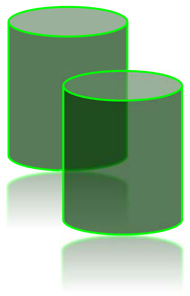


CZ2007
Introduction to Databases

Querying Relational Databases using SQL
Part-2

Arijit Khan

Assistant Professor
School of Computer Science and Engineering
Nanyang Technological University, Singapore

arijit.khan@ntu.edu.sg

Schedule after Recess Week

SQL

8 Lectures
- Week 8 (Oct 07-Oct 11)
- Week 9 (Oct 14-Oct 18)
- Week 10 (Oct 21-Oct 25)
- Week 11 (Oct 28-Nov 01)

Semi-Structured Data, Quiz-2

2 Lectures
- Week 12 (Nov 02-Nov 08)
- Quiz during Tutorial session
- Quiz syllabus: everything on SQL (Week 8, 9, 10 11)

Summary

- Week 13 (Nov 11-Nov 15)

Recap: Best Practice

- Run your query in the Lab (they usually provide MySQL?)

- (It may not compile, but might still be correct!)
  Always check in **Google**

- Consult with the **Book** and course material
  - Database Systems: The Complete Book; Hector Garcia-Molina Jeffrey D. Ullman, Jennifer Widom
  - (Book available online)

Recap: Roadmap (SQL)

- Introduction to SQL
- Querying single relation

✓ Lecture-1

- **Ordering Tuples**
- **Multi-relation queries**
- **Subqueries**

Lecture-2

Today's lecture: Chapter 6.2, 6.3 of the Book "Database Systems: The Complete Book; Hector Garcia-Molina Jeffrey D. Ullman, Jennifer Widom

- Set operations
- Bag semantics
- Join expressions
- Aggregation

Lectures-3 & 4

Recap: Roadmap (SQL)

- Groupings
- Creation of tables
- Database modifications
- Constraints
- Views

Lecture-5 & 6

- Triggers
- Indexes

Lecture-7 & 8

That would be all about Quiz-2!!

Today's Lecture

- Ordering Tuples
- Multi-relation queries
- Subqueries

Study-at-Home slides at the end of every lecture

- They will be in the syllabus of Quiz-2 and Final Exam
- More examples
- Study them at home, will be discussed at the beginning of next lecture
- If any questions, ask me !!

Questions?

The important thing is not to stop questioning.

Albert Einstein

ORDER BY: Sorting the Results

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

```
SELECT PName, Price, Manufacturer
FROM   Product
WHERE  Category = 'Gadgets' AND Price < 50

ORDER BY Price, PName
```

PName	Price	Manufacturer
Gizmo	19.99	GizmoWorks
Powergizmo	29.99	GizmoWorks

ORDER BY: Sorting the Results

Product

PName	Price	Category	Manufacturer
Gizmo	19.99	Gadgets	GizmoWorks
Powergizmo	29.99	Gadgets	GizmoWorks
SingleTouch	149.99	Photography	Canon
MultiTouch	203.99	Household	Hitachi

```
SELECT PName, Price, Manufacturer
FROM   Product
WHERE  Category = 'Gadgets' AND Price < 50

ORDER BY Price, PName
```

PName	Price	Manufacturer
Gizmo	19.99	GizmoWorks
Powergizmo	29.99	GizmoWorks

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

- Ties are broken by the second attribute on the ORDER BY list, etc.

ORDER BY: Sorting the Results

What about NULL?

NULL is normally treated as less than all non-null values.

- Ordering is ascending, unless you specify the DESC keyword.
- Ties are broken by the second attribute on the ORDER BY list, etc.

Multi-Relation Queries

Sailors

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Reserves

Boat reserved by sailors

sid	bid	day
1	102	9/12
2	102	9/13

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND R.bid=102
```

sname
Fred
Jim

Multi-Relation Queries

Boat reserved by sailors

Sailors

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Reserves

sid	bid	day
1	102	9/12
2	102	9/13

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND R.bid=102
```

```
SELECT S.sname
FROM   Sailors AS S, Reserves AS R
WHERE  S.sid=R.sid AND R.bid=102
```

Both are OKAY (Semantically)

sname
Fred
Jim

Multi-Relation Queries

Boat reserved by sailors

Sailors

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Reserves

sid	bid	day
1	102	9/12
2	102	9/13

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND R.bid=102
```

sname
Fred
Jim

Also OKAY (Semantically)

