

Let's Take a Look at the Intricacies of the Outer Join SQL Operation

By Kalman Zilberman

Atlantic OTC, Dec. 9, 2005

The following database sample would be used in all following examples:

Suppliers-Parts Database

Suppliers

Supplier_ID	Name	Status	City
S1	Jones	10	Paris
S2	Spiritoso	12	London
S3	Spira	10	New York
S4	McNamara	17	(null)
S7	Kohl	10	Paris

Parts

Part#	Name	Color	Weight
P1	Bolt	Yellow	13
P2	Nut	Red	2
P3	Frame	Red	97
P4	Wheel	Blue	55
P5	Spring	Brown	7

Supplier Parts

Supplier ID	Part#	Qty
S1	P1	500
S1	P3	257
S1	P4	125
S2	P3	(null)
S3	P2	200
S3	P3	300
S7	P4	342
S8	P1	435

Relational Algebra operations

<u>Operation</u>	<u>Operator</u>
RESTRICTION <ul style="list-style-type: none">○ Sometimes called SELECTION<ul style="list-style-type: none">▪ Not to be confused with the SQL SELECT command	
PROJECTION	
UNION	UNION
INTERSECTION	INTERSECT
DIFFERENCE	MINUS
CARTESIAN PRODUCT	TIMES
JOIN	
DIVISION	DIVIDEBY

Relational Assignment operation

We are going to show its implementation through following examples

RESTRICTION operation

Syntax:

$R2 = R1 \text{ WHERE } \langle \text{restriction-condition} \rangle$

- $\langle \text{restriction-condition} \rangle$
 - is a Boolean expression

- *Example:* Provide a list of parts that are Red in color

Parts_Red = Parts
WHERE Color = 'Red'

Parts_Red *

Part#	Name	Color	Weight
P2	Nut	Red	2
P3	Frame	Red	97

PROJECTION operation

Syntax:

$R2 = R1$ (attribute-list)

- Comma-separated attribute-list: attr1, attr2, ..., attrN
- attribute-list (with surrounding parenthesis) may be specified explicitly or implicitly

Example 1: Identify a list of available part colors

$Part_Colors = Parts$ (Color)

Part Colors

Color
Yellow
Red
Brown
Blue

Example 2: Show all parts with corresponding properties

$Parts_New = Parts$

Parts New

Part#	Name	Color	Weight
P1	Bolt	Yellow	13
P2	Nut	Red	2
P3	Frame	Red	97
P4	Wheel	Blue	55
P5	Spring	Brown	7

- *Example 3:* Identify a list of Supplier Names with their corresponding Statuses

$Supplier_Name_Statuses = Suppliers$ (Name, Status)

Supplier Name Statuses

Name	Status
Jones	10
Spiritoso	12
Spira	10
McNamara	17
Kohl	10

Assumptions

We assume that the audience is familiar with use of

- Aliases
- Qualified names
- Relational algebra operations
 - UNION
 - INTERSECTION
 - DIFFERENCE
- *Example:* Provide a complete list of Suppliers from Suppliers and Supplier_Parts relations (show use of aliases just for demo purposes)

```
Suppliers_Complete_List =  
    Suppliers (Supplier_ID      S_ID)      S  
    UNION  
    Supplier_Parts (Supplier_ID  S_ID)      SP
```


is equivalent to

(Suppliers S1
TIMES
 Parts P1)
 (S1.Supplier_ID, S1.Name S_Name, P1.Part#,
 P1.Name P_Name, P1.Color))

