IBM Information On Demand 2006

What's new with the Optimizer in DB2 for z/OS V8?

Terry Purcell & Brian Baggett, IBM Silicon Valley Lab

TAKE BACK CONTROL

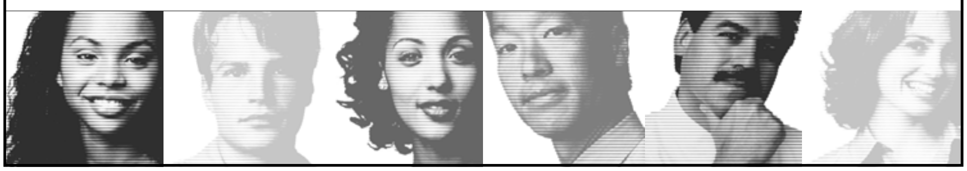
IBM INFORMATION ON DEMAND 2006
October 15 – 20, 2006
Anaheim Convention Center
Anaheim, Cali

Agenda

1. Statistics Collection
2. Predicate Processing
3. Query Tuning Tools
4. Indexing, Partitioning, and Clustering
5. Misc Optimization Enhancements
6. Materialized Query Tables
7. Complex Join

Statistics Collection

TAKE BACK CONTROL



4

Enhanced statistics for non-indexed columns

- RUNSTATS enhancements
  - ►New COLGROUP keyword to collect correlation and/or distribution statistics on any column(s) in a table
    - –Indexed and/or non-indexed columns
  - ►For columns or column groups that are not the leading columns of an index
    - ►Frequency distributions
      - ►Combination of COLGROUP & FREQVAL keywords
      - ►LEAST frequently, MOST frequently, or BOTH
    - ►Correlation cardinality
      - ►COLGROUP keyword

Non-Uniform Distribution - Example

5

- Run a count to demonstrate data distribution
  - Evenly distributed or skewed?

```
SELECT COUNTRY, COUNT(*)
FROM    PERSON
GROUP BY COUNTRY
ORDER BY 2 DESC
```

```
COD_COUNTRY
---------+---------+------
AT         4418514 (90.6%)
            160065
DE          140247
CZ           28996
HU           19654
SI           15499
IT           15461
HR           14113
CH           12204
SK           8078
... Not all displayed
```

Even distribution 1/227 = 0.44%
Actual distribution 'AT' = 90.6%

Single Column Frequency – Indexed

6

- How to collect
  - V7 RUNSTATS only collects single column frequencies on the leading column of index

INDEX (I1) columns (C1,C2,C3)

Collect more than top 10:
RUNSTATS INDEX (I1 FREQVAL NUMCOLS(1) COUNT(20))

Eliminate existing frequencies:
RUNSTATS INDEX (I1 FREQVAL NUMCOLS(1) COUNT(0))

7

Single Column Frequency – Non-indexed

- How to collect
  - V8 RUNSTATS allows collection on almost any column

  RUNSTATS TABLESPACE DB1.TS1
  TABLE (T1) COLUMNS(C1,C2,C3)
  COLGROUP (C1) FREQVAL COUNT(1) MOST
  COLGROUP (C2) FREQVAL COUNT(10) LEAST
  COLGROUP (C3) FREQVAL COUNT(20) BOTH

  - Eliminate existing frequencies:

  COLGROUP (C3) FREQVAL COUNT(0) MOST

8

Multi-column Frequency – Indexed

- How to collect
  - V7 RUNSTATS only collects on leading concatenated column of index
  - RUNSTATS does NOT collect multi-column frequencies by default.
  - Must be explicitly requested.

  INDEX (I1) columns (C1,C2,C3)

  - Collect top 15 values for column group (C1,C2)

  RUNSTATS INDEX (I1 FREQVAL NUMCOLS(2) COUNT(15))

  - Eliminate frequencies on column group (C1,C2):

  RUNSTATS INDEX (I1 FREQVAL NUMCOLS(2) COUNT(0))

Multi-column Frequency – Non-indexed

9

- DB2 V8 allows collection of multi-column frequencies on almost any column group

  – Examples

```
RUNSTATS TABLESPACE DB1.TS1
TABLE (T1) COLUMN(C1,C2,C3)
COLGROUP(C1,C3) FREQVAL COUNT(10) MOST
COLGROUP(C2,C3) FREQVAL COUNT(1) LEAST
```

  – Eliminate frequencies on column group (C1,C3):

```
COLGROUP (C1,C3) FREQVAL COUNT(0)
```


Correlation Statistics

10

```
SELECT  COUNT(*)
FROM    USA
WHERE   CITY = 'ORLANDO'
AND     ZIPCODE = '32821'
```

1,000 cities
Filter factor 0.1%

10,000 Zipcodes
Filter factor 0.01%

- Filtering applied by this WHERE clause is
  - 0.1% of 0.01% = 0.00001% without correlation statistics
  - 0.01% with correlation statistics
  - Assume 300 million table rows
    - 30 rows Vs 30,000 rows

Once I know the zipcode, does the city provide any more filtering?

11

Multi-column Cardinality – Indexed

- How to collect
  - V7 RUNSTATS only collects KEYCARD on leading column of index
  - By default, RUNSTATS only collects FIRST/FULLKEYCARDF

INDEX I1 (C1,C2,C3,C4)
RUNSTATS INDEX(I1 KEYCARD)

  - MCARD on leading concatenated column groups:
  - MCARD(C1,C2), MCARD(C1,C2,C3)

12

Multi-column Cardinality – Non-indexed

- DB2 V8 allows collection of MCARD on any column group

RUNSTATS TABLESPACE DB1.TS1
TABLE(T1) COLUMN(C1,C2,C3,C4)
COLGROUP(C1,C4)
COLGROUP(C2,C3,C4)

Specifying COLGROUP with multiple columns collects multi-column cardinality on the group.

Predicate Processing

TAKE BACK CONTROL

14

Predicate Sargability (stage one)

Column

Op
=
<
<=
>=
>
Between
...

Literal
Column
Exp.

Up through V7

Same type, length, ccsid | **Same type, length, ccsid**

In V8

All except ...