```
SELECT sname
FROM   Sailors, Reserves
WHERE  Sailors.sid=Reserves.sid AND bid=102
```

Multi-Relation Queries

Sailors

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Boat reserved by sailors

Reserves

sid	bid	day
1	102	9/12
2	102	9/13

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND R.bid=102
```

sname
Fred
Jim

```
SELECT Sailors.sname
FROM   Sailors, Reserves
WHERE  Sailors.sid=Reserves.sid
AND Reserves.bid=102
```

Also OKAY (Semantically)

Multi-Relation Queries (On Same Table)

Sailors

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Find pairs of sailor names both of whose ages are below 30

S1.sname	S2.sname
Fred	Nancy

Multi-Relation Queries (On Same Table)

Sailors AS S1

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Sailors AS S2

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Find pairs of sailor names both of whose ages are below 30

```
SELECT S1.sname, S2.sname

FROM   Sailors S1, Sailor S2

WHERE  S1.age<30 AND S2.age<30

AND S1.sname < S2.sname
```

S1.sname	S2.sname
Fred	Nancy

Multi-Relation Queries (On Same Table)

Sailors AS S1

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Sailors AS S2

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

Find pairs of sailor names both of whose ages are below 30

```
SELECT S1.sname, S2.sname
FROM   Sailors S1, Sailor S2
WHERE  S1.age<30 AND S2.age<30
AND S1.sname < S2.sname
```

