Lecture Notes

DATABASES

INTRODUCTION

In this, you were introduced to the basic concepts of the database, database management systems(DBMS) and the SQL commands. SQL is one of the strongest tools for data manipulation and is heavily used across all industries.
Earlier data used to be stored in files but the file system had its own limitations as mentioned below.

- Data Inconsistency
- Data integrity
- Data Redundancy
- Data security
- Data dependency
- Atomicity
- Concurrent access to data

To overcome the limitations mentioned above, databases were introduced to integrate with the web application.

Data
One critical element of any software application today is Data. As a software engineer, you must know how to work with data.
In general, there are two types of data -

1. **Transient data**:- Transient data is not stored in a database and is the logical data that is used in programming logic. It disappears when the system is shut down.

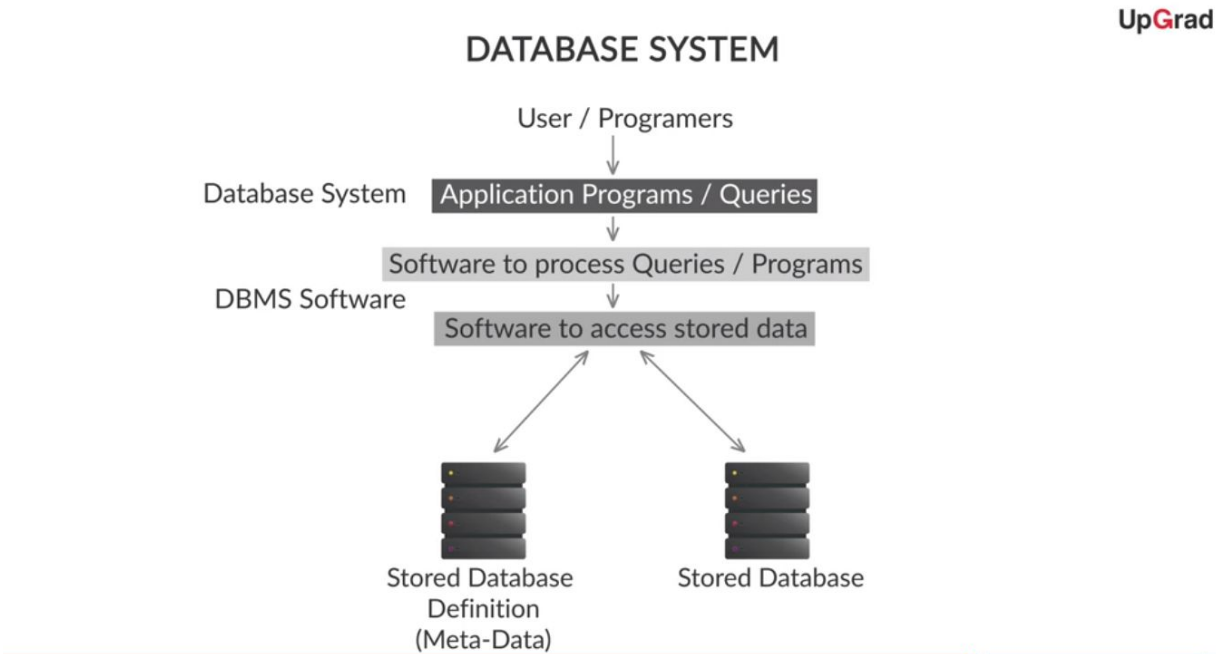
2. **Persistent data**:-. Persistent data is stored in a database and accessible whenever required. In this, you are able to retrieve the data back again when you restart the system. To work with persistent data you need persistent storage.

In order to store the persistent data, you will work with the database management systems(DBMS).

DATABASE SYSTEMS

The database system comprises three major components:-

1. **Application programs/ Queries** - This consists of the application logic and interacts with the DBMS software to make any request for storing/retrieval of data.
2. **DBMS software** - The software interacts with the application program and stored data, such that it helps to process the queries of application and accordingly perform the data storing/retrieval operations.
3. **Stored data** - This is the place where all the required information/data is stored.



