



IBM Informix Dynamic Server, Version 9.30 における機能拡張

*by Carlton Doe
Technical Sales Manager
BM Data Management*

目次

概要	1
1 IDS 9.30 と IDS with J/Foundation 9.30 の違い	3
1.1 <i>IDS 9.30</i>	3
1.2 <i>IDS with J/Foundation 9.30</i>	4
2 IDS Version 9.30 の新機能	4
2.1 パフォーマンスに関する機能	5
2.1.1 <i>SQL ステートメント・キャッシュ</i>	5
2.1.2 <i>ファジー・チェックポイント</i>	5
2.1.3 <i>メモリ常駐表</i>	7
2.1.4 <i>オブティマイザ・ディレクティブ</i>	8
2.1.5 <i>オブティマイザの機能拡張: 副問合せのフラット化</i>	9
2.1.6 <i>オブティマイザの機能拡張: キー優先インデックス走査</i>	10
2.1.7 <i>エンタープライズ・レプリケーションの機能拡張</i>	10
2.2 SQL に関する機能	11
2.2.1 <i>ロング識別子</i>	11
2.2.2 <i>"select文" トリガ</i>	12
2.2.3 <i>SQL の機能拡張: "RENAME INDEX" 文</i>	12
2.2.4 <i>SQL の機能拡張: ANSI 外部結合構文</i>	12
2.2.5 <i>更新ロックの保持</i>	12
2.2.6 <i>SQL 関数</i>	13
2.2.7 <i>ロウ表</i>	13
2.2.8 <i>インプレース表変更</i>	14
2.2.9 <i>フラグメントの関連付け/切離し機能拡張</i>	14
2.2.10 <i>DELETE 文における "FROM" のオプション化</i>	14
2.2.11 <i>非実行エクスプレイン</i>	15
2.2.12 <i>構成可能なデフォルト・ロック・モード</i>	15
2.2.13 <i>ユーザとしてのRevoke</i>	16
2.3 DBA の機能 - ユーティリティ	16
2.3.1 <i>動的なロック割り当て</i>	16
2.3.2 <i>動的論理ログ</i>	17

目次

2.3.3	<i>Archecker</i>	17
2.3.4	<i>Onsmsync</i>	18
2.3.5	<i>High-Performance Loader (HPL) のコマンド・ライン・インターフェイス</i>	18
2.3.6	<i>ロックなしの Oncheck</i>	19
2.3.7	<i>Informix 格納域マネージャ (ISM) のインターフェイス変更</i>	19
2.3.8	<i>外部バックアップおよび復元 (EBR)</i>	19
2.3.9	<i>IDS 7.3 からの再開可能復元機能</i>	20
2.3.10	<i>アラームの新しいスクリプト <code>ex_alarm.sh</code></i>	21
2.3.11	<i>UNIX Bundle Installer</i>	21
3	MaxConnect	21
3.1	<i>MaxConnect の説明</i>	21
3.2	<i>MaxConnect の構成オプション</i>	22
4	IDS のベンチマーク結果	24
4.1	<i>MaxConnect を使用しない場合</i>	24
4.2	<i>MaxConnect を使用した場合</i>	25
5	IDS の概要	25
5.1	機能とバージョン	25
5.1.1	<i>パフォーマンスに関する機能</i>	25
5.1.2	<i>SQL に関する機能</i>	26
5.1.3	<i>DBA の機能 - ユーティリティ</i>	27
5.1.4	<i>その他の機能</i>	27
5.2	IDS と他の Informix 製品との互換性	29
5.2.1	<i>IDS 向けの認定 IBM Infomix DataBlade モジュール</i>	29
5.2.2	<i>IDS 向け認定製品</i>	30



概要

本文書には、IBM Informix® Dynamic Server (IDS) 9.30 の機能および利点、特に IDS 7 と比較した場合の機能および利点に関する詳細なサマリが記載されています。重要な技術的詳細を掘り下げる前に、IDS 9.30 の全体的な利点と、IDS 7 に満足されているユーザにとって、IBM と協力してこのすばらしい新テクノロジーに移行する価値があるかどうかを見ていきましょう。

まず最初に、IBM は別バージョンへの移行を強制しているわけではありません。それでもやはり、IDS 製品ラインの新規の機能拡張および開発は主に IDS 9.x バージョンが対象となります。それらの作業のほとんどは新機能に重点を置いて行われますが、IDS 9.30 では、コード・パス短縮や関数の再設計などの領域で内部最適化が多数行われています。その結果、システムの CPU およびメモリ・リソースの効率的な使用方法が著しく改善されています。IBM は、お使いのデータベース環境を見直し、エンジンを新バージョンに移行する価値がそれらの新機能に含まれているかどうか検討することをお勧めします。

IDS 9.30 は、管理に必要なリソースが競合他社の製品よりも少ないという Informix データベースの伝統を受け継いでいます。IDS 9.30 ベータ版の顧客である、S1 の James Horn 氏は、「Informix は管理とサポートが容易な上、当社にとって非常にコストパフォーマンスに優れていました。これが、当社成功の秘訣の 1 つです... 当社の収益性は基礎システムのコストを低く抑えられるかどうかにかかっているため、TCOが鍵なのです」と証言しています。IBM の顧客は一般に、IBM IDS 製品の管理に必要なスタッフの数が競合製品を大きく下回ると報告しています。

IDS 9.30 は、使いやすさ、管理のしやすさ、ハイパフォーマンス、エンタープライズ・レプリケーション、相互運用性、e-ビジネス統合、および豊富なマルチメディアのサポート機能を備えています。「パフォーマンスに関連する IDS 9.30 の機能は DBA の夢です... IDS 製品はおそらく現在利用できる最も高速かつスケーラブルなエンジンでしょう」Berkeley Information Systems の John Lusty 氏はこう述べています。

IDS 9.30 はすばらしい価値を提供します。競合他社であればたいい追加料金を請求されるような機能がバンドルされているか、基本製品に含まれています。その例としては、区画化、マルチメディア・サポート・モジュール、管理モジュール、および管理/アプリケーション開発ツールなどが挙げられます。



IDS 9 の将来のバージョンでは、パフォーマンスおよび機能の強化を継続すること、64 ビットのサポートなど、ハードウェア・プラットフォームおよびソフトウェア・プラットフォームのアップグレードを進めることに重点が置かれます。使いやすさは常に IBM ソフトウェアの目標であり、IDS は今後も、オートノミック・コンピューティングおよび IBM SMART イニシャチブの追加によって強化されます。これらの機能により、当社製品は使いやすく、管理しやすくなります。

もう 1 つの重点領域は、WebSphere[®]、Tivoli[®] および Lotus[®] ソフトウェア、そしてもちろん、IBM DB2[®] Universal Database[™] など、他の IBM ソフトウェア・コンポーネントとの統合をより緊密にすることです。当社顧客はこれによって、業界をリードする IBM ベンダーの 1 社から強力な e-ビジネス・ソフトウェア・スタックを入手できます。IDS 9 は既に IBM WebSphere Application Server、IBM WebSphere MQ、IBM Tivoli Storage Manager、IBM DB2 Relational Connect、IBM DB2 Table Editor 4.3、および IBM DB2 Web Query Tool 1 と統合されています。

では、IDS 7 から 9 への移行には何が必要でしょうか。IDS は常にアップグレードのしやすさを重視しており、その伝統は Version 9 でも続いています。アップグレードの情報、マニュアル、およびヒントに加え、Informix に関連する一連のコースが <http://www-3.ibm.com/software/data/infomix/education/> にあります。これらは、スキルを最新に保ち、IDS 9 の最適化方法を学ぶためのコースです。

