SQL
Aggregation and Grouping

Davood Rafiei

Outline

- Aggregation functions
- Grouping
- Examples using the following tables:

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

Aggregates

- Functions that operate on sets:
  - COUNT, SUM, AVG, MAX, MIN
- Produce numbers (not tables)
- Not part of relational algebra

SELECT COUNT(*)
FROM customer

SELECT MAX (*assets*)
FROM branch

Aggregation – Example 1

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Find the number of customers in Edmonton.

```
SELECT   COUNT(*)
FROM     customer
WHERE    city = 'Edmonton'
```

Aggregation – Example 2

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Find the total assets of branches in Edmonton.

```
SELECT    SUM(assets)
FROM      branch
WHERE     city = 'Edmonton'
```

- See your DBMS SQL Reference for more fancy aggregate and other functions.

Aggregation – Use of Distinct

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Find the number of different branches where John Doe has a deposit account.

```
SELECT   COUNT(DISTINCT bname)
FROM     deposit
WHERE    cname = 'John Doe'
```

Aggregation & Nesting

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Find the number of customers who have deposit accounts in at least 3 different branches.

```
SELECT      COUNT(*)
FROM        customer c
WHERE       3 <=
          (SELECT COUNT(DISTINCT bname)
           FROM   deposit d
           WHERE  c.cname = d.cname)
```

Note: We have dropped ALL/SOME before the subquery!

Aggregation & Nesting (Cont)

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Find the names of branches which have assets greater than the average assets of all branches.

```
SELECT  bname
FROM    branch
WHERE   assets >
        (SELECT  AVG(assets)
         FROM    branch)
```

Grouping

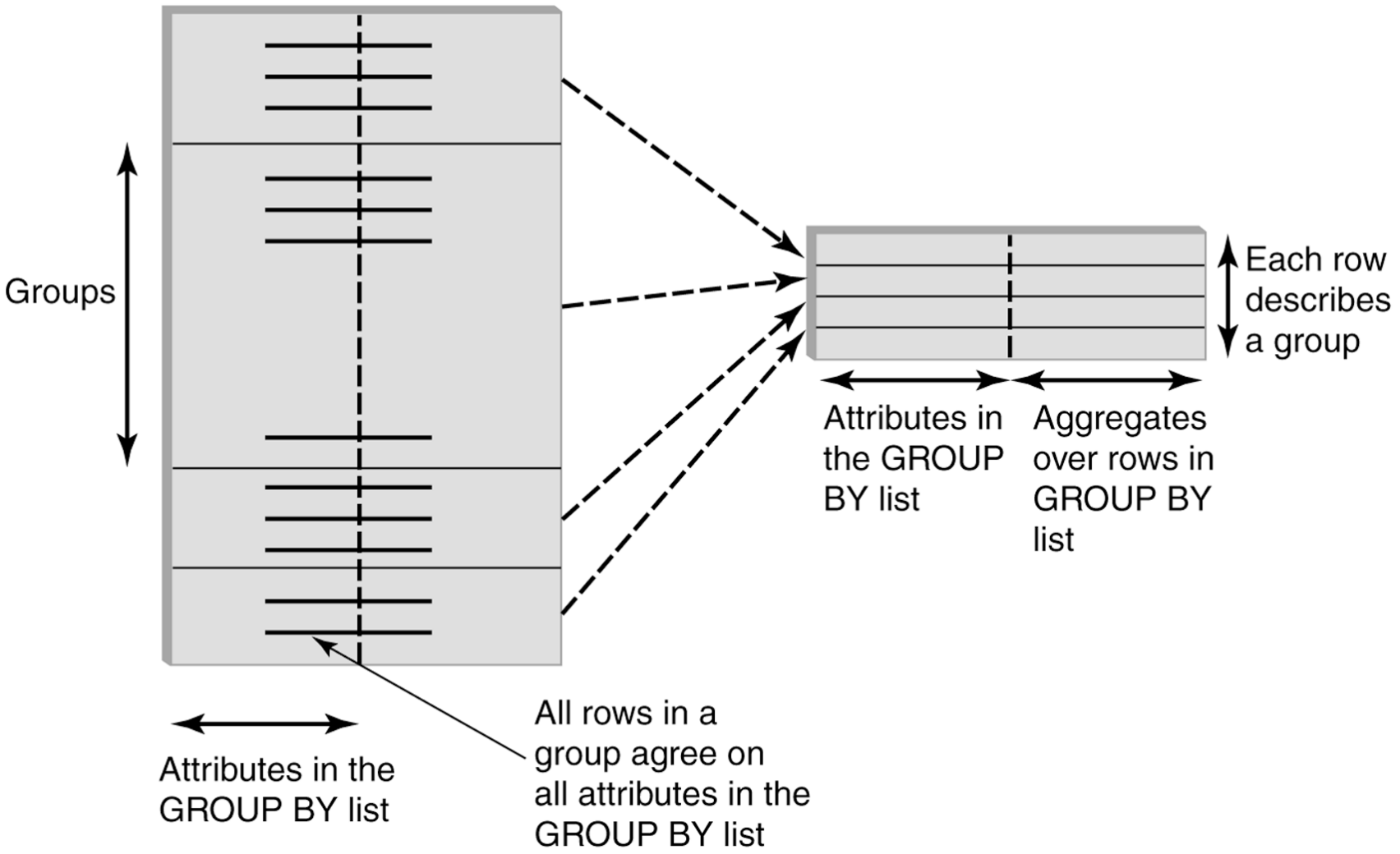
- How to compute the number of customers *per city*?
  - 1) fire off a separate query for each city:

```
SELECT   COUNT(*)
FROM     customer
WHERE  city  = 'Edmonton'
```

