
MySQL®
ව්‍යුහගත විමසුම් බස
තොරතුරු හා සන්නිවේදන තාක්ෂණය
අ.පො.ස (උ.පෙළ)
දිනේෂ් එම්. බණ්ඩාර
Lecturer In ICT

MySQL – හැඳින්වීම

MySQL යනු SQL(Structured Query Language) මගින් ව්‍යුත්පන්න වූ Free and Open DBMS(Database Management System) එකකි. මෙය Server එකක් ලෙස ක්‍රියා කරන බැවින් එය හා සම්බන්ධ වීමට අපට තවත් මෘදුකාංග ඉතා සහසුවෙන් භාවිතා කල හැක.තවද වෙනත් පරිගණක භාෂාවන් සමඟ වුවද ඉතාමත් පහසුවෙන් සම්බන්ධ කල හැකි නිසා අද වන විට ඉතාමත් ජනප්‍රිය වී ඇත.එමෙන්ම මෙය පරිගණක ජාලයක් තුල වුවද ඉතාමත් පහසුවෙන් ස්ථාපිත කර භාවිතා කලහැකි බැවින් Servers වල දත්ත කළමණාකරණය සඳහා භාවිතා කරන ප්‍රභල මෙවලමක් වී ඇත.

MySQL හි විශේෂතා

- MySQL Server එක සඳහා අවශ්‍ය වන ඉඩ ප්‍රමාණය අඩුය උපරිම වශයෙන්).100Mb)
- ස්ථාපනය කිරීමට වෙනත් Software අනවශ්‍යයි.
- වෙනත් පරිගණක භාෂා)Java,PHP) මගින් පහසුවෙන් MySQL Server එකට සම්බන්ධ විය හැක.
- සාමාන්‍ය ඉංග්‍රීසි වචන භාවිතා වන නිසා ඉගෙන ගැනීම සහ භාවිතය පහසුය.

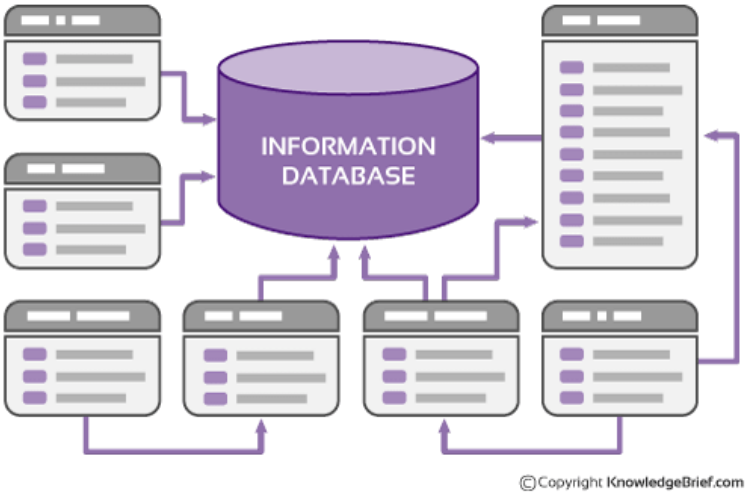
MySQL භාවිතා කිරීමට නම්....

- පළමුව MySQL Server එක download කරගෙන install කරගත යුතුය.

  Server Side Programming කිරීමට බලාපොරොත්තු වේ නම්:උදා)PHP) WAMP,XAMPP,LAMP වැනි Package එකක් වුවද install කර ගතහැක.
- එය නිවැරදිය configure කරගත යුතුය.
- Third Party Software අවශ්‍ය නම් ඒවාද install කර ගන්න.

  උදා:
  - MySQL Query Browser
  - Heidi SQL
  - MySQL GUI Tools
  - MySQL Administrative Tools
  - PHPMyAdmin(PHP සහිත server එක්ක භාවිතා කරන්නේ නම් පමණි(.
- දත්ත සමුදා කළමණාකරණය ගැන න්‍යායික දැනුම.

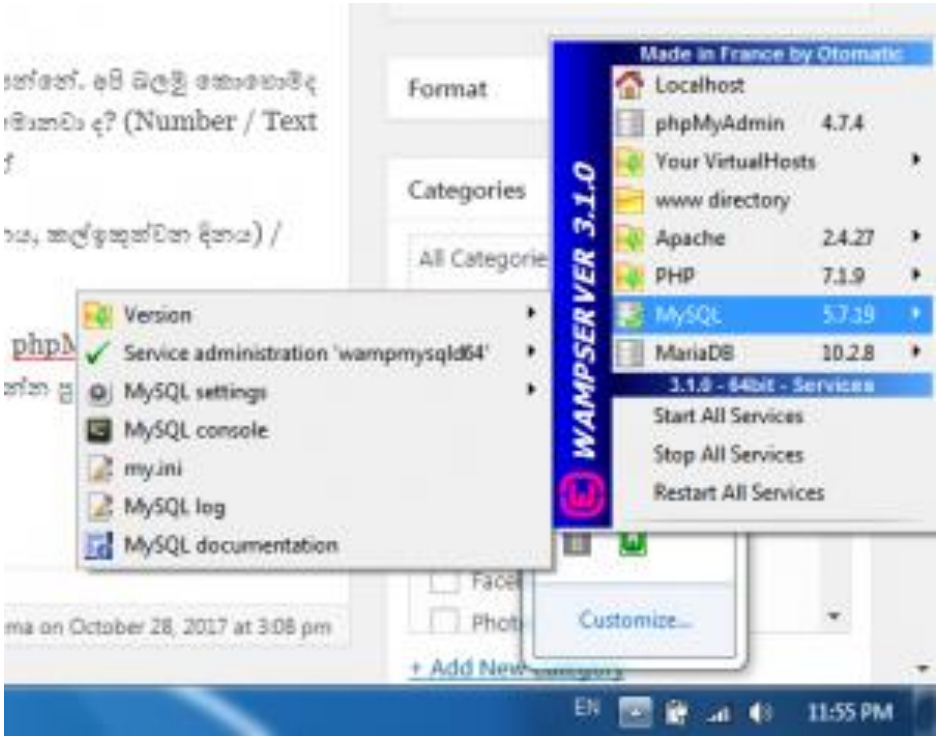
  නිවැරදිව Database එකක් නිර්මාණය කර එය නිවැරදිව හා කාර්යක්ෂමව ක්‍රියා කරවීම සඳහා න්‍යායික කරුණු අත්‍යාවශ්‍ය වේ



මේ විදියට දත්ත ගබඩා කරන ක්‍රම ගොඩක් තියෙනවා. SQL Azure, MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgresql, MongoDB ආදිය දක්වන්න පුළුවන්.

