SQL II
More SQL

Today:

- Nested Queries
- More SQL
Nested Queries in SQL

- Queries containing other queries
  - “outer query”
  - “inner query”

- Inner query:
  - Can appear in FROM or WHERE clause

Example:
```
SELECT   cname
FROM       borrower
WHERE    cname  IN  (SELECT cname
FROM    depositor)
```

<table>
<thead>
<tr>
<th>cname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson</td>
</tr>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Jones</td>
</tr>
<tr>
<td>Smith</td>
</tr>
</tbody>
</table>

think this as a function that returns the result of the inner query
Another example:

```
SELECT DISTINCT cname
FROM   borrower as b, loan as l
WHERE b.lno = l.lno
       AND bname = "Central"
       AND (bname, cname) IN
           ( SELECT bname, cname
             FROM   depositor as d, account as a
             WHERE d.acct_no = a.acct_no)
```

Q: Describe what this query returns

Ans: Names of borrowers at the Central branch who also have savings account at the same branch
Nested Queries in SQL

Another example:

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

Q: Describe what this query returns!

Q: What happens if I replace IN with NOT IN?
Nested Queries in SQL

- What can we do with queries in the WHERE clause?

- Let A = SELECT ... FROM ... WHERE ....

1. WHERE <expr> IN A
   WHERE <expr> NOT IN A

2. WHERE <expr> < ANY (A) (∃)
   WHERE <expr> < ALL (A) (∀)

3. WHERE EXISTS(A) (∃)
   WHERE UNIQUE(A)
Nested Queries in SQL

ANY ( \( \exists \) ): (or SOME)

\[ e < \text{ANY}(A) \equiv \exists t \in A (e < t) \]

\[
\begin{array}{c}
0 \\
5 \\
6
\end{array}
\]

(5 < \text{any } 5 ) = true \quad \text{(read: 5 < some tuple in the relation)}

\[
\begin{array}{c}
0 \\
5
\end{array}
\]

(5 < \text{any } 5 ) = false

\[
\begin{array}{c}
0 \\
5
\end{array}
\]

(5 = \text{any } 5 ) = true

\[
\begin{array}{c}
0 \\
5
\end{array}
\]

(5 \neq \text{any } 5 ) = true \quad \text{(since 0 \neq 5)}

(\text{= any}) \equiv \text{in}

However, (\neq \text{any}) \neq \text{not in} \quad \text{example:} \quad (3 \neq \text{any } 3 )
Nested Queries in SQL

Example query with ANY:

```sql
SELECT bname
FROM branch
WHERE assets > ANY
  ( SELECT assets
    FROM branch
    WHERE bcity= ‘BkLn’ )
```

Q: Describe what this query returns

Ans: Find all branches that have greater assets than some branch in Brooklyn.
Nested Queries in SQL

- Optimization of queries!

```sql
SELECT bname
FROM branch
WHERE assets > ANY
  ( SELECT assets
      FROM branch
      WHERE bcity = 'Bkln'
  )

Q: Can you rewrite this query to something that is equivalent, but more efficient to execute?

1) SELECT assets
    INTO Temp
    FROM branch
    WHERE bname = 'Bkln'

2) SELECT bname
    FROM branch
    WHERE assets > ANY (Temp)

Q: Why this is better?
Nested Queries in SQL

**ALL**  (\(\forall\)):

\[ e < \text{ALL} (A) \equiv \forall t \in A (e < t) \]

\[
\begin{array}{c|c|c|c}
\hline
& 0 & 5 & 6 \\
\hline
5 < \text{all} & \text{false} \\
\hline
6 & \text{false} \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\hline
& 6 & 10 \\
\hline
5 < \text{all} & \text{true} \\
\hline
\hline
4 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\hline
& 4 & 5 \\
\hline
5 = \text{all} & \text{false} \\
\hline
\hline
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\hline
& 4 & 6 \\
\hline
5 \neq \text{all} & \text{true} \quad \text{(since } 5 \neq 4 \text{ and } 5 \neq 6) \\
\hline
\end{array}
\]

\(\neq \text{all}) \equiv \text{not in}\]

However, \(= \text{all}) \neq \text{in}
Example query with ALL:

```sql
SELECT  bname
FROM     branch
WHERE    assets > ALL
    ( SELECT  assets
        FROM     branch
        WHERE  bcity= 'Bkln'
    )
```

Returns: branches that have greater assets than all branches in Brooklyn
Nested Queries in SQL

Correlated Queries: when the inner query contains tuple variables of the outer query

Example: Find all accounts located at branches that also hold a loan for a smaller amount

```
SELECT DISTINCT A.cct_no
FROM account as A
WHERE EXISTS (  SELECT *
FROM loan as L
WHERE A.bname = L.bname
AND A.balance > L.amt)
```

“correlation”: inner query contains reference to table in outer query
Names of sailors who have reserved boat # 103:

```sql
SELECT S.sname
FROM Sailors S
WHERE EXISTS
  (SELECT *
   FROM Reserves R
   WHERE R.bid = 103 AND S.sid = R.sid)
```
Nested Queries in SQL

EXISTS:

$$\text{EXISTS}(A) \equiv \exists t \in A$$  (i.e., true if A is not empty)

Example: Find all depositors who have accounts at all branches in Brooklyn

```
SELECT  DISTINCT cname
FROM  depositor as S
WHERE  NOT EXISTS (  
    ( SELECT bname
      FROM  branch
      WHERE  bcity = 'Bkln' )
    EXCEPT
    ( SELECT bname
      FROM  depositor as T, account as R
      WHERE  T.acct_no = R.acct_no AND
             S.cname = T.cname))
```

Inner Query: (branches in Brooklyn) - (branches where S has an account)

Example: Find the sailors who have reserved all boats (p.150)
Find the sailors who have 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))
```

All boat bids – bids of boats reserved by Sailor s
Find names of sailors who’ve reserved all boats.
Another approach:

- Example in book, not using EXCEPT:

```
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 that doesn’t have ...
a Reserves tuple showing S reserved B
Sailors who have reserved all boats

SELECT  S.name
FROM  Sailors S, reserves R
WHERE  S.sid = R.sid
GROUP BY S.name, S.sid
HAVING COUNT(DISTINCT R.bid) =
( Select COUNT(*) FROM Boats)

<table>
<thead>
<tr>
<th>sname</th>
<th>sid</th>
<th>bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frodo</td>
<td>1</td>
<td>102</td>
</tr>
<tr>
<td>Bilbo</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>Bilbo</td>
<td>2</td>
<td>102</td>
</tr>
<tr>
<td>Frodo</td>
<td>1</td>
<td>102</td>
</tr>
<tr>
<td>Bilbo</td>
<td>2</td>
<td>103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sname</th>
<th>sid</th>
<th>bid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frodo</td>
<td>1</td>
<td>102, 102</td>
</tr>
<tr>
<td>Bilbo</td>
<td>2</td>
<td>101, 102, 103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frodo</td>
<td>7</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>Bilbo</td>
<td>2</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>Sam</td>
<td>8</td>
<td>27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Nina</td>
<td>red</td>
</tr>
<tr>
<td>102</td>
<td>Pinta</td>
<td>blue</td>
</tr>
<tr>
<td>103</td>
<td>Santa Maria</td>
<td>red</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>102</td>
<td>9/12</td>
</tr>
<tr>
<td>2</td>
<td>102</td>
<td>9/12</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>9/14</td>
</tr>
<tr>
<td>1</td>
<td>102</td>
<td>9/10</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
<td>9/13</td>
</tr>
</tbody>
</table>
Nested Queries in SQL

UNIQUE:

UNIQUE(A) = true, if A has no duplicates

Example: Find all customers who have no more than one account at “Kenmore”

SELECT T.cname
FROM    depositor as T
WHERE   UNIQUE( SELECT R.cname
                FROM    account as A, depositor as R
                WHERE   T.cname = R.cname
                        AND R.acct_no = A.acct_no
                        AND A.bname = “Kenmore”)

Inner Query: Returns T’s cname for every acct held by T at Kenmore
Nested Queries in SQL

Nesting also possible in FROM clause

Example: Another way to express HAVING

```
SELECT  bname, avg(balance) as avgbal
FROM    account
GROUP BY bname
HAVING  AVG(balance) > 1200
```

```
SELECT  *
FROM     (SELECT  bname, AVG(balance) as avgbal
          FROM    account
          GROUP BY bname) AS TempRes
WHERE    avgbal > 1200
```
Nested Queries in SQL

Nested query in FROM clause, another example:

Find the maximum total balance across all branches

```
SELECT  MAX(tot-balance)
FROM     (SELECT  bname, SUM(balance)
          FROM     account
          GROUP BY bname) AS branch-total(bname, tot-balance)
```
More SQL Examples (from the book)...

- Find the sailors with the highest rating (p. 152)
- Find the name and age of the oldest sailor(s) (p. 152)
- Find the rating for which the average age of sailors is the minimum over all ratings (p. 162)

Schema:
Boats(bid, bname, color)
Sailors(sid, sname, rating, age)
Reserves( sid, bid, day)
Examples:

Find names of sailors with the highest rating:

SELECT  S.sname  
FROM    Sailors S  
WHERE   S.rating =  (SELECT  MAX(S2.rating)  
                      FROM    Sailors S2)

Find the name and age of the oldest sailor(s)

SELECT  S.sname, S.age  
FROM    Sailors S  
WHERE   S.age =  (SELECT  MAX(S2.age)  
                     FROM    Sailors S2)
Find the rating for which the average age of sailors is the minimum over all ratings:

```
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 = (SELECT MAX(Temp.avgage)
                        FROM   Temp)
```