An Oracle Technical White Paper
February 2008

Oracle TimesTen Products and Technologies

ORACLE

Introduction

Oracle TimesTen In-Memory Database is a memory-optimized relational database that delivers very low response time and very high throughput for performance-critical systems. It is targeted to run in the application tier—close to applications—and optionally, in process with applications. It can be used as the database of record or as a cache to Oracle Database.

This paper provides an overview of Oracle TimesTen products and technologies and their integration points with other Oracle products. It is intended as a paper evaluation prior to a hands-on experience with the software and manuals.

Milliseconds Matter

Oracle TimesTen products provide application-tier data management for performance-critical systems, and are optimized for blazing-fast response and real-time caching of Oracle data. Companies can extend their software infrastructures with Oracle TimesTen products to create systems that are

- Instantly responsive
- Highly scalable
- Continuously available

These systems are used to

- Increase customer loyalty
- Attract new customers
- Streamline operations
- Avoid the costly alternative of proprietary software development

Oracle TimesTen products have a proven track record, with production deployments since 1998 in real-time enterprises and time-critical environments such as network telecommunication services, operational support systems, contact centers, airline and reservation systems, command and control systems, and securities trading. Hundreds of companies worldwide use Oracle TimesTen products in production applications—including Amdocs, Aspect, Avaya, Bombay Stock Exchange, Cisco, Ericsson, JPMorgan, Lucent, NEC, Nokia, Salesforce.com, and Sprint.

The Growth of Real-Time Applications

Network equipment manufacturers, telecom operators, securities exchanges and brokerages, airlines, shipping and logistics companies, and defense and intelligence agencies are examples of enterprises in which real-time applications are a necessity. The use of real-time processing to capture, analyze, and respond intelligently to key events is increasingly becoming the benchmark for corporate excellence.

Real-Time Industries

For many companies, real-time applications aren't elective—they are a necessity. Network equipment manufacturers, telecom operators, securities exchanges and brokerages, airlines, shipping and logistics companies, and defense and intelligence agencies are prime examples of

enterprises that require real-time applications. In the past, building these applications also required the development of real-time infrastructure software. Fast but inflexible, these systems did the job as long as the applications remained static in their requirements. But dynamic industries quickly outpace static applications, and the cost to develop, test, and maintain specialized infrastructure software is rarely justified when commercial alternatives exist.

Real-Time Enterprises

With the growing velocity of messages moving through business networks, the use of real-time processing to capture, analyze, and respond intelligently to key events is becoming the benchmark for corporate excellence. This isn't important only for the execution and management of critical business processes. Customers expect highly tailored interactions and the utmost responsiveness from any company with which they do significant business.

Business activity monitoring, complex event processing, RFID/sensor-based applications, Web portals, and Web services are contributing to the movement of applications to the edge of the enterprise. Configured as dynamic collections of interrelated routines, these applications are part of an overall approach known as a service-oriented architecture (SOA). However, most data sources still reside in the back office, dominated by large amounts of rarely touched legacy data surrounding smaller amounts of currently active information. A natural extension of SOA concepts would include lightweight, real-time data management in the application tier, connected to corporate data sources to provide real-time performance for currently active data.

Real-Time Data Management Software

The enterprise architectures that derive the greatest benefit from real-time processing provide event, data, and transaction management in the application tier, empowering front-line systems with rapid response and deeper insight. It is not sufficient to merely collect and cache data next to applications, as is often the case with first-generation in-house efforts. Nor is it practical to locate the corporate database on the same platform as one of the applications.

What's needed is a generation of lightweight infrastructure software that presents familiar, powerful interfaces and query languages that are widely in use—that can easily interface with existing back-office databases, messaging systems, and application servers—and that exploits the full performance potential of today's networked, memory-rich computing platforms. This is the generation of infrastructure software for real-time data management provided by Oracle TimesTen products.

Application-Tier Deployment

Much of the new application development today is focused on improving interactions with customers and streamlining internal operations to eliminate delays and excess costs. These are real-time applications that reside near the network edge or, in some cases, within the network as hosted services.

This is the new enterprise application tier, with different platform, performance, and availability requirements than legacy back-office applications. Business events are embodied in network messages to which applications subscribe, triggering real-time processing and additional message publishing.

To meet the response time and scalability goals for these applications, it's often necessary to deploy the infrastructure software and applications on the same platform, including some or all the data that drives the application.

Oracle TimesTen products have been designed to integrate seamlessly into these environments, with an architecture that is optimized for application-tier deployment and configuration options that enable highly tailored solutions.

Oracle TimesTen Products

The three Oracle TimesTen products are based on in-memory database, data replication, and caching technologies.

Oracle TimesTen real-time data management software consists of three products based on in-memory database, data replication, and caching technologies. Next is a general introduction to these products and technologies; subsequent sections present additional detail.

Oracle TimesTen In-Memory Database

Oracle TimesTen In-Memory Database is a memory-optimized relational database that empowers applications with the instant responsiveness and very high throughput required by real-time enterprises and industries. Deployed in the application tier as a cache or embedded database, Oracle TimesTen In-Memory Database fits entirely in physical memory using standard SQL interfaces.

Replication—TimesTen to TimesTen

Replication—TimesTen to TimesTen is an option to the Oracle TimesTen In-Memory Database that enables real-time data replication between servers for high availability and load sharing. Data

replication configurations can be set to active/standby or active/active, using asynchronous or synchronous transmission, with conflict detection and resolution, and automatic resynchronization after a failed server is restored. Data replication is fully compatible with the Cache Connect to Oracle option.

Cache Connect to Oracle

Cache Connect to Oracle is an option to the Oracle TimesTen In-Memory Database that creates a real-time, updatable cache for Oracle data that resides in the application tier. It offloads computing cycles from back-end systems and enables remarkably responsive and scalable real-time applications. Cache Connect to Oracle loads a subset of Oracle data, propagates updates in both directions, automates pass-through of SQL requests for noncached data, and automatically resynchronizes data after failures.

In-Memory Database Technology

In-memory database technology implements a relational database in which all data at runtime resides in the RAM. The data structures and access algorithms exploit this property for breakthrough performance.

In-memory database (IMDB) technology is the foundation technology for Oracle TimesTen products. IMDB technology implements a relational database in which all data at runtime resides in the RAM, and the data structures and access algorithms exploit this property for breakthrough performance. Compared to a fully cached RDBMS, IMDB technology requires far less processing power, because the overhead to manage memory buffers and account for multiple data locations (disk and memory) is eliminated. With IMDB technology, magnetic disks are used for persistence and recovery rather than as the primary database storage location.

