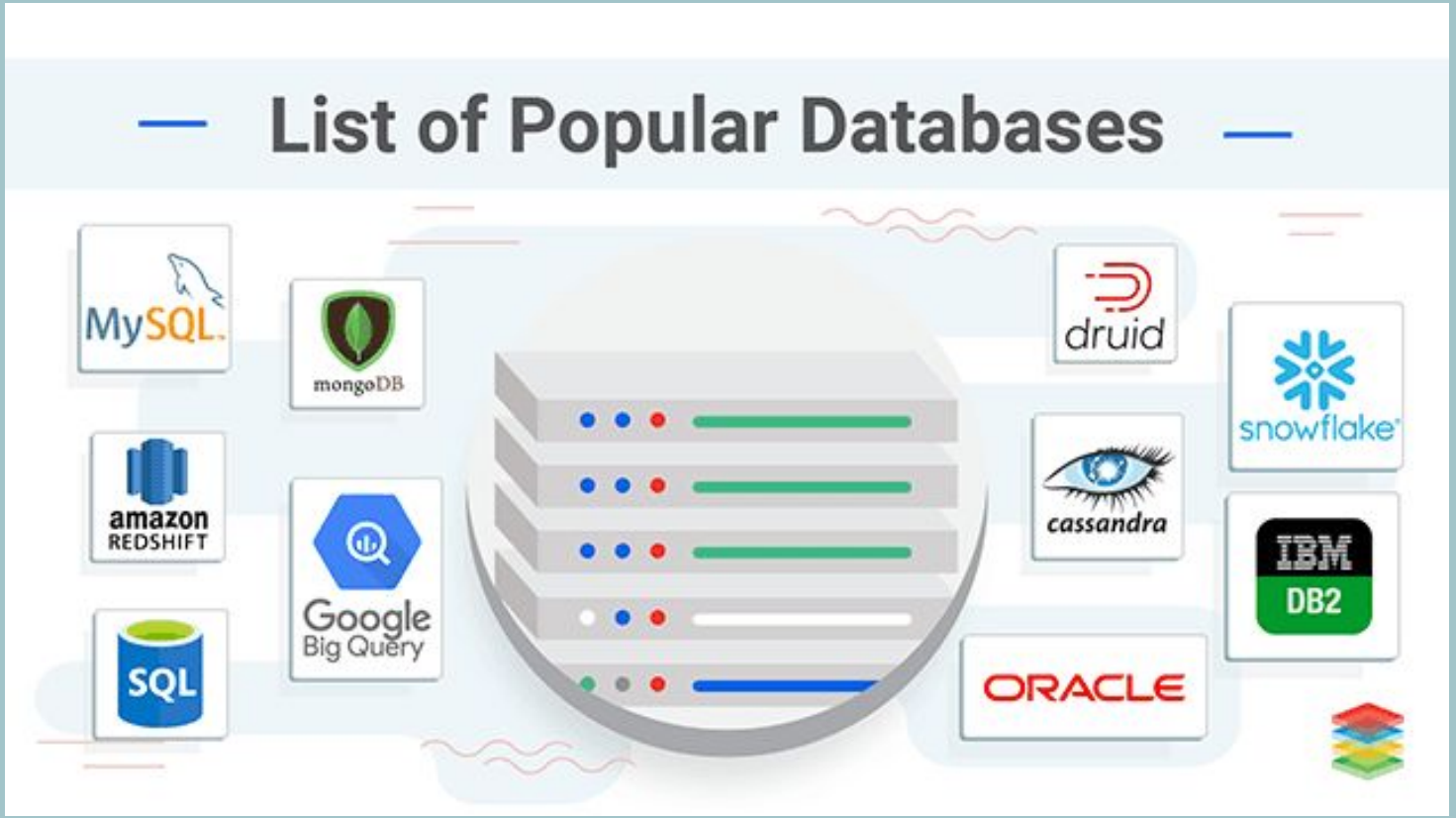
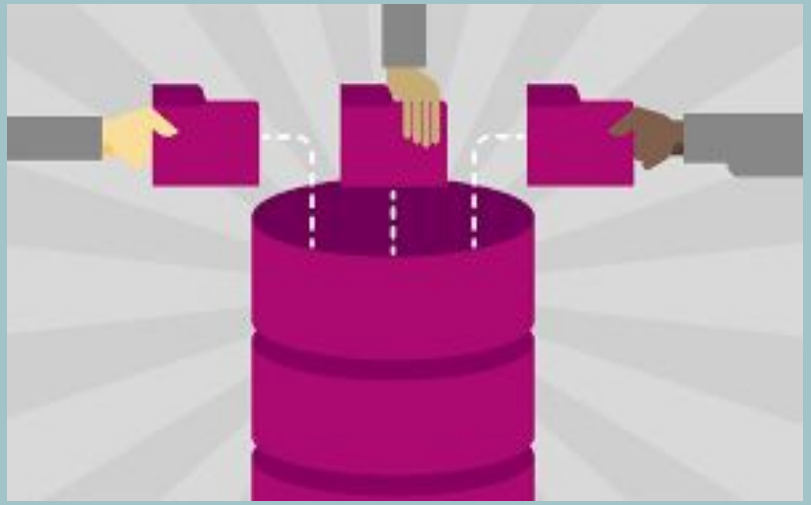
Getting Familiar with Databases

