CSIT115 Data Management and Security

SELECT statement (1)

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

SELECT statement (1)

Outline

Functionality and syntax

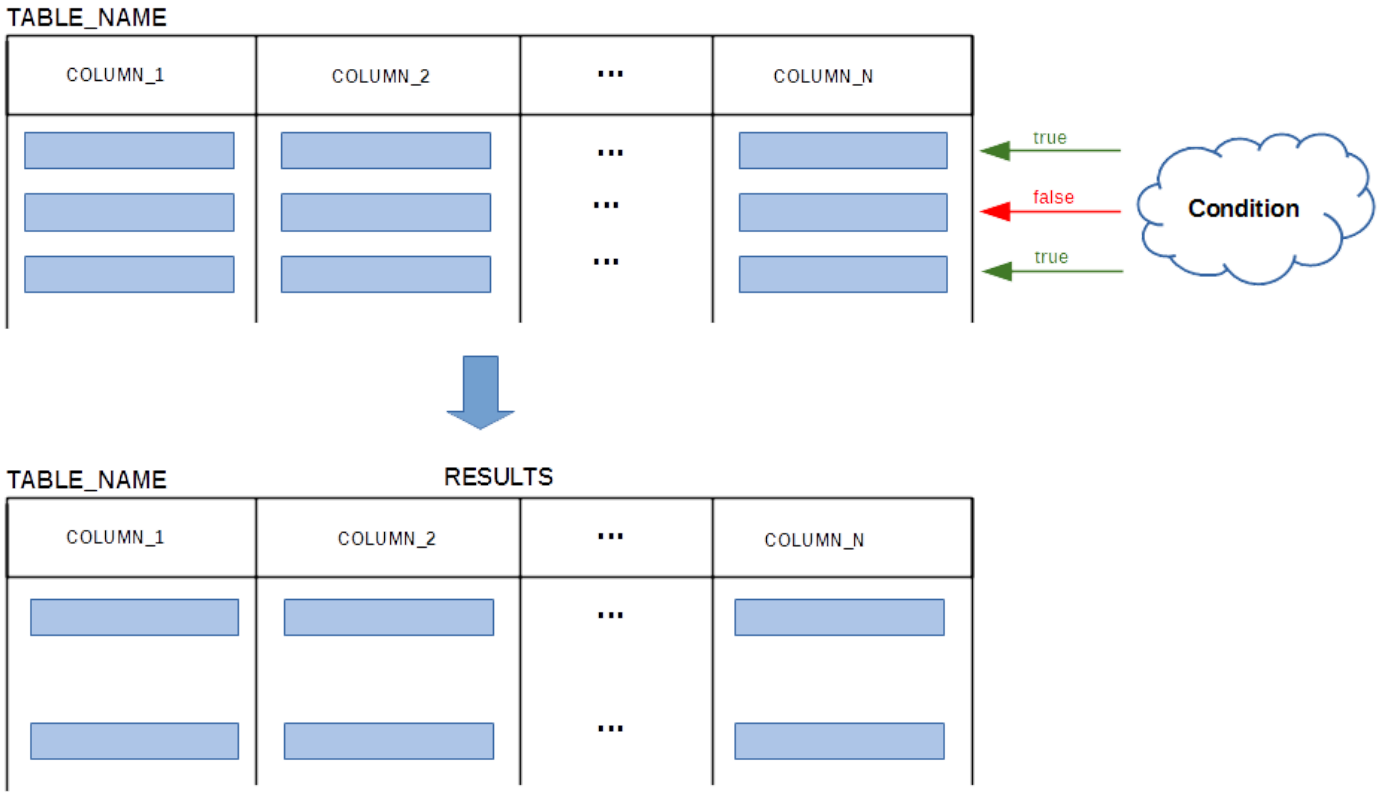
Functionality

- `SELECT` statement retrieves data from a relational database
- The results of `SELECT` statement can be considered as a transient relational table
- The results of `SELECT` statement can be saved as a persistent relational table

