**SQL**

* **SQL** = Structured Query Language.

**SQL** হচ্ছে ডাটাবেস অ্যাক্সেস এবং ম্যানিপুলেট করার জন্য একটি language.

এর মাধ্যমে database এ ডাটা insert, update, table create, এবং database থেকে ডাটা retrieve, delete এবং ডাটাবেস create করা যায়।

* **Some of The Most Important SQL Commands**
* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database
* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index
* **SQL Statement**
* SELECT Statement: database থেকে ডাটা select করার জন্য SELECT Statement ব্যবহার করা হয়। ‍

Specific column select করার জন্য - SELECT CustomerName, City FROM Customers;

Table থেকে সকল ডাটা select করার জন্য - - SELECT \*FROM Customers;;

* **DISTINCT Statement:** database থেকে unique ডাটা return করে - SELECT DISTINCT name FROM `customers`।

**WHERE clause:** records কে filter করার জন্য ব্যবহার WHERE clause করা হয়। WHERE ক্লয়াজ ব্যবহার করা হয় ডেটাবেস থেকে নির্দিষ্ট তথ্য বা রেকর্ড রিট্রিভ করার জন্য, যা একটি বা একাধিক শর্তাদি প্রয়োগ করে সম্পাদিত হয়।

* **ORDER BY Keyword:** ডেটাবেস থেকে প্রাপ্ত রেকর্ডগুলি সর্ট করার জন্য। ORDER BY ডাটাকে ডিফল্ট ভাবে ascending order সর্ট করে।
* **AND Operator:** একাধিক condition উপর ভিত্তি করে ডেটার filter করার জন্য AND Operator ব্যবহার করা হয়। example - SELECT \* FROM `customers` WHERE email = 'm@gmail.com' and mobile = '66';
* **OR Operator :** একাধিক condition উপর ভিত্তি করে ডেটার filter করার জন্য OR Operator ব্যবহার করা হয়। example - SELECT \* FROM `customers` WHERE email = 'm@gmail.com' or mobile = '66';
* **NOT Operator:** **SQL NOT Operator** ব্যবহার করা হয় **ডাটাবেস থেকে নির্দিষ্ট ডেটা বাদ দিতে**। এটি একটি **logical operator** যা **WHERE** clause-এ ব্যবহার করা হয়। NOT operator-এর সাথে যেকোনো **Boolean expression** ব্যবহার করা যায়। **example-**

SELECT \* FROM `customers` WHERE id NOT IN(14,19);

NOT IN(), NOT BETWEEN, NOT LIKE, NOT IS NULL,

NOT Less Than- NOT NOT CustomerID < 50, NOT Greater Than- NOT CustomerID > 50

* **NULL Value** - SQL-এ NULL ভ্যালু হলো এমন একটা বিশেষ মান যা কোনো কলামে \*\* কোনো ডাটা নেই\*\* সেটা নির্দেশ করে। এটা কোনো ফাঁকা স্ট্রিং (empty string) বা শূণ্য (zero) এর মতো নয়। NULL মানের অর্থ হলো সেই কলামের জন্য কোনো তথ্য উপলব্ধ নেই বা প্রযোজ্য নয়।

এসকিউএলে, টেবিলের কলামগুলো ডাটা ধরে রাখে। কখনো কখনো কোনো কারণে কোনো নির্দিষ্ট রেকর্ডের জন্য সেই কলামে কোনো তথ্য নাও থাকতে পারে। এই অবস্থায়, সেই কলামের জন্য NULL মান সেট করা হয়।

* **SQL UPDATE** - SQL UPDATE statement হলো ডাটাবেসের টেবিলের মধ্যে  existing records পরিবর্তন করার জন্য ব্যবহৃত একটা কমান্ড. এটি ডাটা ম্যানিপুলেশন ল্যাংগুয়েজ (DML) এর অংশ, যা টেবিলের গঠন না বদলে কেবলমাত্র তথ্য পরিবর্তন করে।
* **DELETE Statement -** SQL DELETE হলো ডাটাবেসের মধ্যে  existing records থেকে ডাটা মুছে ফেলার জন্য ব্যবহৃত একটা SQL কমান্ড। এটি ডাটা ম্যানিপুলেশান ল্যাঙ্গুয়েজের (DML) অংশ, যা ডাটাবেসের ডাটা পরিবর্তন করতে ব্যবহৃত হয়।

**DELETE** কীভাবে কাজ করে?

DELETE কমান্ডটি নির্দিষ্ট টেবিল থেকে রেকর্ড মুছে ফেলতে ব্যবহার করা হয়। এটি একটি শর্তের উপর ভিত্তি করে নির্দিষ্ট রেকর্ড বা টেবিলের সবগুলি রেকর্ড মুছে ফেলতে পারে।

