SYBASE®

パフォーマンス＆チューニング・ガイド

Sybase® IQ

12.7

ドキュメント ID：DC00283-01-1270-01
改訂：2006 年 6 月

このマニュアルは Sybase ソフトウェアの付属マニュアルであり、新しいマニュアルまたはテクニカル・ノートで特に示されないかぎりは、後続のリリースにも付属します。このマニュアルの内容は、予告なく変更されることがありますが、Sybase, Inc. およびその関連会社では内容の変更に関して一切の責任を負いません。このマニュアルに記載されているソフトウェアはライセンス契約に基づいて提供されるものであり、無断で使用することはできません。

マニュアルの注文

マニュアルの注文を承ります。ご希望の方は、サイベース株式会社営業部または代理店までご連絡ください。マニュアルの変更は、弊社の定期的なソフトウェア・リリース時にのみ提供されます。このマニュアルの内容を弊社の書面による事前許可を得ずに電子的、機械的、手作業、光学的、またはその他のいかなる手段によっても複製、転載、翻訳することを禁じます。

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目次

はじめに

このマニュアルの内容

このマニュアルでは、パフォーマンスとチューニングの推奨事項について説明します。

対象読者

このマニュアルは、パフォーマンス上の問題を理解する必要があるシステム管理者とデータベース管理者を対象としています。リレーショナル・データベース・システムの基礎知識と、Sybase IQ のユーザ・レベルの基本的な経験があることを前提にしています。このマニュアルは、他のマニュアルとともに使用してください。

このマニュアルの使用方法

次のリストは、行う作業や必要性に応じてどの章を参照すべきかを示します。

- SELECT 文の構築については、「第 1 章 データベース・テーブルからのデータの選択」を参照してください。
- ジョイン条件については、「第 2 章 テーブルのジョイン」を参照してください。
- クエリの最適化については、「第 3 章 クエリと削除の最適化」を参照してください。
- メモリ、ディスク I/O、CPU の調整については、「第 5 章 システム・リソースの管理」を参照してください。
- パフォーマンスについては、「第 6 章 パフォーマンスのモニタリングとチューニング」を参照してください。
- Windows パフォーマンスについては、「第 7 章 Windows システムでのサーバのチューニング」を参照してください。

関連マニュアル

Sybase IQ には次のマニュアルが用意されています。

- 『Sybase IQ の概要』－ Sybase IQ と Sybase Central™ データベース管理ツールに慣れていないユーザのための説明と練習が記載されています。
- 『Sybase IQ 12.7 の新機能』－ Sybase IQ の新機能の概略を説明しています。
- 『Sybase IQ パフォーマンス＆チューニング・ガイド』－ 巨大なデータベースのクエリ最適化、設計、チューニングについて説明しています。
- 『Sybase IQ システム管理ガイド』－ Sybase IQ がサポートする、管理面の概念と手順および最適なパフォーマンスのチューニングについて説明しています。IQ ストアの管理方法についても説明しています。

- 『Sybase IQ トラブルシューティングおよびリカバリ・ガイド』－ 問題の解決方法、システム・リカバリの実行方法、データベースの修復方法を紹介しています。
- 『Sybase IQ エラー・メッセージ』－ Sybase IQ エラー・メッセージ (SQLCode、SQLState、Sybase エラー・コードによって参照)、および SQL プリプロセッサのエラーと警告を示します。
- 『Sybase IQ ユーティリティ・ガイド』－ Sybase IQ ユーティリティ・プログラムのリファレンス項目 (使用可能な構文、パラメータ、オプションなど) について説明しています。
- 『Sybase IQ によるラージ・オブジェクト管理』－ Sybase IQ データ・リポジトリ内での BLOB (Binary Large Object) および CLOB (Character Large Object) の格納と取得について説明しています。このオプションの製品をインストールするには、別のライセンスが必要です。
- 『Sybase IQ インストールおよび設定ガイド』－ プラットフォーム固有の Sybase IQ のインストール手順、新バージョンの Sybase IQ へのマイグレート、特定のプラットフォームでの Sybase IQ の設定について説明しています。
- 『Sybase IQ リリース・ノート』－ 製品およびマニュアルに加えられた最新の変更内容について説明しています。
- 『Sybase IQ の暗号化カラム』－ Sybase IQ データ・リポジトリ内でのユーザによるカラムの暗号化の使用について説明しています。このオプションの製品をインストールするには、別のライセンスが必要です。

Sybase IQ と Adaptive Server Anywhere

Sybase IQ は、SQL Anywhere® Studio のコンポーネントである Adaptive Server® Anywhere を拡張した製品のため、Adaptive Server Anywhere と同じ機能を数多くサポートしています。Sybase IQ のマニュアル・セットは、SQL Anywhere Studio のマニュアルの該当する箇所を参照しています。

Adaptive Server Anywhere には、次のマニュアルがあります。

- 『Adaptive Server Anywhere プログラミング・ガイド』－ ODBC、Embedded SQL™、または Open Client™ インタフェースに直接アクセスするプログラムを開発するアプリケーション開発者を対象にしています。このマニュアルでは、Adaptive Server Anywhere アプリケーションの開発方法について説明しています。
- 『Adaptive Server Anywhere データベース管理ガイド』－ すべてのユーザを対象に、データベースとデータベース・サーバの運用、管理、設定について説明しています。

- 『Adaptive Server Anywhere SQL リファレンス・マニュアル』 – Adaptive Server Anywhere で使用する SQL 言語のリファレンスです。さらに、Adaptive Server Anywhere のシステム・テーブルとプロシージャについても説明します。

Sybase Product Manuals Web サイトでも、SQL Anywhere Studio 9.0.2 コレクションの Adaptive Server Anywhere マニュアルを参照できます。Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。

その他の情報ソース

Sybase Getting Started CD、Sybase CD、Sybase Product Manuals Web サイトを利用すると、製品について詳しく知ることができます。

- Getting Started CD には、PDF 形式のリリース・ノートとインストール・ガイド、および SyBooks CD に含まれていないその他のマニュアルや更新情報が収録されています。この CD は製品のソフトウェアに同梱されています。Getting Started CD に収録されているマニュアルを参照または印刷するには、Adobe Acrobat Reader が必要です (CD 内のリンクを使用して Adobe の Web サイトから無料でダウンロードできます)。
- SyBooks CD には製品マニュアルが収録されています。この CD は製品のソフトウェアに同梱されています。Eclipse ベースの SyBooks ブラウザでは、使いやすい HTML 形式のマニュアルにアクセスできます。

  一部のマニュアルは PDF 形式で提供されています。それらのマニュアルは SyBooks CD の PDF ディレクトリに収録されています。PDF ファイルを開いたり印刷したりするには、Adobe Acrobat Reader が必要です。

  SyBooks をインストールして起動するまでの手順については、Getting Started CD の『SyBooks Installation Guide』または SyBooks CD の *README.txt* ファイルを参照してください。
- Sybase Product Manuals Web サイトは、SyBooks CD のオンライン版であり、標準の Web ブラウザを使ってアクセスできます。また、製品マニュアルのほか、EBFs/Maintenance、Technical Documents、Case Management、Solved Cases、ニュース・グループ、Sybase Developer Network へのリンクもあります。

  Sybase Product Manuals Web サイトにアクセスするには、Product Manuals (http://www.sybase.com/support/manuals/) にアクセスしてください。
- Infocenter はオンライン・バージョンの SyBooks であり、標準の Web ブラウザで表示できます。Infocenter Web サイトにアクセスするには、Sybooks Online Help (http://infocenter.sybase.com/help/index.jsp) にアクセスしてください。

Web 上の Sybase 製品の動作確認情報

Sybase Web サイトの技術的な資料は頻繁に更新されます。

❖ **製品動作確認の最新情報にアクセスする**

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [Certification Report] をクリックします。

3 [Certification Report] フィルタで製品、プラットフォーム、時間枠を指定して [Go] をクリックします。

4 [Certification Report] のタイトルをクリックして、レポートを表示します。

❖ **コンポーネント動作確認の最新情報にアクセスする**

1 Web ブラウザで Availability and Certification Reports を指定します。(http://certification.sybase.com/)

2 [Search By Base Product] で製品ファミリとベース製品を選択するか、[Search by Platform] でプラットフォームとベース製品を選択します。

3 [Search] をクリックして、入手状況と動作確認レポートを表示します。

❖ **Sybase Web サイト (サポート・ページを含む) の自分専用のビューを作成する**

MySybase プロファイルを設定します。MySybase は無料サービスです。このサービスを使用すると、Sybase Web ページの表示方法を自分専用にカスタマイズできます。

1 Web ブラウザで Technical Documents を指定します。(http://www.sybase.com/support/techdocs/)

2 [MySybase] をクリックし、MySybase プロファイルを作成します。

Sybase EBF とソフトウェア・メンテナンス

❖ **EBF とソフトウェア・メンテナンスの最新情報にアクセスする**

1 Web ブラウザで Sybase Support Page (http://www.sybase.com/support) を指定します。

2 [EBFs/Maintenance] を選択します。ユーザ名とパスワードの入力が求められたら、MySybase のユーザ名とパスワードを入力します。

3 製品を選択します。

4 時間枠を指定して [Go] をクリックします。EBF/Maintenance リリースのリストが表示されます。

鍵のアイコンは、自分が Technical Support Contact として登録されていないため、一部の EBF/Maintenance リリースをダウンロードする権限がないことを示しています。未登録ではあるが、Sybase 担当者またはサポート・コンタクトから有効な情報を得ている場合は、[Edit Roles] をクリックして、「Technical Support Contact」役割を MySybase プロファイルに追加します。

5 EBF/Maintenance レポートを表示するには [Info] アイコンをクリックします。ソフトウェアをダウンロードするには製品の説明をクリックします。

SQL 構文の表記規則

このマニュアルで、構文の説明に使用する表記規則は次のとおりです。

- **キーワード** SQL キーワードは大文字で示します。ただし、SQL キーワードは大文字と小文字を区別しないので、入力するときはどちらで入力してもかまいません。たとえば、SELECT は Select でも select でも同じです。
- **プレースホルダ** 適切な識別子または式で置き換えられる項目は、斜体で表記します。
- **継続** 省略記号 (…) で始まる行は、前の行から文が続いていることを表します。
- **繰り返し項目** 繰り返し項目のリストは、リストの要素の後ろに省略記号 (ピリオド 3 つ ...) を付けて表します。複数の要素を指定できます。複数の要素を指定する場合は、各要素間はカンマで区切る必要があります。
- **オプション指定部分** 文のオプション指定部分は、角カッコで囲みます。例:

  ```
  RELEASE SAVEPOINT [ savepoint-name ]
  ```

  この例では、*savepoint-name* がオプション部分です。角カッコは入力しないでください。
- **オプション** 項目リストから 1 つだけ選択しなければならない場合、また何も選択する必要のない場合は、項目間を縦線で区切り、リスト全体を角カッコで囲みます。例:

  ```
  [ ASC | DESC ]
  ```

  この例では、ASC、DESC のどちらか 1 つを選択しても、何も選択しなくてもかまいません。角カッコは入力しないでください。
- **選択肢** オプションの中の 1 つを必ず選択しなければならない場合は、選択肢を大カッコ { } で囲みます。例:

  ```
  QUOTES { ON | OFF }
  ```

  この例では、ON、OFF のどちらかを必ず入力しなければなりません。大カッコ自体は入力しないでください。

書体の表記規則

表 1 に、このマニュアルで使用している書体の表記規則を示します。

表 1: 書体の表記規則

項目	説明
`Code`	SQL およびプログラム・コードは等幅 (固定幅) 文字フォントで表記します。
`User entry`	ユーザが入力するテキストには等幅 (固定幅) 文字フォントを使用します。
「強調」	強調する言葉は「 」で囲みます。
file names	ファイル名は斜体で表記します。
database objects	テーブル、プロシージャなどのデータベース・オブジェクトの名前は、印刷物では**ゴシック体**フォントで、オンラインでは斜体で表記します。

サンプル・データベース

Sybase IQ にはサンプル・データベースが用意されています。Sybase IQ マニュアルで紹介している例の多くは、このサンプル・データベースによるものです。

サンプル・データベースは小規模企業の例を示しています。データベースには、この企業の内部情報 (employee、department) とともに、製品情報 (product)、販売情報 (sles_order、customer、contact)、財務情報 (fin_code、fin_data) が入っています。

サンプル・データベース (ファイル名 *asiqdemo.db*) は、UNIX システムでは *$ASDIR/demo* ディレクトリに、Windows システムでは *%ASDIR%¥demo* ディレクトリにあります。

アクセシビリティ機能

このマニュアルには、アクセシビリティを重視した HTML 版もあります。この HTML 版マニュアルは、スクリーン・リーダーで読み上げる、または画面を拡大表示するなどの方法により、その内容を理解できるよう配慮されています。

Sybase IQ 12.7 と HTML マニュアルは、連邦リハビリテーション法第 508 条のアクセシビリティ規定に準拠していることがテストにより確認されています。第 508 条に準拠しているマニュアルは通常、World Wide Web Consortium (W3C) の Web サイト用ガイドラインなど、米国以外のアクセシビリティ・ガイドラインにも準拠しています。

Sybase Central 用 Sybase IQ プラグインのアクセシビリティへの対応については、『Sybase IQ の概要』の「アクセシビリティ機能の使用」を参照してください。この製品のオンライン・ヘルプは、スクリーン・リーダーの読み上げで内容を理解でき、Sybase Central のキーボード・ショートカットなどのアクセシビリティ機能についての説明もあります。

アクセシビリティ・ツールの設定
アクセシビリティ・ツールを効率的に使用するには、設定が必要な場合もあります。一部のスクリーン・リーダーは、テキストの大文字と小文字を区別して発音します。たとえば、すべて大文字のテキスト (ALL UPPERCASE TEXT など) はイニシャルで発音し、大文字と小文字の混在したテキスト (MixedCase Text など) は単語として発音します。構文規則を発音するようにツールを設定すると便利かもしれません。詳細については、ツールのマニュアルと『Sybase IQ の概要』の「スクリーン・リーダの使用」を参照してください。

Sybase のアクセシビリティに対する取り組みについては、Sybase Accessibility (http://www.sybase.com/accessibility) を参照してください。Sybase Accessibility サイトには、第 508 条と W3C 標準に関する情報へのリンクもあります。

Sybase IQ の第 508 条準拠の声明については、Sybase Accessibility (http://www.sybase.com/accessibility) を参照してください。

不明な点があるときは

サポート契約を購入済みの Sybase 製品のインストールには、定められた 1 人以上のユーザに対して、Sybase 製品の保守契約を結んでいるサポート・センタを利用する権利が付属します。マニュアルだけでは解決できない問題があった場合には、担当の方を通して Sybase のサポート・センタまでご連絡ください。

第 1 章

データベース・テーブルからのデータの選択

この章について

この章では、基本的なクエリの構築と、製品設計を活用するための手法について説明します。ここではテーブルの内容の表示、クエリ結果の順序付け、カラムとローの選択、探索条件を使ったクエリの絞り込みを行うチュートリアル作業を実行します。

高度なクエリ・パフォーマンスの推奨事項については、「第 3 章 クエリと削除の最適化」を参照してください。

内容

前提条件

DBISQL の代わりにグラフィカルなフロントエンド・ツールを使用してデータベースへのクエリを実行すると、ツールが生成する SQL 構文を表示できる場合があります。たとえば、InfoMaker では [テーブル] ペインタ・バーの [SQL 構文] ボタンを選択して SQL 文を表示できます。

このチュートリアルでは、データベースから情報を取得するときに使用する SELECT 文について説明します。SELECT 文のことを一般にクエリと呼びます。これは、SELECT 文がデータベース内の情報についてデータベース・サーバに問い合わせるためです。

注意 SELECT 文は用途の広いコマンドです。大きなデータベースから非常に具体的な情報を取得するアプリケーションでは、SELECT 文がきわめて複雑になる場合があります。このチュートリアルでは、単純な SELECT 文だけを使用します。以降のチュートリアルで、より高度なクエリについて説明します。SELECT 文の完全な構文については、『Sybase IQ リファレンス・マニュアル』の「第 6 章 SQL 文」の「SELECT 文」を参照してください。

チュートリアルのレッスンを読んで実行している間は、コンピュータで Sybase IQ ソフトウェアを実行しておくことが理想的です。

このチュートリアルでは、すでに DBISQL を起動し、サンプル・データベースに接続していることを前提にしています。まだこれらを行っていない場合は、『Sybase IQ ユーティリティ・ガイド』の「第 2 章 Interactive SQL (dbisql) の使用」を参照してください。

テーブル情報の表示

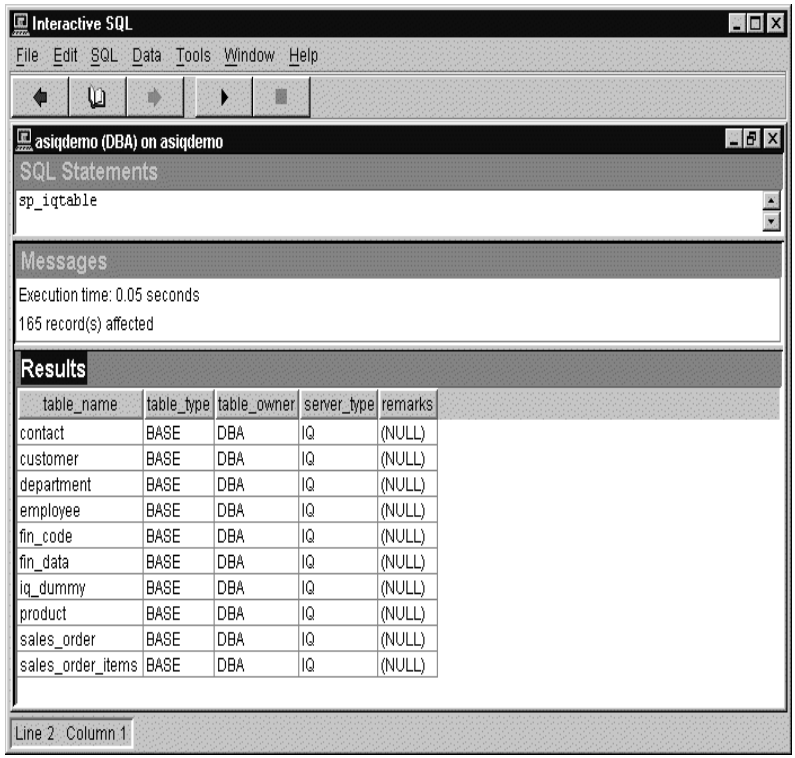
ここでは、*employee* テーブルのデータを表示します。

このチュートリアルで使用するサンプル・データベースは、架空の会社のものです。データベースには、従業員、部署、注文などについての情報が格納されています。すべての情報はテーブルに編成されています。

テーブルのリスト

『Sybase IQ の概要』では、Sybase Central で [テーブル] フォルダを開いてテーブルのリストを表示する方法について説明しました。システム・ストアド・プロシージャの sp_iqtable を使用して、Interactive SQL からユーザ・テーブルをリストすることもできます。システム・ストアド・プロシージャは、Sybase IQ にストアド・プロシージャとして実装されているシステム関数です。

[SQL 文] ウィンドウで `sp_iqtable` と入力し、同じ名前のシステム・ストアド・プロシージャを実行します。



システム・ストアド・プロシージャの詳細については、『Sybase IQ リファレンス・マニュアル』の「第 10 章 システム・プロシージャ」を参照してください。

SELECT 文の使用

このレッスンでは、データベース内のテーブルの 1 つを表示します。使用するコマンドは、employee という名前のテーブル全体を表示します。

次のコマンドを実行します。

```
SELECT * FROM employee
```

アスタリスクは、テーブル内のすべてのカラムを表す省略記号です。

SELECT 文は employee テーブルのすべてのローとカラムを取得し、DBISQL [結果] ウィンドウに次の該当するものが表示されます。

emp_id	manager_id	emp_fname	emp_lname	dept_id
102	501	Fran	Whitney	100
105	501	Matthew	Cobb	100
129	902	Philip	Chin	200
148	1293	Julie	Jordan	300
160	501	Robert	Breault	100

employee テーブルには、**カラム**に編成された複数の**ロー**が格納されています。各カラムには、emp_lname や emp_id などの名前が付いています。会社の従業員 1 人ずつに 1 つのローがあり、それぞれのローは各カラムに値を持ちます。たとえば、従業員 ID が 102 の従業員は Fran Whitney であり、そのマネージャは従業員 ID 501 です。

DBISQL [メッセージ] ウィンドウにも一部の情報が表示されます。この情報については後で説明します。

大文字と小文字の区別

テーブル名 employee は、実際のテーブル名がすべて小文字の場合にも、先頭は大文字の E で表示されます。Sybase IQ データベースは、文字列の比較で大文字と小文字を区別するもの (デフォルト) と区別しないものを作成できますが、その識別子では常に大文字と小文字は区別されません。

注意 このマニュアルの例は、CREATE DATABASE 修飾子の CASE IGNORE を使用して、大文字と小文字を区別しないように作成されています。デフォルトは CASE RESPECT であり、こちらの方がパフォーマンスが向上します。

データベースの作成方法については、『Sybase IQ システム管理ガイド』の「第 5 章 データベース・オブジェクトの使用」を参照してください。

SELECT の代わりに select または Select と入力することもできます。Sybase IQ では、キーワードを大文字、小文字、またはその両方の組み合わせで入力できます。このマニュアルでは、通常、SQL キーワードに大文字を使用しています。

DBISQL 環境の操作方法と DBISQL の使用法は、オペレーティング・システムによって異なります。

データをスクロールして DBISQL 環境を操作する方法については、『Sybase IQ ユーティリティ・ガイド』の「第 2 章 Interactive SQL (dbisql) の使用」を参照してください。

クエリ結果の順序付け

ここでは、SELECT 文に ORDER BY 句を追加して、結果をアルファベット順または数値順に表示します。

特に指定しないかぎり、Sybase IQ ではテーブルのローが順不同で表示されます。テーブルのローを意味のある順序で表示した方が便利なことがよくあります。たとえば、従業員をアルファベット順で表示したいような場合です。

従業員をアルファベット順にリストする

次の例は、SELECT 文に ORDER BY 句を追加して、結果をアルファベット順に取得する方法を示します。

```
SELECT * FROM employee ORDER BY emp_lname
```

emp_id	manager_id	emp_fname	emp_lname	dept_id
1751	1576	Alex	Ahmed	400
1013	703	Joseph	Barker	500
591	1576	Irene	Barletta	400
191	703	Jeannette	Bertrand	500
1336	1293	Janet	Bigelow	300

注意

句の順序は重要です。ORDER BY 句は FROM 句と SELECT 句の後に指定します。

注意　FROM 句を省略した場合、またはクエリ内のすべてのテーブルが SYSTEM dbspace にある場合、クエリは Sybase IQ ではなく Adaptive Server Anywhere によって処理されます。これにより、特に構文上およびセマンティック上の制限とオプション設定の効果に関して、クエリが異なる動作をする場合があります。処理に適用されるルールについては Adaptive Server Anywhere のマニュアルを参照してください。

FROM 句を必要としないクエリを実行する場合は、“FROM iq_dummy” 句を追加して、クエリを強制的に Sybase IQ で処理できます。iq_dummy は、データベースに作成される、ローが 1 つ、カラムが 1 つのテーブルです。

カラムとローの選択

多くの場合、表示する必要がある情報は、テーブル内の一部のカラムだけです。たとえば、従業員への誕生日カードを作成するには、emp_lname、dept_id、birth_date の各カラムを表示すれば十分です。

各従業員の姓、部署、誕生日をリストする

ここでは、各従業員の誕生日、姓、部署 ID を選択します。次のコマンドを入力します。

```
SELECT emp_lname, dept_id, birth_date
FROM employee
```

emp_lname	dept_id	birth_date	...
Whitney	100	1958-06-05	...
Cobb	100	1960-12-04	...
Chin	200	1966-10-30	...
Jordan	300	1951-12-13	...
Breault	100	1947-05-13	...

カラムの並べ替え

この 3 つのカラムは、SELECT コマンドに入力した順序で表示されています。カラムを並べ替えるには、コマンドで指定するカラム名の順序を変更します。たとえば、birth_date カラムを左側に配置するには、次のコマンドを使用します。

```
SELECT birth_date, emp_lname, dept_id
FROM employee
```

ローの順序付け

次のように、特定のカラムだけを表示すると同時に、ローの順序を指定できます。

```
SELECT birth_date, emp_lname, dept_id
FROM employee
ORDER BY emp_lname
```

次のコマンドのアスタリスクは、テーブル内のすべてのカラムを表す省略記号です。

```
SELECT * FROM employee
```

探索条件の使用

ここでは、WHERE 句の複合探索条件、パターン・マッチング、探索条件ショートカットを使用して、日付を比較する手順について説明します。

employee テーブルにある一部の従業員の情報だけを表示したいことがあります。SELECT 文に WHERE 句を追加すると、テーブルから一部のローだけを選択できます。

たとえば、John という名前の従業員だけを表示するとします。

❖ John という名前のすべての従業員をリストするには

- 次のコマンドを入力します。

```
SELECT *
FROM employee
WHERE emp_fname = 'John'
```

emp_id	manager_id	emp_fname	emp_lname	dept_id
318	1576	John	Crow	400
862	501	John	Sheffield	100
1483	1293	John	Letiecq	300

アポストロフィおよび大文字と小文字の区別

- 名前 'John' はアポストロフィ (一重引用符) で囲む必要があります。アポストロフィは、John が文字列であることを示します。引用符 (二重引用符) には別の意味があります。引用符を使用すると、無効な文字列を有効なカラム名やその他の識別子として使用できるようになります。
- サンプル・データベースでは大文字と小文字が区別されないため、'JOHN'、'john'、'John' のいずれで検索しても同じ結果が返ります。

次のように、これまで学習した句を組み合わせて実行できます。

```
SELECT emp_fname, emp_lname, birth_date
FROM employee
WHERE emp_fname = 'John'
ORDER BY birth_date
```

注意

- 句を指定する順序は重要です。FROM 句を最初に指定し、その後に WHERE 句、ORDER BY 句の順に指定します。これ以外の順序で句を入力すると、構文エラーが返されます。
- 文を複数の行に分ける必要はありません。[SQL 文] ウィンドウに自由なフォーマットで文を入力できます。入力した文が画面の行数を超えると、[SQL 文] ウィンドウのテキストがスクロールします。

クエリでの日付の比較

検索対象の正確な値がわからない場合や、一連の値を表示したい場合があります。WHERE 句で比較を使用すると、探索条件を満たす一連のローを選択できます。

1964 年 3 月 3 日より前に生まれた従業員をリストする

次の例は、日付の不等号探索条件の使い方を示します。次のコマンドを入力します。

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date < 'March 3, 1964'
```

emp_lname	birth_date
Whitney	1958-06-05 00:00:00.000
Cobb	1960-12-04 00:00:00.000
Jordan	1951-12-13 00:00:00.000
Breault	1947-05-13 00:00:00.000
Espinoza	1939-12-14 00:00:00.000
Dill	1963-07-19 00:00:00.000

Sybase IQ は、birth_date カラムに日付が格納されていることを認識し、自動的に *'March 3, 1964'* を日付に変換します。

WHERE 句での複合探索条件

これまでに、比較演算子の等号 (=) と未満 (<) を見てきました。Sybase IQ では、より大きい (>)、以上 (>=)、以下 (<=)、等しくない (<>) などのその他の比較演算子もサポートされています。

これらの条件を AND や OR を使って組み合わせると、より複雑な探索条件を作成できます。

リストの修飾

1964 年 3 月 3 日より前に生まれた従業員のうち、Whitney という名前の従業員を除くすべての従業員をリストするには、次のコマンドを入力します。

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date < '1964-3-3'
AND emp_lname <> 'Whitney'
```

emp_lname	birth_date
Cobb	1960-12-04 00:00:00.000
Jordan	1951-12-13 00:00:00.000
Breault	1947-05-13 00:00:00.000
Espinoza	1939-12-14 00:00:00.000
Dill	1963-07-19 00:00:00.000
Francis	1954-09-12 00:00:00.000

探索条件でのパターン・マッチング

もう 1 つの便利な検索方法が、パターンによる検索です。SQL では、LIKE という語を使用してパターンを検索します。LIKE の使い方について、例を挙げて説明します。

姓が BR で始まる従業員をリストする

次のコマンドを入力します。

```
SELECT emp_lname, emp_fname
FROM employee
WHERE emp_lname LIKE 'br%'
```

emp_lname	emp_fname
Breault	Robert
Braun	Jane

探索条件内の % は、BR という文字の後に別の文字が何文字続いてもかまわないことを示します。

姓検索の修飾

姓が BR で始まり、その直後または数文字後に T という文字を含み、T で終わるかさらに別の文字が続くすべての従業員をリストするには、次のコマンドを使用します。

```
SELECT emp_lname, emp_fname
FROM employee
WHERE emp_lname LIKE 'BR%T%'
```

emp_lname	emp_fname
Breault	Robert

最初の % 記号は文字列 “eaul” と一致し、2 番目の % 記号は空の文字列 (文字なし) と一致します。

LIKE で使用できるもう 1 つの特殊文字に _ (アンダースコア) 文字があります。これは 1 文字と一致します。

BR_U% というパターンは、BR で始まり、4 番目の文字が U であるすべての名前と一致します。Braun では、_ が A という文字と一致し、% が N と一致します。

発音によるローのマッチング

SOUNDEX 関数を使用すると、スペルだけでなく読みによってもローをマッチングできます。たとえば、電話メッセージが残されていて、その宛先が “Ms. Brown” のように発音されていたとします。社内で Brown のように発音される名前を持つ従業員を見つける必要があります。

発音による姓の検索

Brown のように発音される姓を持つ従業員をリストするには、次のコマンドを入力します。

```
SELECT emp_lname, emp_fname
FROM employee
WHERE SOUNDEX( emp_lname ) = SOUNDEX( 'Brown' )
```

emp_lname	emp_fname
Braun	Jane

この探索条件に一致する従業員は Jane Braun だけです。

探索条件を入力するためのショートカット

省略形 BETWEEN の使用

SQL には、探索条件を入力するための省略形が 2 つあります。1 つは BETWEEN であり、値を範囲で検索するときに使用します。この例を次に示します。

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date BETWEEN '1964-1-1'
AND '1965-3-31'
```

これは次のコマンドに相当します。

```
SELECT emp_lname, birth_date
FROM employee
WHERE birth_date >= '1964-1-1'
AND birth_date <= '1965-3-31'
```

省略形 IN の使用

もう 1 つの省略形 IN は、複数のいずれかの値を検索するときに使用します。次にコマンド例を示します。

```
SELECT emp_lname, emp_id
FROM employee
WHERE emp_lname IN ('Yeung','Bucceri','Charlton')
```

上記のコマンドは、次のコマンドと同じです。

```
SELECT emp_lname, emp_id
FROM employee
WHERE emp_lname = 'Yeung'
OR emp_lname = 'Bucceri'
OR emp_lname = 'Charlton'
```

集約データの取得

ここでは、集約情報を返すクエリを構築する方法について説明します。集約情報の例を次に示します。

- カラム内のすべての値の合計
- カラム内のエントリの数
- カラム内のエントリの平均値

集合関数の概要

従業員の人数を調べたいとします。次の文は、employee テーブルのローの数を取得します。

```
SELECT count( * )
FROM employee
```

count(*)
75

このクエリによって、1 つのカラム (count(*) というタイトル) と 1 つのロー (従業員数が格納されている) だけで構成されるテーブルが返されます。

次のコマンドは、やや複雑な集約クエリです。

```
SELECT   count( * ),
min( birth_date ),
max( birth_date )
FROM employee
```

count(*)	min(birth_date)	max(birth_date)
75	1936-01-02	1973-01-18

このクエリの結果セットは、3 つのカラムと 1 つのローで構成されます。3 つのカラムには、従業員数、年齢が最も高い従業員の誕生日、年齢が最も低い従業員の誕生日が格納されています。

COUNT、MIN、MAX を「**集合関数**」と呼びます。これらの各関数は、テーブル全体の情報を要約します。集合関数は、MIN、MAX、COUNT、AVG、SUM、STDDEV、VARIANCE と全部で 7 個あります。すべての関数が、パラメータとしてカラム名または式を使用します。前述のように、COUNT はアスタリスクもパラメータとして使用します。

集合関数によるグループ化されたデータの取得

テーブル全体についての情報を取得することに加えて、集合関数をローのグループに対して使うこともできます。

ローのグループに対する集合関数の使用

各営業担当者が受け持つ注文数をリストするには、次のコマンドを入力します。

```
SELECT sales_rep, count( * )
FROM sales_order
GROUP BY sales_rep
```

sales_rep	count(*)
129	57
195	50
299	114
467	56
667	54

このクエリの結果は、各営業担当者の ID 番号別に、営業担当者の ID が格納されたローと、sales_order テーブル内でその ID 番号を持つローの数で構成されます。

GROUP BY 句を使用すると、結果のテーブルには、GROUP BY で指定したカラムで見つかった値別のローが表示されます。

グループの制限

WHERE 句を使用して、クエリでローを制限する方法についてはすでに説明しました。GROUP BY 句の制限には、HAVING キーワードを使用します。

GROUP BY 句の制限

注文数が 55 を超えるすべての営業担当者をリストするには、次のコマンドを入力します。

```
SELECT sales_rep, count( * )
FROM  sales_order
GROUP BY sales_rep
HAVING count( * ) > 55
```

sales_rep	count(*)
129	57
299	114
467	56
1142	57

注意 GROUP BY は常に HAVING の前に指定します。同様に、WHERE は GROUP BY の前に指定します。

WHERE と GROUP BY の使用

注文数が 55 を超えており、ID が 1000 より大きいすべての営業担当者をリストするには、次のコマンドを入力します。

```
SELECT sales_rep, count( * )
FROM sales_order
WHERE sales_rep > 1000
GROUP BY sales_rep
HAVING count( * ) > 55
```

Sybase IQ クエリ・オプティマイザは、それによってパフォーマンスが向上する場合、述部を HAVING 句から WHERE 句に移動します。たとえば、上記の例で WHERE 句の代わりに述部を次のように指定した場合、クエリ・オプティマイザは述部を WHERE 句に移動します。

```
GROUP BY sales_rep
HAVING count( *) > 55
 AND sales_rep > 1000
```

Sybase IQ は、この最適化を (OR や IN を伴わない) 単純な条件を使って実行します。このため、WHERE 句と HAVING 句の両方を含むクエリを構築するときは、できるだけ多くの条件を WHERE 句で指定するようにします。

小計計算の活用

日付や場所などの次元によって異なるデータがある場合に、各次元でデータがどのように異なるかを調べることが必要になる場合があります。ROLLUP 演算子と CUBE 演算子を使用すると、グループ化カラムへの参照のリストから複数レベルの小計と総計を作成できます。小計は、最も詳細なレベルから総計まで「ロールアップ」します。たとえば、販売データを分析している場合は、同じクエリを使用して全体の平均と年別の平均販売数を計算できます。

ROLLUP の使用

年別、モデル別、色別の合計自動車販売数を選択するには、次のコマンドを使用します。

```
SELECT year, model, color, sum(sales)
FROM sales_tab
GROUP BY ROLLUP (year, model, color);
```

year	model	color	sales
1990	Chevrolet	red	5
1990	Chevrolet	white	87
1990	Chevrolet	blue	62
1990	Chevrolet	NULL	154
1990	Ford	blue	64
1990	Ford	red	62
1990	Ford	white	63
1990	Ford	NULL	189
1990	NULL	NULL	343

year	model	color	sales
1991	Chevrolet	blue	54
1991	Chevrolet	red	95
1991	Chevrolet	white	49
1991	Chevrolet	NULL	198
1991	Ford	blue	52
1991	Ford	red	55
1991	Ford	white	9
1991	Ford	NULL	116
1991	NULL	NULL	314
NULL	NULL	NULL	657

このクエリを処理するときに、Sybase IQ は最初に、指定された 3 つすべてのグループ化式 (year、model、color) によってデータをグループ化し、次に最後の式 (color) を除くすべてのグループ化式によってデータをグループ化します。5 番目のローの NULL は、color カラムの ROLLUP 値、つまり、そのモデルのすべての色の合計販売数を示します。343 は、1990 年のすべてのモデルと色の合計販売数を表し、314 は 1991 年の合計販売数を表します。最後のローは、すべての年のすべてのモデルとすべての色の合計販売数を表します。

ROLLUP 演算子には、引数としてグループ化式の順番リストを指定する必要があります。他のグループを含むグループをリストするときは、先に大きい方のグループをリストします (たとえば、state をリストしてから city をリストします)。

ROLLUP 演算子は、集合関数の SUM、COUNT、AVG、MIN、MAX、STDDEV、VARIANCE とともに使用できます。ただし、ROLLUP は COUNT DISTINCT と SUM DISTINCT をサポートしていません。

CUBE の使用

次のクエリでは、人々の州 (地理的位置)、性別、教育水準、所得を含む国勢調査のデータを使用します。GROUP BY 句の CUBE 拡張を使用すると、census テーブル内の国勢調査データを 1 回参照するだけで、州、性別、教育水準の国勢調査全体の平均所得を計算し、state、gender、education の各カラムの可能なすべての組み合わせの平均所得を計算できます。たとえば、すべての州のすべての女性の平均所得を計算する場合や、教育水準と地理的位置を基準に国勢調査のすべての人々の平均所得を計算する場合に、CUBE 演算子を使用します。

CUBE でグループを計算するときに、CUBE は計算されたグループのカラムに NULL 値を挿入します。各ローが表すグループの種類と、その NULL がデータベースに格納されている NULL なのか、CUBE が挿入した NULL なのかを区別することは困難です。この問題を解決するのが GROUPING 関数です。指定されたカラムが上位レベルのグループにマージされている場合、この関数は 1 を返します。

次のクエリは、GROUPING 関数を GROUP BY CUBE と組み合わせた使用例です。

```
SELECT
CASE GROUPING ( state ) WHEN 1 THEN 'ALL' ELSE state END
AS c_state,
CASE GROUPING ( gender ) WHEN 1 THEN 'ALL' ELSE gender
END AS c_gender,
CASE GROUPING ( education ) WHEN 1 THEN 'ALL' ELSE
education END AS c_education,
COUNT(*), CAST (ROUND ( AVG ( income ), 2 ) AS NUMERIC
(18,2)) AS average
FROM census
GROUP BY CUBE (state, gender, education);
```

このクエリの結果は次のとおりです。CUBE が生成した小計ローを示す NULL 値が、クエリ内の指定によって小計ローで ALL に置き換わっています。

c_state	c_gender	c_education	count(*)	average
MA	f	BA	3	48333.33
MA	f	HS	2	40000.00
MA	f	MS	1	45000.00
MA	f	ALL	6	45000.00
MA	m	BA	4	55000.00
MA	m	HS	1	55000.00
MA	m	MS	3	85000.00
MA	m	ALL	8	66250.00
MA	ALL	ALL	14	57142.86
NH	f	HS	2	50000.00
NH	f	MS	1	85000.00
NH	f	ALL	3	61666.67
NH	m	BA	3	55000.00
NH	m	MS	1	49000.00
NH	m	ALL	4	53500.00
NH	ALL	ALL	7	57000.00
ALL	ALL	ALL	21	57095.24
ALL	ALL	BA	10	53000.00
ALL	ALL	MS	6	72333.33
ALL	ALL	HS	5	47000.00
ALL	f	ALL	9	50555.56

c_state	c_gender	c_education	count(*)	average
ALL	m	ALL	12	62000.00
ALL	f	BA	3	48333.33
ALL	m	HS	1	55000.00
ALL	m	MS	4	76000.00
ALL	m	BA	7	55000.00
ALL	f	MS	2	65000.00
ALL	f	HS	4	45000.00
NH	ALL	HS	2	50000.00
NH	ALL	MS	2	67000.00
MA	ALL	MS	4	75000.00
MA	ALL	HS	3	45000.00
MA	ALL	BA	7	52142.86
NH	ALL	BA	3	55000.00

ROLLUP と CUBE は、データ・ウェアハウス管理者が次のような処理を行うときに特に役立ちます。

- 地理や時間などの階層的な次元での小計 (たとえば、年／月／日や国／州／市)
- 要約テーブルへのデータの格納

ROLLUP と CUBE を使用すると、レベルごとに別々のクエリを使用する代わりに、1 つのクエリを使用して、複数レベルのグループ化を使ってデータを計算できます。

ROLLUP 演算子と CUBE 演算子の詳細については、『Sybase IQ リファレンス・マニュアル』の「第 6 章 SQL 文」の「SELECT 文」を参照してください。

分析データの取得

ここでは、分析情報を返すクエリを構築する方法について説明します。統計関数には、ランク付けと逆分散統計の 2 種類があります。ランク付け統計関数は、グループ内の項目をランク付けしたり、分散統計を計算したり、結果セットを複数のグループに分割したりします。逆分散統計関数は、K- 理論パーセンタイル値を返します。これは、ひとまとまりのデータの値として許容し得るしきい値を決定する際に使用します。

ランク分析関数には、RANK、DENSE_RANK、PERCENT_RANK、NTILE があります。逆分散統計関数には、PERCENTILE_CONT と PERCENTILE_DISC があります。

たとえば、自動車販売店の販売状況を調べたいとします。NTILE 関数で、各販売店が販売した車の台数に基づいて、販売店を 4 つのグループに分類します。ntile = 1 になっているのは、車の販売台数で上位 25% までのディーラです。

```
SELECT dealer_name, sales,
NTILE(4) OVER ( ORDER BY sales DESC )
FROM carSales;

dealer_name        sales       ntile
Boston             1000        1
Worcester          950         1
Providence         950         1
SF                 940         1
Lowell             900         2
Seattle            900         2
Natick             870         2
New Haven          850         2
Portland           800         3
Houston            780         3
Hartford           780         3
Dublin             750         3
Austin             650         4
Dallas             640         4
Dover              600         4
```

販売台数で上位 10% の販売店を調べるには、この例の SELECT 文で NTILE(10) を指定します。同様に、販売台数で 50% の販売店を調べるには、NTILE(2) を指定します。

NTILE はクエリ結果を指定された数のバケットに分割し、バケット内の各ローにバケット番号を割り当てるランク分析関数です。結果セットは 10 個 (十分位数)、4 個 (四分位数)、その他の数のグループに分割できます。

ランク分析関数では、OVER (ORDER BY) 句を指定する必要があります。ORDER BY 句は、ランク付けを実行するパラメータと、各グループ内でローをソートする順序を指定します。この ORDER BY 句は、OVER 句の中だけで使用されるもので、SELECT の ORDER BY とは異なります。

OVER 句は、関数がクエリの結果セットに対して処理を行うことを示します。結果セットは、FROM、WHERE、GROUP BY、HAVING の各句がすべて評価された後で返されるローです。OVER 句には、ランク付け統計関数の計算の対象となるローのデータ・セットを定義します。

同様に、逆分布関数では WITHIN GROUP (ORDER BY) 句を指定する必要があります。ORDER BY 句は、百分位関数を実行する式と、各グループでローをソートする順序を指定します。この ORDER BY 句は、WITHIN GROUP 句の中でだけ使用されるもので、SELECT の ORDER BY とは異なります。WITHIN GROUP 句は、クエリの結果を並べ替えて、関数が結果を計算するためのデータ・セットを形成します。

分析関数の詳細については、『Sybase IQ リファレンス・マニュアル』の「第 5 章 SQL 関数」の「統計関数」を参照してください。個別の分析関数については、「SQL 関数」の章の各関数の項を参照してください。

重複したローの削除

SELECT 文の結果テーブルに、重複したローが含まれることがあります。DISTINCT キーワードを使用すると、重複したローを削除できます。たとえば、次のコマンドを実行すると、多くの重複したローが返ります。

```
SELECT city, state FROM employee
```

市と州のユニークな組み合わせだけをリストするには、次のコマンドを使用します。

```
SELECT DISTINCT city, state FROM employee
```

注意 ROLLUP 演算子と CUBE 演算子は、DISTINCT キーワードをサポートしていません。

この章では、単一テーブルの SELECT 文の概要について説明しました。単一テーブルの SELECT 文の詳細については、『Sybase IQ システム管理ガイド』の「第 5 章 データベース・オブジェクトの使用」、『Sybase IQ リファレンス・マニュアル』の「第 3 章 SQL 言語の要素」、『Sybase IQ リファレンス・マニュアル』の「第 6 章 SQL 文」の「SELECT 文」を参照してください。

次の章では、SELECT 文の高度な使い方について説明します。

第 2 章

テーブルのジョイン

この章について

この章では、複数のテーブルにある情報を参照する方法と、さまざまな種類のジョインについて説明します。ここでは、テーブルをジョインするチュートリアル作業を実行します。

内容

外積を使用したテーブルのジョイン

サンプル・データベースに、会社の財務データをリストする fin_data というテーブルがあります。各データ・レコードには、そのレコードの部署と、それが支出レコードか収入レコードかを示す code カラムがあります。fin_data テーブルには 84 のローがあります。

2 つのテーブルから同時に情報を取り出すには、SELECT クエリの FROM 句で、両方のテーブルをカンマで区切って指定します。

例

次の dbisql SELECT コマンドは、fin_code テーブルと fin_data テーブルのすべてのデータをリストします。

```
SELECT *
FROM fin_code, fin_data
```

dbisql [データ] ウィンドウに表示されるこのクエリの結果は、fin_code テーブルのすべてのローと fin_data テーブルのすべてのローに一致します。このジョインを完全外積または直積と呼びます。各ローは、fin_code テーブルのすべてのカラム、fin_data テーブルのすべてのカラムの順で構成されます。

外積ジョインは、ジョインを理解するための単純な出発点にすぎず、それ自体はあまり役に立ちません。これ以降の項で、より選択性の高いジョインを構築する方法について説明します。このジョインは、外積テーブルへの制限の適用と考えることができます。

ジョインの制限

外積ジョインを有効に利用するには、何らかの条件を満たすローだけを結果に含める必要があります。ジョイン条件と呼ばれるこの条件では、比較演算子 (=、=>、< など) を使用して、あるテーブルの 1 つのカラムを別のテーブルの 1 つのカラムと比較します。これにより、外積の結果から一部のローを除外します。

たとえば、前の項のジョインを有効に利用するには、sales_order テーブルの sales_rep と employee テーブルの従業員番号が一致するローだけを結果に含めるように指定します。これにより、各ローには注文と、その注文を担当する営業担当者の情報が格納されます。

例 1

これを実行するには、前のクエリに WHERE 句を追加し、従業員とその担当登録のリストを表示します。

```
SELECT *
FROM sales_order, employee
WHERE sales_order.sales_rep = employee.emp_id
```

カラムを識別するために、テーブル名をプレフィクスとして指定します。この例では必ずしも必要ありませんが、テーブル名のプレフィクスを使用すると文が明確になります。2 つのテーブルに同じ名前のカラムがあるときは、このプレフィクスを指定する必要があります。このようなコンテキストで使用するテーブル名を「**修飾子**」と呼びます。

このクエリの結果には 648 のローしかありません (sales_order テーブルの各ローに 1 つずつ)。ジョインした元の 48,600 のローのうち、648 のローにだけ 2 つのテーブルで共通する従業員番号が含まれています。

例 2

次のクエリでは、一部のカラムだけをフェッチし、結果を順序付けするように変更を加えています。

