***DBT Interview Preparation***

***Created by Sanket Patil***

**RDBMS**

A relational database management system (RDBMS or just RDB) is a common type of database that stores data in tables, so it can be used in relation to other stored datasets.

|  |  |
| --- | --- |
| **DBMS** | **RDBMS** |
| DBMS applications store **data as file**. | RDBMS applications store **data in a tabular form**. |
| **Normalization is not** present in DBMS | **Normalization is** present in RDBMS. |
| DBMS uses file system to store data, so there will be **no relation between the tables**. | in RDBMS, data values are stored in the form of tables, so a  **relationship** between these data values will be stored in the form of a table as well. |
| DBMS **does not support distributed database**. | RDBMS  **supports distributed database**. |
| DBMS is meant to be for small organization and **deal with small data**. it supports **single user**. | RDBMS is designed to **handle large amount of data**. It supports **multiple users**. |
| Examples of DBMS are file systems, **xml** etc. | Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc |

**Features:**  
- Provides data to be **stored in tables.**  
- Persists data in the **form of rows and columns.**  
- Provides facility **primary** key, to **uniquely** identify the rows.  
- Creates **indexes** for quicker data retrieval.  
- Provides a **virtual** table creation in which sensitive data can be stored and simplified query can be applied.(views)  
- Sharing a common column in two or more tables(primary key and foreign key).  
- Provides multi user accessibility that can be controlled by individual users.

**Can we alter two column at same time**

No

**Foreign Key cascade**

A foreign key with **cascade delete** means that if a record in the parent table is **deleted**, then the corresponding records in the child table will automatically be **deleted**. This is called a **cascade delete** in **SQL** Server.

**There are five major categories:**

• **DDL**: Data Definition Language

e.g. CREATE, ALTER, DROP, RENAME.

• **DML**: Data Manipulation Language

e.g. INSERT, UPDATE, DELETE.

• **DQL**: Data Query Language

e.g. SELECT.

• **DCL**: Data Control Language

e.g. CREATE USER, GRANT, REVOKE.

• **TCL**: Transaction Control Language

e.g. SAVEPOINT, COMMIT, ROLLBACK.

**1. What is mysql?**

MySQL is an open-source relational database management system (RDBMS)

**2. What is SQL?**

SQL stands for Structured Query Language, and it is used to communicate with the Database.

This is a standard language used to perform tasks such as retrieval, updation, insertion and deletion of data from a database.

## 3. What is a Candidate key?

*Candidate key is a single key or a group of multiple keys that uniquely identify rows in a table.*

For example, in the example that we took earlier, both *Id*and *Email*can act as a Candidate for the table as they contain unique and non-null values.

## 4. What are Composite keys?

*A Composite key is a Candidate key or Primary key that consists of more than one attribute.*

Sometimes it is possible that no single attribute will have the property to uniquely identify tuples in a table. In such cases, we can use a group of attributes to guarantee uniqueness. Combining these attributes will uniquely identify tuples in the table.

**5. What is a CHECK constraint?**

A CHECK constraint is used to limit the values that can be placed in a column. CHECK constraints are most often used to enforce domain integrity.

**6. What is a NOT NULL constraint?**

A not null constraint enforces that the column will not accept null values. Not null constraints are used to enforce domain integrity.

**7. What is a DEFAULT definition?**

A DEFAULT definition is used to add values into a column when values were omitted. The default value must be compatible with the data type of the column to which the DEFAULT definition applies.