具体的なアップグレードのコストを調べるには、IBM の営業担当者までお問い合わせください。

さて、言い忘れているといけないので先に申し上げますが、IBM によるこそ。私たちは貴社をお客様として迎えることを喜ばしく思い、引き続きのご愛顧をお願いいたします。少し時間をかけて、Informix 9.30 テクノロジーの採用をご検討ください。採用したことによって満足されると思います。



IDS 9.30

このホワイト・ペーパーの目的は、IBM IDS データベース・エンジンの最新の機能拡張について、簡単な概略を述べることです。本文書で強調されている機能は、このエンジンの 9.30 バージョンにおける新機能、または大幅に強化された機能です。

ここで強調されているテクノロジーはおおむね、データ・ウェアハウジングではなくオンライン・ランザクション処理 (OLTP) 環境で実行しているユーザの関心対象となります。しかしこのバージョンでデータ・マートやより小規模なデータ・ウェアハウス環境を実行するユーザにとって価値のある機能も含まれています。

1 IDS 9.30 と IDS with J/Foundation 9.30 の違い

1.1 IDS 9.30

IDS Version 9.21 は、IDS Version 7.3 のコードと、Informix 製品ラインで Informix Universal Server (IUS) Version 9.14 として表記される旧 Illustra 製品との統合によって開発されました。全般的には成功しましたが、IDS Version 9.21 には改善および修正が必要でした。IDS Version 9.30 は IDS Version 9.21 を大幅にアップグレードしたものであり、必要な修復および修正がほとんど組み込まれています。

IDS Version 9.30 を、ハイエンド OLTP アプリケーションの基本ビルディング・ブロックと考えてください。IDS Version 9.30 はマルチスレッド・コア・エンジンとオブジェクト・リレーショナル・テクノロジーを備えているため、データベースおよびアプリケーションの設計者は、競合他社製品には無い機能を利用できます。そのような機能としては、現実世界のビジネス・データをより厳密にモデリングできる新データ型の定義機能や、ビジネス・ロジックおよびアクセス方法を作成してサーバ内に直接格納できる機能などがあります。エンジン内にロジックと方法を格納することで、情報の操作方法に関して単一かつ一貫した視点を確保できます。IDS は堅牢性、スケーラビリティ、および拡張性に優れているため、企業全体にわたるデータ管理用として真っ先に選ばれる製品です。



1.2 IDS with J/Foundation 9.30

IDS Version 9.2 コア・エンジンのリリースと並行して、Informix も追加のデータ管理機能を備えた製品バンドルをいくつかリリースしました。“Foundation” と呼ばれるこれらのバンドルには通常、1 つまたは複数の DataBlade[®] と Java[™] 仮想マシンがコア・エンジンのコンポーネントとして付属しています。これらのバンドルのどれかを購入すれば、各コンポーネント（供給されている場合）を個別に購入するよりもコストを削減できることとなります。

IDS with J/Foundation Version 9.30 は、IDS Version 9.30 エンジンと、より大きな Foundation バンドルで最も要求の多かった機能、つまり Java 仮想マシンを合わせた、新しいエンジン・バンドルです。現時点では、IDS with J/Foundation Version 9.30 は Java HotSpot Virtual Machine をサポートする 32 ビット・プラットフォームのみで利用可能です。IDS J/Foundation では、ストアド・プロシジャまたは JavaBean の形式で Java ルーチンをデータベース・エンジン内で直接構築および実行できます。

Financial Foundation や Law Enforcement Foundation など、その他の Foundation バンドルも引き続き入手可能です。これらの Foundation バンドルには、Time Series、Biometric、および Numerical Analysis Group (NAG) のアルゴリズム DataBlade などのテクノロジーが採用されています。

2 IDS Version 9.30 の新機能

このセクションでは、IDS Version 9.30 の新機能をいくつか取り上げます。一般に、これらの機能はパフォーマンス、SQL 機能拡張、および管理ユーティリティという 3 つの主要カテゴリーに分類されます。



2.1 パフォーマンスに関する機能

2.1.1 SQL ステートメント・キャッシュ

SQL ステートメント・キャッシュは、同じ文を実行するほかのユーザが再利用できるよう、作成および最適化された SQL 文アクセス計画をバッファ・プールの特殊なセットで管理します。旧バージョンのエンジンでは、各文の作成およびアクセス計画は、その文を実行するセッションのみで利用可能でした。OLTP 環境で通常行われるように、1 つまたは複数の後続セッションで同じ文が実行される場合、セッションごとにその文を繰り返し作成し、最適化する必要がありました。これは全体的なトランザクション・スループットという点で非常に非効率的です。

インスタンスで SQL ステートメント・キャッシュが有効になっていると、INSERT、UPDATE、または DELETE などのデータ操作 (DML) 文を受信したとき、エンジンはその文がインスタンス内で既に実行されていて、その計画がキャッシュ内に存在するかどうかチェックします。キャッシュ内に存在する場合は、既存の計画が再利用され、文が即座に実行されます。

その文がキャッシュ内に存在しない場合は、文が最適化され、実行され、キャッシュの管理方法に応じてその計画が他のセッションで再利用できるようキャッシュに格納されます。IBM によるテストでは、キャッシュから計画を再利用することで文のパフォーマンスが最大 5 倍に向上する場合がありますと示されています。エンジンの SQL ステートメント・キャッシュは、アプリケーションレベルのステートメント・キャッシュの代わりにも使用できます。これによって、アプリケーションを実行およびサポートするマシン上に必要なリソースの量が削減されます。

SQL ステートメント・キャッシュは、インスタンス・レベルまたはセッション・レベルでオン/オフを切り替えられます。最高のパフォーマンスを維持するために、キャッシュ・プール内の文の数、サイズ、およびアクティビティを必要に応じて監視できます。

2.1.2 ファジー・チェックポイント

ファジー・チェックポイントは IDS Version 9.2 で導入されました。ファジー・チェックポイントの目標は、データが共有メモリ・バッファ・プールからディスクにフラッシュされる間のインスタンスの待機時間を解消して、トランザクション・スループットを高めることです。

旧バージョンのエンジンでは、新しく入力または変更されたデータは、インスタンスによって管理されるバッファ・プールに格納されていました。変更の記録も、論理ログという一連のログに書き込まれていました。これは、削除を含むデータ変更を追跡するログです。データの論理的・物理的な一貫性を確保するために、それらのバッファ・プールの内容が定期的にディスクに書き込まれました。今では”同期チェックポイント”と呼ばれるそれらのバッファ・プールのフラッシュは、ディスクにフラッシュする必要があるデータの量と、インスタンスがどの程度十分に管理されているかによって、1 秒未満から数分までのさまざまな長さで完了します。このタイプのチェックポイントでは、特定のエンドユーザ・アクティビティが一時停止され、アプリケーションが応答待ちを強制されます。これによってユーザも待機を強制されるため、ユーザの業務遂行能力に影響が生じる可能性があります。

IDS Version 9.2 では、特定のタイプのトランザクション、特に INSERT 文、UPDATE 文、DELETE 文が、潜在的に“ファジーな”操作として再分類され、新しい“ファジー”・チェックポイントの使用によって処理できるようになりました。ファジー・チェックポイントでは、インスタンスはバッファ・プールをディスクにフラッシュしません。その代わりに、使用済みバッファのレコード、および使用済みバッファと論理ログに記録されたトランザクションとの関係が論理ログに書き込まれます。バッファをフラッシュするためにインスタンス処理を停止するのではなく、チェックポイント後に、トリクルフィード・アプローチによって、使用済みバッファが時間をかけて徐々にディスクに書き込まれます。ファジー・チェックポイントでは、エンドユーザ処理に対する割り込みは、変更の記録が論理ログに書き込まれるときにだけ発生します。これはたいてい完了するまで 1 秒もかからないので、エンドユーザ・アクティビティの中断は事実上なくなります。