```
SELECT employee.emp_lname, sales_order.id,
sales_order.order_date
FROM sales_order, employee
WHERE sales_order.sales_rep = employee.emp_id
ORDER BY employee.emp_lname
```

SELECT コマンドに多くのテーブルがある場合は、修飾子名をいくつも入力しなければならないことがあります。このようなときは、相関名を使用して入力の手間を省くことができます。

相関名

相関名は、テーブルの特定のインスタンスのエイリアスです。このエイリアスは、1 つの文中でのみ有効です。相関名を作成するには、テーブル名のすぐ後ろに、テーブル名の省略形をキーワード AS で区切って指定します。それ以降は、修飾子としてテーブル名の代わりにこの省略形を使用する必要があります。

```
SELECT E.emp_lname, S.id, S.order_date
FROM sales_order AS S, employee AS E
WHERE S.sales_rep = E.emp_id
ORDER BY E.emp_lname
```

この例では、sales_order テーブルと employee テーブルに対応する S と E という 2 つの相関名を作成しています。

注意　テーブル名や相関名が必要になるのは、異なるテーブルに同じ名前のカラムがあり、不明確になることを避ける場合だけです。相関名を作成した場合は、テーブル名の代わりに必ず相関名を使用します。相関名を作成していない場合は、テーブル名を使用します。

テーブル間の関係

他の種類のジョインを構築するには、あるテーブルの情報が別のテーブルの情報とどのように関係するかを先に理解する必要があります。

テーブルのプライマリ・キーは、そのテーブル内の各ローを識別します。各テーブルは、外部キーを使って互いに関連付けられます。

ここでは、プライマリ・キーと外部キーを組み合わせて、複数のテーブルからクエリを構築する方法について説明します。

プライマリ・キーによるローの識別

asiqdemo データベースのすべてのテーブルには、プライマリ・キーが設定されています (各テーブルにプライマリ・キーを定義することをおすすめします)。プライマリ・キーは、テーブル内のローをユニークに識別する 1 つまたは複数のカラムです。たとえば、従業員番号は従業員をユニークに識別するため、emp_id は employee テーブルのプライマリ・キーになります。

sales_order_items テーブルは、2 つのカラムでプライマリ・キーを構成しているテーブルの例です。注文 ID だけでは、sales_order_items テーブルのローがユニークに識別されません。注文には複数の項目が含まれる場合があるからです。また、line_id 番号も sales_order_items テーブルのローをユニークに識別しません。sales_order_items テーブルのローをユニークに識別するには、注文 ID 名と line_id の両方が必要です。両方のカラムが一緒になってテーブルのプライマリ・キーになります。

外部キーによって関連付けられたテーブル

asiqdemo データベースのいくつかのテーブルは、データベース内の他のテーブルを参照しています。たとえば、sales_order テーブルには、注文を担当する従業員を示す sales_rep カラムがあります。sales_order テーブルには、従業員をユニークに識別するために必要な最小限の情報だけが格納されています。sales_order テーブルの sales_rep カラムは、employee テーブルに対する外部キーになっています。

外部キー

外部キーは、他のテーブルの候補キーの値を含む 1 つまたは複数のカラムです (候補キーの詳細については、『Sybase IQ システム管理ガイド』の「第 5 章 データベース・オブジェクトの使用」を参照してください)。従業員データベース内の各外部キーの関係は、2 つのテーブル間の矢印によって図示されます。『Sybase IQ の概要』の図 1-1 (11 ページ) のサンプル・データベースの図に、これらの矢印が示されています。矢印は関係の外部キー側を起点とし、候補キー側を指し示しています。

ジョイン演算子

多くの一般的なジョインは、外部キーで関連付けられた 2 つのテーブル間で行われます。最も一般的なジョインでは、外部キーの値がプライマリ・キーの値と等しいものに制限されます。すでに見てきた例では、sales_order テーブルの外部キーの値が、*employee* テーブルの候補キーの値と等しいものに制限されています。

```
SELECT emp_lname, id, order_date
FROM sales_order, employee
WHERE sales_order.sales_rep = employee.emp_id
```

KEY JOIN を使用すると、クエリをより簡単に表現できます。

キー・ジョインを使用したテーブルのジョイン

キー・ジョインは、外部キーで関連付けられたテーブルを簡単にジョインする方法です。例：

```
SELECT emp_lname, id, order_date
FROM sales_order
KEY JOIN employee
```

このコマンドは、次のように 2 つの従業員 ID 番号カラムを結び付ける WHERE 句を使ったクエリと同じ結果をもたらします。

```
SELECT emp_lname, id, order_date
FROM sales_order, employee
WHERE sales_order.sales_rep = employee.emp_id
```

ジョイン演算子 (KEY JOIN) は、単に WHERE 句の入力の手間を省くためのもので、2 つのクエリはまったく同じものです。

『Sybase IQ の概要』の asiqdemo データベースの図では、外部キーがテーブル間の線で表されています。図中で 2 つのテーブルが線で結合されていれば、KEY JOIN 演算子を使用できます。キー・ジョインによるクエリで期待どおりの結果を得るには、アプリケーションで外部キーを強制的に適用する必要があります。

複数のテーブルのジョイン

ジョイン演算子を使用して、複数のテーブルをジョインできます。次のクエリでは、4 つのテーブルを使用して、注文の合計額を顧客別にリストしています。customer、sales_order、sales_order_items、product の 4 つのテーブルを、テーブルの各ペア間の 1 つの外部キー関係で接続しています。

```
SELECT company_name,
CAST( SUM(sales_order_items.quantity *
product.unit_price) AS INTEGER) AS value
FROM customer
KEY JOIN sales_order
KEY JOIN sales_order_items
KEY JOIN product
GROUP BY company_name
```

company_name	value
McManus Inc.	3,156
Salt & Peppers.	4,980
The Real Deal	1,884
Totos Active Wear	2,496
The Ristuccia Center	4,596
...	

このクエリで使用している CAST 関数は、式のデータ型を変換します。この例では、整数として返される合計が値に変換されます。

ナチュラル・ジョインを使用したテーブルのジョイン

NATURAL JOIN 演算子は、共通のカラム名に基づいて 2 つのテーブルをジョインします。言い換えると、Sybase IQ が各テーブルに共通するカラムを結び付ける WHERE 句を生成します。

例

たとえば、次のようなクエリがあるとします。

```
SELECT emp_lname, dept_name
FROM employee
NATURAL JOIN department
```

この例では、データベース・サーバが 2 つのテーブルを参照し、共通するカラム名は dept_id だけであると判断します。次の ON フレーズが内部的に生成され、ジョインの実行に使用されます。

```
FROM employee JOIN department
...
ON employee.dept_id = department.dept_id
```

NATURAL JOIN を使用したときのエラー

このジョイン演算子では、意図しないカラムを結び付けてしまう問題が起きる可能性があります。たとえば、次のクエリは意図しなかった結果をもたらします。

```
SELECT *
FROM sales_order
NATURAL JOIN customer
```

このクエリの結果には、ローが 1 つもありません。

データベース・サーバは、内部的に次の ON フレーズを生成します。

```
FROM sales_order JOIN customer
    ON sales_order.id = customer.id
```

sales_order テーブルの id カラムは、注文の ID 番号です。一方、customer テーブルの id カラムは、顧客の ID 番号です。これらの番号は 1 つも一致しません。たとえ一致する番号があったとしても、当然それは意味を持ちません。

ジョイン演算子をむやみに使用しないように注意してください。ジョイン演算子は、単に強制力のない外部キーや共通のカラム名で WHERE 句を入力する手間を省くためのものであることを忘れないでください。WHERE 句を注意して使用しないと、意図しない結果をもたらすクエリを作成してしまう可能性があります。

アドホック・ジョインとジョイン・インデックスの使用

クエリで参照されるジョイン・カラムにジョイン・インデックスを定義している場合、Sybase IQ は自動的にそれらを使用してクエリ処理を高速化します (ジョイン・インデックスの定義については、『Sybase IQ システム管理ガイド』の「第 6 章 Sybase IQ インデックスの使用」を参照してください)。

ジョイン・インデックスを使用しないジョインを「**アドホック・ジョイン**」呼びます。クエリでいくつものテーブルを参照しており、その中にジョイン・インデックスが定義されていないテーブルがある場合、Sybase IQ は定義されているテーブルではジョイン・インデックスを、それ以外のテーブルではアドホック・ジョインを使用します。

可能なすべてのジョインに対してジョイン・インデックスを作成することはできないため、ときにはアドホック・ジョインが必要になることがあります。Sybase IQ の最適化によって、クエリはジョイン・インデックスなしでも同等かそれ以上のパフォーマンスで実行されます。

ジョイン・インデックスの作成には、次の制約があります。

- インデックス内では、完全な外部ジョインだけがサポートされます。クエリは、インデックス付けされている場合、内部、左外部、右外部のジョインになります。

  完全な外部ジョインでは、指定された左右両方のテーブルのすべてのローが結果に含まれ、対応するカラムに一致する値がないカラムについては NULL が返されます。

- ジョイン述部の ON 句で使用できる比較演算子は EQUALS だけです。
- ON 句の代わりに NATURAL キーワードを使用できますが、1 対のテーブルしか指定できません。
- ジョイン・インデックス・カラムはいずれも同じデータ型、精度、位取りでなければなりません。

ジョインとデータ型

最適なパフォーマンスを得るには、ジョイン・カラムを類似のデータ型にする必要があります。Sybase IQ では、暗黙の変換が存在する任意のデータ型でアドホック・ジョインを行うことができます。ただし、ジョイン・カラムのデータ型が同じでない場合は、データ型とテーブルのサイズによって、パフォーマンスがさまざまな範囲で低下する可能性があります。たとえば、INT を BIGINT のカラムにジョインすることはできますが、このジョインによって特定の種類の最適化ができなくなります。Sybase IQ インデックス・アドバイザは、データ型が異なるジョイン・カラムにパフォーマンス上の問題があると見なします。

暗黙のデータ型変換のテーブルについては、『Sybase IQ システム管理ガイド』の「第 7 章 データベースへのデータの入出力」を参照してください。

ストアまたはデータベース間ジョインのサポート

この項では、ストア間またはデータベース間ジョインに対する現在のサポートを明確にします。

Sybase IQ データベース内でのテーブルのジョイン

指定された Sybase IQ データベース内では、あらゆる種類のジョインがサポートされます。つまり、カタログ・ストアの任意のシステムまたはユーザ・テーブルを、IQ ストアの任意のテーブルに任意の順序でジョインできます。

Adaptive Server Enterprise テーブルと Sybase IQ テーブルのジョイン

Sybase IQ テーブルと Adaptive Server Enterprise データベースのテーブルのジョインは、次の条件下でサポートされます。

- Sybase IQ データベースは、ローカル・データベースとリモート・データベースのどちらでもかまいません。
- ASE で Sybase IQ テーブルをプロキシ・テーブルとして使用する場合は、テーブル名を 30 文字以内にしてください。
- ローカルの Adaptive Server Enterprise テーブルをリモートの Sybase IQ 12 テーブルにジョインするには、ASE のバージョンが 11.9.2 以降である必要があります。また、次の適切なサーバ・クラスを使用してください。
  - Adaptive Server Enterprise 12.5 以降のフロントエンドからリモートの Sybase IQ 12.5 以降に接続するには、ASE 12.5 で追加された ASIQ サーバ・クラスを使用します。
  - Adaptive Server Enterprise 11.9.2 から 12.0 までのフロントエンドからリモートの Sybase IQ 12.x (または Adaptive Server Anywhere 6.x 以降) に接続するには、サーバ・クラス ASAnywhere を使用します。
- ローカルの Sybase IQ テーブルを任意のリモート・テーブルとジョインする場合は、ローカルのテーブルを FROM 句の最初に指定する必要があります。つまり、ローカルのテーブルは、ジョインの最も外側のテーブルになります。

Sybase IQ と Adaptive Server Enterprise の間のジョインは、コンポーネント統合サービス (CIS) に依存します。

Adaptive Server Enterprise データベースから Sybase IQ へのクエリの詳細については、Adaptive Server Enterprise 主要マニュアル・セットの『コンポーネント統合サービス・ユーザーズ・ガイド』を参照してください。

Sybase IQ から他のデータベースへのクエリの詳細については、「リモート・データベースと異種データベースのクエリ」を参照してください。

Adaptive Server Anywhere テーブルと Sybase IQ テーブルのジョイン

データベースが BLANK PADDING OFF を指定して構築された場合、CHAR データ型は Adaptive Server Anywhere と Sybase IQ の間で互換性がありません。文字データをジョイン・キーとして使用して、Adaptive Server Anywhere テーブルと Sybase IQ テーブルの間でデータベースのジョインを実行する場合は、BLANK PADDING ON を指定して CHAR データ型を使用します。

注意　Sybase IQ CREATE DATABASE は、新しいデータベースについては BLANK PADDING OFF をサポートしなくなりました。この変更は、既存のデータベースには影響しません。BlankPadding database プロパティを使用して、既存のデータベースの状態をテストすることができます。

```
select db_property ( 'BlankPadding' )
```

Sybase では、ジョイン結果が正しくなるように、BLANK PADDING OFF によって影響を受ける既存のカラムをすべて変更することを推奨しています。ジョイン・カラムを VARCHAR ではなく CHAR データ型として再作成します。CHAR カラムでは、常にブランクが埋め込まれます。

リモート・データベースと異種データベースのクエリ

ここでは、Sybase IQ をコンポーネント統合サービス (CIS) と組み合わせて使用する方法について説明します。CIS を使用すると、Sybase IQ を通して Adaptive Server Enterprise データベースとリモート・データベースまたは非リレーショナル・データ・ソースにクエリを実行できます。CIS は Sybase IQ の一部としてインストールされます。

CIS を使用すると、リモート・サーバ上のテーブルに、ローカルのテーブルのようにアクセスできます。CIS は、複数のリモート異種サーバのテーブル間でジョインを実行し、1 つのテーブルの内容を、サポートされているリモート・サーバへ転送します。

リモートのデータベースやデータ・ソースにクエリを実行するには、そのテーブルをローカル・プロキシ・テーブルにマッピングする必要があります。CIS は、データがローカルに格納されているかのように、プロキシ・テーブルをクライアント・アプリケーションに示します。テーブルにクエリを実行すると、CIS は実際のサーバ記憶位置を判別します。

❖ リモート・データベースをジョインするには

1. 『Sybase IQ システム管理ガイド』の手順に従って、プロキシ・テーブルを作成します。
2. リモート・テーブルをプロキシ・テーブルにマッピングします。

3 プロキシ・データベース名を各リモート・テーブルの修飾名として使用し、SELECT 文でプロキシ・テーブルを参照します。たとえば、次のような文を発行します。

```
SELECT a.c_custkey, b.o_orderkey
FROM proxy_asiqdemo..cust2 a,
asiqdemo..orders b
WHERE a.c_custkey = b.o_custkey
```

詳細については、『Sybase IQ システム管理ガイド』の「第 16 章 リモート・データへのアクセス」と「第 17 章 リモート・データ・アクセス用のサーバ・クラス」を参照してください。

サブクエリによるジョインの置き換え

ジョインは、複数のテーブルのデータから構築される結果テーブルを返します。サブクエリを使用して、同じ結果テーブルを取得することもできます。サブクエリは、単に別の SELECT 文の中にある SELECT 文です。より複雑で多くの情報を与えるクエリを構築するときに、このツールが役立ちます。

たとえば、注文とその発注先の会社を時系列にリストする必要があり、顧客 ID の代わりに会社名を使いたいとします。この結果を得るには、次のようなジョインを使用します。

ジョインの使用

1994 年の年初以降の各注文の order_id、order_date、company_name をリストするには、次のコマンドを入力します。

```
SELECT  sales_order.id,
sales_order.order_date,
customer.company_name
FROM sales_order
KEY JOIN customer
WHERE order_date > '1994/01/01'
ORDER BY order_date
```

id	order_date	company_name
2473	1994-01-04	Peachtree Active Wear
2474	1994-01-04	Sampson & Sons
2036	1994-01-05	Hermanns
2475	1994-01-05	Salt & Peppers
2106	1994-01-05	Cinnamon Rainbows

外部ジョインの使用

前項のチュートリアルのジョインは、より正確には「**内部ジョイン**」と呼ばれます。

外部ジョインを明示的に指定します。この場合は、GROUP BY 句も必要です。

```
SELECT  company_name,
MAX( sales_order.id ),state
FROM customer
KEY LEFT OUTER JOIN sales_order
WHERE state = 'WA'
GROUP BY company_name, state
```

company_name	max(sales_order.id)	state
Custom Designs	2547	WA
Its a Hit!	(NULL)	WA

サブクエリの使用

在庫が少ない製品の注文項目をリストするには、次のコマンドを入力します。

```
SELECT *
FROM sales_order_items
WHERE prod_id IN
 ( SELECT id
FROM product
WHERE quantity < 20 )
ORDER BY ship_date DESC
```

id	line_id	prod_id	quantity	ship_date
2082	1	401	48	1994-07-09
2053	1	401	60	1994-06-30
2125	2	401	36	1994-06-28
2027	1	401	12	1994-06-17
2062	1	401	36	1994-06-17

カッコで囲まれたフレーズが、この文のサブクエリです。

```
(  SELECT id
FROM product
WHERE quantity < 20 )
```

サブクエリを使用すると、検索を 1 回のクエリだけで実行できるようになります。このため、在庫が少ない製品のリストをクエリで検索し、さらにその製品の注文を別のクエリで検索する必要がなくなります。

このサブクエリは、製品テーブルの id カラムで WHERE 句の探索条件を満たすすべての値をリストします。

クエリの別の表現方法

受注した 10 枚のタンクトップが出荷され、タンクトップの数量カラムの値が 18 になった場合にどうなるかを考えてみます。サブクエリを使ったクエリは、ウールの帽子とタンクトップの両方のすべての注文をリストします。これに対して、最初に使用した文は次のように変更する必要があります。

```
SELECT *
FROM sales_order_items
WHERE prod_id IN ( 401, 300 )
ORDER BY ship_date DESC
```

サブクエリを使用するコマンドは、データベースのデータが変更されてもそのまま機能するように、改善されています。

サブクエリについては、次の点に注意してください。

- NOT EXISTS 述部を使用するクエリなど、ジョインの構築に問題がある場合もサブクエリが役立つことがあります。
- サブクエリが返せるのは 1 つのカラムだけです。
- サブクエリは、比較の引数、IN、または EXISTS 句としてのみ使用できます。
- 外部ジョインの ON 句の中に、サブクエリを使用することはできません。

第 3 章

クエリと削除の最適化

この章について

この章では、次のようなクエリと削除のパフォーマンスに関する推奨事項について説明します。

- 処理速度の速いクエリの構築
- クエリ・プランの使用
- クエリ処理オプションの設定
- 削除オペレーションの最適化

内容

クエリ構築のヒント

ここでは、クエリ構造を改良するためのヒントを示します。

- サブクエリを含むコマンド文をジョインとして構成することによって、実行速度を高めることができる場合があります。
- GROUP BY 句で複数のカラムをグループ化する場合、カラムに対応するユニークな値をもとに降順にカラムをリストします。これによって最適なクエリのパフォーマンスが実現されます。
- ジョイン・インデックスを使用すると、多くの場合、ジョイン・クエリはアドホック・ジョインより高速に実行されますが、より多くのディスク領域が必要となります。ただし、ジョイン・クエリがマルチテーブル・ジョイン・インデックスの最大のテーブルを参照しない場合は、アドホック・ジョインの方がジョイン・インデックスよりパフォーマンスが高くなります。
- 追加のカラムを使用して、頻繁に行う計算の結果を格納すると、パフォーマンスを向上させることができます。

UNION ALL での GROUP BY がクエリ・パフォーマンスに与える影響

パフォーマンスを向上させるために、非常に大きなテーブルを複数の小さなテーブルにセグメント化し、ビューで UNION ALL を使用してアクセスすることがよくあります。このようなビューを GROUP BY とともに使用する特定の非常に個別的なクエリでは、Sybase IQ オプティマイザがいくつかの GROUP BY 処理を UNION ALL の各分岐に挿入して、処理を並列に実行し、結果を結合することでパフォーマンスを向上させることができます。分割 GROUP BY と呼ばれるこの方法では、最上位レベルの GROUP BY で処理されるデータの量が減少し、その結果、クエリ処理時間が減少します。

パフォーマンスが向上するのは、UNION ALL で GROUP BY を使用する特定のクエリだけです。たとえば、次の簡単なクエリは分割 GROUP BY によってパフォーマンスが向上します。

```
CREATE VIEW vtable (v1 int, v2 char(4)) AS
SELECT a1, a2 FROM tableA
UNION ALL
SELECT b1, b2 FROM tableB;

SELECT COUNT(*), SUM(v1) FROM vtable GROUP BY v2;
```

このクエリを分析するときに、オプティマイザは先に tableA で COUNT(*) GROUP BY を実行し、tableB で COUNT(*) GROUP BY を実行した後、結果を最上位レベルの GROUP BY に渡します。最上位レベルの GROUP BY は、2 つの COUNT(*) の結果の SUM を実行し、最終的なクエリ結果を生成します。最上位レベルの GROUP BY の役割が変化していることに注意してください。最上位レベルの GROUP BY が使用している集合関数は COUNT ではなく SUM です。

分割 GROUP BY の制限

分割 GROUP BY によってパフォーマンスが向上する状況とクエリには、いくつかの制限があります。

- クエリで UNION ではなく UNION ALL を使用している場合に、分割 GROUP BY によってクエリのパフォーマンスが向上する可能性があります。次のクエリでは UNION で GROUP BY を使用しているため、分割 GROUP BY によるメリットはありません。

  ```
  CREATE VIEW viewA (va1 int, va2 int, va3 int,
  va4 int) AS
  SELECT b1, b2, b3, b4 FROM tableB
  UNION
  SELECT c1, c2, c3, c4 FROM tableC;

  SELECT SUM(va1) FROM viewA GROUP BY va3;
  ```

- クエリ内の集合関数で DISTINCT が指定されていない場合に、分割 GROUP BY によってクエリのパフォーマンスが向上する可能性があります。次のクエリでは SUM DISTINCT を使用しているため、分割 GROUP BY によるメリットはありません。

```
CREATE VIEW viewA (va1 int, va2 int, va3 int,
va4 int) AS
SELECT b1, b2, b3, b4 FROM tableB
UNION ALL
SELECT c1, c2, c3, c4 FROM tableC;

SELECT SUM(DISTINCT va1) FROM viewA GROUP BY va3;
```

- 分割 GROUP BY によってクエリのパフォーマンスを向上させるには、追加の GROUP BY 演算子の処理に使われる集合情報とデータを格納するために、テンポラリ共有バッファ・キャッシュに十分なメモリが必要です。

```
CREATE VIEW viewA (va1 int, va2 int, va3 int,
va4 int) AS
SELECT b1, b2, b3, b4 FROM tableB
UNION ALL
SELECT c1, c2, c3, c4 FROM tableC
UNION ALL
SELECT d1, d2, d3, d4 FROM tableD
UNION ALL
SELECT e1, e2, e3, e4 FROM tableE
UNION ALL
SELECT f1, f2, f3, f4 FROM tableF
UNION ALL
SELECT g1, g2, g3, g4 FROM tableG;

SELECT SUM(va1) FROM viewA GROUP BY va3;
```

この例では、Sybase IQ オプティマイザが GROUP BY を分割し、6 個の GROUP BY 演算子をクエリ・プランに挿入しています。これにより、集合情報とデータを格納するために、クエリにより多くのテンポラリ・キャッシュが必要となります。システムが十分なキャッシュを割り付けられない場合、オプティマイザは GROUP BY を分割しません。

メモリに空きがある場合は、TEMP_CACHE_MEMORY_MB データベース・オプションを使用してテンポラリ・キャッシュのサイズを増やすことができます。バッファ・キャッシュのサイズの設定方法については、『Sybase IQ リファレンス・マニュアル』の「データベース・オプション」の「バッファ・キャッシュ・サイズの決定」(107 ページ) および「TEMP_CACHE_MEMORY_MB オプション」を参照してください。

- 分割 GROUP BY によってクエリのパフォーマンスを向上させるには、AGGREGATION_PREFERENCE データベース・オプションをデフォルト値の 0 に設定します。これにより、Sybase IQ オプティマイザは GROUP BY に適用する最善のアルゴリズムを判断できるようになります。Sybase IQ オプティマイザが GROUP BY の処理にソート・アルゴリズムを選択するように AGGREGATION_PREFERENCE の値が設定されている場合は、分割 GROUP BY によるメリットはありません。AGGREGATION_PREFERENCE オプションを使用すると、オプティマイザが GROUP BY の処理に選択するアルゴリズムを上書きできます。分割 GROUP BY では、この値を 1 または 2 に設定しないでください。

分割 GROUP BY の例

次の例では、tableA という大きなテーブルを、tabA1、tabA2、tabA3、tabA4 という 4 つの小さなテーブルにセグメント化しています。この 4 つの小さなテーブルと UNION ALL を使用して、unionTab ビューを作成します。

```
CREATE VIEW unionTab (v1 int, v2 int, v3 int, v4 int) AS
SELECT a, b, c, d FROM tabA1
UNION ALL
SELECT a, b, c, d FROM tabA2
UNION ALL
SELECT a, b, c, d FROM tabA3
UNION ALL
SELECT a, b, c, d FROM tabA4;
```

Sybase IQ オプティマイザは GROUP BY の処理を次のクエリに分割し、クエリのパフォーマンスを向上させます。

```
SELECT v1, v2, SUM(v3), COUNT(*) FROM unionTab
GROUP BY v1, v2;

SELECT v3, SUM(v1*v2) FROM unionTab
GROUP BY v3;
```

Adaptive Server Anywhere による処理を引き起こす条件

Sybase IQ アーキテクチャには、Adaptive Server Anywhere のルールに従ってクエリを処理する製品の部分が含まれています。CIS (以前は OMNI) 機能補正と呼ばれるこの機能を使用すると、Sybase IQ のセマンティックで直接サポートされないクエリを処理できますが、パフォーマンスが大幅に低下します。

CIS は次のクエリを傍受します。

- ユーザ定義関数を参照するクエリ
- データベース間のジョインまたはプロキシ・テーブルを含むクエリ
- 特定のシステム関数を含むクエリ
- カタログ・ストア・テーブルまたは SYSTEM dbspace で作成されたテーブルを参照するクエリ

Sybase IQ と Adaptive Server Anywhere の違いの詳細については、『Sybase IQ リファレンス・マニュアル』の「付録 A 他の Sybase データベースとの互換性」を参照してください。

クエリ・プラン

最も効果的な構文を使用していなくても、正しいインデックスを作成していれば、通常は Sybase IQ クエリ・オプティマイザによって、最も効率的な方法でクエリを実行できます。もちろん、クエリを正しく設計することは重要です。クエリを計画する場合に、クエリの実行速度と得られる結果の正確さが主要な問題点となります。

クエリを実行する前に、Sybase IQ クエリ・オプティマイザはクエリ・プランを作成します。Sybase IQ では、これ以降の項で説明するオプションを使用して、クエリ・プランを調査および変更し、クエリを評価できます。このオプションを指定する方法の詳細については、『Sybase IQ リファレンス・マニュアル』を参照してください。

注意　整数値を指定できるデータベース・オプションでは、小数の *option-value* の設定が常に整数値にトランケートされます。たとえば、3.8 という値は 3 にトランケートされます。

クエリ評価オプション

次のオプションは、クエリ・プランの評価に役立ちます。これらのオプションの詳細については、『Sybase IQ リファレンス・マニュアル』を参照してください。

- INDEX_ADVISOR － このオプションを ON に設定すると、インデックス・アドバイザは、Sybase IQ クエリ・プランの一部として、またクエリ・プランが無効の場合には、Sybase IQ メッセージ・ログ・ファイル内の独立したメッセージとして、インデックスの推奨を出力します。これらのメッセージは、“Index Advisor” という文字列で始まります。この文字列を検索することで、Sybase IQ メッセージ・ファイルからこれらのメッセージをフィルタできます。このオプションはメッセージを OWNER.TABLE.COLUMN 形式で出力します。このオプションのデフォルト設定は OFF です。

  『Sybase IQ リファレンス・マニュアル』の「sp_iqindexadvice プロシージャ」も参照してください。

- INDEX_ADVISOR_MAX_ROWS － このオプションはインデックス・アドバイザによって格納されるメッセージの数を制限します。指定された制限値に達すると、INDEX_ADVISOR は新しいアドバイスの保存を停止しますが、既存のアドバイスのカウントとタイムスタンプの更新は続行します。

- NOEXEC − このオプションを ON に設定すると、Sybase IQ はクエリ・プランを生成しますが、クエリを実行しません。ただし、EARLY_PREDICATE_EXECUTION オプションが ON の場合を除きます。
- QUERY_DETAIL − このオプションと、QUERY_PLAN または QUERY_PLAN_AS_HTML の両方が ON の場合、Sybase IQ はクエリ・プランを生成するときに、クエリについての追加情報を表示します。QUERY_PLAN と QUERY_PLAN_AS_HTML が OFF の場合、このオプションは無視されます。
- QUERY_PLAN − このオプションが ON に設定されている場合 (デフォルト)、Sybase IQ はクエリについてのメッセージを生成します。ジョイン・インデックスの使用方法、ジョイン順序、クエリのジョイン・アルゴリズムについてのメッセージなどが生成されます。
- QUERY_PLAN_AFTER_RUN − このオプションを ON に設定すると、クエリの実行が終了した後でクエリ・プランが出力されます。これにより、クエリの各ノードから渡された実際のローの数など、追加情報をプランに含めることができます。このオプションを使用するには、QUERY_PLAN を ON にします。このオプションは、デフォルトでは OFF になっています。
- QUERY_PLAN_AS_HTML − このオプションは、Web ブラウザで表示できるように、HTML 形式のグラフィカルなクエリ・プランを生成します。HTML 形式では、ノード間にハイパーリンクが設定されるため、*.iqmsg* ファイルのテキスト形式よりはるかに使いやすくなります。クエリ・プランのファイル名にクエリ名を含めるには、QUERY_NAME オプションを使用します。このオプションは、デフォルトでは OFF になっています。
- QUERY_PLAN_AS_HTML_DIRECTORY − このオプションを ON に設定し、QUERY_PLAN_AS_HTML_DIRECTORY でディレクトリが指定されている場合、Sybase IQ は指定されたディレクトリに HTML クエリ・プランを書き込みます。
- QUERY_TIMING − このオプションは、サブクエリのタイミング統計の収集などのクエリ・エンジンの反復的な機能を制御するのに使用します。非常に短い相関サブクエリの場合、各サブクエリを実行するタイミングを合わせる処理のために、全体のパフォーマンスが大幅に低下するため、このオプションは、通常、OFF (デフォルト) にします。

注意 クエリ・プランを生成すると、*.iqmsg* ファイルに大量のテキストが追加される場合があります。QUERY_PLAN が ON の場合で、特に QUERY_DETAIL が ON の場合は、IQMSG_LENGTH_MB を正の値に設定し、メッセージ・ログの循環を有効にすることをおすすめします。

クエリ・ツリー

オプティマイザは、クエリ内のデータの流れを表すクエリ「ツリー」を作成します。クエリ・プランでは、クエリ・ツリーが *.iqmsg* ファイル内にテキスト形式で表示されます。オプションで、グラフィカル形式のクエリ・ツリーも作成できます。

クエリ・ツリーはノードで構成されます。それぞれのノードは処理の段階を表します。ツリーの一番下のノードはリーフ・ノードです。各リーフ・ノードは、クエリ内のテーブルまたはプリジョイン・インデックス・セットを表します。

プランの最上部にあるのは、演算子ツリーのルートです。情報はテーブルから上方向に、ジョイン、ソート、フィルタ、格納、集合、サブクエリを表す演算子を通じて流れます。

HTML クエリ・プランの使用

クエリ・プランを初めて使用するときは、QUERY_PLAN_AS_HTML オプションを ON に設定することをおすすめします。このオプションを設定すると、*.iqmsg* ファイルと同じディレクトリにグラフィカル版のクエリ・プランが作成されます。このファイルは、ほとんどの Web ブラウザで表示できます。

HTML クエリ・プランでは、ツリーの各ノードが詳細へのハイパーリンクになっています。各ボックスが上位のツリーへハイパーリンクされています。任意のノードをクリックし、プラン内をすばやく移動できます。

クエリ処理の制御

すべてのユーザが、特定のクエリの処理にかかる時間に制限を設定できます。DBA 権限を持つユーザは、特定のユーザのクエリに他のクエリより高い優先度を与えることや、処理のアルゴリズムを変更し、クエリ処理の速度を操作することができます。この項で説明するオプションの詳細については、『Sybase IQ リファレンス・マニュアル』を参照してください。

クエリの時間制限の設定

MAX_QUERY_TIME オプションを設定すると、ユーザは長い時間がかかるクエリを禁止できます。指定した時間よりクエリの実行時間が長くかかった場合、Sybase IQ は適切なエラーを表示してクエリを停止します。

注意　Sybase IQ では、小数の *option-value* の設定がすべて整数値にトランケートされます。たとえば、3.8 という値は 3 にトランケートされます。

クエリの優先度の設定

処理をキューで待機しているクエリは、そのクエリを送信したユーザの優先度、そしてクエリが送信された順序の順に実行されます。優先度の高いクエリがすべて実行されるまで、優先度の低いキューのクエリは実行されません。

次のオプションは、クエリにユーザ別の処理の優先度を割り当てます。

- IQGOVERN_PRIORITY － 処理キューで待機しているクエリに数字の優先度 (1、2、または 3 で、1 が最も高い) を割り当てます。
- IQGOVERN_MAX_PRIORITY － DBA はユーザまたはグループの IQGOVERN_PRIORITY に上限値を設定できます。
- IQ_GOVERN_PRIORITY_TIME － 優先度の高い (優先度 1 の) クエリが、指定した時間より長く -iqgovern キューで待機している場合に、優先度の高いユーザを開始できます。

クエリの優先度を調べるには、sp_iqcontext ストアド・プロシージャによって返される IQGovernPriority 属性を確認します。

クエリ最適化オプションの設定

次のオプションは、クエリの処理速度に影響を与えます。

- AGGREGATION_PREFERENCE － 集合 (GROUP BY、DISTINCT、SET の各関数) を処理するためのアルゴリズムの選択を制御しますこのオプションは、主に内部用として設計されているため、経験のあるデータベース管理者のみが使用してください。
- DEFAULT_HAVING_SELECTIVITY － クエリ内のすべての HAVING 述部の選択性を設定します。これが、HAVING 句によってフィルタされるロー数についてのオプティマイザの見積もりに優先して使用されます。
- DEFAULT_LIKE_MATCH_SELECTIVITY － `LIKE 'string%string'` (% はワイルドカード文字) などの、汎用 LIKE 述部のデフォルトの選択性を設定します。他の選択性情報が利用できず、照合文字列が一連の定数文字と 1 つのワイルドカードで始まっていない場合、オプティマイザはこのオプションを利用します。
- DEFAULT_LIKE_RANGE_SELECTIVITY － 照合文字列が一連の定数文字と 1 つのワイルドカード文字 (%) でできている `LIKE 'string%'` 形式の先行定数 LIKE 述部のデフォルトの選択性を設定します。他の選択性情報が利用できない場合、オプティマイザはこのオプションを利用します。
- EARLY_PREDICATE_EXECUTION － ジョインの最適化の前に簡単なローカル述部を実行するかどうかを制御します。通常は、このオプションを変更しないでください。

- ENABLED_ORDERED_PUSHDOWN_INSERTION － クエリ・オプティマイザが、ジョイン・オプティマイザによって選択されたプッシュダウン・ジョイン用のセミジョイン述部に追加する方法を制御します。それらのセミジョインによって間接的に影響を受ける可能性のある中間のジョインを再分析します。通常は、このオプションを変更しないでください。
- IN_SUBQUERY_PREFERENCE － IN サブクエリを処理するためのアルゴリズムの選択を制御します。このオプションは、主に内部用として設計されているため、経験のあるデータベース管理者のみが使用してください。
- INDEX_PREFERENCE － クエリ処理に使用するインデックスを設定します。Sybase IQ オプティマイザは、通常最適なインデックスを使用して、ローカルな WHERE 句 の述部など、1 つの IQ インデックスの範囲内で処理できる操作を実行します。このオプションは、テスト目的にオプティマイザの選択を無効にするために使用します。通常の使用の際はこのオプションの値を変更しないでください。
- JOIN_PREFERENCE － ジョインを処理するときのアルゴリズムの選択を制御します。このオプションは、主に内部用として設計されているため、経験のあるデータベース管理者のみが使用してください。
- JOIN_SIMPLIFICATION_THRESHOLD － ジョイン・オプティマイザの単純化が適用される前にジョインされるテーブルの最小数を制御します。通常は、この値を変更する必要はありません。
- MAX_HASH_ROWS － クエリ・オプティマイザがハッシュ・アルゴリズムを使用するときに考慮する最大ロー数の推測値を設定します。デフォルトは、1,250,000 のローです。たとえば、2 つのテーブル間にジョインがあり、両方のテーブルからジョインに入力されるロー数がこのオプションで設定された値を超えると、オプティマイザはハッシュ・ジョインを選択肢から外します。TEMP_CACHE_MEMORY_MB がユーザあたり 50 MB を超えるシステムの場合は、このオプションにさらに大きな値を設定します。
- MAX_JOIN_ENUMERATION － オプティマイザの単純化が適用された後で、ジョイン順のために最適化するテーブルの最大数を設定します。通常は、このオプションを設定する必要はありません。

述部ヒントの設定

Sybase IQ は、選択性、有用性、インデックス設定、実行モードなどの述部単位のヒントを指定できるヒント文字列をサポートします。

選択性は他の 3 つのクエリ最適化と組み合わせて設定できます。

- インデックス設定オプションに相当する機能の設定
- 有用性の設定 (述部の順序付け)
- 1 つ以上の述部の遅延

通常の状況では、評価を遅らせることにメリットはなく、クエリの処理が遅くなるだけです。ただし、これによって次の 4 つの動作のいずれかをクエリ内のもっと後ろに移動できます。

- 最適化の前
- 初回の「最初のフェッチ」時
- 2 回目の「最初のフェッチ」時 (相関サブクエリの内部またはネストループ・プッシュダウン・ジョインの左側のみ)
- インデックスの不使用 (水平処理)

構文、パラメータ、使用例については、『Sybase IQ リファレンス・マニュアル』の「第 3 章 SQL 言語の要素」の「ユーザ指定の条件ヒント文字列」を参照してください。

削除オペレーションの最適化

Sybase IQ は、削除オペレーションを処理するために次の 3 つのアルゴリズムから 1 つを選択します。

- **スモール・デリート**

  スモール・デリートでは、非常に少数のグループからローを削除するときに最適なパフォーマンスが得られます。通常は、削除するローが 1 つだけか、HG (High_Group) インデックスを持つカラムに等号述部がある場合に選択されます。スモール・デリート・アルゴリズムは、HG にランダムにアクセスできます。最悪の場合、I/O はアクセスされるグループの数に比例します。

- **ミッド・デリート**

  ミッド・デリートでは、いくつかのグループからローを削除するときに最適なパフォーマンスが得られます。ただし、それらのグループが十分に分散されているか、十分に少なくて、あまり多くの HG ページがアクセスされないことが条件です。ミッド・デリート・アルゴリズムは、HG への順序付けられたアクセスを提供します。最悪の場合、I/O はインデックス・ページ数によって制限されます。ミッド・デリートは、削除するレコードのソートという追加的なコストを伴います。

- **ラージ・デリート**

  ラージ・デリートでは、多数のグループからローを削除するときに最適なパフォーマンスが得られます。ラージ・デリートでは、すべてのローが削除されるまで HG が順番にスキャンされます。最悪の場合、I/O はインデックス・ページ数によって制限されます。ラージ・デリートは並列処理ですが、並列処理はインデックスの内部構造および削除対象のグループの分散度によって制限されます。HG カラムの範囲述部を使用して、ラージ・デリートのスキャン範囲を狭めることができます。

削除コスト

12.6 より前の HG 削除コスト・モデルでは、最悪の場合の I/O パフォーマンスだけが考慮されていたため、たいていラージ・デリートが優先的に使用されていました。現在のコスト・モデルでは、I/O コスト、CPU コスト、使用可能なリソース、インデックス・メタデータ、並列度、クエリから使用できる述部など、多数の要素が考慮されます。

HG インデックスを持つカラムの述部を指定すると、コストが大幅に改善されます。HG コスト計算でラージ・デリート以外のアルゴリズムを選択するためには、削除によって影響を受ける重複しない個別の値の数を判定できる必要があります。個別カウント数は、初めはインデックス・グループの数および削除されるローの数より少ないものと見なされます。述部は個別カウント数の改善された見積もりや、正確な見積もりでさえも提供できます。

現在のコスト計算では、ラージ・デリートにおける範囲述部の効果を考慮していません。そのため、ラージ・デリートのほうが速いケースでミッド・デリートが選択されることもあります。そうしたケースでは、必要に応じて強制的にラージ・デリート・アルゴリズムを適用できます。これについては、次の項で説明します。

削除パフォーマンス・オプションの使用

HG_DELETE_METHOD オプションを使用すると、HG 削除パフォーマンスを制御できます。

HG_DELETE_METHOD オプションでは、指定した削除アルゴリズムを強制的に適用できます。

- 1 = スモール・デリート
- 2 = ラージ・デリート
- 3 = ミッド・デリート

第 4 章

OLAP の使用

この章について

オンライン分析処理 (OLAP: Online Analytical Processing) は、リレーショナル・データベースに格納されている情報を効率的に分析するための手法です。OLAP を使用すると、データをさまざまな次元で分析し、小計ローを含んだ結果セットを取得し、データを多次元キューブに編成するという処理をすべて 1 つの SQL クエリで行うことができます。また、フィルタを使用してデータを絞り込み、結果セットを迅速に返すことができます。この章では、Sybase IQ がサポートする SQL/OLAP 関数について説明します。

注意 以降で紹介する OLAP の例に出てくるテーブルは、asiqdemo データベースに含まれています。

内容

OLAP について

1999 年の SQL 標準の改正によって、ANSI SQL 標準に複雑なデータ分析操作を行うための拡張機能が導入されました。Sybase IQ では、以前のリリースでこれらの SQL 拡張機能の一部が取り入れられていますが、Sybase IQ 12.7 では、これらの拡張機能が包括的にサポートされています。

この分析機能を使って複雑なデータ分析を 1 つの SQL 文で実行することができますが、これはオンライン分析処理 (OLAP) と呼ばれるソフトウェア・テクノロジに基づいています。OLAP の関数には、GROUP BY 句の拡張機能や、次のような統計関数が含まれています。

- GROUP BY 句の拡張機能 − CUBE、ROLLUP
- 統計関数
  - 単純な集合 − AVG、COUNT、MAX、MIN、SUM、STDDEV、VARIANCE

    注意 Grouping() 以外の単純な集合関数はすべて OLAP ウィンドウ関数と併用できます。

  - ウィンドウ関数
    - ウィンドウでの集合 − AVG、COUNT、MAX, MIN、SUM
    - ランク付け関数 − RANK、DENSE_RANK、PERCENT_RANK、NTILE
    - 統計関数 − STDDEV、STDDEV_SAMP、STDDEV_POP、VARIANCE、VAR_SAMP、VAR_POP
    - 分散統計関数 − PERCENTILE_CONT、PERCENTILE_DISC
  - 数値関数 − WIDTH_BUCKET、CEIL、LN、EXP、POWER、SQRT、FLOOR

データベース製品によっては、OLAP モジュールが独立しており、分析前にデータをデータベースから OLAP モジュールに移動しなければならないものもあります。一方、Sybase IQ では OLAP 機能がデータベースそのものに組み込まれているため、ストアド・プロシージャなどのデータベース機能との配備や統合を簡単かつシームレスに行うことができます。

OLAP の利点

OLAP 関数を GROUPING、CUBE、ROLLUP という拡張機能と組み合わせて使用すると、2 つの大きな利点があります。第一に、多次元のデータ分析、データ・マイニング、時系列分析、傾向分析、コストの割り当て、ゴール・シーク、一時的な多次元構造変更、非手続き型モデリング、例外の警告を多くの場合 1 つの SQL 文で実行できます。第二に、OLAP のウィンドウおよびレポート集合関数では、**ウィンドウ**という関係演算子を使用することができ、これはセルフジョインや相関サブクエリを使用するセマンティック的に等価なクエリよりも効率的に実行できます。OLAP を使用して取得した結果セットには小計ローを含めることができ、この結果セットを多次元キューブに編成することもできます。詳細については、「ウィンドウ」(62 ページ) を参照してください。

さまざまな期間での移動平均と移動和を計算したり、選択したカラムの値が変化したときに集計とランクをリセットしたり、複雑な比率を単純な言葉で表現したりできます。1 つのクエリ式のスコープ内で、それぞれ独自のパーティショニング・ルールを持ついくつかの異なる OLAP 関数を定義することができます。

OLAP の評価について

OLAP の評価は、最終的な結果に影響を及ぼすクエリ実行のいくつかのフェーズとして概念化できます。OLAP の実行フェーズは、クエリ内の対応する句によって識別されます。たとえば、SQL クエリの指定にウィンドウ関数が含まれている場合は、WHERE、JOIN、GROUP BY、HAVING 句が先に処理されます。GROUP BY 句でグループが定義された後、クエリの ORDER BY 句に含まれる最後の SELECT リストが評価される前に、パーティションが作成されます。

グループ化の際には、NULL 値はすべて同じグループと見なされます (それぞれの NULL 値が等しくない場合でも同様です)。

HAVING 句は、WHERE 句に似ており、GROUP BY 句の結果に対するフィルタとして機能します。

ANSI SQL 標準に基づく SQL 文と SELECT、FROM、WHERE、GROUP BY、HAVING 句を含んだ単純なクエリ仕様のセマンティックを考えてみます。

1　クエリにより、FROM 句のテーブル式を満たすロー・セットが取得されます。

2　WHERE 句の述部が、テーブルから取得したロー・セットに適用されます。WHERE 句の条件を満たさない (条件が true にならない) ローが除外されます。

3　残りの各ローについて、SELECT リストおよび GROUP BY 句に含まれている式 (集合関数を除く) が評価されます。

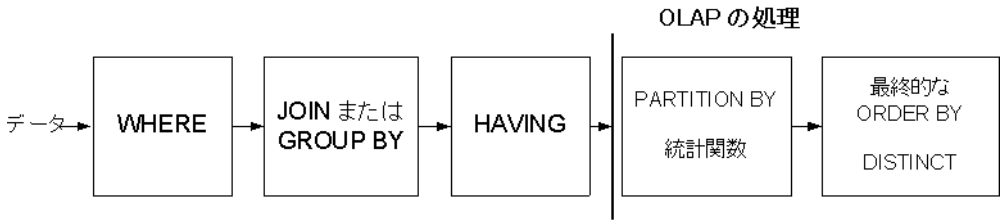
4 GROUP BY 句の式の重複しない値に基づいて、結果のローがグループ化されます (NULL はそれぞれのドメインで特殊な値として扱われます)。PARTITION BY 句がある場合は、GROUP BY 句の式はパーティション・キーとして使用されます。

5 各パーティションについて、SELECT リストまたは HAVING 句の集合関数が評価されます。いったん集合関数を適用すると、中間の結果セットには個々のテーブル・ローが含まれなくなります。新しい結果セットには、GROUP BY の式と、各パーティションについて計算した集合関数の値が含まれます。

6 HAVING 句の条件が結果グループに適用されます。HAVING 句の条件を満たさないグループが除外されます。

7 PARTITION BY 句で定義された境界に基づいて結果が分割されます。結果ウィンドウについて、OLAP ウィンドウ関数 (ランク付け関数および集合関数) が計算されます。

図 4-1: 実行のセマンティック・フェーズ



詳細については、「文法規則 2」(98 ページ) を参照してください。OLAP 構文の詳細については、「OLAP 関数の BNF 文法」(98 ページ) も参照してください。

GROUP BY 句の拡張機能

GROUP BY 句の拡張機能により、次のような処理を行う複雑な SQL 文を書くことができます。

- 入力ローを複数の次元に分割し、結果グループの複数のサブセットを組み合わせる。
- “ データ・キューブ ” を作成し、データ・マイニング分析のための疎密度の多次元結果セットを用意する。
- 元のグループを含んだ結果セットを作成する (必要に応じて、小計ローと合計ローを含める場合もある)。

ROLLUP や CUBE などの OLAP の Grouping() (グループ化) 操作は、プレフィクスや小計ローとして概念化できます。

プレフィクス

GROUP BY 句を含むクエリでは、**プレフィクス**のリストが作成されます。プレフィクスとは、GROUP BY 句の項目のサブセットであり、クエリの GROUP BY 句の項目のうち最も右にある 1 つまたは複数の項目を除外することで作成されます。残りのカラムは**プレフィクス・カラム**と呼ばれます。

ROLLUP 例 1　次に示す ROLLUP のクエリの例では、GROUP BY のリストに 2 つの変数 (*Year* と *Quarter*) が含まれています。