**8. What is the difference between a HAVING clause and a WHERE clause?**

The HAVING clause specifies a search condition for a GROUP BY or an aggregate.

|  |  |  |
| --- | --- | --- |
|  | **Where** | **Having** |
| 1 | WHERE Clause can be used without GROUP BY Clause | HAVING Clause cannot be used without GROUP BY Clause |
| 2 | WHERE Clause implements in row operations | HAVING Clause implements in column operation |
| 3 | WHERE Clause cannot contain aggregate function | HAVING Clause can contain aggregate function |
| 4 | WHERE Clause can be used with SELECT, UPDATE, DELETE statement. | HAVING Clause can only be used with SELECT statement. |
| 5 | WHERE Clause is used before GROUP BY Clause | HAVING Clause is used after GROUP BY Clause |

**9. Difference between primary unique & foreign key.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Primary key** | **Unique key** | **Foreign key** |
| 1. | Primary key is a column of table Which uniquely identifies the rows in that table. | Unique key constraints also uniquely identifies the rows in that table. | It refers to the field in a table which is the primary key of another table. |
| 2. | Primary Key does not allow NULL Values. | Unique Key is like Primary Key, the only difference Unique allows single NULL value. | Foreign Key can have multiple NULL values |
| 3. | Each Table can have only one Primary Key. | Table can have multiple Unique Keys | Table can have multiple Foreign Keys. |
| 4. | Primary Key creates Clustered Index by default. | Unique Key creates Non-Clustered Index by default. | Foreign Key does not create any index by default |
| 5. | Primary key can be created on temporary tables and table variables. | Unique key can be created on temporary tables and table variables. | Foreign key cannot be created on temporary tables and table variables. |

**10. Difference between drop truncate delete**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Delete** | | | **Truncate** | **Drop** |
| 1. | Delete is a DML command | | | truncate is DDL command. | Drop is DDL command |
| 2. | delete statement can be used for deleting the specific data | | | Truncate can be used to delete the entire data of the table | DROP table query removes one or more table |
| 3. | Syntax:  Delete table tablename;  Delete from tablename where ? | | | Syntax:  Truncate table tablename; | Syntax:  Drop table tablename; |
| 4. | | It can be rollback | Truncate not rollback | | Drop not rollback |
| 5. | | It is slow | It is fastest | | It is faster |

**11. What is a difference between Commit, Rollback and Savepoint?**

**COMMIT**: Ends the current transaction by making all pending data changes permanent.

**ROLLBACK**: Ends the current transaction by discarding all pending data changes.

**SAVEPOINT**: Divides a transaction into smaller parts. You can rollback the transaction till a particular named savepoint.

**12. Joins and it's types**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

1. **INNER JOIN / Equijoin**: Returns records that have matching values in both tables
2. **Cross join / Cartesian Join** - The absence of join condition it takes **Cartesian** product of table.

That is no of rows in table A & no of rows in table B.

1. **Natural join**: Returns records that have matching values in both tables.
2. **Non-Equi join**: Returns records base on non-matching values in both tables.
3. **Self-join**: Returns records that have matching values in same (self) table.
4. **LEFT OUTER JOIN**: Return all records from the left table, and the matched records from the right table
5. **RIGHT OUTER JOIN**: Return all records from the right table, and the matched records from the left table
6. **FULL OUTER JOIN**: Return all records when there is a match in either left or right table.

**13. What is normalization?**

Normalization is a way of organizing the data in the database.

Normalization is used to remove redundant data from the database and to store non-redundant data into it.

Refer the link for better understanding.  
<https://beginnersbook.com/2015/05/normalization-in-dbms/>

Normalization rules divides larger tables into smaller tables and links them using relationships.

Without [Normalization](https://en.wikipedia.org/wiki/Database_normalization) in SQL, we may face many issues such as

1. ***Insertion anomaly***: It occurs when we cannot insert data to the table without the presence of another attribute
2. ***Update anomaly***:  It is a data inconsistency that results from data redundancy and a partial update of data.
3. ***Deletion Anomaly***: It occurs when certain attributes are lost because of the deletion of other attributes.

Database normal forms:

**1NF :**

* Each table cell should contain a single value.
* Each record needs to be unique.

**2NF :**

* Be in 1NF
* Single Column Primary Key

**3NF :**

* Be in 2NF
* Has no transitive functional dependencies

**BCNF:**

Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate**Key.

Sometimes is BCNF is also referred as **3.5 Normal Form.**

**4NF**

**5NF**6NF

**14. What is Denormalization?**

Denormalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. Note that denormalization does not mean not doing normalization. It is an optimization technique that is applied after doing normalization.