Supplier Parts All

Supplier_ID	S_Name	Part#	P_Name	Color
S1	Jones	P1	Bolt	Yellow
S1	Jones	P2	Nut	Red
S1	Jones	P3	Frame	Red
S1	Jones	P4	Wheel	Blue
S1	Jones	P5	Spring	Brown
S2	Spiritoso	P1	Bolt	Yellow
S2	Spiritoso	P2	Nut	Red
S2	Spiritoso	P3	Frame	Red
S2	Spiritoso	P4	Wheel	Blue
S2	Spiritoso	P5	Spring	Brown
S3	Spira	P1	Bolt	Yellow
S3	Spira	P2	Nut	Red
S3	Spira	P3	Frame	Red
S3	Spira	P4	Wheel	Blue
S3	Spira	P5	Spring	Brown
S4	McNamara	P1	Bolt	Yellow
S4	McNamara	P2	Nut	Red
S4	McNamara	P3	Frame	Red
S4	McNamara	P4	Wheel	Blue
S4	McNamara	P5	Spring	Brown
S7	Kohl	P1	Bolt	Yellow
S7	Kohl	P2	Nut	Red
S7	Kohl	P3	Frame	Red
S7	Kohl	P4	Wheel	Blue
S7	Kohl	P5	Spring	Brown

JOIN operation

Syntax:

$R3 = (R1 \text{ TIMES } R2) \text{ WHERE } \langle \text{join-condition} \rangle$

- Let θ be any comparison operator (=, >, etc.)
- Let attribute r1 from relation R1 and attribute r2 from relation R2 be drawn from the same domain
- Let θ be applicable to the aforementioned domain
- θ -Join of relation R1 on attribute r1 with relation R2 on attribute r2 is defined as

$(R1 \text{ TIMES } R2) \text{ WHERE } R1.r1 \theta R2.r2$

- Informally - It is a
 - Cartesian Product + (Restriction to the result) + (Restriction condition *cross-compares* attributes in the joined relations)
- If θ is “EQUALS” – **EquiJoin**
- An **EquiJoin** with one of the two identical attributes removed is a **Natural Join**
- Informally, Join usually means natural join
- θ does not have to be “EQUALS”
- Joining attributes do not have to be keys (PK or FK)
- Joining attributes do not have to have the same name
- Any relation can be joined with any other relation – if it makes sense
- A relation can be joined with itself – if it makes sense
- Any number of relations can be joined (not just two) – because of closure property

- *Example I:* For every Supplier show a list of Parts stocked with the corresponding part's quantity. Also show the part's name and color
 - For simplicity reasons use suppliers only from the Supplier Parts relation

```
Supplier_Parts_List =
  (Supplier_Parts      SP
   TIMES
   Parts              P)
  WHERE SP.Part# = P.Part#  (SP.Supplier_ID,
  P.Part#, Qty, Name, Color)
```

Supplier Parts List

	Part#	Qty	Name	Color
S1	P1	500	Bolt	Yellow
S1	P3	257	Frame	Red
S1	P4	125	Wheel	Blue
S2	P3		Frame	Red
S3	P2	200	Nut	Red
S3	P3	300	Frame	Red
S7	P4	342	Wheel	Blue
S8	P1	435	Bolt	Yellow

- *Example II:* What relational algebra operations are performed in the following statement?

```
Supplier_Parts_List1 =
  (Supplier_Parts      SP
   TIMES
   Parts              P)
  WHERE Qty < 300    (Supplier_ID, P.Part#, Qty,
  Name, Color)
```

- How many rows would there be in the resulting relation?

OUTER JOIN operation

- What is an outer join
- Is Outer Join derivable through other operations?
- *Example I:* For every Supplier show a list of Parts stocked with the corresponding part's quantity. Show the Supplier_ID, Part Number, supplier's Name and Qty. Provide Parts information even if a particular part is not provided by any supplier

```
(
  Parts                P
    TIMES
  Supplier_Parts      SP
)
WHERE P.Part# = SP.Part#
(Supplier_ID, P.Part#, P.Name, Qty)
UNION
Parts
WHERE Part# in (Parts(Part#)
                MINUS
                Supplier_Parts (Part#)
                )
(null Supplier_ID, P.Part#, P.Name, null Qty)
```

- Another, simpler way of expressing the same solution:

```
(
  Parts (+)           P
    TIMES
  Supplier_Parts      SP
)
WHERE P.Part# = SP.Part#
(Supplier_ID, P.Part#, P.Name, Qty)
```

