NATIONAL UNIVERSITY
COLLEGE OF COMPUTING AND
INFORMATION TECHNOLOGIES

VER 1.0

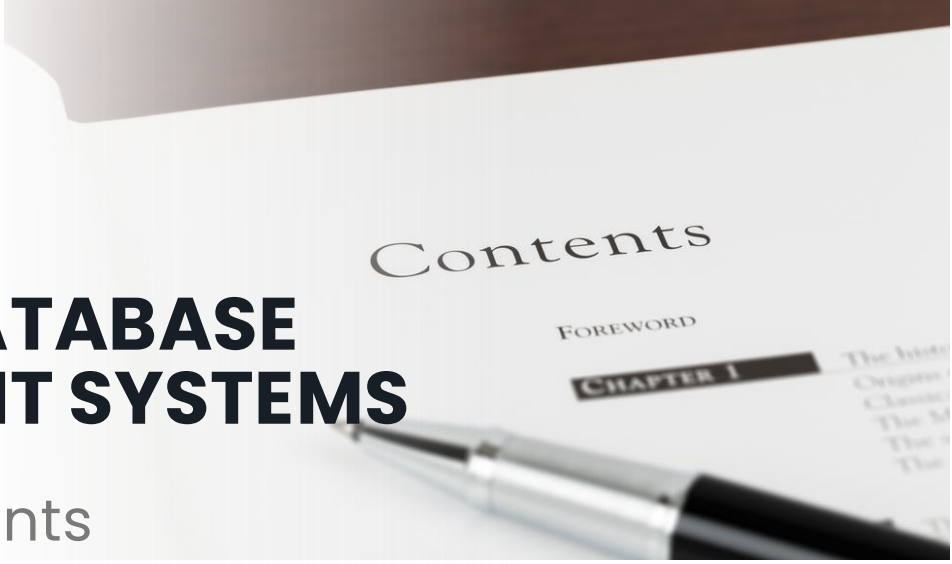
# Course MANUAL

## DATABASE SYSTEMS

This course provides students with the theoretical knowledge and practical skills in the utilization of databases and database management systems in the ICT applications. The logical design, physical design and implementation of relational databases are all covered in this course.

INFORMATION TECHNOLOGY

JULIE ANNE ANGELES-CRYSTAL




# ADVANCE DATABASE MANAGEMENT SYSTEMS

## Table of Contents

MODULE

#3

NATIONAL UNIVERSITY

## MODULE GOALS

- Indexing and Views
- Clustered and Non-clustered Index
- Creating and Managing Views

FLEX Course Material

# *Indexing and Views*

In MS SQL Server, **Indexing** and **Views** play crucial roles in enhancing database performance and simplifying data retrieval.

- **Indexing** helps speed up search queries by creating a data structure that allows SQL Server to quickly locate data.
- **Views** are virtual tables created by storing a SQL query, allowing users to access complex queries as if they were simple tables.

Education that works.

LESSON #

#3

# Indexing

**What is an Index?**

•An index is a data structure that improves the speed of data retrieval.

•It works like an index in a book, helping SQL Server locate rows in a table without scanning every row.

**Types of Indexes:**

**1.Clustered Index:**

1. Sorts and stores rows in the table based on the indexed column.
2. There can be only one clustered index per table.

**2.Non-Clustered Index:**

1. Contains pointers to rows in the data pages.
2. A table can have multiple non-clustered indexes.

**3.Unique Index:**

1. Ensures that all values in the indexed column(s) are unique.

**Creating Indexes:**

```sql
-- Creating a Clustered Index
CREATE CLUSTERED INDEX IX_Employee_ID
ON Employees(EmployeeID);

-- Creating a Non-Clustered Index
CREATE NONCLUSTERED INDEX IX_Employee_Name
ON Employees(LastName);
```

**Dropping Indexes:**

```sql
-- Dropping an Index
DROP INDEX IX_Employee_Name ON Employees;
```

LESSON #

#3

# Views

**What is a View?**

•A view is a virtual table based on a SQL query.

•It does not store data itself but shows data from one or more tables.

•Views simplify complex queries by presenting results as if they were from a table.