The memory-optimized performance of Oracle TimesTen products is complemented by functionality that supplies transactional properties, persistence mechanisms, and recovery from system failures. A variety of choices are available for locking, multiuser isolation, and logging, accommodating a range of application scenarios from transient look-up caches to core transactional trading and billing systems.

Durability is achieved by logging the changes from committed transactions to disk and periodically updating a disk image of the database, termed a "checkpoint." The timing of the disk write for the log is configurable by the application; it can be either synchronous with the end of the transaction or deferred until after. Many situations favor higher throughput over synchronous logging, particularly when the monetary value of a transaction is low or the transaction data is short-lived, such as when tracking the location of mobile phones that communicate their cell location every few seconds.

The native interfaces supported by Oracle TimesTen products are standards-compliant and generally compatible with any other standards-compliant relational database. Applications issue structured query language (SQL) commands through Java database connectivity (JDBC) or open database connectivity (ODBC) interfaces. The statements for defining databases, replication configurations, and cache groups also adhere to SQL syntax conventions. Simple network management protocol is used to issue standardized system management alerts.

An open transaction log API (XLA) with a standard Java message service (JMS) interface is provided for reading the transaction log. This is useful for creating applications that react to database updates. In this regard, XLA is a lightweight trigger. It's also a way to build custom data replication from Oracle TimesTen products to other database systems.

The Physical Structures of Oracle TimesTen Products

This section describes the system components of Oracle TimesTen's products—what is installed, running, and consuming computing resources.

Oracle TimesTen products consist of

- Shared libraries
- Memory-resident data structures
- System processes
- Administrative programs
- Checkpoint files and log files on disk

Application-Tier Shared Libraries

The routines that implement SQL operations and related functions are embodied in a set of shared libraries that developers link to their applications and execute as a part of the application's process. This shared library approach is in contrast to a more conventional RDBMS, which is implemented as a collection of executable programs that applications connect to, typically over a client/server network.

Normally, embedding data manager libraries into the application could make the database vulnerable to corruption if the application process were terminated abnormally. Oracle TimesTen products solved this challenge. Using a patent-pending algorithm known as MicroLogging, Oracle TimesTen libraries self-protect against application process failures. In-memory databases remain consistent, and other applications continue without impact.

IMDB technology is implemented as in-memory databases accessed by applications, utility programs, and system processes via shared library routines. Disk files for logs and checkpoints (backup copies) are optionally maintained for recovery purposes.

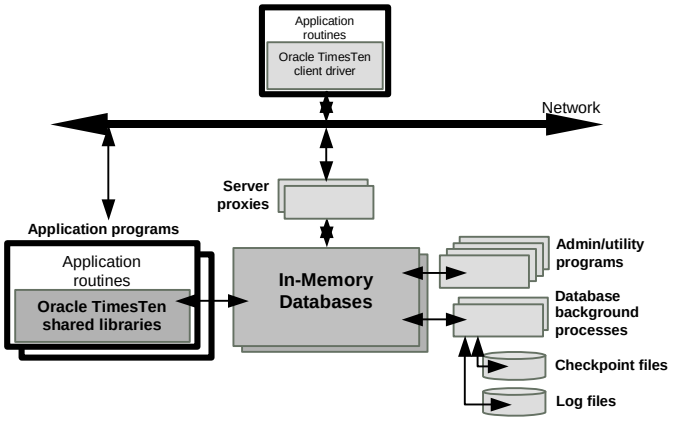

Figure 1. Components of Oracle TimesTen In-Memory Database

Applications may use a client/server connection to access an Oracle TimesTen database, though in most cases the best performance will be realized with a directly linked application.

Memory-Resident Data Structures

In-memory databases are maintained in an operating system's shared memory segments, and contain all user data, indexes, system catalogs, log buffers, lock tables, and temp space. Multiple applications can share an Oracle TimesTen In-Memory Database, and a single application can access multiple Oracle TimesTen databases on the same system.

Memory-resident databases are maintained in an operating system's shared memory segments and contain user data, indexes, system catalogs, log buffers, lock tables, and temp space. Multiple applications can share a database, and a single application can access multiple databases.

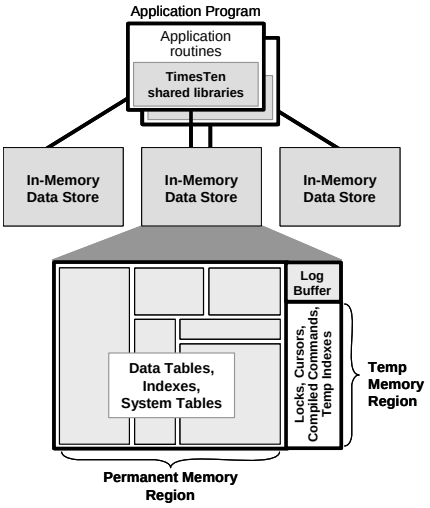

Figure 2. In-memory databases

System Processes

Background processes provide services for startup, shutdown, and application failure detection at the system level, and provide loading, checkpointing, and deadlock handling at the database level. There is one instancewide Oracle TimesTen daemon (a system may have multiple instances), and a separate subdaemon for each database.

Administrative Programs

Utility programs are explicitly invoked by users, scripts, and applications to perform services such as interactive SQL, bulk copy, backup/restore, database migration, and system monitoring.

Checkpoint and Log Files

Changes to the database and transaction logs are written to disk periodically. Should a database need to be recovered, Oracle TimesTen products merge the database checkpoint on disk with the completed transactions that are still in the log files. Normal disk file systems are used for checkpoints and log files.

Data Replication Technology

When near-continuous availability or workload distribution is desired, data replication can be configured to send updates between two or more servers. A master server is configured to send

updates, and a subscriber server is configured to receive them, but a server can be both a master and a subscriber for bidirectional replication. Time-based conflict detection and resolution is used to establish precedence in the rare event of the same data being updated in multiple locations at the same time.

When near-continuous availability or workload distribution is desired, replication can be configured to send updates between two or more servers. A server can be both a master and subscriber for bidirectional replication.