ファジー・チェックポイントを使用する場合は、インスタンスと論理ログの両方を定期的にバックアップすることと、それらのバックアップの整合性を時折検証することが重要です。同期チェックポイントと異なり、ファジー・チェックポイントの場合はデータの完全な論理的・物理的一貫性が存在することが認識される瞬間がありません。それでもすべての変更が論理ログに書き込まれるので、環境を完全に復元できますから安心してください。ただ、インスタンスまたは DB 領域を完全に復元するために、最近使用された論理ログのほとんどを使って、復元プロセスの論理的部分をより広範に実行する必要があるだけです。



2.1.3 メモリ常駐表

インスタンスの共有メモリ・プール内に、新規データまたは変更されたデータのプールがあります。データが要求されると、インスタンスは最初にこのプールをチェックして、そのデータが別の操作のために既に抽出されていて、再利用可能かどうか調べます。データが見つかったら、プールにあるデータのコピーが使用されるので、ディスクからのデータの抽出にともなう待ち時間がなくなります。いずれかのデータが絶えず要求されない限り、より新しい操作のためにほかのデータを格納できるよう、古いデータが最終的にプールから削除されます。

このプールは、独立したデータを最新の操作のためにキャッシュするだけでなく、表、インデックス、またはフラグメントが、プールから削除されないように格納するためにも使用できます。これは表またはインデックスの“メモリ常駐”宣言と呼ばれ、表のコピー、インデックス、またはフラグメントがバッファ・プールに格納されます。それらはインスタンスがシャット・ダウンされるか、表／インデックスまたはフラグメントがデータベースから削除されるか、DBA が表／インデックスまたはフラグメントを“非常駐”として宣言するまで、バッファ・プールに残ります。

データベース・オブジェクトをメモリ常駐として宣言すると、そのオブジェクトからのデータの問合せに大きな影響が出る可能性があることは明らかです。問合せに応じるために、あまり使用されないデータがディスクから読み出されるまで待つのではなく、キャッシュされたオブジェクトの内容全体が即座に利用可能になり、一般にインスタンスのアップタイムが持続する限り使用可能です。

データベース・オブジェクトの常駐または非常駐としての宣言は、簡単な SQL 文で行われます。しかし、メモリ常駐データベース・オブジェクトの使用は慎重に検討する必要があります。常駐として宣言するオブジェクトのサイズによっては、インスタンスの共有メモリ・リソースをすぐに使い果たしてしまう可能性があります。

2.1.4 オプティマイザ・ディレクティブ

IDS には、あらゆるデータベース・エンジンと同様、ユーザ・セッションによって要求されたデータにアクセスする最も高速かつ低コストな方法を調べるための問合せオプティマイザを備えています。オプティマイザはありふれたテクノロジーではありません。オプティマイザが機能する方法を支配する高度な数学モデルおよび定理の作成を扱う研究分野があります。それらのモデルや定理は、ハードウェア・テクノロジーの変遷によってデータ・アクセスの“真の”コストが変化するのにもとない、適応し変化せざるを得ませんでした。その結果 IDS オプティマイザは、新しい数学動作モデルの出現にもとない継続的に改善されてきました。

IDS オプティマイザは、核心部分では“コストベース”の SQL オプティマイザとみなされていますが、それ以上に複雑なものです。エンジンのオブジェクト・リレーショナル・アーキテクチャにより、オプティマイザにも、純粋なリレーショナル・データベース・エンジンならば気にする必要のないユーザ定義のデータ型、関数、その他のテクノロジーを使って操作を正しく処理するオブジェクト指向オプティマイザ・テクノロジーが多く採用されています。

しかし、どのような状況であっても、現実のテストと経験に基づいた最善の問合せ計画をオプティマイザが算出しない可能性があります。IDS はエンジンによって導出された計画に限定されるのではなく、DBA がオプティマイザに対して特定文の処理方法に関する推奨を行うための手段を提供します。“オプティマイザ・ディレクティブ”と呼ばれるそれらのヒントには、“do this” および “don't do this” という 2 つの形式があります。ディレクティブにより、DBA は、特定のインデックス、結合テクノロジー、ハッシュ命令を使用する（または使用しない）こと、あるいは選択基準に一致するすべての行を取得するのか、最初の n 行を取得するのかをオプティマイザに指示できます。ディレクティブは、アプリケーション開発時の“what-if”分析において、動作環境でのパフォーマンスを高めるためにインデックスの作成または削除などのデータベース・モデルの変更が必要かどうかを判別するために使用できます。



使用されるアプリケーション言語によっては、3つの方法のいずれかによってヒントを SQL 文に埋め込むことができます。必要であれば、1つの文に複数のディレクティブを追加することもできます。どの方法を採用するかにかかわらず、オブティマイザによる文の処理方法は時とともに変化することを認識することが重要です。必要に応じて“UPDATE STATISTICS” コマンドが実行され、データ・サイズ、偏り変更、および新規統計情報がオブティマイザに供給されると、変化が起こります。そのため、オブティマイザ・ディレクティブを使用する問合せを定期的に見直し、そのディレクティブが依然として必要かどうか調べることが重要です。不適切なディレクティブを使用すると、マイナスの影響が生じる可能性があります。

オブティマイザ・ディレクティブの例については、セクション 2.2.11「非実行エクスペイン」を参照してください。

2.1.5 オブティマイザの機能拡張: 副問合せのフラット化

IDS オブティマイザは、ネストされたループ半結合について、“副問合せのフラット化”と呼ばれる機能をサポートします。これを説明する最善の方法は、例を使用することです。出荷先がテキサスである顧客について、すべての顧客注文を調べる必要があるとします。問合せの条件要素はおそらく、“注文の出荷先の州”の値が“texas”でなければならず、“注文の顧客番号”と“顧客の顧客番号”の間に相関が存在する必要がある、となるでしょう。

ほとんどのオブティマイザはこのタイプの要求を解析し、注文表全体を走査してテキサスの所在地の検出を試行した後、顧客番号を介してそれらの所在地を顧客関連情報に結合しようと試みます。一般に、注文表は顧客表よりはるかに大きいので、エンジンは最大の表を最初に走査することになります — これはもちろん、最も遅いデータ・アクセス方法です。

顧客表の“state”列にインデックスを追加し、オブティマイザによる副問合せのフラット化を行えば、問合せが再実行されたとき、オブティマイザは顧客表を解析し、まずテキサスの所在地を検索してから、表の中を1回通過するだけで注文情報を自動的に結合します。言うまでもなく、パフォーマンス特性は大幅に改善されます。



2.1.6 オプティマイザの機能拡張: キー優先インデックス走査

キー優先インデックス走査の概念は、ディスクにアクセスして行を読み込み、データが問合せ条件を見たすかどうか実際に調べる必要が生じる前に、可能性のあるデータの不一致を最も低コストな方法で除外するというものです。そのためには、エンジンが表に存在するすべてのインデックスにデータ条件（問合せの“WHERE”節）を適用することで、行をディスクから読み込む前に、可能性のある一致を検索します。通常、インデックスの走査はデータの読み込みよりもはるかに高速で行われるため、問合せのパフォーマンスが改善します。

