**Databases Introduction**

**Data** is a term for raw bits & pieces of information (i.e. – *Images, Words, Phone Numbers).*

**Databases** are Collections of Data organized into Files (*Tables*) as a logical way of accessing/managing/updating data. *They can be Small Scale/Local, Distributed across a network or Aurora Instance in the AWS Cloud.*

**Data Models** represent logical structure of data stored within databases. It has the following Types: *Relational, Semi-Structured, Entity Relationship, Object-Based.*

**Schema** defines the database organization, based on the data model describing the tables, columns, relationships & constraints.

**Relational & Non-Relational Databases**

|  |  |
| --- | --- |
| **Database Definitions** | |
| **Relational (SQL)** | Collection of Data Items with Predefined Relationships, requires fixed structure definition & stored in tables of rows/columns. |
| **Non-Relational (NoSQL)** | Doesn’t follow Relational Model, doesn’t require fixed structure definition & doesn’t use a table structure to store data. |

|  |  |
| --- | --- |
| **Database Properties** | |
| **Relational Databases** (i.e. – *MySQL, Aurora, Oracle, etc*) | **Non-Relational Databases** |
| * Natively Supports SQL * Provides Data Integrity * Supports Transactions   Used in Ecommerce, Customer Relationship Management & Business Intelligence Tools. | * Fraud Detection * Internet of Things (IoT) * Social Networks   NoSQL Examples include Amazon DB, MongoDB & Apache Hbase. |
| **Advantages & Disadvantages** | |
| * Known & Reliable Technology. * Simply-to-Write Complex Queries. * Well-Known SQL Language. * Well-Supported Transactions. * User Vertical Scaling. * Include Fixed Schema. | * Flexible Schema. * Good for Storing & Fast Retrieval of Massive Data of Different Types. * Horizontal Scaling. * Good Fit for Hierarchical Data. * Relatively New Technology. * Doesn’t Guarantee Data Integrity. * Bad Fit for Complex Queries/ Transactional Apps. |

**Database Management Systems (DBMS)**

**DBMS** is software/Database as a Service (DBaaS) providing database functionality. It avoids the cost of installing/maintaining servers through allowing data to be stored in the cloud instead of on-premises.

|  |  |  |
| --- | --- | --- |
| *Creating Databases.* | *Inserting Data into Database.* | *Storing/Retrieving/Updating/Deleting Data.* |
| *Hosted by 3rd Party Providers.* | *Reduced Cost of Maintenance & Faster.* | *Fully Managed (No Provisioning/Patching/etc).* |

|  |  |
| --- | --- |
| **Database Management Systems Variations** | |
| **Single-User** | i.e. – *Microsoft Access*. |
| **Multiple-User** | i.e. – *Oracle Database, Microsoft SQL Server, MySQL, IBM Db2.* |

**Database Interactions & Transactions**

|  |  |
| --- | --- |
| **Database Roles & Forms of Interaction** | |
| **Application Developer** | Creates Apps to populate/maintain data in DB according to an application’s functional requirements. |
| **End User** | User Reports created from information within DB *sometimes interacting directly*. |
| **Data Analyst** | Collects/Cleans/Interprets Data within DB System, *entering SQL Commands.* |
| **Database Administrator** | Designs/Implements/Administers/Monitors Data within DB System ensuring Consistency/Quality/Security *entering SQL Commands.* |

|  |  |
| --- | --- |
| **Data Interaction Models** | |
| **Client-Server** | **Three-Tier Web Application** |
| 1. Users use Devices with Client Apps using SQL to request. 2. Apps use SQL sent to server across network to communicate with DB. 3. Server runs DBMS to receive request, process SQL & return response. | 1. User Device with Web Browser, Webpage captures input & requests. 2. Web Server gathers info & forwards to app server for processing. 3. App Server Web App receives request & runs SQL command for it, sending to the DB Server. 4. DBMS receives request, process SQL & returns response to App Server. 5. App Server Web App processes & returns to Web Server. 6. Web Server formats into Webpage. 7. Web Browser displays Webpage. |