Functionality and syntax

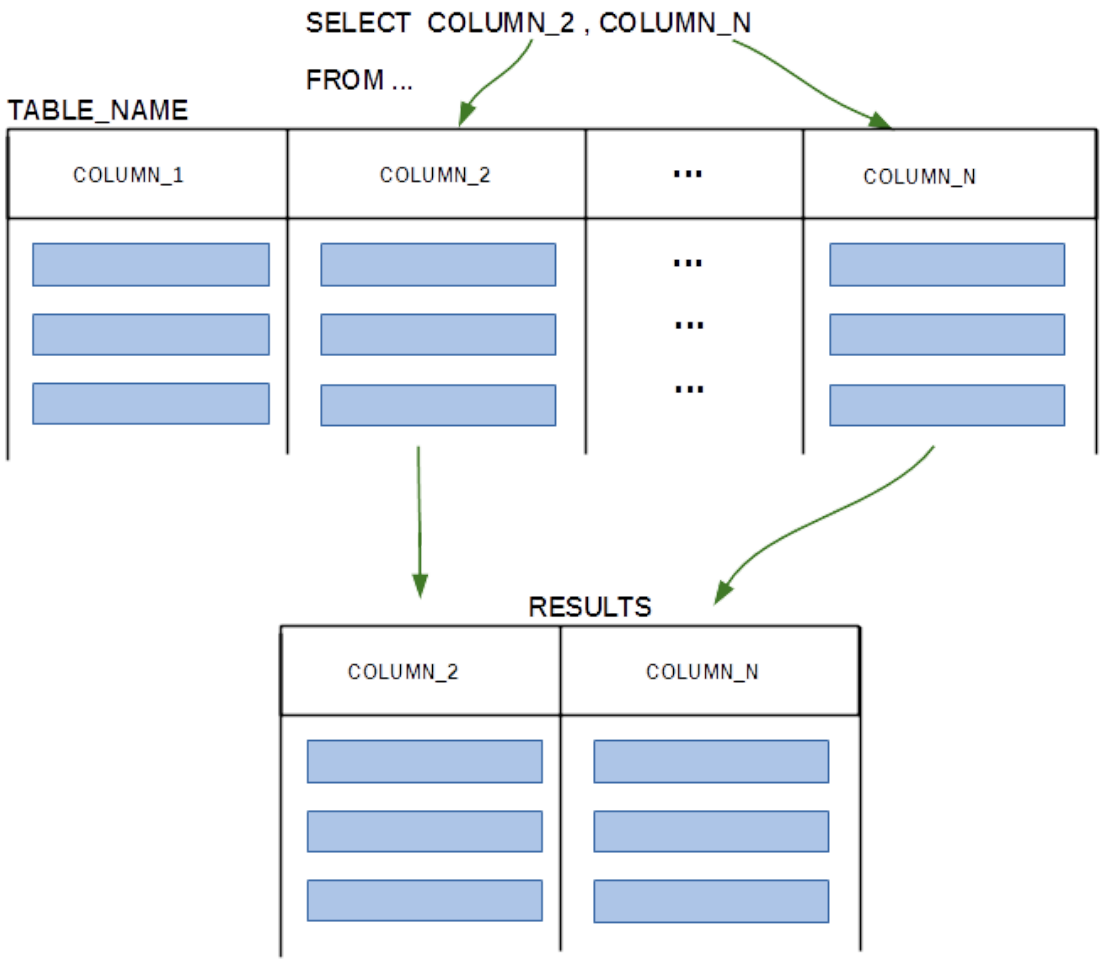
Selection queries select complete rows from a relational table

```
SELECT *
FROM ...
WHERE Condition ;
```



Functionality and syntax

Projection queries select complete columns from a relational table



Functionality and syntax

Sample database

CREATE TABLE statement

```
CREATE TABLE DEPARTMENT(
 name               VARCHAR(50)        NOT NULL,
 code               CHAR(5)            NOT NULL,
 total_staff_number DECIMAL(2)         NOT NULL,
 chair              VARCHAR(50)            NULL,
 budget             DECIMAL(9,1)           NULL, /* In this presentation */
  CONSTRAINT dept_pkey PRIMARY KEY(name),        /*  budget can be NULL  */
  CONSTRAINT dept_ckey1 UNIQUE(code),
  CONSTRAINT dept_ckey2 UNIQUE(chair),
  CONSTRAINT dept_check1 CHECK (total_staff_number BETWEEN 1 AND 50) );
```

CREATE TABLE statement

```
CREATE TABLE COURSE(
cnum               CHAR(7)           NOT NULL,
title              VARCHAR(200)      NOT NULL,
credits            DECIMAL(2)        NOT NULL,
offered_by         VARCHAR(50)           NULL,
 CONSTRAINT course_pkey PRIMARY KEY(cnum),
 CONSTRAINT course_check1 CHECK (credits IN (6, 12)),
 CONSTRAINT course_fkey1 FOREIGN KEY(offered_by)
                      REFERENCES DEPARTMENT(name) ON DELETE CASCADE );
```