OUTER JOIN Syntax:

R3 = (R1 (attr-list1) (+) **TIMES** R2 (attr-list2) (+))
WHERE <join-condition>

- (+) - is optional, but at least one (+) must be present to have an outer join
- One-sided
- Two-sided
- Many-sided
- Left, Right, Full, Inner
- The (+), following the relation name, notation allows to specify a many-sided join
- *Example II:* For every Supplier show a list of Parts stocked with the corresponding part's quantity. Show the Supplier, its Name and Status for every Supplier. Provide Supplier information even if the Supplier provides no Parts

```
Supplier_Parts_List =
  (Supplier_Parts      SP
   TIMES
   Suppliers (+)      S)
  WHERE SP.Supplier_ID = S.Supplier_ID
  (SP.Supplier_ID Spl, Part#, Qty, S.Supplier_ID, Name,
  Status)
```

Supplier Parts List

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P1	500	S1	Jones	10
S1	P3	257	S1	Jones	10
S1	P4	125	S1	Jones	10
S2	P3		S2	Spiritoso	12
S3	P2	200	S3	Spira	10
S3	P3	300	S3	Spira	10
			S4	McNamara	10
S7	P4	342	S7	Kohl	10

- Same expressed in old Oracle's SQL notation

```

SELECT SP.Supplier_ID SP_Supplier, Part#, Qty, S.Supplier_ID
       S_Supplier, Name, Status
FROM   Supplier_Parts      SP
       , Suppliers         S
WHERE  SP.Supplier_ID (+) = S.Supplier_ID
;

```

- Does not allow for (+) notation on the **FROM** clause level
 - The WHERE clause (+) notation does not allow to specify a more than one-sided join
- *Example III:* For every Supplier show a list of Parts stocked with the corresponding part's quantity *if the quantity is less than 200*. Show the Supplier_ID, its Name and Status **for every Supplier**. Provide Supplier information **even if the Supplier provides no Parts**

```

Supplier_Parts_List
=
  (Supplier_Parts      SP
   TIMES
   Suppliers (+)      S
  )
  WHERE SP.Supplier_ID = S.Supplier_ID
        and Qty < 200
  (SP.Supplier_ID     Spl, Part#, Qty, S.Supplier_ID, Name,
   Status)

```

is equivalent to (we assume):

```

Supplier_Parts_List =
  ((Supplier_Parts
    WHERE Qty < 200
  )
    SP
    TIMES
  Suppliers (+)
    S
  )
  WHERE SP.Supplier_ID = S.Supplier_ID
  (SP.Supplier_ID Spl, Part#, Qty, S.Supplier_ID, Name,
  Status)

```

Supplier Parts List

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P4	125	S1	Jones	10
			S2	Spiritoso	12
			S3	Spira	10
			S4	McNamara	10
			S7	Kohl	10

but not equivalent to (we assume):

```

( (Supplier_Parts
  TIMES
  Suppliers (+)
    S)
  WHERE SP.Supplier_ID = S.Supplier_ID )
  WHERE Qty < 200
  (SP.Supplier_ID Spl, Part#, Qty, S.Supplier_ID, Name,
  Status)

```

Supplier Parts List

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P4	125	S1	Jones	10

- Both of the preceding assumptions could be valid assumptions
 - But they are producing different results

- Thus, the initial solution to the aforementioned problem

```

Supplier_Parts_List
=
    (Supplier_Parts      SP
     TIMES
     Suppliers (+)      S)
     WHERE SP.Supplier_ID = S.Supplier_ID
           and Qty < 200
(Supplier_Parts_List, SP.Supplier_ID, Part#, Qty, S.Supplier_ID, Name,
Status)

```

- is ambiguous
 - unless we properly define the order of operations within the outer join
- Same *example* expressed in old Oracle's SQL notation