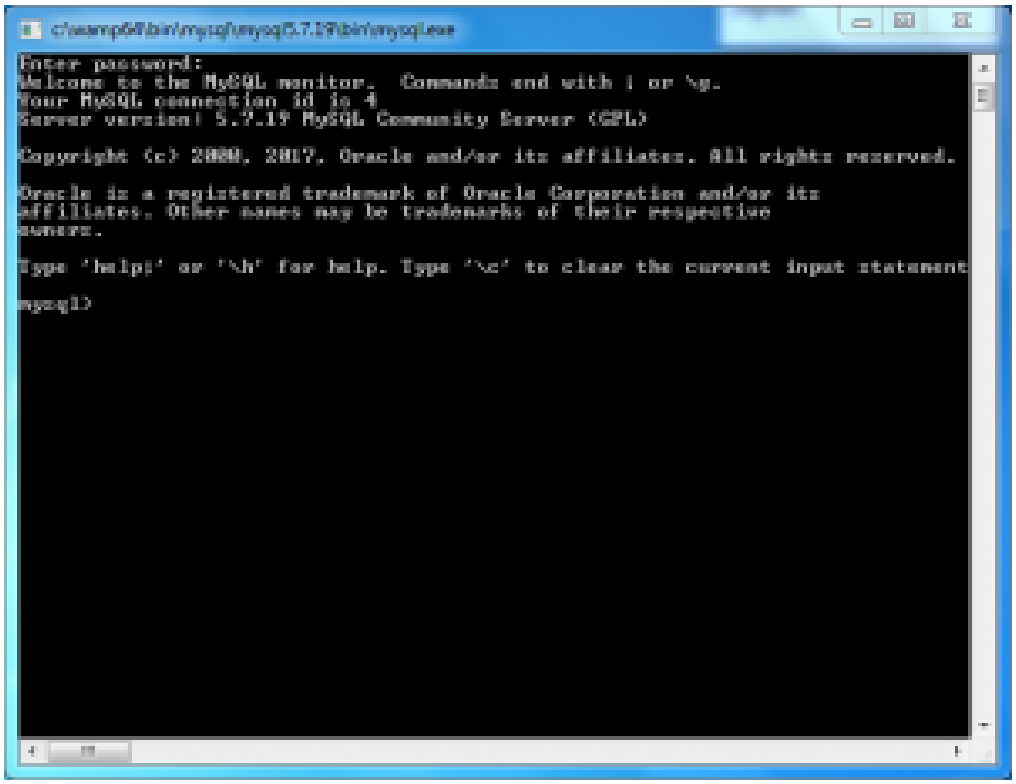
අපි දැන් WampServer එක දැනට ස්ථාපනය කරලා තියෙන නිසා MySQL එකත් ඉන්ස්ටෝල් වෙලා තියෙන්නේ. අපි බලමු කොහොමද මුලින්ම ඩේටා බේස් එකක් හදාගන්නේ කියලා. ඒකට අපි දැනගන්න ඕන අපි මේ ගබඩා කරන තොරතුරු මොනවා ද? (Number / Text / Date & Time) කියලා සහා අපේ තොරතුරු වගුගත කරන්න හැකි පරිදි වර්ග කරලා පහත උදාහරණ වගේ
වර්ගීකරණය (පුද්ගල තොරතුරු (නම,වයස,ගම,රැකියාව) / භාණ්ඩ තොරතුරු (භාණ්ඩය,මිල, නිශ්පාදිත දිනය, කල්ඉකුත්වන දිනය) / සේවක තොරතුරු (නම, දෙපාර්තමේන්තුව, බැඳුන දිනය, සේවක අංකය)

MySQL සඳහා අපිට පරීශීලක මුහුණත් දෙකක් තිබෙනවා එකක් තමා console එක හරහා භාවිතයත් අනික phpMyAdmin අතුරු මුහුණත. මේ දෙකෙන් වඩාත් පහසු මොකක් ද කීවොත් phpMyAdmin තමා. නමුත් මේ SQL Query ලියන්න පුරුදු වෙන්න නම් console එක ලොකු උදව්වක්.

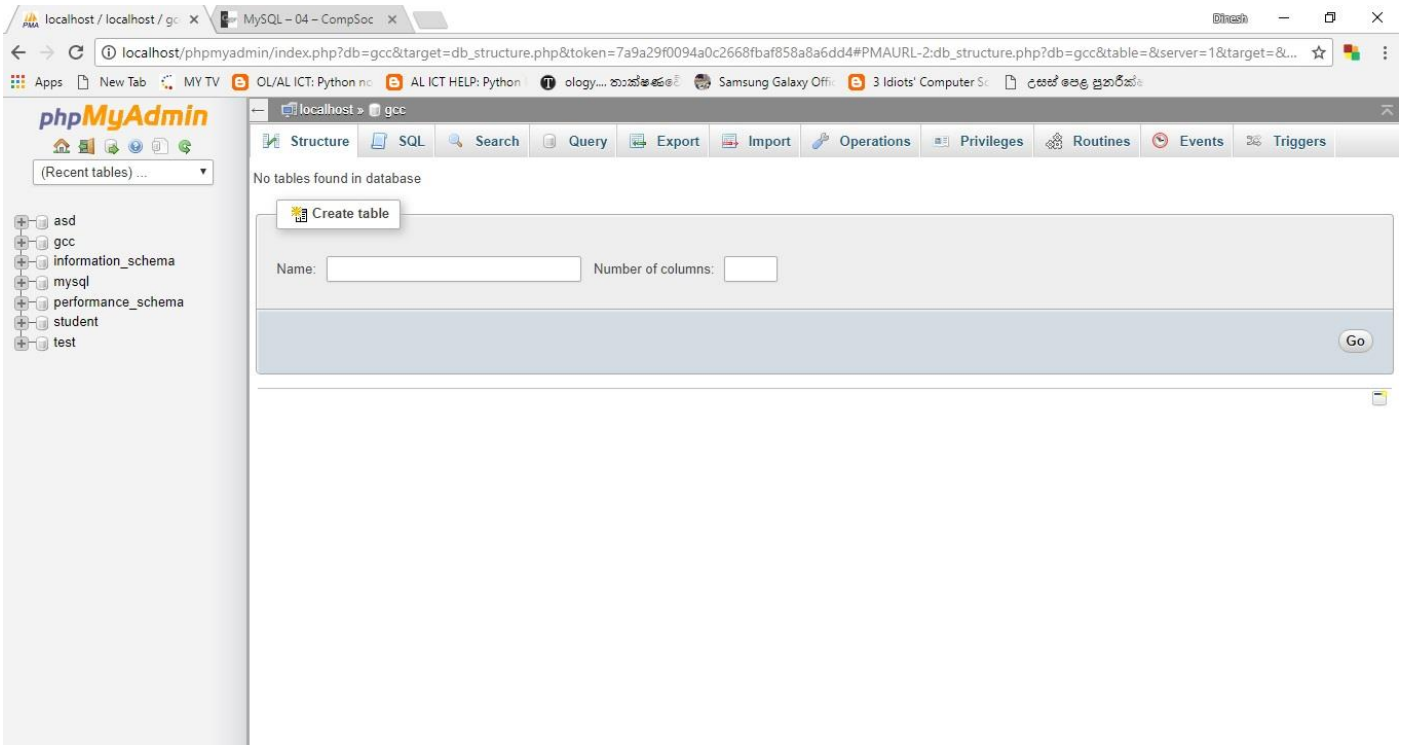


toolbar එකේ තියෙන වැම්ප් අයිකන් එක උඩ click කරාම ඉහත ආකාරයේ මෙනුවෙන් MySQL > MySQL console එක හෝ ඉහල ඇති phpMyAdmin තෝරාගන්න පුළුවන්.

