SINGAPORE INSTITUTE OF TECHNOLOGY

INF1001: Introduction to Computing

L2b: Database – SQL

Aug 2023

Topic outline: Database-SQL

Relational Queries & SQL

Overview

ER-Schema-SQL

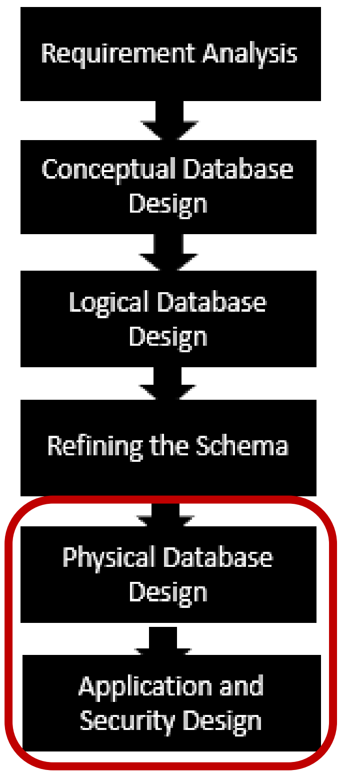
SQL Statements

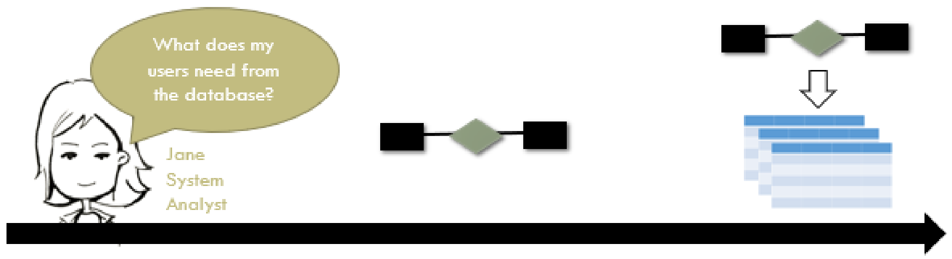
CREATE, INSERT, SELECT, Nested Queries

Syntax

Examples

Recap on Stages of Database Design





1. Requirement Analysis

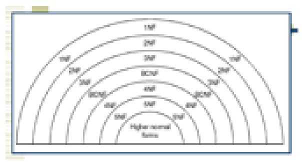
- Understand what data needs to be stored in the database
- What kind of applications uses the data?
- What operation needs to be performed?

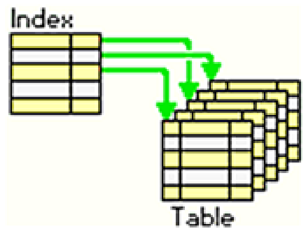
2. Conceptual Database Design

- Uses information from requirement analysis to develop a high-level description of the data
- Understand what are the constraints that need to be modeled
- Uses the ER model for modeling

3.Logical Database Design

- Conceptual database design -> Database Schema
- Determine the DBMS that will be used to implement the database design
- Convert ER to a relational database schema





4. Schema Refinement

- Analyze the relations to identify potential problems and refine it
- Use design theory for normalization

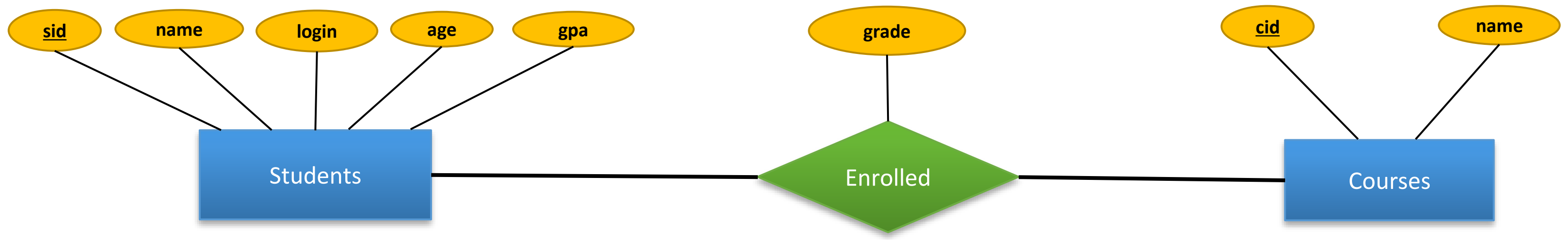
5. Physical Database Design

- Consider typical workloads and further refine database design to ensure it meets performance criteria
- Build indexes on tables and cluster some tables

