



Enhancements in IBM Informix Dynamic Server, Version 9.30

*by Carlton Doe
Technical Sales Manager
IBM Data Management*

Table of Contents

Overview	1
1 Differences between IDS 9.30 and IDS with J/Foundation 9.30	3
1.1 <i>IDS 9.30</i>	3
1.2 <i>IDS with J/Foundation 9.30</i>	4
2 New features in IDS Version 9.30	4
2.1 Performance features	5
2.1.1 <i>SQL statement cache</i>	5
2.1.2 <i>Fuzzy checkpoints</i>	5
2.1.3 <i>Memory resident tables</i>	7
2.1.4 <i>Optimizer directives</i>	8
2.1.5 <i>Optimizer extension: subquery flattening</i>	9
2.1.6 <i>Optimizer extension: key-first index scan</i>	10
2.1.7 <i>Enterprise Replication enhancements</i>	10
2.2 SQL features	11
2.2.1 <i>Long identifiers</i>	11
2.2.2 <i>“On Select” trigger</i>	12
2.2.3 <i>SQL extension: “rename index” statement</i>	12
2.2.4 <i>SQL extension: ANSI outer join syntax</i>	12
2.2.5 <i>Retain update locks</i>	12
2.2.6 <i>SQL functions</i>	13
2.2.7 <i>Raw tables</i>	13
2.2.8 <i>In-place alter table</i>	14
2.2.9 <i>Attach/detach fragment extensions</i>	14
2.2.10 <i>Optional “from” in a delete statement</i>	14
2.2.11 <i>Explain without execute</i>	15
2.2.12 <i>Configurable default lock mode</i>	15
2.2.13 <i>Revoke as user</i>	16
2.3 DBA features—utilities	16
2.3.1 <i>Dynamic lock allocation</i>	16
2.3.2 <i>Dynamic logical logs</i>	17

Table of Contents

2.3.3	<i>Archecker</i>	17
2.3.4	<i>Onsmsync</i>	18
2.3.5	<i>Command-line interface for the High-performance Loader (HPL)</i>	18
2.3.6	<i>Oncheck without locks</i>	19
2.3.7	<i>Informix Storage Manager (ISM) interface change</i>	19
2.3.8	<i>External backup and restore (EBR)</i>	19
2.3.9	<i>Restartable restore also a feature from 7.3</i>	20
2.3.10	<i>New ex_alarm.sh script for alarms</i>	21
2.3.11	<i>UNIX Bundle Installer</i>	21
3	MaxConnect	21
3.1	<i>Description of MaxConnect</i>	21
3.2	<i>Configuration options for MaxConnect</i>	22
4	IDS benchmark results	24
4.1	<i>IDS without MaxConnect</i>	24
4.2	<i>IDS with MaxConnect</i>	25
5	IDS overview	25
5.1	<i>Features and versions</i>	25
5.1.1	<i>Performance features</i>	25
5.1.2	<i>SQL features</i>	26
5.1.3	<i>DBA features—utilities</i>	27
5.1.4	<i>Additional features</i>	27
5.2	<i>Compatibility between IDS and other Informix products</i>	29
5.2.1	<i>Certified IBM Informix DataBlades Modules for IDS</i>	29
5.2.2	<i>Certified products for IDS</i>	30



Overview

This document contains a detailed summary of the features and benefits of IBM Informix® Dynamic Server (IDS) 9.30, specifically as compared to IDS 7. Before delving into the important technical details, let's look at some overall benefits of IDS 9.30 and whether it is worth your while, as a satisfied IDS 7 customer, to work with IBM to start your plans to move to this exciting new technology.

First of all, IBM is not forcing you to move to another version and has pledged continued support for IDS 7 into the future. Having said this, new enhancements and development of the IDS product line will be primarily directed to the 9.x version. While most of this work is focused on new features, IDS 9.30 contains a number of internal optimizations in the area of code path reduction and function re-design that have resulted in marked improvement in the use and efficiency of system CPU and memory resources. IBM would like to encourage you to review your database environment and see if there is enough value in these newer features to warrant migrating to this version of the engine.

IDS 9.30 continues the Informix database tradition of needing fewer resources to manage than our competitors' offerings. IDS 9.30 beta customer James Horn of S1 testified, "Informix is easy to maintain and support and has been very cost-effective for us, and that's one of the secrets of our success...the total cost of ownership is key since our profitability depends on being able to keep the cost of the underlying system low." Our customers typically report needing significantly less staff to manage IBM IDS products than competitive products.

IDS 9.30 offers ease of use and administration, high performance, enterprise replication, interoperability, e-business integration and the ability to support rich multimedia. "The performance-related features of IDS 9.30 are a DBA's dream...the IDS product (is) perhaps the fastest and most scalable engine available today," notes John Lusty of Berkeley Information Systems.

IDS 9.30 provides tremendous value. Functionality that our competitors often charge extra for is bundled with or included in the base product. This includes features such as partitioning, multimedia support modules, administration modules and administration/application development tools.



Future versions of IDS 9 will focus on continuing performance and feature enhancements as well as ongoing hardware and software platform upgrades, including 64-bit support. Ease of use is a continual goal of IBM software, and IDS will continue to be enhanced with autonomic computing and IBM SMART initiative additions, which will make our products easier to use and manage.

Another focus area will be the tighter integration with other IBM software components such as WebSphere[®], Tivoli[®] and Lotus[®] software and, of course, IBM DB2[®] Universal Database[™]. This allows our customers to source a powerful e-business software stack from one industry-leading vendor. IDS 9 is already integrated with IBM WebSphere Application Server, IBM WebSphere MQ, IBM Tivoli Storage Manager, IBM DB2 Relational Connect, IBM DB2 Table Editor 4.3 and IBM DB2 Web Query Tool 1.

So, what's involved in moving from IDS 7 to 9? IDS has always focused on ease of upgrades and the tradition continues with Version 9. In addition to upgrade information, documentation, hints and tips, you can also find an array of courses related to Informix at <http://www-3.ibm.com/software/data/informix/education/> to keep your skills current and to learn how to optimize IDS 9.