දැන් මේ දෙකෙන් මොකක් විවෘත කරත් මුලින්ම database එකේ ලොගින් ඉල්ලනවා. එතකොට ස්ථාපනය කරන අවස්ථාවේ දුන්න මුරපදය ඇතුලත් කරන්න ඕන. ගොඩක් වෙලාවට default username එක විදියට root තමා ඇතුලත් වෙන්නේ සහා ඊට මුරපදයක් නැහැ. Enter කිරීමෙන් පිවිසෙන්න පුළුවන්.

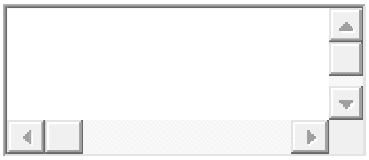


දැන් phpMyAdmin වෙත පිවිසුනාම බලාගන්න පුළුවන් දැනට අපේ පවතින database මොනවාද ඒ ඒ database එකට අදාල table වර්ග ඒ ඒ table වල තියෙන දත්ත. ආරක්ෂණ විධිවිධාන ඇතුලු සැකසුම් රාශියක්. ඉතින් මේක ගැන වෙනම ඉගෙනගන්න ඕන.



ටිකක් පුරුදු වෙනකම් මම කොන්සෝල් එක තුලින්ම වැඩකරන්නම්.

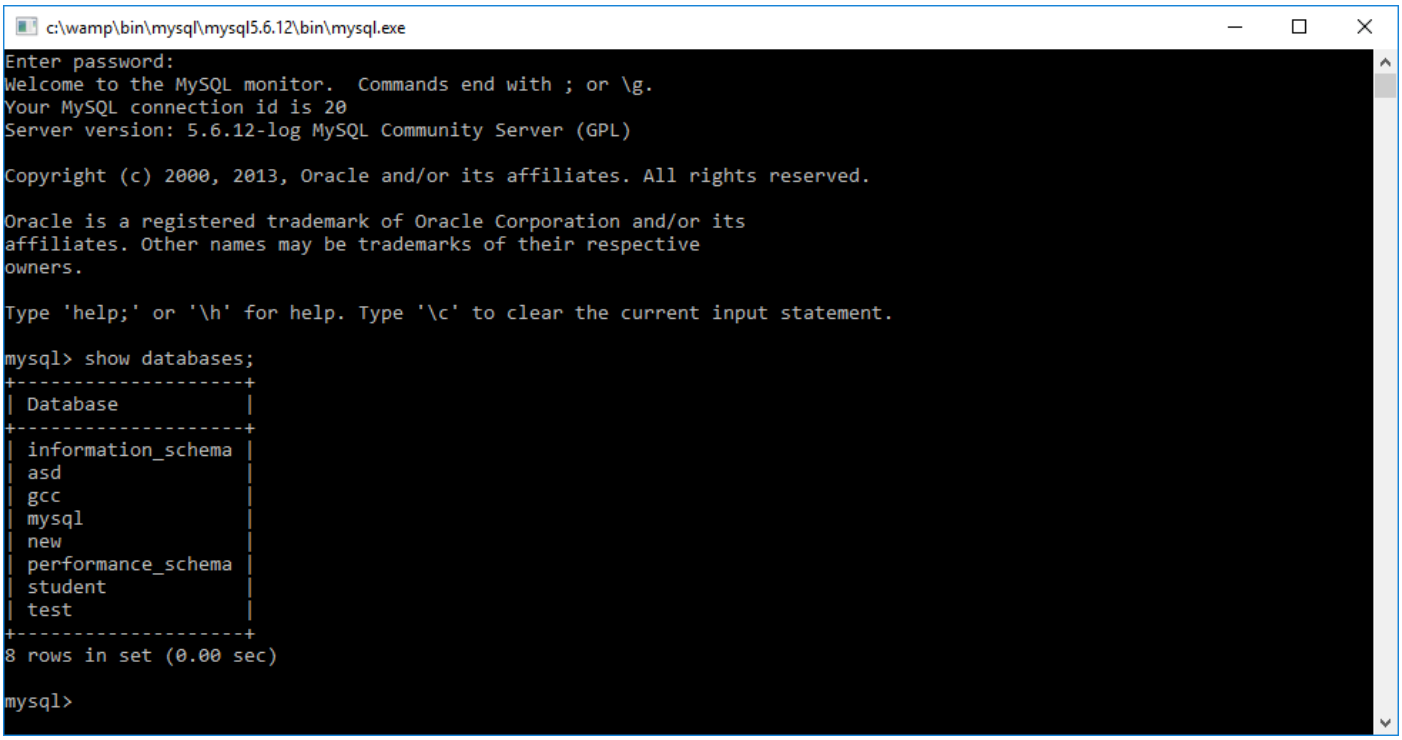
console එකේ නම් ඔයාට තියෙන database බලාගන්න පහත query එකම ඇතුලත් කරන්න ඕන.



1 show databases;

මතක තියාගන්න SQL කියන්නෙ query language එකක් ඔයාගේ කමාන්ඩ් එකක් අවසානය දැක්වෙන delimiter එක හැම කමාන්ඩ් එකක් අවසානයේදීම ඇතුලත් කරන්නම ඕන. නැතිනම් ඔයා ඒක දෙනකම් අදාල කමාන්ඩ් එක වැඩ කරන්නේ නෑ. සාමාන්‍යයෙන් delimiter එක වෙන්නේ “;” .

දැන් ඉහත කමාන්ඩ් එක දැම්මාම පහතින් තියෙනවා වගේ සියලුම database පෙන්වාවි.



MySQL – Basics

දැන් MySQL හි භාවිතා වන මූලික command ගැන සලකා බලමු.

show databases ;

දැනට පවතින MySQL Databases වල list එකක් ලබා ගැනීම සඳහා භාවිතා කරයි.

create database ;

අළුතින් database එකක් නිර්මාණය කිරීම සඳහා භාවිතා කරයි.

උදා : `create database myDB ;`

drop database ;

දැනට පවතින database එකක් delete කිරීම සඳහා භාවිතා කරයි.

උදා : `drop database myDB ;`

use ;

Database එකක් open කර ගැනීමට භාවිතා කරයි.

උදා : `use myDB ;`

show tables ;

දැනට Open කර ඇති database එකේ ඇති tables වල list එකක් ලබා ගැනීම සඳහා භාවිතා කරයි.

desc ;

දෙන ලද table එකක meta data ලබා ගැනීම සඳහා භාවිතා කරයි.

උදා : `desc myTable ;`

NOTE :

`create` සහ `drop` commands සඳහා `if exists` හෝ `if not exists` යන යෙදුම් භාවිතා කිරීමෙන් ඒවා ක්‍රියාත්මක වීමේදී ඇතිවන errors නවතාලිය හැක එසේ .MySQL Script එකක් run කරවීමේදී ඇති විය හැකි errors අවම කලයුතු අතර මන්දයත්, මෙම errors නිසා සමහර විට අපගේ ඉතිරි commands run නොවීමට ඉඩ ඇති බැවිනි.

පැහැදිලි කිරීම : `create database myDB ;` ලෙස ලබා දුන් විට දැනටමත් myDB ලෙස database එකක් පවතී නම් error එකක් ලබා දේඑවිට . `create database if not exist myDB;` ලෙස ලබා දුන් විට myDB ලෙස database එකක් නැතිනම් පමණක් එය නිර්මාණය කරයි. `drop database myDB ;` ලෙස ලබා දුන් විට දැනට myDB ලෙස database එකක් නැතිනම් error එකක් ලබා දේඑවිට . `drop database if exist myDB;` ලෙස ලබා දුන් විට myDB ලෙස database එකක් තිබේ නම් පමණක් එය ඉවත් කරයි.

