Agenda

- What is a Database
- What, What Not, Why, How of Scaler SQL Curriculum
- Types of Databases
- Intro to Relational Databases
- Intro to Keys

What is a Database

In your day to day life whenever you have a need to save some information, where do you save it? Especially when you may need to refer to it later, maybe something like your expenses for the month, or your todo or shopping list?

Many of us use softwares like Excel, Google Sheets, Notion, Notes app etc to keep a track of things that are important for us and we may need to refer to it in future. Everyone, be it humans or organizations, have need to store a lot of data that me useful for them later. Example, let's think about Scaler. At Scaler, we would want to keep track of all of your's attendance, assignments solved, codes written, coins, mentor session etc! We would also need to store details about instructors, mentors, TAs, batches, etc. And not to forget all of your's email, phone number, password. Now, where will we do this?

For now, forget that you know anything about databases. Imagine yourself to be a new programmer who just knows how to write code in a programming language. Where will you store data so that you are able to retrieve it later and process that?

You will store it in files. You will write code to read data from files, and write data to files. And you will write code to process that data. For example you may create separate CSV (comma separated values, you will understand as we proceed) files to store information about let's say students, instructors, batches.

Examples of CSV

```
students.csv
name, batch, psp, attendance, coins, rank
Naman, 1, 94, 100, 0, 1
Amit, 2, 81, 70, 400, 1
Aditya, 1, 31, 100, 100, 2
```

```
name, subjects, average_rating
Rachit, C++, 4.5
Rishabh, Java, 4.8
Aayush, C++, 4.9
```

```
id, name, start_date, end_date
1, AUG 22 Intermediate, 2022-08-01, 2023-08-01
2, AUG 22 Beginner, 2022-08-01, 2023-08-01
```

What happens if we want to find average now? Finding average is cumbersome in CSV

Now, let's say you want to find out the average attendance of students in each batch. How will you do that? You will have to write code to read data from students.csv, and batches.csv, and then process it to find out the average attendance of students in each batch. Right?

Question

Do you think this will be very cumbersome?

Choices

- [ ] Yes
- [ ] No

Issues with using files as a database

Okay, let's think out problems that can happen writing such code. Before that, take a while and think about what all can go wrong?

Correct! There are a lot of issues that can happen. Let's discuss these:

1. Inefficient

While the above set of data is very small in size, let's think of actual Scaler scale. We have 2M+ users in our system. Imagine going through a file with 2M lines, reading each line, processing it to find your relevant information. Even a very simple task like finding the psp of a student named Rahul will require you to open the file, read each line, check if the name is Rahul, and then return the psp. Time complexity wise, this is O(N) and very slow.

2. Integrity

Is there anyone stopping you from putting a new line in above file `students.csv` as `Rahul, 1, Hello, 100, 0, 1` . If you see that `Hello` that is unexpected. The psp can't be a string. But there is no one to validate and this can lead to very bad situations. This is known as data integrity issue, where the data is not as expected.

3. Concurrency

Later in the course, you will learn about multi-threading and multi-processing. It is possible for more than 1 people to query about the same data at the same time. Similarly, 2 people may update the same data at the same time. On save, whose version should you save? Imagine you give same Google Doc to 2 people and both make changes on the same line and send to you. Whose version will you consider to be correct? This is known as concurrency issue.

4. Security

Earlier we talked about storing password of users. Imagine them being stored on files. Anyone who has access to the file can see the password of all users. Also anyone who

