

# DB2 Advanced SQL – Working with Complex Queries

**Tony Andrews, Application Tuning Consultant**

**Themis, Inc.**

[tandrews@themisinc.com](mailto:tandrews@themisinc.com)

[www.themisinc.com](http://www.themisinc.com)



Themis and Themis, Inc. are trademarks of Themis, Inc.

DB2 is a trademark of the IBM Corporation.

Other products and company names mentioned herein may be trademarks of their respective companies. Mention of third party products or software is for reference only and constitutes neither a recommendation nor an endorsement.

© Copyright Themis, Inc. March 2012

***Tony Andrews is currently a trainer, consultant, and technical advisor at Themis, Inc. He teaches courses on SQL, application programming, database design, and performance tuning. He also has more than 23 years experience in the development of IBM DB2 relational database applications. Most of this time, he has provided development and consulting services to Fortune 500 companies and government agencies. For the last 10 years, Tony has been splitting his time between performance and tuning consulting engagements and DB2/SQL training. His main focus is to teach today's developers the ways of RDMS application design, development, and SQL programming -- always with a special emphasis on improving performance. He is a current IBM Champion, and regular speaker at many user groups, IDUG NA, and IDUG EMEA.***



# Table of Contents

|   |           |
|---|-----------|
| <b>DB2 ADVANCED SQL – WORKING WITH COMPLEX QUERIES.....</b> | <b>1</b>  |
| <b>Table Review .....</b>                                   | <b>6</b>  |
| <b>Scalar Fullselects .....</b>                             | <b>7</b>  |
| Option 1 .....  | 8         |
| Option 2 .....  | 9         |
| Option 3 .....  | 10        |
| Option 4 .....  | 11        |
| <b>Table Expressions.....</b>                               | <b>12</b> |
| Option 1 .....  | 13        |
| Correlated Subquery.....                                    | 14        |
| Adding to the SELECT List.....                              | 15        |
| Option 2 – Nested Table Expression .....                    | 17        |
| Option 3 – Common Table Expression .....                    | 18        |
| Table Expressions – Another Problem .....                   | 19        |
| <b>Ranking.....</b>   | <b>22</b> |
| Option #1 .....   | 23        |
| Option #2 .....   | 25        |
| Option #3 .....   | 29        |
| Dealing with Duplicates .....                               | 30        |
| Restricted Quotas.....                                      | 31        |
| <b>Relational Divide.....</b>                               | <b>35</b> |
| Option #1 .....   | 36        |
| Option #2 .....   | 37        |
| <b>Distinct Options.....</b>                                | <b>38</b> |
| Option #1 .....   | 39        |
| Option #2 .....   | 40        |
| Option #3 .....   | 41        |
| <b>Set Operations .....</b>                                 | <b>42</b> |
| Intersect .....   | 43        |
| Other Options .....   | 45        |
| Except.....   | 46        |
| Other Options .....   | 47        |
| <b>SELECT from MERGE .....</b>                              | <b>48</b> |
| <b>CASE Statement.....</b>                                  | <b>49</b> |
| Counting with CASE.....                                     | 50        |
| Table Pivoting.....   | 52        |
| <b>DB2 V8 Features .....</b>                                | <b>53</b> |
| <b>DB2 9 Features.....</b>                                  | <b>54</b> |
| <b>DB2 10 Features.....</b>                                 | <b>56</b> |

# Objectives

By the end of this presentation, developers will have a better understanding of:

- Nested and Common Table Expressions
- Complex Joins
- OLAP Ranking
- Correlated / Non Correlated Subqueries
- Quota Queries
- Relational Difference
- Set Operations
- Merge SQL
- Complex Case Statements

© 2012 Themis, Inc. All rights reserved.



## My Experience Shows

- Often times there are multiple ways of getting the same answer in programming. This pertains to SQL also.
- Everyone needs to be stronger in SQL.
- Strong SQL skills greatly enhance one's ability to do performance tuning of queries, programs, and applications.
- There are many new SQL functions available to developers in V8, V9, and V10.

# Table Review

## EMP

| EMPNO (PK) | LASTNAME | FIRSTNAME | DEPTNO | JOB      | EDLEVEL |
|------------|----------|-----------|--------|----------|---------|
| 000010     | HAAS     | CHRISTINE | A00    | PRES     | 18      |
| 000020     | THOMPSON | MICHAEL   | B01    | MANAGER  | 18      |
| 000030     | KWAN     | SALLY     | C01    | MANAGER  | 20      |
| 000050     | GEYER    | JOHN      | E01    | MANAGER  | 16      |
| 000060     | STERN    | IRVING    | D11    | MANAGER  | 16      |
| 000070     | PULASKI  | EVA       | D21    | MANAGER  | 16      |
| 000110     | VINCENZO | LUCCHESI  | A00    | SALESREP | 19      |
| 000120     | SEAN     | O'CONNELL | A00    | CLERK    | 14      |

## DEPT

| DEPTNO (PK) | DEPTNAME                     | MGRNO (FK) | ADMRDEPT | LOCATION |
|-------------|------------------------------|------------|----------|----------|
| A00         | SPIFFY COMPUTER SERVICE DIV. | 000010     | A00      |          |
| B01         | PLANNING                     | 000020     | A00      |          |
| C01         | INFORMATION CENTER           | 000030     | A00      |          |
| D01         | DEVELOPMENT CENTER           | <null>     | A00      |          |
| D11         | MANUFACTURING SYSTEMS        | 000060     | D01      |          |
| D21         | ADMINISTRATION SYSTEMS       | 000070     | D01      |          |

© 2012 Themis, Inc. All rights reserved.



# Table Review

# Scalar Fullselects

## Problem #1:

Provide a report of employees with employee detail information along with department aggregate information. Show:

| EMPNO  | LASTNAME | FIRSTNME  | SALARY   | DEPTNO | DEPT_AVG_SAL |
|--------|----------|-----------|----------|--------|--------------|
| 000010 | HAAS     | CHRISTINE | 52750.00 | A00    | 45312.50     |

---

## Scalar Fullselects

This example will require a correlated subquery since each individual piece of data (an employee's EDLEVEL) will need to be compared to an aggregate piece of data (the average EDLEVEL) where the aggregate is determined by information from the individual (the average for the department the employee works in).