* **SQL LIMIT ক্লজ** - SQL ডেটাবেজ থেকে data সেটের আকার সীমাবদ্ধ করতে LIMIT ক্লজ ব্যবহার করা হয়। এটি প্রাথমিকভাবে নির্দিষ্ট করা সংখ্যক রো (Rows) return করে।
* **SQL Aggregate Functions –** একটি ডাটা সেট থেকে ক্যালকুলেশন করে একটি single value return করে।
* MIN() - returns the smallest value within the selected column
* MAX() - returns the largest value within the selected column
* COUNT() - returns the number of rows in a set
* SUM() - returns the total sum of a numerical column
* AVG() - returns the average value of a numerical column
* COUNT() ফাংশন null value ignore করে।
* **SQL LIKE Operator -** SQL LIKE অপারেটরটি কোনো টেবিলের কলামের মধ্যে কোনো নির্দিষ্ট প্যাটার্ন অনুসন্ধান করতে WHERE ক্লজের সাথে ব্যবহার করা হয়। এটি হল এমন একটি উপায় যা আপনাকে ঠিক সঠিক মিলের চেয়ে আরো নমনীয়ভাবে ডাটা খুঁজে বের করতে দেয়। LIKE অপারেটরটির সাথে সাথে বিশেষ কিছু অক্ষর ব্যবহার করা হয়, যা "wildcard" নামে পরিচিত। এই ওয়াইল্ডকার্ডগুলো আপনার নির্দিষ্ট করা প্যাটার্নের সাথে মেলে এমন একাধিক মানের সাথে মিল খুঁজতে সাহায্য করে।
* **SQL Wildcard Characters -** SQL ওয়াইল্ডকার্ড হল বিশেষ ক্যারেক্টার যা স্ট্রিং এর মধ্যে এক বা একাধিক ক্যারেক্টারের প্রতিস্থাপন হিসাবে ব্যবহৃত হয়। LIKE অপারেটরের সাথে এরা ব্যবহৃত হয় টেবিল থেকে ডাটা নির্বাচন করার সময় স্ট্রিং এর মধ্যে নির্দিষ্ট প্যাটার্ন অনুসন্ধান করতে।
* **SQL IN Operator -** SQL IN অপারেটর হলো WHERE cluse এ multiple ভালু নির্দিষ্ট করার জন্য ব্যবহৃত একটি way। এটি একাধিক OR কন্ডিশনের একটি সংক্ষিপ্ত রূপ।
* **SQL BETWEEN Operator -** SQL BETWEEN অপারেটর ডাটাবেসের টেবিল থেকে নির্দিষ্ট পরিসীমার মধ্যে থাকা ডাটা খুঁজে বের করতে ব্যবহৃত হয়। এটি WHERE ক্লজের সাথে কাজ করে এবং একটি সহজ উপায়ে ফলাফল ফিল্টার করতে দেয়।
* **SQL aliases -** হল সিকুয়াল টেবিলের নাম এবং কলাম নেইম কে করেন্ট অবস্থানে কিংবা স্থানীয় অংশের একটি নাম দেয়া। এই অ্যালিয়াসগুলি ব্যবহার করা হয় কুয়েরি পলকে কিংবা টেবিলের নাম লেখার সময় যাতে সংক্ষেপে ব্যবহার করা যায়। এটি কুয়েরি লেখার পার্থক্য তৈরি করে এবং সংক্ষেপে লেখা সুবিধা দেয়
* **SQL JOIN -** SQL JOIN হলো এমন এক ধরণের ক্যuerী, যা দুটি বা তার বেশি টেবিলের ডাটা একত্রিত করে নতুন ডাটা সেট তৈরি করে। এই JOIN কাজ করে দুটি টেবিলে থাকা common INNER JOIN: এটি এমন JOIN, যা কেবল সেই রো (row) গুলোকেই ফলাফল দেয় যেগুলোর মধ্যে দুটি টেবিলেই মিল রয়েছে।
* **LEFT JOIN:** এটি এমন JOIN, যা বাম দিকের টেবিল থেকে সব রো (row) সহ, মিলি রয়েছে এমন ডান দিকের টেবিলের রো (row) গুলো ফলাফল দেয়। বাম দিকের টেবিলে মিল না থাকলে, ডান দিকের টেবিলে সেই কলামের জন্য NULL মান দেখাবে।
* **RIGHT JOIN:** এটি LEFT JOIN এর বিপরীত। এটি ডান দিকের টেবিল থেকে সব রো (row) সহ, মিলি রয়েছে এমন বাম দিকের টেবিলের রো (row) গুলো ফলাফল দেয়। বাম দিকের টেবিলে মিল না থাকলে, বাম দিকের টেবিলে সেই কলামের জন্য NULL মান দেখাবে।
* **FULL JOIN:** এটি এমন JOIN, যা দুটি টেবিলেই থাকা সমস্ত রো (row) ফলাফল দেয়, ভেবে মিল আছে কিনা।
* **SQL সেলফ জয়েন -** (Self Join) হলো একটি SQL কোয়েরি প্রযুক্তি যেখানে একটি টেবিলের সাথে নিজের নিজেকে জয়েন করা হয়। এটি মূলত একটি টেবিলের ভেতরের ডেটা সেটের মধ্যে সম্পর্ক নির্ধারণে ব্যবহৃত হয়। কলামের উপর ভিত্তি করে।
* **SQL UNION Operator -** SQL UNION অপারেটর দুটি বা ততোধিক সিলেক্ট স্টেটমেন্টের output কে একটি সিঙ্গেল result সেটে মার্জ করে। UNION অপারেটর কেবল সমান সংখ্যক এবং সমান ধরনের কলামের মধ্যেই কাজ করতে পারে। প্রতিটি সিলেক্ট স্টেটমেন্টের ফলাফলের কলামের সংখ্যা, ধরণ এবং ক্রম সমমান হতে হবে।

