PEOPLETOOLS 8.48/8.49 NEW DATABASE FEATURES

David Kurtz, Go-Faster Consultancy Ltd.

INTRODUCTION

PeopleTools 8.44 was a very significant release; arguably the most important feature to DBA or System Administrator was the instrumentation of PeopleTools to support the PeopleSoft Performance Monitor. Since PeopleSoft's takeover by Oracle, there have been further developments relating to 'Fusion Readiness', integration and messaging.

PeopleTools 8.48 was another very significant release. There have also been some significant changes to incorporate databases features, many relating to how PeopleSoft runs on an Oracle database. PeopleTools 8.49 introduced new third-party product versions[1].

This paper deals with the following areas:

- Changes DDL Models and %UpdateStats
- %Update Stats –v- Oracle Dynamic Sampling
- NOLOGGING
- Column Default Values
- Long Columns, Unicode & CLOBs
- Descending Indexes
- Dirty Read (PT8.49, SQLServer)

There is no overall theme to these sections. Each one can be read in isolation.

[1] And since then BEA has also been acquired by Oracle.

DDL MODELS & %UPDATESTATS

A recent thread on Oracle-L[2] led me to look at how Oracle has changed the way that PeopleTools 8.48 collects Oracle Cost-Based Optimiser statistics. It now uses DBMS_STATS instead of the ANALYZE command. This has also caused me to reconsider some options for managing statistics for a PeopleSoft system.

Application Engine programs can collect Cost-Based Optimiser statistics on specific tables by calling the *%UpdateStats([,high/low]);* PeopleCode macro. This uses one of two DDL models depending on whether the high or low option is specified. However, these DDL models only exist for Oracle and DB2/MVS. *%UpdateStats()* has no function on other platforms.

This was PeopleSoft's platform generic solution (before their takeover by Oracle, and before Dynamic Sampling was available in the Oracle database) to the very real problem that occurs when statistics on a working storage or reporting table, that is emptied, repopulated and used during a batch process, do not accurately represent the content of the table and hence cause the optimiser to choose an inappropriate execution plan. PeopleSoft provided a method of refreshing the statistics during the process, and introduced new DDL models because each database platform would have its own command. However, this approach relies upon developers to add the *%UpdateStats()* macro for every occasion where data has changed sufficiently to require refreshing the statistics. Unfortunately, developers are not always best placed to make that decision. There are still plenty of places in delivered PeopleSoft code where this macro could be usefully added.

Up to PeopleTools 8.47, PeopleSoft delivered two DDL models that used the *ANALYZE* command. The *%UpdateStats(,high)* ran a full compute of the table:

```
ANALYZE TABLE [TBNAME] COMPUTE STATISTICS;
```

While *%UpdateStats(,low)* estimated statistics with the default sample size:

```
ANALYZE TABLE [TBNAME] ESTIMATE STATISTICS;
```

From PeopleTools 8.48, these DDL models now call the Oracle supplied PL/SQL package *DBMS_STATS*. The low option collects statistics on a 1% sample.

```
DBMS_STATS.GATHER_TABLE_STATS (ownname=> [DBNAME], tabname=>[TBNAME], estimate_percent=>1,
method_opt=> 'FOR ALL COLUMNS SIZE 1',cascade=>TRUE);
```

While the high option estimates statistics with the automatic sample size.

```
DBMS_STATS.GATHER_TABLE_STATS (ownname=> [DBNAME], tabname=>[TBNAME], estimate_percent=>
dbms_stats.auto_sample_size, method_opt=> 'FOR ALL INDEXED COLUMNS SIZE 1',cascade=>TRUE);
```

So it would appear that PeopleSoft now follow the recommendations that Oracle have been making since version 8*i* of the database to use *DBMS_STATS* instead of the ANALYZE command. This is certainly a step in the right direction. It also makes good sense to use the automatic sample size. ESTIMATE_PERCENT defaults to *DBMS_STATS.AUTO_SAMPLE_SIZE* from Oracle 10g. Previously it was NULL, which caused a full compute.

