SELECT statement (3)

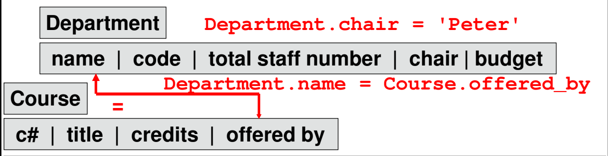
Sample database

```
CREATE TABLE Department(
name                VARCHAR2(50),
code                CHAR(5),
total_staff_number NUMBER(2)                NOT NULL,
hair                VARCHAR2(50),
budget              NUMBER(9,1)             NULL,
CONSTRAINT dept_pkey PRIMARY KEY(name),
CONSTRAINT dept_ckey1 UNIQUE(code),
CONSTRAINT dept_ckey2 UNIQUE(chair),
CONSTRAINT dept_check1
CHECK (total_staff_number BETWEEN 1 AND 50)  );
```

```
CREATE TABLE Course(
c#                 CHAR(7),
title              VARCHAR2(200)             NOT NULL,
credits            NUMBER(1)                 NOT NULL,
offered_by         VARCHAR2(50)              NULL,
CONSTRAINT course_pkey PRIMARY KEY(c#),
CONSTRAINT course_check1
        CHECK (credits IN (6, 12) ),
CONSTRAINT course_fkey1 FOREIGN KEY(offered_by)
        REFERENCES Department(name) );
```

Join queries

Find the titles of all courses offered by a department chaired by `Peter`


Department
Department.chair = 'Peter'
name | code | total staff number | chair | budget
Department.name = Course.offered_by
Course
=
c# | title | credits | offered by

```
SELECT Course.title
FROM Course, Department
WHERE Department.name = Course.offered_by AND
      Department.chair = 'Peter';
```

Join queries

Find the titles of all courses offered by a department chaired by `Peter`

```
SELECT Course.title
FROM Course, Department
WHERE Department.name = Course.offered_by AND
      Department.chair = 'Peter';
```

```
SELECT title
FROM Course, Department
WHERE name = offered_by AND
      chair = 'Peter';
```

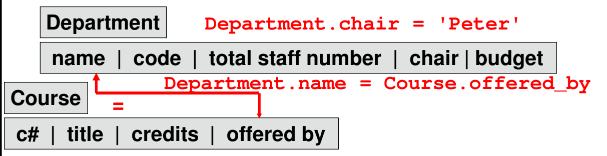
Computational model (join queries)

```
SELECT <ATTRIBUTES>
FROM <TABLE_1>, <TABLE_2>
WHERE <JOIN_CONDITION> AND <CONDITION>;
```

```
forall rows t in <TABLE_1>
  forall rows u in <TABLE_2>
    if evaluate(<JOIN_CONDITION_1>, t, u) AND
       evaluate(<CONDITION>, t, u>) then
         output(t.<ATTRIBUTES>, u.<ATTRIBUTES>)
    endif;
  endforall;
endforall;
```

Join queries (ANSI SQL syntax)

Find the titles of all courses offered by a department chaired by `Peter`


Department
Department.chair = 'Peter'
name | code | total staff number | chair | budget
Department.name = Course.offered_by
Course
=
c# | title | credits | offered by

```
SELECT Course.title
FROM Course JOIN Department
ON Department.name = Course.offered_by
WHERE Department.chair = 'Peter';
```

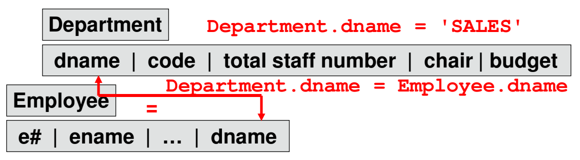
Computational model (join queries ANSI SQL)

```
SELECT <ATTRIBUTES>
FROM <TABLE_1> JOIN <TABLE_2>
ON <JOIN CONDITION>
WHERE <CONDITION>;
```

```
forall rows t in <TABLE_1>
  forall rows u in <TABLE_2>
    if evaluate(<JOIN_CONDITION_1>, t, u) AND
       evaluate(<CONDITION>, t, u>) then
         output(t.<ATTRIBUTES>, u.<ATTRIBUTES>)
    endif;
  endforall;
endforall;
```

Natural join queries (ANSI SQL syntax)

Find the names of all employees from 'SALES' department


Department
Department.dname = 'SALES'
dname | code | total staff number | chair | budget
Employee
Department.dname = Employee.dname
=
e# | ename | … | dname

```
SELECT ename
FROM Employee NATURAL JOIN Department
WHERE dname = 'SALES';
```

