DB2 Express-C 入門

V9.7 対応

コミュニティーによるコミュニティーのための学習書

RAUL CHONG, IAN HAKES, RAV AHUJA

FOREWORD BY DR. ARVIND KRISHNA

日本アイ・ビー・エム株式会社　訳・監訳

THIRD EDITION

Third Edition (June 2009)

This edition has been updated for IBM® DB2® Express-C Version 9.7 for Linux®, UNIX® and Windows®.

目次

本書について

注意事項および商標

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

IBM は、ここに挙げられた内容に関連する何等の保証責任を負いません。また特に、特定の目的のための商品性または適合性に関してはいかなる黙示的な保証責任も負わないものとします。IBM は、本書における誤りに対しては責任を負いません。本書の情報は、予告なしに変更される場合があります。IBM は改訂や変更について何人にも通知する義務なしに変更する権利を有するものとします。IBM は今現在ここに含まれている情報を保持し続けることを確約するものではありません。

本書における IBM 以外の製品に関する情報は、その製品の供給者から入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する性能、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml の「Copyright and trademark information」をご覧ください。

Java およびすべての Java 関連の商標は、Sun Microsystems, Inc. の米国およびその他の国における商標です。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

本書に記載の IBM 製品、またはサービスが日本においては提供されていない場合があります。日本で利用可能な製品、プログラム、またはサービスについては、日本 IBM の営業担当員にお尋ねください。

本書の対象読者

本書は、データベース管理者 (以下、DBA)、アプリケーション開発者、コンサルタント、ソフトウェア・アーキテクト、プロダクト・マネージャー、インストラクター、および学生など、データベースを現在操作、あるいはこれから操作することを予定している読者を対象に作成されています。

本書の構成内容

「パート I - 概要とセットアップ」では、DB2 Express-C エディションの概要ならびに DB2 製品ファミリーとその機能を紹介します。また、データベースのインストールおよび作成手順と、DB2 で使用できるツールについて説明します。

「パート II - DB2 の学習: データベース管理」では、DB2 の環境、アーキテクチャー、リモート接続、データベース・オブジェクト、データの移動 (インポート/エクスポート/ロード)、セキュリティー、バックアップとリカバリー、並行性とロック、その他一般的な管理作業について説明します。

「パート III - DB2 の学習: アプリケーション開発」では、サーバー･サイドおよびクライアント･サイドの開発を含む DB2 アプリケーションの開発について説明します。また、SQL/XML、XQuery、そして pureXML® についても説明します。

「付録」には、トラブルシューティングに役立つ情報が記載されています。

ほとんどの章には、演習が用意されています。これらの演習に必要な入力ファイルは本書に付属の **`expressc_book_exercises_9.7.zip`** という圧縮ファイルに含まれています。

この第 3 版では内容を変更および追加しています。そのため、DB2 9.5 を対象とした第 2 版を読んだ読者が、DB2 バージョン 9.7 で新しく追加されたフィーチャーまたは更新内容についてわかりやすいようにしてあります。変更箇所は以下のアイコンで示されているので、簡単に見分けることができます。

コミュニティーのためのガイド

本書は DB2 Express-C チームによって作成され、DB2 Express-C コミュニティーに無償で公開されています。DB2 9.7 対応版を執筆時点で、本書のダウンロード数は 45,000 を超え、世界中のボランティアによって 9 つの言語に翻訳されています。まさにコミュニティーによる作業の賜物です。本書に関するフィードバック、新しい教材の提供、既存の教材の改善、または他の言語への翻訳でご協力いただける場合は、「DB2 Express-C book changes」という件名で、その内容を明記した E メールを db2x@ca.ibm.com に送ってください。

本書の成功に刺激を受け、25 を超える無料のオンライン・ブックが新たに作成されています。IBM 製品だけでなく、IBM 以外の技術を取り上げるこれらのオンライン・ブックは、2009 年 10 月にリリースされる **Community Book Series** に含まれる予定です。

オンライン・ブックまたは Community Book Series についての詳細を調べるには、IBM® DB2 Express-C の Web サイト、www.ibm.com/db2/express にアクセスしてください。

著者および貢献者

以下は、本書の内容およびその他の分野で多大な貢献を果たしたメンバーです。

Raul F. Chong – 主執筆者

Raul は、IBM Toronto Lab の DB2 on Campus プログラム・マネージャーです。

Ian Hakes – 共著者兼編集者

Ian は、以前は DB2 Express-C コミュニティーのファシリテーターをしていましたが、現在はユーザビリティーのエキスパートとして IBM Toronto Lab に勤務しています。

Rav S. Ahuja – 共著者兼発行者

Rav は、IBM Toronto Lab の DB2 シニア・プロダクト・マネージャーです。

謝辞

本書で使用している教材の開発に協力し、支援してくださった以下の方々に深く感謝します (敬称略)。

- IBM Toronto Lab の Ted Wasserman、Clara Liu、Paul Yip は、本書のフレームワークとなった教材を開発してくれました。
- Don Chamberlin と Cindy Saracco は XQuery について彼らが著した IBM developerWorks の記事で、Matthias Nicola は pureXML のプレゼンテーションで協力してくれました。
- Kevin Czap と Grant Hutchison は、DB2 テクニカル・ブリーフィング資料を作成してくれました。
- Katherine Boyachok と Grant Hutchison は本書のカバーをデザインしてくれました。
- Susan Visser は本書のレビューをしてくれるとともに、本書の発行を支援してくれました。

序文

技術の進歩に、技術革新は欠かせません。IBM での今までのデータ・サーバーの進化にも、常に技術革新が伴ってきました。1960 年代と 70 年代にかけてデータ管理技術の先駆者として活躍した IBM は、その技術者が考案した数千ものデータ管理に関する特許に反映されているように、革新的な情報管理技術を絶えず生み出し続けてきました。その結果、世界屈指の規模を誇る組織では、その極めて要求の厳しいミッション・クリティカルなデータ管理ソリューションの駆動力として DB2 をはじめとする IBM 製品に依存するようになっています。

そんな DB2 が、いまや大企業だけのものではなくなりました。DB2 Express-C が公開されたことにより、中小企業でも、受賞の栄誉に輝いた DB2 技術を利用できる (しかも無料で) ようになっています。他にも無料またはオープンソースのデータ・サーバーは出回っていますが、DB2 Express-C にはそのどれにも勝る独特のメリットがあります。

DB2 Express-C には数多くの技術革新が息づいています。これらの技術革新によって、新しい機能がもたらされ、管理作業の負担の軽減やパフォーマンスの改善、そしてインフラストラクチャーのコストの削減が実現されます。

DB2 Express-C のハイブリッド技術は、リレーショナル・データと XML データの両方をそれぞれのネイティブ・フォーマットで管理できるようにします。そのため、XML データが大量に受け渡しされる SOA や Web 2.0 といった新しい世代のアプリケーションを強化するには、DB2 はまさに理想的なデータ・サーバーです。他の「無料の」データ・サーバーとは違い、DB2 Express-C では、データベースに保管できるデータの量にも、システム上で作成できるデータベースの数にも制限はありません。そして当然のことながら、IBM のサポートまたは支援が必要な場合にはクリック 1 つでサポートが得られます。

本書は DB2 Express-C の入門ガイドとして、DB2 の概念を理解する手助けをし、読者の皆さんが DB2 の管理と DB2 アプリケーション開発のスキルを磨けるようにすることを目的としています。本書によって得られるスキルと知識は、DB2 for Linux, UNIX, and Windows の他の拡張エディションにも関連しています。

DB2 Express-C はオープンソースの製品ではありませんが、IBM では確固たる信念を持ってコミュニティー・イニシアチブのサポートおよび助成に取り組んでいます。嬉しいことに、本書は DB2 Express-C コミュニティーのメンバーによって作成され、コミュニティーの誰もが無料で入手できるようになっています。本書をさらに充実させるために、皆さんにもぜひ、ご自身のノウハウと経験で本書の情報を更新してくださること、そして別の言語に翻訳して他の人々が活用できるようにご協力くださることをお願いします。

Arvind Krishna
General Manager
Information Management, IBM Software Group

パート I – 概要およびセットアップ

1

第 1 章 – DB2 Express-C 概要

DB2 Express-C データ･サーバー･ソフトウェア (「DB2 Express-C」) は、リレーショナル・データと XML データの両方を管理する強力なデータ・サーバー・ソフトウェア、IBM DB2 ファミリーのメンバーです。無料で制限なく使用できる DB2 Express-C は、手軽に使用できる DB2 のエディションとなっています。DB2 Express-C の「C」はコミュニティーの略で、オンライン、オフラインともにユーザーが一団となって互いに協力しあう DB2 Express-C ユーザーのコミュニティーを指します。DB2 Express-C コミュニティーを構成するのは、データベース・ソリューションを設計、開発、デプロイ、または使用するあらゆる職種の人々と企業です。例えば、コミュニティーには以下のメンバーが含まれます。

- スタンドアロン・アプリケーション、クライアント・サーバー・アプリケーション、Web ベースのアプリケーション、そしてエンタープライズ・アプリケーションを構築するためのオープン・スタンダードのデータベース・ソフトウェアを必要とするアプリケーション開発者
- フル機能のデータ・サーバーをソリューションにバンドルしたり、またはその一部として組み込む必要のある、ISV、ハードウェア・ベンダー、インフラストラクチャー・スタック・ベンダー、およびその他の類のソリューション・プロバイダー
- トレーニング、スキル開発、評価、およびプロトタイピングに適した堅牢なデータ・サーバーを必要とするコンサルタント、データベース管理者、および IT アーキテクト
- アプリケーションおよび業務用に信頼性の高いデータ・サーバーが必要な新興企業や中小企業
- Web 2.0 および次世代アプリケーションを構築する際に手軽に使用できるデータ・サーバーを求めているデータベース愛好家および最先端技術のマニア
- 教育、コースウェア、プロジェクト、研究に使用できる、極めて用途の広いデータ・サーバーを必要とする学生、教師、その他学術関連のユーザー

DB2 Express-C のコア機能とコード・ベースは、DB2 for Linux, UNIX, and Windows の有料エディションと同じです。DB2 Express-C は Linux あるいは Widows オペレーティング・システムを使用した 32bit または 64bit システムのどちらでも実行することができます。さらに、Solaris (x64) 上でも実行することができ、Mac OS X (x64) の場合はベータ版が利用可能です。使用するシステムのプロセッサー数やメモリー容量についての前提条件はありません。また、特殊なストレージやシステムのセットアップ要件もありません。DB2 Express-C には、無料で pureXML も組み込まれています。pureXML は、XML 文書をネイティブに保管し、処理する DB2 ならではの独自の技術です。

1.1 開発、デプロイメント、配布は無料、しかも制限はありません！

この一文に、DB2 Express-C の背後にある以下の重要な理念が要約されています。

- **無料で開発できます**: 開発中のアプリケーションにデータベースが必要なアプリケーション開発者は、DB2 Express-C を使用できます。
- **無料でデプロイメントできます**: 本番環境での作業で、重要なレコードを保管するデータベース管理システムが必要な場合には DB2 Express-C を使用できます。
- **無料で配布できます**: 開発中のアプリケーションやツールに組み込みデータ・サーバーが必要な場合に、DB2 Express-C を使用できます。アプリケーションに DB2 Express-C を組み込んで販売したとしても、アプリケーションに組み込まれた DB2 Express-C に料金が発生することはありません。DB2 Express-C を再配布するには IBM への登録が必要になりますが、登録にも費用はかかりません。
- **制限はありません**: 競合他社のデータベース・オファリングではデータベースのサイズ、データベースの数、ユーザーの数に制限が設けられていますが、DB2 Express-C の場合、何の制限もありません。ライセンス契約に違反することなく、思いのままにデータベースを拡張できます。さらに、接続数やサーバーごとのユーザー数に関する制限もライセンスで規定されることはありません。

1.2 ユーザー支援および技術サポート

DB2 Express-C に関する技術的な質問は、DB2 Express-C フォーラムに投稿することができます。この無料のフォーラムは IBM の DB2 エキスパートがモニターしていますが、ほとんどの回答は、コミュニティーが自主的に提供するものです。

IBM では低価格のDB2 Express データ･サーバー･ソフトウェア (「DB2 Express-C」) 年間サブスクリプション (FTL (Fixed Term License) としても知られています) を購入するという選択肢も用意しています。このサブスクリプションにより、技術サポートとソフトウェア更新が提供されます。サポートとソフトウェア保守に加え、この低価格の年間サブスクリプション (米国ではサーバー 1 台あたりの年間費用は $2,995 ですが、この価格は国によって異なります) を購入すると追加フィーチャーも使用できるようになります。追加フィーチャーには、HADR (High Availability and Disaster Recovery を目的としたクラスタリング)、SQL レプリケーション (他の DB2 サーバーにデータを複製する機能)、Backup Compression (データベースの圧縮バックアップ･コピーを作成する機能) があります。このサブスクリプションについての詳細は、以下のリンクにアクセスしてください。
www.ibm.com/db2/express/support.html

1.3 DB2 サーバー

すべての DB2 サーバー・エディションには、同じコア･コンポーネントが含まれます。これらのコア・コンポーネントは、ユーザーが必要な機能を価格に応じて選択できるようにパッケージ化されています。図 1.1 に、DB2 製品のエディション間の関係を示します。

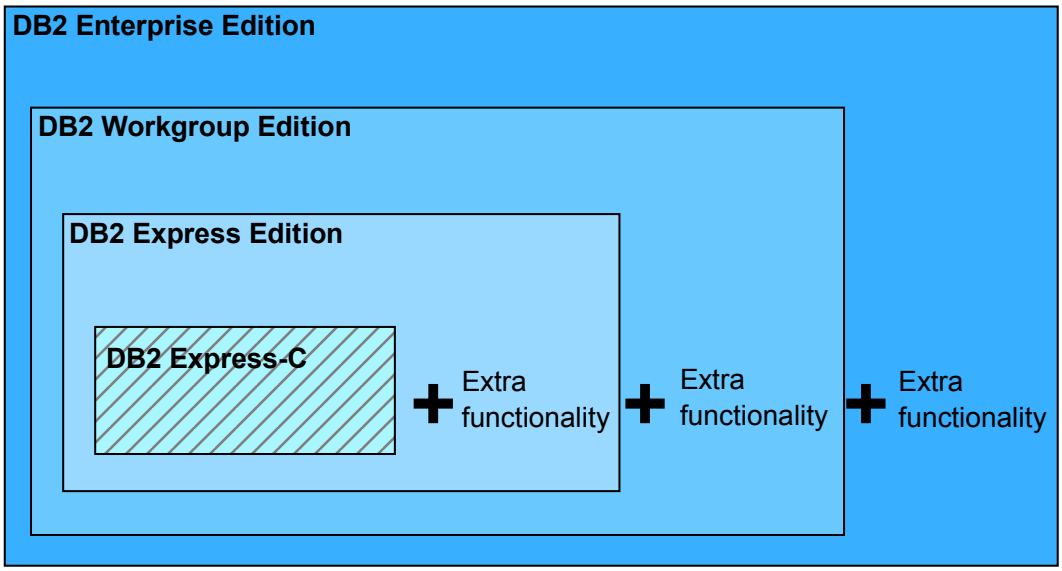


図 1.1 – DB2 サーバー

図 1.1 に示したように、DB2 Express-C はいくつかのコンポーネントが省かれているという点を除けば、DB2 Express と同じです。DB2 Express-C はコミュニティーに開放されています。無料のオンライン・フォーラムを利用すると技術支援を受けることができます。あるいは年間サブスクリプション (FTL) を購入すると、公式 IBM DB2 技術サポートを受けることができます。

図 1.1 から、DB2 Express-C のアップグレードが簡単である理由も明白です。将来、他の DB2 サーバーのいずれかにアップグレードすることを計画している場合、すべての DB2 サーバーのコア・コンポーネントは同じなので、簡単にアップグレードすることができます。このことはまた、いずれかのエディションを対象に開発したアプリケーションは、他のエディションでも変更を加えずに動作するということを意味します。さらに、あるエディションで学んだスキルは他のエディションにも適用できるということでもあります。

1.4 DB2 クライアントおよびドライバー

DB2 クライアントには、DB2 サーバーに接続するために必要な機能が組み込まれています。ただし、DB2 クライアントは必ずインストールしなければならないというわけではありません。例えば、JDBC Type 4 アプリケーションの場合、JDBC ドライバーがインストールされてさえいれば、DB2 サーバーに直接接続することができます。DB2 クライアントおよびドライバーには、何種類かのフレーバーが用意されています。

- IBM Data Server Client: GUI ツール、ドライバーが組み込まれた最も完全なクライアントです。
- IBM Data Server Runtime Client: 基本的な機能を備えた軽量のクライアントで、ドライバーが組み込まれています。
- DB2 Runtime Client Merge Modules for Windows: 主に、Windows アプリケーション・システムに DB2 ランタイム・クライアントを統合する場合に使用します。

- IBM Data Server Driver for JDBC and SQLJ: 完全なクライアントをインストールしなくても、Java アプリケーションが DB2 サーバーに接続できるようにします。
- IBM Data Server Driver for ODBC and CLI: ODBC および CLI アプリケーションから DB2 サーバーに接続できるようにします。クライアントをインストールする必要がない分、フットプリントを大幅に節約することができます。

new in V9.7

- IBM Data Server Driver Package: ODBC、CLI、およびオープンソースのプロジェクトに加え、.NET 環境もサポートする Windows 専用のドライバーです。このドライバーは、以前は IBM Data Server Driver for ODBC, CLI and .NET という名前で提供していました。

図 1.2 に、利用可能な DB2 クライアントとドライバーを示します。

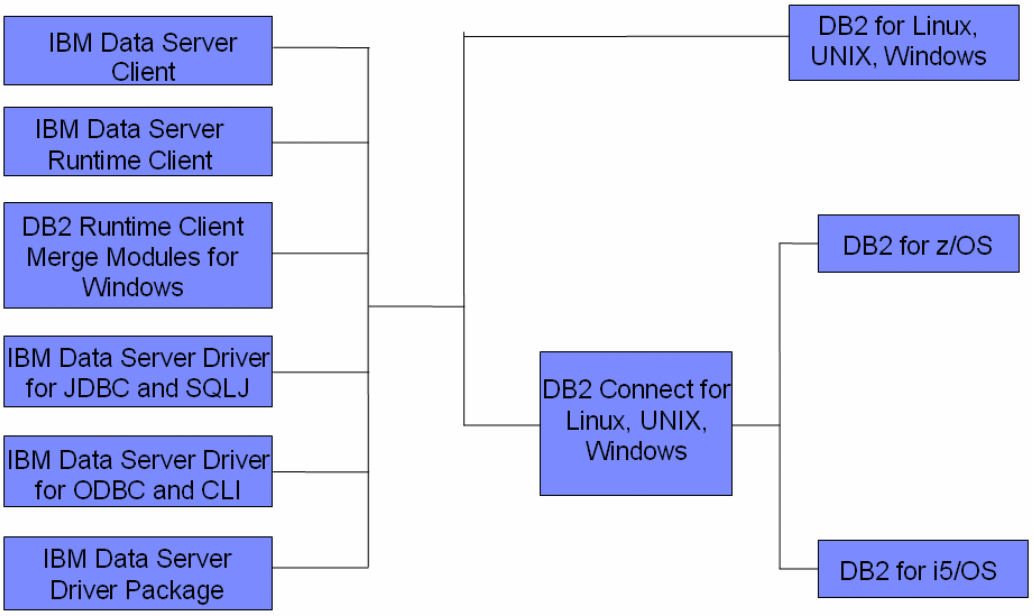


図 1.2 - DB2 クライアントおよびドライバー

図 1.2 の左側には、すべての DB2 クライアントおよびドライバーが示してあります。どの DB2 クライアントにも必要なドライバーは組み込まれていますが、DB2 データ・サーバー・ソフトウェア (「DB2」) v.9 以降では、ドライバーを単独でも提供するようになっています。DB2 クライアントおよびドライバーはいずれも無料で提供されており、入手するには DB2 Express-C Web サイトからダウンロードします。Linux、UNIX または Windows 上の DB2 サーバーには、これらのクライアントとドライバーを使用して直接接続することができますが、DB2 for z/OS® サーバーあるいは DB2 for i5/OS® サーバーに接続するには DB2 Connect™ サーバーを経由する必要があります。DB2 Connect ソフトウェア (「DB2 Connect」) については第 2 章で説明します。

注:

本書で重点を置いているのは DB2 データ・サーバーですが、IBM Data Server Client は IBM ファミリーの他のデータ・サーバー (Informix など) にも接続することができます。したがって、具体的に「DB2 クライアント」とするのではなく、「IBM Data Server Client」という総称名を使用します。

1.5 自由自在なアプリケーション開発

DB2 が提供するアプリケーション開発環境は、標準をベースにした、DB2 ファミリー全体でトランスペアレントな環境になっています。DB2 製品ライン全体での SQL 標準化により、データベースのアクセスには、一連の共通したアプリケーション・プログラミング・インターフェースを使用することができます。

その上、DB2 製品のそれぞれには SQL プリコンパイラーとアプリケーション・プログラミング・インターフェース (API) が用意されているため、開発者は移植可能なアプリケーション・プログラムに静的 SQL と動的 SQL を組み込むことができます。さらに DB2 にはネイティブ .NET 管理プロバイダーがあり、Microsoft® Visual ツールが統合されています。

DB2 では、以下の言語と標準を使用できます。

- SQL, XQuery, XPath
- C/C++ (CLI、ODBC、および組み込み SQL)
- Java (JDBC および SQLJ)
- COBOL
- PHP
- Perl
- Python
- Ruby on Rails
- .NET 言語
- OLE-DB
- ADO
- MS Office: Excel, Access, Word
- Web サービス

1.6 DB2 バージョンと DB2 エディションとの違い

DB2 に馴染みのない方にとって、DB2 バージョンと DB2 エディションとの違いは多少分かりにくいかもしれません。

IBM では数年ごとに新しい DB2 バージョンを正式に発表しています。新しいバージョンには、製品に追加された新しいフィーチャーや重要な改善が含まれます。現在、IBM が公式にサポートしているのは DB2 バージョン 9 です。1 つのバージョンには、いくつかのリリースがある場合もあります。リリースは新しい機能を含んだ更新であることがありますが、一般に、これらの機能には新しいバージョンを発表するほどの重要性はありません。例えば DB2 バージョン 9 の場合、9.5 と 9.7 がリリース・レベルです。過去数年を振り返ると、IBM は 1、2 年ごとに DB2 の新規リリースを発表しているようですが、新しいバージョンの発表となると通常は 3 年以上の間隔が空いています。最新のリリースである V9.7 が一般出荷可能 (GA) レベルになったのは、2009 年の 6 月です。さらに、

各リリースにいくつかのモディフィケーション・レベルがある場合もあります。通常、モディフィケーション・レベルには修正が含まれていたり、あるいはモディフィケーション・レベル自体が fix pack レベルに相当するものであったりしますが、新しい機能が含まれることはほとんどありません。本書を作成している時点での DB2 Express-C の最新のバージョン、リリース、モディフィケーション (V,R,M) レベルは 9.7.0 です。つまり、コード・レベルは 9.7 で fix pack は 0 なので、GA レベルであることを意味します。

一方、エディションとは選択オファリング、または各バージョンに含まれるパッケージ・グループのことです。前述のとおり、エディションには特定の価格とライセンスに応じて、異なる機能がパッケージ化されます。DB2 Version 9.7 (別名、DB2 9.7) には複数のエディションがあります。例えば、DB2 Express-C 9.7、DB2 Express 9.7、DB2 Workgroup 9.7、DB2 Enterprise 9.7 はすべて DB2 9.7 のエディションです (図 1.1 参照)。