**15. What is indexing?**

Indexing is a data structure technique which allows you to quickly retrieve records from a database file. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of pointers for holding the address of the disk block where that specific key value stored.

reduced i/o cost.

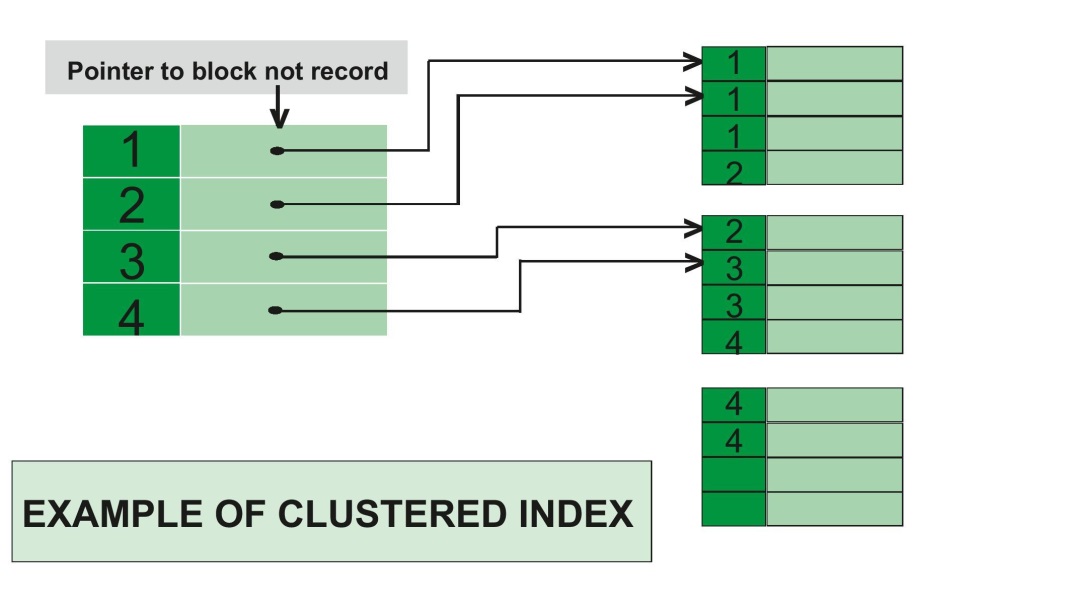
There are two types of indexing:

1. **Clustered Index:** A clustered index determines the physical order of data in a table. For this reason a table can have only one clustered index.

Primary key constraint create clustered indexes automatically if no index already exists on the table.

Eg. Clustered index is as same as dictionary where the data is arranged by alphabetical order.

