Informix Internet Foundation.2000

For UNIX, Linux, and Windows NT

TECHNICAL BRIEF

Informix® Internet Foundation.2000™ combines the performance and scalability of Informix's proven relational database technology with the extensibility and flexibility of object-oriented technology. Specifically designed for the Internet, Informix Internet Foundation.2000 enables companies to quickly and efficiently integrate dynamic data types such as video, image, HTML, geospatial, and other complex data. It offers sophisticated tools to extend the transactional engine to the Web, incorporating server-managed rich data with Java and XML programmability. With Informix Internet Foundation.2000, Informix lets businesses handle any type of information to do just about anything imaginable—it is the smartest data engine for the Internet.

ABOUT INFORMIX INTERNET FOUNDATION.2000

The Internet has become a major defining force in the success of today's businesses. Implemented correctly, a business can gain significant competitive edge, enabling it to expand its market, improve operating efficiencies, and retain valuable customers.

However, the Internet has brought about a data management problem. Complex and dynamic data types such as video, image, sound, and other complex data are the norm of the Internet. To implement a successful Internet strategy, businesses need to choose the right database management solution—one that is capable of storing, retrieving, and managing rich data types and performing complicated operations such as keyword searches and aggregations on this non-traditional data quickly and efficiently. Furthermore, the database management solution must be easily extended to the Web—providing the ability to publish, transact, and analyze business data—at a low cost.

The Informix Internet Foundation.2000 is the smartest Internet solution in the market today. It combines the best features of Informix's industry-leading relational database management system (RDBMS) technology with the flexibility and power of Informix's advanced object-relational technology to produce a first-in-its-class data engine for the next generation of Internet computing. As such, Informix Internet Foundation.2000 lets businesses quickly and reliably extend their enterprise to the Web with the same degree of performance and reliability as traditional OLTP applications.

At the core of Informix Internet Foundation.2000 is Informix Dynamic Server.2000™, Informix's powerful, multi-threaded data engine designed to deliver breakthrough database scalability, manageability, and performance. Informix Dynamic Server.2000 takes database technology to the next level by incorporating extensibility directly into the database, allowing users to manage business logic, create non-traditional data types, and define complex database functions in an integrated, intelligent information management system. With Informix Dynamic Server.2000, users benefit from the performance and scalability offered by traditional relational database, while gaining all the advantages of object-oriented technology and unlimited extensibility.

Developed specifically to handle the complex needs of the Internet, Informix Internet Foundation.2000 provides unparalleled power to publish, transact, and analyze business application data. It is designed to publish business data on the Internet through delivering relevant and timely content to customers and to consistently capture critical information about customers. Informix Internet Foundation.2000 is also an Internet transaction and analysis engine that powers secure, reliable, and highly available Web applications for business-to-consumer and business-to-business systems. With Informix Internet Foundation.2000, businesses are provided with the ability to analyze Internet transactions to better understand customers, market trends, and demographics, allowing them to make critical business decisions to succeed in today's competitive climate.

Informix Internet Foundation.2000 supports all Internet programming standards such as Java and XML. These programming languages have been tightly integrated with the database server to deliver maximum performance and scalability, making Informix Internet Foundation.2000 an ideal platform for hosting scalable Internet application development and deployment. By incorporating extensibility directly into its core server, Informix provides businesses with unlimited capacity to grow and adapt to ever-changing needs.

With Informix Internet Foundation.2000, organizations now have the best platform to rapidly and easily move their business to the Internet.

INFORMIX INTERNET FOUNDATION.2000 ADVANTAGES

Key advantages of Informix Internet Foundation.2000 include:

- support for any kind of data imaginable, whether they are complex data types such as spatial, time-series, or 3-D, or user-defined data types which empower customers to define data structures according to their business needs;
- DataBlade® modules manage newly defined data types with the same flexibility and reliability as built-in data types;
- application programming standards such as Java and XML server programmability are tightly integrated with the database server;
- single database architecture across all operating environments (UNIX, Linux, and Windows NT) ensures maximum performance, transactional consistency, and data integrity;
- full RDBMS functionality across all hardware architectures (uniprocessor, symmetric multiprocessing, and clustered systems) enables seamless migration of applications, data, and skills;
- maximum performance and scalability through a superior multithreaded parallel processing architecture;
- high database availability for supporting a wide range of business-critical applications on open systems platforms;
- dynamic, distributed on-line system administration for monitoring tasks and distributing workloads.

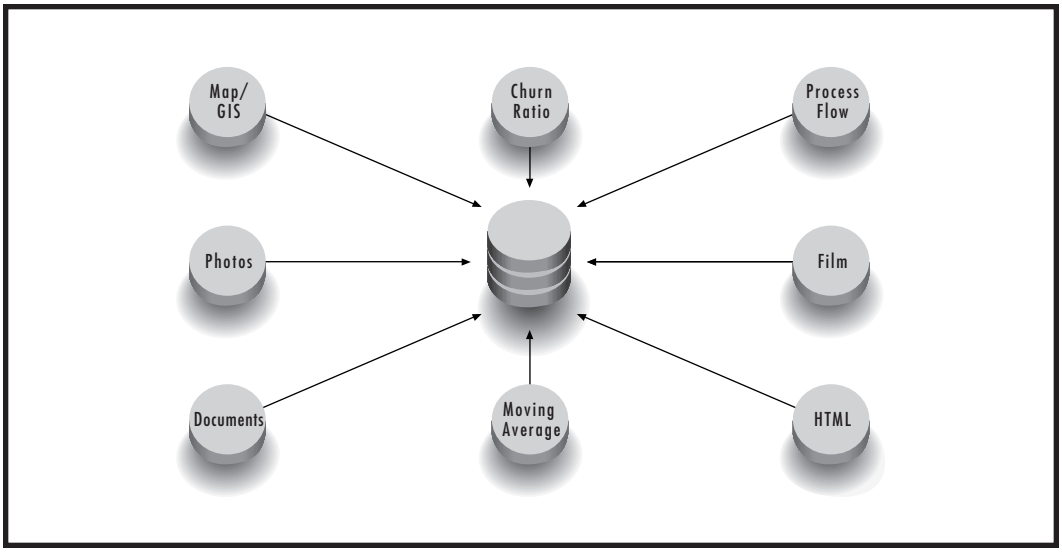

Figure 1: Informix Internet Foundation.2000 supports all kinds of data.

Support for Any Kind of Data Imaginable

Informix Internet Foundation.2000 lets organizations support a universal spectrum of emerging data types (including images, sound, video, HTML pages, 2-D and 3-D spatial data, text documents, etc.) along with unanticipated information types (including reporting period, profit, cost per unit, etc.) that are unique to a specific business requirement. Moreover, these rich data types are stored in the database as native data types, providing users with unprecedented flexibility for publishing content-rich Web sites. Furthermore, by supporting complex data as native type, Informix Internet Foundation.2000 is less expensive to deploy and support.

Add New Functionality Through DataBlade Module Technology

DataBlade modules are Informix's exclusive technology that lets developers add new intelligence to Informix Internet Foundation.2000. DataBlade modules can be viewed as object-oriented "packages" that encapsulate specialized data types, operations that process the data, and access methods that index the data. They can be "plugged" into the data engine to do anything with any data.

DataBlade modules can be used to encapsulate existing application code to enable more efficient execution, as well as offer the flexibility to run the application code wherever it will best perform: within the database server, within an application server, or within the client. While multidata type queries run best on the database server, users are provided with the flexibility to place the logic wherever it makes most sense.

Supports Internet Application Development Standards

Informix Internet Foundation.2000 supports Internet application development standards such as Java, XML, and HTML, and lowers total cost of ownership by leveraging existing standards in development tools, systems infrastructure, and customer skill sets. By providing an Internet-ready data engine, Informix Internet Foundation.2000 is capable of hosting scalable Internet application development and deployment for new business process applications such as e-commerce and media asset management.

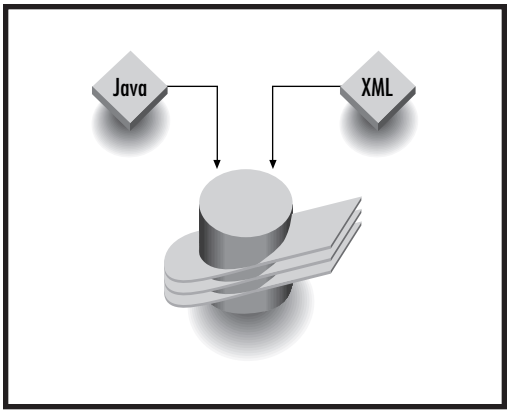

Figure 3: Informix Internet Foundation.2000 supports popular Internet development standards such as Java, XML, and HTML.

Single-Server Architecture

Informix Internet Foundation.2000 employs a single-server architecture across UNIX, Linux, and Windows NT, providing customers complete freedom to select the operating environment that best meets their current information needs. To accommodate future growth, Informix provides feature parity for all supported operating systems, ensuring ease of database and application migration. Additionally, all components of Informix Internet Foundation.2000 are tightly integrated with each operating environment, making application services a simple and natural extension to LAN and OS services. For example, on Windows NT, Informix Internet Foundation.2000 fully exploits native Windows NT threads, asynchronous disk and network I/O drivers, and is integrated with Windows NT Registry, Event Log, and various other Windows NT services. On Linux, Informix Internet Foundation.2000 supports features such as Apache integration, HOWTO, RPM, ESQL/C, ODBC, etc.

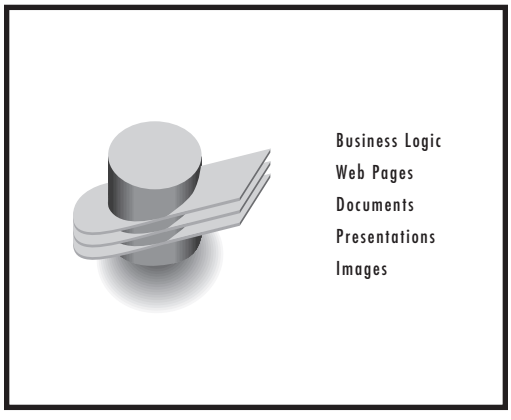

Figure 2: Informix DataBlade technology lets Informix Internet Foundation.2000 do anything with any data.

By implementing a single-server architecture, Informix Internet Foundation.2000 can be extended to integrate any type of data and function. This centralized server implementation ensures transactional consistency and data integrity, as well as simplifies data administration and management tasks. Furthermore, since only one server is involved, the optimizer can efficiently determine the optimum execution path for multi data type queries, thus resulting in better query performance.

Other competitive database management systems rely on middleware to link multiple servers managing different data types. This type of solutions not only compromises performance but also transactional consistency and integrity, as problems with the network can lead to problem with the data.

Seamless Migration

Informix Internet Foundation.2000 allows customers to migrate their database across computing environments (uniprocessors, SMP, and clusters) and operating systems (UNIX, Linux, and Windows NT). As such, data, applications, and skills developed from one Informix database environment can be easily transferred to a different hardware and operating environment without a tedious migration process.

Informix is unique in its support of application transparency, which is achieved through isolating parallelism at the database server level. The server functions based on an automated determination of how the data is distributed and the amount of processing resources that are available to complete a query. It then automatically determines whether the SQL statement can be parallelized and, if so, the degree of parallelism. Applications do not need to be tailored to the specific platform they will run on. Therefore, as users outgrow their current hardware environments, they can easily move to more powerful systems without worrying about recoding their applications.

