

SQL

Chapter 5

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Basic SQL Query

```
SELECT [DISTINCT] target-list
FROM relation-list
[WHERE condition]
```

```
SELECT S.sname
FROM Sailors S
WHERE S.age > 25
```

```
SELECT DISTINCT S.sname
FROM Sailors S
WHERE S.age > 25
```

- Default is that duplicates are *not* eliminated!
 - Need to explicitly say "DISTINCT"

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

SQL Query

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

Sailors

| sid | sname | rating | age |
|-----|--------|--------|------|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 58 | rusty | 10 | 35.0 |

Reserves

| sid | bid | day |
|-----|-----|----------|
| 22 | 101 | 10/10/96 |
| 58 | 103 | 11/12/96 |

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Conceptual Evaluation Strategy

```
SELECT [DISTINCT] target-list
FROM relation-list
[WHERE condition]
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Example of Conceptual Evaluation

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

| (sid) | sname | rating | age | (sid) | bid | day |
|-------|--------|--------|------|-------|-----|----------|
| 22 | dustin | 7 | 45.0 | 22 | 101 | 10/10/96 |
| 22 | dustin | 7 | 45.0 | 58 | 103 | 11/12/96 |
| 31 | lubber | 8 | 55.5 | 22 | 101 | 10/10/96 |
| 31 | lubber | 8 | 55.5 | 58 | 103 | 11/12/96 |
| 58 | rusty | 10 | 35.0 | 22 | 101 | 10/10/96 |
| 58 | rusty | 10 | 35.0 | 58 | 103 | 11/12/96 |

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

A Slightly Modified Query

```
SELECT S.sid
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND R.bid=103
```

- Would adding DISTINCT to this query make a difference?

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: Find sid's of sailors who've reserved a red or a green boat

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: What Does This Query Compute?

```
SELECT S.sid
FROM Sailors S, Boats B1, Reserves R1, Boats B2, Reserves R2
WHERE S.sid=R1.sid AND R1.bid=B1.bid AND
      S.sid=R2.sid AND R2.bid=B2.bid AND
      B1.color='red' AND B2.color='green'
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: Find sid's of sailors who've reserved a red and a green boat

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Expressions and Strings

```
SELECT S.age, S.age-5 AS age1, 2*S.age AS age2
FROM Sailors S
WHERE S.sname LIKE 'B_%B'
```

- AS is used to name fields in result.
- LIKE is used for string matching
 - '_' stands for any one character
 - '%' stands for 0 or more arbitrary characters.

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Nested Queries (with Correlation)

Find names of sailors who have reserved boat #103:

```
SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
              FROM Reserves R
              WHERE R.bid=103 AND S.sid=R.sid)
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Nested Queries (with Correlation)

*Find names of sailors who have **not** reserved boat #103:*

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (SELECT *
                  FROM Reserves R
                  WHERE R.bid=103 AND S.sid=R.sid)
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Division in SQL

Find sailors who've reserved all boats

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS ((SELECT B.bid
                   FROM Boats B)
                  EXCEPT
                  (SELECT R.bid
                   FROM Reserves R
                   WHERE R.sid=S.sid))
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Division in SQL (without Except!)

Find sailors who've reserved all boats.

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (SELECT B.bid
                  FROM Boats B
                  WHERE NOT EXISTS (SELECT R.bid
                                    FROM Reserves R
                                    WHERE R.bid=B.bid
                                       AND R.sid=S.sid))
```

Sailors S such that ...
there is no boat B without ...
a Reserves tuple showing S reserved B

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

More on Set-Comparison Operators

↻ *op* ANY, *op* ALL

- *op* can be >, <, =, >=, <=, <>

↻ Find sailors whose rating is greater than that of all sailors called Horatio:

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Aggregate Operators

☞ Significant extension of relational algebra.

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)

single column

```
SELECT COUNT (*)  
FROM Sailors S  
  
SELECT AVG (S.age)  
FROM Sailors S  
WHERE S.rating=10  
  
SELECT COUNT (DISTINCT S.rating)  
FROM Sailors S  
WHERE S.sname='Bob'
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: Find name and age of the oldest sailor(s) with rating > 7

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Aggregate Operators

- So far, we've applied aggregate operators to all (qualifying) tuples
- Sometimes, we want to apply them to each of several groups of tuples.
- Consider: *Find the age of the youngest sailor for each rating level.*
 - If rating values go from 1 to 10; we can write 10 queries that look like this:

```
For i = 1, 2, ..., 10: SELECT MIN (S.age)  
FROM Sailors S  
WHERE S.rating = i
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

GROUP BY

```
SELECT [DISTINCT] target-list
FROM relation-list
[WHERE condition]
GROUP BY grouping-list
```

Find the age of the youngest sailor for each rating level

```
SELECT S.rating, MIN(S.Age)
FROM Sailors S
GROUP BY S.rating
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Conceptual Evaluation Strategy

```
SELECT [DISTINCT] target-list
FROM relation-list
[WHERE condition]
GROUP BY grouping-list
```

- Semantics of an SQL query defined as follows:

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Find the age of the youngest sailor with age ≥ 18 , for each rating with at least one such sailor

