IBM System i™

Session:

The ABC's of Coding High-Performance SQL Apps

Presented by Jarek Miszczyk
DB2 for i5/OS Technology Team
IBM Rochester, MN USA

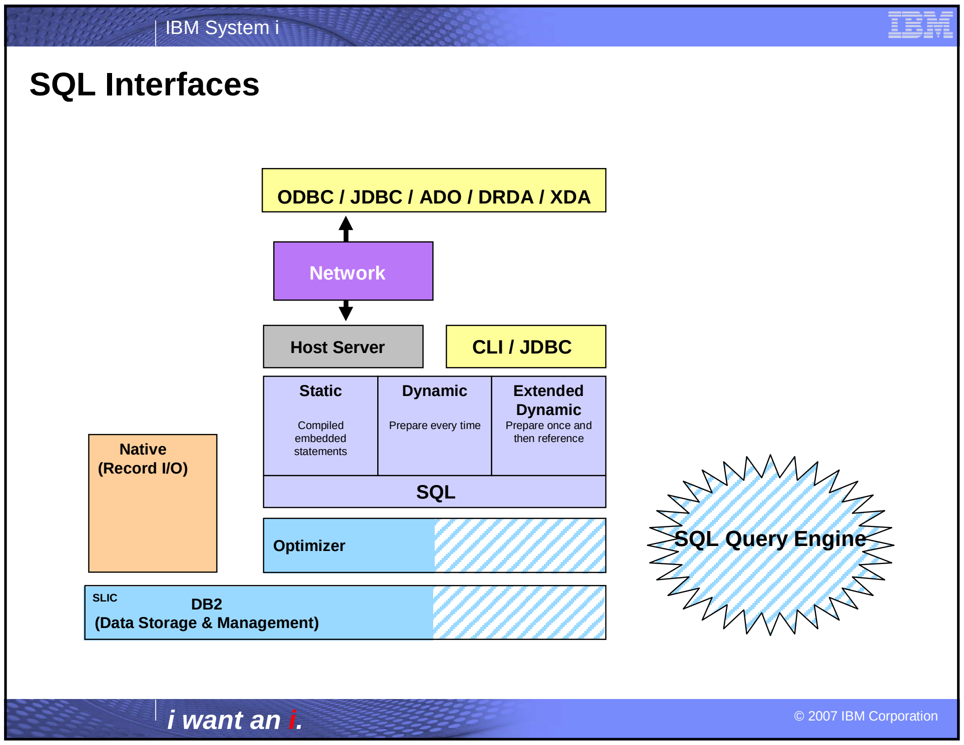
i want stress-free IT.
i want control.
i want an i.

Wright brothers software engineering:

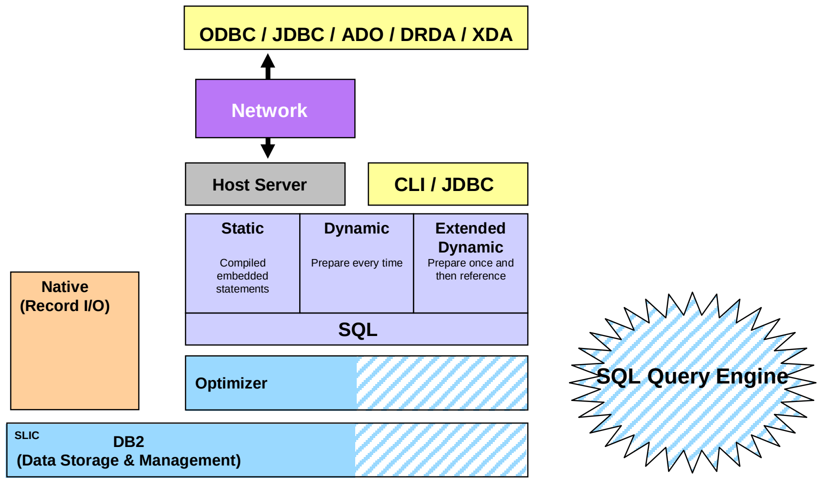
~

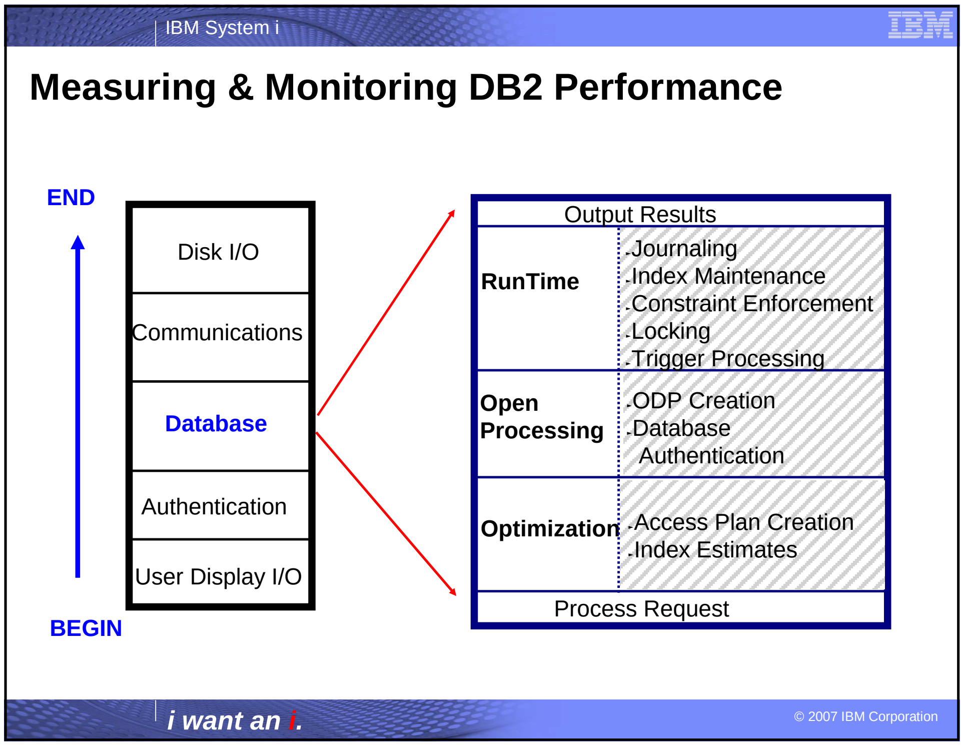
"Put it (the query) all together and push it off a cliff to see if it flies."

i want an i.

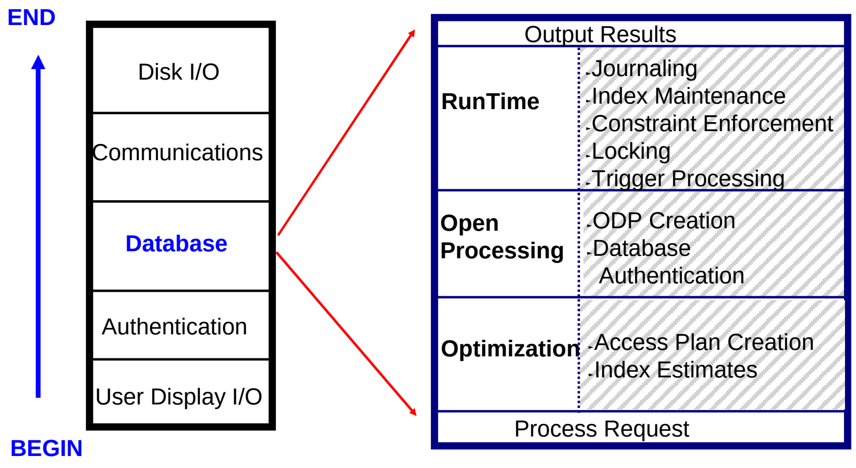


SQL Interfaces





Measuring & Monitoring DB2 Performance



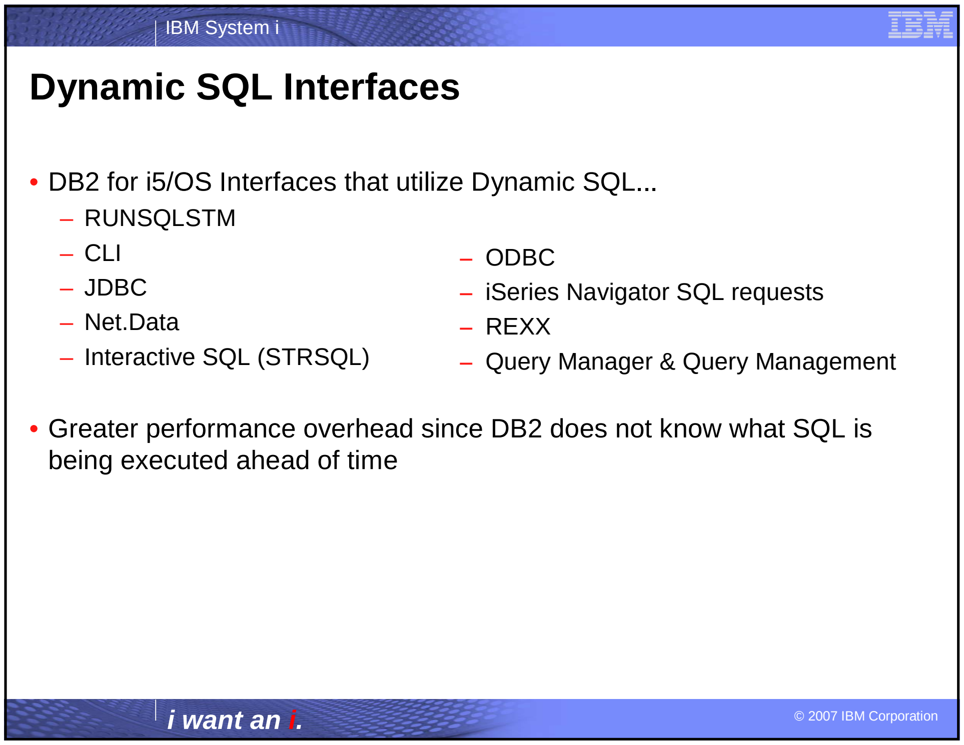
Static SQL

- Non-dynamic SQL statements embedded in application programs
- Languages Supported:
  - RPG
  - COBOL
  - C, C++
  - SQL Procedural Language (SQL embedded in C)
  - PL/I
- Most efficient SQL interface on iSeries

Dynamic SQL

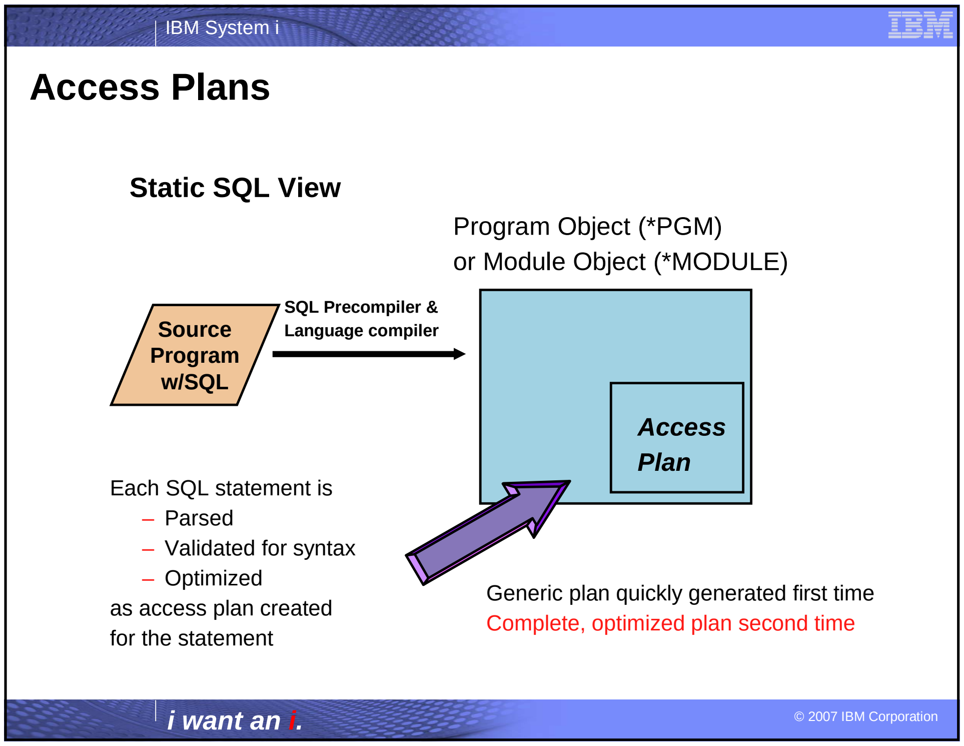
- SQL statements are dynamically created on the fly as part of application logic:
  PREPARE, EXECUTE, EXECUTE IMMEDIATE

```
DSTRING = 'DELETE FROM CORPDATA.EMPLOYEE
WHERE EMPNO = 33';

EXEC SQL
        PREPARE S1 FROM :DSTRING;

EXEC SQL
        EXECUTE S1;
```