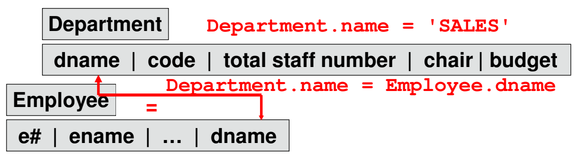
Computational model (natural join queries ANSI SQL)

```
SELECT <ATTRIBUTES>
FROM <TABLE_1> NATURAL JOIN <TABLE_2>
WHERE <CONDITION>;
```

```
forall rows t in <TABLE_1>
  forall rows u in <TABLE_2>
    if t.X = u.Y AND
       evaluate(<CONDITION>, t, u>) then
         output(t.<ATTRIBUTES>, u.<ATTRIBUTES>)
    endif;
  endforall;
endforall;
```

Column name join queries (ANSI SQL syntax)

Find the names of all employees from 'SALES' department


Department
Department.name = 'SALES'
dname | code | total staff number | chair | budget
Employee
Department.name = Employee.dname
=
e# | ename | … | dname

```
SELECT ename
FROM Employee JOIN Department
USING (dname)
WHERE dname = 'SALES';
```

Computational model (column join queries ANSI SQL)

```
SELECT <ATTRIBUTES>
FROM <TABLE_1> JOIN <TABLE_2>
USING (A,B)
WHERE <CONDITION>;
```

```
forall rows t in <TABLE_1>
  forall rows u in <TABLE_2>
    if t.A = u.A AND t.B = u.B
       evaluate(<CONDITION>, t, u>) then
         output(t.<ATTRIBUTES>, u.<ATTRIBUTES>)
    endif;
  endforall;
endforall;
```

Cross join queries (ANSI SQL syntax)

Find all pairs: (employee name, chair person name)

Department

dname | code | total staff number | chair | budget

Employee

e# | ename | … | dname

```
SELECT ename, chair
FROM Employee CROSS JOIN Department;
```

is equivalent to:

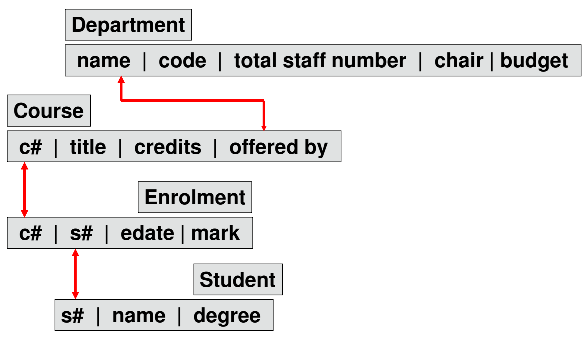
```
SELECT ename, chair
FROM Employee, Department;
```

Computational model (cross join queries)

```
SELECT <ATTRIBUTES>
FROM <TABLE_1>, <TABLE_2>;
```

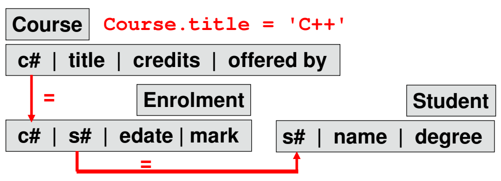
```
forall rows t in <TABLE_1>
  forall rows u in <TABLE_2>
     output(t.<ATTRIBUTES>, u.<ATTRIBUTES>);
  endforall;
endforall;
```

Sample database


Department
name | code | total staff number | chair | budget
Course
c# | title | credits | offered by
Enrolment
c# | s# | edate | mark
Student
s# | name | degree

Join queries

Find the names of all students who enrolled C++ course



Course.c# = Enrolment.c#

Enrolment.s# = Student.s#

Join queries

Find the names of all students who enrolled C++ course

```
SELECT Student.name
FROM Course, Enrolment, Student
WHERE Course.title = 'C++' AND
      Course.c# = Enrolment.c# AND
      Enrolment.s# = Student.s#;
```

```
SELECT name
FROM Course JOIN Enrolment
            ON Course.c# = Enrolment.c#
            JOIN Student
            ON Enrolment.s# = Student.s#
WHERE title = 'C++';
```

Natural join queries (ANSI SQL syntax)

Find the names of all students who enrolled C++ course

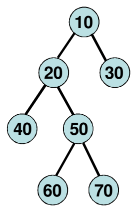
```
SELECT Student.name
FROM Course NATURAL JOIN Enrolment
            NATURAL JOIN Student
WHERE Course.title = 'C++';
```