Relational Databases

There are various ways to arrange and manage data in a database. The most common is to arrange the data in tables, which is similar to an excel file. The first step is to design the database then only you can store and retrieve the data.

So, there are different ways of designing the database and the most famous way of describing data is relational data model. XML is another language for describing the data. A majority of applications today uses relational databases. The language of the relational database is called the relational data model. After understanding relational data model you will be able to describe the relational schema. The relational schema represents the outline of how the different elements of data are related to each other.

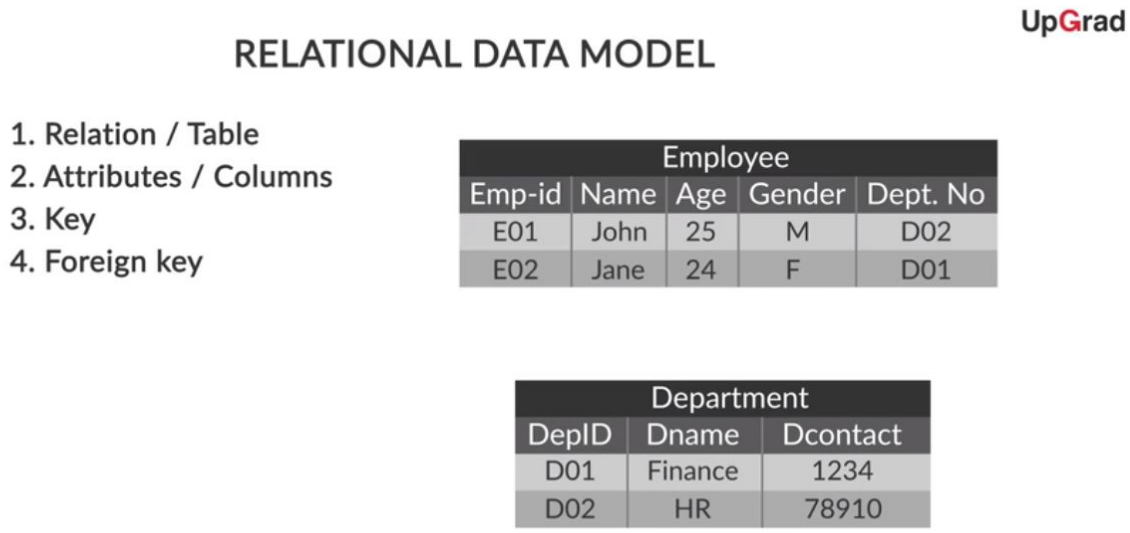
After this, you were introduced with the language used for databases ie **Structured Query Language(SQL).**

Then you will work with the relational database management system, **PostgreSQL**. A **relational database management system(RDBMS)** can work with many databases and in general, a database consists of all the information corresponding to a given application. A database can consist of as many tables as required to store all the required information.

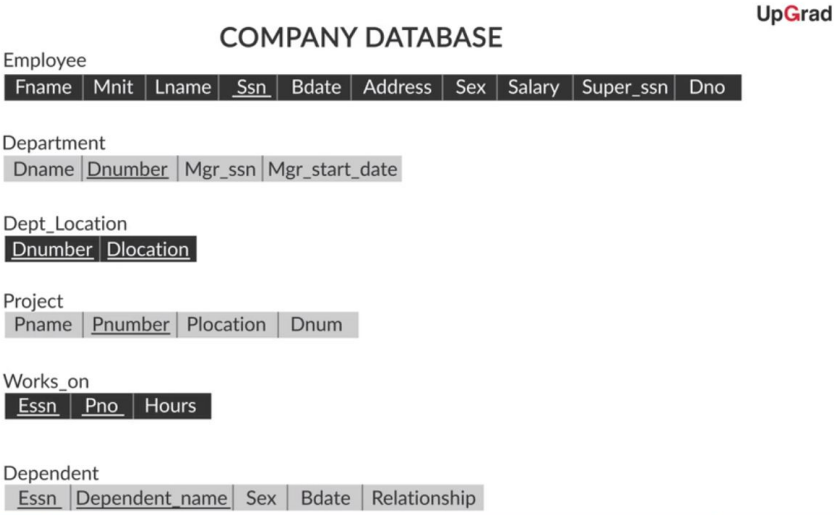
Relational Data model
A relational data model will contain one database and in a database, there can be any number of tables and rows/columns.

