



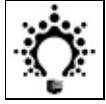
विमलम

विमलम

दा

সূচিপত্র

কোর্স পরিচিতি	0
উপক্রমণিকা	1
ইনস্টলেশন	2
ডাটাবেজ	3
ডাটা টাইপ	4
ডাটাবেস ডিজাইন	5
রিজার্ডড কিওয়ার্ড	5.1
টেবিল অপারেশন	6
এসকিউএল কমান্ড'স	7
স্টিং অপারেশন	7.1
গাণিতিক অপারেশন	7.2
লজিক্যাল অপারেশন	7.3
JOIN	7.4
সময় ও তারিখ এর ব্যবহার	7.5
অপারেটর'স	7.6
ডাটাবেস অপ্টিমাইজেশান	8
স্টেটমেন্ট অপ্টিমাইজেশান	8.1
কুয়েরী অপ্টিমাইজেশান	8.2
ইনডেক্স অপ্টিমাইজেশান	8.3
ডাটাবেস স্ট্রাকচার অপ্টিমাইজেশান	8.4
বাফারিং এবং ক্যাশিং	8.5
MySQL সার্ভার অপ্টিমাইজেশান	8.6
Benchmarking	8.7



howtocode.com.bd

পৃষ্ঠা পছন্দ করুন 9.9হাজার পছন্দগুলি

আপনার বন্ধুদের মধ্যে আপনিই এটা প্রথম পছন্দ করুন



কোর্স এর মূল পাতা | HowToCode মূল সাইট | সবার জন্য প্রোগ্রামিং ব্লগ | পিডিএফ ডাউনলোড

SQL - রিলেশনাল ডাটাবেজের ভাষা

gitter join chat



saiful 129



thesabbir 11



niloyniloy 6



nuhil 6

সংক্ষেপ

ডাটাবেস ম্যানেজমেন্ট সিস্টেম (DBMS) হল সফটওয়্যার নিয়ন্ত্রিত একটি ব্যবস্থা যার মাধ্যমে ডাটাবেস পরিচালনা, তথ্যের স্থান সংকুলান, নিরাপত্তা, ব্যাকআপ, তথ্য সংগ্রহের অনুমতি ইত্যাদি নির্ধারণ করা হয়। আমাদের নিত্য দিনের ব্যবহৃত সকল সফটওয়্যার, ওয়েব সাইট, ওয়েব এ্যাপ, মোবাইল এ্যাপের সকল তথ্য সংরক্ষণের জন্য ব্যবহার করা হয় ডাটাবেস। কিছু জনপ্রিয় ডাটাবেস ম্যানেজমেন্ট সিস্টেম হল ওরাকল, এসকিউএল, এসকিউএল-লাইট, মাইএসকিউএল, পোস্টজিআরই-এসকিউএল, মাইক্রোসফট এসকিউএল সার্ভার, আইবিএম ডিবি২, মাইক্রোসফট এক্সেস।

এই বইতে মূলত জনপ্রিয় ওপেন সোর্স ডাটাবেজ ম্যানেজমেন্ট সিস্টেম MySQL (মাইএসকিউএল/মাইসিকুয়েল) নিয়ে আলোকপাত করা হয়েছে।

ওপেন সোর্স

এই বইটি মূলত স্বেচ্ছাশ্রমে লেখা এবং বইটি সম্পূর্ণ ওপেন সোর্স। এখানে তাই আপনিও অবদান রাখতে পারেন লেখক হিসেবে। আপনার কন্ট্রিবিউশান গৃহীত হলে অবদানকারীদের তালিকায় আপনার নাম যোগ করে দেওয়া হবে।

এটি মূলত একটি গিটহাব রিপোজিটরি যেখানে এই বইয়ের আর্টিকেল গুলো মার্কডাউন ফরম্যাটে লেখা হচ্ছে। রিপোজিটরিটি ফর্ক করে পুল রিকুয়েস্ট পাঠানোর মাধ্যমে আপনারাও অবদান রাখতে পারেন।



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#).

সংক্ষিপ্ত ইতিহাস

১৯৭০ দশকের প্রথম দিকে [Donald D. Chamberlin](#) এবং [Raymond F. Boyce](#) আইবিএমের "System R" নামক রিলেশনাল ডাটাবেজ এ ব্যবহারের জন্য একটি লাংগুয়েজ তৈরি শুরু করেন। তারা লাংগুয়েজটির প্রথম ভার্সনটির নাম দেন SEQUEL (Structured English Query Language) যা পরবর্তীতে ট্রেডমার্ক জনিত সমস্যার কারণে পরিবর্তিত হয়ে SQL (Structured Query Language) হয়।

এসকিউএল হচ্ছে [Edgar F. Codd](#) এর রিলেশনাল মডেল এর উপর ভিত্তি করে বানানো প্রথম লাংগুয়েজ। যদিও এটি তখন তার রিলেশনাল মডেলের সাথে পুরোপুরি সামঞ্জস্য পূর্ণ ছিল না। পরবর্তীতে এটিই বিশ্বের সবচেয়ে বহুল ব্যবহৃত ডাটাবেজ লাংগুয়েজে পরিণত হয়।

এসকিউএল কেন শিখবেন ?

- আপনি সহজে অর্থ উপার্জন করতে পারবেন। (যদিও SQL শেখার জন্য এটা প্রধান কারণ না)।
- ডাটাবেস ম্যানেজমেন্ট সম্পর্কিত চাকুরী করতে চাইলে SQL জানা লাগবে।
- কোন সমস্যায় পড়লে সবচাইতে কম সময়ে এবং সহজে সমাধান পাবেন।
- কুয়েরি কিভাবে লিখতে হয় সেটা জানতে পারবেন, পরে অন্য কোন DBMS সফটওয়্যার এর কাজ শেখা সহজ হবে।
- SQL শিখলে MySQL, SQLite, PostgreSQL ইত্যাদি সহজে ব্যবহার করতে পারবেন।
- ডাটা মাইনিং এর কাজ সহজে করতে পারবেন।
- ডাটা ম্যানিপুলেশন এর কাজ সহজে করতে পারবেন।
- বড় ডাটা সোর্স নিয়ে কাজ করতে পারবেন।
- ওয়েব সাইট / ওয়েব এ্যাপ নিয়ে কাজ করতে চাইলে SQL ভিত্তিক ডাটাবেস ব্যবহার সুবিধাজনক এবং সহজলভ্য।
- পিএইচপি, পাইথন, C#, রুবি, জি, সি++, ইত্যাদির সাথে SQL ব্যবহার করতে পারবেন।

NoSQL এর যুগে SQL ?