SELECT name, price FROM `product\_two` UNION SELECT name, price FROM product\_one;

* **SQL GROUP BY Statement -** GROUP BY ক্লজ SQL-এ একটি ডেটাবেস থেকে ডেটা গ্রুপ করার জন্য ব্যবহৃত হয়। এটি একই ধরণের ডেটা একত্রিত করে এবং প্রতিটি গ্রুপের জন্য output প্রদান করে।

SELECT price, sum(price) as "total price" FROM `product\_one` GROUP BY price;

* **SQL HAVING Clause -** HAVING clause ব্যবহার করা হয় GROUP BY clause দ্বারা গ্রুপ করা ডেটা সেটের উপর aggregate functions ফিল্টার করার জন্য। এটি WHERE clause এর মত কাজ করে, তবে WHERE clause ব্যবহার করে GROUP BY clause এর পূর্বেই ডেটা ফিল্টার করা হয়।

SELECT name,avg(price) FROM `product\_one` GROUP BY name HAVING sum(price) > 10;

* **SQL EXISTS Operator -** EXISTS অপারেটরটি একটি subquery ব্যবহার করে পরীক্ষা করে যে একটি টেবিলে একটি নির্দিষ্ট শর্ত পূরণ করে এমন কোন রেকর্ড আছে কিনা।

EXISTS অপারেটরটি subqueries ব্যবহার করে ডেটা ফিল্টার করার জন্য একটি শক্তিশালী হাতিয়ার। এটি JOIN operations এর চেয়ে দ্রুত হতে পারে কারণ এটি subquery কে কেবলমাত্র প্রয়োজনীয় তথ্য খুঁজে পেতে ব্যবহার করে।

EXISTS অপারেটরটি কেবলমাত্র true বা false ফেরত দেয়। এটি কোন ডেটা ফেরত দেয় না।

EXISTS অপারেটরটি subqueries এর সাথে ব্যবহার করা হয় যা 1 রেকর্ড বা 0 রেকর্ড ফেরত দেয়।

EXISTS অপারেটরটি IN অপারেটরের বিকল্প হিসাবে ব্যবহার করা যেতে পারে।

https://www.geeksforgeeks.org/sql-server-exists/

* **SQL ANY and ALL Operators-**

ANY **এবং** ALL অপারেটরগুলো SUBQUERY-এর সাথে ব্যবহৃত হয়। SUBQUERY হলো একটি SQL প্রশ্ন যা অন্য SQL প্রশ্নের ভেতরে ব্যবহৃত হয়।

ANY অপারেটর TRUE retrun করে যদি SUBQUERY-এর ফলাফলে **কমপক্ষে একটি** মান SUBQUERY-এর শর্ত পূরণ করে।

ALL অপারেটর TRUE retrun করে যদি SUBQUERY-এর ফলাফলের **সকল** মান SUBQUERY-এর শর্ত পূরণ করে।