Functionality and syntax

Examples

- Find a name and code of a department that has more than 30 staff members

SELECT statement

```
SELECT code, name
FROM DEPARTMENT
WHERE total_staff_number > 30;
```

- Find a code of a department and course number such that a course is offered by a department and a course provides 6 credit points

SELECT statement with JOIN operation

```
SELECT DEPARTMENT.code, COURSE.cnum, title
FROM DEPARTMENT JOIN COURSE
                ON DEPARTMENT.name = COURSE.offered_by
WHERE credits = 6;
```

- Find all information about courses offered by a department whose chair is James Bond

Nested SELECT statement

```
SELECT *
FROM COURSE
WHERE offered_by IN ( SELECT name
                      FROM DEPARTMENT
                      WHERE chair = 'James Bond' );
```

Functionality and syntax

Examples

- Find all information about courses offered by a department whose chair is James Bond

SELECT statement with inline view

```
SELECT *
FROM COURSE JOIN ( SELECT name
                   FROM DEPARTMENT
                   WHERE chair = 'James Bond' ) JB
             ON COURSE.offered_by = JB.name;
```

- Find all information about courses offered by a department whose chair is James Bond

SELECT statement with WITH clause

```
WITH JAMES AS
           ( SELECT name
             FROM DEPARTMENT
             WHERE chair = 'James Bond' ),
     JAMESCOURSE AS
          ( SELECT *
            FROM COURSE JOIN JAMES
            ON COURSE.offered_by = JAMES.name )
SELECT * FROM JAMESCOURSE;
```

Functionality and syntax

Keywords

SELECT statement

```
SELECT code, name
FROM DEPARTMENT
WHERE total_staff_number > 30;
```

SELECT statement

```
SELECT code, cnum, title
FROM DEPARTMENT, COURSE
WHERE name = offered_by AND credits = 6;
```

Nested SELECT statement

```
SELECT *
FROM COURSE
WHERE offered_by IN ( SELECT name
                      FROM DEPARTMENT
                      WHERE chair = 'James Bond' );
```

SELECT statement with inline view

```
SELECT *
FROM COURSE JOIN ( SELECT name
                   FROM DEPARTMENT
                   WHERE chair = 'James Bond' ) JB
             ON COURSE.offered_by = JB.name;
```

Functionality and syntax

Selected columns

SELECT statement

```
SELECT code, name
FROM DEPARTMENT
WHERE total_staff_number > 30;
```

SELECT statement

```
SELECT code, cnum, title
FROM DEPARTMENT, COURSE
WHERE name = offered_by AND credits = 6;
```

Nested SELECT statement

```
SELECT *
FROM COURSE
WHERE offered_by IN ( SELECT name
                      FROM DEPARTMENT
                      WHERE chair = 'James Bond' );
```

SELECT statement with inline view

```
SELECT *
FROM COURSE JOIN ( SELECT name
                   FROM DEPARTMENT
                   WHERE chair = 'James Bond' ) JB
             ON COURSE.offered_by = JB.name;
```

Functionality and syntax

Relational tables used

SELECT statement

```
SELECT code, name
FROM DEPARTMENT
WHERE total_staff_number > 30;
```

SELECT statement

```
SELECT code, cnum, title
FROM DEPARTMENT, COURSE
WHERE name = offered_by AND credits = 6;
```

Nested SELECT statement

```
SELECT *
FROM COURSE
WHERE offered_by IN ( SELECT name
                      FROM DEPARTMENT
                      WHERE chair = 'James Bond' );
```

SELECT statement with inline view

```
SELECT *
FROM COURSE JOIN ( SELECT name
                   FROM DEPARTMENT
                   WHERE chair = 'James Bond' ) JB
             ON COURSE.offered_by = JB.name;
```

Functionality and syntax

Conditions

SELECT statement

```
SELECT code, name
FROM DEPARTMENT
WHERE total_staff_number > 30;
```

SELECT statement

```
SELECT code, cnum, title
FROM DEPARTMENT, COURSE
WHERE name = offered_by AND credits = 6;
```

Nested SELECT statement

```
SELECT *
FROM COURSE
WHERE offered_by IN ( SELECT name
                      FROM DEPARTMENT
                      WHERE chair = 'James Bond' );
```

SELECT statement with inline view

```
SELECT *
FROM COURSE JOIN ( SELECT name
                   FROM DEPARTMENT
                   WHERE chair = 'James Bond' ) JB
             ON COURSE.offered_by = JB.name;
```

Functionality and syntax

Subqueries

Nested SELECT statement

```
SELECT *
FROM COURSE
WHERE offered_by IN ( SELECT name
                      FROM DEPARTMENT
                      WHERE chair = 'James Bond' );
```

