Table of Contents

1. Associating tables on conditions other than equality 1
2. Self-Joins 2
3. Legacy comma style inner join 5

Many of our joins are joining tables by matching the fk and pk of two tables and doing an equality match. There are a few more join conditions that may be useful.

1. Associating tables on conditions other than equality

We have a table to supply a customer's credit rating based on their credit limit. This holds descriptive terms for various credit levels assigned to a customer. But there is no relationship defined between these two tables since the credit levels do not generally exactly match the values in the credit ratings table. This type of join is often used for a lookup up table that is based on a range of values.

Demo 01: Displaying oe_credit-ratings. These are integer values and the ranges do not overlap.

```
select *
from a_oe.credit_ratings;
+-----------+------------+-----------+
| low_limit | high_limit | rating    |
+-----------+------------+-----------+
|         0 |       1000 | Standard  |
|      1001 |       2000 | Good      |
|      2001 |       5000 | High      |
|      5001 |      10000 | Excellent |
|     10001 |      99999 | Superior  |
+-----------+------------+-----------+
```

Demo 02: Displaying the rating for customers. Notice that the joining clause uses a Between operator instead of equality. An equality join would not work well here.

```
select cust_id, credit_limit, rating
from a_oe.customers
join a_oe.credit_ratings on credit_limit between low_limit and high_limit
order by cust_id;
+---------+--------------+-----------+
| cust_id | credit_limit | rating    |
+---------+--------------+-----------+
|  400300 |         6000 | Excellent |
|  400801 |          750 | Standard  |
|  401250 |          750 | Standard  |
|  401890 |         1750 | Good      |
|  402100 |          750 | Standard  |
|  402110 |          750 | Standard  |
|  402120 |          750 | Standard  |
|  403000 |         6000 | Excellent |
|  403010 |         6000 | Excellent |
|  403050 |         6000 | Excellent |
|  403100 |         6000 | Excellent |
|  403500 |         6000 | Excellent |
|  403750 |         6000 | Excellent |
|  403760 |         6000 | Excellent |
|  404000 |         3500 | High      |
|  404100 |         3500 | High      |
|  404150 |         3500 | High      |
|  404180 |         3500 | High      |
. . .  rows omitted
```

Demo 03: We can also write this query using the following syntax which filters the Cartesian product.

```
select cust_id, credit_limit, rating
from a_oe.customers
   , a_oe.credit_ratings
where credit_limit between low_limit and high_limit
order by cust_id
;
```

Pay close attention to this query. This is using a Cartesian product- we have two tables in the From clause separated by a comma. The Where clause is supplying the test to associate these two tables and I get 32 rows returned in my data set. If I omit the Where clause then I am doing a Cartesian product with no filter for associating rows and I get back 170 rows in the result set (5 rows in the credit_ratings table times 34 rows in the customer tables). And 138 of these rows are meaningless.

```
select cust_id, credit_limit, rating
from a_oe.customers, a_oe.credit_ratings
order by cust_id;
```

Suppose I use following filter to find customers with an Excellent credit rating and I get a result set that shows all of my customers!

```
select cust_id, credit_limit, rating
from a_oe.customers, a_oe.credit_ratings
where rating in ('Excellent')
order by cust_id;
```

Cartesian products are not always a mistake but they need to be examined closely.

Demo 04: This uses a join that involves two attributes to check if any items were sold at more than their list price. This does not use a Where clause.

```
select a_prd.products.prod_id, quoted_price, prod_list_price, ord_id
from a_oe.order_details od
join a_prd.products
   on  od.prod_id = a_prd.products.prod_id
   and quoted_price > prod_list_price
;
+---------+--------------+-----------------+--------+
| prod_id | quoted_price | prod_list_price | ord_id |
+---------+--------------+-----------------+--------+
|    1010 |       175.00 |          150.00 |    390 |
|    1010 |       195.00 |          150.00 |    395 |
|    1010 |       175.00 |          150.00 |    550 |
|    1010 |       175.00 |          150.00 |    551 |
|    1010 |       175.00 |          150.00 |    609 |
|    1010 |       175.00 |          150.00 |   2120 |
|    1010 |       175.00 |          150.00 |   2121 |
|    1100 |       205.00 |           49.99 |    301 |
|    1150 |         7.25 |            4.99 |    223 |
+---------+--------------+-----------------+--------+
9 rows in set (0.05 sec)
```

2. Self-Joins

You can join a table to itself. You need to use a table alias to distinguish the two copies of the table involved in the join. The following is the traditional self-join of employees and their managers

