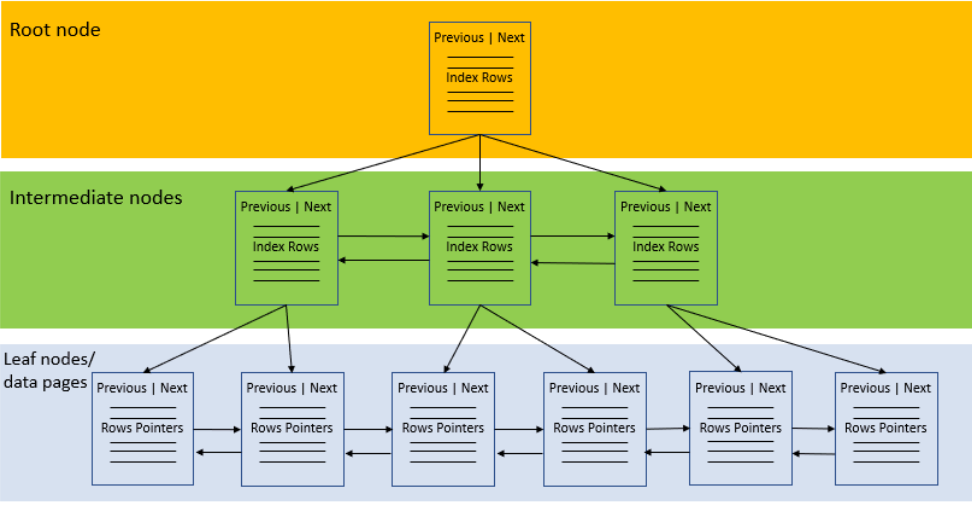


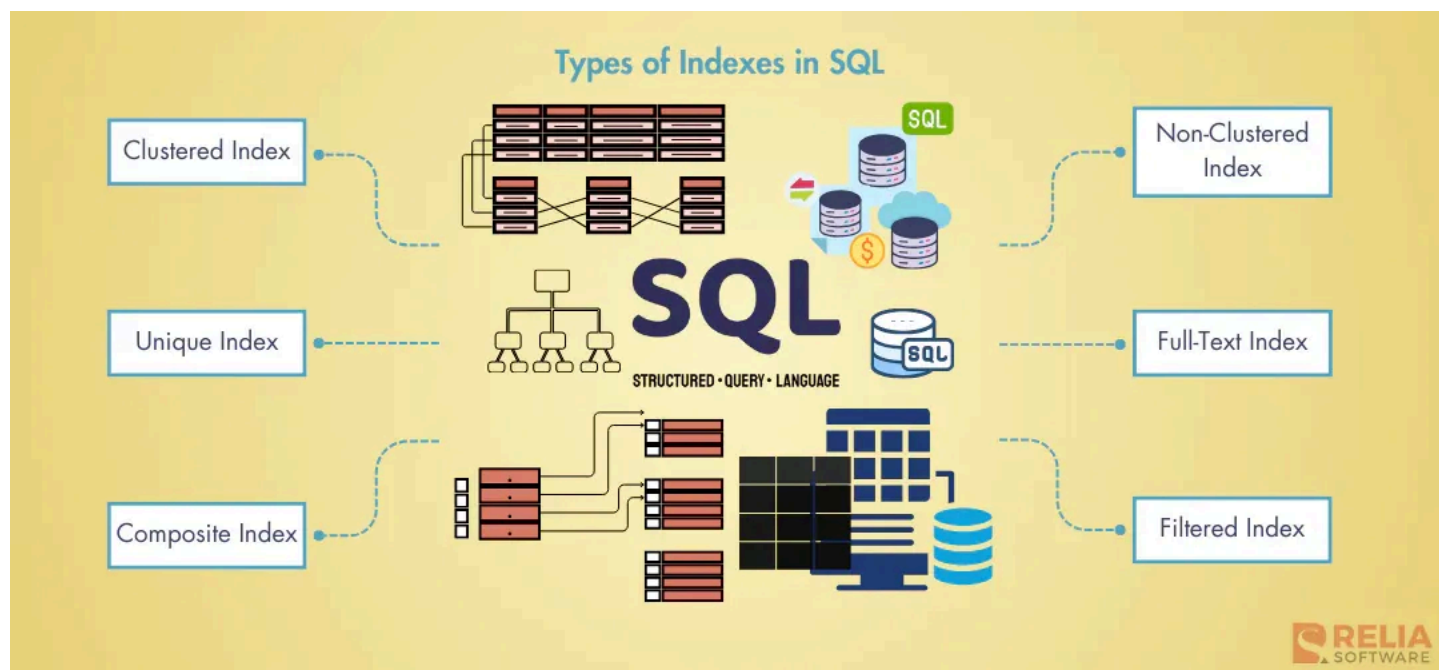
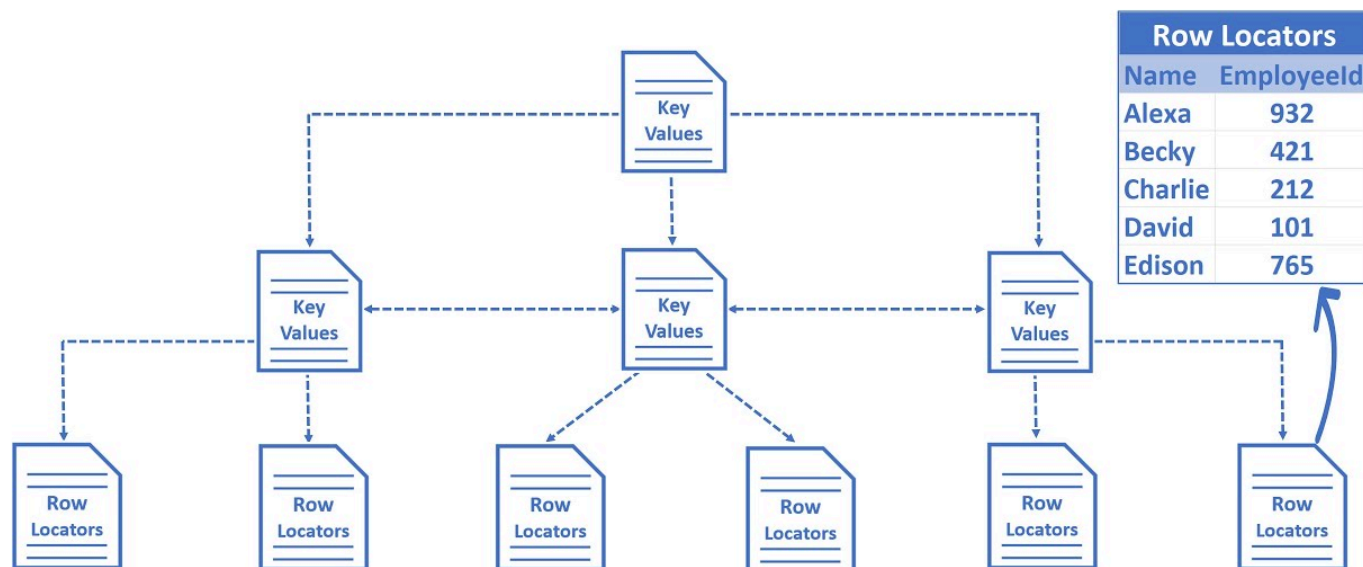
## Minutes of session for Day 14 : Working with Views, Indexing & Normalization

Day 14	Working with Views & Indexing	<p><b>Views</b></p> <p>A Virtual table that is based on the result set of an SQL Query. It doesn't store data physically, but provides a way to look at data from one or more tables.</p> <pre>CREATE VIEW StudentCourses AS SELECT s.StudentName, c.CourseName FROM Student s JOIN Enrollment e ON s.SudentID = e.StudentID JOIN Courses c ON e.CourseID = c.CourseID;</pre> <p>Update data using a View</p> <p>Dropping a view</p>
		<p><b>Indexing</b></p> <p>Indexing helps in speeding up the retrieval of rows by using a pointer.</p>  <p>Data is sorted in sorted order by the index key values</p>
		<p><b>Clustered index:</b> Stores the actual data rows at the leaf level of the index.</p> <p><b>Non-clustered index :</b> Contains pointers to the actual data</p> <p><b>Unique index :</b> Ensures all the values in the index column are unique</p>
	Normalizations	<p><b>Normalizations</b></p> <p>Normalization is the process of organizing data to reduce redundancy and improve data integrity.</p> <p>First normal form : Atomic values   One column per phone number   no manual data for calculated values like subtotal , current age</p> <p>We remove multivalues attributes</p> <p>Second normal form : 1NF + No partial dependency : Create separate tables for courses as per ex</p> <p>Third normal form : 2NF + Transitive dependency :</p>

Ensure all non- key attributes depend only on the primary key.