Tables නිර්මාණය කිරීම

මේ සඳහා `create table`command එක භාවිතා කරයි මෙහි.syntax එක විස්තර කරනවාට වඩා උදාහරණ මගින් පැහැදිලි කරන විට වඩා හොඳින් අවබෝධ කරගත හැකි වනු ඇත.NOTE:

MySQL Scripts run කිරීම සඳහා අප භාවිතා කරන්නේ MySQL CommandLine Client බැවින් type කරන commands edit කර ගැනීම අපහසුය එනිසා ඒවා වෙනම .text file එකක type කර පසුව MySQL CommandLine Client හි paste කරන්න එවිට.type කල command එක දෝෂ සහගත වුවත් text file එකේ ඇති copy එක වෙනස් කර නැවත භාවිතා කල හැක.

සරල Table එකක් නිර්මාණය කිරීමට ලියා ඇති සරලම ආකාරයේ MySQL script එකක් පහත දැක්වේ මෙය) .run කිරීමට පෙර ඔබට අවශ්‍ය database එක නිර්මාණය කර `use` command එක මගින් එය තුලට ඇතුලත් වී තිබිය යුතුය(.

```
create table student(student_id varchar(10) not null,student_name varchar(50)
not null,age int(2) default 5);
```

මෙය කියවීමට අපහසු බැවින් පහත පරිදි එය පේලි වලට වෙන් කර type කිරීම සම්ප්‍රදායිකව සිදු කරනු ලබයි.

```
create table student(
        student_id varchar(10) not null,
        student_name varchar(50) not null,
        age int(2) default 5
);
```

පැහැදිලි කිරීම :

- `create table student` – මෙහි student යනු නිර්මාණය වීමට අවශ්‍ය table එකේ name එක වේ.
- ; – ඕනෑම MySQL command එකක් අවසානයේ semi-colon (;) එකක් යෙදිය යුතුය.
- `student_id varchar(10) not null`

  `student_id` යනු අදාල column එකේ name එකයි.

  `varchar(10)` හි varchar යනු student_id සඳහා ලබාදෙන data type එකයි වරහන් තුල ලබා දී ඇත්තේ ඒ . .සඳහා තිබිය හැකි උපරිම අකුරු ගණන වේ

  `not null` යන්නෙන් නිරූපණය වන්නේ student_id සඳහා අනිවාර්යෙන්ම value එකක් තිබිය යුතු බවයි.
- `student_name varchar(50) not null` – ඉහත අයුරින්ම වේ.
- `age int default 5`

  `age` යනු අදාල column එකේ name එකයි.

  `int` යනු age සඳහා ලබාදෙන data type එකයි.

  `default 5` යන්නෙන් නිරූපණය වන්නේ age සඳහා value එකක් ලබා නොදුනහොත් ඒ සඳහා MySQL විසින් ලබාදිය යුතු default value එකයි.

Columns සඳහා යෙදිය හැකි ප්‍රධාන data types

varchar	Variable Characters	අකුරු, ඉලක්කම් වැනි ගණිතමය වටිනාකමක් රහිත විවිධ දිගින් යුත් text සඳහා භාවිතා කරයි . උපරිම අකුරු ගණන255 කි: උදා. `varchar(10)` ලෙස ඇති නම් උපරිම වශයෙන් අකුරු 10 කින් යුත් text සඳහා භාවිතා කරයි.
char	Characters	දී ඇති නිශ්චිත දිගකින් යුත් අකුරු, ඉලක්කම් වැනි ගණිතමය වටිනාකමක් රහිත text සඳහා භාවිතා කරයි උපරිම අකුරු ගණන .255 කි: උදා. `nic char(10)` ලෙස ඇති නම් අනිවාර්යෙන් අකුරු 10 කින් යුත් text සඳහා භාවිතා කරයි.
int	Integers	නිඛිල)Integers) ගබඩා කිරීමට භාවිතා කරයි- .2147483648 සිට 2147483647 දක්වා හෝ 0 සිට 4294967295 දක්වා UNSIGNED (ධන සංඛ්‍යා පමණක්: උදා.වශයෙන් භාවිතා කලහැක (`int points` ලෙස ඇති නම් `int age unsigned` ලෙස ඇති නම් තිබිය හැක්කේ ධන සංඛ්‍යා පමණි. `int(5)` වශයෙන් ලියූ විට ZeroFill on කර තිබේ නම් ඉලක්කමක අනිවාර්යෙන් තිබිය යුතු ඉක්නම් ගණන නිරූපණය කලහැක එම ඉලක්කම් ගණනට අඩුවෙන් ඉලක්කම් ඇති අගයක් දුන් විය ඉතිරිය.0 වලින් පිරේ: උදා. `int(5) distance unsigned zerofill` ලෙස ඇති විට එම column එකට 3 ඇතුලත් කල විට එය ගබඩා වන්නේ 00003 ලෙසයි.

double	Floating Point Numbers	දශම සංඛ්‍යා ගබඩා කිරීමට භාවිතා කරයි- .1.7976931348623157E+308 සිට -2.2250738585072014E-308 දක්වා, 0, සහ 2.2250738585072014E-308 සිට 1.7976931348623157E+308 ඉලක්කම් භාවිතා කලහැක. `double(7,2)` වශයෙන් ලියූ විට ගබඩා කලහැකි ඉලක්කම් වල උපරිම වශයෙන් සංඛ්‍යා 7ක් ද දශම තිතෙන් පසුව උපරිම වශයෙන් ඉලක්කම් 2 ක් ද තිබිය හැකි බව නිරූපණය කලහැක: උදා. `qty double(6,2)` ලෙස ඇති නම් උපරිම වශයෙන් ඉලක්කම් දශම සංඛ්‍යාද) (ඇතුළුව6 ක් තිබිය හැකි අතර දශම කොටසේ ඉලක්කම් 2ක් උපරිම වශයෙන් තිබිය හැකි බව නිරූපණය කරයි.
boolean	True/False	true හෝ false ගබඩා කර ගබා ගනී: උදා. `isProcessed boolean`
date	Date	YYYY-MM-DD ආකාරයේ දිනයක් ගබඩා කර ගබා ගනී. `'1000-01-01'` සිට `'9999-12-31'` දක්වා දිනයන් ඇතුලත් කලහැක: උදා. `joinedDate date`
time	Time	HH:MM:SS ආකාරයේ වේලාවක් ගබඩා කර ගබා ගනී.`'-838:59:59'` සිට `'838:59:59'` දක්වා වේලාවන් ඇතුලත් කලහැක: උදා. `retunTime time`
datetime	Date & Time	YYYY-MM-DD HH:MM:SS ආකාරයෙන් දිනයක් සහ වේලාවක එකතුවක් ගබඩා කර ගබා ගනී. `'1000-01-01 00:00:00'` සිට `'9999-12-31 23:59:59'` දක්වා ඇතුලත් කලහැක: උදා. `fileCreated datetime`

MySQL – Data Integrity

MySQL තුළ දත්ත තළමනාකරණය කරන ක්‍රියාවලීන් මෙලෙස හඳුන්වයි ඒවා ප්‍රධාන කොටස් .3 කි.