Maximum Performance and Scalability

At the core of Informix Internet Foundation.2000 is Informix Dynamic Server.2000, Informix's premier data engine. Informix Dynamic Server.2000 is based on a parallel database architecture, which has been built from the ground up to provide core internal parallelism, enabling all major database operations, such as I/O, complex queries, index builds, log recovery, and backups and restores to execute in parallel across all available system resources. This parallel database architecture provides the unique ability to intelligently optimize performance and fully exploit the inherent processing power of any hardware.

High Availability

To respond to the increasing need for higher availability, a multitude of features are provided to ensure round-the-clock database processing. These features include on-line administrative utilities for dynamic tuning, backup and recovery, and table reorganization; functionality such as enterprise replication that promotes fault resilience; cluster and data failover, and software mirroring; and enhancements that enable Informix technical support to diagnose problems quicker. Together, these features minimize planned downtime by allowing administrators to perform database maintenance operations online and reduce unplanned downtime by working around any faults that may occur, delivering a highly available database environment for all types of mission- and business-critical processing.

Dynamic, On-Line Administration Tools

Informix delivers a suite of mainframe-caliber administration/management tools that are optimized for performance, availability, and ease of use. To ensure regular administrative tasks are executed in the fastest time possible, many utilities have been parallelized to maximize performance. Informix also offers a complete set of on-line utilities to maximize availability. And through Informix Enterprise Command Center (IECC), administrators are provided with the flexibility to manage multiple remote databases from a single, centralized console, hence significantly reducing the amount of work associated with managing databases in a distributed environment.

COMPONENTS OF INFORMIX INTERNET FOUNDATION.2000

Informix Internet Foundation.2000 consists of a core database engine, the Informix Dynamic Server.2000 and a set of tools for facilitating Internet application development and deployment.

Informix Dynamic Server.2000

Informix Dynamic Server.2000 is designed to deliver unprecedented flexibility and extensibility. This is accomplished through Informix Dynamic Server.2000's object-oriented technology, which provides support for rich new content by the way of defining new data types as objects and manipulating them through user-defined functions. Leveraging Informix's advanced, extensible DataBlade technology, newly defined data types and functions can be encapsulated into a reusable plug-in package, ensuring management of any kind of data and function with fully optimized relational access.

Informix Dynamic Server.2000's core architecture was designed from the ground up to provide built-in multithreading and parallel-processing capabilities. Multithreading is achieved through managing user requests in the form of lightweight mechanisms called threads, which are scheduled and processed via a pool of database processes (called virtual processors). Parallel processing is achieved through dividing large user tasks into subtasks, thus enabling processing to be distributed across all available resources. Together, multithreading and parallel processing ensure the most efficient use of all system resources, delivering the scalability and performance needed for all types of database processing.

Object-Relational Extensibility

Informix Dynamic Server.2000 provides a complete set of features to extend the database server. They include support for new data types, database routines, access methods, DataBlade module support, server-side application programming interfaces (APIs), and client-side APIs.

Data Types

Informix Dynamic Server.2000 supports three data-type categories: built-in data types, user-defined data types, and complex data types. Built-in data types are standard RDBMS data types (such as alphanumeric, integer, etc.) User-defined and complex data types allow data to be more intelligently stored and processed in a way that meet a company's true business objective. They are new data types that are defined to support an application domain. Once defined, they are treated like standard data types managed by the server to provide the highest level of performance and scalability. Values of complex and user-defined types may be stored, examined using queries or function calls, passed as arguments to database functions, and indexed in the same way as the core built-in types.

Built-In Data Types

Built-in data types are data types supported in traditional relational database servers. The following are examples of built-in data types:

- character data types such as CHAR, VARCHAR, and LVARCHAR;
- numeric data types such as DECIMAL, MONEY, SMALLINT, INTEGER, and FLOAT;
- large-object data types including simple-large-object types such as TEXT and BYTE and smart-large-object types such as CLOB and BLOB;
- time data types such as DATE, DATETIME, and INTERVAL; and,
- miscellaneous data type such as BOOLEAN.

User-Defined Data Types

User-defined data types allow customers to define the data structures according to their business needs. These data types are treated like standard data types such that they can be indexed and optimized as any other type of data, and standard calculations can be performed against them. User-defined data types can be either `OPAQUE` or `DISTINCT`:

- `OPAQUE` types allow any data that can be represented in C or C++ to be natively stored and processed by the server. This allows applications already implement data types as C structures to be easily encapsulated in a DataBlade module. `OPAQUE` types provide developers complete control over how the data is stored and processed, enabling highly efficient performance through specialized access. Like other types, `OPAQUE` types still gain the proven manageability and integrity of Informix's database architecture. Furthermore, `OPAQUE` type data is fully and automatically recoverable and managed by all the same database facilities as built-in types.
- `DISTINCT` types provide a fast way to create new data types that are structurally equivalent to an existing type (including other user-defined types) but, additionally, have customized processing characteristics. For example, *PROFIT* could be a `DISTINCT` type created with the same representation as REAL. All routines that operate on REAL values have database server-simulated counterparts for *PROFIT* values. However, *PROFIT* values and REAL values cannot be added, subtracted, or compared with one another without converting one data type to another data type. Additionally, because *PROFIT* is a `DISTINCT` type, it cannot be confused with any other type, even if it is numeric in nature, such as temperature.

Complex Data Types

Complex data types is a composite of other existing data types. For example, it may consist of built-in types, `opaque` types, `distinct` types, or other complex types. A key differentiator of complex types is that individual components of a complex data type can be accessed and manipulated using SQL statements. Two complex data types are supported: `ROW` and `COLLECTION`.

A `ROW` type can be thought of as a row of columns, of varying data types, stored in a single database table column or row. `ROW` types follow essentially the same rules as database tables. The columns within a `ROW` type are called fields. They can be almost any data type, including other complex types, and can be access individually.

For example, address information can be composed of several columns such as street, city, state, and zip code. These columns can be combined into a single column for more rapid access. Composed of built-in or user-defined data types, this data type can streamline database design, reduce application time, and allow applications to run more efficiently.

Instead of having additional columns in the EMPLOYEE table, the ROW type groups data that is most often accessed together in one column. The table EMPLOYEE consists of the columns Name(VARCHAR(30)), Address(address_t), and Contact_info(SET(LVARCHAR)). The ROW type address_t consists of the named fields Street(VARCHAR(20)), City(VARCHAR(20)), State(CHAR(2)), and Zip_code(CHAR(5)).

COLLECTION types are like a nested table within a column. This data type avoids the costly duplication of key values and frequent joins caused by a typical normalization approach. It solves the problem of relational databases assigning multiple values to a column for a single record, thereby maximizing database design efficiency, reducing amount of coding, and simplifying code management.

Elements within a COLLECTION can be of any type: built-in data types, OPAQUE, DISTINCT, or ROW. It can be accessed and updated in much the same way as tables, using minor extensions to the current SQL language.

Figure 5 illustrates a collection in a column called Contact_info.

Instead of putting contact information in a separate table, all the information is contained in one row, using a COLLECTION type. Elements within the COLLECTION type can be added or removed without altering the table's columns.

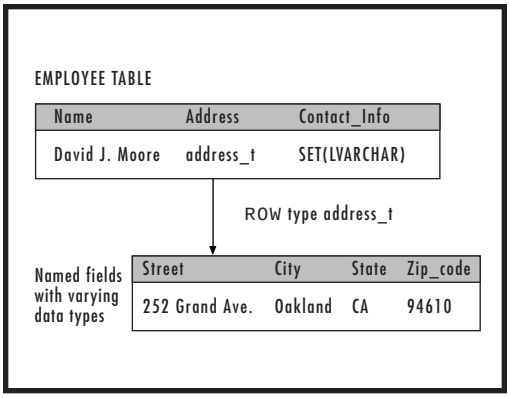

Figure 4: Sample ROW type.

Support for Large Objects

Informix Dynamic Server.2000 supports both simple large objects (also known as binary large objects or BLOBs) as well as smart large data objects (called character large objects or CLOBs). BLOBs support BYTE and TEXT data types, which can be used to store up to two gigabytes of binary data as a field in a database record. BLOBs are treated like any other database data and are accessible through SQL.

Informix Dynamic Server.2000 also supports smart large objects or CLOBs. Smart large objects are used to support user-defined data types such as video and audio clips, pictures, large text documents, and spatial objects such as drawings and maps. In essence, smart large objects provide the ability to seek, read from, and write to segments within the object and may be partially read or written to from the client or the server. Additionally, smart large objects are recoverable and obey transaction isolation modes.

Inheritance

Inheritance is a process that allows a data type or a table to acquire the properties of another type or table, thus simplifying data design by easing the definition of new objects from others. Inheritance encourages incremental modification; a data type or table can inherit a general set of properties and add properties that are specific to itself.

For example, an application developer may be creating a medical information system in which an orthopedic patient is defined as a subclass of patient. Using inheritance, the orthopedic patient class derives much of its structure and function from the patient class, adding only what makes it unique, perhaps a set of X-rays, or other information required by the orthopedic specialization. This avoids the need to redesign and recode from scratch for data types or table.

Typed-Based System

Type system refers to how a database server handles comparisons among dissimilar data types. Informix Dynamic Server.2000 is a strongly typed system, which means that unlike data types cannot be directly compared without casting them into a comparable form. For example, a price quoted in U.S. dollars will not be directly compared with a price quoted in Japanese Yen without executing a function that looks up the exchange rate and performs a conversion. This feature prevents errors common in weakly typed system, which allows simple numerical comparison on prices quoted in different currencies.

User-Defined Routines

User-defined routines extend the processing and aggregation functions of the database to provide new domain-specific capabilities that are integrated with the server. They can be used to capture business logic or commonly used application logic and run that logic in the server, where there is the greatest processing power. By centralizing this logic, the developer can reduce the time it takes to develop an application, as well as increase the application's speed.

Once defined, a user-defined routine is registered in the system catalog tables and is invoked by an SQL statement or another routine. This routine can either be a function or a procedure, and can perform a wide variety of tasks such as encapsulating multiple SQL statements and support new data types. User-defined routines can also accomplish tasks that address new technologies such as search graphical data and collect data from Internet end users.

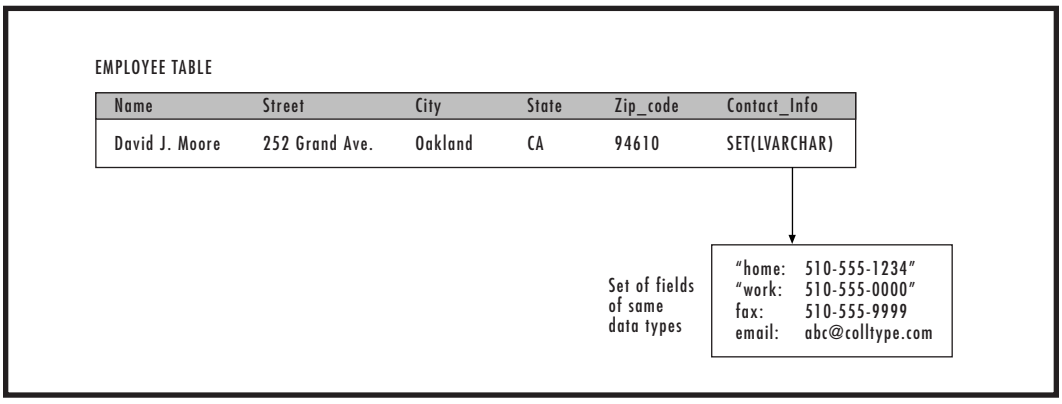

Figure 5: Sample `COLLECTION` type.

