**DATABASE**

1. **Define database model.**

A database model shows the logical structure of a database, including the relationships

and constraints that determine how data can be stored and accessed.

1. **Define DDL and DML?**

**DDL** is a data definition language used to define data structures.

For example:

Create

Alter

Drop

Truncate

**DML** is a data manipulation language used to manipulate the data itself.

For Example:

Insert

Update

Delete

1. **What is cardinality?**

**In context of data models:**

In terms of data modeling, cardinality refers to the relationship between two tables.

They can be of four types:

One to One – A single row of table 1 associates with single row of table 2

One to Many – A single row of table 1 associates with more than one rows of table 2

Many to One – Many rows of table 1 associate with a single row of table 2

Many to Many – Many rows of table 1 associate with many rows of table 2

**In context of query optimization:**

In terms of query, the cardinality refers to the uniqueness of a column in a table. The

column with all unique values would be having the high cardinality and the column with

all duplicate values would be having the low cardinality.

1. **What are the elements of an ERD?**

There are three basic elements in ER-Diagrams:

* Entities are the "things" for which we want to store information. An entity is a person,

place, thing or event.

* Attributes are the data we want to collect for an entity.
* Relationships describe the relations between the entities.

1. Define Normalization?

Normalization is the process of organizing the columns and tables of a relational

database to reduce data redundancy and improve data integrity.

Advantages:

* No duplicate entries
* Saves storage space
* Better performance.

1. Explain about different types of normalization?

**First Normal Form:**

As per First Normal Form, no two Rows of data must contain repeating group of

information i.e each set of column must have a unique value, such that multiple

columns cannot be used to fetch the same row. Each table should be organized into

rows, and each row should have a primary key that distinguishes it as unique

**Student Table:**

|  |  |  |
| --- | --- | --- |
| **Student** | **Age** | **Subject** |
| Adam | 15 | Biology, Maths |
| Alex | 14 | Maths |
| Stuart | 17 | Maths |

In First Normal Form, any row must not have a column in which more than one value

is saved, like separated with commas. Instead of that we separate such data into multiple

rows.

**Student Table following 1NF will be:**

|  |  |  |
| --- | --- | --- |
| **Student** | **Age** | **Subject** |
| Adam | 15 | Biology |
| Adam | 15 | Maths |
| Alex | 14 | Maths |
| Stuart | 17 | Maths |

Using the First Normal Form, data redundancy increases, as there will be many columns

with same data in multiple rows but each row as a whole will be unique.

**Second Normal Form:**

A database is in second normal form if it satisfies the following conditions:

* It is in first normal form
* All non-key attributes are fully functional dependent on the primary key

**New Student Table following 2NF will be:**

|  |  |
| --- | --- |
| **Student** | **Age** |
| Adam | 15 |
| Alex | 14 |
| Stuart | 17 |

In Student Table the candidate key will be **Student** column, because all other column i.e **Age** is

dependent on it.

**New Subject Table introduced for 2NF will be:**

|  |  |
| --- | --- |
| **Student** | **Subject** |
| Adam | Biology |
| Adam | Maths |
| Alex | Maths |
| Stuart | Maths |

In Subject Table the candidate key will be {**Student**, **Subject**} column. Now, both

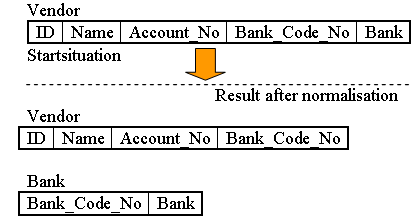
the above tables qualifies for Second Normal Form

**Third Normal Form:**

A table is in third normal form when the following conditions are met:

* It is in second normal form.
* All nonprimary fields are dependent on the primary key.

Example:



The table in this example is in 1NF. But there is a transitive dependency between bank code No and Bank, because Bank Code is not the primary key of this relation. To get to the third normal form(3NF) we have to put the bank name in separate table together with the clearing number to identify it.

Reference link for 3NF: <http://www.gitta.info/LogicModelin/en/html/DataConsiten_Norm3NF.html>

1. **Define Union All operator and Union.**

**UNION:**

The UNION operator is used to combine the result-set of two or more SELECT statements.

UNION remove duplicate record in table.

**UNIONALL:**

The SQL UNION ALL operator is used to combine the result sets of 2 or more SELECT statements. It does not remove duplicate rows between the various SELECT statements (all rows are returned).

