Database Management Systems

QUERYING DATA IN T-SQL

September 2019

Contents

1

BASIC QUERIES

SQL Query

Basic form:

```
SELECT <attributes>
FROM   <one or more relations>
WHERE  <conditions>
```

(SFW query)

Notes

SQL **commands** are case insensitive:

Same: SELECT, Select, select

Same: Product,   product

Values are **not:**

<u>Different:</u> ‘Seattle’, ‘seattle’

Use single quotes for constants:

‘abc’ - yes

“abc” - no

Selection (WHERE Clause)

ID	name	dept_name	salary
14365	Lembr	Accounting	32241.56
15347	Bawa	Athletics	72140.88
16807	Yazdi	Athletics	98333.65
19368	Wieland	Pol. Sci.	124651.41

```
SELECT *
FROM  instructor
WHERE dept_name = ‘Athletics';
```

ID	name	dept_name	salary
15347	Bawa	Athletics	72140.88
16807	Yazdi	Athletics	98333.65

LIKE: Simple String Pattern Matching

```
SELECT *
FROM   instructor
WHERE  name LIKE '_a[wz]%';
```

s LIKE p: pattern matching on strings

p may contain two special symbols:

- %   any sequence of characters
- _   any single character
- [ ]   a single character from the list
- [-] a single character within given range
- [^] a single character not listed after ^

BETWEEN, IN

```
SELECT *
FROM instructor
WHERE salary BETWEEN 75000 AND
80000;
```

name	salary
Katz	75000.00
Singh	80000.00
Kim	80000.00

```
SELECT *
FROM instructor
WHERE left(name,1) IN ('C','E');
```

name	dept_name
Einstein	Physics
El Said	History
Califieri	History
Crick	Biology

Projection (SELECT Clause)

ID	name	dept_name	salary
14365	Lembr	Accounting	32241.56
15347	Bawa	Athletics	72140.88
16807	Yazdi	Athletics	98333.65
19368	Wieland	Pol. Sci.	124651.41

```
SELECT ID, name, salary
FROM  instructor
WHERE dept_name = 'athletics';
```

ID	name	salary
15347	Bawa	72140.88
16807	Yazdi	98333.65

Literals

Attributes in select clause can be literals (e.g. numbers)

- Without from clause: single row
- With from clause: *N* rows

```
SELECT 25;
```

25

```
SELECT 25
FROM instructor;
```

25
25
25
25
25
25
25
25
25

Rename

ID	name	dept_name	salary
14365	Lembr	Accounting	32241.56
15347	Bawa	Athletics	72140.88
16807	Yazdi	Athletics	98333.65
19368	Wieland	Pol. Sci.	124651.41

```
SELECT name AS Instructor, salary [Annual Salary]
FROM instructor;
```

Instructor	Annual Salary
Lembr	32241.56
Bawa	72140.88
Yazdi	98333.65
Wieland	124651.41

- AS is optional
- [ ] for column name with special characters

And More …

ID	name	dept_name	salary
14365	Lembr	Accounting	32241.56
15347	Bawa	Athletics	72140.88
16807	Yazdi	Athletics	98333.65
19368	Wieland	Pol. Sci.	124651.41

```
SELECT ID, name [Instructor Name], salary/12 [Annual Salary]
FROM instructor;
```

ID	Instructor Name	Annual Salary
14365	Lembr	2686.796666
15347	Bawa	6011.740000
16807	Yazdi	8194.470833
19368	Wieland	10387.617500

- Arithmetic expressions
- String expressions
- Functions

DISTINCT: Eliminating Duplicates

```
SELECT DISTINCT dept_name
FROM   instructor;
```

dept_name
Accounting
Athletics
Biology

Versus

```
SELECT dept_name
FROM   instructor;
```

dept_name
Accounting
Athletics
Athletics
Pol. Sci.

