

SQL Advanced 2: Aggregates and Nulls

22 December 2009
Lecture 10

Topics for Today

- Aggregates (finishing)
- Null values
- Complex integrity constraints

Source: RG 5.5-5.7

Group By / Having

- Break up the results into groups
 - Grouping conditions – GROUP BY
 - Checks on groups – HAVING

```
SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification
GROUP BY grouping-list
HAVING group-qualification
```

Having rules

- If you use a group-qualification
 - One value per group to compare
 - Decides whether include the group
- SQL:1999
 - New set functions per group
 - Any row: HAVING ANY (condition)
 - GROUP BY S.rating HAVING ANY S.sname = 'Bob'
 - All rows: HAVING ALL (condition)
 - GROUP BY S.rating HAVING ALL S.sname LIKE 'B%'
 - Not supported by MS SQL
- If you have a HAVING and no GROUP BY
 - The table is all one group

Example

- Find the age of the youngest sailor who is eligible to drink (i.e is at least 18 years old) for each rating level with at least two **such** sailors

```
SELECT S.rating, MIN(S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT(*) > 1
```

Sailors(sid:integer, sname:string, rating:integer, age:real)
Boats(bid:integer, bname:string, color:string)
Reserves(sid:integer, bid:integer, day:date)

Example

```
SELECT S.rating, MIN(S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT(*) > 1
```

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Analy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

rating	minage
3	25.5
7	35.0
8	25.5

Example

- Two more equivalent queries:

```
SELECT S.rating, MIN(S.age) AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING 1 < (SELECT COUNT(*)
           FROM Sailors S2
           WHERE S.rating = S2.rating
           AND S2.age >= 18)
```

```
SELECT Temp.rating, Temp.minage
FROM
  (SELECT S.rating, MIN(S.age) AS minage, COUNT(*) AS ratingcount
   FROM Sailors S
   WHERE S.age >= 18
   GROUP BY S.rating) AS Temp
WHERE Temp.ratingcount > 1
```

This is legal!

December 22, 2009

ISE 322: Database Systems

7

Harder Example

Find the rating(s) with the smallest average age

First try:

```
SELECT S.rating
FROM Sailors S
WHERE AVG(S.age) = (SELECT MIN(AVG(S2.age)) FROM
                   Sailors S2 GROUP BY S2.rating)
```

What's wrong?

December 22, 2009

ISE 322: Database Systems

8

Harder Example

Find the rating(s) with the smallest average age

Second try:

```
SELECT Temp.rating, Temp.avgage
FROM
  (SELECT S.rating, AVG(S.age) AS avgage
   FROM Sailors S
   GROUP BY S.rating) AS Temp
WHERE Temp.avgage <= ALL
      (SELECT AVG(S2.age)
       FROM Sailors S2
       GROUP BY S2.rating)
```

December 22, 2009

ISE 322: Database Systems

9

Harder Example

Find the rating(s) with the smallest average age

Second try:

```
SELECT Temp.rating, Temp.avgage
FROM
  (SELECT S.rating, AVG(S.age)
   AS avgage
   FROM Sailors S
   GROUP BY S.rating) AS Temp
WHERE Temp.avgage <= ALL
      (SELECT AVG(S2.age)
       FROM Sailors S2
       GROUP BY S2.rating)
```

- Is this query the same?

```
SELECT Temp.rating,
       MIN(Temp.avgage)
FROM
  (SELECT S.rating, AVG(S.age) AS
   avgage
   FROM Sailors S
   GROUP BY S.rating) AS Temp
GROUP BY Temp.rating
```

Why not?