Dec(p,s), where p > 15	**Float**
string types	**Date, Time or Timestamp**
Unicode or same CCSID	**Unicode or same CCSID**

☆ PK12389 – Datatype mismatch indexability added to compatibility mode

Datatype/Length mismatch – V8

15

- Unmatched data type: numeric types

```
Employee ( Name    character (20),
           Salary  decimal (12,2),
           deptID  character (3) );
```

```
SELECT * FROM employee
   WHERE salary > :hv_float ;
```

Prior to V8

- Stage-2
- RSCAN

V8

- Stage-1
- Sargable and indexable

Datatype/Length mismatch – V8

16

- Unmatched types: string types

```
SELECT * FROM employee
   WHERE deptID = '6S5A' ;
```

or   char(3)

```
SELECT * FROM employee
   WHERE deptID = :CHAR4HV ;
```

Prior to V8

- **Stage-2**
- **RSCAN**

V8

- **Stage-1**
- **Sargable and indexable**

Join Dependent Indexability in V8

- Unknown join sequence: Column-expression
  - Without datatype/length match

```
SELECT e1.*
   FROM emp AS e1, emp AS e2, dept
   WHERE e1.deptID = dept.id   AND
         e1.salary > e2.salary * 1.10 ;
```

dec(12,2) → **e1.salary**

V7 prior to PQ54042
- ☞**Stage-2 predicate**

V7 after PQ54042
- ☞**If e2 is inner table**
  - **Stage-2 predicate**
- ☞**If e1 is inner table**
  - **Stage-1, indexable on e1.salary**

☆
- Move expression to the other side for an alternate join sequence
  - e1.salary / 1.10 > e2.salary

Join Dependent Indexability in V8

- BETWEEN as join predicate

```
SELECT emp.*
   FROM emp, salRange AS s, dept
   WHERE emp.level = s.level  AND
         emp.salary BETWEEN s.low AND s.mid;
```

dec(12,2) → **emp.salary**

V7 prior to PQ54042
- ☞**Stage-2 predicate**

V7 after PQ54042
- ☞**If salRange is inner table**
  - **Stage-2 predicate**
- ☞**If emp is inner table**
  - **Stage-1, indexable on e1.salary**

☆
- Change BETWEEN (on join predicates only) to >= AND <=
  - For indexability in either join sequence
  - emp.salary >= s.low AND emp.salary <= s.mid

IBM

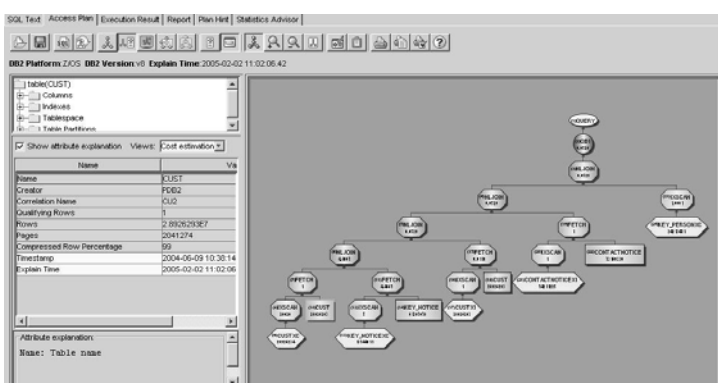
Query Tuning Tools

TAKE BACK CONTROL

V8 Visual Explain Enhancements

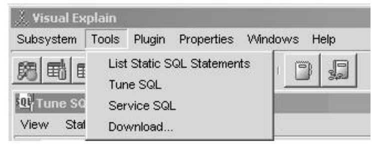
20

- Significant improvements in V8:
  - ►More statistical details for each node in the access path graph
  - ►Statistics Advisor
  - ►Easier collection of information to send for PMRs (V7 also)



21

Visual Explain Input Options



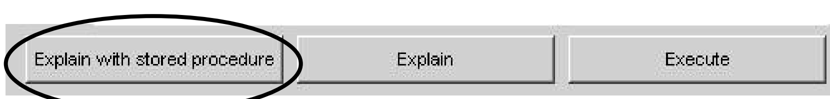
- Tune SQL option
  - Type or Cut & Paste
  - Retrieved previously saved SQL
- Static SQL
  - By plan/package
    - Applying filters
      - Cost, object or access path

- Dynamic Statement Cache
  - By any statement cache filter

22

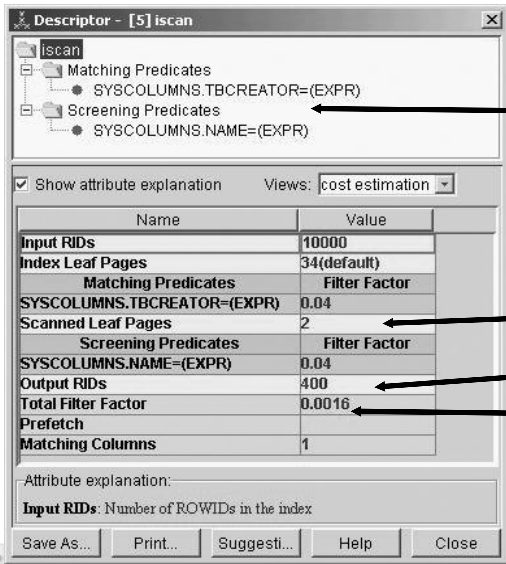
Explain with Stored Procedure

- Requires stored procedure DSN8EXP
  - See APAR PQ90022
- Explain SQL against objects you do not have authority to execute queries against.
  - Eg. Developer has SQL performance responsibilities for application, but does not have SELECT / INSERT / UPDATE / DELETE access to the production objects.
  - Use stored procedure to execute the explain.
- Do not require access to the objects, but do require authority to execute the stored procedure.