Dynamic SQL Interfaces

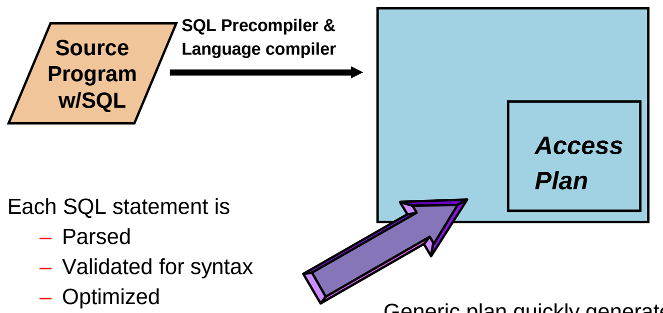
- DB2 for i5/OS Interfaces that utilize Dynamic SQL...
  - RUNSQLSTM
  - CLI
  - JDBC
  - Net.Data
  - Interactive SQL (STRSQL)
  - ODBC
  - iSeries Navigator SQL requests
  - REXX
  - Query Manager & Query Management
- Greater performance overhead since DB2 does not know what SQL is being executed ahead of time



Access Plans

Static SQL View

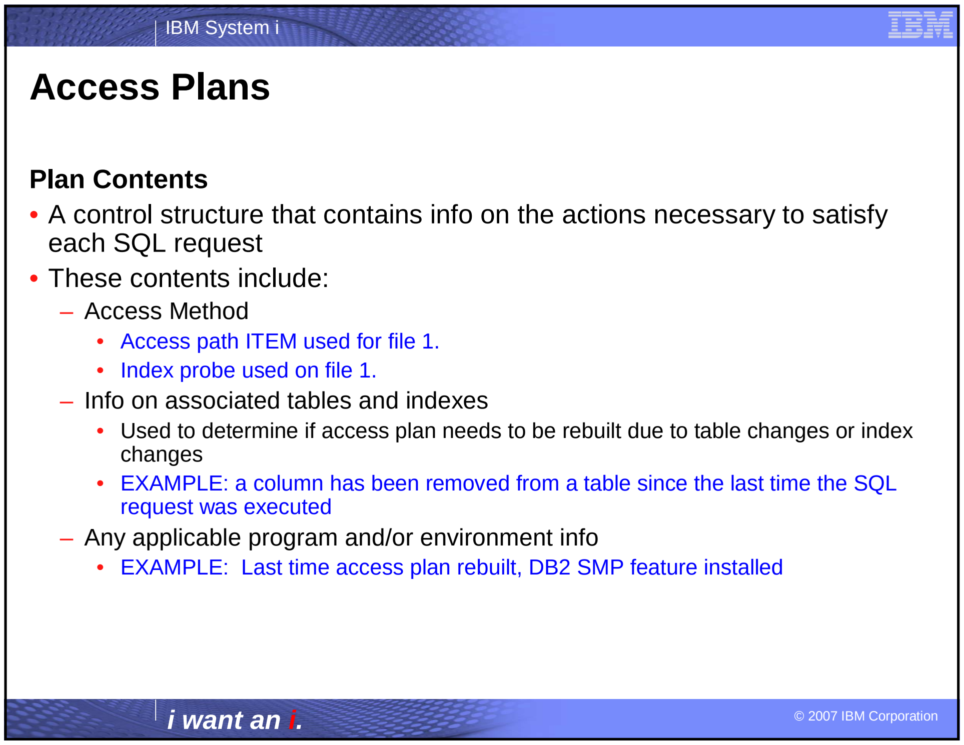
Program Object (*PGM)
or Module Object (*MODULE)



Each SQL statement is
- Parsed
- Validated for syntax
- Optimized

as access plan created for the statement

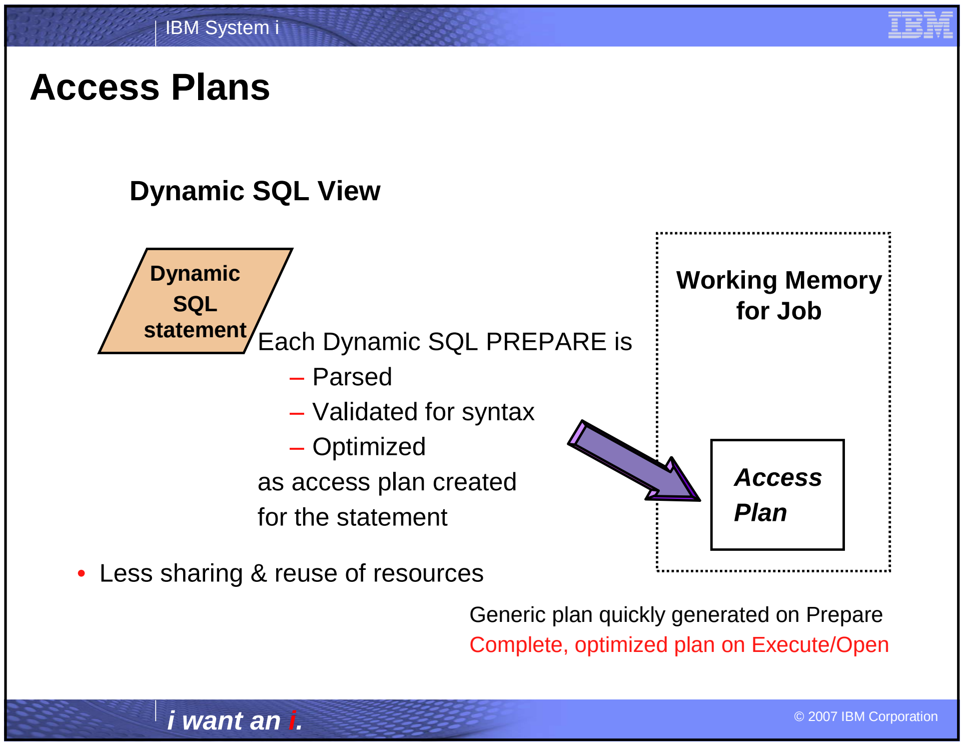
Generic plan quickly generated first time
Complete, optimized plan second time



Access Plans

Plan Contents

- A control structure that contains info on the actions necessary to satisfy each SQL request
- These contents include:
  - Access Method
    - Access path ITEM used for file 1.
    - Index probe used on file 1.
  - Info on associated tables and indexes
    - Used to determine if access plan needs to be rebuilt due to table changes or index changes
    - EXAMPLE: a column has been removed from a table since the last time the SQL request was executed
  - Any applicable program and/or environment info
    - EXAMPLE: Last time access plan rebuilt, DB2 SMP feature installed



Access Plans

Dynamic SQL View

Dynamic SQL statement

Each Dynamic SQL PREPARE is
- Parsed
- Validated for syntax
- Optimized

as access plan created for the statement

Working Memory for Job

Access Plan

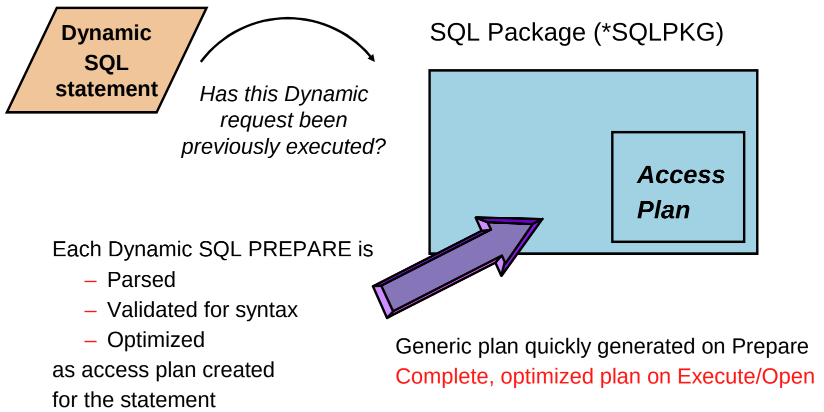
- Less sharing & reuse of resources

Generic plan quickly generated on Prepare

Complete, optimized plan on Execute/Open

Access Plans

Extended Dynamic SQL View


Dynamic SQL statement
Has this Dynamic request been previously executed?
SQL Package (*SQLPKG)
Access Plan
Generic plan quickly generated on Prepare
Complete, optimized plan on Execute/Open

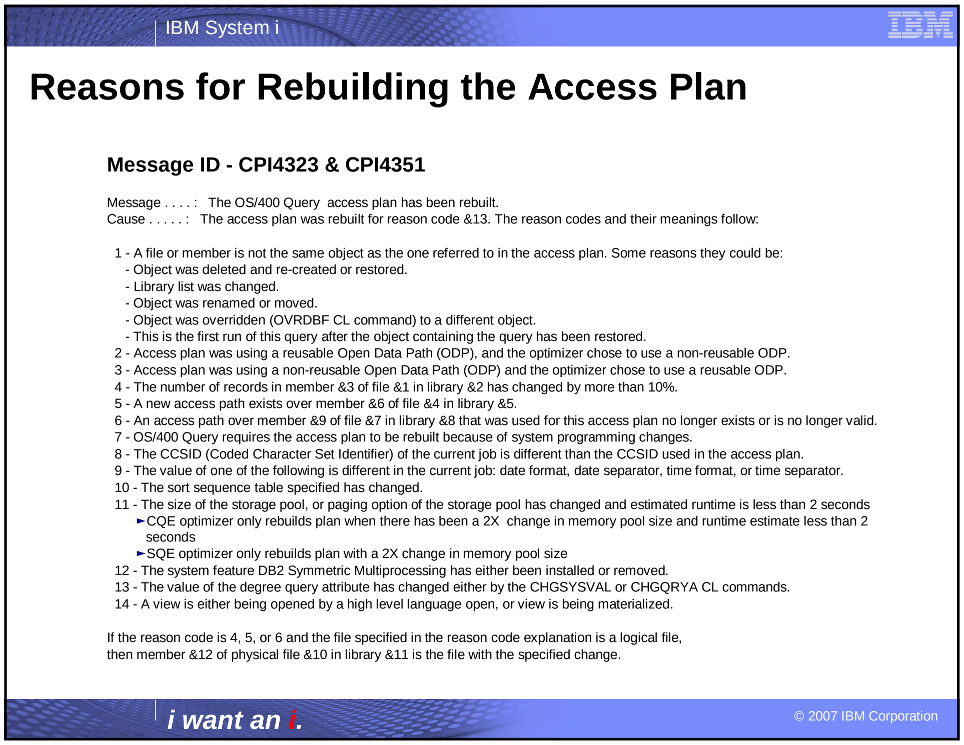
Each Dynamic SQL PREPARE is
- Parsed
- Validated for syntax
- Optimized

as access plan created for the statement

OPENing the Access Plan

- Validate the Access Plan
- IF NOT Valid, THEN Reoptimize & update plan (late binding)
  - Some of the possible reasons:
    - Table size greatly increased
    - Index added/removed
    - Significant host variable value change
- Implement Access Plan: CREATE ODP (Open Data Path)

NOTE: If optimizer has to rebuild access plan stored in a program or package object, then users may have to build a temporary access plan in some cases.