String comparison to avoid duplicates

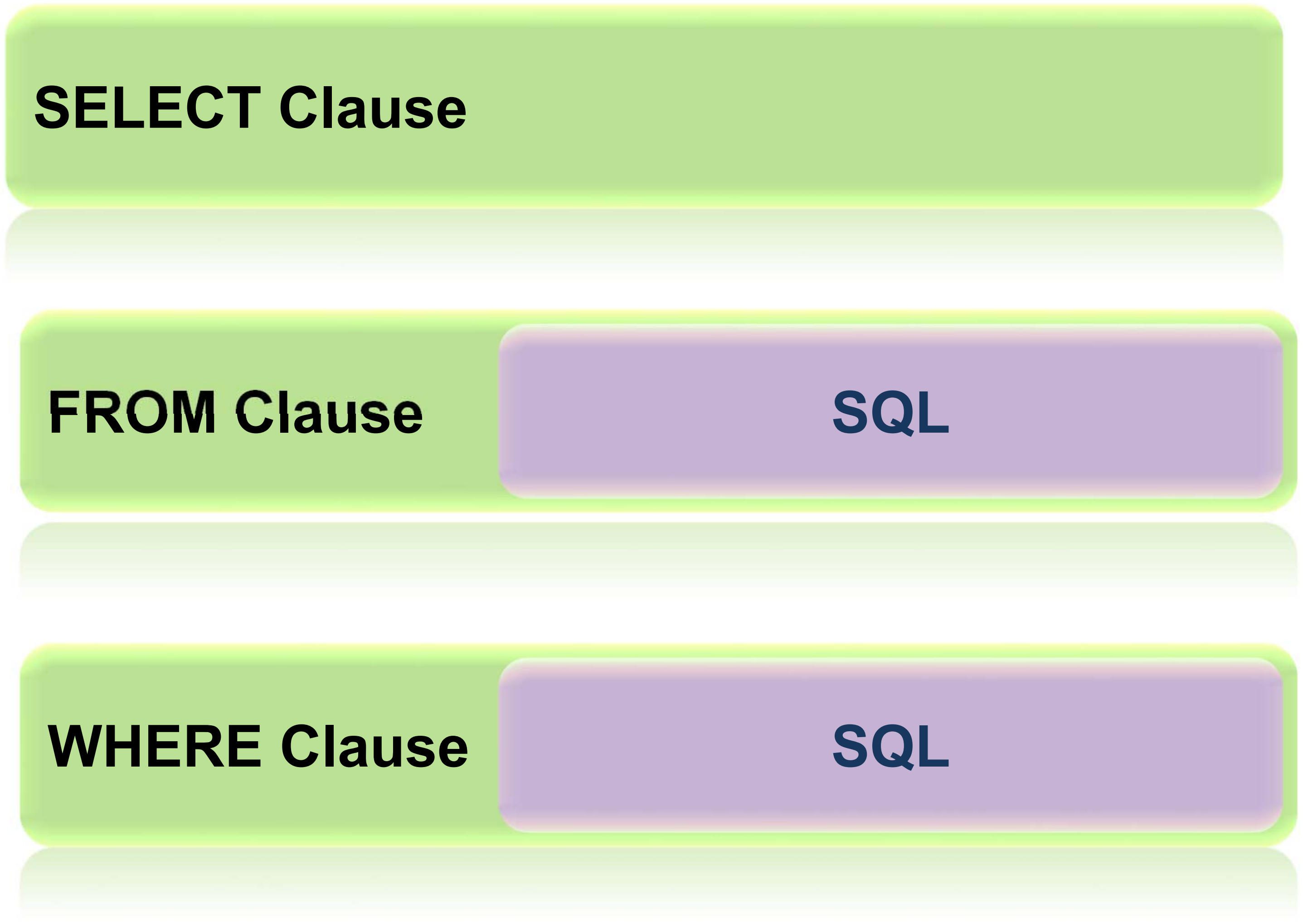
S1.sname	S2.sname
Fred	Nancy

Questions?

Today's Lecture

- Ordering Tuples ✓
- Multi-relation queries ✓
- **Subqueries**

Subqueries



Subqueries

- Also called nested queries
- We can do nested queries because SQL is compositional:

    - Everything (inputs / outputs) is represented as multisets- the output of one query can thus be used as the input to another (nesting)!

- This is extremely powerful!

Types of Subqueries

Scalar Subquery

- returns a single value which is then used in a comparison.
- if query expects a single value from a subquery, and it returns multiple values or no values, a run-time error occurs.

Row Subquery

- returns a single row which may have multiple columns

Table Subquery

- returns one or more columns and multiple rows.

Scalar Subquery

Query

From Sells(bar, beer, price), find the bars that serve Heineken for the same price WOOBAR charges for Bud.

```
SELECT    bar
FROM      Sells
WHERE     beer = `Heineken'
          AND price = [price of Bud @ WOOBAR];
```

Subqueries

- Find the price WOOBAR charges for Bud.
- Find the bars that serve Heineken at that price.

Scalar Subquery

Sells

Bar	Beer	Price
Southbridge	Heineken	7.90
Southbridge	Bud	6.60
WOOBAR	Bud	7.90
WOOBAR	Heineken	8.10
Emerald Hill	Heineken	8.00

Price
7.90

```
SELECT      price
FROM        Sells
WHERE       bar = `WOOBAR'
            AND beer = `Bud';
```

Scalar Subquery

Sells

Bar	Beer	Price
Southbridge	Heineken	7.90
Southbridge	Bud	6.60
WOOBAR	Bud	7.90
WOOBAR	Heineken	8.10
Emerald Hill	Heineken	8.00

Bar
Southbridge

```
SELECT      price
FROM        Sells
WHERE       bar = `W
            AND be
```

```
SELECT      bar
FROM        Sells
WHERE       beer = `Heineken’
            AND price = 7.90;
```

Scalar Subquery

```
SELECT      bar
FROM        Sells
WHERE       beer = ‘Heineken’ AND
            price = ( SELECT price
                      FROM   Sells
                      WHERE bar = ‘WOOBAR’
                      AND beer = ‘Bud’);
```

Without using Scalar Subquery?

```
SELECT      S1.bar
FROM        Sells S1, Sells S2
WHERE       S1.beer = ‘Heineken’
AND         S2.bar = ‘WOOBAR’
AND         S2.beer = ‘Bud’
AND         S1.price = S2.price;
```

Use two copies of the table

Row Subquery

Row Subquery

- returns a single row which may have multiple columns

Operators in Row Subquery

IN

<tuple> IN <relation> is true if and only if the tuple is a member of the relation.

Row Subquery

Row Subquery

- returns a single row which may have multiple columns

Operators in Row Subquery

ALL

x <> ALL(<relation>) is true if and only if for every tuple *t* in the relation, *x* is not equal to *t*.

Row Subquery

Row Subquery

- returns a single row which may have multiple columns

Operators in Row Subquery

ANY/SOME

- *x* = SOME(<relation>) is a Boolean condition. Meaning that *x* equals at least one tuple in the relation.
- “Equal to at least one”
- Early version of SQL allowed ANY

“IN” - Row Subquery

IN

<tuple> IN <relation> is true if and only if the tuple is a member of the relation.

Query

From Beers(name, manf) and Likes(drinker, beer), find the name and manufacturer of each beer that Fred likes.

```
SELECT      *
FROM        Beers
WHERE       name IN [What Fred likes]
```

"IN" - Row Subquery

IN

<tuple> IN <relation> is true if and only if the tuple is a member of the relation.