```
SELECT year (order_date) Year, quarter(order_date)
   Quarter, COUNT(*) Orders
FROM alt_sales_order
GROUP BY ROLLUP(Year, Quarter)
ORDER BY Year, Quarter
```

このクエリには次の 2 つのプレフィクスがあります。

- Quarter を除外するプレフィクス ― プレフィクス・カラムには 1 つのカラム (Year) が含まれます。
- Quarter と Year の両方を除外するプレフィクス ― プレフィクス・カラムは存在しません。

Quarter と Year の両方を除外するプレフィクス

Quarter を除外するプレフィクス

Year	Quarter	Orders
(NULL)	(NULL)	648
2000	(NULL)	380
2000	1	87
2000	2	77
2000	3	91
2000	4	125
2001	(NULL)	268
2001	1	139
2001	2	119
2001	3	10

注意　GROUP BY リストには、項目と同じ数のプレフィクスが含まれます。

GROUP BY での ROLLUP と CUBE

プレフィクスに関する一般的なグループ化を簡単に指定するために、2 つの重要な構文簡略化パターンが用意されています。1 つ目のパターンは ROLLUP、2 つ目のパターンは CUBE と呼ばれます。

GROUP BY ROLLUP

ROLLUP 演算子には、引数として適用するグループ化の式を、次の構文の中で順序リストで指定します。

SELECT … [**GROUPING** (*column-name*) …] …
GROUP BY [*expression* [, …]
| **ROLLUP** (*expression* [, …])]

GROUPING は、カラム名をパラメータとして受け取り、表 4-1 に示すようにブール値を返します。

表 4-1: ROLLUP 演算子が指定された GROUPING によって返される値

結果値の種類	GROUPING の戻り値
ROLLUP 演算子によって作成された NULL	1 (真)
ローが小計であることを示す NULL	1 (真)
ROLLUP 演算子によって作成された以外の NULL	0 (偽)
格納されていた NULL	0 (偽)

ROLLUP は、まず GROUP BY 句に指定された標準的な集合関数値を計算します。次に、ROLLUP はグループ化を行うカラムのリストを右から左に移動し、より高いレベルの小計を連続して作成します。最後に総計が作成されます。グループ化するカラムの数が *n* 個の場合、ROLLUP は *n*+1 レベルの小計を作成します。

SQL 構文の例	定義されるセット
`GROUP BY ROLLUP (A, B, C);`	(A, B, C) (A, B) (A) ( )

ROLLUP と小計ロー

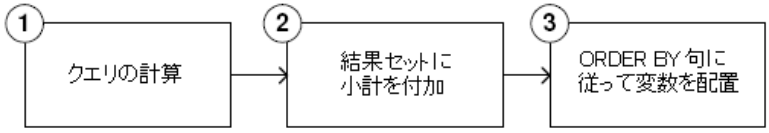
ROLLUP は、GROUP BY のクエリ・セットに対して UNION を行うのと同じことです。次の 2 つのクエリの結果セットは等しくなります。GROUP BY (A, B) の結果セットは、A と B に定数が含まれているすべてのローについての小計から成ります。UNION を可能にするために、カラム C には NULL が割り当てられます。

ROLLUP クエリの例	ROLLUP を使用せずに記述した同じ内容のクエリ
SELECT A, B, C, SUM(D) FROM T1 GROUP BY ROLLUP (A, B, C);	SELECT * FROM ((SELECT A, B, C, SUM(D) GROUP BY A, B, C) UNION ALL (SELECT A, B, NULL, SUM(D) GROUP BY A, B) UNION ALL (SELECT A, NULL, NULL, SUM(D) GROUP BY A) UNION ALL (SELECT NULL, NULL, NULL, SUM(D)))

小計ローはデータの分析に役立ちます。特に、データが大量にある場合、データにさまざまな次元がある場合、データがさまざまなテーブルに含まれている場合、あるいはまったく異なるデータベースに含まれている場合に威力を発揮します。たとえば販売マネージャが、売上高についてのレポートを営業担当者別、地域別、四半期別に整理して、売上パターンの理解に役立てることができます。データの小計は、販売マネージャが売上高の全体像をさまざまな視点から分析するのに役立ちます。販売マネージャが比較したいと考える基準に基づいて要約情報が提供されていれば、データの分析を容易に行うことができます。

OLAP を使用すると、ローおよびカラムの小計を分析、計算する処理をユーザの目から隠すことができます。図 4-2 に、Sybase IQ での小計の計算の概念を示します。

図 4-2: 小計



1　このステップで、まだ ROLLUP とは見なされない中間の結果セットが生成されます。

2　小計が評価され、結果セットに付加されます。

3　クエリ内の ORDER BY 句に従ってローが並べられます。

NULL 値と小計ロー

GROUP BY 操作に対する入力のローに NULL が含まれているときは、その中に、ROLLUP または CUBE 操作によって追加された小計ローと、最初の入力データの一部として NULL 値を含んでいるローが混在している可能性があります。

Grouping() 関数は、小計ローをその他のローから区別します。具体的には、GROUP BY リストのカラムを引数として受け取り、そのカラムが小計ローであるために NULL になっている場合は 1 を返し、それ以外の場合は 0 を返します。

次の例では、結果セットの中に Grouping() カラムが含まれています。強調表示されているローは、小計ローであるために NULL を含んでいるのではなく、入力データの結果として NULL を含んでいるローです。Grouping() カラムは強調表示されています。このクエリは、employee テーブルと sales_order テーブルの間の外部ジョインです。このクエリでは、テキサス、ニューヨーク、またはカリフォルニアに住んでいる女性従業員を選択しています。営業担当者でない (したがって売上がない) 女性従業員については、カラムに NULL が表示されます。

```
SELECT employee.emp_id AS Employee, year(order_date) AS
   Year, COUNT(*) AS Orders, GROUPING(Employee) AS
   GE, GROUPING(Year) AS GY
   FROM employee LEFT OUTER JOIN alt_sales_order ON
   employee.emp_id = alt_sales_order.sales_rep
   WHERE employee.sex IN ('F') AND employee.state
   IN ('TX', 'CA', 'NY')
GROUP BY ROLLUP (Year, Employee)
ORDER BY Year, Employee
```

このクエリの結果セットを次に示します。

```
emp_id      year      Orders     GY     GE
------      ----      ------     --     --
NULL        NULL           1      1      0
NULL        NULL         165      1      1
1090        NULL           1      0      0
NULL        2000          98      1      0
 667        2000          34      0      0
 949        2000          31      0      0
1142        2000          33      0      0
NULL        2001          66      1      0
 667        2001          20      0      0
949         2001          22      0      0
1142        2001          24      0      0
```

個々のプレフィクスについて、プレフィクス・カラムに同じ値が含まれているすべてのローに関する**小計ロー**が作成されます。

ROLLUP の結果を具体的に説明するために、前述のクエリの例をもう一度詳しく見ていきます。

```
SELECT year (order_date) AS Year, quarter
   (order_date) AS Quarter, COUNT (*) Orders
FROM sales_order
   GROUP BY ROLLUP (Year, Quarter)
   ORDER BY Year, Quarter
```

このクエリでは、Year カラムを含んでいるプレフィクスにより、Year=2000 の合計ローと Year=2001 の合計ローが作成されます。このプレフィクスに関する 1 つの合計ローはカラムを含んでいません。これは、中間の結果セットに含まれているすべてのローの小計です。

小計ローの各カラムの値は、次のようになっています。

- プレフィクスに含まれているカラム － そのカラムの値です。たとえば前述のクエリでは、Year=2000 のローに関する小計ローの Year カラムの値は 2000 になります。
- プレフィクスから除外されたカラム － NULL です。たとえば、Year カラムから成るプレフィクスにより生成された小計ローでは、Quarter カラムの値は NULL になります。
- 集合関数 － 除外されているカラムの値を計算した結果です。

  小計値は、集計されたローではなく基本データのローに対して計算されます。多くの場合、たとえば SUM や COUNT などでは結果は等しくなりますが、AVG、STDDEV、VARIANCE などの統計関数では結果が異なってくるため、この区別は重要です。

ROLLUP 演算子には次の制限があります。

- ROLLUP 演算子は、COUNT DISTINCT と SUM DISTINCT を除き、GROUP BY 句で使用可能なすべての集合関数をサポートしています。
- ROLLUP は SELECT 文でのみ使用できます。サブクエリでは ROLLUP を使用できません。
- 1 つの GROUP BY 句の中で複数の ROLLUP、CUBE、および GROUP BY カラムを組み合わせるグループ化の指定は、現時点ではサポートされていません。
- GROUP BY のキーに定数式を指定することはできません。

式の一般的なフォーマットについては、『Sybase IQ リファレンス・マニュアル』の「式」と「SQL 言語の要素」を参照してください。

ROLLUP 例 2　次は、ROLLUP と GROUPING の使用例です。GROUPING によって作成される一連のマスク・カラムを表示します。カラム S、N、C に表示されている数字 0 と 1 は、GROUPING からの戻り値で ROLLUP の結果の値を表現しています。マスクが “011” であれば小計のローであり、“111” であれば総計のローであると特定できます。これを利用して、クエリの結果をプログラムで分析することが可能です。

```
SELECT size, name, color, SUM(quantity),
   GROUPING(size) AS S,
   GROUPING(name) AS N,
   GROUPING(color) AS C
FROM product
GROUP BY ROLLUP(size, name, color) HAVING (S=1 or N=1 or C=1)
ORDER BY size, name, color;
```

このクエリの結果セットを次に示します。

```
size          name          color     SUM  S  N  C
----          -----         ------    ---  -  -  -
(NULL)        (NULL)        (NULL)    496  1  1  1
```

```
Large               (NULL)          (NULL)      71  0  1  1
Large               Sweatshirt      (NULL)      71  0  0  1
Medium              (NULL)          (NULL)     134  0  1  1
Medium              Shorts          (NULL)      80  0  0  1
Medium              Tee Shirt       (NULL)      54  0  0  1
One size fits all   (NULL)          (NULL)     263  0  1  1
One size fits all   Baseball Cap     (NULL)     124  0  0  1
One size fits all   Tee Shirt       (NULL)      75  0  0  1
One size fits all   Visor           (NULL)      64  0  0  1
Small               (NULL)          (NULL)      28  0  1  1
Small               Tee Shirt       (NULL)      28  0  1  1
```

ROLLUP 例 3 次の例は、GROUPING を使用して、最初から格納されていた NULL 値と ROLLUP 操作によって生成された “NULL” 値とを区別する方法を示しています。このクエリで指定されているとおり、最初から格納されていた NULL 値はカラム prod_id に [NULL] として表示され、ROLLUP によって生成された “NULL” 値はカラム PROD_IDS で ALL に置き換えられます。

```
SELECT year(ship_date) AS Year, prod_id, SUM(quantity)
   AS OSum, CASE WHEN GROUPING(Year) = 1 THEN 'ALL' ELSE
   CAST(Year AS char(8)) END, CASE WHEN
   GROUPING(prod_id) = 1 THEN 'ALL' ELSE CAST(prod_id
   as char(8)) END
FROM alt_sales_order_items
GROUP BY ROLLUP(Year, prod_id) HAVING OSum > 36
ORDER BY Year, prod_id;
```

このクエリの結果セットを次に示します。

```
ship_date     prod_id   SUM   SHIP_DATES   PROD_IDS
---------     -------   ---   ----------   --------
NULL           NULL   28359   ALL          ALL
2000           NULL   17642   2000         ALL
2000           300     1476   2000         300
2000           301     1440   2000         301
2000           302     1152   2000         302
2000           400     1946   2000         400
2000           401     1596   2000         401
2000           500     1704   2000         500
2000           501     1572   2000         501
2000           600     2124   2000         600
2000           601     1932   2000         601
2000           700     2700   2000         700
2001           NULL   10717   2001         ALL
2001           300      888   2001         300
2001           301      948   2001         301
2001           302      996   2001         302
2001           400     1332   2001         400
2001           401     1105   2001         401
2001           500      948   2001         500
```

```
2001          501      936   2001          501
2001          600      936   2001          600
2001          601      792   2001          601
2001          700     1836   2001          700
```

ROLLUP 例 4　次のクエリ例は、注文数を年別および四半期別に集計したデータを返します。

```
SELECT year(order_date) AS Year, quarter(order_date)
   AS Quarter, COUNT(*) AS Orders
   FROM alt_sales_order
GROUP BY ROLLUP(Year, Quarter)
ORDER BY Year, Quarter
```

次の図は、このクエリの結果を示しています。結果セット内の小計ローは強調表示されています。各小計ローでは、その小計の計算対象になったカラムに NULL 値が格納されています。

	Year	Quarter	Orders
1	**(NULL)**	**(NULL)**	**648**
2	**2000**	**(NULL)**	**380**
3	2000	1	87
	2000	2	77
	2000	3	91
	2000	4	235
2	**2001**	**(NULL)**	**268**
3	2001	1	139
	2001	2	119
	2001	3	10

ロー [1] は、両方の年 (2000 年および 2001 年) のすべての四半期の注文数の合計を示しています。このローは、Year カラムと Quarter カラムの両方が NULL であり、すべてのカラムがプレフィクスから除外されています。

注意　ROLLUP 操作によって返される結果セットには、集合カラムを除くすべてのカラムが NULL であるローが必ず 1 つ含まれています。このローは、集合関数に対する全カラムの要約を表しています。たとえば、集合関数として SUM を使用している場合は、このローはすべての値の総計を表します。

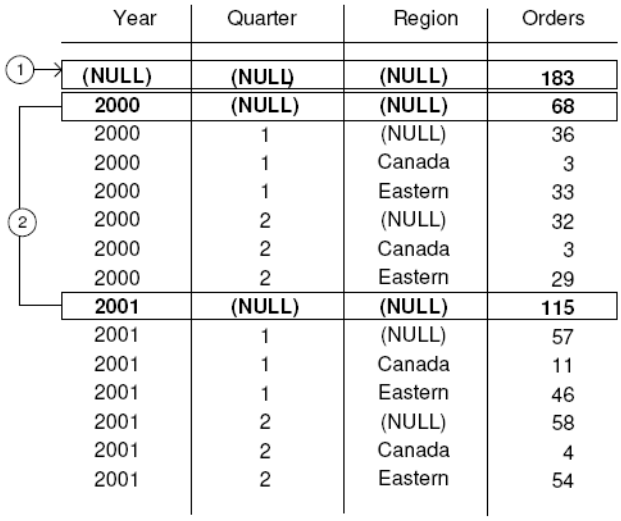
ロー [2] は、2000 年および 2001 年の注文数の合計をそれぞれ示しています。どちらのローも、Quarter カラムの値は NULL になっています。このカラムの値を加算して、Year の小計を出しているためです。結果セットにこのようなローがいくつ含まれるかは、ROLLUP クエリに登場する変数の数によって決まります。

[3] としてマークされている残りのローは要約情報を示し、それぞれの年の各四半期の注文数の合計を表しています。

ROLLUP 例 5 この ROLLUP 操作の例では、年別、四半期別、地域別の注文数を集計するというやや複雑な結果セットを返します。この例では、第 1 および第 2 四半期と 2 つの地域 (カナダと東部地区) だけを分析します。

```
SELECT year(order_date) AS Year, quarter(order_date)
   AS Quarter, region, COUNT(*) AS Orders
   FROM alt_sales_order WHERE region IN ('Canada',
   'Eastern') AND quarter IN (1, 2)
GROUP BY ROLLUP (Year, Quarter, Region)
ORDER BY Year, Quarter, Region
```

次の図は、このクエリの結果セットを示しています。各小計ローでは、その小計の計算対象になったカラムに NULL が格納されています。

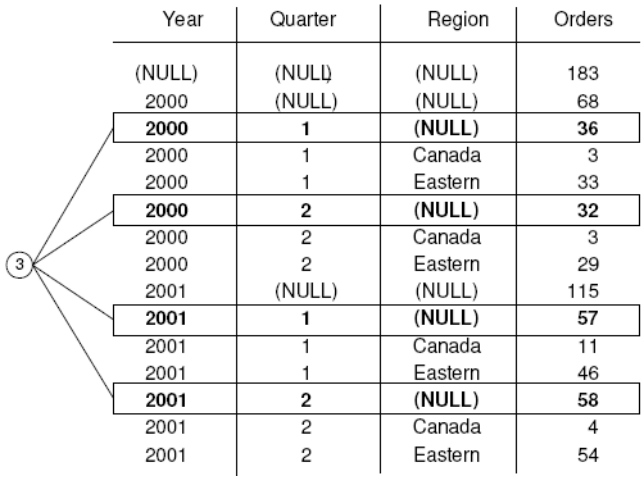


	Year	Quarter	Region	Orders
1	**(NULL)**	**(NULL)**	**(NULL)**	**183**
	2000	**(NULL)**	**(NULL)**	**68**
	2000	1	(NULL)	36
	2000	1	Canada	3
	2000	1	Eastern	33
2	2000	2	(NULL)	32
	2000	2	Canada	3
	2000	2	Eastern	29
	2001	**(NULL)**	**(NULL)**	**115**
	2001	1	(NULL)	57
	2001	1	Canada	11
	2001	1	Eastern	46
	2001	2	(NULL)	58
	2001	2	Canada	4
	2001	2	Eastern	54

ロー [1] はすべてのローの集計結果であり、Year、Quarter、Region カラムに NULL が含まれています。このローの Orders カラムの値は、カナダと東部地区の 2000 年および 2001 年の第 1 および第 2 四半期の注文数の合計を示しています。

[2] としてマークされているローは、それぞれの年 (2000 年と 2001 年) におけるカナダと東部地区の第 1 および第 2 四半期の注文数の合計を示しています。ロー [2] の値を足すと、ロー [1] に示されている総計に等しくなります。

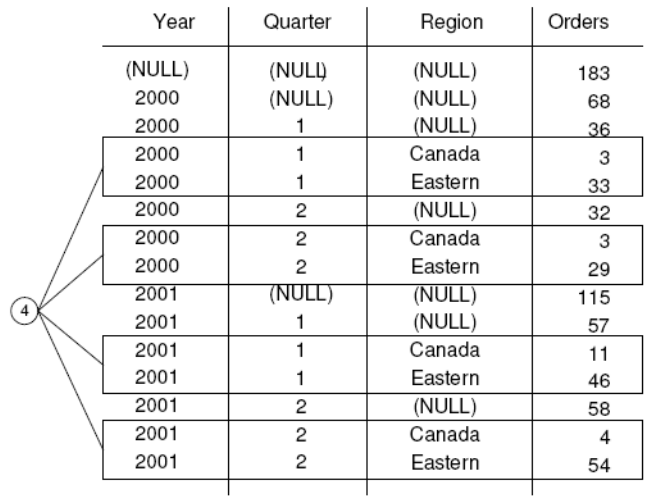
[3] としてマークされているローは、特定の年および四半期の全地域の注文数の合計を示しています。



Year	Quarter	Region	Orders
(NULL)	(NULL)	(NULL)	183
2000	(NULL)	(NULL)	68
2000	**1**	**(NULL)**	**36**
2000	1	Canada	3
2000	1	Eastern	33
2000	**2**	**(NULL)**	**32**
2000	2	Canada	3
2000	2	Eastern	29
2001	(NULL)	(NULL)	115
2001	**1**	**(NULL)**	**57**
2001	1	Canada	11
2001	1	Eastern	46
2001	**2**	**(NULL)**	**58**
2001	2	Canada	4
2001	2	Eastern	54

3

[4] としてマークされているローは、結果セット内のそれぞれの年の各四半期の各地域の注文の合計数を示しています。



Year	Quarter	Region	Orders
(NULL)	(NULL)	(NULL)	183
2000	(NULL)	(NULL)	68
2000	1	(NULL)	36
2000	1	Canada	3
2000	1	Eastern	33
2000	2	(NULL)	32
2000	2	Canada	3
2000	2	Eastern	29
2001	(NULL)	(NULL)	115
2001	1	(NULL)	57
2001	1	Canada	11
2001	1	Eastern	46
2001	2	(NULL)	58
2001	2	Canada	4
2001	2	Eastern	54

4

GROUP BY CUBE

GROUP BY 句の CUBE 演算子は、データを複数の次元 (グループ化の式) でグループ化することでデータを分析します。CUBE に次元の順序リストを引数として指定すると、SELECT 文の中で、そのクエリに指定した次元の考えられるすべての組み合わせの小計を計算し、選択した複数のカラムのすべての値の組み合わせについての要約を示す結果セットを生成することができます。

CUBE の構文は次のとおりです。

SELECT … [**GROUPING** (*column-name*) …] …
GROUP BY [*expression* [,…]
| **CUBE** (*expression* [,…])]

GROUPING は、カラム名をパラメータとして受け取り、表 4-2 に示すようにブール値を返します。

表 4-2: CUBE 演算子が指定された GROUPING によって返される値

結果値の種類	GROUPING の戻り値
CUBE 演算子によって作成された NULL	1 (真)
ローが小計であることを示す NULL	1 (真)
CUBE 演算子によって作成された以外の NULL	0 (偽)
格納されていた NULL	0 (偽)

CUBE は、同じ階層の一部ではない次元を扱うときに特に威力を発揮します。

SQL 構文の例	定義されるセット
`GROUP BY CUBE (A, B, C);`	(A, B, C) (A, B) (A, C) (A) (B, C) (B) (C) ( )

CUBE 演算子には次の制限があります。

- CUBE 演算子は GROUP BY 句で使用可能なすべての集合関数をサポートしますが、CUBE は現在 COUNT DISTINCT および SUM DISTINCT ではサポートされていません。
- CUBE は、現在、逆分散統計関数である PERCENTILE_CONT と PERCENTILE_DISC ではサポートされていません。
- CUBE は SELECT 文でのみ使用できます。CUBE を SELECT のサブクエリで使用することはできません。
- 1 つの GROUP BY 句の中で ROLLUP、CUBE、GROUP BY カラムを組み合わせる GROUPING の指定は、現時点ではサポートされていません。

- GROUP BY のキーに定数式を指定することはできません。

注意 キューブのサイズがテンポラリ・キャッシュのサイズを超えると、CUBE のパフォーマンスが低下します。

GROUPING と CUBE 演算子を併用すると、格納されていた NULL 値と CUBE によって作成されたクエリ結果の “NULL” 値を区別することができます。

GROUPING 関数を使用して結果を分析する方法については、ROLLUP 演算子の説明で紹介した例を参照してください。

CUBE 操作が返す結果セットには、集計カラムを除くすべてのカラムの値が NULL であるローが少なくとも 1 つは含まれています。このローは、集合関数に対する全カラムの要約を表しています。

CUBE 例 1　次の例は、対象者の州 (地理的な位置)、性別、教育レベル、および収入などで構成される調査データを使用したクエリです。最初に紹介するクエリには GROUP BY 句が指定されています。この句は、クエリの結果を census テーブルの state、gender、education カラムの値に応じてロー・グループに分類し、収入の平均とローの合計数をグループごとに計算します。このクエリには GROUP BY 句だけを使用し、ローのグループ化に CUBE 演算子を使用していません。

```
SELECT state, sex as gender, dept_id, COUNT(*),
   CAST(ROUND(AVG(salary),2) AS NUMERIC(18,2))
   AS average
FROM employee WHERE state IN ('MA' , 'CA')
GROUP BY state, sex, dept_id
ORDER BY 1,2;
```

このクエリの結果セットを次に示します。

```
state  gender  dept_id  count(*)  avg salary
-----  ------  -------  --------  ----------
CA       F       200       2         58650.00
CA       M       200       1         39300.00
MA       F       500       4         29950.00
MA       F       400       8         41959.88
MA       F       300       7         59685.71
MA       F       200       3         60451.00
MA       F       100       6         58243.42
MA       M       300       2         58850.00
MA       M       500       5         36793.96
MA       M       400       8         45321.47
MA       M       100      13         58563.59
MA       M       200       8         46810.63
```

GROUP BY 句の CUBE 拡張機能を使用すると、調査データを 1 回参照するだけで、調査データ全体における州別、性別、教育別の平均収入を計算し、state、gender、education カラムの考えられるすべての組み合わせにおける平均収入を計算することができます。CUBE 演算子を使用すると、たとえば、すべての州における全女性の平均収入を計算したり、調査対象者全員の平均収入を、各自の教育別および州別に計算したりすることができます。

CUBE でグループを計算するときには、計算されたグループのカラムに NULL 値が挿入されます。最初からデータベース内に格納されていた NULL なのか、CUBE の結果として生成された NULL なのかを区別するためには、GROUPING 関数を使用する必要があります。GROUPING 関数は、指定されたカラムが上位レベルのグループにマージされている場合は 1 を返します。

CUBE 例 2 次のクエリは、GROUP BY CUBE と GROUPING 関数を併用する方法を示しています。

```
SELECT case grouping(state) WHEN 1 THEN 'ALL' ELSE state
   END AS c_state, case grouping(sex) WHEN 1 THEN 'ALL'
   ELSE sex end AS c_gender, case grouping(dept_id)
   WHEN 1 THEN 'ALL' ELSE cast(dept_id as char(4)) end
   AS c_dept, COUNT(*), CAST(ROUND(AVG(salary),2) AS
   NUMERIC(18,2))AS AVERAGE
FROM employee WHERE state IN ('MA' , 'CA')
GROUP BY CUBE(state, sex, dept_id)
ORDER BY 1,2,3;
```

このクエリの結果は次のとおりです。クエリで指定されているとおり、小計ローを示すために CUBE によって生成された NULL は、小計ロー内で ALL に置き換えられています。

state	sex	dept_id	count	avg salary
ALL	ALL	100	19	58462.48
ALL	ALL	200	14	50888.43
ALL	ALL	300	9	59500.00
ALL	ALL	400	16	43640.67
ALL	ALL	500	9	33752.20
ALL	ALL	ALL	67	50160.38
ALL	F	100	6	58243.42
ALL	F	200	5	59730.60
ALL	F	300	7	59685.71
ALL	F	400	8	41959.88
ALL	F	500	4	29950.00
ALL	F	ALL	30	50713.08
ALL	M	100	13	58563.59
ALL	M	200	9	45976.11
ALL	M	300	2	58850.00
ALL	M	400	8	45321.47
ALL	M	500	5	36793.96
ALL	M	ALL	37	49712.25
CA	ALL	200	3	52200.00

```
CA          ALL       ALL           3        52200.00
CA          F         200           2        58650.00
CA          F         ALL           2        58650.00
CA          M         200           1        39300.00
CA          M         ALL           1        39300.00
MA          ALL       100          19        58462.48
MA          ALL       200          11        50530.73
MA          ALL       300           9        59500.00
MA          ALL       400          16        43640.67
MA          ALL       500           9        33752.20
MA          ALL       ALL          64        50064.78
MA          F         100           6        58243.42
MA          F         200           3        60451.00
MA          F         300           7        59685.71
MA          F         400           8        41959.88
MA          F         500           4        29950.00
MA          F         ALL          28        50146.16
MA          M         100          13        58563.59
MA          M         200           8        46810.63
MA          M         300           2        58850.00
MA          M         400           8        45321.47
MA          M         500           5        36793.96
MA          M         ALL          36        50001.48
```

CUBE 例 3　この例のクエリは、注文数の合計を要約する結果セットを返し、次に、年別および四半期別の注文数の小計を計算します。

注意　比較する変数の数が増えると、キューブの計算のコストが急激に増大します。

```
SELECT year(order_date) AS Year, quarter(order_date)
   AS Quarter, COUNT(*) AS Orders
FROM alt_sales_order
GROUP BY CUBE(Year, Quarter)
ORDER BY Year, Quarter
```

次の図は、このクエリの結果セットを示しています。この結果セットでは、小計ローが強調表示されています。各小計ローでは、その小計の計算対象になったカラムに NULL が格納されています。

	Year	Quarter	Orders
1	(NULL)	(NULL)	648
2	(NULL)	1	226
	(NULL)	2	196
	(NULL)	3	101
	(NULL)	4	125
3	2000	(NULL)	380
	2000	1	87
	2000	2	77
	2000	3	91
	2000	4	125
3	2001	(NULL)	268
	2001	1	139
	2001	2	119
	2001	3	10

先頭のロー [1] は、両方の年のすべての四半期の注文数の合計を示しています。Orders カラムの値は、[3] としてマークされている各ローの値の合計です。これは、[2] としてマークされている 4 つのローの値の合計でもあります。

[2] としてマークされている一連のローは、両方の年の四半期別の注文数の合計を示しています。[3] としてマークされている 2 つのローは、それぞれ 2000 年および 2001 年のすべての四半期の注文数の合計を示しています。

統計関数

Sybase IQ では、1 つの SQL 文内で複雑なデータ分析を実行できる機能を備えた単純な集合関数とウィンドウ集合関数の両方を提供しています。これらの関数を使用して、たとえば“ダウ工業株 30 種平均の四半期の移動平均はどうなっているか”または“各部署のすべての従業員とその累積給与を一覧表示せよ”というクエリに対する答えを計算することができます。さまざまな期間における移動平均と累積和を計算したり、パーティション値が変化したときに集合計算がリセットされるような方法で集計とランクを分割したりできます。1 つのクエリ式のスコープ内で、それぞれ独自のパーティショニング・ルールを持ついくつかの異なる OLAP 関数を定義することができます。統計関数は 2 つのカテゴリに分けられます。

- 単純な集合関数 (AVG、COUNT、MAX、MIN、SUM など) は、データベースに含まれるローのグループのデータを要約します。SELECT 文の GROUP BY 句を使ってグループを形成します。
- 1 つの引数を取る単項の統計集合関数には、STDDEV()、STDDEV_SAMP()、STDDEV_POP()、VARIANCE()、VAR_SAMP()、および VAR_POP() があります。

単純な集合関数でも単項の集合関数でも、データベース内のローのグループに関するデータを要約することができ、ウィンドウ指定と組み合わせて、処理の際に結果セットに対する移動ウィンドウを計算することができます。

注意　集合関数　AVG、SUM、STDDEV、STDDEV_POP、STDDEV_SAMP、VAR_POP、VAR_SAMP、VARIANCE は、バイナリ・データ型である BINARY と VARBINARY をサポートしていません。

単純な集合関数

単純な集合関数 (AVG、COUNT、MAX、MIN、SUM など) は、データベースに含まれるローのグループのデータを要約します。ローのグループを形成するには SELECT 文の GROUP BY 句を使用します。集合関数は、select リストと、SELECT 文の HAVING 句および ORDER BY 句の中だけで使用できます。

注意　Grouping() 関数を除き、単純な集合関数と単項の集合関数はどちらも、SQL クエリの指定に「ウィンドウ句」(**ウィンドウ**) を組み込むウィンドウ関数として使用できます。これにより、処理時に結果セットに対して概念的に移動ウィンドウを作成することができます。詳細については、「ウィンドウ」(62 ページ) を参照してください。

詳細については、Sybase IQ リファレンス・マニュアルの「第 5 章 SQL 関数」の「集合関数」を参照してください。

ウィンドウ

OLAP に関する ANSI SQL 拡張機能で導入された主な機能は、**ウィンドウ**と呼ぶ構造です。このウィンドウ拡張機能により、ユーザはクエリの結果セット (または、クエリの論理パーティション) をパーティションと呼ばれるローのグループに分割し、現在のローについて集計するローのサブセットを決定することができます。

1 つのウィンドウには 3 つのクラスのウィンドウ関数として、ランク付け関数、ロー・ナンバリング関数、およびウィンドウ集合関数を使用できます。

```
<WINDOWED TABLE FUNCTION TYPE> ::=
   <RANK FUNCTION TYPE> <LEFT PAREN> <RIGHT PAREN>
   | ROW_NUMBER <LEFT PAREN> <RIGHT PAREN>
   | <WINDOW AGGREGATE FUNCTION>
```

詳細については、「文法規則 6」(98 ページ) を参照してください。

ウィンドウ拡張機能は、ウィンドウ名または指定に対するウィンドウ関数の種類を指定し、1 つのクエリ式のスコープ内のパーティション化された結果セットに適用されます。ウィンドウ・パーティションは、特殊な OVER 句の 1 つ以上のカラムで定義されている、クエリから返されるローのサブセットです。

```
olap_function() OVER (PARTITION BY col1, col2...)
```

ウィンドウ操作では、パーティション内の各ローのランク付け、パーティション内のローの値の分布、および同様の操作などの情報を設定できます。また、データの移動平均や合計を計算し、データおよび操作に対するそのデータの影響を評価する機能を拡張することもできます。

ウィンドウ・パーティションは、特殊な OVER() 句の 1 つ以上のカラムで定義されている、クエリから返されるローのサブセットです。

```
OLAP_FUNCTION() OVER (PARTITION BY col1, col2...)
```

OLAP ウィンドウの 3 つの重要な側面

OLAP ウィンドウは、ウィンドウ・パーティション、ウィンドウ順序、ウィンドウ・フレームという 3 つの重要な側面から成ります。それぞれの要素は、その時点でウィンドウ内で可視となるデータ・ローに大きな影響を与えます。また、OLAP の OVER 句は、次の 3 つの特徴的な機能により、OLAP 関数を他の統計関数やレポート関数から区別します。

- ウィンドウ・パーティションの定義 (PARTITION BY 句)。詳細については、「ウィンドウ・パーティション」(64 ページ) を参照してください。
- パーティション内でのローの順序付け (ORDER BY 句)。詳細については、「ウィンドウ順序」(64 ページ) を参照してください。
- ウィンドウ・フレームの定義 (ROWS/RANGE 指定)。「ウィンドウ・フレーム」(65 ページ)。

OLAP のウィンドウ指定に関して名前を指定することができます。冗長なウィンドウ定義を避けるために、この名前を使用して複数のウィンドウ関数を指定できます。その場合は、キーワード WINDOW の後に少なくとも 1 つのウィンドウ定義を指定します (複数指定する場合はカンマで区切ります)。ウィンドウ定義には、クエリ内でウィンドウを識別するための名前と、ウィンドウのパーティション、順序、フレームを定義するためのウィンドウ指定の詳細を含めます。

```
<WINDOW CLAUSE> ::= <WINDOW WINDOW DEFINITION LIST>

<WINDOW DEFINITION LIST> ::=
    <WINDOW DEFINITION> [ { <COMMA> <WINDOW DEFINITION>
    } . . . ]

<WINDOW DEFINITION> ::=
    <NEW WINDOW NAME> AS <WINDOW SPECIFICATION>

<WINDOW SPECIFICATION DETAILS> ::=
    [ <EXISTING WINDOW NAME> ]
    [ <WINDOW PARTITION CLAUSE> ]
    [ <WINDOW ORDER CLAUSE> ]
    [ <WINDOW FRAME CLAUSE> ]
```

ウィンドウ・パーティション内の各ローについて、ウィンドウ・フレームを定義することができます。ウィンドウ・フレームにより、パーティションの現在のローに対して計算を実行するときに使われるローの範囲を変更することができます。現在のローは、ウィンドウ・フレームの開始ポイントと終了ポイントを決定するための参照ポイントとなります。

ウィンドウのサイズは、物理的なローの数 (ウィンドウ・フレーム単位 ROWS を定義するウィンドウ指定を使用) または論理的な数値の間隔 (ウィンドウ・フレーム単位 RANGE を定義するウィンドウ指定を使用) に基づきます。詳細については、「ウィンドウ・フレーム」(65 ページ) を参照してください。

OLAP のウィンドウ操作では、次のカテゴリの関数を使用できます。

- 「ランク付け関数」(75 ページ)
- 「ウィンドウ集合関数」(80 ページ)
- 「統計集合関数」(81 ページ)
- 「分散統計関数」(82 ページ)

ウィンドウ・パーティション

ウィンドウ・パーティションとは、PARTITION BY 句を使用して、ユーザ指定の結果セット (入力ロー) を分割することです。パーティションは、カンマで区切られた 1 つ以上の値の式によって定義されます。パーティションに分割されたデータは暗黙的にソートされ、デフォルトのソート順序は昇順 (ASC) になります。

```
<WINDOW PARTITION CLAUSE> ::=
   PARTITION BY <WINDOW PARTITION EXPRESSION LIST>
```

ウィンドウ・パーティション句を指定しなかった場合は、入力が 1 つのパーティションとして扱われます。

注意 統計関数に対してパーティションという用語を使用した場合は、結果セットのローを PARTITION BY 句に基づいて分割することだけを意味します。

ウィンドウ・パーティションは任意の式に基づいて定義できます。また、ウィンドウ・パーティションの処理はグループ化の後に行われるので (GROUP BY 句が指定されている場合)、SUM、AVG、VARIANCE などの集合関数の結果をパーティションの式で使用することができます。したがって、パーティションを使用すると、GROUP BY 句や ORDER BY 句とはまた別に、グループ化と順序付けの操作を実行することができます。たとえば、ある数量の最大 SUM を求めるなど、集合関数に対して集合関数を計算するクエリを記述できます。

GROUP BY 句がない場合でも、PARTITION BY 句を指定できます。

ウィンドウ順序

ウィンドウ順序とは、ウィンドウ・パーティション内の結果 (ロー) をウィンドウ順序句に基づいて並べることです。ウィンドウ順序句には、1 つ以上の値の式をカンマ区切りで指定します。ウィンドウ順序句を指定しなかった場合は、入力ローが任意の順序で処理されることがあります。

```
<WINDOW ORDER CLAUSE> ::= <ORDER SPECIFICATION>
```

OLAP のウィンドウ順序句は、非ウィンドウ・クエリの式に指定できる ORDER BY 句とは異なります。詳細については、「文法規則 31」(100 ページ) を参照してください。

OLAP 関数で使用する ORDER BY 句は、通常はウィンドウ・パーティション内のローをソートするための式を定義しますが、PARTITION BY 句がなくても ORDER BY 句を使用することができます。その場合は、このソート指定によって、確実に意味のある (かつ意図どおりの) 順序で並べられた中間の結果セットに OLAP 関数を適用することができます。

OLAP のランク付け関数には順序の指定が必須であり、ランキング値の基準は、ランク付け関数の引数ではなく ORDER BY 句で指定します。OLAP の集合関数では、通常は ORDER BY 句の指定は必須ではありませんが、ウィンドウ・フレームを定義するときには必須とされています (「ウィンドウ・フレーム」(65 ページ) を参照してください)。これは、各フレームの適切な集合値を計算する前に、パーティション内のローをソートしなければならないためです。

この ORDER BY 句には、昇順および降順のソートを定義するためのセマンティックと、NULL 値の取り扱いに関する規則を指定します。OLAP 関数は、デフォルトでは昇順 (最も小さい値が 1 番目にランク付けされる) を使用します。

これは SELECT 文の最後に指定する ORDER BY 句のデフォルト動作と同じですが、連続的な計算を行う場合にはわかりにくいかもしれません。OLAP の計算では、降順 (最も大きい値が 1 番目にランク付けされる) でのソートが必要になることがよくあります。この要件を満たすには、ORDER BY 句に明示的に DESC キーワードを指定する必要があります。

注意　ランク付け関数は、ソートされた入力のみを扱うように定義されているため、「ウィンドウ順序句」の指定を必要とします。「クエリ指定」の「order by 句」と同様に、デフォルトのソート順序は昇順です。

「ウィンドウ・フレーム単位」で RANGE を使用する場合も、「ウィンドウ順序句」を指定する必要があります。RANGE の場合は、「ウィンドウ順序句」に 1 つの式のみを指定します。「ウィンドウ・フレーム」(65 ページ) を参照してください。

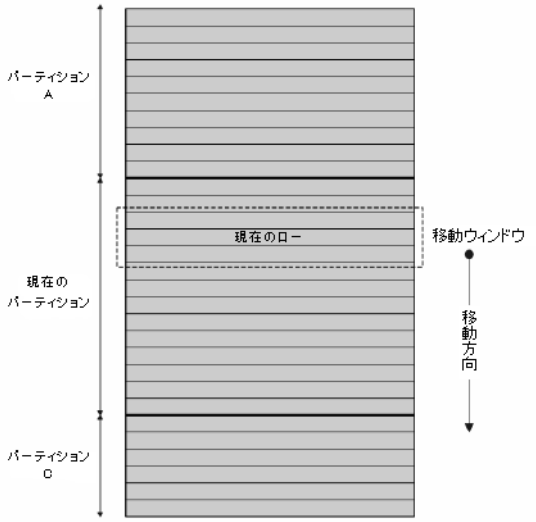
ウィンドウ・フレーム

ランク付け関数を除く OLAP 集合関数では、ウィンドウ・フレーム句を使用してウィンドウ・フレームを定義することができます。ウィンドウ・フレーム句には、現在のローを基準としてウィンドウの開始位置と終了位置を指定します。

```
<WINDOW FRAME CLAUSE> ::=
    <WINDOW FRAME UNIT>
    <WINDOW FRAME EXTENT>
```

これにより、パーティション全体の固定的な内容ではなく、移動するフレームの内容に対して OLAP 関数を計算できます。定義にもよりますが、パーティションには開始ローと終了ローがあり、ウィンドウ・フレームは開始ポイントからパーティションの終了位置に向けてスライドします。

図 4-3: 分割された入力と、3 ロー分の移動ウィンドウ



UNBOUNDED PRECEEDING と FOLLOWING

ウィンドウ・フレームは、パーティションの先頭 (UNBOUNDED PRECEDING)、最後 (UNBOUNDED FOLLOWING)、または両方まで到達する無制限の集合グループによって定義されます。

UNBOUNDED PRECEDING には、パーティション内の現在のロー以前にあるすべてのローが含まれており、ROWS または RANGE で指定できます。UNBOUNDED FOLLOWING には、パーティション内の現在のロー以後にあるすべてのローが含まれており、ROWS または RANGE で指定できます。詳細については、「ROWS」(67 ページ) と「RANGE」(70 ページ) を参照してください。

FOLLOWING の値では、現在のロー以降にあるローの範囲または数を指定します。ROWS を指定する場合、その値には、ローの数を表す正の数を指定します。RANGE を指定する場合、そのウィンドウには、現在のローに指定の数値を足した数よりも少ないローが含まれます。RANGE を指定する場合、そのウィンドウ値のデータ型は、ORDER BY 句のソート・キー式の型に対応している必要があります。指定できるソート・キー式は 1 つだけで、このソート・キー式のデータ型は「加算」を許可していなければなりません。

PREDCEEDING の値では、現在のロー以前にあるローの範囲または数を指定します。ROWS を指定する場合、その値には、ローの数を表す正の数を指定します。RANGE を指定する場合、そのウィンドウには、現在のローから指定の数値を引いた数よりも少ないローが含まれます。RANGE を指定する場合、そのウィンドウ値のデータ型は、ORDER BY 句のソート・キー式の型に対応している必要があります。指定できるソート・キー式は 1 つだけで、このソート・キー式のデータ型は「減算」を許可していなければなりません。1 つ目のバインドされたグループで CURRENT ROW または FOLLOWING の値を指定している場合は、2 つ目のバインドされたグループにこの句を指定することはできません。

BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING の組み合わせを使用すると、グループ化したクエリとのジョインを構築しなくても、パーティション全体についての集合を計算できます。パーティション全体についての集合は、レポート集合とも呼ばれます。

CURRENT ROW の概念

物理的な集合グループでは、現在のローに対する相対位置に基づき、隣接するローの数に応じて、ローを含めるか除外するかが判断されます。現在のローは、クエリの中間結果における次のローへの参照にすぎません。現在のローが前に進むと、ウィンドウ内に含まれる新しいロー・セットに基づいてウィンドウが再評価されます。現在のローをウィンドウ内に含めるという要件はありません。

ウィンドウ・フレーム句を指定しなかった場合のデフォルトのウィンドウ・フレームは、ウィンドウ順序句を指定しているかどうかによって異なります。

- ウィンドウ指定にウィンドウ順序句が含まれている場合は、ウィンドウの開始ポイントは UNBOUNDED PRECEDING、終了ポイントは CURRENT ROW になり、累積値の計算に適した可変サイズのウィンドウになります。
- ウィンドウ指定にウィンドウ順序句が含まれていない場合は、ウィンドウの開始ポイントは UNBOUNDED PRECEDING、終了ポイントは UNBOUNDED FOLLOWING になり、現在のローに関係なく固定サイズのウィンドウになります。

注意　ウィンドウ・フレーム句はランク付け関数とは併用できません。

ローベース (ロー指定) または値ベース (範囲指定) のウィンドウ・フレーム単位を指定してウィンドウを定義することもできます。