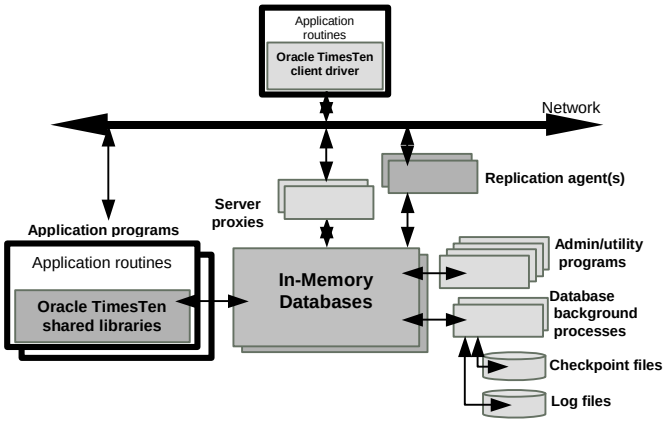

Figure 3. Adding data replication for high availability

When replication is configured, a replication agent process is started for each database. If multiple databases on the same server are configured for replication, there is a separate replication agent for each database. Each replication agent can send updates to one or more subscribers, and receive updates from one or more masters. Each of these connections is implemented as a separate thread of execution inside the replication agent process. Replication agents communicate through TCP/IP stream sockets.

For maximum performance, the replication agent detects updates to a database by monitoring the existing transaction log, and sends updates to the subscribers in batches if possible. Only committed transactions are replicated. On the subscriber node, the replication agent updates the database though an efficient low-level interface, avoiding the overhead of the SQL layer.

Caching Technology

When Oracle TimesTen products are used to cache portions of an Oracle Database in an in-memory database, a cache group is created to hold the cached data, additional shared library

routines are invoked by the applications, and a system agent (the cache agent) performs all asynchronous data transfers between the cache and Oracle Database.

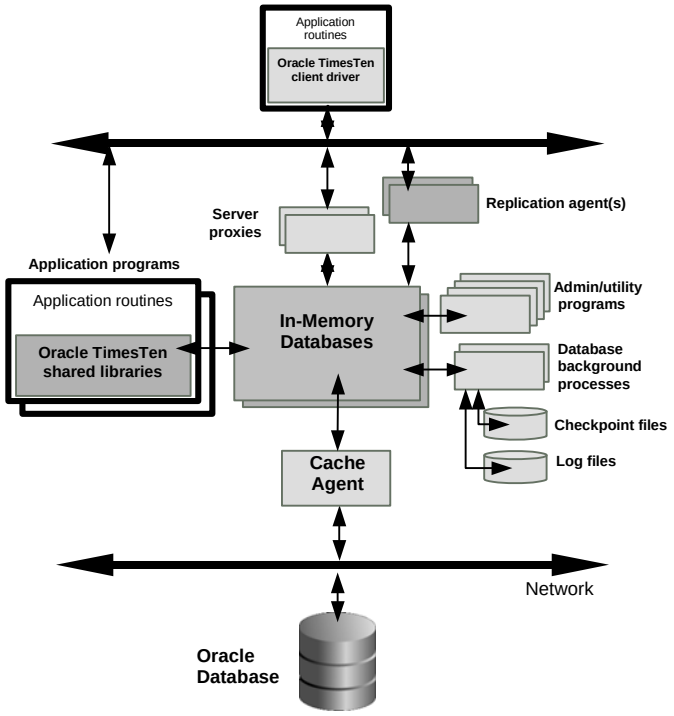

Figure 4. Adding caching for Oracle data

A cache group is a collection of one or more tables arranged in a logical hierarchy via primary/foreign key relationships. Each table in a cache group is related to an Oracle Database table. A cache group table can contain all or a subset of the rows and columns in the related Oracle Database table. Cache groups can be created and modified by means of the browser-based cache administrator or via SQL statements. Cache groups support the following features:

- Applications can both read from and write to tables in the cache groups
- Cache groups can be refreshed (adding new Oracle Database data into cache group) automatically or manually
- Cache groups can be flushed (propagating cache updates to Oracle Database tables) automatically or manually
- Changes to either Oracle Database tables or the cache group can be tracked automatically

When rows in a cache group are updated by applications, the corresponding rows in Oracle Database tables are updated synchronously as part of the same transaction, or asynchronously

immediately afterward depending on the type of cache group that was created. The asynchronous configuration produces significantly higher throughput and much faster application response times.

If a synchronous commit to Oracle Database fails for any reason, the transaction will be rolled back and consistency will be preserved, whereas an asynchronous configuration will only log the failure in the Oracle Database and not automatically roll back the Oracle TimesTen transaction, since it was already committed. However, the advantage of the asynchronous configuration (in addition to performance) is that it will continue to operate even if the connection to Oracle Database is lost, and will apply updates to Oracle Database automatically when the connection is restored.

Changes that originate in Oracle Database are refreshed into the cache via the cache agent. Updates to cached data are tracked and periodically refreshed by the agent.

IMDB Technology in Depth

By managing all data in memory and optimizing for that environment, in-memory database technology can operate much more efficiently, and thus offer dramatic improvements in performance..

In-memory database technology delivers impressive performance by changing the assumptions about where data resides at runtime. By managing all data in memory and optimizing for that environment, in-memory database technology can operate much more efficiently, and thus offer dramatic improvements in responsiveness and throughput.

But what's at the heart of this performance? Couldn't an application get the same results just by placing all its RDBMS data into main memory?

Much of the work that is done by an RDBMS is done under the assumption that data is primarily on the disk. Oracle TimesTen products, on the other hand, assume the data resides in main memory and can therefore take more direct routes to it, reducing code-path length and simplifying both algorithm and structure.

In comparing these architectures, there are a number of key differences that clearly illustrate how a many-fold performance improvement is attainable. To illustrate, just a few of these differences can be found in query optimization algorithms, buffer pool management, and indexing structures.

Query Optimization

Because disk input/output is far more expensive than memory access, disk-based RDBMSs have to assume that the data resides on disk.

Query optimization algorithms are different for disk-based systems than for memory-based systems. RDBMS optimization decisions are based on the assumption that data resides primarily on the disk. In a dynamic runtime environment, data might be on disk or cached in main-memory at any given moment. Because disk input/output (I/O) is far more expensive than memory access, disk-based RDBMSs have to assume that the data resides on the disk. They are optimized to reduce the point of performance bottleneck, so a disk-based optimizer will not always produce the optimal plan for data that resides—primarily or fully—in main memory.

IMDB technology, on the other hand, knows that the data resides within main memory and optimizes its queries under simpler assumptions. It does not need to make a worst-case scenario based on disk residency, and therefore, its cost estimates can be simpler and more consistently accurate.

Buffer Pool Management

Although necessary in disk-based data management solutions, buffer pools become unnecessary in memory-based data management because the data already resides within memory.

In conventional RDBMS architectures, buffer pools must be maintained for data that has been cached in main memory. When the SQL query processor requires a page of data, it must first search the buffer pool for that data, and even if that data is there, in many cases it must be copied out of the pool for processing. This buffer pool maintenance and management, coupled with additional data copies, add significantly to the original burden of making the data available to an application.