**Transactions** are collections of changes made to a Database that must be performed as a logical unit of work. It has various statuses that are listed below:

|  |  |
| --- | --- |
| **Transaction Statuses** | |
| **Active** | Occurs in the Initial State of every transaction & when the transaction is ran. |
| **Failed** | Occurs when any checks made by the Database Recovery System fail. |
| **Aborted** | Occurs if it’s in a Failed State & it rolls back to its original state before running. |
| **Committed** | Occurs when all Transaction Operations have been Performed Successfully. |

**Transaction Use Cases** are as follows:

1. Run Operations so the Database Never contains Partial Operation Results. If 1 fails, it’s rolled back to original state, if no errors the full statement set changes the Database.
2. Provide Isolation between Programs accessing Database Simultaneously, without isolation the outcomes may be incorrect.

|  |  |
| --- | --- |
| **Transaction Properties** | |
| **Atomicity** | Ensures Changes Successfully Completed all at once or not at all. |
| **Consistency** | Ensures Changes don’t violate Database Integrity including constraints. |
| **Isolation** | Ensures Transactions Isolated to not interfere with each other. |
| **Durability** | Ensures as soon as Transaction Committed, the change is permanent. |

**Tables & Data Types**

**SQL** can perform many necessary actions within a database, and has sublanguage groups that are as follows:

|  |  |
| --- | --- |
| **SQL Sublanguage Groups** |  |
| **Data Manipulation Language (DML)** | Used to View/Add/Change/Delete Data in a Table. *SELECT/INSERT/UPDATE/DELETE Commands.* |
| **Data Definition Language (DDL)** | Define/Maintain Database Objects (Schema). *CREATE/ALTER TABLE/DROP Commands.* |
| **Data Control Language (DCL)** | Control Access to Data in Database. *REVOKE/GRANT Commands.* |

|  |  |  |
| --- | --- | --- |
| **SQL Data Type** | **Description** | **Example** |
| **INTEGER** | Represents an Integer. Max/Min Values depend on the DBMS. | 12000 |
| **SMALLINT** | Smaller Range of Values than INT. | 10 |
| **BIGINT** | Larger Range of Values than INT. | 45345345 |
| **CHAR(length)** | Represents a fixed-length character string. | ‘United States’ |
| **VARCHAR(length)** | Represents a Variable Length Character String. |  |
| **CLOB(length)** | Large Character Length String/Text Data with potential length in order of GBs. |  |
| **DECIMAL(p,s)** | Represents Exact Number with (p)recision and (s)cale. A Decimal Number. | (10,3) can be 1234567 or 1234567.123. |
| **FLOAT** | Represents a Floating-Point Number. | 3.14159 |
| **REAL** | Same as FLOAT but DBMS sets precision. |  |
| **DATETIME** | Represents a Date & Time Combination. | ‘2023-07-18 17:43:02’ |
| **DATE** | Represents a Date. | Yyyy-mm-dd |
| **TIME** | Represents a Time-of-Day without Time Zone. | Hh:mm:ss |
| **TIMESTAMP** | Represents Moment in Time with a date & time. | Yyyy-mm-dd hh:mm:ss |

**Identifiers** represent User-Created Object Names in contrast to Language Keywords/Statements. *Language keywords & commands are often capitalized, identifiers are lowercase.*

A screenshot of a computer code

Description automatically generated

|  |  |
| --- | --- |
| **Data Constraints** | |
| **NOT NULL** | Ensures a Column doesn’t hold a NULL value. |
| **UNIQUE** | Requires Column(s) have values unique to them. |
| **DEFAULT** | If Column Value isn’t provided, DEFAULT provides a value when the DBMS inserts a row. |

**Reserve Terms** are SQL Keywords/Symbols with Specific Meanings when processed. *They shouldn’t be used in names of database objects for clarity & avoiding errors*

i.e. - *#,;,:,@,ADD,CLOSE,DATABASE,EXISTING*

|  |  |
| --- | --- |
| **TABLE Command Statements** | |
| **CREATE TABLE** | Creates a Table within a Database. |
| **DROP TABLE** | Removes a Table within a Database. *DROP TABLE table\_name1 [, table\_name2] ... [, table\_nameN]* |
| **column\_name** | Creates a Column within a Database. *column\_name DATA\_TYPE [(length)][NOT NULL][DEFAULTvalue]* |