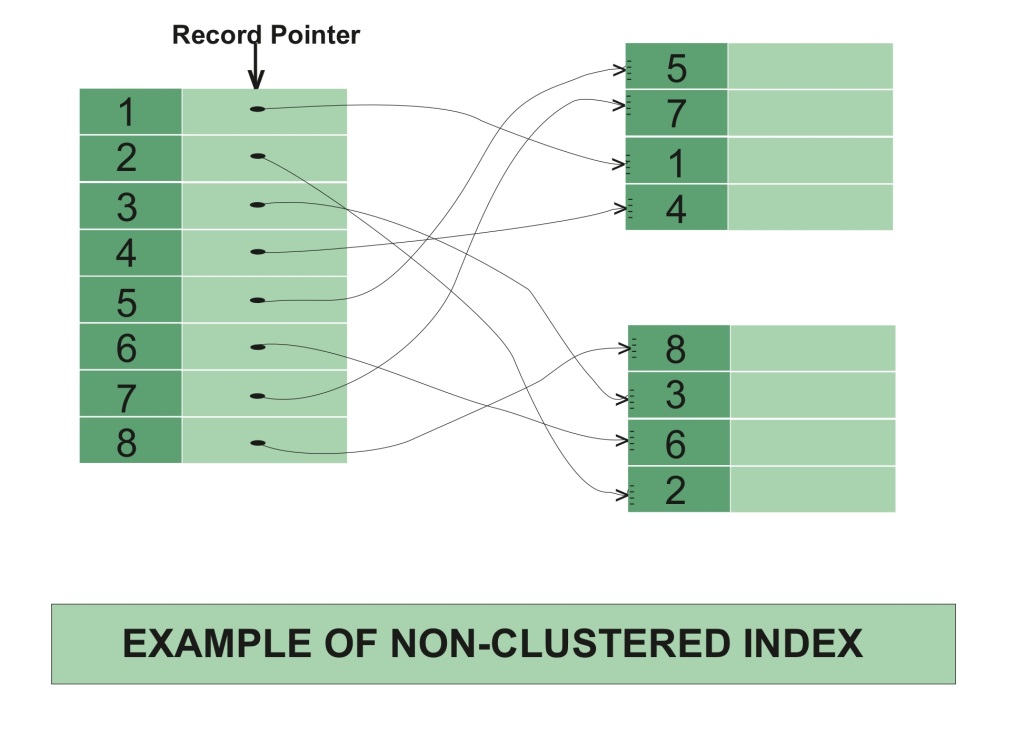
In clustered index, index contains pointer to block but not direct data.



1. **Non- Clustered Index:** Non-Clustered Index is similar to the index of a book. The index of a book consists of a chapter name and page number, if you want to read any topic or chapter then you can directly go to that page by using index of that book. No need to go through each and every page of a book.

The data is stored in one place, and index is stored in another place. Since, the data and non-clustered index is stored separately, then you can have multiple non-clustered index in a table.

In non-clustered index, index contains the pointer to data.



**16. What is Cursor?**

Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML operations on Table by User.

Cursors are used to store Database Tables. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors.

1. **Implicit Cursors:**

Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

1. **Explicit Cursors :**

Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

**17. Which are the steps for using an Explicit Cursor?**

There are four steps in using an Explicit Cursor.

1) DECLARE the cursor in the declaration section.

2) OPEN the cursor in the Execution Section.

3) FETCH the data from cursor into PL/SQL variables or records in the Execution Section.

4) CLOSE the cursor in the Execution Section before you end the PL/SQL Block.

**18. What is Trigger?**

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

A trigger uses the special table to keep a copy of the row which we have just inserted, deleted or modified.

**19. What are the different types of triggers?**

Basically we have 13 types of triggers.

**1. Row level triggers (6 types):** Row-level triggers execute once for each row in a transaction.

**2. Statement level triggers (6 types**): A statement level trigger is also called as table level

trigger. A table level trigger is a trigger that doesn't fire for each row to be changed.

**3. Instead of trigger**: Instead of trigger is use to write trigger on Views.

**Row Level Triggers:**

1. before insert on each row
2. before update on each row
3. before delete on each row
4. after insert on each row
5. after update on each row
6. after delete on each row

**Statement Level Triggers:**

1. before insert
2. before update
3. before delete
4. after insert
5. after update
6. after delete

**Instead of Trigger:** mainly using for complex views.

**20. Which are Different Parameters in Procedure and Functions?**

**1) IN type parameter**: These types of parameters are used to send values to stored procedures.

**2) OUT type parameter**: These types of parameters are used to get values from stored

procedures. This is similar to a return type in functions.

**3) INOUT parameter:** These types of parameters are used to send values and get values from stored procedures.

**21. Subquery**

In SQL a Subquery can be simply defined as a query within another query. In other words we can say that a Subquery is a query that is embedded in WHERE clause of another SQL query.

* You can place the Subquery in a number of SQL clauses: [WHERE](https://www.geeksforgeeks.org/sql-where-clause/) clause,[HAVING](https://www.geeksforgeeks.org/having-vs-where-clause/) clause, FROM clause.  
  Subqueries can be used with SELECT, UPDATE, INSERT, DELETE statements along with expression operator
* A subquery is a query within another query. The outer query is called as **main query** and inner query is called as**subquery**.
* Subquery must be enclosed in parentheses.
* [ORDER BY](https://www.geeksforgeeks.org/sql-order-by/) command **cannot** be used in a Subquery. [GROUPBY](https://www.geeksforgeeks.org/sql-group-by/)command can be used to perform same function as ORDER BY command.

