SQL Nested Queries
Loading our example tables

- Look at createdb.sql (converted to HTML by <pre>)
- Note portable datatypes
- Note added “not null” constraints, missing in text.
- Same createdb.sql and dropdb.sql work for both Oracle and mysql.
- Table loading is different in the two cases
  - We could use inserts portably, but would run slower.
  - So we are using “bulk loads”, with special commands: sqlldr for Oracle and the load data command in mysql.
Note that the tables form five groups, or “databases”:

- Student, faculty, class, enrolled
- Emp, dept, works
- Flights, aircraft, employees, certified
- Suppliers, parts, catalog
- Sailors, boats, reserves

In all 5 groups, the last-named table has two FKs to PKs of other tables: this is an indication of a relationship table:

- A student is enrolled in a class
- An emp works in a dept
- An employee is certified to work on an aircraft
- A part is cataloged by a supplier
- A sailor reserves a boat
Nested Queries (subqueries)

- An SQL query can be used to help the evaluation of another query
  - E.g., a condition may need to be evaluated on a computed relation, not one readily available
  - Multiple levels of nesting are possible
  - Semantics similar to those of nested loops

- Nested queries do not appear in relational algebra
  - But it is possible to write relational algebra expressions to obtain same result (unless count, sum, etc. are in use)
  - Using nested queries leads to more concise solutions
Connecting queries and subqueries

- **Subquery:** Select … from … inside another Select … from …

- A subquery can return:
  - A scalar value (1x1 table) – such a subquery can appear in a query in the same place where a constant appears, in SQL-99-compliant DBs.
  - A relation

- **Where can subqueries appear?**
  - Most often in `WHERE` clause of parent query
  - Also used in `FROM` clause followed by range variable
    - … FROM Sailors, (SELECT bid FROM Boats) Bids …
  - In `HAVING` clauses
    - Will discuss later on
  - Also, if scalar, in the select-list. Not covered in text, and not in Entry-level SQL-92.
## Example Schema

### Sailors

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

### Boats

<table>
<thead>
<tr>
<th>bid</th>
<th>name</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>interlake</td>
<td>red</td>
</tr>
<tr>
<td>103</td>
<td>clipper</td>
<td>green</td>
</tr>
</tbody>
</table>

### Reserves

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>
Subqueries that return a constant

- Also referred to as subqueries that return a scalar
- Most easy case to understand

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

- If subquery returns more than one value or zero values, a runtime error occurs! FRAGILE, AVOID! (OK if “select max(...)...”, etc.)

- Next, we focus on subqueries that return relations
Correlated subquery

- Test that a relation satisfies some condition
  ... WHERE EXISTS (SELECT ...) - TRUE if subquery result is not empty
  more common in use: WHERE NOT EXISTS (...)

SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
               FROM Reserves R
               WHERE R.bid=103 AND S.sid=R.sid)

- Find names of sailors who’ve reserved boat #103

- Subquery is CORRELATED with parent query
Conditions involving relations and tuples

- Typically have some sort of set operations semantics
  - ... WHERE field IN (SELECT … )
  - ... WHERE field op ANY (SELECT …) (alternative: SOME)
  - ... WHERE field op ALL (SELECT …)

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

- Find names of sailors who’ve reserved boat #103
# How the ALL keyword works

<table>
<thead>
<tr>
<th>Condition</th>
<th>Equivalent expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>x &gt; ALL (1, 2)</td>
<td>x &gt; 2</td>
</tr>
<tr>
<td>x &lt; ALL (1, 2)</td>
<td>x &lt; 1</td>
</tr>
<tr>
<td>x = ALL (1, 2)</td>
<td>(x = 1) AND (x = 2)</td>
</tr>
<tr>
<td>x &lt;&gt; ALL (1, 2)</td>
<td>(x &lt;&gt; 1) AND (x &lt;&gt; 2)</td>
</tr>
</tbody>
</table>
## How the ANY and SOME keywords work