However, there are various problems:

- Oracle has swapped over the DDL models. I believe that the automatic sample size was supposed to be in the low option, and a 100% sample was supposed to be used in the high, but there is a typo in the script.
- PeopleSoft have chosen to specify the *METHOD_OPT* as *FOR ALL INDEXED COLUMNS SIZE 1*. If you have specified histograms on any of your columns, or generated them automatically with DBMS_STATS, the PeopleSoft command will remove them from indexed columns and will leave any histograms on unindexed columns unchanged, and potentially out of date.
    - The default in Oracle 9*i* is *FOR ALL COLUMNS SIZE 1*. This removes all histograms on all columns, although this is at least the same behaviour as the ANALYZE command.

[2] http://www.freelists.org/archives/oracle-l/04-2007/msg00959.html

- In Oracle 10g, METHOD_OPT defaults to *FOR ALL COLUMNS SIZE AUTO*. The Oracle manual states that the database '…determines the columns to collect histograms based on data distribution and the workload of the columns'. So, Oracle may remove histograms if it judges that they are not necessary.

I have no hesitation in recommending that value for METHOD the delivered DDL models should be changed. However, I might consider using FOR ALL COLUMNS SIZE REPEAT.

%UPDATESTATS –V- DYNAMIC SAMPLING

When I first wrote the previous section about DDL models, it made me to think about whether %UpdateStats() PeopleCode macro is the best solution to managing statistics on working storage tables in Oracle.

Optimizer Dynamic Sampling was introduced in Oracle 9.0.2. as a solution to the same problem. When a query is compiled Oracle can collect some optimiser statistics based upon a small random sample of blocks for tables that do not have statistics and that meet certain other criteria depending upon the parameter OPTIMIZER_DYNAMIC_SAMPLING. In Oracle 10g the default value for this parameter changed from 1 to 2 and so dynamic sampling applies to ALL unanalysed tables.

Thus, it should be possible to resolve the problem of incorrect statistics on a working storage table without explicitly collecting statistics during an Application Engine program, and therefore without needing a code change to add %UpdateStats(). Instead, simply delete statistics from the table, and lock them. A subsequent GATHER_SCHEMA_STATS will skip any locked tables. When a query references the table it will dynamically sample statistics and use them in determining the execution plan.

However, there is one more problem to overcome. GATHER_TABLE_STATS will raise an exception on a table with locked statistics. If you want to use Dynamic Sampling on a table where %UpdateStats() is already used to update the statistics, the PeopleCode macro will raise an exception that will cause Application Engine programs to terminate with an error. The workaround is to encapsulate GATHER_TABLE_STATS in a procedure that handles the exception, and reference the procedure in the DDL model. It is not possible to put a PL/SQL block in DDL model.

```
CREATE OR REPLACE PACKAGE wrapper AS
PROCEDURE ps_stats (p_ownname VARCHAR2, p_tabname VARCHAR2, p_estpct NUMBER);
END wrapper;
/

CREATE OR REPLACE PACKAGE BODY wrapper AS
PROCEDURE ps_stats(p_ownname VARCHAR2, p_tabname VARCHAR2, p_estpct NUMBER) IS
table_stats_locked EXCEPTION;
  PRAGMA EXCEPTION_INIT(table_stats_locked,-20005);
BEGIN
  IF p_estpct = 0 THEN
  sys.dbms_stats.gather_table_stats
  (ownname=>p_ownname
  ,tabname=>p_tabname
  ,estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE
  ,method_opt=>'FOR ALL COLUMNS SIZE REPEAT'
   ,cascade=>TRUE);
  ELSE
   sys.dbms_stats.gather_table_stats
   (ownname=>p_ownname
   ,tabname=>p_tabname
   ,estimate_percent=>p_estpct
   ,method_opt=>'FOR ALL COLUMNS SIZE REPEAT'
   ,cascade=>TRUE);
  END IF;
EXCEPTION
  WHEN table_stats_locked THEN NULL;
END ps_stats;
END wrapper;
/
```

At this time I have no data to determine which method is more likely to produce the better execution plan. However, when

performance problems occur in production they are instinctively routed to the DBA, who is likely to have difficulty introducing a code change at short notice. Dynamic Sampling has some clear advantages.