**22. Stored procedure vs stored function**

|  |  |  |
| --- | --- | --- |
|  | **Functions** | **Procedures** |
| 1. | A function has a return type and returns a value. | A procedure does not have a return type. |
| 2. | You cannot use a function with Data Manipulation queries. Only Select queries are allowed in functions. | You can use DML queries such as insert, update, select etc… with procedures. |
| 3. | You cannot call stored procedures from a function | You can call a function from a stored procedure. |
| 4. | A function does not allow output parameters | A procedure allows both input and output parameters. |
| 5. | You cannot manage transactions inside a function. | You can manage transactions inside a function. |

**23. What is Transaction n ACID properties**

It is a set of operations used to perform logical unit of work.

transaction generally represent change in database.

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Transactions access data using read and write operations.

In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.

**ACID** (an acronym for Atomicity Consistency Isolation Durability)

For a reliable database, all four of these attributes should be achieved:

**Atomicity** is an all-or-none rule for database modifications.

**Consistency** guarantees that a transaction never leaves your database in a half-finished state.

**Isolation** keeps transactions separated from each other until they are finished.

**Durability** guarantees that the database will keep track of pending changes in such a way that the server can recover from an abnormal termination and committed transactions will not be lost.

**24. View**

Views in SQL are kind of virtual table that is used to view & manipulate parts of the table.

A view also has rows and columns as they are in a real table in the database.

A view is a virtual table based on the result-set of an SQL statement.

It can be used for retrieving data as well as updating or deleting rows.

It cannot hold physical data or physical memory.

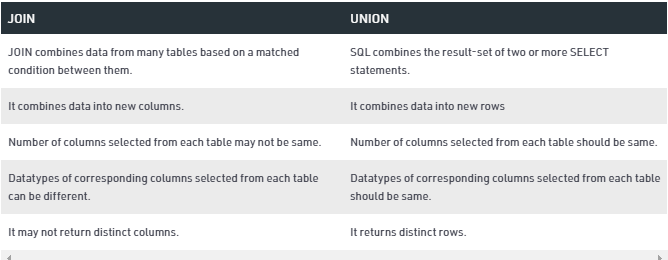
Two main purposes of creating a view are

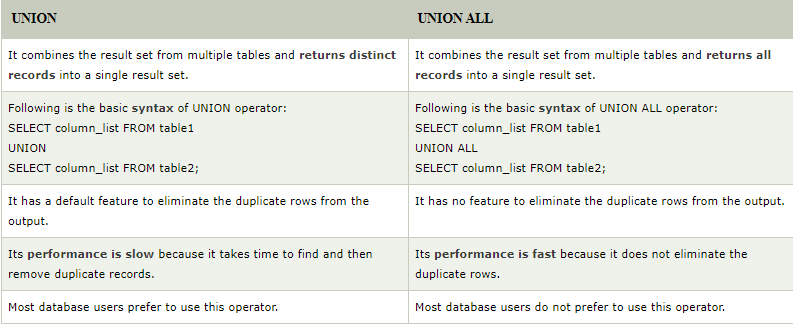
1) Provide a security mechanism which restricts users to a certain subset of data.

2) Provide a mechanism for developers to customize how users can logically view the data

**Difference between union & join**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

****



**Suppose u have 1 millions records in our db then how to fetch all records**

It's depends on your requirement. You have to write an optimized query. In the table you can do indexing for better performance. Or you can divide the single table into multiple tables on the basis of different criteria like date created, alphabetical, etc.

Queries:

|  |  |
| --- | --- |
| EMPNO | SAL |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. **Find the second highest salary of employee**

Select max(sal) from emp

where sal not in (select max(sal) from emp);

or……………………………………………………………..

