SQL: Structured Query Language

Abdu Alawini

University of Illinois at Urbana-Champaign

CS411: Database Systems

June 15, 2020

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

Leaning Objectives

After this lecture, you should be able to:

- Write **single-relation SQL** queries
- Write SQL queries with **complex WHERE conditions**
- Write **Multi-relation** and **JOIN** SQL queries
- Write advanced queries with **subqueries** in the **FROM**, **WHERE** and **SELECT** clauses
- Write advanced queries with **Set Operators**

Outline

- Multi-Relation SQL Queries
- Join SQL Queries
- Advanced SQL
  - Subqueries
  - Boolean Operators (IN, ANY, EXISTS, ALL)
  - Set operations

Multi-relation Queries

- Interesting queries often combine data from more than one relation.
- We can address several relations in one query by listing them all in the FROM clause.
- Distinguish attributes of the same name by "<relation>.<attribute>"

Example of Multi-Relation Query

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

How does this work?

Which rows, from which tables,

are evaluated in the WHERE clause?

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.

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

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

We must check every combination of one row from Account with one row from Deposit.

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

notice The columns

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount

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

Yes! Place in query answer.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00

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

Yes! Place in query answer.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

All combinations fail!

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

No! Skip it.

(Why?)

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

No! Skip it.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

The first three fail.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00

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

Yes! Place in query answer.

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

WHERE D.AcctNo = A.Number and A.Balance > 1000;

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
105	H. Martin	10,000.00	checking	105	4	11/02/00	10,000.00

Intermediate result

(after processing the FROM & WHERE clauses)

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
105	H. Martin	10,000.00	checking	105	4	11/02/00	10,000.00

Process the SELECT

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

Final query answer: (notice that W. Wei appears twice)

Owner	Balance
W. Wei	2000.00
W. Wei	2000.00
H. Martin	10,000.00

Intermediate result

(after processing the FROM & WHERE clauses)

Number	Owner	Balance	Type	AcctNo	T-id	Date	Amount
102	W. Wei	2000.00	checking	102	1	10/22/00	500.00
102	W. Wei	2000.00	checking	102	2	10/29/00	200.00
105	H. Martin	10,000.00	checking	105	4	11/02/00	10,000.00

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

If we use the word DISTINCT, then duplicates are removed from the query answer. W. Wei only appears once.

Owner	Balance
W. Wei	2000.00
H. Martin	10,000.00

Operational Semantics of Multi-relation queries

Almost the same as for single-relation queries:

1. Start with the product of all the relations in the FROM clause.
2. Apply the selection condition from the WHERE clause.
3. Project onto the list of attributes and expressions in the SELECT clause.

Queries

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

Notice that a query is Expressed against the schema.

But the query runs or executes against the instance (the data)

And may give different answers on different instances

```
SELECT  Owner
FROM    Account
WHERE Type = "checking";
```

	Owner
	J. Smith
	W. Wei
	M. Jones
	H. Martin

Comments on Queries

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

Notice that the answer to a query is always a **table**!

It doesn't always have a name (for the table).

The attribute names are deduced from the input tables (or supplied by the query author). It might or Might not have any rows.

	Owner
	J. Smith
	W. Wei
	M. Jones
	H. Martin

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.

Outline

- ✓ Multi-Relation SQL Queries
- Join SQL Queries
- Advanced SQL
  - Subqueries
  - Boolean Operators (IN, ANY, EXISTS, ALL)
  - Set operations

Example Database

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Cross Product

Department × Employee

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

```
SELECT *
FROM Department, Employee
```

ILLINOIS

Department.DepartmentName	Department.DepartmentID	Employee.LastName	Employee.DepartmentID
Sales	31	Rafferty	31
Sales	31	Jones	33
Sales	31	Steinberg	33
Sales	31	Smith	34
Sales	31	Robinson	34
Sales	31	John	NULL
Engineering	33	Rafferty	31
Engineering	33	Jones	33
Engineering	33	Steinberg	33
Engineering	33	Smith	34
Engineering	33	Robinson	34
Engineering	33	John	NULL
Clerical	34	Rafferty	31
Clerical	34	Jones	33
Clerical	34	Steinberg	33
Clerical	34	Smith	34
Clerical	34	Robinson	34
Clerical	34	John	NULL
Marketing	35	Rafferty	31
Marketing	35	Jones	33
Marketing	35	Steinberg	33
Marketing	35	Smith	34
Marketing	35	Robinson	34
Marketing	35	John	NULL