## Option 1: Scalar Fullselect

```
SELECT E1.EMPNO,  
       E1.LASTNAME,  
       E1.SALARY  
       E1.DEPTNO,  
       (SELECT AVG(E2.SALARY)  
        FROM EMP E2  
        WHERE E2.DEPTNO = E1.DEPTNO)  
        AS DEPT_AVG_SAL  
FROM EMP E1  
ORDER BY E1.DEPTNO, E1.SALARY
```

Scalar Fullselect  
came in V8

© 2012 Themis, Inc. All rights reserved.



### Option 1

To add the average to the result, the subquery must be repeated as a scalar fullselect in the SELECT clause. Scalar fullselects were introduced in DB2 Version 8 and may be used as an expression anywhere in the statement provided they only return 1 column and 1 row.



## Option 2: Join

Join Provides  
different optimizer  
options

```
SELECT E1.EMPNO, E1.LASTNAME,  
       E1.DEPTNO, E1.SALARY,  
       AVG(E2.SALARY) AS DEPT_AVG_SAL  
FROM EMP E1 INNER JOIN  
       EMP E2 ON E1.DEPTNO = E2.DEPTNO  
GROUP BY E1.EMPNO, E1.LASTNAME,  
         E1.DEPTNO, E1.SALARY  
ORDER BY E1.DEPTNO, E1.SALARY
```

© 2012 Themis, Inc. All rights reserved.



## Option 2

With a join, the optimizer will have all the options of any join process (Nested Loop, Merge Scan, or Hybrid). There also may be a sort specific to the Group By and/or the Order By.

## Option 3: Nested Table Expression

```
SELECT E.EMPNO, E.LASTNAME, E.DEPTNO,  
       E.SALARY, DAS.DEPT_AVG_SAL  
FROM EMP E INNER JOIN  
  (SELECT DEPTNO,  
   AVG(SALARY) AS DEPT_AVG_SAL  
   FROM EMP  
   GROUP BY DEPTNO) AS DAS  
  
ON E.DEPTNO = DAS.DEPTNO  
ORDER BY E.DEPTNO, E.SALARY
```

Table Expression  
most likely gets  
materialized

© 2012 Themis, Inc. All rights reserved.



### Option 3

With a table expression containing aggregation and a Group By, most likely will see materialization.

## Option 4: Common Table Expression

```
WITH DEPT_AVG_SAL AS  
(SELECT DEPTNO,  
  AVG(SALARY) AS DEPT_AVG_SAL  
FROM EMP  
GROUP BY DEPTNO)  
  
SELECT E.EMPNO, E.LASTNAME, E.DEPTNO,  
       E.SALARY, DAS.DEPT_AVG_SAL  
FROM EMP E INNER JOIN  
       DEPT_AVG_SAL DAS ON E.DEPTNO = DAS.DEPTNO  
ORDER BY E.DEPTNO, E.SALARY
```

Table Expression  
most likely gets  
materialized

© 2012 Themis, Inc. All rights reserved.



### Option 4

With a table expression containing aggregation and a Group By, most likely will see materialization. With only 1 table expression, there will be no difference between the nested table and the common table.

# Table Expressions

## Problem #2:

Provide a report of employees whose education levels are higher than the average education level of their respective department.

Example: EMPNO '000010' works in department 'A00'. Is this employee's EDLEVEL > the average of all employees in 'A00'. If so, send their information to the result set.

© 2012 Themis, Inc. All rights reserved.



---

## Table Expressions

This example will require a correlated subquery since each individual piece of data (an employee's EDLEVEL) will need to be compared to an aggregate piece of data (the average EDLEVEL) where the aggregate is determined by information from the individual (the average for the department the employee works in).

## Option 1: Correlated Subquery

```
SELECT E1.EMPNO,  
       E1.LASTNAME,  
       E1.DEPTNO,  
       E1.EDLEVEL  
FROM EMP E1  
WHERE E1.EDLEVEL >  
      (SELECT AVG(E2.EDLEVEL)  
       FROM EMP E2  
       WHERE E2.DEPTNO = E1.DEPTNO)  
AND E1.DEPTNO < 'D01'
```

Join predicate in  
subquery!

© 2012 Themis, Inc. All rights reserved.



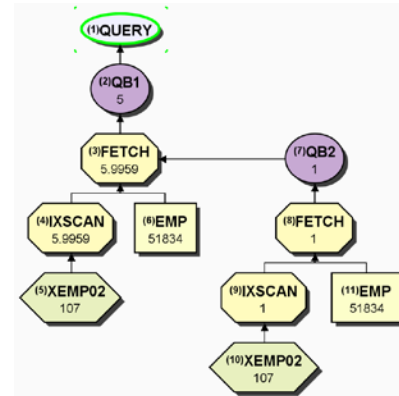
### Option 1

This example will require a correlated subquery since each individual piece of data (an employee's EDLEVEL) will need to be compared to an aggregate piece of data (the average EDLEVEL) where the aggregate is determined by information from the individual (the average for the department the employee works in).

## Solution Using Correlated Subquery

| EMPNO  | LASTNAME | DEPTNO | EDLEVEL |
|--------|----------|--------|---------|
| 000010 | HAAS     | A00    | 18      |
| 000011 | HAAS     | A00    | 18      |
| 000030 | KWAN     | C01    | 20      |
| 000110 | LUCCHESI | A00    | 19      |

Now try adding the average EDLEVEL into the result...



© 2012 Themis, Inc. All rights reserved.



## Correlated Subquery

Here is the solution and the access path graph from IBM Data Studio. Notice that the table is accessed twice. The access path for the correlated subquery will actually be run multiple times.

## Adding the Average to the Result

```
SELECT E1.EMPNO,  
       E1.LASTNAME,  
       E1.DEPTNO,  
       E1.EDLEVEL,  
       (SELECT AVG(E3.EDLEVEL)  
        FROM EMP E3  
        WHERE E3.DEPTNO = E1.DEPTNO) AS AVG  
FROM EMP E1  
WHERE E1.EDLEVEL >  
      (SELECT AVG(E2.EDLEVEL)  
       FROM EMP E2  
       WHERE E2.DEPTNO = E1.DEPTNO)  
AND E1.DEPTNO < 'D01'
```

Scalar Fullselect  
came in V8

© 2012 Themis, Inc. All rights reserved.