1. Entity Integrity
2. Referencial Integrity
3. Domain Integrity

මෙහිදී මූලික වශයෙන් මුල් කරුණු 2 සලකා බලමු.

Entity Integrity

Table එකක් සැලකූ විට එහි ඇති records එකිනෙකින් වෙන් කර හඳුනා ගැනීමට හැකි විය යුතුය එනම් එකම දත්තය . ඒ සඳහා .පුනරාවර්ථ වීම වැලැක්විය යුතුයfields එකක් හෝ කිහිපයක් සලකා බලමින් එහි ඒවායේ ඇති දත්ත එකම විටෙක/ එසේ විශේෂිත වූ .නැවත යෙදීම වැලැක්විය යුතුයfields, Primary Key ලෙස හඳුන්වයි. උදා :

පාසලකට ළමයෙක් ඇතුලත් වන විට ඔහුට .ඇයට ඇතුලත්වීමේ අංකයක් හෝ ලියාපදිංචි වීමේ අංකයක් ලබා දේ/ එය, ඔහුව එනිසා එම ඇතුලත්වීමේ .ඇයව ලේඛන වලදී අන් අයගෙන් වෙන් කර හඳුනා ගැනීමට භාවිතා කරයි/ අංකය හෝ ලියාපදිංචි වීමේ අංකය ඇති තවත් අයෙක් සිසුන්ගේවිස්තර ඇති වගුවේ සිටිය නොහැකි බැවින්

අළුතින් තවත් ළමයින් ඇතුලත් කිරීමේදී එම ඇතුලත්වීමේ අංකය හෝ ලියාපදිංචි වීමේ අංකය වෙනත් ළමයෙක්ට ලබා දිය නොහැක.

ඒ අනුව student table එකේ index_no යනු Primary Key එකයි.

පුස්ථකාලයක පොත් ගැනීමේදී අදාල පොත්වල සහ සාමාජිකයාගේ විස්තර ගබඩා කරගත යුතුය ඒ සඳහා පහත. .ආකාරයක වගුවක් භාවිතා කරයි

Borrowing						
Book_No	Member_No	Date	Time	Due_Date	Returned_Date	Fine
B-001	M-001	2012-05-24	1:20:25	2012-05-31	2012-05-31	0.00
B-002	M-002	2012-05-24	1:25:30	2012-05-31	2012-05-25	0.00
B-002	M-001	2012-05-26	2:54:08	2012-06-02	2012-06-05	20.00

මෙහි Book_No පමණක් Primary Key විය නොහැක එසේ වුවහොත් එකම .Book_No එක නැවතත් මෙම වගුවේ ඇතුලත් විය නොහැකි නිසා එක් පොතක් ගැනිය හැක්කේ කවුරුන් හෝ එක් සාමාජිකයෙක්ට පමණි.

මෙහි Member_No පමණක් Primary Key වීමද සිදු විය නොහැක එසේ වුවහොත් එකම .Member_No එක නැවතත් මෙම වගුවේ ඇතුලත් විය නොහැකි නිසා එක් සාමාජිකයෙක්ට ගැනිය හැක්කේ කුමන හෝ හෝ එක් පොතක් පමණි.

ඊට අමතරව Book_No සහ Member_No යන දෙකම Primary Key කලහොත් යම් සාධාරණයක් සිදුවේ නමුත් . එවිට එකමBook_No එක, එකම Member_No එකක් සමඟ තවත් වතාවක් යෙදිය නොහැක එනම් එක් . .සාමාජිකයෙකුට එකම පොත දෙවතාවක් ගැනිය නොහැක

තවදුරටත් විමසා බැලුවිට, Book_No, Member_No සහ Date යන තුනම Primary Key කලහොත් වඩා නිවැරදි වේඑවිට එකම සාමාජිකයාට එකම පොතක් එකම දිනයක් තුලදී දෙවතාවක් ලබා ගත . නොහැක.

තවත් දුරට සලකා බැලූ විට, Book_No, Member_No, Date සහ Time යන හතරම Primary Key කලහොත් වඩාත්ම නිවැරදි වේ එවිට එකම සාමාජිකයාට එකම පොතක් එකම දිනයක් තුලදී වුවද වෙනස් වෙලාවන් හිදී . .ලබා ගත හැක

එනම් Book_No, Member_No, Date සහ Time යන හතරම Primary Key වේ. එසේ Fields එකකට වඩා වැඩියෙන් Primary Key වේ නම්, එවැනි Primary Key, Composite Primary Key ලෙස හඳුන්වයි.

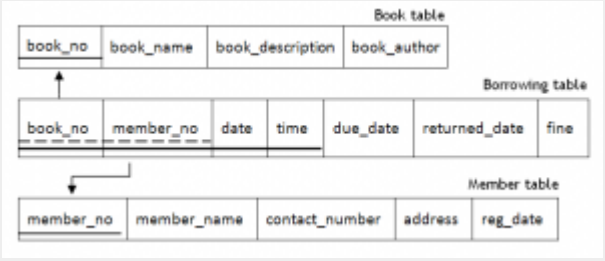
MySQL table එකක් තුලට Primary Key අන්තර්ගත කිරීම

පහත දක්වා ඇත්තේ පෙර ලිපියක සඳහන් කරන ලද studenttable එකට primary key එකක් ඇතුලත් කර ඇති ආකාරයයි.

```
create table student(              student_id varchar(10) not null,
student_name varchar(50) not null,              age int(2) default 5,
       primary key(student_id)     );
```

පහත දක්වා ඇත්තේ මෙම ලිපියේ මුලින් සඳහන් කරන ලද borrowing table එකට primary key එකක් ඇතුලත් කර ඇති ආකාරයයි.

```
      create table borrowing(           book_no varchar(10) not null,
      member_no varchar(10) not null,
      date date,          time time,          due_date date,
      returned_date date,         fine double(7,2) default '0.00',
      primary key(book_no,member_no,date,time)  );
```