What about cost? IDS 7 and 9 are currently priced the same so new licenses are identical. For existing customers, IBM has recently announced that IDS 7 and 9 customers are eligible to participate in the Trade Up to Data Management Gold Transaction Processing Offering. Current customers now have the flexibility of purchasing either IDS 7 or 9, or DB2, as their needs dictate, as long as they have a current maintenance contract. To understand what your specific upgrade costs would be, contact your sales representative. More information on the Gold Offerings is available at <http://www-3.ibm.com/software/data/programs/goldoffer.html>.

So, in case we haven't said it before, welcome to IBM. We are delighted to have you as a customer and we hope to earn your continued business. Please take a moment to consider Informix 9.30 technology. We think you will be glad you did.



IDS 9.30

The purpose of this white paper is to provide you with a brief overview of some of the newest enhancements to the IBM IDS database engine. The functions highlighted in this document are either new or significantly enhanced in the 9.30 version of the engine.

For the most part, the technologies highlighted here will be of interest to those running in an online transaction processing (OLTP) environment as opposed to data warehousing. There are features, though, that will be of value to those running data mart and smaller data warehouse environments with this version of the engine.

1 Differences between IDS 9.30 and IDS with J/Foundation 9.30

1.1 IDS 9.30

IDS Version 9.21 was developed through the merger of IDS Version 7.3 code with the former Illustra product, represented in the Informix product line as Informix Universal Server (IUS) Version 9.14. While generally successful, there were issues in IDS Version 9.21 that needed improvement and correction. IDS Version 9.30 is a significant upgrade from IDS Version 9.21 and incorporates almost all of the needed fixes and corrections.

IDS Version 9.30 should be considered the basic building block for high-end OLTP applications. With its multi-threaded core engine and object-relational technology, database and application designers can take advantage of features not available from our competitor's products. Some of these features include the ability to define new data types that can more closely model real-world business data as well as the creation and storage of business logic and access methods directly in the server. Storing logic and methods within the engine provides a consistent, single point-of-view on how information should be manipulated. The robustness, scalability and extensibility of IDS should make it the first choice for enterprisewide data management.



1.2 IDS with J/Foundation 9.30

Concurrent with the release of IDS Version 9.2 core engine, Informix also released several bundles of products that provide additional data management functionality. Called “Foundations,” these bundles usually include one or more DataBlades[®] as well as a Java[™] virtual machine as a component of the core engine. Purchasing one of these bundles represents a cost savings over buying those components (where available) on an individual basis.

IDS with J/Foundation Version 9.30 is a new engine bundle bringing together the IDS Version 9.30 engine with the most requested feature from the larger Foundation bundles—the Java virtual machine. At this time, IDS with J/Foundation Version 9.30 is only available on 32-bit platforms that support the Java HotSpot Virtual Machine. IDS J/Foundation enables the building and execution of Java routines, either in the form of stored procedures or JavaBeans, directly in the database engine.

Other Foundation bundles, such as the Financial and Law Enforcement Foundations, are still available. These Foundation bundles contain technologies such as the TimeSeries, Biometric and Numerical Analysis Group (NAG) Algorithm DataBlades.

2 New features in IDS Version 9.30

This section will focus on some of the new features in IDS Version 9.30. Generally speaking, these features fall into three major categories: performance, SQL enhancements and administration utilities.



2.1 Performance features

2.1.1 SQL statement cache

The SQL statement cache maintains prepared and optimized SQL statement access plans in a special set of buffer pools for re-use by other users executing the same statement. In earlier versions of the engine, each statement's preparation and access plan was only available to the session executing the statement. If the same statement was being executed in one or more subsequent sessions, as is usually the case in an OLTP environment, each of those sessions had to prepare and optimize their iteration of the statement. This is grossly inefficient in terms of overall transaction throughput.

With the SQL statement cache enabled in the instance, when a data manipulation (DML) statement such as an insert, update or delete is received, the engine checks to see if the statement has already been executed within the instance and if its plan exists in the cache. If so, the existing plan is reused and the statement is immediately executed.

If the statement does not exist in the cache, it is optimized, executed and, depending on how the cache is being managed, its plan is entered into the cache for re-use by other sessions. Tests by IBM indicate that the re-use of plans from the cache can lead to an increase in performance of the statement of up to five times. The engine SQL statement cache can be used in place of application-level statement caches as well, reducing the amount of resources required on machines executing and supporting applications.

The SQL statement cache can be turned on or off at either an instance or session level. The number, size and activity within the cache pools can be monitored and managed as needed to maintain peak performance.

2.1.2 Fuzzy checkpoints

Fuzzy checkpoints were introduced in IDS Version 9.2. The goal of fuzzy checkpoints is to increase transaction throughput by virtually eliminating instance wait time while data is flushed from shared memory buffer pools to disk.



In earlier versions of the engine, newly entered or modified data was stored in buffer pools maintained by the instance. A record of the change was also written to a series of logs, called logical logs, that track changes to data, including deletes. On a regular basis, the contents of these buffer pools were written to disk to ensure logical and physical consistency of the data. Now called a “sync checkpoint,” these buffer pool flushes can take from as little as a sub-second to many minutes to complete depending on how much data must be flushed to disk and how well the instance is managed. During this type of checkpoint, certain end-user activities are temporarily suspended, forcing applications to wait for a reply. This, in turn, forces users to wait, which can have an impact on their ability to do their jobs.

In IDS Version 9.2, certain types of transactions, specifically insert, update and delete statements, were re-classified as potentially “fuzzy” operations and could be handled through the use of a new “fuzzy” checkpoint. During a fuzzy checkpoint, the instance doesn’t flush the buffer pool to the disks. Instead, a record of the dirty buffers and their relationship to transactions recorded in the logical logs is written into the logical logs. Rather than suspending instance processing to flush the buffers, the dirty buffers are gradually written to disk over time in a trickle-feed approach following the checkpoint. During a fuzzy checkpoint, the only interruption to end-user processing occurs when the record of changes is written to the logical logs. This generally takes less than a second to complete, virtually eliminating interruption of end-user activities.

