Another Technology Report
From Ken North Computing, LLC

Benchmarks of SQL Query Performance for ODBC and Oracle Call Interface

October 2002

Prepared by Ken North
ken_north@compuserve.com

Benchmarks of SQL Query Performance for ODBC and Oracle Call Interface

Table of Contents

1.0 Background

Many applications for the enterprise, the Web and the desktop use SQL databases. The widespread adoption of SQL databases has put a spotlight on SQL application programming interfaces (API) and performance issues related to SQL APIs.

Because SQL is declarative and non-procedural, application performance does not depend on techniques used to navigate a database. SQL application performance is instead a function of several factors, including database design, network latency, and query optimization. Another factor is the influence of constituent software such as components and libraries. Poorly tuned queries and sub-optimal coding techniques often contribute more to performance problems than the choice of SQL API. Nonetheless, the decision whether to use a proprietary (native) API or Open Database Connectivity (ODBC) is the subject of much debate.

ODBC has been a *de facto* multi-database programming standard since the mid-1990s. In 1994, the ODBC Software Developer Kit was ported to Unix and other operating platforms. In 1995, there was an alignment of ODBC 3.0 with the SQL Call-Level Interface standard (SQL/CLI).

1.1 ODBC Benchmark Tests

ODBC has been put under the microscope on more than one occasion. In October 1995, Oracle published the results of benchmarks it ran when evaluating ODBC for use with the Oracle Open Gateway product. The tests used an Open Gateway that connected to an Oracle database without using ODBC and a second with ODBC. Results showed that in 80% of the tests the performance difference was less than 5% and in the remaining 20% of tests the difference was less than 10%. [1]

[1] "Integrating Enterprise Data Using ODBC", October 1995, Bert Simonis (Oracle Mainframe and Integration Technologies white paper)

In August 1995, Ken North conducted SQL API performance tests using the APIBench application with ODBC drivers and native APIs for Informix, Oracle, and Sybase. The Informix tests measured comparative performance of ODBC and embedded SQL. The Sybase tests compared ODBC and CT-Library performance. The Oracle tests measured ODBC and Oracle Call Interface performance. In 29 tests, native performance was better in only 12 instances, and in only two of those instances the advantage was greater than 10%. [2]

Neither of those benchmark studies produced evidence of the mythical ODBC performance penalty.

2.0 Valid Performance Metrics

The lack of understanding of how to measure ODBC performance has been a persistent problem since the days of ODBC 1.0. Developers often fail to account for the performance effect of software that's an intermediary between their application code and the SQL API.

Developers evaluating SQL APIs have not always understood the efficacy of data access libraries, components, and frameworks such as .NET. When attempting to measure ODBC performance, developers have sometimes used test cases involving data-aware controls, database engines, object layers and class libraries. Software such as the Borland Database Engine, Microsoft Jet, Data Access Objects (DAO), ActiveX Data Objects (ADO), Remote Data Objects (RDO) and Microsoft Foundation Classes raise the level of abstraction when writing data access program. They simplify programming but they are part of the application performance equation.

[2] "Comparative Performance Tests of ODBC Drivers and Proprietary Database Applications Programming Interfaces", August 3, 1995, Ken North (Resource Group technology report)

ODBC applications use multiple tiers of software. The ODBC Driver Manager fields function calls before invoking an ODBC driver. The Driver Manager is part of the performance equation but its performance effect is negligible and less important than the quality of ODBC drivers. First-class ODBC drivers enhance performance by techniques such as pre-fetching rows and using static SQL for some queries.

To do an “apples to apples” comparison of ODBC and native APIs, it’s necessary to use the API directly without encapsulating it in intermediary software. For the tests reported here, the APIBench and APIBench-B software used direct function calls to ODBC and the Oracle Call Interface (OCI). It did not use intermediary software to wrap or encapsulate the SQL APIs.

3.0 Architecture

SQL servers are an example of a client-server model for distributed computing. The database client issues SQL requests that travel over a network to the database server. Whether operating over the Internet or a local-area network, transporting SQL queries requires an application-level protocol operating over a low-level protocol such as TCP/IP.

