

The Unconstrained Primary Key



Dan Cruikshank



In this presentation I build upon the concepts that were presented in my article “The Keys to the Kingdom”. I will discuss how primary and unique keys can be utilized for something other than just RI. In essence, it is about laying the foundation for data centric programming. I hope to convey that by establishing some basic rules the database developer can obtain reasonable performance. The title is an oxymoron, in essence a Primary Key is a constraint, but it is a constraint that gives the database developer more freedom to utilize an extremely powerful relational database management system, what we call DB2 for i.

Agenda

- **Keys to the Kingdom**
- **Exploiting the Primary Key**
- **Pagination with ROW_NUMBER**
- **Column Ordering**
- **Summary**

I will review the concepts I introduced in the article “The Keys to the Kingdom” published in the Centerfield. I think this was the inspiration for the picture. I offered a picture of me sitting on the throne, but that was rejected.

I will follow this with a discussion on using the primary key as a means for creating peer or subset tables for the purpose of including or excluding rows in a result set.

The ROW_NUMBER function is part of the OLAP support functions introduced in 5.4. Here I provide some examples of using ROW_NUMBER with the BETWEEN predicate in order to paginate a result set.

Finally, I will wrap up with some column ordering techniques. I provide examples of using DYNAMIC SQL to provide the ORDER BY string for the ROW_NUMBER function.

There is a method to this madness as ordering is the last thing the DB2 optimizer addresses.

We will then summarize and answer any questions, time permitting.

Keys to the Kingdom

- **Constraints**
- **Types of Constraints**
- **Where constraints are used**

Lets talk about constraints or what I consider are the true keys to the database kingdom.

Constraints

- **Define unconstrained**

- Adjective

- Meaning

- free from constraint

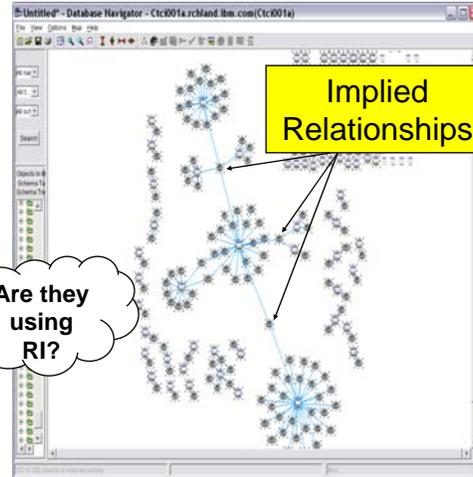
- *Usage example*

- *“they were unconstrained by the boundaries associated with standards or best practices, thus they coded with complete abandon”*



Are they using RI?

System i Database Navigator



Lets look up unconstrained on the internet and find out what it means, simply without constraint. Well, that says it all. Nothing like a usage example to provide more understanding. Some of you may recognize the usage example from an Application Performance Review I may have recently performed at your location.

One thing I love to do is use System i Database Navigator to map your database. Mapping a single database file can produce a display that is like Google Earth, <click> as we zoom in on the ink blots they begin to take shape and form relationships.

<click> This brings up one of my favorite questions “ I wonder if they are using Referential Integrity”. And then my next favorite question “What would happen if I deleted this row here, or that row there?” This usually grabs the attention of the CIO.

Types of Constraints

- **Unique Key**
 - One or more columns, more than 1 per table
- **Primary Key**
 - Only 1 per table, BIGINT AS IDENTITY
- **Referential (Foreign Key)**
 - Ensures integrity of relationships
- **Check**
 - Ensures integrity of the data
- **Note: Primary, Unique and Referential constraints are radix indexes under the covers**

These are the constraints that you can define on DB2 for i. Using my housing database example from the keys to the kingdom, I like to think of these as the plumbing and wiring of a house. Get this wrong and you can end up paying through the nose for a long time to come.

A good database design practice is to define a unique key for each and every database table. A unique key allows you to retrieve a single row. It also informs DB2 that the value contained in the unique column will never appear in more than 1 row. Can make a difference when DB2 is producing a plan for a query that has the unique key column on a WHERE clause.

DB2 for i supports both a Primary and Unique Key constraint. Why have 2, you ask. Well, as I pointed out in the keys to kingdom, a unique key is the application key. The primary key is used to establish relationships between tables. We see an example of this in upcoming slides.

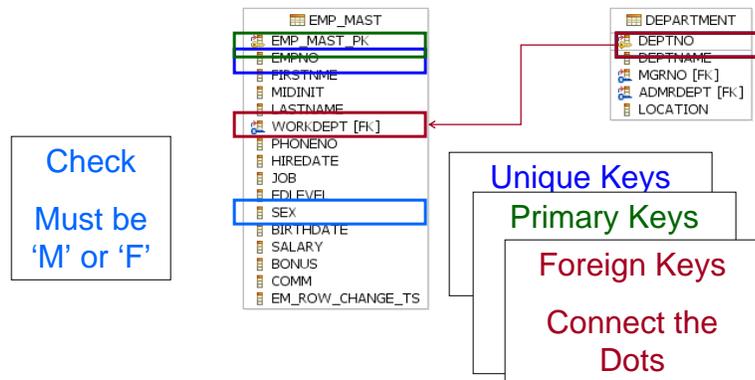
The referential or foreign key constraint ensures the integrity of the relationship by protecting the database from inadvertent deletes or inserts. More on this later.

A check constraint validates the column data within a row.

The key constraints create binary radix indexes which can be used for both optimization and implementation.

Where constraints are used?

▪ Data Model of the Employee Master and Department tables



6

www.ibm.com/systems/services/labservices

© 2009 IBM Corporation

Let's use a logical data model (LDM) of the Employee and Department tables to demonstrate where constraints are used.

<click>

1. These are the unique keys for the database. The main purpose of the unique key is to retrieve a single row. This may also be referred to as a natural key. I call it the application key.

<click>

2. These are the primary keys. The primary key is used to establish referential integrity between related tables. It is hidden from the applications. Note that the DEPTNO unique key for the department table also serves as a primary key..

<click>

3. This is the RI foreign key constraint between the two tables. This constraint ensures the integrity between dependent relations. In the case of a validation or lookup relationship, we would probably restrict the department table row from being deleted if it is referenced by an employee row. We would also restrict an employee row from being added,

<click>

4. This is the check constraint used for the column SEX. This ensures that the data entered into this column is valid. I assume that this means gender. It is amazing what these fields contain when they are unconstrained.