Demo 05: Employees and managers . Note that the first row here has no Manager. Employee 100 is at the top of the chart . Remember if we want to sort by an alias, we use the back ticks on the order by keys

```
select concat(m.emp_id, ' ' , m.name_last) as "Manager"
, concat(e.emp_id, ' ' , e.name_last) as "Supervises"
from a_emp.employees e
left join a_emp.employees  m on  m.emp_id = e.emp_mng
order by  `Manager`, `Supervises`
;
+------------+------------+
| Manager    | Supervises |
+------------+------------+
| NULL       | 100 King   |
| 100 King   | 101 Koch   |
| 100 King   | 102 D'Haa  |
| 100 King   | 145 Russ   |
| 100 King   | 146 Partne |
| 100 King   | 201 Harts  |
| 101 Koch   | 108 Green  |
| 101 Koch   | 162 Holme  |
| 101 Koch   | 200 Whale  |
| 101 Koch   | 203 Mays   |
| 101 Koch   | 205 Higgs  |
| 102 D'Haa  | 103 Hunol  |
| 103 Hunol  | 104 Ernst  |
| 108 Green  | 109 Fiet   |
| 108 Green  | 110 Chen   |
| 145 Russ   | 150 Tuck   |
| 145 Russ   | 155 Hiller |
| 145 Russ   | 207 Russ   |
| 146 Partne | 160 Dorna  |
| 146 Partne | 161 Dewal  |
| 205 Higgs  | 204 King   |
| 205 Higgs  | 206 Geitz  |
+------------+------------+
22 rows in set (0.00 sec)
```

This is another self-join. The following query returns pairs of employees who have the same job id. We are joining on the job id and also on an inequality between the employees' ids. If we do not add that second joining condition, then each employee would be paired with themselves (since the job id values would match). The output shows one row if there are two employees with the same job id; and three rows if there are three employees with the same job id due to the pair matching .

Demo 06: Pairing Employees who have the same job id

```
select emp_1.job_id
, emp_1.emp_id as Emp1, emp_2.emp_id as Emp2
from a_emp.employees emp_1
join a_emp.employees emp_2
   on   emp_1.job_id = emp_2.job_id
   and  emp_1.emp_id < emp_2.emp_id
order by emp_1.job_id, emp_1.emp_id, emp_2.emp_id
;
+--------+------+------+
| job_id | Emp1 | Emp2 |
+--------+------+------+
|      8 |  150 |  155 |
|      8 |  150 |  207 |
|      8 |  155 |  207 |
|     16 |  101 |  108 |
|     16 |  101 |  161 |
|      16 |  101 |  162 |
|      16 |  101 |  200 |
|      16 |  101 |  203 |
|      16 |  101 |  205 |
|      16 |  108 |  161 |
|      16 |  108 |  162 |
|      16 |  108 |  200 |
|      16 |  108 |  203 |
|      16 |  108 |  205 |
|      16 |  161 |  162 |
|      16 |  161 |  200 |
|      16 |  161 |  203 |
|      16 |  161 |  205 |
|      16 |  162 |  200 |
|      16 |  162 |  203 |
|      16 |  162 |  205 |
|      16 |  200 |  203 |
|      16 |  200 |  205 |
|      16 |  203 |  205 |
|      32 |  104 |  109 |
|      32 |  104 |  110 |
|      32 |  104 |  160 |
|      32 |  104 |  204 |
|      32 |  104 |  206 |
|      32 |  109 |  110 |
|      32 |  109 |  160 |
|      32 |  109 |  204 |
|      32 |  109 |  206 |
|      32 |  110 |  160 |
|      32 |  110 |  204 |
|      32 |  110 |  206 |
|      32 |  160 |  204 |
|      32 |  160 |  206 |
|      32 |  204 |  206 |
|      64 |  102 |  103 |
|      64 |  102 |  146 |
|      64 |  103 |  146 |
+--------+------+------+
42 rows in set (0.00 sec)
```

Demo 07: Finding employees who earn more than other employees. This has a lot of rows of output

```
select
   e1.emp_id, e1.salary ,' earns more than '
,  e2.emp_id ,e2.salary
from a_emp.employees e1 ,
     a_emp.employees e2
where e1.salary > e2.salary
order by  e1.salary desc, e1.emp_id
;
```

