SQL/DML (Data Manipulation Sublanguage)

References

- Elmasri & Navathe – Ch 6 + Ch 7
- The SAILORS DB that we implemented in Oracle for class use.
  - ***The examples below are from that database***
- Oracle's online SQL documentation: **http://www.cis.gvsu.edu/Facilities/EOS/#oracle**

Agenda

1. The Basic Form of a SQL Query
2. Conceptual Evaluation
3. The SAILORS DB Instance
4. SQL Queries: Various types

1. The Basic Form of a SQL Query

- The basic form of a SQL query block is:

SELECT	*<target list (a list of attributes and/or functions)>*	- mandatory
FROM	*<table list. >*	- mandatory
WHERE	*<condition. >*	- optional
GROUP BY	*<grouping attribute(s)>*	- optional
HAVING	*<grouping condition >*	- optional
ORDER BY	*<attribute list>*	- optional

- The FROM clause can contain an ***inline query***.
- The WHERE and HAVING clauses can contain ***nested subqueries***.

  A query can be made up of several nested and/or non-nested subqueries. ***Draw diagrams here***

  - ***A query consisting of one block only.***

  - ***A query containing two non-nested subqueries.***

  - ***A query containing a nested subquery.***

2. Conceptual Evaluation

- Conceptual evaluation helps only in understanding the result of the query.
- The DBMS does not use conceptual evaluation to actually evaluate the query. Rather, it does it in a more efficient way.

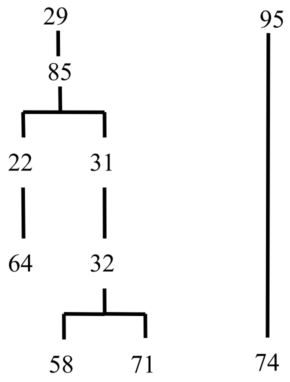
- The steps in conceptualizing query evaluation are:

  1. The cross product of the tables in the **`FROM`** clause is generated.
  2. The cross product is restricted by applying all the conditions in the **`WHERE`** clause on each row in the cross product individually.
  3. Groups are generated according to the grouping attributes mentioned in the **`GROUP BY`** clause.
  4. The **`HAVING`** clause is applied to each group individually.
  5. The **`SELECT`** clause is applied to retain only those attributes that are mentioned in it.
  6. The **`ORDER BY`** clause is applied to sort the result.

3. The SAILORS DB Instance

Sailors	SID	SNAME	RATING	AGE	TRAINEE
	22	Dave	7	45	85
	29	Mike	1	33	
	31	Mary	8	55	85
	32	Albert	8	25	31
	58	Jim	10	35	32
	64	Jane	7	35	22
	71	Dave	10	16	32
	74	Jane	9	40	95
	85	Art	3	25	29
	95	Jane	3	63	

Is_trained_by relationship
(e.g. 85 is trained by 22 and 31)



Boats …	BID	BNAME	COLOR	RATE	LENGTH	logKeeper
	101	Interlake	blue	350	30	95
	102	Interlake	red	275	23	22
	103	Clipper	green	160	15	85
	104	Marine	red	195	24	22
	105	Weekend Rx	white	500	43	31
	106	C#	red	300	27	32
	107	Bayside	white	350	32	85
	108	C++	blue	100	12	95

Reservations	BID	forDate	SID	onDate
	101	10-OCT-12	22	07-OCT-12
	102	14-OCT-12	22	10-OCT-12
	103	17-NOV-12	22	10-OCT-12
	105	14-OCT-12	58	13-OCT-12
	102	20-OCT-12	31	10-OCT-12
	103	22-NOV-12	31	20-OCT-12
	104	23-NOV-12	31	20-OCT-12
	101	05-SEP-12	64	27-AUG-12
	102	20-NOV-12	64	03-NOV-12
	103	18-OCT-12	74	04-AUG-12

Acknowledgment: This database is an extended & enhanced version from one that appears in Ramakrishnan & Gehrke's book

4. SQL Queries: A sample

Important Notes:

- ***See the textbook for numerous other good examples of queries***

- Our purpose here is to learn important SQL/DML constructs.
  - Therefore, we often write the same query in the examples that follow in different ways using different constructs.