Winnie Cai
Dana Hausman
Serge Mavuba
Corinne Haley


List of Popular Databases
MySQL
mongoDB
druid
snowflake
amazon
REDSHIFT
cassandra
IBM
DB2
Google
Big Query
SQL
ORACLE

How do you define a Database?

- A database is a collection of bits of data that is organized into files, which are called tables. Tables are a logical way of accessing, managing, and updating data.
- In a database, data can also appear in other formats, such as figures, graphics, images, and audio-video recordings.



- **Database Types:**
  - Relational databases
  - Non-relational databases

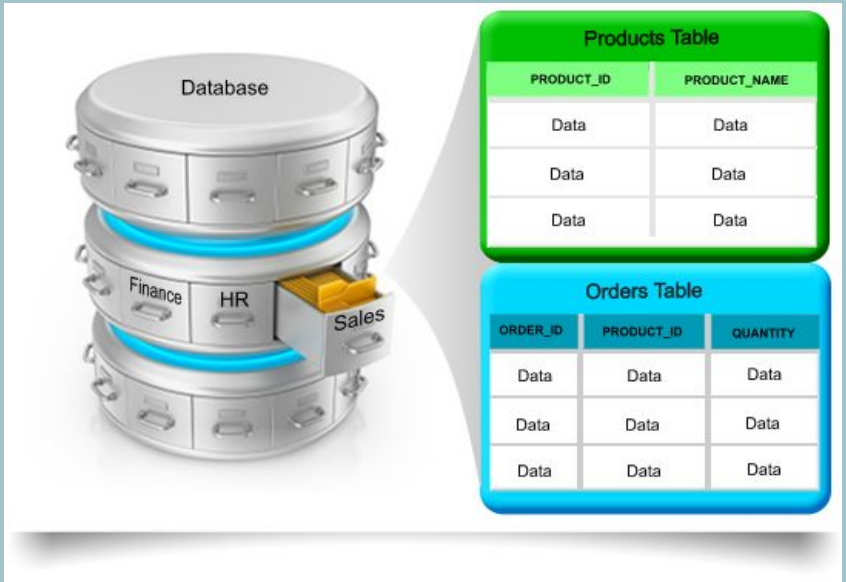
What’s the difference?

Relational (SQL)

- Structured query language
- Rigid schema
- Run transactional or analytical queries
- Used primarily in relational databases
- Easily update, delete, or add to existing DBs
- Uses vertical scaling, which means changing the resources (CPU,RAM), inside the computer or virtual machine (WM) to fit the needs of the database

Use cases

- Traditional application, enterprise resource planning (ERP), customer relationship management (CRM), ecommerce