ORDER BY: Sorting the Results

```
SELECT   name, dept_name, salary
FROM     instructor
WHERE    salary > 50000 AND salary < 80000
ORDER BY dept_name, salary DESC
```

Ordering is **ascending**, unless you specify the **DESC** keyword.

name	dept_name	salary
Moreira	Accounting	71351.42
Romero	Astronomy	79070.08
Bawa	Athletics	72140.88
Murata	Athletics	61387.56

2

JOIN QUERIES

Cartesian Product (FROM Clause)

```
SELECT i.ID, i.name, i.salary, d.ID, d.name, d.salary
FROM instructor i, instructor d
```

ID	name	salary	ID_1	name_1	salary_1
10101	Srinivasan	65000.00	10101	Srinivasan	65000.00
12121	Wu	90000.00	10101	Srinivasan	65000.00
15151	Mozart	40000.00	10101	Srinivasan	65000.00
22222	Einstein	95000.00	10101	Srinivasan	65000.00
…	…	…	…	…	…

Theta Joins

```
SELECT distinct i.salary
FROM instructor i INNER JOIN instructor d ON
    i.salary < d.salary;
```

INNER is optional

OR

```
SELECT distinct i.salary
FROM instructor i, instructor d
WHERE i.salary < d.salary;
```

salary
40000.00
60000.00
62000.00
65000.00
72000.00
75000.00
80000.00
87000.00
90000.00
92000.00

Equi Joins

```
SELECT i.ID, i.name, i.dept_name, t.course_id
FROM instructor i JOIN teaches t ON i.ID = t.ID
WHERE i.dept_name = 'Comp. Sci.';
```

OR

```
SELECT i.ID, i.name, i.dept_name, t.course_id
FROM instructor i, teaches t
WHERE i.ID = t.ID and i.dept_name = 'Comp. Sci.';
```

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
10101	Srinivasan	Comp. Sci.	CS-315
10101	Srinivasan	Comp. Sci.	CS-347

Outer Joins

```
SELECT i.ID, i.name, i.dept_name, t.course_id
FROM instructor i LEFT OUTER JOIN teaches t
    on i.ID = t.ID
WHERE i.dept_name = 'Physics';
```

ID	name	dept_name	course_id
22222	Einstein	Physics	PHY-101
33456	Gold	Physics	None

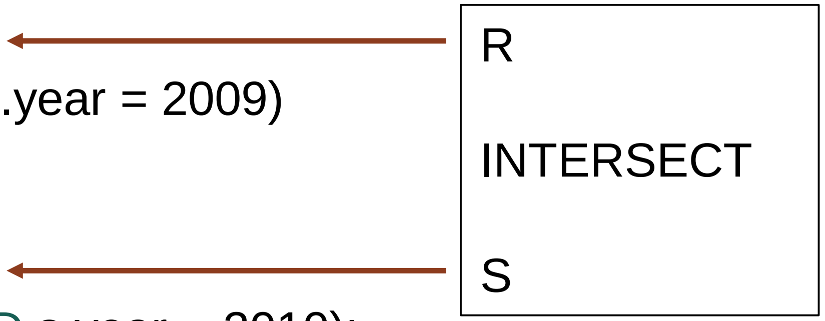
3

SET OPERATIONS

INTERSECT

Courses taught in Fall 2009 and Spring 2010

```
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Fall' AND s.year = 2009)
INTERSECT
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Spring' AND s.year = 2010);
```

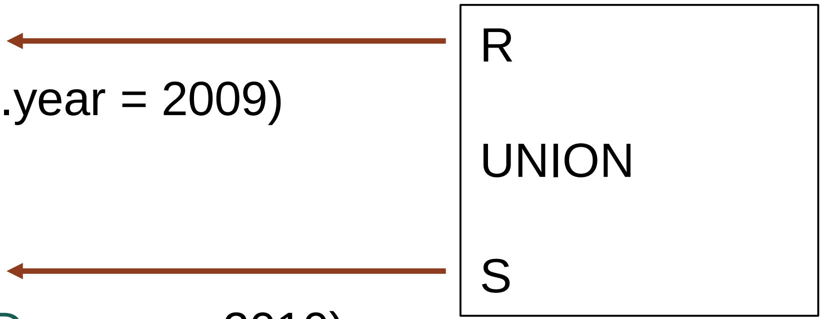


course_id
CS-101

UNION

Courses taught in Fall 2009 OR Spring 2010

```
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Fall' AND s.year = 2009)
UNION
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Spring' AND s.year = 2010);
```



UNION ALL

Courses taught in Fall 2009 OR Spring 2010 (with duplicates)

```
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Fall' AND s.year = 2009)
UNION ALL
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Spring' AND s.year = 2010);
```