1.7 別の DB2 エディションへのアップグレード

お使いのデータベースの拡張が必要になるにつれ、さらに大規模なハードウェア構成をサポートする DB2 エディションにアップグレードしていかなければならない場合があります。このような場合には、別の DB2 エディションに簡単にアップグレードすることができます。

- 同じコンピューター・システムで別の DB2 エディションにアップグレードするには、DB2 Ex-press-C をアンインストールしてから新しい DB2 エディションをインストールしてください。これによってデータベースが削除されることはありません (ただし、必ずバックアップしておくことをお薦めします)。
- DB2 をアップグレードして、同じオペレーティング・システムを使用した別の容量の大きいコンピューターに新しいエディションをインストールする場合には、新しい DB2 エディションをその別のコンピューターにインストールして、元のコンピューターのデータベースをバックアップします。そして、そのバックアップ・イメージを新しい DB2 エディションがインストールされたコンピューターに移動して、そのコンピューターのデータベースにリストアします。また、元のコンピューターのインスタンス構成設定 (dbm cfg) を保存して、この構成を新しく使うコンピューターに適用する必要もあります。バックアップおよびリストア・コマンドについての詳細は、第 11 章「バックアップおよびリカバリー」で説明しています。dbm cfg についての詳細は、第 5 章「DB2 環境」を参照してください。
- いずれの場合にしても、クライアント・アプリケーションを変更する必要はありません。

1.8 DB2 Express-C の保守および更新

DB2 Express-C インストール・イメージは定期的に更新されます。通常、更新が行われるのは、新しいリリースまたはバージョンが利用可能になる時点、または製品に加えられたバグ修正がかなりの数にまで蓄積された時点です。これまでのところ、DB2 Express-C のインストール・イメージは 1 年に 1 回更新されるのが通常となっています。ただし、DB2 Express-C は保証されていない無料のオファリングであるため、正式な保守リリースや (年に数回リリースされる) 定期的な fix pack は提供されないことに注意してください。新しい更新、または DB2 Express-C の新規リリースが使用可能になると、それ以前のリリースの DB2 Express-C は保守の対象外となります。

前述したように、セキュリティー・パッチと定期的にスケジュールされたソフトウェア更新、あるいはバグ修正が含まれる fix pack を使用できるように、IBM では DB2 Express 年間サブスクリプション・ライセンス (FTL) を用意しています。このサブスクリプションを購入すると、サブスクリプション・ライセンスの有効期間中、FTL のライセンス・キーによって DB2 技術サポート、ならびに更新および fix pack を利用する資格が与えられ、DB2 Express-C システムを更新できるようになります。また、サブスクリプション・ライセンスには無料のバージョン・アップグレードの利用資格も含まれます。特定のバージョンやリリースを使用し続けたいという場合でも、年間サブスクリプションの有効期間中であれば、そのリリースがサポートされている限り、fix pack とセキュリティー・パッチを適用することができます。

1.9 無料の関連ソフトウェアと DB2 コンポーネント

DB2 Express-C ダウンロード・ページ (www.ibm.com/db2/express/download.html) からダウンロードできるソフトウェアはすべて無料です。DB2 Express-C ソフトウェアの他、以下の便利なソフトウェア・パッケージを無料でダウンロードして使用することができます。

- Visual Studio アドイン
- DB2 Spatial Extender

さらに IBM alphaWorks Web サイト (www.alphaworks.ibm.com/datamgmt) からダウンロードできる以下の DB2 Express-C ベースのスターター・ツールキットも役に立つ場合があります。

- Starter Toolkit for DB2 on Rails (www.alphaworks.ibm.com/tech/db2onrails/)
- Web 2.0 Starter Toolkit for DB2 (www.alphaworks.ibm.com/tech/web2db2)

無料の軽量アプリケーション・サーバーをお探しの方のために、IBM では以下のサーバーを用意しています。

- WebSphere® Application Server – Community Edition (WAS CE)

1.9.1 IBM Data Studio

IBM Data Studio は、データベースを管理するための Eclipse ベースのツールで、XQuery、SQL スクリプト、ユーザー定義関数、およびストアード・プロシージャーの開発を支援します。このツールにはデバッガーが統合されています。さらに IBM Data Studio では、全体的な表の相関関係を理解できるように、物理データ・モデリング (PDM = Physical Data Modeling) ダイアグラムで作業できるようになっています。また、プログラミングを行わずに、ドラッグ・アンド・ドロップ方式でデータを Web サービスとして開発、パブリッシュすることもできます。IBM Data Studio は、現在使用が推奨されない「コントロール・センター」や「コマンド・エディター」などの DB2 ツールに置き換わるものです (「コントロール・センター」、「コマンド・エディター」は DB2 には組み込まれているものの、現在開発は行われていません)。IBM Data Studio については、第 5 章「DB2 ツール」で詳しく説明します。

1.9.2 DB2 Text Search

DB2 Text Search は、オプションで DB2 に統合できるコンポーネントです。IBM の OmniFind™ 技術をベースとするこのコンポーネントは、DB2 にそのままの形式で保管された XML 文書を含め、テキスト文書を対象に、強力で高速かつ詳細な全文検索を実行することができます。このコンポーネントは言語処理を使用してテキスト内でさまざまな形の検索語を見つけ出します。例えば、「study」という単語を検索すると、DB2 Text Search は「studies」や「studied」などの他の形でもこの単語を探します。

DB2 Text Search コンポーネントをインストールするには、DB2 Express-C のカスタム・インストールを選択し、サーバー・サポート・カテゴリーで DB2 Text Search Feature を選択してください。

注:

Net Search Extender (NSE) という DB2 エクステンダーでも、同様の機能を使用できますが、NSE は廃止が予定されています。DB2 Text Search は、NSE に置き代わるコンポーネントです。

1.9.3 WebSphere Application Server – Community Edition

IBM WebSphere Application Server - Community Edition (WASCE) は無料で入手できる軽量の Java EE 5 アプリケーション・サーバーです。Apache Geronimo 技術をベースとしたこのサーバーは、オープンソース・コミュニティーの最新の革新技術を利用して、Java アプリケーションの開発およびデプロイメントの基盤を提供します。その基盤は、1 つに統合されていて、利用しやすく、柔軟性の高いものになっています。WASCE ではオプションで、年間サブスクリプションによる技術サポートを利用することができます。

1.10 まとめ

DB2 Express-C エディションは、極めて優れた製品を無料で提供します。このエディションにはデータベースのサイズに関する制限がなく、データベースを自由に開発、デプロイ、配布できるだけでなく、他の DB2 エディションと同じコア機能および pureXML 技術が備わっています。広範なクライアント、ドライバー、開発言語をサポートする DB2 Express-C は、他の DB2 エディションにも簡単にアップグレードすることができます。

2

第 2 章 – 関連フィーチャーおよび製品

この章では、DB2 Express の年間サブスクリプション・ライセンス (FTL (Fixed Term License)) を購入すると提供される DB2 のフィーチャーについて説明します。また、(条件によっては) 追加料金で使用できる、他の DB2 エディションに含まれるフィーチャーについても説明します。

表 2.1 に、無料 (保証なし) の DB2 Express-C と、年間サブスクリプション・オプションを購入した場合の DB2 Express との違いを示します。

フィーチャー	無料 (保証なし)	有料サブスクリプション* (FTL)
DB2 のコア機能	あり	あり
無料の管理ツール	あり	あり
無料の開発ツール	あり	あり
オートノミック機能	あり	あり
pureXML Feature	あり	あり
コミュニティーによる無料支援***	あり	あり
IBM 公式サポート	なし	あり
fix pack	なし	あり
High Availability (HADR)	なし	あり
SQL Data Replication	なし	あり
Backup Compression	なし	あり
プロセッサー最大使用数	コア 2 基	コア 4 基 (最大 2 つのソケット)
メモリー最大使用量	2GB	4GB
使用可能な更新	新規リリース時に完全更新。通常、1 年に 1 回	年に数回のセキュリティー・パッチおよび fix pack
以前のバージョン/リリースのインストール・イメージ	使用不可。最新リリースおよびベータ版イメージのみ使用可能	使用可。IBM Passport Advantage を使用
サーバーの年間単価**	0	2,995 米ドル

表 2.1: 無料の DB2 Express-C と有料サブスクリプション (FTL) との比較

* サブスクリプションによって利用資格が与えられるフィーチャーは、サブスクリプションの有効期間中にのみ使用することができます。

** サブスクリプションの料金は米国での料金であり、予告なしに変更される場合があります。米国以外の国では、料金設定が異なる場合もあります。

*** コミュニティーによる無料支援は、オンライン・フォーラムによって提供されます。

表 2.2 に、製品のフィーチャーと、そのフィーチャーがどの DB2 9.7 エディションに組み込まれているのかを記載します。別途購入可能なフィーチャーは、該当する DB2 エディションでのフィーチャー名で記載してあり、背景を灰色にしてあります。

フィーチャー	DB2 Express サブスクリプション (FTL)	DB2 Express Edition	DB2 Workgroup Server Edition (WSE)	DB2 Enterprise Server Edition (ESE)
Homogenous SQL Replication	あり	あり	あり	あり
Homogenous Federation	あり	あり	あり	あり
Net Search Extender、DB2 Text Search	あり	あり	あり	あり
Spatial Extender	あり	あり	あり	あり
Backup Compression	あり	あり	あり	あり
pureXML	あり	あり	あり	あり
High Availability Disaster Recovery	あり	High Availability Feature	あり	あり
Tivoli® System Automation	あり		あり	あり
Advanced Copy Services	なし		あり	あり
Online Reorganization	なし		あり	あり
MQT	なし	なし	なし	あり
MDC	なし	なし	なし	あり
Query Parallelism	なし	なし	なし	あり

Connection Concentrator	なし	なし	なし	あり
Table partitioning	なし	なし	なし	あり
Governor	なし	なし	なし	あり
Compression: row level, index, XML, temp table	なし	なし	なし	Storage Optimization Feature
Label Based Access Control (LBAC)	なし	なし	なし	Advanced Access Control Feature
Geodetic Extender	なし	なし	なし	Geodetic Data Management Feature
Query Patroller	なし	なし	なし	Performance Optimization Feature
DB2 Workload Management	なし	なし	なし	
Performance Expert	なし	なし	なし	
Homogenous Q Replication	なし	なし	なし	Homogeneous Replication Feature for ESE
Database Partitioning	なし	なし	なし	なし

表 2.2: DB2 9.7 エディション: フィーチャーおよび機能サポート

その他の DB2 エディションで使用可能なフィーチャーには、以下のものがあります。

有料で使用できる DB2 Express Edition のフィーチャー

- High Availability Feature

DB2 Workgroup Edition に無料で含まれているフィーチャー

- High Availability and Disaster Recovery (HADR)、Tivoli System Automation、Online Reorganization、Advanced Copy Services
- UNIX プラットフォーム (AIX®、Solaris、HP-UX) への対応

DB2 Enterprise Edition に無料で含まれているフィーチャー

- Table (Range) Partitioning
- Materialized Query Tables (MQT)
- Multi-dimensional Clustering (MDC)
- Query Parallelism
- Connection Concentrator
- Governor

有料で使用できる DB2 Enterprise Edition のフィーチャー

- Storage Optimization Feature (圧縮を含む)
- Advanced Access Control (細分化された拡張セキュリティー)
- Performance Optimization (Workload Management、Performance Expert、Query Patroller)
- Geodetic Data Management (地理的位置を分析する機能)

DB2 関連の有料の製品

- DB2 Connect
- InfoSphere Warehouse の各 Edition
- InfoSphere Balanced Warehouse
- WebSphere Federation Server
- WebSphere Replication Server

2.1 DB2 Express サブスクリプション (FTL) に含まれるフィーチャー

このセクションでは、DB2 の fix pack、HADR、SQL Replication の概要を説明します。

2.1.1 fix pack

DB2 fix pack とは、インストール済みの DB 2 製品に適用するコード修正のセットのことで、製品がリリースされた後に報告されたさまざまな問題を修正することを目的としています。サブスクリプション・ライセンスがインストールされている場合、fix pack は無料でダウンロードしてインストールすることができます。fix pack は通常 4 ヶ月ごと、もしくは必要がある場合に配布されます。

最新の fix pack をダウンロードするには、DB2 技術サポート・サイト http://www.ibm.com/software/data/db2/support/db2_9/ にアクセスしてください。

2.1.2 High Availability Disaster Recovery (HADR)

HADR はデータベースの信頼性に関するフィーチャーの 1 つで、サイトの全体的または部分的な障害に対処するための高可用性および災害時リカバリー・ソリューションを提供します。HADR 環境は通常、1 次データ・サーバーと 2 次データ・サーバーの 2 つで構成されます (この 2 つは地理的に離れた位置に配置することができます)。1 次サーバーにはソース・データベースが保管され、クライアント・アプリケーションはこのサーバーにアクセスします。トランザクションが 1 次データベースで処理されると、データベースのログ・レコードがネットワーク経由で自動的に 2 次サーバーに配信されます。通常、このコピーは 1 次データベースのバックアップを作成し、それを 2 次システム上にリストアすることによって作成されます。2 次サーバーが 1 次データベースのログを受信すると、このログが 2 次データベースで再現され、適用されます。このように、2 次データベースはログ・レコードを継続的に再現することによって、1 次データベースのレプリカを 1 次データベースと同期のとれた状態に維持し、1 次データベースに障害が発生した場合にはこのレプリカが引き継げるようにします。

DB2 が完全にサポートする HADR ソリューションにより、以下の利点がもたらされます。

- カスタマーおよびクライアント・アプリケーションにまったく認識されないほど極めて高速にフェールオーバーが行われます。
- トランザクションの完全な原子性により、データ損失を防ぐことができます。
- 明らかなサービスの中断を伴わずにシステムまたはアプリケーションをアップグレードすることができます。
- リモート・システムへのフェールオーバーにより、ローカル側でデータ・センターが災害に見舞われたとしても完全なリカバリーをすることができます。
- DB2 グラフィカル・ツールを使用して容易に管理を行えます。
- このいずれの機能にしても、システムの全体的なパフォーマンスに悪影響を与えることはありません。

new in V9.7

DB2 9.7 では新たに、クライアントがスタンバイ・サーバー上で読み取り操作を実行できるようになります。この「スタンバイでの読み取り」機能は、DB2 9.7 Fix Pack 1 で使用可能になりました。

2.1.3 Data Replication

このフィーチャーは、データの変更を取り込むソース・サーバーと、データの変更が適用されるターゲット・サーバーとの間でデータを複製 (レプリケーション) できるようにするものです。図 2.1 に、レプリケーションがどのように行われるかを大まかに説明します。

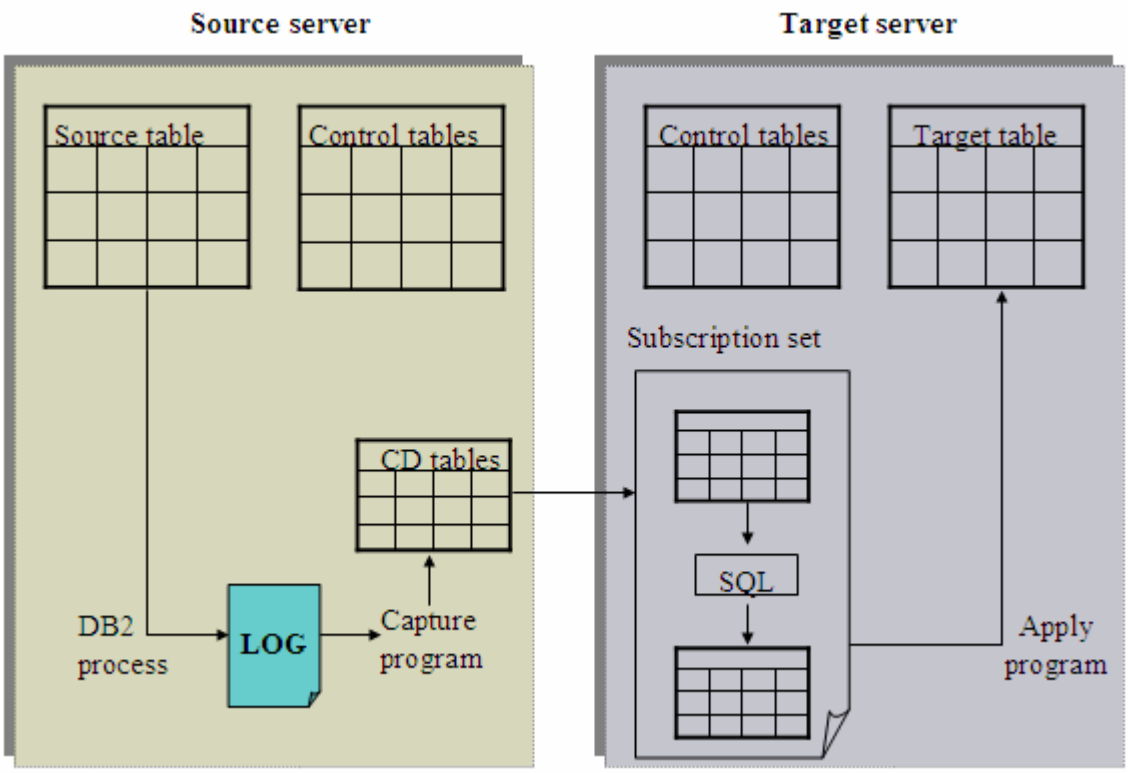


図 2.1 - SQL レプリケーション

図 2.1 には、ソース・サーバーとターゲット・サーバーの 2 つのサーバーがあります。ソース・サーバーでは、キャプチャー・プログラムがデータベースに対して行われた変更を取り込みます。ターゲット・サーバーでは、適用プログラムがデータベース・レプリカに変更を適用します。レプリケーションは、例えば空き容量を増やす場合、データウェアハウスやデータマートにデータを提供する場合、変更履歴を監査する場合など、複製されたデータが必要となるさまざまな状況で役立ちます。SQL レプリケーション・フィーチャーを使用すれば、DB2 Express とその他の IBM データ・サーバー (これには、別の Linux、UNIX、z/OS、i5/OS システム上の DB2 サーバーも含まれます) との間でデータを複製することができます。

2.2 DB2 Express-C の標準装備ではないフィーチャー

このセクションでは、DB2 Express-C や DB2 Express の年間サブスクリプション・ライセンスでは使用できない一方、他の DB2 エディションでは使用できるフィーチャーのいくつかを取り上げます。

2.2.1 Database Partitioning

Database Partitioning Feature (DPF) は、クエリーの処理を複数のデータベース・サーバーからなるクラスターに分散させて処理します。DPF は InfoSphere Warehouse のエディションにのみ用意されているフィーチャーです。このフィーチャーは、複数のデータベース・パーティションやノードにデータを分散させて、いくつかの異なるサーバーにそのデータを配置することができます。DPF はシェアード・ナッシング・アーキテクチャーをベースとしており、データベース・パーティションごとの独立したディスクに、データ全体のうちのサブセットが保管されます。

コンピューターがデータベース・クラスターに追加されると、そのコンピューターの CPU、メモリー、およびディスクを使えるようになるためデータ処理能力が向上し、大規模なタスクや複雑なクエリーを小さく分割してさまざまなデータベース・ノードに分散し、各ノードで並行処理できるようになります。したがって、データベースが単一のサーバーのみに存在する場合よりも、並行性が高くなり、応答時間が短縮される結果となります。DPF が特に役立つのは、大規模なデータウェアハウス環境や、数百ギガバイドから数百テラバイトのビジネス情報データによるワークロードが関与する環境においてです。

2.2.2 Connection Concentrator

Connection Concentrator は、多数の同時接続ユーザーをサポートするためのフィーチャーです。以前は、すべてのデータベース接続にそれぞれ 1 つのデータベース・エージェントが必要でした。Connection Concentrator は「論理エージェント」の概念を導入し、1 つのエージェントが複数の接続を処理できるようにしています。エージェントについての詳細は、第 6 章「DB2 アーキテクチャー」で説明します。

2.2.3 Geodetic Extender

DB2 Geodetic Extender は、DB2 Enterprise Server Edition の有料オプションとして用意されています。このエクステンダーを利用すると、地理的位置の分析が必要なビジネス・インテリジェンスおよび電子政府アプリケーションの開発をとても簡単に行えるようになります。DB2 Geodetic Extender では任意のスケールで仮想地球儀を構成することができます。ほとんどの位置情報は、全地球航行測位衛星 (GPS) などを使った世界規模のシステムによって収集され、緯度と経度の座標 (ジオコード) で表すことが可能です。住所などのビジネス・データは DB2 Geodetic Extender によってジオコードに変換することができるので、エンタープライズ・アプリケーションがより快適な動作をするためには、そうしたデータを、地球儀から平面の地図へ投影する前のジオコード形式で保持するようにします。そして、地球儀から平面の地図へ投影したデータは、地図の表示および出力用にプレゼンテーション層にそのまま残しておくようにします。

2.2.4 Label-based Access Control (LBAC)

LBAC は、行および列レベルで、きめ細かなセキュリティーを実現します。このフィーチャーは、表内のデータへのアクセスを許可するために、ユーザー・セッションとデータ行または列の両方に関連付けられたラベルを利用します。図 2.2 に LBAC の動作の様子を示します。

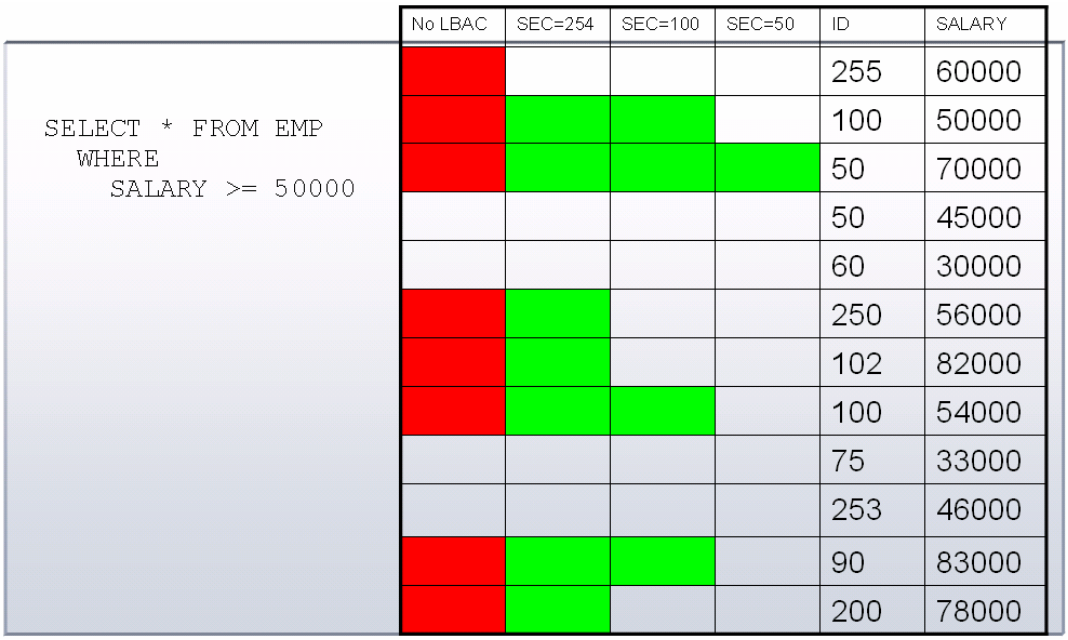


図 2.2 - LBAC の動作例

この図に示す ***EMP*** 表には、***SALARY*** という列と、特定の行のラベルが含まれる ***ID*** という内部列があります。この 2 つ以外は、説明のためだけに使用する列です。図に示されているクエリーが実行されて、その結果としてユーザーに表示される行は、そのユーザーが持っているラベルによって異なります。「No LBAC」というタイトルが付いた列は、LBAC が実装されていない場合に選択される行を表します。この図を見るとわかるように、選択される行はいずれも SALARY (給与) が 50,000 以上となっています。