SELECT statement with inline view

```
SELECT *
FROM COURSE JOIN   ( SELECT name
                     FROM DEPARTMENT
                     WHERE chair = 'James Bond' ) JB
            ON COURSE.offered_by = JB.name;
```

Functionality and syntax

WITH clause

Nested SELECT statement

```
WITH JAMES AS
          ( SELECT name
            FROM DEPARTMENT
            WHERE chair = 'James Bond' ),
     JAMESCOURSE AS
          ( SELECT *
            FROM COURSE JOIN JAMES
            ON COURSE.offered_by = JAMES.name )
SELECT *
FROM JAMESCOURSE;
```

SELECT statement (1)

Outline

Projection queries

Projection queries select the entire columns from a relational table and do not have `WHERE` clause

Find full information about all departments

```sql
SELECT code, name, total_staff_number, chair, budget
FROM DEPARTMENT;
```

```sql
SELECT *
FROM DEPARTMENT;
```

Find the names and chairpersons of all departments

```sql
SELECT name, chair
FROM DEPARTMENT;
```

Find the titles of all courses

```sql
SELECT title
FROM COURSE;
```

Projection queries with duplicates/ no duplicates

A keyword `DISTINCT` can be used to eliminated duplicates from the results of a query

Find the credit points of all courses

```sql
SELECT credits
FROM COURSE;
```

Find the distinct credit points of all courses

```sql
SELECT DISTINCT credits
FROM COURSE;
```

Find the distinct total number of staff members in each department

```sql
SELECT DISTINCT total_staff_number
FROM DEPARTMENT;
```

SELECT statement (1)

Outline

Queries with row functions

A row function is called and it is processed one time for each row selected from a relational table

List the names of departments in uppercase format

```
SELECT UPPER(name)
FROM DEPARTMENT;
```

UPPER row function

Find the first three characters from all course codes and full titles of all courses

```
SELECT SUBSTR(cnum, 1, 3), title
FROM COURSE;
```

SUBSTR row function

Display the name of departments and budgets increased by 10%

```
SELECT name, 1.1*IFNULL(budget,0)
FROM DEPARTMENT;
```

IFNULL row function

SELECT statement (1)

Outline

Queries with group functions

A group function is called and it is processed one time for a group of rows

Find the total number of courses

```
SELECT COUNT(*)
FROM COURSE;
```

COUNT group function

Find the total number of all staff members in all departments

```
SELECT SUM(total_staff_number)
FROM DEPARTMENT;
```

SUM group function

Find an average budget per each department

```
SELECT AVG(IFNULL(budget, 0))
FROM DEPARTMENT;
```

AVG group function

Find the total number of staff members in the largest department

```
SELECT MAX(total_staff_number)
FROM DEPARTMENT;
```

MAX group function

SELECT statement (1)

Outline

Special queries

SQL as a calculator

Compute 30 hours * $90.30 per hour

```
SELECT 30 * 90.30
FROM DUAL;
```

Arithmetic expression in SELECT clause

SQL as a diary

What date is tomorrow ?

```
SELECT DATE_ADD(SYSDATE(), INTERVAL 1 DAY)
FROM DUAL;
```

Date arithmetic

Add 2 months to a current date

```
SELECT DATE_ADD(SYSDATE(), INTERVAL 2 MONTH )
FROM DUAL;
```

Date arithmetic

Special queries

How many days have passed since 1 January 2001 ?

Date arithmetic

```
SELECT DATEDIFF(SYSDATE(),'2001-01-01')
FROM DUAL;
```

SQL as word processor

Who am I ?

String concatenation and user name

```
SELECT CONCAT('I am ', CURRENT_USER())
FROM DUAL;
```

Hello world !

The famous Hello world ! program

```
SELECT 'Hello world !'
FROM DUAL;
```

Substring of 'Hello world' that starts from the first 'e'

String processing

```
SELECT SUBSTR('Hello world', INSTR('Hello world','e'), LENGTH('Hello world'))
FROM DUAL;
```

References

T. Connoly, C. Begg, Database Systems, A Practical Approach to Design, Implementation, and Management, Chapters 6.3.1 - 6.3.4 Data Manipulation, Pearson Education Ltd, 2015

D. Darmawikarta, SQL for MySQL A Beginner's Tutorial, Chapters 2 - 5, Brainy Software Inc. First Edition: June 2014

How to ... ? Cookbook, How to implement queries in SQL, Recipe 5.1 How to implement SELECT statements with simple Boolean expressions ?