**Interview Questions of Different Software Companies**

=> Partial Dependency

Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.

The 2nd Normal Form (2NF) eliminates the Partial Dependency.

=>Transitive Dependency

When an indirect relationship causes functional dependency it is called Transitive Dependency.

If  P -> Q and Q -> R is true, then P-> R is a transitive dependency.

To achieve 3NF, eliminate the Transitive Dependency.

=>Clustered and unclustered indexing

**1. Clustered Index:**

Clustered index is created only when both the following conditions satisfy –

1. The data or file, that you are moving into secondary memory should be in sequential or sorted order.
2. There should be a key value, meaning it cannot have repeated values.

Whenever you apply clustered indexing in a table, it will perform sorting in that table only. You can create only one clustered index in a table like primary key.

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.

**2. 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.

| CLUSTERED INDEX | NON-CLUSTERED INDEX |
| --- | --- |
| Clustered index is faster. | Non-clustered index is slower. |
| Clustered index requires less memory for operations. | Non-Clustered index requires more memory for operations. |
| In clustered index, index is the main data. | In Non-Clustered index, index is the copy of data. |
| A table can have only one clustered index. | A table can have multiple non-clustered index. |
| Clustered index has inherent ability of storing data on the disk. | Non-Clustered index does not have inherent ability of storing data on the disk. |
| Clustered index store pointers to block not data. | Non-Clustered index store both value and a pointer to actual row that holds data. |
| In Clustered index leaf nodes are actual data itself. | In Non-Clustered index leaf nodes are not the actual data itself rather they only contains included columns. |
| In Clustered index, Clustered key defines order of data within table. | In Non-Clustered index, index key defines order of data within index. |
| A Clustered index is a type of index in which table records are physically reordered to match the index. | A Non-Clustered index is a special type of index in which logical order of index does not match physical stored order of the rows on disk. |
| The size of clustered index is large. | Size of non-clustered index is comparatively smaller. |
| Primary Keys of the table by default are clustered index. | Composite key when used with unique constraints of the table act as non-clustered index. |

=> how to remove many to many

To avoid this problem, you can break the many-to-many relationship into **two** [**one-to-many relationships**](https://fmhelp.filemaker.com/help/18/fmp/en/FMP_Help/glossary.html#ww1027974) by using a third table, called a **join table.**

Each record in a join table includes a match field that contains the value of the [primary keys](https://fmhelp.filemaker.com/help/18/fmp/en/FMP_Help/glossary.html#ww1060118) of the two tables it joins. (In the join table, these match fields are [foreign keys](https://fmhelp.filemaker.com/help/18/fmp/en/FMP_Help/glossary.html#ww1060063).) These foreign key fields are populated with data as records in the join table are created from either table it joins.

=> A database query is slow. What could be the reasons?

Queries can become slow for various reasons ranging from **improper index usage to bugs in the storage engine itself**. However, in most cases, queries become slow because developers or MySQL database administrators neglect to monitor them and keep an eye on their performance.

=> How many keys?