例えばクエリーを実行したユーザーのセキュリティー・ラベル (SEC) が 100 だとすると、選択される行は、左から数えて 3 番目の列に色が付いている行です。この場合、DB2 は SALARY が 50,000 以上の行を検索し、見つかった行のセキュリティー・ラベルを調べることになります。例えば、最初の行の SALARY は 60000 で、ラベル ID は 255 となっています。クエリーを実行したユーザーのラベル ID は 255 より小さい 100 であるため、このユーザーが最初の行を見ることはできず、したがってクエリーによる出力は返されないことになります。

LBAC セキュリティーを実装するのは、SECADM 権限を持つセキュリティー管理者でなければなりません。

2.2.5 Workload Manager (WLM)

WLM は、ユーザーとアプリケーションの優先順位と、リソースの可用性およびワークロードのしきい値との組み合わせに応じて、データベース全体でのワークロードを管理します。このフィーチャーによって、重要かつ優先順位の高いクエリーを即座に実行できるようにし、適切でないクエリーがシステム・リソースを独占しないようにデータベースのワークロードとクエリーを調整することで、システムを効率的に運用できます。WLM は DB2 9.7 でさらに強化され、それ以前の DB2 バージョンで提供されている Query Patroller や DB2 Governor ツールより強力な機能を提供します。

new in V9.7

2.2.6 Deep Compression

DB2 は、以下のさまざまな圧縮をサポートします。

- NULL およびデフォルト値の圧縮

  このタイプの圧縮は、通常は値が NULL、またはシステム・デフォルト値 (0 など) であるため、ディスク・ストレージを消費しない列に適用されます。

- 多次元クラスタリング

  多次元クラスタリング (MDC = Multidimensional Clustering) 表とは、物理データ・ページが複数の次元でクラスター化されている表のことです。これらの表が使用するブロック索引は、単一のレコードではなく、レコードのブロックを指すため、ある意味、索引を圧縮する方法と言えます。

- データベース・バックアップの圧縮

  このタイプの圧縮は、バックアップ・イメージに適用されます。圧縮されるのは、索引と LOB 表スペースです。

- データ行の圧縮

  行の圧縮は、データ行のなかで何度も繰り返されるストリングを遥かに短い記号に置き換えて表すことで機能します。この短い記号とストリングとのマッピングは、ディクショナリーに保持されます。行の圧縮によって行のサイズが小さくなり、ディスクとメモリーとの間でやり取りできる行の数が多くなることから、行の圧縮は I/O バウンドのワークロードでのパフォーマンスを大幅に改善することができます。また、企業の IT 予算のなかで一般に最大の経費として考えられるストレージを節約できるという利点もあります。ただし、CPU バウンドのワークロードの場合には、圧縮された行は圧縮を解除しないと処理できないことから、オーバーヘッドが増えることになります。圧縮されたレコードのログ・データも同じく圧縮された形式になることに注意してください。

XML 列や LOB 列にアクセスする場合は通常、DB2 はバッファー・プール (メモリー) を使用する代わりに、ディスクに対して直接入出力操作を行います。これは、XML および LOB のサイズは一般に大きいため、メモリーに取り込むとなると、ページをメモリーから除去しなければならなくなるためです。ただし DB2 9.5 では、サイズの小さな XML 文書 (32K 未満) の場合には XML インライン化が許可されます。つまり、サイズの小さな XML 文書であれば、XDA として知られる別個の内部ストレージ・オブジェクトにではなく、基本表の行で保管できるということです。この手法には、二重の利点があります。まず 1 つは、XML 文書にバッファー・プールを介して XML 文書にアクセ

スできるようになること、そしてもう 1 つは XML 文書もデータ行の圧縮によるメリットを受けられることです。

new in V9.7

DB2 9.7 では、圧縮にさらに新しい機能拡張が加えられています。

- XDA 内部オブジェクト (XML が保管される場所) も圧縮できるようになりました。
- 索引および一時表 (システムおよびユーザー) を圧縮することができます。
- XML のインライン化と同じような方法で LOB をインライン化することができます。

2.2.7 SQL Compatibility

new in V9.7

多くのベンダーが SQL 92 および SQL/PSM 標準に従ってはいるものの、これらの標準のすべてのフィーチャーがサポートされているわけではなく、また、標準に含まれていないフィーチャーもあります。DB2 9.7 の SQL Compatibility Feature により、DB2 では DB2 独自の SQL PL だけでなく、他の RDBMS ベンダーがサポートする PL/SQL 構文のほとんどをサポートできるようになりました。図 2.3 に、このサポートの仕組みを要約します。

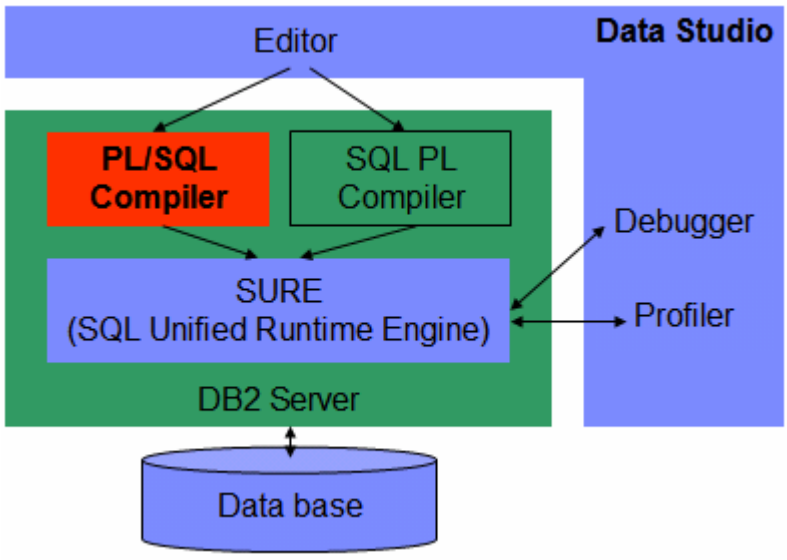


図 2.3 – DB2 での PL/SQL サポート

この図を見るとわかるように、PL/SQL コンパイラーが開発され、DB2 エンジンに組み込まれました。

SQL Compatibility Feature にも、CLPPlus と呼ばれるツールのサポートが組み込まれています。CLPPlus は、SQL や他のコマンドを実行できるようにするコマンド行ツールで、既存の DB2 コマンド行プロセッサー (CLP) と同様です。図 2.4 に、CLPPlus ツールの実行画面を示します。

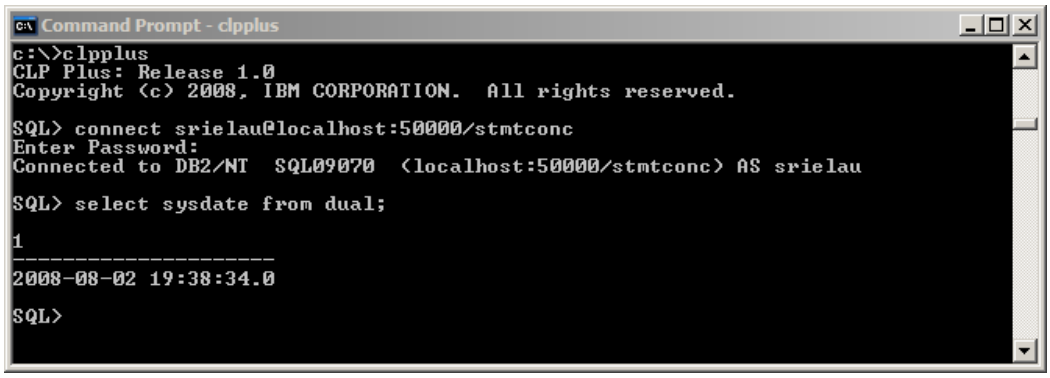


図 2.4 –CLPPlus

BINARY_INTEGER、RAW をはじめ、ほとんどの PL/SQL データ型に対するサポートも組み込まれています。他の Oracle データ型 (VARCHAR2 など) は、SQL Compatibility Feature がなくてもサポートされますが、DB2_COMPATIBILITY_VECTOR レジストリー変数を使用してサポートを有効にする必要があります。Oracle データ型とこのレジストリー変数については、この後の章で詳しく説明します。

上記で概説した SQL Compatibility Feature が現在提供されているのは DB2 9.7 DB2 Express (年間サブスクリプション・オプションつまり FTL を含め)、 Workgroup および Enterprise エディションです。DB2 Express-C エディションでは提供されていません。

PL/SQL サポートおよび CLPPlus フィーチャーの機能は DB2 Express-C 9.7 では使用できませんが、このエディションに組み込まれている他のフィーチャーにより、DB2 で Oracle アプリケーションを簡単に使えるようになります。これらのフィーチャーには、新規データ型、新規スカラー関数、モジュール・サポート、カーソル固定 (CS = Cursor Stability) の分離レベルに応じた現在コミット済み (CC = Currently Committed) のセマンティクスなどがあります。これらのフィーチャーについては、この後の章で説明します。

2.3 DB2 関連の有料の製品

このセクションでは、DB2 で使用できる無料の製品およびオファリングについて簡単に説明します。

2.3.1 DB2 Connect

DB2 Connect は、DB2 for Linux, UNIX, and Windows クライアントを DB2 for z/OS または DB2 for i5/OS サーバーに接続するための有料ソフトウェアです (図 2.5 を参照)。反対方向、つまり DB2 for z/OS または DB2 for i5/OS から DB2 for Linux, UNIX, and Windows サーバーに接続する場合には、DB2 Connect は必要ありません。DB2 Connect には、接続のニーズに応じて 2 つの主要なエディション、DB2 Connect Personal Edition と DB2 Connect Enterprise Edition が用意されています。

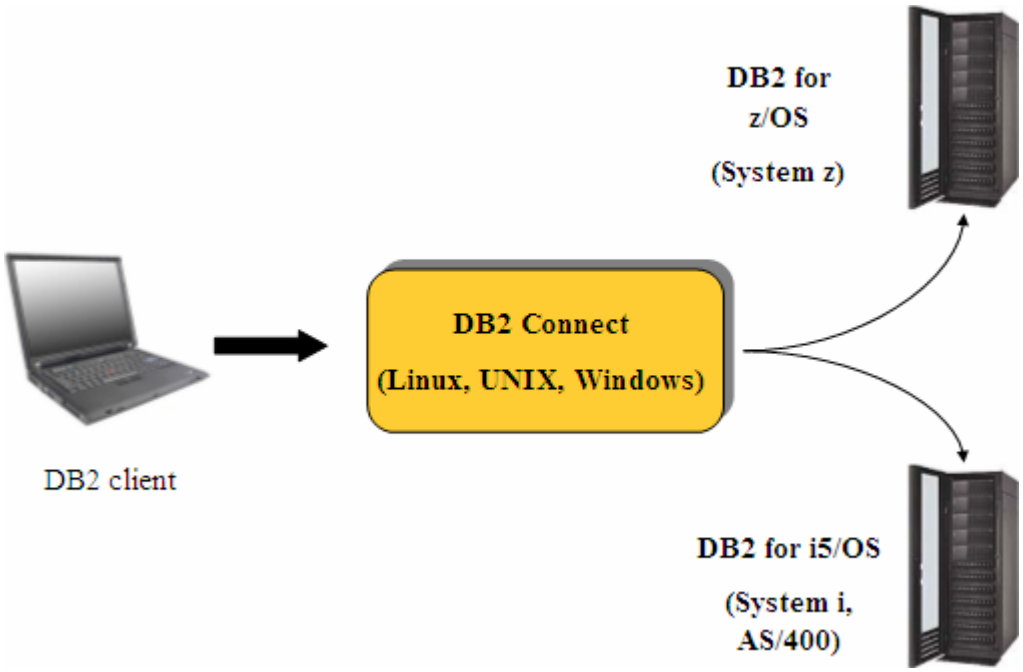


図 2.5 – DB2 Connect

2.3.2 InfoSphere Federation Server

以前は WebSphere Information Integrator (フェデレーション・サポート用) として知られていた WebSphere Federation Server は、データベースのフェデレーションを可能にします。これはつまり、さまざまなリレーショナル・データベース・システムのオブジェクトを処理可能なデータベース・クエリーを実行できるということです。WebSphere Federation Server を購入すると、例えば以下のリスト 2.1 に示すクエリーを実行することができます。

```
SELECT  *
FROM      Oracle.Table1  A
             DB2.Table2  B
       SQLServer.Table3  C
WHERE
            A.col1 < 100
        and B.col5 = 1000
        and C.col2 = 'Test'
```

リスト 2.1 – フェデレーション・クエリー

図 2.6 に、WebSphere Federation Server を使用した場合の処理の様子を示します。

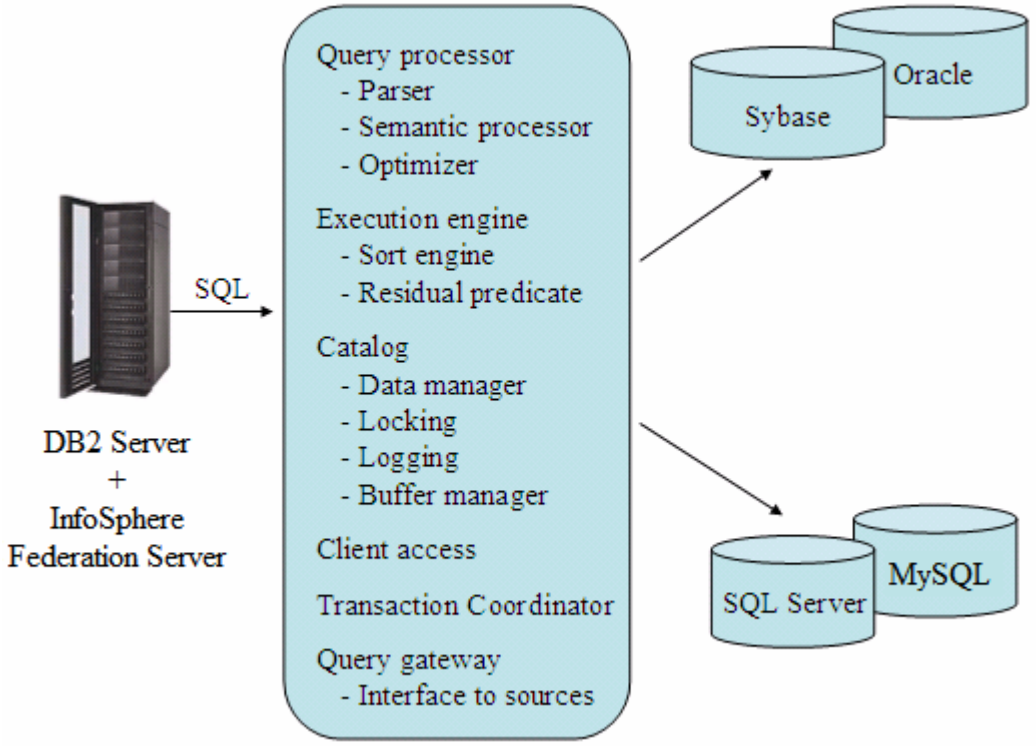


図 2.6 – InfoSphere Federation Server

IBM ファミリーに含まれるリレーショナル・データベース管理システムに対しては、DB2 Express-C にはすでにフェデレーション・サポートが組み込まれています。つまり、例えば 2 つの異なる DB2 データベース間、または DB2 データベースと Informix データベース (041Informix は IBM ファミリーです) の間でクエリーを実行するのであれば、WebSphere Federation Server 製品は必要ありません。

2.3.3 InfoSphere Replication Server

以前は WebSphere Information Integrator (レプリケーション・サポート用) として知られていた InfoSphere Replication Server は、IBM 以外のデータ・サーバーが関係する場合のデータベース・レコードの SQL レプリケーションを可能にします。このサーバーには、メッセージ・クエリーを使用してデータを複製する、Q-Replication として知られるフィーチャーも組み込まれています。

2.3.4 Optim Development Studio (ODS)

以前は Data Studio Developer として知られていた ODS は、Data Studio と簡単に統合して同じ Eclipse を共有できる Eclipse ベースのツールです。ODS では、既存の Oracle または DB2 データベースからのコピー・アンド・ペースト操作によって開発用データベースを作成することができます。

2.3.5 Optim Database Administrator (ODA)

以前は Data Studio Administrator として知られていた ODA は、Data Studio と簡単に統合して同じ Eclipse を共有できる Eclipse ベースのツールです。ODA には変更管理機能が提供されているため、ODA を使用することでスキーマの変更をより簡単に自動化することができます。

2.4 Amazon Elastic Compute Cloud での DB2 オファリング

特筆すべき点として、IBM では Amazon Web Services (AWS) と提携を結び、Amazon の Elastic Compute Cloud (EC2) で DB2 を実行できるようにしています。AWS が提供する一連の統合サービスで構成された「クラウド内」のコンピューティング・プラットフォームは、従量課金モデルで使用することができます。つまり、ユーザーは AWS から計算能力 (仮想サーバーおよびストレージ) を「レンタル」し、使用した分だけ料金を支払うということです。例えば、通常のデータベース操作に対して 1 つの EC2 仮想サーバーをプロビジョニングし、ピーク時や特定の期間に必要な数時間の間、追加のデータベース・サーバーをプロビジョニングするとします。この場合、AWS に追加で支払う料金は、追加のデータベース・サーバーを稼働させた時間の分だけです。

IBM では、Amazon のクラウド・プラットフォーム上の DB2 について、以下の 3 つのデプロイメント・オプションを用意しています。

- 評価および開発用の DB2 Express-C AMI
- DB2 Express および DB2 Workgroup での従量課金による本番対応 AMI
- DB2 ライセンスで作成した独自の AMI

Amazon EC2 での DB2 に関する詳細および使用方法については、www.ibm.com/db2/cloud にアクセスしてください。

2.5 まとめ

DB2 Express-C は、データベース・アプリケーションを開発して本番環境にデプロイしたり、さらにはアプリケーションとサード・パーティー・ソリューションを統合して配布したりする際の、無料で使いやすく、堅牢な基盤となります。コミュニティーによる支援があれば十分な場合、そして最新のフィックスや高度なフィーチャーが必要でない場合には、このエディションは理想的です。ただし、IBM による正式な技術サポートと定期的なソフトウェア更新 (fix pack) が必要であったり、追加リソースの使用、高可用性クラスタリング・サポートを必要とする場合に備え、IBM では低価格の DB2 Express 年間サブスクリプション・ライセンス (FTL) を用意しています。一方、ミッション・クリティカルなワークロードおよび大規模なデータベース・アプリケーションに対応した高度な機能が必要であれば、IBM がさらにスケーラブルな DB2 エディションおよび関連製品を提供します。したがって、まずは DB2 Express-C から初め、ビジネス・デマンドに合わせてエディションをアップグレードしていくことも可能です。

3

第 3 章 – DB2 のインストール

DB2 のインストールは非常に簡単です。通常のインストールでは、デフォルトのオプションを選択していけば、短時間で DB2 サーバーが起動して稼働状態になります。

3.1 インストールの前提条件

DB2 Express-C は Linux®、Sun Solaris (x64)、Microsoft Windows® 2003、XP、Vista に対応しています。また Mac OS X に対応したベータ版も用意されています。対応可能なプロセッサー・アーキテクチャーは、32bit、64bit、および PowerPC (Linux) です。この他のプラットフォーム (UNIX など) で DB2 を実行する場合には、本書で前に説明したデータ・サーバー・エディションのいずれかを購入する必要があります。すべての DB2 エディションに共通のオペレーティング・システム要件については、以下のドキュメントに記載されています。http://www.ibm.com/software/data/db2/udb/sysreqs.html

DB2 Express-C をインストールするシステムのハードウェア・リソースに関しては、CPU のコア数やメモリーについての要件はありませんが、保証無しの無料ライセンス・バージョンで使用されるのは最大で 2 コアと 2GB のメモリーのみとなります。DB2 Express のサブスクリプション・バージョンを購入すると、最大で 4 コアと 4GB のメモリーを使用できるようになります。システムは物理システムでも、パーティションや仮想マシン・ソフトウェアを使用した仮想システムでも構いません。もちろんお望みの場合には、これよりも小規模なシステム (例えば、1 GB のメモリーが搭載されたシングル・プロセッサーのシステムなど) で実行することもできます。

DB2 Express-C のハードウェア要件に関する最新情報については、以下の DB2 Express-C Web ページで、をご確認ください。

http://www.ibm.com/software/data/db2/express/about.html

3.2 インストールに必要なオペレーティング・システム権限

DB2 Express-C を Linux または Windows にインストールするには、十分な権限を持つオペレーティング・システム・ユーザーとして操作する必要があります。

Linux の場合、DB2 Express-C はルート (スーパーユーザー) としてインストールする必要があります。ルート以外のユーザーとして DB2 Express-C をインストールすることもできますが、その場合、製品で実行できる機能が制限されます。その一例として、ルート以外のユーザーとしてインストールすると、インストール時に作成されたデフォルトのインスタンス数を超えるインスタンスを作成できなくなります。

Windows の場合、インストールを実行するマシンの管理者グループに属するユーザー・アカウントを使用する必要があります。Windows 2008、Windows Vista、またはそれ以降の場合、管理者以外でもインストールを実行することはできますが、DB2 セットアップ・ウィザードから管理資格情報の入力を求めるプロンプトが出されます。

インストールでドメイン・アカウントの作成または検証が必要な場合、インストール・ユーザー ID はそのドメインの「ドメイン管理者」グループに属していなければなりません。

また、インストールを実行するには、(推奨されていない場合でも) 標準装備の「ローカル・システム」アカウントを使用する必要があります。「ローカル・システム」アカウントにはパスワードは不要ですが、このアカウントでネットワーク・リソースにアクセスすることはできません。

ユーザー・アカウントには、「ネットワークからこのコンピューターにアクセスする」権利が必要です。

3.3 インストール・ウィザード

DB2 Express-C をインストールする方法はいくつかありますが、そのうち最も簡単なのは、GUI ベースの DB2 インストール・ウィザードを使用する方法です。DB2 Express-C のイメージをダウンロードして解凍したら、以下のようにしてウィザードを起動することができます。

- Windows の場合: EXP/image/ ディレクトリーにある **setup.exe** ファイルを実行します。
- Linux の場合: exp/disk1/ ディレクトリーにある **db2setup** コマンドを実行します。

DB2 インストール・ウィザードの手順に従うと、簡単に DB2 Express-C をインストールすることができます。ほとんどの場合はデフォルト設定で十分なので、必要な作業はご使用条件を承諾し、「完了」ボタンが表示されるまで「次へ」ボタンをクリックし、最後に「完了」ボタンをクリックすることだけです。数分後にインストールが完了すると、DB2 が立ち上がり、稼働状態になります。

図 3.1 に、「セットアップ・ランチパッド」画面を示します。この画面では「製品のインストール」をクリックし、「新規インストール」を選択して DB2 Express-C を新たにシステムにインストールします。以前に DB2 Express-C またはその他の DB2 エディションをインストールしたことがある場合は、「既存の製品を操作」というボタンが表示されます。DB2 では、製品を複数回インストールすることができ、またそれらのインストール済み環境はバージョンまたはリリースのレベルが異なっていてもかまいません。

図 3.1 – DB2 セットアップ・ランチパッド画面

ご使用条件の条項に同意した後の画面では、通常は図 3.2 に示すように「標準」インストール (デフォルト) を選択します。DB2 Text Search をインストールに含めたい場合は「カスタム」を選択します。