```
SELECT S.rating, MIN(S.age)
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
```

| sid | sname | rating | age |
|-----|---------|--------|------|
| 29 | brutus | 1 | 33.0 |
| 22 | dustin | 7 | 45.0 |
| 64 | horatio | 7 | 35.0 |
| 58 | rusty | 10 | 35.0 |

| sid | sname | rating | age |
|-----|---------|--------|------|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 15.5 |
| 71 | zorba | 10 | 16.0 |
| 64 | horatio | 7 | 35.0 |
| 29 | brutus | 1 | 33.0 |
| 58 | rusty | 10 | 35.0 |

| rating | |
|--------|------|
| 1 | 33.0 |
| 7 | 35.0 |
| 10 | 35.0 |

Answer relation

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: What does this query compute?

```
SELECT B.bid, COUNT (*) AS scout
FROM Reserves R, Boats B
WHERE R.bid=B.bid AND B.color='red'
GROUP BY B.bid
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: Find those ratings for which the average age is the minimum over all ratings

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: What does this query compute?

```
SELECT Temp.rating, Temp.minage
FROM (SELECT S.rating, MIN (S.age) AS minage, COUNT(*) AS cnt
      FROM Sailors S
      WHERE S.age >= 18
      GROUP BY S.rating) AS Temp
WHERE Temp.cnt >= 2
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Queries With GROUP BY and HAVING

```
SELECT [DISTINCT] target-list
FROM relation-list
[WHERE qualification]
GROUP BY grouping-list
HAVING group-qualification
```

Find the age of the youngest sailor with age ≥ 18 for each rating level with at least 2 such sailors

```
SELECT S.rating, MIN(S.Age)
FROM Sailors S
WHERE S.age  $\geq 18$ 
GROUP BY S.rating
HAVING COUNT(*)  $\geq 2$ 
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Conceptual Evaluation Strategy

- Semantics of an SQL query defined as follows:
 - Compute the cross-product of *relation-list*
 - Discard resulting tuples if they fail *condition*.
 - Delete attributes that are not in *target-list*
 - Remaining tuples are partitioned into groups by the value of the attributes in *grouping-list*
 - The *group-qualification* is applied to eliminate some groups
 - One answer tuple is generated per qualifying group
- Note: Does not imply query will actually be evaluated this way!

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Find the age of the youngest sailor with age ≥ 18 , for each rating with at least 2 such sailors

```
SELECT S.rating, MIN(S.age)
FROM Sailors S
WHERE S.age  $\geq 18$ 
GROUP BY S.rating
HAVING COUNT(*)  $> 1$ 
```

| sid | sname | rating | age |
|-----|---------|--------|------|
| 22 | dustin | 7 | 45.0 |
| 31 | lubber | 8 | 55.5 |
| 71 | zorba | 10 | 16.0 |
| 64 | horatio | 7 | 35.0 |
| 29 | brutus | 1 | 33.0 |
| 58 | rusty | 10 | 35.0 |

| rating | age |
|--------|------|
| 1 | 33.0 |
| 7 | 45.0 |
| 7 | 35.0 |
| 8 | 55.5 |
| 10 | 35.0 |

| rating | age |
|--------|------|
| 7 | 35.0 |

Answer relation

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

ICE: Find the age of the youngest sailor with age ≥ 18 , for each rating with at least 2 sailors (of any age)

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Find the average age for each rating, and order results in ascending order on avg. age

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

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Null Values

- Field values in a tuple are sometimes *unknown*
 - e.g., a rating has not been assigned
- Field values are sometimes *inapplicable*
 - e.g., no spouse's name
- SQL provides a special value *null* for such situations.

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Queries and Null Values

```
SELECT S.Name
FROM Sailors S
WHERE S.Age > 25
```

- What if S.Age is NULL?

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Three-valued Logic

```
SELECT S.Name
FROM Sailors S
WHERE NOT(S.Age > 25) OR S.rating > 7
```

- What if one or both of S.age and S.rating are NULL?

NOT Truth Table

| A | NOT(A) |
|-------|--------|
| True | |
| False | |
| NULL | |

OR Truth Table

| A/B | True | False | NULL |
|-------|------|-------|------|
| True | | | |
| False | | | |
| NULL | | | |

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

General Constraints

```
CREATE TABLE Reserves
( sname CHAR(10) NOT NULL,
  bid INTEGER NOT NULL,
  day DATE NOT NULL,
  PRIMARY KEY (bid,day),
  CONSTRAINT noInterlakeRes
  CHECK ('Interlake' <>
        (SELECT B.bname
         FROM Boats B
         WHERE B.bid=bid)))
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

Constraints Over Multiple Relations

*Number of boats
plus number of
sailors is < 100*

```
CREATE ASSERTION smallClub  
CHECK  
( (SELECT COUNT (S.sid) FROM Sailors S)  
+ (SELECT COUNT (B.bid) FROM Boats B) < 100 )
```
