More SQL Select

Database Systems Lecture 8
Natasha Alechina
In This Lecture

• More SQL Select
  • Aliases
  • ‘Self-joins’
  • Subqueries
  • IN, EXISTS, ANY, ALL

• For more information
  • Connoly and Begg Chapter 5
  • Ullman and Widom Chapter 6.3.
SQL SELECT Overview

SELECT

[DISTINCT | ALL] <column-list>
FROM <table-names>
[WHERE <condition>]
[ORDER BY <column-list>]
[GROUP BY <column-list>]
[HAVING <condition>]

([] - optional, | - or)

More SQL SELECT
More SQL SELECT

This ‘AS’ is optional, but Oracle doesn’t accept it at all
Example

**Employee**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>John</td>
</tr>
<tr>
<td>124</td>
<td>Mary</td>
</tr>
</tbody>
</table>

**WorksIn**

<table>
<thead>
<tr>
<th>ID</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Marketing</td>
</tr>
<tr>
<td>124</td>
<td>Sales</td>
</tr>
<tr>
<td>124</td>
<td>Marketing</td>
</tr>
</tbody>
</table>

More SQL SELECT

```
SELECT
    E.ID AS empID,
    E.Name, W.Dept
FROM
    Employee E,
    WorksIn W
WHERE
    E.ID = W.ID
```
Example

```
SELECT
    E.ID AS empID,
    E.Name, W.Dept
FROM
    Employee E,
    WorksIn W
WHERE
    E.ID = W.ID
```

<table>
<thead>
<tr>
<th>empID</th>
<th>Name</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>124</td>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>124</td>
<td>Mary</td>
<td>Marketing</td>
</tr>
</tbody>
</table>
Aliases and ‘Self-Joins’

Aliases can be used to copy a table, so that it can be combined with itself:

```sql
SELECT A.Name FROM Employee A,
      Employee B
WHERE A.Dept=B.Dept
  AND B.Name='Andy'
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
</tr>
<tr>
<td>Andy</td>
<td>Marketing</td>
</tr>
<tr>
<td>Anne</td>
<td>Marketing</td>
</tr>
</tbody>
</table>
### Aliases and Self-Joins

#### Employee A

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
</tr>
<tr>
<td>Andy</td>
<td>Marketing</td>
</tr>
<tr>
<td>Anne</td>
<td>Marketing</td>
</tr>
</tbody>
</table>

#### Employee B

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
</tr>
<tr>
<td>Andy</td>
<td>Marketing</td>
</tr>
<tr>
<td>Anne</td>
<td>Marketing</td>
</tr>
</tbody>
</table>
Aliases and Self-Joins

SELECT ... FROM Employee A, Employee B ...
Aliases and Self-Joins

SELECT ... FROM Employee A, Employee B
WHERE A.Dept = B.Dept

<table>
<thead>
<tr>
<th>A.Name</th>
<th>A.Dept</th>
<th>B.Name</th>
<th>B.Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Marketing</td>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>Andy</td>
<td>Marketing</td>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>Anne</td>
<td>Marketing</td>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>Mary</td>
<td>Sales</td>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
<td>Peter</td>
<td>Sales</td>
</tr>
<tr>
<td>Mary</td>
<td>Sales</td>
<td>Peter</td>
<td>Sales</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
<td>Andy</td>
<td>Sales</td>
</tr>
<tr>
<td>John</td>
<td>Marketing</td>
<td>Andy</td>
<td>Marketing</td>
</tr>
<tr>
<td>Andy</td>
<td>Marketing</td>
<td>Andy</td>
<td>Marketing</td>
</tr>
<tr>
<td>Anne</td>
<td>Marketing</td>
<td>Andy</td>
<td>Marketing</td>
</tr>
</tbody>
</table>
Aliases and Self-Joins

```
SELECT ... FROM Employee A, Employee B
WHERE A.Dept = B.Dept AND B.Name = 'Andy'
```

<table>
<thead>
<tr>
<th>A.Name</th>
<th>A.Dept</th>
<th>B.Name</th>
<th>B.Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Marketing</td>
<td>Andy</td>
<td>Marketing</td>
</tr>
<tr>
<td>Andy</td>
<td>Marketing</td>
<td>Andy</td>
<td>Marketing</td>
</tr>
<tr>
<td>Anne</td>
<td>Marketing</td>
<td>Andy</td>
<td>Marketing</td>
</tr>
</tbody>
</table>
Aliases and Self-Joins

SELECT A.Name FROM Employee A, Employee B
WHERE A.Dept = B.Dept AND B.Name = 'Andy'

<table>
<thead>
<tr>
<th>A.Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
</tr>
<tr>
<td>Andy</td>
</tr>
<tr>
<td>Anne</td>
</tr>
</tbody>
</table>

The result is the names of all employees who work in the same department as Andy.
Subqueries

• A `SELECT` statement can be nested inside another query to form a subquery

• The results of the subquery are passed back to the containing query

• E.g. get the names of people who are in Andy’s department:

```sql
SELECT Name
FROM Employee
WHERE Dept =
  (SELECT Dept
   FROM Employee
   WHERE Name='Andy')
```
Subqueries

```
SELECT Name
FROM Employee
WHERE Dept =
(SELECT Dept
 FROM Employee
 WHERE Name='Andy')
```

- First the subquery is evaluated, returning the value ‘Marketing’
- This result is passed to the main query

```
SELECT Name
 FROM Employee
 WHERE Dept =
 'Marketing'
```
Subqueries

- Often a subquery will return a set of values rather than a single value
- You can’t directly compare a single value to a set

Options
- **IN** - checks to see if a value is in the set
- **EXISTS** - checks to see if the set is empty or not
- **ALL/ANY** - checks to see if a relationship holds for every/one member of the set
(NOT) IN

• Using IN we can see if a given value is in a set of values

• NOT IN checks to see if a given value is not in the set

• The set can be given explicitly or from a subquery

```
SELECT  <columns>
FROM    <tables>
WHERE   <value>
        IN <set>
```

```
SELECT  <columns>
FROM    <tables>
WHERE   <value>
        NOT IN <set>
```
(NOT) IN

SELECT *
FROM Employee
WHERE Department IN ('Marketing', 'Sales')
(NOT) IN

<table>
<thead>
<tr>
<th>Name</th>
<th>Department</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Marketing</td>
<td>Chris</td>
</tr>
<tr>
<td>Mary</td>
<td>Marketing</td>
<td>Chris</td>
</tr>
<tr>
<td>Chris</td>
<td>Marketing</td>
<td>Jane</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
<td>Jane</td>
</tr>
<tr>
<td>Jane</td>
<td>Management</td>
<td>Jane</td>
</tr>
</tbody>
</table>

SELECT * FROM Employee WHERE Name NOT IN (SELECT Manager FROM Employee)

More SQL SELECT
(NOT) IN

• First the subquery
  SELECT Manager
  FROM Employee
• is evaluated giving

<table>
<thead>
<tr>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chris</td>
</tr>
<tr>
<td>Chris</td>
</tr>
<tr>
<td>Jane</td>
</tr>
<tr>
<td>Jane</td>
</tr>
</tbody>
</table>

• This gives
  SELECT *
  FROM Employee
  WHERE Name NOT IN ('Chris', 'Jane')

<table>
<thead>
<tr>
<th>Name</th>
<th>Department</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Marketing</td>
<td>Chris</td>
</tr>
<tr>
<td>Mary</td>
<td>Marketing</td>
<td>Chris</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
<td>Jane</td>
</tr>
</tbody>
</table>

More SQL SELECT
(NOT) EXISTS

- Using EXISTS we see if there is at least one element in a set
- NOT EXISTS is true if the set is empty
- The set is always given by a subquery

\[
\text{SELECT } <\text{columns}> \\
\text{FROM } <\text{tables}> \\
\text{WHERE EXISTS } <\text{set}>
\]

\[
\text{SELECT } <\text{columns}> \\
\text{FROM } <\text{tables}> \\
\text{WHERE NOT EXISTS } <\text{set}>
\]
(NOT) EXISTS

SELECT *
FROM Employee E1
WHERE EXISTS (
    SELECT * FROM Employee E2
    WHERE E1.Name = E2.Manager
)

More SQL SELECT
ANY and ALL

- ANY and ALL compare a single value to a set of values.
- They are used with comparison operators like =, >, <, <=, >=.

- `val = ANY (set)` is true if there is at least one member of the set equal to the value.
- `val = ALL (set)` is true if all members of the set are equal to the value.
Find the names of the employee(s) who earn the highest salary

```
SELECT Name
FROM Employee
WHERE Salary >= ALL (SELECT Salary FROM Employee)
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>20,000</td>
</tr>
<tr>
<td>John</td>
<td>15,000</td>
</tr>
<tr>
<td>Jane</td>
<td>25,000</td>
</tr>
<tr>
<td>Paul</td>
<td>30,000</td>
</tr>
</tbody>
</table>
Find the names of employee(s) who earn more than someone else

```
SELECT Name
FROM Employee
WHERE Salary > ANY (SELECT Salary FROM Employee)
```
Word Searches

- Word Searches
  - Commonly used for searching product catalogues etc.
  - Want to be able to use word stemming for flexible searching
- For example: given a database of books,
  - Searching for "automata" should return everything with "automata" somewhere in the title
Word Searches

To search we can use queries like

```sql
SELECT * FROM Book
WHERE Title LIKE '%%Automata%%';
```

which returns all titles which have a substring `Automata`. `%` stands for `any other string`.

More SQL SELECT
Next Lecture 4 March!

• No lectures the week of the 22\textsuperscript{nd} Feb
• Yet more SQL
  • ORDER BY
  • Aggregate functions
  • GROUP BY and HAVING
  • UNION etc.
• For more information
  • Connoly and Begg Chapter 5
  • Ullman and Widom Chapter 6.4