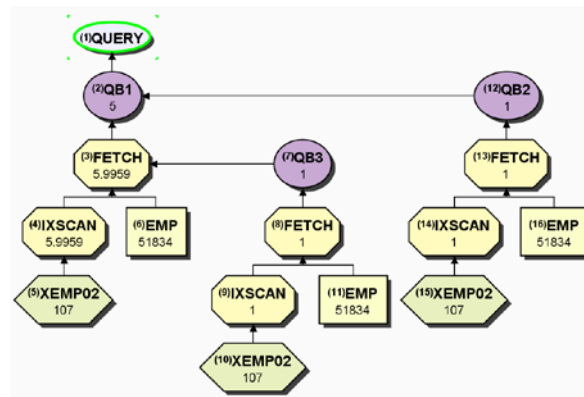


## Adding to the SELECT List

To add the average to the result, the subquery must be repeated as a scalar fullselect in the SELECT clause. Scalar fullselects were introduced in DB2 Version 8 and may be used as an expression anywhere in the statement provided they only return 1 column and 1 row.

# Adding the Average to the Result

| EMPNO  | LASTNAME | DEPTNO | EDLEVEL | AVG |
|--------|----------|--------|---------|-----|
| 000010 | HAAS     | A00    | 18      | 17  |
| 000011 | HAAS     | A00    | 18      | 17  |
| 000030 | KWAN     | C01    | 20      | 18  |
| 000110 | LUCCHESI | A00    | 19      | 17  |



© 2012 Themis, Inc. All rights reserved.





## Option 2: Nested Table Expression

```
SELECT E1.EMPNO,  
       E1.LASTNAME,  
       E1.DEPTNO,  
       E1.EDLEVEL,  
       DEPTAVG.AVG  
FROM EMP E1 JOIN  
      (SELECT DEPTNO, AVG(EDLEVEL) AS AVG  
       FROM EMP  
       WHERE DEPTNO < 'D01'  
       GROUP BY DEPTNO) AS DEPTAVG  
ON E1.DEPTNO = DEPTAVG.DEPTNO  
AND E1.EDLEVEL > DEPTAVG.AVG
```

© 2012 Themis, Inc. All rights reserved.



## Option 2 – Nested Table Expression

This query could be rewritten using a nested table expression to eliminate the redundancy. Think about the placement of the `DEPTNO < 'D01'` predicate. It is likely better to place it inside the nested table expression to limit the amount of data materialized (if materialization is required).

## Option 3: Common Table Expression

```
WITH DEPTAVG AS  
(SELECT DEPTNO, AVG(EDLEVEL) AS AVG  
FROM EMP  
WHERE DEPTNO < 'D01'  
GROUP BY DEPTNO) AS DEPTAVG
```

```
SELECT E1.EMPNO,  
       E1.LASTNAME,  
       E1.DEPTNO,  
       E1.EDLEVEL,  
       DEPTAVG.AVG  
FROM EMP E1 JOIN  
       DEPTAVG  
ON E1.DEPTNO = DEPTAVG.DEPTNO  
AND E1.EDLEVEL > DEPTAVG.AVG
```

© 2012 Themis, Inc. All rights reserved.

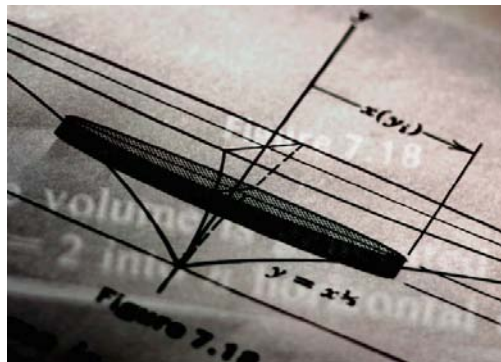


## Option 3 – Common Table Expression

# Table Expressions

## Problem #3:

**Return the department number and the total payroll for the department that has the highest payroll. Payroll will be defined as the sum of all salaries and bonuses for the department.**



© 2012 Themis, Inc. All rights reserved.



## Table Expressions – Another Problem

An additional layer of complexity arises when multiple levels of aggregation are needed. Here SUM will be needed to find the total payroll for each department and then a MAX will be needed to figure out which is the largest.

## Option1: Nested Table Expressions

```
SELECT DEPTNO, DEPT_TOT
FROM
  (SELECT DEPTNO,
    SUM(SALARY + BONUS) AS DEPT_TOT
    FROM EMP
    GROUP BY DEPTNO) DETPAY
WHERE DEPT_TOT =
  (SELECT MAX(TOT2)
    FROM (SELECT DEPTNO,
      SUM(SALARY + BONUS) TOT2
      FROM EMP
      GROUP BY DEPTNO) DETPAY2
  )
```

© 2012 Themis, Inc. All rights reserved.



Nested table expressions may be used to solve this problem, but because the MAX is needed in more than one query block, the nested table expression must be repeated.

## Option #2: Common Table Expression

```
WITH DEPTPAY AS  
  (SELECT DEPTNO,  
        SUM(SALARY + BONUS) AS DEPT_TOT  
   FROM EMP  
  GROUP BY DEPTNO)
```

Redundant  
Expression  
Eliminated!

```
SELECT DEPTNO, DEPT_TOT  
   FROM DEPTPAY  
  WHERE DEPT_TOT = (SELECT MAX(DEPT_TOT)  
                    FROM DEPTPAY)
```

© 2012 Themis, Inc. All rights reserved.



Common Table Expressions are defined outside of the query that uses the table expression and may therefore be used in any query block as shown here. Note that the SQL on this slide is all one query and will return the result from the bottom SELECT.

# Ranking

## Problem #4:

**Return the employees with the top 5 salaries.**  
**Could be 5 employees with different salaries.**  
**Could be many employees having the same salaries.**



© 2012 Themis, Inc. All rights reserved.



# Ranking

## Option #1: Rank() Function

```
SELECT    LASTNAME, SALARY,  
          RANK() OVER (ORDER BY SALARY DESC) AS R  
FROM EMP  
FETCH FIRST ?????? ROWS ONLY
```

How many do we  
fetch to get all of  
the top 5 ?

| LASTNAME     | SALARY   | R |
|--------------|----------|---|
| HAAS         | 52750.00 | 1 |
| HAAS         | 52750.00 | 1 |
| LUCCHESI     | 46500.00 | 3 |
| THOMPSON     | 41250.00 | 4 |
| GEYER        | 40175.00 | 5 |
| Others ..... |          |   |

© 2012 Themis, Inc. All rights reserved.



## Option #1

The RANK function orders and ranks a result. In this example, the result is being ranked by SALARY in descending sequence, so the highest salary has a rank of 1. When there is a tie, all rows with the value receive the same rank and an appropriate number of ranks are “skipped”. In this example, since there were 2 salary values in second place, the value of 3 is skipped.