NoSQL এর সময়ে SQL শেখার অনেক কারণ আছে। NoSQL এখনো স্ট্যান্ডার্ড না, তাছাড়া এটা এখন পর্যন্ত বড় ধরনের পরীক্ষার মধ্যদিয়ে যায় নি। আগামী ১০-১২ বছরে হয়তো এটা ভালো পর্যায়ে যেতে পারবে। তাছাড়া এটি SQL থেকে পুরানো হলেও এটি সকল কাজে ব্যবহার করার জন্য উপযুক্ত না। এটি রিলেশনাল ডাটা মডেল নিয়ে কাজ করতে পারে না। NoSQL শেখার ইচ্ছে থাকলে শেখা উচিত।

এসকিউএল স্ট্যান্ডার্ড

এসকিউল মূলত একটি স্ট্যান্ডারাইজড লাংগুয়েজ যার নির্দিষ্ট স্পেসিফিকেশন রয়েছে। এর প্রথম স্ট্যান্ডার্ডটি ANSI ১৯৮৬ সালে প্রকাশ হয়। এর সর্বশেষ সংস্করণ SQL:2009। বিভিন্ন ডাটাবেজ সিস্টেমস এ এসকিউল ইমপ্লিমেন্টেশন সমূহ শতভাগ স্ট্যান্ডার্ড মেনে চলে না। তাদের কিছু অনন্য সুবিধাও এসকিউলে সংযুক্তি করেছে।

লিনাক্স/*nix

টার্মিনালে নিচের কমান্ড দিয়ে রিপো আপডেট করুন।

```
sudo apt-get update
```

টার্মিনালে নিচের কমান্ড দিয়ে MySQL সার্ভার ইন্সটল করুন।

```
sudo apt-get install mysql-server
```

টার্মিনালে নিচের কমান্ড দিয়ে MySQL কনফিগার করুন।

```
sudo mysql_secure_installation
```

টার্মিনালে `mysql --version` কমান্ড দিয়ে দেখুন MySQL এর কোন ভার্সন ইন্সটল হয়েছে।

ম্যাক

টার্মিনালে নিচের কমান্ড দিয়ে MySQL সার্ভার ইন্সটল করুন।

```
sudo port install mysql5-server && sudo -u _mysql /opt/local/bin/mysql_install_db5
```

MySQL সার্ভার চালু করতে টার্মিনালে নিচের কমান্ড দিন।

```
sudo port load mysql5-server
```

MySQL সার্ভার বন্ধ করতে টার্মিনালে নিচের কমান্ড দিন।

```
sudo port unload mysql5-server
```

উইন্ডোজ

MySQL এর [অফিশিয়াল সাইট](#) থেকে ইন্সটলার নামিয়ে ইন্সটল করুন।

phpmyadmin

গ্রাফিক্যাল ইন্টারফেস দিয়ে MySQL ব্যবহার করতে চাইলে phpmyadmin ইন্সটল করতে পারেন। তবে এর জন্য পিএইচপি এবং Apache/Nginx/Web সার্ভার ইন্সটল করা থাকতে হবে। phpmyadmin ইন্সটল করতে টার্মিনালে নিচের কমান্ড দিন।

```
sudo apt-get install phpmyadmin
```

উইন্ডোজ এর জন্য [WAMP/XAMPP](#) ব্যবহার করুন।

ডাটাবেজ

ডাটাবেজ তৈরি

টার্মিনালে/phpmyadmin এর SQL query তে নিচের মত করে কমান্ড লিখুন।

```
CREATE DATABASE your_database_name;
```

ডাটাবেজ মোছা

টার্মিনালে/phpmyadmin এর SQL query তে নিচের মত করে কমান্ড লিখুন।

```
DROP DATABASE your_database_name;
```

ডাটা টাইপ

যে কোন ডাটার একটি নির্দিষ্ট ধরন থাকে, যা প্রোগ্রামিং ল্যাঙ্গুয়েজকে বলে করে দেয় এটি কি ধরনের ডাটা জমা রাখতে পারবে অথবা এই ডাটার ধরন কি, ডাটার এই ধরনকে ডাটা টাইপ বলা হয়ে থাকে। MySQL এ বেশ কয়েক প্রকার ডাটা টাইপ আছে, নিচে এদের বর্ণনা দেওয়া হল।

সংখ্যা

সংখ্যা সম্পর্কিত তথ্য সংরক্ষণের জন্য নিচের ডাটা টাইপ সমূহ ব্যবহার করা হয়ে থাকে।

ডাটা টাইপ	ডাটা সাইজ	সর্বনিম্ন মান	সর্বোচ্চ মান	রেঞ্জ
TINYINT	1 বাইট	-128	127	2^8
SMALLINT	2 বাইট'স	-32,768	32,767	2^{16}
MEDIUMINT	3 বাইট'স	-8,388,608	8,388,607	2^{24}
INT	4 বাইট'স	-2,147,483,648	2,147,483,647	2^{32}
BIGINT	8 বাইট'স	-9,223,372,036,854,775,808	9,223,372,036,854,775,807	2^{64}

অক্ষর

অক্ষর সম্পর্কিত তথ্য সংরক্ষণের জন্য নিচের ডাটা টাইপ সমূহ ব্যবহার করা হয়ে থাকে।

ডাটা টাইপ (টেক্সট)	ডাটা টাইপ (ব্লব)	ডাটা সাইজ	অক্ষর সংখ্যা (বাইটে)
TINYTEXT	TINYBLOB	L + 1 বাইট'স	255
TEXT	BLOB	L + 2 বাইট'স	65,535
MEDIUMTEXT	MEDIUMBLOB	L + 3 বাইট'স	16,777,215
LONGTEXT	LOB	L + 4 বাইট'স	4,294,967,295
VARCHAR	-	L + 2 বাইট'স	0-65535

উল্লেখ্য L = সংরক্ষিত মান এর দৈর্ঘ্য / Length

ব্লব / BLOB = Binary Large Object

সময় ও তারিখ

সময় ও তারিখ সম্পর্কিত তথ্য সংরক্ষণের জন্য নিচের ডাটা টাইপ সমূহ ব্যবহার করা হয়ে থাকে।

ডাটা টাইপ	ফরম্যাট	ডাটা সাইজ
DATE	YYYY-MM-DD	3 বাইট'স
TIME	HHH:MM:SS	3 বাইট'স
YEAR	YYYY	1 বাইট
DATETIME	YYYY-MM-DD HH:MM:SS	8 বাইট'স
TIMESTAMP	YYYY-MM-DDHH:MM:SS	8 বাইট'স

অন্যান্য

ENUM, SET, BOOLEAN

ডাটাবেস ডিজাইন

“ডাটাবেস ডিজাইন” হচ্ছে এমন এক প্রক্রিয়া যেখানে একটি ডাটাবেসের ডাটার ধরন, ডাটার মধ্যবর্তী সম্পর্ক ইত্যাদি দিয়ে ডাটার মডেল প্রস্তুত করা হয়। এই ডাটার মডেল সাধারণত “ডাটা ডেফিনেশন ল্যাঙ্গুয়েজ (DDL)” এ কোড জেনারেট করে, যা দিয়ে পরবর্তীতে ডাটাবেস তৈরি করা হয়।

“ডাটাবেস ডিজাইন” সফল ভাবে সম্পন্ন করার জন্য কয়েকটি ধাপ অনুসরণ করা হয়ে থাকে। নিচে ধাপসমূহ উল্লেখ করা হল।

