IS 441 SQL Handout 2 – Joining Tables; ALTER/INSERT/UPDATE

Dr. Yue “Jeff” Zhang, Version 3.2, April 13, 2016

=========================

\*\*\*\*PLEASE study all four materials:\*\*\*\*

1. This handout; 2. The password-protected handout “SQL\_Handout\_pswd”; 3. PPT; 4. Textbook.

=========================

**I. JOINING TABLES**

1. Syntax of JOINs

SQL implements the joining of tables in the JOIN clause. A JOIN operation can be conducted in the following syntaxes:

1. SELECT *field-list* FROM *Table1*, *Table2*, *Table3*, …

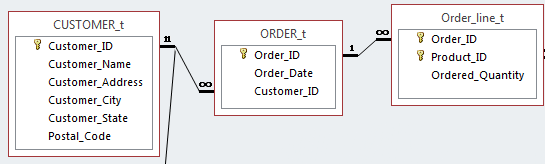
WHERE *Table1.PK=Table2.FK*

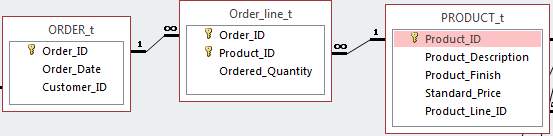
AND *Table2.PK=Table3.FK*

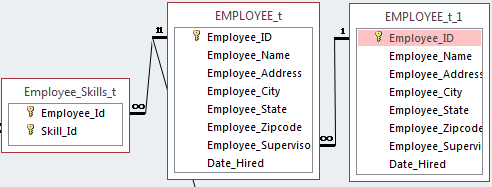
1. SELECT *field-list* FROM (*Table1* JOIN *Table2* ON *Table1.PK=Table2.FK*)

JOIN *Table3* ON *Table2.PK=Table3.FK*

Joining two tables are straightforward, BUT joining THREE tables needs more attention: one needs to clearly understand the three tables’ relationship – more so than joining two tables. Examples:







What can we say about the two tables on the right? – based on the fields and the referential integrity?

2. Outer Join Logic and Outcomes

[OUTER JOIN also has the same options in syntax as (INNER) JOIN: use WHERE or ON]

Assuming that there are two tables CUSTOMER and ORDER with entity instances and relationship instances as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cust\_ID | Cust\_City | … | (Lines here means the relationship) | Ord\_ID | Ord\_Date | … | Cust\_ID |
| C01 | … |  |  | O1011 | … |  | C01 |
| **C03** | … |  |  | O1012 | … |  | C08 |
| C08 |  |  |  | O1013 |  |  | C32 |
| … |  |  |  | … |  |  |  |
| C32 |  |  |  | O1234 |  |  | C01 |
| **C36** |  |  |  | O1235 |  |  | C78 |
| … |  |  |  | … |  |  |  |
| C78 |  |  |  | O9876 |  |  | C78 |
| … |  |  |  | … |  |  |  |

The SQL code -

SELECT CUSTOMER.Cust\_ID, Ord\_ID

FROM CUSTOMER LEFT OUTER JOIN [ORDER]

WHERE CUSTOMER.Cust\_ID = [ORDER].Cust\_ID;

Will generate the following results:

C01 O1011

Results:

All orders related to their customers,

AND

ALL customers, even those w/o order.

C01 O1234

**C03**

C08 O1012

C32 O1013

**C36**

C78 O1235

C78 O9876

3. Outer Join Syntax, with Variations

Syntax:

SELECT *field\_list*

FROM TABLE1 LEFT OUTER JOIN TABLE2

WHERE TABLE1.common\_column = TABLE2.common\_column

produces ALL records in TABLE1, plus those related records in TABLE2.

\*\*\*NOTE:

The above logic can be executed and produce exactly the **same outcome** with a variation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table on LEFT | |  | Table on RIGHT | |
| FROM | TABLE1 | **LEFT** OUTER JOIN | TABLE2 | “Whole table” **LEFT** join the other table |
| FROM | TABLE2 | **RIGHT** OUTER JOIN | TABLE1 | The other table **RIGHT** join “whole table” |

Please note: the effect of switching the table names AND (simultaneously) change “LEFT” to “RIGHT” will generate the same outcome.

The “Whole table” is the table –

* To the left of LEFT JOIN, or
* To the right of RIGHT JOIN.

The outcome of

|  |  |  |
| --- | --- | --- |
| A LEFT JOIN B | CUSTOMER LEFT JOIN ORDER | EMPLOYEE LLEFT JOIN MANAGER |
| \*\*\*\* ---- is the same as ---- \*\*\*\* | | |
| B RIGHT JOIN A | ORDER RIGHT JOIN CUSTOMER | MANAGER RIGHT JOIN EMPLOYEE |

4. “Bridge table”: Tables needed to be joined even no field from the table is to be listed

Scenario:

Table4

Table3

Table2

Table1

SELECT Table1.fied, Table2.field, Table4.field -- Note Table 3 has no field to be listed!!

FROM Table1, Table2, Table3, Table4 -- Note Table 3 still needs to be joined!!

WHERE Table1.common\_column\_w\_2 = Table2.common\_column\_w\_1

AND Table2.common\_column\_w\_3 = Table3.common\_column\_w\_2

AND Table3.common\_column\_w\_4 = Table4.common\_column\_w\_3;

Note: Even in the SELECT clause we only listed fields from tables 1, 2, and 4, we need to include the join conditions involving Table 3 in order to be able to connect Table 4.

5. CAUTION: While “nested” or multiple (INNER) JOINs are doable just as the JOIN between two tables, multi-table OUTER JOIN is more difficult to handle: unexpected outcomes would result if a multi-table outer join is attempted without very careful analysis. The advice is: do NOT attempt to perform a multi-table (more than two) outer join in one SELECT statement.

**II. ALTER, INSERT, and UPDATE**

Slide 30 of Chap 6 PPT:

* ALTER: changing the columns of the table
  + ALTER TABLE CUSTOMER\_T ADD *field*…
  + Note: About ALTER – a table contains two rows or 200 rows are the same in logic, since those 198 extra rows are just more (much more) logical repetitions (with some different values) of the original 2 rows. HOWEVER, a column having 10 columns and a table with 12 columns – 2 columns added to the original table – are very different: the STRUCTURE has changed. Hence the term for the operation of adding columns: ALTER.
* INSERT: adding records based on the existing table
  + For a complete-row insertion: INSERT INTO CUSTOME\_T VALUES (values of fields, in the order corresponding to the order of the fields in the table)
  + For a partial-row insertion: INSERT INTO CUSTOME\_T *field-list* VALUES (values of fields, in the order corresponding to the order of the fields in the *field-list*)
* UPDATE: changing the values of some fields in selected records
  + UPDATE CUSTOMER\_T SET *field* = *value* …WHERE…(the condition to have the update applied to rows)

**III. Re-Emphasizing on Normalization:**

Please study **Normalization example with step-by-step solution**