When using fuzzy checkpoints, it is important to regularly back up both your instance and logical logs and to verify the integrity of those backups on an occasional basis. As opposed to sync checkpoints, with fuzzy checkpoints you do not have known moments in time when there is complete logical and physical consistency of the data. Rest assured, you can completely restore an environment since all the changes are still written into the logical logs. You’ll just need to go through a more extensive logical portion of the restore process using most of the recently used logical logs to completely restore an instance or dbspace.



2.1.3 Memory resident tables

Within an instance's shared memory pools, there is a pool for the storage of new or modified data. When a request for data is made, the instance checks this pool first to see if the data has already been retrieved for another operation and can be re-used. If found, the pool's copy of the data is used, eliminating the wait time associated with retrieving the data from disk. Unless any given piece of data is constantly being requested, it will eventually be dropped from the pool so that other pieces of data can be stored for more current operations.

In addition to caching independent pieces of data for current operations, this pool can also be used to store tables, indexes or fragments of either such that they are never dropped from the pool. Called declaring the table or index "memory resident," a copy of the table, index or fragment thereof is stored in the buffer pool, where it remains until the instance is shut down, the table/index or fragment is dropped from the database or the DBA declares the table/index or fragment "nonresident."

Obviously, declaring a database object to be memory resident can have a significant impact on querying data from that object. Instead of waiting for the less commonly used data to be read from disk to satisfy a query, the entire contents of the cached object are instantly available and, generally speaking, remain so for the duration of the instance up-time.

Declaring a database object to be either resident or non-resident is done through a simple SQL statement. Use of memory resident database objects should be carefully considered however. Depending on the size of the object(s) you want to declare resident, you can quickly use up instance shared memory resources.



2.1.4 Optimizer directives

IDS, like all database engines, has a query optimizer to figure out the quickest and least expensive way to access data requested by a given user session. Optimizers are not trivial pieces of technology. There are fields of study devoted to the creation of sophisticated mathematical models and theorems to govern how optimizers can and should work. These models and theorems have had to adopt and change over the years as changes in hardware technology have changed the “real” costs of accessing data. As a result, the IDS optimizer has been continually improved as new mathematical behavior models have emerged.

While at its heart, the IDS optimizer is considered a “cost-based” SQL optimizer, it is more complicated than that. With the engine’s object-relational architecture, the optimizer also contains a significant amount of object-oriented optimizer technology to correctly process operations using user-defined datatypes, functions and other technologies that a purely relational database engine does not need to worry about.

In any given situation though, the optimizer may not yield the best query plan based on real-life testing and experience. Rather than be limited to the plan derived by the engine, IDS provides a way whereby the DBA can make recommendations to the optimizer on how to process a specific statement. Called “optimizer directives,” these hints can take two forms—“do this” or “don’t do this.” With a directive, the DBA can tell the optimizer to use (or not use) a specific index, join technology, hash order, or whether or not to prepare to bring back all rows that match the selection criteria or just the first n rows. Directives can be used in “what-if” analyses during application development to determine whether or not database model changes, such as the creation or deletion of indexes, are necessary to increase production performance.



Depending on the application language being used, hints can be embedded into the SQL statement in one of three ways. Multiple directives can be added to a single statement if needed as well. Regardless of the method used, it is important to realize that over time, the optimizer will change how it processes statements. Changes will occur as data size, skew change and new statistical information is provided to the optimizer through the execution of the “update statistics” command on an as-needed basis. As such, it is important to regularly revisit any queries using optimizer directives to see if the directives are still needed. Improper directive use can have a negative impact.

See section 2.2.11 “Explain without execute” for one example of an optimizer directive.

2.1.5 Optimizer extension: subquery flattening

The IDS optimizer also supports what is known as “subquery flattening” for nested loop semi-joins. The best way to explain this is to use an example. Suppose you need to know all the customer orders for customers whose shipping address is in Texas. The conditional elements of your query would most likely state that the “order shipping address state” value must be “texas” and that there should be a correlation between the “order customer number” and the “customer customer number.”

Most optimizers would parse this type of request and attempt to scan the entire order table looking for addresses in Texas, then attempt to join them to customer-related information through the customer number. Generally speaking, the order table should be a lot larger than the customer table so the engine scans the largest table first—which is, of course, the slowest way to access data.

With the addition of an index on the “state” column in the customer table and optimizer subquery flattening, when the query is re-run, the optimizer will parse the customer table, first looking for Texans and then automatically joining the order information in one pass through the table. Needless to say, the performance characteristics are significantly better.



2.1.6 Optimizer extension: key-first index scan

The concept of key-first index scans is to use the least costly manner to eliminate possible data mismatches before having to go to disk and read the row to see if, in fact, it satisfies the query conditions. To do this, the engine applies the data conditions (the “where” clause of the query) to all the index(es) that exist on the table to find possible matches before reading a row from disk. Query performance is improved since index scans usually occur significantly faster than data reads.

2.1.7 Enterprise Replication enhancements

IDS Version 9.30 provides significant Enterprise Replication (ER) feature enhancements and performance improvements. Of particular note are the following:

- Datasync improvements including support for intra-replicate parallelism and out-of-order commits. Customers in the 9.30 beta program reported two- to three-fold improvements in their deployments.
- Decrease in logical log consumption.
- Interoperability with ER running on IDS Version 7.31 and Version 9.2x.
- Serial column primary key support (both standard and serial 8 types are supported).
- Smart LOB support.
- Distinct and opaque datatype support (no complex datatype support, however).
- Automatic addition/removal of shadow columns for conflict resolution during in-place alter table operations in most cases.
- Replicate sets that replace the “groups” feature by managing collections of replicates. An extension to this feature, exclusive replicate sets, preserves referential integrity in time-based replication by guaranteeing that replicated transactions are applied at the targets in the same order as in the source.