- ডাটাবেসে কোন ধরনের ডাটা সংরক্ষণ করা হবে তা নির্ধারণ করা।
- কোন ডাটার সাথে কোন ডাটার সম্পর্ক / রিলেশন থাকবে তা নির্ধারণ করা।
- যেসব রিলেশন লেখা হয়েছে তা যুক্তিযুক্ত কিনা তা নির্ধারণ করা।

নেমিং কনভেনশন

- প্রাইমারি কি হিসাবে `id` সবসময় ব্যবহার না করা। কারণ জয়েন কুয়েরিতে সকল টেবিলে যদি `id` থাকে তবে এই সকল `id` এর জন্য এলিয়াস(alias) লিখতে হবে।
- টেবিলের নাম / কলামের নামে **রিজার্ড কি-ওয়ার্ড** ব্যবহার করা যাবে না। সবচাইতে ভালো পদ্ধতি হচ্ছে টেবিলের নামে প্রিফিক্স ব্যবহার করা। প্রিফিক্স এর উদাহরণঃ `mydb_tableName`।
- হাইফেন (-), কোট (‘), স্পেস এগুলো ব্যবহার করা যাবে না।
- টেবিলের / কলামের নাম সিঙ্গুলার দেওয়া উত্তম, যদিও প্লুরাল দেখতে অথবা শুনতে যুক্তিযুক্ত মনে হতে পারে। তবে প্লুরাল ব্যবহার করলে কোন ক্ষতি নেই।

ডাটা সংরক্ষণের ধরন নির্ধারণ করা


এই ধাপে সাধারণত লক্ষ রাখা হয় যে কাজের জন্য ডাটাবেস ডিজাইন করা হচ্ছে সেখানে কি ধরনের ডাটা সংরক্ষণ প্রয়োজন। এখানে “চাহিদা বিশ্লেষণ” করা হয়ে থাকে। এই কাজের জন্য এপ্লিকেশন এর গঠন, ডাটা ম্যানিপুলেশন, স্কেল এভিলিটি, সিস্টেম স্পেসিফিকেশন ইত্যাদি ব্যাপারে প্রয়োজনীয় জ্ঞান দরকার। আপনাকে জানতে হবে কোন কোন ডাটা আপনি সার্চ করবেন, কোন কোন ডাটা স্ট করা হবে, কিভাবে ডাটা রাখলে সবচাইতে কম কুয়েরি করে প্রয়োজনীয় ডাটা দেখানো যাবে, কোন ফরম্যাটে ডাটা রাখলে ডাটার সাইজ কম হবে এবং ম্যানিপুলেশন সহজ হবে।


উদাহরণঃ মনে করুন যে এপ্লিকেশন এর জন্য ডাটাবেস ডিজাইন করতে হবে সেটা একটা ব্লগ। প্রথমেই চিন্তা করুন আপনার ব্লগে কি কি ডাটা থাকতে পারে, কি কি টেবিল লাগতে পারে। সাধারণত ব্লগে USER একাউন্ট থাকে, পোস্ট থাকে, কमेंট থাকে। প্রথমে যদি আমরা User Table নিয়ে কাজ করতে যাই তাহলে আমাদের যা যা লাগবেঃ


- `id` / `user_id` যা প্রাইমারি কি, অটো ইনক্রিমেন্ট এর ডাটা টাইপ আনসাইন্ড ইন্টিজার। কারণ `id` কখনো ঋণাত্মক

হবে না।

- `username` এর ডাটা টাইপ `VARCHAR` এবং সাইজ ২০ এবং এটি ইউনিক। কারণ আমরা একই `username` বার বার দেখতে চাইনা।
- `password` এর ডাটা টাইপ `VARCHAR` এবং সাইজ ৩২। কারণ আমরা পাসওয়ার্ড এর হ্যাশ সংরক্ষণ করবো, আমাদের হ্যাশ হবে MD5 যার সাইজ হচ্ছে ৩২ ক্যারেক্টার।

abc_user		
	<code>user_id</code>	<code>integer</code>
	<code>username</code>	<code>varchar(20)</code>
	<code>password</code>	<code>varchar(32)</code>
	<code>email</code>	<code>varchar(100)</code>
Add field		

abc_post		
	<code>post_id</code>	<code>integer</code>
	<code>title</code>	<code>varchar(255)</code>
	<code>details</code>	<code>text</code>
	<code>user_id</code>	<code>integer</code>
Add field		

abc_comment		
	<code>comment_id</code>	<code>integer</code>
	<code>user_id</code>	<code>integer</code>
	<code>post_id</code>	<code>integer</code>
	<code>details</code>	<code>varchar(255)</code>
Add field		

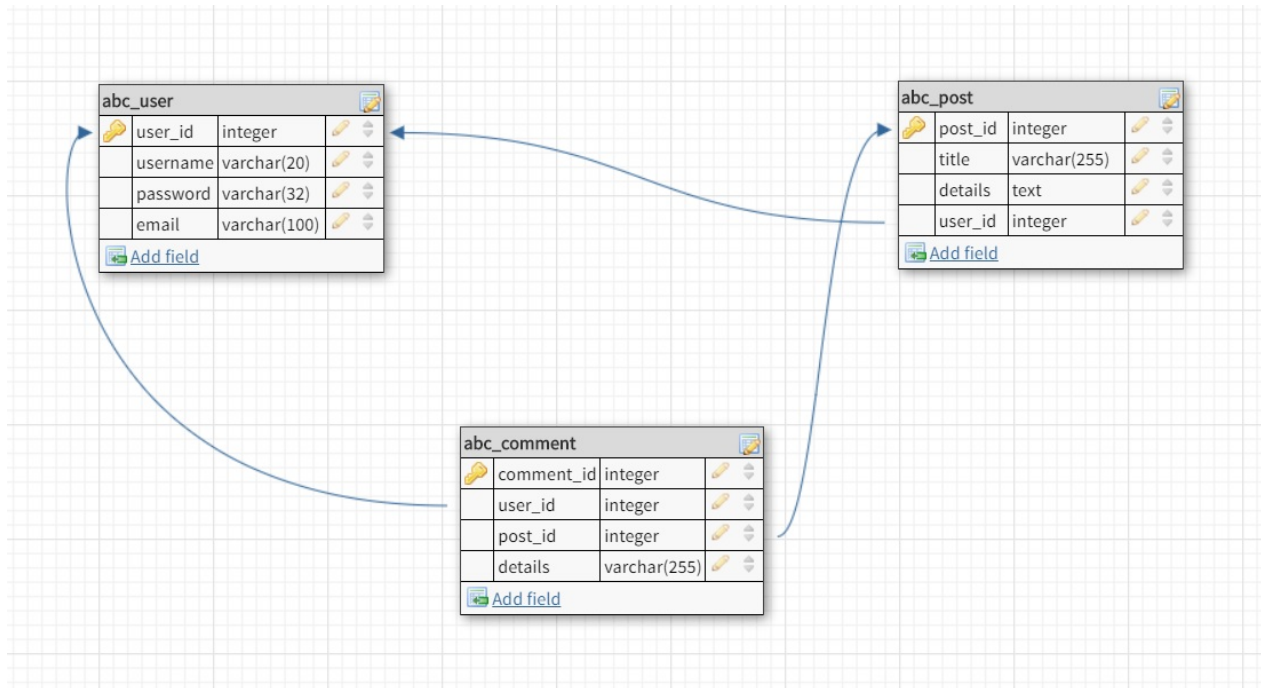
চিত্রঃডাটাবেস ডিজাইন (রিলেশন ছাড়া)

