Database Management Systems

Ehsan Noei
e.noei@utoronto.ca

UNIVERSITY OF TORONTO

SQL Language

- The Data Manipulation Language (DML)
- The Data Definition Language (DDL)
- Triggers and Advanced Integrity Constraints
- Embedded and Dynamic SQL
- Client-Server Execution and Remote Database Access
- Transaction Management
- Advanced Features

The Data Definition Language (DDL)

- This subset of SQL supports the creation, deletion, and modification of definitions for tables and views.

The Data Manipulation Language (DML)

- This subset of SQL allows users to pose queries and to insert, delete, and modify rows.

Triggers

- This subset of SQL allows users to pose queries and to insert, delete, and modify rows.

Embedded and Dynamic SQL

- Embedded SQL features allow SQL code to be called from a host language.

Client-Server Execution and Remote Database Access

- These commands control how a *client* application program can connect to an SQL database *server,* or access data from a database over a network.

Transaction Management

- Various commands allow a user to explicitly control aspects of how a transaction is to be executed.

Advanced Features

- Security
- Object-oriented features
- Recursive queries
- etc.

Basic SQL Query

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

Basic SQL Query

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

- *relation-list* A list of relation names (possibly with a *range-variable* after each name).
- *target-list* A list of attributes of relations in *relation-list*
- *qualification* Comparisons (Attr *op* const or Attr1 *op* Attr2, where *op* is one of $<, >, =, \leq, \geq, \neq$) combined using AND, OR and NOT.

Basic SQL Query

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

- DISTINCT is an optional keyword indicating that the answer should not contain duplicates.
- Default is that duplicates are *not* eliminated!

SQL Query Language

```
SELECT Name, Salary
        FROM Employee
        WHERE eID = ‘123’
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
        WHERE eID = ‘123’
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
        WHERE Name = ‘Ehsan’ AND Salary = ‘10$’
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
        WHERE Name = ‘Ehsan’ OR Salary = ‘10$’
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT *
        FROM Employee
```

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

SQL Query Language

```
SELECT E.Salary, N.Address
      FROM Employee E, Manager N
      WHERE E.Name = N.Name
```

Employee

eID	Name	Salary
123	Ehsan	$10
321	Steve	$9

Manager

mID	Name	Address
666	Joe	CA
667	Steve	CA

Conceptual Evaluation Strategy

- Semantics of an SQL query defined in terms of the following conceptual evaluation strategy:
  - Compute the cross-product of *relation-list*.
  - Discard resulting tuples if they fail *qualifications*.
  - Delete attributes that are not in *target-list*.
  - If DISTINCT is specified, eliminate duplicate rows.
- This strategy is probably the least efficient way to compute a query! An optimizer will find more efficient strategies to compute *the same answers*.

Example

Sailors (sid: integer, sname: string, rating: integer, age: real)

Boats (bid: integer, bname: string, color: string)

Reserves (sid: integer, bid: integer, day: date)

R1

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

S1

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

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

A Note on Range Variables

- Really needed only if the same relation appears twice in the FROM clause. The previous query can also be written as:

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

OR

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

It is good style, however, to use range variables always!

Find the' names and ages of all sailors

```
SELECT S.sname, S.age
FROM Sailors S
```

Find the' names and ages of all sailors

```
SELECT DISTINCT S.sname, S.age
FROM Sailors S
```

Find all sailors with a rating above 7.

```
SELECT S.sid, S.sname, S.rating, S.age
FROM Sailors S
WHERE S.rating > 7
```

Find all sailors with a rating above 7.

```
SELECT *
FROM Sailors S
WHERE S.rating > 7
```

Find the sid of sailors who've reserved a red boat

```
SELECT  R.sid
FROM  Boats B, Reserves R
WHERE  R.bid=B.bid AND B.color =  'red'
```

- Query contains a join of two tables (cross product), followed by a selection on the color of boats
- If we wanted the name of the sailors, we must include the Sailors relation as well

Find the name of sailors who've reserved a red boat

```
SELECT  S.sname
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color = 'red'
```

- Query contains a join of three tables, followed by a selection on the color of boats

Find the colors of boats reserved by Lubber.