Reasons for Rebuilding the Access Plan

Message ID - CPI4323 & CPI4351

Message . . . . : The OS/400 Query access plan has been rebuilt.
Cause . . . . . : The access plan was rebuilt for reason code &13. The reason codes and their meanings follow:

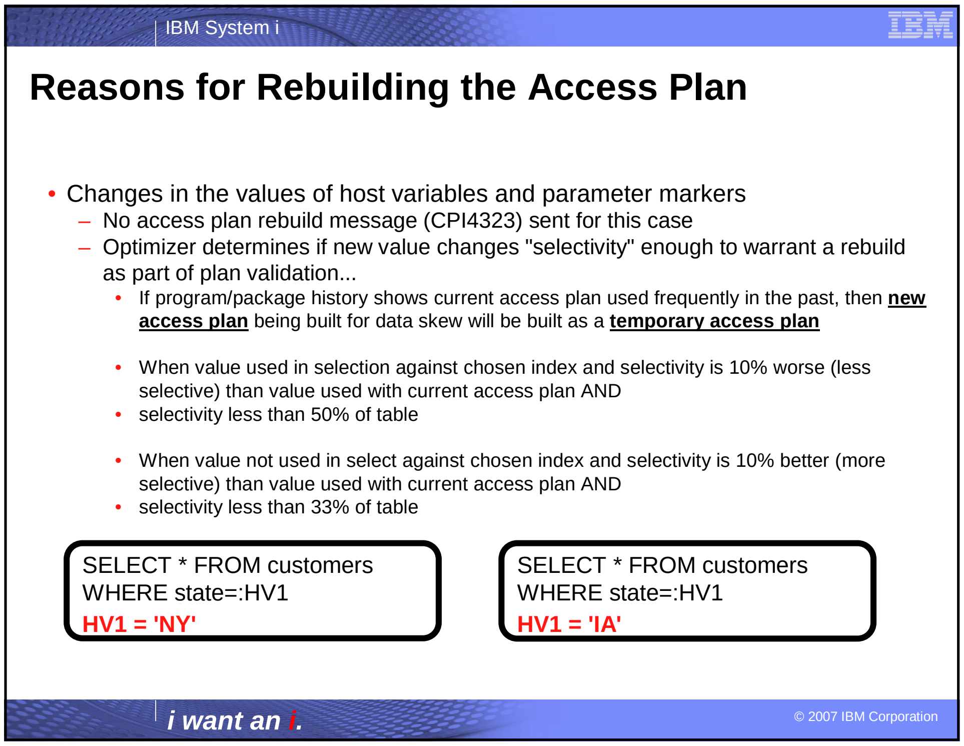
1 - A file or member is not the same object as the one referred to in the access plan. Some reasons they could be:
- Object was deleted and re-created or restored.
- Library list was changed.
- Object was renamed or moved.
- Object was overridden (OVRDBF CL command) to a different object.
- This is the first run of this query after the object containing the query has been restored.

2 - Access plan was using a reusable Open Data Path (ODP), and the optimizer chose to use a non-reusable ODP.
3 - Access plan was using a non-reusable Open Data Path (ODP) and the optimizer chose to use a reusable ODP.
4 - The number of records in member &3 of file &1 in library &2 has changed by more than 10%.
5 - A new access path exists over member &6 of file &4 in library &5.
6 - An access path over member &9 of file &7 in library &8 that was used for this access plan no longer exists or is no longer valid.
7 - OS/400 Query requires the access plan to be rebuilt because of system programming changes.
8 - The CCSID (Coded Character Set Identifier) of the current job is different than the CCSID used in the access plan.
9 - The value of one of the following is different in the current job: date format, date separator, time format, or time separator.
10 - The sort sequence table specified has changed.
11 - The size of the storage pool, or paging option of the storage pool has changed and estimated runtime is less than 2 seconds
- ►CQE optimizer only rebuilds plan when there has been a 2X change in memory pool size and runtime estimate less than 2 seconds
- ►SQE optimizer only rebuilds plan with a 2X change in memory pool size

12 - The system feature DB2 Symmetric Multiprocessing has either been installed or removed.
13 - The value of the degree query attribute has changed either by the CHGSYSVAL or CHGQRYA CL commands.
14 - A view is either being opened by a high level language open, or view is being materialized.

If the reason code is 4, 5, or 6 and the file specified in the reason code explanation is a logical file, then member &12 of physical file &10 in library &11 is the file with the specified change.

i want an i.



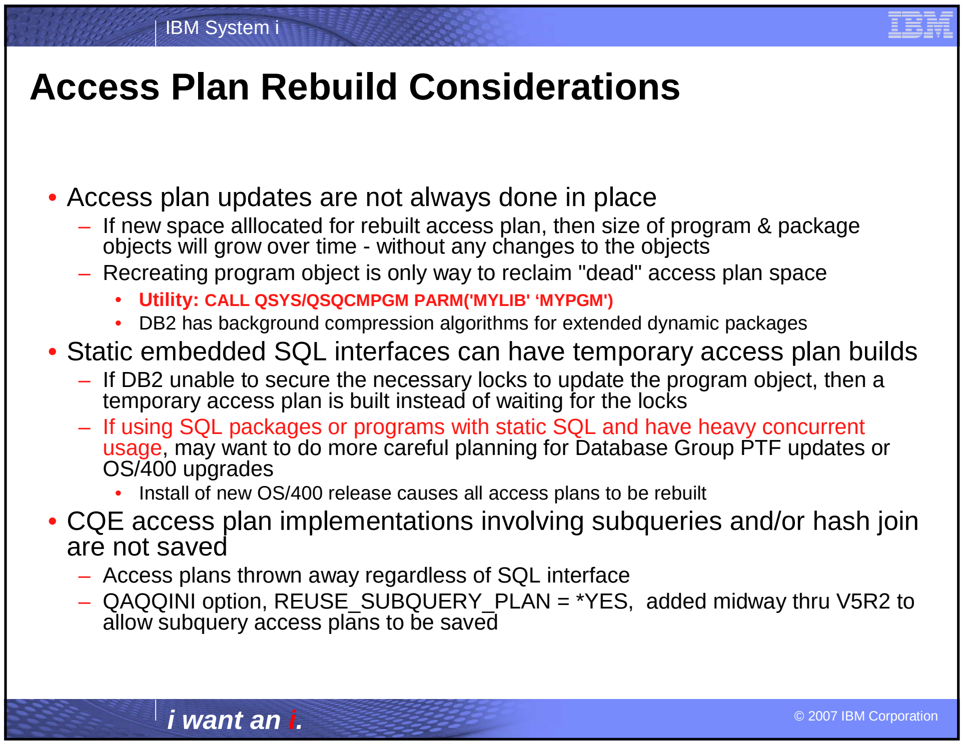
Reasons for Rebuilding the Access Plan

- Changes in the values of host variables and parameter markers
  - No access plan rebuild message (CPI4323) sent for this case
  - Optimizer determines if new value changes "selectivity" enough to warrant a rebuild as part of plan validation...
    - If program/package history shows current access plan used frequently in the past, then **new access plan** being built for data skew will be built as a **temporary access plan**
    - When value used in selection against chosen index and selectivity is 10% worse (less selective) than value used with current access plan AND
    - selectivity less than 50% of table
    - When value not used in select against chosen index and selectivity is 10% better (more selective) than value used with current access plan AND
    - selectivity less than 33% of table

```
SELECT * FROM customers
WHERE state=:HV1
HV1 = 'NY'
```

```
SELECT * FROM customers
WHERE state=:HV1
HV1 = 'IA'
```

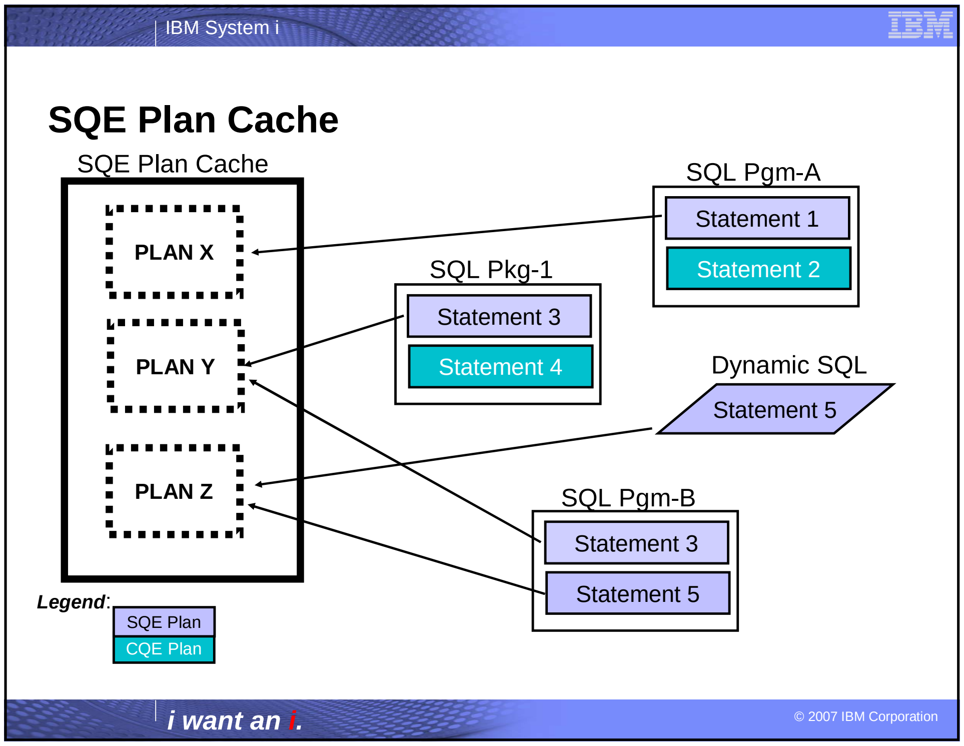
i want an i.



Access Plan Rebuild Considerations

- Access plan updates are not always done in place
  - If new space alllocated for rebuilt access plan, then size of program & package objects will grow over time - without any changes to the objects
  - Recreating program object is only way to reclaim "dead" access plan space
    - **Utility: CALL QSYS/QSQCMPGM PARM('MYLIB' 'MYPGM')**
    - DB2 has background compression algorithms for extended dynamic packages
- Static embedded SQL interfaces can have temporary access plan builds
  - If DB2 unable to secure the necessary locks to update the program object, then a temporary access plan is built instead of waiting for the locks
  - If using SQL packages or programs with static SQL and have heavy concurrent usage, may want to do more careful planning for Database Group PTF updates or OS/400 upgrades
    - Install of new OS/400 release causes all access plans to be rebuilt
- CQE access plan implementations involving subqueries and/or hash join are not saved
  - Access plans thrown away regardless of SQL interface
  - QAQQINI option, REUSE_SUBQUERY_PLAN = *YES, added midway thru V5R2 to allow subquery access plans to be saved

i want an i.



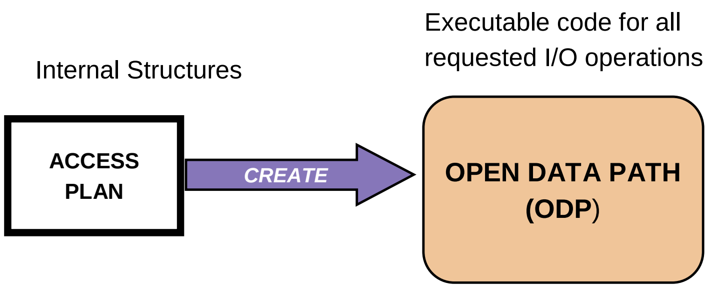
SQE Plan Cache

i want an i.

SQE Plan Cache

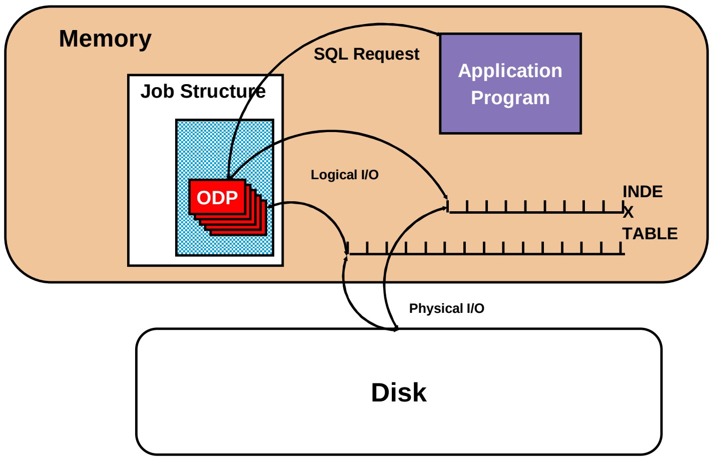
- Self-managed cache for all plans produced by SQE Optimizer
  - Allows more reuse of existing plans regardless of interface for identical SQL statements
    - Room for about 6000-10000 SQL statements
    - Plans are stored in a compressed mode
    - Up to 3 plans can be stored per SQL statement
  - Access is optimized to minimize contention on plan entries across system
  - Cache is automatically maintained to keep most active queries available for reuse
  - Foundation for a self-learning query optimizer to interrogate the plans to make wiser costing decisions
- SQE Access Plans actually divided between Plan Cache & Containing Object (Program, Package, etc)
  - Plan Cache stores the optimized portion (e.g., the index scan recipe) of the access plan
  - The access plan components needed for validating an SQL request (such as the SQL statement text and object information) is left in the original access plan location along with a virtual link to the plan in the Plan Cache
  - Plan cache entry also contains information on automatic stats collection & refresh
- Plan Cache is cleared at IPL (& IASP vary off)