ডাটার মধ্যবর্তী সম্পর্ক নির্ধারণ করা

প্রাথমিক ডাটাবেস ডিজাইন শেষ হলে রিলেশন নির্ধারণ শুরু করতে হয়। কোন ডাটা আপডেট করলে কোন ডাটাতে ইনক্রিমেন্ট হবে, কোন ডাটা ডিলিট করলে কোন কোন ডাটা অটোমেটিক ডিলিট হয়ে যাবে এই সব ঠিক করে দিতে হবে। কোন টেবিলের কোন কলাম অন্য টেবিলের কোন কলামের সাথে সম্পর্কযুক্ত তা নির্ধারণ করে দিতে হবে।

উদাহরণ: `abc_post` টেবিলের `user_id` `abc_user` টেবিলের `user_id` এর সাথে সম্পর্কিত। একই ভাবে `abc_comment` টেবিলের `user_id` `abc_user` টেবিলের `user_id` এর সাথে সম্পর্কিত এবং `abc_comment` টেবিলের `post_id` `abc_post` এর `post_id` এর সাথে সম্পর্কিত।

>



চিত্রঃডাটাবেস ডিজাইন (রিলেশন সহ)

```
DROP TABLE IF EXISTS `abc_user`;
CREATE TABLE IF NOT EXISTS `abc_user` (
  `user_id` int(11) NOT NULL,
  `username` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `email` (`email`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `abc_post`;
CREATE TABLE IF NOT EXISTS `abc_post` (
  `post_id` int(11) NOT NULL,
  `title` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `details` mediumtext COLLATE utf8_unicode_ci NOT NULL,
  `user_id` int(11) NOT NULL,
  PRIMARY KEY (`post_id`),
  KEY `abc_post_fk0` (`user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `abc_comment`;
CREATE TABLE IF NOT EXISTS `abc_comment` (
  `comment_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL,
  `post_id` int(11) NOT NULL,
  `details` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`comment_id`),
  KEY `abc_comment_fk0` (`user_id`),
  KEY `abc_comment_fk1` (`post_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

ভিজুয়ালই স্কিমা ডিজাইন করতে [এই সাইট](#) ব্যবহার করতে পারেন।

রিজার্ডড কি-ওয়ার্ড এর তালিকা

MySQL এর সকল রিজার্ডড কি-ওয়ার্ড / ওয়ার্ড এর তালিকা নিচে দেওয়া হল।

কি-ওয়ার্ড	কি-ওয়ার্ড
ACCESSIBLE (R)	MAX_ROWS
ACTION	MAX_SIZE
ADD (R)	MAX_UPDATES_PER_HOUR
AFTER	MAX_USER_CONNECTIONS
AGAINST	MEDIUM
AGGREGATE	MEDIUMBLOB (R)
ALGORITHM	MEDIUMINT (R)
ALL (R)	MEDIUMTEXT (R)
ALTER (R)	MEMORY
ANALYZE (R)	MERGE
AND (R)	MESSAGE_TEXT
ANY	MICROSECOND
AS (R)	MIDDLEINT (R)
ASC (R)	MIGRATE
ASCII	MINUTE
ASENSITIVE (R)	MINUTE_MICROSECOND (R)
AT	MINUTE_SECOND (R)
AUTHORS	MIN_ROWS
AUTOEXTEND_SIZE	MOD (R)
AUTO_INCREMENT	MODE
AVG	MODIFIES (R)
AVG_ROW_LENGTH	MODIFY
BACKUP	MONTH
BEFORE (R)	MULTILINESTRING
BEGIN	MULTIPOINT
BETWEEN (R)	MULTIPOLYGON

BIGINT (R)	MUTEX
BINARY (R)	MYSQL_ERRNO
BINLOG	NAME
BIT	NAMES
BLOB (R)	NATIONAL
BLOCK	NATURAL (R)
BOOL	NCHAR
BOOLEAN	NDB
BOTH (R)	NDBCLUSTER
BTREE	NEW
BY (R)	NEXT
BYTE	NO
CACHE	NODEGROUP
CALL (R)	NONE
CASCADE (R)	NOT (R)
CASCADED	NO_WAIT
CASE (R)	NO_WRITE_TO_BINLOG (R)
CATALOG_NAME	NULL (R)
CHAIN	NUMERIC (R)
CHANGE (R)	NVARCHAR
CHANGED	OFFSET
CHAR (R)	OLD_PASSWORD
CHARACTER (R)	ON (R)
CHARSET	ONE
CHECK (R)	ONE_SHOT
CHECKSUM	OPEN
CIPHER	OPTIMIZE (R)
CLASS_ORIGIN	OPTION (R)
CLIENT	OPTIONALLY (R)
CLOSE	OPTIONS
COALESCE	OR (R)
CODE	ORDER (R)

COLLATE (R)	OUT (R)
COLLATION	OUTER (R)
COLUMN (R)	OUTFILE (R)
COLUMNS	OWNER
COLUMN_NAME	PACK_KEYS
COMMENT	PAGE
COMMIT	PARSER
COMMITTED	PARTIAL
COMPACT	PARTITION
COMPLETION	PARTITIONING
COMPRESSED	PARTITIONS
CONCURRENT	PASSWORD
CONDITION (R)	PHASE
CONNECTION	PLUGIN
CONSISTENT	PLUGINS
CONSTRAINT (R)	POINT
CONSTRAINT_CATALOG	POLYGON
CONSTRAINT_NAME	PORT
CONSTRAINT_SCHEMA	PRECISION (R)
CONTAINS	PREPARE
CONTEXT	PRESERVE
CONTINUE (R)	PREV
CONTRIBUTORS	PRIMARY (R)
CONVERT (R)	PRIVILEGES
CPU	PROCEDURE (R)
CREATE (R)	PROCESSLIST
CROSS (R)	PROFILE
CUBE	PROFILES
CURRENT_DATE (R)	PROXY[h]
CURRENT_TIME (R)	PURGE (R)
CURRENT_TIMESTAMP (R)	QUARTER

CURRENT_USER (R)	QUERY
CURSOR (R)	QUICK
CURSOR_NAME	RANGE (R)
DATA	READ (R)
DATABASE (R)	READS (R)
DATABASES (R)	READ_ONLY
DATAFILE	READ_WRITE (R)
DATE	REAL (R)
DATETIME	REBUILD
DAY	RECOVER
DAY_HOUR (R)	REDOFILE
DAY_MICROSECOND (R)	REDO_BUFFER_SIZE
DAY_MINUTE (R)	REDUNDANT
DAY_SECOND (R)	REFERENCES (R)
DEALLOCATE	REGEXP (R)
DEC (R)	RELAY[i]
DECIMAL (R)	RELAYLOG
DECLARE (R)	RELAY_LOG_FILE
DEFAULT (R)	RELAY_LOG_POS
