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)
```

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

<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>
Nested Queries in SQL

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:

```
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?

Schema:

Boats(bid, bname, color)
Sailors(sid, sname, rating, age)
Reserves( sid, bid, day)
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) (∀)
   Also ( ‘>’, ‘<=’, ‘>=’, ‘=’, ‘≠’, ... )

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

ANY ( ∃): (or SOME)

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

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

\((5 < \text{any} \begin{array}{c} 0 \\ 5 \\ 6 \end{array}) = \text{true}\) (read: 5 < some tuple in the relation)

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

\((5 < \text{any} \begin{array}{c} 0 \\ 5 \end{array}) = \text{false}\)

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

\((5 = \text{any} \begin{array}{c} 0 \\ 5 \end{array}) = \text{true}\)

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

\((5 \neq \text{any} \begin{array}{c} 0 \\ 5 \end{array}) = \text{true}\) (since 0 \neq 5)

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

However, \((\neq \text{any}) \not\equiv \text{not in}\) example: \((3 \neq \text{any} \begin{array}{c} 3 \\ 5 \end{array})\)
Example query with ANY:

```
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!

```
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 (\(\forall\))**:

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5 < all 5 ) = false

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5 < all 10 ) = true

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5 = all 5 ) = false

<table>
<thead>
<tr>
<th></th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

(5 \(\neq\) all 6 ) = true (since 5 \(\neq\) 4 and 5 \(\neq\) 6)

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

However, \((= \text{all}) \neq \text{in}\)
Nested Queries in SQL

Example query with ALL:

```
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
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
Nested Queries with Correlation

- 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:**

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

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

```sql
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)
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

What about query optimization?

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

SELECT DISTINCT A.acct_no
FROM account as A, loan as L
WHERE A.bname = L.bname AND A.bal > L.amt
```

Above an example of Query Unnesting.

advantageous because joins evaluate in more ways than nested queries
Nested Queries in SQL

Correlation with Aggregates:

```sql
SELECT bname
FROM account as A
WHERE bal > (SELECT AVG(amt)
            FROM loan as L
            WHERE A.bname = L.bname)
```

Returns the branch names where accounts are held whose balances are more than the average loan taken at the same branch.

Kim’s technique:

1. ```sql
   SELECT bname, AVG(amt) as avgloan
   INTO Temp
   FROM loan
   GROUP BY bname
   ```

2. ```sql
   SELECT A.bname
   FROM account as A, temp as L
   WHERE A.bname = L.bname AND
   A.bal > L.avgloan
   ```
Kim’s Unnesting Technique

- Why is the rewrite better than the original query?
  Ans: the rewrite computes the avg loans JUST ONCE per branch

Is the rewrite always better than the nested query?

Ans: NO: if loan has branch names not in account
Kim’s Unnesting Technique

```sql
SELECT  bname
FROM      branch as B
WHERE   B.numloans   = ( SELECT COUNT(*)
                          FROM    loan as L
                          WHERE  B.bname = L.bname)
```

returns branches whose loan count agrees with that specified

1. ```sql
   SELECT  bname, COUNT(*) as numloans
   INTO       Temp
   FROM      Loan
   GROUP   BY  bname
```

2. ```sql
   SELECT   B. bname
   FROM      branch as B, temp as L
   WHERE   B.bname = L.bname AND
           B.numloans = L.numloans
```

Q: What is the problem with this rewrite?
Kim’s Bug (the COUNT bug)

Ans: Suppose branch relation includes a branch (Kenmore) with numloans = 0 and the loan is the same.

Is Kenmore in the result of the nested query? Why???
Is kenmore in the result of the unnested query?

Nested query:
............ WHERE A.x = (SELECT COUNT(*) FROM B WHERE A.y = B.z) If A.x = 0 and ... no such B, then result empty
COUNT(empty) = 0

So, it is in the nested query.. but not in the unnested!!

What about SUM? No problem because SUM returns NULL and comparing something to NULL is never true
Nested Queries in SQL

Nesting also possible in FROM clause

Example: Another way to express HAVING

```sql
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
- Find the sailors who have reserved all boats
- Find the name and age of the oldest sailor
- Find the rating for which the average age of sailors is the minimum over all ratings

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