ER in IDS Version 9.30 can now construct hierarchical replication (HR) scenarios through the definition of root, hub and leaf servers. These scenarios can be in one of two forms—“forest of trees” or “hierarchical trees” depending on your needs. In earlier versions of the engine, all IDS instances participating in an ER scenario had to have a direct connection to all the servers in the scenario. While this option is still available, it can create complex replication networks that are difficult to maintain, particularly in geographically large and distributed environments.

2.2 SQL features

2.2.1 Long identifiers

One of the most popular user requests was the elimination of the “8/18” rule where user identifiers were limited to 8 characters in length while database object names (such as table or index names) had to be 18 characters or less. IDS Version 9.30 provides much-needed relief from these restrictions.

With IDS Version 9.30, user identifiers can be up to 32 characters long. Database object names can be up to 128 characters. Objects benefiting from this longer naming scheme include table names, server names, database names, column names, index names, synonym names, constraint names, procedure names, dbspace names, blob space names, optical cluster names, trigger names, type names, routine language names, access method names, operator class names, trace message class names, trace message names and procedure variable names.

With these longer identifiers (both user and object), you can use a more intuitive naming scheme as well as more easily migrate from other database environments to IDS.



2.2.2 “On Select” trigger

In earlier versions of IDS, you could only create triggers that operated on “delete”, “insert” or “update” SQL operations. With IDS Version 9.30, you can now create a trigger that executes in conjunction with an SQL “select” event. With this feature, DBAs and engine administrators have new options for creating auditing or security features, click stream analysis, etc.

2.2.3 SQL extension: “rename index” statement

The “rename index” statement makes it easy to rename an existing index in a database.

Prior to IDS Version 9.30, to rename an index, you had to drop and re-create it, which required exclusive access to the table and, sometimes, long build times. This SQL statement extension changes the index name in the system tables, eliminating the need to recreate the index.

2.2.4 SQL extension: ANSI outer join syntax

This SQL extension significantly increases the performance of complex outer join queries. In IDS Version 9.30, IBM has implemented the American National Standards Institute (ANSI) syntax standard for outer joins. In third-party benchmarks, IDS Version 9.30 outer join queries now perform significantly better than comparable queries executed in competing database systems.

In addition to providing better performance, this feature facilitates easier application migration from other database systems to IDS.

2.2.5 Retain update locks

In earlier versions of IDS, data locks are placed and released based on a couple of environmental conditions, including the query isolation level and the type of operation being executed (e.g. cursor-based read or update). In some isolation levels, data is unlocked as the cursor moves from one data element to another—assuming, of course, that the data itself was not changed while locked. In other levels, only committed data is read and locked. In others, no locks are placed and all rows are read, both committed and uncommitted.



IDS Version 9.30 has a new SQL command—“retain update locks”—which, regardless of the isolation level, locks all rows read for potential update until the transaction is closed. This prevents other sessions from locking updated rows that have been unlocked by the updating session but have not been committed. By using this feature, you can eliminate inefficient coding techniques such as dummy updates or the (unnecessary) usage of the “repeatable read” isolation level.

2.2.6 SQL functions

IDS Version 9.30 offers several new SQL functions that perform tasks such as upper/lower/mixed case conversion, sub-string functions and a “case” statement. In many cases these new, built-in functions can replace homegrown stored procedures, improving performance.

Some of these new functions are:

- “NVL”—maps a NULL to a given value
- “Case” statement—support of SQL-92 conditional expressions
- “Decode”—similar in operation to a “case” statement, IDS Version 9.30 now supports “decode” syntax structures
- String/date/datatype conversions between different datatypes
- Union views—the ability to create a “view” that incorporates the “union” syntax in the “create view” statement
- Updates to the output from the “describe” SQL keyword. Depending on the type of SQL statement in which the “describe” keyword is used, the engine will reply with information about the SQL statement. The information could be the datatypes of the columns affected, the potential number of rows that might be affected if the statement were executed and what the “where” clause is.

2.2.7 Raw tables

“Raw” tables have been a feature of the IBM XPS product for some time. In contrast to IDS “standard” tables, a raw table is a permanent, non-logged table that can exist in a logged database. Originally designed for supporting massive data loads in a data warehouse environment, raw tables use “light appends” to add data to the end of the table’s fragments. Update and delete operations to data in raw tables are supported as well, but in all cases, transaction information is not captured in the logical logs.



Standard tables can be converted to raw mode and vice versa with the “alter table” command as needed. Any indexes on the table must be dropped prior to converting from standard to raw mode.

It is important to remember that because changes to raw tables are not tracked in the logical logs, there is no recovery mechanism. Once a table has been converted back to standard mode, you should create a backup of the dbspace to capture any data changes.

2.2.8 In-place alter table

The “in-place alter table” algorithm functionality has been extended to handle nearly all “alter table” operations in fractions of a second, without requiring double the disk space for each table to be altered. This reduced disk space requirement, together with the rapid execution of the command, make it much easier to apply table changes to a production system.

2.2.9 Attach/detach fragment extensions

The ability to attach and detach fragments from a table has been optimized in IDS Version 9.30. Existing indexes will be used more efficiently during an attach operation, eliminating some of the cost associated with rebuilding all the fragments.

2.2.10 Optional “from” in a delete statement

With IDS Version 9.30, the “from” keyword is optional in an SQL “delete” statement. As a result, both of these statements would be correct in terms of syntax:

```
delete from customer where cust_id = 1550
```

```
delete customer where cust_id = 1550 clause
```

This feature can make it easier to migrate applications from another database environment to IDS.



2.2.11 Explain without execute

Another long-asked-for user request has been the ability to see optimizer plan information without actually executing the SQL statement. This has been particularly critical when trying to review and evaluate update or delete statements. IDS Version 9.30 has new functionality that directs the optimizer to stop processing the statement once the prepare phase has completed and has output its plan information to a file. Unlike IBM XPS, the IDS optimizer puts optimizer plan output in a file called “sqexplain.out” in the current working directory.