DEFINER	RELAY_THREAD
DELAYED (R)	RELEASE (R)
DELAY_KEY_WRITE	RELOAD
DELETE (R)	REMOVE
DESC (R)	RENAME (R)
DESCRIBE (R)	REORGANIZE
DES_KEY_FILE	REPAIR
DETERMINISTIC (R)	REPEAT (R)
DIRECTORY	REPEATABLE
DISABLE	REPLACE (R)
DISCARD	REPLICATION
DISK	REQUIRE (R)
DISTINCT (R)	RESET

DISTINCTROW (R)	RESIGNAL (R)
DIV (R)	RESTORE
DO	RESTRICT (R)
DOUBLE (R)	RESUME
DROP (R)	RETURN (R)
DUAL (R)	RETURNS
DUMPFIL	REVOKE (R)
DUPLICATE	RIGHT (R)
DYNAMIC	RLIKE (R)
EACH (R)	ROLLBACK
ELSE (R)	ROLLUP
ELSEIF (R)	ROUTINE
ENABLE	ROW
ENCLOSED (R)	ROWS
END	ROW_FORMAT
ENDS	RTREE
ENGINE	SAVEPOINT
ENGINES	SCHEDULE
ENUM	SCHEMA (R)
ERROR[a]	SCHEMAS (R)
ERRORS	SCHEMA_NAME
ESCAPE	SECOND
ESCAPED (R)	SECOND_MICROSECOND (R)
EVENT	SECURITY
EVENTS	SELECT (R)
EVERY	SENSITIVE (R)
EXECUTE	SEPARATOR (R)
EXISTS (R)	SERIAL
EXIT (R)	SERIALIZABLE
EXPANSION	SERVER
EXPLAIN (R)	SESSION

EXTENDED	SET (R)
EXTENT_SIZE	SHARE
FALSE (R)	SHOW (R)
FAST	SHUTDOWN
FAULTS	SIGNAL (R)
FETCH (R)	SIGNED
FIELDS	SIMPLE
FILE	SLAVE
FIRST	SLOW[j]
FIXED	SMALLINT (R)
FLOAT (R)	SNAPSHOT
FLOAT4 (R)	SOCKET
FLOAT8 (R)	SOME
FLUSH	SONAME
FOR (R)	SOUNDS
FORCE (R)	SOURCE
FOREIGN (R)	SPATIAL (R)
FOUND	SPECIFIC (R)
FRAC_SECOND[b]	SQL (R)
FROM (R)	SQLException (R)
FULL	SQLSTATE (R)
FULLTEXT (R)	SQLWARNING (R)
FUNCTION	SQL_BIG_RESULT (R)
GENERAL[c]	SQL_BUFFER_RESULT
GEOMETRY	SQL_CACHE
GEOMETRYCOLLECTION	SQL_CALC_FOUND_ROWS (R)
GET_FORMAT	SQL_NO_CACHE
GLOBAL	SQL_SMALL_RESULT (R)
GRANT (R)	SQL_THREAD
GRANTS	SQL_TSI_DAY
GROUP (R)	SQL_TSI_FRAC_SECOND[k]
HANDLER	SQL_TSI_HOUR

HASH	SQL_TSI_MINUTE
HAVING (R)	SQL_TSI_MONTH
HELP	SQL_TSI_QUARTER
HIGH_PRIORITY (R)	SQL_TSI_SECOND
HOST	SQL_TSI_WEEK
HOSTS	SQL_TSI_YEAR
HOURL	SSL (R)
HOURL_MICROSECOND (R)	START
HOURL_MINUTE (R)	STARTING (R)
HOURL_SECOND (R)	STARTS
IDENTIFIED	STATUS
IF (R)	STOP
IGNORE (R)	STORAGE
IGNORE_SERVER_IDS[d]	STRAIGHT_JOIN (R)
IMPORT	STRING
IN (R)	SUBCLASS_ORIGIN
INDEX (R)	SUBJECT
INDEXES	SUBPARTITION
INFILE (R)	SUBPARTITIONS
INITIAL_SIZE	SUPER
INNER (R)	SUSPEND
INNOBASE[e]	SWAPS
INNODB[f]	SWITCHES
INOUT (R)	TABLE (R)
INSENSITIVE (R)	TABLES
INSERT (R)	TABLESPACE
INSERT_METHOD	TABLE_CHECKSUM
INSTALL	TABLE_NAME
INT (R)	TEMPORARY
INT1 (R)	TEMPTABLE
INT2 (R)	TERMINATED (R)

INT3 (R)	TEXT
INT4 (R)	THAN
INT8 (R)	THEN (R)
INTEGER (R)	TIME
INTERVAL (R)	TIMESTAMP
INTO (R)	TIMESTAMPADD
INVOKER	TIMESTAMPDIFF
IO	TINYBLOB (R)
IO_THREAD	TINYINT (R)
IPC	TINYTEXT (R)
IS (R)	TO (R)
ISOLATION	TRAILING (R)
ISSUER	TRANSACTION
ITERATE (R)	TRIGGER (R)
JOIN (R)	TRIGGERS
KEY (R)	TRUE (R)
KEYS (R)	TRUNCATE
KEY_BLOCK_SIZE	TYPE
KILL (R)	TYPES
LANGUAGE	UNCOMMITTED
LAST	UNDEFINED
LEADING (R)	UNDO (R)
LEAVE (R)	UNDOFILE
LEAVES	UNDO_BUFFER_SIZE
LEFT (R)	UNICODE
LESS	UNINSTALL
LEVEL	UNION (R)
LIKE (R)	UNIQUE (R)
LIMIT (R)	UNKNOWN
LINEAR (R)	UNLOCK (R)
LINES (R)	UNSIGNED (R)
LINESTRING	UNTIL

LIST	UPDATE (R)
LOAD (R)	UPGRADE
LOCAL	USAGE (R)
LOCALTIME (R)	USE (R)
LOCALTIMESTAMP (R)	USER
LOCK (R)	USER_RESOURCES
LOCKS	USE_FRM
LOGFILE	USING (R)
LOGS	UTC_DATE (R)
LONG (R)	UTC_TIME (R)
LOB (R)	UTC_TIMESTAMP (R)
LONGTEXT (R)	VALUE
LOOP (R)	VALUES (R)
LOW_PRIORITY (R)	VARBINARY (R)
MASTER	VARCHAR (R)
MASTER_CONNECT_RETRY	VARCHARACTER (R)
MASTER_HEARTBEAT_PERIOD[g]	VARIABLES
MASTER_HOST	VARYING (R)
MASTER_LOG_FILE	VIEW
MASTER_LOG_POS	WAIT
MASTER_PASSWORD	WARNINGS
MASTER_PORT	WEEK
MASTER_SERVER_ID	WHEN (R)
MASTER_SSL	WHERE (R)
MASTER_SSL_CA	WHILE (R)
MASTER_SSL_CAPATH	WITH (R)
MASTER_SSL_CERT	WORK
MASTER_SSL_CIPHER	WRAPPER
MASTER_SSL_KEY	WRITE (R)
MASTER_SSL_VERIFY_SERVER_CERT (R)	X509
MASTER_USER	XA

MATCH (R)	XML
MAXVALUE (R)	XOR (R)
MAX_CONNECTIONS_PER_HOUR	YEAR
MAX_QUERIES_PER_HOUR	YEAR_MONTH (R)
ZEROFILL (R)	-

টেবিল

টেবিল তৈরি

CREATE TABLE কমান্ড দিয়ে টেবিল তৈরি করতে হয়। নিচে এটি টেবিল তৈরি করার জন্য SQL কমান্ড লেখা হয়েছে। বলা হয়েছে যদি `table_name` নামে কোন টেবিল না থাকে তবে টেবিল তৈরি করতে, যার ইঞ্জিন MyISAM, ডিফল্ট ক্যারেক্টার সেট utf8 এবং কোলেট utf8_unicode_ci।