This is the foundation for data centric programming. Not one line of application code needs to be written to ensure the integrity of the data.

Agenda

- Keys to the Kingdom
- **Exploiting the Primary Key**
- Pagination with ROW_NUMBER
- Column Ordering
- Summary

I will now discuss how we utilize the primary key in data centric or database programming.

Exploiting the Primary Key

- **Inclusion and Exclusion Tables**

- New Column or New Table?
 - Unconstrained or Constrained?

- **Data Centric**

- Using database features, functions and programming techniques to solve business problems.
 - Auto-Generated Fields
 - Identity column, row change timestamp, etc.
 - Referential Integrity
 - SQL Joins and Views
 - Instead of Triggers

We will begin to look at ways that we can exploit this concept of a primary key. Here I introduce the concept of inclusion and exclusion, aka peer or subset tables. This table is characterized by a one to maybe one relationship with the parent table.

I also introduce the concepts of application versus data centric development. In essence, by data centric I mean utilizing the capabilities of the database to solve business problems. This includes the use of identity columns and other types of auto-generated values. Utilizing join technology, views and instead of triggers are other forms of data centric development.

A Simple Request

- **Employee Listing within Department Request**

- **Need to view by:**

- Active only
- Inactive Only
- All
- Ability to order on any column



- **Issues**

- EMP_MAST does not contain a status column
- Sensitive data
- Ordering on multiple columns

EMP_MAST
EMP_MAST_PK
EMPNO
FIRSTNME
MIDINIT
LASTNAME
WORKDEPT [FK]
PHONENO
HIREDATE
JOB
EDLEVEL
SEX
BIRTHDATE
SALARY
BONUS
COMM
EM_ROW_CHANGE_TS

So you just received a request for a new HR application. HR needs to see a list of all active or inactive employees. A department may have dozens of employees so they would like to be able to order the list by any column. Seems like a reasonable request until you review the employee master table.

<click>

There is no column to designate whether an employee is active or inactive. In addition, the employee master contains sensitive data. And you know they will want to order on multiple columns (e.g. lastname, firstnme, etc). What approach do you use to solve this business problem.

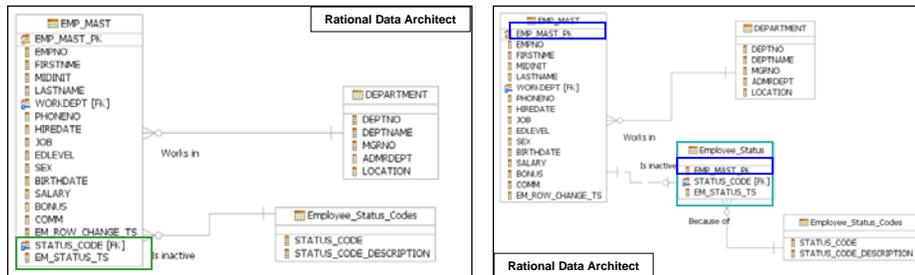
Inclusion and Exclusion Tables

Unconstrained

- Requires modification to EMP_MAST
 - Impact on existing apps
- Status changes are made via updates
 - Journal (logging) issues
- Potential for many new EMP_MAST indexes
 - Adversely effects scalability

Constrained

- No modification to existing table
- Status changes occur to separate table
 - Minimizes journal impact
- No new indexing requirement for EMP_MAST
 - Highly scalable



10

www.ibm.com/systems/services/labservices

© 2009 IBM Corporation

How do you address the problem?

<click>

The unconstrained approach is to add the necessary columns to the employee master table. This may seem like a simple solution, until you consider the impact on the rest of the system.

What about all the journal logs that may contain the entire record image including sensitive data? And don't forget the impact of additional index requirements for performance. Every column is a potential index. Two new columns, two new indexes (A+B, B+A). Wheel in 2 existing columns, WORKDEPT and LASTNAME and you now have 24 potential indexes (4*3*2*1). Maybe it's time to consider an alternative approach.

<click>

Now is the time to take advantage of DB2 for i and exploit it's capabilities. Here I create an inclusion/exclusion table which contains the primary key <click> from the employee master, along with the new status code field and row change timestamp column which contains the date the row is created. All update activity occurs against this table, reducing the overhead of journal and index maintenance and increasing the overall scalability of the employee database.

So how do you keep it all together?

IBM Systems Lab Services and Training IBM

Data Centric Programming Tip #1 Use Referential Integrity

EMP_MAST

- EMP_MAST_PK
- EMPNO
- FIRSTNAME
- MIDINIT
- LASTNAME
- WORKDEPT [FK]
- PHONENO
- HIREDATE
- JOB
- EDLEVEL
- SEX
- BIRTHDATE
- SALARY
- BONUS
- COMM
- EM_ROW_CHANGE_TS

Check for existence (SETLL, CHAIN)
If found then Add

Participation
One to Maybe One

Mandatory

Optional

Employee_Status

- EMP_MAST_PK [FK]
- STATUS_CODE [PK]
- EM_STATUS_TS



Constraint created here

EM must exist before adding ES
Deleting EM deletes ES
ES can be deleted
EM = Emp_MAST, ES = Employee_Status

**ALTER TABLE ES ADD CONSTRAINT
EM_ES_FK FOREIGN KEY
(EMP_MAST_PK) REFERENCES EM
(EMP_MAST_PK) ON DELETE
CASCADE;**

11 | www.ibm.com/systems/services/labservices | © 2009 IBM Corporation

This brings us to data centric programming tip # 1 – Use Referential Integrity. Lets see an example.

This is a data model of the Employee Master (EM) table and the Employee Status (ES) table showing the relationship between the two tables.

<Click>

The FK designation indicates EM_PK is a foreign key of EM implying EM is the parent table. As the parent, the participation of the EM table is mandatory. The circle followed by a single bar indicates that the participation of the ES table is optional. The degree of participation (aka cardinality) for EM and ES is 1, but no more than 1. This is referred to as a one-to-maybe one relationship. A unique key constraint on EM_PK in ES prevents more than 1 row per EM_PK. In the application centric world I would need to perform an existence check using a random IO against either table before deleting or inserting a new row. Not necessary with data centric programming.

<click>

Adding the referential constraint pushes the existence checking down to the database layer. DB2 for i ensures that the EM row exists before creating an ES row. The CASCADE rule ensures the ES row is deleted when the EM row is deleted. In addition an EM row can be created without an EA row. An EA row can be deleted at any time.