As is the case when defining new data types, there are several ways of defining new routines. User-defined routines can be written using Informix's stored procedure language (SPL), or third-generation languages such as C, C++, or Java. SPL routines contain SQL statements that are parsed, optimized, and stored in the system catalog tables in executable format, making it ideal for SQL-intensive tasks. Routines written in third-generation languages are compiled and loaded into a shared object file or dynamic link library (DLL). The routine and name of the shared object are declared to the server and once invoked, the shared object is linked to the server. Since C, C++, and Java are powerful, full-function development languages, routines written in these languages can carry out much more complicated computations than SPL functions.

User-Defined Aggregates

User-defined routines can be used to extend the functionality of aggregates in the database server. Informix Dynamic Server.2000 provides two ways to extend aggregates: 1) extension of built-in aggregates, and 2) creating new aggregates. Extension of built-in aggregates refers to extending the capability of system-defined aggregates such as AVG, SUM, MIN, MAX, and COUNT to work with user-defined data types. This functionality allows reuse of existing client applications because it requires no new SQL syntax for aggregates.

Informix Dynamic Server.2000 also supports creation of new aggregates to provide new aggregate functions that the database server does not provide. For example, system-defined aggregates do not offer capability such as the median employee salary, the second largest employee salary, and the standard deviation of employee salaries. In order to perform such calculation, users must retrieve a collection of values into a user program to perform the computation manually. This may create a serious performance bottleneck because a large data set must be transmitted over a client-server connection. By allowing users to define new aggregate operations inside the database server, users can now easily calculate complex operations with one command. Furthermore, like system-defined aggregates, user-defined aggregates can be executed across multiple threads to increase performance.

Expensive Function Optimization

Some user-defined routines, such as ones that perform image processing functions, can be expensive to evaluate and can affect the performance of queries that utilizes these user-defined routines. To improve performance, the Informix Dynamic Server.2000 optimizer uses cost and selectivity information to estimate the number of rows returned from a query to calculate the total cost of executing a query and uses this information to select the best query plan. The optimizer also makes sure that expensive user-defined routines are evaluated last.

Parallelizing User-Defined Routines

To enhance performance, a user-defined routine can be executed across multiple processors if it is executed within the context of parallel data query. Two classes of virtual processors can be used for parallelizing user-defined routines: the user-defined virtual processors for routines written in C, and Java virtual processors for routines written in Java.

Access Methods

An access method consists of software routines that access disk storage, retrieve data into memory, and write data back to permanent storage. Informix Dynamic Server.2000 supports two different access methods: primary and secondary. Developers can also create their own access method for accessing data stored in a non-Informix database server.

Primary Access Method

A primary access method provides a relational-table interface for direct read and write access. A primary access method reads directly from and writes directly to source data. It provides a means to combine data from multiple sources into a common relational format that the database server, users, and application software can use.

Secondary Access Method

A secondary access method provides a means to index data for alternate or accelerated access. An index consists of entries, each of which contains one or more key values and a pointer to the row in a table that contains the corresponding value or values. The secondary access method maintains the index to coincide with inserts, deletes, and updates to the primary data.

Informix Dynamic Server.2000 supplies two secondary access methods: B-tree index and R-tree index. A B-tree index organizes index information in a less than, greater than order and is arranged as a hierarchy of pages. It is ideal for accelerating queries on linearly ordered, one-dimensional data. However, many new kinds of rich data are multidimensional in nature and take advantage of nonlinear ordering. For example, spatial data can be 2-D or 3-D. To address these rich data types, Informix Dynamic Server.2000 introduces R-tree index, which can index points or volumes, in two or more dimensions, or ranges in a single dimension.

User-Defined Access Methods

Informix Dynamic Server.2000 also provides the ability to define new access methods. User-defined access methods provide SQL access to data in a table in either an external location or in a smart large object. The ability to create new access methods allows Informix Dynamic Server.2000 to unify all heterogeneous data distributed throughout an organization (refer to “Server-Side Application Programming Interface [API]” for more information).

DataBlade Module Support

A DataBlade module is a collection of database objects and code that extend Informix Dynamic Server.2000 by enabling developers to add new functionality attuned to the needs of a specific application. They can be viewed as an object-oriented “package” that encapsulates specialized data types, operations that process the data, and access methods that index the data. By doing so, a DataBlade module lets the server provide the same level of support new data types that it provides for built-in data types, thereby adding greater intelligence to the database server and enabling users to manage any kind of information.

In addition to specialized data types, routines that work across those data types, and access methods to allow specific indexes work on those types of data, a DataBlade module also includes a SQL interface to enable DataBlade modules to interoperate with one another. Figure 6 illustrates the components of a DataBlade module.

The SQL interface is a collection of functions that conforms to a standard and exports a predictable service, enabling DataBlade modules to share with each other the services they rely upon or provide. For example, a DataBlade module that performs image matching might allow users to supply text captions for images. If a keyword search interface is registered in the database, then the image matching DataBlade module can use the keyword search routines to match on the captions. Informix is standardizing several DataBlade modules, such as those for text and image matching, so that they provide consistent interfaces, enabling customers to choose the DataBlade module most appropriate for their needs and allowing DataBlade modules to be combined like building blocks.

Prepackaged DataBlade Modules

With Informix Dynamic Server.2000, users can choose from a library of DataBlade modules written by both Informix and third-party vendors. These professionally certified extensions are designed specifically to enable users to efficiently store, retrieve, update, and manipulate any new kinds of data. Each DataBlade module can be easily "snapped" into the database server and used individually or in conjunction with other DataBlade modules, just as a general-purpose utility knife can be extended to perform different cutting jobs by inserting special-purpose blades.

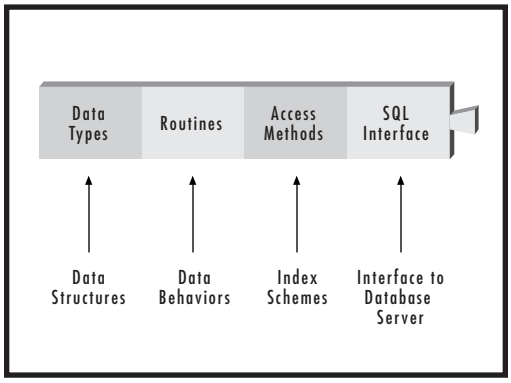

Figure 6: DataBlade module components.

DataBlade Development Kit

If a DataBlade module does not exist to fit the customer's particular need, the customer can create his or her own DataBlade modules using the DataBlade Development Kit. This level of flexibility lets the database server accommodate new business requirements as they evolve, allowing customers to leverage their existing investments in database technology and still preserve the flexibility required to react competitively in any situation.

Server-Side Application Programming Interface (API)

Informix Dynamic Server.2000 offers two server-side APIs for the developing of user-defined access methods. They include the virtual table interface and virtual index interface, and the DataBlade API.

Virtual Table Interface and Virtual Index Interface

Users can utilize Informix's Virtual Table Interface™ (VTI) and virtual index interface (VII) to create user-defined access methods. VTI is an open interface for implementing primary access method in an external or specialized data source. VII is an open interface for implementing a secondary access method in an external or specialized data source.

To increase performance, Informix has integrated query parallelism with user-defined access method. This feature allows scans on tables defined via VTI and indices defined via VII to be performed in parallel, resulting in faster response time.

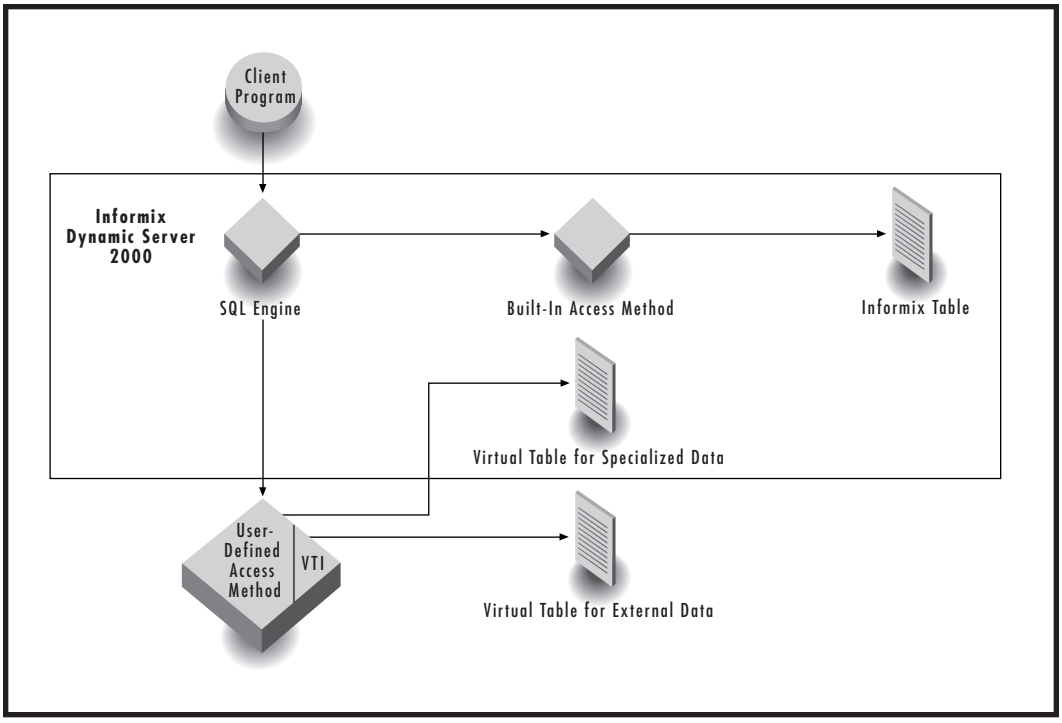

Figure 7: Using VTI to access information inside and outside of Informix Dynamic Server.2000.

DataBlade API

The DataBlade API includes functions and data structures that enable an application to add functionality to the database server. It provides functions that allow developers to manage database connections, execute SQL statements, process query results, manage server events and errors, and manage database server memory.

Client-Side APIs

Informix Dynamic Server.2000 offers a set of client APIs to allow programmers to embed SQL statements directly into programming languages. These APIs include:

- the C++ interface, which lets programmers develop client applications using object-oriented C++ programming languages by encapsulating Informix database server features into an easy-to-use class hierarchy and extensible object library;
- the Informix JDBC Driver, version 1.x and 2.0, which provide standard connectivity between Java applications and any Informix database on all platforms;
- Informix Embedded SQLJ, which allows developers to embed SQL statements into Java applications;
- the GLS interface, which allows programmers to write programs (or change existing programs) to handle different languages, cultural conventions, and code sets;
- the INFORMIX-ESQL/C, which allows programmers to embed SQL statements directly into a C program;
- the INFORMIX-ODBC, which enables programmer to create custom applications using ODBC for accessing external data sources; and,
- the Informix OLE DB Provider, which lets programmers develop OLE DB applications for accessing Informix Dynamic Server.2000.

Multithreading

Within the database server, a configurable pool of database server processes called "virtual processors" is used to schedule and manage user requests. User requests are represented by lightweight mechanisms called "threads," which are a single sequential flow of control that represents a discrete task within a database server process. Unlike other database products that manage user requests at the operating system process level, virtual processors manage active threads at the database level in the form of multithreading.

Virtual processors are designed with built-in intelligence to efficiently coordinate among multiple concurrent threads. Threads are spawned, queued, and serviced by the first available virtual processor, ensuring efficient hardware utilization with no bottlenecks. A "thread scheduler" is able to take advantage of in-depth knowledge of database objects and algorithms, so it can provide "smarter" scheduling than a general-purpose operating system scheduler.