```
CREATE TABLE IF NOT EXISTS `table_name` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `column_name1` varchar(100) NOT NULL,  
  `column_name2` varchar(100) NOT NULL,  
  `column_name3` varchar(50) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

টেবিল সম্পাদনা

ALTER TABLE কমান্ড দিয়ে টেবিলের স্কিমা/স্ট্রাকচার পরিবর্তন করা যায়।

উদাহরণঃ টেবিলে `name` নামে একটি নতুন কলাম যুক্ত করা হয়েছে নিচের কমান্ড দিয়ে।

```
ALTER TABLE `table_name` ADD `name` VARCHAR(50) NOT NULL AFTER `id`;
```

টেবিল মুছে ফেলা

DROP TABLE কমান্ড দিয়ে টেবিল ডাটাবেস থেকে মুছে ফেলা যায়।

```
DROP TABLE table_name
```

বেসিক এসকিউএল কমান্ড'স

সিলেক্ট

SQL SELECT সিনট্যাক্স

```
SELECT column_name,column_name  
FROM table_name;
```

এবং

```
SELECT * FROM table_name;
```

ইনসার্ট

SQL INSERT INTO সিনট্যাক্স

কলামের নাম ছাড়া

```
INSERT INTO table_name  
VALUES (value1,value2,value3,...);
```

কলামের নাম সহ

```
INSERT INTO table_name (column1,column2,column3,...)  
VALUES (value1,value2,value3,...);
```

আপডেট

SQL UPDATE সিনট্যাক্স

```
UPDATE table_name  
SET column1=value1,column2=value2,...  
WHERE some_column=some_value;
```

ডিলিট

SQL DELETE সিনট্যাক্স

```
DELETE FROM table_name
WHERE some_column=some_value;
```

WHERE

SQL WHERE সিনট্যাক্স

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....
```

WHERE এর অপারেটর সমূহঃ

অপারেটর	বর্ণনা
=	সমান বোঝাতে
<>	সমান নয় বোঝাতে
!=	সমান নয় বোঝাতে
>	বৃহত্তর বোঝাতে
<	ক্ষুদ্রতর বোঝাতে
>=	বৃহত্তর অথবা সমান বোঝাতে
<=	ক্ষুদ্রতর অথবা সমান বোঝাতে
BETWEEN	দুইয়ের মধ্যে আছে বোঝাতে
LIKE	খোজা বোঝাতে
IN	একাধিক সম্ভাব্য মান আছে বোঝাতে

উদাহরণঃ

```
SELECT * FROM Customers
WHERE Country='Bangladesh';
```

```
SELECT * FROM Customers
WHERE Id=1;
```

```
SELECT * FROM Customers
WHERE Id=1 AND Country='Bangladesh';
```

```
SELECT * FROM Customers
WHERE Id=1 OR Country='Bangladesh';
```

ORDER BY

এক বা একাধিক কলামের ডাটা সর্ট করতে ORDER BY ক্লস ব্যবহৃত হয়ে থাকে। ডিফল্ট ORDER ASC এসেন্ডিং (আরোহী)।

ASC = **Ascending** / আরোহী / উর্ধ্বক্রম

DESC = **Descending** / অবরোহী / নিম্নক্রম

SQL ORDER BY সিনট্যাক্স

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

উদাহরণঃ

```
SELECT * FROM Customers
ORDER BY Country;
```

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

GROUP BY

একই ধরনের ডাটার GROUP তৈরি করতে GROUP BY ক্লস ব্যবহৃত হয়ে থাকে।

SQL ORDER BY সিনট্যাক্স

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;
```

উদাহরণঃ

```
SELECT Shippers.ShipperName,COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders
LEFT JOIN Shippers
ON Orders.ShipperID=Shippers.ShipperID
GROUP BY ShipperName;
```

```
SELECT Shippers.ShipperName, Employees.LastName,
COUNT(Orders.OrderID) AS NumberOfOrders
FROM ((Orders
INNER JOIN Shippers
ON Orders.ShipperID=Shippers.ShipperID)
INNER JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID)
GROUP BY ShipperName,LastName;
```

SQL Alias

আপনি চাইলে কোন টেবিল অথবা কলামের নাম সাময়িক ভাবে পরিবর্তন করতে পারেন এলিয়াস ব্যবহার করে।

SQL Alias সিনট্যাক্স

```
SELECT column_name AS alias_name
FROM table_name;
```

উদাহরণঃ

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

```
SELECT CustomerName AS Customer, ContactName AS Contact_Person
FROM Customers;
```

```
SELECT CustomerName, CONCAT(Address,',',City,',',PostalCode,',',Country) AS Address
FROM Customers;
```

স্ট্রিং অপারেশন

কোন কলামের স্ট্রিং ডাটা আপার-কেসে দেখানোর জন্য `UCASE()` ফাংশন টি ব্যবহার করা হয়।

SQL UCASE() সিনট্যাক্স

```
SELECT UCASE(column_name) FROM table_name;
```

কোন কলামের স্ট্রিং ডাটা লোয়ার-কেসে দেখানোর জন্য `LCASE()` ফাংশন টি ব্যবহার করা হয়।

SQL LCASE() সিনট্যাক্স

```
SELECT LCASE(column_name) FROM table_name;
```

কোন কলামের স্ট্রিং ডাটা ছোট করে দেখানোর জন্য `MID()` ফাংশন টি ব্যবহার করা হয়। একই কাজ `SUBSTRING()` ফাংশন দিয়েও করা যায়।

SQL MID() সিনট্যাক্স

```
SELECT MID(column_name,start,length) AS some_name FROM table_name;
```