SELECT B.color

FROM Sailors S, Reserves R, Boats B

WHERE S.sid = R.sid AND R.bid = B.bid AND S.sname = 'Lubber'

Find sailors who've reserved at least one boat

```
SELECT DISTINCT S.name
FROM Sailors S, Reserves R
WHERE  S.sid=R.sid
```

Rename output columns

```
Select S.name AS NewName
From Sailors S
Where S.sid = ‘1’
```

Expressions and Strings

- SQL supports a more general version of the select-list than just a list of columns.

Compute increments for the rating of persons who have sailed two different boats on the same day.

```
SELECT S.sname, S.rating+1 AS rating
FROM Sailors S, Reserves R1, Reserves R2
WHERE S.sid = R1.sid AND S.sid = R2.sid
        AND R1.day = R2.day AND R1.bid <> R2.bid
```

Expressions and Strings

- Each item in a *qualification* can also be as general as *expression1 = expression2.*

```
SELECT S1.sname AS name1, S2.sname AS name2
FROM Sailors S1, Sailors S2
WHERE 2*S1.rating = S2.rating-1
```

Find the ages of sailors whose name begins and ends with B and has at least three characters.

SELECT S.age

FROM Sailors S

Where S.sname LIKE ‘B_%B'

Expressions and Strings

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

- Illustrates use of arithmetic expressions and string pattern matching: *Find triples (of ages of sailors and two fields defined by expressions) for sailors whose names begin with B and contain at least two characters.*
- AS is a way to name fields in the result.
- LIKE is used for string matching. `_’ stands for any one character and `%’ stands for 0 or more arbitrary characters.

UNION, INTERSECT, AND EXCEPT

- SQL provides three set-manipulation constructs that extend the basic query form presented earlier.
- Since the answer to a query is a multiset of rows, it is natural to consider the use of operations such as union, intersection, and difference.

- Find sid's of sailors who've reserved a red or a green boat.

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid
  AND (B.color= 'red'  OR B.color= 'green' )
```

- If we replace OR by AND, what do we get?

Range

- Find sid's of sailors who've reserved a red and a green boat.

```
SELECT S.sname
FROM Sailors S, Reserves R1, Boats B1, Reserves R2, Boats B2
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'
```

- Difficult to understand
- Inefficient to execute

- Find sid's of sailors who've reserved a red and a green boat.

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
             AND B.color= 'red'
INTERSECT
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
             AND B.color= 'green'
```

- Included in the SQL/92 standard, but some systems don’t support it.

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
            AND B.color= ‘red’
INTERSECT
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
            AND B.color= ‘green’
```

- Find sid’s of sailors who’ve reserved a red or a green boat.

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid
  AND (B.color= ‘red’  OR B.color= ‘green’ )
```

- Find sid’s of sailors who’ve reserved a red or a green boat.

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
            AND B.color= ‘red’
UNION
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
            AND B.color= ‘green’
```

- Find the sids of all sailor's who have reserved red boats but not green boats.

```
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
          AND B.color= ‘red’
EXCEPT
SELECT  S.sid
FROM  Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND
R.bid=B.bid
          AND B.color= ‘green’
```

- Find the sids of all sailor's who have reserved red boats but not green boats.

```
SELECT  S.sid
FROM Boats B, Reserves R
WHERE  R.bid=B.bid
          AND B.color= ‘red’
EXCEPT
SELECT  S.sid
FROM Boats B, Reserves R
WHERE  R.bid=B.bid
          AND B.color= ‘green’
```

- Find all sids of sailors who have a rating of 10 or reserved boat 104.

```
SELECT S.sid
FROM Sailors S
WHERE S.rating = 10
UNION
SELECT R.sid
FROM Reserves R
WHERE R.bid = 104
```