* **INSERT INTO SELECT Statement -** INSERT INTO SELECT স্টেটমেন্ট ব্যবহার করে একটি টেবিলে ডেটা অন্য টেবিল থেকে কপি করা হয়।
* **SQL CASE Expression -** এটি ডেটাবেস থেকে রেকর্ড নির্বাচন করার সময়, শর্তের উপর নির্ভর করে বিভিন্ন মান ফেরত দিতে ব্যবহার করা হয়।
* একটি বা একাধিক শর্ত নির্ধারণ করা হয়।
* প্রতিটি শর্তের সাথে একটি মান যুক্ত করা হয়।
* একটি ডিফল্ট মান (ঐচ্ছিক) নির্ধারণ করা হয় যদি কোন শর্তই পূরণ না হয়।
* CASE Expression মূল্যায়ন করে এবং শর্তের সাথে মিলে যায় এমন প্রথম মান ফেরত দেয়।
* কোন শর্তই না মিললে ডিফল্ট মান ফেরত দেয়।
* **IFNULL() এবং ISNULL() –**

**IFNULL():** expression1 যদি NULL হয়, তাহলে expression2 ফেরত দেয়। অন্যথায়, expression1 ফেরত দেয়।

IFNULL(expression1, expression2)

**ISNULL():** expression1 যদি NULL হয়, তাহলে 1 ফেরত দেয়। অন্যথায়, expression1 ফেরত দেয়।

ISNULL(expression1)

* **SQL Stored Procedures -** SQL Stored Procedures হলো পূর্ব-সংজ্ঞায়িত কোডের ব্লক যা ডেটাবেজের সাথে ইন্টারঅ্যাক্ট করার জন্য ব্যবহৃত হয়। এগুলো SQL স্টেটমেন্টের সমন্বয়ে গঠিত হয় যা একসাথে সংকলিত হয় এবং একটি নাম দেওয়া হয়।
* **Stored Procedures** এর সুবিধা:
* কোড পুনরায় ব্যবহার:
* একই কাজ বারবার করার জন্য একই কোড বারবার লেখার প্রয়োজন হয় না।
* **SQL CREATE DATABASE Statement –এই** Statement এর মাধ্যমে নতুন ডাটাবেস তৈরী করা হয়।
* **DROP DATABASE Statement -** SQL database থেকে existing database ডিলেট করার জন্য
* **SQL BACKUP DATABASE** হল একটি SQL কমান্ড যা একটি ডাটাবেজের একটি ব্যাকআপ তৈরি করে। এই ব্যাকআপটি পরবর্তীতে ডাটাবেজটি পুনরুদ্ধার করতে ব্যবহার করা যেতে পারে যদি এটি দুর্ঘটনাবশত মুছে ফেলা হয়, নষ্ট হয় বা দূষিত হয়।
* **SQL BACKUP DATABASE** কমান্ডটি নিম্নলিখিত সিনট্যাক্স ব্যবহার করে:
* SQL

Syntax--

BACKUP DATABASE database\_name

TO backup\_device = 'backup\_location'

WITH (

MEDIA = backup\_type,

FORMAT = backup\_format,

...

);

* **database\_name** হল ব্যাকআপ নেওয়া হবে এমন ডাটাবেজের নাম।
* **backup\_device** হল ব্যাকআপ সংরক্ষণের জন্য ডিভাইসের পথ। এটি একটি ফাইল পথ, একটি টেপ ডিভাইসের নাম বা অন্য কোনও বৈধ ব্যাকআপ ডিভাইস স্পেসিফিকেশন হতে পারে।
* **backup\_type** হল ব্যাকআপের জন্য ব্যবহৃত মিডিয়ার ধরণ। এটি DISK, TAPE, URL ইত্যাদির মতো মানগুলির মধ্যে একটি হতে পারে।
* **backup\_format** হল ব্যাকআপের জন্য ব্যবহৃত বিন্যাস। এটি FULL, DIFFERENTIAL, TRANSACTION LOG ইত্যাদির মতো মানগুলির মধ্যে একটি হতে পারে।
* **SQL CREATE TABLE Statement -** এই Statement এর মাধ্যমে ডাটাবেসে নতুন টেবিল তৈরী করা হয়।
* **SQL DROP TABLE Statement -** SQL database থেকে existing table ডিলেট করার জন্য এই ‍statement ব্যবহার করা হয়।
* **SQL ALTER TABLE Statement -** ডেটাবেস ডিজাইনের পরিবর্তন করার জন্য ALTER TABLE স্টেটমেন্ট ব্যবহার করে।