6. Application & Security Design

- Identify parts of database that must be accessible and parts that must not be accessible
- Ensure access rules are enforced

ER Diagram and Relational Schema



Relational Schema:

Students(<u>sid varchar(20)</u>, name varchar(20), login varchar(10), age integer, gpa real)

Courses(<u>cid varchar(20)</u>, name varchar(20))

Enrolled (<u>cid varchar(20), sid varchar(20)</u>, grade varchar(5))

Relational Queries

- A major strength of the relational model: supports simple, powerful ***querying*** of data.

- Queries can be written intuitively, and the DBMS is responsible for ***efficient*** evaluation.

- Query:
  - Input and output of a query is a relation
  - Evaluated using instances of each input relation
  - Produces ***instance*** of output relation

Structured Query Language (SQL)

- Most widely used commercial relational database (DB) language
- Originally developed by IBM
- SQL is a very-high-level language.
  - Say “**what** to do” rather than “**how** to do it.”
  - Avoid a lot of data-manipulation details needed in procedural languages like C++ or Java.
- Database management system figures out “best” way to execute query.
  - Called “query optimization.”

Structured Query Language (SQL)

- SQL statements:
  - CREATE
  - INSERT
  - SELECT
  - Nested Queries

Outline

Relational Queries & SQL

Overview

ER-Schema-SQL

SQL Statements

CREATE, INSERT, SELECT, Nested Queries

Syntax

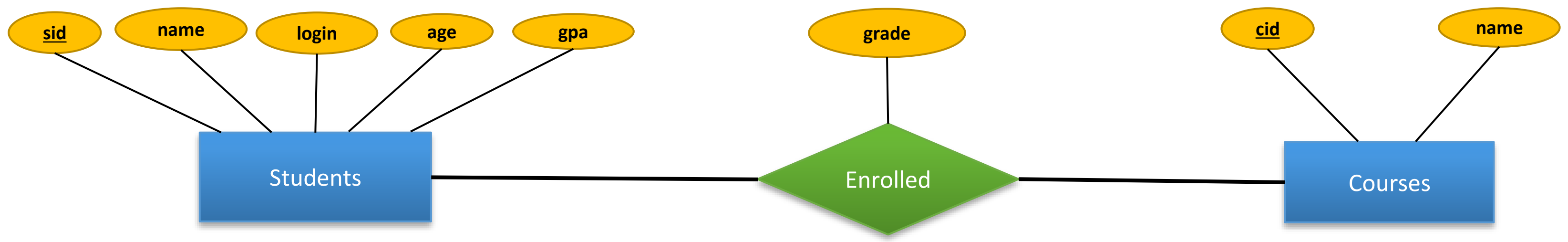
Examples

1. SQL - CREATE

- Creating tables/relations

```
CREATE TABLE table-name (
      attribute-name₁ attribute type₁,
            …,
      attribute-nameₙ attribute typeₙ
      PRIMARY KEY (attribute-name)
      )
```

1. SQL – CREATE



Schema

```
Students(sid varchar(20),
         name varchar(20),
         login varchar(10),
         age integer,
         gpa real)
```

```
CREATE TABLE Students
          (sid VARCHAR(20),
           name VARCHAR(20),
           login VARCHAR(10),
           age INTEGER,
           gpa REAL)
```

- Creates the Students relation.
- Observe that the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

1. SQL – CREATE

```
CREATE TABLE Students
         (sid VARCHAR(20),
          name VARCHAR(20),
          login VARCHAR(10),
          age INTEGER,
          gpa REAL,
          PRIMARY KEY (sid))
```

