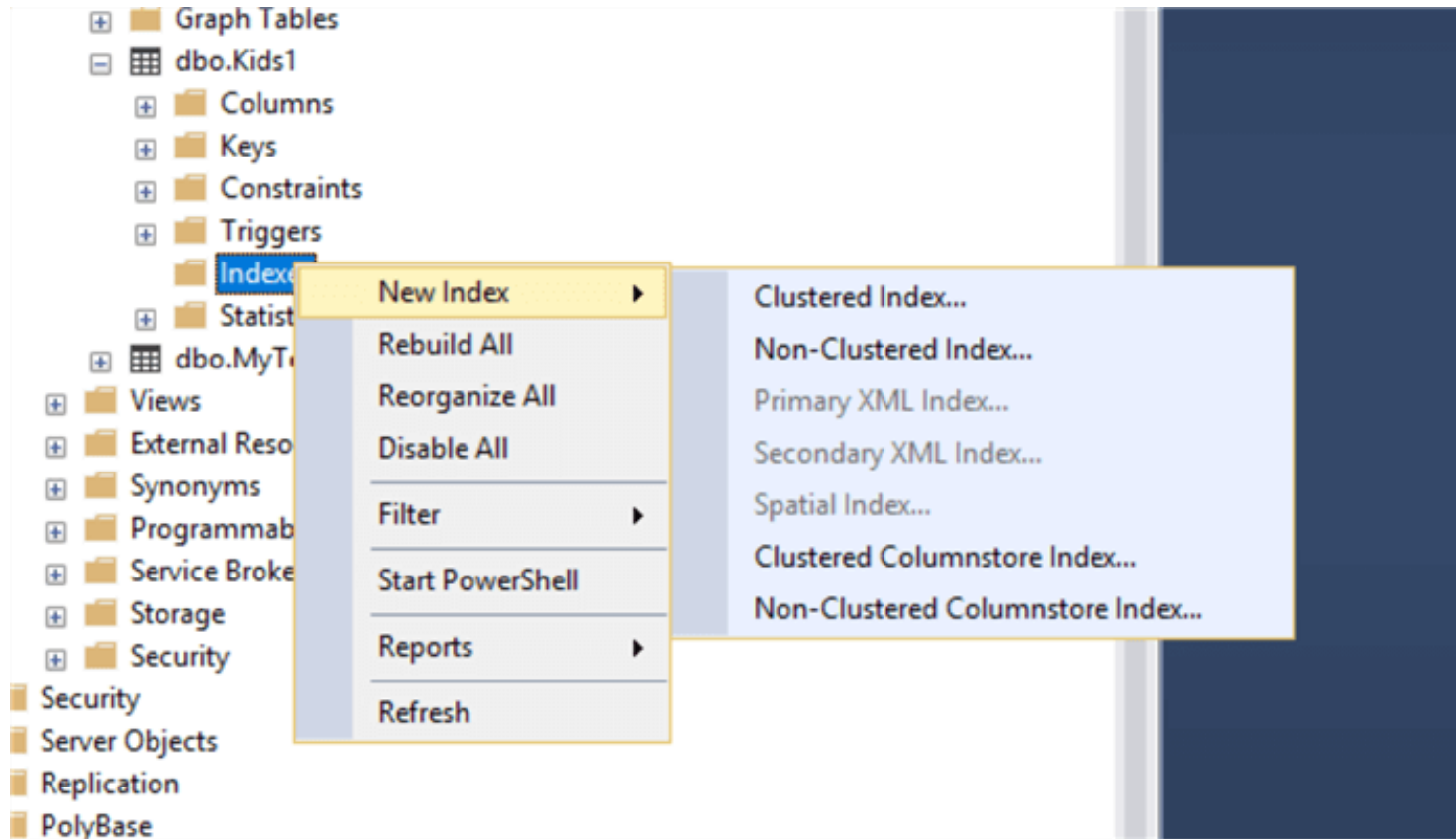
Clustered Index

- A clustered index defines the order in which data is physically stored in a table.
 - ❑ Table data can be sorted in only way, therefore, there can be only one clustered index per table.
 - ❑ In SQL Server, the primary key constraint automatically creates a clustered index on that particular column.
 - ❑ Example :

Non-Clustered Indexes

- A non-clustered index doesn't sort the physical data inside the table.
 - ❑ In fact, a non-clustered index is stored at one place and table data is stored in another place.
 - ❑ This is similar to a textbook where the book content is located in one place and the index is located in another.
 - ❑ This allows for more than one non-clustered index per table.

Creating an Index (SQL Server)



Creating an Index

CREATE INDEX index ON TABLE column;

- ❑ **index** is the name given to that index and
- ❑ **TABLE** is the name of the table on which that index is created
- ❑ **column** is the name of that column for which it is applied.

Creating an Index SQL Code

```
create table Customer_2 (  
    Id            int            identity,  
    FirstName     nvarchar(40)   not null,  
    LastName      nvarchar(40)   not null,  
    City          nvarchar(40)   null,  
    Country       nvarchar(40)   null,  
    Phone         nvarchar(20)   null,  
    constraint PK_CUSTOMER2 primary key (Id));
```

```
create index IndexCustomerName on Customer_2 (  
    LastName ASC,  
    FirstName ASC );
```

In Next Lecture

- Database Schema Designing
 - Entity Relationship Diagram (ER-D)

Thanks