January 23, 2012 – Week 3

slides updated on January 24 (changes shown in green font)

CS 386/586

Winter 2013

Outline

- Why bother to define or use relational algebra?
  - precise spec. for implementing query processing
  - internal representation for SQL queries
  - relational algebra equivalences – used for opt.
  - But … we must do it for bags (not sets)
  - And … we need enough operators to cover SQL
- Relational algebra for bags
  - select, project, cross, join – activity: how many for cross
  - union, intersection, set difference – with examples
  - SQL (postgresql) queries – demo & activity
- Extended operators:
  - delta (distinct); tau (order by); outer join symbols (left, right, full)
  - quick exercise
  - fancy project; aggregates
  - quick exercise
- Group by/having …
  - French fries
  - SQL queries to write - exercise

WHY DO WE LEARN RELATIONAL ALGEBRA? (WHAT IS A RELATIONAL ALGEBRA EQUIVALENCE??)

Why do we use Relational Algebra? (Update to slide shown in green)

Because:

- It is mathematically defined ; precise specification of semantics of queries, e.g., for implementors.
- We can prove that two relational algebra expressions are equivalent. For example:

$$\sigma_{cond1}(\sigma_{cond2} R) \equiv \sigma_{cond2}(\sigma_{cond1} R) \equiv \sigma_{cond1\ AND\ cond2} R$$
$$\equiv (\sigma_{cond1} R) \cap (\sigma_{cond2} R)$$

- Note: since we have two languages: relational algebra defined to work on and produce sets as well as relational algebra defined to work on and produce bags – we need to make sure we use the correct equivalences in the correct context. Some equivalences are true for sets but not bags.

(this slide added after class; 3rd equiv. added)
Some relational algebra equivalences

$\sigma_{cond1}(\sigma_{cond2} R) \equiv \sigma_{cond2}(\sigma_{cond1} R) \equiv \sigma_{cond1\ AND\ cond2} R$

$\equiv (\sigma_{cond1} R) \cap (\sigma_{cond2} R)$

$R1 \bowtie_{cond12} R2 \equiv (\sigma_{cond1} R1) \bowtie_{cond2} R2$

provided: "cond12" can be split (into "cond1 and cond2") and "cond1" addresses only attribute of R1.

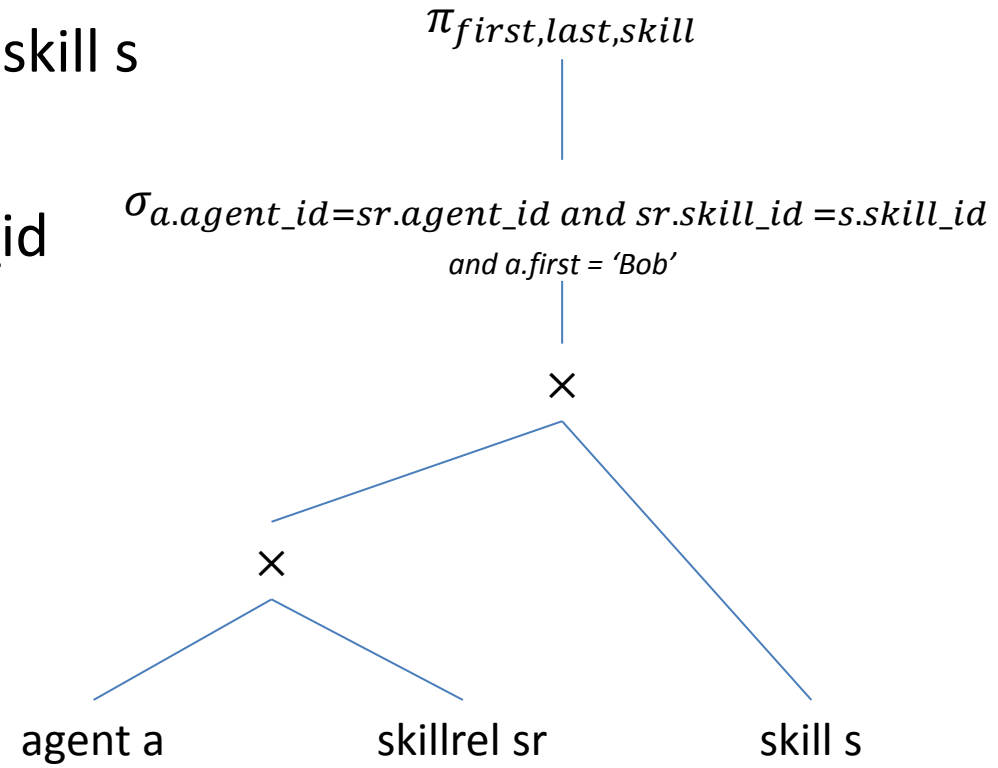
$\sigma_{cond12}(R1 \times R2) \equiv \sigma_{cond1}(R1 \bowtie_{cond2} R2)$