Data Centric Programming Tips 2 & 3 Use SQL **Joins** and **Views**

- Active only view

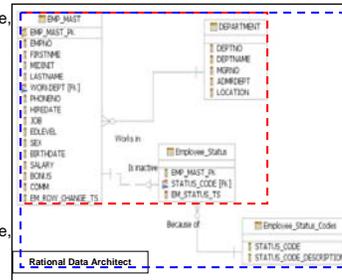
```
CREATE VIEW ACTIVE_EMPLOYEES AS
SELECT em.empno, em.workdept deptno, d.deptname, em.lastname,
       em.firstname, em.midinit, CHAR('Activ',5)
       status_code,em.hiredate active_since
FROM   emp_mast EM EXCEPTION JOIN
       Employee_Status EX USING(emp_mast_PK)
       JOIN dept_mast D ON em.workdept = d.deptno;
```

- Inactive Only view

```
CREATE VIEW INACTIVE_EMPLOYEE_LIST AS
SELECT em.empno, em.workdept deptno, d.deptname, em.lastname,
       em.firstname, em.midinit, es.status_code_description,
       DATE(ex.em_status_ts) as inactive_date
FROM   emp_mast EM
       JOIN Employee_Status EX USING(emp_mast_PK)
       JOIN Employee_Status_Codes ES USING(status_code)
       JOIN dept_mast d ON em.workdept = d.deptno;
```

- Notes

- EXCEPTION JOIN same as NOT IN or NOT EXISTS subquery
- JOIN same as IN or EXISTS subquery
- Other join types: RIGHT EXCEPTION, LEFT or RIGHT OUTER and CROSS



It's a Wrap

This brings us to data centric programming tips 2 and 3-Use SQL joins and views. An SQL statement is like a program. A common mistake made by new SQL programmers is to treat SQL like an IO operation, Fetch a row then Select from this and Select from that. Put as much as you can into a single statement. Let DB2 for i do the heavy lifting.

My data model contains 4 tables. In order to produce an active employee list I must check for existence in the i/x table. If a matching row is not present then the EM row is selected. I accomplish using an EXCEPTION JOIN followed by an inner join to the department table. Note that the active list uses a derived field for the status.

The LEFT OUTER JOIN can be used to produce a list of all employees with the inactive status description.

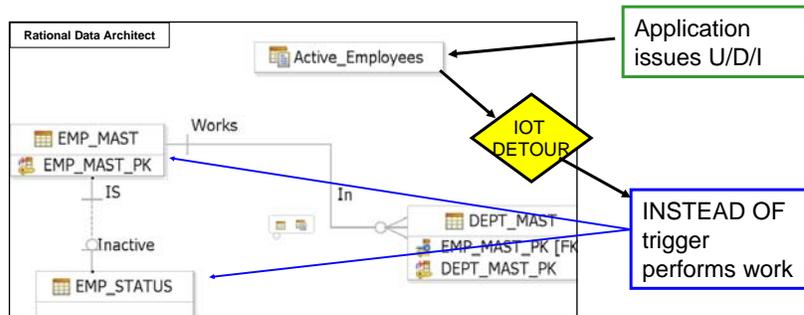
The inactive employee list is a 4 table join. In this case an INNER JOIN is used and if the i/x row is present then the EM row is selected. The status code from the ES table is used to join to the code lookup table to return a description. This is not a derived column.

The SQL view serves two purposes; first it hides the plumbing and second it makes the complex SQL statement reusable. Any application program can use the view.

How do you activate or inactivate an Employee?

Data Centric Programming Tip # 4 Use Instead Of Triggers

- **INSTEAD OF Trigger (IOT)**
 - Enables updates/deletes or inserts against an SQL view which is considered Read-Only
 - A join would be one such view
- **In the diagram, the view Active_Employees is a join of three tables**
 - An UPDATE, DELETE or INSERT to the Active_Employees view will fail without an INSTEAD OF Trigger



13

www.ibm.com/systems/services/labservices

© 2009 IBM Corporation

Brings us to tip number 4-Instead of Triggers.

An INSTEAD OF trigger enables the use of update, delete and/or insert operations to be performed against an SQL view which is considered to be non-updateable, non-deleteable and/or non-insertable. The new ACTIVE_EMPLOYEES view contains a join making it READ ONLY.

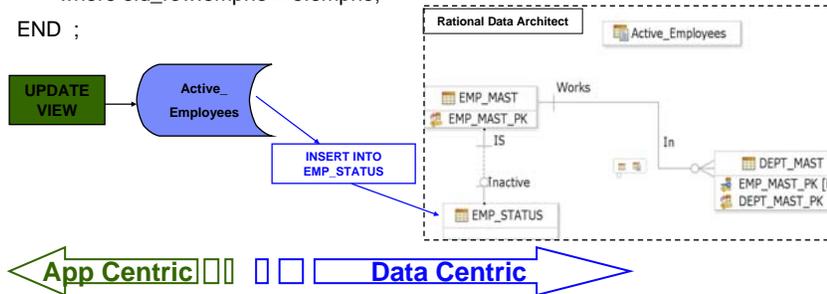
<click> An UPDATE, DELETE or INSERT performed against a READ ONLY view will fail unless <click> an INSTEAD OF trigger is attached to the view. The original statement is detoured and <click> the heavy lifting is performed by the IOT.

Inactivate Employee using INSTEAD OF Triggers

```

CREATE TRIGGER INACTIVATE_EMPLOYEE
INSTEAD OF UPDATE ON ACTIVE_EMPLOYEES
REFERENCING OLD Old_row NEW New_row
FOR EACH ROW
BEGIN ATOMIC
  INSERT INTO emp_status (emp_mast_pk) select emp_mast_pk from emp_mast e
  where old_row.empno = e.empno;
END ;

```



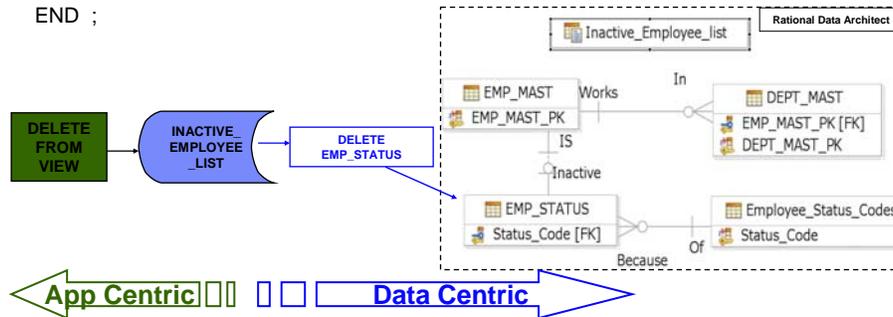
The application issued an **UPDATE** against the view which is transformed into an **INSERT** against the ES table.