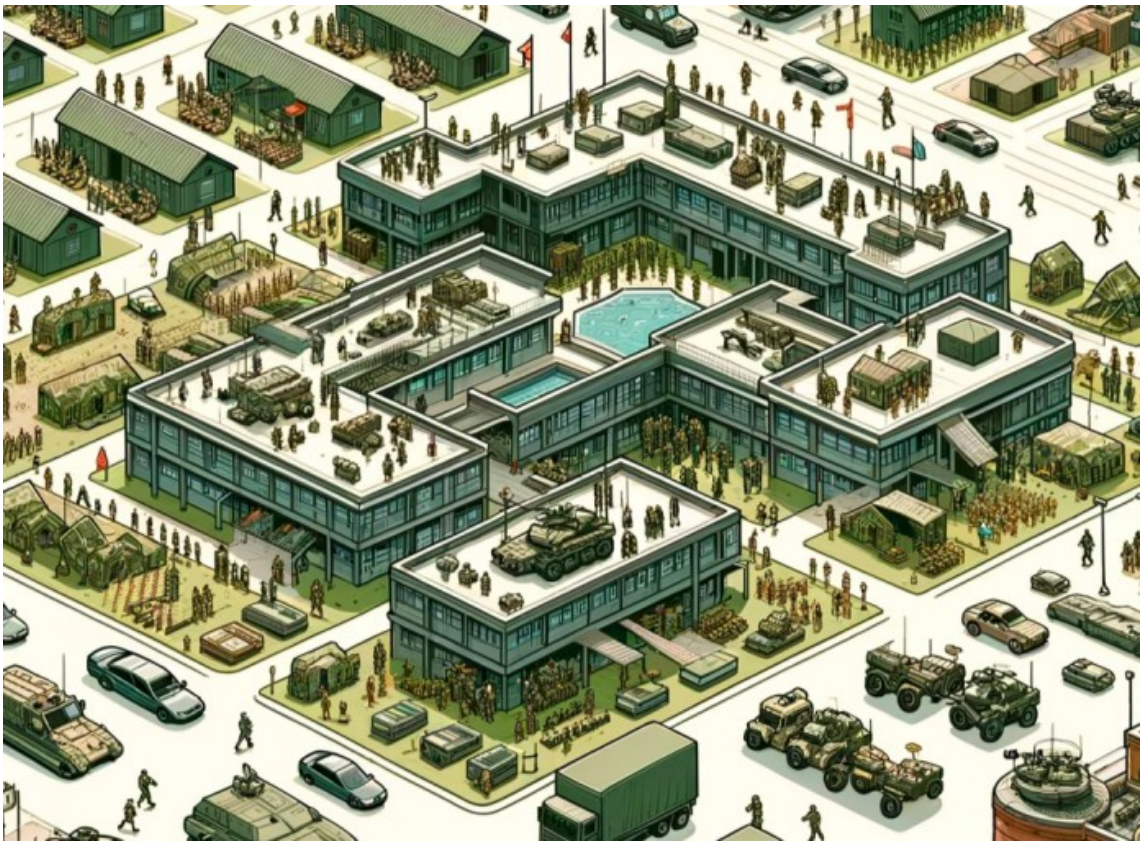
has access to the file can update it as well. There is no authorization at user level. Eg: a particular person may be only allowed to read, not write.

What's a Database

Now let's get back to our main topic. What is a database? A database is nothing but a collection of related data. Example, Scaler will have a Database that stores information about our students, users, batches, classes, instructors, and everything else. Similarly, Facebook will have a database that stores information about all of it's users, their posts, comments, likes, etc. The above way of storing data into files was also nothing but a database, though not the easiest one to use and with a lot of issues.

Analogy to understand Databases:

- Like we have army personnels at Army base:



pic credits: Unknown

- We have airforce personnel at Airbase:

pic credits: Unknown

Similarly we have data at Database:

pic credits: Unknown

What's DBMS

What's a Database Management System (DBMS)

A DBMS as the name suggests is a software system that allows to efficiently manage a database. A DBMS allows us to create, retrieve, update, and delete data (often called CRUD operations). It also allows to define rules to ensure data integrity, security, and concurrency. It also provides ways to query the data in the database efficiently.

Eg: find all students with psp > 50, find all students in batch 1, find all students with rank 1 in their batch, etc. There are many database management systems, each with their tradeoffs. We will talk about the types of databases later.

Types of Databases

Welcome back after the break. Hope you had a good rest and had some water, etc. Now let's start with the next topic for the day and discuss different types of databases that exist. Okay, tell me one thing, when you have to store some data, for example, let's say you are an instructor at Scaler and want to keep a track of attendance and psp of every student of you, in what form will you store that?

Correct! Often one of the easiest and most intuitive way to store data can be in forms of tables. Example for the mentioned use case, we may create a table with 3 columns:

name, attendance, psp and fill values for each of my students there. This is very intuitive and simple and is also how relational databases work.

Databases can be broadly divided into 2 categories:

1. Relational Databases
2. Non-Relational Databases

Relational Databases

Relational Databases allow you to represent a database as a collection of multiple related tables. Each table has a set of columns and rows. Each row represents a record and each column represents a field. Example, in the above case, we may have a table with 3 columns: name, attendance, psp and fill values for each of my students there. Let's learn some properties of relational databases.

Non-Relational Databases

Now that we have learnt about relational databases, let's talk about non-relational databases. Non-relational databases are those databases that don't follow the relational model. They don't store data in form of tables. Instead, they store data in form of documents, key-value pairs, graphs, etc. In the DBMS module, we will not be talking about them. We will talk about them in the HLD Module.

In the DBMS module, our goal is to cover the working of relational databases and how to work with them, that is via SQL queries.

Property of RDBMS - 1

1. Relational Databases represent a database as a collection of tables with each table storing information about something. This something can be an entity or a relationship between entities. Example: We may have a table called students to store information about students of a batch (an entity). Similarly we may have a table called student_batches to store information about which student is in which batch (a relationship betwen entities).

Property of RDBMS - 2

2. Every row is unique. This means that in a table, no 2 rows can have same values for all columns. Example: In the students table, no 2 students can have same name, attendance and psp. There will be something different for example we might also want to store their roll number to distingusih 2 students having the same name.

Property of RDBMS - 3

3. All of the values present in a column hold the same data type. Example: In the students table, the name column will have string values, attendance column will have integer values and psp column will have float values. It cannot happen that for some students psp is a String.