Primary Key:

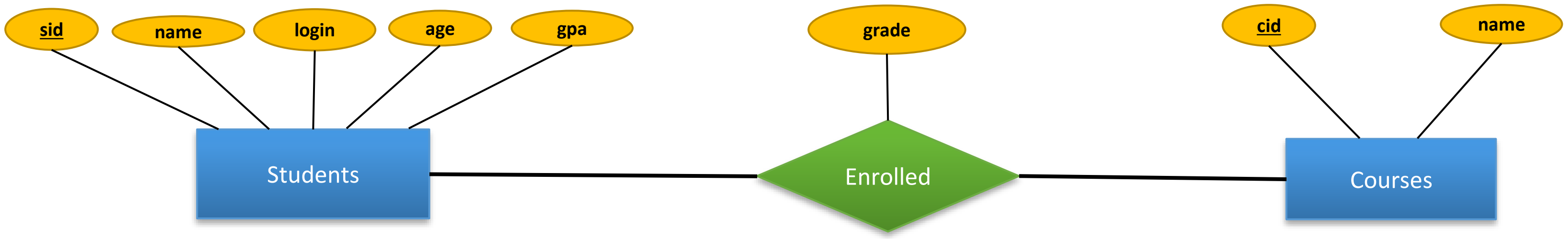
- Uniquely identifies each record in a table
- Must contain unique values
- Cannot contain NULL values

Table can have **only 1 primary key**, which may consist of a single or multiple fields

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

1. SQL – CREATE



Schema

```
Courses(cid varchar(20), name varchar(20))
```

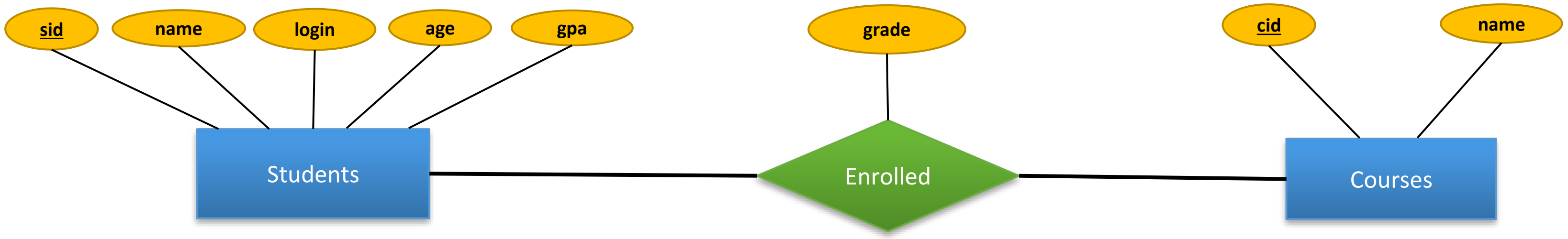
```
CREATE TABLE Courses
        (cid VARCHAR(20),
         name VARCHAR(20),
         PRIMARY KEY (cid))
```

Courses

cid	name
ICT1001	Introduction to ICT
ICT1002	Programming
ICT1003	Computer Architecture

1. SQL – CREATE

- Enrolled table holds information about courses that students take.



```
Students(sid varchar(20), name varchar(20), login varchar(10), age integer, gpa real)
Courses(cid varchar(20), name varchar(20))
Enrolled (sid varchar(20), cid varchar(20), grade varchar(5))
```

```
CREATE TABLE Students
          (sid VARCHAR(20), name VARCHAR(20),
           login VARCHAR(10),  age INTEGER,  gpa REAL,
           PRIMARY KEY  (sid) )
```

```
CREATE TABLE Courses
           (cid VARCHAR(20),
            name VARCHAR(20),
            PRIMARY KEY  (cid) )
```

```
CREATE TABLE Enrolled
          (sid VARCHAR(20), cid VARCHAR(20),
           grade VARCHAR(5))
```

Only students listed in the Students relation should be allowed to enroll for courses. How to enforce this?

