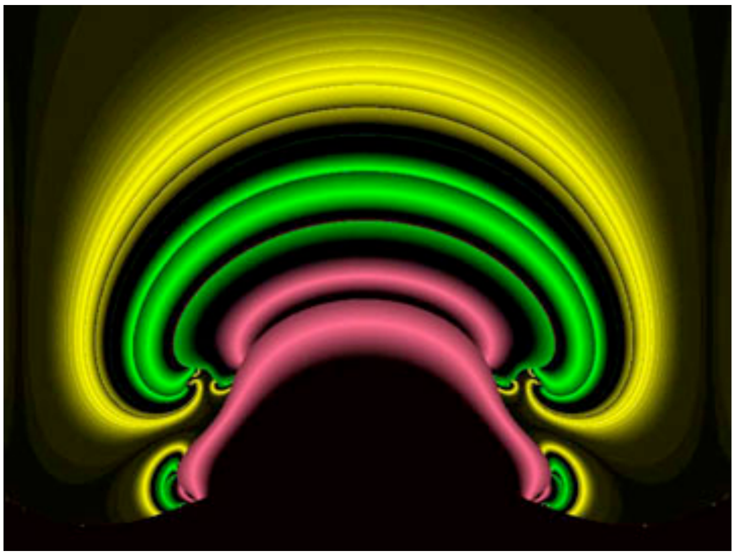
Oracle Facts 2



Ever wonder why Oracle compares its performance with Microsoft SQL Server but never with Sybase? While Oracle may be 14% faster than Microsoft SQL Server, Sybase is 20% to 3000% faster than Oracle.

Do you want to prove that? Go grab Data Direct Technologies ODBC drivers and run psOTestThreads and see the difference for yourself.

Oracle's omission of Sybase in comparative ads is a marketing ploy. Oracle does not want you to know about Sybase. If Oracle does not mention Sybase, then either you will think that Sybase does not exist, does not have the technical ability to compete, or that Sybase is not worth mentioning because of its market share. These are silent lies.

Download the Microsoft Word document containing these Oracle facts.

32	Oracle will log every row in the index to its rollback segments. Oracle uses rollback segments for everything without exception, and even for objects in the temporary tablespace. An index will be logged twice--one for the rows being generated in the permanent tablespace and again for rows being temporary written to in the temporary tablespace. All versions of Oracle use rollback segments. It is just that if your are running Oracle 9i or 10g, Oracle still uses rollback segments behind the scenes even if you are using locally managed tablespaces.
33	SQL*Plus will not inform you if you have run out of rollback segment while creating an index. It will just hang forever. You have to suspect something is up. You have to continuously view the alert log. *tail -f alert.log* . . . is normal for Oracle.
34	Point 34 used to slam Oracle's inability to create indexes based on dictionary order. However, in the light of Oracle's function-based indexes, this is not a concern. For review, Sybase users can set up their servers to process search clauses, order by, and case senstivity by setting up server-wide collating sequences. While Oracle doesn't handle that type of in-built automation, Oracle does support function-based indexes. Though function-based indexes require more syntax and are probably never needed aside from collating sequences, you could say that Oracle beats Sybase in flexibility for this one aspect.
35	In 8i, the network configuration assistant program (netca) on Solaris will erase your entire tnsnames.ora and listener.ora files when you tell it to "Cancel and discard your changes." To the Sybase user, that is like quitting sybsetup or asecfg and having it erase the entire interfaces file. Oracle makes these blunders in most releases. In 9i, Oracle shipped UNIX installation CDs that wouldn't install. The 1st CD wouldn't release the CD drive to allow you to insert the 2nd CD. Also, due to a global memory contention, you couldn't install Oracle if a previous version was already running. In 10g, Oracle uses a 9i server to store enterprise management data. All really huge problems and/or really stupid.
36	The create database command does not load all the necessary SQL scripts to make the database a database. In Sybase, the equivalent is issuing a create database command and finding that the created database does not have system tables.
37	In Oracle, a database *is* a server. Oracle only supports one database per server. For each database you want in Oracle, you must run the equivalent of asecfg; that is, dbassist. So, a Sybase DBA who is used to managing 5000 databases on a single server is going to have to run 5000 iterations of dbassist to create his 5000 databases. Now dbassist is a trip in itself. Note that when dbassist generates a script to create the database, the script does not do any error checking. You can literally get thousands of errors and Oracle will say, "Database successfully installed." Remember that DDL errors are