- Different optimizers differ in how well they optimize different query structures.

  - To be on the safe side, write queries in the simplest forms
  - Avoid the use of inline queries in the FROM clause in queries
  - Avoid, whenever possible, the use of nested subqueries – optimizers handle joins much more efficiently.

- ***Why learn SQL/DML well?***

1. GUI tools (e.g. QBE in Access) do not support the full SQL capabilities.
2. In most applications, SQL in embedded in programs; two options exist then:

    - The BAD option: Have a simple SQL query get the data from the database, and then post-process it in the program.

    - The GOOD option: Have a sophisticated SQL query get and post-process the data, and then deliver it to the program. Advantages include:
        - The DBMS will do the optimization of data access and post-processing.
        - Program code will be simpler.
        - Data traffic will be less.

    - A very simple example

Find how many people over 21 reside in each zip code that has more than 10,000 such people.

MichiganPeople (SSnum, name, …, …, age, …, …, zipcode, …, …)

BAD idea:
The program issues:
 	
SELECT *
FROM MichiganPeople
WHERE age > 21;
 	
Then the program has to do the rest of data extraction.
	GOOD idea:
	The program issues:
	
	SELECT zipCode, COUNT(*)
	FROM MichiganPeople
	WHERE age > 21
	GROUP BY zipcode
	Having COUNT (*)>10,000;
	
	Done by the DBMS!!!

It is often helpful to draw a query graph before writing the SQL code for complex queries. We'll present several examples of graphs in class when we discuss the queries below.

/*(10) A simple one-relation query

Find the sid and name of every sailor whose rating is above 7 and whose age is less than 40.

*/

```
SELECT     sid, sname
FROM       Sailors
WHERE      rating > 7 AND age < 40;
```

/*(12) Strings manipulation … also see the SQL manual for the richness of string processing in SQL.

Find every sailor whose name contains an 'a' as the second letter and contains an 'e'

*/

```
SELECT *
FROM   Sailors
WHERE  sname LIKE '_a%'  AND
        sname LIKE '%e%';
```

/*(14) Comparing `NULL` values … more on NULL later in these notes.

Find every sailor who doesn't train anybody.

*/

```
SELECT *
FROM   Sailors
WHERE  trainee IS NULL;   -- What does trainee = NULL do and mean?
```

Beware NULLS!!! Try the above query by trying:

```
      WHERE  trainee = NULL;

      WHERE  trainee != NULL;
```

/*(15) THIS IS AN IMPORTANT BASE QUERY … it demos:

- Cross Product + Conceptual Evaluation + Query Graph

Experiment with it by adding various conditions in the WHERE clause, and by modifying the SELECT clause.

*/

```
SELECT *
FROM   Sailors S, Boats B, Reservations R;
```

/*(20) Joining two tables – *Draw a query graph*

Find the sid and name of every sailor whose rating is above 7 and who has a reservation.

*/

```
SELECT S.sid, S.sname
FROM   Sailors S, Reservations R
WHERE  S.rating > 7 AND
       S.sid = R.sid;
```

/*(22) Removing the duplicates from query (20)

Find the sid and name of every sailor whose rating is above 7 and who has a reservation. Remove duplicate rows in the result

*/

```
SELECT DISTINCT S.sid, S.sname
FROM   Sailors S, Reservations R
WHERE  S.rating > 7   AND
       S.sid = R.sid;
```

/*(30) Joining three tables – *Draw a query graph*
Find the sid and name of every sailor who has reserved a red **OR** a green boat; remove duplicates.
*/

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

/*(32) WATCH OUT!!!!
Find the sid and name of every sailor who has reserved a red **AND** a green boat.
*/
/* This will ***NOT work*** –*/

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

/*(40) Self-join (joining a table with itself)
Find pairs of sids where the first sailor (in the pair) has a rating above 7 and the second sailor (in the pair) has a similar rating. List each pair once only.
*/

```
SELECT S1.sid, S2.sid
FROM   Sailors S1, Sailors S2
WHERE  S1.rating > 7          AND
       S1.rating = S2.rating  AND
       S1.sid < S2.sid;
```

