SQL: Advanced Structured Query Language - Review

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

Some slides are adopted from D. Maier, S. Davidson and A. Parameswaran

Leaning Objectives

After this lecture, you should be able to:

- Write **Multi-relation** and **JOIN** SQL queries
- Write advanced queries with **subqueries** in the **FROM**, **WHERE** and **SELECT** clauses
- Write advanced queries with Boolean operators in the **WHERE** clause (IN, Any, All, Exists)

Example of Multi-Relation Query

```
SELECT    A.Owner, A.Balance
FROM      Account A, Deposit D
WHERE     D.AcctNo = A.Number and A.Balance > 1000;
```

"A" is a correlation name for Account

and

"D" is a correlation name for Deposit.

Correlation names are like local variables – they hold one tuple or row from the corresponding table.

You choose correlation names when you write the query.

Exercise 1

Q1 - What would be the result of this query?

```
SELECT *
FROM Account A, Deposit D
WHERE A.Balance >= 1000
AND   D.Amount <= 10000;
```

Account

Number	Owner	Balance	Type
101	J. Smith	1000.00	checking
102	W. Wei	2000.00	checking
103	J. Smith	5000.00	savings
104	M. Jones	1000.00	checking
105	H. Martin	10,000.00	checking

Deposit

AcctNo	T-id	Date	Amount
102	1	10/22/00	500.00
102	2	10/29/00	200.00
104	3	10/29/00	1000.00
105	4	11/02/00	10,000.00

A. Empty set

B. The combination of every row from Account and Deposit.

C. All rows of Account followed by all rows of Deposit

D. All rows of Deposit followed by all rows of Account

E. None of the above

Inner Joins

- Cross Product:
  - Include the combination of every row from input tables
- Equi-join (theta-join):
  - Include input tables rows that match the join (theta) condition
- Natural join:
  - Same as equi-join but doesn't need a condition.
  - Joins based on columns with the same name
  - Outputs only one copy of the joining columns

Outer Joins

- Left outer join:
  - Include the left tuple even if there's no match
- Right outer join:
  - Include the right tuple even if there's no match
- Full outer join:
  - Include both the left and right tuples even if there's no match

Exercise 2

R

Col1
Null
Null

S

Col1
Null
Null

What is the answer of the following query

```
SELECT *
FROM R Natural Join S
```

A

Col1

B

Col1
Null
Null

C

Col1
Null
Null
Null

D

Col1
Null
Null
Null
Null

Exercise 3

R

Col1
Null
Null

S

Col1
Null
Null

What is the answer of the following query

```
SELECT *
FROM R Left Outer Join S on R.Col1 = S.Col1
```

A

R.Col1	S.Col1

B

R.Col1	S.Col1
Null	Null
Null	Null

C

R.Col1	S.Col1
Null	Null
Null	Null
Null	Null

D

R.Col1	S.Col1
Null	Null
Null	Null
Null	Null
Null	Null

Exercise 4

R

Col1
Null
Null

S

Col1
Null
Null

What is the answer of the following query

```
SELECT *
FROM R Full Outer Join S on R.Col1 = S.Col1
```

A

R.Col1	S.Col1

B

R.Col1	S.Col1
Null	Null
Null	Null

C

R.Col1	S.Col1
Null	Null
Null	Null
Null	Null

D

R.Col1	S.Col1
Null	Null
Null	Null
Null	Null
Null	Null

Comments on Queries

Because the answer to a relational query is always a table

- ✓ we can use the answer from one query as input to another query.
- ✓ This means that we can create arbitrarily complex queries!
- ✓ A query language is **closed** if it has this property.

Subqueries

- A parenthesized SELECT-FROM-WHERE statement (*subquery*) can be used as a value in a number of places, including FROM and WHERE clauses.

Boolean Operators

- <tuple> IN <relation> is true if and only if the tuple is a member of the relation.
- EXISTS(<relation>) is true if and only if the <relation> is not empty.
- *x* = ANY(<relation>) is a boolean condition meaning that *x* equals at least one tuple in the relation.
- *x* <> ALL(<relation>) is true if and only if for every tuple *t* in the relation, *x* is not equal to *t*.