1. SQL – CREATE: Foreign Key

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
  - The table with the foreign key is called the **referencing** or **child** table
  - Table with the primary key is called the **referenced** or **parent** table.
- The FOREIGN KEY constraint is used to prevent actions (such as deleting tuples) that would destroy links between tables.
- The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table that it points to.
  - E.g., FAIL/ERROR if trying to add a `grade` for the student `sid` in the child `Enrolled` table that references a non-existent `sid` in the parent `Students table`

1. SQL – CREATE: Foreign Key

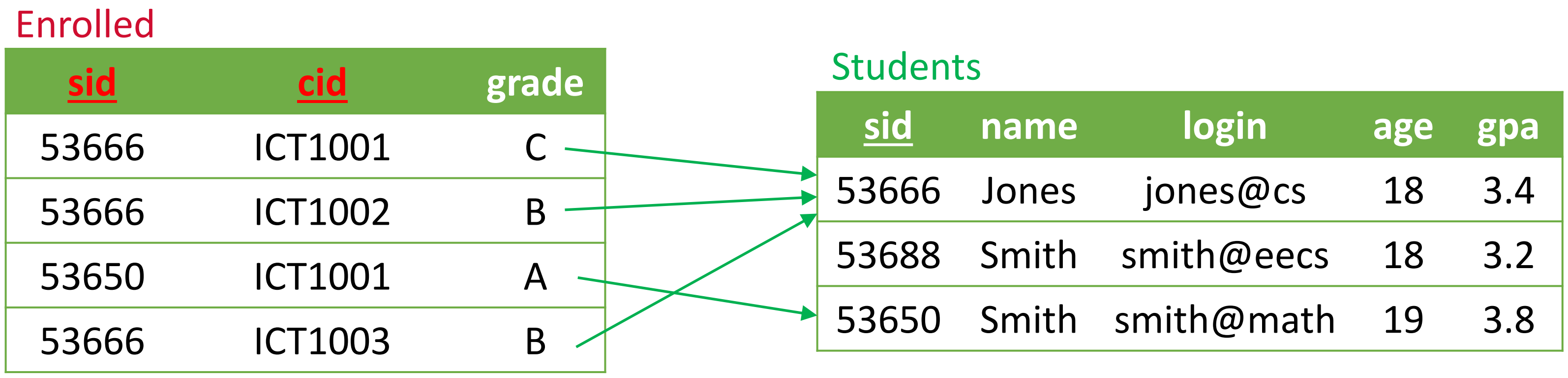
- Only students listed in the Students relation should be allowed to enroll for courses.

```
CREATE TABLE Students
           (sid VARCHAR(20), name VARCHAR(20),
            login VARCHAR(10),  age INTEGER,  gpa REAL,
            PRIMARY KEY  (sid))
```

```
CREATE TABLE Courses
           (cid VARCHAR(20),
            name VARCHAR(20),
            PRIMARY KEY  (cid))
```

```
CREATE TABLE Enrolled
   (sid VARCHAR(20), cid VARCHAR(20),  grade VARCHAR(5),
     PRIMARY KEY  (sid, cid),
     FOREIGN KEY (sid) REFERENCES Students(sid)
     FOREIGN KEY (cid) REFERENCES Courses(cid))
```

Enrolled is the referencing/child relation, while Students is the referenced/parent relation



Enrolled

sid	cid	grade
53666	ICT1001	C
53666	ICT1002	B
53650	ICT1001	A
53666	ICT1003	B

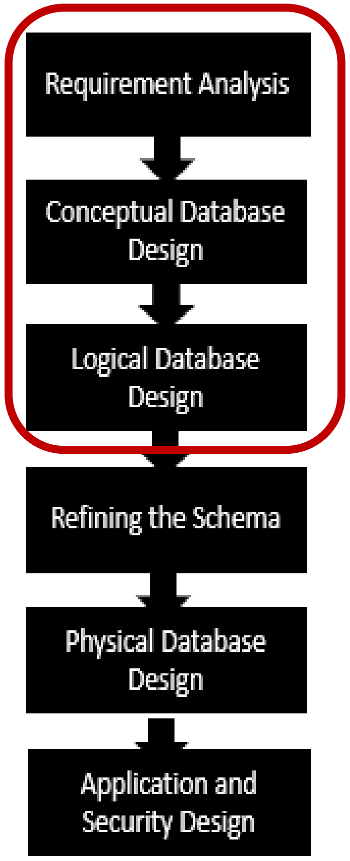
Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