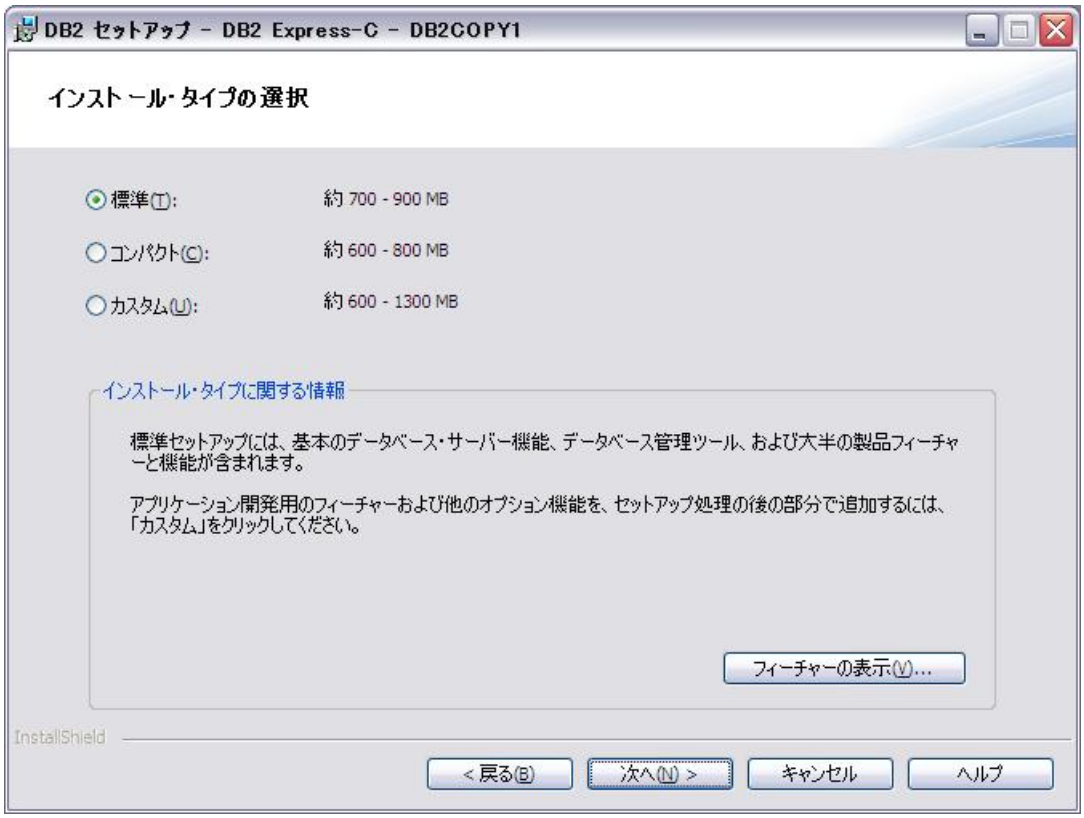


図 3.2 – インストール・タイプ

図 3.3 に示す次のステップでは、製品をインストールするか、インストールの設定を応答ファイルに保管するか、またはその両方を行うかを選択することができます (応答ファイルについてはセクション 3.4 「サイレント・インストール」で説明します)。この画面では通常、デフォルトの選択「このコンピューターに DB2 Express-C をインストールし、設定を応答ファイルに保管する」) のままにします。

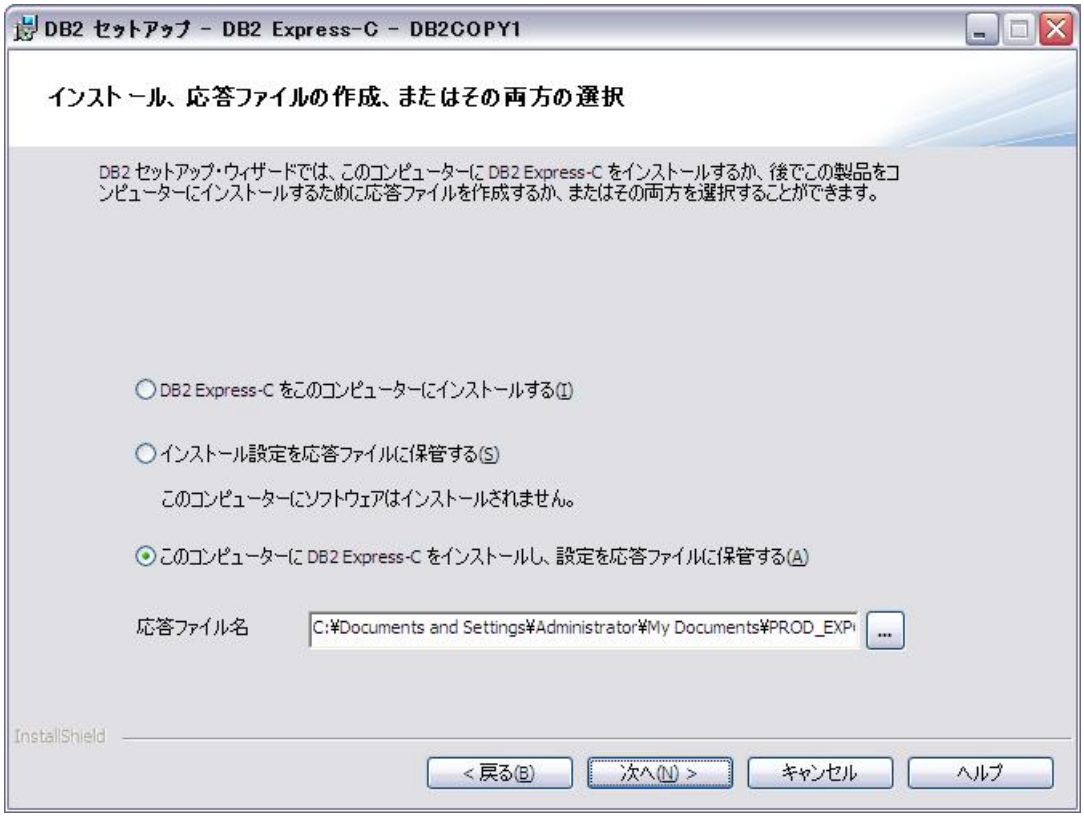


図 3.3 – インストールの選択

続いて表示されるいくつかの画面ではデフォルト値を選択し、図 3.4 に示すパネルが表示されたら、インスタンスやその他のサービスをセットアップおよび実行する際に使用するユーザー ID を入力します。

Windows で既存のユーザー ID を使用する場合には、そのユーザーはローカル管理者グループに属していなければなりません。

ユーザー ID が既存のユーザーの ID ではない場合、そのユーザー ID がローカル管理者として作成されます。ユーザー ID がドメインに属していなければ、ドメイン入力用のフィールドは空白のままにして構いません。

Windows でのデフォルトのユーザー ID は **db2admin** です。Linux の場合に作成されるデフォルトのユーザー ID は **db2inst1** です。

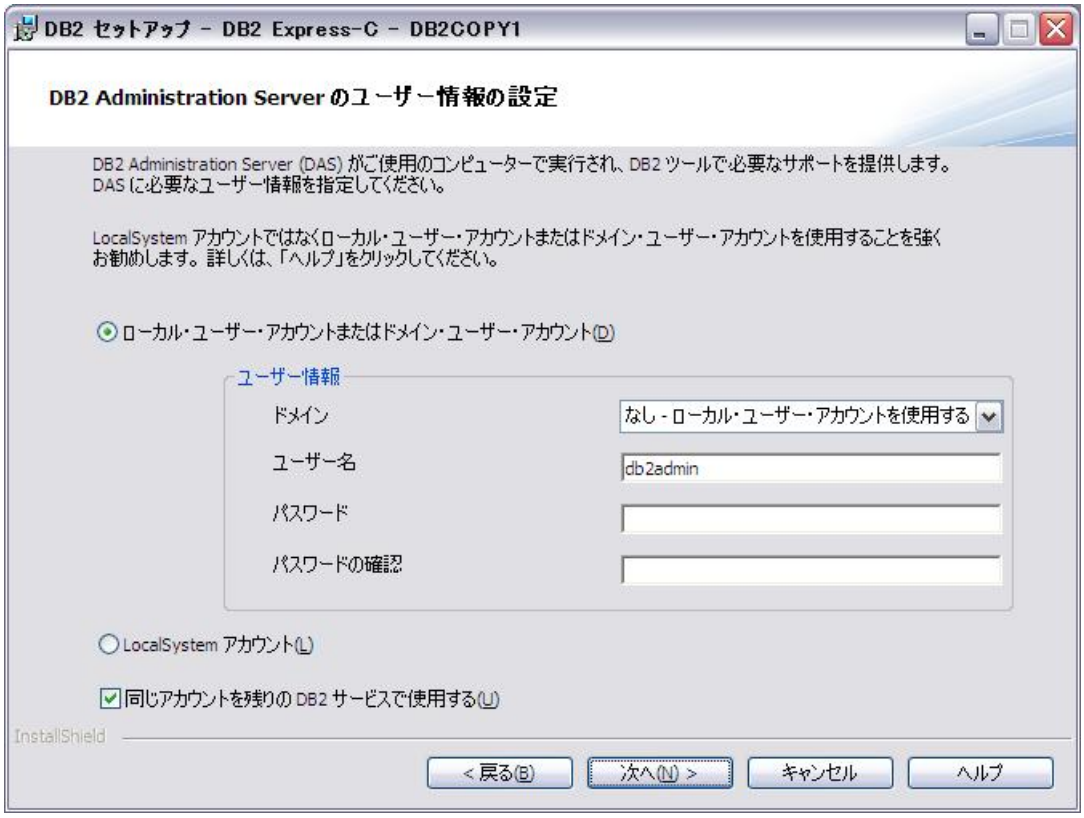


図 3.4 – DB2 Administration Server のユーザー情報の指定

最後にインストール・ウィザードには、インストール内容の要約と、これまでのステップで指定されたさまざまな構成値が表示されます (図 3.5)。この画面で「完了」ボタンをクリックするとインストールが開始され、プログラム・ファイルがシステムに導入されます。

図 3.5 – インストール内容の要約

インストールが完了すると、図 3.6 に示すようなウィンドウが表示され、インストール・ウィザード・プロセスの結果と、インストールを完了するために必要な残りのステップが示されます。

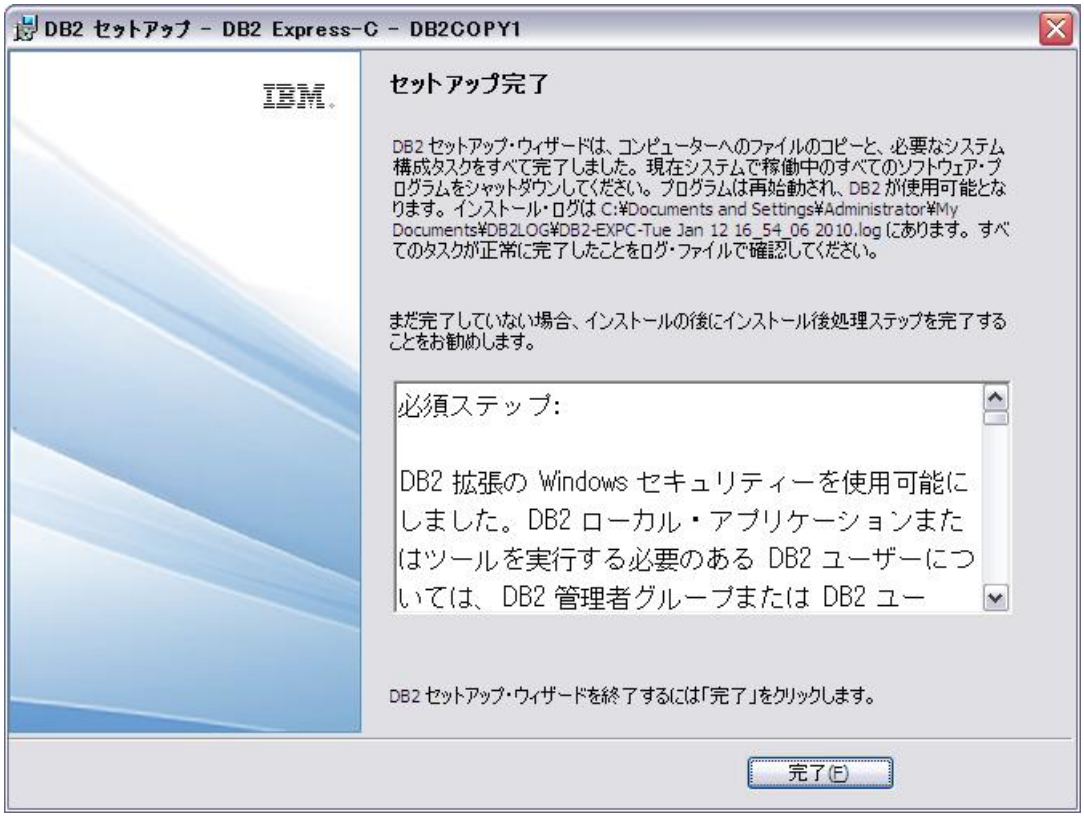


図 3.6 – インストールの完了

インストールの結果を示す図 3.6 のウィンドウで「完了」をクリックすると、図 3.7 に示すように、「DB2 ファースト・ステップ」アプリケーションが起動されます。

この小さなアプリケーションは、デフォルトのサンプル・データベース (**SAMPLE** という名前が付けられています) の作成や、ユーザー独自の新規データベースの作成など、DB2 の使用を始めるためのいくつかのオプションの概要を示します。この時点では「ファースト・ステップ」アプリケーションを使って DB2 の作業を行わない場合は、ウィンドウを閉じ、後で呼び出すことができます。

Windows で DB2 ファースト・ステップを手動で開始するには、「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「セットアップ・ツール」 -> 「ファースト・ステップ」の順に選択するか、またはコマンド・プロンプトで **db2fs** コマンドを実行します。

Linux では、ターミナル・ウィンドウから **db2fs** コマンドを実行します。

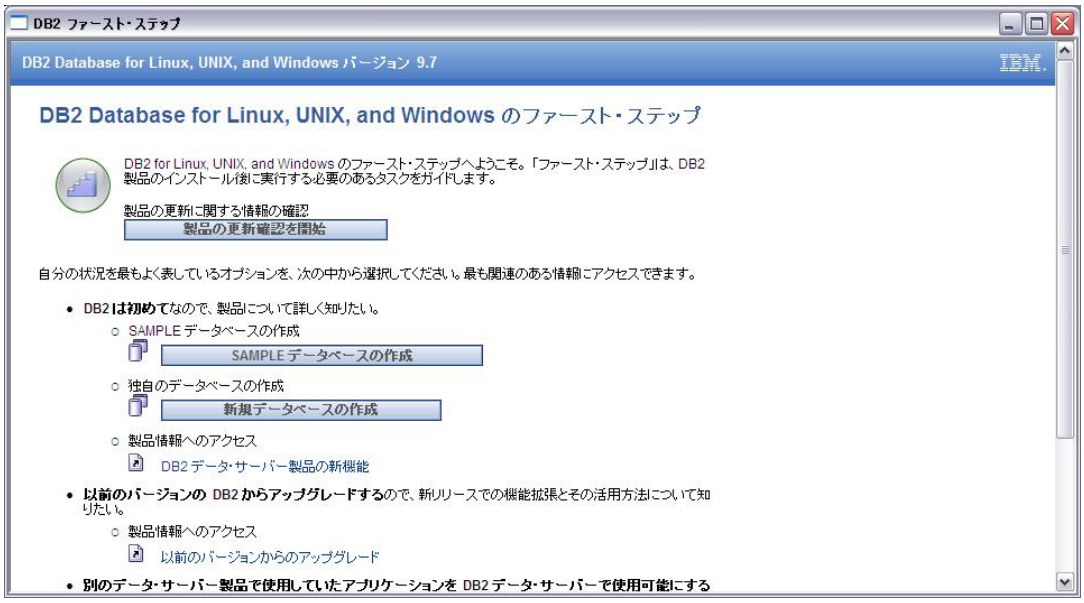


図 3.7 – ファースト・ステップ

3.4 インストールの検証

DB2 のインストール後、DB2 コマンド・ウィンドウ (Windows の場合) またはターミナル・ウィンドウ (Linux の場合) から 3 つのコマンドを実行して、インストールが正常に行われたかどうかを確認することができます。

- **db2level**: このコマンドは、インストールされた DB2 製品、fix pack のレベル、およびその他の詳細情報を表示します。
- **db2licm -l**: このコマンドは、インストールされた DB2 製品に固有のライセンス情報をすべてリストアップします。

new in V9.7

- **db2val**: これは DB2 9.7 で使用可能な新しいコマンドで、DB2 コピーのコア機能を検証することによって、インストールの検証を行います。ご使用のインスタンスに矛盾がないか、またデータベース作成およびデータベース接続が機能するかどうかを確認します。

下の図 3.8 に、これらの 3 つのコマンドの出力例を示します。

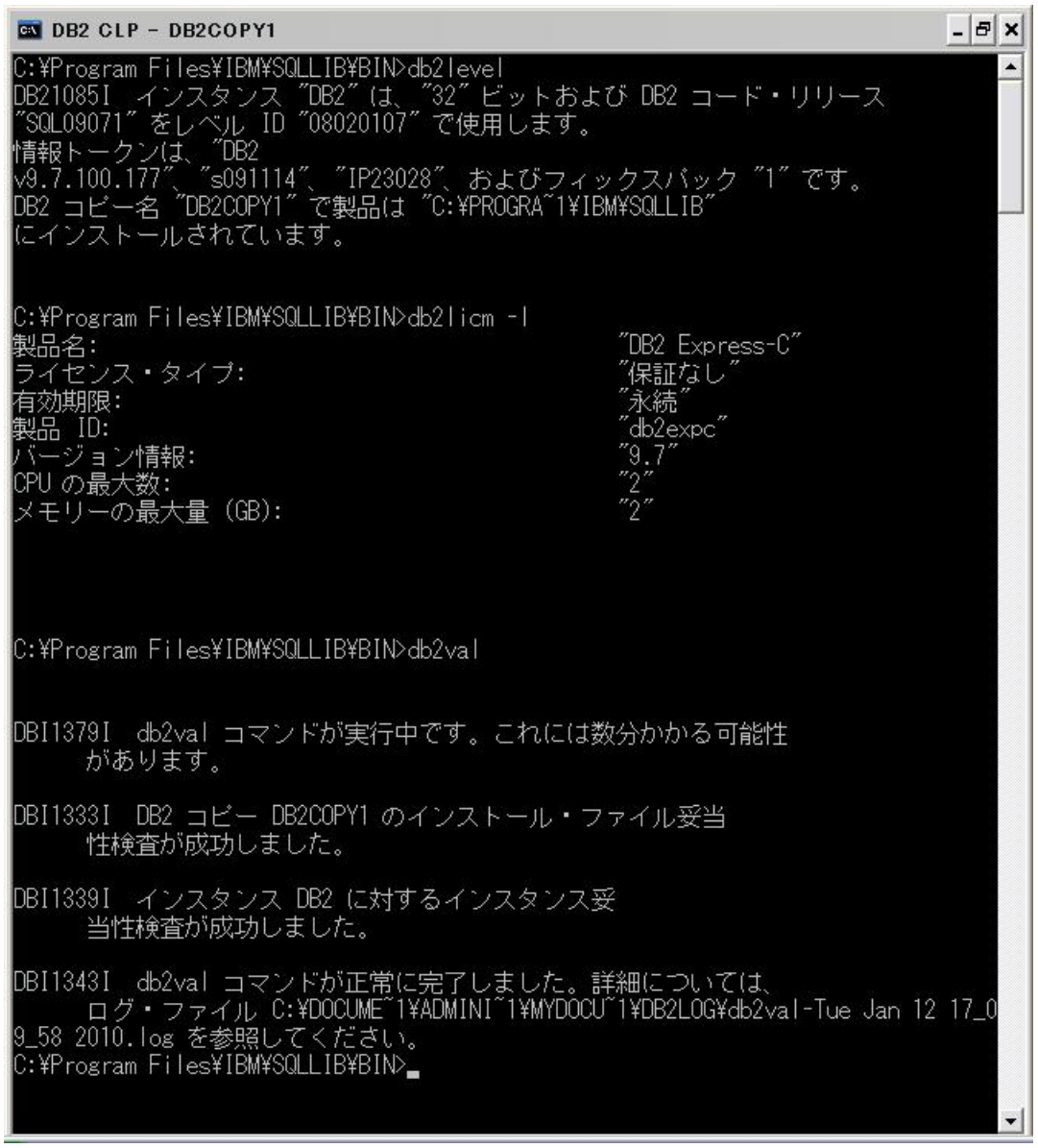


図 3.8 – インストール検証用の db2level、db2licm、および db2val コマンド

この図で、**db2level** コマンドの出力は、Fix Pack 0 の DB2 9.7 (DB2 v9.7.0.441) を実行していることを示しています。これは、ご使用の DB2 コードが、フィックスが適用されていない基本 (GA) レベルであることを意味します。**db2licm -l** コマンドの出力は、最大 2 つのコアの使用を許可す

る永続かつ無保証のライセンスを持つ DB2 Express-C エディションをインストールしたことを示しています。**db2val** コマンド出力についてはご覧のとおりです。

注:

データベースの一貫性をいつでも好きなときに検証したい場合は、INSPECT ユーティリティーを使用してください。

3.5 サイレント・インストール

DB2 クライアントを複数のコンピューターにインストールしなければならない状況や、DB2 データ・サーバーをアプリケーションの一部として組み込まなければならないような状況で、アプリケーションのインストール・プロセスの一部として DB2 をインストールしたいという場合があるかもしれません。そのような場合には、サイレント・インストールにより DB2 をインストールするのが理想的です。

DB2 では、インストール情報が単純なテキストの選択肢として保管された応答ファイルを使用してサイレント・インストールを行うことができます。以下に示すのは、サンプル応答ファイルのソース・コードです。

```
PROD=UDB_EXPRESS_EDITION
LIC_AGREEMENT=ACCEPT
FILE=C:\Program Files\IBM\SQLLIB\
INSTALL_TYPE=TYPICAL
LANG=EN
INSTANCE=DB2
DB2.NAME=DB2
DEFAULT_INSTANCE=DB2
DB2.SVCENAME=db2c_DB2
DB2.DB2COMM=TCPIP
...
```

リスト 3.1 – サンプル応答ファイル

応答ファイルを生成するには、いくつかの方法があります。

- DB2 インストール・ウィザードを使用して、DB2 Express-C のインストールを開始します。ウィザードの始めのほうのオプション (図 3.3 を参照) の 1 つに、インストールの設定内容を応答ファイルに保存するためのチェック・ボックスがあります。このチェック・ボックスを選択すると、ウィザードはインストールの終了時に、指定したディレクトリーに指定したファイル名で応答ファイルを生成します。このファイルはテキスト・ファイルなので、後ほど手作業で編集することができます。

- DB2 Express-C イメージに提供されているサンプル・応答ファイルを編集します。このサンプル・ファイル (.rsp というファイル拡張子のファイル) は、db2/platform/samples/ ディレクトリーにあります。
- Windows の場合は、以下の応答ファイル生成プログラム・コマンドを使用することもできます。:

```
db2rspgn -d <output directory>
```

上記のいずれかの方法で生成した応答ファイルを使用して DB2 のサイレント・インストールを行うには、Windows の場合、以下のコマンドを実行します。

```
setup -u <response filename>
```

Linux の場合は、以下のコマンドを実行します。

```
db2setup -r <response filename>
```

3.6 まとめ

この章では、DB2 Express-C のインストールについて詳しく説明しました。この DB2 エディションは、Linux、Solaris、および大半の Windows に対応しており、32 ビット、64 ビット、および Power PC アーキテクチャーで実行することができます。この章ではさらに、DB2 のインストールに必要なユーザー権限について説明した後、DB2 インストール・ウィザード GUI を使用した容易なインストールの手順を説明しました。続いて、DB2 ファースト・ステップの実行、インストールの検証などのインストール後の作業についても触れ、最後に、DB2 応答ファイルを使用したサイレント・インストールの作成方法および実行方法を確認しました。

