School of Computing Science and Engineering

Program: BCA

Course Code:BCAC2102

Course Name: Database Management System

Lecture-15

Topic- Relational database design: pitfalls

Faculty:-Dr. Satyajee Srivastava

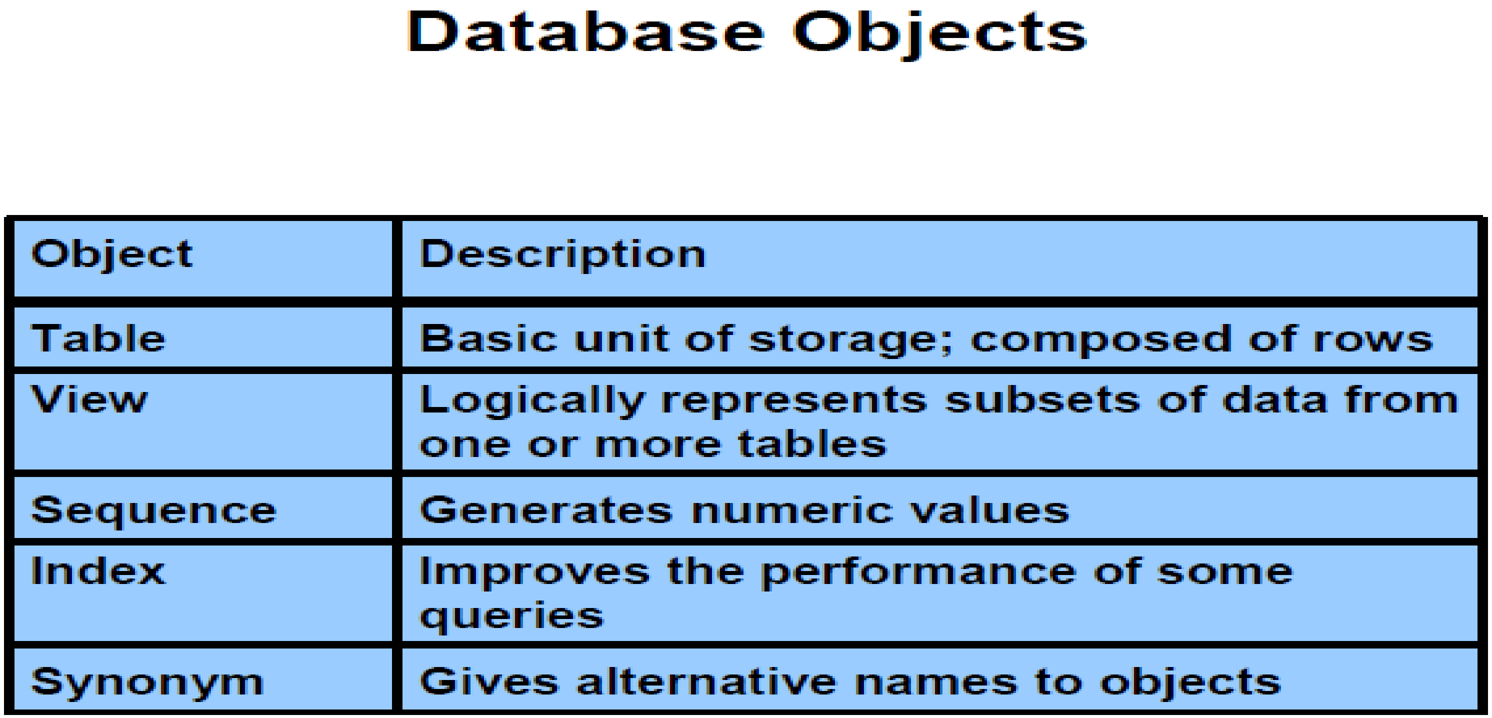
Lecture-14(RECAP)

Topic- Views

Objective :

Understand Views and How To Create Views?

Lecture-14



Database Objects

Object	Description
Table	Basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Lecture-14

VIEW

View is a logical table. It is a physical object which stores data logically. View just refers to data that is tored in base tables.

A view is a logical entity. It is a SQL statement stored in the database in the system tablespace. Data for a view is built in a table created by the database engine in the TEMP tablespace.

Lecture-14

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

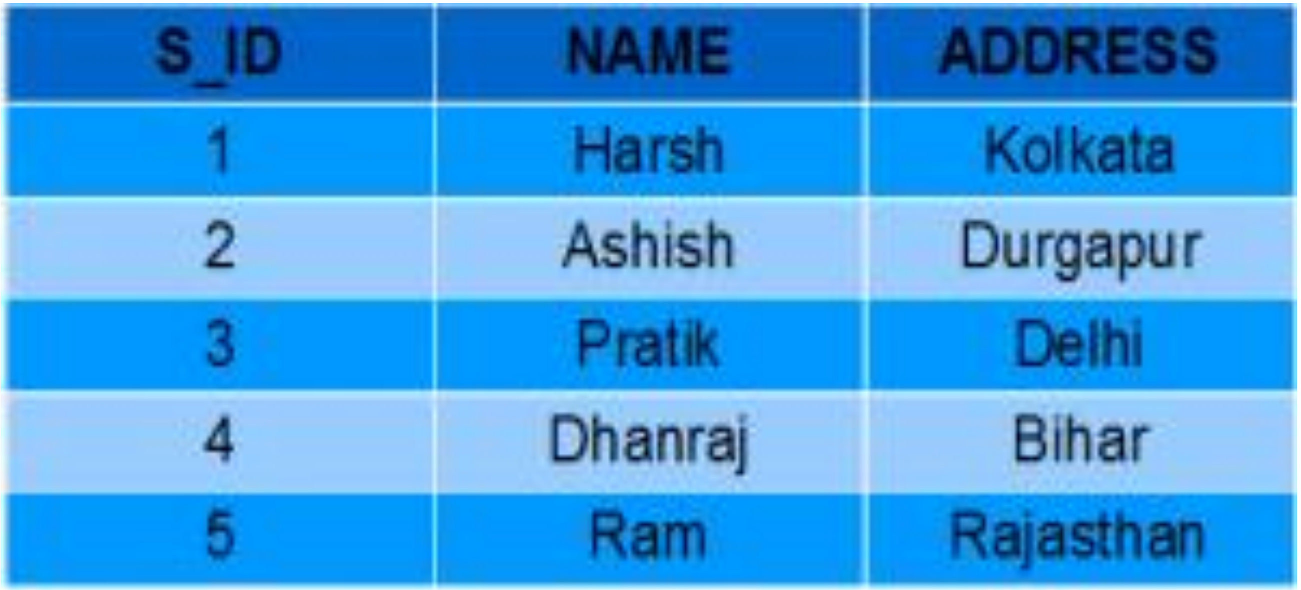
Lecture-14

Creating , deleting and updating Views

Lecture-14

Sample Tables

StudentDetails

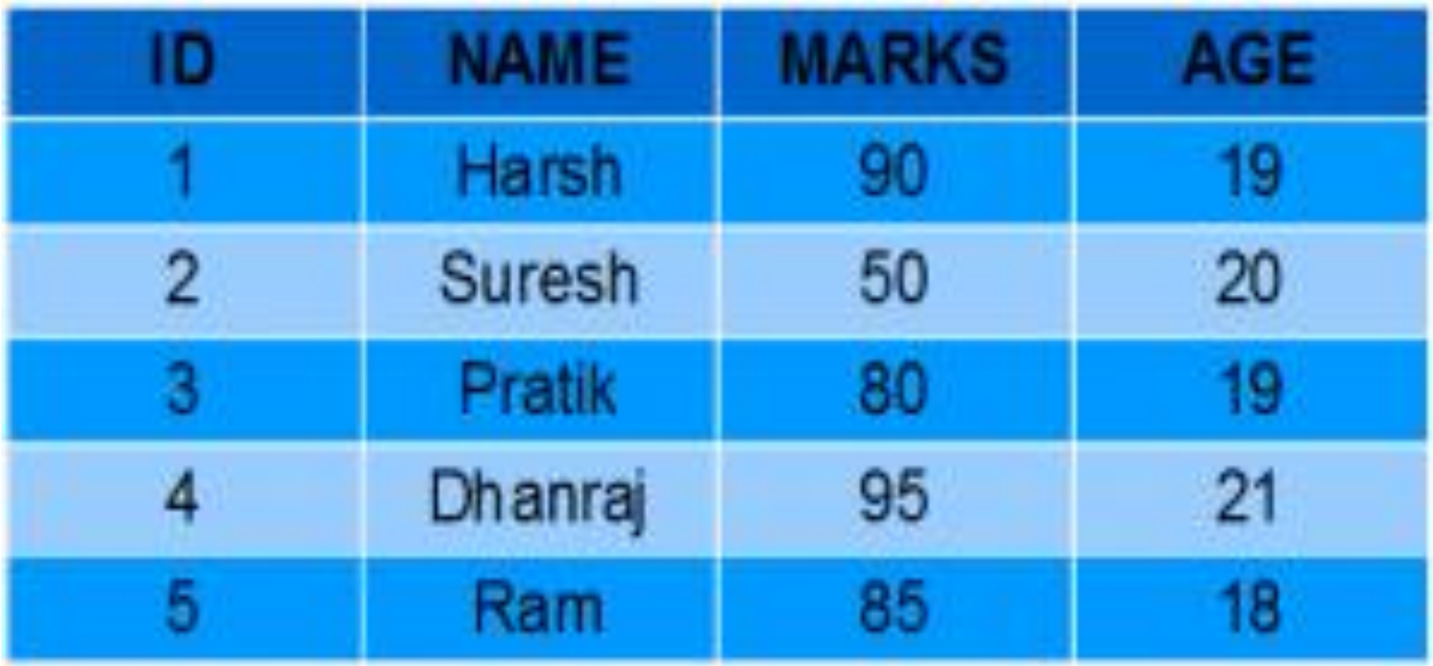


S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

Lecture-14

Sample Tables

StudentMarks



ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

Lecture-14

We can create View using **CREATE VIEW** statement. A View can be created from a single table or multiple tables.