Select max(sal) from emp

where sal <(select max(sal) from emp);

1. **Display the highest payed employee in each department**

Select max(sal), deptno

from emp

group by deptno;

1. **Find number of employee in each department**

Select count(\*) ,deptno

from emp

group by deptno;

1. **Display alternate records in sql**

//odd no records

select \* from

(select empno, ename, sal, rownum rn

from emp

order by rn)

where mod (rn, 2) != 0;

//even no records

select \* from

(select empno, ename, sal, rownum rn

from emp

order by rn)

where mod (rn, 2) = 0;

1. **Find duplicate values and its frequency of a column**

select ename , count(\*)

from emp

group by ename

having count(\*)>1;

1. **Display employee whose name starts with m**

select ename from emp

where ename like ‘m%’;

1. **Name ends with n**

Select ename from emp

Where ename like ‘%n’;

1. **Having name in anyposition in name**

Select ename from emp

Where ename like ‘%m%’;

1. **Having name does not contain m anywhere**

Select ename from emp

Where ename NOT like ‘%m%’;

1. **Display names of all emplyoees whose name contains exactly 4 letters**

Select ename from emp

Where ename like ‘\_\_\_\_’;

1. **Display names of all emplyoees whose name contains the second letter as ‘L’ & forth letter as ‘ M’**

Select ename from emp

Where ename like ‘\_L%’;

Select ename from emp

Where ename like ‘\_\_\_M%’;

1. **Display the employee names and hire dates for the employees joined at the month of December**

Select hiredate, ename from emp

Where hiredate like ‘%DEC%’;

1. **Display names of all emplyoees whose name contains exactly 2 ‘ L’s**

Select ename from emp

Where ename like ‘%LL%’;

1. **Display names of employee whose name starts with ‘J’ and ends with ‘S’**

Select ename from emp

Where ename like ‘J%S’;

1. **Display nth row in sql**

**rownum cannot deal with greater than or equal to operater**

……………………………………………………………………………………………

//**display 4th row**

Select \* from emp

Where rownum<=4

Minus

Select \* from emp

Where rownum<=3

**Or**……………………………………….

select \* from

(select rownum r, ename, sal from emp)

where r=4;

…………………………………………………………….

**To print all columns of 4th row**

Select \* from(select rownum r, emp.\* from emp)

Where r=4;

…………………………………………………………………………

**To print 2nd , 3rd , 7th record**

Select \* from (select rownum r, emo.\* from emp)

Where r in(2,3,7);

1. **Display employees who are working in location Chicago from emp and dept table**

Select ename, sal, d.deptno,dname,loc

From emp e, dept d

Where e.deptno=d.deptno and LOC=’CHICAGO’;

1. **Display the department name and total salaries from each department**

Select dname, sum(sal)

from emp e, dept d

where e.deptno=d.deptno

group by deptno;

1. **First 1st & last nth rows**

Select \* from(select rownum r, ename, sal from emp)

Where r=1 or r = (select count(\*) from emp);

1. **Display last two rows of the table**

Select \* from emp

Minus

Select \* from emp

Where rownum <=(select count(\*)-2 from emp;

Or…………………………………………………………………………

Select \* from (select rownum r , ename, sal from emp)

Where r>(select count(\*)-2 from emp);

For 1st two numbers

Select \* from (select rownum r , ename, sal from emp)

Where r>(select count(\*)-2 from emp) or r in(1,2);

1. **Nth highest salary**

**1st 3 highest salary**

Select \* from(select distinct sal from emp order by sal desc)

Where rownum<=3;

………………………………………………………………………………………….

**3rd highest salary**

Select \* from(select distinct sal from emp order by sal desc)

Where rownum<=3

Minus

Select \* from(select distinct sal from emp order by sal desc)

Where rownum<=2

---------------------------------------------------------------------------------------------------------

**select empno, sal**

**from emp e1**

**where 3=(select count(distinct sal)**

**from emp e2**

**where e2.sal>e1.sal);**