**Primary Keys** are Special Columns with Unique Values for each row & uniquely identifies the row.

**Foreign Keys** are Special Columns that hold Primary Keys from another table, creating a relationship between the two.

**Referential Integrity** is a Database Quality where all non-NULL Foreign Keys match an Existing Primary Key Value.

**Inserting Data into a Database**

**A .csv file** is a simple text file with information separated by commas, *the process of importing one is as follows:*

1. Verify .csv has Data Matching Table Column No. & Data Type in each Column.
2. Create MySQL Table with Name corresponding to desired importing .csv.
3. Import with LOAD DATA Statement.
   1. *IGNORE 1 ROWS* If Row 1 is Column Headers.
   2. *TERMINATED BY ‘/N’* If Rows are Terminated by Newline Character.

|  |  |
| --- | --- |
| ***Importing a .CSV File*** | ***Exporting a .CSV File*** |
|  |  |

|  |  |
| --- | --- |
| **Insertion Commands** | |
| **LEFT/RIGHT/TRIM** | Selects Certain Elements of Strings & Remove Certain Characters. |
| **CONCAT** | Combine Several Column Strings & Combine them. |
| **LOWER** | Force Whole String Characters to Lowercase. |
| **UPPER** | Force Whole String Characters to Uppercase. |
| **DESCRIBE** | Provides Description of Specified Table/View. *DESCRIBE country;* |
| **INSERT INTO** | DML Command which Inserts Single/Multiple Records into a Table. *INSERT INTO tableName (col\_name, col\_name, etc) VALUES (‘val\_1’, ‘val\_2’, etc)* |
| **NULL** | Placeholders/Represent Missing Value to Improve Readability, Clarifying Meaning/Actions of Conditional Statements. |

**Selecting Data from a Database**

|  |  |
| --- | --- |
| **SELECT Command & Description** | |
| **SELECT** | Used to Access Subset or Rows/Columns/Both.  The format is *SELECT id, name, countrycode FROM city WHERE countrycode = ‘BRA’;*   * A Single Column: *SELECT colname1 FROM table\_name;* * A single + Optional Columns: *SELECT colname1[, colname2, etc] FROM table\_name;* * All Columns: *SELECT \* FROM table\_name;* |
| **FROM** | From a Location, Retrieve all Data. |
| **WHERE** | *SELECT id, name, countrycode FROM city WHERE countrycode = ‘BRA’;* Get Table Data Ignoring all Rows except the rows where the query is met, Returning Only ID, NAME and COUNTRYCODE columns: SELECT |
| **GROUP BY** | *SELECT continent, COUNT(\*) FROM country GROUP BY continent;* Select Rows from Table, Grouping by Continent, Counting Row No. in each group. |
| **HAVING** | *HAVING COUNT(\*) > 1;* Filters GROUP BY to select continents with more than one country after rows grouped by continent. |
| **ORDER BY** | *ORDER BY id;* Get all Data in Table Ordered by ID, sorts queries by 1 or more column in ascending/descending order. |

**Performing a Conditional Search**

Using the WHERE feature within SELECT, SQL uses the same Arithmetic (+, -, etc), Comparison (>, <, etc) & Logical (AND, OR etc) Operators as Python when searching for thing in a query. However, there are some unique ones:

|  |  |
| --- | --- |
| **Unique SQL Operators** | |
| **IN** | Matches Multiple Data Items in a WHERE clause using a List of Conditional Values. |
| **LIKE** | Matching Multiple Data Items within a WHERE clause using Partially Matching Conditional Values (Wildcards). |
| **BETWEEN** | Specifies a Range on Matching Conditional Values. |

**Order Precedence** Dictates which Operator you deal with first, *like BIDMAS for SQL:*

***A person standing next to a screen

Description automatically generated***

***Aliases are used to Assign Temporary Names to a Table/Column within a SQL query.*** i.e. – *SELECT name, lifeexpectancy, lifeexpectancy+5.5 AS newlifeexpectancy (They require ‘’ if there’s spaces).*