Key:-
A table can contain lakhs of rows and if you need to find one particular row from those lakhs of rows. The basis for accessing that table is specified using a **key.** Key is a combination of one or more attributes which helps in uniquely identifying a particular row in a table.

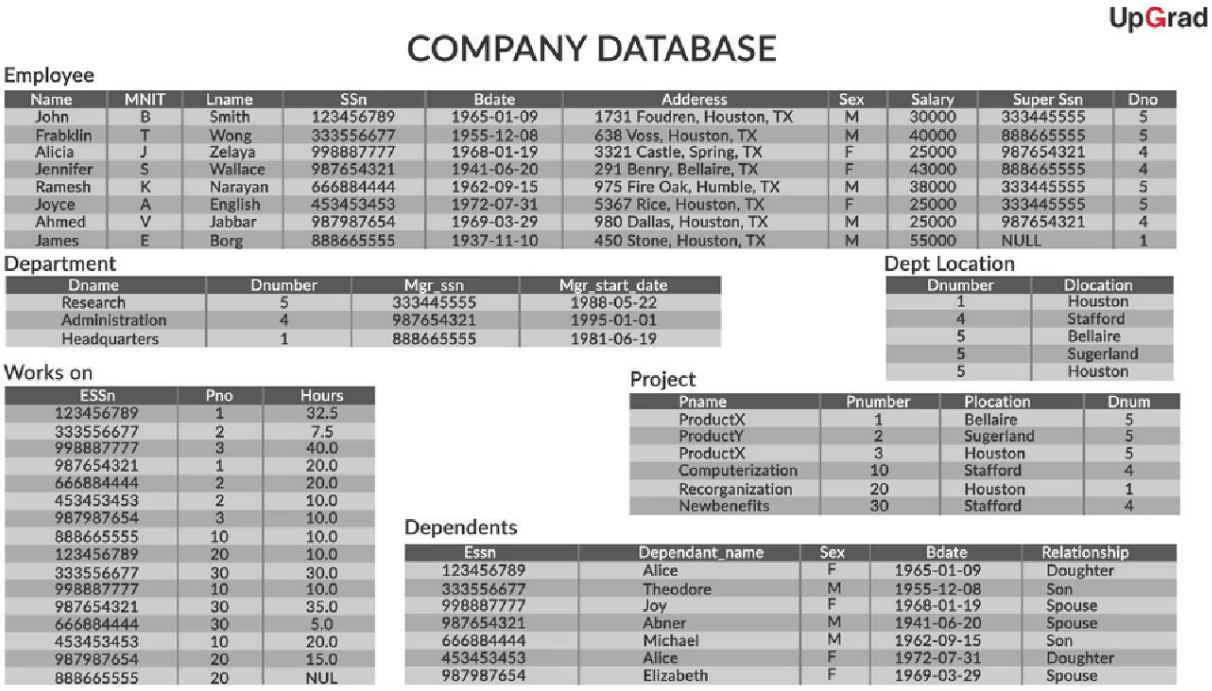
Foreign Key:-
In the picture below **Dept. No** is a foreign key as it refers to the **DeptId** of the other table. Using Dept.No you can obtain the information about that department from the other table named Department



In this, you referred company database as an example to study SQL. In this, all the underlined attributes are the keys.