When one thread is waiting for a resource, a virtual processor can work on behalf of another thread. By providing this flexibility, only a small number of operating system processes are required to manage a larger number of users because each virtual processor can respond to multiple user requests. Although many virtual processors may be spawned to respond to user requests, to the user it appears as one database server.

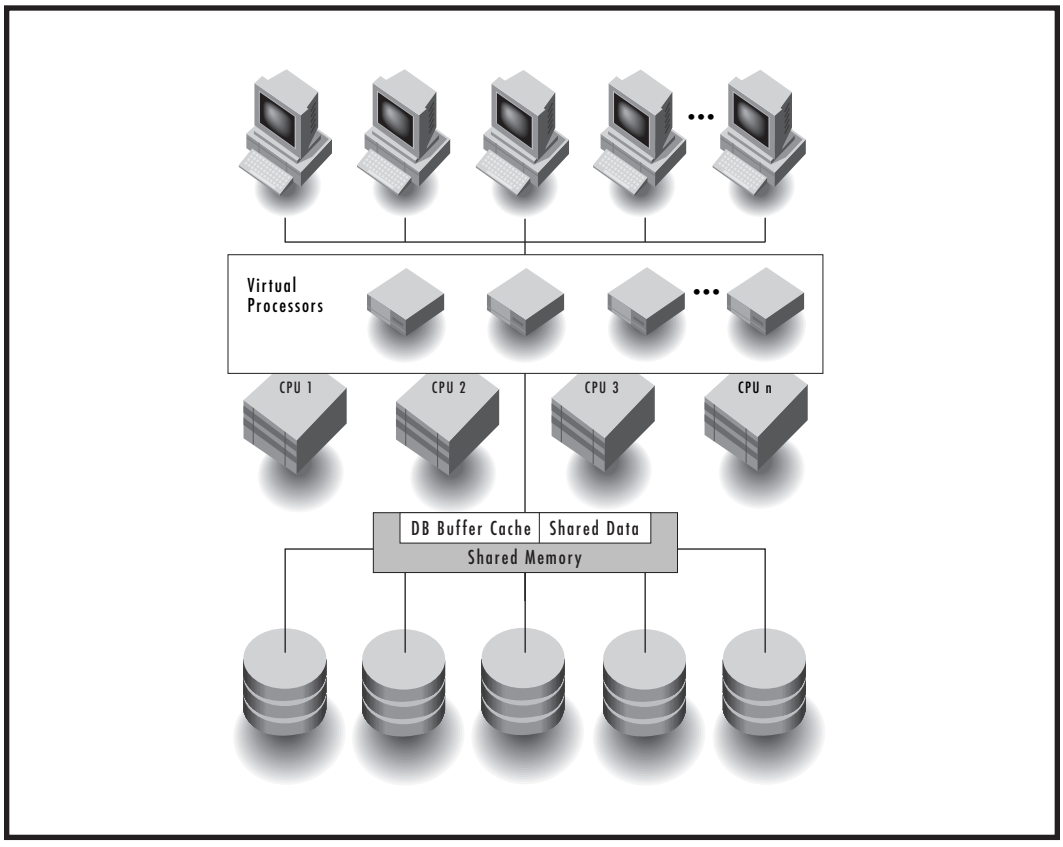

Figure 8: Informix Dynamic Server.2000 consists of a configurable pool of database server processes, called virtual processors, that can respond to any client's request.

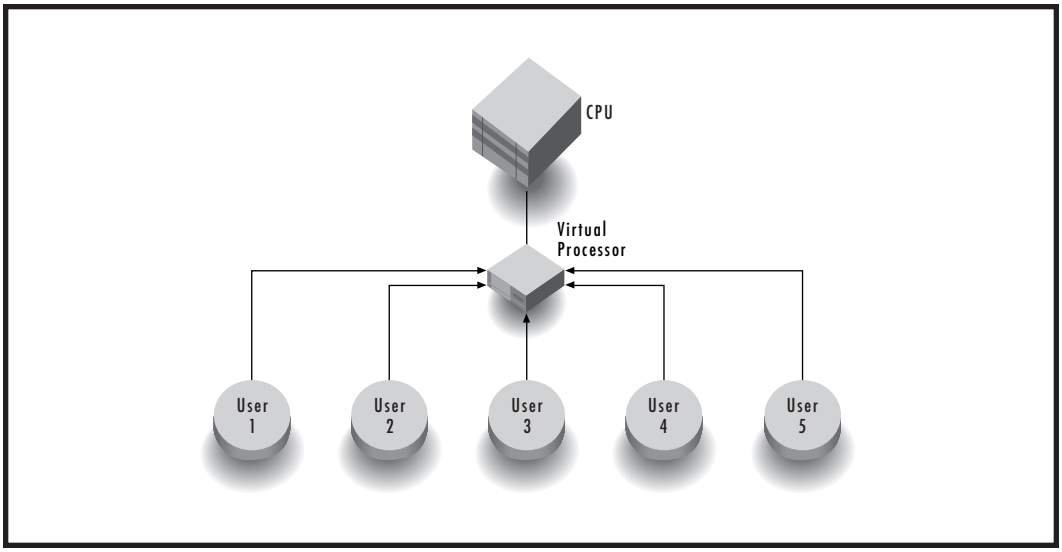

Figure 9: A virtual processor can respond to many user requests.

Because the number of database server processes required is significantly reduced, less context switching within the operating system is required—which enables the Informix Dynamic Server.2000 to bypass most performance restrictions and burdens typically imposed by the operating system. And with Informix Dynamic Server.2000, you can take advantage of special scheduling features provided by hardware vendors (such as processor affinity), without adversely affecting your overall system performance.

Parallel Processing

To ensure most effective utilization of system resources, large database tasks are broken into individual subtasks so that they can be executed in parallel across multiple CPUs and disks. By dividing tasks into subtasks and running the subtasks in parallel, Informix Dynamic Server.2000 can significantly decrease the execution time of complex operations. For example, for a processing-intensive request such as a multitable join, the task can be divided into multiple database subtasks and spread across all the available virtual processors in the system.

In addition to a multithreaded architecture, parallel processing is facilitated through a number of features including parallel data query (PDQ), table partitioning, and specialized virtual processor classes.

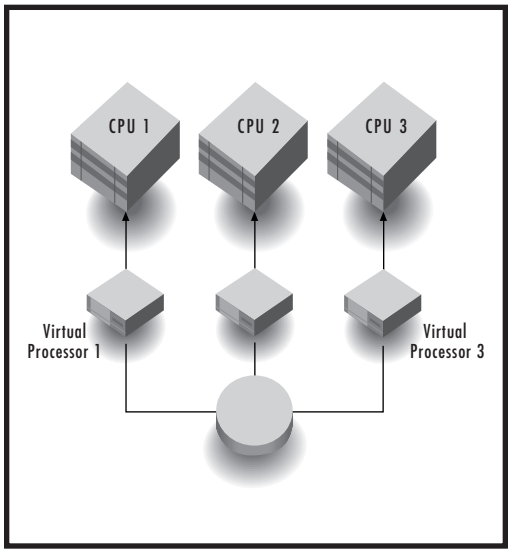

Figure 10: Many virtual processors can be spawned to respond to a user request.

Parallel Data Query

As symmetric multiprocessor, loosely coupled clusters, and massively parallel processor architectures are utilized for database processing, PDQ will take complete advantage of the underlying CPUs and execute tasks many times faster than competing database architectures. In some cases, queries could complete in minutes instead of hours, or seconds instead of minutes. This performance benefit applies to all complex database operations that require sorting, scanning large amounts of data, joining tables, and performing aggregations.

Let's take a simple join, for example. When joins happen serially, users have to wait for one task to finish before the database begins the next one. In other words, the database must first scan, then join, then sort, then send the results to the user. PDQ achieves two processing economies. First it processes these tasks concurrently, in parallel. Second, it breaks each individual task into subtasks, taking full advantage of the server's built-in core parallelism. This ability dramatically reduces overall processing time.

With PDQ, users can perform all types of database operations in parallel, whether it is within an SMP node, or across multiple loosely coupled SMP or MPP nodes. These database operations include parallel sort, scan, insert, delete, join, aggregation, index build, and a number of database administration functions. This full functionality ensures scalability for all database operations and enables the implementation of very large databases on open systems for a wide range of applications including OLTP, e-commerce, and media asset management.

Table Partitioning

Table partitioning improves parallel processing performance and high availability. It also makes managing very large databases much easier by breaking the database into smaller sections.

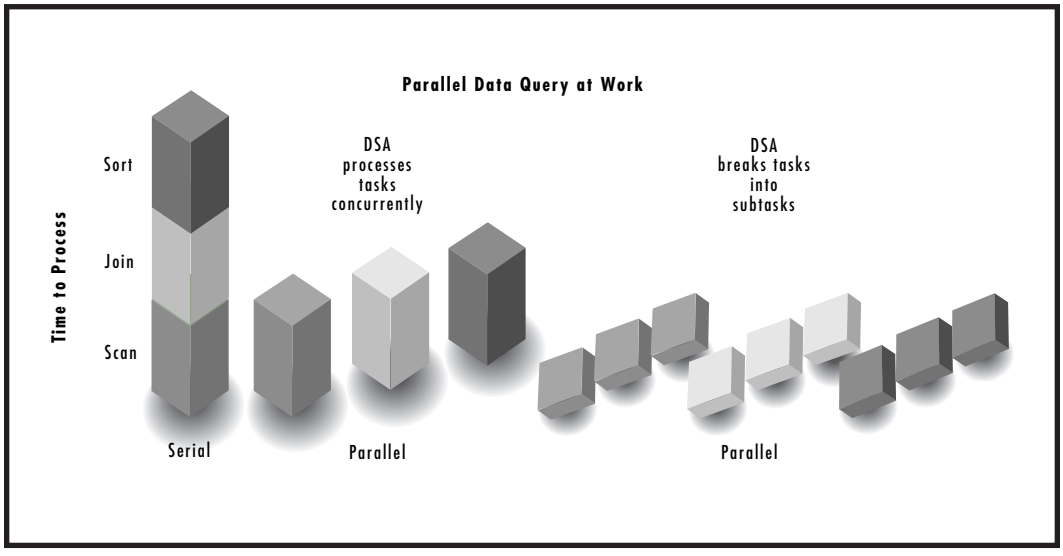

Figure 11: Dynamic Scalable Architecture (DSA) is designed to process tasks (such as scan, join, and sort) concurrently, then break them into subtasks to greatly reduce processing time.

Informix Dynamic Server.2000 allows table partitions to be set and data distribution changes to be altered without interrupting the database server. In addition, partitioning schemes are transparent to applications and end users. The partitioning schemes can be set using simple round robin (every record goes to the next partition in sequence), hashed (an algorithm applied to the record key determines its partition number), or expression methods (each partition gets a set of records based on their key values) via SQL statements such as CREATE TABLE and ALTER TABLE. These partitions can be monitored and tuned when necessary.

The ability to divide a single table partition into two or three new table partitions is essential in order to reap the benefits of parallel processing. For instance, by recognizing the scheme by which the data has been partitioned, the database server is able to skip partitions that are uninvolved in a particular query. At the same time, the database server can skip partitions unavailable due to a system crash, maintaining high availability.

In addition to partitioning data tables, Informix Dynamic Server.2000 can also partition indexes to allow for maximum data layout flexibility, resulting in optimal parallel processing performance. DBAs can place a partitioned index on a different partition than the data. These partitioned indexes can have their own partitioning scheme, separate from data tables.