The result set may be ordered using an ORDER BY for the entire SELECT, and this order need not be the same as the column being ranked. When this is done, 2 sorts may need to be performed to achieve the desired result.

## Option #1: Rank() Function

```
WITH RANK_TBL AS  
(SELECT LASTNAME, SALARY,  
    RANK() OVER (ORDER BY SALARY DESC) AS RANK_NUM  
 FROM THEMIS90.EMP)  
  
SELECT *  
FROM RANK_TBL  
WHERE RANK_NUM < 6  
;
```

| LASTNAME | SALARY   | R |
|----------|----------|---|
| HAAS     | 52750.00 | 1 |
| HAAS     | 52750.00 | 1 |
| LUCCHESI | 46500.00 | 3 |
| THOMPSON | 41250.00 | 4 |
| GEYER    | 40175.00 | 5 |

© 2012 Themis, Inc. All rights reserved.





## Option #2: Dense\_Rank() Function

```
SELECT LASTNAME, SALARY,
       DENSE_RANK() OVER (ORDER BY SALARY DESC) AS R
FROM EMP
```



| LASTNAME   | SALARY   | R |
|------------|----------|---|
| HAAS       | 52750.00 | 1 |
| HAAS       | 52750.00 | 1 |
| LUCCHESI   | 46500.00 | 2 |
| THOMPSON   | 41250.00 | 3 |
| GEYER      | 40175.00 | 4 |
| KWAN       | 38250.00 | 5 |
| Others ... |          |   |

© 2012 Themis, Inc. All rights reserved.



## Option #2

The DENSE\_RANK function orders and ranks a result. In this example, the result is being ranked by SALARY in descending sequence, so the highest salary has a rank of 1. When there is a tie, all rows with the value receive the same rank and no ranks are “skipped”. In this example, there were 2 salary values for first, then the next value becomes rank = 2.

The result set may be ordered using an ORDER BY for the entire SELECT, and this order need not be the same as the column being ranked. When this is done, 2 sorts may need to be performed to achieve the desired result.

## Option #2: Dense\_Rank() Function

```
WITH RANK_TBL AS  
(SELECT LASTNAME, SALARY,  
  DENSE_RANK() OVER (ORDER BY SALARY DESC) AS RANK_NUM  
  FROM THEMIS90.EMP)  
  
SELECT *  
FROM RANK_TBL  
WHERE RANK_NUM < 6  
;
```

| LASTNAME | SALARY   | R |
|----------|----------|---|
| HAAS     | 52750.00 | 1 |
| HAAS     | 52750.00 | 1 |
| LUCCHESI | 46500.00 | 2 |
| THOMPSON | 41250.00 | 3 |
| GEYER    | 40175.00 | 4 |
| KWAN     | 38250.00 | 5 |

Note: 6 rows make the top 5

*This page intentionally left blank*

## Option #3: Quota Queries

Find the 5 *highest* salaries:

```
SELECT LASTNAME, SALARY
FROM EMP E
WHERE 5 >
      (SELECT COUNT(*)
       FROM EMP E1
        WHERE E1.SALARY > E.SALARY)
ORDER BY SALARY DESC
```

| LASTNAME | SALARY   |
|----------|----------|
| HAAS     | 52750.00 |
| HAAS     | 52750.00 |
| LUCHESSI | 46500.00 |
| THOMPSON | 41250.00 |
| GEYER    | 40175.00 |

© 2012 Themis, Inc. All rights reserved.



## Option #3

Quota query is the name given to the general class of problems seeking to return a ranked list of the highest or lowest values of some column or columns. The top 10 customers; the 15 highest salaries in the company; the 25 employees with the shortest tenure; are representative quota queries.

SQL solutions involving many levels of nested subselect (or many joins) are also impractical for all but the simplest quota. The following example shows a subselect solution for finding the 2 highest salaries on the EMP table:

```
SELECT EMPNO, SALARY  
FROM EMP  
WHERE SALARY = (SELECT MAX(SALARY)  
                FROM EMP)  
    OR SALARY = (SELECT MAX(SALARY)  
                FROM EMP  
                WHERE SALARY <> (SELECT MAX(SALARY)  
                    FROM EMP));
```

If the 10 highest salaries were required, this template would take on a prohibitive level of complexity. The queries on the facing page solve the problem using a counting solution. The inner select counts the number of salaries greater than the salary being considered for the result. Since no one makes more than the highest salary, a count of 0 to 4 would indicate the employee in the outer query was among the top 5.

## Option #3: Quota Queries - Duplicates

```
SELECT    LASTNAME, SALARY
FROM      EMP E
WHERE     5 > (SELECT COUNT(DISTINCT SALARY)
              FROM EMP E1
              WHERE E1.SALARY > E.SALARY)
ORDER BY  SALARY DESC
```

| LASTNAME | SALARY   |
|----------|----------|
| HAAS     | 52750.00 |
| HAAS     | 52750.00 |
| LUCHESSI | 46500.00 |
| THOMPSON | 41250.00 |
| GEYER    | 40175.00 |
| KWAN     | 38250.00 |

© 2012 Themis, Inc. All rights reserved.



## Dealing with Duplicates

This query addresses duplicate salaries by elimination of duplicates during COUNT(DISTINCT) processing. Eliminating the duplicate salaries is only one way of interpreting the semantics of the top 5 salaries. If 5 people in the company made the same salary and it was coincidentally the maximum salary, should the result only contain these employees? By removing the duplicates we are implying the top 5 *different* salaries. Other questions regarding semantics, such as, *suppose there were only 4 employees; should the result be empty; should the query return only the 4 employees, should it be an error;* must be considered and built into the solution.

# Restricted Quotas

## PROBLEM #5:

**Find all employees who major in math (MAT) and computer science (CSI).**



EMPMAJOR TABLE

| EMPNO | MAJOR |
|-------|-------|
| E1    | MAT   |
| E1    | CSI   |
| E2    | MAT   |
| E3    | CSI   |
| E4    | ENG   |

© 2012 Themis, Inc. All rights reserved.



## Restricted Quotas

Restricted quotas filter rows based on a certain condition before performing the required ranking. Either by grouping or counting or both, a condition is associated with a count. Those groups having the expected count are qualified to the condition.

