2015.10.06.

- ව්‍යුහගත විමසුම් භාෂාව (Structured Queries Language - SQL)

- SQL යනු

දත්ත සමුදාය නිර්මාණය, වැඩ නිර්මාණය, දත්ත ඇතුළත් කිරීම, අවශ්‍ය දත්ත නැවත ලබාගැනීම, එකතුකිරීම හා යාවත්කාලීන කිරීම ආදී වූ කාර්යයන් සඳහා භාවිත කළ හැකි පරිගණක භාෂාවක් මෙම භාෂාව අකුරු සංවේදී (Case sensitive) නොවේ.

SQL ප්‍රධාන ආකාර 02 කි.

(i) Data Definition Language - DDL
(ii) Data Manipulation Language - DML

(i) Data Definition Language (DDL)

මෙය භාවිතයෙන් දත්ත වැඩ නිර්මාණය හා වගුවල ව්‍යුහයන්ගේ සිදුවන වෙනස්කම් සිදුකරයි. මෙහිදී පහත විධාන භාවිතා කරයි.

(01) CREATE
(02) ALTER
(03) DROP

(01) CREATE

- CREATE DATABASE

මෙමගින් දත්ත සමුදාය නිර්මාණය කරයි.

Syntax.

CREATE DATABASE database_name/databaseName

උදා: mySchool ලෙස දත්ත සමුදායක් නිර්මාණය කිරීම.

CREATE DATABASE mySchool/my_school

Scanned by CamScanner

- CREATE TABLE

දත්ත කට්ටලය තුළ අඩංගු විය යුතු දත්ත වගු නිර්මාණය සඳහා මෙය යොදාගනී. මෙහිදී වගු තුළ අඩංගු කරනු ලබන විය යුතු උපලක්ෂණයන්ගේ දත්ත වර්ග (Data types) ඇතුළත් කළ යුතු වේ.

Syntax

```
CREATE TABLE table_name
(columnName1 dataType,
 columnName2 dataType,
 columnName3 dataType,
 ----------
 ----------
 primaryKey(columnName) )
```

උදා: පහත උපලක්ෂණ ඇතුළත් වන ලෙස student වගුව නිර්මාණය කරමු.

stuId - char(5)
address - varchar(20)
marks - int

```
CREATE TABLE student
(stuId char(5),
(address varchar(20),
(marks int,
primaryKey(stuId) )
```

Scanned by CamScanner

(ii) ALTER

- ALTER TABLE

මෙමගින් වගුවක ව්‍යුහයෙහි යම් යම් වෙනස්කම් සිදුකරයි.

එකම වගුවකට අලුතින් උපලක්ෂණ එක් කිරීම උපලක්ෂණ මකා දැමීම හා දත්ත වර්ගය වෙනස් කිරීම සිදුකරයි.

- උපලක්ෂණයක් වගුවකට එක් කිරීම.

Syntax:

```
ALTER TABLE table_name
ADD column_name dataType
```

උදා: student වගුවට stu_name ලෙස උපලක්ෂණයක් එක් කිරීම.

```
ALTER TABLE student
ADD stu_name cha(10)
```

- උපලක්ෂණයක් වගුවකින් මකා දැමීම.

Syntax:

```
ALTER TABLE table_name
DROP COLUMN column_name
```

උදා: Student වගුවෙන් marks යන උපලක්ෂණය මකා දැමීම.

```
ALTER TABLE student
DROP COLUMN marks.
```

- දැනටමත් ඇති උපලක්ෂණයක දත්ත වර්ගය වෙනස් කිරීම.

Syntax:

```
ALTER TABLE table_name
ALTER COLUMN column_name dataType.
```

Scanned by CamScanner

උදා: stu_name වලට අති උපරිමයෙන් දත්ත ප්‍රමාණය varchar(25) වලට වෙනස් කිරීම

```
ALTER TABLE student
ALTER COLUMN stu_name varchar(25)
```

(03.) DROP

- DROP DATABASE

  මෙමගින් දත්ත සමුදාය මකා දමයි.

Syntax:

```
DROP DATABASE database_name
```

උදා: myschool දත්ත සමුදාය මකා දැමීම

```
DROP DATABASE myschool
```

Scanned by CamScanner

2015.10.08.

- DROP TABLE

මෙමගින් දත්ත වගුවක් මකාදමයි.