Partitioning is also critical for efficient system administration. The larger the database, the more important it becomes for the system administrator to be able to perform operations—such as archive and restore, and bulk load and unload—at the table partition level, rather than having to archive or restore the entire database or table. Such operations can occur in parallel, greatly reducing the time required to load, unload, or restore data.

Virtual Processor Classes

For efficient execution and versatile tuning, virtual processors are grouped into classes—each optimized for a particular function such as CPU operations, disk I/O, client/server communication, and administrative tasks. By delegating the virtual processors into classes, Informix Dynamic Server.2000 is able to effectively schedule and prioritize operations.

Threads are transparently scheduled across the virtual processors in the relevant class. You can configure your system with enough virtual processors in each class to handle the specific type of workload on that particular system. And, since the pool of virtual processors can be easily adjusted on line, you can quickly tune the number within each class, or make changes to accommodate periods of heavy activity or load mixes.

While there are many virtual processor classes, the most important is the CPU class, which can be increased and decreased as CPU processing needs dictate. If there is a long queue for CPU processing, you can dynamically start up a CPU virtual processor to alleviate that bottleneck without interrupting a single user on the system.

Since the number of virtual processors required to handle client requests is often unpredictable, several virtual processor classes for client/server communication protocols (such as IPX/SPX, TCP/IP, and shared memory) can be implemented to handle client communication. An optical virtual processor class can even be added to handle the I/O processing if you need to add optical media storage devices to your system.

Dynamic Shared Memory

All memory used by Informix Dynamic Server.2000 is shared among the pool of virtual processors. That way, the database server can be configured to automatically add more memory to its shared memory pool in order to process client requests more expediently.

Data from the read-only data dictionary (system catalog) and stored procedures is shared among users rather than copied, resulting in optimized memory utilization and fast execution of heavily used procedures. This feature can provide substantial benefit in many applications, particularly those accessing many tables with a large number of columns and/or many stored procedures.

Informix Dynamic Server.2000 also allocates an area, called the threads stack, in the virtual portion of shared memory to store non-shared data for the functions that a thread executes. The threads stack tracks the state of a user session and enables a virtual processor to protect a thread's non-shared data from being overwritten by other threads concurrently executing the same code. Informix Dynamic Server.2000 dynamically grows the stack for certain operations such as recursive stored procedures.

Informix Dynamic Server.2000's shared memory minimizes fragmentation so that memory utilization does not degrade over time. Beyond the initial allocation, shared memory segments are automatically added in large chunks as needed, but can also be added by the administrator while the database is running. The memory management system will also attempt to automatically

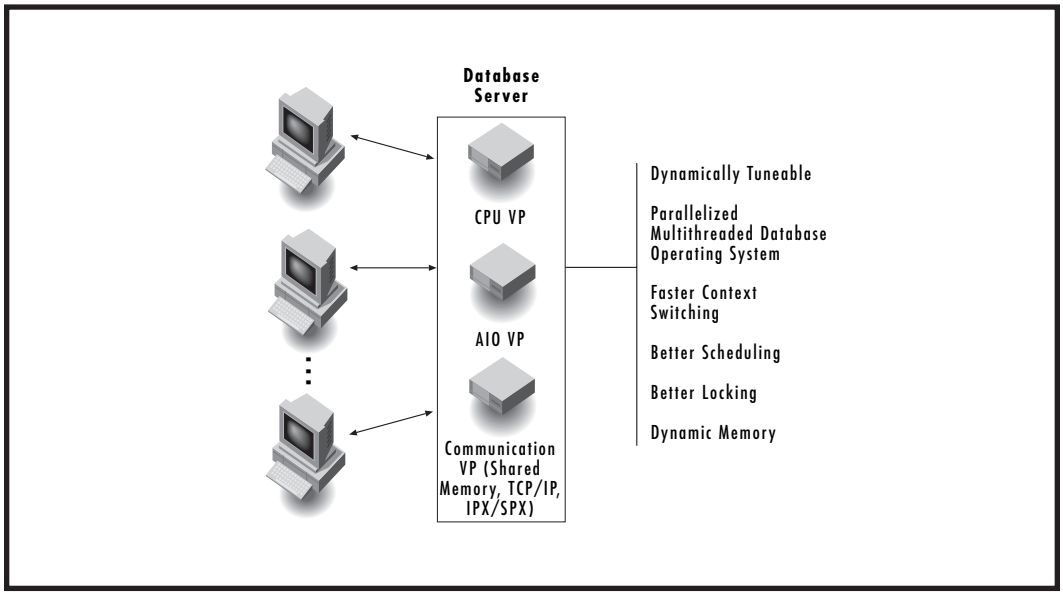

Figure 12: Virtual processors are grouped into classes which are optimized for a particular function. Informix Dynamic Server.2000 may be configured with the appropriate number of virtual processors in each class to handle the specific workload on the system.

grow the memory segment when it runs out of memory. When a user session terminates, the memory it used is freed and reused by another session. Memory can be reclaimed by the operating system by freeing the memory allocated to the database. User threads can therefore easily migrate among the virtual processors, contributing to Informix Dynamic Server.2000's scalability as the number of users increase.

Asynchronous I/O
I/O is typically the slowest component of database processing. Informix Dynamic Server.2000 uses its own asynchronous I/O (AIO) package (or the operating system's kernel AIO when available) to speed up I/O processing. Because Informix Dynamic Server.2000's virtual processors service user I/O requests asynchronously, a virtual processor will never have to wait for an I/O to complete before beginning work on another service request.

There are four specific classes of I/O virtual processors: logical log I/O, physical log I/O, AIO, and kernel asynchronous I/O (KIO). Separating I/O into these classes allows for an efficient prioritization scheme. And since I/O requests are uniformly scheduled, Informix Dynamic Server.2000 can effectively keep all available disks busy. In fact, you can use Informix Dynamic Server.2000's System Monitoring Interface to detect long queues for reads or writes from disk, then start up additional virtual processors specializing in I/O to alleviate I/O bottlenecks.

Read Ahead
For sequential table or index scans, Informix Dynamic Server.2000 can be configured to asynchronously read several pages ahead from disk while the current set of pages in shared memory is being processed. With this ability, throughput is maintained, because applications spend less time waiting for disk accesses to complete.

Other Performance Features
In addition to parallel processing and data partitioning, Informix Dynamic Server.2000 provides a host of performance features designed to improve both OLTP and decision-support processing.

Memory Grant Manager
The memory grant manager (MGM) allows DBAs and programmers to control the degree of parallelism by balancing the priority of user requests with available system resources. MGM performs the following:

- adjusts the amount of system resources needed for PDQ-type tasks;
- sets the priority of each query;
- adjusts the number of complex queries allowed to run concurrently;
- adjusts the maximum amount of memory utilization for both decision support and OLTP; and,
- works in conjunction with the cost-based optimizer to ensure the fullest degree of parallelism at all levels.

Memory grant manager's efficient memory utilization is the key to optimizing parallel processing. MGM also enforces aggregate resource limits for PDQ processing. And with MGM, all system-wide priority settings can be changed dynamically.

Cost-Based Optimizer
Informix Dynamic Server.2000's cost-based optimizer will automatically determine the fastest way to retrieve data from a database table based on detailed information about the distribution of that data within the table's columns. The optimizer collects and calculates statistics about this data distribution and will pick the return path that has the least impact on system resources—in some cases this will be a parallelized return path, but in others it might be a sequential process. All that is needed to control the degree of parallelism is the memory grant manager.

To provide users with a degree of control, Informix Dynamic Server.2000 offers optimizer directives that let users bypass the optimizer. Areas which users can control include:

- Access Methods—This lets users specify how to access a table. For example, a user can direct the optimizer to use a specific index.
- Join Methods—This lets users specify how to join a table to the other tables in the query. For example, users can specify that the optimizer uses a hash join.
- Join Order—This lets users direct the optimizer to join tables in a specific order.
- Optimization Goal—This lets users specify whether a query is to be optimized by response time (which returns the first set of rows) or by total time (which returns all rows).

Raw Disk Management

For fast data access, Informix Dynamic Server.2000 achieves contiguous disk-space storage through raw disk devices, since the native UNIX file system does not guarantee contiguous disk space allocation. Raw disk space allocation, conversely, makes it possible for Informix Dynamic Server.2000 to create its own data storage system.

Data storage on raw disk allows Informix Dynamic Server.2000 to perform direct memory access (DMA). DMA writes data directly from memory to disk, bypassing intermediate operating system buffering mechanisms that are required when data storage is performed through a standard file system. This process makes it faster to commit transactions to disk.

Select First N Rows

Many decision support queries call for only the first few records of a select statement for analysis. For example, the top 20 selling products, the worst five performing regions, the first 100 customers matching the marketing profile, etc. To address these decision support queries, Informix introduces a new function called Select First N Rows. With this new feature, users can now limit the result of a query to the first N rows. This new function can deliver significant performance advantages by minimizing I/Os, and in some cases, reducing CPU cycles required since only a fraction of the rows are processed.

Union Within Views

One method for improving performance of decision-support queries is to create a view to reduce the amount of data the query has to access. Views can also be used to decrease the complexity of the query the user has to write. Since many decision support queries involve combining results from several SELECT statements, incorporating the UNION function within a view not only can reduce the complexity of the query but also speed up execution time. For example, instead of writing a query that involves multiple SELECTs, create a view that uses UNION to merge the results of the SELECT statements. By doing so, an end user only needs to issue a simple SELECT statement against the view.

Memory Resident Tables

To improve performance of frequently accessed data, users can specify one or more fragments of a table or index to remain resident in the Informix shared memory for as long as possible. Once specified, these memory resident tables will be considered last for replacement when a free buffer is requested. Ideal for small tables that are accessed frequently but are swapped out

of buffer cache due to randomly accessed tables, memory resident tables can significantly improve response time of data access.

Correlated Subquery Enhancements

Informix Dynamic Server.2000 utilizes several optimization strategies for improving performance of queries involving subqueries. These strategies include subquery flattening, query caching, and prediction promotion. Subquery flattening refers to combining query blocks into a single query block, thus minimizing new table scans, index scans, and joins. Subquery caching refers to caching the results of a subquery that is called multiple times, enabling the optimizer to avoid execution of the same query. Predicate promotion refers to substituting the constant value of a column for the occurrence of the same column in the correlated subquery, thus transforming a correlated subquery to an uncorrelated subquery.

Shared Statement Cache

Informix Dynamic Server.2000 employs a shared statement cache to cache query plans of SQL statements, so that they can subsequently be used by another user session. This feature can dramatically reduce memory consumption for systems with users issuing identical SQL statements. Additionally, since sessions executing cached statements will not have to re-parse or re-optimize the statements, the query is processed much faster.

Long Identifiers

Informix Dynamic Server.2000 lets users define identifiers that are up to 128 bytes in length. By supporting long identifiers, users can easily convert a non-Informix application to run with Informix Dynamic Server.2000. This feature eliminates the need to apply a wrapper to an application, which can slow down development process as well as the performance of the application.

64-Bit Support and Large Memory Addressability

Most UNIX and Windows NT systems are limited to two gigabytes of memory due to the limitations of the underlying 32-bit address space. Such limitations are relaxed with 64-bit architectures, providing a significant breakthrough in database performance.