Example III: For every Supplier show a list of Parts stocked with the corresponding part's quantity ***if the quantity is less than 200***. Show the Supplier_ID, its Name and Status **for every Supplier**. Provide Supplier information **even if the Supplier provides no Parts**

```

SELECT   SP.Supplier_ID      SP_Supplier
           , Part#
           , Qty
           , S.Supplier_ID      S_Supplier
           , Name
           , Status
FROM     Supplier_Parts      SP
           , Suppliers         S
           WHERE SP.Supplier_ID (+) = S.Supplier_ID
           and SP.Qty (+) < 200
;

```

Supplier Parts List

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P4	125	S1	Jones	10
			S2	Spiritoso	12
			S3	Spira	10
			S4	McNamara	17
			S7	Kohl	10

is equivalent to (Oracle assumes):

```
SELECT  SP.Supplier_ID SP_Supplier
          , Part#
          , Qty
          , S.Supplier_ID   S_Supplier
          , Name
          , Status
FROM (SELECT *
          FROM Supplier_Parts
          WHERE Qty < 200)          SP
          , Suppliers
WHERE SP.Supplier_ID (+) = S.Supplier_ID
;
```

but not equivalent to (Oracle assumes):

```

SELECT    SP_Supplier
          , Part#
          , Qty
          , S_Supplier
          , Name
          , Status
FROM (SELECT    SP.Supplier_ID SP_Supplier
          , Part#
          , Qty
          , S.Supplier_ID    S_Supplier
          , Name
          , Status
FROM Supplier_Parts        SP
  , Suppliers              S
WHERE SP.Supplier_ID (+) = S.Supplier_ID
)
WHERE Qty < 200
;

```

Supplier Parts List

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P4	125	S1	Jones	10

the preceding query, though, is equivalent to:

```

SELECT    SP.Supplier_ID SP_Supplier
          , Part#
          , Qty
          , S.Supplier_ID    S_Supplier
          , Name
          , Status
FROM Supplier_Parts        SP
  , Suppliers              S
WHERE SP.Supplier_ID (+) = S.Supplier_ID
and Qty < 200
;

```

- Order of execution is established in old Oracle's SQL syntax using (+) sign in the WHERE clause
- If the (+) sign is used for an additional restriction condition then
 - this additional condition is applied before the OUTER part of the JOIN is done
 - otherwise, it is applied after the OUTER part of the JOIN is done

Order of sub-operations when executing an OUTER JOIN – Summary Analysis

- **Possibility (I)**
 - JOIN using the cross-comparison of attributes without additional restriction conditions **(a)**
 - Perform the OUTER part of the JOIN – include tuples/rows that do not match the join criteria **(b)**
 - Apply additional restriction conditions **(c)**
- This possibility has the appeal of being most desirable
BUT
- Because (c) applies after the outer join (b) this possibility would give us, possibly, a not desirable result
 - The additional restriction condition (c) would eliminate, most of the time, the “outer joined” rows (the rows that were not a part of the base Join)
- It is possible that this possibility is the desirable result, though
- In general, there are six possibilities
 - **(a) (b) (c)**
 - **(a) (c) (b)**
 - ~~(b) (a) (c)~~
 - ~~(b) (c) (a)~~
 - **(c) (a) (b)**
 - ~~(c) (b) (a)~~

- Following possibilities do not make sense (~~(b)(a)(e)~~, ~~(b)(e)(a)~~, ~~(e)(b)(a)~~)
 - because a Join must be followed by an Outer part of the Join
 - all these remaining possibilities would require that the Outer part of the Join (b) is done before the Join (a)
 - this does not make sense
- The other possibility (II)
 - JOIN using the cross-comparison of attributes without additional restriction conditions (a)
 - Apply additional restriction conditions (c)
 - Perform the OUTER part of the JOIN – include tuples that do not match the join criteria (b)
- The third possibility (III)
 - Apply additional restriction conditions to corresponding underlying relation/table (c)
 - JOIN using the cross-comparison of attributes (a)
 - Perform the OUTER part of the JOIN – include tuples that do not match the join criteria (b)
- *We previously assumed that this possibility is, by definition, the default order of executing an outer join*
- Possibilities (II) and (III) are equivalent?
 - **Extra credit:** Formal proof of this hypothesis
- We assume that Possibility (III) introduces the “proper” implied order of execution (by definition)!!!
 - **Possibility (III) (or its equivalent – Possibility II) should be a part of the Outer Join definition**
 - This implies:
 - when we perform an Outer Join operation
 - the defined “proper” implied order of execution will be followed
 - Again, this does not imply that every problem requires Possibility II or III resolution - the solution to the problem may be Possibility I or even something different than that