Although necessary in disk-based data management solutions, buffer pools are unnecessary in memory-based data management. The data already resides within main memory, so Oracle TimesTen has no buffer pool. Therefore, code path length and engine footprint are reduced, copying is avoided, the algorithm is simplified, and data is delivered to the application more quickly.

Moreover, RDBMS products are not designed to dynamically reverse these disk-based assumptions at runtime. Disk-based assumptions are too tightly interwoven within the system's code base to reshape them with a few well-placed if-then-else statements. One example of this interwoven, disk-based assumption can be seen in the indexing tree structures of the two architectures.

Index Structures

B+-Trees

In a typical RDBMS B+-tree index page, both the key value and the pointer to data are held within a B+-tree entry. A B+-tree node (a page on the disk) is made up of many B+-tree entries, each entry holding the index data value and a page number of either the next appropriate index node or a page number of the disk block that holds the sought data row. This structure results in a flat and wide tree—ideal for the reduction of disk I/O. In a B+-tree structure, the main goal is to reduce the amount of disk I/O needed to accomplish an indexed lookup of data files. A B+-tree accomplishes this by holding the key values within the B+-tree node itself (which reduces disk I/O), and holding as many index entries within the node as possible, (which increases the number of B+-tree entries that can be serviced with a single I/O).

This structure, although ideal when the data and index files reside on the disk, is less than ideal when the data resides in memory. Once data is held within memory, the goal of an indexing scheme should be to reduce CPU cycles, not I/O. CPU cycles are spent expanding and comparing compressed index values within a B+-tree. A significant number of CPU cycles are also spent managing buffers that hold data and indexes already read into memory from the disk.

T-Trees

In Oracle TimesTen products, the picture is simpler. A T-tree is optimized for main memory access. It has an index entry that is more economical both in terms of size and algorithm than its B+-tree cousin. Connections between T-tree nodes are achieved through less-than-or-equal-to and greater-than pointers. These pointers reference memory locations, not pages on the disk. With just two comparisons, the T-tree search algorithm knows whether the value it is searching for is on the current node or elsewhere in memory. And with every new index node pointer indirection, its search area is cut in half—the definition of a true binary search.

T-tree indexes are optimized for architectures that store date in memory. Since all rows are already in memory, T-trees do not include key values.

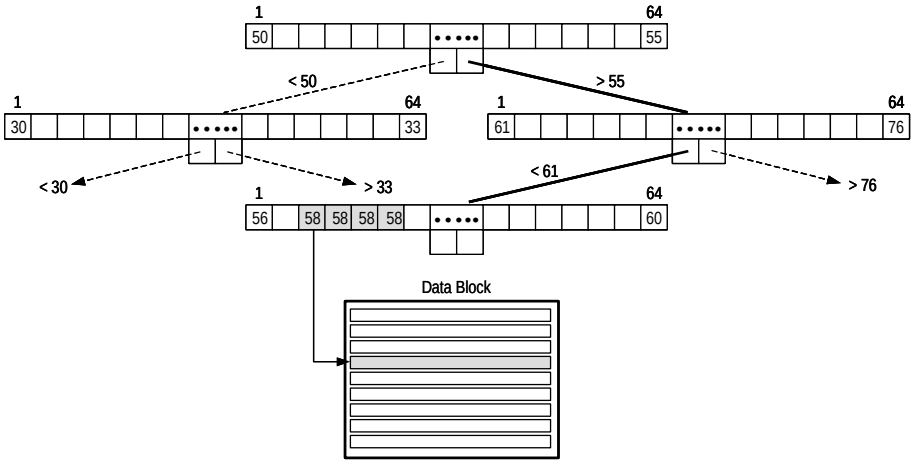

Figure 5. T-tree indexes for memory-resident databases

In main memory data management, the goal is to reduce space requirements; eliminate disk I/O; and shrink the algorithm, code path, and footprint. T-trees reduce space requirements by not holding the key value within the index node itself, but simply pointing to the value that already resides in memory within the data row. T-trees have no disk I/O because the index tree structures all reside in memory. T-trees shrink the complexity of code and reduce code path length by being dramatically simpler algorithmically. With T-trees, you get all the benefits of B+-tree-based access, but with tremendous economies of motion and size.

The Benefits of the Difference

When the assumption of data residing on the disk (or rather, the ambiguity of data location) is removed, complexity is dramatically reduced. The number of machine instructions drops by at least a factor of 10, buffer pool management disappears, extra data copies aren't needed, index pages shrink, and their structure is simplified. When data residing in memory is the bedrock assumption, everything gets simpler, more elegant, more compact, and faster.

A New Look at Price Performance

For many applications, leveraging in-memory database technology offers a new way to gain a competitive edge, improve user satisfaction, and boost return on investment.

Performance improvements can be measured in multiples rather than fractions. An application that spends half its processing time managing data and half its time within its application space or business logic can nearly double its capacity by using an in-memory data management product.

This dramatically increases the options in terms of market strategies, architectural economies, and system choice.

Exceptional Performance

Scalability

Oracle TimesTen technology is engineered to take advantage of multiple CPUs on symmetric multiprocessor computers. Figure 6 shows the transaction throughput of Oracle TimesTen products on a four-CPU Linux/Intel system. Each transaction executes a single SQL update or select (read) operation, as indicated. The results are shown for one, two, and four CPUs.

Oracle TimesTen products are capable of exceptional throughput, ranging from tens of thousands of update transactions to hundreds of thousands of read operations per second, when the transaction contains a single SQL operation.

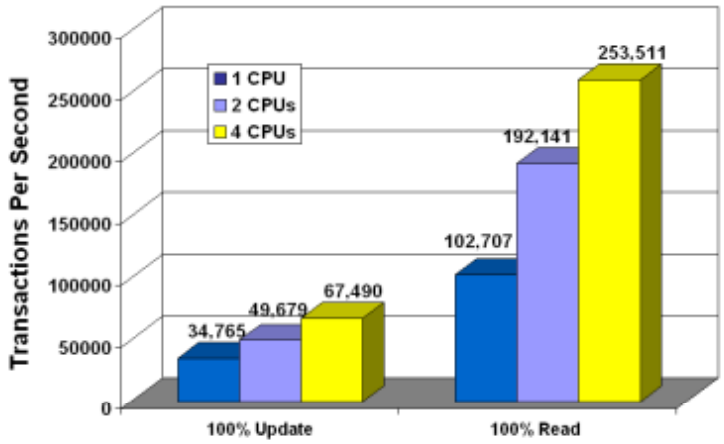

Figure 6. The throughput of Oracle TimesTen products

The simplest and therefore highest volume workload is the "100% reads" test, which measures how many individual records can be retrieved using a key value lookup. In this case 253,511 reads were completed every second, on average. With only one CPU, the read rate was 102,707 per second.