NOLOGGING

From PeopleTools 8.48, PeopleSoft builds all indexes with the NOLOGGING option, and then alters them to logged objects. This is done to reduce redo generation during the index build process. It can save time by reducing redo, and can be useful in development environments.

However, this has implications when running Oracle Data Guard. If an operation is not logged on the primary, it will not be replicated to the standby databases.

Here is a simple test. I created a table on the primary database, but the index will be created NOLOGGING, which exactly what PeopleSoft does. I have deliberately set the database to allow NOLOGGING operations

```
ALTER DATABASE NO FORCE LOGGING;

CREATE TABLE t (a number);
INSERT INTO t SELECT rownum FROM dba_objects WHERE ROWNUM <= 1000;

CREATE UNIQUE INDEX t_idx ON t (a) NOLOGGING;
ALTER INDEX t_idx LOGGING;

ALTER SYSTEM SWITCH LOGFILE;
```

Switching the log file forces the changes to be replicated to the secondary database. Now open the secondary database in read only:

```
SELECT /*+ FULL(t)*/ COUNT(a) FROM t;

  COUNT(A)
----------
      1000
```

So there are 1000 rows in the table, but if I force Oracle to count the rows in the index using a hint I get a corrupt block error:

```
SELECT /*+ INDEX_FFS(t t_idx)*/ COUNT(a) FROM t
*
ERROR at line 1:
ORA-01578: ORACLE data block corrupted (file # 1, block # 134419)
ORA-01110: data file 1: '/u01/oradata/hcm89/system01.dbf'
ORA-26040: Data block was loaded using the NOLOGGING option
```

This could happen every time you build an index using a script generated by the application designer. The status on the corrupt index is still VALID, so you would have no way of knowing whether this problem existed before you open your standby and run on it (because you can't run PeopleSoft on a standby database in read only mode).

So how can you prevent this from happening in the first place?

You could alter the DDL models to remove the NOLOGGING option from the CREATE INDEX DDL model. But the same argument applies to any NOLOGGING operation performed in the database. You are likely to get corrupt objects in your standby database. Although, I know of no other NOLOGGING operations in a vanilla PeopleSoft database, if any are introduced during development then DBAs are unlikely to be able to change them.

It was for exactly this reason that Oracle provided the ability to force NOLOGGING operations to generate redo

information. It is set with the following command:

```
ALTER DATABASE FORCE LOGGING;
```

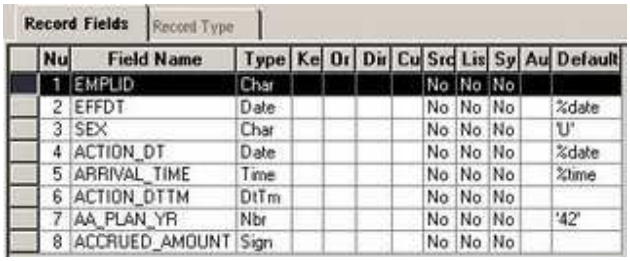
Now the database will log all operations regardless and all objects and operations will be fully replicated to the standby database. However, nothing is free. NOLOGGING is often used to improve performance by reducing redo, for example when populating working storage tables which will never need to be recovered. These performance improvements will be lost when logging is forced.

COLUMN DEFAULT VALUES

PeopleSoft's Mobile Synchronization Framework is designed to support off-line access to a subset of the PeopleSoft application. Application Designer has been able to generate database triggers to maintain Timestamp and System ID fields on tables that support synchronization of mobile clients since PeopleTools 8.44. In 8.48 it is now possible to set column defaults on character and numeric fields via Application Designer. However, this has potential for much wider application in PeopleSoft applications.

There are two new checkboxes on the record properties dialogue box.

If the User -> Server box is checked, Application Designer will add column defaults to the fields in the generated create and alter table DDL. As an example, I created the following record in Application Designer.