3.7 演習　DB2 Express-C をインストールして、SAMPLE データベースを作成する

この演習では、DB2 Express-C をインストールして SAMPLE データベースを作成します。

目的

DB2 Express-C に付属のすべてのフィーチャーとツールを体験するには、まず DB2 Express-C をお使いのシステムにインストールする必要があります。この演習では、Windows での DB2 Express-C の基本的なインストール手順を実行します。Linux でも同じインストール・ウィザードを使用できるので、Linux プラットフォームでのインストール手順も Windows の場合とほとんど変わりません。

手順

1. **「DB2 Express-C イメージの入手」**
   DB2 Express-C Web サイト (www.ibm.com/db2/express) から該当する DB2 Express-C イメージをダウンロードします。このファイルを任意のディレクトリーに解凍します。

2. **「ファイルの配置場所の確認」**
   解凍した DB2 製品インストール・ファイルがどのディレクトリー (またはドライブ) に置かれているか、確認します。

3. **「DB2 セットアップ・ランチパッドの実行」**
   DB2 セットアップ・ランチパッドを起動するには、`setup.exe` ファイルをダブルクリックします。Linux では、ルートとして `db2setup` コマンドを実行します。「ランチパッド」ウィンドウの左ペインにある「製品のインストール」オプションをクリックします。

4. **「DB2 セットアップ・ウィザードの実行」**
   DB2 セットアップ・ウィザードは、すべてのシステム要件が満たされていることを確認するとともに、既存の DB2 インストールの有無を調べます。「新規インストール」をクリックしてウィザードを起動し、「次へ」をクリックします。

5. **「ご使用条件の確認」**
   ご使用条件を読んでこれに同意し (「...同意します。」ラジオ・ボタンを選択)、「次へ」ボタンをクリックして先に進みます。

6. **「インストール・タイプの選択」**
   この演習では、デフォルトの「標準」オプションを選択します。ちなみに、「コンパクト」オプションを選択すると基本的なインストールが実行され、「カスタム」オプションを選択すると特定の機能を選択してインストールすることができます。「次へ」ボタンをクリックして先に進みます。

7. **「インストール、応答ファイル作成のいずれか、またはその両方の選択」**
   デフォルトのままにすると、DB2 はインストールされ、応答ファイルも作成されます。「次へ」ボタンをクリックして先に進みます。

8. **「インストール先フォルダーの選択」**
   この画面では、DB2 コードをインストールするシステム上のインストール先ドライブおよびディレクトリーを選択することができます。コードをインストールするのに十分なスペースがあることを確認してください。この例では、以下のデフォルトのドライブおよびディレクトリー設定を使用します。

   ドライブ: `C:`

   ディレクトリー: `C:\Program Files\IBM\SQLLIB`

   「次へ」ボタンをクリックして先に進みます。

9. **ユーザー情報を設定します。** DB2 Express-C のインストールが完了すると、特定の DB2 プロセスがシステム・サービスとして実行されることになります。これらのサービスを実行するには、オペレーティング・システムのアカウントが必要です。Windows 環境では、デフォルトの **db2admin** ユーザー・アカウントを使用することをお薦めします。ユーザー・アカウントがまだない場合には、DB2 が自動的にユーザー・アカウントをオペレーティング・システムに作成します。既存のアカウントを使用するように指定することもできますが、そのアカウントにはローカル管理者権限があることが条件となるので、提案されたデフォルトのアカウントを使用することをお薦めします。アカウントには必ずパスワードを指定してください。Linux の場合、デフォルト・ユーザー ID は、インスタンス所有者には

db2inst1、fenced ユーザーには **db2fenc1**、DB2 Administration Server のユーザーには **dasusr1** が設定されます。「次へ」ボタンをクリックして先に進みます。

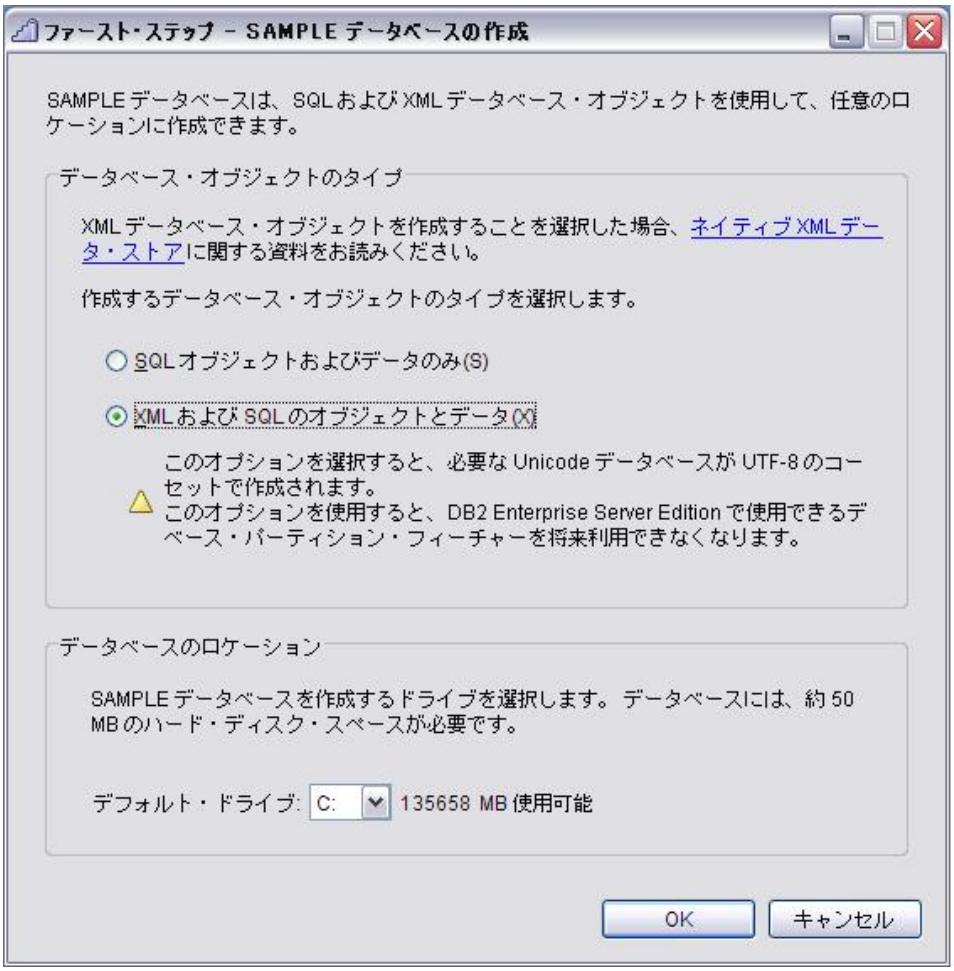
10. DB2 インスタンスを構成します。DB2 インスタンスとは、データベースのコンテナーのようなものです。コンテナーとなるインスタンスが存在しなければ、その内部にデータベースを作成することができません。Windows の場合、インストール中に **DB2** というインスタンスが自動的に作成されます。Linux 環境でのデフォルト・インスタンス名は **db2inst1** です。インスタンスについては本書の後で説明します。

    デフォルトでは、DB2 インスタンスはポート 50000 で TCP/IP 接続をリッスンするように構成されます。デフォルトのプロトコルとポートを変更するには、「構成」ボタンをクリックします。この例ではデフォルト設定を使用することをお薦めします。「次へ」ボタンをクリックして先に進みます。

11. インストールを開始します。インストールの概要を示すこれまでに選択したオプションを確認してください。「完了」ボタンをクリックすると、インストール・ロケーションへのファイルのコピーが始まります。さらに、DB2 が初期構成プロセスを実行します。

12. インストールが完了すると、「ファースト・ステップ」という名前の別の起動ユーティリティーが表示されます。このファースト・ステップは、コマンド **db2fs** を実行して後から起動することもできます。

13. **SAMPLE** データベースは、テスト目的で使用できるデータベースです。このデータベースは、ファースト・ステップから、「SAMPLE データベースの作成」ボタンをクリックすることで作成することができます。2 番目のオプション (XML および SQL のオブジェクトとデータ) を選択してください。SAMPLE データベースは、コマンド **db2sampl -xml -sql** を使用して作成することもできます。



14. 数分後、データベースが作成されていることを確認することができます。そのためには、DB2 「コントロール・センター」ツールを開き、「スタート」 -> 「すべてのプログラム」 -> IBM DB2 -> 「DB2COPY1 (デフォルト)」 -> 「汎用管理ツール」 -> 「コントロール・センター」の順に選択します。コントロール・センターを起動するには、コマンド **db2cc** を実行するという方法もあります。初めてコントロール・センターを起動すると、ポップアップ・ウィンドウが表示され、使用するコントロール・センター・ビューの選択を求められるので、デフォルト (「詳細」) のままにし、「OK」をクリックしてください。左のパネルに、「すべてのデータベース」フォルダーの詳細が示されます。そのフォルダーに SAMPLE データベースがない場合は、「ビュー」 -> 「リフレッシュ」を選択して、ビューを最新の表示にしてください。

15. コンピューターを再起動します。このステップはオプションであるため、公式の DB2 インストール・マニュアルには記載されていませんが、システムを再起動することをお薦めします (少なくとも Windows では、可能であれば再起動してください)。これにより、すべてのプロセスが正常に開始されること、そして正しくクリーンアップされていない可能性のあるメモリー・リソースがクリーンアップされることを確実にすることができます。

16. コマンド **db2level**、**db2licm**、および **db2val** を実行して、DB2 インストールを検証します。Windows の「スタート」メニューから、「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「コマンド行ツール」 -> 「コマンド・ウィンドウ」の順に選択して、DB2 コマンド・ウィンドウを開きます。コマンド・ウィンドウ (Linux の場合はシェル) から、**db2level** と入力して出力を調べます。コマンド **db2licm -l** でも同じことを行います。次に **db2val** コマンドを実行します。**db2val** が正常に終了した場合、インストールは正常に行われています。エラーがある場合は、エラー・メッセージに示されたログ・ファイルで詳細を調べてください。3 つのコマンドの出力例は、前に図 3.8 で示してあります。

4

第 4 章 – DB2 環境

DB2 環境にはさまざまなデータベース･オブジェクトや構成ファイルが含まれています。図 4.1 は DB2 を操作するためのさまざまなコマンドとツールの概要を示しており、右側に赤い楕円で示しているのがこの章で焦点を当てる DB2 環境です。図の左側には DB2 データ・サーバーを操作する際に使用できるさまざまな DB2 コマンド、SQL ステートメント、SQL/XML ステートメント、そして XQuery ステートメントが示されています。図の中央には DB2 データ・サーバーを操作する際に使用する各種ツールの名前が示されています。

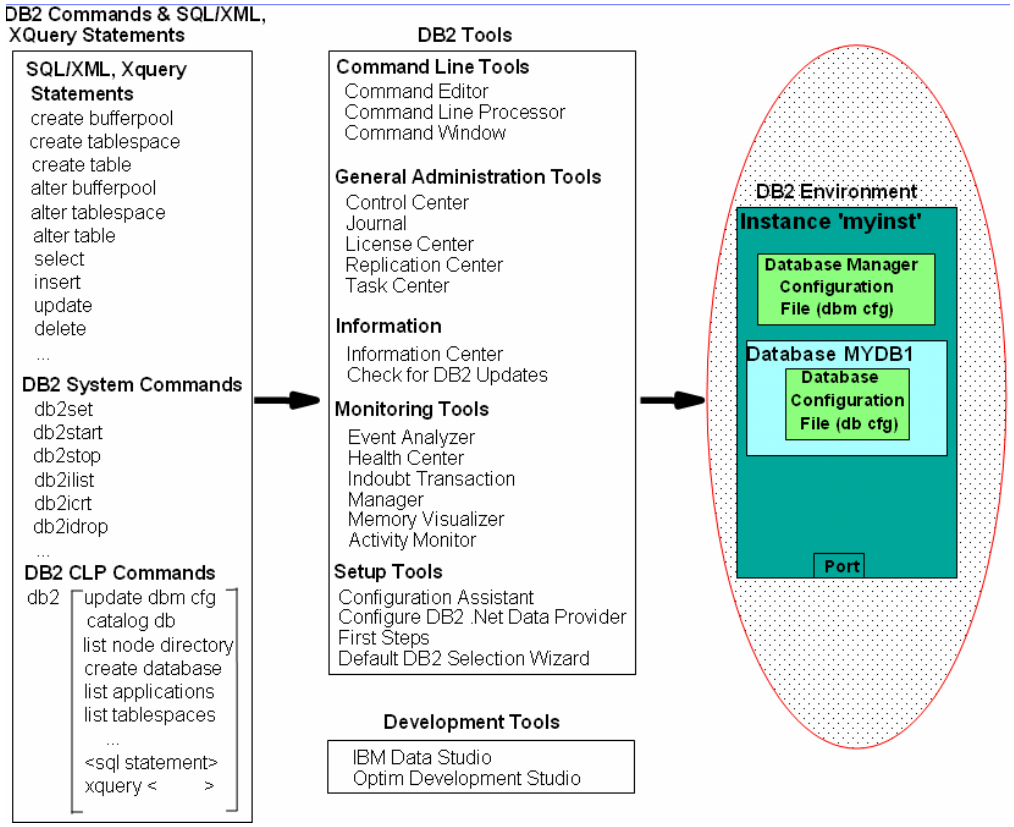


図 4.1 – DB2 の全体像: DB2 環境

DB2 環境を説明するため、その構成要素を 1 つひとつ図に追加しながら説明していきます。まず、DB2 Express-C 9.7 のインストールが完了した時点での DB2 データ・サーバーを図 4.2 に示します。

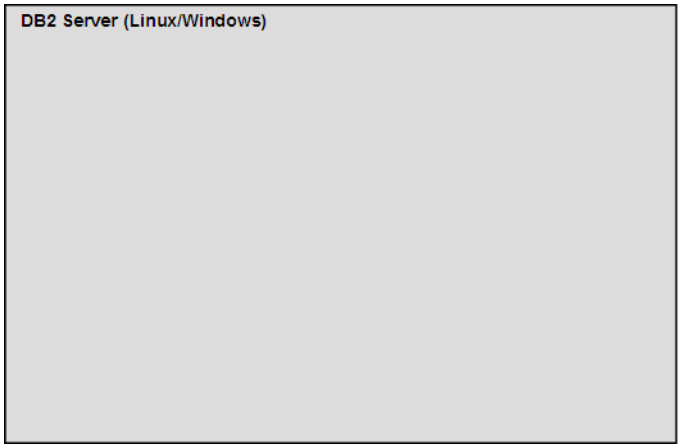


図 4.2 – DB2 Express-C 9.5 インストール後の DB2 サーバー

Windows ではインストールの一環として、**DB2** というデフォルト・インスタンス (Linux の場合は **db2inst1**) が作成されます。図 4.3 の左側に、このデフォルト・インスタンスを緑のボックスで示します。インスタンスとは単なる独立した環境のことで、この環境のなかで、アプリケーションを実行し、データベースを作成することができます。データ・サーバーでは複数のインスタンスを作成し、それぞれを異なる目的で使用することができます。例えば、あるインスタンスを本番環境で使用するためのデータベースとして使用し、別のインスタンスをテスト環境のデータベースとして使用し、さらに別のインスタンスを開発環境として使用するなどです。これらのインスタンスはすべて独立しています。つまり、あるインスタンスで操作を行っても、他のインスタンスには影響しません。

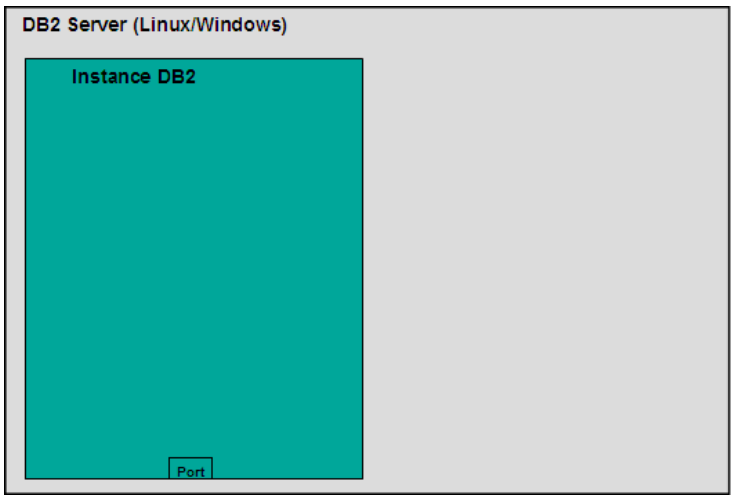


図 4.3 – デフォルトで作成された DB2 インスタンス

新しい DB2 インスタンスを作成するには、コマンド **db2icrt <instance name>** を実行します。ここで、**<instance name>** は任意の 8 文字からなるインスタンス名に置き換えてください。例えば ***myinst*** というインスタンスを作成する場合のコマンドは、**db2icrt myinst** となります。

図 4.4 の右側に、**myinst** という名前の新しいインスタンスを別の緑のボックスで示します。

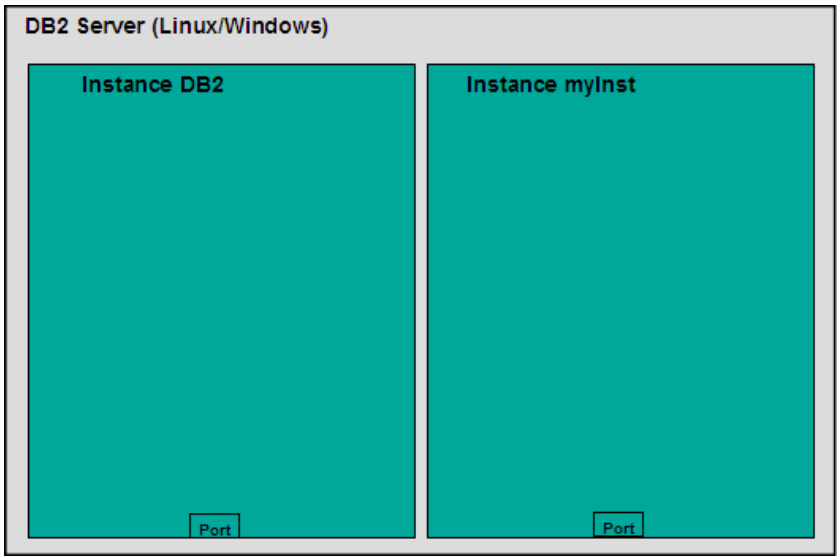


図 4.4 – 2 つのインスタンスを持つ DB2 サーバー

それぞれのインスタンスには固有のポート番号があります。そのため、リモート・クライアントから TCP/IP を使用して特定のインスタンス内にあるデータベースに接続するときには、その固有のポート番号によってインスタンスを区別することができます。DB2 インスタンスをアクティブなインスタンスにするには、DB2 コマンド・ウィンドウを使用します。Windows の場合には、以下のオペレーティング・システム・コマンドを実行します。

```
set db2instance=myinst
```

等号記号 (=) の前後には空白スペースを入れてはいけないので注意してください。この例の場合、コマンド・ウィンドウからデータベースを作成すると、新しいデータベースがインスタンス ***myinst*** 内に作成されます。

お使いのシステムのインスタンスをリストアップするには、以下のコマンドを実行します。

```
db2ilist
```

Linux では、インスタンスと Linux オペレーティング・システムのユーザーは一致していなければならないため、ユーザーを切り替えるだけでインスタンスが切り替わります。このユーザーは、インスタンス所有者と呼ばれます。他のインスタンスのインスタンス所有者になるにはログオフしてからログインするか、**su** コマンドを使用します。

表 4.1 に、インスタンス・レベルの便利なコマンドを記載します。

コマンド	説明
db2start	現行のインスタンスを開始する
db2stop	現行のインスタンスを停止する
db2icrt	新しいインスタンスを作成する
db2idrop	インスタンスをドロップする
db2ilist	システム上にあるすべてのインスタンスの一覧を表示する
db2 get instance	現行のアクティブ・インスタンスの一覧を表示する

表 4.1 - インスタンス・レベルの便利な DB2 コマンド

上記のコマンドのなかには、コントロール・センターから実行できるものもあります。コントロール・センターで、例えば「インスタンス」フォルダーを展開し、目的のインスタンスを右クリックすると、DB2 コマンド・ウィンドウから実行する **db2start** コマンドに相当する「開始」、あるいは **db2stop** コマンドに相当する「停止」を選択することができます (図 4.5 を参照)。

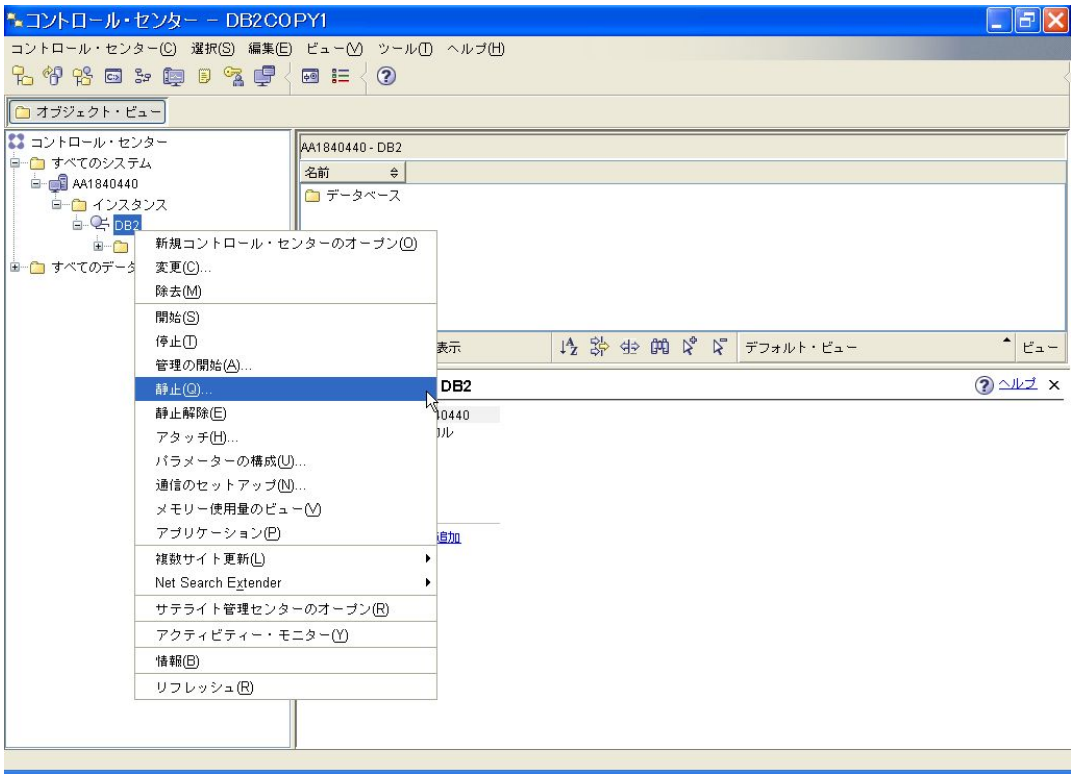


図 4.5 - コントロール・センターのインスタンス・コマンド

アクティブ・インスタンスのなかにデータベースを作成するには、DB2 コマンド・ウィンドウで以下のコマンドを実行します。

```
db2 create database mydb1
```

作成されたすべてのインスタンスの一覧を表示するには、以下のコマンドを実行します。

```
db2 list db directory
```

1 つのインスタンス内には、複数のデータベースを作成することができます。データベースは、表、ビュー、索引などのオブジェクトの集合です。データベースは独立した単位なので、データベース間でオブジェクトが共有されることはありません。図 4.6 に、インスタンス **DB2** 内に作成されたデータベース ***MYDB1*** を示します。