Re-Activate Employee Using INSTEAD OF Triggers

```

CREATE TRIGGER REINSTATE_EMPLOYEE
INSTEAD OF DELETE ON INACTIVE_EMPLOYEE_LIST
REFERENCING OLD Old_row
FOR EACH ROW BEGIN ATOMIC
  DELETE FROM emp_status e
  WHERE emp_mast_pk = (select emp_mast_pk from emp_mast e
                      where old_row.empno = e.empno);
END ;

```



In this example the application issued a **DELETE** against the view which is transformed to a **DELETE** against the ES table.

Example of IOT In Action

```
CALL DB2SANDBOX/LIST_ACTIVE_EMPLOYEES
('D11', 30,1, '3',0,0);
```

```
UPDATE ACTIVE_EMPLOYEES SET
STATUS_CODE = 'I' WHERE EMPNO = '587268';
```

```
UPDATE ACTIVE_EMPLOYEES SET
STATUS_CODE = 'MLOA' WHERE EMPNO =
'587268';
```

```
COMMIT;
```

```
CALL
DB2SANDBOX/LIST_ACTIVE_EMPLOYEES
('D11', 30,1, '3',0,0);
```

```
SELECT * FROM INACTIVE_EMPLOYEE_LIST;
```

```
DELETE FROM INACTIVE_EMPLOYEE_LIST
WHERE EMPNO = '587268';
```

```
COMMIT;
```

```
CALL
DB2SANDBOX/LIST_ACTIVE_EMPLOYEES
('D11', 30,1, '3',0,0);
```

Message: [SQL0930] Operation not allowed by referential constraint EMPLOYEE_STATUS_EMPLOYEE_STATUS_CODES_PK in 'MLOA'. If this is an INSERT or UPDATE statement, the value is not valid for the foreign key because it does not have a matching value in the parent key. If this is a DELETE statement affected by a SET DEFAULT delete rule, the default value is not valid for the same reason. If this is an ALTER TABLE statement, the result of the operation would violate the constraint EMPLOYEE_STATUS_EMPLOYEE_STATUS_CODES_PK. Constraint EMPLOYEE_STATUS_EMPLOYEE_STATUS_CODES_PK in 'MLOA' requires that any non-null value of the foreign key have a matching value in the parent key. Recovery ...: To conform to the constraint rule, you must either: - change the INSERT or UPDATE value to ...

EMPNO	DEPTNO	DEPTNAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	ACTIVE
587268	D11	MAN.FA...	Borell	Ruben	K	Actv	2008-12-
200009	D11	MAN.FA...	Kohts	Reed	B	Actv	2008-12-
418112	D11	MAN.FA...	Ehagator	Gilbert	H	Actv	2008-12-
109100	D11	MAN.FA...	Francisque	Dylan	S	Actv	2008-12-
973314	D11	MAN.FA...	Grandsstaff	Marc	G	Actv	2008-12-
000060	D11	MAN.FA...	STERN	IRVING	F	Actv	1973-09-14
000150	D11	MAN.FA...	ADAMSON	BRUCE		Actv	1972-02-12
000160	D11	MAN.FA...	PLANKA	ELIZABETH	R	Actv	1977-10-11

NAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	DESCRIPTION
FACTURING SYSTEMS	Borell	Ruben	K		Maternity Leave of Absence
ISTRATION SYSTEMS	PEREZ	MARIA	L		Sick Leave of Absence

EMPNO	DEPTNO	DEPTNAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	ACTIVE
587268	D11	MAN.FA...	Borell	Ruben	K	Actv	2008-12-
200009	D11	MAN.FA...	Kohts	Reed	B	Actv	2008-12-
418112	D11	MAN.FA...	Ehagator	Gilbert	H	Actv	2008-12-
109100	D11	MAN.FA...	Francisque	Dylan	S	Actv	2008-12-
973314	D11	MAN.FA...	Grandsstaff	Marc	G	Actv	2008-12-
000060	D11	MAN.FA...	STERN	IRVING	F	Actv	1973-09-09
000150	D11	MAN.FA...	ADAMSON	BRUCE		Actv	1972-02-
000160	D11	MAN.FA...	PLANKA	ELIZABETH	R	Actv	1977-10-
000170	D11	MAN.FA...	YOSHIMU...	MASATO...	J	Actv	1976-09-
000180	D11	MAN.FA...	SCOUTTEN	MARILYN	S	Actv	1973-07-

In this example, the end user executes the LIST_ACTIVE_EMPLOYEES program to display the list of employees. We will examine this procedure in a few moments. The user needs to make Ruben Borrell inactive. <click>

The user enters an I in the status code field and the program fails. Examination of the joblog reveals that a referential constraint violation has occurred. <click>

The user enters a valid code and a refresh of the active list verifies Ruben is gone. <click>

A sanity check reveals that Ruben is not in the inactive employee listing, however the Status code used was not appropriate. <click>

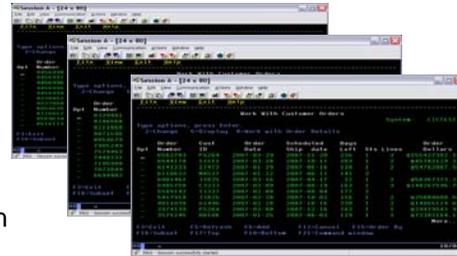
Oh no, call for backup.