    - ✓ Cumbersome
    - ✓ What if the number of cities changes? Add another query?
  - 2) define a special *grouping operator*:

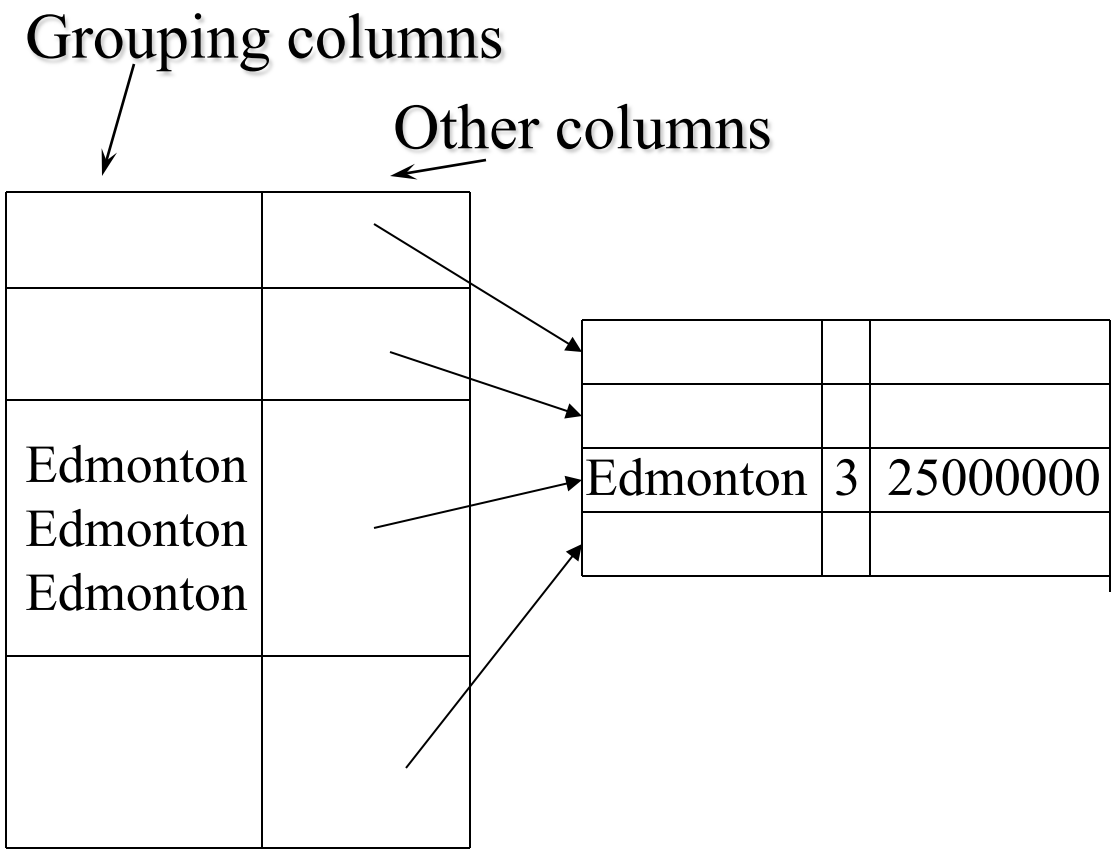
```
SELECT     city,  COUNT(cname)
FROM       customer
GROUP BY   city
```