SQL SUBSTRING সিনট্যাক্স

```
SELECT SUBSTRING(column_name,start,length) AS some_name FROM table_name;
```

উদাহরণঃ

```
SELECT MID(city,1,4) AS short_city  
FROM customers;
```

```
SELECT SUBSTRING(city,1,4) AS short_city  
FROM customers;
```

দুই বা ততোধিক কলামের ডাটা এক সাথে যুক্ত করার জন্য `CONCAT()` ফাংশনটি ব্যবহার করা হয়।

SQL CONCAT() সিনট্যাক্স

```
SELECT CONCAT(str1,str2,...) AS al_name  
FROM table;
```

উদাহরণঃ

```
SELECT CONCAT(col_1,col_2) AS al_name  
FROM table;
```

স্ট্রিং সম্পর্কিত আরও ফাংশন সম্পর্কে জানতে [এই সাইট](#) ভিজিট করুন ।

গাণিতিক অপারেশন

কোন কলামের ডাটাসমূহের গড় বের করার জন্য `AVG()` ফাংশনটি ব্যবহার করা হয়।

SQL AVG() সিনট্যাক্স

```
SELECT AVG(column_name) FROM table_name
```

উদাহরণঃ

```
SELECT product_name, price FROM products  
WHERE price > (SELECT AVG(price) FROM products);
```

কোন কলামের ডাটাসমূহের থেকে সর্বোচ্চ মান বের করার জন্য `MAX()` ফাংশনটি ব্যবহার করা হয়।

SQL MAX() সিনট্যাক্স

```
SELECT MAX(column_name) FROM table_name;
```

উদাহরণঃ

```
SELECT MAX(price) AS max_price FROM products;
```

কোন কলামের ডাটাসমূহের থেকে সর্বনিম্ন মান বের করার জন্য `MIN()` ফাংশনটি ব্যবহার করা হয়।

SQL MIN() সিনট্যাক্স

```
SELECT MIN(column_name) FROM table_name;
```

উদাহরণঃ

```
SELECT MIN(price) AS min_price FROM products;
```

কোন কলামের ডাটাসমূহের যোগফল বের করার জন্য `SUM()` ফাংশনটি ব্যবহার করা হয়।

SQL SUM() সিনট্যাক্স

```
SELECT SUM(column_name) FROM table_name;
```

উদাহরণঃ

```
SELECT SUM(quantity) AS total_order FROM orders;
```

কোন কলামের ডাটাকে দশমিকের নির্দিষ্ট ঘর অথবা দশমিক সংখ্যাকে পূর্ণসংখ্যা হিসাবে দেখানোর জন্য `ROUND()` ফাংশনটি ব্যবহার করা হয়।

SQL ROUND() সিনট্যাক্স

```
SELECT ROUND(column_name,decimals) FROM table_name;
```

উদাহরণঃ

```
// price = 21.3545
SELECT products, ROUND(price,2) AS new_price
FROM products; // price = 21.35

SELECT products, ROUND(price) AS new_price
FROM products; // price = 21
```

লজিক্যাল অপারেশন (ড্রাফট)

```
CASE case_value
  WHEN when_value THEN statement_list
  [WHEN when_value THEN statement_list] ...
  [ELSE statement_list]
END CASE
```

Or:

```
CASE
  WHEN search_condition THEN statement_list
  [WHEN search_condition THEN statement_list] ...
  [ELSE statement_list]
END CASE
```

```
IF search_condition THEN statement_list
  [ELSEIF search_condition THEN statement_list] ...
  [ELSE statement_list]
END IF
```

```
[begin_label:] WHILE search_condition DO
  statement_list
END WHILE [end_label]
```

JOIN

দুই বা ততোধিক টেবিলের রো একসাথে যুক্ত করতে `SQL JOIN` ব্যবহার করা হয়। SQL এ JOIN মোট ৪ প্রকারের।

INNER JOIN

INNER JOIN কি-ওয়ার্ড দিয়ে দুই বা ততোধিক টেবিলের রো একসাথে যুক্ত করতে হলে বাম ও ডান কলামের মান সমান হতে হয়।

`SQL INNER JOIN` সিনট্যাক্স

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name=table2.column_name;
```

অথবা,

```
SELECT column_name(s)
FROM table1
JOIN table2
ON table1.column_name=table2.column_name;
```

উদাহরণঃ

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
INNER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

LEFT JOIN

`LEFT JOIN` কি-ওয়ার্ড দিয়ে দুই বা ততোধিক টেবিলের রো একসাথে যুক্ত করতে হলে বাম কলামের রো এর ডাটার সাথে ডান কলামের রো এর ডাটা তুলনা করা হয়। যেসব স্থানে উভয় কলামের রো এর ডাটা সমান নয় সেসকল ক্ষেত্রে শুধুমাত্র বাম কলামের ডাটা সিলেক্ট করা হয়, আর যেসব স্থানে উভয় কলামের রো এর ডাটা সমান সে ক্ষেত্রে উভয় কলামের রো এর ডাটা সিলেক্ট করা হয়।

`SQL LEFT JOIN` সিনট্যাক্স

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

অথবা,

```
SELECT column_name(s)
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

উদাহরণঃ

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

RIGHT JOIN

RIGHT JOIN কি-ওয়ার্ড দিয়ে দুই বা ততোধিক টেবিলের রো একসাথে যুক্ত করতে হলে বাম কলামের রো এর ডাটার সাথে ডান কলামের রো এর ডাটা তুলনা করা হয় । যেসব স্থানে উভয় কলামের রো এর ডাটা সমান নয় সেসকল ক্ষেত্রে শুধুমাত্র ডান কলামের ডাটা সিলেক্ট করা হয় , আর যেসব স্থানে উভয় কলামের রো এর ডাটা সমান সে ক্ষেত্রে উভয় কলামের রো এর ডাটা সিলেক্ট করা হয় । এটি LEFT JOIN এর বিপরীত ।

SQL RIGHT JOIN সিনট্যাক্স

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name;
```

অথবা,

```
SELECT column_name(s)
FROM table1
RIGHT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

উদাহরণঃ

```
SELECT Orders.OrderID, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees  
ON Orders.EmployeeID=Employees.EmployeeID  
ORDER BY Orders.OrderID;
```

FULL JOIN

FULL JOIN মূলত LEFT JOIN এবং RIGHT JOIN এর ফলাফল একসাথে যুক্ত করে প্রকাশ করে।

SQL FULL OUTER JOIN সিনট্যাক্স

```
SELECT column_name(s)  
FROM table1  
FULL OUTER JOIN table2  
ON table1.column_name=table2.column_name;
```

উদাহরণঃ

```
SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders  
ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;
```