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Equijoin

Employee ⋈ Department

Employee.DeptID = Department.DeptID

```
SELECT *
FROM Employee emp JOIN Department dept
        ON emp.DepartmentID = dept.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Robinson	34	Clerical	34
Jones	33	Engineering	33
Smith	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Natural Join

Employee ⋈ Department

```
SELECT *
FROM Employee emp NATURAL JOIN Department dept
```

DepartmentID	Employee.LastName	Department.DepartmentName
34	Smith	Clerical
33	Jones	Engineering
34	Robinson	Clerical
33	Steinberg	Engineering
31	Rafferty	Sales

Nulls and Joins

- Sometimes need special variations of joins:
  - I want to see all employees and their departments
  - … But what if there's a department with no employees?
  - Or what if an employee has not been assigned to a department?
- Outer join:
  - Most common is *left outer join*

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

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Left Outer Join

Employee ⟕ Department

Employee.DepartmentID = Department.DepartmentID

```
SELECT *
FROM Employee emp LEFT OUTER JOIN Department dept
                  ON emp.DepartmentID = dept.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
John	NULL	NULL	NULL
Steinberg	33	Engineering	33

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Right Outer Join

Employee ⟖ Department

Employee.DepartmentID = Department.DepartmentID

```
SELECT *
FROM Employee emp RIGHT OUTER JOIN Department dept
        ON emp.DepartmentID = dept.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	*Marketing*	*35*

Employee table

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
John	NULL

Department table

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

Full Outer Join

Employee ⟗ Department

Employee.DepartmentID = Department.DepartmentID

```
SELECT *
FROM Employee FULL OUTER JOIN Department dept
        ON emp.DepartmentID = dept.DepartmentID
```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Smith	34	Clerical	34
Jones	33	Engineering	33
Robinson	34	Clerical	34
John	NULL	NULL	NULL
Steinberg	33	Engineering	33
Rafferty	31	Sales	31
NULL	NULL	*Marketing*	*35*

Outline

- ✓ Multi-Relation SQL Queries
- ✓ Join SQL Queries
- Advanced SQL
  - Subqueries
  - Boolean Operators (IN, ANY, EXISTS, ALL)
  - Set operations

Subqueries

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

```
SELECT *
FROM (SELECT * FROM Customer WHERE
name LIKE 'A%') as temp
WHERE temp.phone LIKE '5%';
```

Subqueries that Return Scalar

- If a subquery is guaranteed to produce one tuple with one component, then the subquery can be used as a value.
  - "Single" tuple often guaranteed by key constraint.
  - A run-time error occurs if there is no tuple or more than one tuple.

Example

From Sells(cafe, drink, price), find the café shops that serve Latte for the same price Espresso Royal charges for Cappuccino.

Two queries would surely work:

1. Find the price Espresso Royal charges for Cappuccino.
2. Find the café shops that serve Latte at that price.

Query + Subquery Solution

Find the café shops that serve Latte at that price

```
SELECT cafe
FROM Sells
WHERE drink = 'Latte' AND
  price = (SELECT price
           FROM Sells
           WHERE cafe = 'Espresso Royal'
           AND drink = 'Cappuccino');
```

Find the price Espresso Royal charges for Cappuccino.

Outline

- ✓ Multi-Relation SQL Queries
- ✓ Join SQL Queries
- Advanced SQL
  - ✓ Subqueries
  - Boolean Operators (IN, ANY, EXISTS, ALL)
  - Set operations

Boolean Operators: The IN Operator

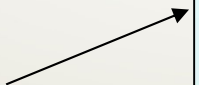
- <tuple> IN <relation> is true if and only if the tuple is a member of the relation.
  - <tuple> NOT IN <relation> means the opposite.
- IN-expressions can appear in WHERE clauses.
- The <relation> is often a subquery.

Example

- From Drinks(name, manf) and Likes(customer, drink), find the name and manufacturer of each drink that Fred likes.