Access Plan to ODP



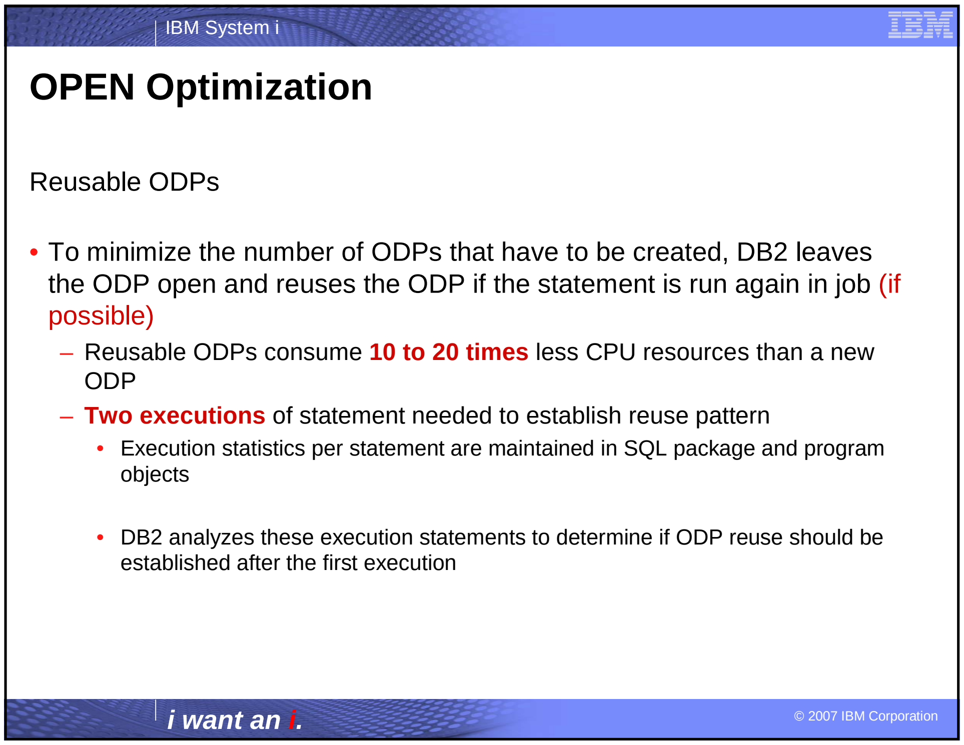
- Create process is EXPENSIVE
  - Longer execution time the first time an SQL statement is executed
- Emphasizes the need of REUSABLE ODPs

ODP's "In Action"



OPEN Optimization

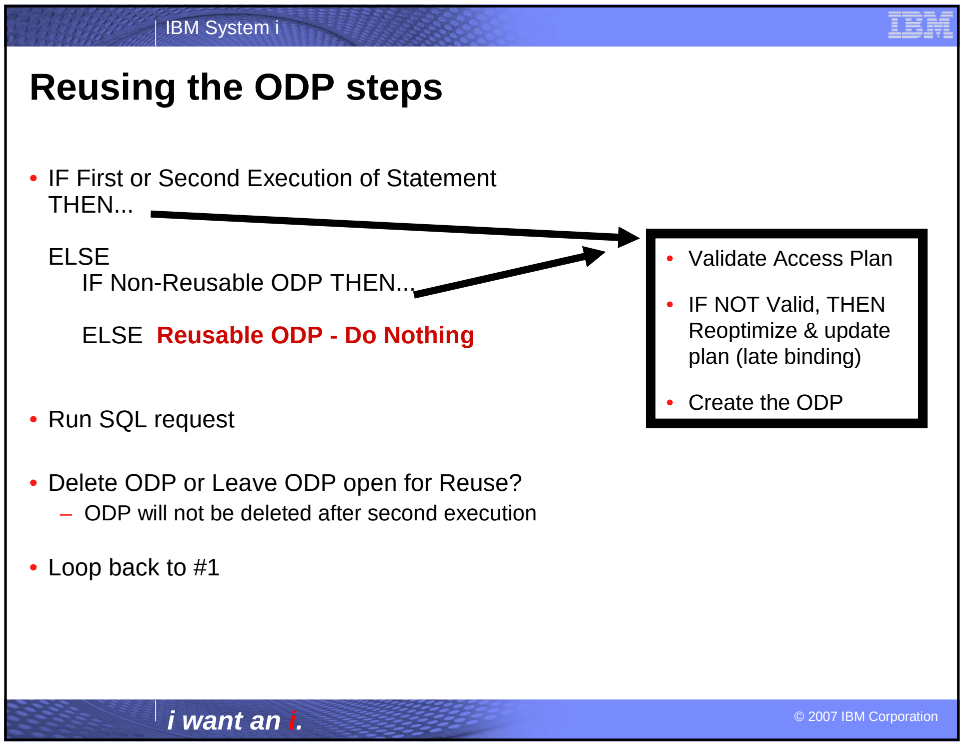
- OPENs can occur on:
  - OPEN Statement
  - SELECT Into Statement
  - INSERT statement with a VALUES clause
  - INSERT statement with a SELECT (2 OPENs)
  - Searched UPDATE's
  - Searched DELETE's
  - Some SET statements
  - VALUES INTO statement
  - Certain subqueries may require one Open per subselect
- The request and environment determine if the OPEN requires an ODP Creation ("Full" Open)



OPEN Optimization

Reusable ODPs

- To minimize the number of ODPs that have to be created, DB2 leaves the ODP open and reuses the ODP if the statement is run again in job (if possible)
  - Reusable ODPs consume **10 to 20 times** less CPU resources than a new ODP
  - **Two executions** of statement needed to establish reuse pattern
    - Execution statistics per statement are maintained in SQL package and program objects
    - DB2 analyzes these execution statements to determine if ODP reuse should be established after the first execution



Reusing the ODP steps

- IF First or Second Execution of Statement
  THEN...

  ELSE
     IF Non-Reusable ODP THEN...

     ELSE **Reusable ODP - Do Nothing**

  - Validate Access Plan
  - IF NOT Valid, THEN Reoptimize & update plan (late binding)
  - Create the ODP

- Run SQL request
- Delete ODP or Leave ODP open for Reuse?
  - ODP will not be deleted after second execution
- Loop back to #1

OPEN Optimization

Reusable ODP Example

```
INSERT INTO resultTable
    SELECT id, name
        FROM customers
            WHERE region = 'Central'
```

```
SQL7912   ODP created.
SQL7912   ODP created.             ←
...
SQL7913   ODP deleted.
SQL7913   ODP deleted.
SQL7985   CALL statement complete
SQL7912   ODP created.
SQL7912   ODP created.
...
SQL7914   ODP not deleted.
SQL7914   ODP not deleted.         ←
SQL7985   CALL statement complete
SQL7911   ODP reused.
SQL7911   ODP reused.              ←
...
...
SQL7914   ODP not deleted.
SQL7914   ODP not deleted.
SQL7985   CALL statement complete
```

Miscellaneous considerations

Reusable ODP Control - QSQPSCLS1 Data Area

- Existence of data area allows the reuse behavior after first execution of SQL statement instead of the second execution
  - DB2 checks for data area named QSQPSCLS1 in job's library list - existence only checked at the beginning of the job (first SQL ODP)
  - **USE CAREFULLY** since cursors that are not reused will consume extra storage
  - Data area contents, type, and length are not applicable

Reusable ODP Tips & Techniques

OPEN Optimization - Reuse Roadblocks

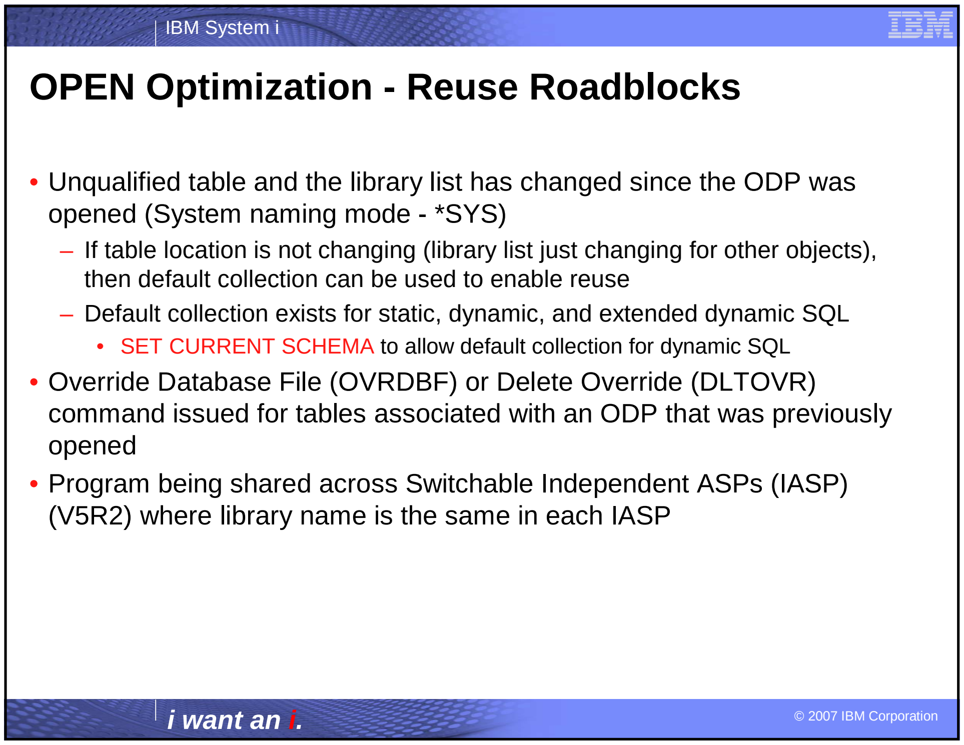
- With embedded SQL, DB2 only reuses ODPs opened by the same statement
  - If same statement will be executed multiple times, need to code logic so that statement is in a shared subroutine that can be called
  - ODPs for the same static SQL statements in different programs or stored procedure are NOT reused

NON-REUSABLE ODP

```
SELECT name FROM emptbl
WHERE id=:hostvar

...
SELECT name FROM emptbl
WHERE id=:hostvar
...
```

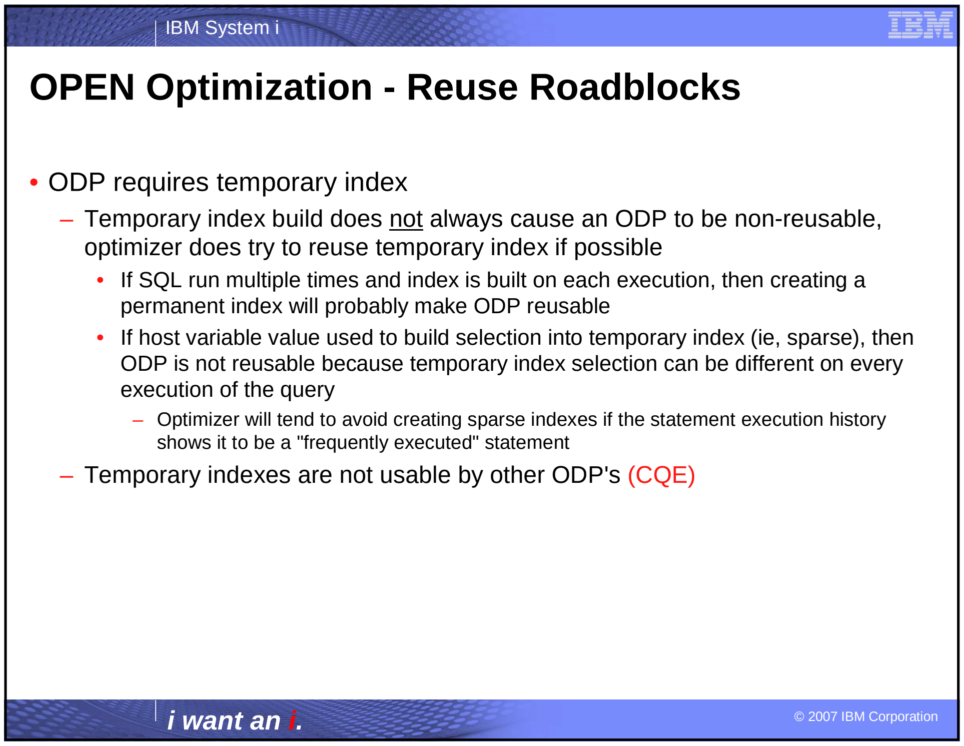
REUSABLE ODP

```
CALL Proc1
...
CALL Proc1
...

Proc1:=========
SELECT name FROM emptbl
WHERE id=:hostvar
```



OPEN Optimization - Reuse Roadblocks

- Unqualified table and the library list has changed since the ODP was opened (System naming mode - *SYS)
  - If table location is not changing (library list just changing for other objects), then default collection can be used to enable reuse
  - Default collection exists for static, dynamic, and extended dynamic SQL
    - SET CURRENT SCHEMA to allow default collection for dynamic SQL
- Override Database File (OVRDBF) or Delete Override (DLTOVR) command issued for tables associated with an ODP that was previously opened
- Program being shared across Switchable Independent ASPs (IASP) (V5R2) where library name is the same in each IASP



OPEN Optimization - Reuse Roadblocks

- ODP requires temporary index
  - Temporary index build does <u>not</u> always cause an ODP to be non-reusable, optimizer does try to reuse temporary index if possible
    - If SQL run multiple times and index is built on each execution, then creating a permanent index will probably make ODP reusable
    - If host variable value used to build selection into temporary index (ie, sparse), then ODP is not reusable because temporary index selection can be different on every execution of the query
      - Optimizer will tend to avoid creating sparse indexes if the statement execution history shows it to be a "frequently executed" statement
  - Temporary indexes are not usable by other ODP's (CQE)

OPEN Optimization

UPDATE WHERE CURRENT OF Reuse

- If an UPDATE WHERE CURRENT OF request contains a function or operator on the SET clause, then an open operation must be performed
- Can avoid this open by performing the function or operation in the host language
  - Code operation into host language...

```
FETCH EMPT INTO :Salary;

Salary = Salary + 1000;

UPDATE EMPLOYEE
      SET Salary = :Salary
      WHERE CURRENT OF Empt;
```

  - Instead of...

```
FETCH EMPT INTO :Salary;
UPDATE Employee
       SET Salary = :Salary+1000
       WHERE CURRENT OF Empt;
```

OPEN Optimization - Reuse Considerations

- Reusable ODP's do have one shortcoming... once reuse mode has started access plan is NOT rebuilt when the environment changes
  - What happens to performance if Reusable ODP is now run against a table that started out empty and has now grown 5X in size since the last execution?
  - What if selectively of host variable or parameter marker greatly different on 5th execution of statement?
  - What if index added for tuning after 5th execution of statement in the job?