Update operations require more processing than a read operation, partly because changes must be logged for recovery purposes. In this result, more than 67,000 records were updated per second on four CPUs, and more than 34,000 on a single CPU.

Although these pure workloads aren't representative of any particular business application, these results validate the extreme efficiency of Oracle TimesTen products, and any realistic mixture of SQL operations would likely result in a maximum throughput in the thousands- to tens-of-thousands-per-second range. If higher volumes are required, Oracle TimesTen products scale well beyond a four-CPU system. Note that performance results vary on different processor platforms.

Response Time

Throughput is a byproduct of individual transaction response times. As throughput increases, average response time decreases. The exceptional throughput of Oracle TimesTen products yield microsecond-level response times on systems across a range of data management workloads. The average response times for the throughput workloads previously described are shown in Figure 7.

Oracle TimesTen products' response time is measured in microseconds for single-record operations. Data-intensive applications will benefit the most, since overall response time for the application includes more than just the data management portion.

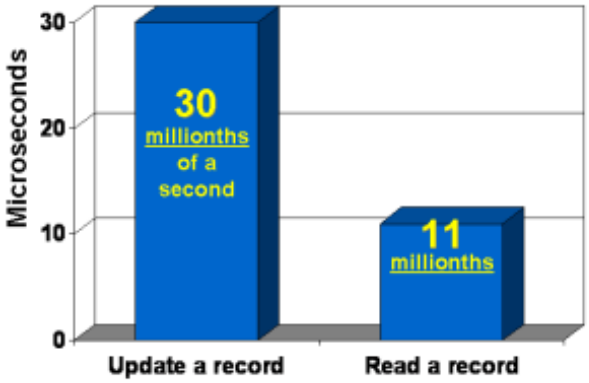

Figure 7. Latency in Oracle TimesTen products

It's important to keep in mind that a transaction's response time is the aggregate of the response times of the constituent processing steps, such as business logic, networking, I/O, and data management. The performance of the Oracle TimesTen product's portion is not the only determinant of overall response time. Sometimes one of the steps, usually I/O or networking, will dominate the others.

For instance, if client/server network is used instead of a direct connection between the application and the data manager, an additional millisecond or more of response time can be added for each round trip over the network. Similarly, if synchronous I/O is used for logging

changes, the relative slowness of a disk operation is included in the response time. In both cases, if the data management component is completed within microseconds, the overall application response time is still going to be a factor of the network and I/O latencies.

For applications that have a dominant data management component, the speedup of Oracle TimesTen technology could achieve a major decrease in response time. The highest-performing application is one that is directly connected, able to use asynchronous logging, and is data management–intensive. Search-intensive online reservation systems, presence, and location-based communication services and contact center routing engines are good examples.

As always, performance results vary greatly depending on many factors, such as the size of a transaction, the amount of nondata access logic, and platform performance.

Real-Time Functionality

This section presents a functional view of Oracle TimesTen products, describing the interfaces and features used by developers when writing real-time applications.

Data Management

The data manager feature in Oracle TimesTen products implements a true relational database model. It is not a hybrid model or a different model with a relational view grafted on top. Developers who are familiar with popular RDBMS products should be immediately productive.

Relational Database Model

In the relational model, user data is stored as rows (records) within tables. A row is a list of columns (data fields), each of a specified data type. Indexes are explicitly or implicitly created to speed key-based or value-range searches of the tables. Oracle TimesTen products support both hash indexes and T-tree indexes. Defined constraints, such as uniqueness or referential integrity (parent/child relationships between tables), are enforced automatically.

Within a table, a primary key—comprised of one or more columns—may be designated. A primary key is the unique identifier for a row, and Oracle TimesTen products automatically prohibit the insertion of additional rows with the same primary key value. One or more foreign keys may also be designated per table, each associated with the primary key of another table. Foreign keys enforce a parent/child relationship between two tables. For example, a row with a primary key will not be deleted if an associated foreign key in another table has the same value.

A materialized view is a read-only table derived from regular (base) tables. It can include parts or all of the data from one or more tables, and can include calculated values (such as totals and

averages). Since materialized views are automatically updated by the system whenever necessitated by a change to a base table, they provide applications with faster access to frequently derived data. Without materialized views, the overhead of joins and other calculations would be incurred whenever an application needed to read the data. Materialized views are especially useful in support of event processing (discussed in more detail later in this article).

If Cache Connect to Oracle is used, one or more cache groups can be defined in an in-memory database. A cache group is a hierarchical set of tables (enforced through foreign keys) that map to a corresponding set of Oracle Database tables, and contain some or all of the same data. Cache groups are both read and write, and updates to the data can propagate automatically between Oracle TimesTen products and Oracle Database.

Data management operations are expressed via the industry-standard Structured Query Language (SQL). The major operations are SELECT (read), UPDATE, INSERT, and DELETE. The relational join operation, which combines columns from more than one table into a result set, is also supported, as are most of the SQL options in the baseline SQL-92 standard definition.

Tables, indexes, materialized views, and cache groups are created and maintained within the databases. Applications connect to an Oracle TimesTen database and request data management operations against the data within. Multiple databases may exist on the same computer system and be accessed by the same application. However, no single operation (for example, a join across two tables) may span more than one database. Applications can participate in a distributed transaction through Oracle TimesTen products' support of the standard XA (or JTA for Java) interface for two-phase commit.

The earlier discussion of Oracle TimesTen IMDB technology explained how data management services are performed in real time. It's worth repeating that the entire database is guaranteed to reside in memory, enabling the elimination of processing instructions—such as searching algorithms, buffer management overhead, data copies, and suboptimal index structures and execution plans—that would otherwise have to account for the ambiguity of the data location.

In addition, the layout of information inside an Oracle TimesTen database is optimized for fast processing and memory conservation. In a disk-based RDBMS, disk structures and memory structures are almost identical; a memory page is the same size as an I/O block size to minimize I/O delays. With an Oracle TimesTen database, that's not important, so the memory structures can be designed to minimize processing instructions.

Durability and Concurrency

Oracle TimesTen products conform to the atomicity, consistency, isolation, and durability properties of data management systems.

Oracle TimesTen products conform to the atomicity, consistency, isolation, and durability properties of data management systems. These properties ensure that, in a multiuser system, each transaction operates as if it were the only transaction being executed at the time, and that the system can guarantee that the effects of a committed transaction are not lost. These are the most rigid properties required of data managers, and Oracle TimesTen products ensure full conformance.