VE Sample Output – Index Details

23


Descriptor - [5] iscan

iscan
Matching Predicates
SYSCOLUMNS.TBCREATOR=(EXPR)
Screening Predicates
SYSCOLUMNS.NAME=(EXPR)

Show attribute explanation   Views: cost estimation

Name	Value
Input RIDs	10000
Index Leaf Pages	34(default)
Matching Predicates	Filter Factor
SYSCOLUMNS.TBCREATOR=(EXPR)	0.04
Scanned Leaf Pages	2
Screening Predicates	Filter Factor
SYSCOLUMNS.NAME=(EXPR)	0.04
Output RIDs	400
Total Filter Factor	0.0016
Prefetch	
Matching Columns	1

Attribute explanation:
Input RIDs: Number of ROWIDs in the index

Save As... Print... Suggesti... Help Close

Predicate info
- Matching
- Screening

Scanned leaf pages

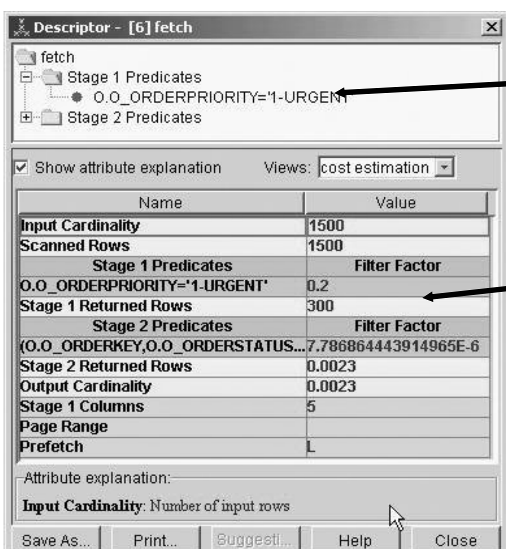
Output RIDs

Total filter factor

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

VE Sample Output – Fetch Details

24


Descriptor - [6] fetch

fetch
Stage 1 Predicates
O.O_ORDERPRIORITY='1-URGENT
Stage 2 Predicates

Show attribute explanation   Views: cost estimation

Name	Value
Input Cardinality	1500
Scanned Rows	1500
Stage 1 Predicates	Filter Factor
O.O_ORDERPRIORITY='1-URGENT'	0.2
Stage 1 Returned Rows	300
Stage 2 Predicates	Filter Factor
(O.O_ORDERKEY,O.O_ORDERSTATUS...	7.786864443914965E-6
Stage 2 Returned Rows	0.0023
Output Cardinality	0.0023
Stage 1 Columns	5
Page Range	
Prefetch	L

Attribute explanation:
Input Cardinality: Number of input rows

Save As... Print... Suggesti... Help Close

Predicates
- Stage 1
- Stage 2

Access information
- Stage 1 rows
- Stage 2 rows
- Result rows
- Page range flag
- Prefetch flag

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

V8 VE – New information Overview

- Estimated number of records
  - Know single table qualified row estimates
    - What tables have worst estimates?
    - Can affect join sequence selected.
  - Join size estimate
    - Over estimation early in query can cause problems later with join sequence, join method

V8 VE – New information Overview

- Predicate information
  - Stage of predicate application
    - Matching
    - Screening
    - Non-indexed stage 1
    - Non-indexed stage 2
  - Filter factor estimation
    - Single predicate filter factor
    - Applicable bounds also apply

V8 VE – New information Overview

27

- Limited partition scan information
  - What partitions are scanned?
  - How many page ranges?
  - How many partitions in specific range?
- Sort information
  - Sort key columns/length
  - Sort record length
  - Estimated sort records
  - Estimated pages scanned
- Parallelism information
  - Mode of parallelism?
  - How many degrees?
  - Divided on page range or key range?
  - What range is each parallel task accessing?
  - What partitions is each task accessing?

Service SQL

28

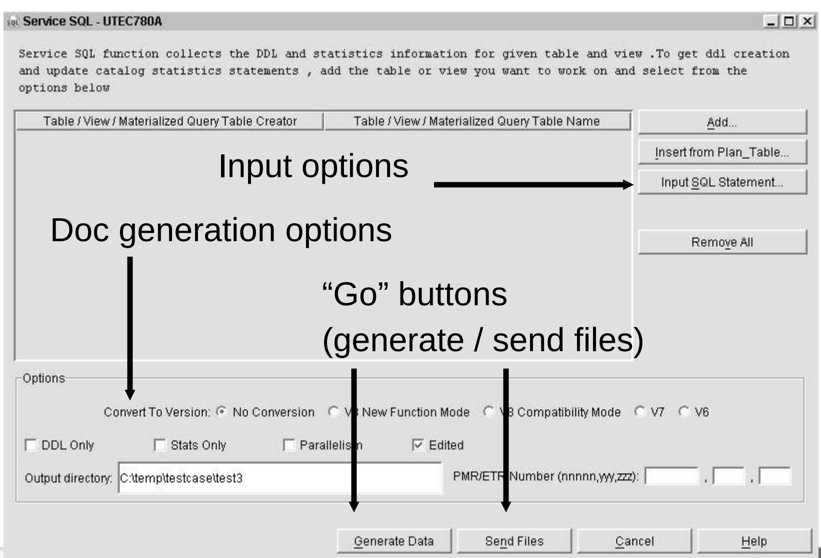
- Provides problem recreation information to service team
  - Uses SQL statement or PLAN_TABLE as input
  - SQL statement (if SQL used as input)
  - DDL
  - Catalog statistics
  - Zparms (if DSNWZP stored procedure available)
  - Environment specific information
    - CPU speed
    - Bufferpool, ridpool, sortpool sizes
    - Number of processors

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

Service SQL (cont)

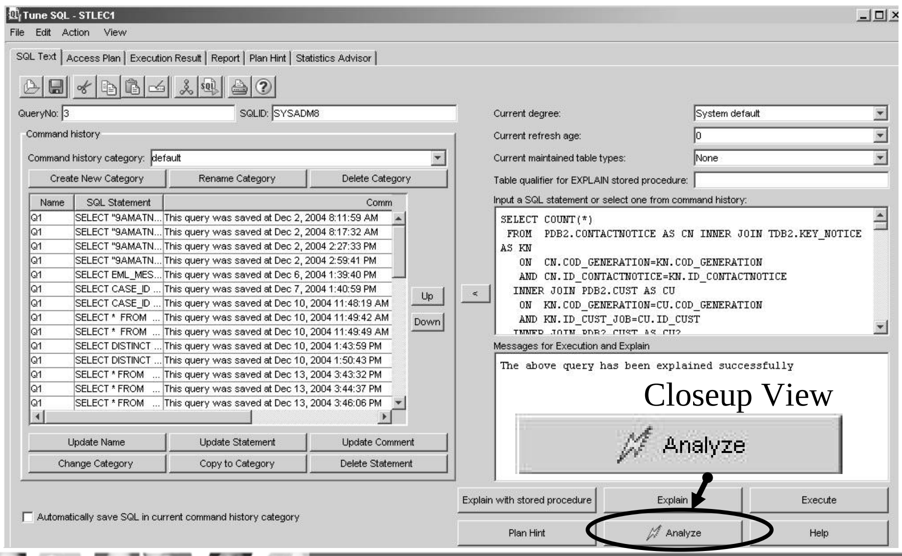
- Eliminate common problems / frustrations
  - Input pmr number
  - SQL statement
  - SQL explained
  - Click to generate documentation
    - File names based on pmr
  - Click to FTP documentation
    - Appropriate FTP settings already set