Syntax:

```
DROP TABLE table_name
```

උදා: දත්ත සමුදායෙන් student යන දත්ත වගුව මකා දැමීම.

```
DROP TABLE student.
```

(ii) Data Manipulation Language (DML)

මෙහිදී දත්ත සමුදායක ඇති වගුවක, අවංග කර දත්ත හෙවත් දත්ත යාවත්කාලීන කිරීම, පරිශීලක අවශ්‍යතා මත තාවත ලබාගැනීම ආදිය සිදුකරයි. ඒ සඳහා තොත පුකාශන යොදාගනී.

(01) SELECT - දත්ත වැඩ කාර්යයෙන් දත්ත තාවත ලබාගැනීම (Retrive)

(02) INSERT - වගු තුළට දත්ත පිටතින් ඇතුළු කරීම.

(03) UPDATE - දත්ත යාවත්කාලීන කිරීම.

(04) DELETE - වගු තුළ ඇති දත්ත මකා දැමීම.

Student

StuNo	fname	lname	tpNo	addresse
S_1	Amal	Perera	0332252322	80, gampaha rd, Kalu
S_2	Nimal	Silva	033 2211345	15, royal rd, colomb
S_3	Kamal	Perera	033 2222123	23, rose rd, Kalutar
S_4	Saman	Soyza	033 2255371	52, lotus rd, colomb
S_5	Amal	Dias	033 2232321	09/2, Gampaha
S_6	Vimal	Silva	0332295340	93, colombo rd, gampa

Scanned by CamScanner

sports

rank	stuNo	house	marks
1	S_4	Mahamaya	53
2	S_1	Vishaka	42
3	S_3	Sangamitta	31
4	S_6	Gothami	20
5	S_5	Mahamaya	19
6	S_2	Vishaka	07

(v) SELECT

- SELECT statement

මෙමගින් එක් වගුවකින් හෝ වගු කිහිපයකින් අවශ්‍ය දත්ත සමූහයක් නැවත ලබාගැනීම සිදුවේ.

(i) එක් වගුවක ඇති සියලුම තීරුවලට දත්ත ලබාගැනීම.

Syntax:

All

SELECT * FROM table_name

උදා: student වගුවෙහි සියලුම තීරුවලට ඇති දත්ත නැවත ලබා ගැනීම.

SELECT * FROM student

(ii) වගු කිහිපයක් තුළ ඇති සියලුම දත්ත එකවර ලබාගැනීම.

Syntax:

SELECT * FROM table_name1, table_name2....

උදා: student සහ sports යන වගුවල ඇති සියලුම දත්ත ලබාගැනීම.

SELECT * FROM student, sports

Scanned by CamScanner

(iii) එක් වගුවකින් අවශ්‍ය (උප ලක්ෂණයක්) තීරු පමණක් ලබා ගැනීම

Syntax:

```
SELECT column1, column2, ....
FROM table_name
```

උදා: student වගුවෙන් සියලුම ළමුන්ගේ පළමු නම සහ ඉරිතාලා අංකය ලබා ගැනීම.

```
SELECT fname, tpNo
FROM student
```

2015.10.26

(iv) වැඩි කිහිපයකින් අවශ්‍ය තීරු පමණක් ලබා ගැනීම.

Syntax:

```
SELECT table_name1.column, table_name2.column.....
FROM table_name1, table_name2....
```

උදා: සිසුන්ගේ පළමු නම fname, lname සහ අයිති කිරීම ලබා ගැනීම.

```
SELECT student.fname, student.lname,
                                sports.house
FROM student, sports
```

Scanned by CamScanner

- WHERE clause

යම්කිසි කොන්දේසියක් අනුව දත්ත වගුවකින් හෝ කිහිපයකින් අවශ්‍ය උපලක්ෂණ පමණක් ලබා ගැනීම වෙනුවෙන් භාවිත කරයි.

(v.) එක් වගුවකින් දෙනු ලබන කොන්දේසියක් අනුව අවශ්‍ය උපලක්ෂණ පමණක් ලබා ගැනීම.

Syntax:

```
SELECT * FROM table_name
WHERE column_name = value
```

condition

උදා: student වගුවෙන් lname Perera වන සියලුම උපලක්ෂණ ලබාගැනීම.

```
SELECT * FROM student
WHERE lname = "Perera"
```

2015.10.28.

යම් කොන්දේසියක ලබා දීමේදී ඒවා සහන කිරීමේ AND සහ OR යන තාර්කික කර්ම වලින් සම්බන්ධ කර අවශ්‍ය උපලක්ෂණ පමණක් ලබාගත හැකිවේ.

(vi.) වගු කිහිපයකින් දෙනු ලබන කොන්දේසි සියල්ලම සහන වන උපලක්ෂණ පමණක් ලබාගැනීම.