Property of RDBMS - 4

4. Values are atomic. What does atomic mean? What does the word `atom` mean to you?

Correct. Similarly, atomic means indivisible. So, in a relational database, every value in a column is indivisible. Example: If we have to store multiple phone numbers for a student, we cannot store them in a single column as a list. How to store those, we will learn in the end of the course when we do Schema Design. Having said that, there are some SQL databases that allow you to store list of values in a column. But that is not a part of SQL standard and is not supported by all databases. Even those that support, aren't most optimal with queries on such columns.

Property of RDBMS - 5

5. The columns sequence is not guaranteed. This is very important. SQL standard doesn't guarantee that the columns will be stored in the same sequence as you define them. So, if you have a table with 3 columns: name, attendance, psp, it is not guaranteed that the data will be stored in the same sequence. So it is recommended to not rely on the sequence of columns and always use column names while writing queries. While MySQL guaranteees that the order of columns shall be same as defined at time of creating table, it is not a part of SQL standard and hence not guaranteed by all databases and relying on order can cause issues if in future a new column is added in between.

Property of RDBMS - 6

6. The rows sequence is not guaranteed. Similar to columns, SQL doesn't guarantee the order in which rows shall be returned after any query. So, if you want to get rows in a particular order, you should always use `ORDER BY` clause in your query which we will learn about in the next class. So when you write a SQL query, don't assume that the first row will always be the same. The order of rows may change across multiple runs of same query. Having said that, MySQL does return rows in order of their primary key (we will learn about this later on), but again, don't rely on that as not guaranteed by SQL standard.

Property of RDBMS - 7

7. The name of every column is unique. This means that in a table, no 2 columns can have same name. Example: In the students table, we cannot have 2 columns with name `name`. This is because if I have to write a query to get the name of a student, we will have to write `SELECT name FROM students`. Now if there are 2 columns with name `name`, how will the database know which one to return? Hence, the name of every column is unique.

Keys in Relational Databases

Now we are moving to probably the most important foundational concept of Relational Databases: Keys. let's say you are working at Scaler and are maintaining a table of every students' details. Someone tells you to update the psp of Rahul to 100. How will you do that? What can go wrong?

Correct. If there are 2 Rahuls, how will you know which one to update? This is where keys come into picture. Keys are used to uniquely identify a row in a table. There are 2 important types of keys:

1. Primary Key and
2. Foreign Key.

There are also other types of keys like Super Key, Candidate Key etc. Let's learn about them one by one.

Super Keys

To understand this, let's take an example of a students table at scaler with following columns.

name	psp	email	batch	phone number
Rahul	1	94	100	0
Amit	2	81	70	400
Aditya	1	31	100	100

Which are the columns that can be used to uniquely identify a row in this table?

Can name be Super Key?

Let's start with name. Do you think name can be used to uniquely identify a row in this table?

Correct. Name is not a good idea to recognize a row. Why? Because there can be multiple students with same name. So, if we have to update the psp of a student, we cannot use name to uniquely identify the student. Email, phone number on the other hand are a great idea, assuming no 2 students have same email, or same phone number.

Can a combination of columns be Super Key?

Do you think the value of combination of columns (name, email) can uniquely identify a student? Do you think there will be only 1 student with a particular combination of name and email. Eg: will there be only 1 student like (Rahul, rahul@scaler.com)?

Correct, similarly do you think (name, phone number) can uniquely identify a student? What about (name, email, phone number)? What about (name, email, psp)? What about (email, psp)?

The answer to each of the above is Yes. Each of these can be considered a `Super Key`. A super key is a combination of columns whose values can uniquely identify a row in a table. What do you think are other such super keys in the students table?

In the above keys, did you ever feel something like "but this column was useless to uniquely identify a row.." ? Let's take example of (name, email, psp). Do you think psp is required to uniquely identify a row? Similarly, do you think name is required as you anyways have email right? This means a Super key can have redundant/extra columns.

Time for a few quizzes

Quiz 1

Which of the following is a Super Key for the Student table?

> *Consider StudentID to be unique in students table.*

Choices

- [ ] {StudentID, CourseID}
- [ ] {FirstName, LastName}
- [ ] {Age, CourseName}
- [ ] {LastName, CourseID}

Quiz 2

Which of these combinations could also be a Super Key for the Student table?

> *Consider StudentID to be unique in students table.*

Choices

- [ ] {StudentID, CourseName}
- [ ] {FirstName, Age}
- [ ] {LastName, Age}
- [ ] {CourseID, CourseName}

Quiz 3

Given the uniqueness of the StudentID, which of these could be a potential Super Key for the Student table?

Choices

- [ ] {StudentID, FirstName}
- [ ] {StudentID, Age}
- [ ] {StudentID, LastName}
- [ ] All of the above

> *Answers for Quizzes:*
>
> 1. *Option 1*
> 2. *Option 1*
> 3. *Option 4*