IS 441 SQL Handout 4 – Extra Review Session Notes

Dr. Yue “Jeff” Zhang, May 10, 2014

This handout is the notes from the “Extra review session” that I gave Friday morning. About 8 students attended and we discussed a lot of important issues with examples. I would like to share the notes with the whole class so we can all benefit from them.

The review session covers the following topics:

1. Correct implementation of the logic of the problem (w examples from HW6)
2. JOIN syntax
3. OUTER JOIN logic and syntax
4. ORDER BY vs GROUP BY
5. Self join
6. Subquery: relationship of inner and outer query
7. Given functional dependencies (FDs) in a relation, perform normalization to 2NF and 3NF
8. “Read” a given table to determine functional dependencies (FDs)

**I.** Correct implementation of the logic of the problem

Example 1:

Problem 8.

Wrong: … (in WHERE clause) Orderitem.OrderID NOT IN

(SELECT OrderID FROM [Order] WHERE Month(Orderdate) IN (12,11,10) );

PLEASE NOTE that the problem asked us to list the “items NOT being ordered in 4Q”, not “the orders not being placed in 4Q”. See the following created dataset to illustrate the mistake (I used simplified data format to save typing):

|  |  |  |  |
| --- | --- | --- | --- |
| OrderID | Order in 4Q | ItemID | Remarks |
| 1 | Not in 4Q | 01, 23, 45 | 1) According to the above WRONG logic, all the orders |
| 2 | No | 01, 34, 56 | made in 4Q are excluded. HOWEVER, |
| 3 | Yes (in 4Q) | 23, 45, 67 | 2) did we exclude all those ***items*** that have been |
| 4 | No | 45, 67, 89 | ordered in 4Q? – how about those in Order 1? |
| 5 | Yes | 89, 01, 23 | How about 01 in Order 2 and 45&67 in Order 4? |

Now you know that we cannot “translate” the logic as we wrongly interpreted it – we must “faithfully” implement the logic AS GIVEN/REQUIRED in the problem.

(BTW, a student emailed me and asked me his HW6 grade and I told him, and he emailed back for me to tell his mistakes; this is one of the mistakes.)

Example 2:

Problem 11 (the last problem).

SELECT (…) WHERE (join condition) AND Orderitem.ItemID NOT IN (SELECT … [category = “gar” and order date in June & July]...)

My remark: Should be EID NOT IN, not ItemID NOT IN – “mis-translated” the logic.

II. JOIN

1. Logic: Given tables A, B, C, D, will list fields from A, B, D. PLEASE remember to also join table C – w/o which D can’t be joined w B!
2. Syntax: Some students liked to use “… (A INNER JOIN B ON … ) INNER JOIN C ON… ) INNER …” which can easily miss certain parentheses and/or join wrong things or fail the joins. Recommendation: Just used the ESAIEST:

FROM A, B, C, D

WHERE (condition to join A and B)

AND (condition to join B and C)

AND (condition to join C and D)

-- You can’t go wrong with this syntax

III. OUTER JOIN

1. Logic: If you want all records in A to display, but only those Bs that are related to A to display, then

A LEFT OUTER JOIN B or

B RIGHT OUTER JOIN A

-- Hope my color coding, underline, and spacing help you to understand the logic.

1. Syntax: Please AVOID

“(A LEFT OUTER JOIN B…) INNER JOIN C…”

* - often have troubles

IV. ORDER BY vs GROUP BY

Say our College has 7,000 students and 10 majors.

SELECT …

|  |  |
| --- | --- |
| ORDER BY Major | GROUP BY Major |
| * Ends up 7,000 rows, w students of the same major listed one after another | * Ends up 10 rows, with majors ONLY but NO students |

Suggestion or hints: If the problems says “average/total/total number of … by XYZ or for each XYZ”, where the “XYZ” could be Major, Gender, Department, City, ProductID or CustID (when it can appear in many orders), …, then we need to think whether that calls for a GROUP BY this “XYZ”.

V. Self-Join (using HW6 as example)

Total sales under each manager = sales are achieved by each employee; now the individual employee’s sales need to be SUMmed within the same group – those employees of the same manager.

* Sum the employees’ sales GROUP BY their manager ID or manager name.
* Employee is one “physical table”; but now we need to view it as tweo tables: one contains the managers (“Mgr”), and the other contains employees who are NOT managers (“Emp”).
* To sum the employees’ sales for the group of employees who are under the saem manager (rather than summing randomly, or summing all employees in the company), we need to relate employees with their managers:
* “The employee’s manager ID = his/her manger’s employee ID”

Other possible questions of the type: “employees who are older than his/her manager”; “employees who were hired before his/her manager”; “employees whose salary is more than his/her manager”; “employees whose salary is half of his/her manager”; etc.

VI. Subquery

IF the subquery is in WHERE or HAVING clause, and if it’s not “correlate” (such as THERE EXIST), then

* The inner query is “in a different world” from the outer query, that is
* their tables used can be totally different;
* their aliases are totally different; or, the alias you named in the inner query will NOT be “seen” in the outer query, and vice versa;
* Please NOTE: IF the subqury is in FROM, then the outputs will be able to be SELECTed, and its column alias will be treated as a field, which can be displayed – that’s why we sometimes need to use subquery in FROM

VII. Given functional dependencies (FDs) in a relation, perform normalization to 2NF and 3NF

**G**

**E**

**F**

**D**

**C**

**B**

**A**

A-B-C: Full – the FD that originates from the key is FULL.

* Because every table MUST have a key (could be 1 field, or two or three fields – composite key, so every table MUST have one and only one full FD.

B-D: Partial FD, because B is part of the composite key; so an FD originated from part of the key is “partial” FD.

C-E-F: Transitive, because C is non-key, and it determines some other fields; so that is “transitive”.

F-G: what would that be? Should be clear and easy right?

2NF: remove the partial FD.

**G**

**E**

**F**

**C**

**B**

**A**

**D**

**B**

3NF: Based on 2NF, further remove the transitive FDs

**C**

**B**

**A**

**D**

**B**

**E**

**F**

**C**

**G**

**F**

VIII. “Read” a given table to determine functional dependencies (FDs)

(Using the “**Normalization**[**example with step-by-step solution**](http://www.csun.edu/~yz73352/441/441-ppt/Normalization%20Example%20Steps.docx)” that was posted right above “04/19” row in the class schedule)

1. Identify the composite key (if any). This is often hinted in the problem.
2. Identify different themes (such as EMPLOYEE and DEPARTMENT and PRODUCT, etc);
3. Quickly determine the easy ones, such as department ID determines department name and manager); -- now we have reduced the undecided FDs to a smaller number!
4. Examine those non-key fields that are likely to be related to one of possibly both of the fields in the composite key (say EmpID and ProductID):
   1. Given a specific EmpID (for example), will all the products sold by this employee the same?
      1. If not, then examine – given the same product, no matter who sod it, would it have the same price? If yes, then ProductID 🡺 ProductPrice;
   2. Do the same type of analysis for the other fields.
   3. One more example: Customer, Employee, Service, Price
      1. Given a customer, is the employee always the same? If yes, then Customer 🡺 Employee; if not, does the same SerivceDate always have the same Employee”? If yes, then ServiceDate 🡺 Employee; if not, then ServicceDate and Customer together determine Employee (a specific customer, comes in on a specific date, being served by a specific employee – but if the same cust comes in a different date, s/he would be served by a different employee. On the other hand, the same date, different customers could be served by different employees).