The ODBC specification does not define the division of labor between client and server or the format of the messages sent across the network. ODBC client software must use communications protocols that will be understood by the database server. Database vendors such as IBM, Sybase and Oracle have defined their own protocols for client-server SQL processing. Sybase and Microsoft use the Tabular Data Stream (TDS) protocol. Oracle uses Transparent Network Substrate (TNS) and IBM uses Distributed Relational Database Architecture (DRDA).

Traditional ODBC drivers use platform-specific libraries that are supplied with the DBMS product. This type of driver uses the DBMS’s network libraries and maps ODBC functions to a library that implements the proprietary or native API. The traditional ODBC driver uses data access libraries such as Sybase Client-Library™ and network libraries such as Oracle SQL*Net, Net8 and Oracle Net.

The architecture of ODBC drivers has evolved from the original "fat client" model that requires multiple libraries. ODBC server technology supports thin clients and multi-database access. Wire protocol drivers eliminate the need for native network and data access libraries.

Wire protocol driver technology reduces the number of components required for a client-server connection. An ODBC application can connect, for example to a Sybase database without using the Sybase client software stack (Open Client™, Net-Library, Net-Library protocol drivers, net listener, CT-Library or DB-Library). Likewise, to connect to Oracle, an ODBC application does not require SQL*Net™, Net8™, Oracle Net™ libraries or Oracle Call Interface (OCI).

4.0 SQL APIs and Data Access Benchmarks

DataDirect Technologies traces its lineage back through more than a decade of shipping data access middleware, including traditional ODBC drivers. In 2001, DataDirect introduced wire protocol ODBC drivers that do not require native libraries. The introduction of new software often raises questions whether the new version performs as well as earlier versions, and in this case, as well as native APIs. To answer those questions, Ken North Computing, LLC conducted performance tests using Oracle Call Interface and wire protocol drivers from DataDirect Technologies. The driver used for these tests was DataDirect Connect *for* ODBC, release 4.1 for Oracle.

4.1 Methodology

All testing used identical queries and the same test configuration when running the API comparison tests. For the 1995 tests, the APIBench application ran on the same client, using the same network transport, and executed the same queries against the same tables and physical database. The hardware, database, tables and SQL queries were held constant. The test variable was the SQL API in use by the benchmark software. The 2002 benchmarks used the APIBench-B application. The methodology was similar but there was an additional variable. The OCI tests used an Oracle-supplied application-level network library. The wire protocol ODBC tests did not use Oracle network libraries.

The test suite included 10 different types of SQL queries (one INSERT, two DELETEs, two UPDATEs, and five different SELECTs). Query execution times were recorded to find the slowest and fastest queries of each type and to determine the mean execution time for five executions of each query type.

The testing configuration for the 1995 tests used an Oracle 7 database server, a version 7 OCI library, SQL*Net and a version 2.0 ODBC driver. A second set of benchmarks used Oracle 8*i* (8.1.6), corresponding versions of NET8 and OCI, and a version 3.5 ODBC driver. A third set of benchmarks used Oracle 9*i* (9.0.1) with corresponding versions of the OCI library and Oracle Net.

Tests for Oracle 9i used variable size row sets. Earlier tests used 2500 rows so results for that population are reported here.

4.2 Oracle Call Interface versus ODBC

Performance tests conducted in August 1995 and May 2002 compared query execution times when using different SQL APIs for Oracle. The 1995 tests compared ODBC and Oracle Call Interface. The 2002 tests compared ODBC and Oracle Call Interface, but used a wire protocol ODBC driver for Oracle.

4.2.1 SQL API Benchmarks for Oracle 7

The 1995 benchmarks of APIs for Oracle 7 showed a performance advantage using ODBC. As compared to Oracle Call Interface, the mean execution time for ODBC was faster in five of nine tests. The performance advantage ranged between 3.09% and 24.96% for ODBC. For four queries, OCI had a performance advantage that ranged between 1% and 13%.