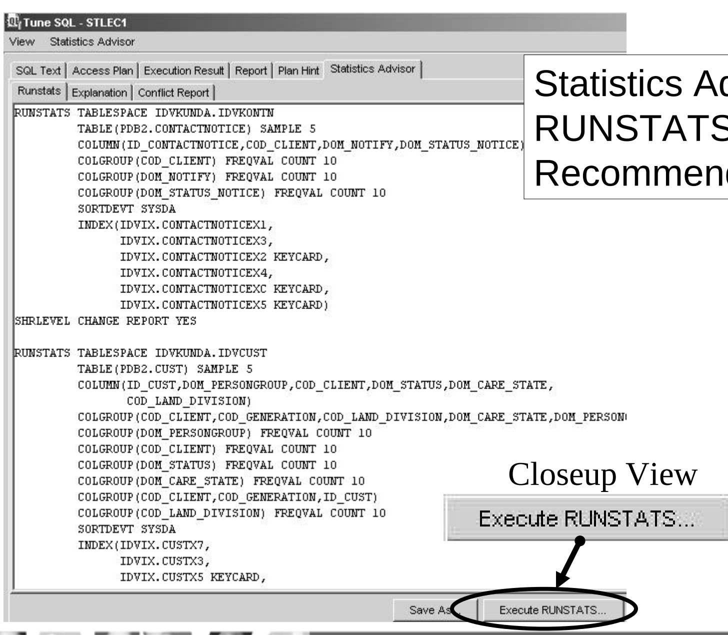
Service SQL main screen



IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

Automating the SQL Tuning Process

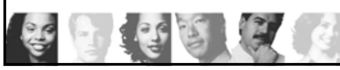




Statistics Advisor – RUNSTATS Recommendations

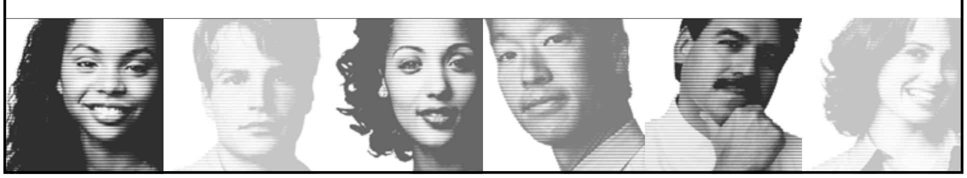
Statistics Advisor

- Automated statistics determination
  - Often queries have inefficient OR unstable performance due to lack of statistics
  - SA automates the analysis of statistics required for an SQL statement
- Goal
  - Automate SOLUTION to many common SQL performance problems
  - Solve SQL performance problems quickly and easily



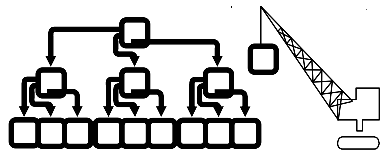
Indexing Partitioning & Clustering

TAKE BACK CONTROL



35

Index Improvements

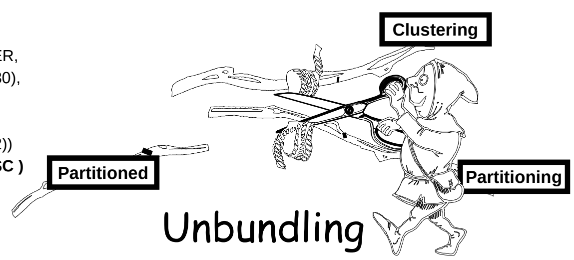


- Variable length index keys
- Index-only access for varchar data
- Predicates indexable for unlike types
- Backward Index Scan
- Partitioning separate from clustering
- Data-partitioned secondary indexes (DPSI)
- Add column to index
- Alter clustering index ☆

36

Table Controlled Partitioning

```
CREATE TABLE CUSTOMER (
      ACCOUNT_NUM          INTEGER,
      CUST_LAST_NM         CHAR(30),
      . . .
      LAST_ACTIVITY_DT     DATE,
      STATE_CD             CHAR(2))
   PARTITION BY ( ACCOUNT_NUM   ASC )
   ( PARTITION 1   ENDING AT (199),
     PARTITION 2   ENDING AT (299),
     PARTITION 3   ENDING AT (399),
     PARTITION 4   ENDING AT (499)  ) ;
```



No indexes are required for partitioning!!

TB	101 102 103 104 105 106	201 202 203 204 205 206	301 302 303 304 305 306	401 402 403 404 405 406

Partitioned table

37

Version 8 classification of indexes

- An index may / may not be correlated with the partition**ing** columns of the table
  - Partitioning index (PI)
  - Secondary index
- An index may / may not be physically partition**ed**
  - Partitioned
  - Non-partitioned
- Clustering index:
  - Any index may be the clustering index
  - The clustering index can be non-unique

38

Partition**ing** indexes

- A partition**ing** index
  - Same leftmost columns as the columns which partition the table
  - These columns have the same collating sequence (ASC / DESC)

```
CREATE TABLE CUSTOMER (
        ACCOUNT_NUM             INTEGER,
        LAST ACTIVITY_DT        CHAR(3),
        CCODE                   CHAR(2),
        . . .
      )
        PARTITION BY (ACCOUNT_NUM         ASC)
        ...
CREATE . . . INDEX  part_ix_1  ON CUSTOMER (
        ACCOUNT_NUM                      ASC)
```