2.1.7 エンタープライズ・レプリケーションの機能拡張

IDS Version 9.30 ではエンタープライズ・レプリケーション（ER）の機能が大幅に拡張され、パフォーマンスも改善されています。特に注目すべき点は次のとおりです。

- レプリケート間の並列処理およびソートを行わないコミットのサポートなどの、データ同期の改善。9.30 ベータ・プログラムの顧客は、各社での配備において 2～3 倍に改善されたと報告しています。
- 論理ログ消費量の削減
- IDS Version 7.31 および Version 9.2x 上で動作する ER との相互運用性
- シリアル列主キーのサポート（標準とシリアル 8 タイプの両方をサポート）
- スマート LOB のサポート
- DISTINCT データ型と OPAQUE データ型のサポート（ただし複合データ型はサポート外）
- ほとんどの場合インプレース変更表の操作における衝突解消に使用される、シャドウ列の自動追加／削除
- レプリケートのコレクションを管理することで“グループ”機能に取って代わるレプリケート・セット。この機能の拡張である排他レプリケート・セットは、レプリケート対象のトランザクションがターゲットにおいてソースと同じ順序で適用されることを保証して、時間ベースのレプリケーションにおける参照整合性を確保します。



IDS Version 9.30 の ER は、ルート・サーバ、ハブ・サーバ、およびリーフ・サーバの定義によって階層レプリケーション (HR) シナリオを構築できるようになりました。それらのシナリオは必要に応じて、“ツリーの有向グラフ” または “階層ツリー” という 2 つの形式のどちらかになります。旧バージョンのエンジンでは、ER シナリオに参加するすべての IDS インスタンスが、シナリオ内のすべてのサーバに直接接続されている必要がありました。このオプションは依然として使用できますが、管理しにくい複雑なレプリケーション・ネットワークが生じる可能性があります。地理的に大きく分散した環境では特にそうです。

2.2 SQL に関する機能

2.2.1 ロング識別子

ユーザから最もよく寄せられる要求の 1 つに、データベース・オブジェクト名 (表名、インデックス名など) は 18 文字以下にする必要があるのに、ユーザ識別子の長さが 8 文字に制限される “8/18” ルールの解消がありました。IDS Version 9.30 はこの制限に対して、大いに必要とされていた救済策を備えています。

IDS Version 9.30 では、ユーザ識別子の長さを最大 32 文字にできます。データベース・オブジェクト名は最大 128 文字にできます。この長い命名スキームの恩恵を受けるオブジェクトとしては、表名、サーバ名、データベース名、列名、インデックス名、シノニム名、制約名、プロシジャ名、DB 領域名、BLOB 領域名、オブティカルクラス名、トリガ名、タイプ名、ルーチン言語名、アクセス方法名、演算子クラス名、トレース・メッセージ・クラス名、トレース・メッセージ名、プロシジャ変数名などがあります。

これらのロング識別子 (ユーザとオブジェクトの両方) により、より直観的な命名スキームを使用できるだけでなく、ほかのデータベース環境から IDS への移行が容易になります。



2.2.2 “選択時” トリガ

旧バージョンの IDS では、“DELETE”、“INSERT”、または “UPDATE” の SQL 操作に関して動作するトリガしか作成できませんでした。IDS Version 9.30 では、SQL の “SELECT” イベントとともに実行されるトリガを作成できます。この機能により、DBA およびエンジンの管理者は、監査機能またはセキュリティ機能、クリック・ストリーム分析などを作成するという新しい選択肢を得ることになります。

2.2.3 SQL の機能拡張: “RENAME INDEX” 文

“RENAME INDEX” 文により、データベース内にある既存のインデックスの名前を簡単に変更できます。

IDS Version 9.30 以前は、インデックス名を変更するためにインデックスを削除して再作成する必要があり、それには表に対する排他的アクセス権と、場合によっては長い構築時間が必要でした。

この SQL 文の機能拡張によって、インデックスを再作成することなくシステム表内のインデックス名を変更できます。

2.2.4 SQL の機能拡張: ANSI 外部結合構文

この SQL の機能拡張は、複雑な外部結合問合せのパフォーマンスを大幅に向上させます。IBM は IDS Version 9.30 において、外部結合の米国規格協会 (ANSI) 構文標準を実装しました。サードパーティのベンチマークでは、IDS Version 9.30 の外部結合問合せのパフォーマンスは競合データベース・システムで実行される同等の問合せを大きく上回っています。

この機能により、パフォーマンスが改善されるだけでなく、ほかのデータベース・システムから IDS へのアプリケーション移行が容易になります。

2.2.5 更新ロックの保持

旧バージョンの IDS では、データ・ロックは問合せ排他レベルや実行される操作タイプ (カーソルベースの読み込みまたは更新など) などのいくつかの環境条件に基づいて配置および解放されます。一部の排他レベルでは、カーソルが 1 つのデータ要素から別のデータ要素に移動するとデータのロックが解除されます — これはもちろん、ロック中にデータ自体が変更されていないという前提に立っています。また、コミットされたデータだけが読み込まれ、ロックされるレベルもあります。ロックが配置されず、コミットされていてもいなくてもすべての行が読み込まれるレベルもあります。



IDS Version 9.30 には新しい SQL コマンド “RETAIN UPDATE LOCKS” があります。このコマンドは、排他レベルにかかわらず、潜在的な更新のために読み込まれたすべての行を、トランザクションが閉じられるまでロックします。これによって、更新セッションによってロック解除されたがコミットされていない更新済みの行が、ほかのセッションによってロックされることを防止します。この機能を使用すると、ダミー更新や “REPEATABLE READ” 排他レベルの (不要な) 使用などの非効率的なコーディング・テクニックを排除できます。

2.2.6 SQL 関数

IDS Version 9.30 は、大文字 / 小文字 / 混合の変換、サブ文字列関数、“CASE” 文などのタスクを実行する新しい SQL 関数を備えています。多くの場合、これらの新しい組み込み関数を自家製のストアド・プロシジャの代わりに使用して、パフォーマンスを改善することができます。

新しい関数の一部を次に示します。

- “NVL” — 指定された値に NULL をマッピングします。
- “CASE” 文 — SQL-92 条件式のサポート
- “DECODE” — IDS Version 9.30 は、“CASE” 文と同様の操作の “DECODE” 構文構造をサポートします。
- 各種データ型間の文字列 / 日付 / データ型変換
- UNION ビュー — “CREATE VIEW” 文に “UNION” 構文が組み込まれた “ビュー” を作成します。
- “DESCRIBE” SQL キーワードからの出力の更新。“DESCRIBE” キーワードが使用される SQL 文のタイプによっては、エンジンはその SQL 文に関する情報によって応答します。この文が実行された場合に影響を受ける列のデータ型、影響を受ける可能性のある潜在的な行数、および “WHERE” 節の内容などの情報が使用されます。

2.2.7 ログ表

“ログ” 表は長い間 IBM XPS 製品の機能でした。IDS の “標準” 表と異なり、ログ表はログ付きデータベース内に存在できる、ログなしの永続表です。ログ表は元来データウェアハウス環境での大量データ・ロードをサポートするためのものであり、“簡単な追加” によって表のフラグメントの末尾にデータを追加します。ログ表内のデータの更新操作と削除操作もサポートされますが、いずれの場合にも、トランザクション情報は論理ログ内に取得されません。