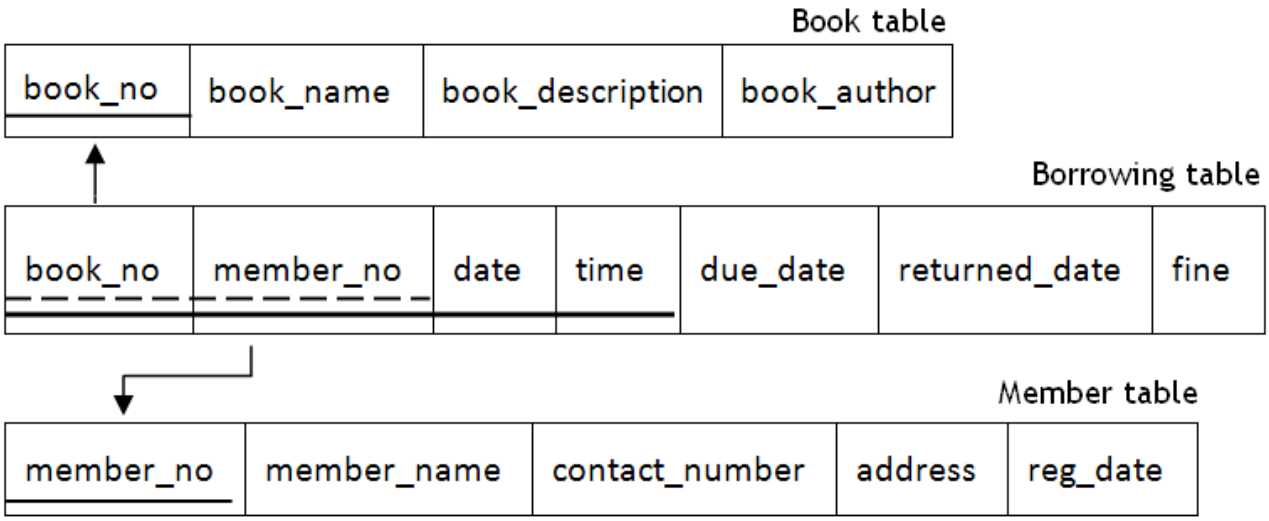
මෙහිදී primary key ලෙස වරහන් තුල ඇති field එක හෝ fields භාවිතා කිරීමට සලස්වා ඇත.

Composite Primary Key බොහෝ විට ඇත්තේ එම fields වෙනත් tables වල field හා සම්බන්ධ වන විටදීය.

Referencial Integrity

Tables වලට වෙනත් tables වල ඇති fields අන්තර්ගත කිරීමෙන් එම tables අදාල table එක සමඟ සම්බන්ධ කලහැක . උදාරණ වශයෙන් ඉහත පුස්ථකාලයේBorrowing table එක දැක්විය හැක එහි .Book_No සහ Member_No යනු වෙනත් tables වලින් ලැබෙන දත්ත සමඟ සම්බන්ධ වීමට භාවිතා කරන fields වේ.

එනම් Book_No යන field එක Book table එකෙන්ද, Member_No යන field එක Member table එකෙන්ද දත්ත ලබා ගනීඑහි අදහස වන්නේ ., Borrowing table එකට දත්ත පේලියක්)record එකක් ඇතුලත් කිරීමේදී එහි ඇති (Book_No එක දැනට Book table එකේද, Member_No එක දැනට Member table එකේද තිබිය යුතුය එසේ එක් .table එකක ඇති දත්ත refer කිරීමට, එම table එකේ Primary key එක වෙනත් table එකක ඇතුලත් කලවිට එය Foreign Key ලෙස හඳුන්වයි . එසේ දත්ත සපයනtable එක, Parent Table එක ලෙසත්, එම Primary Key එක භාවිතා කරමින් එම table එක සමඟ සම්බන්ධ වන table එක Child Table එක ලෙසත් හඳුන්වයි එනම් .Parent table එකේ Primary key එකක් හෝ කිහිපයක් Child table එකේදී Foreign keys ලෙස භාවිතා කලහැක.එය අවබෝධ කර ගැනීමට පහත රූප සටහන සලකා බලන්න .



(මෙහි යටින් ඇඳි කඩ ඉරි වලින් Foreign Keys ද තද ඉරි වලින් Primary Keys ද නිරූපණය කර ඇත(.

මෙම රූප සටහනට අනුව Foreign Keys ඇතුලත් කල පසුව Borrowing table එකට අදාල MySQL code එක පහත දැක්වේ.

```
      create table borrowing(           book_no varchar(10) not null,
      member_no varchar(10) not null,          date date,          time
time,         due_date date,              returned_date date,         fine
double(7,2) default '0.00',               primary
key(book_no,member_no,date,time),         foreign key(book_no) references
```

```
book(book_no),              foreign key(member_no) references book(member_no)
      );
```

පැහැදිලි කිරීම :

foreign key(book_no) references book(book_no) හි

`references book(book_no)` යන්නෙහි book යනු Parent table එක වන අතර වරහන් තුල ඇති book_no යනු එම Parent table එකේ refer වන field එක වේ.

`foreign key(book_no)` හි වරහන් තුල ඇති book_no යනු මෙම table එකේ Foreign Key ලෙස ක්‍රියා කරන field එකයි.

ඒ අනුව ඉහත රූප සටහනේ ඇති tables නිර්මාණය කිරීමට අදාල MySQL code එක පහත දැක්වේ.

```
      drop database if exists myDB;      create database myDB;       use myDB;
      create table member(               member_no varchar(10) not null,
      member_name varchar(50) not null,         contact_number varchar(11),
            address varchar(100),               reg_date date,
      primary key(member_no)      );     create table book(          book_no
varchar(10) not null,              book_name varchar(50) not null,
      book_description varchar(100),            book_author varchar(50),
      primary key(book_no)        );     create table borrowing(
      book_no varchar(10) not null,             member_no varchar(10) not
null,         date date,           time time,           due_date date,
      returned_date date,         fine double(7,2) default '0.00',
      primary key(book_no,member_no,date,time),              foreign
key(book_no) references book(book_no),           foreign key(member_no)
references book(member_no)  );
```

සැලකිය යුතුයි :

Foreign key යෙදීමේදී relationship එක සෑදෙන fields එකම data type එකෙන් තිබිය යුතුය උදාහරණ ලෙස . ඉහතmember_no යන්න member table එකේ varchar(10) ලෙස ඇති නිසා එය refer කරන සියළුම child tables වල member_no යන field එක varchar(10) ලෙසම තිබිය යුතුය.

මෙසේ Referencial Integrity පවතින විටදී, parent table එකක් හෝ කිහිපයක් update/delete වන විටදී එය child tables වලට බලපායි: උදා .එය නිවැරදිව කළමනාකරණය කලයුතුය.

ඉහත උදාහරණයේ member table එකේ member කෙනෙක්ගේ member_no එකක් වෙනස් කිරීමට අවශ්‍ය යැයි සිතන්න නමුත් දැනටමත් එම.member_no එක borrowing table එකේ records වල ඇතුලත් වී තිබිය හැක එවිට එම .member_no එක member table එකේ පමණක් වෙනස් කල නොහැක මක්නිසාද යත් එවිට . Referencial Integrity යන සංකල්පය බිඳ වැටෙන නිසාය එසේ වෙනස් කිරීමට උත්සහ කලවිටද).error එකක් ලැබේ(.

එ සඳහා නිවැරදි ලෙස tables define කල යුතුයඑනම් ., parent table එකේ record එකක් update/delete කලවිට එය child tables වලට කෙසේ බලපාන්නේද යන්න define කලයුතුය ඒ සඳහා .child tables මෙසේ define කලහැක .parent table එක වෙනස් කිරීමේදී

1. child table එකේ foreign key ඊට අනුව වෙනස් වේ.

   උදා :

   ඉහත member table එකේ member_no එක වෙනස් කලවිට borrowing table එකේ අදාල member_no එක යෙදී ඇති instances සියල්ලම අනුරූපව වෙනස් කරයි.

1. child table එකේ foreign key වශයෙන් එම records භාවිතා වී ඇත්නම් වෙනස් කිරීමට අවස්ථාව ලබා නොදීම.