**যার মধ্যে রয়েছে:**

**কলাম যোগ করা:** নতুন কলাম যোগ করার জন্য ALTER TABLE ADD ব্যবহার করা হয়।

**কলাম মুছে ফেলা:** ALTER TABLE DROP ব্যবহার করে বিদ্যমান কলাম মুছে ফেলা হয়।

**কলাম পরিবর্তন করা:** ALTER TABLE MODIFY ব্যবহার করে কলামের ডেটা টাইপ বা অন্যান্য বৈশিষ্ট্য পরিবর্তন করা হয়।

**টেবিলের নাম পরিবর্তন করা:** ALTER TABLE RENAME ব্যবহার করে টেবিলের নাম পরিবর্তন করা হয়।

**কনস্ট্রেন্ট যোগ করা:** ALTER TABLE ADD CONSTRAINT ব্যবহার করে টেবিলের কনস্ট্রেন্ট যোগ করা হয়।

কনস্ট্রেন্ট মুছে ফেলা: ALTER TABLE DROP CONSTRAINT ব্যবহার করে টেবিল থেকে কনস্ট্রেন্ট মুছে ফেলা হয়।

* **SQL Constraints-** SQL Constraints হলো ডাটাবেস টেবিলের ডাটার উপর নির্দিষ্ট করা কিছু নিয়ম। এগুলো টেবিলে কোন ধরনের ডাটা যাবে সেটা নিয়ন্ত্রণ করে। এতে করে ডাটাবেসের ডাটার সঠিকতা এবং নির্ভরযোগ্যতা নিশ্চিত হয়। যদি কোনো ডাটা ইনসার্ট, আপডেট বা ডিলিট করার সময় কোনো Constraint না মানে, তাহলে সেই কাজটি বন্ধ হয়ে যাবে এবং একটি error message দেখাবে।
* Constraints কলাম লেভেল বা টেবিল লেভেল হতে পারে। কলাম লেভেল Constraint একটি নির্দিষ্ট কলামের উপর Apply হয়, এবং টেবিল লেভেল Constraint (পুরো টেবিল) এর উপর Apply হয়। সাধারণত ব্যবহৃত কিছু SQL Constraints হলো:
* **NOT NULL:** এই Constraint নিশ্চিত করে যে কোনো কলামে NULL মান থাকতে পারবে না।
* **UNIQUE:** এই Constraint নিশ্চিত করে যে, কোনো কলামের সব মানই আলাদা আলাদা হবে। অর্থাৎ, একই কলামের কোনো দুটি Row মান রাখতে পারবে না।
* **PRIMARY KEY:** এটি NOT NULL এবং UNIQUE Constraint এর সম্মিলন। এটি টেবিলে প্রতিটি Row (রো) কে Uniqueভাবে Identify করে।
* **FOREIGN KEY:** এটি দুটি বা ততোধিক টেবিলের মধ্যে সম্পর্ক (Relation) স্থাপন করে এবং ডাটা Delete বা Update এর সময় Data Integrity (data integrity) নিশ্চিত করে।
* **CREATE INDEX :** ডাটাবেজের টেবিলগুলোতে ইনডেক্স তৈরি করার জন্য ব্যবহৃত SQL statement। টেবিলের ডাটা অনেক দ্রুত খুঁজে পাওয়া যায় কারণ হলো ইন্ডেক্সগুলি ডাটাবাসকে সঠিক ডাটার অবস্থান নির্দেশ করে, ফলে হার্ডডিস্কে অ unnecessary ভাবে স্ক্যান করতে হয় না।
* **CHECK:** এই Constraint কোনো কলামের মানের উপর নির্দিষ্ট কোনো শর্ত Apply করে। এই শর্ত মেটাতে হবে সেই কলামে মান Insert (insert) (ইনসার্ট) করার জন্য।
* **DEFAULT:** যদি কোনো কলামে কোনো মান Insert (insert) (ইনসার্ট) না করা হয়, তাহলে এই Constraint কলামটির জন্য একটি Default Value (default value) সেট করে দেয়।
* **SQL Date Data Types –**
* **DATE:** এই টাইপটি শুধুমাত্র তারিখের অংশ (yyyy-mm-dd) জন্য ব্যবহৃত হয়, সেকেন্ড বা মিনিটের মতো সময়ের কোনো অংশ থাকে না। সাপোর্টেড রেঞ্জ সাধারণত 1000-01-01 থেকে 9999-12-31 ।
* **DATETIME:** এই টাইপটি তারিখ (yyyy-mm-dd) এবং সময় (hh:mm:ss) উভয়ই ধারণ করে। সাপোর্টেড রেঞ্জ টি ডাটাবেস সিস্টেমের উপর নির্ভর করে তবে সাধারণত 1900-01-01 00:00:00 থেকে 2079-12-31 23:59:59 ।
* **TIMESTAMP:** এই টাইপটিও তারিখ এবং সময় ধারণ করে তবে DATETIME এর চেয়ে কিছুটা ভিন্নভাবে। TIMESTAMP সাধারণত একটি নির্দিষ্ট সময়ের রেফারেন্স (যেমন, ডেটা কবে insert /update করা হয়েছিল) নির্দেশ করে। সাপোর্টেড রেঞ্চ ডাটাবেস সিস্টেমের উপর নির্ভর করে।