Nu	Field Name	Type	Ke	Or	Dir	Cu	Src	Lis	Sy	Au	Default
1	EMPLID	Char					No	No	No		
2	EFFDT	Date					No	No	No		%date
3	SEX	Char					No	No	No		'U'
4	ACTION_DT	Date					No	No	No		%date
5	ARRIVAL_TIME	Time					No	No	No		%time
6	ACTION_DTTM	DtTm					No	No	No		
7	AA_PLAN_YR	Nbr					No	No	No		'42'
8	ACCRUED_AMOUNT	Sign					No	No	No		

And this was the DDL that Application Designer generated.

```
CREATE TABLE PS_DMK (EMPLID VARCHAR2(11) DEFAULT ' ' NOT NULL,
EFFDT DATE,
SEX VARCHAR2(1) DEFAULT 'U' NOT NULL,
ACTION_DT DATE NOT NULL,
ARRIVAL_TIME DATE NOT NULL,
ACTION_DTTM DATE,
AA_PLAN_YR SMALLINT DEFAULT 42 NOT NULL,
ACCRUED_AMOUNT DECIMAL(15, 2) DEFAULT 0 NOT NULL)
…
```

- All of the numeric and character columns now have default values, but none of the date columns have defaults.
- If a literal value is specified in Application Designer it is carried forward to the DDL, otherwise a character field defaults to a single space and a numeric field to zero.
- Whether a field is required or not does not affect the default value.
- It is a pity that the system variables to specify current date or time do not get converted to SYSDATE in the DDL, thus

```
…
ACTION_DT DATE DEFAULT TRUNC(SYSDATE) NOT NULL,
ARRIVAL_TIME DATE DEFAULT SYSDATE NOT NULL,
…
```

I assume that this is because it is platform specific, but then so is the rest of the field list.

There is another problem to using the Up Sync check box. When checked, you are prevented you from making a field required. The unchecked box is greyed out, but fields that are already required remain so.

I think that the ability to specify column defaults could have advantages during batch processing. All numeric and character columns in PeopleSoft (with minor exceptions) are NOT NULL in the database. Thus, if a process is inserting a row into that table, it must provide a value for every mandatory column. Otherwise, you generate an Oracle error.

```
ORA-01400: cannot insert NULL into ("SYSADM"."table name"."field name")
```

Specifying a default value will prevent this error. It would also save developers from having to add this default values to their code, thus simplifying development and maintenance.

Many batch processes in PeopleSoft process working storage or reporting tables. Often they insert rows, filling in key and some other columns, and then update other columns later on. If these columns have defaults they do not need to be referenced in the insert statement.

When a new column is added to a table, it is necessary to find every place where a row is inserted, and add a value. Specifying a column default would save adding a space or zero to those inserts.

LONG COLUMNS, UNICODE & CLOBS

Unicode and the use of Long columns in PeopleSoft have been subjects that I have grumbled about in the past. However, Oracle now appear to have addressed them in PeopleTools 8.48. Unfortunately I cannot find any guiding documentation. I stumbled across these changes by accident. However, if they work, they are both things that customers should know about.

PeopleSoft deliver two scripts in the PT8.48 distribution in *%PS_HOME%/scripts*: *upgradedboptions_enable.sql* and *upgradedboptions_disable.sql*, but I cannot find any documentation.

They set PSSTATUS.DATABASE_OPTIONS to 2 and 0 respectively. This one setting controls both features.

UNICODE

I found the following in the platform advisory note: Operating System, RDBMS & Additional Component Patches Required for Installation on PT 8.48 under RDBMS Patches for 10g[3].

"Note#3: Required INIT.ORA Parameters for building 'Unicode' databases with PT8.48 and Enterprise application releases 9 or later... Note. If it is your intention to create a Unicode DB for an application release 9 or later database, then the following init.ora parameter is mandatory. The PeopleSoft Unicode implementation for PT8.48 no longer triples the VARCHAR2 datatype columns. Instead we rely on an Oracle feature called CHARACTER LENGTH SEMANTICS. This parameter is not needed for non-unicode DB's or for pre-9 unicode application databases.

```
NLS_LENGTH_SEMANTICS=CHAR"
```

However, this advice is not in the same note for PeopleTools 8.49. Tests have shown PeopleTools 8.48 no longer trebles the length of string columns if PSTATUS.UNICODE_ENABLED = 1 and PSSTATUS.DATABASE_OPTIONS is set to 2 (I haven't tested 8.49 yet, but the installation guide now says that you now 'must' use Character Semantics for Unicode databases).