උදා :

ඉහත member table එකේ member_no එක වෙනස් කලවිට borrowing table එකේ අදාල member_no එක දැනටමත් යෙදී ඇති ඇත්නම් parent table එක වෙනස් කිරීමට අවස්ථාව ලබා නොදේ ඒසේ නැත්නම් වෙනස්. .කලහැක

1. child table එකේ foreign key සඳහා වෙනත් value එකක් ලබාදිය හැක.

   උදා :

   ඉහත member table එකේ member_no එක වෙනස් කලවිට borrowing table එකේ අදාල member_no එක යෙදී ඇති instances සියල්ලම දැනට සිටින වෙනත් member කෙනෙක්ගේ member_no එකක් හෝ වෙනත් දෙන ලද value එකක් බවට වෙනස් කරයි.

මෙසේ parent table එකේ records update කිරීමේදී සහ delete කිරීමේදී සිදු විය යුතු දේ define කිරීමට foreign key එක define කරන තැනදීම on update සහ on delete ලෙස අදාල option එක සඳහන් කලහැක එසේ භාවිතා කලහැකි . options ඉහත පැහැදිලි කරන ලදී ඒවා.MySQL වලදී මෙසේ භාවිතා වේ.

1. `cascade`

   ඉහත 1 හි සඳහන් කරන ලදී.

1. `restrict` හෝ `no action`

   ඉහත 2 හි සඳහන් කරන ලදී.

1. `set null` හෝ `set default`
   ඉහත 3 හි සඳහන් කල අයුරින් parent table එක වෙනස් වීමේදී child table එකේ foreign key එකට අදාල field එක `null` හෝ child table එකේ field definition එකේ `default` value එකක් සඳහන් කරන ලද නම් එය assign වේ.

	update	**delete**
cascade	on update cascade	on delete cascade
restrict	on update restrict	on delete restrict
no action	on update no action	on delete no action
set null	on update set null	on delete set null
set default	on update set default	on delete set default

ඉහත පැහැදිලි කිරීමට අනුව borrowing table එක වෙනස් කර ඇති අයුරු සලකා බලමු.

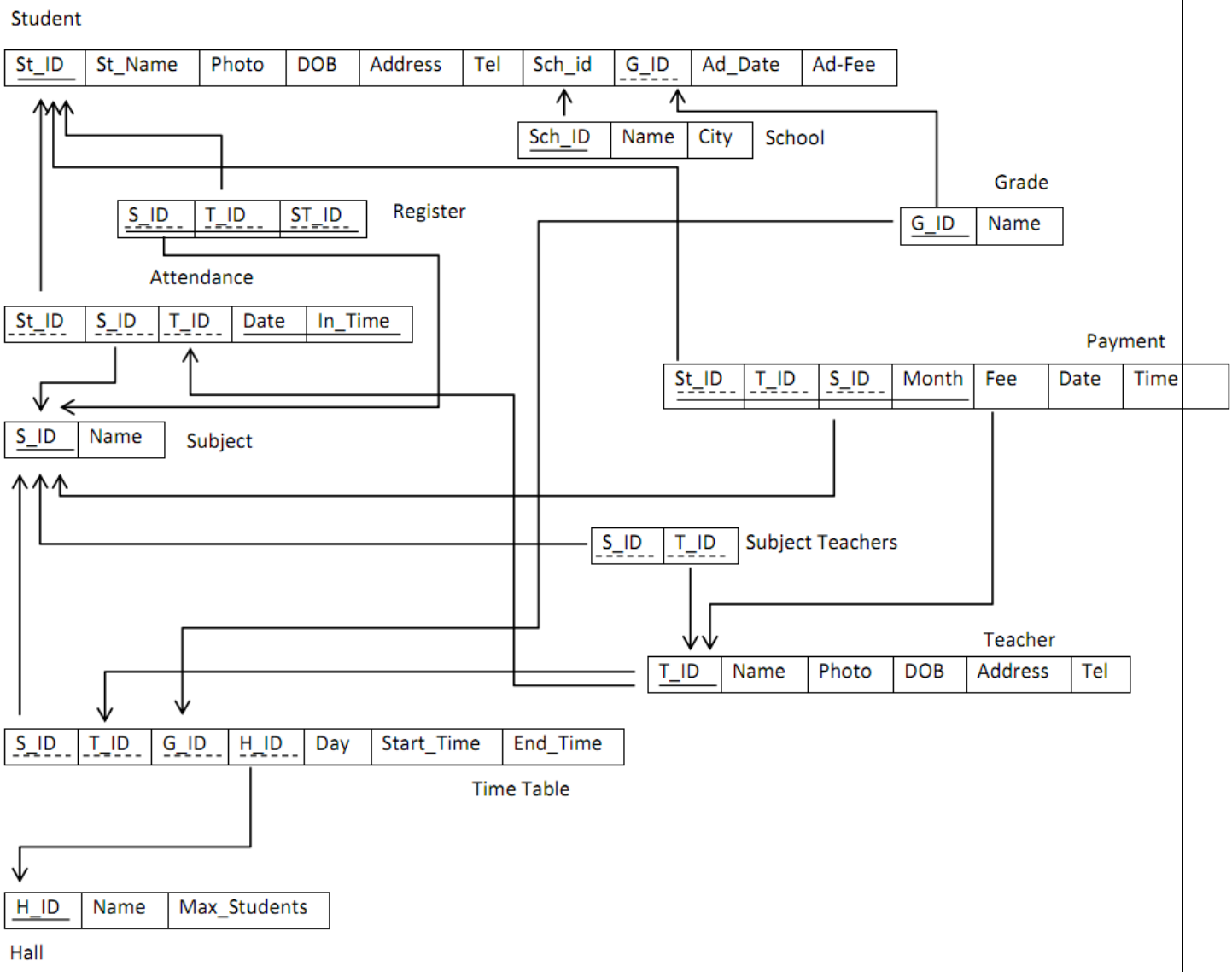
```
       create table borrowing(            book_no varchar(10) not null,
       member_no varchar(10) not null,          date date,          time
time,         due_date date,               returned_date date,        fine
double(7,2) default '0.00',               primary
key(book_no,member_no,date,time),         foreign key(book_no) references
book(book_no) on update cascade on delete cascade,            foreign
```

```
key(member_no) references member(member_no) on update cascade on delete
restrict      );
```

මෙහිදී තමාගේ අවශ්‍යතාවයට අනුව අදාල options එක භාවිතා කිරීමට වල බලා ගන්න

MySQL – Database Example

පහත දක්වා ඇත්තේ අධ්‍යාපන ආයතනයක සිසුන්ගේ සහ ආචාර්යවරුන්ගේ දත්ත කළමණාකරණය සඳහා නිර්මාණය කිරීමට සැලසුම් කරන ලද මෘදුකාංගයකට අදාල relations diagram එකකි.