You can turn this functionality on or off as needed within any given session in one of two ways—an optimizer directive or through an SQL statement. Both require the use of the new “avoid_execute” keyword to stop the optimizer at the prepare phase. For example:

```
set explain on avoid_execute;  
select c1, c2, c3 from....
```

Using optimizer directives:

```
select—+explain avoid_execute c1, c2, c3 from....
```

2.2.12 Configurable default lock mode

In earlier versions of IDS, the default lock mode for newly created tables was at a page level. Even if you only needed to lock one row of data, the entire page was locked. This led to concurrency issues and was partially resolved by the ability to specify the lock mode (either page or row) for a table at its creation in later versions of IDS. IDS Version 9.30 provides a way to override the engine’s default table lock mode for newly created tables.

The default lock mode can be assigned, and changed, on an instance-wide or session-by-session basis through the use of either an engine configuration file parameter (DEF_TABLE_LOCKMODE) or an environment variable (IFX_DEF_TABLE_LOCKMODE). In either case, a user can over-ride the default by putting specific locking instructions (“lock mode row/page”) in the “create table” statement. Please note that this functionality only extends to newly created tables; it has no effect on existing tables. The lock mode of an existing table can be changed with the “alter table” command.



2.2.13 Revoke as user

The “grant” and “revoke” statements allow the owner of a database object to assign or revoke the privileges of other users on that object. Earlier versions of the engine supported the following syntax:

```
GRANT ... TO <user1> AS <user2>
```

IDS Version 9.30 adds new syntax and functionality to the “revoke” and “revoke fragment” commands so that <user2> can revoke the privileges for <user1> with the following command:

```
REVOKE ... FROM <user1> AS <user2>
```

2.3 DBA features—utilities

2.3.1 Dynamic lock allocation

In earlier versions of IDS, database locks were a fixed, tunable commodity. Each lock used X KBS of the instance’s shared memory and engine administrators tried to ensure that statements requiring more locks than were configured were not executed. If these statements were executed, the dreaded “lock table overflow” event occurred with the possibility of data corruption. Significant changes have been made to lock processing in IDS Version 9.30.

With known Smart LOBs, as well as the ability to create new and different user-defined datatypes of unknown size or content, IDS can no longer function with a finite set of locks. In addition, the engine has to be able to provide concurrent access to almost all the objects in the database instead of, for example, single-threading access to Smart LOBs. To accomplish this, IDS Version 9.30 has changed how the instance approaches object-locking as well as the behavior of the LOCKS configuration file parameter.

The LOCKS parameter is now the beginning allocation of locks within the instance and should be set to a reasonable number of locks expected to be used, instead of a worst-case scenario. This should release more shared memory for use in other portions of the instance. The instance itself will manage the number of locks and increase or decrease the number of locks as needed. These additional locks are managed in the virtual portion of the instance’s shared memory.



If the preliminary allocation of locks is not sufficient, the instance will attempt to create up to 16 additional allocations of locks. This process is managed such that if an individual session will be locking more than 10 percent of the available locks, or between 50 percent to 70 percent of a Smart LOB, a single, exclusive lock on the object will be set.

2.3.2 Dynamic logical logs

In earlier versions of IDS, logical log maintenance required estimating the size and number of logs that might be needed to support the instance's work load. If wrong, the instance had to be shut down to make any changes. With IDS Version 9.30, the engine administrator can configure the instance to automatically create, insert and activate a new logical log without interrupting instance processing. While this does eliminate the dreaded long-transaction event, activating this functionality eliminates a significant portion of the event's penalty, especially on instance startup.

The functionality is configurable so that the administrator can grant one of three levels of control to the instance when a new log is needed. In conjunction with the ability to add logical logs, an administrator can also drop logical logs that are no longer needed without interrupting instance processing.

The instance's ability to add new logs does not eliminate the requirement to regularly back up the logs to tape. They still need to be backed up to free them for re-use within the instance. These backed up logical logs will be required for instance recovery, particularly in the event of media failure.

2.3.3 Archecker

Archecker is a utility that has been available from the International Informix Users Group (IIUG) Web site for several years. Originally a utility for internal Informix developer use only, it is now bundled with IDS Version 9.30. Archecker is used to check the ability to recover data from instance backup media (e.g. tape or disk) without actually performing the restore. It goes without saying that a wise administrator should periodically check the integrity of his or her backup jobs for viability.



While Archecker is integrated into the “OnBar” utility (“onbar -v”), it can be used to check the integrity of backups created with the “ontape” utility. Archecker verification can be done on the same machine on which the backup was generated, or on another. It only requires a few megabytes of disk space to run.

2.3.4 Onsmsync

The “onsmsync” utility synchronizes the OnBar database (“sysutils”), the emergency boot file (“ixbar.servernum”) and information managed by the storage manager (ISM, Legato, Omniback, IBM Tivoli Storage Manager, etc.).

In earlier versions of IDS, these three elements had to be managed independently by the administrator. For example, if backup objects maintained by the storage manager expired and were deleted, the administrator had to make the appropriate changes in the “sysutils” database and the emergency boot file. Most of the time, this didn’t happen, leading to situations where a restore operation could not be executed because of missing information. In other cases, if the emergency boot file was allowed to continually grow, restore operations took longer to complete while the system parsed out-of-date entries. Use of this utility should make management of OnBar-related backup information easier to manage.

2.3.5 Command-line interface for the High-Performance Loader (HPL)

Prior to IDS Version 9.30, an administrator had to use a graphically based tool to create and execute High-Performance Loader (HPL) jobs. While not the worst interface in the world, it certainly was not the best. Administrators can now take advantage of the “onpladm” command line interface to the HPL.

Through this command line interface, administrators can dynamically generate HPL jobs through shell scripts, adding more flexibility and integration of the HPL into their existing workflow.



2.3.6 Oncheck without locks

As briefly discussed in Section 2.3.1, Dynamic Lock Allocation, the locking mechanism in IDS Version 9.30, has changed from earlier versions of the engine. In addition to the benefits discussed in that section, an administrator can now execute index consistency checks (“oncheck -ci / -cl”) without having exclusive access to the table. There is, however, one caveat to these types of checks—the “oncheck” command will not re-evaluate index pages that may have changed after being scanned by the utility.