* **DATETIME এবং TIMESTAMP** দুটি ধরনের ডেটা/সময় মান ডাটাবেস সিস্টেমে ব্যবহৃত হয়, তবে এই দুটি ধরনের মধ্যে কিছু পার্থক্য রয়েছে:
* **DATETIME**: এটি একটি ফিক্সড সময় প্রকার, অর্থাৎ এটি তথ্যকে সময় প্রদর্শন করে কিন্তু কোনো সময়ে তথ্য পরিবর্তন হয় না। DATETIME ধরনের মান একটি সংখ্যার একটি স্ট্রিং আকার হতে পারে, যেটি তারিখ, সময় বা উভয়ই ধরে নিতে পারে।
* **TIMESTAMP**: TIMESTAMP হল একটি স্পেশাল টাইপের DATETIME যা একটি অটোম্যাটিক আপডেট হওয়ার সুযোগ দেয়। যখন কোনো নতুন ডেটা ইনসার্ট বা আপডেট হয়, সিস্টেম অটোম্যাটিক ভাবে TIMESTAMP কলামের মান আপডেট করে তা দর্শায়। TIMESTAMP মানের গঠন সাধারণত টাইম স্ট্যাম্পের রূপে প্রদর্শিত হয়, এটি যে সময়ে ডেটা ইনসার্ট বা আপডেট হয়েছে সেটি নিখুঁতভাবে প্রদর্শিত করে।
* **সাধারণত, DATETIME** ব্যবহার করা হয় যখন আপডেটের সময় গুরুত্বপূর্ণ না হয়,
* তবে **TIMESTAMP** ব্যবহার করা হয় যখন ডেটা ইনসার্ট বা আপডেটের সময় অত্যন্ত গুরুত্বপূর্ণ হয়। এছাড়াও, TIMESTAMP মানের গঠন সাধারণত সাইজ ছোট হয় এবং সিস্টেমের ডিফল্ট সেটিংস অনুসারে অটোম্যাটিক ভাবে আপডেট হয়।
* **SQL Injection -** SQL Injection হলো একটি ধরনের সাইবার হ্যাকিং প্রযুক্তি যা একটি ওয়েব অ্যাপ্লিকেশনের সাথে সংস্করণের প্রধানত সার্ভারের ডাটাবেস পরিচালনা করার জন্য ব্যবহৃত হলে তা ব্যবহার করা হয় যাতে একটি কোয়েরি ইনজেক্ট করা যায়। সাধারণত, এটি হ্যাকাররা ডাটাবেস থেকে গুরুত্বপূর্ণ তথ্য অর্জন করতে ব্যবহার করে, যেমন ব্যবহারকারীর তথ্য, পাসওয়ার্ড ইত্যাদি।
* **SQL Hosting -** SQL Hosting হলো একটি পরিষেবা যা ডেটাবেস সার্ভার ইন্টারনেটে হোস্ট করার অনুমতি দেয়। এটি ডেটাবেস-চালিত ওয়েবসাইট এবং অ্যাপ্লিকেশনগুলির জন্য অপরিহার্য, কারণ এটি ডেটা সংরক্ষণ, পরিচালনা এবং অ্যাক্সেস করার জন্য একটি নিরাপদ এবং নির্ভরযোগ্য প্ল্যাটফর্ম প্রদান করে।
* **SQL Hosting কিভাবে কাজ করে-** SQL Hosting সাধারণত নিম্নলিখিত পদক্ষেপগুলি অন্তর্ভুক্ত করে:
* **একটি হোস্টিং প্রদানকারীর সাথে একটি অ্যাকাউন্ট তৈরি করা -** একটি হোস্টিং প্রদানকারী বেছে নিতে হবে এবং একটি হোস্টিং প্ল্যান কিনতে হবে যা চাহিদার জন্য উপযুক্ত। বিভিন্ন ধরণের SQL Hosting প্ল্যান উপলব্ধ রয়েছে,
* **একটি ডেটাবেস তৈরি করা:** একবার হোস্টিং অ্যাকাউন্ট সেট আপ হয়ে গেলে, ডেটা সংরক্ষণের জন্য একটি ডেটাবেস তৈরি করতে হবে। হোস্টিং প্রদানকারীর নিয়ন্ত্রণ প্যানেল ব্যবহার করে এটি করা হয়।
* **ডেটা আপলোড করুন:** ডেটাবেস তৈরি হয়ে গেলে, ডেটা আপলোড করা হয়। এটি বিভিন্ন উপায়ে করা যায়, যেমন একটি SQL ডাম্প ফাইল import করা বা ডেটা ম্যানুয়ালি ইনপুট করা।
* **ডেটা অ্যাক্সেস করা:** ডেটা আপলোড হয়ে গেলে, SQL কোয়েরি ব্যবহার করে এটি অ্যাক্সেস করা যায়।হোস্টিং প্রদানকারীর নিয়ন্ত্রণ প্যানেল বা একটি তৃতীয় পক্ষের ডেটাবেস ক্লায়েন্ট ব্যবহার করে এটি করা যায়।
* **SQL Data Types –**   
  MySQL এর জন্য বিভিন্ন ধরনের ডেটা টাইপ রয়েছে যা ডেটাবেস ফিল্ডে ভ্যালু সংরক্ষণে ব্যবহৃত হয়। কিছু গুরুত্বপূর্ণ ডেটা টাইপ হলো:

**INTEGER**: এটি পূর্ণাংক সংখ্যা সংরক্ষণের জন্য ব্যবহৃত হয়।

**VARCHAR**: টেক্সট স্ট্রিংগুলোর জন্য ব্যবহৃত হয় এবং এর দৈর্ঘ্য পরিবর্তনশীল।

**CHAR**: ফিক্সড লেংথের টেক্সট স্ট্রিংগুলোর জন্য ব্যবহৃত হয়।

**DATE**: তারিখ সংরক্ষণের জন্য ব্যবহৃত হয়।

**TIME**: সময় সংরক্ষণের জন্য ব্যবহৃত হয়।

**DATETIME**: তারিখ এবং সময় সংরক্ষণের জন্য ব্যবহৃত হয়।

**FLOAT**: দশমিক সংখ্যা সংরক্ষণের জন্য ব্যবহৃত হয়।

**DOUBLE**: দশমিক সংখ্যা সংরক্ষণের জন্য ব্যবহৃত হয়, কিন্তু FLOAT এর তুলনায় এটি আরও বড় সংখ্যাসমূহ সংরক্ষণের জন্য ব্যবহৃত হয়।

**প্রিসিশন (Precision):** এটি ডেটাটির মোট দশমিক স্থানসহ কয়টি গুলি সংখ্যা রাখতে পারে তা নির্ধারণ করে। উদাহরণস্বরূপ, ডিসি (5,2) ডাটাতে মোট ৫টি সংখ্যা থাকতে পারে, যার মধ্যে দশমিকের পরে থাকবে ২টি সংখ্যা।

**স্কেল (Scale):** এটি নির্ধারণ করে যে দশমিকের পরে কয়টি গুলি সংখ্যা থাকতে পারে। উদাহরণস্বরূপ, (5,2) ডাটাতে দশমিকের পরে সর্বোচ্চ ২টি সংখ্যা থাকতে পারে।

* **DBMS (Database Management System)** হলো এমন একটি সফটওয়্যার যা ডেটাবেস তৈরি, সংগঠিত, পরিচালনা এবং নিয়ন্ত্রণ করে। এটি ব্যবহারকারীদের ডেটা অ্যাক্সেস, সংশোধন এবং ম্যানেজ করার জন্য একটি ইন্টারফেস প্রদান করে।
* **RDBMS (Relational Database Management System)** হলো DBMS-এর একটি নির্দিষ্ট ধরণ যা ডেটা রিলেশনাল মডেলে সংগঠিত করে। রিলেশনাল মডেল টেবিল এবং তাদের মধ্যে সম্পর্ক ব্যবহার করে ডেটা সংগঠিত করে।
* **DBMS এবং RDBMS এর মধ্যে প্রধান পার্থক্যগুলি হল:**