Syntax:

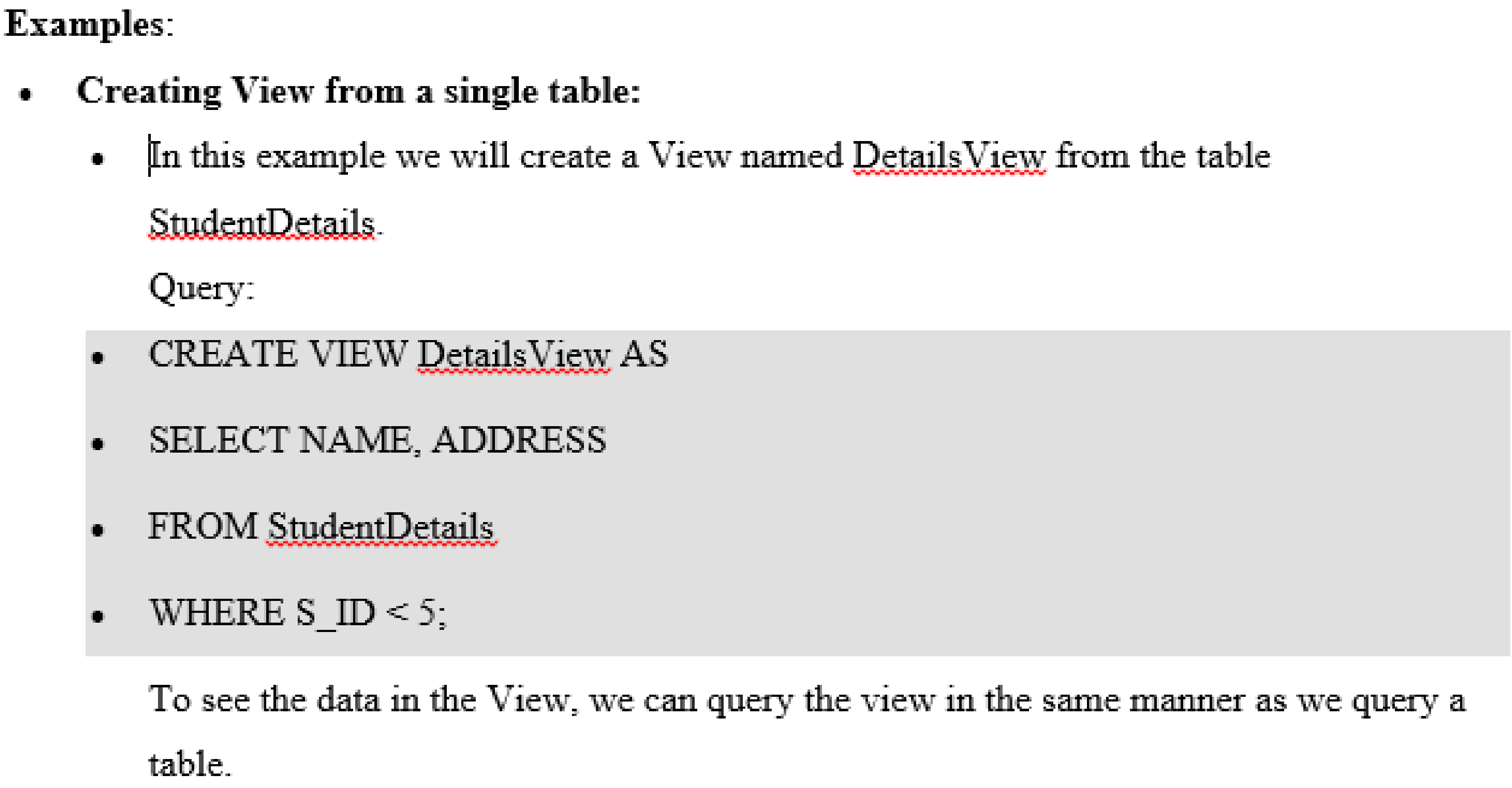
```
CREATE VIEW view_name AS

SELECT column1, column2.....

FROM table_name

WHERE condition;
```

view_name: Name for the View

table_name: Name of the table

condition: Condition to select rows

Lecture-14



Examples:

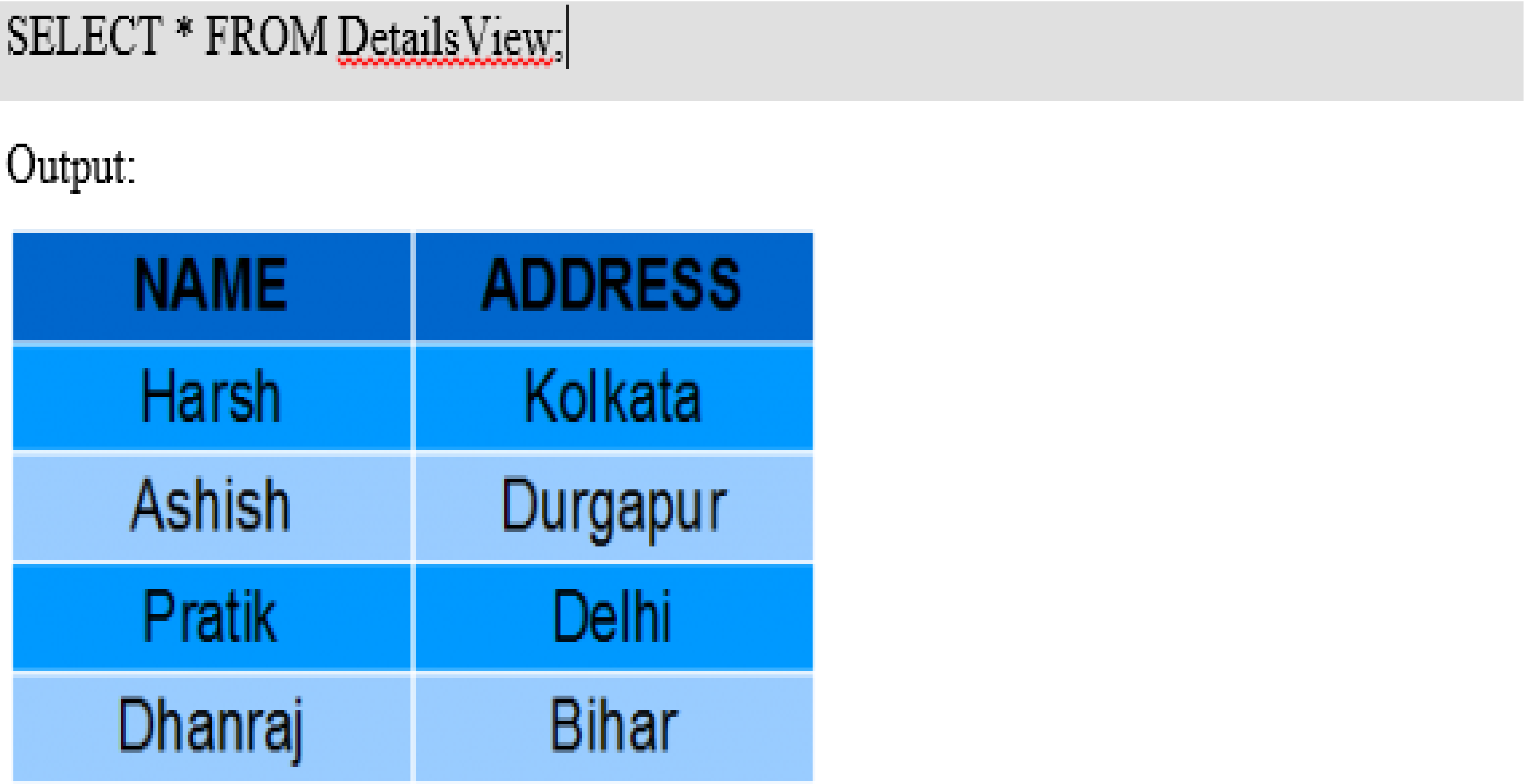
- **Creating View from a single table:**
    - In this example we will create a View named DetailsView from the table StudentDetails.

      Query:

    - CREATE VIEW DetailsView AS
    - SELECT NAME, ADDRESS
    - FROM StudentDetails
    - WHERE S_ID < 5;

  To see the data in the View, we can query the view in the same manner as we query a table.