This change will be a huge improvement for PeopleSoft customers who need Unicode support. The length checking constraints can dramatically increase the overhead of parsing SQL in Oracle (I have seen as much as a 300% increase in parse time - see my Unicode Oddity presentation[4]).

LONG -V- CLOB COLUMNS

If PSSTATUS.DATABASE_OPTIONS is set 2, columns that would normally be created as type LONG, are now created as CLOBs.

Use of CLOBs has a number of advantages.

- It is not possible to partition a table with a LONG, but it is permitted with a CLOB.
- It permits the use of SQL string functions on CLOBs that cannot be used on LONGs
- It is possible to move data in CLOBs across database links (or at least first 32K).
- In Oracle 9i, it was not possible to use Streams to replicate a table with a long column. This restriction was removed in 10g. However, most PeopleSoft customer upgrading to PeopleTools 8.48 will also take the opportunity to upgrade to Oracle 10g

[3] http://www4.peoplesoft.com/psdb.nsf/6285c9ef6101571388256fdb0074e082/33440ec2de7c886788257051005aeb72?OpenDocument

[4] http://www.go-faster.co.uk/unicode_oddity.pps

EXAMPLES

The settings in these fields on the table PSSTATUS control the generation by Application Designer and Data Mover of the column list *[TBCOLLIST]* in the create table DDL model.

PSSTATUS. DATABASE_OPTIONS	PSSTATUS. UNICODE_ENABLED	Create Table DDL

0	0	`CREATE TABLE PS_DMK`
		`(EMPLID VARCHAR2(11) NOT NULL`
		`,EMPL_RCD SMALLINT NOT NULL`
		`,EFFDT DATE`
		`,DESCRLONG LONG VARCHAR) ...`

0	1	`CREATE TABLE PS_DMK`
		`(EMPLID VARCHAR2(33) NOT NULL`
		`CHECK(LENGTH(EMPLID)<=11)`
		`,EMPL_RCD SMALLINT NOT NULL`
		`,EFFDT DATE`
		`,DESCRLONG LONG VARCHAR) ...`

2	either 0 or 1	`CREATE TABLE PS_DMK`
		`(EMPLID VARCHAR2(11) NOT NULL`
		`,EMPL_RCD SMALLINT NOT NULL`
		`,EFFDT DATE`
		`,DESCRLONG CLOB) ...`

DESCENDING INDEXES ARE BACK!

In Application Designer, you can mark a key field as descending. When that record is used in a scroll on a page, PeopleTools automatically orders the query generated to populate the scroll in key order, and that includes specifying descending order for descending keys. The JOB record in HR is a good example.

On entering the JOB component, the PIA generates the SQL to retrieve the job history for a particular employee. The two known keys are specified in the where clause, but all the keys were used to generate the ORDER BY clause. EFFDT and EFFSEQ are descending keys on the JOB record, so the query is in descending order on those columns.

```
SELECT EMPLID,...
FROM PS_JOB
WHERE EMPLID=:1
AND EMPL_RCD=:2
ORDER BY EMPLID, EMPL_RCD, EFFDT DESC, EFFSEQ DESC
```

The descending key attribute has another purpose. Prior to PeopleTools 8.14, if you specified a key field as descending in Application Designer, that column would also be indexed in descending order where it appeared in indexes. However, PeopleSoft removed this functionality because of Oracle Bug 869177 which in most versions of Oracle 8.1 could cause ORA-3113 when accessing the index.

Now that Oracle has fixed the bug in the database, PeopleTools once again builds indexes with descending columns on Oracle RDBMS from PeopleTools 8.48.