Syntax:

```
SELECT table1.column, table2.column_name
FROM table_name1, table_name2
WHERE column_name1 = value1 AND column_name2 = value2
                                        AND ....
```

උදා: මහාමාය කණ්ඩායමේ සිටින fname Amal වන සියලුම තොරතුරු ලබාගැනීම

```
SELECT student.fname, sports.house
FROM student, sports
WHERE student.fname = "Amal" AND sports.house = "Mahamaya"
```

Scanned by CamScanner

(vii) දෙකු ලබන කොන්දේසි කිහිපයක් අතරින් අවම වශයෙන් එකක් හෝ සත්‍ය වන විට උපලක්ෂියාන ලබාගැනීම.

Syntax:

```
SELECT table1.column_name , table2.column_name
FROM table_name1 , table_name2
WHERE column_name1 = value1 OR column_name2 = value2
```

උදා- (i) student වගුවෙහි අති උපලක්ෂියාන අතරින් fname Amal හෝ lname Perera ලෙස සඳහන්ව අති උපලක්ෂියන සමස්ත ලබාගැනීම.

```
SELECT*
FROM student
WHERE fname="Amal" OR lname="Perera"
```

output:

StuNo	fname	lname	tpNo	address
S_1	Amal	Perera	0332252322	80, gamaba rd, Yakkala
S_3	Kamal	Perera	0332222123	23, rose rd, Kalutara.
S_5	Amal	Dias	0332232321	29/2, Gampaha.

(ii) කීඩායය මහාමායා හෝ විශාකා වන සියලුම සිසුන්ගේ fname සහ tpNo ලබාගැනීම.

```
SELECT student.fname, student.tpNo
FROM student, sports
WHERE sports.house="Mahamaya" OR sports.house="Vishaka"
```

Scanned by CamScanner

WAMP server download කරන්න

2015.11.02.

- ORDER BY

දත්ත ලකාගනිමින් තීරුවක් හෝ කිහිපයක් ආරෝහණ හෝ අවරෝහණ ආකාරයට සකස් කිරීම මෙය සාධිත කරයි.

(viii)

Syntax:

Default order Ascending (ආරෝහණ)

```
SELECT *
FROM table_name
ORDER BY column_name DESC/ASC , column_name ASC
```

Descending Ascending

- DESC හෝ ASC ලෙස දක්වා කොමාව විට සෑම විටම දත්ත ආරෝහණ (Ascending) ආකාරයට ලකාදෙයි.

උදා: Student වගුවෙහි ඇති සිසුන්ගේ fname සහ lname පළමු නම අනුව ආකාරවලට පිළිවෙළට ලකාගනීම.

```
SELECT fname, lname
FROM student
ORDER BY fname ASC , lname DESC , .....
```

Scanned by CamScanner

(ii) INSERT

- INSERT INTO

වගුවකට දත්ත ඇතුළු කිරීම සඳහා මෙය භාවිත කරයි. මෙමගින් උපලක්ෂණයක් හෝ එක් උපලක්ෂණයකට දත්ත කිහිපයකට දත්ත ඇතුළත් කළ හැකි වේ.

Syntax:

~~record එකේ කිහිපයක් Data දැමෙන්න නම්~~

```
I  INSERT INTO table_name
   VALUES (value1, value2, ....)
```

```
II INSERT INTO table_name (column1, column2, ...)
   VALUES (value1, value2, ....)
```

උදා: (i) student වගුවේ ප්‍රකෘතිය අංකය 0332292822 වන Saman Silva ගේ තොරතුරු S_7 යන අංකය යටතේ ඇතුළු කිරීම.

```
INSERT INTO student (stuNo, fname, lname, tpNo)
VALUES ("S7", "Saman", "Silva", "0332292822")
```

(ii) S_7 සිසුවා ලකුණු ලබා ගොගත් අතර බහුමේ පිටිකය 7 ලෙස සටහන් විය. මෙම සිසුවා ගෝතමී නිවාසයට අයත් වන්නේ නම් එම දත්ත sports වගුවේ ඇතුළත් කරන්න.

```
INSERT INTO sports
VALUES ( 7 , "S7" , "Gothami" , "0" )
```

(iii) student වගුවේ පහත තොරතුරු ඇතුළත් කිරීම.

S_8, Nayana Kumari, 0335234322, ~~70~~, Gampaha.

S_9, Meena Kumari, 0335522321 1, ~~80~~, Yakkala.

Scanned by CamScanner

```
INSERT INTO student
VALUES ("S8", "Nayana", "kumari", "0335234322", "Gampaha")
AND ("S9", "Meena", "Kumari", "0335522321", "Yakkala")
```

(03) UPDATE statement.