Lecture-14

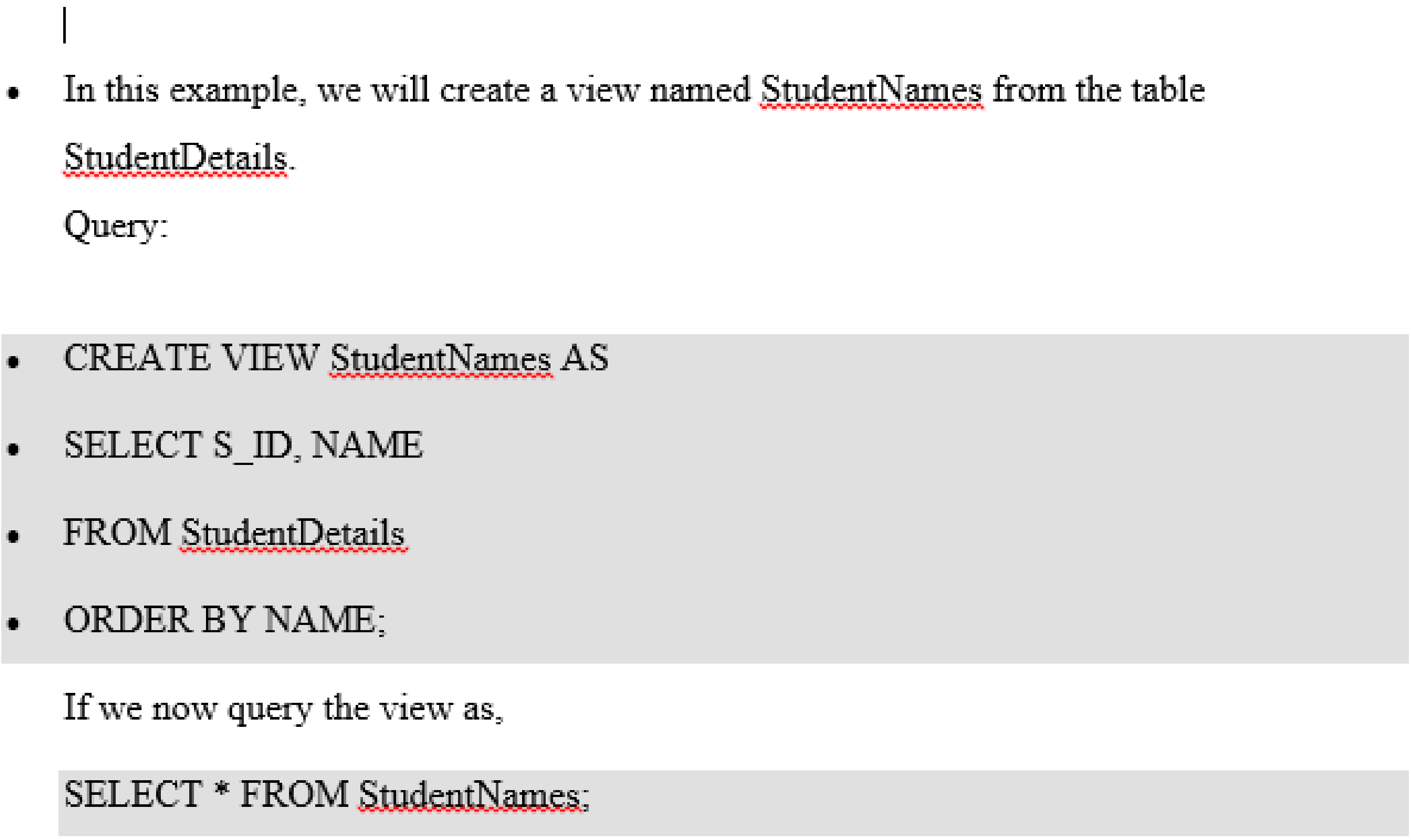


```
SELECT * FROM DetailsView;
```

Output:

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

Lecture-14



- In this example, we will create a view named StudentNames from the table StudentDetails.

  Query:

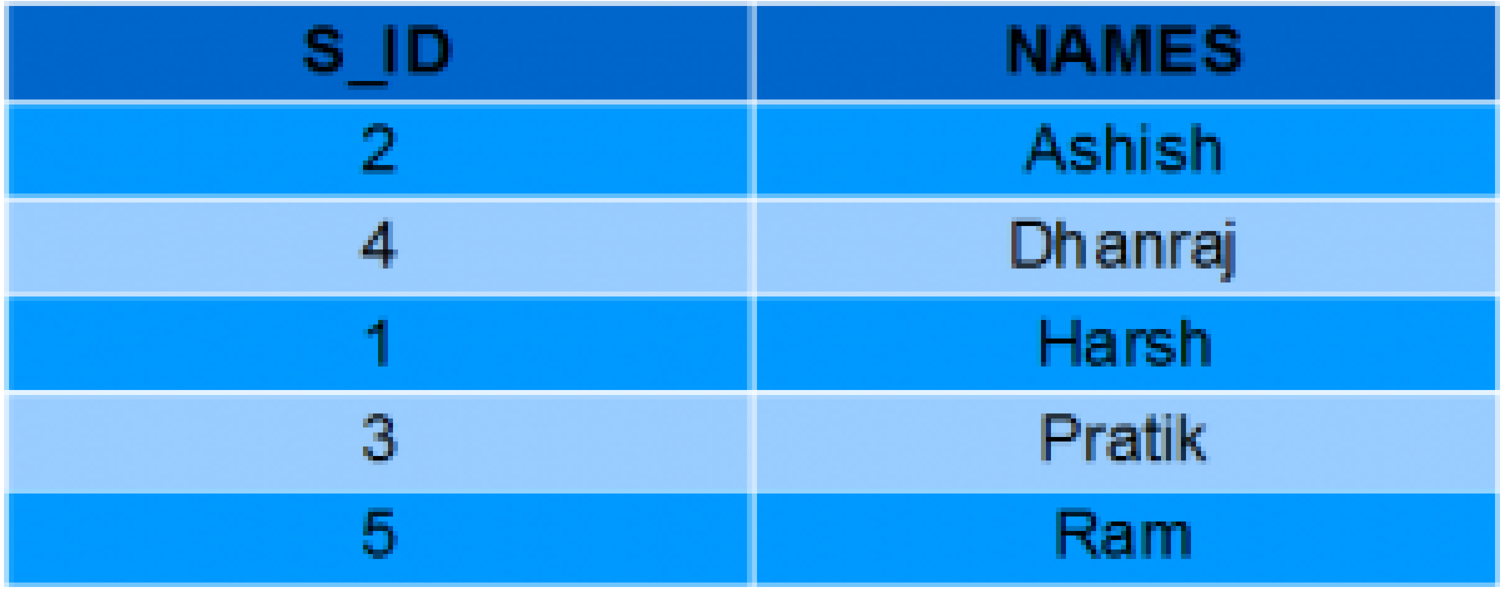
- CREATE VIEW StudentNames AS
- SELECT S_ID, NAME
- FROM StudentDetails
- ORDER BY NAME;

If we now query the view as,

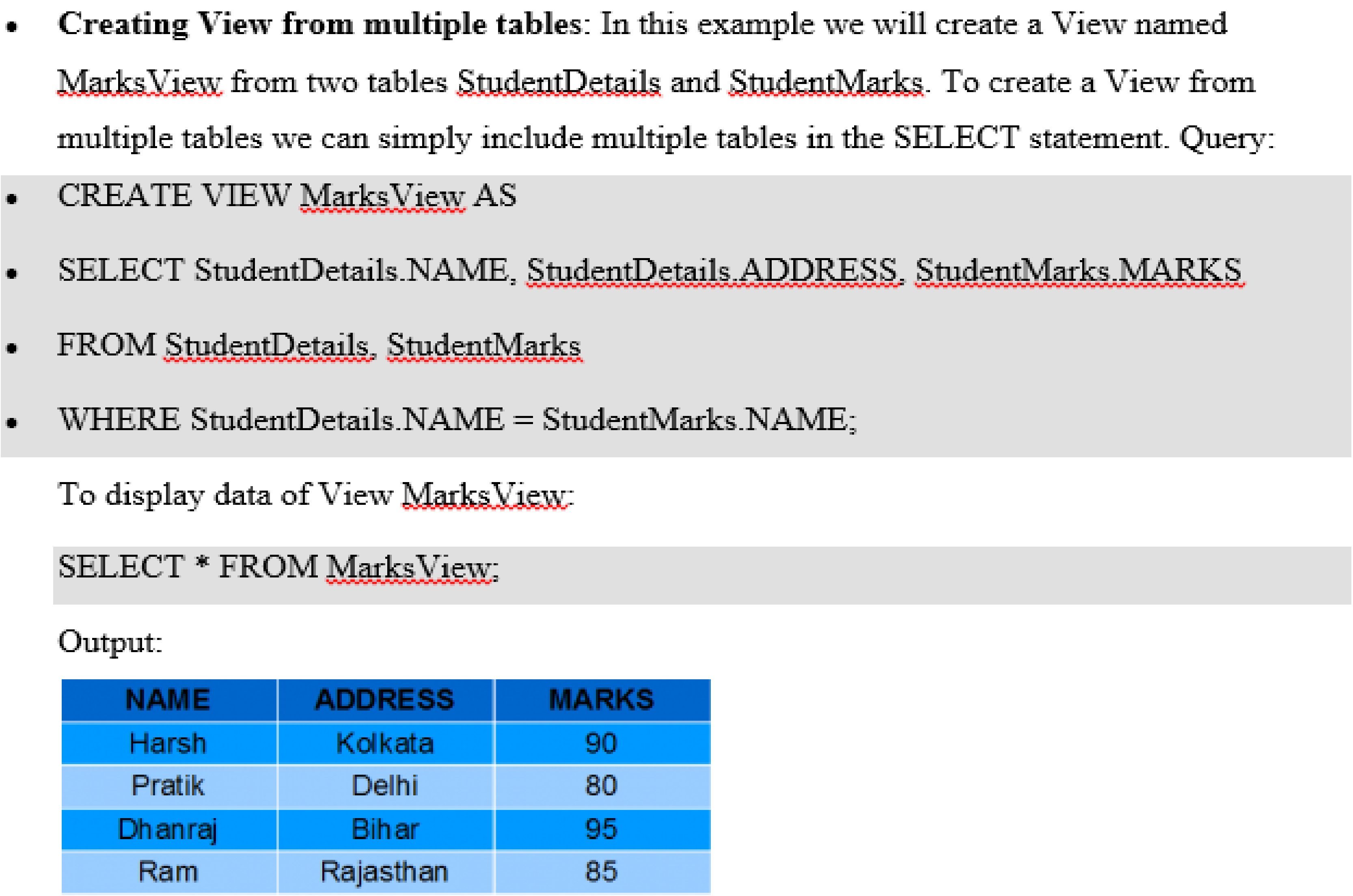
SELECT * FROM StudentNames;