provided: "cond12" can be split (into "cond1 and cond2") where cond2 is a join condition for R1 and R2.

SQL → rel. alg. expression tree

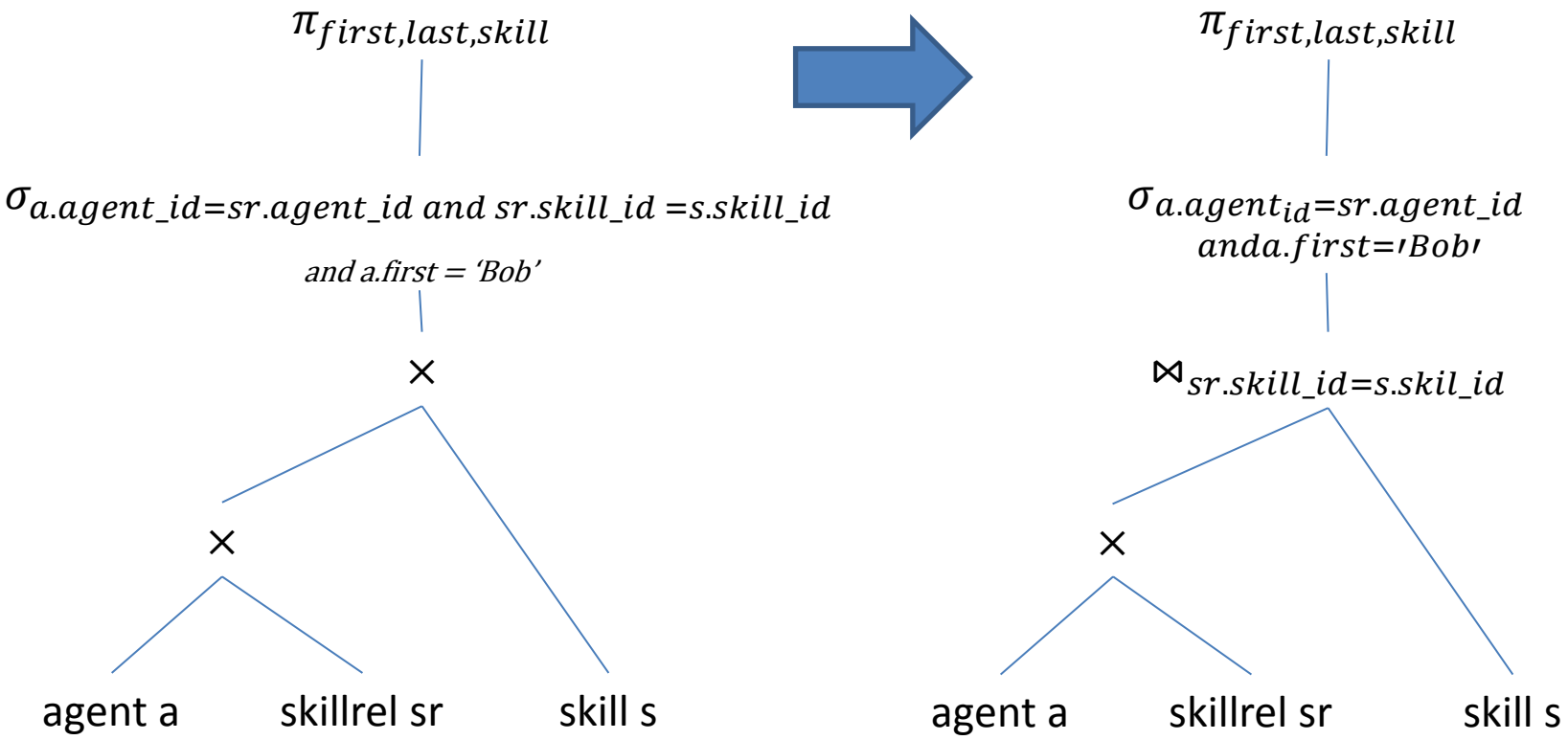
SQL query:

```
select   first, last, skill
from     agent a, skillrel sr, skill s
where    a.agent_id = sr.
and      sr.skill_id = s.skill_id
and      a.first = "Bob"
```



modify this expression tree using this equivalence:

(based on σ_{cond12} (R1 × R2) ≡ σ_{cond1}(R1 $\bowtie_{cond2}$ R2)



Here cond1 is $a.agent_{id} = sr.agent_{id} and\ a.first = 'Bob'$ and cond2 is the join condition $sr.skill\_id = s.skil\_id$

More equivalences: "AND", "OR", and "NOT"

$$\sigma_{cond1\ OR\ cond2}\ R \equiv (\sigma_{cond1}\ R)\ \cup\ (\sigma_{cond2}\ R)$$

$$\sigma_{cond1\ AND\ cond2}\ R \equiv (\sigma_{cond1}\ R)\ \cap\ (\sigma_{cond2}\ R)$$

$$\sigma_{cond1\ AND\ NOT\ cond2}\ R \equiv (\sigma_{cond1}\ R)\ -\ (\sigma_{cond2}\ R)$$

But wait … the first one doesn't hold for bags.

Also, we need enough rel. alg. operators for all features of SQL!

Relational Algebra Equivalences:

- help query writers write queries in different ways
- help query optimizers choose among different query plans

We must know for sure that the two queries (the original and the replacement) are identical – that they will produce the same answer on all database instances. So, relational algebra equivalences must be proved correct.

RELATIONAL ALGEBRA FOR SETS (REMINDER)

Relational Algebra Operators: How should we define them – for bags?

There are seven operators plus renaming:

- π project
- σ select
- $\cup$ union
- $\cap$ intersection
- $-$ difference
- $\times$ cross product
- $\bowtie$ join
- renaming (to provide names for the relation & attributes of answer)

Union: set theory vs. relational algebra for sets

- In set theory, the elements of a set can be all different types
  S = {'a', 7053, (1, 2, 'Smith'), (3, 4, 5, 6, 7, 8, 9)}
  (atomic values as well as tuples of different lengths)
- In set theory, you can take the union (or intersection or difference) of any two sets.
  A = {1, (3, 4, 'a'), 5.3} B = {7, 1, (2, 3)}
  A ∪ B = {1, (3, 4, 'a'), 5.3, 7, (2, 3)}
  A ∩ B = {1}
  A − B = {(3, 4, 'a'), 5.3}

But in relational algebra, relations must have the "same shape" (be *union-compatible)* before you can use ∩, ∪, −.

Union Compatible

- Two relations are *union-compatible* if they have:
  - the same number of attributes
  - the corresponding attributes have the same name (use the rename operator, if necessary)
  - corresponding attributes and are defined on the same domains.

Checking-Account (num, owner, balance)

Savings-Account (num, owner, balance)

These relations are union-compatible relations provided corresponding attributes are defined on same domain.

∪ Union in Relational Algebra

Consider this query: Checking-account ∪ Savings-account

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00
	103	J. Smith	5000.00

∩ Intersection in Relational Algebra (example 1)

Consider this query: Checking-account ∩ Savings-account

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

What's the answer to this query?

∩ Intersection in Relational Algebra (ex. 1 cont.)

What is the answer to this query:

Checking-account ∩ Savings-account

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

What's the answer to this query?

It's empty. There are no tuples that are in both relations.

∩ Intersection in Relational Algebra (example 2)

What's the answer to this query?

$(\pi_{\text{owner}}\text{Checking-account}) \cap (\pi_{\text{owner}}\text{Savings-account})$

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

∩ Intersection in Relational Algebra (ex. 2 cont.)

Intermediate query answers

$(\pi_{\text{owner}}\text{Checking-account}) \cap (\pi_{\text{owner}}\text{Savings-account})$

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

∩

	owner
	J. Smith

Query answer is (using attribute name from Checking-account):

	owner
	J. Smith

— Set Difference: Relational Algebra (ex. 1)

Consider this query: Checking-account — Savings-account

Find all the tuples (rows) that are in the Checking-account relation that are not in the Savings-account relation.

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

What is the answer?

— Set Difference: Relational Algebra (ex. 1 cont.)