Table 1: ODBC and Oracle Call Interface Performance
Query Execution Times for 2,500 Rows (Oracle 7)

API	INS	SEL1	SEL2	SEL3	SEL4	SEL5	UPD1	UPD2	DEL1
OCI 7	206.15	14.78	2.54	30.59	7.16	68.80	8.81	145.23	19.34
ODBC	199.78	14.02	2.58	33.32	6.81	51.63	9.93	146.16	17.98
Diff	-6.37	-0.76	+0.04	+2.73	-0.35	-17.17	+1.12	+0.93	-1.36
R %	96.91	94.86	101.57	108.92	95.11	75.04	112.71	100.64	92.97
Diff %	-3.09	-5.14	1.57	8.92	-4.89	-24.96	12.71	0.64	-7.03

INS	= populate tables	SEL5	= select 1 employee
SEL1	= select all employees	UPD1	= update all employees
SEL2	= select by hire date	UPD1	= update 1 by emp ID
SEL3	= join and order by	DEL1	= delete all employees
SEL4	= salary grouped by department	DEL2	= delete 1 by emp ID

4.2.1.1 Understanding Table 1

In table 1, the mean (average) execution times for OCI and ODBC, respectively, are in rows 1 and 2. The third row of data is the difference (Diff) between the mean execution time for OCI and ODBC. The mean execution times and difference are measured in seconds. In the difference row, a negative number means the ODBC query took less time than its OCI counterpart. The absolute difference in mean execution times for OCI and ODBC varied from .04 seconds to 17.17 seconds.

The row below Diff is the ratio of ODBC execution time to OCI7 execution time. In column one, for example, the ODBC query took 96.91% as much time as the OCI query. The final line (Diff %) is the percentage difference between OCI7 execution time and ODBC execution time. A negative number indicates the ODBC query was faster. For example, the INSERT query (INS1) was 3.09% faster using ODBC. A positive percentage means the ODBC query was slower.

4.2.2 Performance Measurements (May 2002)

The APIBench-B software executed a suite of SQL queries to compare Oracle Call Interface with wire protocol ODBC technology. APIBench-B used the DataDirect Connect *for* ODBC, release 4.1 Oracle Wire Protocol driver for tests with both Oracle 8*i* and Oracle 9*i*. The first round of tests ran on a Windows NT 4.0 SP6 system connecting to an Oracle 8.1.6 server. OCI tests used the Net8 network library. The second round of tests ran on a Windows XP Pro system connecting to an Oracle 9.0.1 server. OCI tests used the Oracle Net network library.

The test procedure executed identical queries against the same tables using the same TCP/IP transport, but varied the API and network library. For Oracle API tests, the employee table was populated with 2500 rows.

4.2.3 SQL API Benchmarks for Oracle 8*i*

The API tests for Oracle 8i used ten distinct SQL queries with a mix of direct execution and prepared queries. OCI tests used Net8 but ODBC tests used a DataDirect Connect *for* ODBC, release 4.1 Wire Protocol driver without Net8. The queries with Oracle 8*i* produced average execution times for ODBC and OCI/Net8 that were very close. The minimum and maximum times (fastest and slowest queries) were also very close. All differences in minimum, maximum and average execution time were less than a second for a row set size of 2500 rows.

Figure 1 illustrates how close the OCI / Net8 and ODBC Wire Protocol average execution times were with Oracle 8.1.6.