R

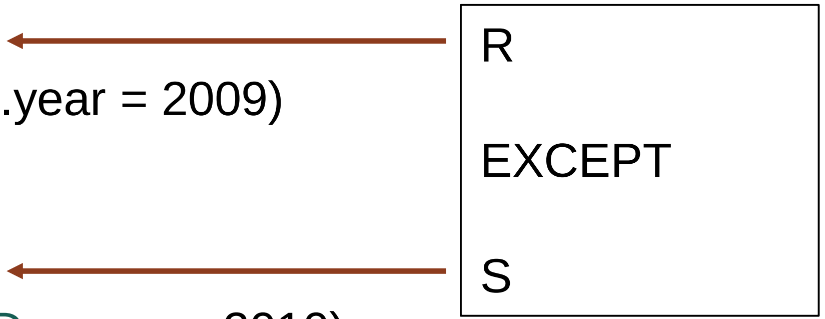
UNION (Multiset operation)

S

EXCEPT

Courses taught in Fall 2009
but NOT in Spring 2010

```
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Fall' AND s.year = 2009)
EXCEPT
(SELECT distinct course_id
FROM section s
WHERE s.semester = 'Spring' AND s.year = 2010);
```



course_id
CS-347
PHY-101

4

NESTED QUERIES

Nested Queries

SELECT $A_1, A_2, ..., A_n$
    FROM $r_1, r_2, ..., r_m$
    WHERE P

- A_i can be replaced be a subquery that generates a single value.
  - Scalar subquery
- r_i can be replaced by any valid subquery
  - Subquery as derived table
- P can be replaced with an expression of the form:
  attribute <operation> (subquery)
  - Test for set membership
  - Set comparison
  - Test for empty relations

Test for Set Membership

Courses taught in Fall 2009 and Spring 2010

```
SELECT distinct course_id
FROM section
WHERE semester = 'Fall' and year = 2009
    and course_id IN (
        SELECT course_id
        FROM section
        WHERE semester = 'Spring' and year = 2010
    );
```

R

INTERSECT

S

course_id
CS-101

Test for Set Membership

Courses taught in Fall 2009
but not in Spring 2010

```
SELECT distinct course_id
FROM section
WHERE semester = 'Fall' and year = 2009
    and course_id NOT IN (
        SELECT course_id
        FROM section
        WHERE semester = 'Spring' and year = 2010
    );
```

```
R

EXCEPT

S
```

course_id
CS-101

Sets Comparison (SOME/ANY, ALL)

```
SELECT name
FROM instructor
WHERE salary > SOME(
    SELECT salary
    FROM instructor
    WHERE dept_name = 'Biology‘
);
```

Instructors with salary greater than that of some (at least one) instructor in the Biology department.

name
Wu
Einstein
Gold
Katz
Singh
…

- F <comp> **some** $r \Leftrightarrow \exists\, t \in r$ such that (F <comp> t)
- F <comp> **all** $r \Leftrightarrow \forall\, t \in r$ (F <comp> t)

Test for Empty Relations

- **exists** $r \Leftrightarrow r \neq \emptyset$
- **not exists** $r \Leftrightarrow r = \emptyset$

All students who have taken all courses offered in the Biology department.

```
SELECT distinct s.ID, s.name
FROM student s                    ← Correlation name
WHERE NOT EXISTS(
   (SELECT course_id
    FROM course
    WHERE dept_name = 'Biology')
   EXCEPT
   (SELECT t.course_id
    FROM takes t                  ← Correlated subquery
    WHERE t.ID = s.ID)
);
```

All courses offered by Biology dept. (X) →

$X - Y = \emptyset \quad \Leftrightarrow \quad X \subseteq Y$

All courses taken by student s_i. (Y) →

Subqueries in From Clause (Derived Tables)

Find the average instructors' salaries of those departments where the average salary is greater than $42,000."

```
SELECT r.dept_name, r.avg_salary
FROM (
        SELECT dept_name, avg(salary) avg_salary
        FROM instructor
        GROUP BY dept_name) r
WHERE r.avg_salary > 42000;
```

Name assigned to the subquery

dept_ name	**avg_ salary**
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Physics	91000

WITH Clause (Common Table Expressions for Derived Tables)

Find all departments with the maximum budget

```
WITH max_budget (value) AS(
    SELECT max(budget)
    FROM department
)
SELECT d.dept_name
FROM department d, max_budget m
WHERE d.budget = m.value;
```

CTE

dept_name
Finance

Scalar Subquery in Select Clause

All departments along with the number of instructors in each department

```
SELECT d.dept_name, (
    SELECT count(*)
    FROM instructor i
    WHERE i.dept_name = d.dept_name
    ) [Number of Instructors]
FROM department d;
```

dept_name	**No_ Instructors**
Biology	1
Comp. Sci.	3
Elec. Eng.	1
Finance	2
History	2
Music	1
Physics	2