## Option #1: Restricted Quotas

**PROBLEM #5:** Find all employees who major in math (MAT) and computer science (CSI).

EMPMAJOR

| EMPNO | MAJOR |
|-------|-------|
| E1    | MAT   |
| E1    | CSI   |
| E2    | MAT   |
| E3    | CSI   |
| E4    | ENG   |

```
SELECT EMPNO
FROM EMPMAJOR
WHERE MAJOR IN ('MAT', 'CSI')
GROUP BY EMPNO
HAVING COUNT(*) = 2
```

Solution 1



© 2012 Themis, Inc. All rights reserved.

Option 1 retrieves employees who major in math or computer science. A given employee appearing in the result before the grouping has 1 row or 2 rows. Only groups with count = 2 result from employees who major in both.



## Option #2: Restricted Quotas

**PROBLEM #5:** Find all employees who major in math (MAT) and computer science (CSI).

EMPMAJOR

| EMPNO | MAJOR |
|-------|-------|
| E1    | MAT   |
| E1    | CSI   |
| E2    | MAT   |
| E3    | CSI   |
| E4    | ENG   |

```
SELECT DISTINCT EMPNO
FROM EMPMAJOR EM
WHERE 2 = (SELECT COUNT(*)
           FROM EMPMAJOR EM2
           WHERE EM.EMPNO =
                 EM2.EMPNO
           AND EM2.MAJOR IN
                 ('MAT', 'CSI'))
```

Solution 2

© 2012 Themis, Inc. All rights reserved.



Option 2 uses the template from the previous example as a model. For a given employee evaluated in the outer query, the inner query returns the rows for that employee restricted to math or computer science. As in solution 1, a count of 2 indicates that the employee majors in both.

## Option #3: Restricted Quotas

**PROBLEM #5:** Find all employees who major in math (MAT) and computer science (CSI).

EMPMAJOR

| EMPNO | MAJOR |
|-------|-------|
| E1    | MAT   |
| E1    | CSI   |
| E2    | MAT   |
| E3    | CSI   |
| E4    | ENG   |

```
SELECT EMPNO
FROM EMPMAJOR AS EMP1
JOIN EMPMAJOR AS EMP2
ON EMP1.EMPNO = EMP2.EMPNO
WHERE EMP1.MAJOR = 'MAT'
AND EMP2.MAJOR = 'CSI';
```

Solution 3

© 2012 Themis, Inc. All rights reserved.



This problem could also be solved with a self-join. Although not a quota query template, this solution is a straightforward approach to AND conditions posed on a single table.

# Relational Divide

## Problem #6:

**Find employees who work on *all* activities between 90 and 110**

**EMPPROJACT Table (partial data)**

**ACT Table (partial data)**

| ACTNO | ACTDESC          |
|-------|------------------|
| 90    | ADM QUERY SYSTEM |
| 100   | TEACH CLASSES    |
| 110   | DEVELOP COURSES  |

| EMPNO  | PROJNO | ACTNO |
|--------|--------|-------|
| 000130 | IF1000 | 90    |
| 000130 | IF1000 | 100   |
| 000140 | IF1000 | 90    |
| 000140 | IF2000 | 100   |
| 000140 | IF2000 | 100   |
| 000140 | IF2000 | 110   |
| 000140 | IF2000 | 110   |
| 000150 | MA2112 | 60    |
| 000150 | MA2112 | 180   |

© 2012 Themis, Inc. All rights reserved.



---

# Relational Divide

The problem on this page is part of a general set of problems known as set divide problems. Set divide problems represent the notion of FORALL quantification. *In this problem, an employee will be on the result table, if FORALL activities between 90 and 110, there exists a row on EMPPROJACT identifying that employee.* Employee 140 satisfies this property, but 130 and 150 do not.

**SQL doesn't support a divide operator, nor does it support direct FORALL quantification. Only EXISTS is supported in SQL.** Therefore we must use a double negative to form the equivalent complement using EXISTS.

## Option #1: Relational Divide

**PROBLEM #6: Find employees who work on all activities between 90 and 110.**

```
SELECT EPA.EMPNO
FROM EMPPROJECT EPA
WHERE EPA.ACTNO BETWEEN 90 AND 110
GROUP BY EPA.EMPNO
HAVING COUNT(DISTINCT ACTNO) =
    (SELECT COUNT(*)
     FROM ACT A
     WHERE A.ACTNO BETWEEN 90 AND 110)
```

The number of activities this person works

...is equal to the number of activities

| EMPNO  |
|--------|
| 000140 |



© 2012 Themis, Inc. All rights reserved.

## Option #1

This solution uses a quota approach similar to the problem involving employee majors presented earlier. Rows of EMPROJECT are grouped resulting in 1 row per employee. The count of the distinct ACTNOs that an employee works on is compared to the total count of the rows on the ACT table that meet the criteria to provide the answer.

## Option #2: Relational Divide

```

SELECT EMPNO
FROM EMP E
WHERE NOT EXISTS
  (SELECT ACTNO
   FROM ACT A
   WHERE ACTNO BETWEEN 90 AND 110
    AND NOT EXISTS
      (SELECT 1
       FROM EMPPROJECT EPA
       WHERE E.EMPNO = EPA.EMPNO
        AND A.ACTNO = EPA.ACTNO
      ) )
        
```

| EMPNO  |
|--------|
| 000140 |

There *isn't*  
an activity  
(between 90  
and 110)

...that this  
employee  
doesn't work

© 2012 Themis, Inc. All rights reserved.



## Option #2

This solution is the standard, most flexible template for the divide problem. The 2 NOT EXISTS correspond to the double negative derived earlier and give rise to the name *double double* for the general class of solutions using this approach. The search of all customers is the outermost level (E level). Since the E level is dependent on a NOT EXIST, an employee will go to the result if the search at the next level (A level) is empty. The search at the A level will be empty if the search at the EPA level (also a NOT EXISTS predicate) always finds a match.

The EPA level always return exactly 1 row (if the employee at the E level works on an activity at the A level) or will be empty (if the employee at the E level does not work the activity at the A level).

A non match at the EPA level returns TRUE to the A level which immediately returns FALSE to the E level rejecting that employee.

## Distinct Options

### Problem #7:

**Find employees who are currently working on a project or projects. Employees working on projects will have a row(s) on the EMPPROJECT table.**

| EMP TABLE | EMPPROJECT TABLE |        |     |
|-----------|------------------|--------|-----|
| HAAS      | HAAS             | MA2100 | 10  |
| KWAN      | HAAS             | MA2100 | 20  |
| ...       | KWAN             | IF1000 | 90  |
| ...       | KWAN             | IF2000 | 100 |
|           | ...              |        |     |

© 2012 Themis, Inc. All rights reserved.