সময় ও তারিখ এর ব্যবহার

মাইসিকুয়েল এ সময় এবং তারিখ নিয়ে কাজ করার জন্য বেশ কিছু বিল্ট-ইন ফাংশন আছে।

NOW() ফাংশনটি বর্তমান সময় Y-m-d H:i:s ফরম্যাটে রিটার্ন করে।

উদাহরণঃ

```
SELECT NOW() FROM table_name;
```

CURDATE() ফাংশনটি বর্তমান তারিখ Y-m-d ফরম্যাটে রিটার্ন করে।

উদাহরণঃ

```
SELECT * FROM table_name WHERE date_col = CURDATE();
```

CURTIME() ফাংশনটি বর্তমান সময় H:i:s ফরম্যাটে রিটার্ন করে।

উদাহরণঃ

```
SELECT * FROM table_name WHERE date_col = CURTIME();
```

DATE() ফাংশনটি ডাটা থেকে Y-m-d ফরম্যাটে তারিখ রিটার্ন করে।

উদাহরণঃ

```
SELECT ProductName, DATE(OrderDate) AS OrderDate  
FROM Orders  
WHERE OrderId=1
```

DATEDIFF() ফাংশনটি দুইটি তারিখের মধ্যবর্তী সময়/দিন প্রকাশ করে।

উদাহরণঃ

```
SELECT DATEDIFF('2014-11-29','2014-11-30') AS DiffDate
```


অপারেটর'স

অপারেটর হচ্ছে এক ধরনের সংরক্ষিত শব্দ যা সাধারণত SQL এর WHERE স্টেটমেন্ট এ গাণিতিক এবং তুলনা সম্পর্কিত কাজ করতে সহায়তা করে।

এরিথম্যাটিক (গাণিতিক) অপারেটর

এই অপারেটর গুলো গাণিতিক হিসাব নিকাশ করতে ব্যবহৃত হয়।

অপারেটর	কাজ
+	যোগ করতে ব্যবহৃত হয়।
-	বিয়োগ করতে ব্যবহৃত হয়।
*	গুন করতে ব্যবহৃত হয়।
/	ভাগ করতে ব্যবহৃত হয়।
%	ভাগশেষ বের করতে ব্যবহৃত হয়।

কম্পারিসন (তুলনা) অপারেটর

এই অপারেটর গুলো দুই বা ততোধিক উপাদানের মধ্যে তুলনা করতে ব্যবহৃত হয়।

অপারেটর	কাজ
=	উভয় পক্ষ সমান বোঝাতে ব্যবহৃত হয়।
!=	উভয় পক্ষ সমান নয় বোঝাতে ব্যবহৃত হয়।
<>	উভয় পক্ষ সমান নয় বোঝাতে ব্যবহৃত হয়।
>	বামপক্ষ ডানপক্ষের তুলনায় বড় বোঝাতে ব্যবহৃত হয়।
<	বামপক্ষ ডানপক্ষের তুলনায় ছোট বোঝাতে ব্যবহৃত হয়।
>=	বামপক্ষ ডানপক্ষের তুলনায় বড় অথবা সমান বোঝাতে ব্যবহৃত হয়।
<=	বামপক্ষ ডানপক্ষের তুলনায় ছোট অথবা সমান বোঝাতে ব্যবহৃত হয়।
!<	বামপক্ষ ডানপক্ষের তুলনায় ছোট বোঝাতে ব্যবহৃত হয়।
!>	বামপক্ষ ডানপক্ষের তুলনায় বড় বোঝাতে ব্যবহৃত হয়।

লজিক্যাল (যুক্তি) অপারেটর

অপারেটর	কাজ
ALL	এক সেটের সকল ড্যালা অন্য সেটের সকল ড্যালার সাথে তুলনা করতে ব্যবহৃত হয়।
AND	WHERE এ একাধিক কন্ডিশন ব্যবহার করতে এই অপারেটর ব্যবহৃত হয়।
BETWEEN	সর্বোচ্চ এবং সর্বনিম্ন মানের মধ্যে কাঙ্ক্ষিত মান খুঁজতে এই অপারেটর ব্যবহৃত হয়।
IN	একটি লিস্টের মান সমূহ খুঁজতে এই অপারেটর ব্যবহৃত হয়।
OR	WHERE এ একাধিক কন্ডিশন ব্যবহার করতে এই অপারেটর ব্যবহৃত হয়।
IS NULL	কোন ডাটার মান NULL কিনা জানতে অপারেটর ব্যবহৃত হয়।
UNIQUE	সকল রো তে যা ডাটা আছে তা মৌলিক কিনা জানতে অপারেটর ব্যবহৃত হয়।

ডাটাবেস অপটিমাইজেশন

কুয়েরী অপটিমাইজেশন (ড্রাফট)

“কুয়েরী অপটিমাইজেশন” হচ্ছে ডাটাবেজ ম্যানেজমেন্ট সিস্টেম এর একটা ফাংশন। পসিবল কুয়েরী প্লান থেকে কিভাবে এফিশিয়েন্ট কুয়েরী লেখা এবং এক্সিকিউট করা যায় তা ঠিক করে কুয়েরী অপটিমাইজার।

যেভাবে কুয়েরী অপটিমাইজেশন করতে হয়

- Join ordering
- Query planning for nested SQL queries
- Cost estimation

Join ordering

কুয়েরী প্লানের পারফরমেন্স মূলত কোন অর্ডারে টেবিল জয়েন করা হয়েছে তার উপর নির্ভর করে। মনে করুন তিনটি টেবিল A,B এবং C এদের রো আছে যথাক্রমে ১০,১০০,২০০। কোন কুয়েরী প্লানে যদি প্রথমে B ও C জয়েন করানো হয় এবং শেষে তাদের সাথে A জয়েন করানো হয় তাহলে কুয়েরী এক্সিকিউট করতে সময় ও মেমোরি বেশি লাগবে। কিন্তু যদি প্রথমে A ও B এবং শেষে C জয়েন করানো হয় তাহলে সময় এবং মেমোরি কম লাগবে। কারন A তে রো এর সংখ্যা কম।

Cost estimation

1)

```
SELECT * FROM `users`
```

সাধারণত * দ্বারা সব কলাম বা ফিল্ড এর ড্যালাু এর পাওয়া যায়। কিনতু যদি আমাদের শুধু মাত্র কয়েকটি কলাম বা ফিল্ড এর দরকার পড়ে, তাহলে আমরা শুধুমাত্র ওই কয়েকটি কলাম বা ফিল্ড এর নাম উল্লেখ করে দিতে পারি।

```
SELECT email,password FROM `users`
```

2)

```
SELECT * FROM `users` WHERE name='niloy'
```

যখন আপনি একটি রো এক্সপেক্ট করছেন তখন LIMIT 1 ব্যবহার করাই ভাল

```
SELECT * FROM `users` WHERE name='niloy' LIMIT 1
```

3) ইনডেক্সিং । ইনডেক্সিং ব্যবহার করে আপনার কুয়েরী টাইম অনেক কমিয়ে নিয়ে আসতে পারেন । তবে আপনি সেইসব সব ফিল্ডই ইনডেক্সিং করবেন , যে সব ফিল্ড আপনি WHERE কন্ডিশন এ ব্যবহার করছেন ।