December 22, 2009

ISE 322: Database Systems

10

So Far

- Aggregates (finishing)
- Null values
- Complex integrity constraints

December 22, 2009

ISE 322: Database Systems

11

Null values

- For unknown values, we use *null*
 - No equivalent in relational algebra
 - We have a 3-valued logic now
 - Introduce the **unknown** result in addition to true and false
- We can't compare using <, <=, >, >=, =
- Instead, we use "IS NULL" and "IS NOT NULL"

Example: Show the boats which do not have a color

```
SELECT *
FROM Boats B
WHERE B.color IS NULL
```

Example: Show the boats which do have a color

```
SELECT * FROM Boats B WHERE B.color IS NOT NULL
```

December 22, 2009

ISE 322: Database Systems

12

Null values

- Truth table:

x	op	y	result
T	AND	Unknown	Unknown
F	AND	Unknown	False
Unknown	AND	Unknown	False
T	AND	T	True
T	AND	F	False
F	AND	F	False
T	OR	Unknown	T
F	OR	Unknown	Unknown
Unknown	OR	Unknown	Unknown
T	OR	F	T
F	OR	F	F
T	OR	T	T
	NOT	Unknown	Unknown
	NOT	T	F
	NOT	F	T

December 22, 2009

ISE 322: Database Systems

13

Null Values

- Impact on SQL constructs
 - Excluded from WHERE clauses
 - EXISTS and UNIQUE (on subqueries) do not count them
- Two null rows are considered duplicates
 - With DISTINCT you get only one
- COUNT(*) counts all rows, even nulls
- Arithmetic ignore them
 - Unless *all* of the entries are null
- Disallowing null values
 - Use NOT NULL in the schema
 - All columns in the primary key are implicitly NOT NULL
- Outer joins
 - We discussed them before

December 22, 2009

ISE 322: Database Systems

14

So Far

- Aggregates (finishing)
- Null values
- Complex integrity constraints

December 22, 2009

ISE 322: Database Systems

15

Complex Integrity Constraints

- More powerful than the constraints we have seen until now
- Use the CHECK statement in the schema
 - Anything we can write in SQL can go in

December 22, 2009

ISE 322: Database Systems

16

Example: Range Constraint

- Ensure the rating of a sailor is between 1 and 10

```
CREATE TABLE Sailors (
  sid INTEGER,
  sname CHAR(10),
  rating INTEGER,
  age REAL,
  PRIMARY KEY (sid),
  CHECK (rating >= 1 AND rating <= 10))
```

Works in MS SQL.

December 22, 2009

ISE 322: Database Systems

17

Complex Integrity Constraints

- Ensure no boat named Interlake is reserved

Not supported in MS SQL.

```
CREATE TABLE Reserves (
  sid INTEGER,
  bid INTEGER,
  day DATETIME,
  FOREIGN KEY (sid) REFERENCES Sailors,
  FOREIGN KEY (bid) REFERENCES Boats,
  CONSTRAINT noInterlakeRes CHECK ('Interlake' <>
    (SELECT B.bname
     FROM Boats B
     WHERE B.bid = Reserves.bid)))
```

December 22, 2009

ISE 322: Database Systems

18

Complex Integrity Constraints

- Whenever rows are added or modified in Reserves, this is checked
 - If we change the name of a boat in the Boats table, this won't raise an error
- If it's false, the update is rejected
- MS SQL Server only supports scalar checks in the CHECK conditions, no subqueries
 - Vendor specific implementations
 - MySQL doesn't support CHECKs at all to my knowledge

December 22, 2009

ISE 322: Database Systems

19

Domain Constraints

- Create a custom type with specific integrity or range constraints
 - CREATE DOMAIN
- Creating a special type for rating value of a sailor
- You can then define fields
 - `rating ratingval`

```
CREATE DOMAIN ratingval INTEGER DEFAULT 1 CHECK  
( VALUE >= 1 AND VALUE <= 10)
```

December 22, 2009

ISE 322: Database Systems

20

Domain Constraints: MS SQL

```
CREATE TYPE type_name FROM base_type [ ( precision [ , scale ] ) ] [ NULL | NOT NULL ]
```

- Type_name: the new alias
- Base_type: the system base type
- Precision: (numeric) the number of digits it holds (total)
- Scale: (numeric) the number of digits past the decimal (<= precision)
- NULL/NOT NULL: whether the type can hold NULL (default NULL)
- Examples:
 - CREATE TYPE ratingval FROM INTEGER
 - CREATE TYPE TZ FROM varchar(11) NOT NULL
- You may compare the new types against the original base type

December 22, 2009

ISE 322: Database Systems

21

Conclusion

- Aggregates (finishing)
- Null values
- Complex integrity constraints

December 22, 2009

ISE 322: Database Systems

22