Introduction to Database Systems

Aggregate Queries

Stéphane Bressan

NUS
National University
of Singapore

The order of the records in the tables is in never guaranteed. However, you can sort the results of a query by adding the "ORDER BY" clause followed by a list of fields. The following query displays the last name and first name of registered clients from Singapore in descending alphabetical order of the first names. The keyword to get an ascending order is "ASC". This is the default mode and the keyword is often omitted. For descending order, add the keyword "DESC".

```
SELECT c.last_name, c.first_name
FROM customers c
WHERE c.country = 'Singapore'
ORDER BY c.first_name ASC;
```

NUS National University of Singapore

last_name	first_name
Griffin	Aaron
Green	Adam
Stone	Adam
Romero	Adam
Wijaya	Adam
Hansen	Alan
Richards	Alan
Porter	Alan
Khoo	Albert
Arnold	Albert
Torres	Albert
Dean	Albert
. . .	

It is possible to indicate a list of fields (or of calculated fields) in the "ORDER BY" clause. Additional fields will be used to sort records that have the same value for previous fields. The order of fields in the list is important. The query below displays the name, version and price of the games in ascending alphabetical order of the names and alphanumeric order of the versions. Two games with the same names are ordered according to their version.

```
SELECT g.name, g.version, g.price
FROM games g
ORDER BY g.name ASC, g.version DESC;
```

name	version	price
Aerified	3.0	3.99
Aerified	2.1	12
Aerified	2.0	5
Aerified	1.2	1.99
Aerified	1.1	3.99
Aerified	1.0	12
Alpha	3.0	5
Alpha	2.1	2.99
Alpha	2.0	12
. . .		

It is possible to sort the displayed result of a query according to fields that are not displayed. The query below displays the name of the games in descending numerical order of their price (from the most expensive to the cheapest). Note that the order fits the domain.

```
SELECT g.name FROM games g
ORDER BY g.price DESC;
```

This is why you can not use "`DISTINCT`" with "`ORDER BY`".

name
Duobam
It
Stim
Cookley
Daltfresh
Daltfresh
Daltfresh
Domainer
Sonsing
Domainer
Sonair
Redhold
Duobam
…

NUS National University of Singapore

Aggregate Queries: Counting Rows

Aggregate queries use aggregate functions like "COUNT()" to combine results over entire tables or columns.

Find the total number of (different) games.

```
SELECT COUNT(*)
FROM games g;
```

Find the average price of a game.

```
SELECT AVG(g.price)
FROM games g;
```

The following are some of the available aggregate functions: "COUNT()", "MAX()", "MIN()", "AVG()", "STD()", "SUM()" etc.

NUS National University of Singapore

Aggregate Queries: Counting Rows

Find the total number of (different) games.

```
SELECT COUNT(g.name)
FROM games g;
```

```
SELECT COUNT(ALL g.name)
FROM games g;
```

Find the total number of games, regardless of their version.

```
SELECT COUNT(DISTINCT a.name)
FROM games g;
```

count
430

count
82

NUS National University of Singapore

Aggregate Queries: Grouping

Find the total number of versions for each game. Print the name of the game and the number of versions.

```
SELECT g.name, COUNT(*)
FROM games g
GROUP BY g.name;
```

```
SELECT g.name, COUNT(g.version)
FROM games g
GROUP BY g.name;
```

name	count
'Matsoft'	6
'Hatity'	6
'Tampflex'	4
'Voyatouch'	5
'Cookley'	6
…	

NUS National University of Singapore

Aggregate Queries: Grouping

Let us order instead of grouping.

```
SELECT g.name, g.version
FROM games g
ORDER BY a.name;
```

We see the groups appearing.

The "GROUP BY" clause creates groups before the aggregation (but after the "WHERE" clause).

name	version
Aerified	1
Aerified	1.1
Aerified	1.2
Aerified	2
Aerified	2.1
Aerified	3
Alpha	1
Alpha	1.1
Alpha	1.2
Alpha	2
Alpha	2.1
Alpha	3
Alphazap	1.1
Alphazap	1.2
Alphazap	2
Alphazap	2.1
Alphazap	3

Aggregate Queries: Grouping

The `GROUP BY` clause creates groups.
We count for each group.

```
SELECT g.name,
COUNT(g.version)
FROM games g
GROUP BY g.name;
```

name	count
Aerified	6
Alpha	6
Alphazap	5

name	version	
Aerified	1	6
Aerified	1.1	
Aerified	1.2	
Aerified	2	
Aerified	2.1	
Aerified	3	
Alpha	1	6
Alpha	1.1	
Alpha	1.2	
Alpha	2	
Alpha	2.1	
Alpha	3	
Alphazap	1.1	5
Alphazap	1.2	
Alphazap	2	
Alphazap	2.1	
Alphazap	3	

Can we get 0?