- **Assumed proper order of execution** - rephrased
 - First
 - Join
 - Additional restriction conditions
 - or**
 - Additional restriction conditions
 - Join
 - Followed by
 - **Outer part of the operation -- is performed last**
-
- **Assumed proper order of execution** – rephrased in simpler terms
 - **Outer part of the operation -- is performed last, by definition**

One more *example*

- *Example IV:* For every Supplier show a list of Parts stocked with the corresponding part's quantity. Show the Supplier, its Name and Status for every Supplier. Provide Supplier information even if the Supplier does not supply any Parts. Show the Part# and Qty even if there is no corresponding Supplier information available

```

SELECT SP.Supplier_ID SP_Supplier, Part#, Qty, S.Supplier_ID
      S_Supplier, Name, Status
  FROM      Supplier_Parts      SP
           , Suppliers          S
  WHERE SP.Supplier_ID (+) = S.Supplier_ID
        UNION
SELECT SP.Supplier_ID SP_Supplier, Part#, Qty, S.Supplier_ID
      S_Supplier, Name, Status
  FROM      Supplier_Parts      SP
           , Suppliers          S
  WHERE SP.Supplier_ID = S.Supplier_ID(+)
;

```

Supplier Parts List

SP_Supplier	Part#	Qty	S_Supplier	Name	Status
S1	P1	500	S1	Jones	10
S1	P3	257	S1	Jones	10
S1	P4	125	S1	Jones	10
S2	P3		S2	Spiritoso	12
S3	P2	200	S3	Spira	10
S3	P3	300	S3	Spira	10
			S4	McNamara	17
S7	P4	342	S7	Kohl	10
S8	P1	435			

- The same expressed in relational algebra notation:

Supplier_Parts_List

=

(Supplier_Parts (+)SP

TIMES

Suppliers (+) S

)

WHERE SP.Supplier_ID = S.Supplier_ID

(SP.Supplier_ID Spl, Part#, Qty, S.Supplier_ID, Name, Status)

Notes:

- The preceding Example III question is ambiguously stated
 - “Show the Supplier_ID, its Name and Status for every Supplier” – does it include
 - every supplier, even if they do not provide Parts with an available quantity of less than 200?
 - every supplier, except if they do not provide Parts with an available quantity of less than 200?
 - every supplier who does not provide any parts at all?
 - every existing supplier?
 - This is what we assumed to be the desirable result
 - “Provide Supplier information even if the Supplier provides no Parts” – does this mean that we need to show suppliers S2, S3, S4, S7? Suppliers S2, S3,

S7 supply parts but S4 does not. Does this mean that we should show supplier information only for S4 but not S2, S3, S7? Like,

Supplier Parts List

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P4	125	S1	Jones	10
			S4	McNamara	10

- *Example III* question sounds, at the first review, deceptively clearly stated – but, apparently, needs further analysis
- Again, it is not immediately clear that the original *Example III* question is ambiguous – one got to be very careful with such questions, these
 - May be not easy to answer
 - Require additional clarification from the people who originally stated the question
- In real life applications we need to clarify this question and define an unambiguous question – this is a programmer or analyst responsibility – thus providing the “proper” result the user is expecting from us, despite the fact that the user was not able to state the original question correctly
- It may be not a simple task to state the question clearly and unambiguously
- This is a different issue than the outer join “proper” implied order of execution issue
- One of the possible ways to improve the *Example III* question:
 - For every Supplier show a list of Parts stocked with the corresponding part’s quantity if the quantity is less than 200. Show the Supplier_ID, its Name and Status for every Supplier even if the Supplier provides no Parts with an available quantity of less than 200 or the supplier does not provide any parts at all
 - In this case the result will be