必要に応じて、“ALTER TABLE” コマンドにより標準表をロウ・モードに、またはその逆に変換できます。標準モードからロウ・モードに変換する場合は、その前に表のインデックスを削除する必要があります。

ロウ表の変更は論理ログで追跡されないため、復旧メカニズムがないことに注意してください。表を変換して標準モードに戻したら、DB 領域のバックアップを作成してデータ変更を取得する必要があります。

2.2.8 インプレース表変更

“インプレース表変更” アルゴリズムの機能が拡張され、ほぼすべての “ALTER TABLE” 操作が瞬時に処理されるようになるとともに、変更する表ごとに 2 倍のディスク容量を確保する必要がなくなりました。そのため、必要なディスク容量が削減されるとともに、コマンドを高速で実行でき、動作環境システムに対する表変更の適用がはるかに容易になりました。

2.2.9 フラグメントの関連付け／切離し機能拡張

IDS Version 9.30 では表に対するフラグメントの関連付け機能および切離し機能が最適化されています。関連付け操作の中で既存のインデックスがより効率的に使用されるので、全フラグメントの再構築にともなうコストが解消されます。

2.2.10 DELETE 文における “FROM” の最適化

IDS Version 9.30 では、“FROM” キーワードは “DELETE” SQL 文でのオプションです。したがって、構文という点では次のどちらの文も正しいことになります。

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

この機能により、アプリケーションを別のデータベース環境から IDS に移行させる作業がはるかに簡単になります。



2.2.11 非実行エクスプレイン

長い間ユーザから求められていたもう 1 つの機能は、SQL 文を実際に行わずに最適化計画情報を調べることができる機能でした。これは、UPDATE 文または DELETE 文のレビューおよび評価を試行する場合に特に重要でした。IDS Version 9.30 には、PREPARE 段階が完了してその計画情報がファイルに出力された後、文の処理を停止するよう最適化に命令する新機能があります。IDS オプティマイザは IBM XPS と異なり、現在の作業ディレクトリ内の “sqxplain.out” というファイルに最適化計画出力を格納します。

この機能は任意のセッションの中で必要に応じてオン/オフを切り替えることができ、それには 2 つの方法があります — オプティマイザ・ディレクティブまたは SQL 文経由です。どちらの場合も、新しい “AVOID_EXECUTE” キーワードを使って最適化を PREPARE 段階で停止する必要があります。次に例を挙げます。

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

オプティマイザ・ディレクティブを使用:

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

2.2.12 構成可能なデフォルト・ロック・モード

旧バージョンの IDS では、新規作成された表のデフォルト・ロック・モードはページ・レベルでした。データを 1 行だけロックする必要がある場合でも、ページ全体がロックされました。これは同時実行での問題につながり、IDS の後続バージョンでは、表の作成時にロック・モード (ページまたは行) を指定する機能によって部分的に解決されました。IDS Version 9.30 は、新規作成される表について、エンジンのデフォルトの表ロック・モードを上書きする手段を備えています。

デフォルト・ロック・モードは、エンジン構成ファイル・パラメータ (DEF_TABLE_LOCKMODE) または環境変数 (IFX_DEF_TABLE_LOCKMODE) を使って、インスタンス全体またはセッション別に割り当て、変更することができます。どちらの場合でも、ユーザは “CREATE TABLE” 文に特定のロック命令 (“LOCK MODE ROW/PAGE”) を入れることで、デフォルトを上書きできます。この機能は新規作成される表だけに適用されることに注意してください。既存の表には効果がありません。既存の表のロック・モードは、“ALTER TABLE” コマンドで変更できます。



2.2.13 ユーザとしての Revoke

“GRANT” 文および “REVOKE” 文を使用すると、データベース・オブジェクトの所有者はそのオブジェクトに対する権限をほかのユーザに割り当てること、その権限を取り消すことができます。旧バージョンのエンジンでは次の構文がサポートされていました。

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

IDS Version 9.30 では “REVOKE” コマンドと “REVOKE FRAGMENT” コマンドに新しい構文と機能が追加され、<user2> が次のコマンドによって <user1> の権限を取り消すことができます。

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

2.3 DBA の機能 — ユーティリティ

2.3.1 動的なロック割り当て

旧バージョンの IDS では、データベース・ロックは調整可能な固定コモディティでした。ロックごとにインスタンスの共有メモリの X KB が使用され、エンジン管理者は構成されている以上のロックを必要とする文が実行されないよう努めていました。それらの文が実行されると、懸念されている “ロック表オーバーフロー” イベントが発生し、データが破損する可能性があります。IDS Version 9.30 ではロック処理が大幅に変更されています。

既知のスマート LOB と、サイズまたは内容の不明な各種のユーザ定義データ型を新規作成する機能により、IDS はロックの有限セットでは機能しなくなりました。また、エンジンはたとえば スマート LOB へのシングルスレッド・アクセスではなく、データベース内のほとんどのオブジェクトに対する並行アクセスを提供できる必要があります。これを実現するため、IDS Version 9.30 ではオブジェクトのロックおよび LOCKS 構成ファイル・パラメータの動作に対するインスタンスのアプローチ方法が変更されています。

LOCKS パラメータはインスタンス内で初期ロック割り当てとなり、ワーストケース・シナリオではなく、使用されると予想される妥当なロック数に設定する必要があります。これによってより多くの共有メモリが解放され、インスタンスのほかの部分で使用できるようになります。インスタンス自体がロック数を管理し、必要に応じてロック数を増減します。それらの追加ロックはインスタンスの共有メモリの仮想部分で管理されます。



事前のロック割り当てが十分でない場合、インスタンスは最大 16 の追加ロック割り当ての作成を試行します。このプロセスは、個別のセッションが利用可能ロックの 10 %を超えてロックするか、スマート LOB の 50~70 % をロックする場合は、そのオブジェクトに対して単一の排他ロックが設定されるという方法で管理されます。

2.3.2 動的論理ログ

旧バージョンの IDS では、論理ログを管理するためにインスタンスの作業負荷のサポートに必要となる可能性のあるログのサイズと数を見積もる必要がありました。この見積りが間違っていると、変更を行うためにインスタンスをシャット・ダウンする必要がありました。IDS Version 9.30 では、エンジン管理者はインスタンスの処理を中断せずに新規論理ログを自動的に作成、挿入および有効化するようにインスタンスを構成できます。これによって、懸念されているロングランザクシオンのイベントが解消されるものの、この機能を有効化することでイベントのペナルティ、特にインスタンス起動時のペナルティの大部分が排除されます。

この機能は構成可能であるため、管理者は新規ログが必要な場合に 3 つの制御レベルのいずれかをインスタンスに付与できます。管理者は論理ログを追加できるだけでなく、インスタンスの処理を中断することなく、必要のなくなった論理ログを削除することもできます。

インスタンスの新規ログ追加機能によって、ログを定期的にテープにバックアップする必要がなくなるわけではありません。依然として、ログをインスタンス内で再利用できるよう解放するためにバックアップする必要があります。バックアップされた論理ログは、インスタンスの復旧、特にメディア障害が発生した場合の復旧に必要となります。

2.3.3 Archecker

Archecker は、数年にわたって International Informix Users Group (IIUG) Web サイトから入手可能なユーティリティです。当初は Informix 社内の開発者専用でしたが、現在は IDS Version 9.30 にバンドルされています。Archecker は、実際に復元を実行せずに、インスタンス・バックアップ・メディア (テープやディスクなど) からデータを復旧できるかどうかチェックするために使用されます。賢明な管理者ならば、復旧が可能かどうかバックアップ・ジョブの整合性を定期的に検査すべきであることは言うまでもありません。



