|  |  |
| --- | --- |
| * Database is a collection of related data * **DBMS (Database Management System)** * Special software program that helps users create and maintain a database * **C.R.U.D – create, read, update and delete** * **Two types of databases** * **1. Relational databases ie SQL databases-** organizes data into one or more tables. * **2. Non-relational ie noSQL databases –** organize data into anything but a traditional table. * **RDBMS – Relational DBMS** * Helps create and maintain a relational database eg MySQL, oracle. * They use Structures Query Language SQL * SQL is language used to interact with RDBMS. * **Non-relational databases.** * Stores data in anything but a table. Eg key and value or JSON * NRDBMS – eg mongoDB, firebase, dynamodb * It is implementation specific and has no standard language. * **Core concepts** * **Database query** * Queries are requests made to the DBMS for information * **Primary key**  - an attribute that uniquely defines a row in a table. * **Surrogate key –** a key that does not have any maning in the real world and can be used as primary key **opposite is natural key eg ID** * **Foreign key -**  stores tehe primary key of a row in another table and allows us to link to that table where the foreign key is a primary key * **Composite key -**  primary key that is made up of two attributes / columns   **SQL BASICS**   * SQL is a language that is used to interact with RDBMS * SQL varies but there is a formal definition but not all RDBMS follow the same implementation. * SQL is a hybrid language of 4 languages   + **Data Query Language (DQL) –** used to query for information   + **Data definition Language (DFL) –** used to define schemas   + **Data Control Language (DCL) –** used to control access to the data   + **Data manipulation Language (DML) –** used for inserting, updating and deleting data.   **QUERIES**   * Set of instructions given to RDBMS telling it what information to retrieve.   **DATA TYPES**  **BASIC**   * INT – integer / whole number * DECIMAL(M, N) – decimals (total digits, digits after decimal) * VARCHAR (l) – string of text of length l * BLOB – Binary large object * DATE – YYYY – MM - DD * TIMESTAMP – used for recording when things happen YYYY-MM-DD HH:MM:SS   **CREATE TABLES**   * CREATE TABLE (name()) * DROP TABLE name; * DESCRIBE TABLE name * ALTER TABLE name ADD column datatype; * ALTER TABLE name DROP column; * INSERT INTO student VALUES (2, 'cate', 'sociology'); * INSERT INTO student (id , name) VALUES (2, 'cate'); - used when one of the values in the table is unkown   **CONSTRAINTS**   * NOT NULL is used to ensure that the data is not empty * UNIQUE is used to ensure unique values are entered * DEFAULT ‘Default’ – creates a default if the value is blank * AUTO\_INCREMENT - increments data automatically when data is added.   **UPDATING AND DELETE DATA IN TABLES**   * UPDATE student SET major = ‘Bio’ WHERE major = ‘Biology’   **BASIC QUERIES**  **SELECT STATEMENTS**   * SELECT name / table.name FROM name * Add ORDER BY name – arranges the results * Add DESC or ASC for descending or ascending * Use LIMIT to limit results * Use column IN () – returns all values that are in the parenthesis | **Functions and aggregation**   * Called to perform a function like count, avg * Used with GROUPBY to get better results   **Wildcards**   * A way of defining patterns * Used with LIKE keyword * % - represents any number of characters * \_ - represents one character   **Union**   * Used to combine two select statements into one output. * Restrictions of datatype and output columns apply   **Joins**   * Used to join two tables and select data into one output. * Define the specific table using table.column. * Example * **SELECT employee.id, employee.fname, branch.id**   **FROM employee**  **JOIN branch**  **ON employee.id = branch.id;**   * **Types of join** * Inner join – example above – only rows in that are matched are shown * Left join – all rows in the left table are used. Keyword **LEFT JOIN**  and only the matched in the right are shown * Right join – all rows in the right table are shown and only matched from the left are displayed * Full outer join – all of the rows in the left and all in the right are displayed   **Nested Queries**   * Use multiple select statement. * Example   **SELECT employee.fname, employee.lname**  **FROM employee**  **WHERE employee.id IN (**  **SELECT sales.employee\_id**  **FROM sales**  **WHERE sales.total\_sales > 30000**  **)**  **SELECT client.client\_name**  **FROM client**  **WHERE client.branchid = (**  **SELECT branch.branchid**  **FROM branch**  **WHERE branch.mgr\_id = 102**  **LIMIT 1**  **);**  **ON DELETE**   * ON DELETE SET NULL – situation where when a foreign key is located the value is set to null * ON DELETE CASCADE – when the foreign key is deleted on another table, the entire row data on the current table is deleted   **TRIGGERS**   * Used for automation * Example * Normal delimiter is ; at the end of an SQL statement * Start by changing delimiter * Next create the trigger. * Delimiter is changed to differentiate the SQL statements created in the trigger from the end of the trigger. * End Trigger * Change delimiter back   **DELIMITER $$**  **CREATE**  **TRIGGER my\_trigger BEFORE INSERT**  **ON employee**  **FOR EACH ROW BEGIN**  **INSERT INTO trigger\_test VALUES (‘added new ’);**  **END $$**  **DELIMITER ;**   * Triggers can be used with if and ifelse statements * Used for Insert, Dlelete, Update * To remove DROP TRIGGER name   **ER DIAGRAMS**   * Takes data requirements to data schema * Representations  1. Rectangle – entity /table 2. Oval – attributes 3. Line – connector 4. Oval with underline – attribute that is primary key 5. Oval connected to an oval – composite attribute e.g name attribute has composite attributes fname and lname 6. Oval inside an oval – multivalued attribute e.g club as an attribute; a student can belong to many clubs 7. Dotted oval – derived attribute from other attributes. Eg honors in school derived from grades 8. Diamond – denotes relationships between entities   Relationships are defined by verbs   1. Single line – partial participation eg. A student can take a class or not 2. Double line – total participation eg a class must have a student 3. Relationship cardinality –   1:1  1:N  N:M  N and M are many   1. Rectangle in a rectangle – weak entity that depends on another entity. It cant exist on its own   **CREATING AN ER DIAGRAM**   * Create the diagram from the user requirements * Use it to create schema * Weak entities have to have a composite key of their primary key and the primary key of the parent * For schema   + **Start with 1:1 relations but favor total relations side. Which means add the primary key of partial as foreign in total**   + **1: N relations – add primary key of 1 as foreign key in N**   + **N:M – create a table that has a combination of both the primary keys and include relation attributes.** |