Consider this query: Checking-account — Savings-account

Find all the tuples (rows) that are in the Checking-account relation that are not in the Savings-account relation.

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

What is the answer? All of the rows in the checking-account table.

— Set Difference: Relational Algebra (ex. 2)

$(\pi_{owner}\text{Checking-account}) - (\pi_{owner}\text{Savings-account})$

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

Compute the intermediate query answers. Then, what is the final query answer?

— Set Difference: Relational Algebra (ex. 2 cont.)

$(\pi_{owner}\text{Checking-account}) - (\pi_{owner}\text{Savings-account})$

Checking-account	num	owner	balance
	101	J. Smith	1000.00
	102	W. Wei	2000.00
	104	M. Jones	1000.00
	105	H. Martin	10,000.00

Savings-account	num	owner	balance
	103	J. Smith	5000.00

Final query answer is:

	owner
	W. Wei
	M. Jones
	H. Martin

EXTENDING RELATIONAL ALGEBRA TO WORK WITH BAGS

Select, Project, Cross Product, Join with bags

For each of these queries, consider what happens if there are duplicate student or faculty tuples.

$\sigma_{\text{age}<35}\text{Student}$

$\pi_{\text{first}}\text{Student}$

$\text{Student} \times \text{Faculty}$

$\text{Student s} \bowtie_{\text{s.advisor=f.id}} \text{Faculty f}$

Operators that work as you would expect with bags (similar to the way they work with sets)

- σ – select – simply evaluate each tuple in the bag
- π – project – simply process each tuple in the bag
- $\times$ - cross product – simply concatenate one tuple from each bag together; do so for every possible combination of one tuple from one bag and one tuple from the other bag
- $\bowtie$ - join – works like cross product with the added processing of checking each element of the cross product to see if the join condition evaluates to true.

Exercise with Bags
(a row in the table represents one visit)

starbucks-visit

personid	date
1	Jan. 2
1	Jan. 3
1	Jan. 3
1	Jan. 4
2	Jan. 2
4	Jan. 5

Note: each row represents a visit; so the number of duplicates of a row is important!

mcdonalds-visit

personid	date
1	Jan. 2
1	Jan. 3
1	Jan. 4
1	Jan. 5
2	Jan. 2
2	Jan. 2
2	Jan. 6
3	Jan. 8

What does it mean:

starbucks-visit ∩ mcdonalds-visit
starbucks-visit ∪ mcdonalds-visit
starbucks-visit – mcdondalds-visit
mcdonalds-visit – starbucks-visit

Describe these queries in English; compute the query answer.

Sets vs. Bags What about ∩,∪, and −?

Consider sets S1={a,b}, S2={a,b,c,d}, S3={b,d}

- What is S1 ∪ S2?
- What is S3 – S1?
- What is S2 ∩ S3?

Now consider bags B1={a,a,b,b,b}, B2={a,b,b,c,c,d,d}, B3={b,b,b,b,d,d,d}

- What is B1 ∪ B2?
- What is B3 – B1?
- What is B2 ∩ B3?

Is there any correspondence between S1 and B1? etc.?

Is there any correspondence between set and bag results?

Sets vs. Bags (also called Multi-sets) answers

Consider sets S1={a,b}, S2={a,b,c,d}, S3={b,d}

- What is S1 ∪ S2? **{a, b, c, d} = S2**
- What is S3 – S1? **{d}**
- What is S2 ∩ S3? **{b, d} = S3**

Now consider bags B1={a,a,b,b,b}, B2={a,b,b,c,c,d,d}, B3={b,b,b,b,d,d,d}

- What is B1 ∪ B2? **{a, a, a, b, b, b, b, b, c, c, d, d}**
- What is B3 – B1? **{b, d, d, d}**
- What is B2 ∩ B3? **{b, b, d, d}**

Is there any correspondence between set and bag results?

Working with bags

- Consider the first relational algebra equivalence from slide 8:

  $\sigma_{cond1\ OR\ cond2}\ R \equiv (\sigma_{cond1}\ R) \cup (\sigma_{cond2}\ R)$

  Exercise: prove that this is not true for bags
  (provide a counterexample)

In SQL, the ALL keyword is for bag semantics the default is set semantics!!!