1. SQL – CREATE: Foreign Key

- If all foreign key constraints are enforced, **referential integrity** is achieved
- Consider parent Students and child Enrolled tables; *sid* in Enrolled is a foreign key that references Students relation.
  - What should be done if an Enrolled tuple with a non-existent student sid is **inserted**?
    - Reject it!
  - What should be done if a Students tuple is **deleted**? Several options:
    - Disallow deletion of the Students tuple that is referred to
    - Delete all Enrolled tuples that refer to it
    - Set sid in Enrolled tuples that refer to it to a default sid (for e.g. 00000)
  - Similar actions are taken if primary key Student is **updated**

From Requirements to Logical Database Design (Schema)



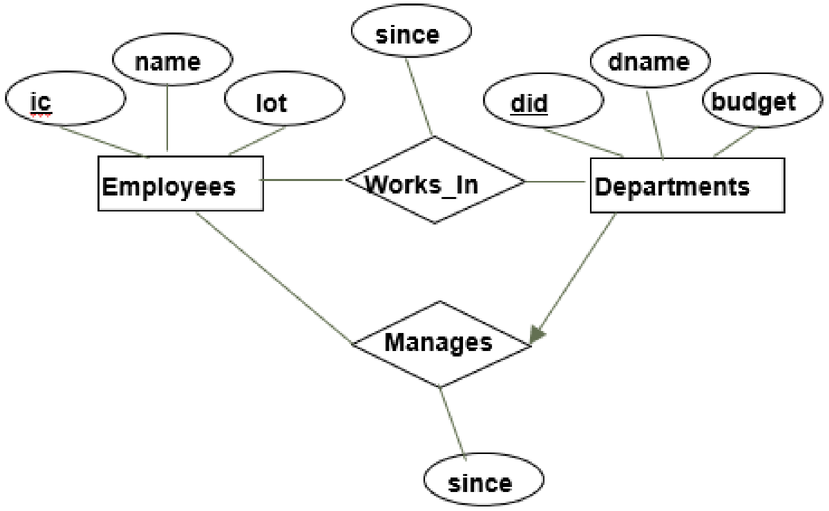
Employee Management System

We would like to develop a database application to keep information of our Employees.

We would like to keep track of the employee information such IC, name, and parking lot.

We would also like to keep track of departments through their unique ID, name and budget. In addition, we would like to keep track of the department that the employee is working in and the time the employee joined.

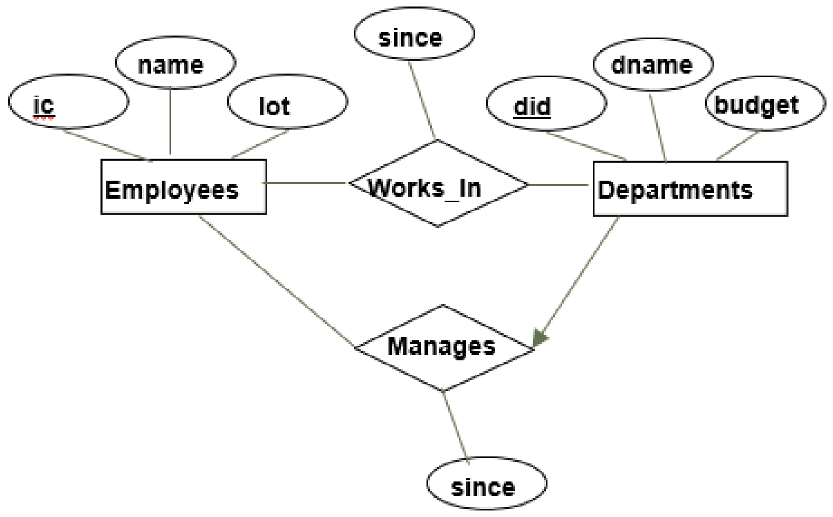
A department is managed by one employee. The same employee can manage multiple departments.