Agenda

- Keys to the Kingdom
- Exploiting the Primary Key
- **Pagination with ROW_NUMBER**
- **Column Ordering**
- **Summary**

Pagination

- a.k.a. Page at a time
- Stateful
 - Persistent
 - 5250 Subfile application
- Stateless
 - Connectionless
 - Client based or
 - Web Browser application



DEPTNO	DEPTNAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	ACTIVE SINCE
1	D11	MANUFACT...	ADAMSON	BRUCE	Actv	1970-02-12
2	D11	MANUFACT...	Actv	1986-02-08
3	D11	MANUFACT...	Actv	...
4	D11	MANUFACT...	Actv	...
5	D11	MANUFACT...	Eberkamp	May	t	2009-09-09
6	D11	MANUFACT...	Starkamp	Wb	V	1996-09-10
7	D11	MANUFACT...	Actv	...
8	D11	MANUFACT...	Actv	...
9	D11	MANUFACT...	Actv	...
10	D11	MANUFACT...	Actv	...
11	D11	MANUFACT...	Actv	...
12	D11	MANUFACT...	Actv	...
13	D11	MANUFACT...	Actv	...
14	D11	MANUFACT...	Actv	...
15	D11	MANUFACT...	Actv	...
16	D11	MANUFACT...	Actv	...
17	D11	MANUFACT...	Actv	...
18	D11	MANUFACT...	Actv	...
19	D11	MANUFACT...	Actv	...
20	D11	MANUFACT...	Actv	...
21	D11	MANUFACT...	Actv	...
22	D11	MANUFACT...	Actv	...
23	D11	MANUFACT...	Actv	...
24	D11	MANUFACT...	Actv	...
25	D11	MANUFACT...	Actv	...
26	D11	MANUFACT...	Actv	...
27	D11	MANUFACT...	Actv	...
28	D11	MANUFACT...	Actv	...
29	D11	MANUFACT...	Actv	...
30	D11	MANUFACT...	Actv	...

RDA – Data Source Explorer

Stateful means the computer or program keeps track of the state of interaction, usually by setting values in a storage field designated for that purpose.

Stateless means there is no record of previous interactions and each interaction request has to be handled based entirely on information that comes with it.

The term *connectionless* is also used to describe communication in which a connection is made and terminated for each message that is sent. IP is connectionless as well as stateless. Each communication is discrete and unrelated to those that precede or follow. In order to have stateful communication, a site developer must furnish a special program that the server can call that can record and retrieve state information. Some Web browsers provide an area in their subdirectories where state information can be stored and accessed. The area and the information that Web browsers and server applications put in this area is called a [cookie](#).

Those of us that have written subfile programs understand pagination. As the subfile was paged forward the program kept the next file position. As long as the program remained active the cursor position would point to the next record. A page down request simply allowed the program to read the next n rows to fill the page. When the job or program ended the cursor positions and subfile data were flushed.

With client or web based applications the list application is always disconnected after retrieving a set of rows. When the end of a set is reached a new connection is made to the server. Somehow the application has to begin

Data Centric Programming Tip # 5 Pagination Using ROW_NUMBER

```

CREATE PROCEDURE LIST_ACTIVE_EMPLOYEES (
  IN p_WORKDEPT CHAR(3), IN p_rows_to_return INTEGER,
  IN p_Starting_Row INTEGER, OUT p_ROW_COUNT INTEGER)
  RESULT SETS 1 LANGUAGE SQL SPECIFIC SPACTEMP
P1: BEGIN
  DECLARE v_dynSQL VARCHAR(1000);
  DECLARE v_Ending_Row INTEGER;
  DECLARE SP_CT CURSOR WITH RETURN TO CLIENT FOR SP_s1;
  SET v_dynSQL =
    'WITH cte_row_number as (' ||
    '  SELECT EMPNO, ' ||
    '    ROW_NUMBER () OVER (ORDER BY lastname, firstname, midinit) AS CTE_RRN ' ||
    'FROM ACTIVE_EMPLOYEES WHERE DEPTNO = ?) ' ||
    'SELECT AE.* FROM ACTIVE_EMPLOYEES AE ' ||
    'JOIN cte_row_number USING (EMPNO) ' ||
    'WHERE CTE_RRN BETWEEN ? AND ?
  ORDER BY lastname, firstname, midinit';
  PREPARE SP_s1 FROM v_dynSQL;

  SET v_Ending_Row = p_Starting_Row + (p_rows_to_return - 1);
  OPEN SP_C1 USING p_WORKDEPT, p_Starting_Row, v_Ending_Row;
END P1

```

Variables used to control pagination

EMPNO used to join CTE to View

Common Table Expression used to gen result set RRN

The ROW_NUMBER function was introduced in 5.4. Basically the function generates an RRN for each row in the result set.

<click> In this example, the stored procedure receives the page positioning variables (rows to return and starting row) and uses these to determine the start and end rows of a page.

<click> A CTE is used to generate the row number based on the ordering criteria. The CTE also contains the unique key as this is known to the application, the PK is not a part of the view. The row number column is used on the WHERE clause to select only the rows that are required for the page.

<click> The CTE is then joined back to the active employee view using the unique key contained in the view and CTE.

Example of Pagination Using ROW_NUMBER

CALL LIST_ACTIVE_EMPLOYEES
('D11',10, 1,0);

Beginning row is 1

CALL LIST_ACTIVE_EMPLOYEES
('D11',10, 11,0);

Beginning row is 11

CALL LIST_ACTIVE_EMPLOYEES
('D11',10, 21,0);

Beginning row is 21

RDA – Data Source Explorer

DEPTNO	DEPTNAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	ACTIVE_SINCE
1	D11	MANUFACTU...	ADAMSON	BRUCE	Actv	1972-02-12
2	D11	MANUFACTU...	Bar	Steve	Q	1994-07-09
3	D11	MANUFACTU...	Borrell	Ruben	K	2009-12-28
4	D11	MANUFACTU...	Byher	Chris	P	1999-01-08
5	D11					
6	D11					
7	D11					
8	D11					
9	D11					
10	D11					
11	D11					
12	D11					
13	D11	MANUFACTU...	Francisqa	Lynn	S	2008-12-29
14	D11	MANUFACTU...	Gordon	Hema	S	1989-11-12
15	D11	MANUFACTU...	Grandstaff	Marc	G	2008-12-30
16	D11	MANUFACTU...	Gubbert	Vaughn	P	2009-01-04
17	D11					
18	D11					
19	D11					
20	D11					
21	D11					
22	D11	MANUFACTU...	Landsberger	Victoria	h	1995-09-02
23	D11	MANUFACTU...	Lerner	Ann	x	1999-12-04
24	D11	MANUFACTU...	LUTZ	JENNIFER	K	1989-09-29
25	D11	MANUFACTU...	Mahija	Lindsay	C	1993-12-01
26	D11	MANUFACTU...	Margolis	Madeline	G	1994-02-11
27	D11					
28	D11	MANUFACTU...	Mather	Chasen	h	1988-12-12
29	D11	MANUFACTU...	Paculci	Taylor	r	1989-07-02

CTE_RRN BETWEEN 1 and 10

CTE_RRN BETWEEN 11 and 20

CTE_RRN BETWEEN 21 and 30

The stored procedure is called from a client. Each request could be serviced by different server jobs.