Query

From Beers(name, manf) and Likes(drinker, beer), find the name and manufacturer of each beer that Fred likes.

```
SELECT      *
FROM        Beers
WHERE       name IN ( SELECT        beer
                      FROM          Likes
                      WHERE         drinker = `Fred');
```

Without using Row Subquery?

Query

From Beers(name, manf) and Likes(drinker, beer), find the name and manufacturer of each beer that Fred likes.

```
SELECT      name, manf
FROM        Beers, Likes
WHERE       name = beer
AND         drinker = `Fred';
```

"ALL" - Row Subquery

ALL

x <> ALL(<relation>) is true if and only if for every tuple *t* in the relation, *x* is not equal to *t*.

Query

From Sells(<u>bar</u>, <u>beer</u>, price), find the beer(s) sold for the highest price.

```
SELECT    beer
FROM      Sells
WHERE     price = [highest price];
```

How to find highest price?

"ALL" - Row Subquery

ALL

x <> ALL(<relation>) is true if and only if for every tuple *t* in the relation, *x* is not equal to *t*.

Query

From Sells(bar, beer, price), find the beer(s) sold for the highest price.

```
SELECT      beer
FROM        Sells
WHERE       price >= ALL ( SELECT     price
                           FROM       Sells );
```

"SOME" - Row Subquery

ANY/SOME

- *x* = SOME(<relation>) is a boolean cond. Meaning that *x* equals at least one tuple in the relation.
- "Equal to at least one"
- Early version of SQL allowed ANY

Query

From Agents(agent_code, agent_name), Customer(agent_code, cust_country), report all agents who belong to the country 'UK'.

"SOME" - Row Subquery

ANY/SOME

- *x* = SOME(<relation>) is a boolean cond. Meaning that *x* equals at least one tuple in the relation.
- "Equal to at least one"
- Early version of SQL allowed ANY

Query

From Agents(agent_code, agent_name), Customer(agent_code, cust_country), report all agents who belong to the country 'UK'.

```
SELECT  agent_code, agent_name
FROM    Agents
WHERE agent_code = SOME (
                SELECT agent_code FROM Customer
                WHERE cust_country = 'UK');
```

More Operators for Subquery

- Any of the comparison operators (<, <=, =, etc.) can be used.

- The keyword NOT can proceed any of the operators (*s* NOT IN *R*)

Table Subquery

Table Subquery

- returns one or more columns and multiple rows.

Operators in Table Subquery: Exists/ No Exists

Table Subquery

Table Subquery

- returns one or more columns and multiple rows.

Operators in Table Subquery: Exists/ No Exists

Product(name, price, category, maker)

```
SELECT  p1.name
FROM     Product p1
WHERE   p1.maker = 'Gizmo-Works'
 AND EXISTS(
              SELECT *
              FROM    Product p2
              WHERE  p2.maker <> 'Gizmo-Works'
                AND      p1.name = p2.name)
```

Find products made by “Gizmo-Works” having the same names as products made by other makers

Questions?

Summary

- Ordering Tuples ✓
- Multi-relation queries ✓
- Subqueries ✓

Study-at-Home

Correlated and uncorrelated subqueries (Slides 42-44)

Will be in the syllabus of Quiz-2 and Final Exam

Uncorrelated Subqueries

```
SELECT      *
FROM        Beers
WHERE       name IN ( SELECT         beer
                      FROM           Likes
                      WHERE          drinker = `Fred’);
```

- Subquery is not related to the outer query

Correlated Subqueries

```
SELECT  p1.name
FROM     Product p1
WHERE   p1.maker = ‘Gizmo-Works’
 AND EXISTS(
             SELECT *
             FROM     Product p2
             WHERE  p2.maker <> ‘Gizmo-Works’
                AND     p1.name = p2.name)
```

•A subquery is **correlated** with the outer query if it contains a reference to an attribute in the outer query.

•A subquery is ***correlated*** with the outside query if it must be re-computed for every tuple produced by the outside query.

Subquery – Rules to Remember

• The ORDER BY clause may not be used in a subquery.

• The number of attributes in the SELECT clause in the subquery must match the number of attributes with the comparison operator.

• Column names in a subquery refer to the table name in the FROM clause of the subquery by default.

• When the result of a subquery is used as an operand, it must be the right operand.

NANYANG TECHNOLOGICAL UNIVERSITY

Questions ??

Thank You !

arijit.khan@ntu.edu.sg