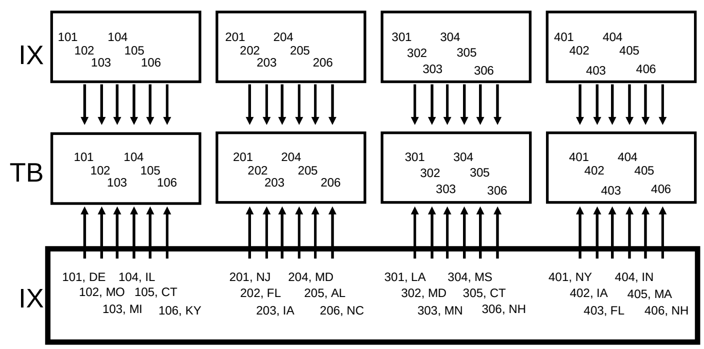
IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

Partitioning indexes -2

39

A partitioning index has the same leftmost columns, in the same collating sequence, as the columns which partition the table

Partitioning index part_IX_1 (ACCOUNT_NUM ASC)




```
CREATE TABLE CUSTOMER (
    ACCOUNT_NUM INTEGER,
    CUST_LAST_NMCHAR(30),
    . . .
        )
    PARTITION BY (
        ACCOUNT_NUM ASC)
    . . .
```

Partitioning index part_IX_2 (ACCOUNT_NUM ASC, STATE_CD)

Secondary indexes

40

A secondary index is any index which is **not** a partitioning index

Secondary Index SI_1 on LAST_ACTIVITY_DT

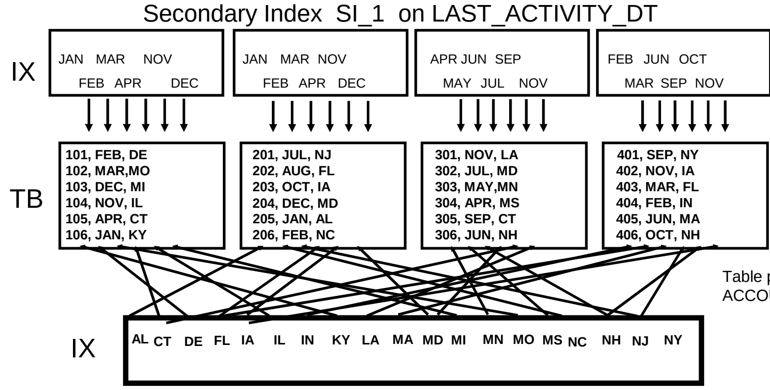


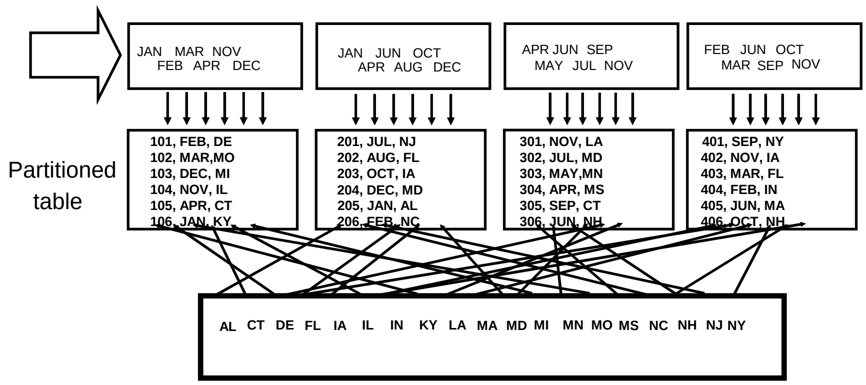
Table partitioned by ACCOUNT_NUM

Secondary Index SI_2 on STATE_CD

41

Secondary indexes (Partition**ed** & Non-Partition**ed**)

Data Partitioned Secondary Index (DPSI) -- data_part_si_1

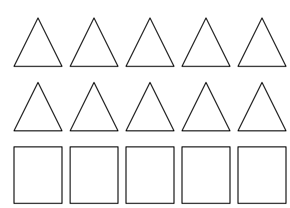


Non-Partitioned Secondary Index (NPSI) -- non_part_si_2

42

Data Partitioned Secondary Indexes

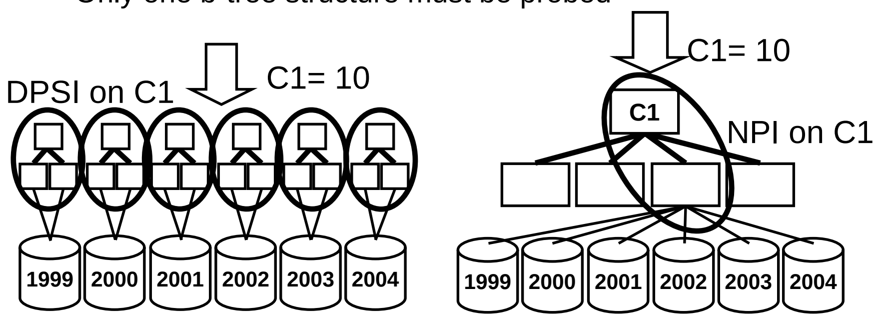
- Benefits include partition independence:
  - More efficient utility processing, no BUILD2
  - Higher availability
  - Streamline partition level operations
  - Potential for lower data sharing overhead
- Potential impact to query performance
  - Partition key is not specified
  - Many partitions to search
  - Not allowed for unique index



Index Access - DPSI vs NPI

43

- Access to a secondary index without specifying a range de-limiting predicate results in:
  - ►For DPSI
    - –All b-tree structures (up to 4096) must be probed
  - ►For NPI
    - –Only one b-tree structure must be probed

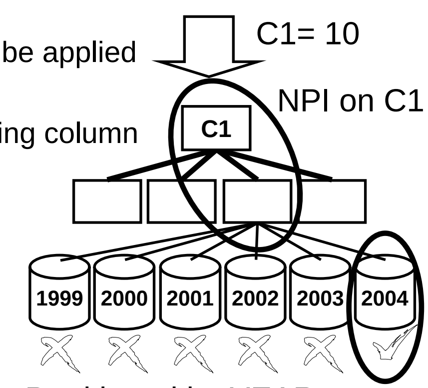


Page Range Screening - NPI

44

- Page Range Screening can be applied
  - ►before data access on a NPI to limit the partitions accessed
    - –if a predicate exists that can be applied
  - ►Similar to index screening
    - –Without requiring the screening column to be indexed