ROW_NUMBER Benefits

- **Server job handles multiple requests**
 - Connection pooling
- **First request builds Open Data Path (a.k.a Full Open)**
 - ODP not deleted after second request
 - Subsequent requests reuse ODP
- **Minimizing Full Opens reduces system resource usage**
 - Increases throughput

System i Navigator-Run SQL Scripts

Message ID Message

```

@SQL7959    Cursor SPACTEMP01_C1 was closed.
@SQL7914    ODP not deleted.
@SQL7963    2 rows fetched from cursor CRSR0007.
@SQL7962    Cursor SPACTEMP01_C1 opened.
@SQL7911    ODP reused.
@SQL7967    PREPARE of statement SPACTEMP01_S1 completed.
@SQL7959    Cursor SPACTEMP01_C1 was closed.
@SQL7914    ODP not deleted.
@SQL7963    10 rows fetched from cursor CRSR0010.
@SQL7962    Cursor SPACTEMP01_C1 opened.
@SQL7916    blocking used for query.
@SQL7912    ODP created.
  
```

Run SQL Scripts-View Joblog

For best performance use connection pooling. This way the cursor (or ODP) can be reused by many different requests.

The System i Navigator figure in the upper right hand corner simulates multiple requests being serviced by different jobs. Each request may be processing different departments and page ranges, however they can reuse the same cursor created for the job.

This is shown in the joblog for one of the server jobs. Note the messages verifying the ODP is not deleted and reused.

Agenda

- Keys to the Kingdom
- Exploiting the Primary Key
- Pagination with ROW_NUMBER
- **Column Ordering**
- **Summary**

Column Ordering

- **Provide ability to sort result set on any column**
 - Multiple columns
 - Ascending or Descending
- **Current stored procedure**
 - CALL LIST_ACTIVE_EMPLOYEES ('D11',10, 1,0);
 - Needs another parameter

Status	Parameters	Result					
	DEPTNO	DEPTNAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	ACTIVE_SINCE
1	D11	MANUFACTURING SYSTEMS	ADAMSON	BRUCE		Activ	1972-02-12
2	D11	MANUFACTURING SYSTEMS	Bax	Steve	Q	Activ	1994-07-09
3	D11	MANUFACTURING SYSTEMS	Borell	Rueben	K	Activ	2008-12-18
4	D11	MANUFACTURING SYSTEMS	Byther	Otto	P	Activ	1999-01-08
5	D11	MANUFACTURING SYSTEMS	BROWN	DAVID		Activ	1966-03-03
6	D11	MANUFACTURING SYSTEMS	Colvard	Jenae	h	Activ	1980-02-07
7	D11	MANUFACTURING SYSTEMS	Deliberato	Caroyln	g	Activ	2006-01-02
8	D11	MANUFACTURING SYSTEMS	Dobyns	Annemarie	R	Activ	1991-07-05
9	D11	MANUFACTURING SYSTEMS	Ehigiator	Gilbert	H	Activ	2008-12-18
10	D11	MANUFACTURING SYSTEMS	Eichenauer	Roy	b	Activ	1987-04-05

RDA – Data Source Explorer

Providing the ability to sort on any column in a list is a very user attractive feature for any graphical interface. Somehow the server must be informed which column or columns to order by based on a user click of the column.

Data Centric Programming Tip # 6 Use DYNAMIC SQL

```

CREATE PROCEDURE LIST_ACTIVE_EMPLOYEES (
  IN p_WORKDEPT CHAR(3) ,IN p_rows_to_return INTEGER ,
  IN p_Starting_Row INTEGER, IN p_ORDER_BY_COLUMNS VARCHAR(50), ...
RESULT SETS 1 LANGUAGE SQL SPECIFIC SPACTEMP
P1: BEGIN
  DECLARE v_dynSQL VARCHAR(1000);
  DECLARE v_Ending_Row INTEGER;
  DECLARE SPACTEMP_C1 CURSOR WITH RETURN TO CLIENT FOR SPACTEMP_s1;
  SET v_dynSQL =
  'WITH cte_row_number as (SELECT EMPNO, '
  '  ROW_NUMBER () OVER (ORDER BY ' || RTRIM(p_ORDER_BY_COLUMNS) || ' AS CTE_RRN) '
  ' FROM ACTIVE_EMPLOYEES WHERE DEPTNO = ?) ' ||
  'SELECT AE.* FROM ACTIVE_EMPLOYEES AE JOIN cte_row_number using (empno) ' ||
  'WHERE CTE_RRN BETWEEN ? AND ? ORDER BY CTE_RRN';
  PREPARE SPACTEMP_s1 FROM v_dynSQL;
  SET v_Ending_Row = p_Starting_Row + (p_rows_to_return - 1);
  OPEN SPACTEMP_C1 USING p_WORKDEPT, p_Starting_Row, v_Ending_Row;
END P1

```

Column ordering
passed as variable

Dynamic
SQL
2MB
Limit

CTE_RRN
Dual Purpose

Providing selective column ordering is easy with Dynamic SQL.

<click> Here I've added another parameter which is used to pass the column names or column positions as a variable. The variable is concatenated to the ORDER BY clause of the ROW_NUMBER function.

<click> The SQL DECLARE statement declares a cursor based on a statement name. The entire SQL statement is a string (**v_dynSQL**) which is built at execution and then associated with the statement name as part of the PREPARE statement.

<click> The ROW_NUMBER function is now performing double duty, pagination and ordering.

