



# DATABASE MANAGEMENT SYSTEMS (IT2040)

## LECTURE 04 - SQL



# LECTURE CONTENT

- Introduction to SQL
- Data definition language
- Data manipulation language

# LECTURE CONTENT

- At the end of this lecture students should be able to
  - Write syntactically correct SQL statements to create and modify relations in a RDBMS
  - Write syntactically correct SQL statements to answer user defined queries in a RDBMS

# SQL

- SQL Initially called SEQUEL (for Structured English QUERy Language)
- It was developed for an experimental relational database system called System R
- A joint effort between ANSI (American National Standard Institute) and ISO (International Standards Organization) led to a standard version of SQL in 1986 (SQL1, SQL-86, etc.)
- Major revisions have been proposed and SQL2 (also called SQL-92) has subsequently been developed

# RELATIONAL MODEL VS. SQL

## ■ Terminology

Relational Model	SQL
Relation	Table
Attribute	Column
Tuple	Row

## SQL: REVIEW (CONTD.)

SQL is a comprehensive database language:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Facilities for security & authorization
- Facilities for transaction processing
- Facilities for embedding SQL in general purpose languages (Embedded SQL)
- ...

# DATA DEFINITION LANGUAGE (DDL)

- DDL is the subset of SQL that supports the creation, deletion and modifications for tables and views.
- Constraints can be defined on the tables

Constraint	Purpose
Not Null	Ensure that column doesn't have null values
Unique	Ensure that column doesn't have duplicate values
Primary key	Defines the primary key
Foreign key	Defines a foreign key
Default	Defines a default value for a column (When no values are given)
Check	Validates data in a column

# CREATING A TABLE - EXAMPLE

## CREATE TABLE STUDENT

```
(  
    studentId INTEGER PRIMARY KEY,  
    sName VARCHAR (30) NOT NULL,  
    nic CHAR(10) UNIQUE,  
    gpa FLOAT,  
    progId VARCHAR(10) DEFAULT 'IT',  
    CONSTRAINT student_prog_fk FOREIGN KEY (progId)  
        REFERENCES programs(id) ON DELETE SET DEFAULT ON  
        UPDATE CASCADE,  
    CONSTRAINT gpa_ck CHECK (gpa <= 4.0 )  
)
```



# MODIFICATIONS TO TABLES

- ALTER commands – to alter the definition of the object
  - Ex : Adding a new column to a table
    - **ALTER TABLE** student **ADD** age **INT**
  - Ex : Adding a new constraint to a column
    - **ALTER TABLE** student **ADD CONSTRAINT** chk\_age **CHECK** (age > 18)
  - Ex : removing a column from a table
    - **ALTER TABLE** student **DROP COLUMN** age
- DROP commands – for dropping objects
  - Ex: Deleting a table
    - **DROP TABLE** Employee

# DATA MANIPULATION LANGUAGE (DML)

- DML is the subset of SQL that allows users to write statements to insert, delete, modify and display rows.
  - Inserting a row
    - **INSERT INTO** student **VALUES** (1000, 'Amal', '123456789V', 3.2, 'BM')
    - **INSERT INTO** student(studentId, sName, nic) **VALUES** (1001, 'Nimali', '234567890V')

StudentID	SName	nic	gpa	progId
1000	Amal	123456789V	3.2	BM
1001	Nimali	234567890V	Null	IT

# DATA MANIPULATION LANGUAGE (DML)(CONTD.)

- Deleting a row
  - **DELETE** student **WHERE** studentId=1000
- Updating a row
  - **UPDATE** student  
**SET** gpa=2.8  
**WHERE** studentId=1001

# SELECT CLAUSE

- Select clause in SQL is the basic statement for retrieving information from a database

- Basic form

**SELECT** <attributes>

**FROM** <one or more relations>

**WHERE** <conditions>

- Ex : display ids of all students whose gpa is above 3.0
  - Select StudentId from student where gpa > 3.0

# CLAUSES AND OPERATORS USED WITH SELECT

- LIKE operator
- IS [NOT] NULL operator
- DISTINCT operators
- BETWEEN operator
- ORDER BY clause
- Joins (inner & outer)
- Nested query (IN/SOME/ANY,ALL), [NOT] EXISTS
- Aggregate functions
- GROUP BY – HAVING clauses

# LIKE OPERATOR

- Used for matching patterns
- Syntax : <string> LIKE <pattern>
  - <pattern> may contain two special symbols:
    - % = any sequence of characters
    - \_ = any single character
- Ex : Find students whose name starts with a 'A'
  - Select Name From student where Name Like 'A%'

**Student**

StudentID	Name	gpa	progId
1000	Amal	3.2	BM
1001	Nimali	Null	IT
1002	Aruni	3.0	SE
1003	Surani	2.5	IT

Name
Amal
Aruni

# IS [NOT] NULL OPERATOR

- IS NULL :Used to check whether attribute value is null
- Ex : Find studentIDs of the students who have not completed a semester yet.