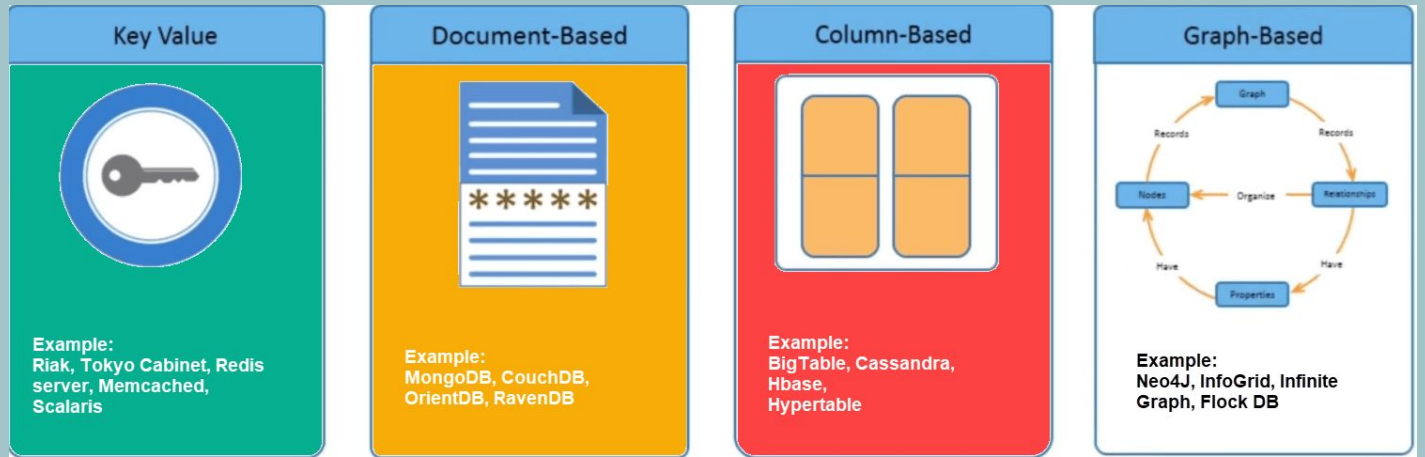


Non-relational (NoSQL)

- Not only SQL
- Not a rigid schema
- Uses different types of data models, such as graph, document, key-value
- Alternative to traditional relational DB
- Used for large data stores in cloud and web applications
- Uses horizontal scaling, which means adding more computers, servers, and more