SELECT cols
FROM T1
WHERE C1 = 10
AND YEAR = 2004



Partitioned by YEAR

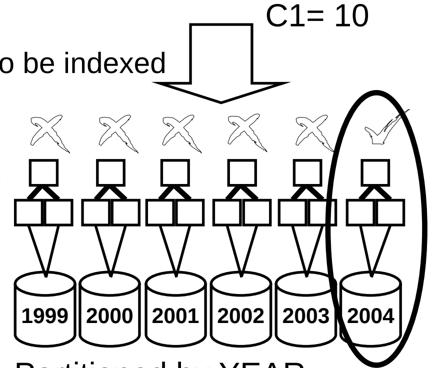
Page Range Screening - DPSI (V8)

45

- Page Range Screening can be applied
  - ►When index access occurs
    - –if a predicate exists that can be applied
  - ►Similar to index matching
    - –Without requiring the column to be indexed
  - ►Effect is 2 matching columns
    - –1 from index, 1 from partition
  - ►Can also function independently

WHERE C1 = 10
AND YEAR = 2004

DPSI on C1
C1= 10



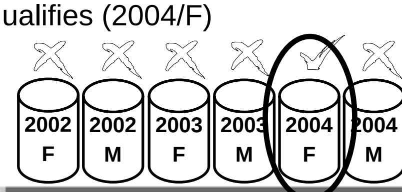
Partitioned by YEAR

Page Range Screening - V7 vs V8

46

- Partitions qualified after Page Range Screening
  - ►In V7
    - –Host variables/parameter markers require REOPT(VARS)
    - –Only the leading limit key is used
      - –2 partitions qualify (2004/F & 2004/M)
      - –Remaining rows disqualified after data access
  - ►In V8
    - –Host var/parameter markers evaluated at run time (no REOPT)
    - –All limit keys can be utilized
      - –1 partition qualifies (2004/F)

WHERE YEAR = 2004
AND GENDER = 'F'



Partitioned by YEAR & GENDER

Join Predicate Page Range Screening

- Join predicates are not eligible for Page Range Screening

```
FROM T1 JOIN T2
ON   T1.C1 = T2.PART_COL
AND T1.C2 = T2.DPSI_COL
```

Not eligible

- Predicate transitive closure predicates are eligible for Page Range Screening

```
FROM T1 JOIN T2
ON   T1.C1 = T2.PART_COL
AND T1.C2 = T2.DPSI_COL
WHERE T1.C1 = ?
```

Eligible

Because DB2 also generates this

```
AND T2.PART_COL  = ?
```


Clustering

- Clustering increases the number of “interesting rows per page”
  - Decreasing the number of I/Os required
  - Most frequent access may not benefit from table clustering on that column
    - What if customers update their accounts on-line
    - Do different customers logon throughout the day in ACCT_ID order?
    - Does clustering on ACCT_ID help?
  - If customer accounts are accumulated in a file, the file is sorted by ACCT_ID and applied in night batch process
    - Clustering by ACCT_ID will benefit batch processing

49

Clustering (cont.)

- Clustering makes sequential access efficient
  - Follow table relationships
    - Child of one to many relationship may benefit from clustering on parent key
    - Cascading relationships can benefit from all tables in same sequence
  - Common range searches benefit from clustering
    - Search by year, month, quarter, etc.
    - Search by geographical area

50

Clustering Enhancements

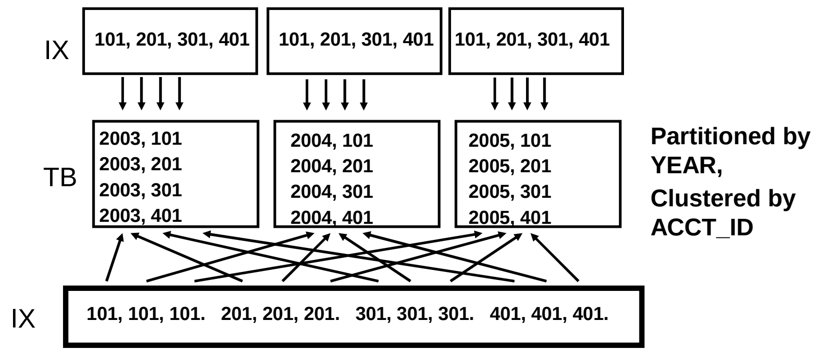
- V8 Provides
  - Altering the clustering index
    - Without drop and recreate
  - Index-only access for VARCHAR columns
    - Consider index-only access to reduce random I/O
      - Poorly clustered index
      - Good clustering, but data scattered after index screening
      - Adding additional columns does not increase index levels

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

Partitioning/Clustering Considerations

- Index Clusterratio may differ depending on DPSI or NPI

DPSI on ACCT_ID – Clustering perfectly aligned with partitions

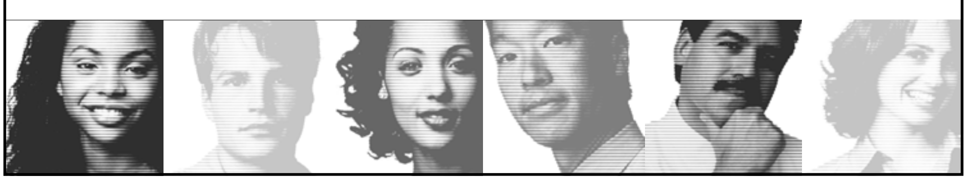


NPI on ACCT_ID – Clustering not aligned, keys cross partitions

IBM

Misc. Optimization Enhancements

TAKE BACK CONTROL



53

SQL statements up to 2MB

DB2 UDB

- SQL statements can be up to 2M bytes in length
- Parse tree has been completely rearchitected to reduce SQL too complex SQLCODE -101 scenarios due to SQL statement length
- Important for SQL Procedure Language applications
- Needed for generated SQL, long names, increased numbers of partitions, …

54

INSERT within SELECT

DB2 UDB

- Elegant technique for retrieving values created / modified by DB2 during INSERT
  - ►identity columns, sequence values
  - ►user-defined defaults, expressions
  - ►columns modified by triggers
  - ►ROWIDs, CURRENT TIMESTAMP, …
  - ►INSERT with return or SELECT from INSERT