Sample database

Employee

e# | name | manager#

```
e# | name  | manager#
---------------------
10 | John  | NULL
20 | Peter | 10
30 | Mary  | 10
40 | Mike  | 20
50 | Kate  | 20
60 | Greg  | 50
70 | Phil  | 50
```



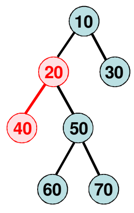
16. `SELECT` (3)

Self-join queries

Find a name of manager of employee no. `40`

```
Employee

e# | name   | manager#
----------------------
10 | John   | NULL
20 | Peter  | 10
30 | Mary   | 10
40 | Mike   | 20
50 | Kate   | 20
60 | Greg   | 50
70 | Phil   | 50
```



16. `SELECT` (3)

Self-join queries

Find a name of manager of employee no. `40`

```
Employee E1                  Employee E2

e# | name   | manager#       e# | name   | manager#
----------------------       ----------------------
10 | John   | NULL           10 | John   | NULL
20 | Peter  | 10             20 | Peter  | 10
30 | Mary   | 10             30 | Mary   | 10
40 | Mike   | 20             40 | Mike   | 20
50 | Kate   | 20             50 | Kate   | 20
60 | Greg   | 50             60 | Greg   | 50
70 | Phil   | 50             70 | Phil   | 50
```

`E1.e# = 40` `E1.manager# = E2.e#` `SELECT E2.name`

16. `SELECT` (3)

Self-join queries

Find a name of manager of employee no. `40`

`E1.e# = 40` `E1.manager# = E2.e#` `SELECT E2.name`

```
SELECT E2.name
FROM Employee E1 JOIN Employee E2
ON E1.manager# = E2.e#
WHERE E1.e# = 40;
```

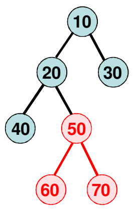
16. `SELECT` (3)

Self-join queries

Find the names of all employees directly managed by `Kate`

```
Employee

e# | name   | manager#
----------------------
10 | John   | NULL
20 | Peter  | 10
30 | Mary   | 10
40 | Mike   | 20
50 | Kate   | 20
60 | Greg   | 50
70 | Phil   | 50
```



16. `SELECT` (3)

Self-join queries

Find the names of all employees directly managed by `Kate`

```
Employee E1                  Employee E2

e# | name   | manager#       e# | name   | manager#
----------------------       ----------------------
10 | John   | NULL           10 | John   | NULL
20 | Peter  | 10             20 | Peter  | 10
30 | Mary   | 10             30 | Mary   | 10
40 | Mike   | 20             40 | Mike   | 20  SELECT
50 | Kate   | 20             50 | Kate   | 20  E2.name
60 | Greg   | 50             60 | Greg   | 50
70 | Phil   | 50             70 | Phil   | 50
```

`E1.name = 'Kate'` `E1.e# = E2.manager#`

16. `SELECT` (3)

Self-join queries

Find the names of all employees directly managed by `Kate`

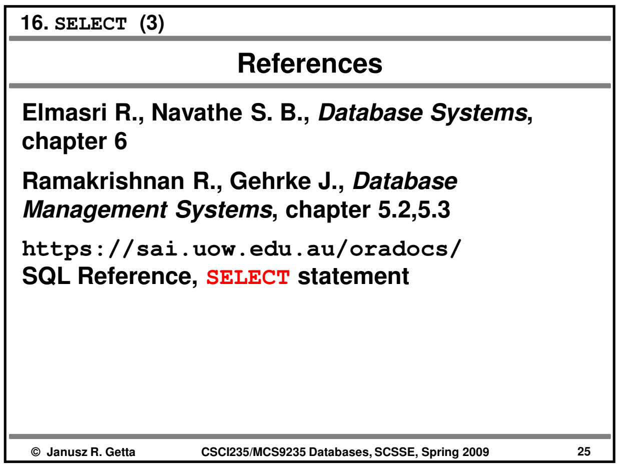
`E1.name = 'Kate'`

`E1.e# = E2.manager#`

`SELECT E2.name`

```
SELECT E2.name
FROM Employee E1 JOIN Employee E2
ON E1.e# = E2.manager#
WHERE E1.name = 'Kate';
```

Find the names of all employees managed by `Peter`???



16. `SELECT` (3)

References

Elmasri R., Navathe S. B., *Database Systems*, chapter 6

Ramakrishnan R., Gehrke J., *Database Management Systems*, chapter 5.2,5.3

`https://sai.uow.edu.au/oradocs/`
SQL Reference, `SELECT` statement