Set semantics (default)	Bag semantics (with ALL)
UNION	UNION ALL
INTERSECT	INTERSECT ALL
EXCEPT	EXCEPT ALL

Union, Intersect, Except (without ALL)

- If you don't specify ALL, then the answer is computed on sets.
- You can think of (set semantics, without ALL) like this:
  - Eliminate duplicates from first table
  - Eliminate duplicates from second table
  - THEN … compute union, intersect, or difference
  - Eliminate duplicates from query answer table

Using the ALL keyword – some examples

- UNION ALL:
  - A has 3 copies of a row, B has 5
  - A ∪ B has 8 copies of the row
- INTERSECT ALL:
  - A has 3 copies of a row, B has 5
  - A ∩ B has 3 copies of the row
- EXCEPT ALL:
  - A has 3 copies of a row, B has 5
  - A – B has 0 copies of the row; B – A has 2

Exercise using bags of tuples

starbucks-visit

personid	date
1	Jan. 2
1	Jan. 3
1	Jan. 3
1	Jan. 4
2	Jan. 2
4	Jan. 5

mcdonalds-visit

personid	date
1	Jan. 2
1	Jan. 3
1	Jan. 4
1	Jan. 5
2	Jan. 2
2	Jan. 2
2	Jan. 6
3	Jan. 8

starbucks-visit ∪ mcdonalds-visit

```
SELECT * from starbucks-visit
UNION ALL
SELECT * from mcdonalds-visit
```

starbucks-visit ∩ mcdonalds-visit

```
SELECT * from starbucks-visit
INTERSECT ALL
SELECT * from mcdonalds-visit
```

starbucks-visit − mdcondalds-visit

```
SELECT * from starbucks-visit
EXCEPT ALL
SELECT * from mcdonalds-visit
```

Set Operations Implementation

	SQL Server 2000	SQL Server 2005	Oracle 10g	PostgreSQL	MySQL 5.1
UNION	x	x	x	x	x
UNION ALL	x	x	x	x	x
INTERSECT		x	x	x	
INTERSECT ALL				x	
EXCEPT		x	x	x	
EXCEPT ALL				x	

Exercise: try all three queries on the right – then try without "All" – explain all six queries, in English

I created three tables using these queries:

```
select   first, clearance_id
into     agentdavid
from     agent
where    first = 'David'

select   first, clearance_id
into     agentbob
from     agent
where    first = 'Bob'
```

```
select   clearance_id from agentdavid
union all
select   clearance_id from agentbob

select clearance_id from agentdavid
intersect all
select clearance_id from agentbob

select clearance_id from agentdavid
except all
select clearance_id from agent bob
```

EXTENDED RELATIONAL ALGEBRA: ADDITIONAL OPERATORS

Additional Operators

- δ (lower case delta) - the duplicate elimination operator. This operator changes a bag into a set.
  In SQL, eliminate duplicates using DISTINCT.
  δ(R)
- τ (lower case tau) – the sort operator. This operator takes a relation or a relational expression and sorts it according to the list of attributes provided (L).
  In SQL, use the ORDER BY <list of attributes>
  $$\tau_L(R)$$
- $\overset{o}{\bowtie}_L$, $\overset{o}{\bowtie}_R$, $\overset{o}{\bowtie}$ - left, right and full outer join
  in SQL, use left join, right join, and full join.

Additional operators (cont.)

- extended project – do arithmetic & aggregates
- γ (lower case gamma) – grouping operator.
  Return one row for each group; there is one group for each assignment of values to the grouping attributes.
  Return the grouping attributes plus any aggregate operators (evaluated on the group).

$$\gamma_{\mathrm{L}}(\mathrm{R})$$

  where L is a list of attributes from R (the grouping attributes) or aggregate operators (applied to each group).
  In SQL, use the GROUP BY clause (with HAVING clause), and with attributes and aggregates in select.

Aggregates in the SELECT Clause

We can use aggregate operators in the SELECT clause: COUNT, SUM, MIN, MAX, and AVG

```
SELECT MIN(CBalance), MAX(CBalance), AVG (CBalance)
FROM   Customer;
```

```
SELECT  MIN(CBalance), MAX(CBalance), AVG (CBalance)
FROM    Customer
WHERE   age > 35;
```

If one aggregate operator appears in the SELECT clause, then ALL of the entries in the select clause must be an aggregate operator (unless the query includes a GROUP BY clause (covered later)).

