View in SQL:
In SQL, a view is a virtual table based on the result-set of an SQL statement. Views do not store any data of their own but display data stored in other tables.
A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

Syntax:
create a view

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

update a view

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

drop a view

```
DROP VIEW view_name;
```

Stored Procedure in SQL:
A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.
So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

Stored Procedure vs View

A view is mostly used when only a SELECT statement is needed. Views should be used to store commonly-used JOIN queries and specific columns to build virtual tables of an exact set of data we want to see. Stored procedures hold the more complex logic, such as INSERT, DELETE, and UPDATE statements to automate large SQL workflows.

Stored Procedure vs Trigger:

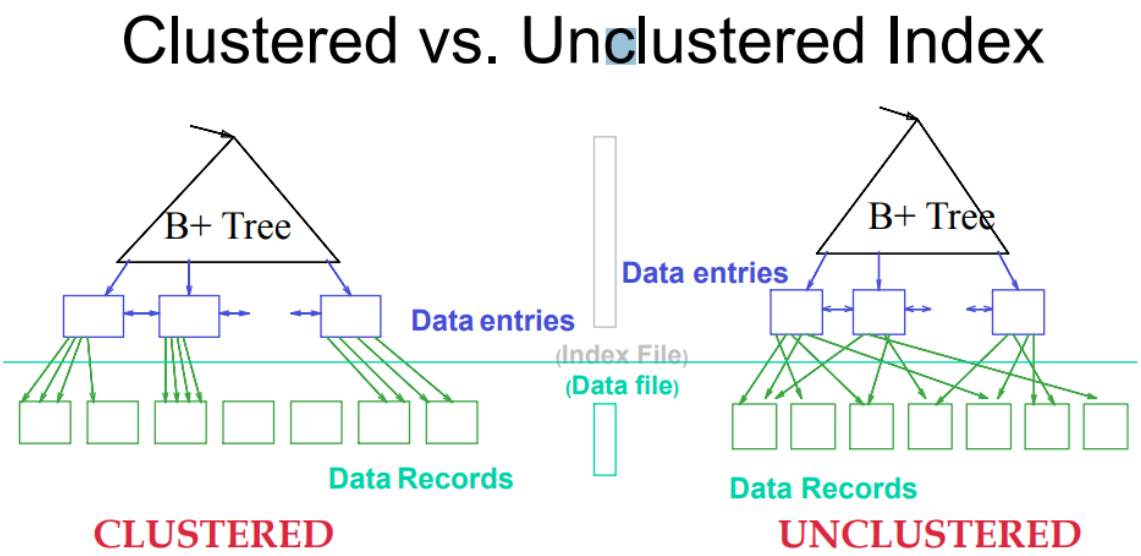
A trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete). A stored procedure is invoked by calling it explicitly.

View vs Materialized View:

Materialized views are **disk based** and are updated periodically based upon the query definition. They are similar to regular views, in that they are a logical view of your data (based on a select statement), however, the **underlying query result set has been saved to a table**. The upside of this is that when you query a materialized view, **you are querying a table**, which may also be indexed. The **downside though is that the data you get back from the materialized view is only as up to date as the last time the materialized view has been refreshed**.

Views are **virtual only** and run the query definition each time they are accessed. The upside of a view is that it will **always return the latest data to you**. The **downside of a view is that its performance** depends on how good a select statement the view is based on.

B+ tree vs B Tree



They are both self-balancing trees that sort the internal nodes in order.

In a B tree search keys and data are stored in internal or leaf nodes. But in a B+-tree data is stored only in leaf nodes.

Full scan of a B+ tree is very easy because all keys are found in leaf nodes. Full scan of a B tree requires a full traversal since keys might not be at the leaf level.
In a B tree, data may be found in leaf nodes or internal nodes. Deletion of internal nodes is very complicated. In a B+ tree, data is only found in leaf nodes. Deletion of leaf nodes is easy.
Insertion in B-tree is more complicated than B+ tree.
B+ trees store redundant search keys but B-tree has no redundant value.
In a B+ tree, leaf node data is ordered as a sequential linked list but in a B tree the leaf node cannot be stored using a linked list. Many database systems' implementations prefer the structural simplicity of a B+ tree.