1. Primary Key
2. Candidate Key
3. Super Key
4. Foreign Key
5. Composite Key
6. Alternate Key
7. Unique Key

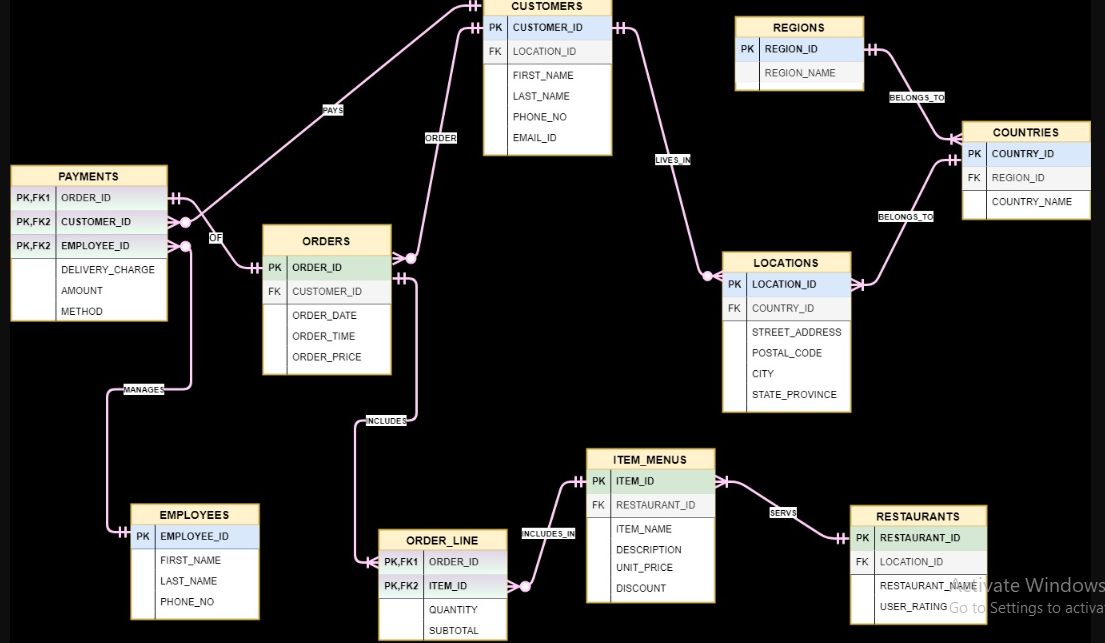
=> bridge entity

A composite entity is also known as a “bridge” entity.

This “bridge” is used to handle the many-to-many relationships that the traditional entity could not handle.

This entity lies between the two entities that are of interest and this composite entity shares the primary keys from both the connecting tables. This composite entity is also known as a “gerund” because it has the characteristics of an entity and a relationship.

=> Design Schema for Food panda



=> 4th max salary? why did you use distinct in query? any other way to find 4th max?

# (How to find Nth highest salary from a table)

By Using **Distinct**:

Suppose the task is to find the employee with the Nth highest salary from the above table. We can do this as follows:

1. Find the employees with top N distinct salaries.
2. Find the lowest salary among the salaries fetched by the above query, this will give us the Nth highest salary.
3. Find the details of the employee whose salary is the lowest salary fetched by the above query.

SELECT \* FROM Employee WHERE sal =

(SELECT MIN(sal) FROM Employee

WHERE sal IN (SELECT DISTINCT TOP *N* sal FROM Employee

ORDER BY sal DESC

)

)

Consider N = 4.

Starting with the most **inner** query, the query:

“*SELECT DISTINCT TOP****4****sal FROM Employee ORDER BY sal DESC*”

will produce the result.

Using **DENSE\_RANK:**

1. DENSE\_RANK computes the rank of a row in an ordered group of rows and returns the rank as a NUMBER. The ranks are consecutive integers beginning with 1.
2. This function accepts arguments as any numeric data type and returns NUMBER.
3. As an analytic function, DENSE\_RANK computes the rank of each row returned from a query with respect to the other rows, based on the values of the value\_exprs in the order\_by\_clause.
4. In the above query the rank is returned based on sal of the employee table. In case of tie, it assigns equal rank to all the rows.

select \* from (

select ename, sal, dense\_rank() over (order by sal desc)r from Emp)

where r=&n;

To find to the 2nd highest sal set n = 2

To find 3rd highest sal set n = 3 and so on.

Using **Limits**

6th highest

use of Limit:

select \* from Employee ORDER BY `sal` DESC limit 5,1;

// will return 6th highest

The DISTINCT keyword in the SELECT clause is used **to eliminate duplicate rows and display a unique list of values**. In other words, the DISTINCT keyword retrieves unique values from a table.

**=> which column do you prefer for indexing?**

**Columns with one or more of the following characteristics are good candidates for indexing:**

* Values are unique in the column, or there are few duplicates.
* There is a wide range of values (good for regular indexes).
* There is a small range of values (good for bitmap indexes).

**Primary key columns** are typically great for indexing because they are unique and are often used to lookup rows.

**=> join vs subquery**

SQL Joins and Subqueries.

An SQL Join statement is used to combine data or rows from two or more tables based on a common field between them.

A subquery is a query that is nested inside a SELECT , INSERT , UPDATE , or DELETE statement, or inside another subquery.

**Joins and subqueries are both used to combine data from different tables into a single result.**

**=> Join query on 3 tables**

**minimum number of join statements to join n tables are (n-1).**

3 tables:

1. Student (sid, sname)
2. Marks (schoolid, sid, score, status)
3. Details (city, email\_id, schoolid, accomplishments)

select sname, score, status, city, email\_id, accomplishments

from student s **inner join** marks m

**ON** s.s\_id = m.s\_id **inner join** details d

**ON** d.school\_id = m.school\_id;