Archecker は “OnBar” ユーティリティ (“onbar -v”) に統合されていますが、“ontape” ユーティリティで作成されたバックアップの整合性チェックに使用できます。Archecker による検証は、バックアップが生成されたのと同じマシン上でも、別のマシン上でも実行できます。実行に必要なディスク容量はわずか数メガバイトです。

2.3.4 Onmsync

“onmsync” ユーティリティは、OnBar データベース (“sysutils”)、緊急ブート・ファイル (“ixbar.servernum”) と、格納域マネージャ (ISM、Legato、Omniback、IBM Tivoli Storage Manager など) によって管理される情報を同期化します。

旧バージョンの IDS では、それら 3 つの要素を管理者が個別に管理する必要がありました。たとえば、格納域マネージャによって管理されているバックアップ・オブジェクトが期限切れとなって削除された場合、管理者は “sysutils” データベースと緊急ブート・ファイルに適切な変更を加える必要がありました。ほとんどの場合はこれが行われず、情報が欠けているために復元操作を実行できないという結果につながりました。ほかのケースでは、緊急ブート・ファイルの増大を放置した場合、システムが古いエントリを解析するので復元操作の完了にかかる時間が長くなりました。このユーティリティを使用すると、OnBar 関連のバックアップ情報の管理が容易になります。

2.3.5 High-Performance Loader (HPL) のコマンド・ライン・インターフェイス

IDS Version 9.30 以前には、管理者は High-Performance Loader (HPL) ジョブの作成と実行にグラフィカルベースのツールを使用する必要がありました。このツールは世界最悪のインターフェイスというわけではありませんが、最高でなかったことは確かです。管理者は HPL への “onpladm” コマンド・ライン・インターフェイスを利用できるようになりました。

管理者はこのコマンド・ライン・インターフェイスによりシェル・スクリプトを介して動的に HPL ジョブを生成できるため、HPL の柔軟性が高まり、HPL が既存のワークフローに統合されるようになりました。



2.3.6 ロックなしの Oncheck

セクション 2.3.1「動的なロック割り当て」で簡単に説明したように、IDS Version 9.30 のロック・メカニズムは旧バージョンのエンジンから変更されています。上記のセクションで説明した利点に加え、管理者は表への排他アクセスがなくてもインデックス一貫性検査 (“oncheck -ci /-cl”) を実行できるようになりました。しかし、このタイプの検査には注意点が 1 つあります — “oncheck” コマンドでは、そのユーティリティによって走査された後に変更された可能性のあるインデックス・ページが再評価されません。

この新機能により、管理者は管理期間にインスタンスをオフラインにするまで待つのではなく、1 日または 1 週間のうち処理の少ない期間に定期的なデータ一貫性検査を実行できるようになりました。

2.3.7 Informix 格納域マネージャ (ISM) のインターフェイス変更

IDS にはかねてより、基本機能を持つ格納域マネージャが製品の一部として付属していました。Informix 格納域マネージャ (ISM) を “OnBar” ユーティリティとともに使用して、最大 4 つの物理デバイスへのシリアル・バックアップまたは並列化バックアップを作成できます。旧バージョンのエンジンでは、ISM はグラフィカル・インターフェイスまたはコマンド・ライン・インターフェイスを介して構成または使用できました。IDS Version 9.30 では、コマンド・ライン・インターフェイスのみを使って ISM にアクセスし、ISM を使用することができます。

2.3.8 外部バックアップおよび復元 (EBR)

標準のインスタンス・バックアップおよび復旧ユーティリティ (つまり、“onbar” および “ontape”) では、データを選択してバックアップ・ユーティリティに渡すプロセスはインスタンス自体によって管理されます。これは実際に機能しますが、大きなインスタンスのバックアップを作成するには時間がかかります。ハードウェア、特に格納域のコストが急落しているため、現在ではほとんどのインスタンスが少なくともハードウェア・ベースのミラーリングによってデータ損失から保護されています。それらのミラーの存在を活用して、高速バックアップを作成できます。また、機能拡張によって復元することもできます。



“外部” バックアップの概念はかなり単純です。ハードウェア・ベースのミラーリング環境では、データの正確なコピーが少なくとも 1 つディスク上に存在します。1 つのディスク・コピーと残りのコピーとの関連付けを一時的に解除することで、データの瞬間ビューが得られます。その後、“onbar” ユーティリティを介して、レベル 0 バックアップと同様にこのスナップショットからデータを格納域マネージャにプッシュできます。このバックアップ操作時にはインスタンスは影響を受けません。走査および取得の操作を管理する必要がないからです。最終的な結果として、インスタンスにほとんど、またはまったく影響を及ぼさずにバックアップ操作が短時間で完了します。その後、ディスクのセットをミラー・セットに再度関連付けることができ、ハードウェア・ベースのミラーリング・メカニズムによってディスクの再同期化が処理されて、すべてのディスクが再び一致します。

外部復元操作は、バックアップが並列で作成されていれば並列にすることもできますが、“ontape” の復元に似ています。インスタンスはディスク・コピーが切り離された時点を認識するので、物理的な復元が完了した後、欠けているトランザクションをロール・フォワードするために要求すべき論理ログを認識します。

EBR を使用すると、非常に大きなデータベース（数百ギガバイト以上）をより簡単かつ定期的にバックアップでき、必要な場合ははるかに短い時間枠で復元できます。

2.3.9 再開可能復元も 7.3 からの機能

インスタンス内に構成されたエンジンの新しい機能である、再開可能復元機能は、失敗した OnBar 復元操作を、先頭からではなく失敗した瞬間から再開し、既に復元された格納域に再度書き込むことができます。この機能をサポートするオーバーヘッドは `sysutils` データベースに格納される事です。このデータベースは、該当箇所まで前方スクロールしてメディアの再読み込みを開始するよう、関連付けられた格納域マネージャに命令します。

この機能は、`RESTARTABLE_RESTORE` 構成パラメータによって、および “onbar” コマンドで “-restart” フラグを使用することで制御されます。



2.3.10 アラームの新しいスクリプト `ex_alarm.sh`

IDS はいくつかのバージョンにわたって、完全論理ログ、ディスク障害などの“アラーム・イベント”を管理者がキャプチャし、そのイベントに応じて特定のルーチンをエンジンに自動的に実行させることができるインターフェイスを提供してきました。これは一般にその構成パラメータ名 (ALARMPROGRAM) で呼ばれ、旧バージョンの IDS は、例として 2 つのアラーム・スクリプトを提供していました — “log_full.sh” と “no_log.sh” です。これらは完全論理ログを自動的にバックアップするかしないかのどちらかでした。

IDS Version 9.30 では、UNIX[®] システム上で “ex_alarm.sh” スクリプトを使って、完全論理ログの自動バックアップなどのイベント・アラームを処理できます。この方法では、1 つのスクリプトを介してすべてのイベント・アラームを処理できるので、管理者のオーバーヘッドを最小限に抑えることができます。

2.3.11 UNIX Bundle Installer

The UNIX Bundle Installer は、複数のエンジン・コンポーネント (コネクティビティ、DataBlade など) をインストールする際の複雑さを解消し、コンポーネントが正しい順序でインストールされることを保証します。この Installer により、デモンストレーション・データベース・インスタンスを手早く作成することもできます。