4

AGGREGATION & GROUP-BY

Aggregation

```
SELECT AVG(salary) avg_salary
FROM instructor
WHERE dept_name = 'Comp. Sci.';
```

```
SELECT COUNT(*)
FROM instructor
WHERE dept_name = 'Comp. Sci.';
```

SQL supports several **aggregation** operations:

- SUM, COUNT, MIN, MAX, AVG

Except COUNT, all aggregations apply to a single attribute

Aggregation: COUNT

COUNT applies to duplicates, unless otherwise stated

```
SELECT COUNT(dept_name) No_Dept
FROM instructor;
```

No_Dept
12

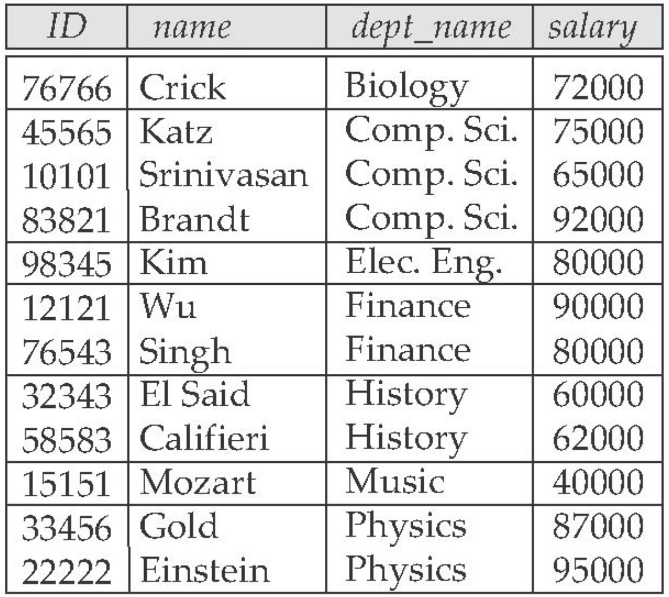
We probably want:

```
SELECT COUNT(DISTINCT dept_name) No_Dept
FROM instructor;
```

No_Dept
7

Grouping & Aggregation

```
SELECT dept_name, AVG(salary)  salary
FROM instructor
GROUP BY dept_name;
```



ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	avg_salary
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

HAVING Clause

```
SELECT dept_name, AVG(salary)  salary
FROM instructor
GROUP BY dept_name
HAVING AVG(salary) > 42000
ORDER BY avg_salary DESC;
```

HAVING clauses contains conditions on **aggregates**

*Whereas WHERE clauses condition on **individual tuples…***

dept_ name	avg_ salary
Physics	91000
Finance	85000
Elec. Eng.	80000
Comp. Sci.	77333
Biology	72000
History	61000

General Form of Grouping and Aggregation

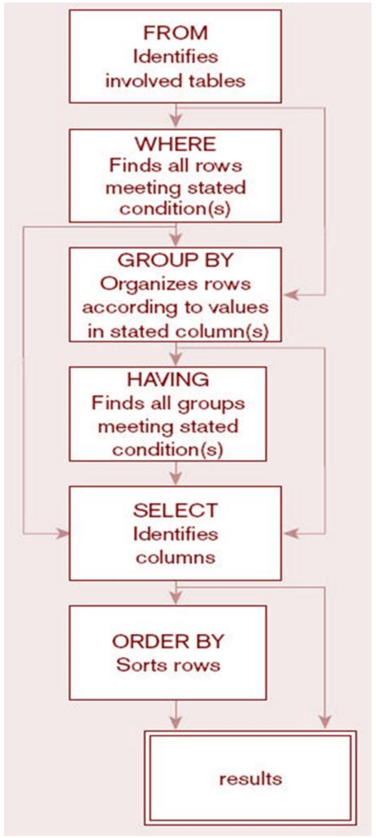
```
SELECT     S
FROM       R_1,…,R_n
WHERE      C_1
GROUP BY   a_1,…,a_k
HAVING     C_2
```

S = Can ONLY contain attributes $a_1,\ldots,a_k$ and/or aggregates over other attributes

C_1 = is any condition on the attributes in $R_1,\ldots,R_n$

C_2 = is any condition on the aggregate expressions

Query processing order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
GROUP BY
Organizes rows according to values in stated column(s)
HAVING
Finds all groups meeting stated condition(s)
SELECT
Identifies columns
ORDER BY
Sorts rows
results