```
<WINDOW FRAME UNIT> ::= ROWS | RANGE

<WINDOW FRAME EXTENT> ::= <WINDOW FRAME START> | <WINDOW
FRAME BETWEEN>
```

ウィンドウ・フレーム句で BETWEEN を使用するときは、ウィンドウ・フレームの開始ポイントと終了ポイントを明示的に指定します。

ウィンドウ・フレーム句でこの 2 つの値のどちらか一方しか指定しなかった場合は、他方の値がデフォルトで CURRENT ROW になります。

ROWS

ウィンドウ・フレーム単位 ROWS では、現在のローの前後に指定の数のローを含んでいるウィンドウを定義します (現在のローは、ウィンドウの開始ポイントと終了ポイントを決定するための参照ポイントになります)。それぞれの分析計算は、パーティション内の現在のローに基づいて行われます。ローで表現されるウィンドウを使用して限定的な結果を生成するには、ユニークな順序付けの式を指定する必要があります。

どのウィンドウ・フレームでも、現在のローが参照ポイントになります。SQL/OLAP の構文には、ローベースのウィンドウ・フレームを、現在のローの前または後にある任意の数のロー (あるいは現在のローの前および後ろにある任意の数のロー) として定義するためのメカニズムが用意されています。

ウィンドウ・フレーム単位の代表的な例を次に示します。

- *Rows Between Unbounded Preceding and Current Row* － 各パーティションの先頭を開始ポイントとし、現在のローを終了ポイントとするウィンドウを指定します。累積和など、累積的な結果を計算するためのウィンドウを構築するときによく使用されます。
- *Rows Between Unbounded Preceding and Unbounded Following* － 現在のローに関係なく、パーティション全体についての固定ウィンドウを指定します。そのため、ウィンドウ集合関数の値は、パーティションのすべてのローで等しくなります。
- *Rows Between 1 Preceding and 1 Following* － 3 つの隣接するロー (現在のローとその前および後のロー) を含む固定サイズの移動ウィンドウを指定します。このウィンドウ・フレーム単位を使用して、たとえば 3 日間または 3 か月間の移動平均を計算できます。詳細については、図 4-3 (66 ページ) を参照してください。

  ウィンドウ値にギャップがあると、ROWS を使用した場合に意味のない結果が生成されることがあるので注意してください。値セットが連続していない場合は、ROWS の代わりに RANGE を使用することを検討してください。RANGE に基づくウィンドウ定義では、重複する値を含んだ隣接ローが自動的に処理され、範囲内にギャップがあるときに他のローが含まれません。

  注意 移動ウィンドウでは、入力の最初のローの前、および入力の最後のローの後ろには、NULL 値を含むローが存在することが想定されます。つまり、3 つのローから成る移動ウィンドウの場合は、入力の最後のローを現在のローとして計算するときに、直前のローと NULL 値が計算に含まれます。

- *Rows Between Current Row and Current Row* － ウィンドウを現在のローのみに制限します。
- *Rows Between 1 Preceding and 1 Preceding* － 現在のローの直前のローだけを含む単一ローのウィンドウを指定します。この指定を、現在のローのみに基づく値を計算する別のウィンドウ関数と組み合わせると、隣接するロー同士のデルタ (値の差分) を簡単に計算することができます。詳細については、「隣接ロー間のデルタの計算」(72 ページ) を参照してください。

ローベースのウィンドウ・フレーム　図 4-4 の例では、ロー [1] ～ [5] は 1 つのパーティションを表しています。それぞれのローは、OLAP のウィンドウ・フレームが前にスライドするにつれて現在のローになります。このウィンドウ・フレームは Between Current Row And 2 Following として定義されているため、各フレームには、最大で 3 つ、最小で 1 つのローが含まれます。フレームがパーティションの終わりに到達したときは、現在のローだけがフレームに含まれます。網掛けの部分は、図 4-4 の各ステップでフレームから除外されているローを表しています。

図 4-4: ローベースのウィンドウ・フレーム

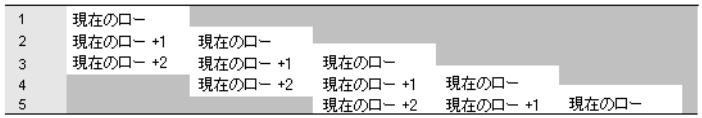
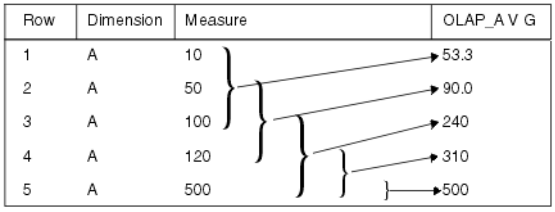


図 4-4 のウィンドウ・フレームは、次のような規則で機能しています。

- ロー [1] が現在のローであるときは、ロー [4] および [5] が除外される。
- ロー [2] が現在のローであるときは、ロー [5] および [1] が除外される。
- ロー [3] が現在のローであるときは、ロー [1] および [2] が除外される。
- ロー [4] が現在のローであるときは、ロー [1]、[2]、[3] が除外される。
- ロー [5] が現在のローであるときは、ロー [1]、[2]、[3]、[4] が除外される。

次の図では、この規則を具体的な値セットに適用し、OLAP の AVG 関数を使用して各ローの計算を行っています。スライド計算により、現在のローの位置に応じて、3 つまたはそれ以下のローを範囲として移動平均を算出しています。



次のクエリは、移動ウィンドウの定義の例を示しています。

```
SELECT dimension, measure,
   AVG(measure) OVER(partition BY dimension
      ORDER BY measure
      ROWS BETWEEN CURRENT ROW and 2 FOLLOWING)
      AS olap_avg
FROM ...
```

平均値は次のようにして計算されています。

- ロー [1] = (10 + 50 + 100)/3
- ロー [2] = (50+ 100 + 120)/3
- ロー [3] = (100 + 120 + 500)/3
- ロー [4] = (120 + 500 + NULL)/3
- ロー [5] = (500 + NULL + NULL)/3

結果セット内の以降のすべてのパーティション (たとえば B、C など) についても、同様の計算が実行されます。

現在のウィンドウにローが含まれていない場合、COUNT 以外のケースでは、結果は NULL になります。

RANGE

範囲ベースのウィンドウ・フレーム　前述のローベースのウィンドウ・フレームの例では、さまざまなローベースのウィンドウ・フレーム定義の中から 1 つを紹介しました。SQL/OLAP 構文では、また別の種類のウィンドウ・フレームとして、物理的なローのシーケンスではなく、値ベース (または範囲ベース) のロー・セットに基づいて境界を定義する方法が用意されています。

値ベースのウィンドウ・フレームは、ウィンドウ・パーティション内で、特定の範囲の数値を含んでいるローを定義します。OLAP 関数の ORDER BY 句では、範囲指定を適用する数値カラムを定義します。このカラムの現在のローの値が、範囲指定の基準となります。範囲指定ではロー指定と同じ構文を使用しますが、構文の解釈の仕方は異なります。

ウィンドウ・フレーム単位 RANGE では、特定の順序付けカラムについて現在のローを基準とする値範囲を指定し、その範囲内の値を持つローを検索して、ウィンドウ・フレームに含めます。これは論理的なオフセットに基づくウィンドウ・フレームと呼ばれ、“3 preceding” などの定数を指定することも、評価結果が数値定数となる任意の式を指定することもできます。RANGE に基づくウィンドウを使用するときは、ORDER BY 句に数値式を 1 つだけ指定します。

たとえば、次のように指定すると、*year* カラムに現在のローの前後数年に当たる値を含んでいるロー・セットをフレームとして定義できます。

```
ORDER BY year ASC range BETWEEN CURRENT ROW and 1 PRECEDING
```

このクエリ例の 1 PRECEDING という部分は、現在のローの *year* 値から 1 を減算することを意味しています。

このような範囲指定は内包的です。現在のローの *year* 値が 2000 である場合は、ウィンドウ・パーティション内で、year 値が 2000 および 1999 であるすべてのローがこのフレームに含まれることになります。パーティション内での各ローの物理的な位置は問われません。値ベースのフレームでは、ローを含めたり除外したりする規則が、ローベースのフレームの規則とは大きく異なります (ローベースのフレームの規則は、ローの物理的なシーケンスに完全に依存しています)。

OLAP の AVG() 関数の例で考えてみます。次の部分的な結果セットは、値ベースのウィンドウ・フレームの概念を具体的に表しています。前述のように、このフレームには次のローが含まれます。

- 現在のローと同じ year 値を持つロー
- 現在のローから 1 を減算したのと同じ year 値を持つロー

Row	Dimension	Year	Measure	Olap_avg
1	A	1999	10000	10000
2	A	2001	5000	3000
3	A	2001	1000	3000
4	A	2002	12000	5250
5	A	2002	3000	5250

次のクエリは、範囲ベースのウィンドウ・フレーム定義の例を示しています。

```
SELECT dimension, year, measure,
   AVG(measure) OVER(PARTITION BY dimension
      ORDER BY year ASC
      range BETWEEN CURRENT ROW and 1 PRECEDING)
      as olap_avg
FROM ...
```

平均値は次のようにして計算されています。

- ロー [1] = 1999 のため、ロー [2] 〜 [5] は除外。したがって AVG = 10,000/1
- ロー [2] = 2001 のため、ロー [1]、[4]、[5] は除外。したがって AVG = 6,000/2
- ロー [3] = 2001 のため、ロー [1]、[4]、[5] は除外。したがって AVG = 6,000/2
- ロー [4] = 2002 のため、ロー [1] は除外。したがって AVG = 21,000/4
- ロー [5] = 2002 のため、ロー [1] は除外。したがって AVG = 21,000/4

値ベースのフレームの昇順と降順　値ベースのウィンドウ・フレームを使用する OLAP 関数の ORDER BY 句では、範囲指定の対象となる数値カラムを特定するだけではなく、ORDER BY 値のソート順序も宣言できます。次の指定により、直前の部分のソート順序 (ASC または DESC) を設定できます。

```
RANGE BETWEEN CURRENT ROW AND n FOLLOWING
```

n FOLLOWING の指定には、次のような意味があります。

- パーティションがデフォルトの昇順 (ASC) でソートされている場合は、*n* は正の値として解釈されます。
- パーティションが降順 (DESC) でソートされている場合は、*n* は負の値として解釈されます。

たとえば、year カラムに 1999 ～ 2002 の 4 種類の値が含まれているとします。次のテーブルは、これらの値をデフォルトの昇順でソートした場合 (左側) と降順でソートした場合 (右側) を示しています。

ORDER BY year ASC	ORDER BY year DESC
1999	2002
2000	2001
2001	2000
2002	1999

現在のローが 1999 で、フレームが次のように指定されている場合、このフレームには値 1999 のローと値 1998 のロー (このテーブルには存在しません) が含まれます。

```
ORDER BY year ASC range BETWEEN CURRENT ROW and 1 FOLLOWING
```

注意 ORDER BY 値のソート順序は、値ベースのフレームに含まれるローの条件をテストするときに重要な要素です。フレームに含まれるか除外されるかは、数値だけでは決まりません。

無制限ウィンドウの使用 次のクエリでは、すべての製品と全製品の総数から成る結果セットが生成されます。

```
SELECT id, description, quantity,
   SUM(quantity) OVER () AS total
FROM product;
```

隣接ロー間のデルタの計算 現在のローと前のローをそれぞれ 1 つのウィンドウとして定義し、この 2 つのウィンドウを使用すると、隣接するロー間のデルタ (つまり差分) を直接的に計算することができます。次のクエリ例と結果を確認してください。

```
SELECT emp_id, emp_lname, SUM(salary) OVER (ORDER BY
   birth_date rows between current row and current row)
   AS curr, SUM(salary) OVER (ORDER BY birth_date rows
   between 1 preceding and 1 preceding) AS prev, (curr
   -prev) as delta
FROM employee WHERE state IN ('MA', 'AZ') AND dept_id
   =100
ORDER BY emp_id, emp_lname;
```

このクエリの結果セットを次に示します。

```
emp_id  emp_lname    curr       prev       delta
------  ---------    ---------  ---------  ----------
 102    Whitney      45700.000  64500.000  -18800.000
 105    Cobb         62000.000  68400.000   -6400.000
 160    Breault      57490.000  96300.000  -38810.000
 243    Shishov      72995.000  59840.000   13155.000
 247    Driscoll     48023.690  87900.000  -39876.310
 249    Guevara      42998.000  48023.690   -5025.690
 266    Gowda        59840.000  57490.000    2350.000
 278    Melkisetian  48500.000  74500.000  -26000.000
 316    Pastor       74500.000  62000.000   12500.000
 445    Lull         87900.000  67890.000   20010.000
 453    Rabkin       64500.000  42998.000   21502.000
 479    Siperstein   39875.500  42500.000   -2624.500
 501    Scott        96300.000  54900.000   41400.000
 529    Sullivan     67890.000  72995.000   -5105.000
 582    Samuels      37400.000  39875.500   -2475.500
 604    Wang         68400.000  45700.000   22700.000
 839    Marshall     42500.000  48500.000   -6000.000
1157    Soo          39075.000  37400.000    1675.000
1250    Diaz         54900.000
```

ここではウィンドウ関数 SUM() を使用していますが、ウィンドウの指定方法により、この合計には現在のローまたは前のローの salary 値だけが含まれています。また、結果セットの最初のローには前のローが存在しないため、最初のローの prev 値は NULL になります。したがって、delta も NULL になります。

ここまでの例では、OVER() 句と一緒に SUM() 集合関数を使用しました。

明示的なウィンドウ句とインラインのウィンドウ句

SQL OLAP では、クエリ内でウィンドウを指定する方法が 2 とおり用意されています。

- 明示的なウィンドウ句。HAVING 句の後でウィンドウを定義します。OLAP 関数を呼び出すときには、このようなウィンドウ句で定義したウィンドウを、ウィンドウの名前を指定して参照します。たとえば次のようにします。

  ```
  SUM ( ...) OVER w2
  ```

- インラインのウィンドウ指定。クエリ式の SELECT リスト内でウィンドウを定義します。これにより、HAVING 句の後のウィンドウ句でウィンドウを定義し、それをウィンドウ関数呼び出しから名前で参照するという方法に加えて、関数呼び出しと一緒にウィンドウを定義するという方法が可能になります。

  注意 インラインのウィンドウ指定を使用する場合は、ウィンドウの名前を指定できません。1 つの SELECT リスト内で複数のウィンドウ関数呼び出しが同じウィンドウを使用する場合には、ウィンドウ句で定義した名前付きウィンドウを参照するか、インラインのウィンドウ定義を繰り返す必要があります。

ウィンドウ関数の例 ウィンドウ関数の例を次に示します。このクエリでは、データを部署別のパーティションに分け、在社年数が最も長い従業員を基点とした従業員の累積給与を計算して、結果セットを返します。この結果セットには、マサチューセッツ在住の従業員だけが含まれます。Sum_Salary カラムには、従業員の給与の累積和が含まれます。

```
SELECT dept_id, emp_lname, start_date, salary,
   SUM(salary) OVER (PARTITION BY dept_id ORDER BY
   start_date rows between unbounded preceding and
   current row) AS sum_salary
FROM employee
WHERE state IN ('MA') AND dept_id IN (100, 200)
ORDER BY dept_id;
```

次の結果セットは部署別に分割されています。

dept_id	emp_lname	start_date	salary	sum_salary
100	Whitney	1984-08-28	45700.000	45700.000
100	Cobb	1985-01-01	62000.000	107700.000
100	Breault	1985-06-17	57490.000	165190.000
100	Shishov	1986-06-07	72995.000	238185.000
100	Driscoll	1986-07-01	48023.690	286208.690
100	Guevara	1986-10-14	42998.000	329206.690
100	Gowda	1986-11-30	59840.000	389046.690
100	Melkisetian	1986-12-06	48500.000	437546.690
100	Pastor	1987-04-26	74500.000	512046.690
100	Lull	1987-06-15	87900.000	599946.690
100	Rabkin	1987-06-15	64500.000	664446.690
100	Siperstein	1987-07-23	39875.500	704322.190
100	Scott	1987-08-04	96300.000	800622.190
100	Sullivan	1988-02-03	67890.000	868512.190
100	Samuels	1988-03-23	37400.000	905912.190
100	Wang	1988-09-29	68400.000	974312.190
100	Marshall	1989-04-20	42500.000	1016812.190
100	Soo	1990-07-31	39075.000	1055887.190
100	Diaz	1990-08-19	54900.000	1110787.190

```
200      Dill        1985-12-06   54800.000     54800.000
200      Powell      1988-10-14   54600.000    109400.000
200      Poitras     1988-11-28   46200.000    155600.000
200      Singer      1989-06-01   34892.000    190492.000
200      Kelly       1989-10-01   87500.000    277992.000
200      Martel      1989-10-16   55700.000    333692.000
200      Sterling    1990-04-29   64900.000    398592.000
200      Chao        1990-05-13   33890.000    432482.000
200      Preston     1990-07-11   37803.000    470285.000
200      Goggin      1990-08-05   37900.000    508185.000
200      Pickett     1993-08-12   47653.000    555838.000
```

ランク付け関数

ランク付け関数を使用すると、データ・セットの値をランク付けされた順序のリストにまとめ、“ 今年度出荷された製品の中で売上合計が上位 10 位の製品名” または “15 社以上から受注した営業部員の上位 5%” といった質問に答えるクエリを 1 つの SQL 文で作成することができます。ランク付け関数には RANK()、DENSE_RANK()、PERCENT_RANK()、NTILE() などがあり、PARTITION BY 句と一緒に使用します。

SQL/OLAP では、次の 4 つの関数がランク付け関数として分類されています。

```
<RANK FUNCTION TYPE> ::=
    RANK | DENSE RANK | PERCENT RANK | NTILE
```

ランク付け関数を使用すると、クエリで指定された順序に基づいて、結果セット内の各ローのランク値を計算することができます。たとえば販売マネージャが、営業成績が最高または最低の営業部員、販売成績が最高または最低の販売地域、あるいは売上が最高または最低の製品を調べたい場合があります。この情報はランク付け関数によって入手できます。

RANK() 関数

RANK 関数は、ORDER BY 句で指定されたカラムについて、ローのパーティション内での現在のローのランクを表す数値を返します。パーティション内の最初のローが 1 位となり、25 のローを含むパーティションでは、パーティション内の最後のローが 25 位となります。RANK は構文変換として指定されており、実際に RANK を同等の構文に変換することも、変換を行った場合に返すはずの値と同等の結果を返すこともできます。

次の例に出てくる ws1 は、w1 という名前のウィンドウを定義するウィンドウ指定を表しています。

```
RANK() OVER ws
```

これは次の指定に相当します。

```
( COUNT (*) OVER ( ws RANGE UNBOUNDED PRECEDING )
- COUNT (*) OVER ( ws RANGE CURRENT ROW ) + 1 )
```

この RANK 関数の変換では、論理的な集合 (RANGE) を使用しています。この結果、同位のロー (順序付けカラムに同じ値が含まれているロー) が複数ある場合は、それらに同じランクが割り当てられます。パーティション内で異なる値を持つ次のグループには、同位のローのランクよりも 1 以上大きいランクが割り当てられます。たとえば、順序付けカラムに 10、20、20、20、30 という値を含むローがある場合、1 つ目のローのランクは 1 になり、2 つ目のローのランクは 2 になります。3 つ目と 4 つ目のローのランクも 2 になりますが、5 つ目のローのランクは 5 になります。ランクが 3 または 4 のローは存在しません。このアルゴリズムは非連続型ランキング (sparse ranking) とも呼ばれます。

『Sybase IQ リファレンス・マニュアル』の「第 5 章 SQL 関数」の「RANK 関数 [統計]」も参照してください。

DENSE_RANK() 関数

RANK 関数は同位のローがあるときに重複したランク値を割り当て非連続的なランキングを返しますが、DENSE_RANK 関数は抜けのないランキングを返します。同位のローに対しては同じように等しいランク値が割り当てられますが、このローのランクは、個々のローの順位ではなく、順序付けカラムに等しい値を含んでいるローの集まりの順位を表しています。RANK の例と同様に、順序付けカラムに 10、20、20、20、30 という値を含むローがある場合、1 つ目のローのランクは同じく 1 となり、2 つ目のローおよび 3 つ目、4 つ目のローのランクも同じく 2 となります。しかし、最後のローのランクは 5 ではなく 3 になります。

DENSE_RANK も、構文変換を通じて計算されます。

```
DENSE_RANK() OVER ws
```

これは次の指定に相当します。

```
COUNT ( DISTINCT ROW ( expr_1, . . ., expr_n ) )
  OVER ( ws RANGE UNBOUNDED PRECEDING )
```

この例では、*expr_1* から *expr_n* の部分が、ウィンドウ w1 のソート指定リストに含まれている値の式のリストを表しています。

『Sybase IQ リファレンス・マニュアル』の「第 5 章 SQL 関数」の「DENSE_RANK 関数 [統計]」も参照してください。

PERCENT_RANK() 関数

PERCENT_RANK 関数は、個別の順位ではなく、パーセンテージでのランクを計算して、0 ～ 1 の小数値を返します。つまり、PERCENT_RANK が返すのはローの相対的なランクであり、この数値は、該当するウィンドウ・パーティション内での現在のローの相対位置を表します。たとえば、順序付けカラムの値がそれぞれ異なる 10 個のローがパーティションに含まれている場合、このパーティションの 3 つ目のローに対する PERCENT_RANK の値は 0.222 ... となります。パーティションの 1 つ目のローに続く 2/9 (22.222...%) のローをカバーしているためです。次の例に示すとおり、ローの PERCENT_RANK は、「ローの RANK - 1」を「パーティション内のローの数 - 1」で割ったものとして定義されています (“ANT” は、REAL や DOUBLE PRECISION などの概数値の型を表します)。

```
PERCENT_RANK() OVER ws
```

これは次の指定に相当します。

```
CASE
    WHEN COUNT (*) OVER ( ws RANGE BETWEEN UNBOUNDED
    PRECEDING AND UNBOUNDED FOLLOWING ) = 1
    THEN CAST (0 AS ANT)
    ELSE
      ( CAST ( RANK () OVER ( ws ) AS ANT ) -1 /
      ( COUNT (*) OVER ( ws RANGE BETWEEN UNBOUNDED
      PRECEDING AND UNBOUNDED FOLLOWING ) - 1 )
END
```

『Sybase IQ リファレンス・マニュアル』の「第 5 章 SQL 関数」の「PERCENT_RANK 関数 [統計]」も参照してください。

ランク付けの例

ランク付けの例 1　次の SQL クエリでは、マサチューセッツ州在住の男性従業員と女性従業員を取得し、給与を基準として降順にランク付けしています。

```
SELECT emp_lname, salary, sex, RANK() OVER (ORDER BY
    salary DESC) AS Rank
FROM employee WHERE state IN ('MA') AND dept_id =100
ORDER BY salary DESC;
```

このクエリの結果セットを次に示します。

emp_lname	salary	sex	rank
Scott	96300.000	M	1
Lull	87900.000	M	2
Pastor	74500.000	F	3
Shishov	72995.000	F	4
Wang	68400.000	M	5
Sullivan	67890.000	F	6
Rabkin	64500.000	M	7
Cobb	62000.000	M	8

```
Gowda          59840.000      M          9
Breault        57490.000      M         10
Diaz           54900.000      M         11
Melkisetian    48500.000      F         12
Driscoll       48023.690      M         13
Whitney        45700.000      F         14
Guevara        42998.000      M         15
Marshall       42500.000      M         16
Siperstein     39875.500      F         17
Soo            39075.000      M         18
Samuels        37400.000      M         19
```

ランク付けの例 2　ランク付けの例 1 のクエリを基にして、データを性別のパーティションに分けることができます。次の例では、性別のパーティションに分けて、従業員の給与を降順にランク付けしています。

```
SELECT emp_lname, salary, sex, RANK() OVER (PARTITION
   BY sex ORDER BY salary DESC) AS RANK
FROM employee WHERE state IN ('MA', 'AZ') AND dept_id
   IN (100, 200)
ORDER BY sex, salary DESC;
```

このクエリの結果セットを次に示します。

emp_lname	salary	sex	rank
Kelly	87500.000	F	1
Pastor	74500.000	F	2
Shishov	72995.000	F	3
Sullivan	67890.000	F	4
Melkisetian	48500.000	F	5
Pickett	47653.000	F	6
Poitras	46200.000	F	7
Whitney	45700.000	F	8
Siperstein	39875.500	F	9
Scott	96300.000	M	1
Lull	87900.000	M	2
Wang	68400.000	M	3
Sterling	64900.000	M	4
Rabkin	64500.000	M	5
Cobb	62000.000	M	6
Gowda	59840.000	M	7
Breault	57490.000	M	8
Martel	55700.000	M	9
Diaz	54900.000	M	10
Dill	54800.000	M	11
Powell	54600.000	M	12
Driscoll	48023.690	M	13
Guevara	42998.000	M	14
Marshall	42500.000	M	15
Soo	39075.000	M	16
Goggin	37900.000	M	17
Preston	37803.000	M	18

```
Samuels          37400.000      M          19
Singer           34892.000      M          20
Chao             33890.000      M          21
```

ランク付けの例 3　この例では、カリフォルニアおよびテキサスの女性従業員を取得し、給与を基準として降順にランク付けしています。累積和を降順で示すために、PERCENT_RANK 関数を使用しています。

```
SELECT emp_lname, salary, sex, CAST(PERCENT_RANK() OVER
   (ORDER BY salary DESC) AS numeric (4, 2)) AS RANK
FROM employee WHERE state IN ('CA', 'TX') AND sex ='F'
ORDER BY salary DESC;
```

このクエリの結果セットを次に示します。

```
emp_lname          salary    sex      percent
---------      ---------   ---   ----------
Savarino       72300.000   F           0.00
Smith          51411.000   F           0.33
Clark          45000.000   F           0.66
Garcia         39800.000   F           1.00
```

ランク付けの例 4　PERCENT_RANK 関数を使用して、データ・セットにおける上位または下位のパーセンタイルを調べることができます。この例のクエリは、給与の額がデータ・セットの上位 5% に入る男性従業員を返します。

```
SELECT * FROM (SELECT emp_lname, salary, sex,
   CAST(PERCENT_RANK() OVER (ORDER BY salary DESC) as numeric
(4, 2)) AS percent
FROM employee WHERE state IN ('MA') AND sex ='F' ) AS
   DT where percent > 0.5
ORDER BY salary DESC;
```

このクエリの結果セットを次に示します。

```
emp_lname          salary    sex     percent
---------      ----------   ---   ---------
Whitney        45700.000    F          0.51
Barletta       45450.000    F          0.55
Higgins        43700.000    F          0.59
Siperstein     39875.500    F          0.62
Coe            36500.000    F          0.66
Espinoza       36490.000    F          0.70
Wetherby       35745.000    F          0.74
Braun          34300.000    F          0.77
Butterfield    34011.000    F          0.81
Bigelow        31200.000    F          0.85
Bertrand       29800.000    F          0.88
Lambert        29384.000    F          0.92
Kuo            28200.000    F          0.96
Romero         27500.000    F          1.00
```

ウィンドウ集合関数

ウィンドウ集合関数を使用すると、複数のレベルの集合を 1 つのクエリで計算できます。たとえば、支出が平均より少ない四半期をすべて列挙することができます。集合関数 (単純な集合関数 AVG、COUNT、MAX、MIN、SUM を含む) を使用すると、1 つの文の中でさまざまなレベルで計算した結果を 1 つのローに書き出すことができます。これにより、ジョインや相関サブクエリを使用しなくても、集合値をグループ内のディテール・ローと比較することができます。

これらの関数を使用して、非集合値と集合値を比較することも可能です。たとえば、営業部員が特定の年にある製品に対して平均以上の注文を出した顧客の一覧を作成したり、販売マネージャが従業員の給与をその部署の平均給与と比較したりすることが考えられます。

SELECT 文の中で DISTINCT が指定されている場合は、ウィンドウ演算子の後に DISTINCT 操作が適用されます (ウィンドウ演算子は、GROUP BY 句が処理された後、SELECT リストの項目やクエリの ORDER BY 句が評価される前に計算されます)。

ウィンドウ集合関数の例 1　この例のクエリは、平均販売数よりも多く売れた製品の一覧を年別に示す結果セットを返します。

```
SELECT * FROM (SELECT year(order_date) AS Y, prod_id,
   SUM(quantity) AS Q,  CAST(AVG(SUM(quantity)) OVER
   (PARTITION BY Y) AS numeric (8, 2)) AS Average
FROM alt_sales_order S, alt_sales_order_items O
WHERE S.id = O.id
GROUP BY Y, O.prod_id ) AS derived_table
   WHERE Q > Average
ORDER BY Y, prod_id;
```

このクエリの結果セットを次に示します。

Year	prod_id	Q	Average
2000	400	2030	1787.00
2000	600	2124	1787.00
2000	601	1932	1787.00
2000	700	2700	1787.00
2001	400	1248	1048.90
2001	401	1057	1048.90
2001	700	1836	1048.90

2000 年の平均注文数は 1,787 であり、4 つの製品 (700、601、600、400) が平均を上回っています。2001 年の平均注文数は 1,048 であり、3 つの製品が平均を上回っています。

ウィンドウ集合関数の例 2　この例のクエリは、給与の額がそれぞれの部署の平均給与よりも 1 標準偏差以上高い従業員を表す結果セットを返します。標準偏差とは、そのデータが平均からどのぐらい離れているかを示す尺度です。

```
SELECT * FROM (SELECT emp_lname AS E_name, dept_id AS
   Dept, CAST(salary AS numeric(10,2) ) AS Sal,
   CAST(AVG(Sal) OVER(PARTITION BY dept_id) AS
   numeric(10, 2)) AS Average, CAST(STDDEV_POP(Sal)
   OVER(PARTITION BY dept_id) AS numeric(10,2)) AS
   STD_DEV
FROM employee
GROUP BY Dept, E_name, Sal) AS derived_table WHERE
   Sal> (Average+STD_DEV )
ORDER BY Dept, Sal, E_name;
```

このクエリの結果セットを次に示します。どの部署にも、給与の額が平均を大きく上回っている従業員が 1 人以上いることがわかります。

Employee	Dept	Salary	Average	Std_Dev
Lull	100	87900.00	58736.28	16829.59
Sheffield	100	87900.00	58736.28	16829.59
Scott	100	96300.00	58736.28	16829.59
Sterling	200	64900.00	48390.94	13869.59
Savarino	200	72300.00	48390.94	13869.59
Kelly	200	87500.00	48390.94	13869.59
Shea	300	138948.00	59500.00	30752.39
Blaikie	400	54900.00	43640.67	11194.02
Morris	400	61300.00	43640.67	11194.02
Evans	400	68940.00	43640.67	11194.02
Martinez	500	55500.80	33752.20	9084.49

従業員 Scott の給与は 96,300.00 ドルで、所属部署の平均給与は 58,736.28 ドルです。この部署の標準偏差は 16,829.00 なので、給与の額が 75,565.88 ドル (58736.28 + 16829.60 = 75565.88) 未満ならば、平均の 1 標準偏差以内の範囲に収まります。Scott の給与 96,300.00 ドルは、この数字を大きく超えています。

統計集合関数

ANSI SQL/OLAP 拡張機能には、数値データの統計的分析を行うための集合関数がこの他にも数多く用意されています。これには、分散、標準偏差、相関、直線回帰を計算するための関数も含まれます。

標準偏差と分散

SQL/OLAP の一般的な関数の中には、STDDEV、STDDEV_POP、STDDEV_SAMP、VARIANCE、VAR_POP、VAR_SAMP のように、1 つの引数を取る関数があります。

```
<SIMPLE WINDOW AGGREGATE FUNCTION TYPE> ::=
   <BASIC AGGREGATE FUNCTION TYPE>
   | STDDEV | STDDEV_POP | STDDEV_SAMP
   | VARIANCE | VARIANCE_POP | VARIANCE_SAMP
```

- STDDEV_POP − グループまたはパーティションの各ロー (DISTINCT が指定されている場合は、重複が削除された後に残る各ロー) に対して評価される「値の式」についての母標準偏差を計算します。これは、母分散の平方根として定義されます。
- STDDEV_SAMP − グループまたはパーティションの各ロー (DISTINCT が指定されている場合は、重複が削除された後に残る各ロー) に対して評価される「値の式」についての母標準偏差を計算します。これは、標本分散の平方根として定義されます。
- VAR_POP − グループまたはパーティションの各ロー (DISTINCT が指定されている場合は、重複が削除された後に残る各ロー) に対して評価される「値の式」についての母分散を計算します。これは、「値の式」と「値の式の平均」との差の 2 乗和をグループまたはパーティション内の残りのローの数で割った値として定義されます。
- VAR_SAMP − グループまたはパーティションの各ロー (DISTINCT が指定されている場合は、重複が削除された後に残る各ロー) に対して評価される「値の式」の標本分散を計算します。これは、「値の式」の差の 2 乗和を、グループまたはパーティション内の残りのローの数より 1 少ない数で割った値として定義されます。

これらの関数と STDDEV および VARIANCE 関数は、クエリの ORDER BY 句の指定に従ってローのパーティションについての値を計算できる集合関数です。MAX や MIN などのその他の基本的な集合関数と同様に、これらの関数は入力データ内の NULL 値を無視します。また、分析される式のドメインに関係なく、分散と標準偏差の計算では必ず IEEE の倍精度浮動小数点数が使用されます。分散関数または標準偏差関数への入力が空のデータ・セットである場合、これらの関数は結果として NULL を返します。VAR_SAMP 関数は 1 つのローに対して計算を行うと NULL を返しますが、VAR_POP は値 0 を返します。

分散統計関数

SQL/OLAP には、順序付きセットを取り扱う関数がいくつか定義されています。PERCENTILE_CONT と PERCENTILE_DISC という 2 つの逆分散統計関数があります。これらの統計関数は、パーセンタイル値を引数として受け取り、WITHIN GROUP 句で指定されたデータのグループまたはデータ・セット全体に対して処理を行います。

これらの関数は、グループごとに 1 つの値を返します PERCENTILE_DISC (不連続) の場合、結果のデータ型は、WITHIN GROUP 句に指定した ORDER BY の項目のデータ型と同じになります。PERCENTILE_CONT (連続) では、結果のデータ型は、numeric (WITHIN GROUP 句の ORDER BY 項目が numeric の場合) または double (ORDER BY 項目が整数または浮動小数点の場合) となります。

逆分散統計関数では、WITHIN GROUP (ORDER BY) 句を指定する必要があります。次に例を示します。

PERCENTILE_CONT (*expression1*)
WITHIN GROUP (**ORDER BY** *expression2* [**ASC** | **DESC**])

expression1 の値には、numeric データ型の定数を、0 以上 1 以下の範囲で指定します。引数が NULL であれば、“wrong argument for percentile” エラーが返ります。引数の値が 0 よりも小さいか、1 よりも大きい場合は、“data value out of range” エラーが返ります。

必須の ORDER BY には、パーセンタイル関数の実行の対象となる式と、各グループ内でのローのソート順を指定します。この ORDER BY 句は、WITHIN GROUP 句の内部でのみ使用するものであり、SELECT 文の ORDER BY とは異なります。

WITHIN GROUP 句は、クエリの結果を並べ替えて、関数が結果を計算するためのデータ・セットを形成します。

expression2 には、カラム参照を含む 1 つの式でソートを指定します。このソート式に、複数の式やランク付け統計関数、set 関数、またはサブクエリを指定することはできません。

ASC と DESC のパラメータでは、昇順または降順の順序付けシーケンスを指定します。昇順がデフォルトです。

逆分散統計関数は、サブクエリ、HAVING 句、ビュー、union で使用することが可能です。逆分散統計関数は、分析を行わない単純な集合関数が使用されるところであれば、どこでも使用できます。逆分散統計関数は、データ・セット内の NULL 値を無視します。

PERCENTILE_CONT 例　この例では、PERCENTILE_CONT 関数を使用して、各地域の自動車販売の 10 番目のパーセンタイル値を求めます。次のようなデータ・セットを使用します。

```
sales       region        dealer_name
-----       ---------     -----------
900         Northeast     Boston
800         Northeast     Worcester
800         Northeast     Providence
700         Northeast     Lowell
540         Northeast     Natick
500         Northeast     New Haven
450         Northeast     Hartford
800         Northwest     SF
600         Northwest     Seattle
500         Northwest     Portland
400         Northwest     Dublin
500         South         Houston
400         South         Austin
300         South         Dallas
200         South         Dover
```

次のクエリ例では、SELECT 文に PERCENTILE_CONT 関数を含めています。

```
SELECT region, PERCENTILE_CONT(0.1)
WITHIN GROUP ( ORDER BY sales DESC )
FROM carSales GROUP BY region;
```

この SELECT 文の結果には、各地域の自動車販売の 10 番目のパーセンタイル値が一覧表示されます。

```
region          percentile_cont
---------       ---------------
Northeast       840
Northwest       740
South           470
```

PERCENTILE_DISC 例　この例では、PERCENTILE_DISC 関数を使用して、各地域の自動車販売の 10 番目のパーセンタイル値を求めます。次のようなデータ・セットを使用します。

```
sales       region        dealer_name
-----       ---------     -----------
900         Northeast     Boston
800         Northeast     Worcester
800         Northeast     Providence
700         Northeast     Lowell
540         Northeast     Natick
500         Northeast     New Haven
450         Northeast     Hartford
800         Northwest     SF
600         Northwest     Seattle
500         Northwest     Portland
400         Northwest     Dublin
500         South         Houston
400         South         Austin
300         South         Dallas
200         South         Dover
```

次のクエリ例では、SELECT 文に PERCENTILE_DISC 関数を含めています。

```
SELECT region, PERCENTILE_DISC(0.1) WITHIN GROUP
   (ORDER BY sales DESC )
FROM carSales GROUP BY region;
```

この SELECT 文の結果には、各地域の自動車販売の 10 番目のパーセンタイル値が一覧表示されます。

```
region          percentile_cont
---------       ---------------
Northeast       900
Northwest       800
South           500
```

分散統計関数の詳細については、『Sybase IQ リファレンス・マニュアル』の「第 5 章 SQL 関数」の「PERCENTILE_CONT 関数 [統計]」と「PERCENTILE_DISC 関数 [統計]」を参照してください。

数値関数

Sybase IQ でサポートされる OLAP 数値関数には、CEILING (エイリアスは CEIL)、EXP (エイリアスは EXPONENTIAL)、FLOOR、LN (エイリアスは LOG)、SQRT、WIDTH_BUCKET があります。

```
<numeric value function> :: =
  <natural logarithm>
|  <exponential function>
|  <power function>
|  <square root>
|  <floor function>
|  <ceiling function>
|  <width bucket function>
```

サポートされる数値関数の構文を表 4-3 に示します。

表 4-3: 数値関数の構文

数値関数	構文
自然対数	LENGTH (*string-expression*)
指数関数	EXP (*numeric-expression*)
累乗関数	POWER (*numeric-expression1*, *numeric-expression2*)
平方根	SQRT (*numeric-expression*)
床関数	FLOOR (*numeric-expression*)
天井関数	CEILING (*numeric-expression*)
等幅ヒストグラム作成関数	WIDTH_BUCKET (*expression*, *min_value*, *max_value*, *num_buckets*)

それぞれの数値関数の機能は次のとおりです。

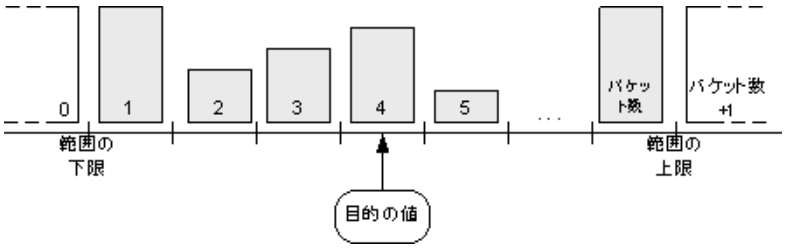
- LN: 引数値の自然対数を返します。引数値がゼロまたは負の場合は、エラー状態が発生します。LN は LOG の同意語です。
- EXP: *e* (自然対数の底) の値を、引数値で指定された指数まで累乗した結果を返します。
- POWER: 1 つ目の引数値を、2 つ目の引数値で指定された指数まで累乗した結果を返します。両方の引数の値が 0 の場合は、1 が返されます。1 つ目の引数が 0 で、2 つ目の引数が正の値である場合は、0 が返されます。1 つ目の引数が 0 で、2 つ目の引数が負の値である場合は、例外が発生します。1 つ目の引数が負の値で、2 つ目の引数が整数でない場合は、例外が発生します。
- SQRT: 引数値の平方根を返します。これは、“POWER (*expression*, *0.5*)” の構文変換です。

- FLOOR: 引数の値以下で、正の無限大に最も近い整数値を返します。
- CEILING: 引数の値以上で、負の無限大に最も近い整数値を返します。CEIL は CEILING の同意語です。

WIDTH_BUCKET 関数

WIDTH_BUCKET 関数は、他の数値関数よりも少し複雑です。この関数は 4 つの引数を取ります。具体的には、「目的の値」、2 つの範囲境界、そしてこの範囲を何個の等しいサイズ (または可能な限り等しいサイズ) の「バケット」に分割するかを指定します。WIDTH_BUCKET 関数は、範囲の上限から下限までの差のパーセンテージに基づき、目的の値が何番目のバケットに含まれるかを示す数値を返します。最初のバケットが、バケット番号 1 となります。

目的の値が範囲境界の外にある場合のエラーを避けるために、範囲の下限よりも小さい目的の値は、先頭の補助バケット (バケット 0) に配置されます。同様に、範囲の上限よりも大きい目的の値は、末尾の補助バケット (バケット N+1) に配置されます。



たとえば、WIDTH_BUCKET (*14, 5, 30, 5*) は 2 を返します。処理の内容は次のとおりです。

- (30-5)/5 = 5 なので、指定の範囲を 5 つのバケットに分割すると、各バケットの幅は 5 になります。
- 1 つ目のバケットは 0.00 ～ 19.999 ...% の値、2 つ目のバケットは 20.00 ～ 39.999 ...% の値を表し、以降同様に続き、5 つ目のバケットは 80.00 ～ 100.00% の値を表します。
- 目的の値を含むバケットは、(5*(14-5)/(30-5)) + 1 という計算によって算出されます。これは、バケットの総数に、指定範囲に対する「下限から目的の値までのオフセット」の比率を掛け、それに 1 を足すという計算です。実際の数式は (5*9/25) + 1 となり、これを計算すると 2.8 になります。これはバケット番号 2 (2.0 ～ 2.999 ...) の範囲に含まれる値であるため、バケット番号 2 が返されます。

WIDTH_BUCKET 例

次の例では、サンプル・テーブル内のマサチューセッツ州の顧客の credit_limit カラムに関する 10 バケットのヒストグラムを作成し、各顧客のバケット番号 (“Credit Group”) を返します。最大値を超える限度額が設定されている顧客は、オーバフロー・バケット 11 に割り当てられます。

注意 これは説明用の例であり、asiqdemo データベースから生成したものではありません。

```
SELECT customer_id, cust_last_name, credit_limit,
   WIDTH_BUCKET(credit_limit, 100, 5000, 10) "Credit
   Group"
   FROM customers WHERE territory = 'MA'
   ORDER BY "Credit Group";
```

CUSTOMER_ID	CUST_LAST_NAME	CREDIT_LIMIT	Credit Group
825	Dreyfuss	500	1
826	Barkin	500	1
853	Palin	400	1
827	Siegel	500	1
843	Oates	700	2
844	Julius	700	2
835	Eastwood	1200	3
840	Elliott	1400	3
842	Stern	1400	3
841	Boyer	1400	3
837	Stanton	1200	3
836	Berenger	1200	3
848	Olmos	1800	4
849	Kaurusmdki	1800	4
828	Minnelli	2300	5
829	Hunter	2300	5
852	Tanner	2300	5
851	Brown	2300	5
850	Finney	2300	5
830	Dutt	3500	7
831	Bel Geddes	3500	7
832	Spacek	3500	7
838	Nicholson	3500	7
839	Johnson	3500	7
833	Moranis	3500	7
834	Idle	3500	7
845	Fawcett	5000	11
846	Brando	5000	11
847	Streep	5000	11

範囲境界の指定が逆になっている場合は、各バケットの間隔が逆になります。たとえば、WIDTH_BUCKET (*credit_limit*, *5000*, *0*, *5*) という関数呼び出しを考えてみます。この例では、バケット番号 1 は (4000 ～ 5000)、バケット番号 2 は (3000 ～ 4000) となり、以降同様に続き、バケット番号 5 は (0 ～ 1000) となります。オーバフロー・バケットは番号 0 (5000 ～ + ∞)、アンダフロー・バケットは番号 6 (- ∞ ～ 0) となります。

参照

『Sybase IQ リファレンス・マニュアル』の「第 5 章 SQL 関数」の「BIT_LENGTH 関数 [文字列]」、「EXP 関数 [数値]」、「FLOOR 関数 [数値]」、「POWER 関数 [数値]」、「SQRT 関数 [数値]」、「WIDTH_BUCKET 関数 [数値]」を参照してください。

OLAP の規則と制限

OLAP 関数を使用できる場合

SQL クエリ内では、次の条件下で OLAP 関数を使用できます。

- SELECT リストの中
- 式の中
- スカラ関数の引数として
- 最後の ORDER BY 句の中 (クエリ内のどこかで定義されている OLAP 関数のエイリアスまたは位置参照を使用)

OLAP 関数を使用できない場合

OLAP 関数は、次の条件下では使用できません。

- サブクエリの中
- WHERE 句の検索条件の中
- SUM (集合) 関数の引数としてたとえば次の式は無効です。

```
SUM(RANK() OVER(ORDER BY dollars))
```

- ウィンドウ集合を、他の集合に対する引数として使用することはできません (ただし、内側の集合がビューまたは抽出テーブル内で生成されたものである場合は例外です)。ランク付け関数についても同じことが言えます。
- ウィンドウ集合関数と RANK 関数は、HAVING 句の中では使用できません。
- ウィンドウ集合関数に DISTINCT を指定することはできません。
- ウィンドウ関数を他のウィンドウ関数の内部にネストすることはできません。
- 逆分散統計関数は、OVER 句ではサポートされていません。
- ウィンドウ定義句では外部参照を使用できません。
- OLAP 関数内での相関参照は認められていますが、相関があるカラムのエイリアスは認められていません。

OLAP 関数から参照するカラムは、その OLAP 関数と GROUP BY 句が含まれている同じクエリ・ブロック内のグループ化カラムまたは集合関数でなければなりません。OLAP の処理は、グループ化操作と集合操作の後、最後の ORDER BY 句が適用される前に行われます。そのため、中間の結果セットから OLAP 式を導出することも可能です。クエリ・ブロック内に GROUP BY 句がない場合、OLAP 関数は select リスト内の他のカラムを参照することができます。

Sybase IQ の制限事項

Sybase IQ で SQL OLAP 関数を使用するときの制限事項を次に示します。

- ウィンドウ・フレーム定義の中でユーザ定義関数を使用することはできません。
- ウィンドウ・フレーム定義で使用する定数は符号なし数値でなければならず、最大値 BIG INT 2^{63-1} を超えてはなりません。
- ウィンドウ集合関数と RANK 関数は、DELETE および UPDATE 文では使用できません。
- ウィンドウ集合関数と RANK 関数は、サブクエリ内では使用できません。
- CUME_DIST は、現時点ではサポートされていません。
- グループ化セットは、現時点ではサポートされていません。
- 相関関数と直線回帰関数は、現時点ではサポートされていません。

その他の OLAP の例

この項では、OLAP 関数を使用したその他の例を紹介します。

ウィンドウの開始ポイントと終了ポイントは、中間の結果ローが処理されるときに変化する可能性があります。たとえば、累積和を計算する場合には、ウィンドウの開始ポイントは各パーティションの最初のローに固定されますが、終了ポイントは現在のローを含めるためにパーティション内のローを移動していきます。詳細については、図 4-3 (66 ページ) を参照してください。

また、ウィンドウの開始ポイントと終了ポイントの両方が可変だが、パーティション全体のローの数は一定であるという例も考えられます。このようなウィンドウを使用すると移動平均を計算するクエリを作成でき、たとえば 3 日間の株価の移動平均を返す SQL クエリを作成できます。

例：クエリ内でのウィンドウ関数

次のクエリは、2005 年の 7 月と 8 月に出荷された全製品と、出荷日別の累積出荷数を一覧にして示します。