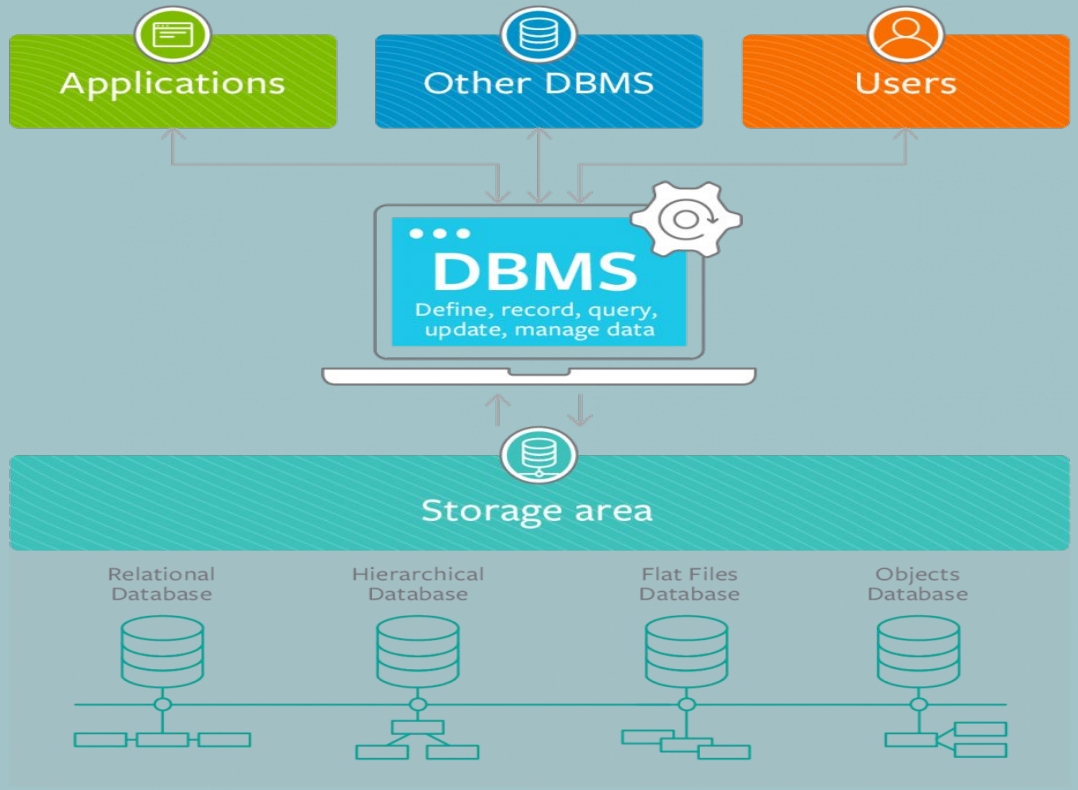
Use cases

- High traffic web applications, ecommerce systems, gaming applications



Database Management System (DBMS)

You manage stored data with a database management system (DBMS).



Managing Databases

Create database Sample

- The **CREATE DATABASE** statement
- **Sample - usersDB;**

Rename a database:

- The **ALTER** statement
- **ALTER DATABASE** *Sample* **MODIFY** *name = usersDB*

Delete or dropping:

- The **DROP DATABASE** statement
- DROP DATABASE SampleDB

Drop database Sample1

Managing Tables

Create table tblEmployee

- The **CREATE TABLE** statement is used to create a new table in a database.
- **CREATE TABLE** tblEMployee (

  EmployeeID int not null Primary Key,

  Fname nvarchar (50) not null,

  Name nvarchar (50) not null,

  GenderID int

  );

Managing Tables(cont'd)

The ALTER TABLE statement is used to add, delete, or modify tables.

Rename a table:

- **ALTER TABLE** tblStudent **RENAME TO** tblGraduate

Change the column within a table:

- **ALTER TABLE Modify Column**

  ALTER TABLE tblStudent
  a add Email nvarchar(100);

Alter table tblStudent:

- To delete a column in the table tblStudent, use the following syntax:
- ALTER TABLE tblStudent
  DROP Email;

Order of Execution

1. FROM	tblLearner
2. ON	tblLearner.genderID = tblGender.ID
3. JOIN	inner/left/right
4. WHERE	jobTitle = ‘cloud engineer’
5. GROUP BY	group by States
6. WITH CUBE OR WITH ROLLUP	group by rollup (States)
7. HAVING	sum(salary) > 100000
8. SELECT	ID, Fname, Lname or *
9. DISTINCT	unique records with one column or combined column
10. ORDER BY	Fname desc
11. TOP	top 5, 10, or by %

Data Integrity

What does the term data integrity mean?

The term data integrity refers to the correctness and completeness of the data in a database. To preserve the consistency and correctness of its data, a RDBMS imposes constraints.

TYPES OF DATA INTEGRITY		EXAMPLES:
• Required data	→	not allowed to have missing data or NULL
• Entity integrity	→	primary key must contain unique value in each row
• Foreign key	→	has to be one of the value contained in the table it points to
• Validity checking	→	set of values permitted for column
• Consistency	→	updates to one table can corresponding change in linked tables
• Referential integrity	→	primary key in parent table links to foreign key in child table
• Business rules	→	quantity in a column must be above a set amount

Example:

Foreign constraint

ALTER TABLE tblPerson **ADD CONSTRAINT FK**_tblPerson_genderID

Foreign key (genderID) references tblGender(ID)

Default constraint

ALTER TABLE tblPerson A**DD CONSTRAINT DF**_tblPerson_GenderID **DEFAULT** 3 **FOR** genderID

SQL SELECT with Order By, Insert, Update and Delete Command

- SQL **INSERT** statement is used to add rows to a table. Insert can be used in several ways:
  - To insert a single complete row, OR insert a single partial row
- SQL **UPDATE** is used to update data in a row or set of rows specified in the filter condition.
  - Update tblLearner set ageRange = ‘20-30’ where ID = 1
- SQL **DELETE** is used to delete a row or set of rows specified in the filter condition.
  - Delete from tblPerson where ID = 4