Lecture-14

Output:-



S_ID	NAMES
2	Ashish
4	Dhanraj
1	Harsh
3	Pratik
5	Ram



- **Creating View from multiple tables**: In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement. Query:

```
CREATE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

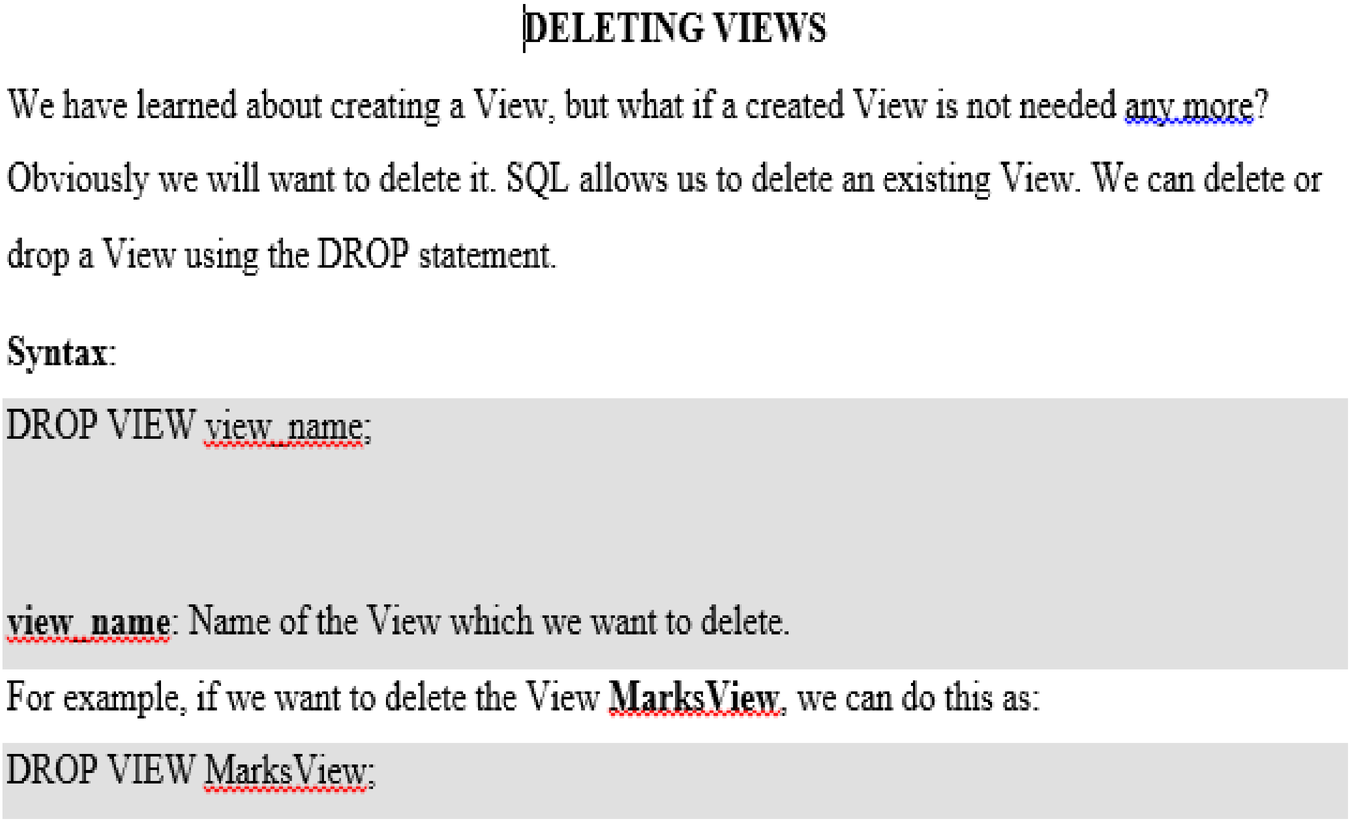
To display data of View MarksView:

```
SELECT * FROM MarksView;
```

Output:

NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

Lecture-14



DELETING VIEWS

We have learned about creating a View, but what if a created View is not needed any more? Obviously we will want to delete it. SQL allows us to delete an existing View. We can delete or drop a View using the DROP statement.

Syntax:

```
DROP VIEW view_name;



view_name: Name of the View which we want to delete.
```

For example, if we want to delete the View **MarksView**, we can do this as:

```
DROP VIEW MarksView;
```

Lecture-14

UPDATING VIEWS

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is **not** met, then we will not be allowed to update the view.

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.

The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view

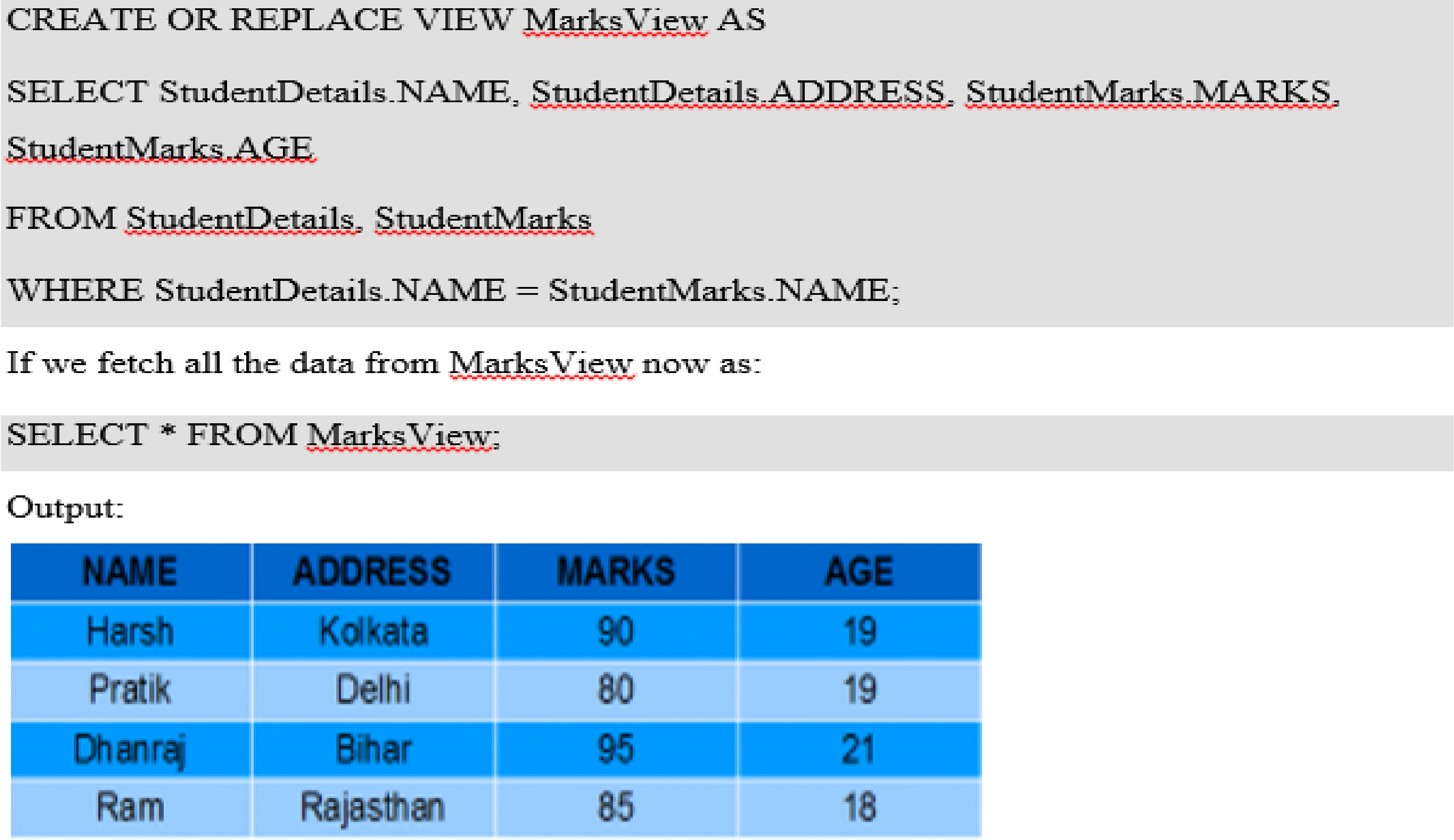
Lecture-14

- We can use the **CREATE OR REPLACE VIEW** statement to add or remove fields from a view.

  Syntax:
- CREATE OR REPLACE VIEW view_name AS
- SELECT column1,coulmn2,..
- FROM table_name
- WHERE condition;

For example, if we want to update the view **MarksView** and add the field AGE to this View from **StudentMarks** Table, we can do this as:

Lecture-14