3 MaxConnect

3.1 MaxConnect の説明

MaxConnect は、データベース・インスタンスとそのクライアントの間に常駐する、別売のソフトウェアです。クライアントはインスタンスに接続するのではなく、MaxConnect インスタンスに接続し、MaxConnect インスタンスはインスタンスへの接続を多重化します。たとえば、パフォーマンスを一切損なうことなく、500 のクライアント接続を少数のインスタンス接続に多重化できます。最終的な効果として、クライアントのコネクティビティと通信をサポートするために必要なインスタンス・リソースが削減されます。クライアントとのデータ送受信に通常使用される CPU VP サイクルを使って、代わりに SQL 操作を処理できるようになりました。

MaxConnect の最も重要な機能は、アプリケーションの透過性です。MaxConnect のインストール方法および構成方法によっては、MaxConnect を使用するためにクライアント側で必要となる変更はほとんど、あるいはまったくありません。MaxConnect を使用するためにアプリケーション自体を変更または再コンパイルする必要はありません。その結果、多数のユーザが IDS インスタンスにアクセスしているような環境にも、MaxConnect を簡単に挿入できます。

3.2 MaxConnect の構成オプション

図 1 および 2 に示すように、MaxConnect は 2 層または 3 層の環境にインストールできます。

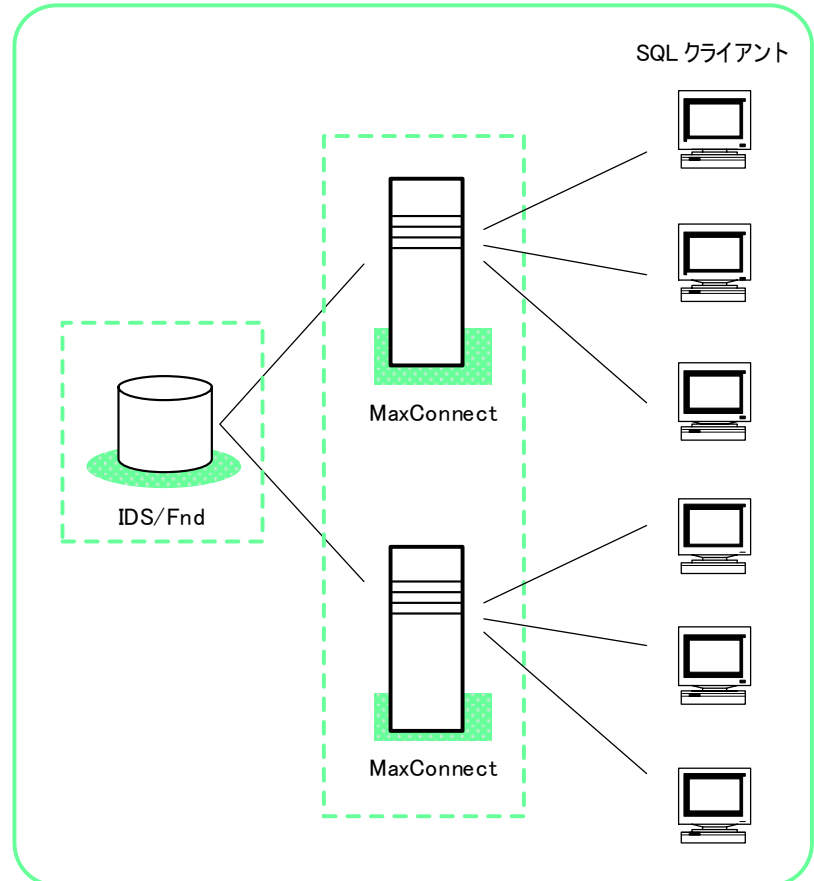


図 1.3 層環境で別個のサーバにインストールされた MaxConnect

ハードウェア区画化の可能な大規模 SMP システムの場合:

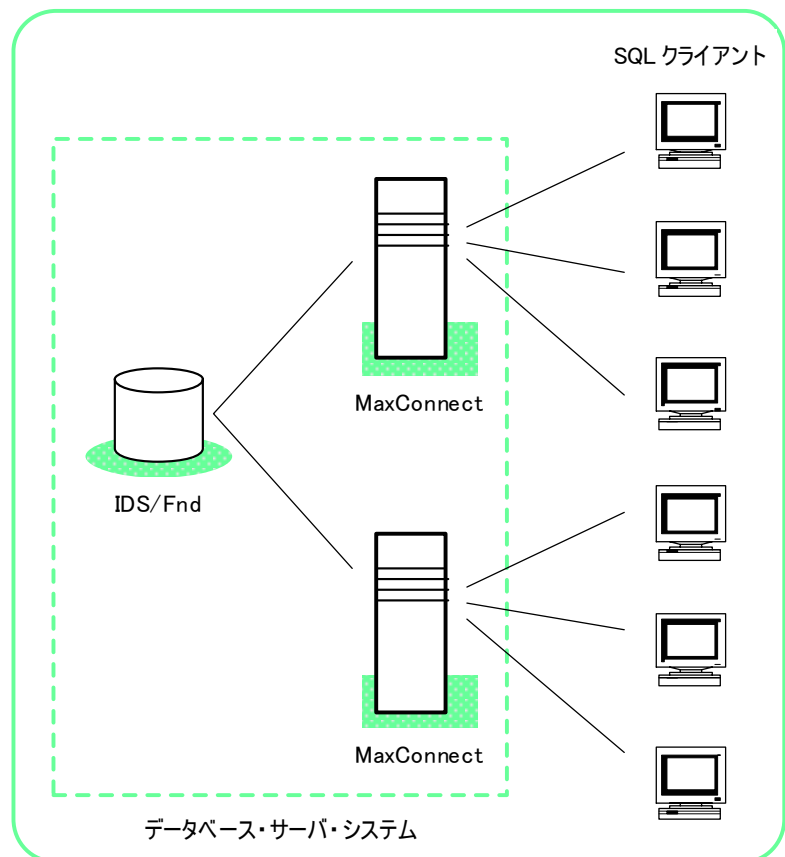


図 2.2 層環境においてハードウェア区画化の可能なサーバ上にインストールされた MaxConnect

通常は、データベース・インスタンスをサポートするサーバ上のネットワーク・トラフィックを削減するため、MaxConnect が別個のマシンにインストールされます (3 層)。複数のネットワーク・インスタンス・カードをサポートできる、ハードウェア区画化の可能な大規模マシンを使用しており、O/S がネットワーク・トラフィックを処理できる場合は、インスタンスをサポートするものと同じサーバ上に MaxConnect をインストールできます。一般に、2 層実装でのパフォーマンス改善は 3 層の場合ほど劇的ではありません。

MaxConnectを使用するには、NETTYPEパラメータおよび INFORMIXSERVERALIASパラメータと、データベース・インスタンス上のSQLHOSTSファイルを若干変更する必要があります。MaxConnect インスタンスには非常に簡単な構成ファイルだけでなく、独自の SQLHOSTS ファイルもあります。

MaxConnect がインストールおよび構成された後、MaxConnect を使用するためにクライアント側でわずかな変更を行う必要があります。2 層配備では、変更は必要ありません。適切に構成された 3 層環境では、クライアントの SQLHOSTS ファイル内のホスト名を変更する必要があるだけです。他のすべての通信パラメータはそのままです。

4 IDS のベンチマーク結果

4.1 MaxConnect を使用しない場合

このホワイト・ペーパーの冒頭で述べたように、IDS 9.30 は必要な修復や修正がほとんど組み込まれた 9.21 からの“重大な”アップグレードです。少し前に主要 ERP ベンダーについてのベンチマークが実行され、IDS Version 9.21 でのパフォーマンスの改善が示されるとともに、MaxConnect の有効性が強調されました。このペーパーでは 9.30 に焦点を当ててきましたが、IDS 9 製品ファミリー全体のパフォーマンスの特長に関して安心感を持っていただくために、この 9.21 のパフォーマンス・ベンチマークについても述べた方が適切と思われる。