Supplier Parts List

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P4	125	S1	Jones	10
			S2	Spiritoso	12
			S3	Spira	10
			S4	McNamara	10
			S7	Kohl	10

For time saving purposes - the highlighted text should not be presented

Oracle's SQL

- Similarities with Relational Algebra are obvious
- Old syntax
 - OUTER JOIN implemented using (+) in the WHERE clause instead of FROM clause
 - The plus sign is on the wrong side of the comparison
 - This creates certain problems
 - Inability to express many-side OUTER JOINS simply
 - The first implementation of OUTER JOIN, probably
- If more than two tables are involved in an outer join
 - The safest method is to resolve them in pairs of tables
 - If joining customer, order and item requiring an outer join on customer
 - The join between order and item should be resolved first
 - Then, customer should be outer joined to the resulting relation

Examples of new (starting with version 9i) Outer-Join syntax

```
SELECT *  
  FROM Supplier_Parts  SP          JOIN Suppliers      S  
  ON SP.Supplier_ID = S. Supplier_ID;
```

```
SELECT *  
  FROM Supplier_Parts  SP          INNER JOIN Suppliers      S  
  ON SP.Supplier_ID = S. Supplier_ID;
```

```
SELECT *  
  FROM Supplier_Parts  SP          LEFT OUTER JOIN Suppliers      S  
  ON SP.Supplier_ID = S. Supplier_ID;
```

```
SELECT *  
  FROM Supplier_Parts  SP          RIGHT OUTER JOIN Suppliers      S  
  ON SP.Supplier_ID = S. Supplier_ID;
```

```
SELECT *  
  FROM Supplier_Parts  SP          FULL OUTER JOIN Suppliers      S  
  ON SP.Supplier_ID = S. Supplier_ID;
```

```

SELECT *
FROM Supplier_Parts SP FULL OUTER JOIN Suppliers S
  USING ( Supplier_ID );

```

- The outer joined is identified here on the FROM clause level – like it was supposed to be, similarly to relational algebra
- Though, the WHERE clause will be performed after the outer join has completed
- The way to achieve the expected by us result, in this case, is to do the extra condition
 - within the ON clause
 - or in a nested subquery in the FROM clause

```

SELECT SP.Supplier_ID SP_Supplier
  , Part#
  , Qty
  , S.Supplier_ID S_Supplier
  , Name
  , Status
FROM Supplier_Parts SP RIGHT OUTER JOIN Suppliers S
  ON SP.Supplier_ID = S.Supplier_ID
WHERE SP.Qty < 200;

```

- Results in one row only

Spl	Part#	Qty	Supplier_ID	Name	Status
S1	P4	125	S1	Jones	10

- Restating the query

```

SELECT  SP.Supplier_ID  SP_Supplier
        , Part#
        , Qty
        , S.Supplier_ID S_Supplier
        , Name
        , Status
FROM Supplier_Parts  SP  RIGHT OUTER JOIN  Suppliers
S
        ON SP.Supplier_ID = S.Supplier_ID and Qty < 200
;

```

- Or restating the query

```

SELECT  SP.Supplier_ID  SP_Supplier
        , Part#
        , Qty
        , S.Supplier_ID S_Supplier
        , Name
        , Status
FROM (SELECT * FROM Supplier_Parts WHERE Qty < 200)  SP
        RIGHT OUTER JOIN
        Suppliers  S
        ON SP.Supplier_ID = S.Supplier_ID
;

```

- Gives us the expected result

Spl	Part#	Qty	Supplier ID	Name	Status
S1	P4	125	S1	Jones	10
			S2	Spiritoso	12
			S3	Spira	10
			S4	McNamara	17
			S7	Kohl	10

- Using the new syntax the many-side **OUTER JOIN** is impossible to achieve except through nested JOINS