Relational Schema

```
Employees (ic: char(11), name: char(11), lot: integer)
Works_In (ic: char(11), did: int, since: date)
Dept_Mgr (did: int, dname: char(11), budget: real,
          ic: char(11), since: date)
```

From Logical Database Design (Schema) to SQL



Relational Schema

```
Employees (ic: char(11), name: char(11), lot: integer)
Works_In (ic: char(11), did: int, since: date)
Dept_Mgr (did: int, dname: char(11), budget: real,
          ic: char(11), since: date)
```

```
CREATE TABLE Works_In (
  ic  VARCHAR(11),
  did  INTEGER,
  since  DATE,
  PRIMARY KEY (ic, did),
  FOREIGN KEY (ic) REFERENCES Employees(ic),
  FOREIGN KEY (did) REFERENCES Dept_Mgr(did))
```

```
CREATE TABLE Employees (
   ic VARCHAR(11),
   name VARCHAR(11),
   lot  INTEGER,
   PRIMARY KEY (ic))
```

```
CREATE TABLE  Dept_Mgr (
   did  INTEGER,
   dname  VARCHAR(11),
   budget  REAL,
   ic  VARCHAR(11),
   since  DATE,
   PRIMARY KEY (did),
   FOREIGN KEY (ic) REFERENCES Employees(ic))
```

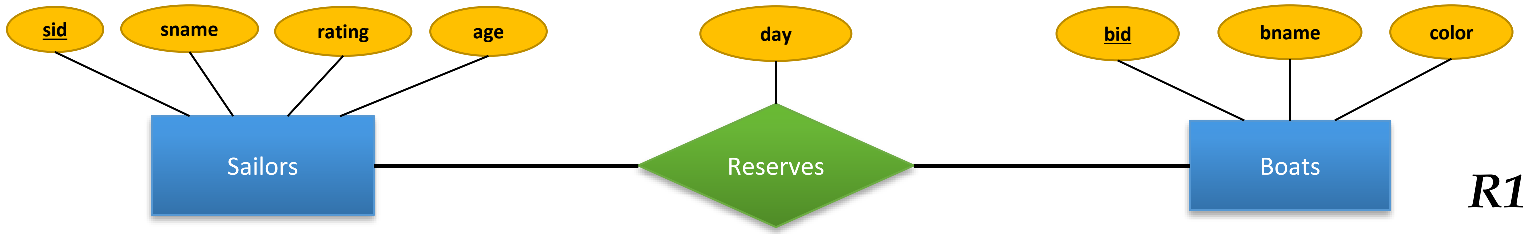
Example Instances

- We will use these instances of the Sailors, Boats and Reserves relations in our examples.

```
Sailors(sid: integer, sname: string, rating: integer, age: real)

Boats(bid: integer, bname: string, color: string)

Reserves(sid: integer, bid: integer, day: date)
```



R1

```
CREATE TABLE Sailors (sid int unsigned, sname varchar(10), rating int, age real,
                      PRIMARY KEY (sid))
```

Sailors

<u>sid</u>	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
74	Rusty	7	35.0

Example Instances

```
CREATE TABLE Boats (bid int unsigned, bname varchar(10), color varchar(10),
                    PRIMARY KEY (bid))
```

```
Sailors(sid: integer, sname: string, rating: integer, age: real)

Boats(bid: integer, bname: string, color: string)

Reserves(sid: integer, bid: integer, day: date)
```

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	green
104	Marine	red

```
CREATE TABLE Reserves (sid int unsigned, bid int unsigned, day date,
                       PRIMARY KEY(sid, bid),
                       FOREIGN KEY(sid) REFERENCES sailors(sid),
                       FOREIGN KEY(bid) REFERENCES boats(bid))
```

