**FEATURES IN SQL SERVER 2016**

* **Always Encrypted**

Enabling Always Encrypted Feature present under the Security Tab will make the SQL Server data always encrypted within the SQL Server. The encrypted data will be available only to the client application owners using the keys. These keys is never passed to the SQL Server, which means Database or System administrators cannot peek on the client sensitive data. The clients can peacefully store their confidential data in the cloud-managed database with the Always Encrypted Feature enabled free of unauthorized access.

* **Dynamic Data Masking**

This new feature in SQL Server 2016, is for those purposes where only some part of the SQL Server data needs to be hidden for security while some data remains visible to other users. Enabling Dynamic Data Masking feature in SQL Server limits sensitive data exposure by masking it to unauthorized users. It is also a part of preventing unauthorized access to sensitive data by allowing users to decide how much percentage of data should be revealed with less impact on application layer. This feature can be set up using the masking rules defined by the users. Example: while storing the credit card numbers or phone numbers, rules can be setup like first four digits should be visible and remaining digits should be hidden.

* **Support for JSON**

SQL Server 2016 has a new feature that provides support for JSON, Java Script Object Notation. This feature allows the JSON data to be exchanged between applications and SQL Server database engine. JSON is based on subset of JavaScript programming language and is a human readable that is also easy for computer to parse and generate. With this support, Microsoft allows SQL Server to parse JSON formatted data allowing it to store in a relation format. In addition to this, it also allows relational data to turn it into JSON formatted data. This feature has been aimed to make the applications easy to exchange JSON data with SQL Server.

* **Row Level Security**

This new feature in SQL Server 2016 provides Row level Security that allows SQL Server to restrict which users can view what data in a table based on a function. It has been helpful in case of multi-tenant environments where data access needs to be keep in limited access based on customer ID. Restriction of Rows can be done by filter predicates defined in inline table-value function and security rules will make the filter predicates execute for every SELECT or DELETE operation. Enabling the feature at database layer does not require application developers to maintain code for restricting data from some logins while allowing other logins to access all data. Though this feature has been already supported by other databases, SQL Server introduces it for the first time considering the need to provide security to row data.

* We can secure on a complete row
* **Temporal Table**

Temporal table is defined as a table holding old versions of rows within a base table. If the database is having temporal tables, SQL Server 2016 can automatically manage shifting old row versions to the temporal table whenever it is updated. Users who are planning to manage row versioning might find this modification quite helpful.

* It will keep track the current and historical data in database
* Data can be restored without down time
* It will provides ability to query the table snapshot at any point in time
* **Polybase :**
* We can query relational and non-relational data

Example : SQL + Big Data

**MSSQL IN SPRINGBOOT:**

## **1. Declare dependency for SQL Server JDBC Driver**

<**dependency**>

    <**groupId**>com.microsoft.sqlserver</**groupId**>

    <**artifactId**>mssql-jdbc</**artifactId**>

    <**scope**>runtime</**scope**>

</**dependency**>

## **2. Specify Data Source Properties**

spring.datasource.url=jdbc:sqlserver://sqlsrv\\sqlexpress;databaseName=customer

spring.datasource.username=username

spring.datasource.password=password

## **3.Connect to SQL Server with Spring Data JPA**

|  |  |
| --- | --- |
| 1  2  3  4 | <**dependency**>      <**groupId**>org.springframework.boot</**groupId**>      <**artifactId**>spring-boot-starter-data-jpa</**artifactId**>  </**dependency**> |

spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect. org.hibernate.dialect.SQLServer2008Dialect

#CREATES TABLE

spring.jpa.generate-ddl=true