```
CREATE OR REPLACE VIEW MarksView AS

SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS,
StudentMarks.AGE

FROM StudentDetails, StudentMarks

WHERE StudentDetails.NAME = StudentMarks.NAME;
```

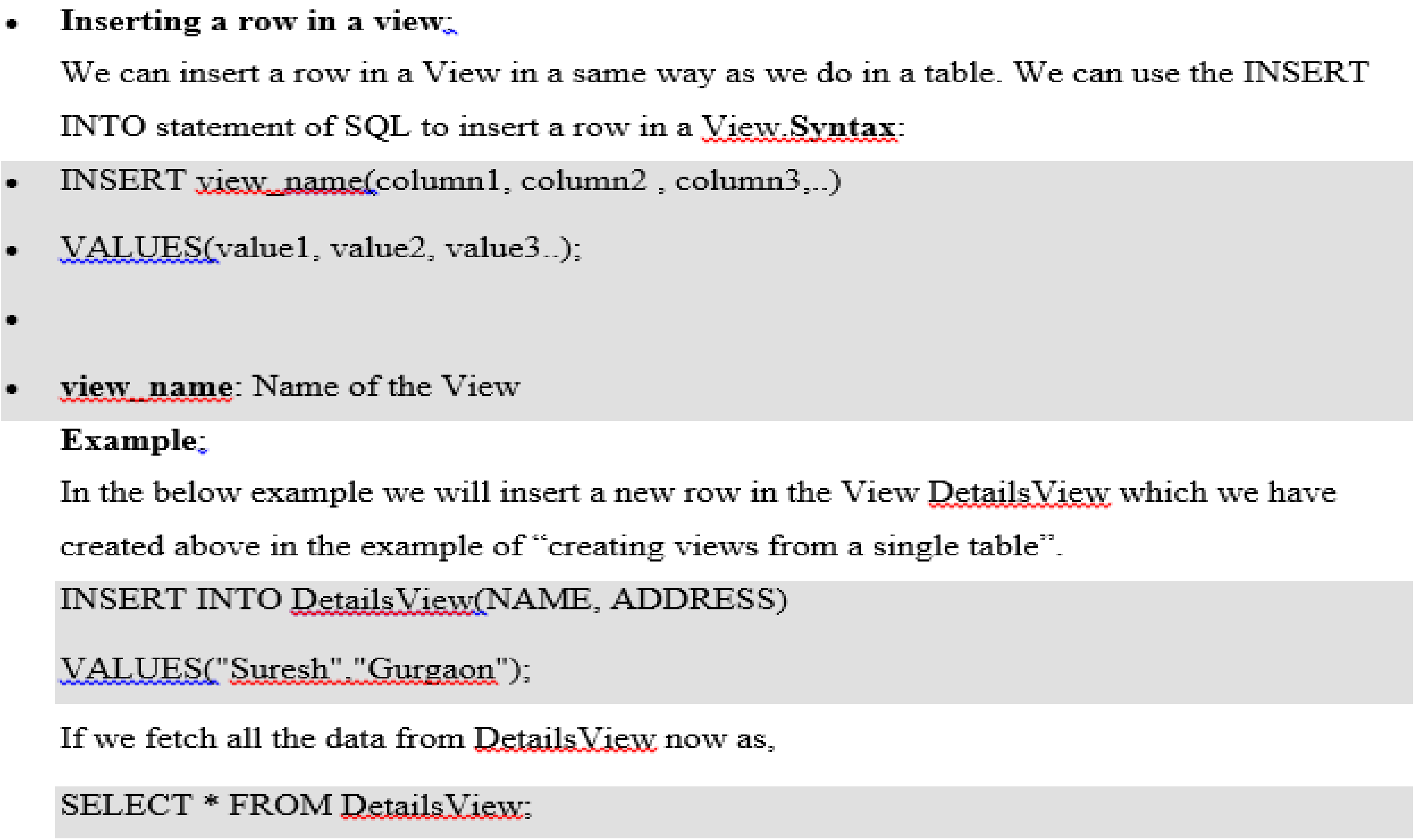
If we fetch all the data from MarksView now as:

```
SELECT * FROM MarksView;
```

Output:

NAME	ADDRESS	MARKS	AGE
Harsh	Kolkata	90	19
Pratik	Delhi	80	19
Dhanraj	Bihar	95	21
Ram	Rajasthan	85	18

Lecture-14



- **Inserting a row in a view:**

  We can insert a row in a View in a same way as we do in a table. We can use the INSERT INTO statement of SQL to insert a row in a View.**Syntax**:

- INSERT view_name(column1, column2 , column3,..)
- VALUES(value1, value2, value3..);
-
- **view_name**: Name of the View

Example:

In the below example we will insert a new row in the View DetailsView which we have created above in the example of "creating views from a single table".

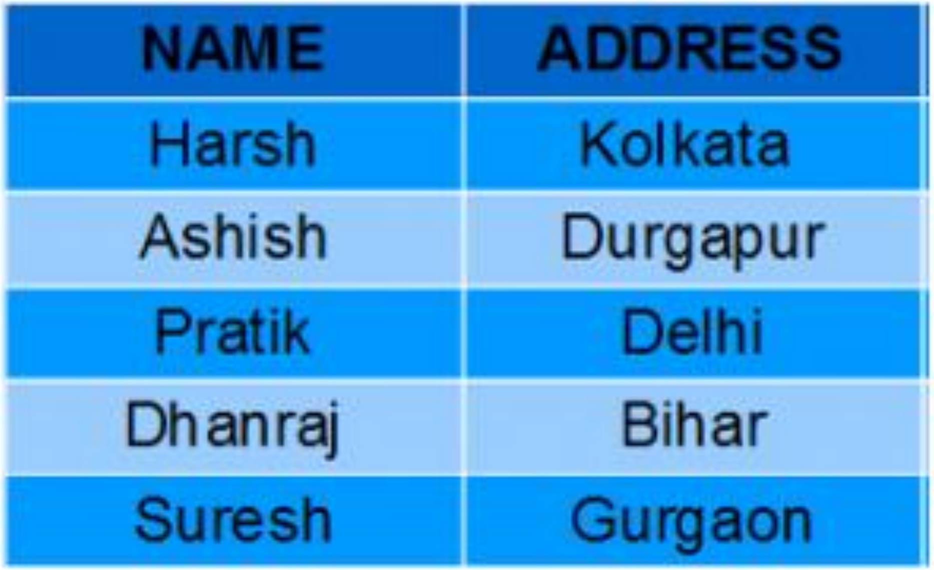
```
INSERT INTO DetailsView(NAME, ADDRESS)

VALUES("Suresh","Gurgaon");
```

If we fetch all the data from DetailsView now as,