Reserves

sid	bid	day
22	101	18/10/21
58	103	25/12/21

2. SQL - INSERT

- Adding tuples to relation

```
INSERT INTO table-name
       (attribute-name₁, …, attribute-nameₙ,)
VALUES
      (attribute-value₁, …, attribute-valueₙ,)
```

- Example: insert a single tuple into Students relation:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES  (53688, ‘Smith’, ‘smith@eecs’, 18, 3.2)
```

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53650	Smith	smith@math	19	3.8
53688	Smith	smith@eecs	18	3.2

3. SQL - SELECT

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

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the *result-set.*
- *column1, column2,* ... are the field names of the table you want to select data from.

If you want to select all the fields available in the table:

```
SELECT * FROM table_name;
```

If you want to return only distinct (different) values:

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

SQL SELECT Statement

```
CREATE TABLE Sailors (sid int unsigned, sname varchar(10), rating int, age real,
                     PRIMARY KEY (sid))
```

Q1: Find the name and age of all sailors

```
SELECT   sname, age
FROM     Sailors
```

Query Result:

sname	age
Dustin	45.0
Lubber	55.5
Rusty	35.0
Rusty	35.0

Q2: Find DISTINCT name and age of all sailors

```
SELECT   DISTINCT sname, age
FROM     Sailors
```

Query Result:

sname	age
Dustin	45.0
Lubber	55.5
Rusty	35.0

Removes duplicates in the result set

SQL SELECT Statement

SINGAPORE INSTITUTE OF TECHNOLOGY

```
CREATE TABLE Sailors (sid int unsigned, sname varchar(10), rating int, age real,
                      PRIMARY KEY (sid))
```

Q3: Find sid, name, age, rating of all sailors

```
SELECT  sid, sname, age, rating
FROM    Sailors
```

OR

```
SELECT  *
FROM     Sailors
```

Query Result:

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
74	Rusty	7	35.0

3. SQL - SELECT

- Adding conditions in your SELECT statement

```
SELECT desired attributes
FROM   one or more tables
WHERE  condition about tuples of tables
```

SQL SELECT Statement

Q4: Find all sailors with rating > 7

```
SELECT  sname, age
FROM    Sailors
WHERE   rating > 7
```

Query Result:

sname	age
Lubber	55.5
Rusty	35.0

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
74	Rusty	7	35.0

Q5: Find all sailors who are older than 35

```
SELECT sname, age
FROM   Sailors
WHERE age > 35
```

Query Result:

sname	age
Dustin	45.0
Lubber	55.5

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
74	Rusty	7	35.0

3. SQL – SELECT: Complex conditions

- Complex condition in WHERE clause
  - Boolean operators AND, OR, NOT.
  - Comparisons =, <>, <, >, <=, >=.
    - And many other operators that produce boolean-valued results.

3. SQL – SELECT: Complex conditions

- SELECT FROM 2 or more tables – Join Tables

Q6: Find the names of sailors who have reserved at least one boat

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid
```

JOIN-ing TWO tables

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Cross product

Reserves

sid	bid	day
22	101	18/10/21
58	103	25/12/21

(sid)	sname	rating	age	(sid)	bid	day
22	Dustin	7	45	22	101	18/10/21
22	Dustin	7	45	58	103	25/12/21
31	Lubber	8	55.5	22	101	18/10/21
31	Lubber	8	55.5	58	103	25/12/21
58	Rusty	10	35.0	22	101	18/10/21
58	Rusty	10	35.0	58	103	25/12/21

Evaluate the WHERE condition S.Sid = R.sid

sname
Dustin
Rusty

3. SQL – SELECT: Complex conditions

- SELECT FROM 2 or more tables – Join Tables

Summary : How is the query computed to produce a JOIN?