normal for Oracle. If you program Oracle, you do not even try to make your program error free because true DDL error handling is not part of Oracle's "architecture." Was the database installed correctly? Of course not.

38 Note that dbassist is incompatible with itself . After you ask dbassist to clean up after an unsuccessful generated script run, dbassist not only removes the database, but also removes the directory structure that your script initially requires. The script you just generated after spending an hour answering dbassist questions, no longer works. You have no choice but to answer all the questions again, even if you answer them the same way.

. Listen to *Willie Taylor* sung by Julee Glaub.

39 You cannot tell Oracle to use a specific rollback segment for an import. If you are importing a large table with a large index, you must first take all the small rollback segments offline so that Oracle must choose the large rollback segment when doing the import. Remember, Oracle even uses rollback segments to create an index. While you can bust up the import of the table data into smaller transactions with a COMMIT=Y, you cannot bust up the index creation into separate smaller transactions. (This index transaction limitation does not exist in Sybase since Sybase doesn't log the individual rows created during a create index.)

40 You cannot create an index organized table from a heap table or vice-versa. In Sybase lingo, you cannot create or drop a clustered index on an existing table. This has awful and severe consequences. When you need to import a lot of data into your clustered index table, it is always much faster (up to 50x faster) to import the data without the clustered index being present, even if it means the table being recopied. In Oracle, you are stuck. You cannot change an index organized table to a heap table for loading. Therefore, the load will take forever. The work-around is to create a non-clustered index instead. That will effectively double your table's lookup time. In addition, if your index contains most of the columns in your table, then you have doubled the size of your table as well.

41 Here's a major Oracle kludge: The LDAP server. The LDAP server is nothing more than a super thick carpet in order to cover up the severe problem that all users in an Oracle database cannot readily share the tables of another user. There is no concept or equivalent of 'dbo' in Oracle. In Sybase, every user in a Sybase database sees the database's dbo's schema. The dbo's schema is common to all users in a table, and is an addition to the user's own schema. As a poor attempt to provide dbo-like functionality, Oracle introduces the Band-Aid concept of synonyms. If you get tired of creating 10,000 synonyms for the 10,000 objects in your database, Oracle introduces an additional Band-Aid called the LDAP server. The LDAP server is a nightmare in itself.

42 Oracle allows you to create a user name with a "." (dot) in it. Dots are the ANSI standard SQL character to separate the parts of a fully qualified object name. For example, "user.mytable". So, Oracle's username's embedded dots conflict with the ANSI standard. If SQL*Plus runs into any dotted user names, it core dumps. FYI, Sybase does not let you "sp_addlogin" a username with a dot in it.

43 Which brings up another limitation of Oracle. In Sybase basic architecture, a fully qualified object name is: server.database.owner.object. In Oracle, a fully qualified object name is: owner.object. Oracle's legacy architecture has no concept of objects belonging to different databases. The Oracle fan will say "But you can query tables from remote servers in Oracle!" And they are right. You do it with database links. The format is owner.object@databaselink. Oracle introduced a syntax inconsistency to get around their shortcoming. Oracle fans do not see the problem with this syntax because they are use to it. However, Sybase and SQL server users are thrown off by the inconsistency and become wary of the implied limitation.