A common misperception is that in-memory data managers cannot prevent data loss from system failures. In fact, the same techniques that make transactions and data durable in a conventional database are used in Oracle TimesTen products. As in all transaction-oriented systems, durability is achieved through a combination of change logging and periodic refreshes of a version of the database that resides on a disk.

Oracle TimesTen products offer applications control as a tradeoff for the degree of durability and the total throughput. More of one means less of the other, and Oracle TimesTen products provide choices along this spectrum.

Disk Operations

Oracle TimesTen databases may be permanent or temporary, and for exclusive or shared access.

If an application requires that no changes be lost, log records are flushed to disk as part of committing the transaction. If maximum performance is more important than possibly losing some transactions, log records can be written to disk less often, asynchronously from each transaction commit. In either case, the Oracle TimesTen data manager will attempt to "group commit" multiple transactions together to minimize disk writes.

Periodically, the database is checkpointed to disk automatically based on user-specified frequency and log volume. Oracle TimesTen products support "fuzzy" checkpoint operations that run in the background with minimal impact on running applications. Recovery from a system failure is a matter of merging log records with the latest checkpoint file.

Oracle TimesTen databases may be permanent or temporary, and for exclusive (single-user) or shared (multiuser) access. Permanent databases (checkpoint files) remain on disk when not in use. Temporary databases reside solely in memory and are removed when not in use.

Transaction logging serves two purposes. First, logging enables the recovery of transactions against persistent databases after a system failure. Second, logging enables the Oracle TimesTen data manager to perform deadlock detection and elimination when used in a shared access mode. A nonlogged database provides better performance. However, with logging disabled, transactions cannot be rolled back or recovered, and locks can only be placed at the database level (no row-

level or table-level locking is allowed). Disabling the log is appropriate only for single-user operations that can be restarted from scratch should a failure occur.

Permanent, shared, disk-logged databases are the most commonly deployed configurations. Temporary databases are common in applications for which the data is so transient that there is no value in recovering the database after a system failure, for example, highly dynamic state information in a call center application.

Locking and Isolation

Oracle TimesTen products apply locks to prevent a user from changing data that is currently being read or changed by another user. A lock is placed on database objects at either the database, table, or row level. Row-level locking provides the greatest multiuser concurrency, and is the default for Oracle TimesTen products.

Applications can call a procedure to change the locking level to row or database during runtime. Table-level locking is used when the Oracle TimesTen optimizer determines that it's advantageous, or when the application calls a procedure that directs the optimizer to apply table-level locking during the duration of a transaction.

Users can select an isolation level to specify the behavior of locks that are applied in response to read operations. The two isolation levels are serializable or read-committed (the default).

Applications select either the serializable or read-committed isolation level. Read-committed isolation provides the greatest multiuser concurrency. It guarantees that readers will only see committed data values, but doesn't wait for writers to release or place locks on records that are being read.

Serializable isolation is used for transactions that demand consistent, predictable data values through the life of a transaction. With serializable isolation, records that are read are prohibited from being updated or deleted by other users until the transaction commits or is rolled back. During this time, other users aren't allowed to insert new records if they would be part of the result set from these read operations. In other words, serializable isolation guarantees that a read operation could be repeated with the same results. With database-level locking, transactions are effectively operating with serializable isolation by default.

Read-committed isolation provides the greatest multiuser concurrency. It guarantees that readers will see only committed data values, but doesn't wait for writers to release or place locks on records that are being read. Oracle TimesTen products create two copies of a record that is being updated: the preupdated and thus "committed" values for readers, and the updatable version for writers. Readers are not blocked access to the read version, and writers don't wait for readers.

Query Processing

A primary objective of Oracle TimesTen products is to adopt relevant open standards. Applications can communicate with Oracle TimesTen databases through SQL, JDBC, ODBC, and JMS interfaces.

Most specialty products designed for high performance require proprietary, "under the covers" APIs. Oracle TimesTen products do not. The only way to access an Oracle TimesTen database is through standards: structured query language (SQL), Java database connectivity (JDBC), open database connectivity (ODBC), and Java Message Service (JMS). The implementation of these interfaces has been highly tuned for the architecture of Oracle TimesTen products.

SQL has been widely adopted for years as the query language standard for relational databases. Abstraction from the underlying storage and indexing details is one of the main benefits of SQL. It's also an easy-to-use language that expresses which action is to be done, rather than how to perform it. SQL has no references to indexes, data types, or physical layouts—just tables, columns, and search conditions. The optimizer feature in Oracle TimesTen products determines the fastest way to respond to the query based on factors such as the presence of indexes and the distribution of key values.

This level of abstraction allows the underlying data model to be tuned or extended without affecting existing applications. New services can be quickly added into a production environment simply by adding application modules and any required data tables and columns. Without a query language such as SQL shielding the application from the internals of the data manager, most applications would be affected by the addition of new data fields.

Data Replication

Replication is configured through SQL statements and can apply to designated tables or an entire database. To enable high efficiency and low overhead, Oracle TimesTen products use a transaction-log-based replication scheme.

Replication is the process of copying data between databases. This can help make data continuously available to mission-critical applications with minimal performance impact. In addition to its role in failure recovery, replication is also useful for distributing user loads across multiple databases for maximum performance and for facilitating online upgrades and maintenance.

Oracle TimesTen products follow a master/subscriber replication model in which committed changes are copied from their source to one or more subscriber databases. Replication is configured through SQL statements and can apply to designated tables or an entire database. To enable high efficiency and low overhead, a transaction-log based replication scheme is used.

Replication at each master and subscriber database is controlled by replication agents that communicate through TCP/IP stream sockets. The replication agent on the master database reads the records from its transaction log and forwards any detected changes to the replication agent on the subscriber database. The replication agent on the subscriber then applies the updates to its local database. If the subscriber agent is not running when the updates are forwarded by the master, the master retains the updates in its transaction log until they can be applied by the subscriber.

Balancing Performance and Consistency

The master and subscriber databases have internal mechanisms to confirm that the updates have been successfully received and committed by the subscriber. These mechanisms, which are completely invisible to the applications, ensure that updates are applied only once.

The replication mechanism in Oracle TimesTen products is by default asynchronous. When using asynchronous replication, an application updates a master database and continues working without waiting for the updates to be received by the subscribers. The master and subscriber databases have internal mechanisms to confirm that the updates have been successfully received and committed by the subscriber. These mechanisms, which are completely invisible to the application, ensure that updates are applied by a subscriber only once.