***NOT an issue with SQE since V5R3 – SQE recognizes new indexes and table size changes while in ODP reuse mode

OPEN Optimization

Actions that Delete ODPs

- SQL DISCONNECT statement
- CLOSQLCSR(*ENDPGM) - ONLY deletes ODP's on program exit, if it's the last SQL program on the call stack
- A Reclaim request is issued:
  Reclaim Activation Group (RCLACTGRP) for ILE programs or
  Reclaim Resource (RCLRSC) for OPM programs
  - A Reclaim will not close ODP when programs precompiled using CLOSQLCUR(*ENDJOB)
  - With COBOL, RCLRSC issued when...
    - First COBOL program on the call stack ends
    - COBOL program issues the STOP RUN statement

OPEN Optimization

Actions that Delete ODPs (continued)

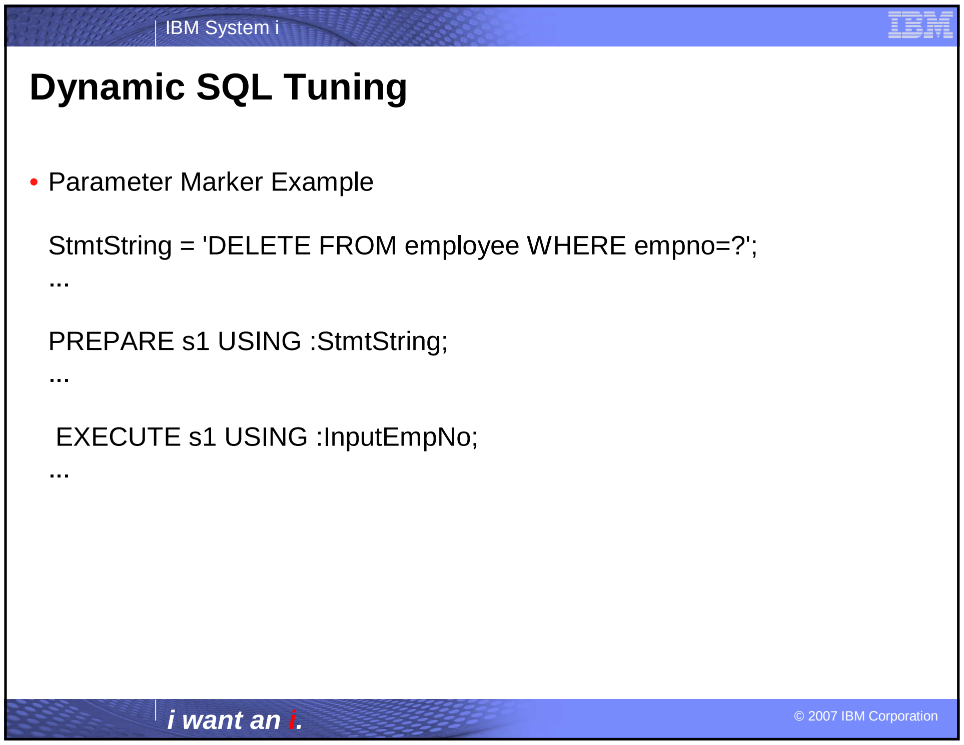
- CONFLICT parameter added to ALCOBJ command that can be used to request that pseudo-closed cursors to be hard closed
  - CONFLICT(*RQSRLS) (not the default) request to release lock sent to each job and thread holding a conflicting lock
    - **Will not release real application locks**
    - **Only releases implicit system locks for Reusable ODP cursors**
    - **Does not release Reusable ODP locks in requestor's job, only other jobs**
- ODP reuse can also be controlled/managed with the QAQQINI options added in V4R5
  - OPEN_CURSOR_THRESHOLD & OPEN_CURSOR_CLOSE_COUNT
- CLI provides special statement attribute & Toolbox JDBC Driver
- OS/400 Extended Dynamic interface gives programmer control of ODP deletion

Dynamic & Extended Dynamic SQL

Dynamic SQL Tuning

- With Dynamic interfaces, full opens are avoided by using a "PREPARE once, EXECUTE many" mentality when an SQL statement is going to be executed more than once
- A PREPARE does NOT automatically create a new statement and full open on each execution
  - DB2 performs caching on Dynamic SQL PREPAREs within a job/connection
  - DB2 caching is not perfect (and subject to change), good application design is the only way to guarantee ODP reuse
  - Job Cache searched by Statement Text & Statement Name to try and reuse existing ODPs or Plans (white space matters on statement)

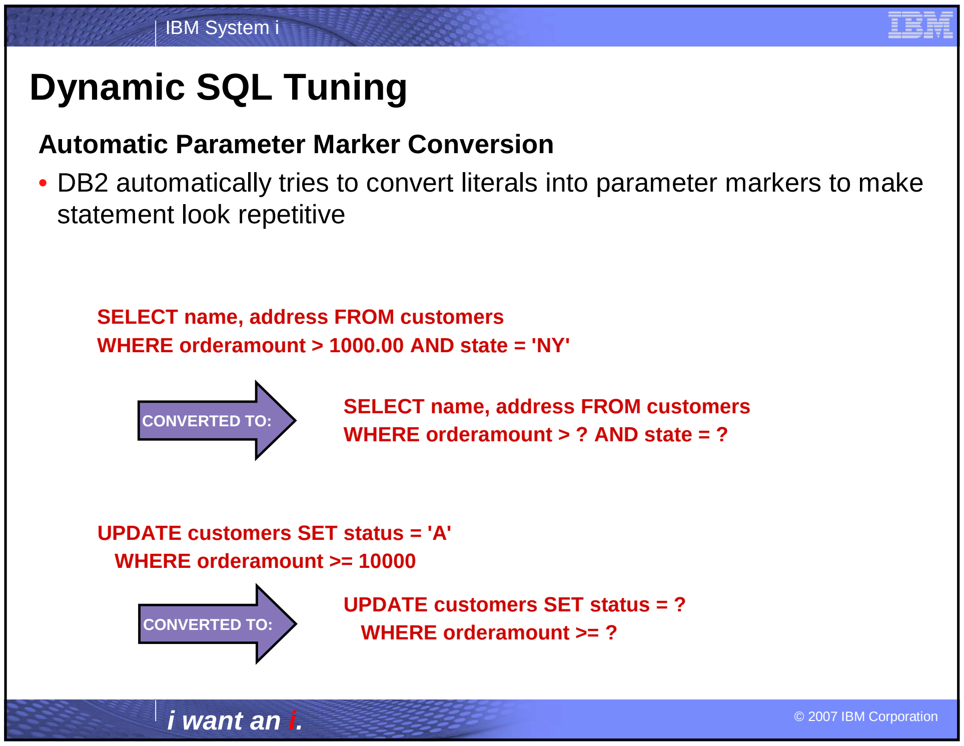
```
PreparedStatement pst = con.prepareStatement
                           ("INSERT INTO c1 VALUES( ?, ?, ?, ?, ?)");
for (int i = 0; i < outerNumOfLoops; i++) {
    for (int j = 0; j < numOfLoops; j++) {
        pst.setString(1, "GenData_" + Integer.toString(j));
        …
        pst.addBatch();
        }
    int [] updateCounts = pst.executeBatch();
    con.commit();                                           }
```



Dynamic SQL Tuning

- Parameter Marker Example

```
StmtString = 'DELETE FROM employee WHERE empno=?';
...

PREPARE s1 USING :StmtString;
...

 EXECUTE s1 USING :InputEmpNo;
...
```



Dynamic SQL Tuning

Automatic Parameter Marker Conversion

- DB2 automatically tries to convert literals into parameter markers to make statement look repetitive

```
SELECT name, address FROM customers
WHERE orderamount > 1000.00 AND state = 'NY'
```

CONVERTED TO:

```
SELECT name, address FROM customers
WHERE orderamount > ? AND state = ?
```

```
UPDATE customers SET status = 'A'
  WHERE orderamount >= 10000
```

CONVERTED TO:

```
UPDATE customers SET status = ?
  WHERE orderamount >= ?
```

Extended Dynamic & Packages

- Package is searched to see if there is a statement with the same SQL and attributes
  - Hash tables used to make statement searches faster
- If a match is found, then a new statement entry name is allocated with a pointer to the existing statement information (access plan, etc)
  - DB Monitor can be used to determine if "packaged" statement used at execution time:
    - SELECT qqc103, qqc21, qq1000 from ‹db monitor table›
      WHERE qqrid=1000 AND qvc18='E'

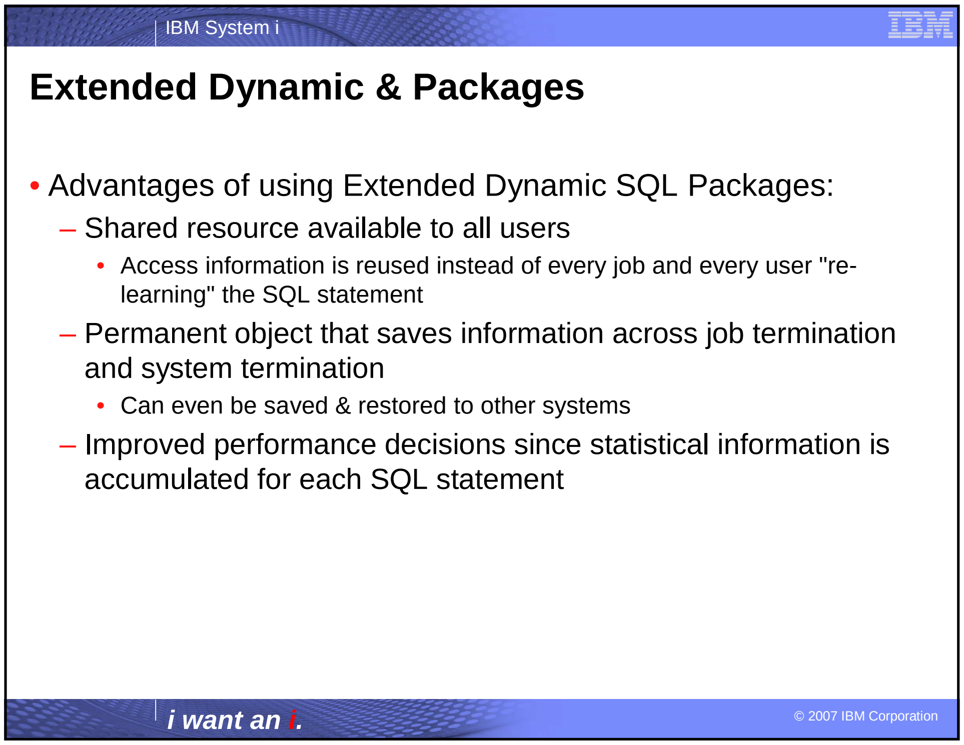
Extended Dynamic & Packages

Package Contents:

- Statement name
- Statement text
- Statement parse tree
- Access Plan

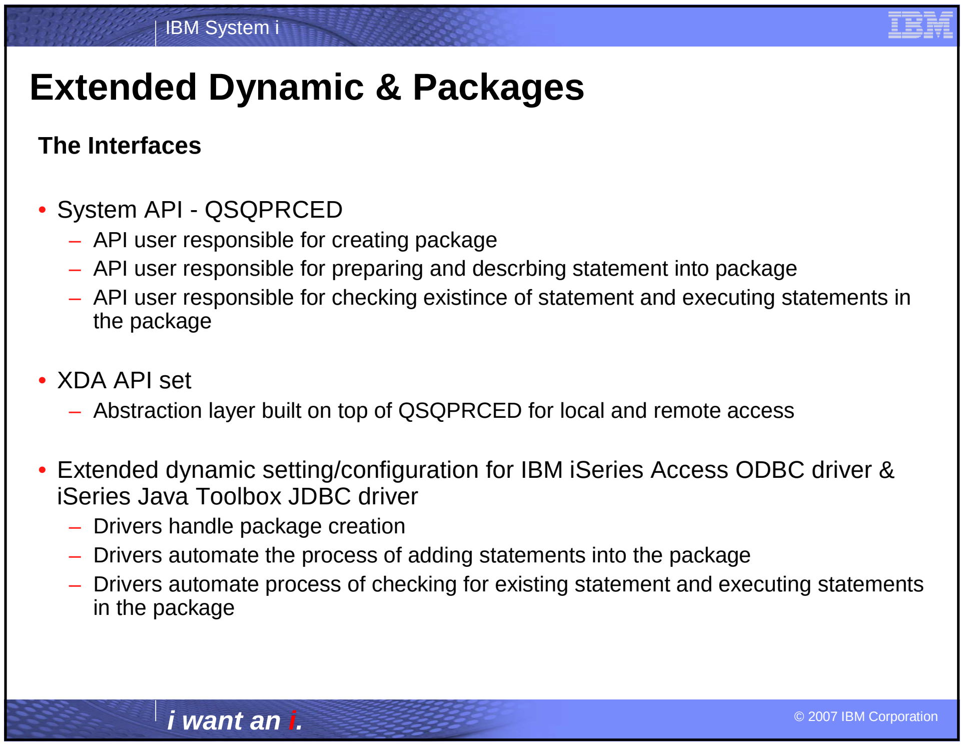
PRTSQLINF output →

```
STATEMENT NAME:  QZ7A6B3E74C31D0000
Select IID, INAME, IPRICE, IDATA from TEST/ITEM where
IID in ( ?, ?, ?, ?)
  SQL4021  Access plan last saved on 12/16/96 at 20:21:45.
  SQL4020  Estimated query run time is 1 seconds.
  SQL4008  Access path ITEM used for file 1.
  SQL4011  Key row positioning used on file 1.
...
STATEMENT NAME:  QZ7A6B3E74DD6D8000
Select CLAST, CDCT, CCREDT, WTAX from  TEST/CSTMR,
TEST//WRHS where  CWID=? and CDID=?
   SQL4021  Access plan last saved on 12/16/96 at 20:21:43.
   SQL4020  Estimated query run time is 1 seconds.
   SQL4007  Query implementation for join position 1 file 2.
   SQL4008  Access path WRHS used for file 2.
   SQL4011  Key row positioning used on file 2.
   SQL4007  Query implementation for join position 2 file 1.
   SQL4006  All access paths considered for file 1.
   SQL4008  Access path CSTMR used for file 1.
   SQL4014  0 join field pair(s) are used for this join position.
   SQL4011  Key row positioning used on file 1.
```



Extended Dynamic & Packages

- Advantages of using Extended Dynamic SQL Packages:
  - Shared resource available to all users
    - Access information is reused instead of every job and every user "re-learning" the SQL statement
  - Permanent object that saves information across job termination and system termination
    - Can even be saved & restored to other systems
  - Improved performance decisions since statistical information is accumulated for each SQL statement



Extended Dynamic & Packages

The Interfaces

- System API - QSQPRCED
  - API user responsible for creating package
  - API user responsible for preparing and descrbing statement into package
  - API user responsible for checking existince of statement and executing statements in the package
- XDA API set
  - Abstraction layer built on top of QSQPRCED for local and remote access
- Extended dynamic setting/configuration for IBM iSeries Access ODBC driver & iSeries Java Toolbox JDBC driver
  - Drivers handle package creation
  - Drivers automate the process of adding statements into the package
  - Drivers automate process of checking for existing statement and executing statements in the package

Extended Dynamic & Packages

Considerations

- Any SQL statement that can be prepared is eligible
  - ODBC & JDBC drivers have further restrictions
- Size limitations
  - Current size limit is 500 MB, about 16K statements
  - Package can grow without new statements being added. Access plan rebuilds require additional storage
  - Background package compression tries to increase life and usefulness of package objects
- Good online SQLPackage FAQ at the DB2 for i5/OS web site, FAQ URL:

  http://www.iseries.ibm.com/db2/sqlperffaq.htm

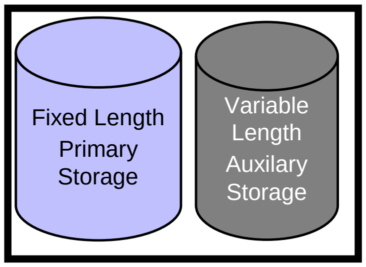
SQL Performance Techniques & Considerations

VARCHAR considerations

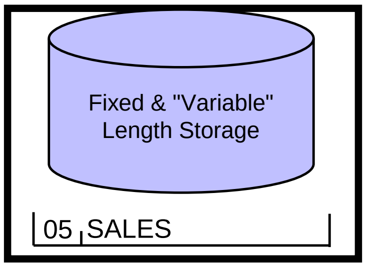
- Variable length columns (VARCHAR/VARGRAPHIC)
  - If primary goal is space saving, include ALLOCATE(0) with VARCHAR definition
  - If primary goal is performance, ALLOCATE value should be wide enough to accommodate 90-95% of the values that will be assigned to the varying length column
    - Minimizes number of times that DB2 has to touch data in overflow storage area
- VARCHAR columns more efficient on wildcard searches
  - DB2 able to stop searching after the end of the string - with fixed length characters it must search to the end of string, even if all blanks

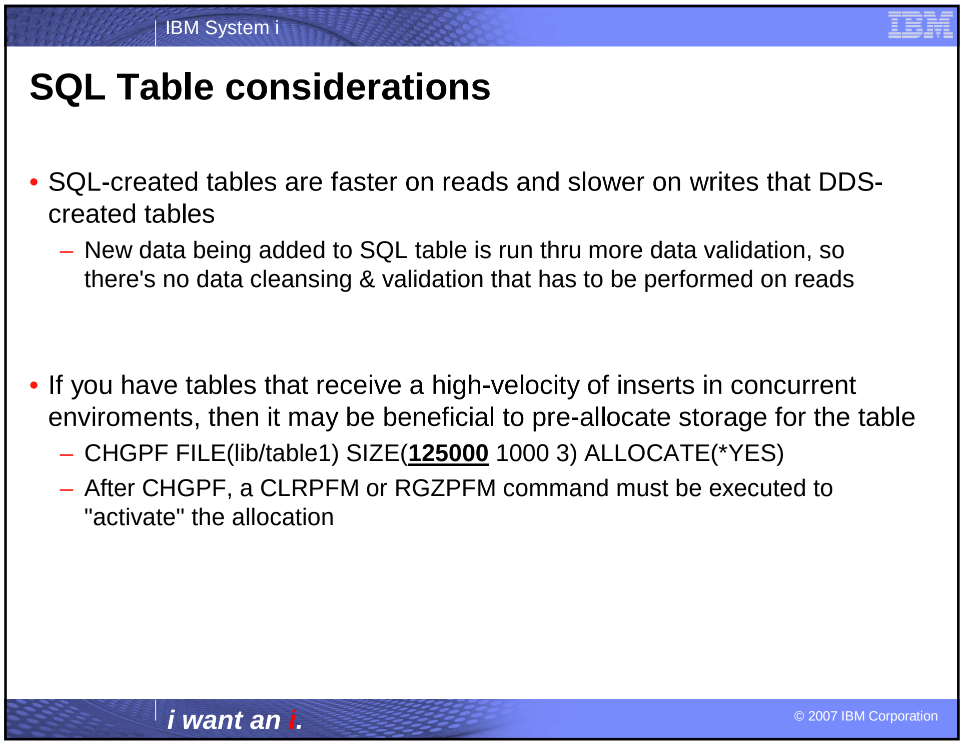
VARCHAR considerations

```
CREATE TABLE dept
(
  id      CHAR(4),
  name VARCHAR(40),
  bldg_num INTEGER
 )
```


Fixed Length Primary Storage
Variable Length Auxilary Storage

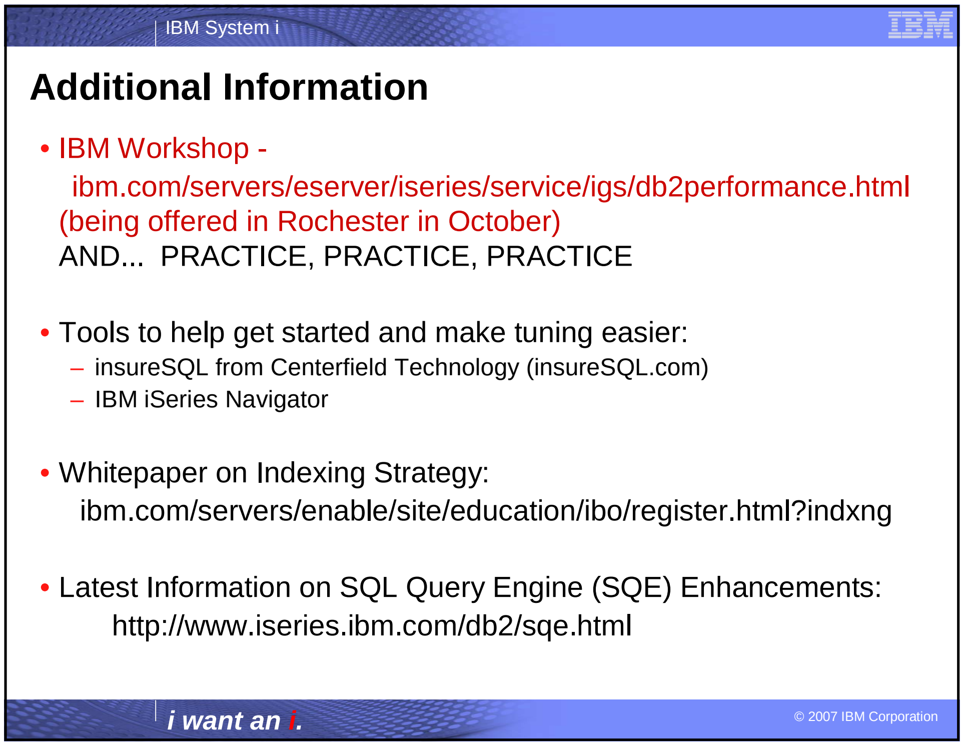
```
CREATE TABLE dept
(
  id      CHAR(4),
  name VARCHAR(40)
           ALLOCATE(40),
  bldg_num INTEGER
 )
```





SQL Table considerations

- SQL-created tables are faster on reads and slower on writes that DDS-created tables
  - New data being added to SQL table is run thru more data validation, so there's no data cleansing & validation that has to be performed on reads
- If you have tables that receive a high-velocity of inserts in concurrent enviroments, then it may be beneficial to pre-allocate storage for the table
  - CHGPF FILE(lib/table1) SIZE(**125000** 1000 3) ALLOCATE(*YES)
  - After CHGPF, a CLRPFM or RGZPFM command must be executed to "activate" the allocation



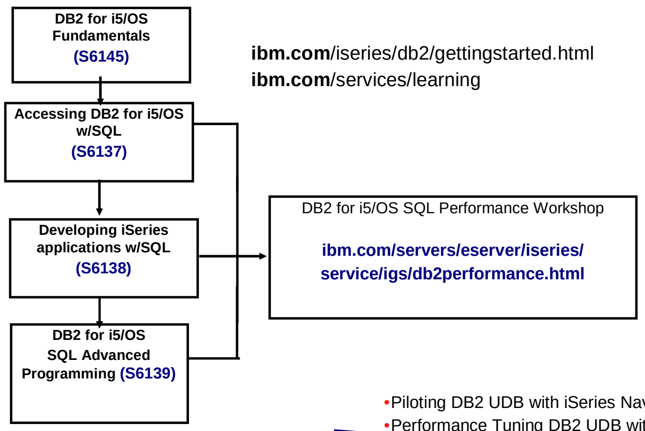
Additional Information

- IBM Workshop -
  ibm.com/servers/eserver/iseries/service/igs/db2performance.html
  (being offered in Rochester in October)
  AND... PRACTICE, PRACTICE, PRACTICE
- Tools to help get started and make tuning easier:
  - insureSQL from Centerfield Technology (insureSQL.com)
  - IBM iSeries Navigator
- Whitepaper on Indexing Strategy:
  ibm.com/servers/enable/site/education/ibo/register.html?indxng
- Latest Information on SQL Query Engine (SQE) Enhancements:
  http://www.iseries.ibm.com/db2/sqe.html