Figure 1: Oracle 8*i* Query Performance with OCI and Wire Protocol ODBC

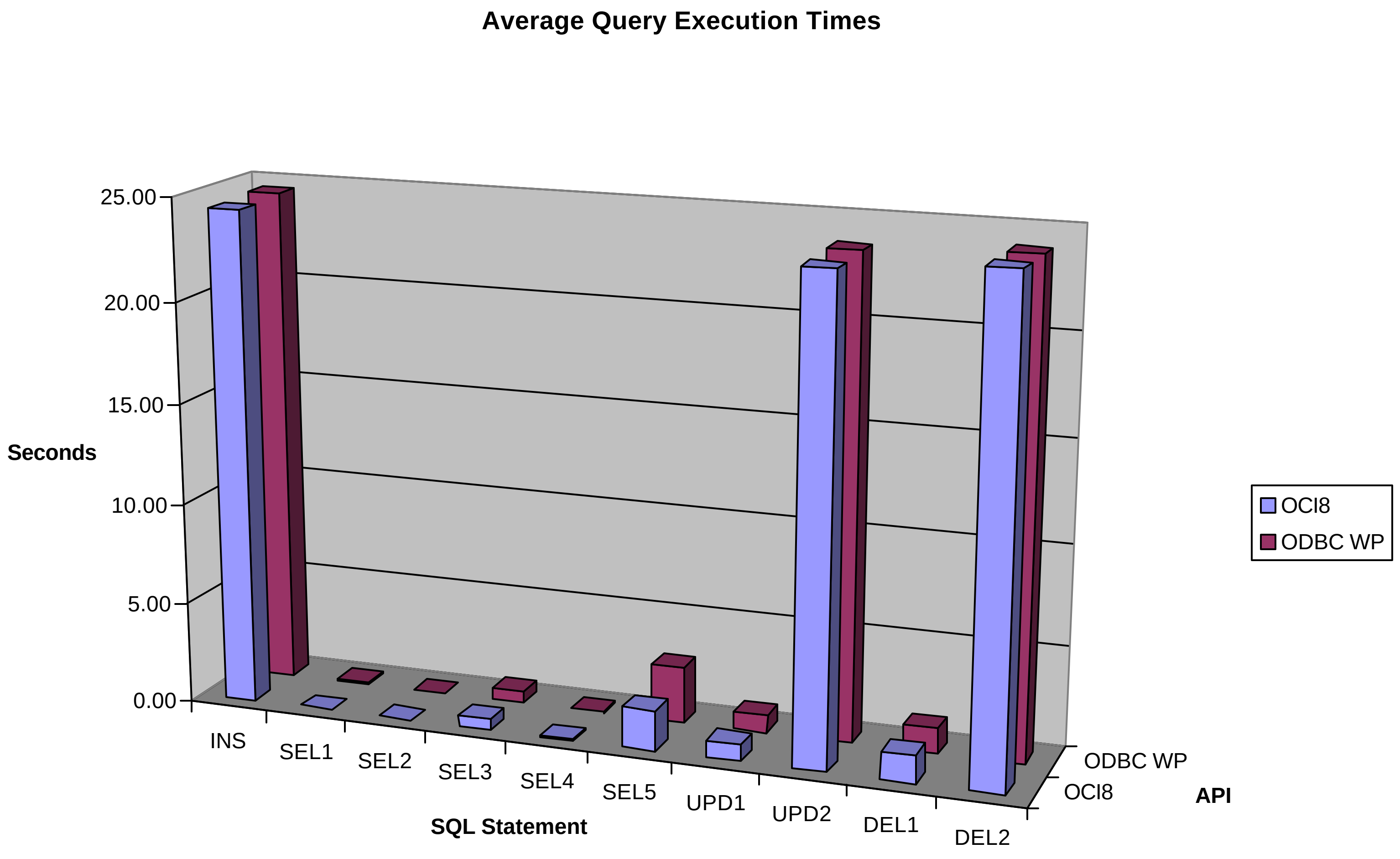


Table 2: Oracle Call Interface Benchmarks
Query Execution Times (seconds) for 2500 Rows (Oracle 8.1.6)

	INS	SEL1	SEL2	SEL3	SEL4	SEL5	UPD1	UPD2	DEL1	DEL2
Min	23.24	0.05	0.01	0.49	0.05	1.95	0.31	23.05	0.90	23.71
Max	24.91	0.06	0.01	0.61	0.06	2.11	1.26	23.98	1.41	24.35
Avg	24.39	0.06	0.01	0.55	0.06	1.99	0.86	23.52	1.37	24.07