GROUP BY



Source: Kifer, Bernstein, Lewis, 2006

GROUP BY - Example



Attributes:

-city

-number of branches

-avg assets

```
SELECT city, COUNT(b.bname), AVG(b.assets)
FROM branch b
GROUP BY b.city
```

Grouping

- Syntax

```
SELECT  a1, …, ak,agg1, …, aggl
FROM    R1, ...
GROUP BY a1 …, ak
```

- Rules:
  - Every column name in the SELECT clause must appear in the GROUP BY clause (this is not required if the column name is only used in aggregation functions).
  - Eg of **agg:** sum(a_i), min(a_j), ...

HAVING Clause

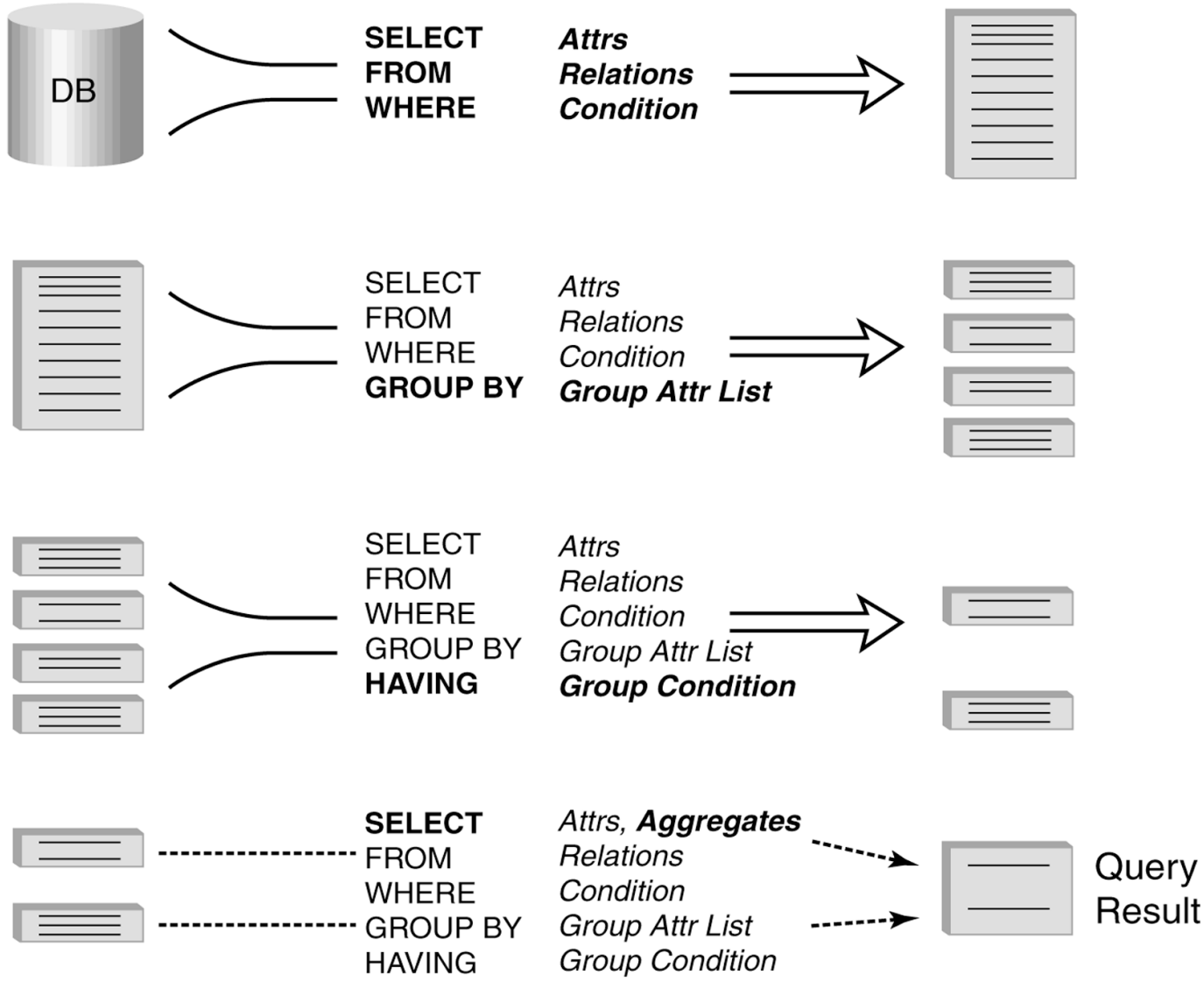
branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Find cities with more than two bank branches.

```
SELECT  city
FROM    branch
GROUP BY  city
HAVING  COUNT(*) > 2
```