Benefits of normalization and when to apply it

# How do SQL Indexes Work



How can **performance of a SQL Query** be improved ?

1. Indexing :

2. Avoid Selecting \*(Select only required Columns)
3. Always Use Where Clause to filter rows early
4. Use Joins Appropriately, Avoid unnecessary joins(Always go with inner joins as compared to Left join).
5. Create and Use Indexed Views(Materialized Views)
6. Use Exist instead of IN for subqueries( Very efficient for large datasets)
7. Avoid using Cursors(replace them with set-based operations), triggers where possible.
8. Use Appropriate Data Types ex Varchar(1000) for phone number)
9. Enable Query Caching
10. Query Execution PLaN(SSMS)( CTRL +M)

Performance : Faster response time and throughput.

Storage: Efficient use of disk and memory

Scalability: Handle large dataset with ease.

Maintainability: Easier to debug and optimize.

Cost : Lower H/W requirement and licensing cost.

SELECT Name FROM Employees Where DepartmentID IN( SELECT DEpartmentID FROM Department);	SELECT Name FROM Employees Where EXIST ( SELECT 1 FROM Departments Where Employees.DepartmentID = Departments.DepartID);
	Here EXISTS stops scanning once the match is found

Limitations with views in MSSQL Server :

No Parameters Support	Views can not accept parameters like Stored procedures	So we use TVF or Stored Procedures for parameterized data retrieval.
Read-Only in many cases	Views involving joins, aggregation or grouping are often not updatable.	Use <b>INstead of Triggers</b> on view to allow updates , insert or delete.
Performance Overhead	Complex views( Nested views) can degrade performance	Use Indexed Views with proper indexing to improve performance
Dependency management	Dropping base table will break views and views don't update schema if base table changes	Use SSSMS dependency tracking or write scripts to validate objects dependencies.
Limited Metadata support	They don't support comments or rich metadata like procedure and tables	Maintain external documentation or extended property for comments
Can not handle Complex logic	Loops, error handling and conditional execution are not supported	We have to use Procedure and function for that
No Dynamic filtering	Filtering data based on user input is	We have to use TVF

	not supported	
View Definition limit	It is limited to 8192 char in view definition	Break logic into multiple views or use SP for larger queries.

## Before Normalization

### Employee\_Department

Emp_ID	Emp_Name	Department	Dept_Location	Emp_Skills
101	Nick Wise	HR	London	Recruitment, Payroll
102	John Cader	Finance	Australia	Budgeting
103	Lily Case	HR	London	Recruitment
104	Ford Dawid	IT	Chicago	Programming, Testing

## Problems in the Employee\_Department Relation

### 1. Insertion Anomaly:

- If a new department is created but no employee is assigned to it yet, we cannot store its location because we need an employee record to insert.

### 2. Update Anomaly:

- If the location of the HR department changes, we must update it in multiple rows (for both Nick Wise and Lily Case). If one row is missed, the data becomes inconsistent.

### 3. Deletion Anomaly:

- If all employees in the IT department leave, we lose the department information, including its location.

### 4. Data Redundancy:

- The department location is repeated for every employee in the same department.

## After Normalization

Employee

Emp_ID	Emp_Name	Dept_ID
101	Nick Wise	D1
102	John Cader	D2
103	Lily Case	D1
104	Ford Dawid	D3

Department

Dept_ID	Department	Dept_Location
D1	HR	London
D2	Finance	Australia
D3	IT	Chicago

Employee\_Skills

Emp_ID	Emp_Skills
101	Recruitment
101	Payroll
102	Budgeting
103	Recruitment
104	Programming
104	Testing