මෙම ප්‍රකාශනය මගින් දත්ත කුට්ටියක් තුළ ඇති උපලක්ෂියක් යාවත්කාලීන කිරීම සිදුවේ.

- මෙහිදී WHERE පදය භාවිතයෙන් යාවත්කාලීන කළ යුතු උපලක්ෂියක් සොයාදීමට අවශ්‍ය වුනු නම්. එසේ නොවන විට එම සව අගය මගින් එම අගයට අදාළ උපලක්ෂියෙහි ඇති සියලුම දත්ත ප්‍රතිස්ථාපනය (replace) වේ.

Syntax:

```
UPDATE table_name
SET column1=value1, column2=value2, ....
WHERE condition
```

උදා:- student වගුවෙහි ඇති S_7 උපලක්ෂියකොහි address සඳහා gampaha යන්න ඇතුළත් කිරීම.

```
UPDATE student
SET address="Gampaha"
WHERE stuNo="S7"
```

(ii) Amal Perera ගේ දුරකථන අංකය 0332292822 ලෙසත් ලිපිනය 90, colombo ලෙසත් වෙනස් කිරීම.

```
UPDATE student
SET tpNo="0332292822", address="90, Colombo"
WHERE fname="Amal" AND lname="Perera"
```

Scanned by CamScanner

(04) DELETE statement

වගු තුළ ඇති දත්ත මකාදැමීම සඳහා මෙය භාවිත කරයි.

- මෙහිදී ද WHERE වාක්‍යයක් අදාළ උපලක්ෂණය හඳුන්වීම අනිවාර්යය වේ. එසේ නොමැති විට වගුව තුළ ඇති සියළුම දත්ත මෙමගින් මකාදමයි.

Syntax:

```
DELETE FROM table_name
WHERE condition
```

උදා: Student වගුවෙහි ඇති S_5 සිසුවාගේ දත්ත මකා දැමීම

```
DELETE FROM student
WHERE stuNo = "S5"
```

Scanned by CamScanner

2015. 11. 03.

2014 A/L

03)

house

houseID	name
HS1	Gamunu
HS2	Tissa
HS3	Wijaya
HS4	Parakum

student

studentID	name	grade	houseID
STU001	Ranjith	13	HS1
STU002	Gopy	12	HS1
STU003	Vipula	12	HS2
STU004	Hakeem	11	HS3

ඉහතින් දී තිබෙන විදුහලක සිසුන් හා වැටක් තුළත්වන නිවාස තුළින් වන වැඩ කිහිපයක් සඳහා කාර්යන් සඳහා SQL කුමක් ලියන්න

(i) student වගුවට STU005 ලෙස නිමේශ් කම තුළින් කරන්න

```
INSERT INTO student (studentID, name)
VALUES ("STU005", "Amith")
```

(ii) student වගුවෙහි රංජිත්ගේ නිවාසය වැරදියට සටහන්ව ඇති නිසා එය HS2 ලෙස වෙනස් කරන්න.

```
UPDATE student
SET houseID="HS2"
WHERE name="Ranjith"
```

(iii) එක් එක් නිවාසය විවිධ ක්‍රීඩාවක් සඳහා ලකුණු ලබාදී කිරීම සඳහා marks ලෙස දත්ත වගුවක් නිර්මාණය කරන්න. එහි houseID සහ marks ලෙස උපලක්ෂණ 02 ක් තබමින් දත්ත ප්‍රභේදයන් පිළිවෙලින් char(4) සහ char(6) වේ.

```
CREATE TABLE marks
(houseID char(4),
 marks char(6),
 sports varchar(20),
 primaryKey(houseID, sports),
 foreignkey(houseID REFERENCES house(houseID) )
```

(iv) student වගුවෙන් සියලුම 13 ලබන සිසුන් ඉවත් කරන්න

```
DELETE FROM student
WHERE grade="13"
```

Scanned by CamScanner

(v) house වගුවෙහි ඇති දත්ත නිසි පරිදි අකාරාදී පිළිවෙලට ලැයිස්තු කර ගන්න

```
SELECT *
FROM house
ORDER BY name ASC
```

2015.11.05

SQL තුළ භාවිතා වන සීමාකරණ (SQL constraints)

- NOT NULL - යම් තීරුවකට මෙම සීමාකරණය ඇතුළත් කර ඇතිවිට යම් (රේකෝඩයක) උපලක්ෂණයක් එම ඇතුළු තීරුවේ අගයක් ඇතුළත් කළ යුතුම වේ.

- UNIQUE - මෙහිදී යම් තීරුවකට ඇතුළත් කරනු ලබන දත්ත වල අනන්‍යතාවය තහවුරු කරයි. එනම් මෙහිදී යම් අගයක් දැනු තිබීම වළක්වයි.