Table 3: DataDirect Connect *for* ODBC Wire Protocol Benchmarks
Query Execution Times (seconds) for 2500 Rows (Oracle 8.1.6)

	INS	SEL1	SEL2	SEL3	SEL4	SEL5	UPD1	UPD2	DEL1	DEL2
Min	24.19	0.09	0.01	0.60	0.06	2.76	0.27	23.28	0.98	23.59
Max	24.96	0.10	0.02	0.63	0.07	2.83	1.35	24.22	1.37	24.26
Avg	24.51	0.10	0.02	0.62	0.06	2.80	0.84	23.60	1.25	23.98

Table 4: DataDirect Connect *for* ODBC (WP) and OCI/Net 8 Performance
Query Execution Times for 2500 Rows (Oracle 8.1.6)

API	INS	SEL1	SEL2	SEL3	SEL4	SEL5	UPD1	UPD2	DEL1	DEL2
OCI 8	24.39	0.06	0.01	0.55	0.06	1.99	0.86	23.52	1.37	24.07
ODBC	24.51	0.10	0.02	0.62	0.06	2.80	0.84	23.60	1.25	23.98
Diff	0.12	0.04	0.01	0.07	0.00	0.81	-0.02	0.08	-0.12	-0.09
R %	100.49	166.6	200.00	112.73	100.0	140.7	97.67	100.34	91.24	99.63
Diff %	0.49	66.67	100.00	12.73	0.00	40.70	-2.33	0.34	-8.76	-0.37

4.2.3.1 Understanding Table 4

In table 4, rows 1 and 2 are the mean (average) execution times in seconds. Row 3 (Diff) is the difference between mean execution times in rows 1 and 2. In the Diff row, a negative number means the ODBC average execution time was less (or faster) than its OCI counterpart. The fourth row is the ratio of ODBC mean execution time to OCI8 mean execution time expressed as a percentage. In column one, for example, the ODBC average execution time was 100.49% of the OCI average. The final row (Diff %) is the percentage difference between OCI8 execution time and ODBC execution time. A negative number indicates the ODBC query was faster. For example, the query to update all rows (UPD1) was 2.33% faster (on average) using ODBC. A positive percentage means the ODBC query was slower.

4.2.3.2 Interpreting the Results

During the 1995 benchmarks, the difference in performance was less than 10% for all but two queries. The percentage difference was greater for several of the Oracle 8*i* queries. More significant, however, is the minimal difference in minimum, maximum, and mean execution times for each query type.

During Oracle 7 testing, the difference in mean query execution time varied as much as 17.17 seconds. For only four query types was the difference less than a second. Oracle 8*i* testing found no such gap between OCI and wire protocol ODBC. The difference between the fastest, slowest and average execution time for each query type was always less than a second. 87% of the difference values were less than 250 milliseconds.

For Oracle 8*i* queries, the difference in mean execution time for OCI and ODBC varied from 0.0 to 0.12 seconds for nine of the ten queries. The difference for the remaining query was .81 seconds (810 milliseconds).

> For 90% of the Oracle 8*i* queries in this test, the average difference between ODBC and OCI performance was 61 milliseconds.

There was also little difference between the fastest query of each type and the slowest of each type. For eight of the ten query types, the fastest execution times for ODBC and the fastest for OCI differed by 230 milliseconds or less. The remaining two values differed by 810 milliseconds and 950 milliseconds. For nine query types, the difference in the slowest ODBC and the slowest OICI query ranged between 10 milliseconds and 240 milliseconds. For the tenth type, the difference between slowest queries was 720 milliseconds.

Simply put, differences in performance between the DataDirect Connect *for* ODBC wire protocol driver and OCI/Net8 are measured in the millisecond range for the average SQL query. The range of values for the slowest, fastest and average query execution time differed by 950 milliseconds or less for each of ten query types. For 85% of the fastest and slowest execution times, the difference between OCI8 and ODBC was 240 milliseconds or less.