**Creating a View:**

```
CREATE VIEW EmployeeDetails AS
SELECT EmployeeID, FirstName, LastName, Department
FROM Employees
WHERE Status = 'Active';
```

**Querying a View:**

```
SELECT * FROM EmployeeDetails;
```

**Updating a View:**

```
ALTER VIEW EmployeeDetails AS
SELECT EmployeeID, FirstName, LastName, Department, Salary
FROM Employees
WHERE Status = 'Active';
```

**Dropping a View:**

```
DROP VIEW EmployeeDetails;
```

LESSON #

#3

# Practice Activity

## Practice Activities

### 1.Index Practice:

1. Create a **clustered index** on the Products table using the ProductID column.
2. Add a **non-clustered index** on the ProductName column.
3. Drop an index and verify using system views.

### 2.View Practice:

1. Create a view named ActiveOrders to display all orders with status 'Processing'.
2. Modify the view to include customer names by joining with the Customers table.
3. Retrieve data from the view.

### 3. Exploration:

1. Use the following query to check existing indexes:

```
SELECT * FROM sys.indexes
WHERE object_id = OBJECT_ID('Employees');
```

1. List all views in the database:

```
SELECT * FROM INFORMATION_SCHEMA.VIEWS;
```

LESSON #

#3

# Assessment Questions

1.What is the main difference between a clustered and non-clustered index?

2.How many clustered indexes can a table have?

3.Write an SQL command to create a view named HighSalaryEmployees showing employees with a salary above 50,000.

4.How can you modify a view to add a new column?

5.Why would you use an index in SQL Server?

LESSON #

#3

# Assessment Answers

1. **What is the main difference between a clustered and non-clustered index?**

   A **clustered index** determines the physical order of data in a table and stores data rows in sorted order based on the indexed column. A **non-clustered index**, on the other hand, creates a separate structure that holds pointers to the data, allowing multiple non-clustered indexes on a table.

2. **How many clustered indexes can a table have?**

   A table can have only **one clustered index** because the data rows can be stored in only one sorted order.

3. **Write an SQL command to create a view named HighSalaryEmployees showing employees with a salary above 50,000.**

   ```
   CREATE VIEW HighSalaryEmployees AS SELECT EmployeeID, FirstName, LastName, Salary FROM Employees WHERE Salary > 50000;
   ```

4. **How can you modify a view to add a new column?**

   ```
   ALTER VIEW HighSalaryEmployees AS SELECT EmployeeID, FirstName, LastName, Salary, Department FROM Employees WHERE Salary > 50000;
   ```

5. **Why would you use an index in SQL Server?**

   Indexes improve query performance by reducing the time it takes to retrieve data from tables. They optimize search operations, enhance sorting, and help speed up joins and filtering conditions.

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Objectives**

By the end of this activity, students will be able to:

- Create and use **Clustered and Non-clustered Indexes.**
- Optimize queries using **Composite and Covering Indexes**.
- Work with **Views**, including **Simple, Updatable, and Indexed Views**.

**Part 1: Setting Up the Database and Table**

**Step 1: Create a New Database**

1. Open **SQL Server Management Studio (SSMS)**.
2. Click **New Query** and enter:

```
CREATE DATABASE CompanyDB;
```

3. Execute the query by clicking **F5** or the **Execute button**.
4. Select the new database to work with:

```
USE CompanyDB;
```

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Part 2: Creating the Employees Table**

**Step 2: Create an Employees Table**

```
CREATE TABLE Employees (
    EmployeeID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Department NVARCHAR(50),
    Salary DECIMAL(10,2),
    Email NVARCHAR(100)
);
```

**Part 3: Inserting Sample Data**

**Step 3: Insert Sample Employees**

1.Run the following query to add employee records:

```
INSERT INTO Employees (FirstName, LastName, Department, Salary, Email)
VALUES
('Aria', 'Angeles', 'HR', 60000, 'aria@email.com'),
('Klay', 'Santos', 'IT', 70000, 'klay@email.com'),
('Noel', 'Santos', 'Finance', 55000, 'noel@email.com'),
('Yuri', 'Hanamitchi', 'IT', 80000, 'yuri@email.com'),
('Jasper Jean', 'Mariano', 'HR', 62000, 'jayjay@email.com'),
('Kiefer', 'Watsons', 'Finance', 58000, 'kiefer@email.com');
```