EXAMPLE:

```
SELECT C1, C2, C3, C4, C5 FROM
  INSERT (C1, C5) INTO T1
      VALUES('ABC', CURRENT DATE);
```

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

Common Table Expressions

55

```
WITH DTOTAL (DEPTNO, TOTALPAY) AS
(SELECT DEPTNO, SUM(SALARY)
 FROM DSN8810.EMP
 GROUP BY DEPTNO)
 SELECT DEPTNO FROM DTOTAL
 WHERE TOTALPAY = (SELECT MAX(TOTALPAY)
                          FROM DTOTAL)
```

Table expression referenced twice

- Pros:
  - Avoids requirement to code or execute nested table expression many times.
  - Closer to DB2 family compatibility
- Cons:
  - Avoidance of repeat execution requires materialization

Recursive SQL

56

```
WITH TEMP (N) AS
   (SELECT 0
    FROM SYSIBM.SYSDUMMY1
    UNION ALL
    SELECT N + 1
    FROM TEMP
    WHERE N < 5)
SELECT *
FROM TEMP;
```

Common Table Expression

Recursive Reference

N
0
1
2
3
4
5

- Pros:
  - Recursive processing now in DB2 z/OS
  - Closer to DB2 family compatibility
- Cons:
  - VALUES clause not supported in common table expression

Fullselect in Select List

57

```
SELECT D.*,
    , (SELECT COUNT(*) FROM EMP E
       WHERE D.DEPTNO = E.WORKDEPT) AS NUMEMP
    , (SELECT SUM(E.SALARY) FROM EMP E
       WHERE D.DEPTNO = E.WORKDEPT) AS SUMSAL
    ,  (SELECT COUNT(*) FROM PROJ P
        WHERE D.DEPTNO = P.DEPTNO) AS NUMPROJ
FROM DEPTNO D
```

- Pros:
  - Materialization avoidance
- Cons:
  - Each subquery can only return a single value

Scalar fullselect - 2

58

```
UPDATE NEW_PARTPRICE N
   SET PRICE =
      CASE
        WHEN(  (SELECT ONHAND#
                    FROM INVENTORY
                    WHERE PART=N.PART)  < 7  )
             THEN 1.1 * PRICE
        WHEN(  (SELECT ONHAND#
                  FROM INVENTORY
                  WHERE PART=N.PART) > 20  )
             THEN .8 * PRICE
        ELSE PRICE
      END;
```

Other SQL Improvements and DB2 Family Compatibility

59

- GROUP BY expression
  - SELECT A+B, … FROM T … **GROUP BY A+B**
- SELECT INTO statement with ORDER BY ☆
  - SELECT … INTO … **ORDER BY A** FETCH FIRST ROW ONLY
- Qualified column names on UPDATE/INSERT clause
  - UPDATE T1 SET **T1.**COL1...
  - INSERT **T1.**COL1 INTO T1 VALUES...
- Multiple DISTINCT clauses
  - SELECT COUNT(**DISTINCT**(A1)), COUNT(**DISTINCT**(A2)) …

VOLATILE Table Support

60

- Encourages index access for tables that have unpredictable cardinality
- Significant performance improvement for some SAP applications

CREATE TABLE XYZ ...... **VOLATILE;**

ALTER TABLE XYZ ...... **VOLATILE;**

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

REOPT(ONCE) BIND Option

61

- Controls when DB2 builds access path for dynamic SQL
- Default, access path is calculated at PREPARE.
  - REOPT(ONCE)
    - Defers access path selection until OPEN
    - Values of host variables on OPEN used to calculate access path
    - Resulting access path cached in global prepare cache

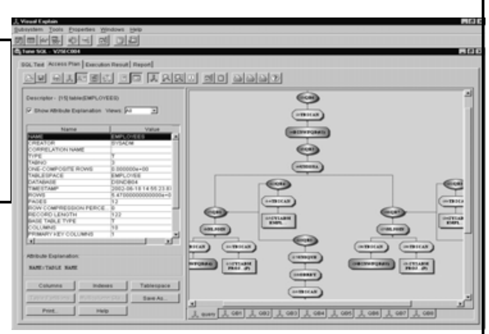
IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

EXPLAIN for global prepare cache

62

- Enhancements to the EXPLAIN statement allow you to obtain EXPLAIN information for entries in the DB2 global prepare cache
- Visual Explain uses this new function.

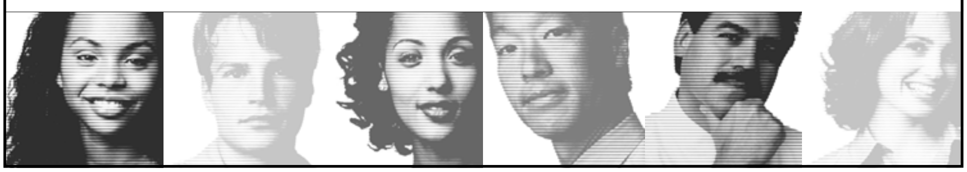
EXPLAIN STMTCACHE
STMTID=integer
STMTTOKEN=string



IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

Materialized Query Tables

TAKE BACK CONTROL



Materialized Query Tables (MQTs) 64

- Report-generation queries usually touch a large amount of data:
  - ►Selection criteria based on a few dimensions
  - ►Aggregating on a few dimension columns
  - ►Column functions applied to the records of interest
- Performance requirement:
  - ►Seconds or minutes elapsed time instead of hours
  - ►Avoid recomputation of same (complex) result
- Resolution:
  - ►Parallelism
  - ►Indexing
  - ►Materialization (precomputation) of common query result

Materialized Query Tables (MQTs)

65

- Sometimes called:
  - ►Summary Tables,
  - ►Automatic Summary Tables,
  - ►Automatic Materialized Query Tables,
  - ►Materialized Views, ...
- Optimizer can rewrite queries to access MQT instead of base table or view
- Pre-computed information
  - ►Significant performance improvement
- Managed by user or system (SQL REFRESH)
- Automatic rewrite or manual
- Informational Referential Integrity (not enforced)

Materialized Query Tables (MQTs)