44 I mentioned this one in passing before: You cannot do "select col1, col2 . . . coln from mytable" in a stored procedure. In other words, Oracle stored procedures *cannot* return a result set. The Sybase user is instantly horrified and rightly so. Returning result sets from procs is so useful, common and easy that it is taken for granted in Sybase. But in Oracle, the way to simulate a row result set is fodder for PhD dissertations. The problem is formidable. Hordes of white papers have been sacrificed on this unholy alter. For giggles, go to google.com. Search on ODBC, Oracle and Stored Procedures. Look at all the material on the subject

The solution is this: You *pass a reference cursor as a parameter* to the procedure and then use a client-side fetch on the cursor. Your stored procedure also must be bundled in an Oracle *package* so that you can *type define* your cursor. The ODBC developer realizes, "Hey! There is no ODBC parameter type SQL_REFERENCE_CURSOR. I can't pass a parameter that's a reference cursor. " The response is, "Your ODBC driver vendor must provide the reference cursor mechanism inside the driver itself . . . and hide it from the ODBC layer." The company Merant does such a thing. Hats off to them. Their ODBC drivers provide this mechanism under the covers. You still have to go through the considerable pain of setting up these special stored procedures,. But Merant has given one the avenue to simulate result sets with Oracle. In DBPowerSuite, I have included a couple of PL/SQL script examples of how to simulate row result sets in an Oracle proc *given* that the user is using Merant ODBC drivers.

You don't even think about result sets in Sybase. Result sets are a natural feature of the territory. Just like a sunrise, one takes it for granted in Sybase. However, if you use Oracle, the sun does not rise. You have to invent the sun and a rotating earth.

Program managers beware. Your database developers and DBAs will spend three times as long accomplishing a task in Oracle than they will in Sybase. That is normal for Oracle. Your developers must invent suns and rotate earths on a daily basis in Oracle. *You will not find Oracle lacking functionality, but you will find its functionality convoluted, illogical, counter-intuitive, missing the mark and buried.* Oracle's functionality comes this way because Oracle's architectural foundation is not adequate.

. Listen to *Two Eyes* by Mike Collins.

45 Oracle is in conflict over the meaning of ';' -- the semicolon. In SQL*Plus, the semicolon executes your command but in PL/SQL means the end of a SQL statement without executing it. To execute a PL/SQL inside SQL*Plus, you type '/' instead. This is a fundamental overlap conflict between execution commands and language syntax. It is a basic architectural flaw which one faces every day and hour when using Oracle. It drives everyone crazy, even Oracle aficionados. The Sybase user, who has grown accustomed to using good software, vomits at this behavior of Oracle. The Oracle user, who has grown accustomed to flaws and quirks, treats this as normal. In Oracle, one

	gets used to such bad things and gets desensitized after a while.
46	The import utility captures ctrl-C. Say you have 3000 tables in your dump. You get 10 tables into it and for whatever reason, you want to quit. You must type ctrl-C 2990 more times before import will exit. Consider kill -9.
47	Oracle's UPDATE and DELETE statements have no FROM clause. Therefore, you *cannot join tables* to restrict the set of rows you are updating. Instead of being consistent with the SELECT statement like Sybase, Oracle forces you to be inconsistent. It's work-around time. You now have to figure out a way to simulate a *join*. You have to think *subqueries*. Oracle forces you to write a non-intuitive subquery for each column you are setting. If you are setting 3 columns, you have to write 3 subqueries--even if your subqueries are all the same. That's 3 times the processing. The resulting syntax is so convoluted, you will have to refer to examples each time you try to do it. This single Oracle problem will cost your company thousands of dollars both in writing the original statements, and then maintaining them.
48	Oracle lets you create a temporary table, but then forbids you to use a storage clause. There is no reason for this. It is an annoyance which requires expensive work-arounds. For an app I wrote, I needed to create and populate a large temporary table. I declared a big minextentsso that the app would not abort hours later in case of insufficient space. I made the table temporary so that the table would automatically go away if the database connection abnormally went away. But Oracle forbids me to declare minextents for a temporary table. In order to get around this problem, I had to create permanent tables, and write a stored procedure which drops such permanent tables on demand. Another employee of the company had to write a GUI interface to stored procedure so that the customer could maintain his own database. You realize these problems never occur in Sybase?

Home | Buy Software | Privacy