NUS National University of Singapore

Aggregate Queries: Grouping

Find the total number of downloads for each game regardless of the version. Print the name of the games and the number of downloads.

```
SELECT d.name, COUNT(*)
FROM downloads d
GROUP BY d.name;
```

Find the total number of downloads for each game and version. Print the name of the games and the number of downloads.

```
SELECT d.name, COUNT(*)
FROM downloads d
GROUP BY d.name, d.version;
```

name	count
Aerified	73
Alpha	68
Alphazap	52
Andalax	40
Asoka	47
…	

name	count
Aerified	12
Aerified	10
Aerified	11
Aerified	13
Aerified	13
…	

Things that do not Work or should not Work

```
SELECT g.name, COUNT(*)
FROM games g;
```

Only attributes in the "GROUP BY" clause and aggregate functions can be used in the "SELECT" (and "HAVING") clauses.

SQLite does not yield an error but picks a random value!

Things that do not Work (even if they Makes Sense)

Find the total number of downloads for each game. Print the name, the version and the price of the game and the number of downloads.

```
SELECT d.name, d.version, g.price, COUNT(*)
FROM downloads d, game g
WHERE d.name = g.name AND d.version =
g.version
GROUP BY d.name, d.version;
```

It does not work! (except in SQLite).

NUS National University of Singapore

How things should be

Find the total number of downloads for each game. Print the name, the version and the price of the game and the number of downloads.

Only attributes in the "`GROUP BY`" clause and aggregate functions can be used in the "`SELECT`" (and "`HAVING`") clauses. Instead:

```
SELECT d.name, d.version, g.price, COUNT(*)
FROM downloads d, games g
WHERE d.name = g.name AND d.version =
g.version
GROUP BY d.name, d.version, g.price;
```

Aggregate Queries: Duplicate Elimination

Which one eliminates duplicate customers?

```
SELECT c.last_name
FROM customers c;
```

```
SELECT c.last_name
FROM customers c
GROUP BY c.last_name;
```

Interestingly, the "GROUP BY" clause can be used to eliminate duplicates. Sometimes it is the only way to do so. For readability of the queries, unless impossible, prefer "DISTINCT".

NUS National University of Singapore

Aggregate Queries: Duplicate Elimination

`"GROUP BY"` allows to eliminate duplicates of attributes that do not appear in the result.

Find the names of the different games downloaded and the corresponding number of downloads.

```
SELECT d.name, COUNT(*)
FROM downloads d
GROUP BY d.name, d.version;
```

NUS National University of Singapore

Aggregate Queries: Condition

Find the number of games in different versions downloaded by each customer. Print the identifier of the customer and the number of games.

```
SELECT d.customerid, COUNT(*) AS c
FROM downloads d
GROUP BY d.customerid
ORDER BY c;
```

Find the identifiers of customers who have downloaded more than 10 games in different versions.

```
SELECT d.customerid
FROM downloads d
GROUP BY d.customerid
HAVING COUNT(*) >= 10;
```

customerid
'Amy1990'
'Helen1992'
'Thomas2000'
'Matthew1996'
'Clarence1985'
…

Things that do not Work (or should not Work)

Find the name and version of the most expensive games.

```
SELECT g.name, g.version
FROM games g
WHERE g.price = MAX(g.price);
```

```
SELECT g.name, g.version
FROM games g
HAVING g.price = MAX(g.price);
```

NUS National University of Singapore

How Things should be Done

Find the name and version of the most expensive games

```
SELECT g.name, g.version
FROM games g
WHERE g.price >= ALL (
      SELECT g1.price FROM games g1);
```

NUS National University of Singapore

Subqueries in the HAVING Clause

Find the name and version of the games that have the largest number of downloads (ignore games that have not been downloaded).

```
SELECT d.name, d.version
FROM downloads d
GROUP BY d.name, d.version
HAVING COUNT(*) >= ALL (
      SELECT COUNT(*)
      FROM downloads d
      GROUP BY d.name, d.version);
```

name	version
'Y-find'	'1.2'

NUS National University of Singapore

Summary

Syntax

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING
6. ORDER BY

(UNION, INTERSECT, MINUS)

Semantics

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. ORDER BY
6. SELECT

(UNION, INTERSECT, MINUS)

NUS National University of Singapore

Credits

The content of this lecture is based on chapter 5 of the book "Introduction to database Systems" By S. Bressan and B. Catania, McGraw Hill publisher

Images and clips used in this presentation are licensed from Microsoft Office Online Clipart and Media

For questions about the content of this course and about copyrights, please contact Stéphane Bressan

steph@nus.edu.sg