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
Aliases

- Aliases rename columns or tables to:
  - Make names more meaningful
  - Make names shorter and easier to type
  - Resolve ambiguous names

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

Two forms:
- Column alias
  `SELECT column AS newName...`
- Table alias
  `SELECT ... FROM table AS newName`
Example

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

```sql
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>

More SQL SELECT
### Aliases and Self-Joins

Select ... FROM Employee A, Employee B ...

<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>Mary</td>
<td>Sales</td>
<td>John</td>
<td>Marketing</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</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>John</td>
<td>Marketing</td>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Mary</td>
<td>Sales</td>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Peter</td>
<td>Sales</td>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Andy</td>
<td>Marketing</td>
<td>Mary</td>
<td>Sales</td>
</tr>
<tr>
<td>Anne</td>
<td>Marketing</td>
<td>Mary</td>
<td>Sales</td>
</tr>
</tbody>
</table>

More...
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>
</tbody>
</table>
Aliases and Self-Joins

```sql
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></td>
</tr>
</tbody>
</table>
```

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

- First the subquery
  ```
  SELECT Manager
  FROM Employee
  ```
- is evaluated giving

- 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

SELECT <columns> FROM <tables> WHERE EXISTS <set>

SELECT <columns> FROM <tables> WHERE NOT EXISTS <set>
(NOT) EXISTS

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

Employee

<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>

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 =, >, <, <=, >=, <>
- \( \text{val} = \text{ANY} \ (\text{set}) \) is true if there is at least one member of the set equal to the value
- \( \text{val} = \text{ALL} \ (\text{set}) \) 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

```sql
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

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

• Word Searches
  • Commonly used for searching product catalogues etc.
  • Want to be able to search by keyword
  • Want to be able to use word stemming for flexible searching

• For example: given a database of books,
  • Searching for “crypt” would return
    • “Cryptonomicon” by Neil Stephenson
    • “Applied Cryptography” by Bruce Schneier
Word Searches

- To do a word search we can keep
  - A table of items to be searched
  - A table of keywords
  - A linking table saying which keywords belong to which items

<table>
<thead>
<tr>
<th>Items</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>itmID</td>
<td>itmTitle</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keywords</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>keyID</td>
<td>keyWord</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ItemKey</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>itmID</td>
<td>keyID</td>
<td></td>
</tr>
</tbody>
</table>
Word Searches

To search we can use queries like

```sql
SELECT * FROM Items
WHERE itmID IN (
    SELECT itmID FROM ItemKey
    WHERE keyID IN (
        SELECT keyID FROM Keywords
        WHERE keyWord LIKE 'crypt%')
)
```

More SQL SELECT
Word Searches

• Sometimes you need to search for a set of words
  • To find entries with all words you can link conditions with AND
  • To find entries with any of the words use OR

```sql
SELECT * FROM Items
WHERE itmID IN (  
  SELECT itmID FROM ItemKey  
  WHERE keyID IN (  
    SELECT keyID FROM Keywords  
    WHERE keyWord LIKE 'word1%')  
)  
AND  
itmID IN (  
  SELECT itmID FROM ItemKey  
  WHERE keyID IN (  
    SELECT keyID FROM Keywords  
    WHERE keyWord LIKE 'word2%')
)```

More SQL SELECT
Next Lecture

• 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