```
SELECT * FROM DetailsView;
```

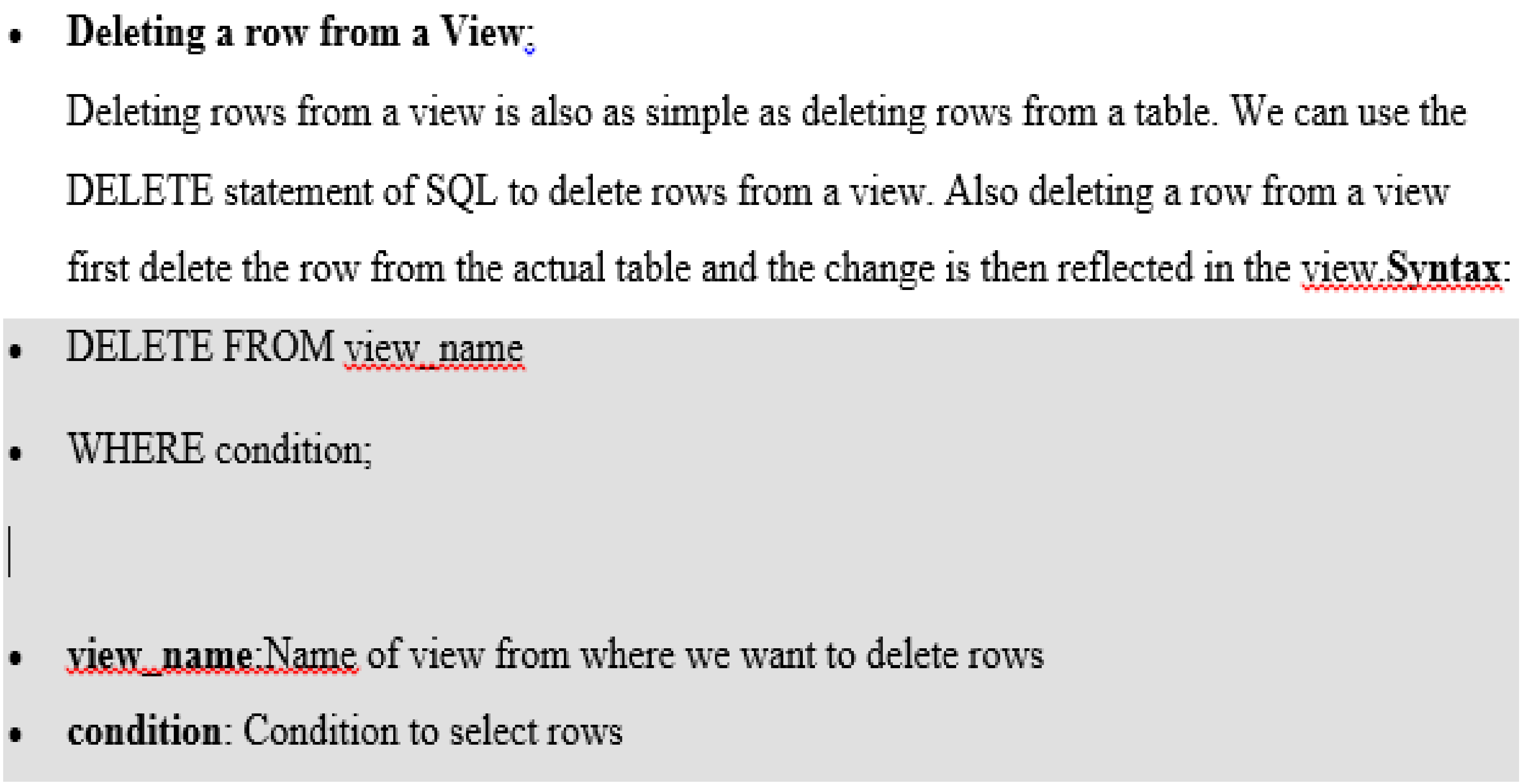
Lecture-14

Output:



NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar
Suresh	Gurgaon

Lecture-14



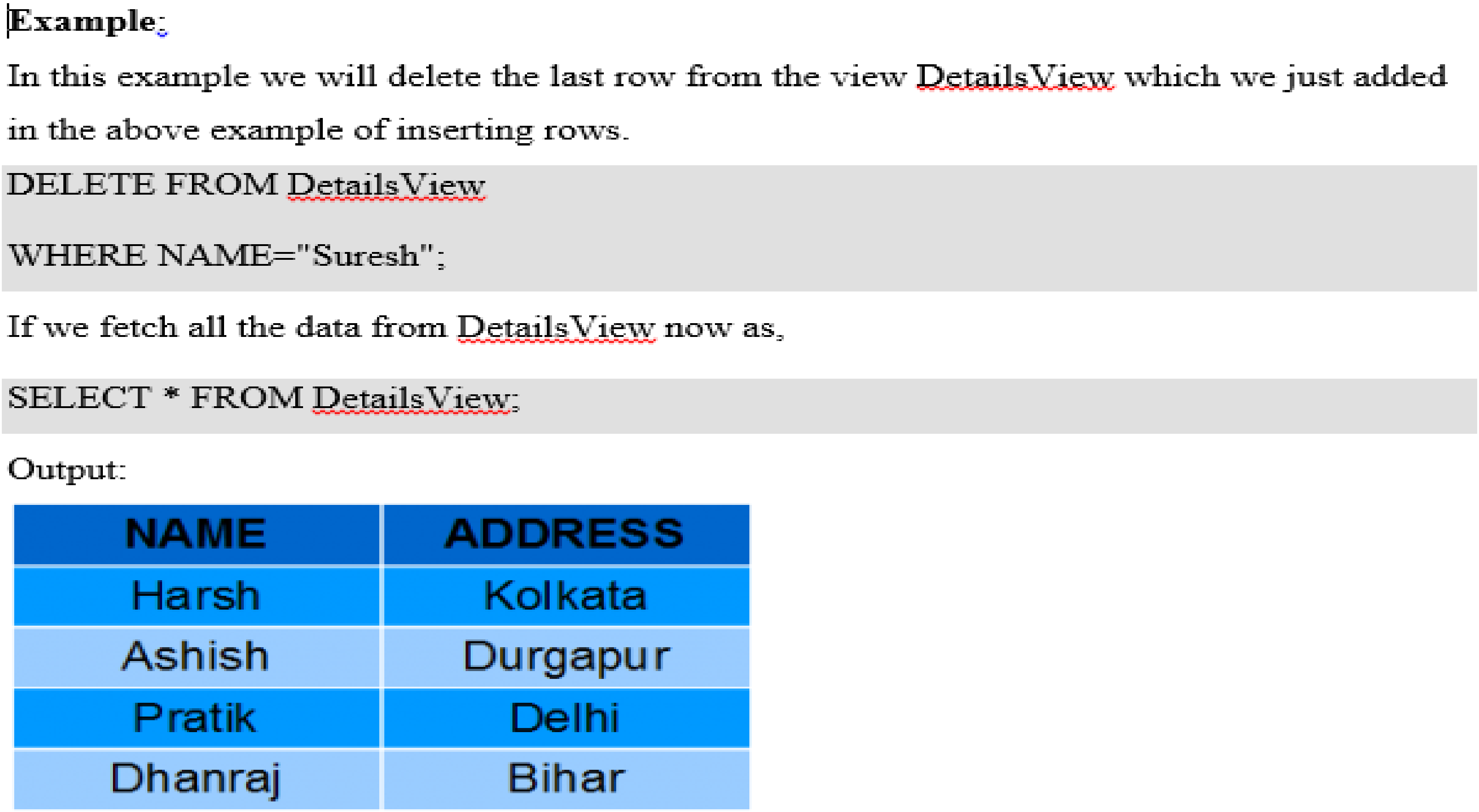
- **Deleting a row from a View:**

  Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view. Also deleting a row from a view first delete the row from the actual table and the change is then reflected in the view.**Syntax**:

- DELETE FROM view_name
- WHERE condition;

- **view_name**:Name of view from where we want to delete rows
- **condition**: Condition to select rows

Lecture-14



Example:

In this example we will delete the last row from the view DetailsView which we just added in the above example of inserting rows.