## Distinct Options

Distinct often times causes sorts to take place of the final result set to look for and eliminate any duplicate rows. At times, there are choices that can also handle duplicates without executing the Distinct.

## Option #1: Distinct Options

**PROBLEM #7: Find employees who are currently working on a project or projects.**

```
SELECT DISTINCT E.EMPNO, E.LASTNAME  
FROM EMP E, EMPPROJECT EPA  
WHERE E.EMPNO = EPA.EMPNO
```

**Or**

```
SELECT E.EMPNO, E.LASTNAME  
FROM EMP E, EMPPROJECT EPA  
WHERE E.EMPNO = EPA.EMPNO  
GROUP BY E.EMPNO, E.LASTNAME
```

Distinct to get rid of duplicates

Group By to get rid of duplicates

© 2012 Themis, Inc. All rights reserved.



## Option #1

There are sort enhancements for both 'Distinct' and 'Group By' with no column function. This was already available prior to V9 with 'Group By' and a column function. It now handles the duplicates more efficiently in the input phase, eliminating a step 2 passing of data to a sort merge.

Prior to V9, 'Distinct' could only use a unique index to avoid a sort. 'Group By' could use both unique and non unique. Now V9 'Distinct' may take advantage of non unique indexes to avoid a sort in order to handle duplicates.

Sometimes, if one of the tables has no columns being selected from it, it can then be moved to a subquery.

## Option #2: Distinct Options

**PROBLEM #7: Find employees who are currently working on a project or projects.**

```
SELECT E.EMPNO, E.LASTNAME  
FROM EMP E  
WHERE E.EMPNO IN  
  (SELECT EMPNO  
   FROM EMPPROJECT EPA)
```

**'In'  
non correlated  
subquery**

© 2012 Themis, Inc. All rights reserved.



## Option #2

This 'In' subquery will build a list of values for the EMP 'In' predicate, and sort those values in ascending order at the same time as getting rid of duplicate values.

New in V9 are optimizer enhancements that sometimes takes an 'In' subquery and:

- Materialize the list of values into a workfile, and feeds the EMP table in a Nested

Loop Join Process

- Transforms the 'In' non correlated subquery to a correlated subquery.

V9 calls this 'Global query Optimization'



## Option #3: Distinct Options

**PROBLEM #7: Find employees who are currently working on a project or projects.**

```
SELECT E.EMPNO, E.LASTNAME  
FROM EMP E  
WHERE EXISTS  
  (SELECT 1  
   FROM EMPPROJACT EPA  
   WHERE EPA.EMPNO = E.EMPNO)
```

**'Exists'  
correlated  
subquery**

© 2012 Themis, Inc. All rights reserved.



### Option #3

This 'Exists' subquery will be checked for each employee in the EMP table. A flag gets sent on the join stating for each employee value whether that value exists or not in the EMPPROJACT table.

Because the subquery will get executed multiple times, you want to make sure an index exists on the join column(s).

# Set Operations

- Union / Union All
- Intersect / Intersect All
- Except / Except All



© 2012 Themis, Inc. All rights reserved.



---

## Set Operations

SQL Supports three set operations for combining and comparing the results of two selects. UNION combines two result sets vertically. INTERSECT and EXCEPT may be used for comparing two results vertically. All three operations require that the result sets being processed are compatible, that is, they have the same number of columns and compatible data types.

## Intersect / Intersect ALL

```
SELECT LASTNAME  
FROM NA_EMP
```

```
INTERSECT
```

```
SELECT LASTNAME  
FROM SA_EMP
```

Finds records where all  
selected columns match

Order of the tables with  
INTERSECT does not  
matter.

© 2012 Themis, Inc. All rights reserved.



## Intersect

Both of the following queries will return the same results.

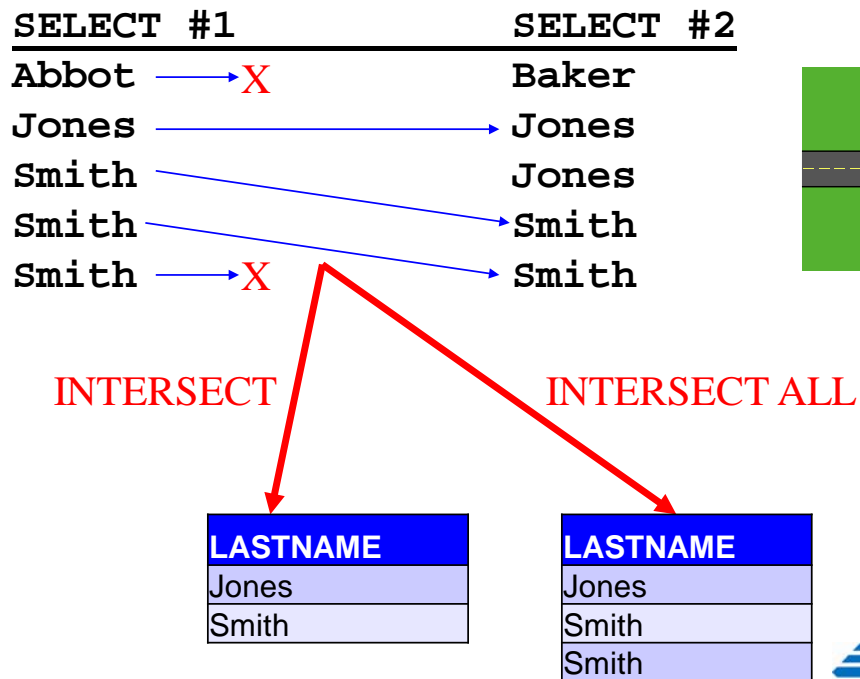
1) 

```
SELECT LASTNAME  
FROM NA_EMP  
INTERSECT  
SELECT LASTNAME  
FROM SA_EMP
```

2) 

```
SELECT LASTNAME  
FROM SA_EMP  
INTERSECT  
SELECT LASTNAME  
FROM NA_EMP
```

## Intersect / Intersect ALL



© 2012 Themis, Inc. All rights reserved.



This example demonstrates the difference between INTERSECT and INTERSECT ALL. Since Smith occurs three times in the first select but only twice in the second, INTERSECT ALL considers that there are two matches.

Once again, a duplicate is considered any record in the result where the values for all selected columns are the same.