- Select studentId

From student

Where gpa IS NULL



StudentID
1001
1004

**Student**

StudentID	Name	gpa	progId
1000	Amal	3.2	BM
1001	Nimali	Null	IT
1002	Aruni	3.0	SE
1003	Surani	2.5	IT
1004	Imali	Null	BM

# DISTINCT OPERATOR

- In a table, a column may contain many duplicate values.
- Duplicates in results can be eliminated using DISTINCT operator

■ Ex :

Select progld  
From student

Progld
BM
IT
SE
IT
BM

**Student**

StudentID	Name	gpa	progld
1000	Amal	3.2	BM
1001	Nimali	Null	IT
1002	Aruni	3.0	SE
1003	Surani	2.5	IT
1004	Imali	Null	BM

Select DISTINCT progld  
From student

Progld
BM
IT
SE



# BETWEEN OPERATOR

- Used to check whether attribute value is within a range
- Ex :Find the students who will be obtaining a first class ( $3.7 \leq \text{gpa} \leq 4.0$ )

Select studentID

From student

Where gpa between 3.7 and 4.00



**Student**

StudentID	Name	gpa	progId
1000	Amal	3.2	BM
1001	Nimali	Null	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	IT
1004	Imali	4.0	BM

StudentID
1002
1004

# ORDER BY CLAUSE

- Used to order results based on a given field
- Ordering is ascending (ASC), unless you specify the DESC keyword
- Ex : display the student names and gpa's in the ascending order of gpa's.

Select Name, gpa



From student

Order by gpa

Name	gpa
Surani	2.5
Nimali	2.8
Amal	3.2
Aruni	3.8
Imali	4.0

## Student

StudentID	Name	gpa	progId
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	IT
1004	Imali	4.0	BM

# [INNER] JOIN

- Joins two tables based on a certain condition
- Ex : Find the names of students who follow programs offered by SLIIT

Select s.Name

From student s, program p

where s.pid=p.ProgId and offerBy='SLIIT'

Or

Select s.Name

From student s INNER JOIN program p on

s.pid=p.progId


Where offerBy='SLIIT'

**Student**

SID	Name	gpa	pid
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	IT

**Program**

progId	years	Offer By
BM	3	Curtin
IT	4	SLIIT
SE	3	SHU



Name
Nimali
Surani

# LEFT OUTER JOIN

- Returns all rows from the table on the left hand side of join, with the matching rows in the table on the right hand side of the join.
- The result is NULL in the right side when there is no match.
  - Ex : For all the students display the name and the offering institute

Select s.Name, p.offerBy

From student s LEFT OUTER JOIN program p  
on s.pid=p.progId

**Student**

SID	Name	gpa	pid
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	IT

**Program**

progId	years	Offer By
BM	3	Curtin
IT	4	SLIIT



Name	offerBy
Amal	Curtin
Nimali	SLIIT
Aruni	NULL
Surani	SLIIT

# RIGHT OUTER JOIN

- Returns all rows from the table on the right hand side of join, with the matching rows in the table on the left hand side of the join.
- The result is NULL in the left side when there is no match.
  - Ex : For all the programs display the offering institute and names of the students following

Select s.Name, p.offerBy

From student s RIGHT OUTER JOIN program p  
on s.pid=p.progld

**Student**

SID	Name	gpa	pid
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	NUL L
1003	Surani	2.5	IT

**Program**

progld	years	Offer By
BM	3	Curtin
IT	4	SLIIT
SE	3	SHU



Name	offerBy
Amal	Curtin
Nimali	SLIIT
Surani	SLIIT
NULL	SHU

# IN OPERATOR

- Used to check whether attribute value matches any value within a value list
- Ex : Find the students who has obtained a 'A'.

Select s.Name

From Student s

Where s.SID IN ( Select SID

from Grades

Where Grade='A')



**Student**

SID	Name	gpa
1000	Amal	3.2
1001	Nimali	2.8
1002	Aruni	3.8

**Grades**

SID	cid	Grade
1000	IT102	A
1000	IT100	B
1001	IT102	A
1002	IT102	C
1002	IT200	C

Name
Amal
Nimali

# EXISTS OPERATOR

- Used to check if subquery returns any rows
- Ex : Find the students who has obtained a 'A'.