With this new functionality, administrators can now perform regular consistency checks on the data during the lighter processing periods of the day or week instead of waiting to take the instance off-line in a maintenance period.

2.3.7 Informix Storage Manager (ISM) interface change

For some time, IDS has included a storage manager with basic functionality as part of the product. Informix Storage Manager (ISM) could be used in conjunction with the “OnBar” utility to create serial or parallelized backups to no more than four physical devices. In earlier versions of the engine, the ISM could be configured or used through either a graphical or command line interface. With IDS Version 9.30, you can only use the command line interface to access and use the ISM.

2.3.8 External backup and restore (EBR)

With the standard instance backup and recovery utilities (i.e. “onbar” and “ontape”), the process of selecting data and passing it to the backup utility is managed by the instance itself. While this can and does work, creating backups of large instances is time consuming. With the plunging cost of hardware, particularly in storage, most instances are now using at least hardware-based mirroring to protect against data loss. The existence of these mirrors can be exploited to create faster backups and, by extension, restores.



The concept of an “external” backup is rather simple. In a hardware-based mirroring environment, there is at least one, if not more, exact copies of the data on disk. If you temporarily disassociate one disk copy from the rest, you have a moment-in-time view of the data. You can then push data from this snapshot through the “onbar” utility to the storage manager as if it were a level 0 backup. During this backup operation, the instance is unaffected because it does not have to manage the scan and retrieve operations. The net result is a backup operation that completes faster with little to no impact on the instance. The set of disks can then be re-associated with the mirror set and the hardware-based mirroring mechanisms will handle resynchronizing of the disks so they all match again.

An external restore operation is not unlike an “ontape” restore, though it can be in parallel if the backup was created in parallel. Since the instance is aware of when the disk copy was broken off, it knows which logical logs to request to roll forward missing transactions once the physical restore is completed.

By using EBR, very large databases (hundreds of gigabytes or more) can be more easily, and regularly, backed up and, if need be, restored in much shorter time frames.

2.3.9 Restartable restore also a feature from 7.3

A relatively new engine feature, when configured into the instance, the restartable restore feature enables a failed OnBar restore operation to restart at the moment of failure, rather than at the beginning, and rewrite to already restored storage spaces. The overhead supporting this functionality is stored in the sysutils database, which directs the associated storage manager to scroll forward to the appropriate location to begin re-reading the media.

This functionality is controlled by the `RESTARTABLE_RESTORE` configuration parameter and through the use of the “-restart” flag in the “onbar” command.



2.3.10 New `ex_alarm.sh` script for alarms

For several versions, IDS has provided an interface through which administrators could capture “alarm events,” such as full logical logs, disk failure, etc., and have the engine automatically execute specific routines depending on the event. Generally referred to by its configuration parameter name (ALARMPROGRAM), earlier versions of IDS provided two alarm scripts as examples—“`log_full.sh`” and “`no_log.sh`,” which either automatically backed up full logical logs or didn’t.

With IDS Version 9.30, the “`ex_alarm.sh`” script can be used on UNIX[®] systems to handle event alarms, including the automatic back-up of full logical logs. In this way, all event alarms can be handled through one script, minimizing administrator overhead.

2.3.11 UNIX Bundle Installer

The UNIX Bundle Installer removes the complexity of installing multiple engine components (such as connectivity, DataBlades, etc.) and assures that the components are installed in the correct order. With the Installer, you can also quickly create a demonstration database instance.

3 MaxConnect

3.1 Description of MaxConnect

MaxConnect is a separately orderable piece of software that sits between a database instance and its clients. Rather than connecting to the instance, clients connect to the MaxConnect instance which, in turn, multiplexes the connections to the instance. For example, 500 client connections can be multiplexed to less than a handful of instance connections without any loss in performance. The net effect is the reduction of instance resources needed to support client connectivity and communication. The CPU VP cycles that would normally be used to receive and send messages to clients can now be used to process SQL operations instead.



The most important feature of MaxConnect is application transparency. Depending on how you install and configure MaxConnect, there could be few to no changes required at the client side to use MaxConnect. The application itself does not need to be changed or recompiled to use MaxConnect. As a result, MaxConnect can be easily inserted into any environment where large numbers of users are accessing IDS instances.

3.2 Configuration options for MaxConnect

As illustrated in Figures 1 and 2, MaxConnect can be installed in a two- or three-tier environment.