1 台の HP K570 (6 CPU) データベース・サーバと 5 台の HP K580 (各 6 CPU) アプリケーション・サーバを使って、次の結果が記録されました。

- IDS 9.21 は IDS7.3 に比べ 24% 多くのユーザをサポートできた。
- システムのメモリ使用率を 22% 削減できた。
- CPU 使用率を 16% 削減できた。

詳細な結果を次に示します。

	IDS 9.21	IDS 7.3	改善率
スループット			
最大ユーザ数 (6 CPU)	4,336	3,485	24% 改善
トランザクション数/分	12,816	10,784	18.7% 改善
ユーザ/CPU	720	580	24% 改善
3,375 ユーザでの結果			
応答時間	~1.4 秒	~1.55 秒	~6% 高速化
ユーザ CPU %	47%	56%	16%削減
メモリ/ユーザ	4.47MB	5.8MB	22% 削減



4.2 MaxConnect を使用した場合

HP N クラスマシン (8 CPU) および MaxConnect で BAAN ベンチマークを実行したとき、データベース・インスタンスをサポートするサーバに CPU を追加することで、ほぼリニアなスケーラビリティが存在しました。

BAAN ベンチマークの実行時に、IBM Informix は参加したすべてのデータベース・ベンダの中で最高の結果を出しました。IDS のベンチマークは、インスタンスへの 18,650 の同時接続をサポートしながら 11,445 BRU (Baan Reference Unit) に達することができました。この結果は、同じハードウェア上の Oracle を21% 上回りました。

5 IDS の概要

5.1 機能とバージョン

5.1.1 パフォーマンスに関する機能

機能	7.2x	7.30	7.31	9.21	9.30
共有ステートメント・キャッシュ				×	×
ファジー・チェックポイント				×	×
メモリ常駐表		×	×	×	×
オブティマイザ・ヒント		×	×	×	×
オブティマイザ機能拡張: 副問合せのフラット化		×	×	×	×
オブティマイザ機能拡張: キー優先インデックス走査			×	×	×
オブティマイザおよび更新統計の改善					×
エンタープライズ・レプリケーションのパフォーマンス改善					×



5.1.2 SQL に関する機能

機能	7.2x	7.30	7.31	9.21	9.30
ロング識別子				×	×
select 文トリガ				×	×
SQL 機能拡張: “RENAME INDEX” 文			×	×	×
SQL 機能拡張:ANSI 外部結合			×	×	×
更新ロックの保持			×	×	×
SQL 関数 (CASE、UPPER 等)		×	×	×	×
ログなしの表			×	×	×
インプレース表変更		×	×	×	×
フラグメント拡張機能の関連付け/切離し			×	×	×
DELETE 文 “FROM” のオプション化					×
問合せを実行せずに問合せ計画を表示					×
ユーザとしての Revoke					×
構成可能なデフォルト・ロック・モード					×



5.1.3 DBA の機能 — ユーティリティ

機能	7.2x	7.30	7.31	9.21	9.30
動的ロック・マネージャ				×	×
動的ログ					×
Archecker			×	×	×
Onsmsync			×	×	×
階層型レプリケーション				×	×
ハイパフォーマンス・ローダ用コマンド・ライン・インターフェイス				×	×
ロックなしの Oncheck		×	×	×	×
ISM Informix 格納域マネージャ		×	×	×	×
EBR 外部バックアップ復元		×	×	×	×
再開可能復元		×	×	×	×
MaxConnect のサポート			×	×	×
新しい ex_alarm.sh スクリプト					×
UNIX Bundle Installer					×
AGS ServerStudio JE を IDS にバンドル					×
ER により任意更新でシリアル・主キーを使用可能					×

5.1.4 その他の機能

IDS は業界をリードする拡張可能データベースです。ビジネス・ロジックは、サーバにおいてユーザ定義関数の形で、およびストアド・プロシジャとして実行できます。ユーザ定義関数は SPL、C、または Java で記述でき、たとえば複数の SQL 文をカプセル化するため、および既存の SQL 関数を拡張してビジネス固有の操作を実行するために使用できます。それらの操作をデータに近づけることで、ネットワーク・トラフィックを削減し、データベースのパフォーマンスを改善するとともに、アプリケーション開発を簡素化し、処理の一貫性を高めることができます。



この文書で説明していないものの、一部のアプリケーションに対して重要となる可能性のある、その他の IDS 機能のリストを次に示します。

機能	7.2x	7.30	7.31	9.21	9.30
Onbar 進捗フィードバック				×	×
Onbar CLI クリーンアップ				×	×
並列復旧				×	×
ロウ・タイプにおけるシリアル・タイプ				×	×
引用符で囲まれた文字列における埋込み改行				×	×
コンパイルされた式				×	×
Microsoft [®] Windows NT [®] 上でのオンライン・オプティカル		×	×	×	×
NT 上でのロウ・デバイスのサポート		×	×	×	×
NT での Wolfpack のサポート		×	×	×	×
XA における Microsoft Transaction Server のサポート				×	×
再帰的更新トリガ				×	×
JVM 1.2 のサポート				×	×
JDBC 2.0 機能セット				×	×
JVP の動的削除				×	×
UDR スレッドの最適化				×	×
R ツリー・ビルドのソート関数およびダイレクト関数の強化				×	×
R ツリーによる直近ネイバ問合せ					×
境界ボックスのみの R ツリー・インデックス					×
R ツリーでの Oncheck/onlog				×	×
可変長 UDT				×	×
C++ フック				×	×
一時スマート BLOB					×
スマート BLOB 領域管理の簡素化					×
スマート BLOB、UDT、スペーシャル・データについてのエンタープライズ・レプリケーションのサポート					×
JVM 1.3 のサポート					×
SAPI インターフェイスの拡張					×



5.2 IDS と他の Informix 製品との互換性

5.2.1 IDS 向けの認定 IBM Informix DataBlade モジュール

次の IBM Informix DataBlade モジュールおよび DataBlade ユーティリティは IDS 向けに認定されています。

- 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 以上
- 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 以上



5.2.2 IDS 向け認定製品

次の製品は IDS 向けに認定されています。

- 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 以上
- Informix Enterprise Gateway Manager 7.2x
- Informix Enterprise Gateway with DRDA® 7.2x 以上
- MaxConnect 1.00
- Office Connect 2.00
- Object Translator 2.00
- Server Studio JE 2.0.JC1 (Object Explorer, SQL Editor および Table Editor を含む)



© 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, e-ビジネス・ロゴ, IBM₁, IBM ロゴ, Informix, Lotus, MetaCube, Tivoli, および WebSphere は、米国または他国、あるいはその両方における International Business Machines Corporation の商標または登録商標です。

Microsoft, Windows, および Windows NT は、米国または他国、あるいはその両方における Microsoft Corporation の登録商標です。

UNIX は、米国または他国、あるいはその両方における The Open Group の登録商標です。

Java および Java ベースのすべての商標は、米国または他国、あるいはその両方における Sun Microsystems, Inc. の商標です。

本出版物において IBM の製品またはサービスに言及していても、それは IBM が営業しているすべての国でそれらを提供する意図を示すものではありません。提供物は予告なしに変更、拡張、または撤回されることがあります。



回収された使用済みファイバーを 10% 含む再生紙に
米国内にて印刷