Asynchronous replication provides maximum performance, but the application is completely decoupled from the receipt process of the replicated elements on the subscriber. Oracle TimesTen products also provide two return service options for applications to confirm that the replicated data is consistent between the master and subscriber databases.

- The return receipt service loosely couples or synchronizes the application with the replication mechanism by blocking the application until replication confirms that the update has been received by the subscriber
- The return twosafe service enables fully synchronous replication by blocking the application until replication confirms that the update has been both received and committed by the subscriber

Applications that use the return services trade some performance to ensure higher levels of data integrity and consistency between the master and subscriber databases. However, the return receipt service has less performance impact than the return twosafe service, at the expense of less synchronization.

Replication Topologies

By designating a database as both a master and a subscriber, bidirectional replication may be configured. Additionally, Oracle TimesTen products allow for multinode "n-way" replication, which offers a wide range of possible replication topologies, including both hot-standby and active-active configurations. The latter can also be broken into a split workload (each replicated table has only one master) and a distributed workload (a table has more than one master). In a distributed workload configuration, the application must divide the work between the two systems so that replication collisions do not occur. In the event that collisions do occur, a timestamp-based collision detection and resolution mechanism prevents inconsistent replicas.

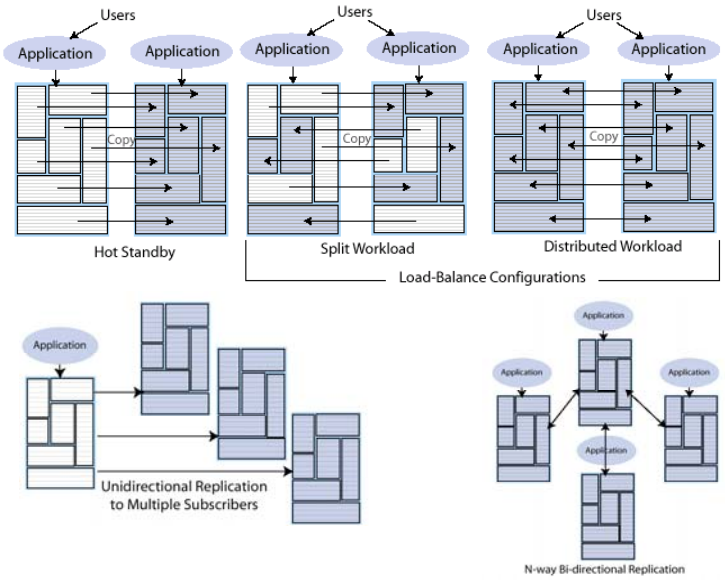

Figure 8. Replication configuration options

You can also define a subscriber to serve as a propagator that receives replicated updates from a master and passes them to subscribers of its own. Propagators are useful for optimizing replication performance over lower-bandwidth network connections, such as those between servers in an intranet.

"Leading service providers rely on our products to manage time-critical billing data, so bottlenecks and downtime cannot be tolerated. With Oracle TimesTen [products] and Oracle Real Application Clusters, we gain a reliable, scalable, proven real-time data management foundation to better serve our customers."

Ed McKee, Director of Applications, Interact, Inc.

Caching

Customized Integration

It's common for Oracle TimesTen products to be deployed in conjunction with a disk-based RDBMS, so that the Oracle TimesTen product is used when data needs to be captured or processed in real time. As the data transitions to a non-real-time state (for example, when a stock trade is complete or a call detail record has been rated) the information is transferred from the Oracle TimesTen product to the back-end RDBMS. There are multiple ways in which this integration can be achieved.

Applications can connect to both Oracle TimesTen products and the back-end database, and move the data through regular API requests. This is the most flexible approach, but is the least transparent to the applications and could require complex programming. For example, an application that needs to cache all active subscribers could first request a record in an Oracle TimesTen database, and if results are not found, connect to the back-end RDBMS and repeat the same request, and insert the results into the Oracle TimesTen database. If there were any updates to the data, they would need to be made in both databases by the application. However, changes made directly to the data in the back-end RDBMS could result in cache coherency issues.

A variation of this approach is to connect an Oracle TimesTen product and the back-end RDBMS through a publish-and-subscribe message bus, write new modules separate from existing applications that listen for changes, and then duplicate changes picked up from the bus. An application can use the Oracle TimesTen product's transaction log API (XLA) to register for and receive notice of updates. Most RDBMSs offer a trigger feature that can signal changes, or provide an API similar to XLA.

Cache Connect to Oracle

An easier and more transparent alternative for integrating Oracle TimesTen databases and Oracle Databases is through the built-in connections of Cache Connect to Oracle.

Unlike most caches, which are read-only, Cache Connect to Oracle supports read/write caching of Oracle data and bidirectional propagation of updates between Oracle TimesTen products and Oracle Databases. Another distinction is cache groups—a group of Oracle TimesTen tables that

map to all or a subset of the tables in an Oracle Database. A cache group can also consist of all or a subset of the rows and columns in these Oracle Database tables.

Cache Connect to Oracle provides controls that specify how long Oracle Database data should remain in the cache. In addition to the ability to set up automatic functions, a cache group can be loaded, refreshed, flushed, and unloaded on demand through SQL statements.

Cache Connect to Oracle interacts with Oracle Database to perform all the synchronous cache group operations, such as creating a cache group, loading the cache group, and propagating updates. In addition, there is a background process, called the Oracle Agent, that performs all the asynchronous cache operations, such as automatically propagating updates from Oracle Database to a cache group in an Oracle TimesTen product.

Cache Connect to Oracle can send SQL statements to either a cache group or Oracle Database through a single connection. This single-connection capability is enabled by a pass-through feature that checks if the SQL statement can be handled locally by the cache tables or if it must be redirected to Oracle Database. Cached data can be updated in either the Oracle TimesTen cache group or in Oracle Database. Cache Connect to Oracle provides the ability to automatically propagate updates from the cache group to Oracle Database, as well from Oracle Database to the cache group in Oracle TimesTen products.

Applications can send SQL statements to either a cache group or Oracle Database through a single connection. This single-connection capability is enabled by a pass-through feature that checks if the SQL statement can be handled locally by the cached tables or if it must be redirected to Oracle Database.

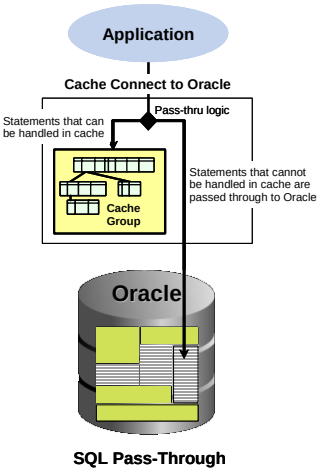

Figure 9. The single connection feature of Cache Connect to Oracle

There are two basic categories of cache groups.