## Exists 1 ....

```
SELECT N.LASTNAME  
FROM NA_EMP N  
WHERE EXISTS  
  (SELECT 1  
   FROM SA_EMP S  
   WHERE S.LASTNAME  
         = N.LASTNAME)
```

| LASTNAME |
|----------|
| Jones    |
| Smith    |
| Smith    |
| Smith    |
| Smith    |

## Exists 2 ...

```
SELECT S.LASTNAME  
FROM SA_EMP S  
WHERE EXISTS  
  (SELECT 1  
   FROM NA_EMP N  
   WHERE N.LASTNAME  
         = S>LASTNAME)
```

| LASTNAME |
|----------|
| Jones    |
| Jones    |
| Smith    |
| Smith    |

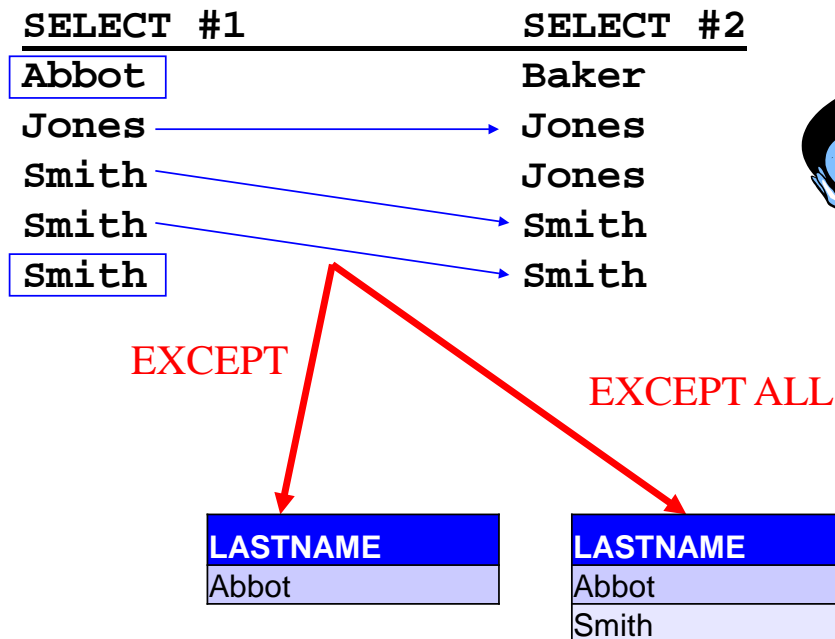
© 2012 Themis, Inc. All rights reserved.



## Other Options

Intersect is not the same as Exists logic as can be seen in the example. Exists checks one at a time whereas the intersect logic looks at all of the same name as a group in each table and compares.

# Except / Except ALL



© 2012 Themis, Inc. All rights reserved.



## Except

This example demonstrates the difference between EXCEPT and EXCEPT ALL. Smith is not returned using EXCEPT, since there are Smiths in the second result. Using EXCEPT ALL, however, one Smith is returned since there were three in the first result, but only two in the second.

Notice that EXCEPT and EXCEPT ALL will produce a different result if the SELECTs are reversed. Order is important!

## Not Exists 1 ....

```
SELECT N.LASTNAME  
FROM NA_EMP N  
WHERE NOT EXISTS  
  (SELECT 1  
   FROM SA_EMP S  
   WHERE S.LASTNAME  
         = N.LASTNAME)
```

| LASTNAME |
|----------|
| Abbot    |

## Not Exists 2 ...

```
SELECT S.LASTNAME  
FROM SA_EMP S  
WHERE NOT EXISTS  
  (SELECT 1  
   FROM NA_EMP N  
   WHERE N.LASTNAME  
         = S>LASTNAME)
```

| LASTNAME |
|----------|
| Baker    |

## Other Options

## SELECT FROM MERGE

```
SELECT ITEMNAME, UPD_IND FROM FINAL TABLE  
(MERGE INTO ITEM I  
  INCLUDE (UPD_IND CHAR(1))  
  USING (VALUES (1, 'SOCKET'))  
    AS NEWITEM (ITEMNO, ITEMNAME)  
  ON I.ITEMNO = NEWITEM.ITEMNO  
  WHEN MATCHED THEN  
    UPDATE SET ITEMNAME = NEWITEM.ITEMNAME,  
      UPD_IND = 'U'  
  WHEN NOT MATCHED THEN  
    INSERT (ITEMNO,ITEMNAME,UPD_IND)  
      VALUES (NEWITEM.ITEMNO, NEWITEM.ITEMNAME,'I') )
```

© 2012 Themis, Inc. All rights reserved.



## SELECT from MERGE

When using from a MERGE statement, it may be desirable to determine whether an INSERT or UPDATE operation was performed. This may be accomplished by using the INCLUDE clause to create an indicator variable. In the example shown, an INCLUDE column is created called UPD\_IND. When an update is performed, the UPD\_IND is set to a “U”. When an insert is performed, the UPD\_IND is set to an “I”. This value may be returned to the application via the SELECT.



# Counting With CASE

**Provide a list of total number of males and total number of females.**

```
SELECT SEX, COUNT(*)  
FROM EMP  
GROUP BY SEX
```

| SEX | 2  |
|-----|----|
| F   | 14 |
| M   | 19 |
|     |    |

© 2012 Themis, Inc. All rights reserved.



---

## CASE Statement

The CASE statement may be used in conjunction with column functions to translate ranges of values into categories. In this example, CASE is being used to supply a value of 1 or 0 for 3 separate columns in a result set that are then summed.

# Counting With CASE

**Provide a list of total number of males and total number of females.**

```
SELECT
  SUM(CASE WHEN SEX = 'F' THEN 1 ELSE 0 END)
      AS NUM_FEMALES,
  SUM(CASE WHEN SEX = 'M' THEN 1 ELSE 0 END)
      AS NUM_MALES
FROM EMP
```

| NUM_FEMALES | NUM_MALES |
|-------------|-----------|
| 14          | 19        |

© 2012 Themis, Inc. All rights reserved.