- '**GROUP BY** city' partitions tuples into groups where all tuples in a group have the same values in city column.
- '**HAVING** …' is applied to each group separately to determine if the group as a whole qualifies.

Evaluation of GroupBy with Having



Source: Kifer, Bernstein, Lewis, 2006

HAVING - Example

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- For every branch, list the branch name and the name of every customer who has more than 3 loans each over $100,000 in that branch.

```
SELECT   bname, cname, COUNT(accno)
FROM     loan
WHERE    amount > 100000
GROUP BY bname, cname
HAVING   COUNT(*) > 3
```

Grouping: Proper Usage

Student (sid, name, phone)
Transcript (sid, cid, sem, grade)

- For every student, list the id, the name and the average grade.

```
SELECT  S.sid, S.name, AVG(grade)
FROM    Student S, Transcript T
WHERE   S.sid = T.sid
GROUP BY  S.sid          -- wrong
          S.sid, S.name  -- right
```

Every attribute that occurs in SELECT *clause must also occur in* GROUP BY *or it must be an aggregate.* S.Name *does not.*

SQLite does not enforce this!!

Aggregates: Proper Usage

SELECT d.cname, AVG (d.balance)

– *makes no sense (in the absence of* GROUP BY *clause)*

SELECT COUNT (*), AVG (d.*balance*)

– *but this is OK*

WHERE d.*balance* > AVG (SELECT ….)

– *aggregate cannot be applied to result of* SELECT *statement*

Grouping Exercises

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Q1. For each customer having more than two deposit accounts, find the name and the city of the customer.

- Q2. Find branches with an average account balance greater than $1,200.

Grouping Exercises

branch (bname, address, city, assets)
customer(cname, street, city)
deposit(accno, cname, bname, balance)
loan(accno, cname, bname, amount)

- Q3. Find branches with an average account balance greater than or equal to the average account balances of all branches.

- Q4. Find cities of all customers who have total deposit balances of over $4,000.