SQL Overview

• Query capabilities
  – SELECT-FROM-WHERE blocks,
  – Basic features, ordering, duplicates
  – Set ops (union, intersect, except)
  – Aggregation & Grouping
  – Nested queries (correlation)
  – Null values
Nested queries

- A **nested query** is a query with another query embedded within it.
- The embedded query is called the **subquery**.
- The subquery usually appears in the **WHERE** clause:

```sql
SELECT S.sname
FROM Sailors S
WHERE S.sid IN ( SELECT R.sid
                 FROM Reserves R
                 WHERE R.bid = 103 )
```

(Subqueries also possible in FROM or HAVING clause.)
Conceptual evaluation, extended

- For each row in cross product of outer query, evaluate the WHERE clause conditions, (re)computing the subquery.

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

equivalent to:

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

• If the inner subquery depends on tables mentioned in the outer query then it is a **correlated subquery**.

• In terms of conceptual evaluation, we must recompute subquery for each row of outer query.

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

• Optional NOT may precede these:
  – EXISTS R -- true if R is non-empty
  – attr IN R -- true if R contains attr
  – UNIQUE R -- true if no duplicates in R

• For arithmetic operator $\text{op} \{<,\leq,=,\geq,>,\geq,>\}$
  – $\text{op} \text{ ALL}$ -- all elements of R satisfy condition
  – attr $\text{op} \text{ ANY R}$ -- some element of R satisfies condition

IN equivalent to = ANY
NOT IN equivalent to <> ALL
Example

• Find the sailors with the highest rating

```
SELECT S.sid
FROM Sailors S
WHERE S.rating >= ALL (SELECT S2.rating
FROM Sailors S2 )
```
Please write SQL

• Find sailors whose rating is higher than **some** sailor named Horatio.

```sql
SELECT S.sid
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
FROM Sailors S2
S2.name = 'Horatio')
```

• Find sailors whose rating is higher than **all** sailors named Horatio.

```sql
SELECT S.sid
FROM Sailors S
WHERE S.rating > ALL (SELECT S2.rating
FROM Sailors S2
S2.name = 'Horatio')
```
Simulating INTERSECT

• Suppose we have tables R(a,b) and S(a,b)
• The following computes \( R \cap S \):

\[
\text{SELECT DISTINCT * FROM R WHERE (R.a, R.b) IN (SELECT * FROM S )};
\]

This can be expressed without nesting:

• Given \( R(a,b), S(a,b) \), what is \( R \bowtie S \) ?

Intersection!

\[
\text{SELECT DISTINCT R.a, R.b FROM R, S WHERE R.a = S.a AND R.b = S.b};
\]
Find the names of sailors who reserved a red and a green boat.

using INTERSECT

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

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

without INTERSECT

SELECT sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
   AND S.sid IN
   (SELECT S2.sid
    FROM Sailors S2, Reserves R2, Boats B2
    WHERE S2.sid = R2.sid AND R2.bid = B2.bid AND B2.color = 'green')

“Find all sailors who have reserved a red boat and, further, have sids that are included in the set of sids of sailors who have reserved a green boat.”
Simulating EXCEPT (set difference)

• What does this query compute?

```
SELECT DISTINCT *
FROM R
WHERE (R.a, R.b) NOT IN (SELECT * 
                            FROM S);
```

Can this be expressed without a nested query? No.

(But this fact is not obvious)
Find boats **not** reserved by sailor with sid = 100.

- R: all boats
- S: boats reserved by sailor with sid=100
- R – S is what we want.

Please write SQL query

```sql
SELECT B.bid
FROM Boats B
WHERE B.bid NOT IN (SELECT R.bid
                      FROM Reserves R
                      WHERE R.sid = 100 );
```
SQL Overview

• Query capabilities
  – SELECT-FROM-WHERE blocks,
  – Basic features, ordering, duplicates
  – Set operations (union, intersect, except)
  – Aggregation & Grouping
  – Nested queries (correlation)
  – Null values
NULLS in SQL

• Whenever we don’t have a value, we can put a NULL
• Can mean many things:
  – Value does not exists
  – Value exists but is unknown
  – Value not applicable
  – Etc.
• The schema specifies for each attribute whether it can be null (nullable attribute)
• How does SQL cope with tables that have NULLs?
Null Values

• If x = NULL then $4*(3-x)/7$ is still NULL

• If x = NULL then x = “Joe” is UNKNOWN

• In SQL there are three boolean values:
  FALSE  =  0
  UNKNOWN =  0.5
  TRUE    =  1
Null Values

- \( C_1 \text{ AND } C_2 = \min(C_1, C_2) \)
- \( C_1 \text{ OR } C_2 = \max(C_1, C_2) \)
- \( \text{NOT } C_1 = 1 - C_1 \)

Example SQL query:

```
SELECT *
FROM Person
WHERE (age < 25) AND (height > 6 OR weight > 190)
```

E.g.
- age = 20
- height = NULL
- weight = 200

Rule in SQL: include only tuples that yield TRUE.
Null Values

Unexpected behavior:

```
SELECT *
FROM Person
WHERE age < 25 OR age >= 25
```

Some Persons are not included!
Null Values

Can test for NULL explicitly:
- x IS NULL
- x IS NOT NULL

```
SELECT *
FROM Person
WHERE age < 25 OR age >= 25 OR age IS NULL
```

Now it includes all Persons
SQL Overview

- SQL Preliminaries
- Integrity constraints
- Query capabilities
  - SELECT-FROM-WHERE blocks,
  - Basic features, ordering, duplicates
  - Set ops (union, intersect, except)
  - Aggregation & Grouping
- Nested queries (correlation)
- Null values
- Modifying the database
- Views

Review in the textbook, Ch 5
Modifying the Database

Three kinds of modifications
• Insertion - creates new tuple(s)
• Deletion - remove existing tuple(s)
• Updates - modify existing tuple(s)

Sometimes they are all called “updates”
Insertions

General form:

```
INSERT INTO R(A1, ..., An) VALUES (v1, ..., vn)
```

Example: Insert a new sailor to the database:

```
INSERT INTO Sailor(sid, sname, rating, age)
VALUES (3212, 'Fred', 9, 44)
```

Missing attribute $\rightarrow$ NULL.
May drop attribute names if give them in order.
Insertions

```sql
INSERT INTO Sailor(sname)
SELECT DISTINCT B.name
FROM Boaters B
WHERE Boaters.rank = "captain"
```

The query replaces the VALUES keyword. Here we insert many tuples into PRODUCT.
Deletions

Example:

```sql
DELETE
FROM   Sailor
WHERE  S.sname = 'Horatio'
```

Factoid about SQL: there is no way to delete only a single occurrence of a tuple that appears twice in a relation.
Updates

Example:

```
UPDATE Sailor S
SET rating = rating + 1
WHERE Sailor.sid IN
    (SELECT sid
     FROM Reserves R
     WHERE R.date = 'Oct, 25');
```