/*(42) A 2-table, 4-range-variable query
Find pairs of sid's of sailors (and their names) where the first sailor (in the pair) has made a reservation for the same day as the second sailor (in the pair). List each pair once only.
*/

```
SELECT S1.sid, S1.sname, S2.sid, S2.sname
FROM   Sailors S1, Sailors S2, Reservations R1, Reservations R2
WHERE  S1.sid = R1.sid        AND
       S2.sid = R2.sid        AND
       R1.forDate = R2.forDate   AND
       S1.sid < S2.sid;
```

/*(44) Using set operations

***1- UNION, INTERSECT, and MINUS remove duplicates … no need for DISTINCT here**.)*

2- Watch out for union compatibility

Find the sid and name of every sailor whose rating is above 7 OR has a reservation.

*/

```
SELECT S.sid, S.sname
FROM   Sailors S
WHERE  rating > 7
UNION
SELECT S.sid, S.sname
FROM   Sailors S, Reservations R
WHERE  S.sid = R.sid;
```

/*(50) Using set operations - Remember the union compatibility - Also compare with queries (52) and (54)

Find the sid's and names of sailors whose rating is >7 and have not reserved any boats.

*/

```
SELECT S.sid, S.sname
FROM   Sailors S
WHERE  rating > 7
MINUS
SELECT S.sid, S.sname
FROM   Sailors S, Reservations R
WHERE  S.sid = R.sid;
```

/*(52) Using a non-correlated subquery ... compare with (50) and (54)

Find the sid's and names of sailors whose rating is >7 and have not reserved any boats.

*/

```
SELECT S.sid, S.sname
FROM   Sailors  S
WHERE  S.rating > 7  AND
       S.sid NOT IN (SELECT R.sid
                     FROM   Reservations R);
```

/*(54) Using a correlated subquery ... compare with (50) and (52)

Find the sid's and names of sailors whose rating is >7 and have not reserved any boats.

*/

```
SELECT S.sid, S.sname
FROM   Sailors  S
WHERE  S.rating > 7  AND
       NOT EXISTS (SELECT *
                   FROM   Reservations R  /*don't add: Sailors S here!!*/
                   WHERE  R.sid = S.sid);
```

/*(56) Using a view ... Compare with (50), (52), and (54)

Find the sid's and names of sailors whose rating is >7 and have not reserved any boats.

*/

```
SELECT L.sid, L.sname
FROM   LazySailors L
WHERE  L.rating > 7;
```

```
/*(60) Using set comparison - compare with (62)
Find the sid and rating of every sailor whose rating is above the rating of EVERY sailor who has reserved
boat 103
*/
SELECT S.sid, S.rating
                            FROM   Sailors S
       WHERE  S.rating > ALL      -- experiment with ANY and SOME
       (SELECT S.rating
        FROM   Sailors S, Reservations R
        WHERE  S.sid = R.sid  AND
               R.bid = 103);
```

```
/*(62) Using an aggregation function - compare with (60)
Find the sid and rating of every sailor whose rating is above the rating of EVERY sailor who has reserved
boat 103
*/
SELECT S.sid, S.rating
FROM   Sailors S
WHERE  S.rating >
       (SELECT MAX (S.rating)
        FROM   Sailors S, Reservations R
        WHERE  S.sid = R.sid  AND
               R.bid = 103);
```

```
/*(65) Counting
Find out how many different ratings there are.
*/
SELECT COUNT (DISTINCT rating)
FROM   Sailors;
```

```
/*(66) Aggregation
Find the maximum, average, and minimum age of all sailors.
*/
SELECT MAX(age) AS maxAge, AVG(age) AS averageAge, MIN(age) as minAge
FROM   Sailors;
```

/*(70) Using GROUP BY

Note: Generally, the attributes list must be a subset of the GROUP BY list.

Find the sid, name, and the number of boats reserved by every sailor whose rating is above 2.

*/

```
SELECT   S.sid, S.sname, COUNT(*)
FROM     Reservations R,  Sailors S
WHERE    S.sid = R.sid  AND
         S.rating > 2
GROUP BY S.sid, S.sname;
```

/*(72) Using GROUP BY and HAVING ... (HAVING is **not** WHERE)

Find the sid, name, and the number of boats reserved by every sailor who has reserved more than two boats.