```
SELECT p.id, p.description, s.quantity, s.ship_date,
    SUM(s.quantity) OVER (PARTITION BY prod_id ORDER BY
    s.ship_date rows between unbounded preceding and
    current row)
FROM alt_sales_order_items s JOIN product p on (s.prod_id =
    p.id) WHERE s.ship_date BETWEEN '2001-05-01' and
    '2001-08-31' AND  s.quantity > 40
ORDER BY p.id;
```

このクエリの結果セットを次に示します。

ID	description	quantity	ship_date	sum quantity
302	Crew Neck	60	2001-07-02	60
400	Cotton Cap	60	2001-05-26	60
400	Cotton Cap	48	2001-07-05	108
401	Wool cap	48	2001-06-02	48
401	Wool cap	60	2001-06-30	108
401	Wool cap	48	2001-07-09	156
500	Cloth Visor	48	2001-06-21	48
501	Plastic Visor	60	2001-05-03	60
501	Plastic Visor	48	2001-05-18	108
501	Plastic Visor	48	2001-05-25	156
501	Plastic Visor	60	2001-07-07	216
601	Zipped Sweatshirt	60	2001-07-19	60
700	Cotton Shorts	72	2001-05-18	72
700	Cotton Shorts	48	2001-05-31	120

この例では、2 つのテーブルのジョインとクエリの WHERE 句を適用した後に、SUM ウィンドウ関数の計算が行われます。このクエリではインラインのウィンドウ指定を使用しており、このウィンドウ指定によって、ジョインからの入力ローが次のように処理されています。

1 prod_id 属性の値に基づいて入力ローがパーティション (グループ) に分けられます。

2 各パーティション内で、ローが ship_date 属性に基づいてソートされます。

3 パーティション内の各ローの quantity 属性について、SUM() 関数が評価されます。その際に、ソート後の各パーティションの最初のローから現在のローまでを含む移動ウィンドウ (現在のローも含む) が使用されます。詳細については、図 4-3 (66 ページ) を参照してください。

このクエリを別の方法で記述するには、関数の外でウィンドウを定義し、そのウィンドウを関数呼び出しから参照します。この方法は、同じウィンドウに基づくウィンドウ関数を複数指定する場合に便利です。このウィンドウ関数を使用するクエリを、独立したウィンドウ句を使用する方法で記述すると次のようになります (cumulative というウィンドウを宣言しています)。

```
SELECT p.id, p.description, s.quantity, s.ship_date,
   SUM(s.quantity) OVER(cumulative
                    ROWS BETWEEN UNBOUNDED PRECEDING
                            and CURRENT ROW
                    ) AS cumulative qty
FROM sales_order_items s JOIN product p On (s.prod_id =
  p.id)
WHERE s.ship_date BETWEEN '2005-07-01' and '2005-08-31'
Window cumulative as (PARTITION BY s.prod_id ORDER BY
  s.ship date)
ORDER BY p.id
```

このクエリ指定では、ウィンドウ句が ORDER BY 句の前にあることに注意してください。ウィンドウ句を使用するときには、次の制限が適用されます。

- インラインのウィンドウ指定に PARTITION BY 句を含めることはできません。
- ウィンドウ句で指定されるウィンドウにウィンドウ・フレーム句を含めることはできません。たとえば、「文法規則 32」(100 ページ) に次のように記述されています。

```
<WINDOW FRAME CLAUSE> ::=
   <WINDOW FRAME UNIT>
   <WINDOW FRAME EXTENT>
```

- インラインのウィンドウ指定にもウィンドウ句のウィンドウ指定にもウィンドウ順序句を含めることができますが、両方に含めることはできません。たとえば、「文法規則 31」(100 ページ) に次のように記述されています。

```
<WINDOW ORDER CLAUSE> ::= <ORDER SPECIFICATION>
```

例：複数の関数で使用されるウィンドウ

1 つの名前付きウィンドウを定義しておき、そのウィンドウに基づいて複数の関数を計算することもできます。具体的な例を次に示します。

```
SELECT p.id, p.description, s.quantity, s.ship_date,
   SUM(s.quantity) OVER ws1, MIN(s.quantity) OVER ws1
FROM sales_order_items s JOIN product p ON (s.prod_id =
   p.id) WHERE s.ship_date BETWEEN '1994-05-01' AND
   '1994-08-31' AND  s.quantity > 40 window ws1 AS
   (PARTITION BY prod_id ORDER BY ship_date rows
   between unbounded preceding and current row)
ORDER BY p.id;
```

このクエリの結果セットを次に示します。

ID	description	quantity	ship_date	sum	min
302	Crew Neck	60	1994-07-02	60	60
400	Cotton Cap	60	1994-05-26	60	60
400	Cotton Cap	48	1994-07-05	108	48
401	Wool cap	48	1994-06-02	48	48
401	Wool cap	60	1994-06-30	108	48
401	Wool cap	48	1994-07-09	156	48
500	Cloth Visor	48	1994-06-21	48	48
501	Plastic Visor	60	1994-05-03	60	60
501	Plastic Visor	48	1994-05-18	108	48
501	Plastic Visor	48	1994-05-25	156	48
501	Plastic Visor	60	1994-07-07	216	48
601	Zipped Sweatshirt	60	1994-07-19	60	60
700	Cotton Shorts	72	1994-05-18	72	72
700	Cotton Shorts	48	1994-05-31	120	48

例：累積和の計算

このクエリでは、ORDER BY start_date の順序に従って、部署別の給与の累積和を計算します。

```
SELECT dept_id, start_date, name, salary,
   SUM(salary) OVER (PARTITION BY dept_id ORDER BY
   start_date ROWS BETWEEN UNBOUNDED PRECEDING AND
   CURRENT ROW)
FROM emp1
ORDER BY dept_id, start_date;
```

このクエリの結果セットを次に示します。

dept_id	start_date	name	salary	sum(salary)
100	1996-01-01	Anna	18000	18000
100	1997-01-01	Mike	28000	46000
100	1998-01-01	Scott	29000	75000
100	1998-02-01	Antonia	22000	97000
100	1998-03-12	Adam	25000	122000
100	1998-12-01	Amy	18000	140000
200	1998-01-01	Jeff	18000	18000
200	1998-01-20	Tim	29000	47000
200	1998-02-01	Jim	22000	69000
200	1999-01-10	Tom	28000	97000
300	1998-03-12	Sandy	55000	55000
300	1998-12-01	Lisa	38000	93000
300	1999-01-10	Peter	48000	141000

例：移動平均の計算

このクエリでは、連続する 3 か月間の売上の移動平均を計算します。使用するウィンドウ・フレームは 3 つのローから成り、先行する 2 つのローと現在のローが含まれます。このウィンドウは、パーティションの最初から最後までスライドしていきます。

```
SELECT prod_id, month_num, sales, AVG(sales) OVER
   (PARTITION BY prod_id ORDER BY month_num ROWS
   BETWEEN 2 PRECEDING AND CURRENT ROW)
FROM sale WHERE rep_id = 1
ORDER BY prod_id, month_num;
```

このクエリの結果セットを次に示します。

prod_id	month_num	sales	avg(sales)
10	1	100	100.00
10	2	120	110.00
10	3	100	106.66
10	4	130	116.66
10	5	120	116.66

```
10              6       110         120.00
20              1        20          20.00
20              2        30          25.00
20              3        25          25.00
20              4        30          28.33
20              5        31          28.66
20              6        20          27.00
30              1        10          10.00
30              2        11          10.50
30              3        12          11.00
30              4         1           8.00
```

例：ORDER BY の結果

この例では、クエリの最上位の ORDER BY 句がウィンドウ関数の最終的な結果に適用されます。ウィンドウ句に指定されている ORDER BY は、ウィンドウ関数の入力データに適用されます。

```
SELECT prod_id, month_num, sales, AVG(sales) OVER
   (PARTITION BY prod_id ORDER BY month_num ROWS
   BETWEEN 2 PRECEDING AND CURRENT ROW)
FROM sale WHERE rep_id = 1
ORDER BY prod_id desc, month_num;
```

このクエリの結果セットを次に示します。

```
prod_id  month_num       sales   avg(sales)
-------  ---------       -----   ----------
30               1          10        10.00
30               2          11        10.50
30               3          12        11.00
30               4           1         8.00
20               1          20        20.00
20               2          30        25.00
20               3          25        25.00
20               4          30        28.33
20               5          31        28.66
20               6          20        27.00
10               1         100       100.00
10               2         120       110.00
10               3         100       106.66
10               4         130       116.66
10               5         120       116.66
10               6         110       120.00
```

例：1 つのクエリ内で複数の集合関数を使用

この例では、1 つのクエリ内で、異なるウィンドウに対して 2 種類の集合関数を実行しています。

```
SELECT prod_id, month_num, sales, AVG(sales) OVER
   (WS1 ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS
   CAvg, SUM(sales) OVER(WS1 ROWS BETWEEN UNBOUNDED
   PRECEDING AND CURRENT ROW) AS CSum
FROM sale WHERE rep_id = 1 WINDOW WS1 AS (PARTITION BY
   prod_id
ORDER BY month_num)
ORDER BY prod_id, month_num;
```

このクエリの結果セットを次に示します。

prod_id	month_num	sales	CAvg	CSum
10	1	100	110.00	100
10	2	120	106.66	220
10	3	100	116.66	320
10	4	130	116.66	450
10	5	120	120.00	570
10	6	110	115.00	680
20	1	20	25.00	20
20	2	30	25.00	50
20	3	25	28.33	75
20	4	30	28.66	105
20	5	31	27.00	136
20	6	20	25.50	156
30	1	10	10.50	10
30	2	11	11.00	21
30	3	12	8.00	33
30	4	1	6.50	34

例：ウィンドウ・フレーム指定の ROWS と RANGE の比較

このクエリでは、ROWS と RANGE を比較しています。ORDER BY 句の指定により、このデータには重複するローが含まれています。

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
   (ws1 RANGE BETWEEN 2 PRECEDING AND CURRENT ROW) AS
      Range_sum, SUM(sales) OVER
   (ws1 ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS
      Row_sum
FROM sale window ws1 AS (PARTITION BY prod_id ORDER BY
   month_num)
ORDER BY prod_id, month_num;
```

このクエリの結果セットを次に示します。

prod_id	month_num	sales	Range_sum	Row_sum
10	1	100	250	100

```
10              1     150         250       250
10              2     120         370       370
10              3     100         470       370
10              4     130         350       350
10              5     120         381       350
10              5      31         381       281
10              6     110         391       261
20              1      20          20        20
20              2      30          50        50
20              3      25          75        75
20              4      30          85        85
20              5      31          86        86
20              6      20          81        81
30              1      10          10        10
30              2      11          21        21
30              3      12          33        33
30              4       1          25        24
30              4       1          25        14
```

例：現在のローを除外するウィンドウ・フレーム

この例では、現在のローを除外するウィンドウ・フレームを定義しています。このクエリは、現在のローを除く 4 つのローの合計を計算します。

```
SELECT prod_id, month_num, sales, sum(sales) OVER
   (PARTITION BY prod_id ORDER BY month_num RANGE
   BETWEEN 6 PRECEDING AND 2 PRECEDING)
FROM sale
ORDER BY prod_id, month_num;
```

このクエリの結果セットを次に示します。

```
prod_id   month_num       sales    sum(sales)
-------   ---------       -----    ----------
10                1         100        (NULL)
10                1         150        (NULL)
10                2         120        (NULL)
10                3         100           250
10                4         130           370
10                5         120           470
10                5          31           470
10                6         110           600
20                1          20        (NULL)
20                2          30        (NULL)
20                3          25            20
20                4          30            50
20                5          31            75
20                6          20           105
30                1          10        (NULL)
30                2          11        (NULL)
30                3          12            10
30                4           1            21
30                4           1            21
```

例：ROW のデフォルトのウィンドウ・フレーム

このクエリは、ROW のデフォルトのウィンドウ・フレームの例を示しています。

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
   (PARTITION BY prod_id ORDER BY month_num RANGE
   BETWEEN 1 FOLLOWING AND 3 FOLLOWING)
FROM sale
ORDER BY prod_id, month_num;
```

このクエリの結果セットを次に示します。

prod_id	month_num	sales	sum(sales)
10	1	100	350
10	1	150	350
10	2	120	381
10	3	100	391
10	4	130	261
10	5	120	110
10	5	31	110
10	6	110	(NULL)
20	1	20	85
20	2	30	86
20	3	25	81
20	4	30	51
20	5	31	20
20	6	20	(NULL)
30	1	10	25
30	2	11	14
30	3	12	2
30	4	1	NULL)
30	4	1	(NULL)

例：UNBOUNDED PRECEDING と UNBOUNDED FOLLOWING

この例では、パーティション内のすべてのローがウィンドウ・フレームに含まれます。このクエリは、パーティション全体 (各月に重複ローは含まれていません) における売上の最大値を計算します。

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
   (PARTITION BY prod_id ORDER BY month_num ROWS
   BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
FROM sale WHERE rep_id = 1
ORDER BY prod_id, month_num;
```

このクエリの結果セットを次に示します。

prod_id	month_num	sales	max(sales)
10	1	100	680
10	2	120	680
10	3	100	680
10	4	130	680

```
10                5          120           680
10                6          110           680
20                1           20           156
20                2           30           156
20                3           25           156
20                4           30           156
20                5           31           156
20                6           20           156
30                1           10            34
30                2           11            34
30                3           12            34
30                4            1            34
```

このクエリは、次のクエリと同じ意味になります。

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
   (PARTITION BY prod_id )
FROM sale WHERE rep_id = 1
ORDER BY prod_id, month_num;
```

例：RANGE のデフォルトのウィンドウ・フレーム

このクエリは、RANGE のデフォルトのウィンドウ・フレームの例を示しています。

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
   (PARTITION BY prod_id ORDER BY month_num)
FROM sale
ORDER BY prod_id, month_num;
```

このクエリの結果セットを次に示します。

```
prod_id   month_num       sales    max(sales)
-------   ---------       -----    ----------
10                1         100           250
10                1         150           250
10                2         120           370
10                3         100           470
10                4         130           600
10                5         120           751
10                5          31           751
10                6         110           861
20                1          20            20
20                2          30            50
20                3          25            75
20                4          30           105
20                5          31           136
20                6          20           156
30                1          10            10
30                2          11            21
30                3          12            33
```

```
30              4           1           35
30              4           1           35
```

このクエリは、次のクエリと同じ意味になります。

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
   (PARTITION BY prod_id ORDER BY month_num RANGE
   BETWEEEN UNBOUNDED PRECEDING AND CURRENT ROW)
FROM sale
ORDER BY prod_id, month_num;
```

OLAP 関数の BNF 文法

次の BNF (Backus-Naur Form) 文法は、さまざまな ANSI SQL 統計関数に関する具体的な構文サポートの概要を示しています。ここに記載されている関数の多くは Sybase IQ で実装されています。

文法規則 1

```
<SELECT LIST EXPRESSION> ::=
   <EXPRESSION>
   | <GROUP BY EXPRESSION>
   | <AGGREGATE FUNCTION>
   | <GROUPING FUNCTION>
   | <TABLE COLUMN>
   | <WINDOWED TABLE FUNCTION>
```

文法規則 2

```
<QUERY SPECIFICATION> ::=
   <FROM CLAUSE>
   [ <WHERE CLAUSE> ]
   [ <GROUP BY CLAUSE> ]
   [ <HAVING CLAUSE> ]
   [ <WINDOW CLAUSE> ]
[ <ORDER BY CLAUSE> ]
```

文法規則 3

```
<ORDER BY CLAUSE> ::= <ORDER SPECIFICATION>
```

文法規則 4

```
<GROUPING FUNCTION> ::=
   GROUPING <LEFT PAREN> <GROUP BY EXPRESSION>
   <RIGHT PAREN>
```

文法規則 5

```
<WINDOWED TABLE FUNCTION> ::=
   <WINDOWED TABLE FUNCTION TYPE> OVER <WINDOW NAME OR
   SPECIFICATION>
```

文法規則 6

```
<WINDOWED TABLE FUNCTION TYPE> ::=
   <RANK FUNCTION TYPE> <LEFT PAREN> <RIGHT PAREN>
   | ROW_NUMBER <LEFT PAREN> <RIGHT PAREN>
   | <WINDOW AGGREGATE FUNCTION>
```

文法規則 7

```
<RANK FUNCTION TYPE> ::=
   RANK | DENSE RANK | PERCENT RANK | CUME_DIST
```

文法規則 8

```
<WINDOW AGGREGATE FUNCTION> ::=
    <SIMPLE WINDOW AGGREGATE FUNCTION>
    | <STATISTICAL AGGREGATE FUNCTION>
```

文法規則 9

```
<AGGREGATE FUNCTION> ::=
    <DISTINCT AGGREGATE FUNCTION>
    | <SIMPLE AGGREGATE FUNCTION>
    | <STATISTICAL AGGREGATE FUNCTION>
```

文法規則 10

```
<DISTINCT AGGREGATE FUNCTION> ::=
    <BASIC AGGREGATE FUNCTION TYPE> <LEFT PAREN>
    <DISTINCT> <EXPRESSION> <RIGHT PAREN>
    | LIST <LEFT PAREN> DISTINCT <EXPRESSION>
    [ <COMMA> <DELIMITER> ]
    [ <ORDER SPECIFICATION> ] <RIGHT PAREN>
```

文法規則 11

```
<BASIC AGGREGATE FUNCTION TYPE> ::=
    SUM | MAX | MIN | AVG | COUNT
```

文法規則 12

```
<SIMPLE AGGREGATE FUNCTION> ::=
    <SIMPLE AGGREGATE FUNCTION TYPE> <LEFT PAREN>
    <EXPRESSION> <RIGHT PAREN>
    | LIST <LEFT PAREN> <EXPRESSION> [ <COMMA>
    <DELIMITER> ]
    [ <ORDER SPECIFICATION> ] <RIGHT PAREN>
```

文法規則 13

```
<SIMPLE AGGREGATE FUNCTION TYPE> ::= <SIMPLE WINDOW AGGREGATE
FUNCTION TYPE>
```

文法規則 14

```
<SIMPLE WINDOW AGGREGATE FUNCTION> ::=
    <SIMPLE WINDOW AGGREGATE FUNCTION TYPE> <LEFT PAREN>
    <EXPRESSION> <RIGHT PAREN>
| GROUPING FUNCTION
```

文法規則 15

```
<SIMPLE WINDOW AGGREGATE FUNCTION TYPE> ::=
    <BASIC AGGREGATE FUNCTION TYPE>
    | STDDEV | STDDEV_POP | STDDEV_SAMP
    | VARIANCE | VARIANCE_POP | VARIANCE_SAMP
```

文法規則 16

```
<STATISTICAL AGGREGATE FUNCTION> ::=
    <STATISTICAL AGGREGATE FUNCTION TYPE> <LEFT PAREN>
    <DEPENDENT EXPRESSION> <COMMA> <INDEPENDENT
    EXPRESSION> <RIGHT PAREN>
```

文法規則 17

```
<STATISTICAL AGGREGATE FUNCTION TYPE> ::=
    CORR | COVAR_POP | COVAR_SAMP | REGR_R2 |
    REGR_INTERCEPT | REGR_COUNT | REGR_SLOPE |
    REGR_SXX | REGR_SXY | REGR_SYY | REGR_AVGY |
    REGR_AVGX
```

文法規則 18

```
<WINDOW NAME OR SPECIFICATION> ::=
    <WINDOW NAME> | <IN-LINE WINDOW SPECIFICATION>
```

文法規則 19

```
<WINDOW NAME> ::= <IDENTIFIER>
```

文法規則 20

```
<IN-LINE WINDOW SPECIFICATION> ::= <WINDOW SPECIFICATION>
```

文法規則 21

```
<WINDOW CLAUSE> ::= <WINDOW WINDOW DEFINITION LIST>
```

文法規則 22
```
<WINDOW DEFINITION LIST> ::=
    <WINDOW DEFINITION> [ { <COMMA> <WINDOW DEFINITION>
    } . . . ]
```

文法規則 23
```
<WINDOW DEFINITION> ::=
    <NEW WINDOW NAME> AS <WINDOW SPECIFICATION>
```

文法規則 24
```
<NEW WINDOW NAME> ::= <WINDOW NAME>
```

文法規則 25
```
<WINDOW SPECIFICATION> ::=
    <LEFT PAREN> <WINDOW SPECIFICATION> <DETAILS> <RIGHT
    PAREN>
```

文法規則 26
```
<WINDOW SPECIFICATION DETAILS> ::=
    [ <EXISTING WINDOW NAME> ]
    [ <WINDOW PARTITION CLAUSE> ]
    [ <WINDOW ORDER CLAUSE> ]
    [ <WINDOW FRAME CLAUSE> ]
```

文法規則 27
```
<EXISTING WINDOW NAME> ::= <WINDOW NAME>
```

文法規則 28
```
<WINDOW PARTITION CLAUSE> ::=
    PARTITION BY <WINDOW PARTITION EXPRESSION LIST>
```

文法規則 29
```
<WINDOW PARTITION EXPRESSION LIST> ::=
    <WINDOW PARTITION EXPRESSION>
    [ { <COMMA> <WINDOW PARTITION EXPRESSION> } . . . ]
```

文法規則 30
```
<WINDOW PARTITION EXPRESSION> ::= <EXPRESSION>
```

文法規則 31
```
<WINDOW ORDER CLAUSE> ::= <ORDER SPECIFICATION>
```

文法規則 32
```
<WINDOW FRAME CLAUSE> ::=
    <WINDOW FRAME UNIT>
    <WINDOW FRAME EXTENT>
```

文法規則 33
```
<WINDOW FRAME UNIT> ::= ROWS | RANGE
```

文法規則 34
```
<WINDOW FRAME EXTENT> ::= <WINDOW FRAME START> | <WINDOW FRAME
BETWEEN>
```

文法規則 35
```
<WINDOW FRAME START> ::=
    UNBOUNDED PRECEDING
    | <WINDOW FRAME PRECEDING>
    | CURRENT ROW
```

文法規則 36
```
<WINDOW FRAME PRECEDING> ::= <UNSIGNED VALUE SPECIFICATION>
PRECEDING
```

文法規則 37
```
<WINDOW FRAME BETWEEN> ::=
    BETWEEN <WINDOW FRAME BOUND 1> AND <WINDOW FRAME
    BOUND 2>
```

文法規則 38
```
<WINDOW FRAME BOUND 1> ::= <WINDOW FRAME BOUND>
```

文法規則 39
```
<WINDOW FRAME BOUND 2> ::= <WINDOW FRAME BOUND>
```

文法規則 40

```
<WINDOW FRAME BOUND> ::=
    <WINDOW FRAME START>
    | UNBOUNDED FOLLOWING
    | <WINDOW FRAME FOLLOWING>
```

文法規則 41

```
<WINDOW FRAME FOLLOWING> ::= <UNSIGNED VALUE SPECIFICATION>
FOLLOWING
```

文法規則 42

```
<GROUP BY EXPRESSION> ::= <EXPRESSION>
```

文法規則 43

```
<SIMPLE GROUP BY TERM> ::=
    <GROUP BY EXPRESSION>
    | <LEFT PAREN> <GROUP BY EXPRESSION> <RIGHT PAREN>
    | <LEFT PAREN> <RIGHT PAREN>
```

文法規則 44

```
<SIMPLE GROUP BY TERM LIST> ::=
    <SIMPLE GROUP BY TERM> [ { <COMMA> <SIMPLE GROUP BY
    TERM> } . . . ]
```

文法規則 45

```
<COMPOSITE GROUP BY TERM> ::=
    <LEFT PAREN> <SIMPLE GROUP BY TERM>
    [ { <COMMA> <SIMPLE GROUP BY TERM> } . . . ]
    <RIGHT PAREN>
```

文法規則 46

```
<ROLLUP TERM> ::= ROLLUP <COMPOSITE GROUP BY TERM>
```

文法規則 47

```
<CUBE TERM> ::= CUBE <COMPOSITE GROUP BY TERM>
```

文法規則 48

```
<GROUP BY TERM> ::=
    <SIMPLE GROUP BY TERM>
    | <COMPOSITE GROUP BY TERM>
    | <ROLLUP TERM>
    | <CUBE TERM>
```

文法規則 49

```
<GROUP BY TERM LIST> ::=
    <GROUP BY TERM> [ { <COMMA> <GROUP BY TERM> } . . . ]
```

文法規則 50

```
<GROUP BY CLAUSE> ::= GROUP BY <GROUPING SPECIFICATION>
```

文法規則 51

```
<GROUPING SPECIFICATION> ::=
    <GROUP BY TERM LIST>
    | <SIMPLE GROUP BY TERM LIST> WITH ROLLUP
    | <SIMPLE GROUP BY TERM LIST> WITH CUBE
    | <GROUPING SETS SPECIFICATION>
```

文法規則 52

```
<GROUPING SETS SPECIFICATION> ::=
    GROUPING SETS <LEFT PAREN> <GROUP BY TERM LIST>
    <RIGHT PAREN>
```

文法規則 53

```
<ORDER SPECIFICATION> ::= ORDER BY <SORT SPECIFICATION LIST>
```

第 5 章

システム・リソースの管理

この章について

この章では、Sybase IQ によるメモリ、ディスク I/O、CPU の使用方法と、これらの要素間の関係について説明します。また、DBA がリソース使用量を調整してパフォーマンスをチューニングする方法についても説明します。

この章の説明は、一般的な内容となっています。ご使用のハードウェアとソフトウェアの設定に合わせて調整してください。プラットフォームごとの推奨事項については、それぞれの『Sybase IQ インストールおよび設定ガイド』を参照してください。

内容

パフォーマンス用語の概要

パフォーマンスとは、コンピュータ・ビジネス・アプリケーションまたは同じ環境内で動作する複数のアプリケーションの効率を表す尺度です。通常、この効率は応答時間とスループットで測定します。

応答時間とは、1 つのタスクが完了するまでにかかる時間のことです。応答時間は、次の項目の影響により変化します。

- 競合の軽減と待機時間 (特にディスク I/O 待機時間) の短縮
- より高速なコンポーネントの使用
- リソースに必要な時間の短縮 (同時実行性の向上)

スループットは、一定の時間にどれだけの作業量が完了したかを表します。スループットは、通常、1 秒あたりのトランザクション数で表されますが、1 分、1 時間、1 日などの単位で測定する場合もあります。

パフォーマンス向上のための設計

適正なデータベース設計、緻密なクエリ分析、適切なインデックス付けを行うことにより、アプリケーションはより高いパフォーマンスを発揮することができます。適正な設計を行い、適切なインデックス付け方式を選択することによって、パフォーマンスを最も向上させることができます。

その他、ハードウェアやネットワークを分析することによって、インストール環境のボトルネックを特定できます。

詳細については、「第 3 章 クエリと削除の最適化」を参照してください。

メモリ使用の概要

Sybase IQ では、次のような目的でメモリを使用します。

- クエリの解析用にディスクから読み込むデータのバッファ
- フラット・ファイルからロードするときにディスクから読み込むデータのためのバッファ
- 接続、トランザクション、バッファ、データベース・オブジェクトを管理するためのオーバヘッド

以降の各項では、オペレーティング・システムが Sybase IQ のメモリ使用をサポートする方法、さまざまな目的のために Sybase IQ でメモリを割り付ける方法、パフォーマンスを改善するためにユーザがメモリ割り付けを調整する方法、また Sybase IQ に十分なメモリが使用できるようにオペレーティング・システムを設定する手順について説明します。

ページングによる使用可能メモリの増加

システムのメモリが不足している場合、パフォーマンスが大幅に低下することがあります。このような場合、使用可能なメモリを増やす必要があります。RDBMS ソフトウェアのように、Sybase IQ にも多くのメモリが必要です。Sybase IQ に割り付け可能なメモリが多ければ多いほど、パフォーマンスも向上します。

ただし、システム内のメモリ量には常に一定の制限があるため、データの一部のみがメモリに格納され、残りのデータはディスク上に格納されるという状況が発生します。オペレーティング・システムが、ディスク上のデータを検索して取り出し、メモリ要求に対応する必要がある場合、これをページングまたはスワッピングと呼びます。メモリを適切に管理することの主な目的は、ページングやスワッピングを回避したり、最小限に抑えたりすることです。

最も頻繁に使用されるオペレーティング・システム・ファイルは、スワップ・ファイルです。メモリが消耗している場合、オペレーティング・システムがメモリのページをディスクにスワップして、新しいデータの領域を確保します。スワップされたページを再び呼び出すと、他のページがスワップされて、要求されたメモリ・ページが元に戻ります。ユーザのディスク使用率が高い場合、スワッピングには時間がかかります。通常は、スワッピングが起こらないようなメモリ編成にして、オペレーティング・システム・ファイルの使用を最小限に抑えてください。スワッピングを最小限に抑えるためのメモリの設定方法については、「プラットフォーム固有のメモリ・オプション」(119 ページ) を参照してください。

Sybase IQ では、物理メモリを最大限利用するために、データベースに対するすべての読み込みと書き込みにバッファ・キャッシュを使用します。

注意 ディスク上のスワップ領域には、少なくとも物理メモリ全体を収容できるだけのサイズを確保します。

スワッピングをモニタするためのユーティリティ

UNIX vmstat コマンド、UNIX sar コマンド、または Windows タスク・マネージャを使用すると、実行中のプロセス数、ページアウト回数、スワップ回数についての統計を表示できます。この統計によって得た情報を使用して、システムでページングが過度に発生していないかどうかを調べてください。必要に応じて、設定を調整します。たとえば、特殊な高速ディスクにスワップ・ファイルを配置します。

vmstat の出力例については、「UNIX システムでのページングのモニタリング」を参照してください。

サーバ・メモリ

さまざまな目的に応じて、Sybase IQ は、サーバ・メモリと呼ばれる単一メモリ・プールからメモリを割り付けます。サーバ・メモリには、バッファ、トランザクション、データベース、サーバの管理のために割り付けられたすべてのメモリが含まれます。

オペレーティング・システム・レベルでは、Sybase IQ サーバ・メモリはヒープ・メモリで構成されます。ほとんどの場合、Sybase IQ で使用されるメモリがヒープ・メモリか共有メモリかを気にする必要はありません。メモリ割り付けは、すべて自動的に処理されます。ただし、Sybase IQ を実行する前に、オペレーティング・システム・カーネルが共有メモリを使用するように正しく構成されていることを確認してください。詳細については、プラットフォームに対応した『Sybase IQ インストールおよび設定ガイド』を参照してください。

マルチプレックスのメモリ管理

マルチプレックスの各サーバは、独自のホスト上にある場合と、ホストを他のサーバと共有している場合があります。複数のサーバが同じシステム上にある場合、作業負荷の処理にかかる CPU 時間は、単一の組み合わされたサーバの場合とほとんど変わりません。しかし、独立した複数のサーバでは、単一の組み合わされたサーバより多くの物理メモリが必要になります。これは、各サーバが使用するメモリを他のサーバが共有できないからです。

ロード、挿入、更新、同期、削除のためのメモリ

マシンの物理メモリが過度に割り付けられるのを防ぐには、ロードが発生する操作に対して LOAD_MEMORY_MB データベース・オプションを設定します。LOAD 操作だけでなく、このオプションは INSERT、UPDATE、SYNCHRONIZE、DELETE の各操作にも影響を与えます。LOAD_MEMORY_MB オプションでは、それ以降のロードで使用できるヒープ・メモリ量の上限を MB 単位で指定します。ロードとバッファ・キャッシュの使用については、「ロードのメモリ要件」(109 ページ) を参照してください。LOAD_MEMORY_MB オプションの詳細については、『Sybase IQ リファレンス・マニュアル』の「第 2 章 データベース・オプション」を参照してください。

プロセスの中止による共有メモリへの影響

警告！ UNIX システムで中止されたプロセスは、自動的にクリーンアップされずに、セマフォまたは共有メモリにそのまま残されます。UNIX 上の Sybase IQ サーバを正常に停止するには、『Sybase IQ システム管理ガイド』の「第 2 章 Sybase IQ の実行」の「データベース・サーバの停止」に説明されている stop_asiq ユーティリティを使用します。ipcs と ipcrm を使用して、異常終了後のクリーンアップを行う方法については、『Sybase IQ トラブルシューティングおよびリカバリ・ガイド』の「第 1 章 トラブルシューティングのヒント」を参照してください。

バッファ・キャッシュの管理

Sybase IQ では、バッファ・キャッシュに最も多くのメモリが必要です。Sybase IQ には、IQ ストア用とテンポラリ・ストア 用の 2 つのバッファ・キャッシュがあります。ページング、データベースへの挿入、バックアップやリストアなどのすべてのデータベース I/O 操作にこの 2 つのバッファ・キャッシュが使用されます。メモリ内にあるデータは、この 2 つのいずれかに格納されます。すべてのユーザ接続は、これらのバッファ・キャッシュを共有します。Sybase IQ が、各接続にどのデータが関連付けられているかを追跡します。

バッファ・キャッシュの管理の詳細については、以降の項を参照してください。

- メモリ要件の計算方法については、「バッファ・キャッシュ・サイズの決定」を参照してください。
- 設定サイズがわかっているバッファ・キャッシュ・サイズの設定方法については、「バッファ・キャッシュ・サイズの設定」を参照してください。
- 適切なバッファ・キャッシュ・サイズを決定する方法の例については、表 5-1 (112 ページ) を参照してください。

バッファ・キャッシュ・サイズの決定

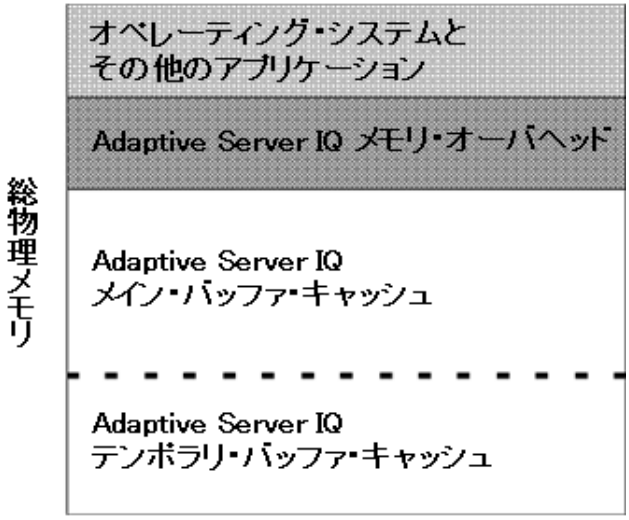
IQ ストアとテンポラリ・ストアに指定するバッファ・キャッシュ・サイズは、さまざまな要因によって決まります。デフォルト値 (メインに 16MB、テンポラリ・キャッシュに 12MB) では、ほとんどのデータベースでサイズが不足します。アプリケーションに必要な実際の値は、次の要因によって決まります。

- システムに搭載されている物理メモリの合計量
- Sybase IQ、オペレーティング・システム、その他のアプリケーションがそれぞれのタスクを実行するのに必要な物理メモリの量
- ロード、クエリ、またはその両方を実行するかどうか

表 5-1 (112 ページ) の例を通して、サイトに最適な設定を決定するときは、これ以降の各項で示されているガイドラインを参考にしてください。

次の図は、バッファ・キャッシュとその他のメモリ消費との関係を示します。

図 5-1: 物理メモリに対するバッファ・キャッシュ



各部分に関する詳細な説明、各部分に必要なメモリ量の判断に役立つガイドラインについては、以降の項で説明します。

オペレーティング・システムとその他のアプリケーション

このメモリの量は、プラットフォームの種類やシステムの使用方法によって異なります。たとえば、UNIX ファイル・システムでは、UNIX ロー・パーティションよりも多くのファイル・バッファリングが実行されるため、オペレーティング・システムに必要なメモリは多くなります。少なくとも、UNIX システムでは 60MB 以上、Windows システムでは 30MB 以上のメモリを使用すると仮定できます。

また、Sybase IQ とともに動作する他のアプリケーション (クエリ・ツールなど) にもそれぞれメモリ要件があります。メモリ要件については、アプリケーションおよびオペレーティング・システムのマニュアルを参照してください。

Sybase IQ メモリ・オーバヘッド

オペレーティング・システムや他のアプリケーションで使用する物理メモリ量を決定すると、Sybase IQ がタスクの実行に必要とする残りのメモリ量を計算できます。このオーバヘッドに影響を与える要因については、以降の各項で説明します。

ロー・パーティションとファイル・システム

UNIX システムの場合、ロー・パーティションではなくファイル・システムを使用するデータベースには、オペレーティング・システムによるファイル・バッファリング処理のために残りのメモリの 30% がさらに必要になります。Windows では、OS_FILE_CACHE_BUFFERING = 'OFF' に設定し (新しいデータベースのデフォルト)、ファイル・システム・キャッシュを無効にしてください。詳細については、プラットフォームに対応した『Sybase IQ インストールおよび設定ガイド』を参照してください。

マルチユーザのデータベース・アクセス

マルチユーザがデータベースをクエリする場合、Sybase IQ には「アクティブ」ユーザ 1 人あたり約 10MB のメモリが必要です。アクティブ・ユーザとは、同時にデータベースにアクセスしたり、データベースに対して問い合わせを行ったりするユーザのことです。たとえば、Sybase IQ に接続しているユーザが 30 人でも 、アクティブにデータベースを同時に使用しているユーザは 10 人ほどしかいないことがあります。

ロードのメモリ要件

Sybase IQ では、バッファ・キャッシュの他に、ロード操作、同期、削除を実行するためのメモリも必要です。このメモリは、フラット・ファイルの I/O バッファリングに使用されます。Sybase IQ ではメモリを使用して、ディスクから読み込んだデータをバッファリングします。この読み込みのサイズは、BLOCK FACTOR に入力レコードのサイズを乗算した値になります。BLOCK FACTOR は、LOAD TABLE コマンドのオプションです。デフォルト値の 10,000 を使用した場合、データの入力ローを 200 バイトとすると、Sybase IQ が I/O バッファリングに使用するサイズは合計 2MB になります。ロードのメモリ要件は、ローの数ではなく、カラムの数と幅によって決まります。

このメモリが必要になるのは、フラット・ファイルからロードする場合、INSERT..LOCATION または INSERT..SELECT を使用する場合だけです。削除と更新に必要なメモリは比較的少量です。

スレッド・スタックのメモリ

スレッドの処理には、少量のメモリが必要です。使用する Sybase IQ 処理スレッドが多くなるにつれ、必要なメモリも多くなります。-iqmt サーバ・スイッチ は、Sybase IQ のスレッド数を制御します。

ユーザの数に比例して、カタログ・ストア処理スレッドに必要なメモリも増加しますが、必要とされるメモリは比較的少量です。-gn スイッチによって、カタログ・ストア処理スレッドを制御します。

スレッドの合計数 (-iqmt と -gn の合計) が、現在のプラットフォームで使用できるスレッド数を超えないようにします。詳細については、『Sybase IQ ユーティリティ・ガイド』の「第 1 章 データベース・サーバの実行」を参照してください。

その他のメモリ使用

すべてのコマンドとトランザクションが、ある程度のメモリを使用します。これまで説明してきた要因の他に、メモリを大量に使用する操作には次のものがあります。

バックアップ　バックアップに使用される仮想メモリの量は、データベース作成時に指定された IQ PAGE SIZE によって決まります。この値はおよそ 2 * CPU の数 * 20 * (IQ PAGE SIZE/16) です。プラットフォームによっては、BACKUP コマンドの BLOCK FACTOR を調整するとバックアップのパフォーマンスが向上する場合がありますが、BLOCK FACTOR を増やすとメモリの使用量も増加します。詳細については、『Sybase IQ システム管理ガイド』の「第 14 章 データのバックアップ、リカバリ、アーカイブ」の「バックアップ中の使用メモリの増加」を参照してください。

データベースの検証と修復　データベース全体を検証すると、sp_iqcheckdb プロシージャは処理を開始する前に、すべての Sybase IQ テーブル、テーブルのそれぞれのフィールドとインデックスを開きます。Sybase IQ テーブルの数、およびテーブル内のカラムとインデックスの累積数によって、sp_iqcheckdb に必要な仮想メモリの量は大幅に異なります。必要なメモリ量を制限するには、sp_iqcheckdb オプションを使って 1 つのインデックスまたはテーブルを検証または修復します。

リーク・ブロックの削除　リークの削除操作でも、すべての Sybase IQ テーブル、ファイル、インデックスを開く必要があるため、データベース全体を検証するときに sp_iqcheckdb が使用するのと同じ容量の仮想メモリを使用します。Sybase IQ テンポラリ・バッファ・キャッシュを使用して、使用ブロックを追跡します。

Sybase IQ のメイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュ

Sybase IQ に必要なオーバヘッド・メモリを決定したら、メイン Sybase IQ バッファ・キャッシュとテンポラリ・バッファ・キャッシュの間でどのように残りのメモリを分散するかを決定してください。図 5-1 では、2 つの領域が点線で分割されていますが、これはさまざまな要因によって分割線の位置が移動する可能性があることを示します。

他のほとんどのデータベースと異なり、Sybase IQ では、通常、メイン・バッファ・キャッシュに 40 %、テンポラリ・バッファ・キャッシュに 60% のメモリを割り当てます。ただし、この比率は固定されているわけではありません。HG インデックスが関連する大きなソートマージ・ジョインまたは挿入を使用するクエリなどの操作では、通常とは異なり、メイン・バッファ・キャッシュよりもテンポラリ・バッファ・キャッシュに多くのメモリが必要です。また、他のアプリケーションでは要件が異なることがあります。Sybase IQ でサポートされるメモリ割り付けの割合は、メイン・バッファ・キャッシュ対テンポラリ・バッファ・キャッシュが 30 % 対 70 % から 70 % 対 30 % の範囲となっています。

注意　上記のガイドラインは、システムでアクティブなデータベースが一度に 1 つであることを前提としています。つまり、すべての Sybase IQ ユーザが 1 つのデータベースだけにアクセスしている場合です。複数のアクティブなデータベースがある場合は、使用するデータベース間で残りのメモリをさらに分ける必要があります。

最初は、ここに記載した一般ガイドラインに従い、「バッファ・キャッシュのモニタリング」(157 ページ) で説明されているモニタ・ツールと、現在のプラットフォームに対応する『Sybase IQ インストールおよび設定ガイド』に記載された個別のツールを使用して、Sybase IQ のパフォーマンスをモニタリングすることを強くおすすめします。

バッファ・キャッシュと物理メモリ

Sybase IQ のメイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュに使用するメモリと、Sybase IQ メモリ・オーバヘッド、オペレーティング・システムとその他のアプリケーションに使用するメモリの合計が、システムの物理メモリを超えないようにしてください。

システムが許容するサイズよりもバッファ・キャッシュ・サイズを大きく設定すると、Sybase IQ はデータベースを開くことができません。サーバ起動オプションの -iqmc (メイン・キャッシュ・サイズ) と -iqtc (テンポラリ・キャッシュ・サイズ) を指定してデータベースを開き、キャッシュ・サイズを再設定してください。デフォルトでは、メイン・キャッシュに 16MB、テンポラリ・キャッシュに 12MB しか割り付けられません。

注意 一部の UNIX プラットフォームでは、他のサーバ・スイッチを設定してバッファ・キャッシュに使用可能なメモリを増やす必要があります。詳細については、「プラットフォーム固有のメモリ・オプション」(119 ページ) を参照してください。

その他の注意事項

Sybase IQ のバッファ・キャッシュ・サイズは、データベースをどのように使うかによって変わります。パフォーマンスを最大にするには、データベースの挿入、問い合わせ、およびその両方を使用するそれぞれの場合に応じて設定を変更してください。ただし、データベースへの挿入と問い合わせを両方使用する環境では、すべてのユーザによるデータベースの使用を中止し、バッファ・キャッシュ・オプションをリセットすることは容易でありません。このような場合は、ロードまたはクエリのどちらかのパフォーマンスを優先させてください。可能なかぎり、データベースで作業を行う前にキャッシュ・サイズを定義してください。

バッファ・キャッシュとメモリ・オーバヘッドのガイドラインも、プラットフォームによって異なります。その他の事項については、『Sybase IQ インストールおよび設定ガイド』を参照してください。

バッファ・キャッシュ・サイズの設定例

次の表には、システムのメモリが消費される要因の一覧と、メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュに残されたメモリ量の例を示します。この例では、システムに 1GB の物理メモリがあること、実行中の Sybase IQ 以外にはハードウェア上に大きなアプリケーションがないこと、アクティブ・データベースは一度に 1 つしかないことを前提にしています。この表では、主なデータベース・アクセスの種類 (クエリまたは挿入) で数字を分けています。

表 5-1: バッファ・キャッシュに使用可能なメモリの例 (MB 単位)

メモリの使用	使用量	使用可能なメモリ : クエリ	使用可能なメモリ : 挿入
使用可能な物理メモリの合計 (MB 単位の概数)		1000	1000
UNIX システムの最小量を前提としたオペレーティング・システム使用量	100[a]	900	900
アクティブ・ユーザ数のオーバヘッド : 接続ユーザ数は約 30 人だが、アクティブ・ユーザ数は約 10 人のみで、1 人あたり 10MB	100	825	
200 バイトのレコード・サイズとデフォルト設定を前提としたときの、フラット・ファイルからの挿入に伴うオーバヘッド	97		828

メモリの使用	使用量	使用可能なメモリ：クエリ	使用可能なメモリ：挿入
メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュ用に残されたメモリ		675	828
iqmc (Main_Cache_Memory_MB) の設定：バッファ・キャッシュ用に残されたメモリの 40%		405	497
iqtc (Temp_Cache_Memory_MB) の設定：バッファ・キャッシュ用に残されたメモリの 60%		270	331

[a]Windows オペレーティング・システムでは、最低 30MB 必要です。

表に示されているように、データベースへの挿入が中心の場合と、問い合わせが中心の場合とでは、バッファ・キャッシュに対して使用する値の組が異なります。これらの値は、挿入と問い合わせの負荷の割合が一般的な場合とは異なります。キャッシュ・サイズを変更する場合は、「バッファ・キャッシュ・サイズの設定」を参照してください。キャッシュ・サイズ・オプションは、データベースを停止して再起動するまで使用できません

バッファ・キャッシュ・サイズの設定

Sybase IQ では、メイン・バッファ・キャッシュは 16MB に、テンポラリ・バッファ・キャッシュは 12MB のサイズにデフォルト設定されています。ほとんどのアプリケーションでは、これよりはるかに大きな値を必要とします。キャッシュのサイズは、物理メモリの合計によって制限されます。使用中のシステムの適切な設定を決定するには、前の各項を参照してください。

必要な設定がわかったら、表 5-2 に示すオプションを使用してバッファ・キャッシュ・サイズを設定します。表 5-3 に示すオプションを使用して、バッファ・キャッシュに使用するメモリを増やすこともできます。

表 5-2: バッファ・キャッシュ・サイズを変更する設定

方法	使用時期	設定の有効期間	参照先
-iqmc および -iqtc サーバ・スイッチ	推奨される方法。データベースとサーバが動作していないときにキャッシュ・サイズを設定する。4GB を超えるキャッシュ・サイズを使用できる。 64 ビット・プラットフォームを使用している場合、またはキャッシュ・サイズのデータベース・オプションがシステムの許容量を超えて設定されている場合に特に有用。	サーバが起動してから停止するまで	「バッファ・キャッシュ・サイズのサーバ・スイッチの設定」(114 ページ)

方法	使用時期	設定の有効期間	参照先
MAIN_CACHE_MEMORY_MB および TEMP_CACHE_MEMORY_MB database データベース・オプション	バッファ・キャッシュ・サイズを最大 4GB に設定。これらの値を設定するにはデータベースを開く必要がある。 この方法は推奨しません。	次回データベースが再始動されてから、これらのオプションをリセットするか、またはサーバ・スイッチでこれらのオプションを上書きするまで	『Sybase IQ リファレンス・マニュアル』の「データベース・オプション」の章

表 5-3: バッファ・キャッシュに使用できるメモリ容量に影響する設定

方法	使用時期	設定の有効期間	参照先
-iqwmem サーバ・スイッチ	バッファ・キャッシュとして使用する追加メモリを割り当てるときに一部の UNIX プラットフォーム上で使用。実際にキャッシュ・サイズの設定はしない。	サーバが起動してから停止するまで	「プラットフォーム固有のメモリ・オプション」(119 ページ)
LOAD_MEMORY_MB データベース・オプション	ロードに使用可能なメモリを制御して、バッファ・キャッシュ・サイズに間接的に影響を与える場合。プラットフォームによっては、ロードに無制限のメモリを許可すると、バッファ・キャッシュに使用可能なメモリが少なくなる。	すぐに有効 (オプションをリセットするまで)	「ロード、挿入、更新、同期、削除のためのメモリ」(106 ページ)

バッファ・キャッシュ・サイズのサーバ・スイッチの設定

サーバ起動オプションの -iqmc と -iqtc を使用するバッファ・キャッシュ・サイズの設定には、次の 2 つの重要な利点があります。

- 4GB を超えるサイズのキャッシュ・サイズを設定できます。32 ビット・プラットフォームでは、このような大きいキャッシュ・サイズは使用できませんが、64 ビット・プラットフォームでは推奨されています。
- システムの許容量を超えるデータベース・オプションを設定してしまい、データベースを開けない場合に、キャッシュ・サイズを変更できます。

データベース・オプションとサーバ・オプションのどちらを使用する場合も、バッファ・キャッシュ・サイズを変更するにはサーバを再起動する必要があります。-iqmc および -iqtc サーバ起動オプションは、サーバが実行されている間だけ有効なため、サーバを再起動するたびに指定する必要があります。

ページ・サイズの指定

データベースを作成するときは、そのページ・サイズを設定します。このパラメータは、バッファ・キャッシュのサイズとともに、そのデータベースのメモリ使用とディスク I/O スループットを決定します。

注意　ページ・サイズを変更することはできません。ページ・サイズによって、一部のデータベース・オブジェクトのサイズの上限が決まります。

ページ・サイズの設定

Sybase IQ では、**ページ**単位でデータがメモリ内外にスワップされます。データベースを作成するときに、カタログ・ストアと IQ ストアに別々のページ・サイズを指定します。テンポラリ・ストアのページ・サイズは、IQ ストアと同じです。

パフォーマンスを最大にするための Sybase IQ ページ・サイズの推奨値については、『Sybase IQ システム管理ガイド』の「第 5 章 データベース・オブジェクトの使用」の「IQ ページ・サイズの選択」を参照してください。

カタログ・ストアで発生する I/O はわずかなため、カタログ・ストアのページ・サイズがパフォーマンスに与える影響はほとんどありません。デフォルト値 4096 バイトで十分です。

IQ ページ・サイズによって、データベースのデフォルト I/O 転送ブロック・サイズと最大データ圧縮の 2 つのパフォーマンスの要因が決まります。以降の項で、これらの要因について説明します。

ブロック・サイズ

すべての I/O は、**ブロック**単位で発生します。これらのブロックのサイズは、Sybase IQ データベースを作成したときに設定したものです。このサイズを変更するには、データベースを再作成します。デフォルトでは、IQ ページ・サイズによって I/O 転送ブロック・サイズが決まります。たとえば、デフォルト IQ ページ・サイズが 128KB の場合、デフォルトのブロック・サイズは 8,192 バイトになります。通常、Sybase IQ はこのページ・サイズに対するデフォルトのブロック・サイズの割合の使用だけでなく、他の要因も考慮します。

ほとんどのシステムでは、デフォルトのブロック・サイズを使用することによって、I/O 転送率とディスク領域の使用率のバランスを最適化できます。ただし、パフォーマンスよりもディスク領域の節約が優先されます。デフォルトのブロック・サイズでシステムのパフォーマンスが不十分な場合は、4,096 ~ 32,768 の間の 2 の累乗の値にブロック・サイズを設定できます。ただし、ブロック数は 1 ページに 2 ～ 16 個となるよう設定してください。次のような場合には、ブロック・サイズを明示的に設定できます。

- ディスク・アレイを使用したロー・ディスク・インストールの場合、ブロックが大きいほどパフォーマンスは向上しますが、使用されるディスク領域が多くなります。
- ファイル・システム・インストールでは、オペレーティング・システムにネイティブ・ブロック・サイズがある場合は、そのブロック・サイズ以上の IQ ブロック・サイズを設定すると、ディスク領域のパフォーマンスが最適化されます。IQ ブロック・サイズがファイル・システムのブロック・サイズと一致する場合は、I/O 率が向上する可能性もあります。

表 5-4 は、各 IQ ページ・サイズのデフォルトのブロック・サイズを示します。

表 5-4: デフォルトのブロック・サイズ

IQ ページ・サイズ (KB)	デフォルトのブロック・サイズ (バイト)
64	4096
128 (新しいデータベースのデフォルト)	8192
256	16384
512	32768

データ圧縮

Sybase IQ では、常にデータを圧縮してディスクに保存します。データ圧縮によって、ディスク領域の必要量が減少し、パフォーマンスも向上します。データ圧縮の量は、IQ ページ・サイズに基づいて自動的に決定されます。

メモリの節約

マシンのメモリが十分でない場合は、次のように調整してメモリを節約します。

バッファ・キャッシュ設定の縮小

バッファ・キャッシュ・サイズを小さくすることによってと、メモリを節約できます。ただし、バッファ・キャッシュを小さくしすぎると、バッファの不足によって、データのロードまたはデータの問い合わせが非効率的になったり、完了できなくなったりすることがあります。

ロードに使用するメモリの削減

LOAD_MEMORY_MB オプションを設定すると、ロードや他の同様の操作に使用するヒープ・メモリ量を制限できます。詳細については、「ロード、挿入、更新、同期、削除のためのメモリ」(106 ページ) を参照してください。

ロードのためのブロッキング係数の調整

BLOCK FACTOR を使用して、フラット・ファイルからロードする場合の I/O を減らします。LOAD コマンドの BLOCK FACTOR オプションにより、入力ファイルの作成時に使用したブロッキング係数 (ブロックあたりのレコード数) を指定します。デフォルトの BLOCK FACTOR は 10,000 です。

このロード・オプションの構文は、次のとおりです。

BLOCK FACTOR = *integer*

BLOCK FACTOR を決定するときは、次のガイドラインに従ってください。