Select Statement

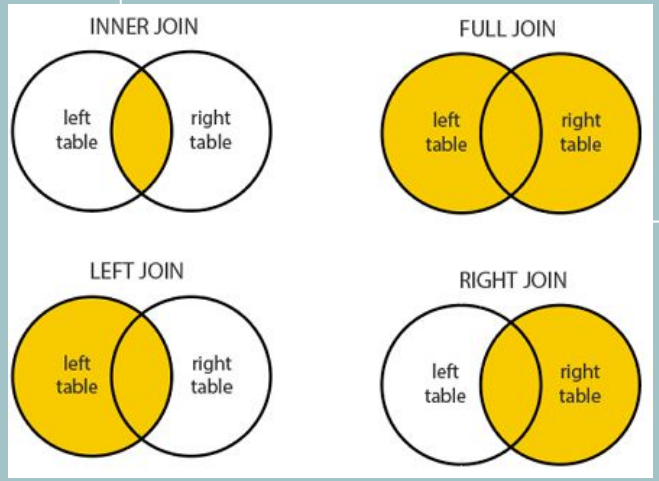
- SQL **SELECT** statement is used to retrieve data from table. For example, to retrieve data from the table given below, following SQL select statements are used:
- In SQL, the **WHERE** clause enables you to apply a filter.
  - “**SELECT * FROM** tblLearner **WHERE** condition; “
- The **WHERE** clause specifies which record(s) should be SELECTED FROM.
- SQL **Alias** name is used to give a alias name to a column or a calculated field. This is used with an aggregate function. In some situations, the aliases make your SQL statements simpler to write and easier to read.
  - **SELECT** jobTitle, sum(salary) as TtlSal **FROM** tblLearner

GROUP BY and Aggregate Functions

- **GROUP BY** is used to group identical rows in a column together
  - Placed **after WHERE** and **before ORDER BY** in a statement
- Often used with aggregate functions to sort the data in a meaningful way
- Example: Say there's a table called **tblLearner** and we're grouping by the column **Salary**
  - **Count ()** - Returns the total number of non-null values
  - **Sum () -** Returns the sum of all non-null values
  - **Avg () -** Returns sum() / count ()
  - **Min () -** Minimum non-null value
  - **Max ()** - Maximum non-null value

Id	Name	Salary
1	A	20
2	B	30
3	C	40

Joins



- A **join** combines rows from two or more tables by merging them through a common column
- **INNER JOIN**
  - Returns records that have matching values in both tables
- **FULL JOIN**
  - Returns all records when there is a match in left or right table records
- **LEFT JOIN**
  - Returns all records from left table, and any that match with the right table
- **RIGHT JOIN**
  - Returns all records from right table, and any that match with the left table

Syntax for Joins

```
SELECT                  Fname, Lname, Salary, Gender,Dept
FROM                    tblLearner
INNER/LEFT/RIGHT JOIN   tblDept
ON                      tblLearner.deptID=tblDept.ID;
```

Transactions

- Transactions are a group of tasks performed as a single execution
- The following commands are used to control transactions:
  - **BEGIN** - Indicates start point of transaction
  - **SET** - Places name on transaction
  - **COMMIT** - To save the changes
  - **ROLLBACK** - Reverses changes in case of an error; failsafe
  - **SAVEPOINT** - Creates points within the transaction to roll back to
  - **RELEASE SAVEPOINT** - Remove a savepoint

```
Create Procedure spHardWorks
As
Begin
        Begin try
                Begin transaction
                        Update tblLearner  set Salary  = '1000000'
                        Where  ID = 1

                        Update tblLearner set Salary = '1M'
                        Where Lname like '[H%]'

                        Update tblLearner set Salary = '1M'
                        Where Fname like '[S%]' and GenderID = 1
                Commit Transaction
        End Try
        Begin Catch
                Rollback Transaction
                Print 'Transaction Rolled Back'
        End Catch
```

Thank you for your time, hope this helps!