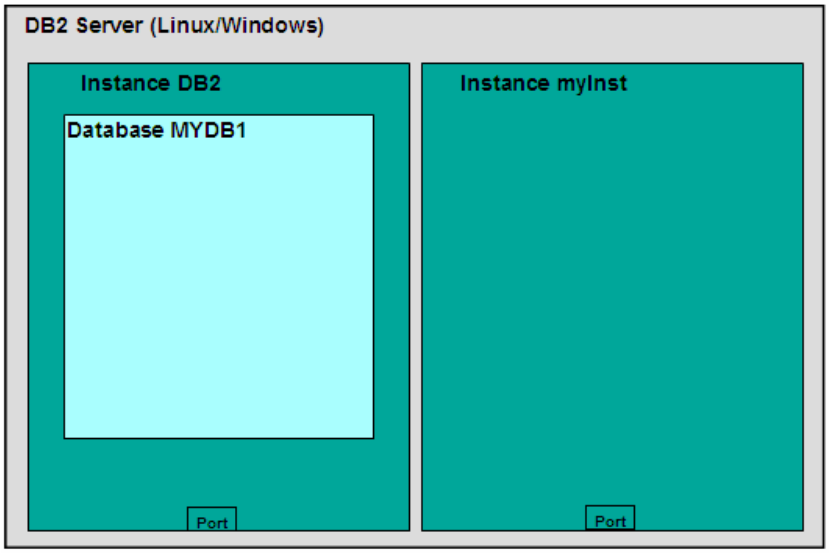


図 4.6 - インスタンス DB2 内に作成されたデータベース MYDB1

表 4.2 に、データベース・レベルで使用できるコマンドをいくつか記載します。

コマンド/SQL ステートメント	説明
`db2 create database`	新しいデータベースを作成する
`db2 drop database`	データベースをドロップする
`db2 connect to <database_name>`	データベースに接続する
`db2 create table/create view/create index`	表、ビュー、索引をそれぞれ作成する SQL ステートメント

表 4.2 - データベース・レベルでのコマンド/SQL ステートメント

同じ名前 (***MYDB1***) を設定した別のデータベースを ***myinst*** インスタンス内に作成する場合には、DB2 コマンド・ウィンドウで以下のコマンドを実行します。

```
db2 list db directory
set db2instance=myinst
db2 create database mydb1
set db2instance=db2
```

図 4.7 に、インスタンス ***myinst*** 内に新しく作成されたデータベース ***MYDB1*** を示します。

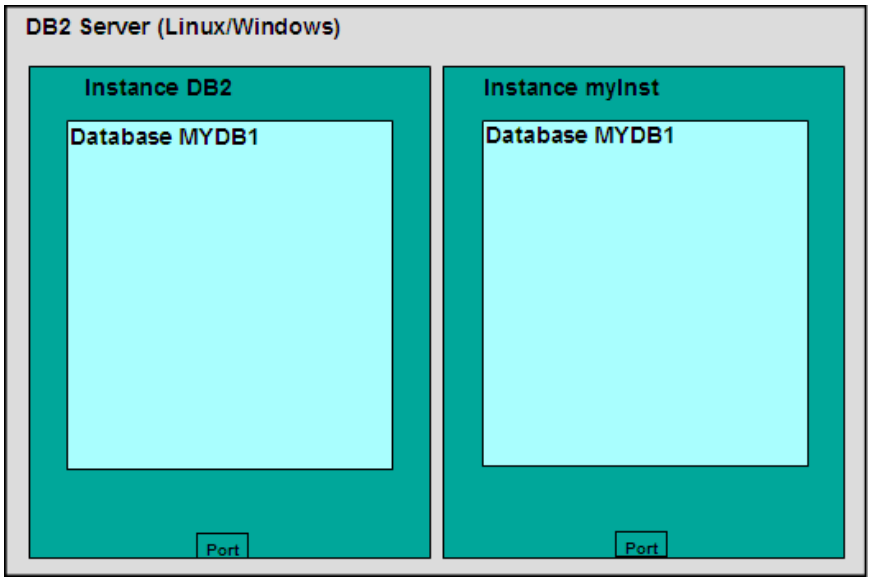


図 4.7 - インスタンス myInst 内に作成されたデータベース MYDB1

データベースを新規に作成すると、デフォルトでいくつかのオブジェクト (表スペース、表、バッファー・プール、ログ・ファイル) が作成されます。これらのオブジェクトを作成するには多少時間がかかるため、**`create database`** コマンドの処理には数分を要します。図 4.8 の左側に、デフォルトで作成された 3 つの表スペースを示します。表スペースについては第 6 章「DB2 アーキテクチャー」で詳しく説明しますが、とりあえずここでは、表スペースとは論理表と物理リソース (ディスクやメモリーなど) の間にある論理層のことであると考えてください。

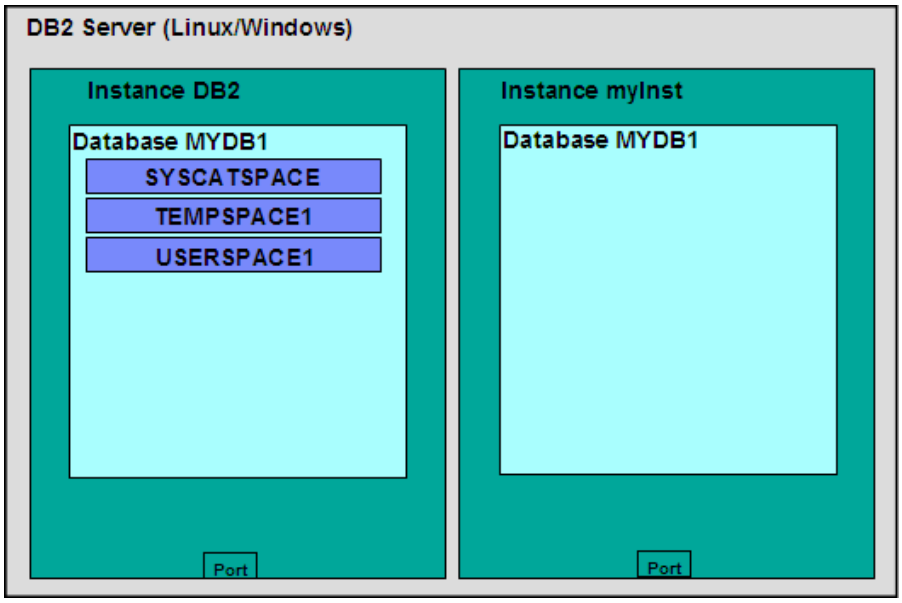


図 4.8 - データベースの作成時にデフォルトで作成された表スペース

表スペース **SYSCATSPACE** にはシステム・カタログ表が含まれます。システム・カタログ表は、他のリレーショナル·データベース管理システムではデータ・ディクショナリーと呼ばれています。この表には基本的に、変更したり削除したりするとデータベースが正常に機能しなくなるシステム情報が含まれています。表スペース **TEMPSPACE1** は、DB2 がソートなどの操作を実行するために追加のスペースが必要になったときに使用されます。表スペース **USERSPACE1** は通常、表の作成時に表スペースが指定されていない場合、ユーザー・データベース表を保管するために使用されます。

独自の表スペースを作成することもできます。その場合に使用するのは、**CREATE TABLESPACE** ステートメントです。図 4.9 に、DB2 インスタンスの ***MYDB1*** データベース内に作成された表スペース、***MYTBLS1*** を示します。表スペースを作成するときには、使用するディスクと使用するメモリー (バッファー・プール) を指定します。したがって、とても頻繁に使用される表がある場合には、表スペースに最も高速なディスクと最大量のメモリーを割り当てることができます。

デフォルトでは、**IBMDEFAULTBP** というバッファー・プール、そしてログ・ファイルも作成されます。図 4.9 に、この 2 つのオブジェクトを追加して示しています。

バッファー・プールは、基本的にはデータベースが使用するメモリー・キャッシュです。1 つ以上のバッファー・プールを作成することもできますが、そのうちの 1 つは必ず、既存の表スペースのページ・サイズと同じページ・サイズでなければなりません。ページとページ・サイズについては、第 6 章「DB2 アーキテクチャー」で詳しく説明します。

ログ・ファイルはリカバリーのために使用されます。データベースでの操作中は、そのデータベースの情報がディスクに保管されるだけでなく、データに対して実行されたすべての操作がログ・ファイルに記録されます。ログはいわば、「自動保存」操作が実行される一時ファイルのようなものです。ログについての詳細は、第 11 章「バックアップおよびリカバリー」で説明します。

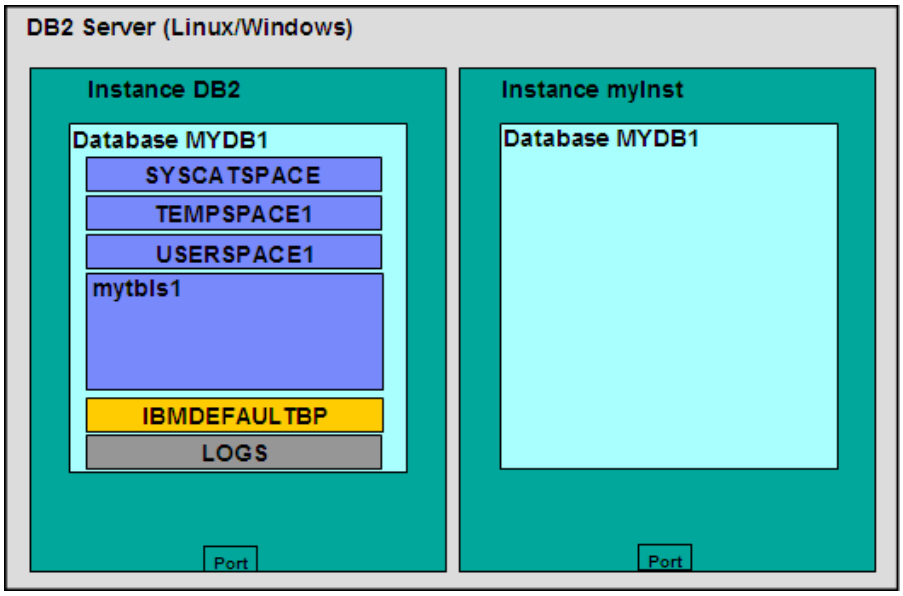


図 4.9 - デフォルトで作成されたバッファー・プールとログ

前に、インスタンスは独立した環境であることから、同じ名前のデータベースを複数のインスタンス内に作成することができると説明しました。インスタンスと同じく、データベースも独立した単位です。そのため、あるデータベース内のオブジェクトが別のデータベース内のオブジェクトと関係を持つことはありません。図 4.10 に、**DB2** インスタンス内の ***MYDB1*** データベースと ***SAMPLE*** データベースそれぞれに含まれる表スペース ***mytbls1*** を示します。このような構成が有効な理由は、データベースは独立した単位だからです。図 4.10 では表示領域が限られているため、**SAMPLE** データベース内のその他のデフォルト・オブジェクトを省略していることに注意してください。

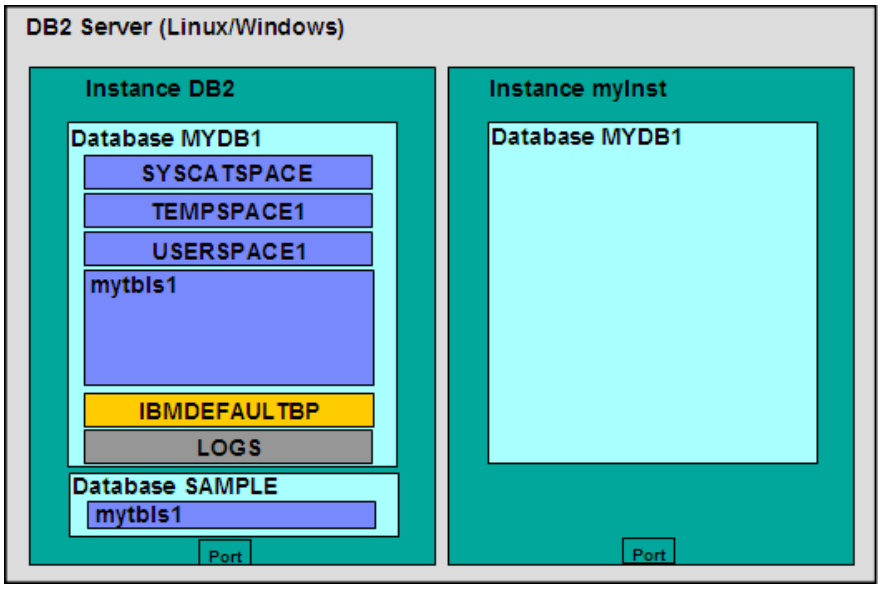


図 4.10 - 異なるデータベース内にある同じ名前の表スペース

表スペースの作成が完了すると、その表スペース内部に表、ビュー、索引などのオブジェクトを作成することができます (図 4.11 を参照)。

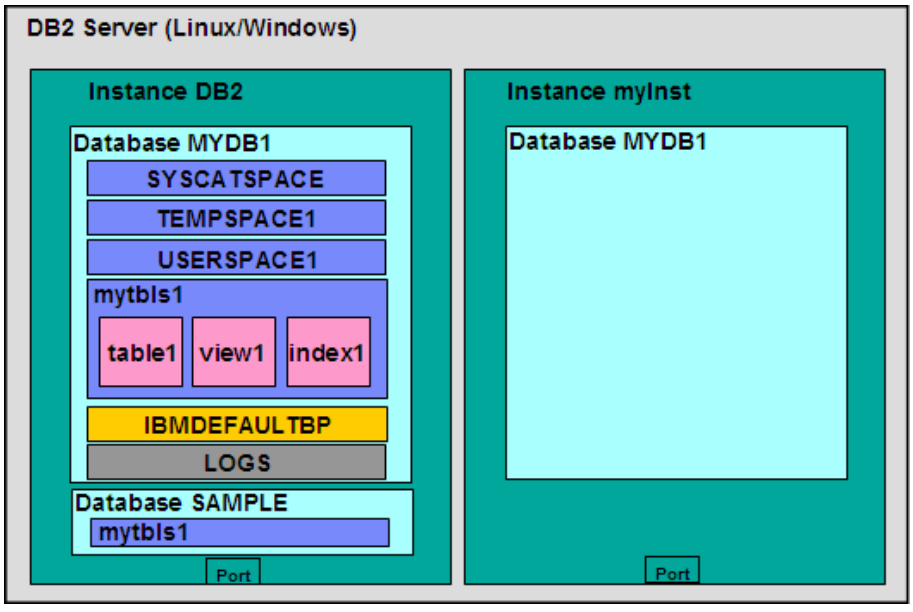


図 4.11 - 表スペース内に作成された表、ビュー、索引

4.1 DB2 の構成

DB2 のパラメーターは、「構成アドバイザー」ツールを使用して構成することができます。コントロール・センターから構成アドバイザーにアクセスするには、データベースを右クリックして「構成アドバイザー」を選択します。システム・リソースとワークロードについての質問に対する答えに応じて、構成アドバイザーが変更の必要がある DB2 のパラメーターと、それぞれのパラメーターに対する推奨値のリストを表示します。DB2 の構成について詳しく知りたい場合は、このまま読み進めてください。詳しい構成がわからなくても、この構成アドバイザーを使用すれば DB2 を十分使いこなせます。

DB2 サーバーは、以下の 4 つのレベルで構成することができます。:

- 環境変数
- データベース・マネージャー構成ファイル (dbm cfg)
- データベース構成ファイル (db cfg)
- DB2 プロファイル・レジストリー

図 4.12 にも上記の 4 つのレベルを示しました。この図のなかで、それぞれのレベルを表すボックスがどこに位置しているかに注目してください。例えば、環境変数はサーバーのオペレーティング・システム・レベルで設定される一方、データベース・マネージャー構成ファイルのパラメーターはインスタンス・レベルで設定されます。また、データベース構成パラメーターが設定されるのはデータベース・レベルであり、DB2 プロファイル・レジストリーが設定されるのはオペレーティング・システムまたはインスタンス・レベルのいずれかです。

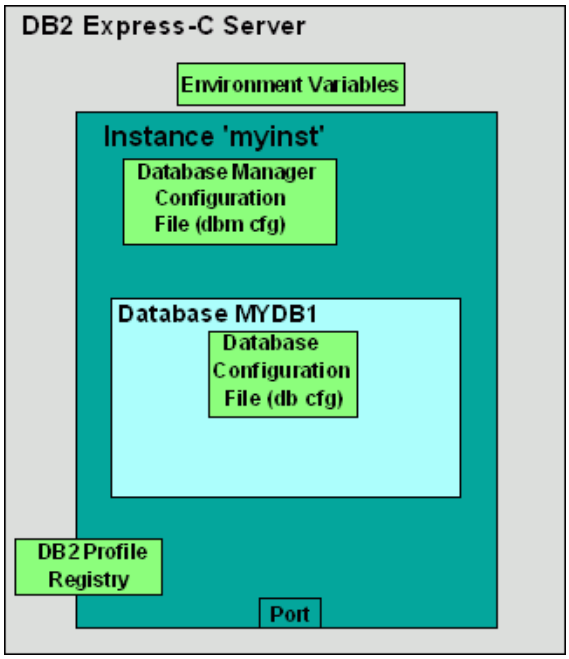


図 4.12 – DB2 構成

4.1.1 環境変数

環境変数は、オペレーティング・システム・レベルで設定される変数です。環境変数のうち重要なのは **DB2INSTANCE** で、この変数は現在作業対象としているアクティブ・インスタンス、つまりDB2 コマンドの適用対象となるインスタンスを示します。例えば Windows でアクティブ・インスタンスを ***myinst*** に設定するには、以下のオペレーティング・システム・コマンドを実行します。

```
set db2instance=myinst
```

4.1.2 データベース・マネージャー構成ファイル (dbm cfg)

データベース・マネージャー構成ファイル (dbm cfg) に組み込まれたパラメーターは、インスタンス、そしてインスタンスに含まれるすべてのデータベースに作用します。データベース・マネージャー構成ファイルは、コマンドラインまたは DB2 コントロール・センターを使用して表示したり、変更したりすることができます。

コントロール・センターから dbm cfg を操作するには、コントロール・センターでインスタンス・フォルダー内のインスタンス・オブジェクトを選択して右クリックします。するとポップアップ・メニューが表示されるので、そこから「パラメーターの構成」を選択します。この操作を図 4.13 に示します。

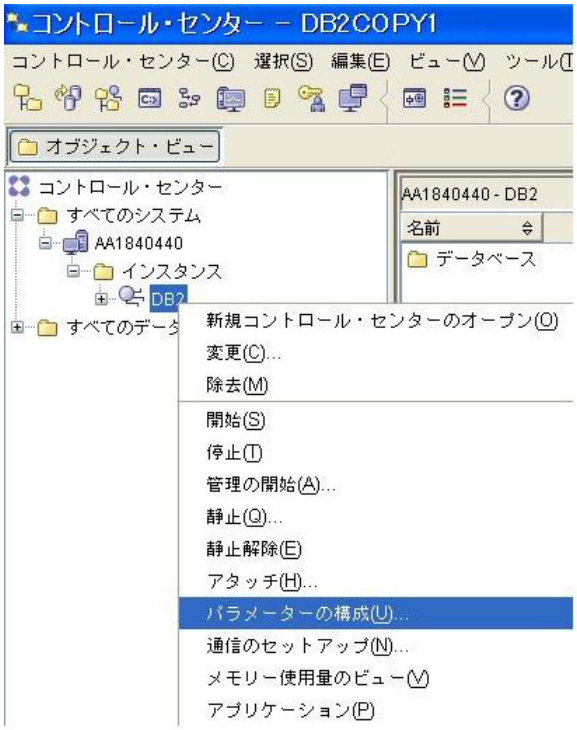


図 4.13 - コントロール・センターでの dbm cfg の構成

「パラメーターの構成」を選択すると、図 4.14 に示す画面に dbm cfg パラメーターの一覧が表示されます。

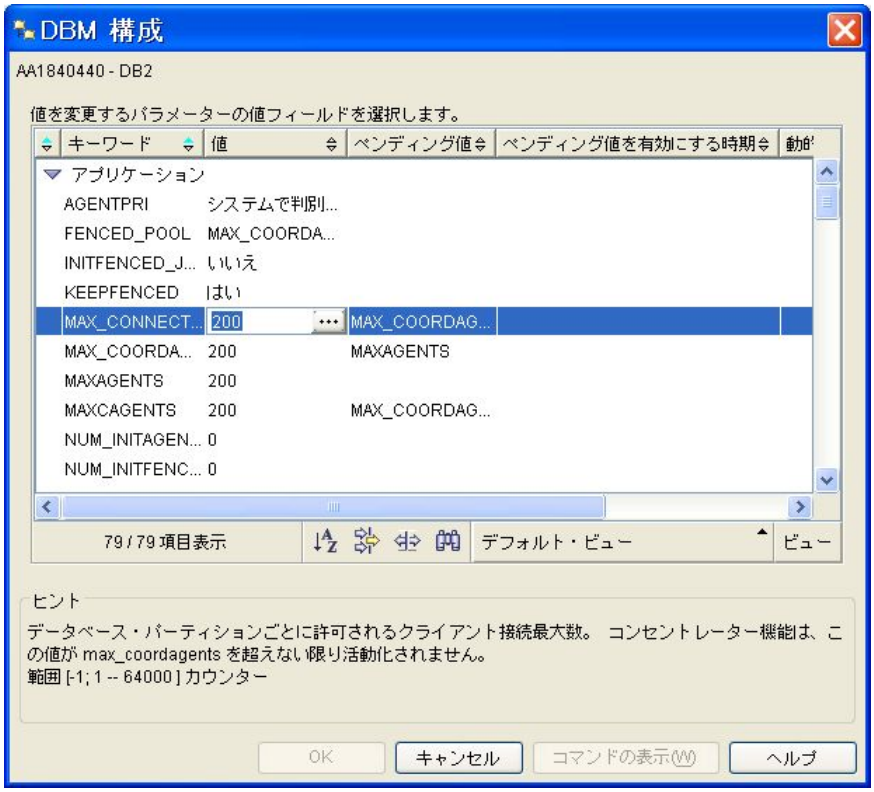


図 4.14 - dbm cfg ダイアログ

多くのパラメーターは動的です。つまり、パラメーターの変更はすぐに有効になります。ただし、パラメーターのなかにはいったんインスタンスを停止し、再び開始してからでないと、その変更が反映されないものもあります。インスタンスを停止し、再び開始するには、コマンドラインから **db2stop** コマンドと **db2start** コマンドを実行します。

すべてのアプリケーションを切断してからでないと、インスタンスを停止することはできません。インスタンスを強制停止させる場合は、**db2stop force** コマンドを使用します。

インスタンスをいったん停止するには、コントロール・センターでインスタンス・オブジェクトをクリックし、「停止」もしくは「開始」のいずれかを選択するという方法もあります。

表 4.3 に、コマンドラインから dbm cfg を管理する際に役立つコマンドをいくつか抜粋します。

コマンド	説明
`db2 get dbm cfg`	dbm cfg に関する情報を取得する
`db2 update dbm cfg using <parameter_name> <value>`	dbm cfg パラメーターの値を更新する

表 4.3 - dbm cfg の操作コマンド

4.1.3 データベース構成ファイル (db cfg)

データベース構成ファイル (db cfg) には、特定のデータベースに作用するパラメーターが組み込まれています。データベース構成ファイルは、コマンドラインまたは DB2 コントロール・センターを使用して表示したり、変更したりすることができます。

コントロール・センターから db cfg を操作するには、コントロール・センターでデータベース・フォルダー内のデータベース・オブジェクトを選択して右クリックします。するとポップアップ・メニューが表示されるので、そこから「パラメーターの構成」を選択します。この操作を図 4.15 に示します。