```
record size * BLOCK FACTOR = memory required
```

このオプションを使用するには、バッファのメモリ以外に追加メモリが必要となります。使用可能なメモリが大量にある場合、または同時にアクティブになっているユーザが 1 人のみの場合は、BLOCK FACTOR の値を大きくすることによって、ロード・パフォーマンスを向上できます。

ユーザが多数存在する場合の最適化

Sybase IQ で処理できるユーザ接続の最大数は、32 ビット・プラットフォーム (Linux と Windows 2000/2003/XP) では 200、64 ビット・プラットフォーム (Sun Solaris、HP-UX と Itanium、AIX) では 1,000 です。64 ビット・システムでこのような多数のユーザをサポートするには、オペレーティング・システムのパラメータと start_asiq サーバ・パラメータの両方を調整する必要があります。推奨事項については、これ以降の各項と『Sybase IQ インストールおよび設定ガイド』を参照してください。

ユーザが多数存在する場合の Sybase IQ コマンド・ライン・オプションの変更

次の start_asiq のスイッチは、ユーザ数が多い場合の処理に影響を与えます。

-gm *#_connections_to_support*

-iqgovern *#_ ACTIVE_ queries_to_support*

-gn *#_Catalog_Store_front_end_threads*

-c *Catalog_Store_cache_size*

-ch *size*

-cl *size*

-gm

これは、サーバで処理できる接続の合計数です。ここで設定する接続の合計数のすべてがアクティブにデータベースを使用するのではなく、一部の接続はアイドル状態となっていることが想定されます。

-iqGovern

Sybase IQ には、1,000 人のユーザが接続できます。ただし、一度にクエリを許可されるユーザが少ないほど、より良いスループットを得ることができ、各ユーザが効率的に操作できるリソースを十分に確保できます。-iqgovern 値を指定すると、一度に実行されるクエリの最大数が制限されます。-iqgovern の制限を超えるユーザがクエリを発行した場合は、アクティブなクエリのいずれかが完了するまで、新しいクエリはキューイングされます。

-iqgovern の最適な値は、クエリの性質、CPU の数、Sybase IQ バッファ・キャッシュのサイズによって異なります。デフォルト値は 2*CPU の数 + 10 です。接続ユーザ数が多い場合は、このオプションを 2*CPU の数 + 4 に設定するとスループットが向上する場合があります。

-gn

-gn の適正値は、-gm の値によって決まります。start_asiq ユーティリティが -gn を計算し、値を適切に設定します。-gn の設定値が小さすぎると、サーバが正しく機能しなくなることがあります。-gn は、480 以下にすることをおすすめします。

-c

カタログ・ストア・バッファ・キャッシュは、カタログ・ストアの汎用メモリ・プールでもあります。MB 単位で指定するには、-c nM 形式を使用します。たとえば、-c 64M のように指定します。Sybase の推奨値は次のとおりです。

表 5-5: カタログ・バッファ・キャッシュの設定

ユーザ数	プラットフォーム	-c で設定する最小値
1000 まで	64 ビットのみ	64MB
200 まで	64 ビット	48MB (64 ビットの場合の start_asiq のデフォルト値)。ユーザ数がこれより多い場合は 64MB に設定すると有効
200 まで	32 ビット	32MB (32 ビットの場合の start_asiq のデフォルト値)

大量の解析を必要とする特定のクエリに対応する場合など、標準のカタログ・キャッシュ・サイズでは小さすぎる場合があります。このような場合は、-cl と -ch を設定すると有効なことがあります。たとえば、32 ビット・プラットフォームでは次のように設定してみます。

```
-cl 128M
-ch 256M
```

-ch または -cl と同じ設定ファイルやコマンド・ラインで -c を使用しないでください。関連情報については、-ch cache-size オプションを参照してください。

警告！　カタログ・ストアのキャッシュ・サイズを明示的に制御するには、サーバ起動用の設定ファイル (*.cfg*) または UNIX コマンド・ラインで、次のいずれか一方を実行します。両方を実行しないでください。

- -c パラメータを設定する
- -ch および -cl パラメータを使用して、カタログ・ストアのキャッシュサイズに特定の上限と下限を設定する

上記のパラメータをこれ以外の組み合わせで指定すると、予期しない結果が生じることがあります。

-iqmt

-iqmt オプションの設定は必須ではありません。指定された接続数に対して -iqmt の設定値が小さすぎる場合は、要求された接続数を処理するためにスレッド数が増加します。つまり、-gm が -iqmt を上書きします。ただし、-iqmt オプションによって Sybase IQ スレッド数が増加した場合は、後述の「ユーザが多数存在する場合のオペレーティング・システム・パラメータの設定」で説明する制限の設定時に、このスレッド数を使用します。

ユーザが多数存在する場合の Sybase IQ テンポラリ領域の増加

場合によっては、より多くのユーザに対応できるようにテンポラリ dbspace を増やす必要がある場合もあります。

新規接続と既存接続との優先順位

Sybase IQ が、接続されているユーザの対応でビジー状態の場合は、新規の接続要求への応答が遅くなることがあります。たとえば、サーバが挿入のためにビジー状態のときに、テスト・スクリプトなど、ループ内の数百個の接続が起動される極端な例の場合、新規の接続がアクティブになるには、しばらく待機することがあります。また、場合によっては、その接続要求がタイムアウトになることもあります。このような場合、サーバは単にビジー状態になっているだけにも関わらず、停止しているように見えます。このような状況が発生した場合は、再び接続を行ってください。

プラットフォーム固有のメモリ・オプション

プラットフォームの種類を問わず、Sybase IQ では次の 4 つの目的ためにメモリを使用します。

- メイン・バッファ・キャッシュ
- テンポラリ・バッファ・キャッシュ
- Sybase IQ メモリ・オーバヘッド (スレッド・スタックを含む)
- ロード・バッファ

図 5-1 (108 ページ) で示した Sybase IQ のメモリ使用の図を参照してください。

すべての 64 ビット・プラットフォームでは、使用可能な合計メモリ量は実質的に無制限です。システムの仮想メモリが唯一の制限となります。

HP-UX システムにおけるパフォーマンス・チューニングのヒントについては、そのプラットフォームの『Sybase IQ インストールおよび設定ガイド』を参照してください。

32 ビット・プラットフォームでは制限があります。詳細については、次の表を参照してください。

表 5-6: 32 ビット・プラットフォームで使用可能な合計メモリ量

プラットフォーム	使用可能メモリ量
RedHat Linux 2.1	約 1.7GB を Sybase IQ に使用可能
RedHat Linux 3.0	約 2.7GB を Sybase IQ に使用可能
Windows 2000/2003/XP[a]	2.75GB を Sybase IQ に使用可能

[a] このメモリ量を確保するには、Windows 2000 Advanced Server または Datacenter Server、Windows Server 2003 Standard、Enterprise または Datacenter Edition、もしくは Windows XP Professional を使用し、/3GB スイッチを設定する必要があります。このスイッチを設定しないと、2GB に制限されます。これはプロセスに使用できる合計メモリ量です。/3GB スイッチを設定した場合でも、Windows サーバではバッファ・キャッシュの合計サイズが 2GB を超えることはできません。詳細については、Windows 用の『Sybase IQ インストールおよび設定ガイド』を参照してください。

Sybase IQ サーバ内での仮想メモリの使用パターンのせいで、Windows プラットフォーム上で仮想メモリの断片化によって処理が過度に増大する可能性があります。このような状況に陥る可能性を小さくするため、Sybase IQ では Windows XP と Windows Server 2003 について Microsoft の LFH (低断片化ヒープ) の使用をサポートしています。

Windows プラットフォームにおけるパフォーマンス・チューニングのこの他のヒントについては、「第 7 章 Windows システムでのサーバのチューニング」を参照してください。

UNIX システムの場合にかぎり、Sybase IQ にはメモリの管理に役立つ 2 つのコマンドライン・オプションが用意されています。

連結メモリ・プール

HP と Sun のプラットフォームでは、指定した量のメモリを「連結」メモリとして指定できます。連結メモリは、物理メモリにロックされた共有メモリです。カーネルはこのメモリを物理メモリからページ・アウトできません。

他のアプリケーションが同じマシン上で同時に実行されている場合は、連結メモリによって Sybase IQ のパフォーマンスが向上することがあります。ただし、連結メモリを Sybase IQ 専用に割り付けると、そのメモリはマシン上の他のアプリケーションから利用できなくなります。

これらの UNIX プラットフォームにのみ「連結」メモリのプールを作成するには、-iqwmem コマンドライン・スイッチを指定して、連結メモリの MB 数を指定します。Sun 以外のプラットフォームで -iqwmem を設定するには、root ユーザになる必要があります。64 ビット・プラットフォームでは、マシンの物理メモリのみが -iqwmem の上限となります。

たとえば、14GB のメモリを搭載するマシンで、10GB の連結メモリを確保するとします。そのためには、次のように指定します。

```
-iqwmem 10000
```

警告！ このスイッチは、連結メモリに指定する余裕がメモリにある場合にのみ使用します。メモリが十分にないときにこのスイッチを使用すると、パフォーマンスが著しく低下することがあります。

注意 このバージョンでは、次の点に注意してください。

- Sun Solaris では、-iqwmem を指定すると、常に連結メモリが有効になります。
- HP では、サーバをルートとして起動した場合に、-iqwmem を指定すると連結メモリが有効になります。ルート以外のユーザでサーバを起動した場合は、非連結メモリが有効になります。この動作は、将来のバージョンで変更される可能性があります。

他のアプリケーションとデータベースの影響

サーバに使用されるメモリは、すべてのアプリケーションとデータベースに使用されるメモリ・プール内のメモリです。複数のサーバまたは複数のデータベースを同時に同じマシン上で実行したり、他のアプリケーションを実行したりしている場合は、サーバが要求するメモリ量を減らす必要があります。

サーバ・ログによって、実際にどれだけのメモリ量があるかがわかります。

```
Created 1073741824 byte segement id 51205 Attached at
80000000

Created 184549376 byte segement id 6151 Attached at
C3576000
```

また、UNIX コマンド `ipcs -mb` を発行して、実際のセグメント数を表示することもできます。

HP のメモリ問題のトラブルシューティング

HP-UX でメモリ問題が発生している場合は、maxdsiz_64bit カーネル・パラメータの値を調べます。このパラメータは、64 ビット HP プロセッサ上で Sybase IQ が使用できる仮想メモリの量を制限します。『Sybase IQ インストールおよび設定ガイド』で推奨値を参照してください。

ファイル・システム・バッファリングの制御

Solaris UFS および Windows ファイル・システムの場合にかぎり、ファイル・システム・バッファリングのオンとオフを切り替えることができます。ファイル・システム・バッファリングをオフにすると、ファイル・システム・バッファ・キャッシュからのデータ・コピーがメイン IQ バッファ・キャッシュに保存されます。通常、データ・コピーの保存によってページングが減り、パフォーマンスが向上します。ただし、例外が 1 つあります。データベースの IQ ページ・サイズがファイル・システムのブロック・サイズよりも小さい場合 (通常は、テスト状況の場合のみ)、ファイル・システム・バッファリングをオフにすると、特にマルチユーザ操作中にパフォーマンスが低下することがあります。

新しく作成される Sybase IQ データベースでは、ファイル・システム・バッファリングがデフォルトでオフになります。

既存のデータベースのファイル・システム・バッファリングを無効にするには、次の文を発行します。

```
SET OPTION "PUBLIC".OS_FILE_CACHE_BUFFERING = OFF
```

このオプションは、PUBLIC グループにのみ設定できます。変更を有効にするには、データベースを停止し、再起動します。

注意 Solaris には、ファイル・システム・バッファ・キャッシュのサイズを制限するカーネル・パラメータはありません。時間の経過とともにファイル・システム・バッファ・キャッシュが大きくなり、IQ バッファ・キャッシュ・ページを置き換えるため、オペレーティング・システムに過度のページング・アクティビティが発生し、Sybase IQ パフォーマンスが低下します。

Windows では、ファイル・システムを犠牲にしてアプリケーションを優先させるページング・アルゴリズムを使用します。これは、Sybase IQ パフォーマンスの向上に役立ちます。詳細については、「第 7 章 Windows システムでのサーバのチューニング」を参照してください。

メモリを増やすその他の方法

特定の環境においては、他のオプションを調整して Sybase IQ に使用可能なメモリを増やすことができます。

Java 実行可能のデータベースのオプション

SET OPTION コマンドの JAVA_HEAP_SIZE オプションは、Java アプリケーションに対して接続ごとに割り付けるメモリの最大サイズ (バイト単位) を設定します。通常、接続ごとに割り当てられるメモリはユーザの作業領域として使用され、その内訳は Java 変数と Java アプリケーション・スタック領域です。Java アプリケーションの実行中、接続ごとの割り付けはデータベース・サーバの固定キャッシュを使用するため、制御できなくなった Java アプリケーションがメモリを使いすぎないようにすることが重要です。

SET OPTION コマンドの JAVA_NAMESPACE_SIZE オプションは、Java アプリケーションに対してデータベースごとに割り付けるメモリの最大サイズ (バイト単位) を設定します。データベースごとのメモリ割り付けは、Java クラス定義を含みます。クラス定義は読み取り専用であるため、接続間で共有されます。したがって、その割り付けには固定キャッシュを使用することになり、このオプションを使って割り付けサイズの制限を設定します。

プロセス・スレッド・モデル

Sybase IQ では、最大限のパフォーマンスを得るために、オペレーティング・システムのカーネル・スレッドを使用します。スレッドは、ユーザ・レベルとカーネル・レベルにあります。ライトウェイト・プロセスは、カーネルでサポートされるコントロールの基本となるスレッドです。オペレーティング・システムによって、どのライトウェイト・プロセス (LWP) をどのプロセッサでいつ実行するかが決定されます。オペレーティング・システムはユーザ・スレッドのことは関知しませんが、ユーザ・スレッドが待機中か実行可能かは認識しています。

オペレーティング・システムのカーネルによって、LWP が CPU リソース上にスケジューリングされます。この場合、LWP のスケジューリング・クラスと優先度を使用します。マルチプロセッサ・システム上で、カーネルによって個別にディスパッチされた LWP は、個別のシステム・コールを実行したり、個別のページ・フォールトを発生させたり、並列実行されます 。

高度にスレッド化された単一のプロセスが、すべての Sybase IQ ユーザの処理を実行します。Sybase IQ は、接続によって実行される処理の種類、使用可能な合計スレッド数、さまざまなオプションの設定に基づいて、各ユーザ接続にさまざまな数のカーネル・スレッドを割り当てます。

スレッド不足エラー

発行したクエリの開始に必要なサーバ・スレッドが不足している場合は、次のようなエラーが発生します。

```
Not enough server threads available for this query
```

この状況は、すぐに解消される場合もあります。他のクエリが完了してからクエリを発行すると、使用可能なスレッドが増えるため、クエリが成功する場合があります。この状況が続く場合は、次の項で説明するように、サーバを再起動し、Sybase IQ スレッド数にさらに大きな値を指定する必要があります。

スレッド使用を管理するための Sybase IQ オプション

Sybase IQ には、スレッド使用の管理に役立つ次のオプションがあります。

- Sybase IQ に使用可能な最大スレッド数を設定するには、サーバ起動オプション -iqmt を設定します。デフォルト値は接続数と CPU 数によって計算され、通常、デフォルト値をそのまま使用できます。
- サーバ内の内部実行スレッドのスタック・サイズを設定するには、サーバ起動オプション -iqtss を設定します。通常はデフォルト値で十分ですが、複雑なクエリを実行したときに、スタックの深さがこの制限を超えていることを示すエラーが返された場合は、値を増やします。-iqmt および -iqtss の詳細については、『Sybase IQ ユーティリティ・ガイド』の「第 1 章 データベース・サーバの実行」を参照してください。
- ユーザ 1 人あたりに使用するスレッド数の最大値を設定するには、SET OPTION MAX_IQ_THREADS_PER_CONNECTION コマンドを発行します。特定の動作に使用するリソースの量を制御する場合にも、このコマンドを使用します。たとえば、DBA は INSERT または LOAD コマンドを使用する前にこのオプションを設定できます。

I/O の分散

この項では、システムの I/O の分散がなぜ重要かについて説明します。パフォーマンスを向上させる 2 つの方法として、ディスク・ストライピングを使用する方法とファイルを別のディスクに格納する方法について説明します。また、メッセージ・ログ・ファイルのサイズを制御する方法についても説明します。

ロー I/O (UNIX オペレーティング・システム)

ほとんどの UNIX ファイル・システムでは、ディスクは固定サイズのパーティションに分割されます。パーティションは、オペレーティング・システムが個別にアクセスする物理的なディスク領域です。ディスク・パーティションは、通常、ファイル・システム・モード UFS ファイル・システムまたはロー・モードのつのモードでアクセスされます。ロー・モード (文字モードとも呼ばれる) ではバッファを使用しない I/O を行い、通常、読み取りまたは書き込みシステム・コールごとにデバイスに対するデータ転送を行います。UNIX ファイル・システム である UFS モードは、バッファを使用する I/O であり、バッファにデータを蓄積してからバッファ全体を一度に転送します。

作成したデータベースまたは dbspace は、ロー・デバイスまたはファイル・システム・ファイルに格納できます。Sybase IQ は指定されたパス名から、それがロー・パーティションかファイル・システム・ファイルかを自動的に判断します。ロー・パーティションは任意のサイズに設定できます。

詳細については、『Sybase IQ システム管理ガイド』の「第 5 章 データベース・オブジェクトの使用」の「データベース・オブジェクトの使用」の項を参照してください。

ディスク・ストライピングの使用

従来のファイル管理システムでは、特定のディスクにファイルを格納できます。したがって、すべてのファイルの操作は、1 台のディスク・ドライブに対して行われます。複数のディスク・ドライブにわたって、論理デバイスまたはボリュームを作成できるオペレーティング・システムもあります。最初のディスク・ドライブでファイルが満杯になると、論理ボリュームの次のドライブに自動的に継続されます。この機能によって最大ファイル・サイズが大きくなり、ディスクが一杯になるまで 1 台のディスクに対してアクティビティが実行されます。

ディスクにファイルを格納するまったく別の方法もあります。ディスク・ストライピングは、複数のディスク・ドライブに 1 つのファイルのデータを分散する場合に使用する一般的な方法です。ディスク・ストライピングを使用すると、連続するディスク・ブロックをストライプ・ディスク・ドライブに格納できます。ストライピングによって、1 つの論理ディスクに 1 つ以上の物理ディスク (またはディスク・パーティション) が結合されます。ストライプ・ディスクでは、I/O の転送が、コンポーネントの複数の物理デバイスに分散され、並列実行されます。このため、1 台のディスクを使用するよりも、パフォーマンスが大幅に向上します。

ディスク・ストライピングでは、異なるディスクにブロックを格納します。最初のブロックは、最初のドライブに格納されます。2 番目のブロックは、2 番目のドライブに格納されます。すべてのドライブを使用すると、最初に戻り、追加のブロックを各ドライブに格納していきます。ディスク・ストライピングの最大の特長は、複数のディスク・ドライブに対してランダムにデータを分散できる点です。ストライプ・ディスクに格納されたファイルにランダムな操作を行うため、ストライプ・セットのすべてのドライブが均等にビジーになり、1 秒あたりのディスク操作の数が最大となります。これは、データベース環境において非常に有効な方法です。

オペレーティング・システムやハードウェアが提供するディスク・ストライピング、または Sybase IQ に内蔵されたディスク・ストライピングを使用できます。

UNIX におけるディスク・ストライピングの設定

ディスク・ストライピングに対応している UNIX システムには、物理ディスクをストライプ・デバイスに設定するユーティリティがあります。詳細については、UNIX システムのマニュアルを参照してください。

Windows におけるディスク・ストライピングの設定

Windows システムでは、適切な SCSI-2 ディスク・コントローラの機能によって、ハードウェア・ディスク・ストライピングを使用します。ハードウェア・ストライピングをサポートしていないマシンでも、データベースに複数のディスクを使用できる場合は、Windows ストライピングを使用して、ディスク I/O を複数のディスクに分散できます。[ディスクの管理] を使用して Windows ストライピングを設定してください。

推奨されるディスク・ストライピング

次に、ディスク・ストライピングの基本的な規則ついて説明します。

- パフォーマンスを最大にするには、ストライプ・ファイル・システムのディスクを複数のディスク・コントローラに分散させる必要があります。ただし、1 台のディスク・コントローラに多くのディスクを割り当てないように注意してください。通常の SCSI マシンでは、1 台のコントローラに 2 ～ 3 台のディスクを割り当てることができます。詳細については、ご使用のハードウェアのマニュアルを参照してください。
- ディスクは、テープ・ドライブまたは CD-ROM などの低速デバイスとは異なるコントローラに配置してください。同じコントローラにディスクを配置すると、ディスク・コントローラの速度が低下します。
- ストライプのサーバ CPU 1 台あたりに 4 台のディスクを割り付けます。
- 個々のディスクは、同等のデバイスである必要があります。つまり、サイズとフォーマットが同じで、できるだけ同一ブランドのデバイスを使用する必要があります。仕様が異なると、通常は最小のディスクのサイズが使用されるため、他のディスク領域が無駄になる場合があります。また、最も低速のディスクの速度が使用されることがあります。
- 通常、ファイル・ストライピングに使用するディスクを、スワップ・パーティションとして使用するなど、他の目的には使用しないでください。
- ルート・ファイル・システムを含むディスクは、決してストライプ・デバイスの一部として使用しないでください。

通常は、可能なかぎりディスク・ストライピングを使用してください。

注意 データをロードする際に最良の結果を得るには、ストライプ・ディスクにあるフラット・ファイルにデータをダンプしてから、LOAD TABLE コマンドを使用して、Sybase IQ にデータを読み込みます。

内部ストライピング

Sybase IQ では、一連の dbspace (ファイルまたはデバイスのロー・パーティション) にブロック単位で情報を格納します。ディスク・ストライピングを使用している場合、Sybase IQ は、使用可能な領域があるすべての dbspace にデータを分散させます。この方法では複数のディスク・スピンドルを使用して、高速な並列ディスク書き込みを可能にします。

ディスク・ストライピング・オプション

この項では、Sybase IQ で提供されているオプションを使用して、サードパーティ製のソフトウェアを使用しないでディスク・ストライピングを行う方法について説明します。サードパーティ製のソフトウェアとハードウェアによるディスク・ストライピングを使用している場合は、次の説明に従う必要はありません。

ディスク・ストライピングの ON/OFF

次に、ディスク・ストライピングの ON/OFF に使用する構文を示します。

```
SET OPTION "PUBLIC".DISK_STRIPING = { ON | OFF }
```

DISK_STRIPING オプションのデフォルト値は、すべてのプラットフォームで ON となります。ディスク・ストライピングが ON で、dbspace に使用可能な領域がある場合は、すべての dbspace に入力データが分散されます。ディスク・ストライピングが OFF の場合は、論理ファイルの先頭から dbspace (ディスク・セグメント) に格納され、一度に 1 つのディスク・セグメントが格納されます。

DISK_STRIPING オプションの値の変更を有効にするには、Sybase IQ サーバを再起動する必要があります。

データベース・オプション MAIN_DISK_KB_PER_STRIPE および TEMP_DISK_KB_PER_STRIPE は、ディスク・ストライピング・アルゴリズムが IQ メイン・ストアと IQ テンポラリ・ストアの次のストライプに移動する前に、各 dbspace に書き込まれるキロバイト (KB) 数を定義します。これらのオプションのデフォルト値は 1 で、これは 1 ページに切り上げられます。

dbspace を削除するには、ディスク・ストラピングがオンのときに DROP DBSPACE コマンドを使用します。ただし、dbspace を削除する前に sp_iqrelocate ストアド・プロシージャを使用して、dbspace 内のすべてのデータを再配置します。ディスク・ストラピングでは、データが複数の dbspace に分散されるため、多数のテーブルとインデックスの再配置が必要になることがあります。sp_iqdbspaceinfo および sp_iqdbspace ストアド・プロシージャを使用して、dbspace 上に存在するテーブルとインデックスを確認します。

複数の dbspace の使用

複数の dbspace を使用することによって、オペレーティング・システムの複数のファイルまたはパーティションに Sybase IQ とテンポラリ・データを分割できます。これらのファイルは、複数のディスクに分散できます。

ディスク・ストライピングと同様に、連続するデータベース・ファイルを複数のドライブに分散することによってランダムに格納できます。CREATE DBSPACE コマンドを使用して、Sybase IQ とテンポラリ・データにセグメントを追加できます。

dbspace の作成時期

可能なかぎり、データベース作成時にすべての dbspace を割り付けます。

dbspace を後で追加する場合、Sybase IQ は古い dbspace と新しい dbspace の両方に新しいデータをストライプします。更新のタイプによって、ストライピングのバランスがとれる場合や、バランスが崩れたままになる場合があります。ストライピングのバランスが再びとれるかどうかは、バージョン管理で「入れ替わる」ページ数によって決まります。

戦略的なファイルの格納

ランダム・アクセス・ファイル専用のディスク・ドライブ数、およびこれらのファイルに対して実行される 1 秒あたりの操作数を増やすことによって、ランダム・アクセス・ファイルに関連するパフォーマンスを向上させることができます。ランダム・ファイルには、IQ ストア、テンポラリ・ストア、カタログ・ストア、プログラム (Sybase IQ 実行ファイル、ユーザおよびストアド・プロシージャ、アプリケーション)、オペレーティング・システム・ファイルのランダム・ファイルがあります。

一方、順次アクセス・ファイルに関連するパフォーマンスは、専用ディスク・ドライブに格納し、他のプロセスとの競合をなくすことによって向上させることができます。順次ファイルには、トランザクション・ログやメッセージ・ログ・ファイルがあります。

ディスク・ボトルネックを防止するために、次の注意に従ってください。

- ランダム・ディスク I/O を順次ディスク I/O から分離する。
- Adaptive Server Enterprise などの他のデータベースのプロキシ・テーブルの I/O から Sybase IQ データベース I/O を分離する。
- IQ ストア、カタログ・ストア、テンポラリ・ストア、Adaptive Server Enterprise などのプロキシ・データベースから、トランザクション・ログとメッセージ・ログを分離する。
- データベース・ファイル、テンポラリ dbspace、トランザクション・ログ・ファイルをデータベース・サーバと同じ物理マシン上に配置する。

トランザクション・ログ・ファイル

トランザクション・ログ・ファイルには、Sybase IQ がシステム障害から復旧するための情報が記録されています。このファイルのデフォルトのファイル拡張子は *.log* です。

トランザクション・ログ・ファイルを移動したり、名前を変更したりするには、トランザクション・ログ・ユーティリティ (dblog) を使用します。構文と詳細については、『Sybase IQ ユーティリティ・ガイド』の「第 3 章 データベース管理ユーティリティ」を参照してください。

警告！ Sybase IQ のトランザクション・ログ・ファイルは、多くのリレーショナル・データベースのトランザクション・ログ・ファイルとは異なります。なんらかの理由で (ログ・ファイルではなく) データベース・ファイルが失われた場合は、データベースが失われます。ただし、バックアップを正しく実行している場合は、データベースを再ロードできます。

トランザクション・ログのトランケーション

Sybase IQ は、システム障害からリカバリするために必要な情報をトランザクション・ログに記録します。コミットされるトランザクションごとにログに記録される情報は少量ですが、トランザクション・ログのサイズは増え続けます。データを変更するトランザクション数が多いシステムでは、時間の経過とともにログが非常に大きくなる場合があります。

ログをトランケートするには、関係する Sybase IQ サーバをオフラインにする必要があります。ログをトランケートする頻度は、実際のところ Sybase IQ システムのサポートを担当している DBA 次第であり、ログ・ファイルの増大の度合いとサイトの運用手順によって異なります。最低でも月 1 回、ログ・ファイルが 100MB を超えている場合はそれより多い頻度でログのトランケーション・プロシージャをスケジュールしてください。

表 5-7 に、Sybase IQ でトランザクション・ログをトランケートする方法を示します。

表 5-7: トランザクション・ログのトランケーション

データベースの種類	使用する方法	詳細の参照先
非マルチプレックス	–m スイッチ。これにより、すべてのデータベースで各チェックポイント後にトランザクション・ログがトランケートされる。	「非マルチプレックス・データベースのトランザクション・ログをトランケートするには」
マルチプレックス	DELETE_OLD_LOGS データベース・オプション	「マルチプレックス・データベースのトランザクション・ログをトランケートするには」
稼働中	dbbackup コマンド・ライン・ユーティリティ	『Sybase IQ ユーティリティ・ガイド』のバックアップ・ユーティリティ (dbbackup)

適切な方法を使用してください。Sybase IQ データベースの複写は、本質的にトランザクション・ログの情報に依存します。このため、マルチプレックス・データベースでは DELETE_OLD_LOGS オプションだけを使用します (「マルチプレックス・データベースのトランザクション・ログのトランケーション」を参照してください)。また、トランザクション・ログには、Sybase の製品の保守契約を結んでいるサポート・センタが問題を診断し、再現するための有用な情報が記録されています。どちらの方法を使用する場合も、後でサポート・センタが診断するときにログが必要になった場合に備えて、既存のログのアーカイブ処理 (ログのコピーの保管) を指定してください。

非マルチプレックス・データベースのトランザクション・ログのトランケーション

–m サーバ起動スイッチを使用して、非マルチプレックス・データベースのトランザクション・ログをトランケートします。–m サーバ起動スイッチを永続的に設定したままにすることはおすすめしません。このスイッチは、トランザクション・ログのトランケーションのために Sybase IQ を起動するときだけ使用してください。これをどのように行うかは DBA 次第ですが、次に示す手順を参考にしてください。

❖ 非マルチプレックス・データベースのトランザクション・ログをトランケートするには

1. サーバ・スイッチ *.cfg* ファイルのコピーを作成し、ログのトランケーション設定用のファイルであることを示す名前を付けます。このファイルを編集し、–m スイッチを追加します。
2. *.db* ファイルおよび *.log* ファイルのコピー作成を含めて、完全なバックアップ手順を実行します。
3. Sybase IQ を停止します。*iq.msg* ファイルに ‘CloseDatabase’ が書き込まれたことを確認します。
4. –m オプションが含まれる設定ファイルを使って Sybase IQ を再起動します。この時点では、ユーザ・アクセスやトランザクションを許可しないでください。
5. Sybase IQ を停止し、–m オプションが設定されていない設定ファイルを使って再起動します。

マルチプレックス・データベースのトランザクション・ログのトランケーション

❖ マルチプレックス・データベースのトランザクション・ログをトランケートするには

1. 書き込みサーバのデータベースをバックアップしていない場合は、バックアップします。
2. 書き込みサーバで、次のように DELETE_OLD_LOGS オプションを設定します。

   ```
   SET OPTION Public.Delete_Old_Logs='On'
   ```

3. 書き込みサーバの dbremote を停止し、-x コマンド・ライン・スイッチを指定して再起動します (そのために、書き込みサーバのデータベース・ディレクトリにある *start_dbremote.bat* スクリプトの特別バージョンを作成します)。これにより、書き込みサーバでログがトランケートされます。次に例を示します。

   ```
   cd ¥Server01¥mpxdb¥cmd /c
   start dbremote -q -v -x -o
   "d:¥Server01¥mpxdb¥dbremote.log" -c
   "uid=DBA;pwd=SQL;eng=Server01;dbf=
   ```

```
d:¥Server01¥mpxdb¥mpxdb;
links=tcpip{port=1704;host=FIONA-PC}"
```

4 書き込みサーバで、次のように DELETE_OLD_LOGS オプションをオフにします。

```
SET OPTION Public.Delete_Old_Logs='Off'
```

注意 クエリ・サーバのトランザクション・ログは、書き込みサーバのログが最後にトランケートされた時期にかかわらず、同期中に常にトランケートされます。

メッセージ・ログ

データベースごとにメッセージ・ログ・ファイルが作成されます。このファイルのデフォルトのファイル名は、*dbname.iqmsg* です。ただし、データベースを作成するときに別の名前を付けることができます。メッセージ・ログ・ファイルは、最初のユーザがデータベースに接続すると作成されます。

デフォルトでは、Sybase IQ はエラー、状態、挿入通知メッセージを含むすべてのメッセージをメッセージ・ログ・ファイルに記録します。LOAD および INSERT 文の通知メッセージを OFF に設定できます。

サイトによっては、挿入の数、LOAD オプションと NOTIFY_MODULUS データベース・オプションの設定、その他の条件が原因で、メッセージ・ログ・ファイルが急速に増大する傾向があります。Sybase IQ では、メッセージ・ログを循環させて、このファイルのサイズを制限できます。

メッセージ・ログの循環を有効にした場合、ファイルが IQMSG_LENGTH_MB データベース・オプションで指定した最大サイズに達すると、新しいメッセージはファイルの先頭から書き込まれます。既存のメッセージは、行単位で上書きされます。

循環を有効にしたときは、新しいメッセージが挿入された場所を <next msg insertion place> タグで知ることができます。ログの循環が有効になっており、ファイル内の最後のメッセージが最新のメッセージとはかぎらないことに注意を促すために、ファイルの先頭と末尾に追加のタグが表示されます。

メッセージ・ログの循環を有効にしてログ・ファイルの最大サイズを設定する方法については、『Sybase IQ リファレンス・マニュアル』の「IQMSG_LENGTH_MB オプション」を参照してください。

挿入、削除、同期のための作業領域

データの挿入や削除、ジョイン・インデックスの同期を行う場合、Sybase IQ では、IQ ストアに作業領域が必要となります。作業領域を必要とするトランザクションがコミットされると、この領域は他の目的に再利用されます。

通常、IQ ストアに適切な割合の空き領域が維持されるかぎり、十分な空き領域を確保できます。ただし、データを削除する場合、データのサイズやデータベース・ページ間のデータの分散によって、大きな作業領域が必要となることがあります。多数のページにデータが分散しているデータベースの大部分を削除する場合は、データベースのサイズを一時的に 2 倍にできます。

予約領域のオプションの設定

MAIN_RESERVED_DBSPACE_MB と TEMP_RESERVED_DBSPACE_MB の 2 つのデータベース・オプションによって、Sybase IQ が特定の処理のために予約する領域の量を制御します。詳細については、『Sybase IQ システム管理ガイド』の「領域不足条件を処理するための領域の予約」を参照してください。

リソース使用を調整するオプション

Sybase IQ データベースの同時ユーザ数、実行するクエリ、使用可能な処理スレッドおよびメモリによって、パフォーマンス、メモリ使用、ディスク I/O に大きな影響を与える場合があります。Sybase IQ にあるリソースの使用を調整するオプションによって、さまざまなユーザ数やクエリに対応できます。次のオプションがあります。

- 現在のデータベースにのみ影響を与える SET OPTION コマンド・オプション
- データベース・サーバ全体に影響を与えるコマンドライン・オプション
- 現在の接続にのみ影響を与える接続パラメータ

パラメータや、オプションが有効になる時期、オプションを単一の接続と PUBLIC グループの両方に設定できるかどうかを含めて、これらのオプションの詳細については、『Sybase IQ リファレンス・マニュアル』を参照してください。

テーブルの最適化については、『Sybase IQ システム管理ガイド』の「記憶領域とクエリ・パフォーマンスの最適化」を参照してください。

同時クエリの制限

-iqgovern コマンドライン・オプションを使用すると、サーバの同時クエリの数を制御できます。これは、ライセンスによって規制される接続数とは異なります。

-iqgovern スイッチは、メモリを最も効果的に使用するために、バッファのデータのディスクへのページングを最適化します。-iqgovern のデフォルト値は、(2 x CPU 数) +4 です。

使用可能な CPU 数の設定

Sybase IQ 起動コマンドの -iqnumbercpus スイッチを使用すると、使用できる CPU の数を指定できます。このスイッチは、次のマシンでのみ使用することをおすすめします。

- Intel® の CPU を搭載し、ハイパースレッディングが有効になっているマシン
- オペレーティング・システムのユーティリティを使用して、Sybase IQ をマシン内の CPU の一部に制限しているマシン

詳細については、『Sybase IQ システム管理ガイド』の「CPU 数の設定」を参照してください。

クエリによるテンポラリ dbspace の使用の制限

SET コマンドの QUERY_TEMP_SPACE_LIMIT オプションを使用すると、1 つのクエリに使用可能なテンポラリ dbspace の量を制限できます。デフォルトでは、クエリは 2000MB のテンポラリ dbspace を使用できます。

クエリが発行されると、Sybase IQ は、クエリの解析に必要な一時領域を推定します。ソート、ハッシュ、ロー・ストアに使用する結果領域の合計が、現在の QUERY_TEMP_SPACE_LIMIT 設定を超える場合、クエリは拒否され、次のようなメッセージが表示されます。

```
Query rejected because it exceeds total space resource limit
```

このオプションを 0 に設定すると、制限がないため、一時領域の条件によってクエリが拒否されることはありません。

返されるローによるクエリの制限

SET コマンドの QUERY_ROWS_RETURNED_LIMIT オプションを設定すると、クエリ・オプティマイザは、大量のリソースを消費する可能性のあるクエリを拒否します。クエリからの結果セットがこのオプションの値を超えると推定される場合、クエリ・オプティマイザはクエリを拒否し、次のメッセージが表示されます。