## Counting with CASE

## Counting With CASE

```
SELECT
SUM (CASE WHEN SALARY < 20000
        THEN 1 ELSE 0 END ) AS LOW,
SUM (CASE WHEN SALARY BETWEEN 20000 AND 45000
        THEN 1 ELSE 0 END ) AS MID,
SUM (CASE WHEN SALARY > 45000
        THEN 1 ELSE 0 END ) AS HI
FROM    EMP
```

| LOW | MID | HI |
|-----|-----|----|
| 7   | 23  | 3  |

# Table Pivoting

```

SELECT JOB,
       SUM(CASE WHEN DEPTNO = 'A00' THEN 1 ELSE 0 END) AS A00,
       SUM(CASE WHEN DEPTNO = 'B01' THEN 1 ELSE 0 END) AS B01,
       SUM(CASE WHEN DEPTNO = 'C01' THEN 1 ELSE 0 END) AS C01,
       SUM(CASE WHEN DEPTNO = 'D11' THEN 1 ELSE 0 END) AS D11,
       SUM(CASE WHEN DEPTNO = 'D21' THEN 1 ELSE 0 END) AS D21,
       SUM(CASE WHEN DEPTNO = 'E01' THEN 1 ELSE 0 END) AS E01,
       SUM(CASE WHEN DEPTNO = 'E11' THEN 1 ELSE 0 END) AS E11,
       SUM(CASE WHEN DEPTNO = 'E21' THEN 1 ELSE 0 END) AS E21
FROM EMP
GROUP BY JOB

```

| JOB      | A00 | B01 | C01 | D11 | D21 | E01 | E11 | E21 |
|----------|-----|-----|-----|-----|-----|-----|-----|-----|
| ANALYST  | 0   | 0   | 2   | 0   | 0   | 0   | 0   | 0   |
| CLERK    | 1   | 0   | 0   | 0   | 5   | 0   | 0   | 0   |
| DESIGNER | 0   | 0   | 0   | 8   | 0   | 0   | 0   | 0   |
| FIELDREP | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 3   |
| MANAGER  | 0   | 1   | 1   | 1   | 1   | 1   | 1   | 1   |
| OPERATOR | 0   | 0   | 0   | 0   | 0   | 0   | 4   | 0   |
| PRES     | 2   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| SALESREP | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

© 2012 Themis, Inc. All rights reserved.

## Table Pivoting

The CASE expression may also be used in conjunction with column functions to change a result from a vertical to horizontal display. In this example a GROUP BY on JOB, DEPTNO would have provided one row for each combination of JOB and DEPTNO with a count. If a tabular result is desired with one column's values as the columns on the table and another for the rows, the result may be pivoted with the CASE statement. CASE must be used to evaluate every desired department.

## New DB2 V8 SQL Features

**Following are some of the new SQL features in DB2 V8:**

- 1) More Stage 1 predicates
- 2) Multi Row Fetch, Update, and Insert
- 3) Multiple Distincts
- 4) Expressions in the 'Group By'
- 5) Common Table Expression
- 6) Dynamic Scrollable Cursors
- 7) Sequences versus Identity Columns
- 8) Materialized Query Tables (MQTs)
- 9) Recursive SQL
- 10) More efficient use of indexes. Forward and Backward scans
- 11) New XML functions and datatypes
- 12) New 'Get Diagnostics' for warning and error information
- 13) Select from an Insert statement
- 14) Scalar Fullselect within a 'Select', 'Case', Update, etc.

© 2012 Themis, Inc. All rights reserved.



---

## DB2 V8 Features

# New DB2 V9 SQL Features

**Following are some of the new SQL features in DB2 V9:**

- 1) Set operations 'Intersect' and 'Except'
- 2) Merge statement for 'Upsert' processing. Insert or Update
- 3) OLAP features for Ranking and Numbering of data
- 4) Native SQL Stored Procedures
- 5) 'Instead of' Triggers
- 6) New support and SQL for XML data
- 7) Optimization Service Center (OSC)
- 8) Distinct sort avoidance with non unique indexes
- 9) Indexing on Expressions
- 10) Statistics on Views
- 11) Skipped locked data
- 12) Truncate statement

© 2012 Themis, Inc. All rights reserved.



---

## DB2 9 Features

Native SQL: No 'C' compilation. Fully integrated into DB2, Versioning

Instead of Triggers: Used on views.

Truncate: Options to bypass and 'delete' triggers, 'Drop' its storage. Fast delete. Useful when deleting tables on a nightly basis for nightly refreshes. Very fast delete.

Skip Locked Data: You can use the SKIP LOCKED DATA option with cursor stability (CS) isolation and read stability (RS) isolation. However, you cannot use SKIP LOCKED DATA with uncommitted read (UR) or repeatable read (RR) isolation levels.

# New DB2 V9 SQL Features

**Following are some of the new SQL features in DB2 V9:**

- 13) Array host variables now supported for stored procedures
- 14) Timestamp auto update for inserts and Updates
- 15) Optimistic locking
- 16) New DECFLOAT datatype
- 17) Select from Update or Delete getting old or new values
- 18) Fetch First, Order BY within subqueries
- 19) REOPT AUTO (Dynamic SQL)
- 20) Data Studio for Native Stored Procedures

© 2012 Themis, Inc. All rights reserved.



**Optimistic locking:** Built in timestamp automatically updated by DB2. Allows for checking that a row has not changed when updating from the last time it was selected.

**REOPT Auto:** The ideas behind REOPT(AUTO) is to come up with the optimal access path in the minimum number of prepares.

# New DB2 V10 SQL Features

**Following are some of the new SQL features in DB2 V10:**

- 1) OLAP Moving Sum, Count, and Average
- 2) Extended Null Indicators
- 3) New timestamp Precisions
- 4) Currently Committed Data
- 5) SQL PI Enhancements
- 6) XML Enhancements
- 7) Row Permissions
- 8) Column Masking
- 9) Temporal Tables

© 2012 Themis, Inc. All rights reserved.



---

## DB2 10 Features



# Training and Consulting. Check Out [www.themisinc.com](http://www.themisinc.com)

- On-site and Public
- Instructor -led
- Hands-on
- Customization
- Experience
- Over 25 DB2 courses
- Over 400 IT courses



US 1-800-756-3000  
Intl. 1-908-233-8900

© 2012 Themis, Inc. All rights reserved.



Education. Check out  
[www.db2sqltuningtips.com](http://www.db2sqltuningtips.com)

**Finally! A book of DB2 SQL tuning tips for developers, specifically designed to improve performance.**

**DB2 SQL developers now have a handy reference guide with tuning tips to improve performance in queries, programs and applications.**



© 2012 Themis, Inc. All rights reserved.



# Thank You for allowing us at Themis to share some of our experience and knowledge!

- We hoped that you learned something new today
- We hope that you are a little more inspired to code with performance in mind

*“I have noticed that when the developers get educated, good SQL programming standards are in place, and program walkthroughs are executed correctly, incident reporting stays low, CPU costs do not get out of control, and most performance issues are found before promoting code to production.”*

© 2012 Themis, Inc. All rights reserved.



***This Page Intentionally Left Blank***