Example of Dynamic Column Ordering with Pagination

CALL LIST_ACTIVE_EMPLOYEES (
'D11',30, 1,'**LASTNAME, FIRSTNAME,
MIDINIT**', 0);

CALL LIST_ACTIVE_EMPLOYEES (
'D11',30, 1,'**7 DESC, 3,4,5**', 0);

DEPTNO	DEPTNAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	ACTVE_SINCE
1	MAN.FA...	ADAMS	BRUCE		Activ	1972-02-12
2	MAN.FA...	Bax	Steve	Q	Activ	1994-07-09
3	MAN.FA...	Borell	Rueben	K	Activ	2009-12-18
4	MAN.FA...	Byher	Otto	P	Activ	1999-01-08
5	MAN.FA...	BROWN	DAVID		Activ	1966-03-03
6	MAN.FA...	Colvard	Jenae	h	Activ	1980-02-07
7	MAN.FA...	Deliberato	Carolyn	g	Activ	2006-01-02
8	MAN.FA...	Dobyns	Annemarie	R	Activ	1991-07-05
9	MAN.FA...	Ehagator	Gilbert	H	Activ	2009-12-18
10	MAN.FA...	Eichenauer	Roy	b	Activ	1987-04-05
11	MAN.FA...	Ellerkamp	May	t	Activ	2005-09-09
12	MAN.FA...	Ellerkamp	Vito	V	Activ	1998-09-10
13	MAN.FA...	Francisque	Dylan	S	Activ	2009-12-18
14	MAN.FA...	Gondik	Hilma	s	Activ	1983-11-12
15	MAN.FA...	Grandstaff	Marc	G	Activ	2009-12-18
16	MAN.FA...	Gulstorf	Vaughn	P	Activ	2008-01-04
17	MAN.FA...	JOHN	REBA	K	Activ	1968-08-29
18	MAN.FA...	JONES	WILLIAM	T	Activ	1979-04-11
19	MAN.FA...	Koib	Reed	B	Activ	2009-12-18
20	MAN.FA...	Krawets	Perry	r	Activ	1996-05-01
21	MAN.FA...	Kruzel	Horace	l	Activ	1985-07-04
22	MAN.FA...	Landerb...	Violeta	h	Activ	1995-09-02
23	MAN.FA...	Lettieri	Nia	x	Activ	1999-12-04
24	MAN.FA...	LUTZ	JENNIFER	K	Activ	1968-08-29
25	MAN.FA...	Mafija	Lindsay	C	Activ	1993-12-01
26	MAN.FA...	Marabos	Madelyn	o	Activ	1994-02-11
27	MAN.FA...	Messler	Cedrick	B	Activ	1985-07-05
28	MAN.FA...	Mohorovich	Korey	v	Activ	2007-01-03
29	MAN.FA...	Naibler	Dwain	h	Activ	1988-10-10
30	MAN.FA...	Paolucci	Taylor	r	Activ	1989-07-02

DEPTNO	DEPTNAME	LASTNAME	FIRSTNAME	MIDINIT	STATUS_CODE	ACTVE_SINCE
1	MAN.FA...	Borell	Rueben	K	Activ	2009-12-18
2	MAN.FA...	Koib	Reed	B	Activ	2009-12-18
3	MAN.FA...	Ehagator	Gilbert	H	Activ	2009-12-18
4	MAN.FA...	Francisque	Dylan	S	Activ	2009-12-18
5	MAN.FA...	Grandstaff	Marc	G	Activ	2009-12-18
6	MAN.FA...	STERN	IRVING	F	Activ	1973-09-14
7	MAN.FA...	ADAMSON	BRUCE		Activ	1972-02-12
8	MAN.FA...	PIANKA	ELEZMETH	R	Activ	1977-10-11
9	MAN.FA...	YOSHIMU...	MASATO...	J	Activ	1978-09-15
10	MAN.FA...	SCOUTTEN	MARILYN	S	Activ	1973-07-07
11	MAN.FA...	WALKER	JAMES	H	Activ	1974-07-26
12	MAN.FA...	BROWN	DAVID		Activ	1966-03-03
13	MAN.FA...	JONES	WILLIAM	T	Activ	1979-04-11
14	MAN.FA...	LUTZ	JENNIFER	K	Activ	1968-08-29
15	MAN.FA...	YAMANO...	KIHOGE		Activ	1978-09-15
16	MAN.FA...	JOHN	REBA	K	Activ	1968-08-29
17	MAN.FA...	Deliberato	Carolyn	g	Activ	2006-01-02
18	MAN.FA...	Paolucci	Taylor	r	Activ	1989-07-02
19	MAN.FA...	Gondik	Hilma	s	Activ	1983-11-12
20	MAN.FA...	Marabos	Madelyn	o	Activ	1994-02-11
21	MAN.FA...	Lettieri	Nia	x	Activ	1999-12-04
22	MAN.FA...	Ellerkamp	May	t	Activ	2005-09-09
23	MAN.FA...	Eichenauer	Roy	b	Activ	1987-04-05
24	MAN.FA...	Colvard	Jenae	h	Activ	1980-02-07
25	MAN.FA...	Truj	Su	o	Activ	2009-06-04
26	MAN.FA...	Landerb...	Violeta	h	Activ	1995-09-02
27	MAN.FA...	Dobyns	Annemarie	R	Activ	1991-07-05
28	MAN.FA...	Mohorovich	Korey	v	Activ	2007-01-03
29	MAN.FA...	Krawets	Perry	r	Activ	1996-05-01
30	MAN.FA...	Bax	Steve	Q	Activ	1994-07-09

RDA - Data Source Explorer

In this example a single stored procedure can process either column names or column positions

Dynamic Column Ordering Considerations

- **Each unique ORDER BY will result in a separate ODP**
 - Each ODP may be reusable
 - DB2 will attempt to manage different ODPs with same cursor name
 - Consider unique statement and cursor names for frequently used transactions
- **Indexing strategy**
 - Primary table
 - Local selection columns, ORDER BY columns, JOIN columns
 - WORKDEPT, LASTNAME, FIRSTNAME, MIDINIT, EMPNO, EMP_MAST_PK
 - Focus on columns used most for ordering
 - Secondary tables
 - JOIN columns, result set columns (Index Only Access-IOA)
 - RI satisfies join requirement
 - IOA for highly used lookup or validation tables
- **SQE Plan Cache or Database Monitor can provide statistics**
 - Database Analysis tools can help identify most used transactions

Each unique dynamic SQL statement will create an ODP. These ODPs may become reusable which is good. DB2 for i will manage different SQL statements for the same cursor until the cursor threshold is met. Then all ODPs are deleted. The application can use logic to assign different statement and cursor names for each unique order by.