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Part 4: Indexing**

**Step 4: Create a Clustered Index**

1. Run the following SQL:

```
CREATE CLUSTERED INDEX IDX_EmployeeID
ON Employees(EmployeeID);
```

TASK: Run the following **before and after** adding the index to compare execution time:

```
SELECT * FROM Employees
WHERE EmployeeID = 3;
```

**Step 5: Create a Non-Clustered Index**

1.Run this SQL to speed up searches on **LastName**

```
CREATE NONCLUSTERED INDEX IDX_LastName
ON Employees(LastName);
```

TASK: Run the following query before and after indexing

```
SELECT * FROM Employees
WHERE LastName = 'Watsons';
```

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Step 6: Create a Composite Index**

Create an index on **Department and Salary**

```
CREATE NONCLUSTERED INDEX IDX_DeptSalary
ON Employees(Department, Salary);
```

TASK: Run and Compare

```
SELECT * FROM Employees
WHERE Department = 'IT' AND Salary > 60000;
```

**Step 7: Create a Covering Index**

Create an index that covers **FirstName and LastName** when searching by Department:

```
CREATE NONCLUSTERED INDEX IDX_Covering
ON Employees(Department) INCLUDE (FirstName, LastName);
```

TASK: Run the query before and after theindexing:

```
SELECT FirstName, LastName
FROM Employees
WHERE Department = 'HR';
```

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Step 8: Create a Unique Index**

Ensure that **Email** is unique

```
ALTER TABLE Employees
ADD CONSTRAINT UQ_Email UNIQUE (Email);
```

TASK: Try inserting a duplicate email to see if it prevents it.

**Part 5: Creating Views**

**Step 9: Create a Simple View**

Create a view to show **high salary employees:**

```
CREATE VIEW vw_HighSalary AS
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employees
WHERE Salary > 60000;
```

TASK: Run:

```
SELECT * FROM vw_HighSalary;
```

Insert a new employee with **Salary > 60,000** and check if they appear in the view.

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Step 10: Create a View with Aggregation**

Create a view for **average salary per department**:

```
CREATE VIEW vw_AvgSalary AS
SELECT Department, AVG(Salary) AS AvgSalary
FROM Employees
GROUP BY Department;
```

**Step 11: Create an Updatable View**

Create a **filtered view** for IT employees

```
CREATE VIEW vw_ITEmployees AS
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employees
WHERE Department = 'IT'
WITH CHECK OPTION;
```

TASK: Try updating an employee's salary in this view:

```
UPDATE vw_ITEmployees
SET Salary = 90000
WHERE EmployeeID = 2;
```

Try inserting an employee from **HR** in the view. Does it work? Why not?

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Step 12: Create an Indexed View**

**Create a view with indexing:**

```
CREATE VIEW vw_IndexedSalary WITH SCHEMABINDING
AS
SELECT Department, COUNT_BIG(*)
AS EmpCount, AVG(Salary) AS AvgSalary
FROM dbo.Employees
GROUP BY Department;
```

**Add a Clustered Index:**

```
CREATE UNIQUE CLUSTERED INDEX IDX_vw_IndexedSalary
ON vw_IndexedSalary(Department);
```

TASK: Run and Compare query speed before and after.

```
SELECT * FROM vw_IndexedSalary
WHERE Department = 'HR';
```

LESSON #

#3

# Laboratory Activity: Indexes and Views in MS SQL Server

**Part 6: Joins and Indexes**

**Step 13: Optimize Joins**

1. Create a Departments Table:

```
CREATE TABLE Departments (
    DeptID INT PRIMARY KEY,
    DeptName NVARCHAR(50)
);
```

2. Insert sample data.

```
INSERT INTO Departments
VALUES (1, 'HR'), (2, 'IT'), (3, 'Finance');
```

3. Add DeptID to Employees.

```
ALTER TABLE Employees ADD DeptID INT;
```

4. Assign DeptID values.

```
UPDATE Employees SET DeptID = 1 WHERE Department = 'HR';
UPDATE Employees SET DeptID = 2 WHERE Department = 'IT';
UPDATE Employees SET DeptID = 3 WHERE Department = 'Finance';
```

5. Optimize the join:

```
CREATE NONCLUSTERED INDEX IDX_DeptID
ON Employees(DeptID);
```

TASK: Run and Compare:

```
SELECT e.EmployeeID, e.FirstName, e.LastName, d.DeptName
FROM Employees e
JOIN Departments d ON e.DeptID = d.DeptID;
```