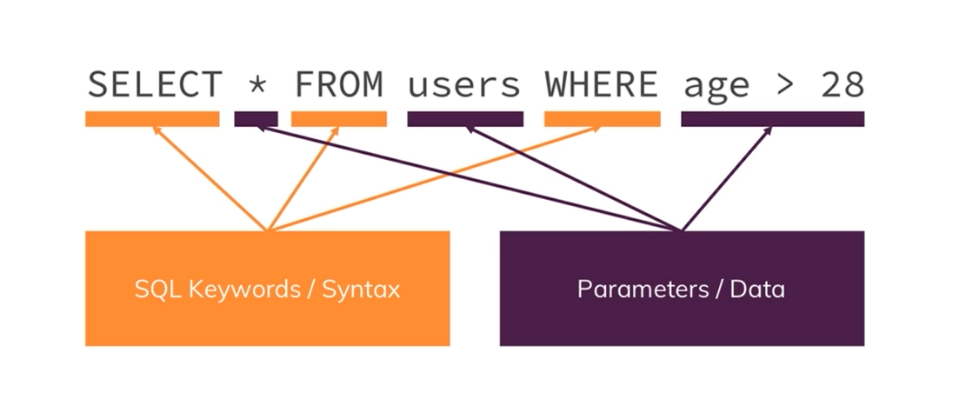
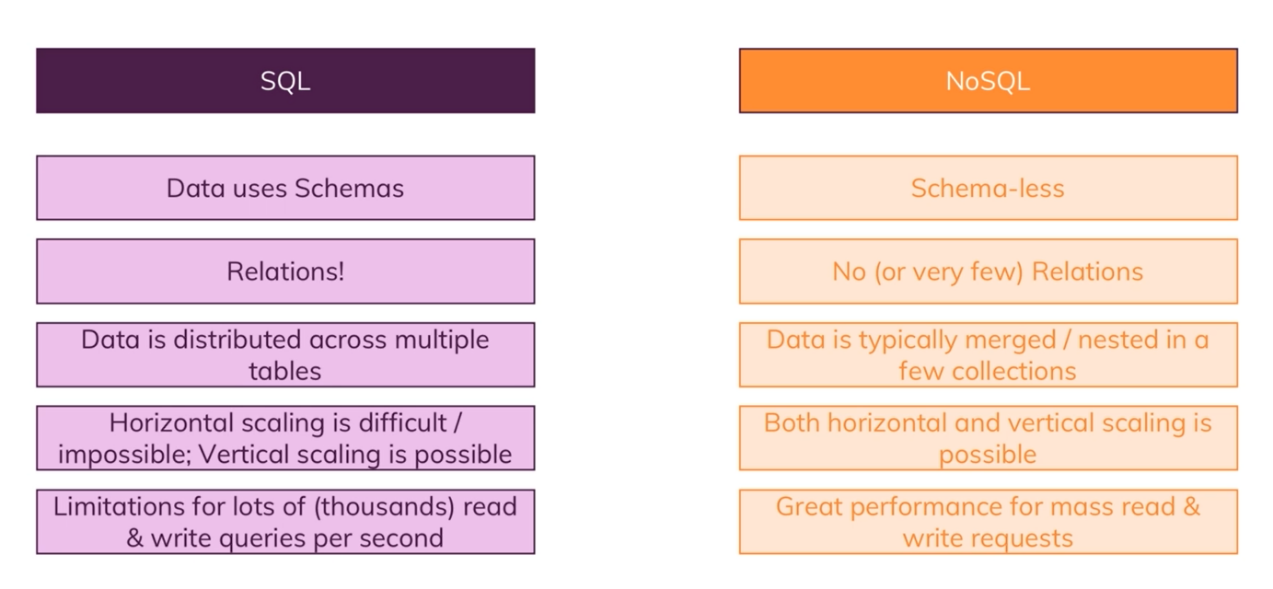
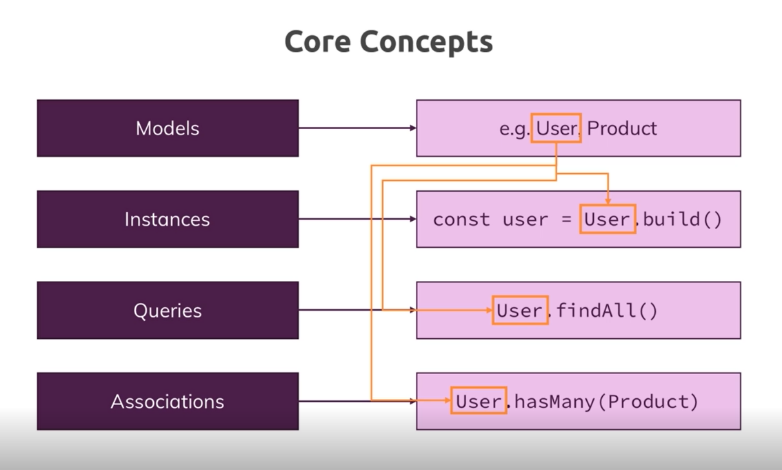
**SQL – Introduction**