To take advantage of 64-bit architectures, Informix Dynamic Server.2000 provides 64-bit support and large memory addressability (LMA) to boost performance in both on-line transaction processing and decision-support environments. 64-bit support allows larger database page sizes, which enhance performance by allowing more data to be transferred from disk to cache in a smaller number of physical I/O operations. With LMA, Informix Dynamic Server.2000 can now support tens of gigabytes of physical memory and hundreds of gigabytes of virtual address space, thus allowing more data to be cached in memory. Additionally, Informix Dynamic Server.2000 can support larger numbers of users and increase database throughput, as well as reduce swapping in heavy OLTP environments.

High Availability Features

Informix Dynamic Server.2000 offers a number of features to minimize both planned and unplanned downtime. To minimize planned downtime, Informix Dynamic Server.2000 supports a complete suite of administrative utilities that lets administrators perform virtually all database administrative tasks on line (features described in the "Administrative/Management Features" section). To minimize unplanned downtime, Informix Dynamic Server.2000 offers features to make the database server more resistant to errors. Furthermore, additional enhancements are provided to enable Informix Technical Support to diagnose

problems more quickly, allowing the server to be brought back on line as quickly as possible in the event of an unanticipated failure.

Database and Log Mirroring

Database and log mirroring provides database administrators with a means of recovering data, in the event of a media failure, without having to take the database server off line. This method is ideal for protecting critical data that require extreme reliability. Examples of data that should be mirrored include root dbspace and logical and physical log files, where if the media that stores any of these data fails, the database is immediately offline.

Fast Recovery

In the event of an unexpected shut down, Informix Dynamic Server.2000 provides a utility called "fast recovery" to quickly bring the system on line without any data loss, thereby maintaining full data integrity. Invoked during the system recovery process, fast recovery applies the transaction logs to the data files to restore the database to a state of physical and logical consistency. During this recovery process, the database is restored to the state of the last checkpoint. Next, all the committed transactions since the last checkpoint are rolled forward and all of the uncommitted transactions are rolled back.

Restartable Restore

During a physical or logical restore process, problems such as an I/O error on the tape or other errors within the server require that the entire process of restoration be restarted from the beginning. The restartable restore feature allows the restoration to be restarted close to where it left off. This is achieved by reducing the granularity of the restore to the dbspace level. As such, restartable restore picks up at the dbspace level where the original restore failed.

Table Reorganization

To enhance high database availability, Informix Dynamic Server.2000 lets administrators alter table schema "in-place." This means that operations such as adding or deleting a column to a table, inserting a column between columns, rearranging columns, or modifying the data type of a column, can be performed without making the table unavailable for normal use. Performance of the alter table operation is also increased since it takes only as long as updating the system catalogs with the new table definition. Furthermore, the database does not have to create a second copy of the table in order for the changes to take place, thereby significantly minimizing disk-space usage, especially when large tables are being altered.

Microsoft Cluster Server Support

Extending high availability support into the Windows NT environment, Informix Dynamic Server.2000 is integrating with Microsoft Cluster Server functionality. Microsoft Cluster Server supports fail-over capability for symmetrical multiprocessing (SMP) systems configured in a cluster, enabling applications on a server to automatically fail over to another server in the event of an unanticipated hardware failure.

Utilizing Microsoft Cluster Server's capability for monitoring, failure detection, and failure communication, Informix Dynamic Server.2000 will immediately perform database recovery procedures upon a node failure. Once the database recovery procedures are completed, Microsoft Cluster Server performs system-level recovery and restarts the cluster-aware application on the surviving node. This feature ensures the highest degree of availability by quickly recovering from unexpected hardware failure, allowing database processing to be resumed in the quickest amount of time.

Exception Handling Improvements

To improve server reliability and availability, Informix Dynamic Server.2000 offers a set of routines to better handle assertion failures and warnings within the server. These routines minimize server downtime by effectively pinpointing and diagnosing the problem areas, and returning appropriate error messages indicating what has transpired. Should a server failure be unavoidable, these exception handling routines provide better diagnostic information to assist in finding and fixing the problems.

Smarter Diagnostics

To decrease the time it takes to diagnose and correct problems, Informix Dynamic Server.2000 offers a set of enhancements to assist Informix development and support organizations in the debugging and analysis of system problems. Smarter diagnostics consist of enhancements in six areas: event alarms, fault isolation, shared memory dumps, stack tracing, additional utility options, and thread blocking routines. Customers benefit from this feature through faster turnaround time for the resolution of reported problems.

Administration/Management Features

Informix Dynamic Server.2000 provides a set of dynamic system administration tools that monitor and fine-tune system parameters such as CPU and memory utilization, asynchronous I/O queuing, decision-support and batch queuing, available disk space, and efficient partitioning schemes. These mainframe-caliber administration tools enable database administrators (DBAs) to perform most system administration functions on line—without bringing the system down. Additionally, many utilities have been parallelized to deliver the highest performance possible. To enhance management of distributed databases, a user-friendly, intuitive administration environment called IECC is provided, which allows DBAs to manage multiple, remote databases from a single, centralized console.

Parallel Load

Informix's parallel load utility can load and unload data very quickly due to its ability to read in data from multiple sources and load the data into the database in parallel. The graphical user interface allows the DBA to:

- configure loading for various types of flat files (such as ASCII, COBOL, or EBCDIC) or applications such as spreadsheet or word processing applications, and perform necessary conversions (e.g., EBCDIC to ASCII);
- perform mapping between the load file and the Informix schema;
- perform selective loading; and,
- view the load file.

Two different modes allow the DBA to decide whether normal load tasks, such as checking referential integrity, logging, and index builds, should be performed during the data load, or after the load is complete (speeding up the load time).

DBAs can utilize user-defined routines (UDRs) to perform special functions within the parallel load utility. For example, a UDR can be used to transform input data x to output data xy+1+xz. The Informix parallel load utility also supports parallel execution of UDR, which can significantly speed up the processing of complex UDRs.

Backup and Restore Utility

Informix Dynamic Server.2000 provides a very sophisticated and well-integrated backup and restore utility that allows every facet of backup and recovery to be configured on the fly—including the ability to perform on-line, table, and partition-level backups in parallel. Additionally, the backup and restore utility are integrated with the Informix enterprise system management environment for assisting administrators with backup and restore of distributed databases.

The backup and restore procedure is separated into two processes. The first process involves extracting and restoring the data to and from Informix Dynamic Server.2000 database using the backup and restore utility. The second process involves writing and extracting data from the storage media using a storage manager. During a backup, the backup and restore utility retrieves the data from the server and passes it to the storage manager. The storage manager then writes the data to the storage media. During restore, the process is reversed. The storage manager retrieves the data from the storage media and passes it through the backup and restore utility, which restores it to the database server.

The Informix backup and restore utility comes equipped with a storage management application programming interface (API) that conforms to the X/Open Backup Services API (X/BSA) standards. With this API, Informix allows users to utilize the many third-party storage management vendor (SMV) subsystems available today, such as IBM ADSM, HP Omniback II, and Legato. Using these SMV subsystems, users can take advantage of advanced backup and restore operations such as unattended backups, scheduled backups, tape and volume management, encryption and decryption support, compression and decompression capabilities, as well as enabling the use of autochanger devices.

For customers who do not need the sophistication of third-party storage managers, the backup and restore utility includes a native storage manager called Informix Storage Manager (ISM). ISM provides a simple, easy-to-use environment to assist administrators with performing backup or restore operations to native tape or disk drives provided by the hardware vendor.

External Backup/Restore

Informix Dynamic Server.2000 also lets DBAs create external backups of their data to further assure data availability in the event of an unlikely disaster. Unlike the internal backup provided by the backup and restore utility (described previously), an external backup creates simultaneous multiple copies of data to host-independent, local, and remote sites using proprietary hardware and software technologies. This could be as simple as disconnecting a mirrored disk and saving it as backup media. By providing this feature, Informix Dynamic Server.2000 allows for faster restoring of data, thereby increasing server availability and minimizing downtime.

Point-In-Time Recovery

Point-in-time recovery lets administrators control the degree to which logical logs are reprocessed, thereby synchronizing the database to a particular point in time. With this feature, administrators simply provide a date or time in which they wish the server to restore. This date or time is then passed to the server during the restore process. Once the server has performed the physical restore, the server then reprocesses the logical log records until it finds the first transaction-bound record that is greater than the specified date or time. At that point, the server would stop reprocessing the logical log records and all transactions that were uncommitted before the user-specified time would be rolled back.

Informix Enterprise Command Center

The IECC provides administrators the ability to manage the entire database environment (UNIX and NT) from a single console. Its advanced systems management solution consists of a easy-to-use graphical user interface (GUI); a suite of automated, open, and

scalable systems management tools; and integrated support for third-party systems management tools and SNMP-based (simple network management protocol) enterprise-management frameworks.

IECC allows customers to manage anywhere from one to thousands of database servers easily from a central console. It offers a complete set of database administration functionality for managing every aspect of the Informix database environment, including server management, job management, event monitoring and resolution, and storage management. And, with its Java-enabled, object-oriented, CORBA/IIOP-based architecture, IECC lets administrators create "lights-out" systems management automation and customize application management objects. IECC also offers users the flexibility to manage distributed databases from a Web browser or Microsoft Windows PC, providing universal systems management from any desktop—regardless of platform or location.

IECC is a completely open platform designed to allow third-party systems management applications and network management tools to easily interoperate, providing complete systems management in heterogeneous environments.

Simple Network Management Protocol (SNMP)

In providing integration with SNMP-based technologies such as HP OpenView and IBM SystemView, Informix Dynamic Server.2000 offers a suite of subagents to provide support for both public and private management information bases (MIBs). Conforming to the Internet Engineering Task Force (IETF) relational database public MIB standards, Informix public MIBs provide generic database information such as database vendor, version number, number of completed transactions, disk utilization, etc. Informix private MIBs provide a host of database information specific to Informix Dynamic Server.2000 such as database buffers, locks, and logs, the number of transactions committed and rolled back, the time of the last checkpoint, memory utilization, the date, time, and level of the last backup, etc. These data enable a network manager to effectively monitor all of the Informix databases that are on a network.

The System Monitoring Interface

Informix Dynamic Server.2000 maintains a system master database—the "sysmaster"—which keeps track of information about the database server. A component of the sysmaster is the System Monitoring Interface (SMI), consisting of a number of tables and pseudo-tables that provide information on the state of the database server. These tables can be queried to identify processing bottlenecks, determine resource usage, track session or database server activity, etc. Some of the information provided includes:

- database names, owners, and logging status;
- status of users who are waiting for database resources;
- performance profiling information;
- user and system CPU usage by virtual processor;
- information about disk space;
- information about transaction log (logical log) status;
- information about specific regions of disk space (i.e., dbspace);
- lock usage information; and,
- extent information—continuous segments of disk space allocated to a tablespace.

OnPerf Utility

The OnPerf utility is a graphical tool that lets DBAs monitor and display performance metrics in real time. Various levels of metrics can be monitored, including classes at the database level, operating system level, CPUs, virtual processors, individual user sessions, and tablespaces. Performance data can be saved to a file and displayed at a later time for simulated real-time analysis or for trend analysis.

Enterprise Replication

Enterprise replication is designed to meet a wide spectrum of business and application requirements. It supports a full peer-to-peer replication model with update-anywhere capability. Enterprise replication replicates data asynchronously among multiple database servers. Updates at one data engine, including configuration changes to enterprise replication, are automatically propagated to other data engines, ensuring all engines have consistent data.

Enterprise replication employs a log-based mechanism for capturing updates. This method minimizes impact on transaction processing because it operates as part of the normal database logging process. Only those records that have been earmarked for replication are collected and transmitted to other data engines, thus incurring minimal interference to the original transactions. Log-based capture is much more efficient than trigger-based capture, which can directly impact the performance of the original user transaction.

