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.

But first...

<table>
<thead>
<tr>
<th>Track</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>cID</td>
<td>Title</td>
</tr>
<tr>
<td>11</td>
<td>Violent</td>
</tr>
<tr>
<td>12</td>
<td>Every Girl</td>
</tr>
<tr>
<td>13</td>
<td>Breath</td>
</tr>
<tr>
<td>14</td>
<td>Part of Me</td>
</tr>
<tr>
<td>12</td>
<td>Star</td>
</tr>
<tr>
<td>12</td>
<td>Teaboy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CD</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mix 9.99</td>
</tr>
<tr>
<td>2</td>
<td>Compilation 12.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Artist</th>
<th>aID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stellar</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cloudboy</td>
<td></td>
</tr>
</tbody>
</table>

Exercise
Find a list of the names of those artists who have a track on the CD with the title ‘Compilation’. (Several versions are on http://www.cs.nott.ac.uk/~nza/G51DBS07/cw3.sql).

SELECT Name
FROM Artist, Track, CD
WHERE (Artist.aID = Track.aID) AND
(Track.cID) = CD.cID) AND
(CD.Title = 'Compilation')

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

Aliases
- Aliases rename columns or tables to
  - Make names more meaningful
  - Make names shorter and easier to type
  - Resolve ambiguous names
- Two forms:
  - Column alias
    SELECT column AS newName...
  - Table alias
    SELECT ...
FROM table
AS newName
This ‘AS’ is optional, but Oracle doesn’t accept it at all
Example

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

Aliases and 'Self-Joins'

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

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

Aliases and Self-Joins

SELECT … FROM Employee A, Employee B …

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

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:

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.

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.

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.

(NOT) IN

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

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')

( 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

( NOT) EXISTS

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

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
**ALL**

Find the names of the employee(s) who earn the highest salary

<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>

**ANY**

Find the names of employee(s) who earn more than someone else

<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:**
  - 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>Keywords</th>
<th>ItemKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>itmID</td>
<td>keyID</td>
<td>itmID</td>
</tr>
<tr>
<td>itmTitle</td>
<td>keyWord</td>
<td>keyID</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%'))
```

---

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
    SELECT itmID FROM ItemKey
    WHERE keyID IN (
        SELECT keyID FROM Keywords
        WHERE keyWord LIKE 'word2%'))
```
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