Index Tuning Basics

Index

What is index? (w.r.t tables in database)

Whenever we create a primary key constraint, an index is created.

That index is also a clustered index

Index has 2 columns

Key - offset

Index will be useful only to speed up the query.

Only when you search based on condition that involves indexed column

Index will be a burden for DML

When your select query returns <= 2% of the total records present in the table,

Then only your index is useful.

Sometimes, people create index like

CREATE INDEX idxJag1 ON UPPER(HR.EMPLOYEES.LAST\_NAME);

Types of index:

Clustered Index

Clustered index is an index, which influence the order of the rows of the table.

A table rows are always in the order of the clustered index.

So only 1 clustered index is allowed per table.

Non-Clustered Index

There can be more than 1 non-clustered index per table. Original rows order can be different.

CREATE INDEX idxStatusJag ON ACCOUNT(ASTATUS);

DROP INDEX IDXSTATUSJAG;

Reminder:

Design pattern types

Creational

Structural

Behavioral

Creational

Helps us in what are all the ways an object of a class can be created.

Employee e1=new Employee();

The way the objects are created or obtained is specified in creational pattern.

Different types of creational pattern:

Singleton

Prototype

Factory

Builder

Observer

Singleton

Allows only one instance of a class.

That instance can be obtained by using a static method

getInstance()

public class Singleton

{

private Singleton instance; //=new Singleton(); //early

private Singleton()

{}

public static Singleton getInstance()

{

if(instance==null)

instance=new Singleton(); //lazy

return instance;

}

}

Prototype

Cloning objects from existing objects

Factory

Instead of

ArrayList<Integer> marks=new ArrayList<>();

Doing like

List<Integer> marks=new ArrayList<>();

marks=new LinkedList<>();

public void displayAll(List<Integer> m){}

Activity:

Understand the different types of Creational Patterns

public void create(){}

public void passivate(){}

public void activate(){}

public void destroy(){}

interface EntityBean

public class EntityAdapter implements EntityBean

{

public void create(){}

public void passivate(){}

public void activate(){}

public void destroy(){}

}

public class MyAdapter extend EntityAdapter

{

}

Activity:

Identify various types of Structural Patterns

Adapter

Bridge

Composite

Decorator

Proxy

Behavioural Pattern:

Chain of responsibility

Iterator

Command

Observer

JDBC

Java DataBase Connectivity

Java program access the database.

Which database?

In our example, PostgreSQL

Database vendors like Oracle corporation, Microsoft corporation, IBM corporation, etc

Not only they provide db product, they also provide java driver programs.

Postgresql driver

Steps for java program to access postgresql:

1. Load the driver (this step we can ignore in new java versions)
2. Connect to the db URL, username, password
3. Create a statement object (that represents sql statement)
4. Execute the statement
5. Display the result
6. Create a maven project
7. In pom.xml add the dependency:

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.5.1</version>

</dependency>

1. In App.java

**package** com.ust.demo;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

/\*\*

\* Hello world!

\*

\*/

**public** **class** App

{

**public** **static** **void** main( String[] args ) **throws** SQLException

{

//load the driver (skip)

Connection con = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres","postgres","password");

Statement st = con.createStatement();

ResultSet rs = st.executeQuery("SELECT \* FROM BRANCH");

**while**(rs.next())

System.***out***.println(rs.getString(1)+"\t"+rs.getString(2)+"\t"+rs.getString(3));

}

}

Ensure that you have used the correct password.

Run the app.