<table>
<thead>
<tr>
<th>Condition</th>
<th>Equivalent expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x &gt; \text{ANY} \ (1, 2)$</td>
<td>$x &gt; 1$</td>
</tr>
<tr>
<td>$x &lt; \text{ANY} \ (1, 2)$</td>
<td>$x &lt; 2$</td>
</tr>
<tr>
<td>$x = \text{ANY} \ (1, 2)$</td>
<td>$(x = 1) \ OR \ (x = 2)$</td>
</tr>
<tr>
<td>$x \neq \text{ANY} \ (1, 2)$</td>
<td>$(x \neq 1) \ OR \ (x \neq 2)$</td>
</tr>
</tbody>
</table>
Conditions involving relations and tuples

- Typically have some sort of set operations semantics
  - ...WHERE field **IN** (SELECT … )
  - ... WHERE field op **ANY** (SELECT … )
  - ... WHERE field op **ALL** (SELECT … )

```sql
SELECT S.sname
FROM  Sailors S
WHERE S.rating > ANY (SELECT S1.rating
  FROM  Sailors S1, Reserves R1
  WHERE S1.sid=R1.sid AND
  R1.bid=103)
```

- Find names of sailors whose rating is higher than the minimum rating among sailors who reserved boat 103
Conditions involving relations and tuples

- Typically have some sort of set operations semantics
  
  ... WHERE field IN (SELECT … )
  
  ... WHERE field \( \text{op} \) ANY (SELECT … )
  
  ... WHERE field \( \text{op} \) ALL (SELECT … )

  SELECT S.sname
  FROM Sailors S
  WHERE S.age >= ALL (SELECT S1.age
                     FROM Sailors S1)

- Find names of sailors with maximum age
Subqueries in the FROM clause

SELECT SQ.sname, SQ.bname
FROM ( SELECT S.sname, B.name AS bname
      FROM Sailors S, Boats B, Reserves R
    ) SQ
WHERE SQ.bname='interlake';

- Find names of sailors who reserved ‘interlake’
Rewriting INTERSECT Queries Using IN

Find sid’s of sailors who’ve reserved both a red and a green boat:

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

- Similarly, EXCEPT queries re-written using NOT IN.
Nested Queries - Review

- Nested queries returning a constant
  - Typically constant is compared with other value in the WHERE clause
  - \( \ldots \text{WHERE field} = (\text{SELECT bid FROM} \ldots) \ldots \)

- Nested queries returning a relation
  - in WHERE clause
    - \( \ldots \text{WHERE EXISTS|UNIQUE} (\text{SELECT bid FROM} \ldots) \ldots \)
    - \( \ldots \text{WHERE field IN} (\text{SELECT bid FROM} \ldots) \ldots \)
    - \( \ldots \text{WHERE field op ANY|ALL} (\text{SELECT bid FROM} \ldots) \ldots \)
  - in FROM clause followed by range variable
    - \( \ldots \text{FROM Sailors, (SELECT bid FROM Boats) Bids} \ldots \)
Example Query 1

<table>
<thead>
<tr>
<th>Sailors</th>
<th>Boats</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>bid</td>
</tr>
<tr>
<td>sname</td>
<td>name</td>
</tr>
<tr>
<td>rating</td>
<td>color</td>
</tr>
<tr>
<td>age</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reserves</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
</tr>
<tr>
<td>bid</td>
</tr>
<tr>
<td>day</td>
</tr>
</tbody>
</table>

- Find sids of sailors who’ve reserved only red boats
Find sids of sailors who’ve reserved only red boats

SELECT R.sid
FROM Reserves R, Boats B
WHERE B.bid = R.bid AND B.color='red' AND R.sid NOT IN
  (SELECT R1.sid FROM Reserves R1, Boats B1
   WHERE B1.color<>‘red’ AND B1.bid = R1.bid)
Find sids of sailors who’ve reserved only red boats