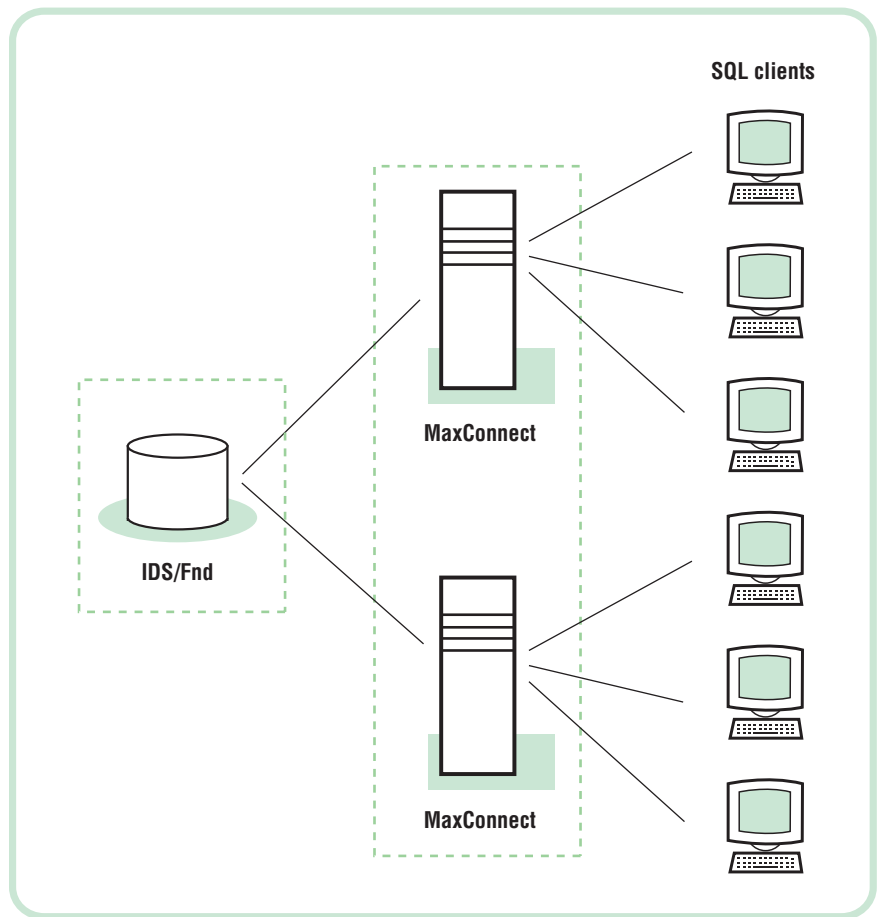


Figure 1. MaxConnect installed on a separate server in a three-tier environment



For large, hardware-partitionable SMP systems:

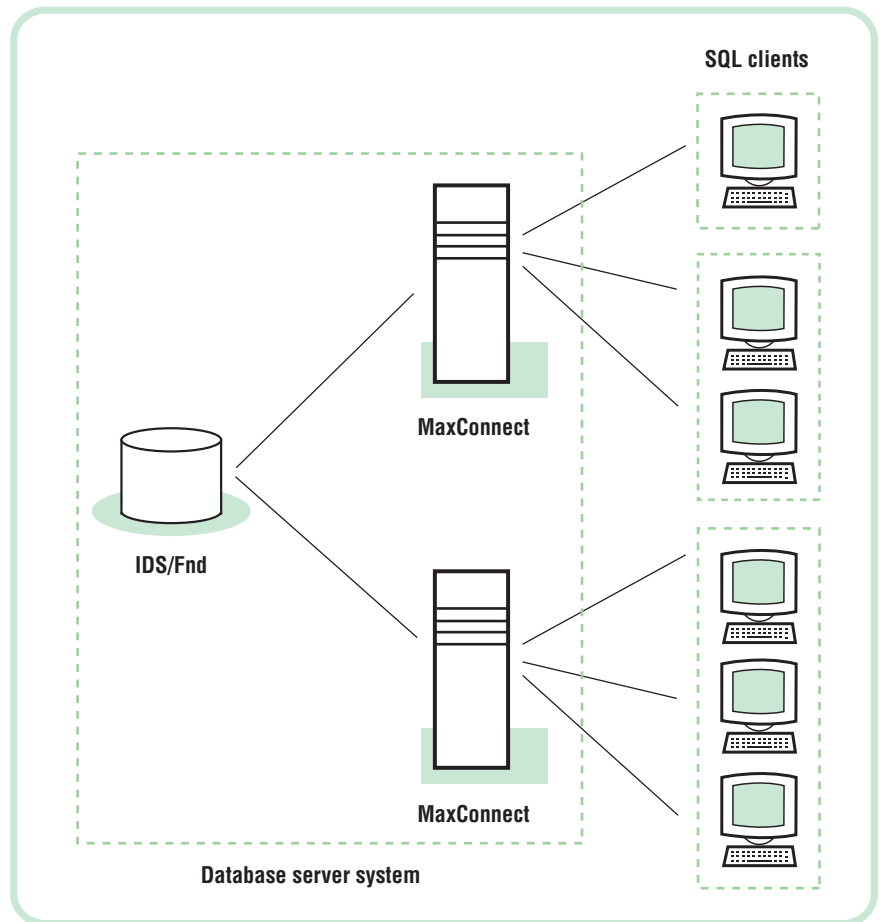


Figure 2. MaxConnect installed on a hardware-partitionable server in a two-tier environment

Usually, MaxConnect is installed on a separate machine (three-tier) to reduce the network traffic on the server supporting the database instance. If you have a larger, hardware-partitionable machine that can support multiple network interface cards, and the O/S can handle the network traffic, you can install MaxConnect on the same physical server supporting the instance. Generally speaking, the performance improvements in two-tier implementations are not as dramatic as in three-tier.

To use MaxConnect, there are minor changes that need to be made to the NETTYPE and INFORMIXSERVERALIAS parameters and in the SQLHOSTS file on the database instance. The MaxConnect instance has a very simple configuration file as well as its own SQLHOSTS file.



Once MaxConnect is installed and configured, minimal changes are required at the client to use it. In two-tier deployments, no changes are required. In a properly configured three-tier environment, you only need to change the host name in the client's SQLHOSTS file. All the other communication parameters remain the same.

4 IDS Benchmark Results

4.1 IDS without MaxConnect

As mentioned at the beginning of this white paper IDS 9.30 is a "significant" upgrade to 9.21, which incorporated almost all of the needed fixes and corrections. A short time ago, a benchmark was run for a major ERP vendor to show the performance improvements made in IDS Version 9.21 as well as to highlight the effectiveness of MaxConnect. While this paper has focused on 9.30 we feel it is relevant to include this 9.21 performance benchmark in order to give you a comfort level with the performance features of the IDS 9 product family as a whole.

Using an HP K570 (six CPUs) database server and five HP K580 (six CPUs each) application servers, the following results were recorded:

- IDS 9.21 could support 24 percent more users
- The system's memory utilization could be reduced by 22 percent
- The CPU utilization could be reduced by 16 percent.

The detailed results are as follows:

	IDS 9.21	IDS 7.3	Improvement
Throughput			
Max users (on 6 CPUs)	4,336	3,485	24% better
Transactions/minute	12,816	10,784	18.7% better
Users/CPU	720	580	24% better
Results for 3,375 users			
Response time	~1.4 sec	~1.55 sec	~6% faster
User CPU %	47%	56%	16% less
Memory per user	4.47MB	5.8MB	22% less



4.2 IDS with MaxConnect

While running a BAAN benchmark on an HP N-Class machine (eight CPUs) and MaxConnect, there was nearly linear scalability by adding additional CPUs to the server supporting the database instance.