```
DELETE FROM DetailsView

WHERE NAME="Suresh";
```

If we fetch all the data from DetailsView now as,

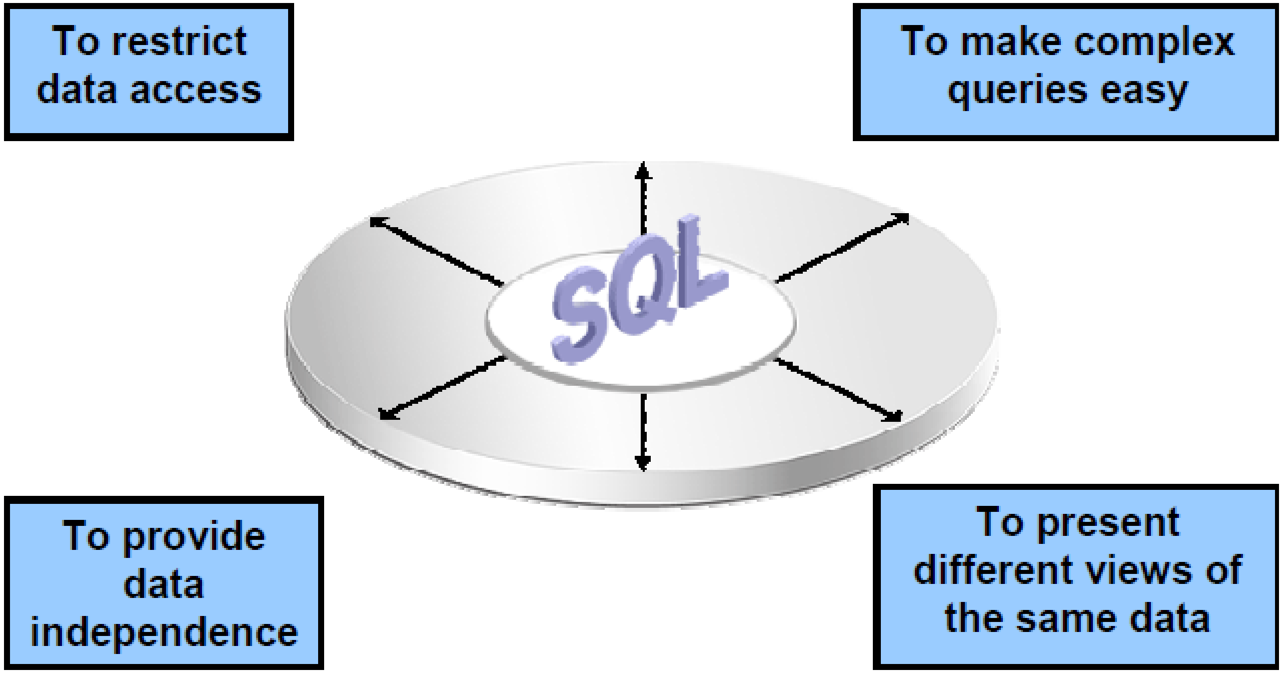
```
SELECT * FROM DetailsView;
```

Output:

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

Lecture-14

Advantages of Views



To restrict data access

To make complex queries easy

To provide data independence

To present different views of the same data

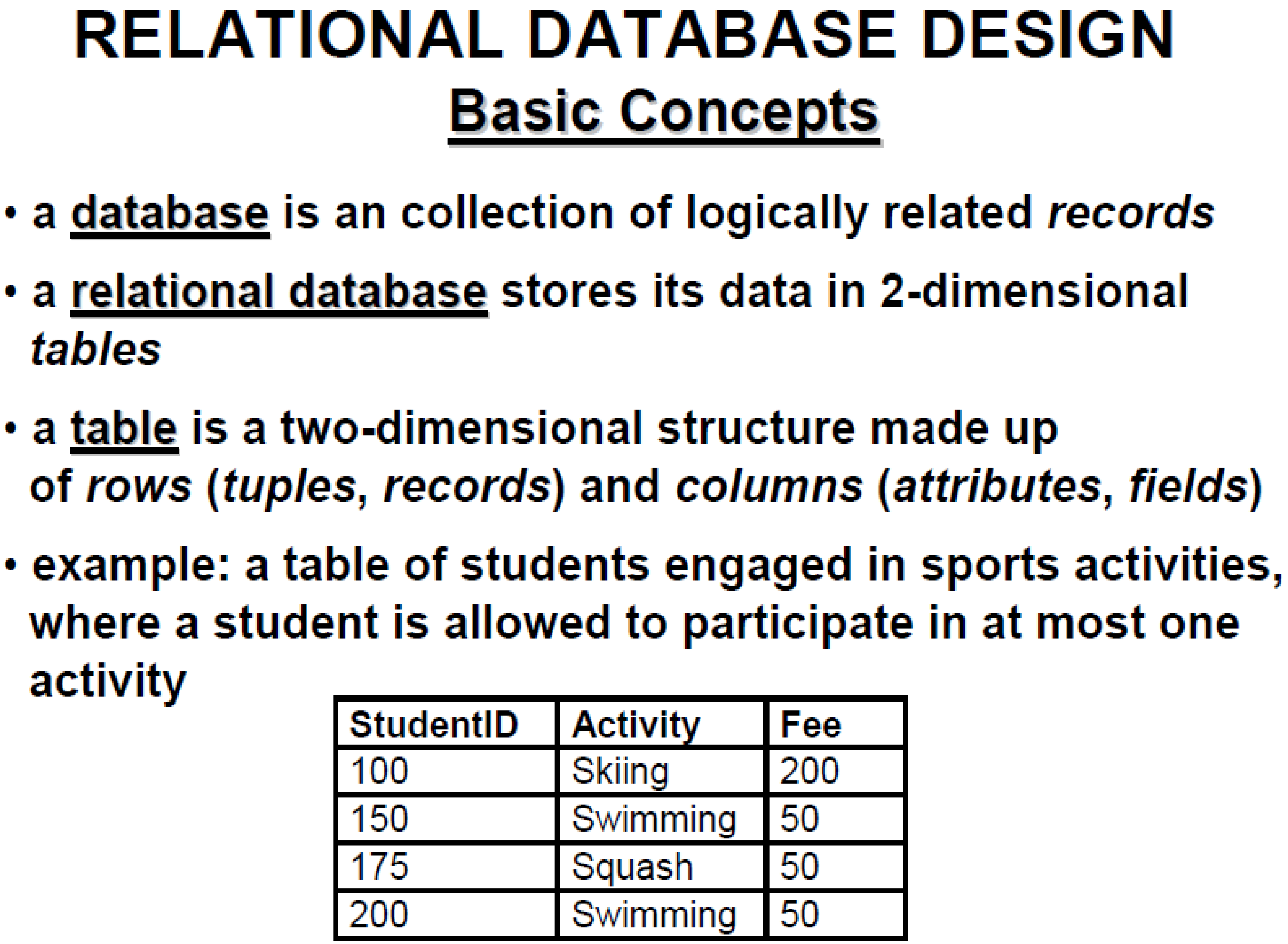
Lecture-15

Topic- Relational database design: pitfalls

Objective :

Discuss Relational database design: pitfalls.

Lecture-15



RELATIONAL DATABASE DESIGN

Basic Concepts

- a database is an collection of logically related *records*
- a relational database stores its data in 2-dimensional *tables*
- a table is a two-dimensional structure made up of *rows* (*tuples*, *records*) and *columns* (*attributes*, *fields*)
- example: a table of students engaged in sports activities, where a student is allowed to participate in at most one activity

StudentID	Activity	Fee
100	Skiing	200
150	Swimming	50
175	Squash	50
200	Swimming	50

Lecture-15

What is Relational Algebra?

Relational algebra is a widely used procedural query language. It collects instances of relations as input and gives occurrences of relations as output. It uses various operation to perform this action.

Relational algebra operations are performed recursively on a relation. The output of these operations is a new relation, which might be formed from one or more input relations.

Lecture-15

Basic Relational Algebra Operations:

Relational Algebra devided in various groups

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol:)

Lecture-15

Relational Algebra Operations From Set Theory

- UNION (u)
- INTERSECTION ( ),
- DIFFERENCE (-)
- CARTESIAN PRODUCT (x)

Binary Relational Operations

- JOIN
- DIVISION

Lecture-15 SELECT (σ)

- $\sigma_p(r)$
- σ is the predicate
- r stands for relation which is the name of the table
- p is prepositional logic
- **Example 1**
- $\sigma_{\text{topic = "Database"}}$ (Subject)

Output - Selects tuples from Subject where topic = 'Dat abase'.

Lecture-15

Example 2

- $\sigma_{\text{topic = "Database" and author = “korth"}}$(Subject)
- **Output** - Selects tuples from Tutorials where the topic is 'Database' and 'author' is korth.
- **Example 3**
- $\sigma_{\text{sales > 50000}}$ (Customers)
- **Output** - Selects tuples from Customers where sales is greater than 50000

Lecture-15

Projection(π)

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

This helps to extract the values of specified attributes to eliminates duplicate values. (pi) The symbol used to choose attributes from a relation. This operation helps you to keep specific columns from a relation and discards the other columns.

Lecture-15

Example of Projection:

Consider the following table

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

Here, the projection of CustomerName and status will give