**Table of Content:**

* **Different kind of Databases**
* **Using SQL in a node.js app**

1. **Choosing a Database**goal is to store data and make is easily accessible and fastly available.   
   2 types of databases:  
      
   SQL Database (eg. MySql):  
   a. it thinks in tables and in each table, there are fields (called as column) and data (called as records) which is rows in our table.  
   b. in SQL database we can relate tables with each other.  
   c. Core SQL Database characteristics are:  
    A. they have very strong data schema. So that for each table we can clearly define type (look like) of data store in each column. All data entered should fit with the rules.  
    B. (Data relation) we relate our data in 3 different relations:  
    1-to-1, 1-to-many, many-to-many. Tables are connected  
   d. Query are imp part of sql. Query are the command use to communicate with database.  
   eg.   
     
     
   NoSQL Database (eg. MongoBD)  
   a. NoSql means it does not follow approach of sql database. (it does not have schema and relation)  
   b. Tables are called Collection and collection contain documents.  
   c. NoSQL is Schemaless means it’s okay to don’t have some fields and it’s okay to have some addition fields as compare to other documents.  
   (we can have different structure of documents in a collection)  
   d. in this we don’t have relation intend we duplicate data.  
   e. NoSQL characteristics:  
    A. No Data Schema  
    B. No Data Relations (no/few connections)
2. **Horizontal vs Vertical Scaling**there are 2 approaches to scale our database:  
     
   Horizontal Scaling  
   a. in this we add more servers and connect them with our database and split our data accordingly.  
   b. we need some process which runs queries on all of them and merge them which is not easy todo but this good way to scale.  
     
   Vertical Scaling  
   a. in this we make our old server stronger by adding more cpu power or memory.  
   b. but there is some limit we can’t fit infinite cpu power into a single machine.
3. **SQL vs NoSQL**SQL where data is not change frequently and data have relations  
   NoSQL where data changes frequently  
   
4. **Connecting app with SQL Database**a. first we have to install page named as mysql2  
   npm install --save mysql2  
   b. now we have to connect our app with mysql db. To-do that we have 2 approaches:  
    mysql.connect(): this will create one connection to execute query and shutdown after execution and the down side is we need to re-execute the code to create connection for every new query. Creating new connection for every query is inefficient both for code and connection to db and this will affect performance of our app.  
     
   better way is to create connection pool  
     
    mysql.createPool(): to pool we extort in a special form pool.promise() this we allow us to use promise when working with these connections which handle asynchronous tasks instead of callbacks.
5. **Add and Fetching data to db**a. we return a promise from model and will use, then() and catch() methods.
6. **Sequelize**due to this package, we will focus on Nodejs not SQL.  
   we will use mqsql db in background but the cod we write will be different (query)   
   a. **what is Sequelize** A. it is an Object Relational Mapping Library (ORM) which means it handle all the heavy lifting of SQL behind the screen and maps it to JSON object.  
     
     
   b.  **Connecting with Sequelize** A. install sequalize package. (and make sure mysql2 is also installed)  
    B. first we connect sequelize with database, by importing sequelize and creating object of it.  
   const sequelize = new Sequelize('node-complete', 'root', 'Samarjeet1!', { dialect: 'mysql', host: 'localhost' }); **c. Defining model** to define a model we import database.js file and use method of reference of sequelize object.   
   sequelize.define(param1, {}); where param1 is name of model and params is map of key (is attribute) and value (is the contrain) of that attribute.  
     
   **d. Syncing Js definition to the database** before running our app we have to make sure that tables of that sequelize model exists in database. If not we have to create that and if exists we don’t need to override it to make new table.  
     
   we can this, with the help of sync() method of sequelize reference in database.js file.  
     
   write this in app.js file at last, in then() we start our app.  
     
     
     
   **e. Inserting data & Creating a Product** A. to insert a add we will create() on the exported model. It takes map as a parameter where key is attribute and value is data.  
     
   **f. Retrieving data & Finding Products** A. to get all data we will findAll() on the exported model. It will return all data of table  
    B. to get conditional data we will **findAll({where: })** on the exported model and pass map of conditions. OR we can use **findByPk(id**) which takes an id as parameter and return single value while findAll return array of value