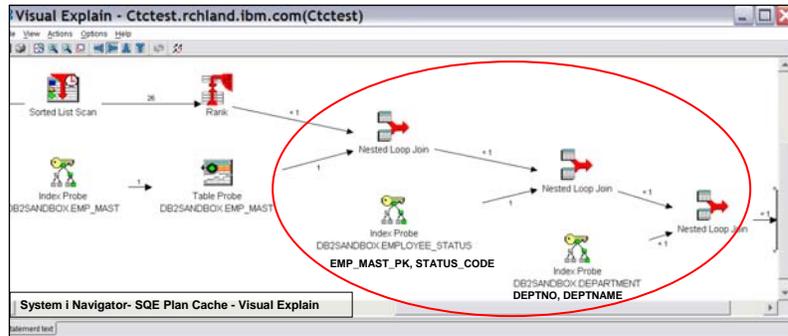
Create indexes for the CTE table focusing on equal selection columns, followed by the ordering columns and including the join columns.

Create indexes for the secondary tables with the join column first, followed by columns used in the result set (if only 1 or 2).

Database analysis tools from IBM or Centerfield can make life simpler for the database plumber.

Data Centric Programming Tip # 7 Exploit SQL Index Only Access

- **Avoid or minimize I/O by only reading data from the index**
 - No need to probe the underlying table
- **All columns used in the query must be represented in the index**
- **Used with index probe or index scan**
 - Used with radix and Encoded Vector indexes
- **Only available to SQL**
 - RLA must always access the table object



Typically the use of an index operation will also include a Table Probe operation to provide access to any columns needed to satisfy the query that cannot be found as index keys. If all of the columns necessary to satisfy the query request for a table can be found as keys of an index, then the Table Probe is not required and the query uses Index Only Access. Avoiding the Table Probe can be an important savings for a query. The I/O associated with a Table Probe is typically the more expensive synchronous random I/O.

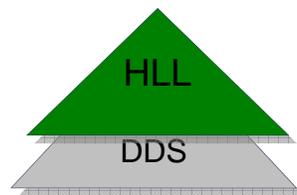
Agenda

- Keys to the Kingdom
- Exploiting the Primary Key
- Pagination with ROW_NUMBER
- Column Ordering
- **Summary**

The State of the Database on IBM i

▪ In the beginning

- Data Description Specifications (DDS)
 - Create external definitions for files (disk, display, printer)
 - Limited capabilities
- Application Centric
 - Majority of work done in High Level Languages
 - CLP, RPG, etc.

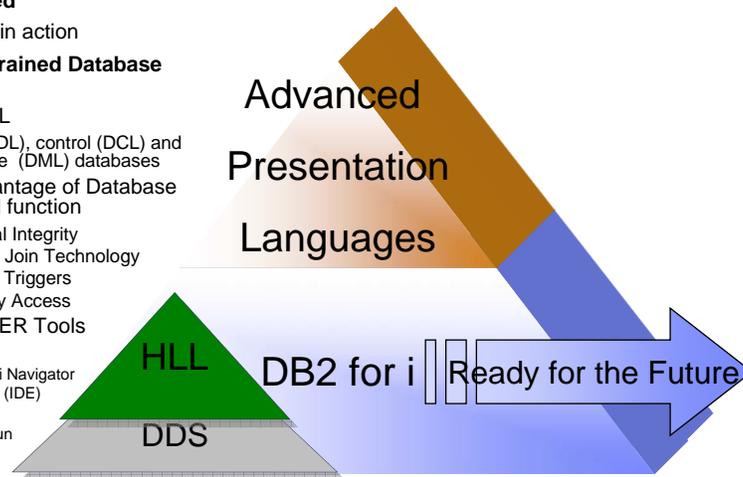


The IBM i platform has always had a solid foundation for building applications. This began with the introduction of DDS as a tool for externally describing the fields that made up a file. This included disk files, display files (green screen) and printer files. Although making life easier for the programmer DDS never was intended as a substitute for programming. Thus many of the business rules were written as part of the HLL application.

The Control Language (CL) provided some commands for expanding the capabilities of a DDS defined database (ADDPFCST, ADDPFTRG, etc). However, since most legacy customers did not have a DBA, these capabilities were never truly exploited, or, if used at all, were used as Band-Aids in a poorly designed database environment.

The State of the Database on IBM i

- **Another definition of unconstrained**
 - Not limited in action
- **The Unconstrained Database Developer**
 - Utilizes SQL
 - Define (DDL), control (DCL) and manipulate (DML) databases
 - Takes advantage of Database feature and function
 - Referential Integrity
 - Advanced Join Technology
 - Instead of Triggers
 - Index Only Access
 - Uses POWER Tools
 - IBM
 - System i Navigator
 - Rational (IDE)
 - Other
 - HomeRun
 - Xcase



All code examples and results were created using SQL and the Rational and System i Navigator tools. Not one line of HLL code was utilized during this process.

Today, the DDS language is no longer enhanced. All database investment is being made within the SQL space. SQL provides an industry standard for defining, controlling and manipulating databases. In addition SQL provides a programming language for the creation of procedures (programs), triggers, constraints, etc. Using this language, the database developer can push more of the business rules and logic down to the database layer, thus facilitating more reuse as the application languages continue to evolve.

<click> This is the strategic development direction for the IBM i platform.

Where to go from here

- **Web sites**
 - Information Center <http://publib.boulder.ibm.com/series/>
 - IBM DB2 for i <http://ibm.com/systems/i/software/db2/>
- **For those who like to read:**
 - DB2 SQL PL (Second Edition)
 - Database Design for Mere Mortals (Michael J. Hernandez)
 - Redbooks
 - Preparing for and Tuning the SQL Query Engine on DB2 for I
 - Modernizing IBM DB2 for i Application Data Access - A Roadmap Cornerstone
 - Stored Procedures, Triggers, and User-Defined Functions on DB2 for I
 - White Papers
 - Improving DB2 for i SQL procedure performance
 - Indexing and statistics strategies for DB2 for i
 - Many more on the IBM DB2 for i website
- **For those who like formal training:**
 - DB2 for i SQL Performance Workshop
 - To enroll select Support-Education from the IBM DB2 for i website
 - Coming soon: DB2 for i Data Access Modernization Workshop



Trademarks

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml: AS/400, DBE, e-business logo, ESCO, eServer, FICON, IBM, IBM Logo, iSeries, MVS, OS/390, pSeries, RS/6000, S/390, VM/ESA, VSE/ESA, Websphere, xSeries, z/OS, zSeries, z/VM

The following are trademarks or registered trademarks of other companies

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation
Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
LINUX is a registered trademark of Linux Torvalds
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
Intel is a registered trademark of Intel Corporation
* All other products may be trademarks or registered trademarks of their respective companies.

NOTES:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use.

The information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.