4.2.4 SQL API Benchmarks for Oracle 9*i*

Tests in May 2002 show wire protocol drivers enjoy even more of a performance advantage for Oracle 9*i* than earlier versions of Oracle.

Table 5: Oracle Call Interface Benchmarks
Query Execution Times (seconds) for 2500 Rows (Oracle 9.0.1)

	INS	SEL1	SEL2	SEL3	SEL4	SEL5	UPD1	UPD2	DEL1	DEL2
Min	23.20	0.08	0.01	0.14	0.05	4.84	0.22	22.45	0.42	23.32
Max	25.99	0.09	0.03	0.14	0.06	4.89	0.27	22.89	2.19	23.83
Avg	24.71	0.08	0.02	0.14	0.06	4.86	0.24	22.70	0.98	23.47

Table 6: DataDirect Connect *for* ODBC Wire Protocol Benchmarks
Query Execution Times (seconds) for 2500 Rows (Oracle 9.0.1)

	INS	SEL1	SEL2	SEL3	SEL4	SEL5	UPD1	UPD2	DEL1	DEL2
Min	23.05	0.05	0.01	0.03	0.01	3.86	0.22	22.36	0.41	23.34
Max	24.22	0.06	0.02	0.03	0.03	6.06	0.23	23.20	1.42	24.00
Avg	23.53	0.05	0.02	0.03	0.02	5.12	0.22	22.58	0.62	23.53

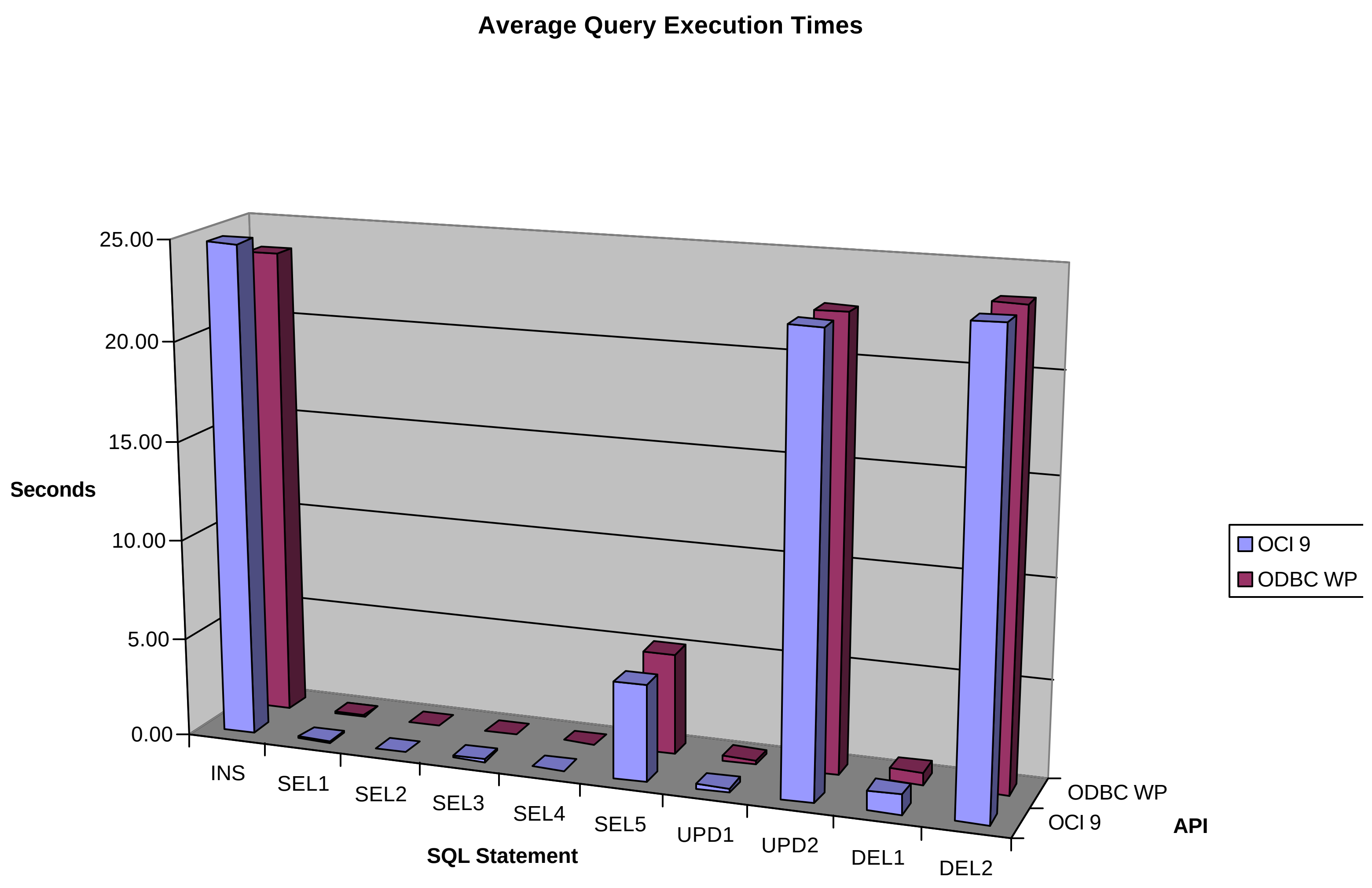
Table 7: DataDirect Connect *for* ODBC and OCI/Oracle Net Performance
Query Execution Times for 2,500 Rows (Oracle 9.0.1)