$\Pi_{\text{CustomerName, Status}}$ (Customers)

Lecture-15

CustomerName	Status
Google	Active
Amazon	Active
Apple	Inactive
Alibaba	Active

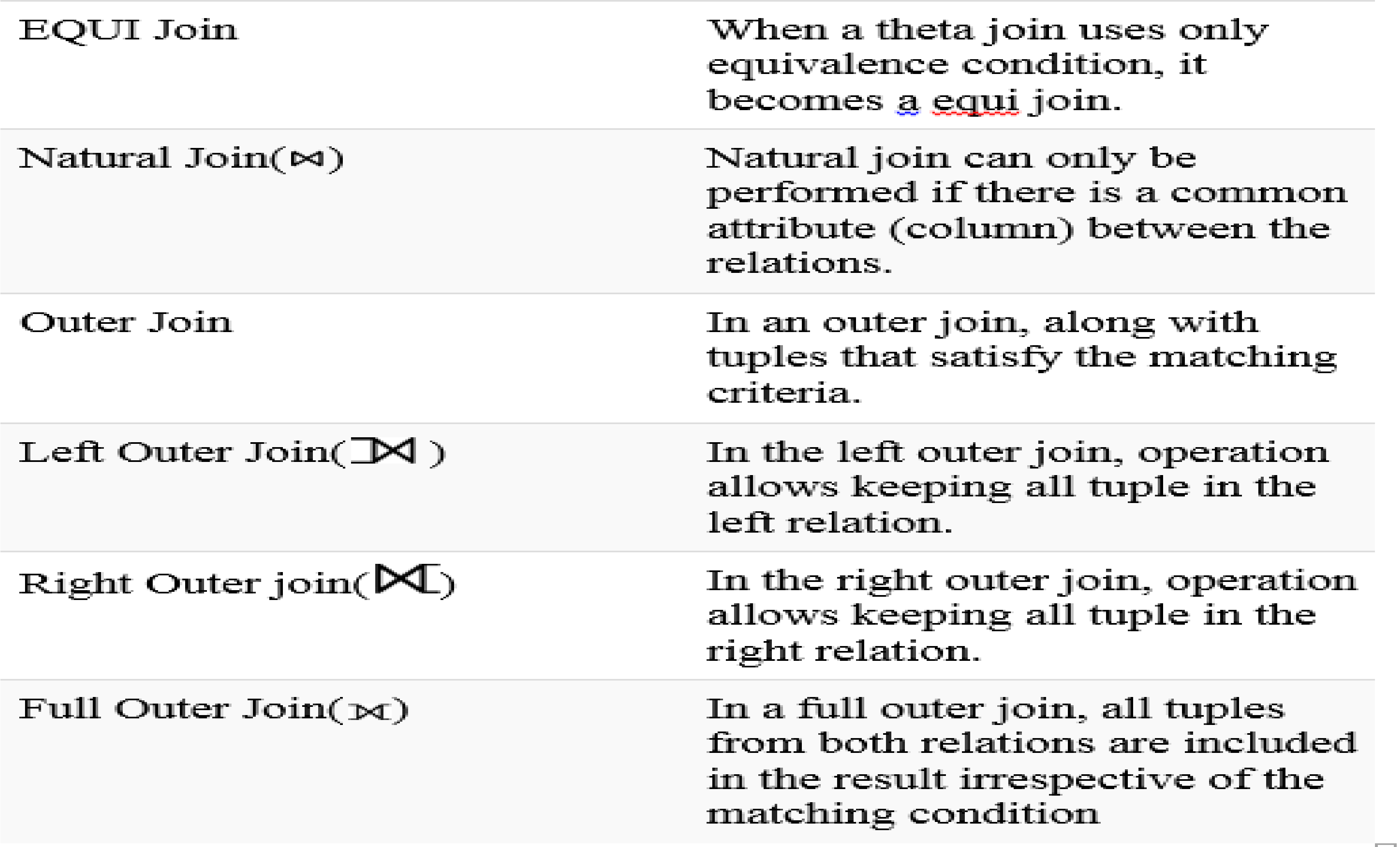
Lecture-15

Operation	Purpose
Select(σ)	The SELECT operation is used for selecting a subset of the tuples according to a given selection condition
Projection(π)	The projection eliminates all attributes of the input relation but those mentioned in the projection list.
Union Operation(∪)	UNION is symbolized by symbol. It includes all tuples that are in tables A or in B.
Set Difference(-)	- Symbol denotes it. The result of A - B, is a relation which includes all tuples that are in A but not in B.
Intersection(∩)	Intersection defines a relation consisting of a set of all tuple that are in both A and B.

Lecture-15

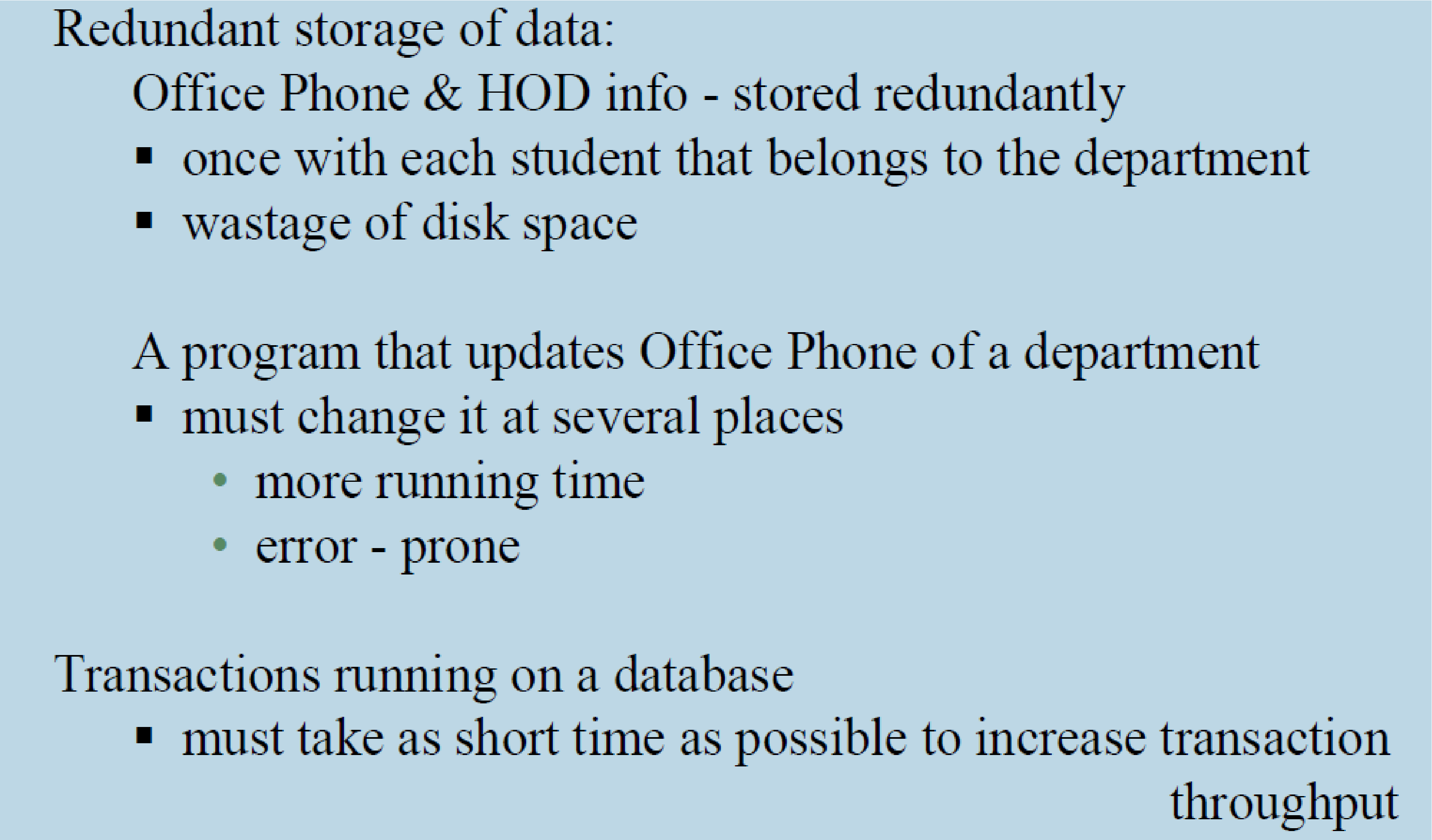
Cartesian Product(X)	Cartesian operation is helpful to merge columns from two relations.
Inner Join	Inner join, includes only those tuples that satisfy the matching criteria.
Theta Join(θ)	The general case of JOIN operation is called a Theta join. It is denoted by symbol θ.

Lecture-15



EQUI Join	When a theta join uses only equivalence condition, it becomes a equi join.
Natural Join(⋈)	Natural join can only be performed if there is a common attribute (column) between the relations.
Outer Join	In an outer join, along with tuples that satisfy the matching criteria.
Left Outer Join(⟕)	In the left outer join, operation allows keeping all tuple in the left relation.
Right Outer join(⟖)	In the right outer join, operation allows keeping all tuple in the right relation.
Full Outer Join(⟗)	In a full outer join, all tuples from both relations are included in the result irrespective of the matching condition

Relational database design: pitfalls



Redundant storage of data:

Office Phone & HOD info - stored redundantly

- once with each student that belongs to the department
- wastage of disk space

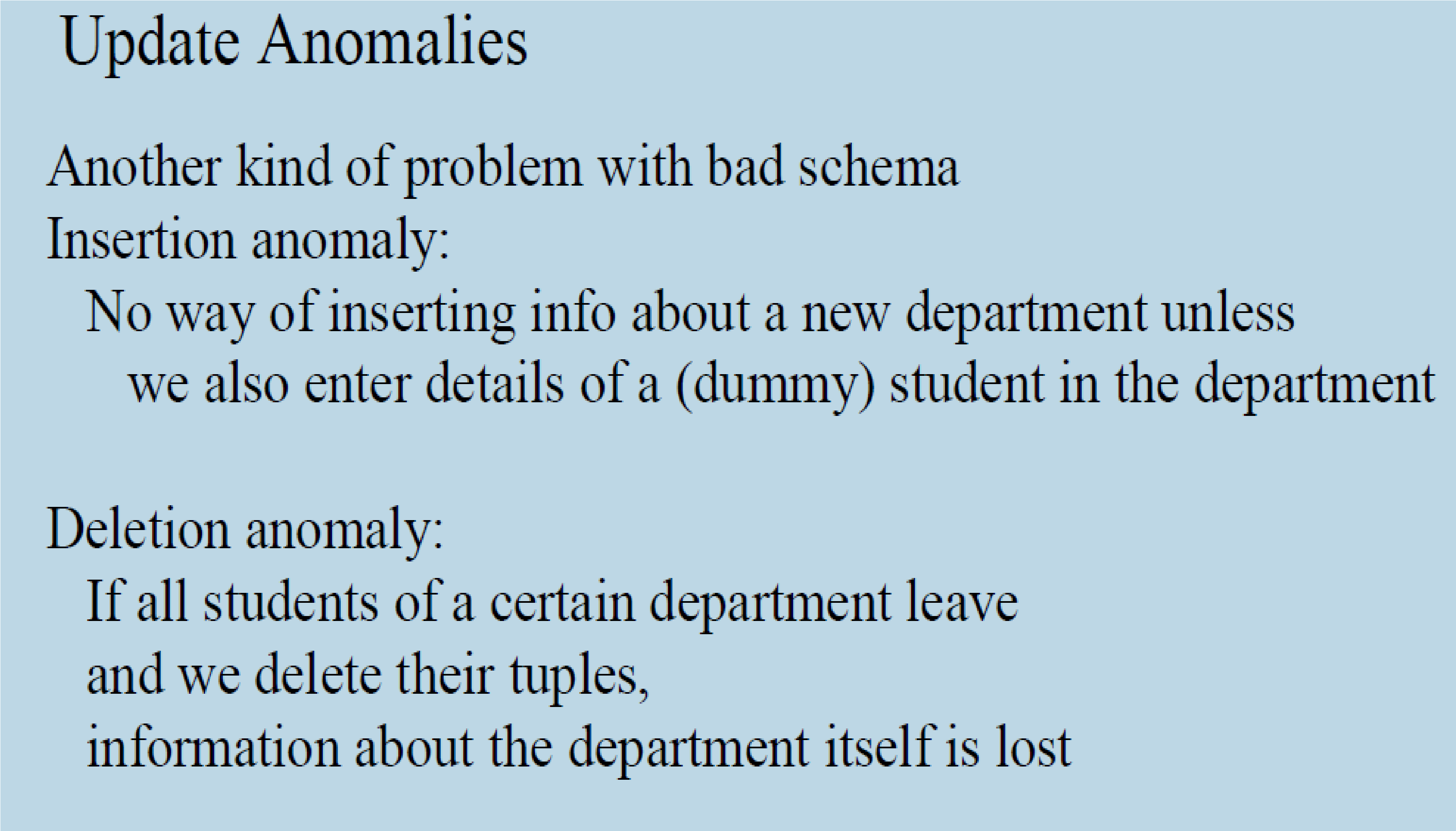
A program that updates Office Phone of a department

- must change it at several places
  - more running time
  - error - prone

Transactions running on a database

- must take as short time as possible to increase transaction throughput

Relational database design: pitfalls



Update Anomalies

Another kind of problem with bad schema

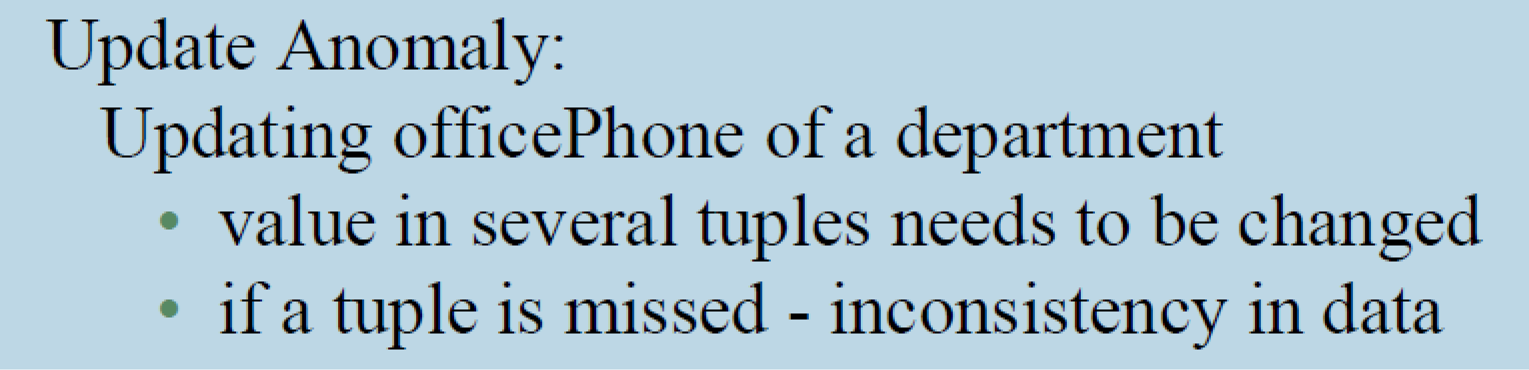
Insertion anomaly:

No way of inserting info about a new department unless we also enter details of a (dummy) student in the department

Deletion anomaly:

If all students of a certain department leave
and we delete their tuples,
information about the department itself is lost

Relational database design: pitfalls



Update Anomaly:

Updating officePhone of a department

- value in several tuples needs to be changed
- if a tuple is missed - inconsistency in data

Lecture-15

(Assignment)

Discuss Relational database design: pitfalls.

Thank You