However, as always, there is a snag (or I wouldn't have bothered to write this). The name of the descending column is not reported in catalogue views. Instead the column name is replaced with a system generated column name.

```
SELECT table_name, index_name, column_position, column_name, descend
FROM   user_ind_columns
WHERE  index_name = 'PS_JOB'

TABLE_NAME INDEX_NAME COLUMN_POSITION COLUMN_NAME  DESCEND
---------- ---------- --------------- ------------ ------------
PS_JOB     PS_JOB                   1 EMPLID       ASC
PS_JOB     PS_JOB                   2 EMPL_RCD     ASC
PS_JOB     PS_JOB                   3 SYS_NC00163$ DESC
PS_JOB     PS_JOB                   4 SYS_NC00164$ DESC
```

This could cause problems with SQL scripts that use the catalogue views. For example, I have had to change my DDL Trigger to protect database objects not managed by Application Designer[5].

So how do you find the name of the underlying column? You have to look at *column_expression* in another view, *user_ind_expressions*. It can be joined to *user_ind_columns* by *column_position*, thus

[5] http://blog.psftdba.com/2006/04/using-ddl-triggers-to-protect-database.html

```
column table_name format a20
column index_name format a25
column column_name format a25
column column_position format 999 heading 'Col|Pos'
column column_expression format a25
column uniqueness format a1 heading 'U'
break on report on table_name on index_name on uniqueness skip 1
SELECT uic.table_name, uic.index_name
,      SUBSTR(ui.uniqueness,1,1) uniqueness
,      uic.column_position
,      uic.column_name
,      uic.descend
,      uie.column_expression
FROM   all_indexes ui
,      all_ind_columns uic
       LEFT OUTER JOIN all_ind_expressions uie
       ON uie.table_owner = uic.table_owner
       AND uie.index_owner = uic.index_owner
       AND uie.table_name = uic.table_name
       AND uie.index_name = uic.index_name
       AND uie.column_position = uic.column_position
WHERE  uic.table_name = UPPER('&1')
AND    ui.owner = uic.index_owner
AND    ui.index_name = uic.index_name
AND    ui.table_owner = uic.table_owner
AND    ui.table_name = uic.table_name
ORDER BY 1,2,3,4
/
```

```
TABLE_NAME           INDEX_NAME               U  Pos COLUMN_NAME              DESC COLUMN_EXPRESSION
-------------------- ------------------------ - ---- ------------------------- ---- -------------------------
PS_JOB               PS_JOB                   U    1 EMPLID                    ASC
                                                   2 EMPL_RCD                  ASC
                                                   3 SYS_NC00214$              DESC "EFFDT"
                                                   4 SYS_NC00215$              DESC "EFFSEQ"
```

My thanks to Noons for pointing out that you can get the underlying column or expression from DBA_IND_EXPRESSIONS.

DIRTY READ (PT8.49, SQLSERVER)

In PeopleTools 8.49 you can configure the application server to use Dirty Reads on SQL Server and DB2/OS390.

Lets start with a quick review of database isolation levels. If you run PeopleSoft on an Oracle database, you hardly ever have to worry about isolation levels, because Oracle defaults to *Read Committed*, where all transactions are transactionally or read consistent, and that doesn't bring any scalability problems.
(*In writing this note, I have spent a fair amount of time reading various Oracle[6] and SQL Server[7] websites. It has been excellent revision!*)

What is read consistency? Well, if I start a long-running query at say 09.00, and then I update some data at 09.01 and commit that update, and then at 09.02 the query comes to the data that was changed at 09.01, it will report the data as at 09.00. So the set of data that is retrieved by the long running query is consistent with itself.
On Oracle, this is done by keeping information to reverse the change, essentially the old data values, in the undo (or rollback) segments from which it can construct, in memory, a read consistent version of data blocks as the start of the transaction. So at 9.02, Oracle creates a copy of the block in the buffer cache and uses the undo information to roll it back to the state it was in when the query began at 9.00, and then it reads the data from that read consistent copy.

Thus, Oracle is capable of maintaining read consistency without locking, so that writers never block readers, and readers never block writers.

Oracle also supports two other isolation levels:

Serializable: queries cannot see changes committed after the transaction began, but before the current query began.

Read-Only: as *Serializable* but no INSERT, UPDATE, or DELETE statements are permitted.