66

- Table containing materialized data derived from one or more source tables specified by a fullselect
  - ►Source tables can be base tables, views, table expressions or user-defined table functions
- MQTs can be accessed
  - ►directly via SQL
  - ►or chosen by the optimizer when a base table or view is referenced, i.e. automatic query rewrite
- Two types:
  - ►Maintained by system
  - ►or by user
- Synchronization between base table(s) and MQT via
  - ►SQL REFRESH TABLE statement
  - ►Manually via batch update, triggers etc. for user-maintained

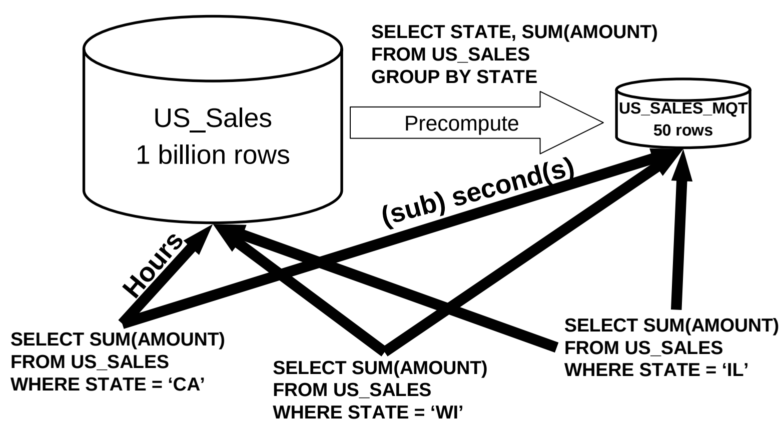
Create MQT - Example

67

```
CREATE TABLE MQT1 AS (
 SELECT T.PDATE, T.TRANSID,
        SUM(QTY * PRICE) AS TOTVAL,
        COUNT(QTY * PRICE) AS CNT
 FROM SCNDSTAR.TRANSITEM TI, SCNDSTAR.TRANS T
 WHERE TI.TRANSID = T.TRANSID
 GROUP BY T.PDATE, T.TRANSID)
DATA INITIALLY DEFERRED
REFRESH DEFERRED
MAINTAINED BY SYSTEM
ENABLE QUERY OPTIMIZATION
IN ...;
```


68

Materialized Query Tables (MQTs)



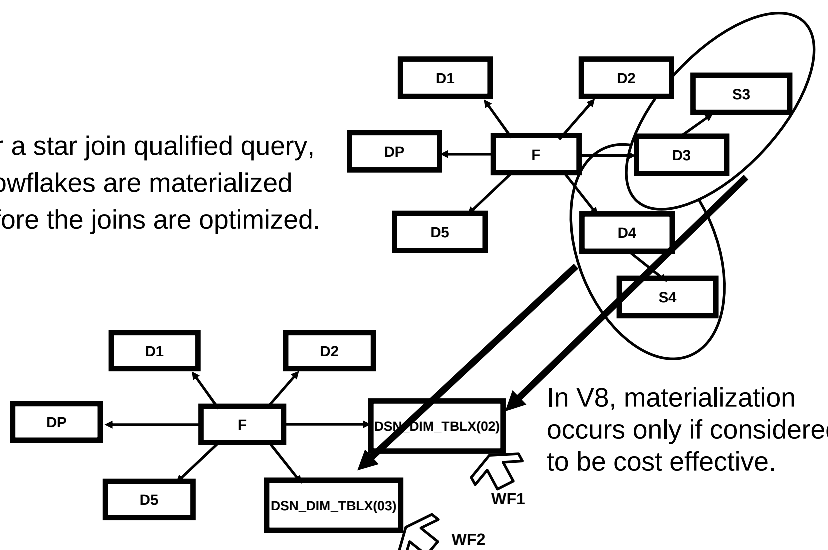
TAKE BACK CONTROL

Complex Join

Star Join - Snowflake Materialization

70

For a star join qualified query, snowflakes are materialized before the joins are optimized.



In V8, materialization occurs only if considered to be cost effective.

71

Star Join Sparse Indexes and in-memory work files

- Optimizer improvement that addresses the same requirement as Hash Join
  - ►ACCESS_TYPE='T' in PLAN table
  - ►Uses sparse index to process the contents of work file
  - ►Improves upon APAR PQ61458 on V7

72

Additional V8 Star Join Enhancements

- ►Improved cost formula
  - – Results in improved table join sequence
- ►Hybrid Join support for star join
- ►Star Join support for low clusterratio indexes
- ►Increased parallelism for non-partitioned fact tables
- ►Table localization when “OR” predicates cross dimensions
  - – Filtering predicates can be applied earlier in join sequence

Complex Join

73

- By default, max number of tables in V8 query is 225.
- For large number of tables, how do we limit resources?
  - Algorithms changed to use less storage and CPU
  - Recognize certain join patterns (ie. Star-join, Siebel)
  - Introduce “Clipping”
    - A process of reducing the search space when there are an excessive number of choices for the optimizer to consider

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

Complex Join – V8 ZPARMS

74

Keyword	Description	Default	Range
MAX_OPT_STOR (SPRMMXOS)	Max amount of RDS OP POOL storage consumed by Optimizer (MB)	20 MB	0 – 100 MB
MAX_OPT_CPU (SPRMMXOC)	Max amount of CPU Time consumed by DB2 Optimizer (Seconds)	100 sec	0 – 1000 sec
MAX_OPT_ELAP (SPRMMXOE)	Maximum amount of elapsed time consumed by the DB2 optimizer (seconds)	100 sec	0 – 1000 sec

- Complement the V7 ZPARMS
  - TABLES_JOINED_THRESHOLD (Default 16)
    - 12 means any join of 12 or more tables will limit resources
    - A setting above 16 may result in -101 SQLCODE or storage abends
  - MXQBCE (Default 32767)
    - Set to the maximum number of combinations to consider
    - Value of 1023 will use no more resources than for a "fully connected" 10 table join.
    - Use formula (2**n) - 1.

IBM INFORMATION ON DEMAND 2006 TAKE BACK CONTROL

IBM Information On Demand 2006

IBM

What's new with the Optimizer in DB2 for z/OS V8?

Terry Purcell, IBM Silicon Valley Lab
tpurcel@us.ibm.com
Session 1818

TAKE BACK CONTROL

IBM INFORMATION ON DEMAND 2006
October 15 – 20, 2006
Anaheim Convention Center
Anaheim, Cali