```
SELECT *
FROM Drinks
WHERE name IN (SELECT drink
               FROM Likes
               WHERE customer = 'Fred');
```

The set of drinks Fred likes



Boolean Operators: The Exists Operator

- EXISTS(<relation>) is true if and only if the <relation> is not empty.

- Being a boolean-valued operator, EXISTS can appear in WHERE clauses.

Example Query with EXISTS

Example: From Drinks(name, manf), and Sells (café, drink, price) find the name and manufacturer of drinks with price >= $4.99

Set of drinks (and their sales information) with the same name as b1 and price >= 4.99

```
SELECT *
FROM Drinks b1
WHERE EXISTS (SELECT *
              FROM Sells
              WHERE drink = b1.name AND price >= 4.99);
```

Read "Correlated Subqueries" Section 6.3.4.

Boolean Operators: The Operator ANY

- *x* = ANY(<relation>) is a Boolean condition meaning that *x* equals at least one tuple in the relation.
- Similarly, = can be replaced by any of the comparison operators.
- Example: *x* >= ANY(<relation>) means *x* is greater than or equal to at least one tuple in the relation.
  - Note tuples must have one component only.

Boolean Operators: The Operator ALL

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

- The <> can be replaced by any comparison operator.
- Example: *x* >= ALL(<relation>) means there is *x* is larger than every tuple in the relation.

Example

- From Sells(cafe, drink, price), find the drink(s) sold for the highest price.

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

price from the outer Sells must not be less than any price.

Outline

- ✓ Multi-Relation SQL Queries
- ✓ Join SQL Queries
- Advanced SQL
  - ✓ Subqueries
  - ✓ Boolean Operators (IN, ANY, EXISTS, ALL)
  - Set operations

Operations on Bags (and why we care)

- Union: {a,b,b,c} U {a,b,b,b,e,f,f} = {a,a,b,b,b,b,b,c,e,f,f}
  - *add* the number of occurrences
- Difference: {a,b,b,b,c,c} – {b,c,c,c,d} = {a,b,b}
  - subtract the number of occurrences
- Intersection: {a,b,b,b,c,c}∩{b,b,c,c,c,c,d} = {b,b,c,c}
  - minimum of the two numbers of occurrences
- The SELECT-FROM-WHERE statement uses bag semantics
  - Selection: preserve the number of occurrences
  - Projection: preserve the number of occurrences (no duplicate elimination)
  - Cartesian product, join: no duplicate elimination

Read Section 5.3 of the book for more detail

Union, Intersection, and Difference

- Union, intersection, and difference of relations are expressed by the following forms, each involving subqueries:
  - (subquery) UNION (subquery)
  - (subquery) INTERSECT (subquery)
  - (subquery) EXCEPT (subquery)

Example

From relations Likes(customer, drink), Sells(cafe, drink, price) and Frequents(customer, cafe), find the customers and drinks such that:

1. The customer likes the drink, and
2. The customer frequents at least one café shop that sells the drink.

How would we do this?

Solution

```
(SELECT * FROM Likes)
  INTERSECT
(SELECT customer, drink
 FROM Sells, Frequents
 WHERE Frequents.cafe = Sells.cafe
);
```

The customer frequents a café shop that sells the drink.

Bag Semantics for Set Operations in SQL

Although the SELECT-FROM-WHERE statement uses bag semantics, the default for union, intersection, and difference is set semantics.

- That is, duplicates are eliminated as the operation is applied.

Controlling Duplicate Elimination

- Force the result to be a set by SELECT DISTINCT . . .
- Force the result to be a bag (i.e., don't eliminate duplicates) by ALL, as in
  - . . . UNION ALL . . .

Outline

- ✓ Multi-Relation SQL Queries
- ✓ Join SQL Queries
- ✓ Advanced SQL
  - ✓ Subqueries
  - ✓ Boolean Operators (IN, ANY, EXISTS, ALL)
  - ✓ Set operations