Additional Information

- DB2 for i5/OS home page - ibm.com/iseries/db2
- Education Resources - Classroom & Online
  - http://ibm.com/iseries/db2/gettingstarted.html
  - ibm.com/servers/enable/site/education/ibo/view.html?oc#db2
  - ibm.com/servers/enable/site/education/ibo/view.html?wp#db2
- DB2 for i5/OS Publications
  - Online Manuals: http://ibm.com/iseries/db2/books.html
  - Porting Help: http://ibm.com/servers/enable/site/db2/porting.html
  - DB2 for i5/OS Redbooks (http://ibm.com/redbooks)
    - Stored Procedures, Triggers, and User-Defined Functions on DB2 for iSeries (SG24-6503)
    - Preparing for & Understanding the SQL Query Engine Redbook (SG24-6598)
    - Modernizing iSeries Application Data Access (SG24-6393)
    - SQL Performance Diagnosis with Database Monitor (SG24-6654)
  - *SQL/400 Developer's Guide* by Paul Conte & Mike Cravitz
    - http://www.iseriesnetwork.com/str/books/Uniquebook2.cfm?NextBook=183

Education Roadmap

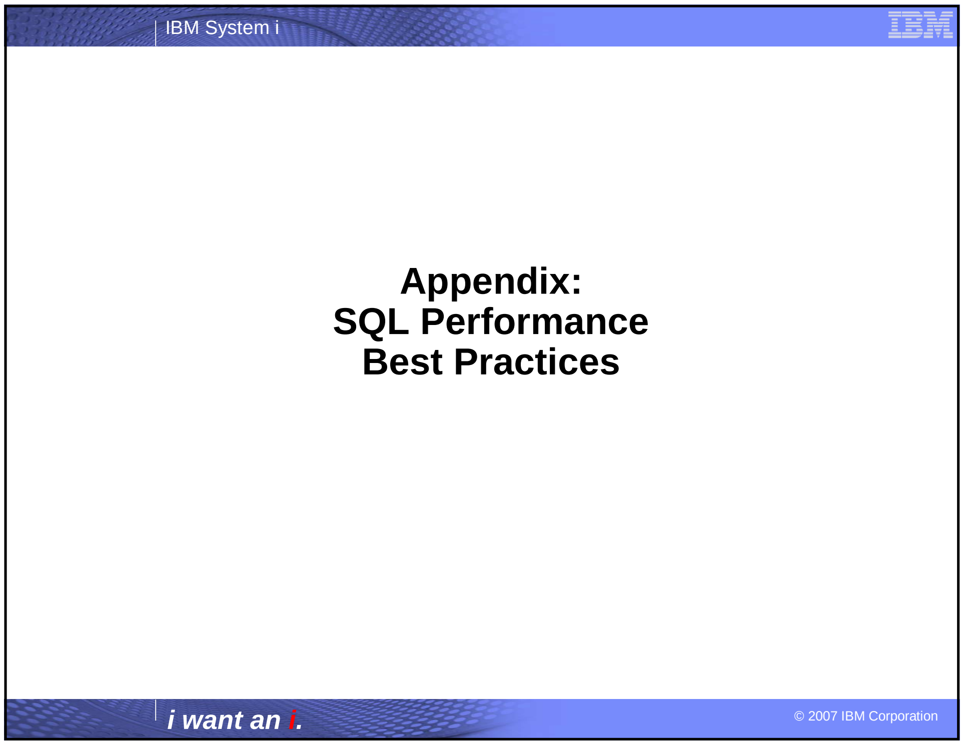

DB2 for i5/OS Fundamentals (S6145)
Accessing DB2 for i5/OS w/SQL (S6137)
Developing iSeries applications w/SQL (S6138)
DB2 for i5/OS SQL Advanced Programming (S6139)
DB2 for i5/OS SQL Performance Workshop
ibm.com/servers/eserver/iseries/service/igs/db2performance.html

ibm.com/iseries/db2/gettingstarted.html
ibm.com/services/learning

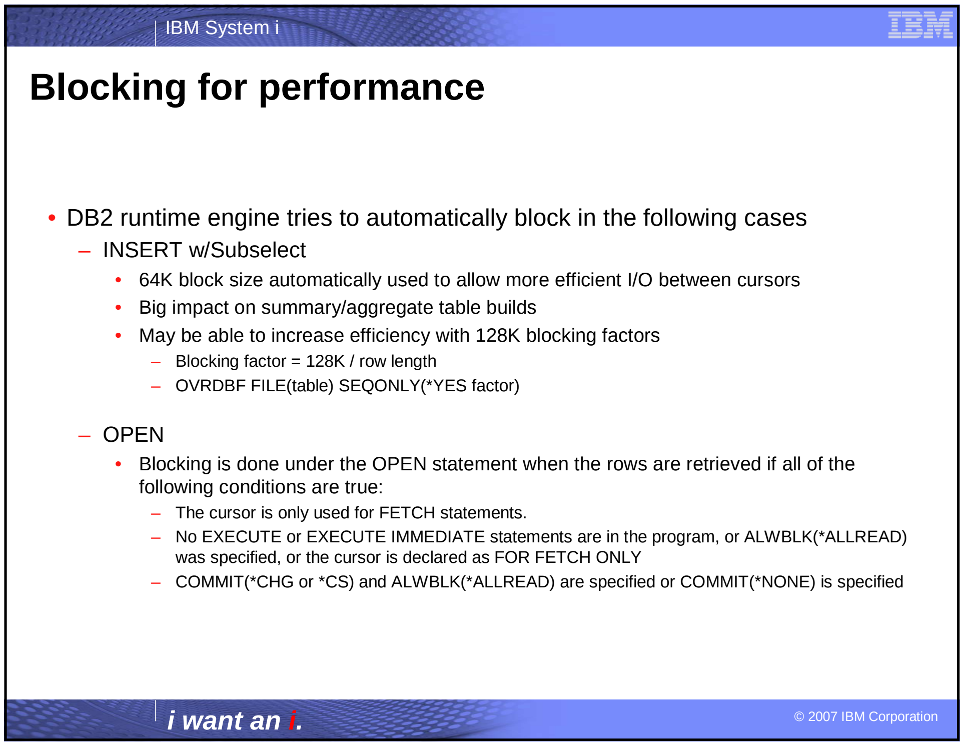
- Self-study iSeries Navigator tutorials for DB2 at:

ibm.com/servers/enable/site/education/ibo/view.html?oc#db2

- Piloting DB2 UDB with iSeries Navigator
- Performance Tuning DB2 UDB with iSeries Navigator & Visual Explain
- Integrating XML and DB2 for i5/OS

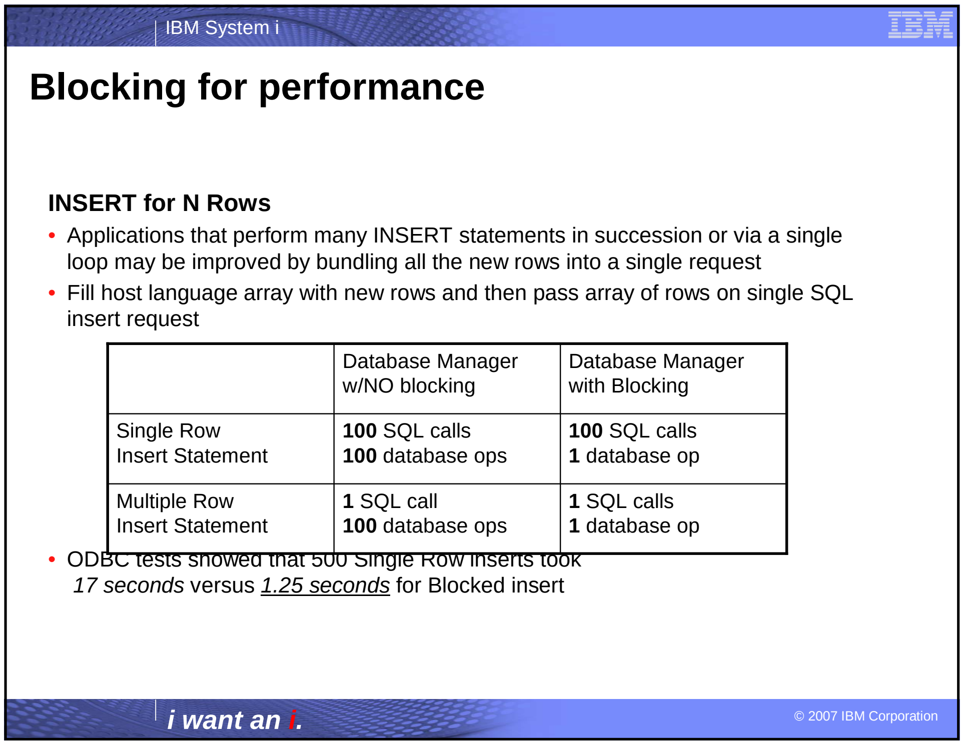


Appendix:
SQL Performance
Best Practices



Blocking for performance

- DB2 runtime engine tries to automatically block in the following cases
  - INSERT w/Subselect
    - 64K block size automatically used to allow more efficient I/O between cursors
    - Big impact on summary/aggregate table builds
    - May be able to increase efficiency with 128K blocking factors
      - Blocking factor = 128K / row length
      - OVRDBF FILE(table) SEQONLY(*YES factor)
  - OPEN
    - Blocking is done under the OPEN statement when the rows are retrieved if all of the following conditions are true:
      - The cursor is only used for FETCH statements.
      - No EXECUTE or EXECUTE IMMEDIATE statements are in the program, or ALWBLK(*ALLREAD) was specified, or the cursor is declared as FOR FETCH ONLY
      - COMMIT(*CHG or *CS) and ALWBLK(*ALLREAD) are specified or COMMIT(*NONE) is specified



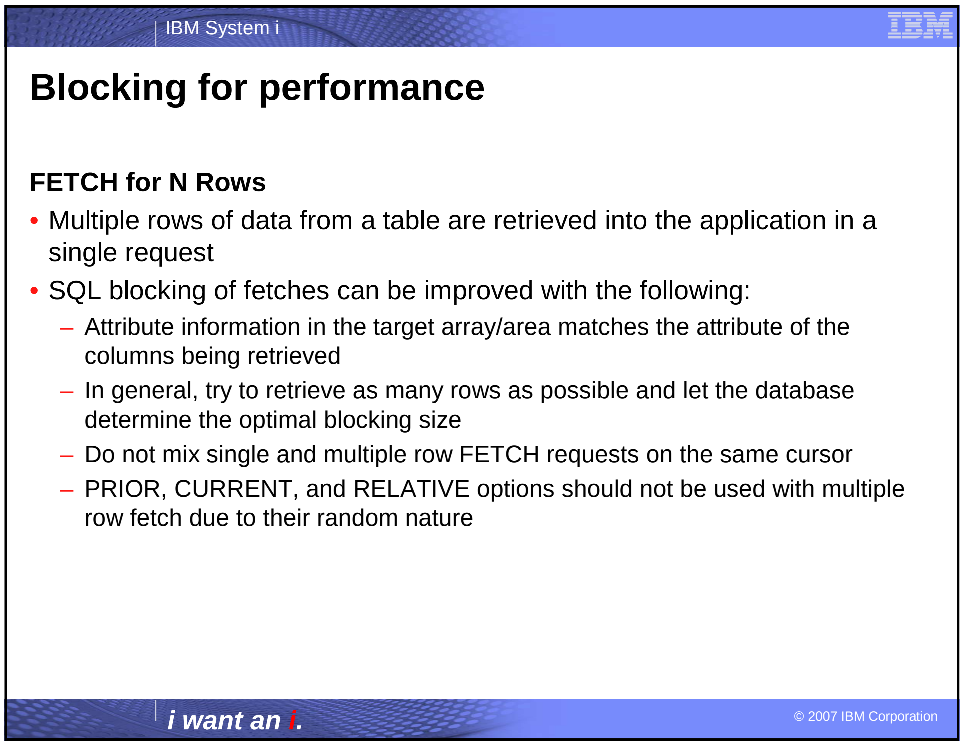
Blocking for performance

INSERT for N Rows

- Applications that perform many INSERT statements in succession or via a single loop may be improved by bundling all the new rows into a single request
- Fill host language array with new rows and then pass array of rows on single SQL insert request