- PRIMARY KEY - යම් වගුවක ඇති උපලක්ෂණයක්හි අනන්‍යතාවය මෙහිදී තහවුරු කරයි. මෙම සීමාකරණය තුළින් NOT NULL සහ UNIQUE යන සීමාකරණ දෙකම ඉටුකිරීමක් තාවත කරයි. මෙම සීමාකරණය එක් තීරුවක හෝ තීරු සංයෝජනයක් සඳහා යොදනු ලැබේ.

- FOREIGN KEY - මෙම සීමාකරණය මගින් එක් වගුවක ඇති දත්ත සමඟ සම්බන්ධතා වෙනත් වගුවක දත්ත අතර ඇති සම්බන්ධතාවය (ක්‍රියාත්මක සංගතිකරණයට-Referential integrity) තහවුරු කරයි.

Scanned by CamScanner

- NOT NULL

Syntax:-

```
CREATE TABLE table_name
(column_name1 dataType NOT NULL,
 column_name2 dataType,
 column_name3 dataType NOT NULL)
```

Ex: පහත ක්ෂේත්‍ර ඇතුළත් වන පරිදි person වගුව නිර්මාණය කරන්න. මෙහි pId සහ pTp ක්ෂේත්‍ර සඳහා අනිවාර්යයෙන්ම අගයක් ඇතුළත් කළ හැකි ලෙස වගුව සවිකිරීම ඉල්ලයි.

pId - char(5)
fname - varchar(15)
lname - varchar(15)
pTp - char(10)

```
CREATE TABLE person
(pId char(5) NOT NULL,
 fname varchar(15),
 lname varchar(15),
 pTp - char(10) NOT NULL)
```

- UNIQUE

Syntax:- (MySQL)

```
CREATE TABLE table_name
(Column_name1 dataType, NOT NULL,
 column_name2 dataType,
 column_name3 dataType,
 UNIQUE (column_name))
```

• SQL Server, Oracal වැනි කුමක් ඉලක්කයක් වලදී මෙය පහත පරිදි කුමක් column name ඉදිරියෙන් ලබාදිය හැකිය.

Scanned by CamScanner

```
CREATE TABLE table_name
(column_name1 dataType NOT NULL UNIQUE,
 column_name2 dataType ,
 column_name3 dataType NOT NULL)
```

2015.11.09

- PRIMARY KEY

Syntax:

```
CREATE TABLE table_name
(column_name_pk dataType NOT NULL,
 column_name1 dataType,
 column_name2 dataType,
 PRIMARY KEY (columnname_pk))
```

```
CREATE TABLE table_name
(Column_name_pk dataType NOT NULL PRIMARY KEY
 column_name1 dataType,
 column_name2 dataType )
```

- ප්‍රාථමික යතුර ලෙස භාවිතා යතුරක් තවමත් පිට එම සංයෝජනයම පැවත පවතී එකවර දක්වයි.

```
CREATE TABLE table_name
(column_name_pk1 dataType NOT NULL,
 column_name_pk2 dataType NOT NULL,
 column_name1 dataType,
 column_name2 dataType,
 PRIMARY KEY (column_name_pk1, column_name_pk2))
```

Scanned by CamScanner

උදා: studentID සහ courseID යන සංයෝජනය ප්‍රාථමික යතුර ලෙස කලකින් student_course වගුව සඳහා පහත සඳහන් ආකාරය සාදමි.

stuId - char(5)
cId - char(5)
Duration - int

```
CREATE TABLE student_course
(stuId char(5) NOT NULL,
 cId char(5) NOT NULL,
 Duration int,
 PRIMARY KEY (stuId, cId))
```

- FOREIGN KEY

Syntax:

```
CREATE TABLE table_name
(column_name_pk dataType NOT NULL,
 column_name1 dataType,
 column_name2 dataType,
 PRIMARY KEY (column_name_pk),
 FOREIGN KEY (column_name REFERENCES other_table
                                (column_name)))
```

උදා: student වගුවක stuId සහ course වගුවේ stuId යන කෙනෙකු කෙනෙක් මෙම වගු ඕ ක අතර සම්බන්ධතාවය ඇතිවන්නේ නම් එවිට student වගුව තුළ අංකනයක් යොදා දැක්වීම.

```
CREATE TABLE student
(stuId char(5) NOT NULL,
 fname char(5),
 l_name char(5),
 PRIMARY KEY (stuId),
 FOREIGN KEY (stuId REFERENCES course(stuId)))
```

Scanned by CamScanner