Oracle has not explicitly implemented Repeatable Read as an isolation level, but the same effect can be achieved if you use *SELECT ... FOR UPDATE*.

However, the other database platforms that are certified for PeopleSoft have additional isolation levels that can be set at database and session level (and hence in the application). In SQL Server the isolation modes are:

Read Uncommitted (also known as *Dirty Read*): Queries simply read whatever is in the data blocks, so it is possible to read changes in the database that have not been committed and that may potentially be rolled back!

Read Committed (SQL Server default): Uncommitted changes cannot be read, queries can be blocked by shared locks created by updates unless using Read Committed Snapshot, or other transaction in higher isolation levels.

Read Committed Snapshot (available SQL Server 2005): row versioning is used to produce a transactionally consistent snapshot of the data. This is effectively equivalent to *Read Committed* on Oracle.

Repeatable Read: Locks are placed on all data that is read.

Serializable: Locks are placed on all data that is read, and rows cannot be inserted ranges of data queried by other statements.

Snapshot: Changes made after the start of the transaction cannot be seen.

---

[6] *Oracle Database Concepts Guide: 13. Data Concurrency and Consistency:*
http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/consist.htm#CNCPT1312

[7] SQL Server 2005 Books Online, Language Reference, Transact-SQL Reference, SET ISOLATION LEVEL
http://msdn2.microsoft.com/en-us/library/ms173763.aspx

PeopleSoft running on Microsoft SQL Server 2000 and Sybase has always had scalability problems with high concurrency, because read consistency is achieved by use of locking. This is addressed in SQL Server 2005 with the *READ_COMMITTED_SNAPSHOT* option, but this version of the database is not certified below PeopleTools 8.47.

DIRTY READS IN PEOPLETOOLS 8.49

Dirty-read mode can be specified for the application server:

```
[PSQRYSRV]
;======================================================================
; Settings for PSQRYSRV
;======================================================================
...
; Use dirty-read(uncommitted read) for PSQRYSRV only on DB2/OS390 or SQL Server
Use dirty-read=0
```

It can also be set for scheduled queries run by Application Engine in the Process Scheduler:

```
[PSAESRV]
...
;-----------------------------------------------------------------------------------------
; Setting to control dirty read for Scheduled PSQueries.
ScheduledQuery-DirtyRead = 0
```

This is what Enterprise PeopleTools 8.49 PeopleBook: System and Server Administration[8] says:

"**Use Dirty-Read**: Enter 1 to enable the application server to read uncommitted data from a table. Enter 0 to disable dirty reads.

This parameter can be used for general reporting and PeopleSoft Query. You can run dirty read queries through the application server, the Process Scheduler, and in a two-tier connection. The Use Dirty-Read setting in the application server configuration controls the behavior of PSAPPSRV, PSQCKSRV, and PSQRYSRV.

Note. *Dirty reads are not recommended if you are reading data and doing subsequent processing based on the disposition of the data at the time that it is read. Between the time the data is read by a subsequent process and the time the unit of work is completed by the first process, any activity affecting the table data at the time a subsequent process read could be rolled back, invalidating the accuracy of the data that a subsequent process read."*

However, the parameter only appears in the query server section (*PSQRYSRV*) of the application server configuration file (*psappsrv.cfg*), so I suspect that it only applies to *PS/Query* queries (and therefore *Crystal* and *BI Publisher* reports) and *nVision* reports. Although, I haven't yet found confirmation of that in PeopleBooks or other documentation.

The delivered comment in the configuration file says that dirty-reads apply to DB2/OS390 and SQL Server, but the documentation doesn't mention which databases it applies to. Dirty-reads do not occur in Oracle, but this isolation level is also available on Sybase and Informix.

CONCLUSION

You can get around the locking problems in SQL Server 2000 and DB2, but at a price. Your reports may not show you entirely accurate data! But, if you are running PeopleTools 8.49 on SQL Server 2000, I would question why you are not running SQL Server 2005.

[8] http://download.oracle.com/docs/cd/E05317_01/psft/acrobat/pt849svt-b0307.pdf