එම රූප සටහනට අනුව නිර්මාණය කර ඇති database එකක් මෙසේ දක්වා ඇත.

```
DROP DATABASE IF EXISTS EDINS; CREATE DATABASE EDINS; USE EDINS;

 DROP TABLE IF EXISTS grade; CREATE TABLE grade (   G_ID varchar(10) NOT NULL,
NAME varchar(20) default NULL,   PRIMARY KEY  (G_ID) ) ;

 DROP TABLE IF EXISTS hall; CREATE TABLE hall (   H_ID varchar(10) NOT NULL,
NAME varchar(20) default NULL,   MAX_ST int(3) default NULL,   PRIMARY KEY
(H_ID) );
```

```
 DROP TABLE IF EXISTS teacher; CREATE TABLE teacher (   T_ID varchar(10) NOT
NULL,   NAME varchar(50) NOT NULL,   PHOTO longblob,   GENDER varchar(6)
default NULL,   DOB date default NULL,   Address varchar(100) default NULL,
Balance double default '0',   PRIMARY KEY  (T_ID) );

 DROP TABLE IF EXISTS school; CREATE TABLE school (  sch_id varchar(10) NOT
NULL,  name varchar(100)  NOT NULL,

 city varchar(45) default '',

 PRIMARY KEY (sch_id)

 );
```

DROP TABLE IF EXISTS student; CREATE TABLE student (ST_ID varchar(10) NOT NULL, NAME varchar(50) NOT NULL, PHOTO longblob, GENDER varchar(6) default ”, DOB date default NULL, ADDRESS varchar(60) default ”, SCH_ID varchar(30) default ”, G_ID varchar(10), AD_DATE date default NULL, AD_FEE double(10,2) default ‘0.00’, PRIMARY KEY (ST_ID), CONSTRAINT FK_student_2 FOREIGN KEY (SCH_ID) REFERENCES school (sch_id) ON UPDATE CASCADE, CONSTRAINT FK_student_1 FOREIGN KEY (G_ID) REFERENCES grade (G_ID) ON UPDATE CASCADE); DROP TABLE IF EXISTS subject; CREATE TABLE subject (S_ID varchar(10) NOT NULL, NAME varchar0) default NULL, PRIMARY KEY (S_ID)); DROP TABLE IF EXISTS st_tel; CREATE TABLE st_tel (St_id varchar(10) NOT NULL, type varchar(10) NOT NULL, number varchar(10) NOT NULL, PRIMARY KEY (St_id,type,number), CONSTRAINT FK_st_tel_1 FOREIGN KEY (St_id) REFERENCES student (ST_ID) ON DELETE CASCADE ON UPDATE CASCADE); DROP TABLE IF EXISTS subject_teachers; CREATE TABLE subject_teachers (T_ID varchar(10) NOT NULL, S_ID varchar(10) NOT NULL, PRIMARY KEY (T_ID,S_ID), CONSTRAINT FK_TEACHER_SUBJECTS_1 FOREIGN KEY (T_ID) REFERENCES teacher (T_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_TEACHER_SUBJECTS_2 FOREIGN KEY (S_ID) REFERENCES subject (S_ID) ON DELETE CASCADE ON UPDATE CASCADE); DROP TABLE IF EXISTS teacher_tel; CREATE TABLE teacher_tel (T_ID varchar(10) NOT NULL, type varchar(10) NOT NULL, number varchar(10) NOT NULL, PRIMARY KEY (T_ID,type,number), CONSTRAINT teacher_tel_ibfk_1 FOREIGN KEY (T_ID) REFERENCES teacher (T_ID) ON DELETE CASCADE ON UPDATE CASCADE); DROP TABLE IF EXISTS attendance; CREATE TABLE attendance (ST_ID varchar(10) NOT NULL, S_ID varchar(10) NOT NULL, T_ID varchar(10) NOT NULL, DATE date default ‘0000-00-00′, TIME_IN time default ’00:00:00’, PRIMARY KEY (ST_ID,S_ID,T_ID,DATE,TIME_IN), CONSTRAINT FK_attendance_1 FOREIGN KEY (ST_ID) REFERENCES student (ST_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_attendance_2 FOREIGN KEY (S_ID) REFERENCES subject (S_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_attendance_3 FOREIGN KEY (T_ID) REFERENCES teacher (T_ID) ON DELETE CASCADE ON UPDATE CASCADE); DROP TABLE IF EXISTS grade_subjects; CREATE TABLE grade_subjects (g_id varchar(10) NOT NULL, s_id varchar(10) NOT NULL, PRIMARY KEY (g_id,s_id), CONSTRAINT FK_grade_subjects_1 FOREIGN KEY (g_id) REFERENCES grade (G_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_grade_subjects_2 FOREIGN KEY (s_id) REFERENCES subject (S_ID) ON DELETE CASCADE ON UPDATE CASCADE); DROP TABLE IF EXISTS payment; CREATE TABLE payment (st_id varchar(10) NOT NULL, t_Id varchar(10) NOT NULL, s_id varchar(10) NOT NULL, month varchar(20) NOT NULL, date date default ‘0000-00-00′, time time default ’00:00:00′, PRIMARY KEY (st_id,t_Id,s_id,month,date), CONSTRAINT FK_payment_1 FOREIGN KEY (st_id) REFERENCES student (ST_ID), CONSTRAINT FK_payment_2 FOREIGN KEY (t_Id) REFERENCES teacher (T_ID), CONSTRAINT FK_payment_3 FOREIGN KEY (s_id) REFERENCES subject (S_ID)); DROP TABLE IF EXISTS register; CREATE TABLE register (S_ID varchar(10) NOT NULL, ST_ID varchar(10) NOT NULL, T_id varchar(10) NOT NULL, PRIMARY KEY (S_ID,ST_ID,T_id), CONSTRAINT FK_register_1 FOREIGN KEY (ST_ID) REFERENCES student (ST_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_register_2 FOREIGN KEY (S_ID) REFERENCES subject (S_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_register_3 FOREIGN KEY (T_id) REFERENCES

teacher (T_ID) ON DELETE CASCADE ON UPDATE CASCADE); DROP TABLE IF EXISTS timetable; CREATE TABLE timetable (S_ID varchar(10) NOT NULL, T_ID varchar(10) NOT NULL, G_ID varchar(10) NOT NULL, H_ID varchar(10) NOT NULL, DAY varchar(15) NOT NULL, START time default '00:00:00', END time default '00:00:00', PRIMARY KEY (S_ID,T_ID,G_ID,H_ID,DAY,START,END), CONSTRAINT FK_TIMETABLE_1 FOREIGN KEY (S_ID) REFERENCES subject (S_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_TIMETABLE_2 FOREIGN KEY (T_ID) REFERENCES teacher (T_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_timetable_3 FOREIGN KEY (G_ID) REFERENCES grade (G_ID) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT FK_timetable_4 FOREIGN KEY (H_ID) REFERENCES hall (H_ID) ON DELETE CASCADE ON UPDATE CASCADE);

NOTE :

Foreign Key නිර්මාණය කිරීමේදී එයට නමක් ලබා දිය හැක එවිට පසුව අවශ්‍ය වූ විටෙක එම නම භාවිතා . මේවා .කරමින් එය වෙනස් කිරීමට හැකියdatabase එකක constraints ලෙස හඳුන්වන අතර `Constraint` වශයෙන් එය define කලහැක.ÂÂÂ

ඉහත MySQL Script එකේ අවසාන table එකේ table definition එක සැලකූ විට, එහිදී එය

```
ÂÂÂ  CONSTRAINT FK_timetable_3 FOREIGN KEY (G_ID) REFERENCES grade
(G_ID) ON DELETE CASCADE ON UPDATE CASCADE
```

ලෙස සඳහන් කර ඇතමෙහි . `FK_timetable_3` යනු එම foreign key එන ඇතුලත් constraint එකේ නමයි.