SELECT R.sid FROM Reserves R
WHERE NOT EXISTS(
    SELECT * from Reserves R1, Boats B
    WHERE B.bid=R1.bid AND R1.sid=R.sid
    AND B.color <> 'red'
)

Find sids of sailors who’ve reserved only red boats

SELECT S.sid FROM Sailors S
WHERE NOT EXISTS(
    SELECT * from Reserves R1, Boats B
    WHERE B.bid=R1.bid AND R1.sid=S.sid
    AND B.color <> 'red'
)

This returns sailors with no reservations at all! (as well as sailors reserving only red boats)
Example Query 2

<table>
<thead>
<tr>
<th>Sailors</th>
<th>Boats</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>bid</td>
</tr>
<tr>
<td>sname</td>
<td>name</td>
</tr>
<tr>
<td>rating</td>
<td>color</td>
</tr>
<tr>
<td>age</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reserves</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
</tr>
<tr>
<td>bid</td>
</tr>
<tr>
<td>day</td>
</tr>
</tbody>
</table>

- Find the name(s) of sailor(s) who have the highest rating (without using MAX because we haven’t covered it yet)
- Recall trick from RA: self-join (cross product) with age1 < age2 to find age1 values under the max
Find the name(s) of sailor(s) who have the highest rating

\[
\text{SELECT S1.sname FROM Sailors S1}
\text{WHERE S1.sid NOT IN (}
\text{SELECT S2.sid FROM Sailors S2, Sailors S3}
\text{WHERE S2.rating < S3.rating)}
\]
Find the name(s) of sailor(s) who have the highest rating

```
SELECT S.sname FROM Sailors S
WHERE S.rating >= ALL (SELECT rating FROM Sailors)
```

Using max, not yet officially covered:

```
SELECT S.sname FROM Sailors S
WHERE S.rating = (SELECT max(rating) FROM Sailors)
```
A query that uses ALL

```
SELECT vendor_name, invoice_number, invoice_total
FROM invoices i JOIN vendors v
    ON i.vendor_id = v.vendor_id
WHERE invoice_total > ALL
    (SELECT invoice_total
     FROM invoices
     WHERE vendor_id = 34)
ORDER BY vendor_name
```

The result of the subquery

<table>
<thead>
<tr>
<th>INVOICE_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

The result set

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>INVOICE_NUMBER</th>
<th>INVOICE_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bertelsmann Industry Svcs. Inc</td>
<td>509786</td>
<td>6940.25</td>
</tr>
<tr>
<td>Cahners Publishing Company</td>
<td>587056</td>
<td>2184.5</td>
</tr>
<tr>
<td>Computerworld</td>
<td>367447</td>
<td>2433</td>
</tr>
<tr>
<td>Data Reproductions Corp</td>
<td>40318</td>
<td>21842</td>
</tr>
</tbody>
</table>

(25 rows)
A query that uses ANY

```sql
SELECT vendor_name, invoice_number, invoice_total
FROM vendors JOIN invoices
    ON vendors.vendor_id = invoices.invoice_id
WHERE invoice_total < ANY
    (SELECT invoice_total FROM invoices
     WHERE vendor_id = 115)
```

The result of the subquery

<table>
<thead>
<tr>
<th>INVOICE_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>25.67</td>
</tr>
</tbody>
</table>

The result set

<table>
<thead>
<tr>
<th>VENDOR_NAME</th>
<th>INVOICE_NUMBER</th>
<th>INVOICE_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Office Products</td>
<td>263253257</td>
<td>22.57</td>
</tr>
<tr>
<td>Reiter's Scientific &amp; Pro Books</td>
<td>25022117</td>
<td>6</td>
</tr>
<tr>
<td>Boucher Communications Inc</td>
<td>24863706</td>
<td>6</td>
</tr>
<tr>
<td>Champion Printing Company</td>
<td>24780512</td>
<td>6</td>
</tr>
<tr>
<td>Dean Witter Reynolds</td>
<td>111-92R-10094</td>
<td>19.67</td>
</tr>
</tbody>
</table>