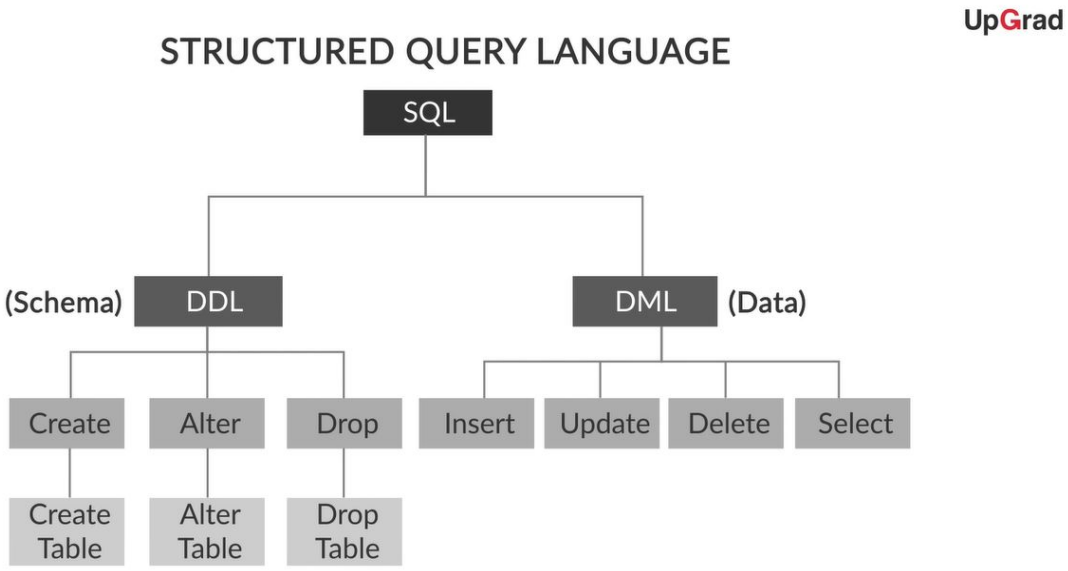
In below image, you will see Company database with actual data. This is a relational schema with an actual relational database.



SQL stands for the structured query language. The commands are broadly categorised as follows:

1. **Data definition language (DDL)**- It is primarily used for working with the schema. **Create**, **alter** and **drop** are the constructs used to specify the schema to the DBMS. So, in the companies database, you will use **create table** construct to create all the tables. To modify the existing database you will use **alter table.** To remove a table from the schema you will use **drop table.**

2. **Data manipulation language (DML)** - It is used to work with actual data. To insert data in a table you will use **insert.** To modify existing data in a given table you will use the **update** script**.** To delete rows you will use **delete.** To retrieve data from a given table you will use **select.**



Data retrieval with SQL

You have learnt to create the 'company' database using pgAdmin3 (GUI), after which you learnt how to perform the data retrieval operations on the 'company' database using structured query language(SQL). Make sure to have the 'company' database ready on your system before moving ahead.

In this, you learnt about the **SELECT** statement
You were asked to download a file Sample SQL commands in which you learned about select statements.
Eg: **select * from employee;** is a select statement. It will retrieve all the rows and columns of table employee.

You can also select only the specific attributes from the table.
Eg: **select fname,lname from employee:** it will only display the first name and the last name of the table employee.

You can also name the columns more user-friendly.
eg: **select fname as “First Name”, lname as “Last Name” from employee;** this will display the fname and lname columns from table employee with column names as First Name and Last Name

If you need to filter the data then you will use **WHERE** clause.
eg: **select * from employee where sex=’M’ and salary >= 30000;** this will filter the data and show only the entries with a male as gender and salary greater than equal to 30000.

After this, you learnt about various queries using examples such as how to write the query for data types like integer, string and working on certain built-in function provided by the PostgreSQL. Also, in the process of working with different queries, you were introduced to the clauses 'like', 'in', 'not in', 'is null' etc.

SQL supports a lot of built-in functions.
If in the date entry, you only need to display the year then comes the built-in function in handy. For this, in SQL you need to use **year** function which is equivalent to **extract** in PostgreSQL.
Eg: **select dname, extract(year from ng_start_date) from department;** in this query, only the year component of the date column will be displayed.

Some more examples of functions
Eg: **select fname from employee where fname like ‘J%’;** it will give all the employees whose name starts with the letter J.