There are also **NULL Values** which can be used. i.e. *– WHERE lifeexpectancy is NULL.*

**Working with Functions**

|  |  |
| --- | --- |
| **Common Aggregate Functions** | |
| **AVG** | Returns Set Average, Can be used for Average Population for Cities within Specified Country. |
| **COUNT** | Returns Item Number in Set, Can be used for Total City No. in Specified Country. |
| **MAX** | Returns Set Max Value, Can be used to find City with Highest Population. |
| **MIN** | Returns Set Min Value, Can be used to find City with Lowest Population. |
| **SUM** | Returns Total of all Values in Set, Can be used to find Total Population for all Cities within Specified Country. |
| **DISTINCT** | Selects only Unique Values. |
| **TRIM** | Removes blank spaces on right end of String Data Values. |
| **LTRIM** | Removes blank spaces of left end of String Data Values. |

**Retrieving Data**

|  |  |
| --- | --- |
| **Set Operators** | |
| **UNION** | Combines Several Result Sets into a Single Set without duplicates. |
| **UNION ALL** | Combines Several Result Sets into a Single Set with duplicates. |
| **INTERSECT** | Combine Two Result Sets & Return Data Common in both Data Sets. |
| **MINUS** | Combine Two Results & Return Data from First that isn’t present in other. |
| **INNER JOIN** | Return Rows Matching in Both Tables. |
| **LEFT JOIN** | Return All Rows from Left Table. |
| **RIGHT JOIN** | Return All Rows from Right Table. |
| **FULL JOIN** | Return All Rows from Both Tables. |

**Amazon Relational Database Service (RDS)**

*A Managed Database Service which Sets Up & Operates a Relational Database in the Cloud.* This is done using**DB Instances** which are are Isolated Database Environments running in the Cloud.

|  |  |  |
| --- | --- | --- |
| **Web/Mobile Apps** | **Ecommerce Apps** | **Mobile/Online Games** |
| High Throughput.  Massive Storage Scalability.  High Availability. | Low-Cost Database.  Data Security.  Fully-Managed Solution. | Rapid Growth Capacity.  Automatic Scaling.  Database Monitoring |
| **Data Types Supported:** | *MySQL, Aurora, Microsoft SQL Server, PostgreSQL, MariaDB & Oracle.* | |

|  |  |
| --- | --- |
| **Backup Options** | |
| **Automatic** | Creates Automated Backups (Data/Transaction Logs) of DB Instances during Backup Window. |
| **Manual** | Creates Storage Volume Snapshots of DB Instances. |

To Create a DB Instance, you’d use the following steps:

1. In the Configuration Section *choose MySQL*.
2. Complete All Other Configuration Questions & *choose CREATE*.
3. Depending on Instance Class & Storage Amount can take 20 minutes before Available.

*High Availability offered by AWS RDS through Multi-AZ Deployment allows for replication, failover, read replicas and scaling (instance-class/storage capacity).*

**Aurora** is a Relational Database Engine using the same code/tools/applications as existing MySQL Databases, with a high-performance subsystem and is created with clusters (consisting of 1/more db instances which are either primary dB instances or aurora replicas).

**Amazon DynamoDB**

*A Fully-Managed, Serverless Key-Value NoSQL Database which improves performance by keeping data in memory, keeps data secure by encrypting data at rest and protects data with backups & automated copying between AWS regions.*

* *Uses Tables like Relational which have names and primary keys specified during creation.*
* *Columns are called Attributes.*

|  |  |
| --- | --- |
| **Amazon DynamoDB terms** | |
| **Items** | Attribute groups uniquely identifiable amongst items. |
| **Single Primary Key** | Consist of one attribute called the Partition/Hash Key. |
| **Composite Primary Key** | Consists of two attributes, the first being Partition/Hash Key and the second being the Sort Key/Range Attribute. |
| **Partitioning** | Done to data splitting within tables, new partitions are automatically added when existing ones are filled with data by the Partition Attribute (Friend ID) |
| **Global Tables** | Create DynamoDB automatically replicated across user-choice of AWS. It must be owned by a single account. Tables within this collection are called Replicas. |