During this BAAN benchmark, IBM Informix generated the best results of all participating database vendors. The IDS benchmark was able to reach 11,445 BRUs (Baan Reference Units) while supporting 18,650 concurrent connections to the instance. This result was 21 percent better than Oracle on the same hardware.

5 IDS overview

5.1 Features and versions

5.1.1 Performance features

Feature	7.2x	7.30	7.31	9.21	9.30
Shared statement cache				√	√
Fuzzy checkpoints				√	√
Memory-resident tables		√	√	√	√
Optimizer hints		√	√	√	√
Optimizer extension: Subquery flattening		√	√	√	√
Optimizer extension: key first index scan			√	√	√
Optimizer & update statistics improvements					√
Enterprise replication performance improvements					√



5.1.2 SQL features

Feature	7.2x	7.30	7.31	9.21	9.30
Long identifiers				√	√
On select trigger				√	√
SQL extension: "Rename index" statement			√	√	√
SQL extension: ANSI outer joins			√	√	√
Retain update locks			√	√	√
SQL functions (case, upper, etc.)		√	√	√	√
Non-logging tables			√	√	√
Alter table in place		√	√	√	√
Attach/detach fragment extensions			√	√	√
Optional "from" in a delete clause					√
Display a query plan without executing query					√
Revoke as user					√
Configurable default lock mode					√



5.1.3 DBA features—utilities

Feature	7.2x	7.30	7.31	9.21	9.30
Dynamic lock manager				√	√
Dynamic logs					√
Archecker			√	√	√
Onsmsync			√	√	√
Hierarchical replication				√	√
Command line interface for High-Performance Loader				√	√
Oncheck without locks		√	√	√	√
ISM Informix Storage Manager		√	√	√	√
EBR external backup restore		√	√	√	√
Restartable restore		√	√	√	√
MaxConnect supported			√	√	√
New ex_alarm.sh script					√
UNIX Bundle Installer					√
AGS ServerStudio JE bundled with IDS					√
ER allows serial primary keys in update-anywhere					√

5.1.4 Additional features

IDS is an industry-leading, extensible database. Business logic can be executed at the server in user-defined functions as well as stored procedures. User-defined functions can be written in SPL, C or Java and can be used, for example, to encapsulate multiple SQL statements and to extend existing SQL functions to perform business-specific operations. By moving these operations closer to the data you can cut down on network traffic and improve database performance, as well as simplify application development and enhance processing consistency.



Here is a list of additional features in IDS not covered in this document that could be important for some applications:

Feature	7.2x	7.30	7.31	9.21	9.30
Onbar progress feedback				√	√
Onbar CLI cleanup				√	√
Parallel recovery				√	√
Serial types in raw types				√	√
Embedded new lines in quoted strings				√	√
Compiled expressions				√	√
Online optical on Microsoft® Windows NT®		√	√	√	√
Raw device support on NT		√	√	√	√
Wolfpack support for NT		√	√	√	√
Microsoft Transaction Server support in XA				√	√
Recursive update triggers				√	√
Support for JVM 1.2				√	√
JDBC 2.0 feature set				√	√
Dynamically drop JVPs				√	√
UDR thread optimization				√	√
Enhanced sort and direct function for R-Tree build				√	√
Nearest neighbor queries with R-Trees					√
Bounding box only R-Tree indexes					√
Oncheck/onlog for R-Trees				√	√
Variable length UDT				√	√
C++ hooks				√	√
Temporary Smart BLOBS					√
Smart BLOB space management simplification					√
Enterprise replication support for Smart BLOB, UDTs, spatial data					√
Support for JVM 1.3					√
SAPI interface enhancements					√



5.2 Compatibility between IDS and other Informix products

5.2.1 Certified IBM Informix DataBlade modules for IDS

The following IBM Informix DataBlade modules and DataBlade utilities have been certified for IDS:

- Blade Manager 4.00
- DBDK 4.00
- Excalibur Image DataBlade 1.21.UC1
- Excalibur Text DataBlade 1.30.UC6
- Geodetic DataBlade 3.00.UC1
- Spatial DataBlade 8.11.UC1
- TimeSeries DataBlade 4.00.UC1 or higher
- IBM Informix Real-Time Loader (version TBD)
- NAG DataBlade (version TBD)
- Image Foundation DataBlade (version TBD)
- Video Foundation DataBlade 2.00.UC3
- Web DataBlade 4.00 or higher.



5.2.2 Certified products for IDS

The following products have been certified for IDS:

- IBM Informix Server Administrator 1.40
- IBM Informix Connect 2.50, 2.60, 2.70
- IBM Informix JDBC 2.20/ESQL J 1.01
- IBM Informix JDBC 1.50
- IBM Informix 4GL 6.05, 7.20, 7.30
- IBM Informix SQL 6.05, 7.20, 7.30
- Informix MetaCube™ 4.x
- Informix Data Director for VisualBasic 3.5 and above
- Informix Enterprise Gateway Manager 7.2x
- Informix Enterprise Gateway with DRDA® 7.2x and above
- MaxConnect 1.00
- Office Connect 2.00
- Object Translator 2.00
- Server Studio JE 2.0.JC1 (includes Object Explorer, SQL Editor and Table Editor).



© Copyright IBM Corporation 2002

IBM Corporation
Silicon Valley Laboratory
555 Bailey Avenue
San Jose, CA 95141
U.S.A.

Printed in the United States of America
10-02
All Rights Reserved

DataBlade, DB2, DB2 Universal Database, DRDA, the e-business logo, IBM, the IBM logo, Informix, Lotus, MetaCube, Tivoli and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries or both.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries or both.

UNIX is a registered trademark of The Open Group in the United States and other countries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries or both.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. Offerings are subject to change, extension or withdrawal without notice.



Printed in the United States on recycled paper containing 10% recovered post-consumer fiber.