In clause
Eg: **select * from dept_locations where dlocation in (‘Houston’, ‘Stafford’);** it will give all the departments where department location is Houston or Stafford.

Concat function
Eg: **Select concat(fname, ‘ ’, minit, ‘ ’, lname) from employee,** It will give one column with all the first name, middle name and last name concatenated.

Null Values
eg: **select * from employee where super_ssn is null;** It will display all the super ssn entries with value null. You can not use equality operator for comparing with **NULL**.instead we use **is** in place of the equality operator

Order by clause
All the records retrieved through the 'select' query do not necessarily follow an order, i.e. they do not appear in alphabetical, increasing or decreasing order by default. **Order by clause is used for the sorting of data.**

Eg: **select ssn, salary,dno**
from employee
where dno=5
order by salary asc;

It will display the salary in ascending order with the where constraint applied. You can also apply this without the where clause. It will simply display all the entries with ascending order of salary.
You can also sort on the basis of multiple fields.

Eg: **select * from works_on**
order by Pno, Hours;
Here Pno is the primary sorting field and Hours is a secondary sorting field. It means that whenever multiple records come with same entries of Pno then it will be sorted on the basis of Hours.

Aggregate Functions

You will often need to find aggregate values of certain variables like the average age, total salary of employees, the number of males or females etc. For which you need to calculate sums, averages, finding highest and lowest, counting the records etc.
Wondering if you can perform these operations using SQL? Of course, you can. SQL provides various built-in functions for such operations. The functions used to generate collected reports are known as '**aggregate functions'**.

Eg: **select avg(hours) as "Average Hours" from works_on;** It will display the average hours from the works_on table.

Some other aggregate functions are:

- Min to find the minimum.
- Max to find the maximum.
- Sum to find the total.
- Count to just count.

Eg: **select count(*) from employee;** this will the number of entries in the employee table in our case it was 8.

'Group by' and 'Having' Clause

'**group by**'clause will help to categorize and retrieve the data, for example, this clause can be used to find the average salary of all the employees working in a specific department, count the number of employees in each department etc.

Eg: **select dno, count(*) as "Number of employees"**
from employee
group by dno;
This will give us the total count department wise and not just the count of all the employees from all the departments.

Let's see one more example
select dno, sex, avg(salary)
from employee

where super_ssn = '333445555'
group by dno, sex;
This will display the average salary of employee (department-wise and gender-wise) which is group by dno and sex.

Till now you are familiarised with group by clause now you learnt about the having clause.
The 'having' clause is typically used when you have to apply a filter condition on an aggregated value. This is because the 'where' clause is applied before the aggregation takes place, and thus it is not useful when you want to apply a filter on an aggregated value.
Eg:
select dno, count(*)
from employee
group by dno, sex
having count(*) >=3;
This will give us group by of only whose value is greater than equal to 3.

SQL commands

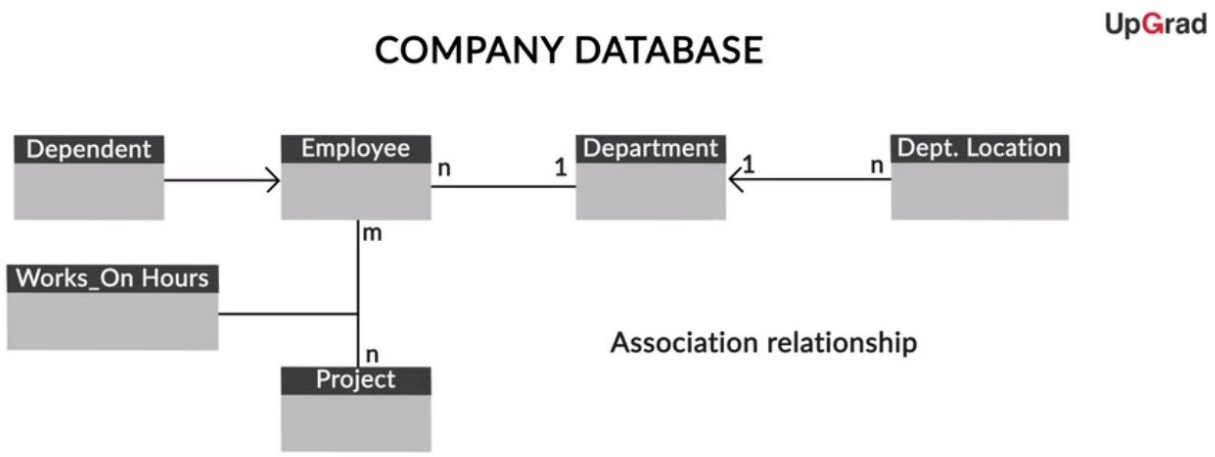
- **Select**
- **From**
- **Where**
- **Group by**
- **Having**
- **Order by**