Select s.Name

From Student s

Where EXISTS ( Select \*

from grades g

Where g.SID=s.SID and  
g.Grade='A')



**Student**

SID	Name	gpa
1000	Amal	3.2
1001	Nimali	2.8
1002	Aruni	3.8

**Grades**

SID	cid	Grade
1000	IT102	A
1000	IT100	B
1001	IT102	A
1002	IT102	C
1002	IT200	C

Name
Amal
Nimali

# COMPARISON OPERATORS WITH SOME, ANY & ALL

- Comparison operators such as `=`, `<>`, `>`, `>=`, `<` and `<=` could be modified using operators `SOME`, `ANY` and `ALL`.
  - `SOME` and `ANY` : used to compare a value to a list or subquery. Return true if at least one of the comparison evaluates as true.
  - `ALL` : used to compare a value to a list or subquery. If all of the comparisons evaluate to true then the result of the `ALL` expression will be true. Otherwise the result will be false



# ANY AND SOME OPERATORS

- Ex : Find BM students who has gpa greater than any of the IT students.

Select s.Name

From student s

Where s.progID='BM' and

s.gpa > ANY (

select sl.gpa

from student sl where  
sl.progId='IT')



**Student**

StudentID	Name	Gpa	progId
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	BM
1004	Imali	4.0	IT

Name
Amal

# ALL OPERATOR

- Ex : Find IT students who has **Student** gpa greater than all the BM students.

Select s.Name

From student s

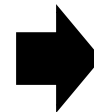
Where s.progID='IT' and

s.gpa > ALL (

select sl.gpa

from student sl where  
sl.progId='BM')

StudentID	Name	gpa	progId
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	BM
1004	Imali	4.0	IT



Name
Imali

# AGGREGATION

- An aggregate function summarizes the results of an expression over a number of rows, returning a single value.
- Some of the commonly used aggregate functions are SUM, COUNT, AVG, MIN and MAX

## AGGREGATION (CONTD.)

- Ex : Find the average, minimum, maximum gpa of students

Student

StudentID	Name	gpa	progId
1000	Amal	3.2	BM
1001	Nimali	2.8	IT
1002	Aruni	3.8	SE
1003	Surani	2.5	BM
1004	Imali	4.0	IT

Select AVG(gpa), MIN(gpa), MAX(gpa)  
From student



AVG(gpa)	MIN(gpa)	MAX(gpa)
3.26	2.5	4.0

# GROUPING (GROUP BY CLAUSE)

- Groups the data in tables and produces a single summary row for each group.
- Grouping is done based on a values in a given field
- When using group by
  - Each item in the SELECT list must be single valued per group.
  - SELECT clause may only contain Columns names, aggregate function, constants or an expression involving combinations of the above.
  - All column names in SELECT list must appear in the GROUP BY clause unless the name is used only in the aggregate function.

# GROUPING (CONTD.)

- Ex: Count the number of students who has followed each module.

Student

SID	CID	Grade
1000	DBII	A
1000	SEI	B
1001	DBII	A
1002	DBII	C
1002	SPD	C

SID	CID	Grade
1000	DBII	A
1001	DBII	A
1002	DBII	C
1000	SEI	B
1002	SPD	C

Select CID, Count(SID)

From Student

Group by CID



CID	Count (SID)
DBII	3
SEI	1
SPD	1

# HAVING CLAUSE

- Used to apply conditions on the groupings
- Ex: Find courses which is followed by more than two students

Select CID, Count(SID)

From course

Group by CID

Having count(SID)>2



CID	Count (SID)
DBII	3

Student

SID	CID	Grade
1000	DBII	A
1000	SEI	B
1001	DBII	A
1002	DBII	C
1002	SPD	C

SID	CID	Grade
1000	DBII	A
1001	DBII	A
1002	DBII	C
1000	SEI	B
1002	SPD	C

# SQL: REVIEW (CONTD.)

## Summary of SQL Queries...

```
SELECT <attribute-list>
FROM   <table-list>
[WHERE   <condition>]
[GROUP BY   <group attribute(s)>]
[HAVING   <group condition>]
[ORDER    BY <attribute list>];
```



# WHAT YOU HAVE TO DO BY NEXT WEEK

- Try out the self-test questions on the course web.
- Complete the tutorial.