API	INS	SEL1	SEL2	SEL3	SEL4	SEL5	UPD1	UPD2	DEL1	DEL2
OCI9	24.71	0.08	0.02	0.14	0.06	4.86	0.24	22.70	0.98	23.47
ODBC	23.53	0.05	0.02	0.03	0.02	5.12	0.22	22.58	0.62	23.53
Diff	-1.18	-0.03	0.00	-0.11	-0.04	0.26	-0.02	-0.12	-0.36	0.06
R %	95.22	62.50	100.00	21.43	33.33	105.3	91.67	99.47	63.27	100.3
Diff%	-4.78	-37.5	0.00	-78.57	-66.67	5.35	-8.33	-0.53	-36.73	0.26

4.2.4.1 Understanding Table 7

In table 7, rows 1 and 2 are the mean execution times in seconds. Row 3 (Diff) is the difference between mean execution times in rows 1 and 2. In the Diff row, a negative number means the ODBC average execution time was less (or faster) than its OCI counterpart. The fourth row is the ratio of ODBC mean execution time to OCI9 mean execution time expressed as a percentage. The final row (Diff %) is the percentage difference between OCI 9.0.1 execution time and ODBC execution time. A negative number indicates the ODBC query was faster. A positive percentage means the ODBC query was slower.

Figure 2: OCI (9.0.1) and ODBC Wire Protocol Query Performance



4.2.4.2 Interpreting the Results

As compared to OCI 9.0.1, the mean execution time for DataDirect Connect *for* ODBC was better in seven of ten tests, with one being equal. When comparing the fastest times for each SQL statement, OCI was faster for only one of ten different SQL statements. When comparing the longest-executing queries of each type, OCI had the slowest queries in seven of ten cases.

Using wire protocol ODBC, INSERT statements, UPDATEs and four of five SELECTs were faster. DELETE all rows was noticeably faster and DELETE by employee ID was slightly slower.

Considering average execution times, the ODBC performance advantage ranged from 0 to 1.18 seconds. For two query types, the OCI performance advantage was 60 milliseconds and 260 milliseconds.

5.0 Conclusions

Examination of these test results shows that:

- Execution time differences are negligible when running SQL queries using DataDirect Connect *for* ODBC wire protocol drivers or Oracle Call Interface with Net8. There is no ODBC performance penalty.
- DataDirect Connect *for* ODBC wire protocol driver technology delivers better performance than programming with Oracle Call Interface 9.0.1.

Production assistance by:

Cathy Mugford, CPA, CISA
Control Consulting Corporation
www.cpacisa.com