Transmission of replicated data can be immediate or postponed through managing enterprise replication connections. A reliable message delivery mechanism stores data locally and propagates the data to the remote server as a separate transaction when a connection is available. In the event of a server or network failure, the surviving server can continue to service the users and queue data for the failed or inaccessible server, providing

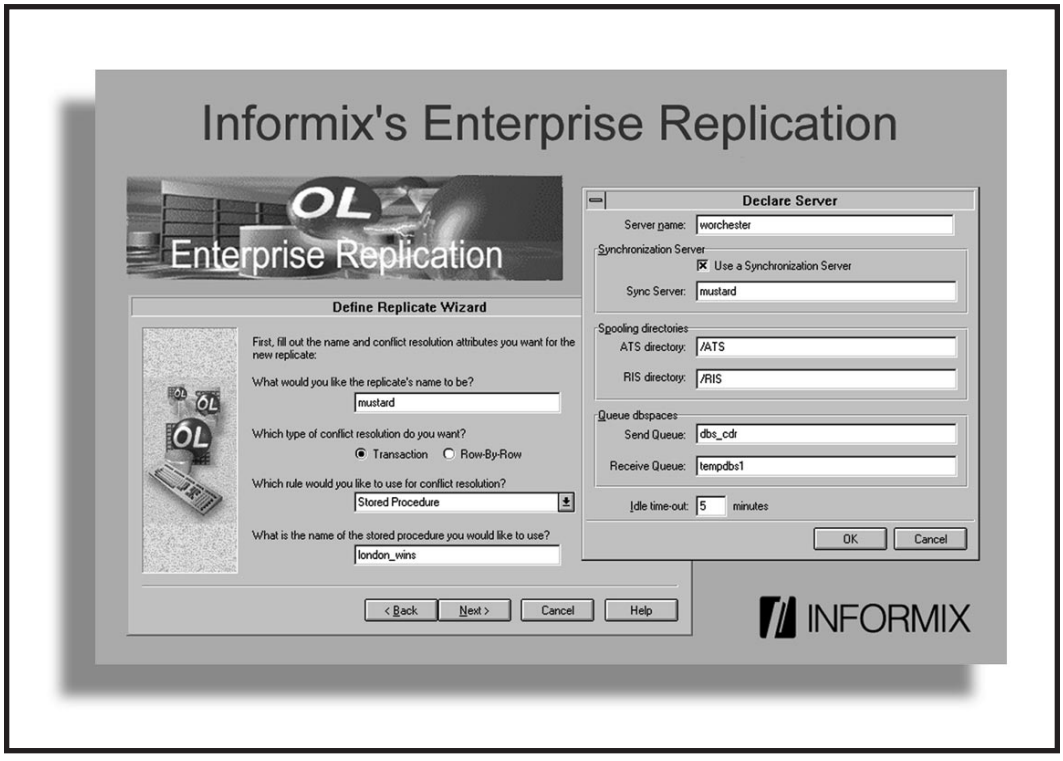

Figure 13: Screen image, Informix Dynamic Server.2000 (202Kb, GIF).

a high degree of data availability. Once the failed server or network is operational again, all changes to the source database are propagated to the database on the affected server.

Through its hierarchical routing feature, enterprise replication can support large replication networks consisting of hundreds or thousands of servers. Unlike replication methods employed by other vendors, enterprise replication's hierarchical routing feature eliminates the need of having to physically connect all the servers in the network, which can incur large overhead as the number of servers in the network grows. This capability is especially important when using replication to consolidate and/or disseminate data between a small set of headquarters based servers and a large set of remotely located servers such as might be found in retail stores or branch offices.

Hierarchical routing lets DBAs specify routing topology through defining servers as one of three types: root, non-root, and leaf, each occupying a different place in the network topology. Once all servers in the network are defined, information can be delivered independent of whether the source server is directly connected to the destination because enterprise replication servers have the ability to route and forward data while maintaining transaction ordering and guaranteeing delivery. This method can drastically reduce the administrative effort required to support a large number of servers.

Data Consistency

While high availability ensures integrity at the system level, data consistency ensures consistency at the transaction level. Informix Dynamic Server.2000 maintains data consistency via transaction logging and internal consistency checking, and by establishing and enforcing locking procedures, isolation levels, and business rules.

Transaction Logging

When an operation is unable to be completed, the partially completed transaction must be removed from the database to maintain data consistency. To remove any partially completed transaction, Informix Dynamic Server.2000 maintains a historical record of all transactions in the logical logs and automatically uses these transaction records as a reference to restore the database to the state prior to the transaction

Checkpoint

Checkpoint refers to the point in the database server operation when the pages on disk are synchronized with the pages in the shared memory buffer pool. To ensure data consistency, Informix Dynamic Server.2000 automatically performs checkpoints. Administrators are provided with the ability to initiate manual checkpoints or specify the checkpoint interval in order to control the frequency of the checkpoints.

The database server performs two types of checkpoints: full checkpoints and fuzzy checkpoints.

Full Checkpoint

In a full checkpoint, the database server flushes all modified pages in the shared-memory buffer pool to disk. When a full checkpoint completes, all physical operations are complete and the database server is said to be physically consistent.

Fuzzy Checkpoint

To speed up checkpoint and improve transaction throughput, administrator can utilize fuzzy checkpoints. In fuzzy checkpoint, the database server does not flush the modified pages of certain types of operations. Known as fuzzy operations, they include inserts, updates, and deletes. By not flushing these modified pages, the time for checkpoints to complete is drastically reduced, thereby improving database throughput.

When a fuzzy checkpoint completes, the checkpointed pages might not be consistent with each other because the database server does not flush all data pages to disk. However, when necessary, the database server will perform a full checkpoint to ensure the physical consistency of all data on disk.

Internal Consistency Checking

Internal consistency checking is designed to alert the Informix Dynamic Server.2000 administrator to data and system inconsistencies. Informix Dynamic Server.2000 contains a data-level layer of checks that can detect data inconsistencies that might be caused by hardware or operating system errors. If inconsistencies are detected, this internal mechanism automatically writes messages to the message log.

In order to better pinpoint the cause of the inconsistency, Informix Dynamic Server.2000 administrators can instruct users to set consistency-checking environment variables. These variables generate diagnostic output—for example, contents of shared memory when the inconsistency occurred—to help locate the cause of the inconsistency.

Locking and Process Isolation

Other important features for maintaining data consistency are locking procedures and process isolation. These security measures prevent other users from changing data that is currently being read or modified.

Locks

The database server prevents errors by imposing a system of locks. A lock is a claim or reservation that a program can place on a piece of data. The database server guarantees that, as long as the data is locked, no other database server process can modify it. When another program requests the data being modified, the database server either makes the program wait or turns it back with an error.

Informix Dynamic Server.2000 also prevents deadlocks—a situation where two users have each locked data that the other user needs. For example, user A has locked one row and won't release it until user A can access the row locked by user B. Informix Dynamic Server.2000 detects a deadlock immediately and returns an error message to the second program to prevent a standstill.

The throughput of transactions or a specific table can be affected by the locking strategy used for the table. Applications that use strategies of exclusive access to data might find that other database server processes are spending time waiting for access to the data. For this reason Informix Dynamic Server.2000 provides several locking levels. The database server can place a lock on a single row, page, table, or database.

Row- and page-level locking are specified when the table is created or altered. Table- and database-level locking is specified in the user's application.

Isolation Levels

The isolation level is the degree to which your read operation is isolated from concurrent actions of other database server processes: what modifications other processes can make to the records you are reading, and what records you can read while other processes are reading or modifying them.

Isolation levels are only in effect for reads, they are not used for statements that insert, update, or delete. Informix Dynamic Server.2000 has four isolation levels: dirty read, committed read, cursor stability, and repeatable read.

The simplest isolation level, dirty read, provides no isolation at all. When a program accesses a row using a dirty read, it places no locks and respects none. For dirty reads, users are able to read all the data—committed or uncommitted.

The committed read isolation level guarantees that database server will read only rows that have been committed to the database. Before retrieving a row, the database server tests to see if an updating process has placed a lock on the row. If no locks have been placed, the database server accesses the row. The committed read isolation level ensures that the database server does not read uncommitted data because locks are placed on the rows that have been updated but not committed.

The next level of isolation is cursor stability. When cursor stability is in effect, the database server places a lock on the latest row read. Only one row is locked at a time; each time a new row is read, the previous lock is released. This isolation level ensures that the current record will not change while the program examines it.

The repeatable read isolation guarantees that the results obtained during a read operation would be identical if the read were repeated later in the same transaction. Not only does a repeatable read place shared locks on the records that were read, but it also prohibits other users from adding records to the database or modifying records if they would have satisfied the criteria of the read request had they been in the database earlier.

Retain Update Locks

Repeatable read isolation is the strictest and the most expensive isolation level since it locks all rows examined for the duration of the transaction. To avoid the overhead of repeatable read isolation level, users can execute a SET ISOLATION statement with the RETAIN UPDATE LOCKS clause. The RETAIN UPDATE LOCKS feature provides a switch to retain update locks for isolation levels dirty read, committed read, and cursor stability such that the locks will not be released until the end of transaction. Thus, RETAIN UPDATE LOCKS effectively achieves the same goal as a repeatable read isolation level, but without the performance overhead.

Business Rules

Business rules enforce data consistency at the column level. They specify possible data values, column defaults, and column-to-column relationships.

Business rules must be enforced by all applications that access or manipulate data. However, Informix Dynamic Server.2000 does not depend on the application to implement these business rules. Instead, it enforces these rules independently. This centralization of responsibilities removes the burden from the user applications and guarantees adherence to business rules.

Informix Dynamic Server.2000 supports integrity constraints, stored procedures, and triggers to enforce business rules.

Integrity Constraints

Informix's implementation of ANSI SQL-compliant integrity constraints ensures that information is not improperly deleted and that inserted data meets column specifications. Informix Dynamic Server.2000 provides two types of integrity constraints: referential integrity and entity integrity.

Referential integrity allows users to define and enforce relationships between columns. For example, Informix Dynamic Server.2000's referential integrity guarantees that information about an entry in a master table is not deleted as long as the corresponding information still exists in the detail table. This prevents users from deleting a customer record if an order still exists for that customer in an order table.

Entity integrity enforces acceptable data values for particular columns. This check allows the DBA to specify a range of permissible values. Default values allow users to specify a default of any value that is compatible with the column datatype.

Stored Procedures

Stored procedures are SQL commands and program statements that are stored in the database as named procedures to ensure consistent implementation of commonly used operations. Stored procedures maintain common optimized application routines in the database rather than in the application program.

Stored procedures reduce the amount of network traffic for database operations because stored procedures can handle multiple tasks—such as insert, update, and delete—with a single command. After processing the request, the stored procedure returns only the result of the request rather than the numerous result sets for the individual queries.

Triggers

A trigger is an alternate method for invoking a stored procedure. Rather than the application program calling a stored procedure to enforce a business rule, a trigger can be defined that will cause the database server to automatically execute a stored procedure (or SQL statement) when any attempt is made to insert, delete, or update a field in the table.

Since triggers are stored in the system catalogs, the need for application programs to maintain redundant code has been removed. Likewise, consistent integrity constraint enforcement across all transactions is guaranteed since triggers cannot be bypassed.