*/

```
SELECT   S.sid, S.sname, COUNT(*)
FROM     Reservations R,  Sailors S
WHERE    S.sid = R.sid
GROUP BY S.sid, S.sname
HAVING   COUNT(*) > 2;
```

/*(73) Another GROUP BY and HAVING query

Find the sid, and name of every sailor who has reserved more than one **red** boat.

*/

```
SELECT   S.sid, S.sname
FROM     Reservations R, Sailors S, Boats B
WHERE    S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
GROUP BY S.sid, S.sname
HAVING   COUNT(*) > 1;
```

/*(74) Another GROUP BY query

Find the sid, name, and sum of the rates of all boats reserved by sailor(s) named Jane

*/

```
SELECT  S.sid, S.sname, SUM (B.rate)
FROM    Sailors S, Reservations R, Boats B
WHERE   S.sname = 'Jane' AND
        S.sid = R.sid      AND
        R.bid = B.bid
GROUP BY  S.sid, S.sname;
```

/*(75) Another GROUP BY query… ***Compare it with the next query***

Find the average age of sailors who are at least 18 years old for each rating that has more than one **such** sailor.

*/

```
SELECT S.rating, AVG (S.age), count (*)
FROM   Sailors S
WHERE  S.age >= 18
GROUP BY S.rating
HAVING  COUNT(*) > 1;
```

/*(76) Another GROUP BY query … ***Compare it with the previous query***

Find the average age of sailors who are at least 18 years old for each rating that has more than **one sailor**.

*/

```
SELECT  S.rating, AVG (S.age), count(*)
FROM    Sailors S
WHERE   S.age >= 18
GROUP BY S.rating
HAVING  (SELECT COUNT(*)
         FROM Sailors S2
         WHERE S2.rating = S.rating) > 1;
```

/*(79) **…. …. AVOID this ABUSE of GROUP BY ……**

What does this query mean? Answer, and then write a simple version of it.

*/

```
SELECT   S.sid, S.sname
FROM     Reservations R, Sailors S, Boats B
WHERE    S.sid = R.sid AND R.bid = B.bid
GROUP BY S.sid, S.sname, B.color
HAVING   B.color = 'red';
```

Quizz: Write a simple version of the above query.

```
/*(80) Our default (INNER) join ... compare with (81)
Find the sid, name, bid, and date for every sailor who has a reservation.
*/
SELECT S.sid, S.sname, R.bid, R.forDate
FROM   Sailors S, Reservations R
WHERE  S.sid = R.sid;
```

```
/*(81) LEFT OUTER JOIN ... Compare with (80)
Find the sid, name, bid, and date for every sailor. Also show the reservations for those who have them.
*/
SELECT S.sid, S.sname, R.bid, R.forDate
FROM   Sailors S LEFT OUTER JOIN Reservations R ON S.sid=R.sid;
```

```
/*(100) Simulating the relational algebra DIVISION operation
Find the sid and name of every sailor who has reserved every boat named 'Interlake'.
*/
SELECT S.sid, S.sname
FROM   Sailors S
WHERE  NOT EXISTS((SELECT  B.bid
                    FROM Boats B
                    WHERE B.bname = 'Interlake')
                  MINUS
                  (SELECT  B.bid
                    FROM    Reservations R, Boats B
                    WHERE   R.sid = S.sid  AND
                            R.bid = B.bid  AND
                            B.bname = 'Interlake'));
```

```
/*(120)  An inline subquery in the FROM clause – presented as a demo, and is NOT recommended
Find the  sailor's sid and name, together with the boat id and name, for every sailor who has reserved a red boat.
*/
SELECT S.sid, S.sName, Reds.bid, Reds.bName
FROM   Sailors S,
       (SELECT DISTINCT R.sid, B.bid, B.bName
        FROM   Reservations R, Boats B
        WHERE  R.bid = B.bid  AND
               B.color = 'red') Reds
WHERE  S.sid = Reds.sid;
```