```
Query rejected because it exceed resource:
Query_Rows_Returned_Limit
```

このオプションを使用する場合は、大量のリソースを消費するクエリのみを拒否するように設定します。

カーソルのスクロールの禁止

ホスト変数を宣言せずにカーソルのスクロールを使用すると、Sybase IQ は、クエリ結果をバッファする一時的なストア・ノードを作成します。これは、テンポラリ・ストア・バッファ・キャッシュとは異なります。百万単位のローなど、非常に多くローを検索する場合、このストア・ノードには多くのメモリが必要となります。

すべてのカーソルがスクロールしないように設定することによって、このテンポラリ・ストア・ノードを除去できます。そのためには、FORCE_NO_SCROLL_CURSORS オプションを ON に設定します。非常に多くのローを検索する場合、このオプションを使用して、一時的な記憶領域の必要量を節約できます。このオプションは、新しいクエリに対してすぐに有効になります。

アプリケーションでカーソルのスクロールを使用しない場合は、これを永続的な PUBLIC オプションに設定します。メモリの節約になるため、クエリのパフォーマンスが大幅に向上します。

カーソル数の制限

MAX_CURSOR_COUNT オプションは、接続が一度に使用できるカーソルの最大数を制限するリソース・ガバナーを指定します。デフォルトの値は 50 です。このオプションを 0 に設定すると、カーソル数は無制限になります。

文の数の制限

MAX_STATEMENT_COUNT オプションは、接続が一度に使用できる準備文の最大数を制限するリソース・ガバナーを指定します。

キャッシュ・ページのプリフェッチ

SET コマンドの PREFETCH_BUFFER_LIMIT オプションは、Sybase IQ がプリフェッチ (データベース・ページの先読み) に使用できるキャッシュ・ページの数を定義します。このオプションのデフォルト値は 0 です。このオプションは、Sybase 製品の保守契約を結んでいるサポート・センタから指示があった場合にだけ設定してください。詳細については、『Sybase IQ リファレンス・マニュアル』の「PREFETCH_BUFFER_LIMIT オプション」を参照してください。

SET コマンドの BT_PREFETCH_MAX_MISS オプションは、特定のクエリでページのプリフェッチを継続するかどうかを決定します。HG インデックスを使用するクエリの実行速度が予想より遅い場合は、このオプションの値を徐々に増やしてみます。詳細については、『Sybase IQ リファレンス・マニュアル』の「BT_PREFETCH_MAX_MISS オプション」を参照してください。

一般的な使用のための最適化

Sybase IQ は、開いたカーソルの数を追跡して、メモリを割り付けます。特定の状況においては、USER_RESOURCE_RESERVATION オプションによって、製品を使用している思われる現在のカーソル数の最小値を調整し、テンポラリ・キャッシュから割り付けるメモリをさらに節約できます。

このオプションは、慎重な分析の結果、実際に必要であると判断された場合にのみ設定する必要があります。このオプションを設定する場合は、Sybase 製品の保守契約を結んでいるサポート・センタに連絡してください。

プリフェッチされるローの数の制御

プリフェッチは、相対位置 1 または相対位置 0 のみをフェッチするカーソルのパフォーマンスを向上させるために使用します。2 つの接続パラメータを使用して、カーソル・プリフェッチのデフォルトを変更できます。PrefetchRows (PROWS) は、プリフェッチされるローの数を設定します。PrefetchBuffer (PBUF) は、プリフェッチされたローを格納するために、この接続に使用できるメモリを設定します。プリフェッチするローの数を増やすと、次の特定の条件ではパフォーマンスが向上する可能性があります。

- アプリケーションが数回の絶対フェッチで数多くのロー (数百ロー以上) をフェッチする場合
- アプリケーションがローを高速でフェッチし、クライアントとサーバが同じマシン上にあるか、高速ネットワークで接続されている
- クライアント／サーバ通信がダイヤルアップ・リンクやワイド・エリア・ネットワークなどの低速ネットワークで行われている場合

リソースを効率的に利用するための他の方法

この項では、パフォーマンスを向上させ、ディスク領域をさらに有効に活用するためのシステムの調整方法について説明します。

マルチプレックス・データベースのディスク領域の管理

ユーザがいずれかのサーバで、古いバージョンのテーブルを必要とするトランザクションを実行している間は、Sybase IQ はそのテーブルを削除できません。このため、マルチプレックス・データベースでテーブルの更新とクエリが同時に発生すると、Sybase IQ が大量のディスク領域を消費することがあります。消費される領域の量は、データとインデックスの性質および更新の頻度によって決まります。

クエリする必要がなくなった古いバージョンを書き込みサーバが削除できるようにすれば、ディスク・ブロックを解放できます。古いテーブル・バージョンをリカバリできるように、すべてのサーバのユーザ全員が現在のトランザクションを定期的にコミットする必要があります。これで、サーバは稼働し続けることができ、すべての機能を利用できます。各クエリ・サーバでのテーブル・バージョンの使用についての最新情報を書き込みサーバに転送するために、dbremote プロセスをすべて実行し続ける必要があります。

クエリ・サーバ間のロード・バランス

IQ ネットワーク・クライアントを使用して、マルチプレックス・クエリ・サーバ間のクエリ負荷のバランスをとれる場合があります。この方法では、プール内のマシンの作業負荷に応じて、各マシンにクライアント接続をディスパッチする中間システムが必要となります。

この方法を使用するには、クライアント・システムで、中間ロード・バランス・システムの IP アドレスとポート番号および汎用サーバ名を指定し、VerifyServerName 接続パラメータを NO に設定した特別な ODBC DSN を作成します。クライアントがこの DSN を使って接続すると、ロード・バランサは負荷が最も少ないと判断したマシンに対して接続を確立します。

クエリ・サーバのロード・バランスで使用する ODBC DSN の定義方法については、『Sybase IQ システム管理ガイド』の「第 4 章 接続パラメータと通信パラメータ」の「VerifyServerName 通信パラメータ (Verify)」を参照してください。

データベース・アクセスの制限

クエリのパフォーマンスを向上させるには、可能なかぎり、データベースを読み取り専用に設定するか、重要な更新を使用頻度の少ない時間帯にスケジュールします。Sybase IQ では、テーブルへの挿入や削除を実行している間に、複数のクエリ・ユーザがそのテーブルを読み込むことができます。ただし、データベースを同時更新している間は、パフォーマンスが低下します。

ディスクのキャッシュ

ディスク・キャッシュとは、ディスク・ブロックのコピーを一時的に格納するために、オペレーティング・システムによって使用されるメモリです。ファイル・システムに基づくディスクの読み書きは、通常、すべてディスク・キャッシュを通じて行われます。アプリケーションから見ると、ディスク・キャッシュによる読み書きは、すべて実際のディスク操作と同等です。

オペレーティング・システムは、固定された方法と動的方法を使用してディスク・キャッシュにメモリを割り付けます。固定された割り当てでは、あらかじめ規定されたメモリ量が使用されます。通常、10 ～ 15% のメモリが割り当てられます。オペレーティング・システムは通常、LRU (一番最後に使用された) アルゴリズムを使用してこの作業領域を管理します。動的割り付けでは、オペレーティング・システムが実行中にディスク・キャッシュの割り付けを決定します。これによって、できるだけ多くのメモリを有効に使用して、実際のメモリの需要とディスクのデータの必要性のバランスを保ちます。

インデックスのヒント

以下の項では、インデックスの選択と管理に関するヒントについて説明します。詳細については、『Sybase IQ システム管理ガイド』の「第 6 章 Sybase IQ インデックスの使用」を参照してください。

正しいインデックス・タイプの選択

カラム・データに適したインデックス・タイプを選択することが重要です。Sybase IQ は、いくつかのインデックスを自動的に設定します。具体的には、射影を最適化するデフォルト・インデックスをすべてのカラムに設定し、UNIQUE、PRIMARY KEYS、FOREIGN KEYS に HG インデックスを設定します。これらのインデックスはいくつかの目的には役立ちますが、特定のクエリをできるだけ迅速に処理するには別のインデックスが必要となります。カラムに設定できるインデックス・タイプが複数ある場合、Sybase IQ は最適なインデックス・タイプを選択します。

Sybase IQ のクエリ・オプティマイザには、クエリの 1 つまたは複数のカラムに追加のインデックスがあるとオプティマイザに有効な場合にメッセージを生成する**インデックス・アドバイザ**という機能があります。インデックス・アドバイザをアクティブにするには、INDEX_ADVISOR オプションを ON に設定します。メッセージはクエリ・プランの一部として出力されます。クエリ・プランが有効になっていない場合は、メッセージ・ログ (*.iqmsg*) に単独のメッセージとして出力されます。出力の形式は OWNER.TABLE.COLUMN となります。詳細については、『Sybase IQ リファレンス・マニュアル』の「データベース・オプション」の「INDEX_ADVISOR オプション」を参照してください。

ジョイン・クエリの WHERE 句で参照されるグループ化カラムの LF または HG に対しては、デフォルト・インデックス以外に LF または HG インデックスを作成する必要があります。WHERE 句で参照されるカラムに LF または HG インデックスが設定されていない場合は、Sybase IQ のクエリ・オプティマイザが最適な実行プランを生成できない可能性があります。HAVING 句で参照される非集合カラムでも、デフォルト・インデックス以外に LF または HG インデックスを設定する必要があります。次に例を示します。

```
SELECT c.name, SUM(l.price * (1 - l.discount))
FROM customer c, orders o, lineitem l
WHERE c.custkey = o.custkey
    AND o.orderkey = l.orderkey
    AND o.orderdate >= "1994-01-01"
    AND o.orderdate < "1995-01-01"
GROUP by c.name
HAVING c.name NOT LIKE "I%"
    AND SUM(l.price * (1 - l.discount)) > 0.50
ORDER BY 2 desc
```

この例に示す l.price と l.discount の近くにあるすべてのカラムには、デフォルト・インデックス以外に、LF または HG インデックスがあります。

ジョイン・インデックスの使用

ユーザは同時に複数のテーブルのデータを参照することがよくあります。このデータは、クエリ作成時にジョインするか、またはジョイン・インデックスを作成することによって事前にジョインできます。常に同じ方法でジョインされるカラムにジョイン・インデックスを作成すると、クエリのパフォーマンスが向上することがあります。

ジョイン・インデックスのロードには、かなりの時間と領域が必要なため、定期的に必要となるジョインにのみジョイン・インデックスを作成します。Sybase IQ のジョイン・インデックスでは、1 対多および 1 対 1 のジョイン関係がサポートされます。

削除のための十分なディスク領域の確保

データ・ローを削除する場合、Sybase IQ は、削除するデータを含むデータベース・ページごとにバージョン・ページを作成します。削除トランザクションが実行されるまで、バージョンは保持されます。このため、データを削除する場合、ディスク領域の追加が必要な場合があります。詳細については、「重複したバージョンと削除」を参照してください。

データベース・サイズと構造の管理

この項では、データベース設計を改善し、データを管理するための考え方について説明します。

データベース・サイズの管理

データベースのサイズは、作成するインデックスと格納するデータ量に大きく依存します。ユーザが発行するクエリに必要とされるインデックスをすべて作成することによって、クエリ処理を高速化できます。ただし、テーブルやインデックスが必要ない場合は、削除できます。削除することによって、ディスク領域を解放し、ロードとバックアップの速度を向上させ、バックアップに必要なアーカイブの記憶領域を小さくできます。

任意のテーブルに格納されたデータ量を制御するには、必要のないデータ・ローを最適な方法で削除してください。Adaptive Server Anywhere データベースのデータがデータベースに含まれる場合は、Anywhere データの削除を実行するだけで不要なデータを削除できます。コマンド構文は互換性があります。Sybase IQ は、Transact-SQL と互換性があるため、Adaptive Server Enterprise データベースのデータも同様に削除できます。

インデックスの断片化の制御

内部インデックスの断片化は、インデックス・ページが最大ボリュームまで使用されていないときに発生します。

ローの断片化は、ローが削除されたときに発生することがあります。ページのロー全体を削除した場合、そのページは解放されますが、ページの一部のローが未使用の場合は、未使用領域がディスクに残ります。

テーブルに対する DML 操作 (INSERT、UPDATE、DELETE) によって、インデックスの断片化が発生します。断片化をレポートする 2 つのストアド・プロシージャがあります。

- sp_iqrowdensity は、デフォルト・インデックス・レベルでのローの断片化をレポートします。「sp_iqrowdensity プロシージャ」を参照してください。
- sp_iqindexfragmentation は、補助インデックス内の内部断片化をレポートします。「sp_iqindexfragmentation プロシージャ」を参照してください。

データベース管理者は、カラムのデフォルト・インデックスを補助する別のインデックスを作成することがあります。テーブルからローが削除されると、これらのインデックスが必要以上の領域を使用する場合があります。

どちらのプロシージャも対応策は示しません。データベース管理者は、レポートされた情報を調べて、インデックスの再作成、再編成、再構築などの対応策をとるかどうかを判断する必要があります。

カタログ・ファイル増大の最小化

カタログ・ファイルが増大するのは正常なことで、その割合はアプリケーションとカタログの内容によって異なります。*.DB* ファイルのサイズがパフォーマンスに影響を与えることはなく、*.DB* ファイル内の空きページが必要に応じて再利用されます。カタログ・ファイルの増大を最小限に抑えるには、次の方法を使用します。

- CREATE TABLE 文で IN SYSTEM を使用しない。
- システム・ストアド・プロシージャを実行した後で COMMIT 文を発行する。
- 長時間実行されるトランザクションの最中に COMMIT 文を発行する。

パフォーマンス向上のための非正規化

正規化フォームでデータベースを作成した場合、ベンチマークを実行して、意図的に正規化を解除してパフォーマンスを向上させることができます。非正規化については、以下のことが言えます。

- テーブルまたはカラムに対して可能
- 事前に正規化されていることが前提
- データの使用方法の理解が必要

非正規化する理由を次に示します。

- すべてのクエリには、「フル」セットのジョイン・データへのアクセスが必要である。
- 導出カラムの計算は複雑なため、selects のための格納が必要である。

非正規化のリスク

非正規化を正しく行うには、アプリケーションに関する十分な知識が必要となるため、パフォーマンスに問題がある場合にのみ非正規化を実行してください。非正規化を行う場合、変更によってデータを最新の状態に保つためにどれだけの作業が必要かを考慮する必要があります。

これは、大量のデータの要約が頻繁に必要とされる意志決定支援アプリケーションと個別にデータ変更を行うトランザクション処理要求との違いを示す良い例です。非正規化を行う場合、特定の処理の効率を向上させるために、他の処理の効率が低下することがあります。

どのような非正規化の方式にも、データの整合性に問題がある可能性があるため、アプリケーションの設計時に慎重に文書化し、注意する必要があります。

非正規化の短所

正規化の解除には、次の短所があります。

- 非正規化を行うと、通常、検索は速くなりますが、更新は遅くなります。これは、DSS 環境ではさほど問題になりません。
- 非正規化は、必ずアプリケーションごとに行われるため、アプリケーションを変更した場合は、再評価が必要となります。
- 非正規化によって、テーブルのサイズが大きくなる場合があります。Sybase IQ では、カラム・データの格納を最適化できるため、このことは問題ではありません。詳細については、『Sybase IQ リファレンス・マニュアル』の「CREATE TABLE 文」の「IQ UNIQUE 制約」と「MINIMIZE_STORAGE オプション」を参照してください。
- 非正規化によって、コーディングが簡単になる場合と、逆に複雑になる場合があります。

非正規化のパフォーマンスの利点

次に、何によってパフォーマンスが向上するかを示します。

- ジョインの必要性の最小化
- 集約値の再計算 (選択時ではなくデータ変更時の計算)
- テーブル数の低減

非正規化の決定

非正規化を行うかどうかを決定する場合、使用中の環境におけるアプリケーションのデータ・アクセス要件と実際のパフォーマンス特性を分析する必要があります。非正規化を行う場合は、次の項目について検討します。

- 重要なクエリおよび予想される応答時間
- 使用するテーブルまたはカラム、および 1 アクセスあたりのロー数
- 通常のソート順
- 同時予測
- アクセス頻度が最も高いテーブルのサイズ
- 要約を計算するプロセスの有無
- パフォーマンス向上のためのジョイン・インデックス作成の有無

ロードを高速化するための UNION ALL ビューの使用

非常に大きいテーブルのロード時間を最小限にするには、UNION ALL ビューを使用します。Sybase IQ では、データを (たとえば日付ごとに) 複数の独立したベース・テーブルに分けて、テーブルを分割できます。データは、これらの小さいテーブルにロードします。そして、UNION ALL ビューを使ってテーブルを 1 つの論理的な統一体に結合し、この統一体に対してクエリを実行します。

UNION ALL ビューは、管理が容易です。たとえば、データを月ごとに分割している場合は、1 つのテーブルを削除し、UNION ALL ビューの定義を更新することによって 1 か月分のデータ全体を削除できます。日付の範囲述部を追加することなく、年、四半期などに対応する多くのビュー定義を作成できます。

UNION ALL ビューを作成するには、ベース・テーブルを別々の物理テーブルに分割する論理的手段を選択します。最も一般的なのは、月ごとに分割する方法です。

たとえば、第一四半期のすべての月を含むビューを作成するには、次のコマンドを入力します。

```
CREATE VIEW
SELECT * JANUARY
UNION ALL
SELECT * FEBRUARY
UNION ALL
SELECT * MARCH
UNION ALL
```

月ごとに、1 つのベース・テーブル (この例では JANUARY、FEBRUARY、または MARCH) にデータをロードできます。次の月は、同じカラムと同じインデックス・タイプで構成された新しいテーブルにデータをロードします。

構文の詳細については、『Sybase IQ リファレンス・マニュアル』の「UNION 演算」を参照してください。

注意 UNION ALL ビューに対して INSERT...SELECT を実行することはできません。

UNION ALL ビューを参照するクエリの最適化

最適化が効果を発揮するためには、*UNION ALL* ビューのすべてのパーティションにすべてのインデックスが定義されている必要があります。

DISTINCT を指定するクエリでは、UNION ALL ビューを使用すると、ベース・テーブルを使用するよりも実行速度が遅くなる傾向があります。

Sybase IQ には、次のような UNION ALL ビューの特許取得済みの最適化が用意されています。

- UNION ALL ビューでの分割 GROUP BY
- UNION ALL ビューへのプッシュダウン・ジョイン

UNION ALL ビューを参照するクエリのパフォーマンスを調整する必要がある場合は、Join_Preference データベース・オプションを設定してください。このオプションは、UNION ALL ビュー間のジョインに影響を与えます。これらのオプションの詳細については、『Sybase IQ リファレンス・マニュアル』の「第 2 章 データベース・オプション」を参照してください。

UNION を分割されたテーブルとして扱えるのは、以下の制約条件がすべて満たされている場合にかぎられます。

- 1 つまたは複数の UNION ALL が含まれる。
- UNION の各アームの FROM 句にテーブルが 1 つだけ含まれており、そのテーブルが物理ベース・テーブルである。
- UNION のどのアームにも、DISTINCT、RANK、集合関数、GROUP BY 句がない。
- UNION の各アームに含まれる SELECT 句の中の各項目がカラムである。
- 最初の UNION アームの SELECT リスト内のカラムのデータ型のシーケンスが、UNION の後続の各アームにおけるシーケンスと同じである。

『Sybase IQ リファレンス・マニュアル』の「SELECT 文」も参照してください。

ネットワーク・パフォーマンス

以降の項では、ネットワーク・パフォーマンスの問題を解決するための方法を示します。

大量のデータ転送の向上

大量のデータを同時に転送すると、スループット全体が低下して、平均応答時間が増加します。次に、このような場合にパフォーマンスを向上させるための方法を示します。

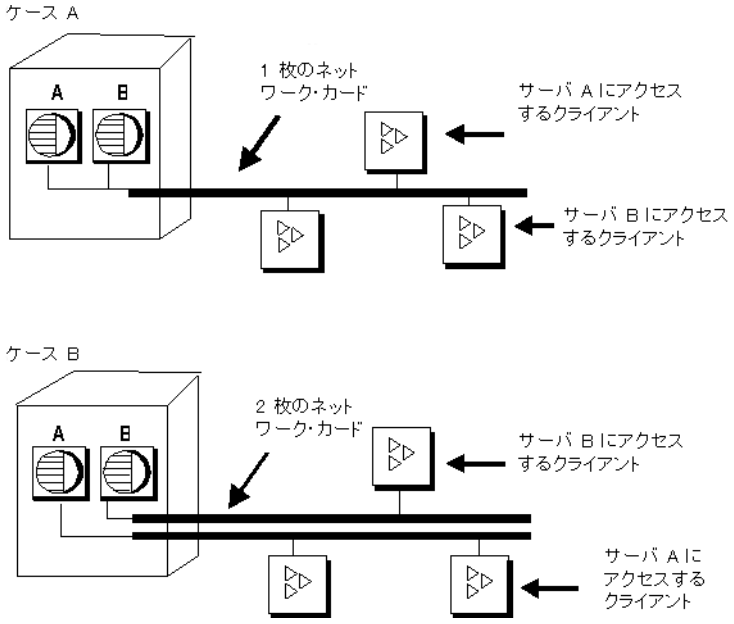
- 大量のデータ転送は、できるかぎり勤務時間外に行う。
- 大量のデータ転送中は同時クエリの数を制限する。
- 大量のデータ転送するときに、クエリと挿入を同時に実行しない。
- ストアド・プロシージャを使用して、トラフィックを低減する。
- ロー・バッファリングを使用して、ネットワーク上の大きなバッチを移動する。
- 大量のデータ転送を日常的に行う場合は、そのような転送に適したネットワーク・ハードウェアの設置を検討する。たとえば次のような方法がある。
    - トークン・リング － 大量のデータを転送する場合、イーサネットより応答が向上する。
    - 光ファイバ － 非常に高い帯域幅を提供するが、ネットワーク全体で使用するには高価すぎる。
    - 別のネットワーク － 最大ボリュームのワークステーションとサーバ間のネットワーク・トラフィックを処理するために使用する。

ヘビー・ネットワーク・ユーザの分離

図 5-2 のケース A では、2 つの異なるデータベース・サーバにアクセスするクライアントが 1 枚のネットワーク・カードを使用しています。このため、サーバ A とサーバ B にアクセスするクライアントは、ネットワーク上とネットワーク・カードで競合します。ケース B では、サーバ A にアクセスするクライアントとサーバ B にアクセスするクライアントが別々のネットワーク・カードを使用しています。

異なるマシンをデータベース・サーバにすると、さらにパフォーマンスが向上します。異なるデータベースのヘビー・ユーザを異なるマシンに分けることもできます。

図 5-2: ヘビー・ネットワーク・ユーザの分離



少量のデータを小さなパケットに入れる

ネットワーク上で少量のデータを送信する場合は、デフォルトのネットワーク・パケット・サイズを小さいまま使用します (デフォルトは 512 バイトです)。-p サーバ起動オプションは、最大パケット・サイズを指定するために使用します。クライアント・アプリケーションを使用してパケット・サイズを設定できます。

大量のデータを大きなパケットに入れる

大量のデータを送受信するアプリケーションが多い場合は、デフォルトのネットワーク・パケット・サイズを大きくします。転送の数は少なくなりますが、データ転送量は多くなります。

サーバ・レベルのプロセス

サーバ・レベルで、できるかぎり多くのデータをフィルタします。

第 6 章

パフォーマンスのモニタリングとチューニング

この章について

この章では、Sybase IQ のパフォーマンスのモニタリングに使用するツールについて説明します。これらのツールを使用して、使用可能なリソースをシステムが最大限に利用しているかどうかを確認します。Sybase IQ でのメモリ、処理スレッド、ディスクの使用方法について、およびリソースの使用を制御するための設定オプションについては、「第 5 章 システム・リソースの管理」を参照してください。また、ここに記載した以外のチューニング上のヒントについては、このマニュアルの他の章にある、パフォーマンスの影響とチューニングについての項を参照してください。

内容

Sybase IQ 環境の表示

Sybase IQ のパフォーマンスをチューニングする最初の手順は、環境を調べることです。これは以下の方法で行うことができます。

- システム・モニタリング・ツールを使用します (システムとサイトごとに、異なるツールが適切な場所に存在します)。
- Sybase IQ についての情報を表示するいずれかのストアド・プロシージャを使用します。詳細については、次の項を参照してください。
- インデックス・タイプが適切かどうかを確認します。インデックス・タイプの選択の詳細については、『Sybase IQ システム管理ガイド』の「第 6 章 Sybase IQ インデックスの使用」を参照してください。
- 画面に表示される情報のうち、挿入通知メッセージと削除通知メッセージを調べます。これらのメッセージの詳細については、『Sybase IQ システム管理ガイド』の「第 7 章 データベースへのデータの入出力」を参照してください。

- Sybase IQ メッセージ・ファイルを調べます。デフォルトは *dbname.iqmsg* です。
- Sybase Central のパフォーマンス・モニタを使用します。
- ストアド・プロシージャ、関数、イベントの実行時間を追跡するためにプロシージャ・プロファイリングを使用します。

ストアド・プロシージャを使用して情報を取得する

Sybase IQ には、データベース情報を表示するストアド・プロシージャがいくつかあります。

- sp_iqconnection は、ユーザ接続とバージョンについての統計を表示します。
- sp_iqcontext は、実行中の文についての情報を表示します。
- sp_iqcheckdb は、現在のデータベースの妥当性を検査します。
- sp_iqdbstatistics は、最後に実行された sp_iqcheckdb の結果をレポートします。
- sp_iqdbsize は、現在のデータベースのサイズを取得します。
- sp_iqspaceinfo は、データベース内の各オブジェクトによる領域の使用状況を表示します。
- sp_iqstatus は、データベースのその他のステータス情報を表示します。
- sp_iqtablesize は、指定したテーブルのサイズを取得します。
- sp_iqgroupsize は、指定したグループのメンバをリストします。

Sybase IQ の全ストアド・プロシージャの構文の詳細と例については、『Sybase IQ リファレンス・マニュアル』を参照してください。

Sybase Central パフォーマンス・モニタの使用

Sybase Central を使用して、次のようにサーバの統計情報をモニタリングできます。

❖ Sybase Central でパフォーマンスをモニタリングするには

1 サーバを選択します。

2 [Statistics] タブで、名前を右クリックし、[Add to Performance Monitor] を選択します。

3　[Performance Monitor] タブをクリックします。Sybase Central は、スナップショット間の差だけを追跡するため、選択する統計によっては、パフォーマンス・モニタにアクティビティが表示されないことがあります。

それぞれの統計の説明を参照するには、[Statistics] タブで統計の名前を右クリックし、[Properties] を選択します。パフォーマンス・モニタ上の統計をグラフ化することもできます。それには、[Properties] タブでチェックボックスをオンにして、[Apply] を選択し、[OK] を選択します。

データベース・プロシージャのプロファイリング

プロシージャ・プロファイリングでは、ストアド・プロシージャ、関数、イベント、システム・トリガ、トリガの実行にかかる時間が表示されます。プロシージャの各行の実行時間も表示できます。データベース・プロファイリング情報を使用すると、チューニングによってデータベース内のどのプロシージャのパフォーマンス向上が可能かを判断できます。

プロファイリングが有効になっている場合、Sybase IQ は、使用されているストアド・プロシージャ、関数、イベント、システム・トリガ、トリガをモニタし、それらの実行時間と、それぞれの呼び出し回数を追跡します。

プロファイリング情報はサーバによってメモリに格納され、Sybase Central の [Profile] タブまたは Interactive SQL で表示できます。プロファイリングが有効になると、プロファイリングを無効にするかサーバが停止するまで、データベースはプロファイリング情報を収集します。

Interactive SQL でのプロファイリング情報の取得の詳細については、「Interactive SQL でのプロファイリング情報の表示」(155 ページ) を参照してください。

プロシージャ・プロファイリングの有効化

プロシージャ・プロファイリングは、すべての接続によるプロシージャとトリガの使用状況を追跡します。Sybase Central または Interactive SQL のいずれかで、プロファイリングを有効化できます。プロシージャ・プロファイリングの有効化および使用には、DBA 権限が必要です。

❖ **プロファイリングを有効にするには (Sybase Central の場合)**

1　DBA 権限のあるユーザとしてデータベースに接続します。

2　左ウィンドウ枠でデータベースを選択します。

3　[ファイル] – [プロパティ] を選択します。

[データベース] プロパティ・シートが表示されます。

4　[プロファイリング] タブで、[このデータベースでプロファイリングを可能にする] を選択します。

5　[OK] をクリックして、プロパティ・シートを閉じます。

注意

Sybase Central でデータベースを右クリックしても、プロファイリングを有効化できます。ポップアップ・メニューから [プロファイリング] － [プロファイリングの開始] を選択します。

❖ **プロファイリングを有効にするには (SQL の場合)**

1 DBA 権限のあるユーザとしてデータベースに接続します。

2 ON 設定を指定して sa_server_option ストアド・プロシージャを呼び出します。

たとえば、次のように入力します。

```
CALL sa_server_option ( 'procedure_profiling', 'ON')
```

もし必要なら、他の接続でのデータベースの使用を妨害せずに、特定のユーザが使用しているプロシージャを確認できます。その接続がすでに存在するか、複数のユーザが同じユーザ ID で接続する場合は、この機能が便利です。

❖ **プロシージャ・プロファイリングをユーザでフィルタするには**

1 DBA 権限のあるユーザとしてデータベースに接続します。

2 次のプロシージャを呼び出します。

```
CALL sa_server_option
('ProfileFilterUser','userid')
```

userid の値は、モニタするユーザの名前です。

プロシージャ・プロファイリングのリセット

プロファイリングをリセットすると、データベースは古い情報をクリアし、プロシージャ、関数、イベント、トリガに関する新しい情報の収集をただちに開始します。

以下の項では、DBA 権限を持つユーザとしてデータベースに接続しており、プロシージャ・プロファイリングが有効になっているものとします。

❖ **プロファイリングをリセットするには (Sybase Central の場合)**

1 左ウィンドウ枠でデータベースを選択します。

2 [ファイル] – [プロパティ] を選択します。

[データベース] プロパティ・シートが表示されます。

3 [プロファイリング] タブで、[すぐにリセット] をクリックします。

4 [OK] をクリックして、プロパティ・シートを閉じます。

注意

Sybase Central で右クリックしても、プロファイリングをリセットできます。ポップアップ・メニューから [プロファイリング] － [プロファイリング情報のリセット] を選択します。

❖ **プロファイリングをリセットするには (SQL の場合)**

- RESET 設定を指定して sa_server_option ストアド・プロシージャを呼び出します。

  たとえば、次のように入力します。

```
CALL sa_server_option ('procedure_profiling',
'RESET')
```

プロシージャ・プロファイリングの無効化

プロファイリング情報の使用が終了したら、プロファイリングを無効にするか、プロファイリングをクリアすることができます。プロファイリングを無効にすると、データベースはプロファイリング情報の収集を停止しますが、その時点までに収集された情報は Sybase Central の [プロファイル] タブに引き続き表示されます。プロファイリングを無効にすると、データベースはプロファイリングをオフにし、Sybase Central の [プロファイル] タブからすべてのプロファイリング・データを削除します。

❖ **プロファイリングを無効にするには (Sybase Central の場合)**

1. 左ウィンドウ枠でデータベースを選択します。
2. [ファイル] – [プロパティ] を選択します。

   [データベース] プロパティ・シートが表示されます。
3. [プロファイリング] タブで、[このデータベースでプロファイリングを可能にする] オプションをクリアします。
4. [OK] をクリックして、プロパティ・シートを閉じます。

注意

Sybase Central でデータベースを右クリックしても、プロファイリングを無効化できます。ポップアップ・メニューから [プロファイリング] – [プロファイリングの停止] を選択します。

❖ **プロファイリングを無効にするには (SQL の場合)**

- OFF 設定を指定して sa_server_option ストアド・プロシージャを呼び出します。

  たとえば、次のように入力します。

```
CALL sa_server_option ('procedure_profiling',
'OFF')
```

❖ **プロファイリングをクリアするには (Sybase Central の場合)**

1. 左ウィンドウ枠でデータベースを選択します。
2. [ファイル] – [プロパティ] を選択します。

   [データベース] プロパティ・シートが表示されます。

3 [プロファイリング] タブで、[すぐにクリア] をクリックします。

プロファイリングをクリアできるのは、プロファイリングが有効になっている場合だけです。

4 [OK] をクリックして、プロパティ・シートを閉じます。

注意 Sybase Central でデータベースを右クリックしても、プロファイリングをクリアできます。ポップアップ・メニューから [プロファイリング] − [プロファイリング情報のクリア] を選択します。

❖ **プロファイリングをクリアするには (SQL の場合)**

- CLEAR 設定を指定して sa_server_option ストアド・プロシージャを呼び出します。

  たとえば、次のように入力します。

```
CALL sa_server_option ('procedure_profiling',
'CLEAR')
```

Sybase Central でのプロファイリング情報の表示

プロシージャ・プロファイリングでは、データベース全体、特定のタイプのオブジェクト、特定のプロシージャのいずれの情報を調べたいかによって、それぞれ異なる情報を表示することができます。次のような情報を表示できます。

- データベース内のプロファイル対象のすべてのオブジェクトについての詳細
- すべてのストアド・プロシージャと関数についての詳細
- すべてのイベントについての詳細
- すべてのトリガについての詳細
- すべてのシステム・トリガについての詳細
- プロファイル対象の個別のオブジェクトについての詳細

プロファイリング情報を表示するためには、データベースに接続し、プロファイリングを有効にする必要があります。

データベース全体のプロファイリング情報を表示すると、次のカラムが表示されます。

- **Name** オブジェクトの名前をリストします。
- **Owner** オブジェクトの所有者をリストします。
- **Table** トリガが属するテーブルをリストします (このカラムはデータベースの [プロファイル] タブにのみ表示されます)。
- **Event** システム・トリガのトリガのタイプを表示します。Update、Delete のいずれかです。
- **Type** オブジェクトのタイプ (たとえばプロシージャ) をリストします。

- **# Exes.**　各オブジェクトが呼び出された回数をリストします。
- **#msecs.**　各オブジェクトの合計実行時間をリストします。

これらのカラムには、データベース内で実行されたすべてのプロシージャに関するプロファイリング情報の要約が表示されます。プロシージャは他のプロシージャを呼び出せるため、ユーザが明示的に呼び出したものより多くの項目が存在する可能性があります。

❖ ストアド・プロシージャと関数の要約プロファイリング情報を表示するには

1　左ウインドウ枠で [プロシージャとファンクション] フォルダを選択します。

2　右ウインドウ枠で [プロファイル] タブをクリックします。

データベース内のすべてのストアド・プロシージャと関数のプロファイリング情報が [プロファイル] タブに表示されます。

❖ イベントの要約プロファイリング情報を表示するには

1　左ウィンドウ枠で [イベント] フォルダを開きます。

データベース内のすべてのイベントのリストが右ウィンドウ枠の [イベント] タブに表示されます。

2　右ウィンドウ枠で [プロファイル] タブをクリックします。

データベース内のすべてのイベントのプロファイリング情報が [プロファイル] タブに表示されます。

❖ トリガの要約プロファイリング情報を表示するには

1　左ウィンドウ枠で [トリガ] フォルダを開きます。

データベース内のすべてのトリガのリストが [トリガ] タブに表示されます。

2　右ウィンドウ枠で [プロファイル] タブをクリックします。

データベース内のすべてのトリガのプロファイリング情報が [プロファイル] タブに表示されます。

❖ システム・トリガの要約プロファイリング情報を表示するには

1　左ウィンドウ枠で [システム・トリガ] フォルダを開きます。

データベース内のすべてのトリガのリストが [システム・トリガ] タブに表示されます。

2　右ウィンドウ枠で [プロファイル] タブをクリックします。

データベース内のすべてのシステム・トリガのプロファイリング情報が [プロファイル] タブに表示されます。

特定のプロシージャのプロファイリング情報の表示

Sybase IQ は、個別のストアド・プロシージャ、関数、イベント、トリガのプロシージャ・プロファイリング情報を提供します。Sybase Central には、個別のプロシージャについての情報が表示されます。この情報は、すべてのストアド・プロシージャ、関数、イベント、またはトリガについて表示される情報とは異なります。

特定のプロシージャのプロファイリング情報を表示すると、次のカラムが表示されます。

- **Calls** オブジェクトが呼び出された回数をリストします。
- **Milliseconds** 各オブジェクトの合計実行時間をリストします。
- **Line** プロシージャの各行に行番号を付加します。
- **Source** SQL プロシージャを 1 行ずつ表示します。

プロシージャは行単位に分割されるので、どの行の実行時間が長いか調べることができ、変更によってプロシージャのパフォーマンスを高められる可能性があります。プロシージャ・プロファイリング情報にアクセスするためには、データベースに接続すること、プロファイリングを有効にすること、DBA 権限を持っていることが必要です。

❖ ストアド・プロシージャまたは関数のプロファイリング情報を表示するには

1. 左ウィンドウ枠でデータベースを展開します。
2. 左ウィンドウ枠で [プロシージャとファンクション] フォルダを選択します。

   データベース内のすべてのストアド・プロシージャと関数のリストが右ウィンドウ枠の [プロシージャとファンクション] タブに表示されます。
3. 左ウィンドウ枠で、プロファイル情報を取得するストアド・プロシージャまたは関数をクリックします。
4. 右ウィンドウ枠で [プロファイル] タブをクリックします。

   指定したストアド・プロシージャまたは関数のプロファイリング情報が右ウィンドウ枠の [プロファイル] タブに表示されます。

❖ イベントのプロファイリング情報を表示するには

1. 左ウィンドウ枠でデータベースを展開します。
2. 左ウィンドウ枠で [イベント] フォルダを選択します。

   データベース内のすべてのイベントのリストが右ウィンドウ枠の [イベント] タブに表示されます。
3. 左ウィンドウ枠で、プロファイル情報を取得するイベントをクリックします。

4　右ウィンドウ枠で [プロファイル] タブをクリックします。

指定したイベントのプロファイリング情報が右ウィンドウ枠の [プロファイル] タブに表示されます。

❖ **トリガのプロファイリング情報を表示するには**

1　左ウィンドウ枠でデータベースを展開します。

2　左ウィンドウ枠で [トリガ] フォルダを開きます。

すべてのトリガのリストが右ウィンドウ枠の [トリガ] タブに表示されます。

3　右ウィンドウ枠で、プロファイル情報を取得するトリガを選択します。

4　右ウィンドウ枠で [プロファイル] タブをクリックします。

指定したトリガのプロファイリング情報が右ウィンドウ枠の [プロファイル] タブに表示されます。

❖ **システム・トリガのプロファイリング情報を表示するには**

1　左ウィンドウ枠でデータベースを展開します。

2　左ウィンドウ枠で [システム・トリガ] フォルダを開きます。

すべてのシステム・トリガのリストが右ウィンドウ枠の [システム・トリガ] タブに表示されます。

3　右ウィンドウ枠で、プロファイル情報を取得するシステム・トリガを選択します。

4　右ウィンドウ枠で [プロファイル] タブをクリックします。

指定したシステム・トリガのプロファイリング情報が右ウィンドウ枠の [プロファイル] タブに表示されます。

Interactive SQL でのプロファイリング情報の表示

ストアド・プロシージャを使用して、プロシージャ・プロファイリング情報を表示できます。Sybase Central でも Interactive SQL でも、表示されるプロファイリング情報は同じです。

sa_procedure_profile_summary ストアド・プロシージャは、データベース内のすべてのプロシージャに関する情報を表示します。このプロシージャを使用すると、同じ結果セット内のストアド・プロシージャ、関数、イベント、システム・トリガ、トリガについてのプロファイリング・データを表示できます。次のパラメータはプロシージャが返すローを制限します。

- **p_object_name**　プロファイル情報を取得するオブジェクトの名前を指定します。
- **p_owner_name**　プロファイル情報を取得するオブジェクトの所有者を指定します。

- **p_table_name** トリガのプロファイル情報を取得するテーブルを指定します。
- **p_object_type** プロファイル情報を取得するオブジェクトのタイプを指定します。次の 5 つの選択肢があります。これらの値のいずれかを選択すると、結果セットは指定したタイプのオブジェクトだけに制限されます。
  - **P** ストアド・プロシージャ
  - **F** 関数
  - **T** トリガ
  - **E** イベント
  - **S** システム・トリガ
- **p_ordering** 結果セットのソート順を指定します。

プロシージャは別のプロシージャを呼び出せるため、ユーザが明示的に呼び出したものより多くの項目がリストされる可能性があります。

以下の項では、DBA 権限を持つユーザとしてデータベースに接続しており、プロシージャ・プロファイリングが有効になっているものとします。

❖ **すべてのプロシージャの要約プロファイリング情報を表示するには**

1 sa_procedure_profile_summary ストアド・プロシージャを実行します。

たとえば、次のように入力します。

```
CALL sa_procedure_profile_summary
```

2 [SQL] – [実行] を選択します。

データベース内のすべてのプロシージャの情報が含まれる結果セットが [結果] ウィンドウ枠に表示されます。

sa_procedure_profile_summary ストアド・プロシージャの詳細については、『Adaptive Server Anywhere SQL リファレンス・マニュアル』を参照してください。

Interactive SQL での特定のプロシージャのプロファイリング情報の表示

sa_procedure_profile ストアド・プロシージャは、特定のプロシージャ内の個々の行に関する情報を表示します。結果セットには、プロシージャ内の各行について、行番号、実行時間、合計実行時間に対する割合が含まれています。次のパラメータによって、プロシージャが返すローを制限できます。

- **p_object_name** プロファイル情報を取得するオブジェクトの名前を指定します。
- **p_owner_name** プロファイル情報を取得するオブジェクトの所有者を指定します。
- **p_table_name** トリガのプロファイル情報を取得するテーブルを指定します。

クエリでパラメータを指定しない場合、プロシージャは、呼び出されたすべてのプロシージャのプロファイリング情報を返します。

❖ **プロシージャ内の特定の行のプロファイリング情報を表示するには**

1 sa_procedure_profile ストアド・プロシージャを実行します。

たとえば、次のように入力します。

```
CALL sa_procedure_profile
```

2 [SQL] – [実行] を選択します。

プロシージャの個々の行のプロファイリング情報が含まれる結果セットが [結果] ウィンドウ枠に表示されます。

sa_procedure_profile ストアド・プロシージャの詳細については、『Adaptive Server Anywhere SQL リファレンス・マニュアル』を参照してください。

バッファ・キャッシュのモニタリング

Sybase IQ には、バッファ・キャッシュのパフォーマンスをモニタリングするツールが用意されています。このモニタは、バッファ・キャッシュ、メモリ、Sybase IQ 内で行われた I/O 関数の統計情報を収集し、それらをログ・ファイルに保管します。

バッファ・キャッシュのパフォーマンスは、Sybase IQ 全体のパフォーマンスにとって重要な要因です。モニタが提供する情報を使用して、メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュに割り付けるメモリの量を微調整できます。あるキャッシュが他のキャッシュよりもかなり多くの I/O を実行している場合は、共有メモリを適切に再割り付けしてください。再割り付けする場合は、10 ～ 50MB 単位で少しずつ、繰り返して行います。再割り付けが終了したら、負荷を再実行し、パフォーマンス上の変化をモニタリングします。

バッファ・キャッシュ・モニタの起動

Sybase IQ バッファ・キャッシュ・モニタを DBISQL から起動します。モニタを起動するたびに、各モニタは Sybase IQ 内で別々のカーネル・スレッドとして実行されます。

次の構文を使って、モニタを起動します。

```
IQ UTILITIES { MAIN | PRIVATE }
 INTO dummy_table_name
 START MONITOR 'monitor_options [ ... ]'
```

MAIN を指定すると、接続先データベースの IQ ストア内のすべてのテーブルについて、メイン・バッファ・キャッシュのモニタリングを開始します。

PRIVATE を指定すると、接続先データベースのテンポラリ・ストア内のすべてのテーブルについて、テンポラリ・バッファ・キャッシュのモニタリングを開始します。

バッファ・キャッシュごとにコマンドを別々に発行する必要があります。モニタが結果を収集している間は、これらの各セッションを開いておく必要があります。接続を閉じると、モニタは実行を停止します。接続は最大で 2 つのモニタの実行まで開くことができます。1 つはメイン・バッファ・キャッシュ用で、もう 1 つはテンポラリ・バッファ・キャッシュ用です。

dummy_table_name には、任意の Sybase IQ ベース・テーブルまたはテンポラリ・テーブルを指定します。他の IQ UTILITIES コマンドと構文上の互換性を持たせるために、テーブル名を指定する必要があります。最も望ましいのは、モニタリング専用のテーブルを作成することです。

モニタリング出力ファイルのディレクトリ位置を制御するには、MONITOR_OUTPUT_DIRECTORY オプションを設定します。このオプションを設定しない場合は、データベースと同じディレクトリに結果が出力されます。モニタを実行している間、すべてのモニタリング出力ファイルが使用されます。モニタの実行が停止した後も、ファイルはそのまま残ります。

マルチプレックス・クエリ・サーバを作成する前に、モニタリングで使用するテンポラリ・テーブルを宣言するか、新しいデータベースの作成時に永続的なダミー・テーブルを作成してください。これによって DDL の変更を回避し、実際の運用稼働時にデータがクエリ・サーバにとどまるようにします。

ヒント
モニタを簡単に使用するためには、ストアド・プロシージャを作成してダミー・テーブルを宣言し、出力ロケーションを指定して、モニタを起動します。

'*monitor_options*' には次の値を 1 つ以上指定できます。

- -summary。メインとテンポラリの両方のバッファ・キャッシュのサマリ情報を表示します。モニタ・オプションを何も指定しないと、サマリ・レポートが表示されます。他のオプションで説明されているフィールドに加えて、次のフィールドが表示されます。
  - [*Users*]：バッファ・キャッシュに接続しているユーザ数。
  - [*IO*]：バッファ・キャッシュによる物理読み込みと物理書き込みの合計。
- -cache。メイン・バッファ・キャッシュまたはテンポラリ・バッファ・キャッシュに対するアクティビティの詳細を表示します。重要なフィールドは [Finds]、[HR%]、[BWaits] です。次のフィールドが表示されます。
  - [*Finds*]：バッファ・キャッシュへの検索要求。Finds の値がゼロに急降下してゼロのままなら、サーバにデッドロックが発生しています。サーバでアクティビティがあれば、Finds はゼロ以外の値を示すはずです。
  - [*Creats*]：データベース内の 1 ページの作成要求。
  - [*Dests*]：データベース内の 1 ページの破棄要求。

- [*Dirty*]：バッファがダーティ (変更) された回数。
- [*HR%*]：ヒット率。I/O 要求なしで、バッファ・キャッシュによって応じることのできたパーセンテージ。ヒット率が高いほど効率が良くなります。キャッシュ・サイズを十分な大きさに設定した場合、通常は 90% ～ 100% になります。大きいクエリの場合、最初はヒット率が下がることがありますが、プリフェッチが機能し始めると上昇します。
- [*BWaits*]：ビジー・ページ (ページ・フレーム競合) のために待機させられた検索要求。通常は小さい数値ですが、特別な場合には大きくなることがあります。たとえば、まったく同じクエリが同時に開始された場合は、両方のクエリが同じページを要求するため、最初の要求がディスクからページを取得するまで 2 番目の要求は待機します。
- [*ReReads*]：同一トランザクション内でストアの同じ部分がキャッシュ内に再読み込みされた概算の回数。この数値は常に小さいはずですが、Sybase IQ 12.4.2 以降では数値が大きくても重要ではありません。
- [*FMiss*]：不正な失敗。バッファ・キャッシュがメモリ内のページの検索に複数回のルックアップを必要とした回数。この数値は 0 か、ごく小さな値になるようにしてください。この値が大きい場合は、ロールバックが発生し、特定の操作が繰り返し要求されていると考えられます。
- [*Cloned*]：Sybase IQ が同時読み込み用に既存のバッファを保持しながら、書き込み用に新しいバッファを作成する必要があった回数。ページのクローンが作成されるのは、他のユーザがそのページを参照している場合だけです。
- [*Reads/Writes*]：バッファ・キャッシュによって実行された物理読み込みと物理書き込み。
- [*PF/PFRead*]：プリフェッチ要求、およびプリフェッチ用に実行した読み込み。
- [*GDirty*]：LRU バッファがダーティな状態で取り込まれたため、Sybase IQ がそのバッファを使用する前に書き出した回数。この数値が 0 より大きい状態が長時間続かないようにしてください。0 より大きい状態が続く場合は、スイーパ・スレッドの数を増やすか、ウォッシュ・マーカを移動する必要があります。
- [*Pin%*]：バッファ・キャッシュ内で使用中でありロックされているページ数のパーセンテージ。
- [*Dirty%*]：変更されたバッファ・ブロックのパーセンテージ。この数値が 85 ～ 90% を超えないようにします。それ以上になると、[GDirty] が 0 より大きくなります。

- -cache_by_type。生成する結果は -cache と同じですが、結果を IQ ページ・タイプごとに集計します ([Bwaits] カラムは例外で、合計だけを表示します)。この形式は、Sybase 製品の保守契約を結んでいるサポート・センタに情報を送る場合にたいへん有効です。
- -file_suffix *suffix*。<dbname>.<connid>-<main_or_temp>-<suffix> の名前で、モニタリング出力ファイルを作成します。suffix を指定しないと、デフォルトで *iqmon* に設定されます。
- -io。指定した期間のメインまたはテンポラリ (プライベート) のバッファ・キャッシュの I/O 率と圧縮比を表示します。これらのカウンタは、サーバのすべてのアクティビティを表します。情報はデバイス別に集計されません。次のフィールドが表示されます。
  - [*Reads*]：バッファ・キャッシュによって実行された物理読み込み。
  - [*Lrd(KB)*]：読み込まれた論理キロバイト数 (ページ・サイズに要求数を乗算した数値)。
  - [*Prd(KB)*]：読み込まれた物理キロバイト数。
  - [*Rratio*]：読み込まれた物理データに対する論理データの圧縮比。読み込み時のディスクに対する圧縮効率の度合い。
  - [*Writes*]：バッファ・キャッシュによって実行された物理書き込み。
  - [*Lwrt(KB)*]：書き込まれた論理キロバイト数。
  - [*Pwrt(KB)*]：書き込まれた物理キロバイト数。
  - [*Wratio*]：書き込まれた物理データに対する論理データの圧縮比。
- -bufalloc。ソート、ハッシュ、ビットマップなどのオブジェクト用にバッファ・キャッシュ内の領域を予約する、メインまたはテンポラリ・バッファ・アロケータ情報を表示します。
  - [*OU*]：User_Resource_Reservation オプション設定 (以前は Optimize_For_This_Many_Users)
  - [*AU*]：現在アクティブなユーザ数。
  - [*MaxBuf*]：バッファ・アロケータに制御されているバッファ数。
  - [*Avail*]：現在ピン・クォータ割り付けに使用可能なバッファ数。
  - [*AvPF*]：現在プリフェッチ・クォータ割り付けに使用可能なバッファ数。
  - [*Slots*]：バッファ・キャッシュ・クォータを使用中の、現在登録されているオブジェクト数。
  - [*PinUser*]：ピン・クォータを使用中のオブジェクト数 (ハッシュ、ソート、B ツリー・オブジェクトなど)。
  - [*PFUsr*]：プリフェッチ・クォータを使用中のオブジェクト数。

- [*Posted*]：あらかじめプランされたクォータ・ユーザであるオブジェクト数。
- [*UnPost*]：特定のクォータ・ユーザであるオブジェクト数。
- [*Locks*]：バッファ・アロケータで処理されたミューテックス・ロック数。
- [*Waits*]：ロックのためにスレッドが待機する必要があった回数。

- -contention。多くの重要なバッファ・キャッシュとメモリ・マネージャ・ロックを表示します。これらのロック・カウンタとミューテックス・カウンタは、バッファ・キャッシュおよびヒープ・メモリ内のアクティビティと、これらのロックがどれだけ迅速に解消されたかを示します。タイムアウトの数値に注目してください。システム時間が 20% を超えている場合は、問題の発生を示しています。

  注意　オペレーティング・システムが進歩したため、Sybase IQ ではスピン・ロックを使用しなくなりました。このため、[woTO]、[Loops]、[TOs] の統計はめったに使用されません。

  - [*AU*]：現在アクティブなユーザ数。
  - [*LRULks*]：LRU がロックされた (テンポラリ・キャッシュ用に繰り返された) 回数。
  - [*woTO*]：タイムアウトなしにロックが付与された (テンポラリ・キャッシュ用に繰り返された) 回数。
  - [*Loops*]：ロックが付与される前に Sybase IQ がリトライした (テンポラリ・キャッシュ用に繰り返された) 回数。
  - [*TOs*]：Sybase IQ がタイムアウトして、ロックのために待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。
  - [*BWaits*]：キャッシュ内のバッファに対する “Busy Waits” の (テンポラリ・キャッシュ用に繰り返された) 回数。
  - [*IOLock*]：Sybase IQ が圧縮化 I/O プールをロックした (テンポラリ・キャッシュ用に繰り返された) 回数。無視してかまわない。
  - [*IOWait*]：圧縮化 I/O プール上のロックのために Sybase IQ が待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。無視してかまわない。
  - [*HTLock*]：Sybase IQ がブロック・マップ・ハッシュ・テーブルをロックした (テンポラリ・キャッシュ用に繰り返された) 回数。
  - [*HTWait*]：ブロック・マップ・ハッシュ・テーブルのために Sybase IQ が待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。HTLock と HTWait は、使用中のブロック・マップ数を示す。

- [*FLLock*]：Sybase IQ がフリー・リストをロックする必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。
- [*FLWait*]：フリー・リスト上のロックのために Sybase IQ が待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。
- [*MemLks*]：Sybase IQ がメモリ・マネージャ (ヒープ) をロックした回数。
- [*MemWts*]：メモリ・マネージャ・ロックのために Sybase IQ が待機する必要があった回数。

- -threads。処理スレッド・マネージャが使用したカウンタを表示します。値はサーバワイドです (つまり、メインとプライベートのどちらでこのオプションを選択するかは無関係です)。レポートの最後のページ以降の新しいイベントを表します。
  - [*cpus*]：Sybase IQ が使用している CPU の数。システムに搭載されている数より少ない場合がある。
  - [*Limit*]：Sybase IQ が使用できるスレッドの最大数。
  - [*NTeams*]：現在使用中のスレッド・チームの数。
  - [*MaxTms*]：今まで使用されたチームの最大数。
  - [*NThrds*]：既存スレッドの現在の数。
  - [*Resrvd*]：システム (接続) での使用のために予約されているスレッドの数。
  - [*Free*]：割り当てに使用可能なスレッドの数。モニタリングが必要。この数値が非常に小さい場合は、スレッドの不足を示している。
  - [*Locks*]：スレッド・マネージャで処理されたロックの数。
  - [*Waits*]：スレッド・マネージャ上のロックのために Sybase IQ が待機する必要があった回数。

注意 オブジェクトまたはクエリが作業を必要としている場合、Sybase IQ はスレッド・チームと呼ばれる処理スレッドのグループを割り付けます。処理スレッドの調整を行う際に、利用可能なオプションとしては、データベース・オプションの MAX_IQ_THREADS_PER_CONNECTION と MAX_IQ_THREADS_PER_TEAM、および Sybase IQ が使用できるスレッド数を指定するサーバ・オプションの -iqmt があります。

- -interval。レポート間隔を秒単位で指定します。デフォルトは 60 秒ごとです。最小値は 2 秒です。通常、クエリの実行中やパフォーマンスに問題があるときに、モニタをデフォルトの間隔で実行すると、有益な結果を取得できます。間隔が短すぎると、意味のある結果を取得できないことがあります。ジョブ時間に見合った間隔を指定してください。通常は 1 分で十分です。

  最初の表示では、サーバの起動からのカウンタが示されます。それ以降の表示では、前の表示との差が示されます。

- -append | - truncate。前者は既存の出力ファイルに追加、後者は既存の出力ファイルをトランケートします。デフォルトでは、トランケートされます。

- -debug。主に、Sybase 製品の保守契約を結んでいるサポート・センタに情報を提供するために使用します。これを指定すると、同じ情報を扱う標準表示モードがあるかどうかにかかわらず、パフォーマンス・モニタで使用可能な情報がすべて表示されます。ページの上部には、ディスク・ブロック・タイプごとの統計の配列が表示されます。次に、他のバッファ・キャッシュの統計、メモリ・マネージャの統計、スレッド・マネージャの統計、フリー・リストの統計、CPU 使用率、そして最後にバッファ・アロケータの統計が表示されます。バッファ・アロケータの統計はさらにクライアント・タイプ (ハッシュ、ソートなど) ごとに集計され、最後に行われたバッファ割り付けのヒストグラムが表示されます。メモリ割り付けは、レポートの最後のページ以降に割り付けられた量を示すことに注意してください。

注意　表示する間隔は、ページ単位ではなく、出力行単位です。ただし、次の 2 つの場合は例外です。-cache_by_type と -debug では、表示ごとに新しいページが開始されます。

モニタ実行中の結果の確認

UNIX システムでは、クエリの実行中にモニタリング出力を確認できます。

たとえば、次のコマンドを使用してモニタを起動するとします。