Does this make sense?

```
SELECT S.Name, S.Phone, AVG(S.Age)
FROM   Student S
WHERE  S.Major = "CS";
```

Consider the data:

Student	ID	Name	Phone	Age	Major
	1	Joe	123	24	CS
	2	Mary	456	28	CS
	3	Arun	789	32	CS
	4	John	999	18	English

WHAT would the query answer be? This SQL query is NOT well-formed and thus not allowed.

Extensions to the SELECT clause

These queries compute the same extent for the query answer but the attributes names are different.

"Important-Customer" is used in the 2nd query answer.

```
SELECT Name, Address
FROM   Customer
WHERE  CRating = 10;

SELECT Name AS Important-Customer, Address
FROM   Customer
WHERE  CRating = 10;
```

The query writer can assign new names to the columns in a query answer with 'AS <new attribute name>' clause.

DISTINCT in the SELECT Clause (and within aggregates)

What is the difference between these two queries?

```
SELECT COUNT(Name)          SELECT COUNT(DISTINCT Name)
FROM   Customer;            FROM   Customer;
```

When will these two queries return the same answer?
(That is, for what sort of database instance would these queries return the same answer?)

Practice Questions

What is the implication of using DISTINCT when computing the SUM or AVG of an attribute?

SUM(DISTINCT(age)) vs. SUM (age)

What is the implication of using DISTINCT when computing the MIN or MAX of an attribute?

MIN(DISTINCT(age)) vs. MIN(age)

Practice Questions Answers

What is the implication of using DISTINCT when computing the SUM or AVG of an attribute?

Answer: The SUM or AVG will be computed only on distinct values (compared to the normal case - without DISTINCT - where the SUM and AVG are computed on all values)

What is the implication of using DISTINCT when computing the MIN or MAX of an attribute?

Answer: There is no difference. The maximum or minimum value is the same whether or not duplicates are eliminated ahead of time.

More Extensions to the SELECT clause

The SELECT clause list can also include simple arithmetic expressions using +, -, *, and /.

```
SELECT  (CAmount – CBalance) AS AvailableCredit, Name
FROM      Customer
WHERE    CAmount > 0;
```

Note: it is very common to use "AS" to name computed columns in queries.

This query computes the available credit, for those Customers that have Camount > 0.

SQL … ORDER BY, GROUP BY, HAVING

SELECT… INTO <temp-table-name>

FROM…

WHERE…

GROUP BY…

HAVING …

ORDER BY…

"ORDER BY" sorts the query answer.

"INTO < temp-table-name>" places the query answer into a table – in the current database – with the name given.

GROUP BY, HAVING

```
SELECT      Salesperson, COUNT(*)
FROM        Customer
GROUP BY    Salesperson;
```

```
SELECT      Salesperson
FROM        Customer
GROUP BY    Salesperson
HAVING      Count(*) > 1;
```

An SQL query can have the answer "grouped". When a query answer is grouped, there is one output row per group.

The HAVING clause is a predicate evaluated against each group. A group participates in the query answer if it satisfies the HAVING predicate

Group by example

Input data:

Customer

(Number, Name, Address, … Salesperson)

101	Mary	…	5
102	John	…	8
103	Quan	…	NULL
106	Susan	…	5
107	David	…	5
109	Mike	…	2
110	Ying	…	8

```
SELECT    Salesperson, count(*)
FROM      Customer
GROUP BY  Salesperson;
```

Example: Group by

Input data:

Customer

(Number, Name, Address, … Salesperson)

101	Mary	…	5
102	John	…	8
103	Quan	…	NULL
106	Susan	…	5
107	David	…	5
109	Mike	…	2
110	Ying	…	8

```
SELECT    Salesperson, count(*)
FROM      Customer
GROUP BY  Salesperson;
```

Intermediate result:
4 groups

103	Quan…		Null
101	Mary …		5
106	Susan	…	5
107	David	…	5
102	John	…	8
110	Ying	…	8
109	Mike	…	2

Example: Group by

Intermediate result:
4 groups

103	Quan…		Null
101	Mary …		5
106	Susan	…	5
107	David	…	5
102	John	…	8
110	Ying	…	8
109	Mike	…	2

Query answer:
one row per group!

salesperson, count(*)
Null, 1
5, 3
8, 2
2, 1

```
SELECT    Salesperson, count(*)
FROM      Customer
GROUP BY  Salesperson;
```