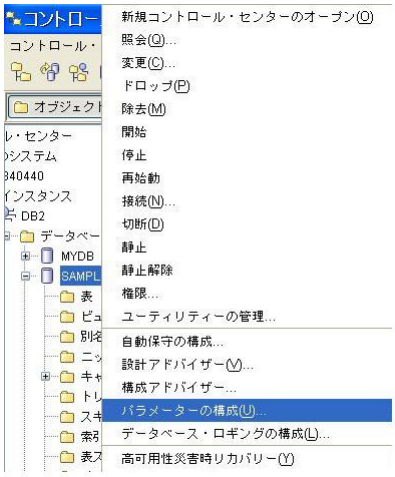


図 4.15 - コントロール・センターでの db cfg の構成

「パラメーターの構成」を選択すると、図 4.16 に示す画面に db cfg パラメーターの一覧が表示されます。

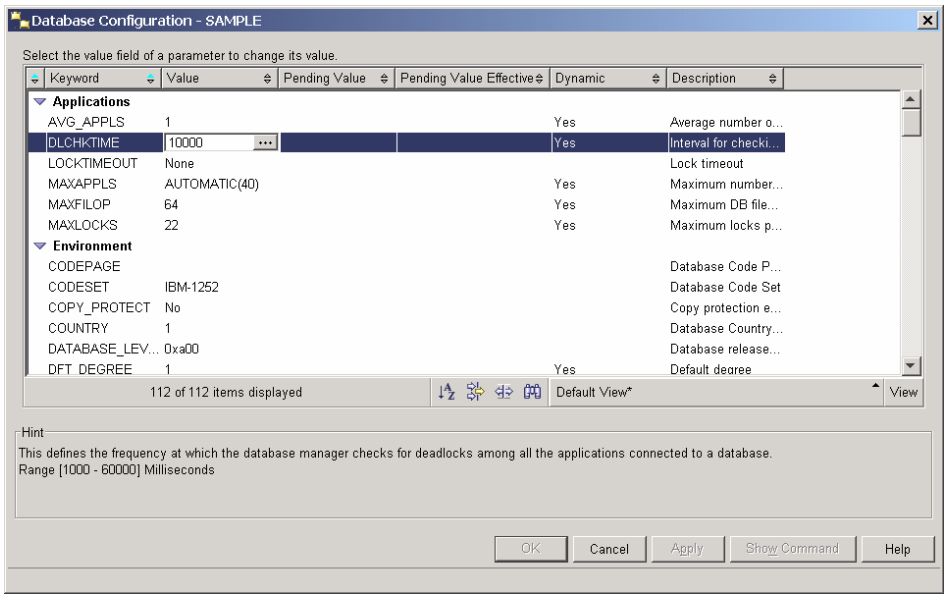


図 4.16 - db cfg

表 4.4 に、コマンドラインから db cfg を管理する際に役立つコマンドをいくつか抜粋します。

コマンド	説明
`get db cfg for <database_name>`	指定されたデータベースの db cfg に関する情報を取得する
`update db cfg for <database_name>` `using <parameter_name> <value>`	db cfg パラメーターの値を更新する

表 4.4 - db cfg の操作コマンド

4.1.4 DB2 プロファイル・レジストリー

DB2 プロファイル・レジストリー変数には、プラットフォームに固有のパラメーターが組み込まれています。これらのパラメーターは、すべてのインスタンスに作用するようにグローバルに設定することも、特定のインスタンスにのみ作用するようにインスタンス・レベルで設定することもできます。

表 4.5 に、DB2 プロファイル・レジストリーを操作する際に役立つコマンドをいくつか抜粋します。

コマンド	説明
`db2set -all`	現在設定されているすべての DB2 プロファイル・レジストリー変数のリストを表示する
`db2set -lr`	すべての DB2 プロファイル・レジストリー変数のリストを表示する
`db2set <parameter>=<value>`	パラメーターに指定された値を設定する

表 4.5 - DB2 プロファイル・レジストリーの操作コマンド

表 4.6 には、最もよく使用される DB2 レジストリー変数を示します。

レジストリー変数	説明
DB2COMM	データベース・マネージャーの起動時に起動させる通信マネージャーを指定します。
DB2_EXTSECURITY	Windows で、DB2 システム・ファイルをロックして DB2 への無許可アクセスを防ぎます。
DB2_COPY_NAME	現在使用中の DB2 コピーの名前を格納します。 インストールされた別の DB2 コピーに切り替えるには、**`installpath\bin\db2envars.bat`** コマンドを実行します。この変数は、DB2 コピーの切り替え目的で使用することはできません。

表 4.6 - よく使用される DB2 プロファイル・レジストリー変数

例えば、TCP/IP を使用した通信を許可するには、以下のように DB2COMM レジストリー変数を TCP/IP に設定します。

```
db2set db2comm=tcpip
```

4.2 The DB2 Administration Server

DB2 Administration Server (DAS) とは、リモート・クライアントが GUI を利用して DB2 サーバーを管理できるように DB2 サーバーで実行されるデーモン・プロセスのことです。DAS は DB2 グラフィカル･ツールを使用する場合にのみ、ローカルまたはリモートに必要になります。図 4.16 に示すように、DAS は実際のコンピューター 1 台あたりに 1 つしかありません。

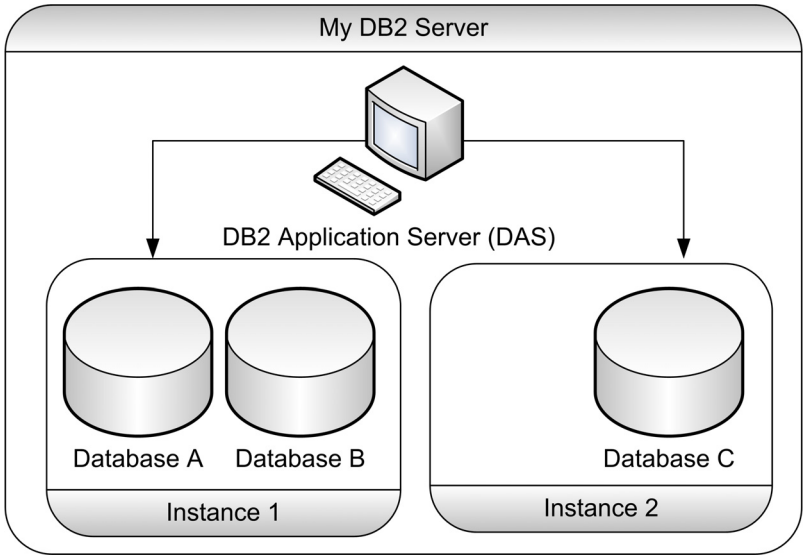


図 4.16 – DB2 Administration Server (DAS)

4.3 まとめ

この章では DB2 環境に焦点を当て、インスタンスおよびデータベースの概念と作成方法、そしてインスタンスとデータベースのそれぞれでよく使用するコマンドを説明しました。続いてインスタンスの他の重要な側面として、使用可能な 3 つのタイプの表スペース、表スペース内に作成できる表、ビュー、索引、そしてバッファー・プールとログについて説明しました。

最後に DB2 の構成と、構成を 4 つのレベル (環境変数、データベース・マネージャー構成ファイル、データベース構成ファイル、DB2 プロファイル・レジストリー) で変更する方法を説明しました。

4.4 演習　新しいデータベースの作成　インスタンス、データベース、構成の操作

このセクションの演習では、この章で説明した概念を復習するとともに、いくつかの DB2 ツールを紹介します。

パート 1: データベース作成ウィザードを使用したデータベースの作成

このパートでは、コントロール・センターのデータベース作成ウィザードを使用してデータベースを作成します。

手順

1. コントロール・センターの「オブジェクト・ビュー」ペインで、「すべてのデータベース」フォルダーを右クリックして「データベースの作成」を選択し、「自動保守」を選択します。この操作によって、データベース作成ウィザードが起動します。
2. ウィザードの「名前」ページで、データベースの名前と場所を指定します。以下の値を使用してください。

   「データベース名」：　EXPRESS

   「デフォルト・パス」(Windows の場合)：　C:

   「デフォルト・パス」(Linux の場合)：　/home/db2inst1

   「別名」：　空白のままにすると、デフォルトで EXPRESS に設定されます。

   「コメント」：　オプションなので、空白のままで構いません。

   「次へ」ボタンをクリックして、ウィザードの次のページに進みます。

new in V9.7

注: Windows のデフォルトでは、データベースを作成できるのはドライブ上にのみであり、パス上に作成することはできません。パス上にデータベースを作成したい場合には、DB2 レジストリー変数 DB2_CREATE_DB_ON_PATH を設定します。

3. 「ストレージ」ページでは何も変更せずに「次へ」をクリックします。
4. 「保守」ページでは、デフォルト設定 (「はい、 ...」) のままにして「次へ」をクリックします。
5. ウィザードの「タイミング」ページで、オフライン保守時間枠を指定します。毎週月曜日から木曜日までの毎日午前 1 時を開始時刻とする 6 時間の時間枠を構成し、「次へ」ボタンをクリックします。
6. ウィザードの「メール・サーバー」ページで、通知機能を構成します。DB2 では、問題や異常な状態が検出されると自動的に E メールまたはページャーへメッセージを送信することができます。この通知機能を構成する場合は、DB2 が E メールの送信に使用できる SMTP サーバーを指定してください。この演習では SMTP サーバーを使用していないため、この構成は空白のままにして「次へ」をクリックします。

ウィザードの「サマリー」ページで、選択したオプションを確認します。確認したら、「完了」ボタンをクリックするとデータベースの作成プロセスが開始されます。データベースの作成には通常、数分かかります。その間は、進行状況が表示されます。

パート 2: インスタンス、データベース、および構成の操作

このパートでは、Windows 上の DB2 サーバーで新しいインスタンスとデータベースを作成した後、構成パラメーターを変更します。この操作はコントロール・センターと DB2 コマンド・ウィンドウのどちらからでも実行できますが、ここでは DB2 コマンド・ウィンドウを使用した場合の手順を説明します。

手順

1. 「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「コマンド行ツール」 -> 「コマンド・ウィンドウ」の順に選択し、DB2 コマンド・ウィンドウを開きます。あるいは、「スタート」 -> 「ファイル名を指定して実行」の順に選択し、**db2cmd** と入力することで DB2 コマンド・ウィンドウを開くという簡単な方法もあります。
2. DB2 コマンド・ウィンドウから、***newinst*** という名前の新しいインスタンスを作成します。

   ```
   db2icrt newinst
   ```

3. ***newinst*** インスタンスに切り替え、インスタンスが実際に切り替わっていることを確認した後、インスタンスを開始します。

   ```
   set db2instance=newinst (注:「=」記号の前後にスペースはありません！)
   db2 get instance (これにより ***newinst*** が現行のインスタンスになります)
   db2start
   ```

4. ***newinst*** インスタンス内に、デフォルト値を使って ***newdb*** というデータベースを作成します。この作業で DB2 はデータベース内に内部オブジェクトを作成し、初期構成を行うので、作業が完了するまでに数分かかります。

   ```
   db2 create database newdb
   ```

5. 新しい ***newdb*** データベースに接続した後、接続を解除してドロップします。

   ```
   db2 connect to newdb
   db2 terminate
   db2 drop db newdb
   ```

6. 現行のインスタンス ***newinst*** を停止します。

   ```
   db2stop
   ```

7. サーバー内にあるすべてのインスタンスのリストを表示します。

   ```
   db2ilist
   ```

8. DB2 インスタンスに切り替え、インスタンスが実際に切り替わっていることを確認します。

```
set db2instance=db2
db2 get instance
```

9. ***newinst*** インスタンスをドロップします。

```
db2idrop newinst
```

10. dbm cfg パラメーター FEDERATED の現在の値を調べます。デフォルトでは、NO の値が設定されているはずです。

```
db2 get dbm cfg
```

ヒント: Linux では、`db2 get dbm cfg | grep FEDERATED` を使用することができます。

11. dbm cfg パラメーター FEDERATED の値を YES に変更し、値が変更されていることを確認します。

```
db2 update dbm cfg using FEDERATED YES
```

FEDERATED は動的パラメーターではないため、インスタンスを停止して開始してからでないと、変更は適用されません。ただし、インスタンスを停止する場合は、接続が存在しないことを確認する必要があります。確認方法 の 1 つは、以下のコマンドを実行することです。

```
db2 force applications all
db2 terminate
```

インスタンスを再起動し、`FEDERATED` の新しい値を確認します。

```
db2stop
db2start
db2 get dbm cfg
```

12. オペレーティング・システムで使用しているユーザー ID とパスワードを使用して SAMPLE データベースに接続します。

```
db2 connect to sample user <userID> using <password>
```

13. 現行のインスタンスで実行中のアプリケーションの数を確認します。

```
db2 list applications
```

14. 別の DB2 コマンド・ウィンドウを開き、今度はユーザー ID とパスワードを指定しないで SAMPLE データベースに接続します。その上で、現在の接続数を確認します。

```
db2 connect to sample
db2 list applications
```

15. DB2 コマンド・ウィンドウのいずれかから、接続を強制的に切断します。以下は、DBA が特定のユーザー (システム・リソースを独占していると考えられるユーザー) の作業を強制終了する例です。

    db2 force application (<application handle for db2bp.exe>)

    ここで、application handle は強制終了するアプリケーションの番号、つまり「ハンドル」です。この番号は、**db2 list applications** コマンドの出力から取得します。アプリケーション db2bp.exe は、DB2 コマンド・ウィンドウを表します。

16. DB2 コマンド・ウィンドウのいずれかでの接続が強制切断されたことを確認します。2 つの DB2 コマンド・ウィンドウのうち、どちらが強制切断されたかわからない場合は、以下のステートメントをそれぞれのコマンド・ウィンドウで実行します。

```
db2 select * from staff
```

    強制切断されたほうの DB2 コマンド・ウィンドウは、コード SQL1224N のエラー・メッセージを返すはずです。もう一方の DB2 コマンド・ウィンドウは、クエリーに対する出力を返します。

17. DAS をドロップして作成しなおした後、起動します。

```
db2admin stop
db2admin drop
db2admin create
db2admin start
```

18. レジストリー変数 DB2COMM の現在の値を調べます。

```
db2set -all
```

19. レジストリー変数 DB2COMM の設定を解除し、解除されたことを確認します。

```
db2set db2comm=
db2stop
```

    (ヒント: 接続が存在する場合は、db2stop の実行時にエラーを受け取ります。その場合にはどうしたらよいでしょうか。この問題を解決するには、前のステップを参照してください。)

```
db2start
db2set -all
```

20. DB2 レジストリー変数 DB2COMM をインスタンスの **tcpip** および **npipe** に設定し、新しい値を確認します。

```
db2set db2comm=tcpip,npipe
db2stop
db2start
db2set -all
```

21. db cfg パラメーター LOGSECOND の現在の値をチェックし、次にこの値を 5 に変更してから新しい値を確認します。

```
db2 connect to sample
db2 get db cfg
db2 update db cfg using LOGSECOND 5
db2 get db cfg
```

5

第 5 章 – DB2 ツール

new in V9.7

この章では、DB2 で使用できるツールをいくつか抜粋して説明します。DB2 9.7 では、この章で説明するツールの多くは現在非推奨とされているため、サポートはされていますが、今後拡張されることはなく、将来のリリースでは製品に組み込まれない可能性があります。これらのツールは IBM Data Studio に置き換えられます。

図 5.1 の赤い円で示しているのが、この章で焦点を当てる領域です。

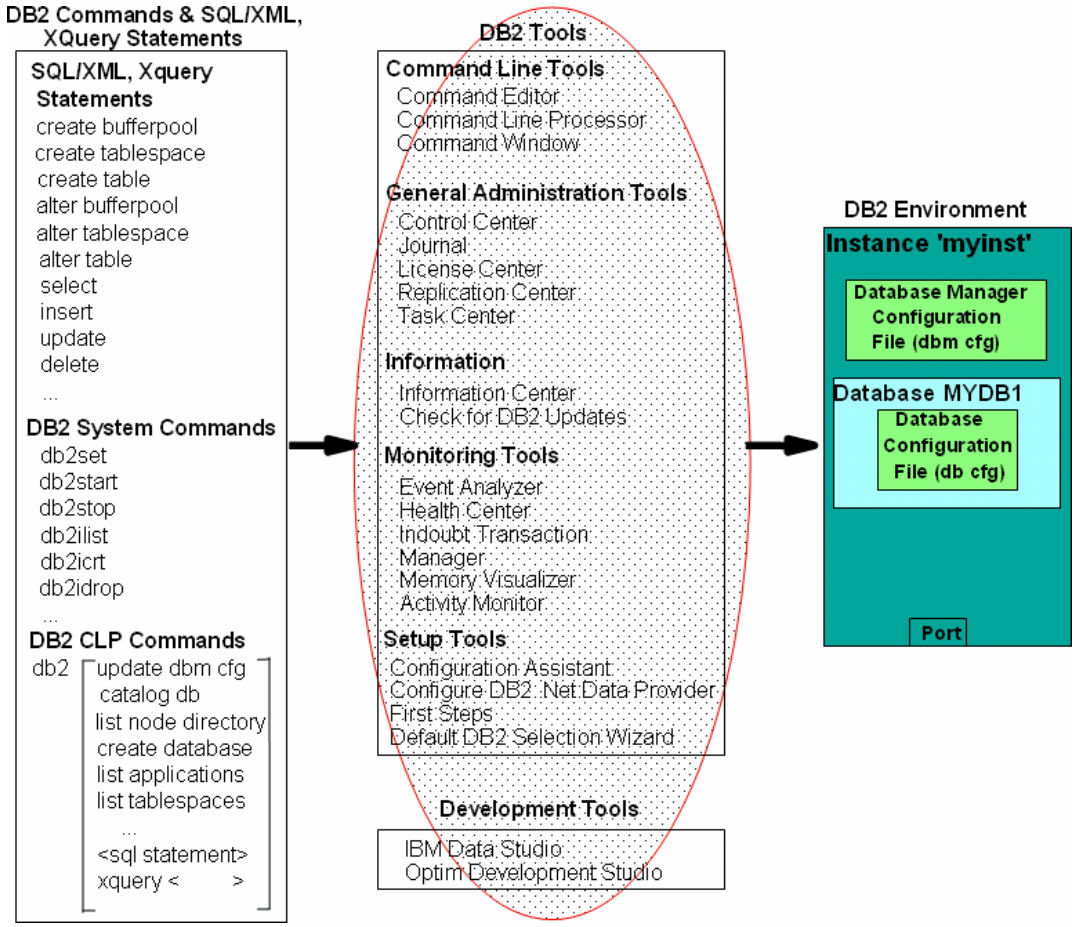


図 5.1 – DB2 の全体像: DB2 ツール

図 5.2 に、IBM DB2 スタート・メニューのショートカットから選択できるすべての DB2 ツールを示します。これらのツールのほとんどは、Linux と Windows に共通しています。

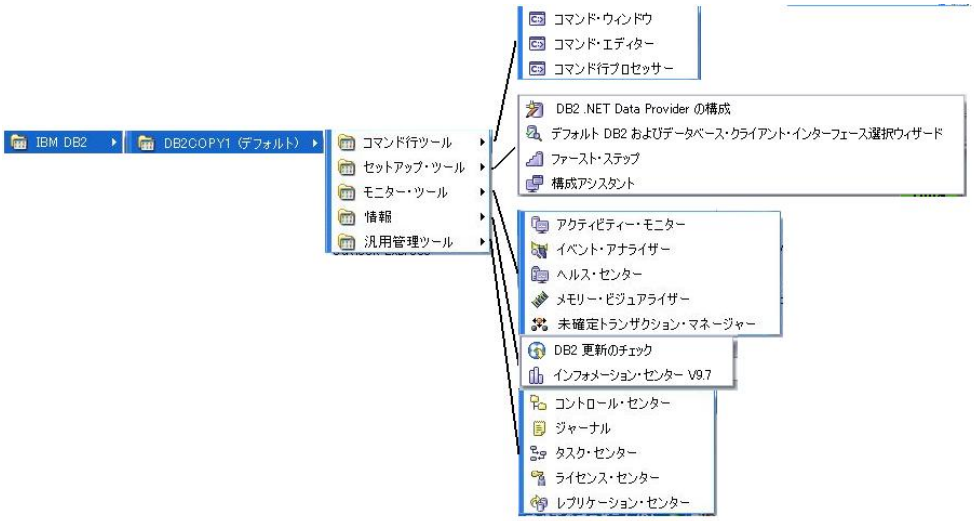


図 5.2 – IBM DB2 スタート・メニューからの DB2 ツール

DB2 ツールを起動させることができるショートカット・コマンドのうち、Linux または Windows で最もよく使用される DB2 ツールのコマンドを抜粋し、表 5.1 に一覧として示します。この表には、DB2 9.7 でどのツールが非推奨になったのかも示してあります。

ツール名	コマンド	非推奨
「コマンド・エディター」	db2ce	はい
「コマンド行プロセッサー」	db2	いいえ
「コマンド・ウィンドウ」(Windows プラットフォームのみ)	db2cmd	いいえ
「コントロール・センター」	db2cc	はい
「タスク・センター」	db2tc	はい
「ヘルス・センター」	db2hc	はい
「構成アシスタント」	db2ca	はい
「ファースト・ステップ」	db2fs	いいえ

表 5.1 – DB2 ツール (一部抜粋) を起動するためのショートカット・コマンド

new in V9.7

5.1 IBM Data Studio

DB2 9.7 では、DB2 によるデータベース管理およびデータベース開発に使用される主要なツールは、IBM Data Studio です。無料で使用できる Data Studio は、Linux およびWindows 上で実行することができ、IBM 製品の統合データ管理ポートフォリオの一部となっています。Data Studio の開発は、DB2 のリリースとは無関係に立てられたスケジュールに従って行われていますが、DB2 製品は可能な限りリリースのスケジュールが Data Studio と合うように調整を行っています。例えば、DB2 9.7 および IBM Data Studio 2.2 は 2009 年 6 月の同じ日にリリースされました。

図 5.3 に、IBM Data Studio の外観を示します。

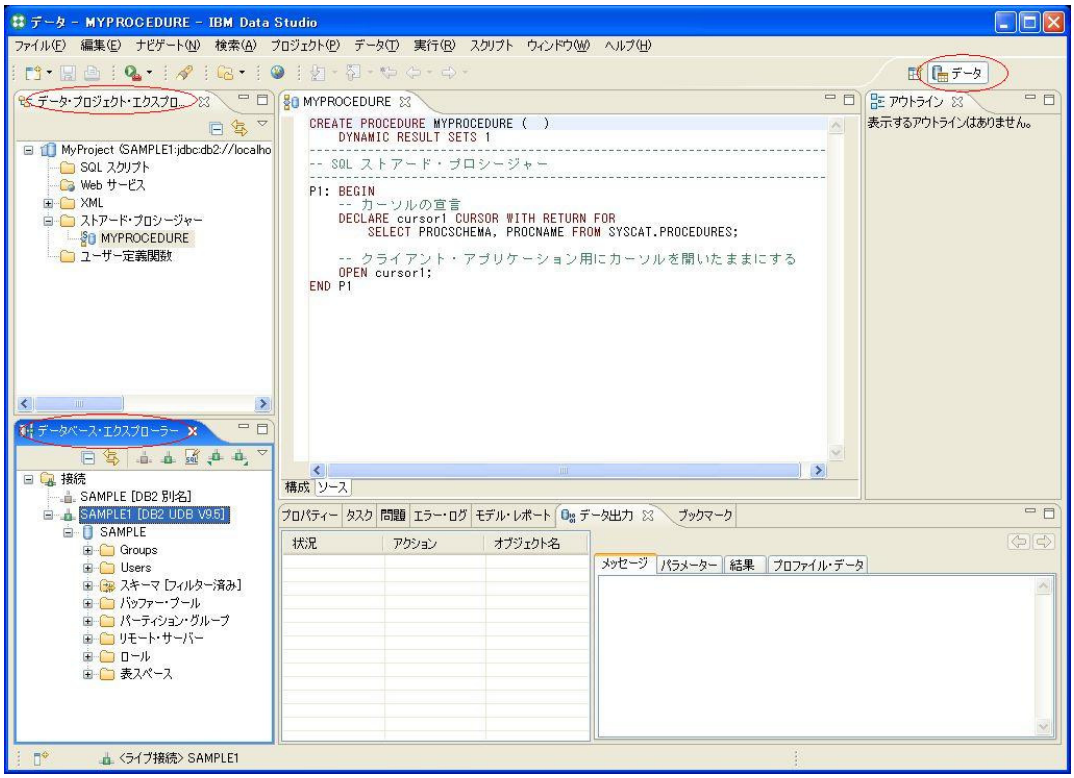


図 5.3 - IBM Data Studio

Eclipse を使い慣れている方は、Data Studio が Eclipse ベースであることに気付かれるはずです。Data Studio では一般に、「データ」パースペクティブ・ウィンドウ (図の右上隅に赤で丸を付けてあります) 内で作業を行います。また、Java プログラムを作成する場合は、「Java」パースペクティブに切り替えることもできます。図では、次の 2 つのビューにも赤で丸が付けてあります。