	Database Manager w/NO blocking	Database Manager with Blocking
Single Row Insert Statement	**100** SQL calls **100** database ops	**100** SQL calls **1** database op
Multiple Row Insert Statement	**1** SQL call **100** database ops	**1** SQL calls **1** database op

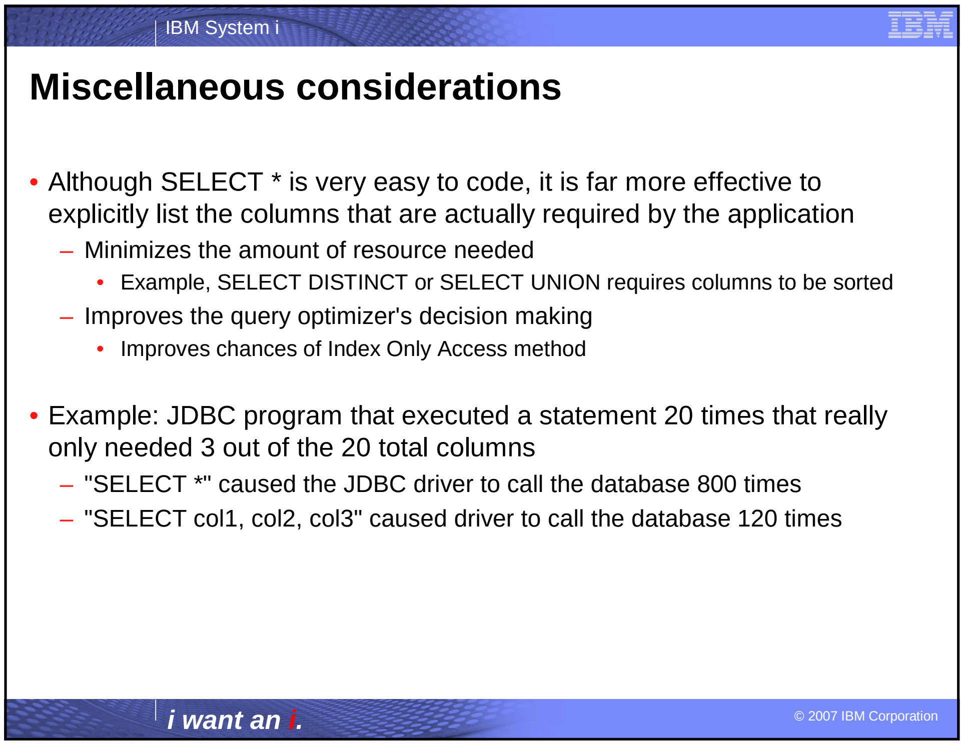
- ODBC tests showed that 500 Single Row inserts took *17 seconds* versus *1.25 seconds* for Blocked insert



Blocking for performance

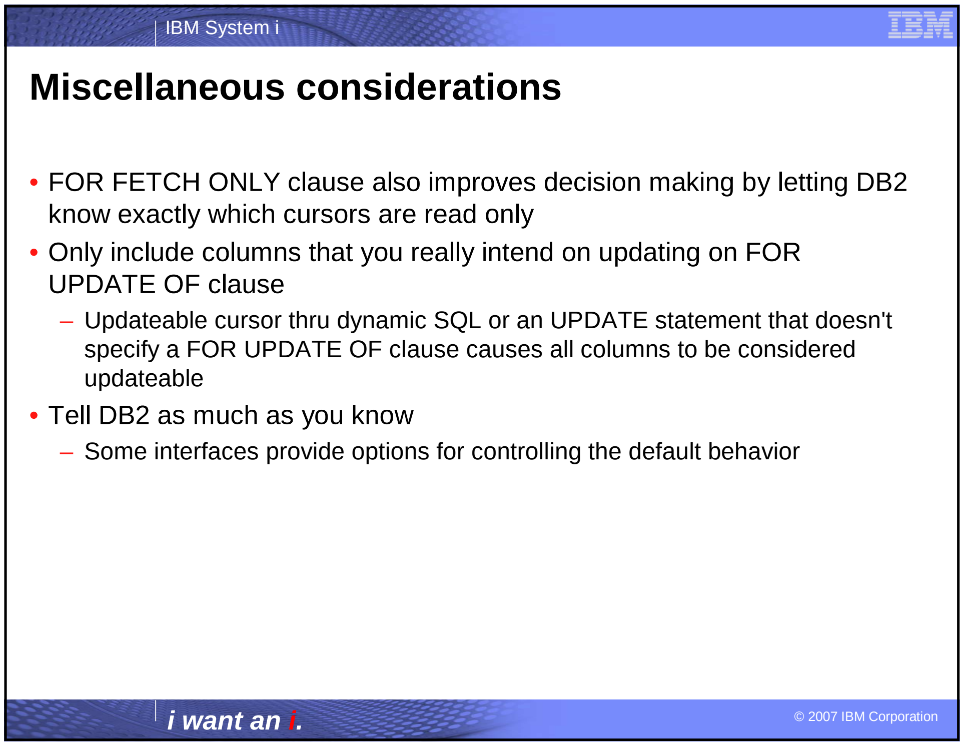
FETCH for N Rows

- Multiple rows of data from a table are retrieved into the application in a single request
- SQL blocking of fetches can be improved with the following:
  - Attribute information in the target array/area matches the attribute of the columns being retrieved
  - In general, try to retrieve as many rows as possible and let the database determine the optimal blocking size
  - Do not mix single and multiple row FETCH requests on the same cursor
  - PRIOR, CURRENT, and RELATIVE options should not be used with multiple row fetch due to their random nature



Miscellaneous considerations

- Although SELECT * is very easy to code, it is far more effective to explicitly list the columns that are actually required by the application
  - Minimizes the amount of resource needed
    - Example, SELECT DISTINCT or SELECT UNION requires columns to be sorted
  - Improves the query optimizer's decision making
    - Improves chances of Index Only Access method
- Example: JDBC program that executed a statement 20 times that really only needed 3 out of the 20 total columns
  - "SELECT *" caused the JDBC driver to call the database 800 times
  - "SELECT col1, col2, col3" caused driver to call the database 120 times



Miscellaneous considerations

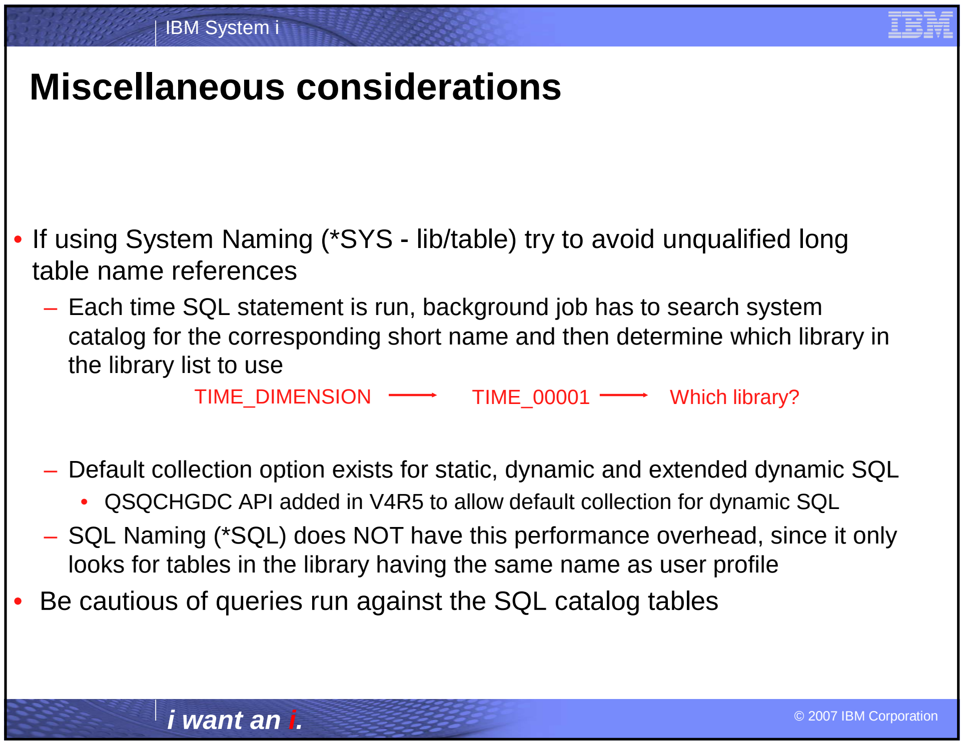
- FOR FETCH ONLY clause also improves decision making by letting DB2 know exactly which cursors are read only
- Only include columns that you really intend on updating on FOR UPDATE OF clause
  - Updateable cursor thru dynamic SQL or an UPDATE statement that doesn't specify a FOR UPDATE OF clause causes all columns to be considered updateable
- Tell DB2 as much as you know
  - Some interfaces provide options for controlling the default behavior

Isolation Level Considerations

- Use lowest isolation level (commitment control) possible in your application
  - The lower the level, the less system resources consumed
  - Avoid Serializable isolation level in concurrent environments, Serializable isolation acquires **exclusive** table locks
- Switching isolation levels can negatively impact ODP reuse if the same SQL statement is executed at different isolation levels
  - Switching to and from the Serializable level is especially problematic

Journal Considerations

- DB2 attempts to journal (log) all SQL created tables automatically
  - Verify that DB2 tables are only journaled when required
- Journals can have a definite impact on SQL performance, so that's another area of investigation when doing database performance analysis. Possible places to start:
  - Journal minimal data option to minimize amount of data copied into the journal and size of the journal object
    - MINENTDTA Option on CRTJRN & CHGJRN CL commands
  - Journal Caching PRPQ (5799-BJC) if running batch jobs with isolation level of No Commit/*NONE
  - HW Configuration: Look for limited Write Cache
  - New Redbook: Striving for Optimal Journal Performance (SG24-6286)



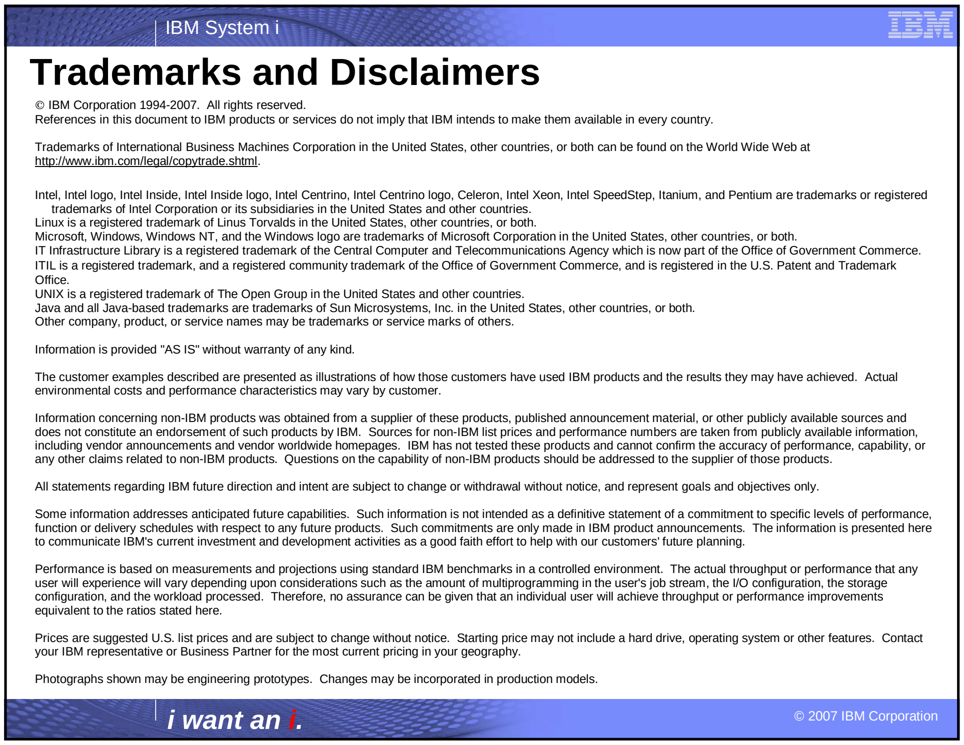
Miscellaneous considerations

- If using System Naming (*SYS - lib/table) try to avoid unqualified long table name references
  - Each time SQL statement is run, background job has to search system catalog for the corresponding short name and then determine which library in the library list to use

    TIME_DIMENSION ⟶ TIME_00001 ⟶ Which library?

  - Default collection option exists for static, dynamic and extended dynamic SQL
    - QSQCHGDC API added in V4R5 to allow default collection for dynamic SQL
  - SQL Naming (*SQL) does NOT have this performance overhead, since it only looks for tables in the library having the same name as user profile
- Be cautious of queries run against the SQL catalog tables

Thank You



Trademarks and Disclaimers