- **System-managed** cache groups provide predetermined caching behaviors that are managed entirely by Cache Connect to Oracle.
- **User-managed** cache groups allow users to select from all the attributes and SQL statements to define customized caching behaviors. Users have full control over the load, aging, and propagation mechanisms.

There are two basic types of system-managed cache groups.

- **Write-through** cache groups load the cached table data from Oracle Database when created. Thereafter, all updates to the cache group are automatically propagated to Oracle Database. Write-through cache groups can be either asynchronous (AWT) or synchronous (SWT). SWT cache groups wait for a commit on Oracle Database before committing in the Oracle TimesTen product's cache. AWT cache groups commit changes in the cache without waiting for a commit on Oracle Database.
- **Read-only** cache groups hold read-only data, so updates on the tables in the cache group are not allowed. Updates must be done on Oracle Database. These updates can either be done directly in the Oracle Database or via Oracle TimesTen products by redirecting the updates to Oracle Database using the pass-through feature. By default, a read-only cache group is automatically refreshed when updates occur on Oracle Database. The refresh frequency is controlled by the application.

Cache group behavior can be defined and managed by Cache Connect to Oracle, or controlled by the user for maximum flexibility. System-managed cache groups implement predefined read-only or write-through configurations.

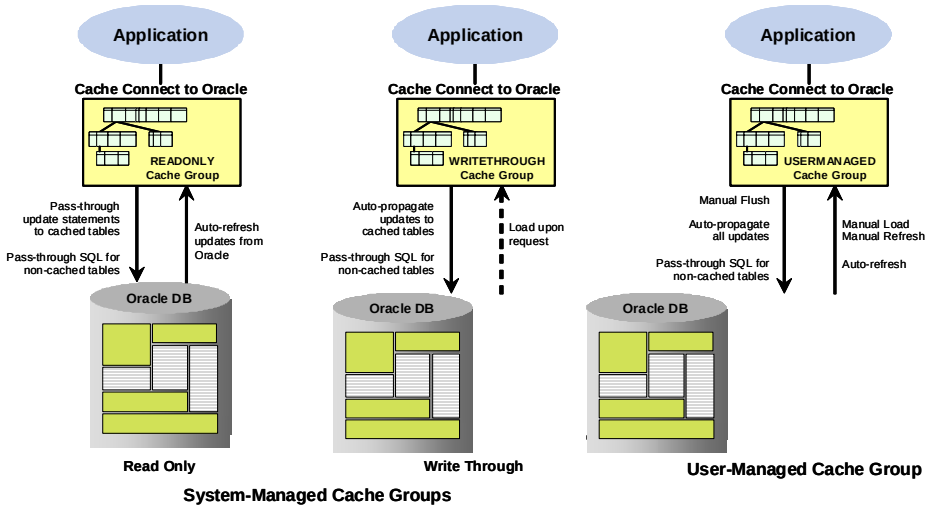

Figure 10. System-managed and user-managed cache groups

Event Processing

Event processing applications can sense and respond to events deemed to have business relevance.

According to Gartner research on the real-time enterprise (RTE), "The RTE monitors, captures, and analyzes root-cause and overt events that are critical to its success the instant those events occur, to identify new opportunities, avoid mishaps, and minimize delays in core business processes. The RTE will then exploit that information to progressively remove delays in the management and execution of its critical business processes."[1]

There are multiple ways in which Oracle TimesTen products enable event-processing applications. At the most basic level, Oracle TimesTen products are designed to support applications that reside at the edge of a network, where messages and business events arrive, and where they can be first detected. The ability to exist with the applications on a wide range of computing servers provides the flexibility to support event-processing applications.

[1] K. McGee, *Gartner Updates Its Definition of Real-Time Enterprise*," March 25, 2004.

However, simple events by themselves often do not reveal their relevance and must be correlated with other information or staged until a pattern is evident. In these cases, the event data must be captured in a data manager and possibly compared to reference data that was pulled from another data source. When an event that requires action is identified, taking the appropriate response must, in many cases, be instantaneous. Such processing is ideally suited to the functionality of Oracle TimesTen products.

The Oracle TimesTen XLA enables the detection of database updates. Applications use XLA to monitor changes in an Oracle TimesTen database and take actions based on those changes. Notification may be desirable when, for example

- The value of any customer's total trades for the day exceeds US$1,000,000
- A customer on the "A" list makes a purchase
- A prepaid balance is depleted
- A network element is taken offline
- A flight changes departure gates

Multiple applications can simultaneously read transaction log updates, and each application can maintain its own bookmark in the log file to maintain its position. Bookmarks are persistent across database connections, shutdowns, and system failures, so event notifications can pick up where they left off.

XLA is often used to build a custom data replication solution to a non-Oracle TimesTen database. Oracle TimesTen's event processing functionality can be achieved in a conventional RDBMS through triggers and stored procedures. However, XLA is designed for lower overhead and higher performance, consistent with real-time expectations. Also, when combined with materialized view functionality, XLA events can be targeted at very granular subsets of the database. Traditional database triggers execute their logic every time a change is made to any record in the table.

Materialized views combined with XLA functionality provide an efficient way to implement fine-grained event notification. That is, a materialized view pulls together data that is related to specific events that might have actions taken on it. An XLA application needs only to monitor update records that are of interest from a single materialized view table. Without a materialized view, the XLA application would have to monitor all the update records from all the base tables, including records reflecting updates to rows and columns of no interest to the application.

Conclusion

With the growing velocity of messages moving through business networks, the use of real-time processing to capture, analyze, and respond intelligently to key events is becoming the benchmark for corporate excellence. This isn't important simply for the execution and management of critical business processes. Customers expect highly tailored interactions and the utmost responsiveness from any company with which they do significant business.

What's needed is a generation of lightweight infrastructure software that presents familiar, powerful interfaces and query languages that are widely in use—that can easily interface with existing back-office databases, messaging systems, and application servers—and exploits the full performance potential of today's networked, memory-rich computing platforms. This is the generation of infrastructure software for real-time data management provided by Oracle TimesTen products.

Oracle TimesTen products provide application-tier data management for performance-critical systems, optimized for blazing-fast response and real-time caching of Oracle data. Hundreds of companies worldwide, including Amdocs, Aspect, Avaya, Bombay Stock Exchange, Cisco, Ericsson, JPMorgan, Lucent, NEC, Nokia, Salesforce.com, and Sprint use Oracle TimesTen products in production applications.

Oracle TimesTen Products and Technologies
February 2008

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

0109