A SELECT trigger is defined on a table or a column and it fires when the table or column appears in certain types of SELECT statements. SELECT triggers can be used by application developers to enforce application-specific auditing or accounting rules. For example, a user can define a SELECT trigger which inserts an audit record to an audit table whenever the EMPLOYEE_SALARY table is selected.

Cascading Deletes

Cascading deletes automatically delete all child records in a table when a parent record is deleted, thereby simplifying user applications. If for any reason the original delete statement or any of the actions triggered as a result of the delete fail, the entire delete statement is rolled back.

The ability to perform cascading deletes not only provides a significant performance advantage, since Informix Dynamic Server.2000 automatically deletes rows, but it also makes the system easier to use and administer by eliminating the need for programmers to hard code deletes into the application. You also don't have to spend the time and effort to code business/referential rules, which results in better database consistency as well as cleaner application programs.

Database Security

Informix Dynamic Server.2000 provides two levels of access privileges to ensure database security. Database privileges control access to the database and the privileges for creating tables and indexes in the database. Table privileges specify the operations that a user is allowed to perform against a specified table.

Informix Dynamic Server.2000 supports ALTER, INSERT and DELETE security at the table level while enforcing SELECT and UPDATE security at the column level. Separate privilege statements are used to grant and revoke the appropriate access level to users. No separate database login is required since Informix Dynamic Server.2000 applies security at the user's login level.

Stored procedures provide an additional security mechanism by establishing their own permissions separately from the data permissions. The owner of a stored procedure grants users the right to execute the stored procedure—allowing the user to perform all the SQL operations in the procedure, but restricting other access against the database. By using stored procedures to prohibit users from performing operations against the database, except through authorized stored procedures, DBAs can elevate security to the procedure level.

Secure Auditing

Informix Dynamic Server.2000 offers secure auditing features to provide traceability and accountability for any database object manipulated by a user. These features have been implemented to model after the Class C2 security requirements set forth by the U.S. National Computer Security Center.

With Informix Dynamic Server.2000, you can selectively monitor the activity of users on the system. The interface for secure auditing is driven from the command line or a configuration parameter and allows you to define which activities you wish to monitor for certain users.

Communications Support Service (CSS)

Communications Support Service (CSS) is an Informix internal component which provides greater network security through supporting third-party security services such as authentication, message integrity, message privacy, data compression, etc. CSS accesses plug-in modules called Communication Support Modules (CSM) to provide support for third-party services such as DCE 1.1 security service via the industry standard API called the Generic Security Service API (GSS API). CSM can also be developed by customers to define their own proprietary security policy.

ANSI SQL92 Entry Level Conformance

Informix Dynamic Server.2000 is the first commercially available RDBMS for the UNIX operating system to receive Entry Level SQL 1992 certification from the National Institute of Standards and Technology (NIST). Formerly the National Bureau of Standards, NIST has the only test suites available for validating SQL standards compliance.

The latest NIST certification states that Informix Dynamic Server.2000 complies with the government's SQL RDBMS Federal Information Processing Standard (FIPS), number 127-2. FIPS 127-2 meets the specifications set by the American National Standards Institute's (ANSI) 1992 Entry Level standard for SQL databases.

Distributed Database and Application Support

With Informix Dynamic Server.2000, distributed, client/server connectivity is automatically built in—ensuring that any Informix application using its associated connectivity libraries can run in client/server mode. Therefore, there is no need to purchase any additional networking products to run Informix Dynamic Server.2000 across heterogeneous client/server networks.

Informix Dynamic Server.2000 can read, join, and update tables on several different computer systems within a single transaction. Its industry-leading optimization methods ensure that these multisite updates are handled quickly, while its two-phase commit recovery procedures ensure that the updated tables remain consistent even after a system failure.

Multisite Read, Joins, and Updates

Informix Dynamic Server.2000's ability to read, join, and update tables on several different computer systems within a single transaction is critical for companies with a need to share data between multiple databases. Informix Dynamic Server.2000 also supports multisite updates within a single transaction because companies often need to modify several distributed databases and ensures consistency between them.

If a system fails during a multisite update, distributed database products should provide special recovery procedures to bring the failed system up to date so that its information is consistent with that of the other database servers. Distributed database products should also implement multisite transactions quickly without wasting CPU or network resources, since most transactions are used for time-critical business applications where wasted time means lost profits.

Informix Dynamic Server.2000 enables users to issue multisite updates within a single transaction. It coordinates changes to distributed databases using a two-phase commit protocol, handles system failures through its automatic recovery procedures, and ensures fast performance by employing presumed abort optimization.

Two-Phase Commit Protocol

Two-phase commit protocols ensure database consistency during multisite updates, and incorporate precommit and commit phases. When the user first connects to an Informix Dynamic Server.2000 system, that system becomes the coordinator for all the other participant database servers involved in the multisite update.

The coordinator receives a COMMIT WORK statement from the user—indicating that there are no more SQL statements in the transaction—and sends messages to the participant database servers asking them to prepare to commit the transaction, beginning the precommit phase of the protocol.

If any of the participants reply that they cannot prepare to commit, or simply don't reply, the coordinator aborts the transaction and requests that the other database servers roll back their portions of the transaction. If all the participant database servers send positive responses, the coordinator asks them to commit the transaction. Deciding to either roll back or commit the transaction starts the commit phase of the two-phase commit protocol.

Multithreaded Clients

Through INFORMIX-ESQL, developers can continue to develop applications using familiar third-generation languages such as C and COBOL for database access. The ESQL libraries are re-entrant, or "thread-safe," enabling users to develop multithreaded client applications that can have multiple active connections to a server concurrently, thereby taking full advantage of Informix Dynamic Server.2000's multithreaded architecture.

Additionally, the ESQL libraries are dynamically linked and shared to significantly reduce the size of the corresponding executables. These shared libraries can be linked to a number of executing programs during run time, resulting in greater efficiency and less memory usage. ESQL includes TP/XA functionality, linking Informix Dynamic Server.2000 with transaction managers to support global transactions—those transactions involving multiple databases and perhaps multiple database management systems from different vendors. Informix is the first RDBMS to provide this standards-based interface to X/Open XA-compliant transaction managers.

Application developers can also create multithreaded applications using INFORMIX-ODBC. Conforming to Microsoft's Open Database Connectivity (ODBC) specification, INFORMIX-ODBC supports all ODBC API calls, datatypes, and SQL syntax as defined in the specification. For larger, more complex applications, INFORMIX-ODBC also supports Microsoft Transaction Server (MTS).

Similarly, developers can use Informix's JDBC driver to create multithreaded Java applications. Informix's JDBC driver is a standards-based (JDBC V1 and V2), object-oriented Java interface that provides full access to the database server for application developers and DataBlade module developers. It achieves optimal performance and robustness because it is a pure Java implementation and uses a native interface to connect to the database server.

Heterogeneous Integration

In today's competitive business climate, immediate access to information throughout an organization is often critical in order to make informed business decisions. Where companies once stored their data on mainframes, today the data may be spread across a variety of different computing platforms and operating systems, including proprietary and open, relational, or non-relational databases. To provide Informix Dynamic Server.2000 users with access to corporate data residing on disparate database servers, Informix offers the Informix Enterprise Gateway™ product family for transparent access to data stored within Oracle, Sybase, IMS, or IBM's DB2 databases. With these gateway technologies, Informix Dynamic Server.2000 users can easily access data regardless of where that data is located.

Open Database Connectivity

For developers and end users wanting to deploy applications that require access across heterogeneous databases, Informix ODBC is the Informix implementation of the Open Database Connectivity (ODBC) 2.5 standard. With Informix ODBC, any ODBC-compliant application can connect directly to Informix Dynamic Server.2000. It supports SQL statements with a library of C-language calls which enables developers to dynamically access Informix Dynamic Server.2000 without the need for an SQL preprocessor or the recompilation of source code for each independent data source. With hundreds of ODBC-based applications for accounting, inventory management, customer tracking, and more already available, customers have immediate access to a variety of solutions for running in a client/server environment.

Global Language Support
Informix's global language support (GLS) implementation conforms to the GLS level 4 specification, a coding standard that allows multibyte characters. By providing GLS support, Informix Dynamic Server.2000 can collate character strings, print dates, and accept monetary input in the rules and formats required by the country in which the products are being used, without the need to distinguish between localized versions of Informix software. Additionally, GLS provides worldwide support of database applications, so applications can be migrated to multiple languages while maintaining the same functionality.

Internet Application Development and Deployment

To accommodate the growing popularity of Internet, Informix Internet Foundation.2000 offers a wide range of Internet-specific features for supporting key component development standards such as Java, making it an ideal database platform for hosting scalable Internet applications. Specifically, Informix Internet Foundation.2000 offers the following:

1. Informix J/Foundation;
2. Specialized Java virtual processor; and
3. Web DataBlade module.

Informix J/Foundation
Informix J/Foundation is the Java cornerstone of the Informix Internet Foundation.2000. J/Foundation is a standard, embedded Java environment that runs directly in the data management server itself, offering performance, scalability, and manageability benefits for Internet applications.

Informix uses standard Java Virtual Machines from our platform vendor partners, which are embedded directly into the server process, giving our customers a completely open and standard environment. J/Foundation enables user-defined routines (UDRs) to be written in Java, as well as complete DataBlade modules, each giving a new level of performance by moving the data-intensive business logic close to the data itself. Development and migration of Java code into the database is easy since Informix use the same JDBC driver in the server.

Java is the ideal programming language for the Internet, and J/Foundation offers a standard platform for the development of database-driven Internet solutions in Java.

Java Virtual Processor

To improve execution efficiency and performance of Java UDRs and Java applications within J/Foundation, Informix has designed a specialized class of virtual processor, called the Java virtual processors (JVPs). Responsible for executing all Java UDRs and general-purpose Java applications, JVPs have the same capabilities as a CPU VPs in that they can process complete SQL queries. Each JVP embeds a Java virtual machine (Java VM) in its code. This embedded VM architecture avoids the cost of shipping Java-related queries back and forth between CPU VPs and Java VPs. Multiple JVPs can exist in the same database server to enable parallel execution of Java UDR.

Web DataBlade Module

The Informix Web DataBlade Module provides the easiest way to Web-enable database content without programming. Using its comprehensive feature set, users can construct anything from a simple query front end to an interactive Web site that retrieves and updates any data stored in Informix Dynamic Server.2000 databases. Through the Web DataBlade module tags, application developers no longer need to write low-level gateway interface code, allowing them to focus on application flow and design. This saves a significant amount of time for application development and maintenance. Furthermore, Informix Web DataBlade module works with all standard Web browsers and servers, offering an open development environment for creating powerful, interactive Web applications.

ABOUT INFORMIX

Based in Menlo Park, California, Informix Corporation specializes in advanced information management technologies that help enterprises in the i.Economy get to market quickly, generate new revenue, build a unique strategic advantage, and solve their most complex business problems. Informix offers customers a complete software infrastructure for the Web that delivers highly scalable transaction processing, personalized content management, integrated business intelligence, full multimedia capabilities and complete e-commerce solutions. For more information, contact the sales office nearest you or visit our Web site at www.informix.com.

Informix®

4100 Bohannon Drive
Menlo Park, CA 94025
Tel. 650.926.6300
www.informix.com

INFORMIX REGIONAL SALES OFFICES

Asia Pacific	65 298 1716	Japan	81 3 5562 4500
Canada (Toronto)	416 730 9009	Latin America	305 591 9592
Europe/Middle East/Africa	44 208 818 1000	North America	800 331 1763
Federal	703 847 2900		650 926 6300

Printed in U.S.A. 1/00
000-21932-77 revision 1