They all used in the context of retrieving data from a single table.

Object Relational Mapping(ORM)

Object-relational mapping(ORM) helps to translate the object-oriented data model to a relational database, such that it maps certain class with its corresponding table in the database. You understood this with the help of the company database.
In this, we have different classes employee, department, dept. location, dependent, project, work_on_hours. In the object-oriented model, we say that there are relationships between these classes. There can be many types of relations from which you studied about association relationship.
Association Relationship:-

1. **One to many relationship**- In below image, there is one to many relationship between department and employee. Here, 1 in front of one class and n in front of the other and it is interpreted as every department is associated with many employees or every employee is associated with one department.
2. **Many to many relationship** - The relationship between employee and project is many to many relationship. It will be interpreted as every employee is associated with many projects or every project has got many employees



Object-relational mapping(ORM) - When you are working with an object-oriented model in java programming and then storing the data in a relational database then to enable the interaction between the incompatible interfaces for which ORM is required.

How to represent a one-to-many or many relationship of an object-oriented model in a relational database?

This is achieved by the appropriate mapping of the primary key and foreign key. One to many relationship is mapped by adding a foreign key to only one table whereas many to many is mapped by creating a new table and adding both the keys as a foreign key.

Retrieving data from multiple tables
Nested Queries and Inner Join

A database is a collection of multiple related tables. While generating insights from the data, you may need to refer to these multiple tables in a query. There are two ways to deal with such types of queries:

1. Nested queries/Subqueries
2. Joins

Nested query-
It is basically a query inside a query.
Eg:-

```
select fname
from employee
where dno = (select Dnumber from department where dname = 'Research')
order by fname;
```

In this example, we have used a nested query to find the department number of the department with name research from table department and simultaneously we used it to find the name from the employee table with the department number research.

Joins-
You also learnt how to retrieve data from multiple tables using inner join keyword. Join is a way to retrieve data from multiple tables. The most common join is the inner join, which selects only those rows from two tables where the common column has the same value.

Eg:-
select fname, dname
from employee inner join department on employee.dno = department.dno;
In this example, you have retrieved data from two different tables using joins. In the from statement, you need to mention the name of the tables that you need to join here it is **employee inner join department** the specify the join condition. And as a result, we were able to retrieve data from both tables.

Summary

Let's have a quick look at all the points you studied about in databases. So we started with the discussion of:

1. Data persistence approach- To develop data all by ourselves but it is really not worth it, not bug-free.

So we moved onto database management systems that will manage all our data in a systematic way
Then we studied different types of data models and studied the relational data model.

Relational Databases:

1. Database
2. Table
3. Attribute/Columns
4. Keys
5. Foreign Key

Then we studied the **structured query language(SQL)**

1. DDL to work with the schema.
2. DML to work with the data.
3. Select command
4. In statement
5. Concat statement
6. Null statements
7. filtering using where clause
8. ‘order by’ clause for sorting the records of a table

9. Aggregate functions such as avg(), count(), etc.
10. 'group by' clause to categorise the data
11. 'having' clause

Then you have learnt about object-relational mapping (ORM)- Object-relational mapping(ORM) helps to translate the object-oriented data model to a relational database, such that it maps certain class with its corresponding table in the database. Then you have learnt to the nested queries or joins to retrieve the data from multiple tables.