1. Produce a temporary table that contains the cross product of the two tables:
   - Each row from table 1 (**size m**) is concatenated with each row from table 2 (**size n**)
   - Produce a **size m × n** cross product table
2. Eliminate rows from the cross product temporary table by evaluating the WHERE condition
3. Outputs final results using the attributes in the SELECT statement

3. SQL – SELECT: Complex conditions

- SELECT FROM 2 or more tables – Join Tables

Q7: Find the names of sailors who have reserved boat 103

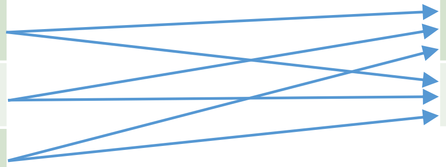
```
SELECT  S.sname
FROM    Sailor S, Reserves R
WHERE   S.sid=R.sid AND R.bid=103
```

JOIN-ing TWO tables

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Cross product



Reserves

sid	bid	day
22	101	18/10/21
58	103	25/12/21

(sid)	sname	rating	age	(sid)	bid	day
22	Dustin	7	45	22	101	18/10/21
22	Dustin	7	45	58	103	25/12/21
31	Lubber	8	55.5	22	101	18/10/21
31	Lubber	8	55.5	58	103	25/12/21
58	Rusty	10	35.0	22	101	18/10/21
58	Rusty	10	35.0	58	103	25/12/21

sname
Rusty

Evaluate the WHERE condition S.Sid = R.sid

Evaluate the WHERE condition R.bid = 103

3. SQL – SELECT: Complex conditions

- SELECT FROM 2 or more tables – Join Tables

Q8: Find the names of student who has obtained an A grade in their course

```
SELECT S.name, E.cid
FROM   Students S, Enrolled E
WHERE  S.sid=E.sid AND E.grade=‘A’
```

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

We get:

S.name	E.cid
Smith	Topology112

4. SQL – SELECT: Nested Queries

- A very powerful feature of SQL: a WHERE clause can itself contain an SQL query!

- To understand semantics of nested queries, think of a *nested loops* evaluation

4. SQL – SELECT: Nested Queries

Q9: Find the names of sailors who have reserved boat 103

```
Previous Example:    SELECT S.sname
(Q7)                 FROM   Sailor S, Reserves R
                     WHERE  S.sid=R.sid AND R.bid=103
```

```
SELECT  S.sname
FROM   Sailors S
WHERE   S.sid IN  (SELECT R.sid
                   FROM   Reserves R
                   WHERE  R.bid=103)
```

IN tests whether a value is in a given set

Reserves

Sid	bid	day
22	101	18/10/21
58	103	25/12/21

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

The nested subquery computes the multi set of sids for sailors who have reserved boat 103
The top level query retrieves the names of sailors whose sid is in the set

- To find sailors who *have not* reserved #103, use NOT IN.
- To understand semantics of nested queries, think of a *nested loops* evaluation

More Examples

Q10: Find the sid of sailors who have reserved a red boat

```
SELECT R.sid
FROM   Reserves R, Boats B
WHERE  R.bid = B.bid AND B.color = ‘red’
```

OR

```
SELECT R.sid
FROM   Reserves R
WHERE  R.bid IN (SELECT B.bid
                 FROM Boats B
                 WHERE B.color = ‘red’
```

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
74	Rusty	7	35.0

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	green
104	Marine	red

Reserves

sid	bid	day
22	101	18/10/21
58	103	25/12/21

More Examples

Q11: Find the names of sailors who have reserved a red boat

```
SELECT S.sname
FROM   Sailors S, Reserves R, Boats B
WHERE  S.sid = R.sid AND R.bid = B.bid AND B.color = ‘red’
```

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
74	Rusty	7	35.0

Boats

bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	green
104	Marine	red

Reserves

sid	bid	day
22	101	18/10/21
58	103	25/12/21

Summary

Relational Queries & SQL

Overview

ER-Schema-SQL

SQL Statements

CREATE, INSERT, SELECT, Nested Queries

Syntax

Examples

To be continued in Year 2 Database courses (for CS and SE students)

More SQL practice

- Kueri.me
- http://www.geektime.com/2016/07/24/kueri-development-like-google-translate-from-natural-language-to-sql/

- http://sqlzoo.net/
- If you want answers, you can just add “?answer=1” at the end of the URL and the answers will display in the text boxes.

- https://www.w3resource.com/sql-exercises/

References

- Good reference: SQL Tutorial at https://www.w3schools.com/sql/