```
iq utilities main into monitor_tab
start monitor "-cache -interval 2 -file_suffix iqmon"
```

このコマンドを実行すると、結果が *dbname.conn#-[main|temp]-iqmon* という名前の ASCII ファイルに出力されます。したがって、データベース asiqdemo では、結果が *asiqdemo.2-main-iqmon* に出力されます。

結果を確認するには、システム・プロンプトで次のコマンドを入力します。

```
$ tail -f asiqdemo.2-main-iqmon
```

バッファ・キャッシュ・モニタの停止

モニタの停止コマンドは起動コマンドとほぼ同じですが、オプションを指定する必要はありません。次の構文を使って Sybase IQ バッファ・キャッシュ・モニタを停止します。

```
IQ UTILITIES { MAIN | PRIVATE }
  INTO dummy_table_name STOP MONITOR
```

注意 特定のオプション設定を有効にするためには、データベースを再起動します。モニタを実行している場合は、データベースを再起動できるようにモニタをシャットダウンする必要があります。

モニタリング結果の検査と保存

モニタは、結果を普通のテキスト・ファイルに保存します。このファイルのデフォルトは、次のとおりです。

- *dbname.connection#-main-iqmon* (メイン・バッファ・キャッシュの結果の場合)
- *dbname.connection#-temp-iqmon* (テンポラリ・バッファ・キャッシュの結果の場合)

プレフィクス *dbname.connection#* は、データベース名と接続番号を示します。接続番号が複数あって、どれが自分のものかわからない場合は、カタログ・ストアド・プロシージャ sa_conn_info を実行してください。このプロシージャを実行すると、アクティブなデータベース接続のそれぞれについて、接続番号、ユーザ ID などの情報が表示されます。

IQ UTILITIES コマンドで -file_suffix パラメータを使用すると、サフィックス *iqmon* を任意のサフィックスに変更できます。

モニタの実行結果を表示するには、テキスト・エディタを使用するか、ファイルの表示や印刷に通常使用している方法がほかにあれば、それを使用してください。

同じデータベースから同じ接続番号を使ってモニタを再度起動する場合、デフォルトでは前回の結果が上書きされます。モニタの実行結果を保存する場合は、ファイルを別の場所にコピーした後で同じデータベースからモニタを再度起動するか、-append オプションを使用してください。

モニタリング結果の例

この項では、いろいろなモニタリング・オプションを使用したサンプル結果を示します。

-summary オプションを使用すると、次のような結果が生成されます。IQ UTILITIES コマンドでの指定にかかわらず、メインとテンポラリ両方のバッファ・キャッシュの統計が表示されることに注意してください。

```
 Sybase Adaptive Server IQ Performance Monitor

                ---------------------------------------------

                             Version 3.2


Options string for Main cache: "-summary -interval 5"

                                                                      Summary

                                                                   2004-07-16 13:53:24

Active|                Main Cache                      |             Temp Cache

 Users|  Finds   HR%  Reads/Writes  GDirty  Pin% Dirty% InUse%|  Finds   HR%
Reads/Writes  GDirty  Pin% Dirty% InUse%

     0     286  99.3      2/34           0   0.0    1.6   26.2     608  99.7
2/47           0   0.0    3.6   20.0

     1    2621  99.4     16/155          0   5.6    8.7   81.7    4121  99.6
16/163          0  11.4   23.2   67.3

     1    2646  99.8      6/48           0   1.6   13.5  100.0    3388  99.8
6/70           1   4.1   40.9   94.5

     1    2684  99.9      7/78           0   5.6   14.3  100.0    3497  99.9
8/103          1  10.9   42.3   99.1

     1    1993  99.9     17/22           0   4.0   31.0  100.0    3342  98.7
122/149         0   8.2   41.4   91.4

     1    2479  99.9     32/110          0   5.6   13.5  100.0    3370  99.8
55/112          0  11.4   45.5   95.9

     1    3273 100.0      0/0            0   5.6   23.8  100.0    3951 100.0
0/108          1  13.6   49.1  100.0

     1    2512  99.9      2/0            0   1.6   31.0  100.0    3916  98.9
88/173          0   5.5   48.6  100.0

     1    1264  99.9     66/131          0   4.0   45.2  100.0    4317  98.9
378/305         0   6.4   40.0   77.3

     1    2122  99.8     30/125          0   5.6   12.7   99.2    3122  99.7
67/127          0  12.3   40.0   90.5

     1    3370 100.0      2/0            0   5.6   23.0  100.0    4034 100.0
2/98           2  13.2   46.4   98.2
```

```
     1    2981  99.9      2/0            0   5.6   31.7  100.0    3715  99.9
2/110          0  14.1   53.2  100.0

     1    3351  99.6     13/3            0   5.6   39.7  100.0    4131  99.7
13/123         0  14.1   57.7  100.0

     1    3286  99.6     13/13           0   5.6   40.5  100.0    4135  99.6
15/139         0  12.3   55.9   97.7

     1     296 100.0      0/0            0   1.6   41.3  100.0    3646  96.9
366/320         0   7.3   53.2  100.0

     1    1230  99.4     71/129          0   6.3   58.7  100.0    4221  98.9
390/297         0   9.5   59.1   91.8

     1    1900 100.0    125/279          0   4.0   50.0  100.0    4102 100.0
344/279         0   7.7   38.6   72.3


                Sybase Adaptive Server IQ Performance Monitor

                ------------------------------------------

                             Shutting Down


     0     422  98.8     16/99           0   0.0    0.8   99.2     853  98.9
34/101         0   0.0    1.8   59.1
```

-cache オプションを使用すると、次のような結果が生成されます。これはテンポラリ・バッファ・キャッシュのものです。

```
Options string for Temp cache: "-cache -interval 10"
                          Temp Shared Buffer Cache
                            2001-02-18 17:43:55
    Finds Creats Dests Dirty   HR% BWaits ReReads FMiss Cloned Reads/     PF/
GDirty Pin% Dirty%
                                                                Writes PFRead
Tm:   640     82    57    84  99.4      0       4     0      0    4/0    0/0
0  0.0   2.8
Tm:  1139    109    83   109 100.0      0       0     0      0    0/0    0/0
0  0.0   5.5
Tm:  6794    754   749   754 100.0      0       0     0      0    0/0    0/0
0  0.0   6.1
Tm: 10759   1646  1646  1646 100.0      0       0     0      0    0/0    0/0
0  0.0   6.1
```

-io オプションを使用すると、次のような結果が生成されます。これはメイン・バッファ・キャッシュのものです。

```
Options string for main cache: "-IO -interval 5"
                         Main Buffer Cache
                        2001-02-18 13:58:48
                Input                           Output
      Reads Lrd(KB) Prd(KB)    Rratio  Writes  Lwrt(KB)  Pwrt(KB)  Wratio
Mn:       10      40      34      1.18      14        56        23    2.43
```

```
Mn:       0       0       0      0.00      21        84        34    2.43
Mn:       0       0       0      0.00       7        28        11    2.43
Mn:       0       0       0      0.00      22        88        35    2.48
Mn:       0       0       0      0.00      63       252       100    2.51
Mn:       0       0       0      0.00      54       216        93    2.32
Mn:       0       0       0      0.00      64       256       101    2.52
Mn:       0       0       0      0.00      62       248        94    2.62
Mn:       0       0       0      0.00      73       292       110    2.65
Mn:       0       0       0      0.00     105       420       121    3.47
```

-buffalloc オプションを使用すると、次のような結果が生成されます。

```
 Options string for Main cache: "-bufalloc -file_suffix bufalloc-iqmon -append -interval
10"

                              Buffer Allocation
2001-02-18 10:58:39

OU/AU MaxBuf  Avail   AvPF   Slots  PinUsr  PFUsr Posted  UnPost Quota  Locks  Waits
1/0    1592   1592     20       0       0      0       0       0      0      1       0

1/1    1592   1592     20       0       0      0       0       0      0      1       0

1/1    1592   1592     20       0       0      0       0       0      0      1       0
```

注意　実際の -contention 出力では、メイン・キャッシュ、テンポラリ・キャッシュ、メモリ・マネージャの結果が同じ行に表示されます。このフォーマットでは横幅が大変広くなってしまうため、ここでは各カラム・セットを別々に示します。

メイン・キャッシュの -contention 結果は、次のとおりです。

```
Options string for Main cache:
"-contention -file_suffix contention-iqmon -append -interval 10"
                          Contention
                          2001-02-18 10:57:03


                         Main Cache                                          |
AU |LRULks woTO  Loops    TOs BWaits IOLock IOWait HTLock HTWait FLLock FLWait

 0     66      0       0      0      0       1       0      5       0       4    0
 1   2958      0       0      0      0     160       0   1117       0       6    0
 1   1513      0       0      0      1     378       0      2       0       8    0
 1    370      0       0      0      0      94       0      2       0      10    0
 1    156      0       0      0      0      46       0      2       0      12    0
 1    885      0       0      0      0     248       0      2       0      14    0
 1   1223      0       0      0      0     332       1      2       0      16    0
 1    346      0       0      0      0      66       0      2       0      18    0
```

テンポラリ・キャッシュの -contention 結果は、次のとおりです。

```
                    Temp Cache
|LRULks   woTO  Loops    TOs BWaits IOLock IOWait HTLock HTWait FLLock FLWait
70        0       0       0      0      1      0      4      0      5      0
466       0       0       0      0      2      0     15      0     12      0
963       0       0       0      0      2      0      8      0     20      1
1186      0       0       0      0      2      0      2      0     23      1
357       0       0       0      0      2      0      2      0     25      1
444       0       0       0      0      2      0      3      0     29      0
884       0       0       0      0      2      0      2      0     31      1
1573      0       0       0      0      2      0      5      0     37      1
```

メモリ・マネージャの結果は、次のとおりです。

```
| Memory Mgr
MemLks MemWts
55483    13
5705      0
2048      0
186       4
2         0
137       0
22        0
203       3
```

-threads オプションを使用すると、次のような結果が生成されます。

```
Options string for Main cache: "-threads -file_suffix threads-iqmon -append -interval 10"


                                Threads

                          2001-02-18 10:59:24


 CPUs  Limit NTeams MaxTms NThrds  Resrvd Free  Locks Waits
 10    100      4     12    100     13     68    106    590
 10    100      6     12    100     12     63      4      6
 10    100      6     12    100     12     63      0      0
 10    100      7     12    100     12     62      1      1
 10    100      7     12    100     12     62      0      0
 10    100      7     12    100     12     58      1      5
 10    100      7     12    100     12     58      0      0
```

バッファ・キャッシュの構造

Sybase IQ では、システム上の CPU の数に応じて、バッファ・キャッシュのキャッシュ・パーティションの数が自動的に計算されます。マルチ CPU 構成でロードまたはクエリのパフォーマンスが予想より悪い場合は、CACHE_PARTITIONS データベース・オプションの値を変更するとパフォーマンスが向上することがあります。詳細については、『Sybase IQ リファレンス・マニュアル』の「CACHE_PARTITIONS オプション」を参照してください。

バッファは、キャッシュの LRU (Least Recently Used) 側の終端に近づくと、ウォッシュ・マーカを越えます。Sybase IQ は最も古いページ (ウォッシュ・マーカを越えたページ) をディスクに書き出して、そのページが占有していたキャッシュ領域を再利用できるようにします。スイーパ・スレッドと呼ばれる Sybase IQ 処理スレッドのチームが、最も古いバッファを一掃し (書き出し) ます。

データのページをキャッシュに読み込む必要がある場合、Sybase IQ は LRU バッファを取り込みます。バッファがまだ「ダーティな」(変更された) 状態の場合は、先にバッファをディスクに書き込む必要があります。モニタ -cache レポートの [Gdirty] カラムは、LRU バッファをダーティな状態で取り込んだために、Sybase IQ がそのバッファを使用する前に書き出す必要があった回数を示します。

通常、Sybase IQ では [Gdirty] の値が 0 に維持されます。この値が 0 より大きい状態が長時間続く場合は、スイーパ・スレッドの数とウォッシュ・マーカを制御するデータベース・オプションを調整する必要があります。詳細については、『Sybase IQ リファレンス・マニュアル』の「第 2 章 データベース・オプション」の「SWEEPER_THREADS_PERCENT オプション」または「WASH_AREA_BUFFERS_PERCENT オプション」を参照してください。

バッファ・マネージャのスラッシングの回避

オペレーティング・システムによるページングは、使用可能な空きメモリを超えるバッファを必要とするクエリに影響します。特に、バッファ・キャッシュにさらに多くの物理メモリを割り付けようとすると、ページングが少々必要です。ただし、物理メモリをバッファ・キャッシュに過度に割り付けると、オペレーティング・システムによるページングの発生頻度が高くなり、システム全体がスラッシングする原因となることがあります。逆に、バッファ・キャッシュに十分なメモリを割り付けないと、Sybase IQ がスラッシングしてしまいます。

オペレーティング・システムがディスクにページ・アウトする最適バッファの量を減らした場合、バッファ・マネージャはこれらのバッファをメモリに戻すためにディスクから余分に読み込みを行わなければならないため、バッファ・マネージャのスラッシングが発生します。Sybase IQ では、ディスクにフラッシュするための最適なバッファを把握しているため、ページ・アウトの全体の数を減らすことによって、このようなオペレーティング・システムによる干渉を回避できます。

バッファ・サイズを設定するときは、次のトレードオフに注意してください。

- Sybase IQ バッファ・キャッシュが大きすぎると、Sybase IQ が全メモリを使用しようとするため、オペレーティング・システムでページングが強制的に行われる。
- Sybase IQ バッファ・キャッシュが小さすぎると、Sybase IQ はクエリ・データをキャッシュに収めきれないため、スラッシングしてしまう。

深刻なパフォーマンスの問題が発生した場合は、ページングをモニタリングして、スラッシングが問題かどうかを確認してください。スラッシングが問題だった場合は、「バッファ・キャッシュの管理」の説明に従って、バッファ・サイズをリセットしてください。

ページングをモニタリングして、スラッシングが問題と判断した場合は、ハッシュ・アルゴリズムを伴うクエリが含まれる文の実行時のスラッシングの量を制限することもできます。HASH_THRASHING_PERCENT データベース・オプションを調整し、許容するハード・ディスク I/O の割合を制御します。この割合を超えると、文がロールバックされてエラーが返されます。

HASH_THRASHING_PERCENT のデフォルト値は 10% です。HASH_THRASHING_PERCENT の値を増やすと、ロールバックが起きるまでに許容するディスクへのページングが増え、HASH_THRASHING_PERCENT の値を減らすと、ロールバックが起きるまでに許容するページングが減ります。

以前のバージョンの Sybase IQ では実行されていた、ハッシュ・アルゴリズムを伴うクエリが、デフォルトの HASH_THRASHING_PERCENT の制限に達するとロールバックされるようになります。Sybase IQ は `Hash insert thrashing detected` または `Hash find thrashing detected` エラーをレポートします。実行に必要なリソースをクエリに割り当てるには、次の 1 つ以上の対応策を講じてください。

- HASH_THRASHING_PERCENT の値を増やし、ページングの制限を緩和します。
- テンポラリ・キャッシュのサイズを増やします (DBA のみ)。テンポラリ・キャッシュのサイズを増やすと、メイン・キャッシュのサイズが減ることに注意してください。
- Sybase IQ がこの文の 1 つ以上のハッシュ・サイズの見積もりを誤っている原因を突き止めて改善します。たとえば、LF または HG インデックスを必要とするすべてのカラムにそれがあるかどうかを確認します。また、複数カラムのインデックスが適切かどうかも検討します。
- データベース・オプション HASH_PINNABLE_CACHE_PERCENT の値を減らします。

これらのデータベース・オプションの詳細については、『Sybase IQ リファレンス・マニュアル』の「第 2 章 データベース・オプション」の「HASH_THRASHING_PERCENT オプション」と「HASH_PINNABLE_CACHE_PERCENT オプション」の項を参照してください。

クエリで起きている可能性のある問題を特定するには、テンポラリ・データベース・オプション QUERY_PLAN = 'ON' と QUERY_DETAIL = 'ON' を指定してクエリを実行し、クエリ・プランを生成します。そして、クエリ・プランの見積もりを調査します。生成されるクエリ・プランは、メッセージ・ログ・ファイルにあります。

Windows システムでのページングのモニタリング

Windows のシステム モニタを使用して、ページングをモニタリングできます。アクセスするには、[Logical Disk] オブジェクト、*PAGEFILE.SYS* ファイルが格納されているディスクのインスタンス、[Disk Transfers/Sec] カウンタを選択します。このファイルはデータベース・ファイルとは別のディスクに格納してください。[Object Memory] と [Pages/Sec] カウンタもモニタリングできます。ただし、この値はソフト・フォールトとハード・フォールトの両方を含む全メモリ・フォールトの合計となります。

UNIX システムでのページングのモニタリング

UNIX のシステム・コマンド vmstat を使用して、ページングなどのシステム・アクティビティをモニタリングできます。コマンドの省略形構文は次のとおりです。

vmstat *interval count*

interval には次の行を出力するまでの時間間隔を、*count* には出力行の表示回数を指定します。オプションとフィールドの説明を含めた vmstat の詳細については、使用しているオペレーティング・システムのマニュアルを参照してください。たとえば次のようになります。

```
>  vmstat 2 3
procs    memory            page            disk     faults      cpu
r b w swap    free  re mf pi po fr de sr s0 s1 sd  in   sy  cs us sy id

0 0 0 3312376 31840 0  8  0  0  0  0  0   0  0  0 297  201 472 82  4 14
0 0 0 3312376 31484 2  3  0  0  0  0  0   0  0  0 260  169 597 80  3 17
0 0 0 3312368 31116 0  8  0  0  0  0  0   0  0  0 205 1202 396 67  4 29
```

上記の出力では、マシンの物理メモリが過度に割り付けられていないため、Sybase IQ のクエリ状態が安定していることが示されています。システムのページ・フォールトはほとんど発生していません。次の例では、問題があることが vmstat 出力に示されています (見やすくなるように、一部のフィールドを省略しています)。

```
procs     memory            page                faults       cpu
r b w swap    free   re   mf  pi po fr de sr  in   sy   cs us sy  id

0 0 0 217348  272784  0  148  11  3  9  0  2 251 1835  601  6  3  91
0 0 0 3487124 205572  0    5   0  0  0  0  0  86  131  133  0  1  99
0 0 0 3487124 205572  0    5   0  0  0  0  0  71  162  121  0  0 100
0 0 0 3483912 204500  0  425  36  0  0  0  0 169  642  355  2  2  96
0 0 0 3482740 203372  0   17   6  0  0  0  0 158  370  210  1  3  97
0 0 0 3482676 203300  0    4  10  0  0  0  0 160 1344  225  1  2  97
0 0 0 3343272 199964  1 2123  36  0  0  0  0 213  131  399  7  8  85
0 0 0 3343264 185096  0  194  84  0  0  0  0 283  796  732  1  6  93
0 0 0 3342988 183972  0   17  58  0  0  0  0 276 1051  746  2  4  94
0 0 0 3342860 183632  0  119 314  0  0  0  0 203 1660  529  3  4  94
0 0 0 3342748 182316  2  109 184  0  0  0  0 187  620  488  4  2  95
0 0 0 3342312 181104  2  147  96  0  0  0  0 115  256  260  9  2  89
0 0 0 3340748 179180  0  899  26  0  0  0  0 163  836  531  4  4  92
0 0 0 3328704 167224  0 2993   6  0  0  0  0  82 2195  222  4  7  89
```

上記の出力の最初の行には、マシンが起動されてからのシステム・アクティビティの概要が示されています。最初の 3 行は、約 200MB の空き物理メモリがあり、マシンがアイドル状態であることを示しています。4 行目は、Sybase IQ が最初に起動されたときと対応しています。8 行目から、空きメモリの量が急速に減少し始めています。これは、Sybase IQ バッファ・キャッシュが割り付けられ、データベース・ページがディスクから読み込まれているときと対応しています (CPU 使用率が増加していることに注意してください)。この時点ではクエリが開始されていないため、ユーザの CPU 時間がほとんどありません。

```
procs     memory               page              faults      cpu
r  b w swap    free   re mf   pi   po fr de sr in    sy  cs   us sy id

 7 0 0 3247636 58920  0 1880 1664 0  0  0  0  1131  442 1668 80 18  3
18 0 0 3246568 43732  0  709 1696 0  0  0  0  1084  223 1308 90 10  1
12 0 0 3246604 37004  0  358  656 0  0  0  0   600  236  722 95  5  0
15 0 0 3246628 32156  0  356 1606 0  0  0  0  1141  226 1317 91  9  0
19 0 0 3246612 26748  0  273 1248 0  0  0  0   950  394 1180 92  7  0
```

上記の出力は少し後の、クエリが進行中のときのものです。これは、ユーザ・モードの CPU レベル (*us* フィールド) を見ると明らかです。依然としてページイン・フォールト (ページインされた KB を示す、*pi* フィールド) が発生しており、空きメモリの量が減少しているため、バッファ・キャッシュはまだ満杯になっていません。

```
procs     memory            page                    faults        cpu
r  b w swap    free  re mf pi   po  fr   de sr  in   sy   cs   us sy id

21 0 0 3246608 22100 0 201 1600   0    0 0    0 1208 1257 1413 88 12  0
18 0 0 3246608 17196 0 370 1520   0  464 0  139  988  209 1155 91  8  0
11 0 0 3251116 16664 0 483 2064 138 2408 0  760 1315  218 1488 88 12  0
30 0 0 3251112 15764 0 475 2480 310 4450 0 1432 1498  199 1717 87 13  0
```

上記の出力は、さらに後のものです。出力の 3 行目は、システムが管理できる空きメモリ量のスレッショルドに達したことを示しています。この時点でページアウト (ページアウトされた KB を示す、*po* フィールド) が発生し、それに応じてシステム・モードの CPU レベル (*sy* フィールド) が増加しています。この状況が発生するのは、物理メモリが過度に割り付けらたためです。つまり、Sybase IQ バッファ・キャッシュのサイズが、このマシンには大きすぎるのです。この問題を解決するには、メインとテンポラリのバッファ・キャッシュのどちらか、または両方のサイズを減らします。

バッファ・キャッシュ・モニタリング・チェックリスト

次の表は、モニタリング結果で着目すべき最も一般的な項目の要約と、動作が正常な範囲を逸脱している場合に講じる必要がある対応策を示します。[統計] カラムには、標準のモニタ・レポートに表示される名前が示されています。デバッグ・レポートでは別の名前で表示される統計については、その名前も示されています。

どのモニタリング統計でも、新しいクエリが開始されたときなど、システムの状態が変化している間は一時的な異常が起きる可能性があることに注意してください。

表 6-1: バッファ・キャッシュ・モニタリング・チェックリスト

統計	正常な動作	調整が必要な動作	推奨される対応策
HR% (Cache hit rate)	90% 以上。 GARRAY、BARRAY、Bitmap (bm)、hash object、sort object、variable-length btree (btreev)、fixed-length btree (btreef)、bit vector (bv)、dbext、dbid、vdo、store、checkpoint block (ckpt) などの個別の内部データ構造体の場合、クエリの実行中はヒット率が 90% を上回る。最初は 90% を下回る場合がある。プリフェッチが機能し始めると (PF または PrefetchReqs > 0)、ヒット率が徐々に上昇して 90% を超える。	プリフェッチが機能した後もヒット率が 90% を下回る。 **注意** オブジェクトによってはプリフェッチを行わないものがあるため、これらのヒット率は一般に低くなる。	-iqmc と -iqtc を調整し、メインとテンポラリのキャッシュ・サイズのバランスをとり直してみる。 PREFETCH_THREADS_PERCENT オプションを調整し、プリフェッチ・スレッド数を増やしてみる。
Gdirty (Grabbed Dirty)	適度なキャッシュ・サイズ (< 10GB) が設定されたシステムでは 0。	GDirty > 0 **注意** スイーパ・スレッドがアクティブになるのは、ダーティ・ページの数がウォッシュ・エリアの一定の割合に達した場合だけである。GDirty/GrabbedDirty が 0 より大きく、I/O 率 (Writes) が低い場合は、単にシステムに軽い負荷がかかっていると考えられるため対応策は不要。	SWEEPER_THREADS_PERCENT オプション (デフォルトは 10%) または WASH_AREA_BUFFERS_PERCENT オプション (デフォルトは 20%) を調整し、ウォッシュ・エリアのサイズを増やす。
BWaits (Buffer Busy Waits)	0	> 0 の状態が持続し、複数のジョブが同じバッファで衝突していることを示している。	I/O 率 (Writes) が高い場合は、キャッシュのスラッシングが原因で Busy Waits が起きていると考えられる。キャッシュ・レポートでヒット率を調べて、メインとテンポラリのキャッシュのバランスをとり直す必要があるかどうかを確認する。 ほぼ同一の多数のクエリを同時にバッチ・ジョブで開始している場合は、開始時刻をずらしてみる。
LRU Waits (デバッグ・レポートでは LRUNum TimeOuts percentage)	20% 以下	> 20%。これは重大な競合問題が起きていることを示す。	オペレーティング・システムのパッチ・レベルやその他の環境設定を確認する。これはオペレーティング・システムの問題の可能性が高い。
IOWait (IONumWaits)	10% 以下	> 10%	ディスク・エラーや I/O リトライを調べる。

統計	正常な動作	調整が必要な動作	推奨される対応策
FLWait (FLMutexWaits)	20% 以下	> 20%	dbspace の設定を確認する。 データベース領域が不足しかかっていないか？ DISK_STRIPING が ON になっているか？ sp_iqcheckdb が 15% を超える断片化をレポートしていないか？
HTWait (BmapHTNum Waits) MemWts (MemNtimesWaited) (PFMgrCondVar Waits)	10% 以下	> 10%	Sybase 製品の保守契約を結んでいるサポート・センタに問い合わせる。
CPU time (デバッグ・レポートでは CPU Sys Seconds、CPU Total Seconds)	CPU Sys Seconds < 20%	CPU Sys Seconds > 20% CPU Total Seconds も低い使用率を示しており、システムがビジー状態になるだけのジョブがある場合は、キャッシュがスラッシングしているか、並列度が失われていると考えられる。	-iqgovern を調整し、実行できる同時クエリの合計数を減らす。 キャッシュ・レポートでヒット率と I/O 率を調べて、キャッシュがスラッシングしていないかどうかを確認する。また、cache_by_type (またはデバッグ) レポートでハッシュ・オブジェクトのヒット率を調べて、ハッシュ・オブジェクトがスラッシングしていないかどうかを確認する：ヒット率が <90% で I/O 率 (Writes) が高くないか？ 試行された並列処理をクエリ・プランで確認する。十分なスレッドが使用可能だったか？ システムに大量の CPU が搭載されているか？ マルチプレックス構成などの対策が必要な場合もある。
InUse% (Buffers in use)	起動時以外は 100% かそれに近い値	100% 未満	バッファ・キャッシュが大きすぎる可能性がある。 -iqmc と -iqtc を調整し、メインとテンポラリのキャッシュ・サイズのバランスをとり直してみる。

統計	正常な動作	調整が必要な動作	推奨される対応策
Pin% (Pinned buffers)	< 90%	> 90 ～ 95%。システムがバッファ不足状態に危険なほど近づいていることを示す。この状態になるとトランザクションがロールバックされる。	メインとテンポラリのキャッシュ・サイズのバランスをとり直してみる。 バッファ・キャッシュ・サイズのバランスをとり直すことができない場合は、-iqgovern を減らして、同時に実行されるジョブの数を制限してみる。
Free threads (ThrNumFree)	空き > 予約済み	空きスレッドの数が予約済みの数まで減少している場合は、システムのスレッドが不足していると考えられる。	次のいずれかを試してみる。 -iqmt を設定してスレッドの数を増やす。 スレッド関連の次のオプションの値を減らす：MAX_IQ_THREADS_PER_CONNECTION、MAX_IQ_THREADS_PER_TEAM、PARALLEL_GBH_UNITS (Group By ハッシュを使ったクエリ) USER_RESOURCE_RESERVATION を設定し、クエリ・エンジンのリソース割り付けを制限する。 -iqgovern を設定し、ジョブの数を制限する。
FlOutOfSpace (デバッグのみ)	0。このストアのフリー・リストが満杯でないことを示す。割り付けられていないページが使用可能	1。このストア (メインまたはテンポラリ) が全部割り付けられていることを示す。	ストアに dbspace を追加する

CPU 使用率をモニタリングするシステム・ユーティリティ

Sybase IQ の使用中に CPU 使用率をモニタリングするには、次のオペレーティング・システムのユーティリティを使用します。

UNIX の場合は、次のとおりです。

- ps コマンド
- vmstat コマンド (前項の例を参照)
- sar コマンド (UNIX SystemV の場合)

Windows の場合は、次のとおりです。

- システム モニタ
- タスク マネージャ

第 7 章

Windows システムでのサーバのチューニング

この章について

この章では、Windows システムで Sybase IQ を実行するときに特有のパフォーマンスとチューニングのガイドラインについて説明します。この章の説明は、「第 5 章 システム・リソースの管理」の説明と併せてお読みください。

内容

パフォーマンスについての一般的なガイドライン

この項では、データのロードとクエリの両方に適用される一般的なガイドラインについて説明します。Windows で Sybase IQ を実行するときに推奨される最小限のメモリ量 (RAM) は 512MB です。これより少ないメモリ構成でも Sybase IQ は正しく機能しますが、パフォーマンスが低下することがあります。

スループットの最大化

Windows で実行する場合は、ネットワーク・サービスのサーバのオプションで [ネットワーク アプリケーションのスループットを最大にする] が有効になっていることを確認してください。このオプションを有効にすると、メモリが制限される環境でも、NTFS キャッシュが Sybase IQ からメモリを横取りして過度のページ・フォールトが発生する事態を回避できます。

❖ **[ネットワーク アプリケーションのスループットを最大にする] を有効にするには**

1 コントロール パネルの [ネットワーク] アイコンをダブルクリックします。

2 [サービス] タブをクリックして、[サーバー] ネットワーク・サービスをダブルクリックします。

3 [ネットワーク アプリケーションのスループットを最大にする] オプションをクリックします。

4 [OK] を 2 回クリックし、マシンを再起動します。

メモリの割り付け超過の防止

マシンの物理メモリ (RAM) を過度に割り付けると、システムのページ・フォールトが頻繁に発生します。ページ・フォールトが頻繁に発生すると、Sybase IQ のパフォーマンスが著しく低下します。Sybase IQ のバッファを慎重に割り付け、Sybase IQ プロセスの仮想アドレス空間と使用可能な物理メモリをモニタリングすることにより、メモリの割り付けの超過を防ぐことができます。この項では、Sybase IQ によるマシンの物理メモリの使用状況をモニタリングするためのガイドラインについて説明します。

物理メモリのモニタリング

アプリケーション (Sybase IQ) が使用できる物理メモリ量は、[物理メモリ (KB)] の下に表示されます。[利用可能] の値が常に 5000 未満である場合は、マシンの物理メモリが過度に割り付けられている可能性があります。これは、値が 5000(KB) 台になると、最低限 5MB の空きメモリを維持するために Windows がページ・フォールトを発生させるからです。

物理メモリをモニタリングするには、[タスク マネージャ] アプレットの [パフォーマンス] タブをクリックします。

ファイル・システム

Windows ファイル・システムでは、ファイル、ディレクトリ、ボリューム・レベルでの圧縮がサポートされています。Sybase IQ データベースを格納するすべてのディスクとボリュームで、Windows ファイル・システムの圧縮を無効にしてください。これは、Sybase IQ が組み込みの圧縮を提供しているためです。ファイル・システムの圧縮を使用しても、それ以上データベースのサイズを縮小することはできず、読み込みや書き込みを実行するときに CPU のオーバヘッドが増えるだけです。

パフォーマンスのモニタリング

Sybase IQ のパフォーマンスをモニタリングするための主なツールは、Sybase IQ パフォーマンス・モニタです。このツールについては、「第 6 章 パフォーマンスのモニタリングとチューニング」に記載されています。さらに、オペレーティング・システムのモニタリング・ツールも使用できます。

Windows には、システムのパフォーマンスをモニタリングするツールが 2 つあります。

Windows タスク マネージャには 2 つのウィンドウがあり、現在のシステム・パフォーマンスの概要を簡単に読み取ることができます。タスク マネージャを起動するには、[Ctrl + Alt + Del] を押し、[タスク マネージャ] ボタンをクリックします。[プロセス] タブをクリックすると、システムで現在実行中のすべてのプロセスがリストされます。表示される列をカスタマイズするには、[表示] - [列の選択] をクリックします。[CPU 使用率] 列、[メモリ使用量] 列、[仮想メモリ サイズ] 列は、CPU やメモリにボトルネックがあるかどうかを判断するのに役立ちます。[パフォーマンス] タブでは、マシンのパフォーマンスのさまざまなカウンタと履歴を確認できます。

パフォーマンス モニタには、マシンのパフォーマンスのより詳細な分析が表示されます。プロセッサ、プロセス、ディスク、ネットワークなど、さまざまなシステムやアプリケーション・オブジェクトに関するカウンタを個別にモニタリングできます。

仮想アドレス空間とワーキング・セットのモニタリング

プロセスの仮想アドレス空間は、プロセスの合計サイズです。プロセスのワーキング・セットは、現在プロセスに割り付けられている物理メモリ量です。ほとんどの環境では、過度なシステムのページ・フォールト避けるために、Sybase IQ プロセスの仮想アドレス空間をマシンの物理メモリより小さくしてください。

Sybase IQ サーバ内での仮想メモリの使用パターンのせいで、Windows プラットフォーム上で仮想メモリの断片化によって処理が過度に増大する可能性があります。このような状況に陥る可能性を小さくするため、Sybase IQ では Windows XP と Windows Server 2003 について Microsoft の LFH (低断片化ヒープ) の使用をサポートしています。

❖ 仮想アドレス空間とワーキング・セットをモニタリングするには

1. パフォーマンス モニタを起動します。
2. [+] アイコンをクリックして、[Process] オブジェクトを選択します。
3. 最初の [Sybase IQ] インスタンスを選択します。
4. [Virtual Bytes] カウンタと [Working Set] カウンタを選択します。

ページ・フォールトのモニタリング

上記と同じように、Windows パフォーマンス モニタで [Sybase IQ] プロセスを選択します。[Page Faults/sec] カウンタを選択します。このカウンタには、「ソフト」ページ・フォールトと「ハード」ページ・フォールトの両方が含まれます。ハード・ページ・フォールトは、ディスク I/O を引き起こすページ・フォールトです。ソフト・ページ・フォールトは一般に、パフォーマンスの問題には関係しません。

ハード・ページ・フォールトの数値を調べるには、[LogicalDisk] オブジェクト、および *pagefile.sys* ファイルが格納されている場所 (このファイルは Sybase IQ データベースとは別のボリュームに格納してください) のインスタンスを選択します。[Disk Transfers/sec] カウンタを選択します。この値を [Page Faults/sec] の値と比較すると、ページ・フォールトに対するハード・ページ・フォールトの割合がわかります。ページ・ファイルへの I/O アクティビティはほとんどないことが理想です。ただし、メモリが少ない構成では、ページングが発生することが多くなります。

ハード・ページ・フォールト率が毎秒 20 を超える状態が持続する場合は、マシンの物理メモリが過度に割り付けられています。

NTFS キャッシュの使用

ネットワーク・サービスのサーバのオプションで [ネットワーク アプリケーションのスループットを最大にする] を有効にした状態で、NTFS とその関連のキャッシュを使用すると、挿入とクエリの両方で Sybase IQ のパフォーマンスが向上します。これは主に、物理メモリ量と、Intel Pentium の Sybase IQ ページの圧縮解除能力が同じでも、NTFS の方が Sybase IQ バッファ・キャッシュより大量のデータを格納できるからです。したがって、Windows プラットフォームで Sybase IQ を使用する場合は、Sybase IQ バッファ・キャッシュのサイズを推奨される標準設定より小さくしてください。

Sybase IQ メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュには、Sybase IQ データ (ページ) が非圧縮形式で格納されます。つまり、100MB の Sybase IQ バッファ・キャッシュには、100MB 分のデータを格納できます。これに対し、NTFS キャッシュでは Sybase IQ データを圧縮形式で管理します。このため、圧縮率が 2:1 の場合、100MB の NTFS キャッシュには 200MB の Sybase IQ データが格納されている可能性があります。したがって、NTFS キャッシュでは、キャッシュのヒット率をより高く維持して、I/O を減らすことが可能です。たとえ NTFS キャッシュから Sybase IQ バッファ・キャッシュに移動するデータを圧縮解除するために計算のオーバヘッドが生じても、I/O の削減効果の方が重要です。

挿入とクエリのチューニング

この項では、Windows プラットフォームでの挿入とクエリをチューニングするための追加のガイドラインについて説明します。

適切にチューニングされた挿入オペレーションの特性

適切にチューニングされた Sybase IQ の挿入オペレーションには、ある特性が見られます。これらの特性は、Windows タスク マネージャと Windows パフォーマンス モニタで確認できます。

- 挿入オペレーションの大部分は、CPU の能力に依存します。システム内のすべての CPU は、能力の 100% 近くを使用して実行されます。CPU の 95% 以上がユーザ・モードで実行されます。このことは、Windows タスク マネージャの [パフォーマンス] タブをクリックし、[表示] – [カーネル時間を表示する] オプションを設定すると簡単に確認できます。
- 物理メモリを過度に割り付けないように注意してください。特に、Sybase IQ プロセスの仮想アドレス空間は、マシンの物理メモリ (RAM) より小さくする必要があります。512MB から 2GB の大容量の物理メモリを備えたマシンでは、これは問題になりません。メモリ量が 256MB 未満のマシンでは、次の項の追加のガイドラインを参照してください。
- ハード・ページ・フォルト (*pagefile.sys* が格納されているボリュームへの I/O) を少なくし、理想的には 0 (ゼロ) に近づけてください。
- IQ ストアへの I/O 処理が安定して実行され、ディスク・サブシステムの I/O 処理能力を超えないようにします。

Sybase IQ では、ファイルを順次アクセス用に読み込むことを指定する Windows CreateFile オプションを (ファイルの作成時および開くときの両方に) 使用します。LOAD TABLE コマンドで指定したファイルには、このオプションが使用されます。これにより、先読みが行われ、NTFS キャッシュ・メモリの使用量が減少するため、ロードのパフォーマンスが向上します。

ロードのパフォーマンスは、さらに向上する可能性があります。場合によっては大幅に向上します。それには、Sybase IQ メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュのサイズを「Sybase IQ のメイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュ」(111 ページ) で推奨されている計算上の値よりもかなり小さく設定します。これでパフォーマンスが向上する理由は、「NTFS キャッシュの使用」(180 ページ) に記載されています。Sybase IQ メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュを、計算上の推奨値より最大 50% 小さく設定できます。

クエリのチューニング

クエリのパフォーマンスも向上する可能性があります。それには、上述のように、メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュのサイズを小さく設定します。ただし、マルチユーザ環境では注意が必要です。テンポラリ・バッファ・キャッシュのサイズを小さくすると、複数ページの位置決めとソート・アルゴリズムに影響し、パフォーマンスの低下を招くことがあります。クエリ・プラン、構造、オプションの詳細については、「第 3 章 クエリと削除の最適化」を参照してください。

バックアップ操作のチューニング

Windows では、固定長の I/O デバイスのみがサポートされています。このため、テープの読み込みや書き込みは、その前後の読み込みや書き込みと同じサイズで行う必要があります。読み込みや書き込みの操作がハードウェア・デバイスの容量を超えた場合、操作は失敗します。したがって、バックアップやリストアの操作では、ハードウェアが設定されているサイズですべての書き込み (または読み込み) を行わないと、バックアップ (またはリストア) が失敗します。

Sybase IQ では、どのプラットフォームでもできるだけ効率的に読み込みと書き込みの操作が行われるようにデフォルトが設定されています。ただし、Sybase IQ データベースの作成時にデフォルトのブロック・サイズを上書きしている場合は、そのデータベースをバックアップするときにブロック係数を調整する必要があります。

バックアップまたはリストアでは、次の式が適用されます。

```
block size x block factor ≈ I/O size
```

Windows システムでブロック係数を調整するには、使用中のテープ・デバイスで処理できる物理ブロックの最大サイズの情報が必要です。通常、ドライブの製造元のマニュアルには、この情報は記載されていません。値 (通常は 64KB) を調べるには、WIN32 API 呼び出しを使って小さなアプレットを作成する必要があります。そして、データベースのブロック・サイズと BACKUP コマンドの BLOCK FACTOR オプションを使用して、バックアップのパフォーマンスを最適化します。完全な構文と使用方法については、『Sybase IQ リファレンス・マニュアル』を参照してください。

各 I/O 操作が最大ブロック・サイズに近づくほど、バックアップのパフォーマンスが向上します。BLOCK FACTOR の整数値にブロック・サイズを乗算したときに、ドライブのブロック・サイズにできるだけ近い値が得られるようにします。

データの整合性を保つために、Sybase IQ は書き込んだ各ブロックに余分なデータを追加することに注意してください。このため、データベースのブロック・サイズが 8192 で、テープ・デバイスで処理できる最大ブロック・サイズが 128KB の場合、本来なら 8192 * 16 = 128KB となるはずですが、ブロック係数に 16 は使用できません。各 I/O 操作で Sybase IQ が追加する余分なデータを計算に入れ、BLOCK FACTOR に 15 を使用する必要があります。15 という値は、Windows でデフォルトのデータベース・ブロック・サイズとデフォルトの IQ ページ・サイズ (128KB) を使用したときのデフォルトのブロック係数です。

索引

J

L

M

N

O

P

Q

R

S

T

U

V

W

あ

い

う

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

ね

は

ひ

ふ

へ

ほ

ま

む

め

も

ゆ

よ

ら

り

れ

ろ

わ