**ডেটা মডেল:**

**ডেটা কাঠামো:** **DBMS** বিভিন্ন ডেটা কাঠামো ব্যবহার করতে পারে, যেমন ফাইল, ইনডেক্স এবং টেবিল।

**RDBMS** টেবিল ব্যবহার করে ডেটা সংগঠিত করে।

**ডেটা অ্যাক্সেস:** **DBMS** বিভিন্ন ডেটা অ্যাক্সেস language ব্যবহার করতে পারে, যেমন SQL, DML এবং DCL।

**RDBMS** SQL ব্যবহার করে ডেটা অ্যাক্সেস করে।

**ব্যবহার:** DBMS বিভিন্ন ধরণের অ্যাপ্লিকেশনের জন্য ব্যবহার করা যেতে পারে।

**RDBMS** ট্রানজেকশনাল অ্যাপ্লিকেশনের জন্য উপযুক্ত যেখানে ডেটা সাবলীলতা এবং সামঞ্জস্যপূর্ণতা গুরুত্বপূর্ণ।

**DBMS ও RDBMS এর মধ্যে পার্থক্যসমুহঃ**

|  |  |
| --- | --- |
| **DBMS** | **RDBMS** |
| DBMS এর পূর্ণরুপ Database Management System. | RDBMS এর পূর্ণরুপ Relational Database Management System. |
| DBMS ডেটা ফাইল হিসেবে সংরক্ষণ করে। | RDBMS ডেটা টেবুলার ফর্মে সংরক্ষণ করে। |
| ডেটা এলিমেন্ট এককভাবে অ্যাকসেস করতে হয়। | একাধিক ডেটা এলিমেন্ট একসাথে একই সময় অ্যাকসেস করা যায়। |
| ডেটা সমূহের মধ্যে কোন রিলেশন থাকে না। | ডেটা সমূহ টেবিলে সংরক্ষণ হয় এবং টেবিলগুলো একে অপরের সাথে সম্পর্কিত। |
| DBMS ডিস্ট্রিবিউটেড ডেটাবেজ সাপোর্ট করে না। | RDBMS ডিস্ট্রিবিউটেড ডেটাবেজ সাপোর্ট করে। |
| **DBMS** ডেটা রিডানডেন্সি থাকে। | ডেটা রিডানডেন্সি থাকে না। |
| অল্প সংখ্যক ডেটা নিয়ে কাজ করার জন্য বিভিন্ন ছোট অরগানাইজেশনে ব্যবহৃত হয়। | অধিক সংখ্যক ডেটা নিয়ে কাজ করার জন্য বিভিন্ন বড় অরগানাইজেশনে ব্যবহৃত হয়। |
| DBMS সিঙ্গেল ইউজার সাপোর্ট করে। | RDBMS একাধিক ইউজার সাপোর্ট করে। |
| DBMS এ ডেটার লো-লেভেল সিকিউরিটি থাকে। | RDBMS এ ডেটার মাল্টি-লেভেল সিকিউরিটি থাকে। |
| উদাহরণ- XML, Microsoft Access ইত্যাদি। | উদাহরণ- MySQL, PostgreSQL, SQL Server, Oracle, ইত্যাদি। |
| **DBMS** বিভিন্ন ডেটা মডেল ব্যবহার করতে পারে, যেমন হায়ারারকিক্যাল, নেটওয়ার্ক এবং রিলেশনাল। | **RDBMS** রিলেশনাল মডেল ব্যবহার করে। |

* **DDL ( Data Definition Language)** হল এসকিউএল (স্ট্রাকচার্ড কোয়েরি ল্যাঙ্গুয়েজ) এর একটি উপসেট যা একটি ডাটাবেসের গঠন এবং এর বস্তু, যেমন টেবিল, ভিউ, ইনডেক্স এবং পদ্ধতিগুলিকে সংজ্ঞায়িত করতে ব্যবহৃত হয়।
* **DML ( Data Manipulation Language)** হল SQL (স্ট্রাকচার্ড কোয়েরি ল্যাঙ্গুয়েজ) এর একটি উপসেট যা একটি ডাটাবেসের মধ্যে ডেটা ম্যানিপুলেট করতে ব্যবহৃত হয়। DML statement একটি ডাটাবেসে ডেটা সন্নিবেশ, আপডেট এবং মুছে ফেলার জন্য ব্যবহৃত হয়।
* **DDL প্রয়োজনীয় হওয়ার বিভিন্ন কারণ রয়েছে:**
* **DDL এবং DML এর মধ্যে পার্থক্য কি?**
* **DDL ( Data Definition Language)** ডাটাবেস স্কিমা সংজ্ঞায়িত করার জন্য SQL কমান্ডের একটি সেট। এটি কেবল ডাটাবেস স্কিমার বর্ণনা নিয়ে কাজ করে এবং ডাটাবেস অবজেক্টের গঠন তৈরি ও পরিবর্তন করতে ব্যবহৃত হয়।
* **DDL স্টেটমেন্টের** উদাহরণগুলির মধ্যে রয়েছে CREATE, ALTER এবং DROP ।
* **DML ( Data Manipulation Language)** হল SQL কমান্ডের একটি সেট যা DDL দ্বারা তৈরি স্কিমার মধ্যে ডেটা ম্যানিপুলেট করতে ব্যবহৃত হয়। এটি প্রকৃত ডেটা নিয়ে কাজ করে এবং ডাটাবেস থেকে ডেটা সন্নিবেশ, আপডেট এবং পুনরুদ্ধার করতে ব্যবহৃত হয়।
* **DML স্টেটমেন্টের উদাহরণগুলির** মধ্যে রয়েছে SELECT, INSERT, UPDATE এবং DELETE ।