1. Define Aggregate functions?

Aggregate functions perform a calculation on a set of values and return a single value.

|  |  |
| --- | --- |
| **Function** | **Description** |
| [AVG()](https://www.w3schools.com/sql/sql_func_avg.asp) | Returns the average value |
| [COUNT()](https://www.w3schools.com/sql/sql_func_count.asp) | Returns the number of rows |
| [FIRST()](https://www.w3schools.com/sql/sql_func_first.asp) | Returns the first value |
| [LAST()](https://www.w3schools.com/sql/sql_func_last.asp) | Returns the last value |
| [MAX()](https://www.w3schools.com/sql/sql_func_max.asp) | Returns the largest value |
| [MIN()](https://www.w3schools.com/sql/sql_func_min.asp) | Returns the smallest value |
| [ROUND()](https://www.w3schools.com/sql/sql_func_round.asp) | Rounds a numeric field to the number of decimals specified |
| [SUM()](https://www.w3schools.com/sql/sql_func_sum.asp) | Returns the sum |

1. Define join and explain its types?

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

* 1. **(INNER) JOIN**: Returns records that have matching values in both tables.



SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name=table2.column\_name;

**2)LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records

from the right table



SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name=table2.column\_name;

3) **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched

records from the left table



SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name=table2.column\_name; 4

**4) FULL (OUTER) JOIN**: Return all records when there is a match in either left or

right table



SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2ON table1.column\_name=table2.column\_name;

1. Explain ACID properties?

**Transaction:** A transaction is a very small unit of a program and it may contain

several lowlevel tasks.

A transaction in database must contain:

**1) Atomicity:** This property ensures that either all the operations of a transaction

reflect in database or none

**Example**: Let’s take an example of banking system to understand this: Suppose Account A has

a balance of 400$ & B has 700$. Account A is transferring 100$ to Account B. This is a transaction

that has two operations a) Debiting 100$ from A’s balance b) Creating 100$ to B’s balance. Let’s

say first operation passed successfully while second failed, in this case A’s balance would be 300$

while B would be having 700$ instead of $800.

This is unacceptable in a banking system. Either the transaction should fail without

Executing any of the operation or it should process both the operations. The Atomicity

property provides that.

**2)Consistency:** To preserve the consistency of database, the execution of transaction

should take place in isolation (that means no other transaction should run concurrently

when there is a transaction already running).

**Example:** account A is having a balance of 400$ and it is transferring 100$ to account

B & C both. So we have two transactions here. Let’s say these transactions run

concurrently and both the transactions read 400$ balance, in that case the final

balance of A would be 300$ instead of 200$. This is wrong. If the transaction were to

run in isolation then the second transaction would have read the correct balance 300$

(before debiting 100$) once the first transaction went successful.

**3)Isolation**: For every pair of transactions, one transaction should start execution

only when the other finished execution.

**4)Durability**: Once a transaction completes successfully, the changes it has made

into the database should be permanent even if there is a system failure. The

recovery-management component of database systems ensures the durability of

transaction.

1. What is Indexing?

Indexing is a data structure technique to efficiently retrieve records from the

database files based on some attributes on which the indexing has been done.

Indexing is defined based on its indexing attributes. Indexing can be of the following types −

* **Primary Index** − Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.
* **Secondary Index** − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
* **Clustering Index** − Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

1. **Define Atomicity and Aggregation.**

**Atomicity:** It’s an all or none concept which enables the user to be assured of incomplete

transactions to be taken care of. The actions involving incomplete transactions are left

undone in DBMS.

**Aggregation:** The collected entities and their relationship are aggregated in this model. It

is mainly used in expressing relationships within relationships.

1. What are the transaction phases?

The various transaction phases are:

* Analysis Phase.
* Redo Phase
* Undo Phase

1. **Define Denormalization.**

Boosting up database performance, adding of redundant data which in turn helps rid

of complex data is called denormalization.

1. ***What are the differences between primary and foreign keys?***

The primary key is the column or set of columns used to uniquely identify the items in a table. A foreign key is used to uniquely identify the items in a different table, allowing join operations to happen.

1. ***What is the difference between an inner and outer join?***

An inner join involves joining two tables where a common id/key exists in both. An outer join is the joining of two tables, but where there is no match in the second (or first).

1. ***What is NoSQL?***

It stands for Not Only SQL and provides an alternative to relational databases. Instead of tabular data stores, they use graph stores, key-value stores, document databases, and wide-column stores. It is popular in the agile development world as developers don’t have to finalize the data model before storing information.