Example: Group by with Having

Intermediate result:
4 groups

103	Quan …		

101	Mary …		5
106	Susan	…	5
107	David	…	5

102	John	…	8
110	Ying	…	8

109	Mike	…	2

Query answer:
one row per group
that satisfies the
HAVING clause!

salesperson, count(*)
5, 3
8, 2

```
SELECT    Salesperson, count(*)
FROM      Customer
GROUP BY  Salesperson
HAVING    Count(*) > 1;
```

GROUP BY, HAVING

```
SELECT      Salesperson, COUNT(*)
FROM        Customer
GROUP BY    Salesperson;
```

```
SELECT      Salesperson, COUNT(*)
FROM        Customer
GROUP BY    Salesperson
HAVING      Count(*) > 1;
```

The only attributes that can appear in a "grouped" query answer are the grouping attribute or aggregate operators (that are applied to the group).

Group By – Order of Clauses

1. Start with the FROM clause to see the tables.
2. Evaluate the WHERE clause for every combination of tuples produced from the FROM clause.
3. Group the intermediate query answer into groups, based on the attributes listed in the GROUP BY clause.
4. Evaluate the HAVING clause for each group.
5. Place one row in the final query answer for each group where the HAVING clause evaluted to true; put only the attributes listed in the SELECT clause.

Question

```
SELECT      Salesperson, AVG(CBalance)
FROM        Customer
GROUP BY    Salesperson
HAVING      AVG(CBalance) > 200;

SELECT      Salesperson, AVG(CBalance)
FROM        Customer
WHERE       CBalance > 200
GROUP BY    Salesperson;
```

Are these queries equivalent (give the same answer on all instances)?

Questions for beverage table (french fries) (in the introdb_store database on postgresql)

- List the stores that appear in the price list; show each store once.
- List the count of whole milk products sold in the price list. Do NOT remove duplicates.
- List the store and the number of whole milk products sold at that store and the max price of the whole mile products sold at that store – for each store.
- List the store, beverage, and count of items sold – for each store/beverage combination in the price list.
- List the store, beverage, and count of items sold with a price greater than $1.00 for each store/beverage combination in the price list.

Queries for the “pricelist ” sample data on next slide.

1. select distinct store from pricelist
2. select count(*) from pricelist where beverage = ‘Whole Milk’
3. select store, count(*), max(price) from pricelist
   where beverage = ‘Whole Milk’
   group by store
4. select store, beverage, count(*)
   from pricelist where price > 1
   group by store, beverage
5. select store, beverage, count(*)
   from pricelist where price > 1
   group by beverage, store

What are the query answers?

Figure the answer out using the paper “rows” for this table.

Check your query answer by running the query in dbclass.cs.pdx.edu

in the introd_store database. Note: I just changed the tablename to pricelist

Data used for GROUP BY Exercise.

Table name is pricelist.

Store	Beverage	Size	Price
Plaid Pantry	Whole Milk	16oz	1.15
Plaid Pantry	2% Milk	16oz	1.05
Plaid Pantry	Whole Milk	64oz	3.20
Plaid Pantry	Pepsi	12oz	.70
Plaid Pantry	Diet Pepsi	12oz	.70
Plaid Pantry	Pepsi	20 oz	.95
7-11	Whole Milk	8oz	.65
7-11	Chocolate Milk	8oz	.65
7-11	Whole Milk	16oz	1.10
7-11	2% Milk	16oz	1.00
7-11	Coke	12oz	.65
7-11	Diet Coke	12oz	.65
7-11	Coke	20oz	1.10
7-11	Diet Coke	20oz	1.10
7-11	Diet Caffeine-Free Cherry Low-Fizz Vitamin-Fortified Coke	20oz	1.10
Circle K	Whole Milk	8oz	.60
Circle K	Whole Milk	16oz	1.20
Circle K	Whole Milk	32oz	2.30
Circle K	Whole Milk	128oz	4.10
Circle K	Coke	20oz	1.15
Circle K	Diet Coke	20oz	1.15
Circle K	Coke	32oz	2.10
Circle K	Diet Coke	32oz	2.05
Safeway	Skim Milk	16oz	1.20
Safeway	Skim Milk	32oz	2.00
Safeway	2% Milk	16oz	1.25
Safeway	2% Milk	64oz	2.75
Safeway	Diet Pepsi	12oz	.60
Safeway	Diet Coke	12oz	.60