The output starts with employee 161 who has the highest salary and is matched with all other employees. The next set of rows starts with employee 100 who has the next highest salary. The last set of rows starts with employee 150 who earns more than only the employee(s) with the lowest salary- in our data set that is employee 201 . Note there is no set of rows that start with this employee id.

```
+--------+-----------+------------------+--------+-----------+
| emp_id | salary    | earns more than  | emp_id | salary    |
+--------+-----------+------------------+--------+-----------+
|    161 | 120000.00 |  earns more than |    100 | 100000.00 |
|    161 | 120000.00 |  earns more than |    204 |  99090.00 |
|    161 | 120000.00 |  earns more than |    101 |  98005.00 |
|    161 | 120000.00 |  earns more than |    162 |  98000.00 |
|    161 | 120000.00 |  earns more than |    146 |  88954.00 |
|    161 | 120000.00 |  earns more than  |    206 |  88954.00 |
|    161 | 120000.00 |  earns more than  |    205 |  75000.00 |
|    161 | 120000.00 |  earns more than  |    103 |  69000.00 |
|    161 | 120000.00 |  earns more than  |    200 |  65000.00 |
|    161 | 120000.00 |  earns more than  |    104 |  65000.00 |
|    161 | 120000.00 |  earns more than  |    109 |  65000.00 |
|    161 | 120000.00 |  earns more than  |    160 |  65000.00 |
|    161 | 120000.00 |  earns more than  |    203 |  64450.00 |
|    161 | 120000.00 |  earns more than  |    108 |  62000.00 |
|    161 | 120000.00 |  earns more than  |    102 |  60300.00 |
|    161 | 120000.00 |  earns more than  |    110 |  60300.00 |
|    161 | 120000.00 |  earns more than  |    145 |  59000.00 |
|    161 | 120000.00 |  earns more than  |    207 |  30000.00 |
|    161 | 120000.00 |  earns more than  |    155 |  29000.00 |
|    161 | 120000.00 |  earns more than  |    150 |  20000.00 |
|    161 | 120000.00 |  earns more than  |    201 |  15000.00 |
|    100 | 100000.00 |  earns more than  |    204 |  99090.00 |
|    100 | 100000.00 |  earns more than  |    101 |  98005.00 |
|    100 | 100000.00 |  earns more than  |    162 |  98000.00 |
|    100 | 100000.00 |  earns more than  |    146 |  88954.00 |
|    100 | 100000.00 |  earns more than  |    206 |  88954.00 |
|    100 | 100000.00 |  earns more than  |    205 |  75000.00 |
. . .  rows omitted for emp 100
|    100 | 100000.00 |  earns more than  |    207 |  30000.00 |
|    100 | 100000.00 |  earns more than  |    155 |  29000.00 |
|    100 | 100000.00 |  earns more than  |    150 |  20000.00 |
|    100 | 100000.00 |  earns more than  |    201 |  15000.00 |
|    204 |  99090.00 |  earns more than  |    101 |  98005.00 |
|    204 |  99090.00 |  earns more than  |    162 |  98000.00 |
|    204 |  99090.00 |  earns more than  |    146 |  88954.00 |
|    204 |  99090.00 |  earns more than  |    206 |  88954.00 |
. . .  rows omitted for many employees
|    207 |  30000.00 |  earns more than  |    155 |  29000.00 |
|    207 |  30000.00 |  earns more than  |    150 |  20000.00 |
|    207 |  30000.00 |  earns more than  |    201 |  15000.00 |
|    155 |  29000.00 |  earns more than  |    150 |  20000.00 |
|    155 |  29000.00 |  earns more than  |    201 |  15000.00 |
|    150 |  20000.00 |  earns more than  |    201 |  15000.00 |
+--------+-----------+-------------------+--------+-----------+
223 rows in set (0.01 sec)
```

3. Legacy comma style inner join

There is a traditional, legacy join that does the attribute matching in the Where clause. You will see this join in a lot of older code (and a lot of code written now).

Logically this syntax does a Cartesian product and adds a filter for the records that match on the joining condition.

Demo 08: This is the join using the column name syntax

```
select cust_id
, oh.ord_id
, prod_id
, quantity_ordered * quoted_price as "extprice"
from a_oe.order_headers oh
join a_oe.order_details od on oh.ord_id = od.ord_id
order by cust_id, oh.ord_id
;
```

```
+---------+--------+---------+----------+
| cust_id | ord_id | prod_id | extprice |
+---------+--------+---------+----------+
|  400300 |    378 |    1120 |  2250.00 |
|  400300 |    378 |    1125 |  2250.00 |
|  401250 |    106 |    1060 |   255.95 |
|  401250 |    113 |    1080 |    22.50 |
|  401250 |    119 |    1070 |   225.00 |
|  401250 |    301 |    1100 |   205.00 |
. . .  rows omitted
```

Demo 09: Using the join of orders and order details in the Where clause

```
select oh.cust_id
, oh.ord_id
, od.prod_id
, od.quantity_ordered * od.quoted_price as "extprice"
from a_oe.order_headers oh
,    a_oe.order_details od
where oh.ord_id = od.ord_id
order by oh.cust_id, oh.ord_id
;
```

The advantage of doing the join in the From clause is that it isolates the join issues from the Where clause filters. If you do the join in the Where clause then you need to take more care with other filters in the Where clause especially if you have both And and Or operators in the Where clause.

This join syntax is not allowed in this class for assignments. I want you to get used to using the more uniform join syntax using the Condition join or Column Name join..