- データ・プロジェクト・エクスプローラー (左上)
- データ・ソース・エクスプローラー (左下)

「データ・ソース・エクスプローラー」ビューは、SQL スクリプト、XQuery、ストアード・プロシージャー、UDF、および Data Web サービスの作業を行うためにデータベース開発者が使用します。

「データ･ソース･エクスプローラー」ビューは、DB2 のインスタンスおよびデータベースを管理するためにデータベース管理者が使用します。このビューでは、コントロール・センターで以前に使用可能だった機能の大半を実行することができます。

上の図で、タイトルが PROCEDURE1 のビューは、「データ・プロジェクト・エクスプローラー」ビューで選択されているプロシージャー用のエディターです。実行しているタスクに応じて、エディターまたはその他のウィンドウが表示され、ここで他の構成をコーディングまたは実行することができます。

IBM Data Studio では、Informix などの他のデータ・サーバーを操作することも可能です。複数のデータ・サーバーを扱い、小規模の DBA またはデータベース開発者チームを持つ企業は、1 つのツール内からそれらのすべてを操作および管理できる利便性が得られます。

注:

Data Studio について詳しくは、本書のシリーズの一部である、無料のオンライン・ブック「Getting Started with Data Studio」を参照してください。

5.2 コントロール・センター

DB2 9.7 より前のバージョンではデータベースの管理で使用する主要な DB2 ツールは、図 5.4 に示すコントロール・センターでした。

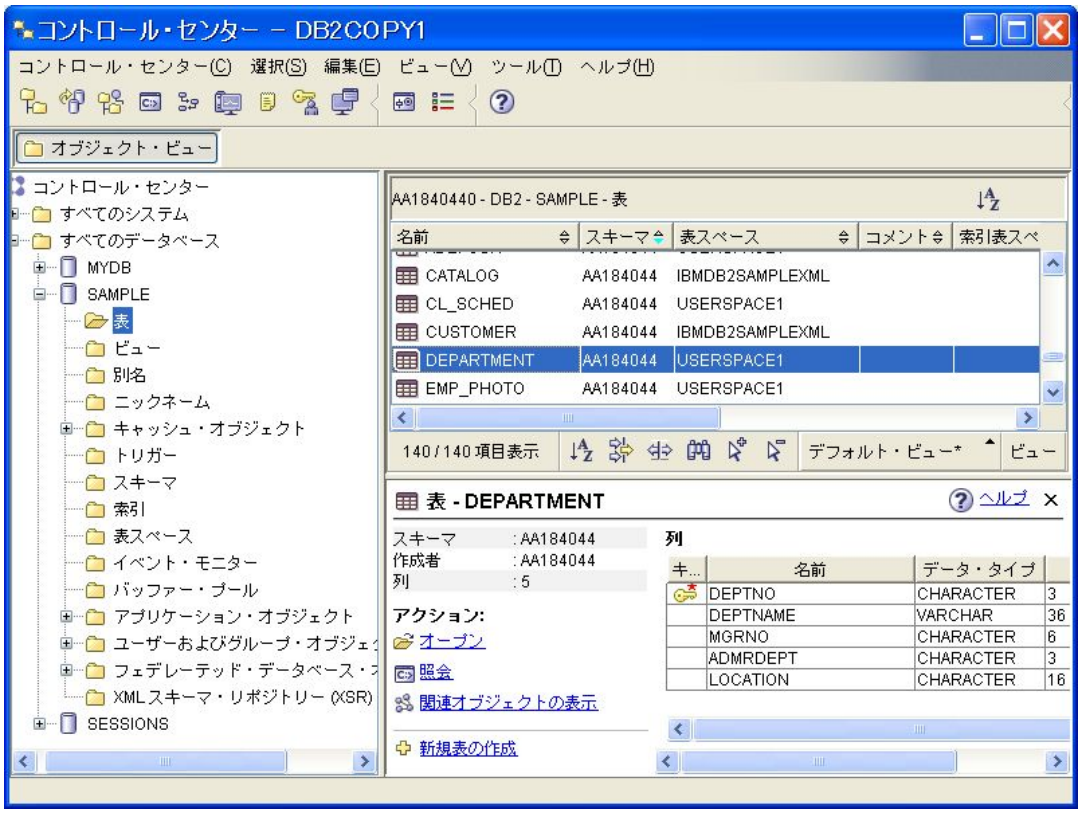


図 5.4 - DB2 のコントロール・センター

コントロール・センターは一元管理を行うためのツールで、このツールによって以下のことが可能になります。

- システム、インスタンス、データベース、およびデータベース・オブジェクトの表示
- データベースおよびデータベース・オブジェクトの作成、変更、管理
- 他の DB2 グラフィカル・ツールの起動

左側のペインには、システム上にあるデータベース・オブジェクトが「表」、「ビュー」などのフォルダーごとの階層で表示されます。「フォルダー」を選択すると、右上のペインにそのフォルダーに関連付けられたすべてのオブジェクトの一覧が表示されます。図 5.4 の例では「表」フォルダーを選択してあるため、**SAMPLE** データベース関連のすべての表が表示されています。右上のペインで特定の表を選択すると、右下のペインに選択した表に固有の詳細情報が表示されます。

オブジェクト・ツリーで別のフォルダーまたはオブジェクトを右クリックすると、そのフォルダーまたはオブジェクトに適用可能なメニューが表示されます。例えば、インスタンスを右クリックして「パラメーターの構成」を選択すると、データベース・マネージャー構成ファイルを表示して更新することができます。同様に、データベースを右クリックして「パラメーターの構成」を選択した場合には、データベース構成ファイルを表示、更新することができます。DB2 環境および構成パラメーターについては、第 5 章「DB2 環境」で詳しく説明しています。

初めてコントロール・センターを起動すると、使用するビューを選択するように求めるダイアログ・ボックスが表示されます。どのビューを選択するかによって、表示されるオプションとデータベース・オブジェクトのタイプが決まります。図 5.5 に、この「コントロール・センターの表示方法」ダイアログ・ボックスを示します。

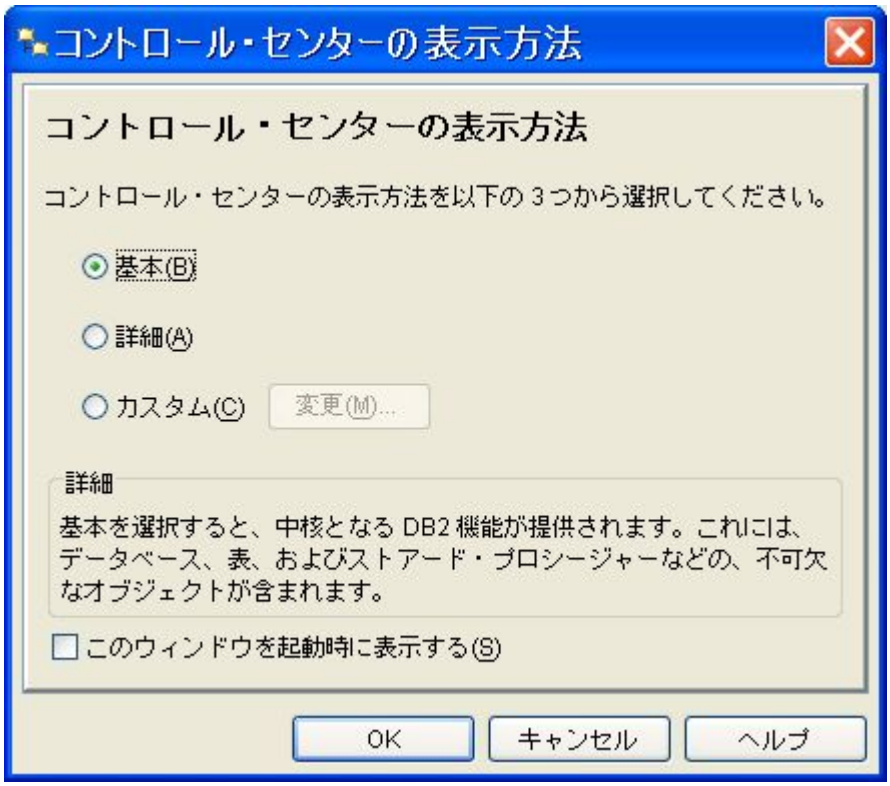


図 5.5 - DB2 のコントロール・センターの表示方法ダイアログ・ボックス

「基本」ビューには、DB2 のコア機能が表示されます。

「詳細」ビューには、さらに多くのオプションと機能が表示されます。

「カスタム」ビューには、特定の機能、オプション、オブジェクトを選択して表示することができます。

「コントロール・センターの表示方法」ダイアログ・ボックスは、「ツール」 -> 「コントロール・センターのカスタマイズ」の順に選択することによって再度表示することができます (図 5.6 を参照)。

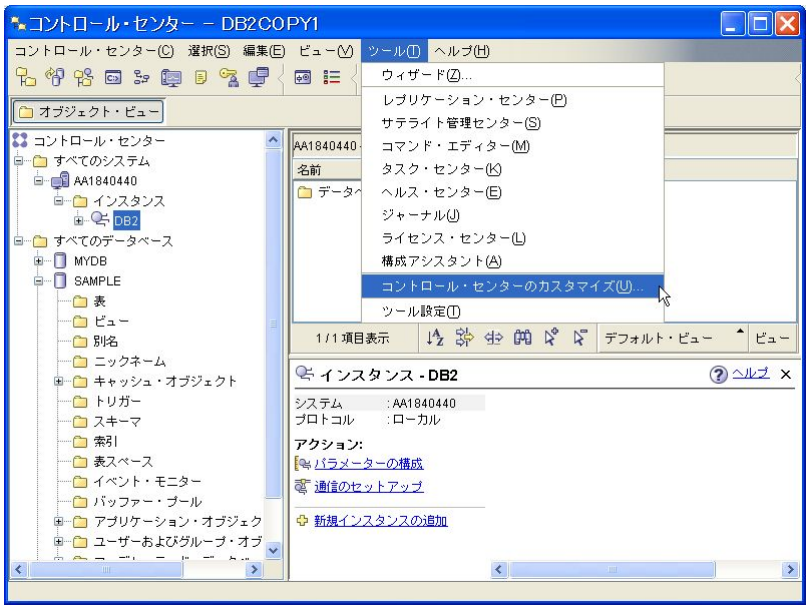


図 5.6 - コントロール・センターのカスタマイズ

5.2.1 コントロール・センターの起動方法

コントロール・センターを起動するには、以下のように複数の方法があります。:

- Windows の スタート・メニューからナビゲートする
- コマンド・プロンプトで **db2cc** を実行する
- コントロール・センター以外の DB2 GUI ツールで、ツールバーの「コントロール・センター」アイコン  をクリックする
- 図 5.7 に示す Windows システム・トレイの DB2 アイコンを使用する (緑色の DB2 アイコンを右クリックして「DB2 コントロール・センター」メニュー・オプションを選択)

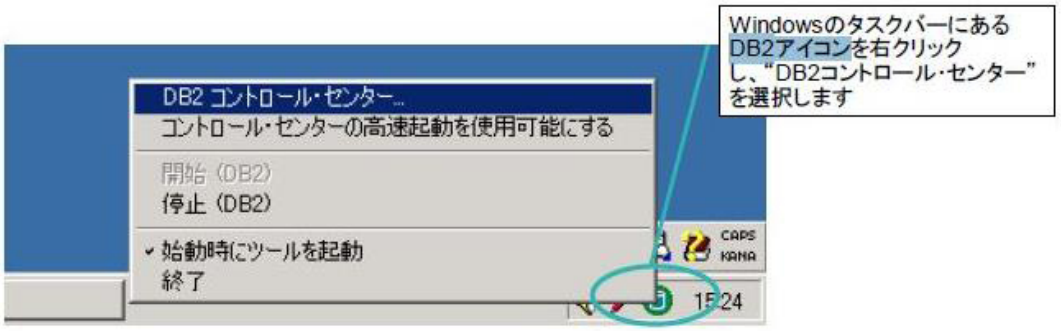


図 5.7 – Windows システム・トレイからの DB2 のコントロール・センターの起動

5.3 コマンド・エディター

DB2 コマンド・エディターでは、DB2 コマンドと SQL および XQuery ステートメントの実行、ステートメントの実行計画の分析、そしてクエリーの結果セットの表示または更新を行うことができます。図 5.8 で、コマンド・エディターの構成要素について説明します。

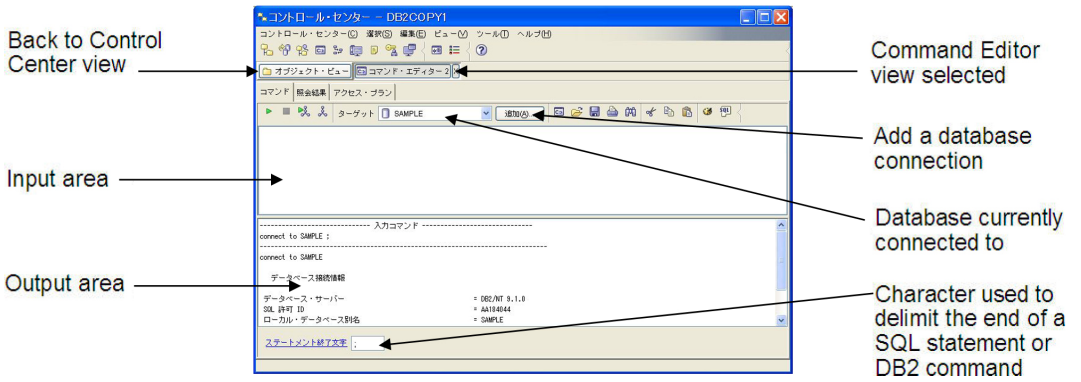


図 5.8 – DB2 コマンド・エディター

入力域には、複数のステートメントを入力することができます。その場合、各ステートメントは終了文字で終わっていなければなりません。実行ボタン (図 5.9 の一番左にあるボタン) をクリックすると、入力域に入力されたすべてのステートメントが順に実行されます。特定のステートメントを明示的に強調表示した場合には、強調表示されたステートメントのみが実行されます。SQL ステートメントを実行するにはデータベース接続が存在している必要がありますが、ステートメントのうちの 1 つを接続ステートメントにすることもできます。

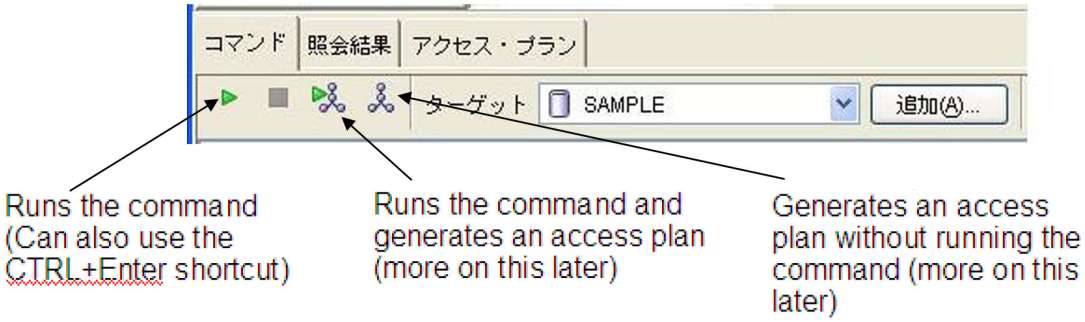


図 5.9 – コマンド・エディター– コマンドタブ

5.3.1 コマンド・エディターの起動方法

コマンド・エディターを起動するには、以下のいくつかの方法があります。

Windows スタート・メニューから以下の順に選択する

- 「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「コマンド行ツール」 -> 「コマンド・エディター」
- コマンド･プロンプトから **db2ce** と入力する

- コントロール・センターの「ツール」メニューを使用する
- コントロール・センターの組み込み機能を使用する
  - コントロール・センターのオブジェクト・ツリー・ペインで **SAMPLE** データベース・アイコンを右クリックし、「照会」メニュー項目を選択する
  - 照会可能なオブジェクト (データベース、表など) が選択されている状態で、コントロール・センターのオブジェクトの詳細ペインに表示された「照会」リンクをクリックしてコマンド・エディターを起動する
- コントロール・センターのツールバーで、図 5.10 に示す「コマンド・エディター」アイコン  をクリックする。

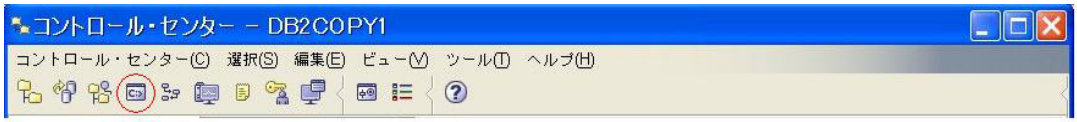


図 5.10 – コントロール・センターの「コマンド・エディター」アイコン

5.3.2 データベース接続の追加方法

データベースへの接続を追加するには、「追加」ボタンをクリックします (図 5.9 を参照)。すると、図 5.11 に示すダイアログが表示されます。

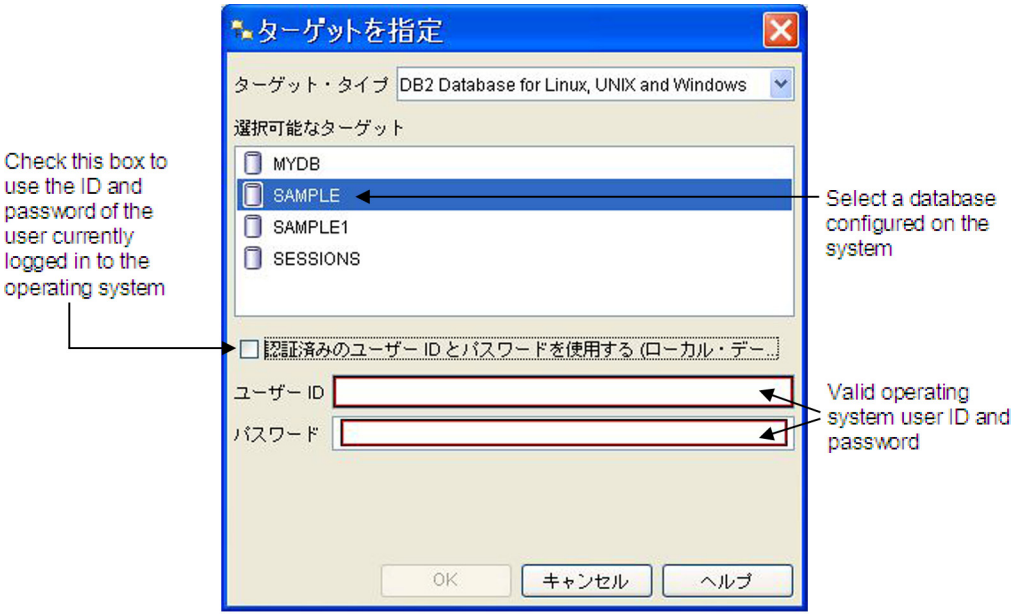


図 5.11 – データベース接続の追加

5.4 SQL Assist ウィザード

SQL 言語を使い慣れていない方のために、コマンド・エディターには SQL コードの生成を支援する SQL Assist ウィザードが用意されています。コマンド・エディターからこのウィザードを起動するには、図 5.12 に示す SQL のシンボルが付いた右端のアイコンをクリックします。このアイコンは、データベースに接続してから出ないと表示されません。

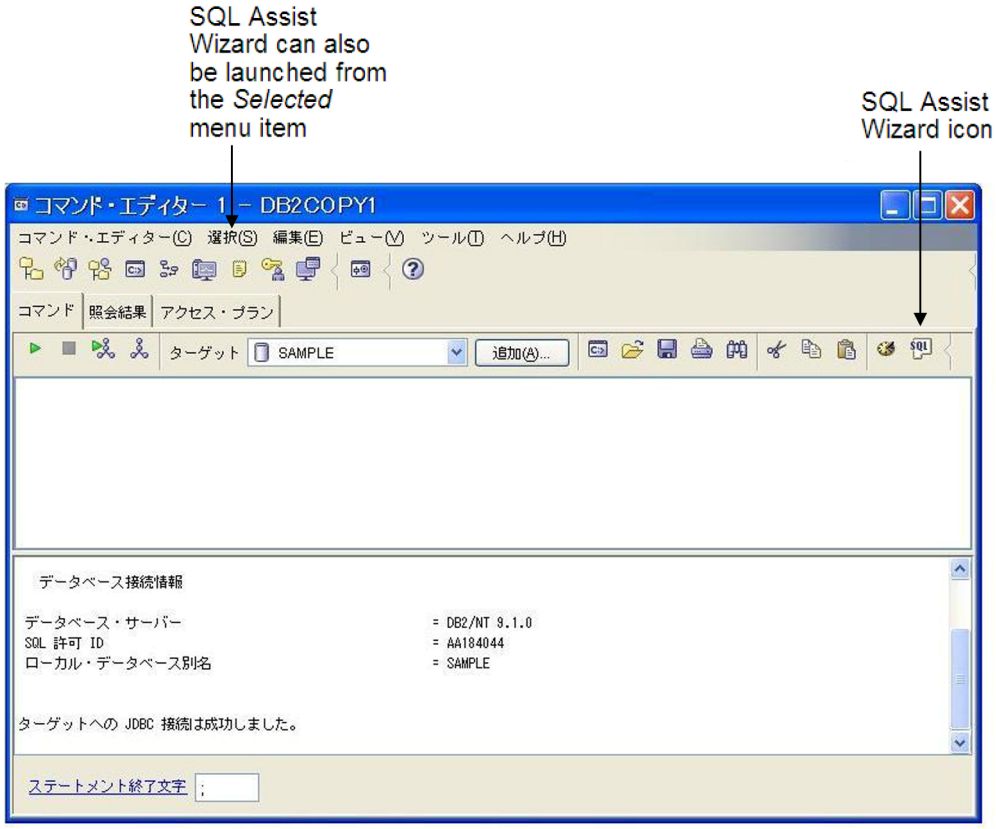


図 5.12 – SQL Assist ウィザードの起動

図 5.13 に SQL Assist ウィザードを示します。このウィザードの使用方法は至って簡単です。まず、支援が必要な SQL ステートメントのタイプ (SELECT、INSERT、UPDATE、DELETE) を指定します。選択したステートメントによって、表示されるオプションは異なります。ウィンドウの下部では、ウィザードで選択した項目に従って作成される SQL ステートメントを確認することができます。

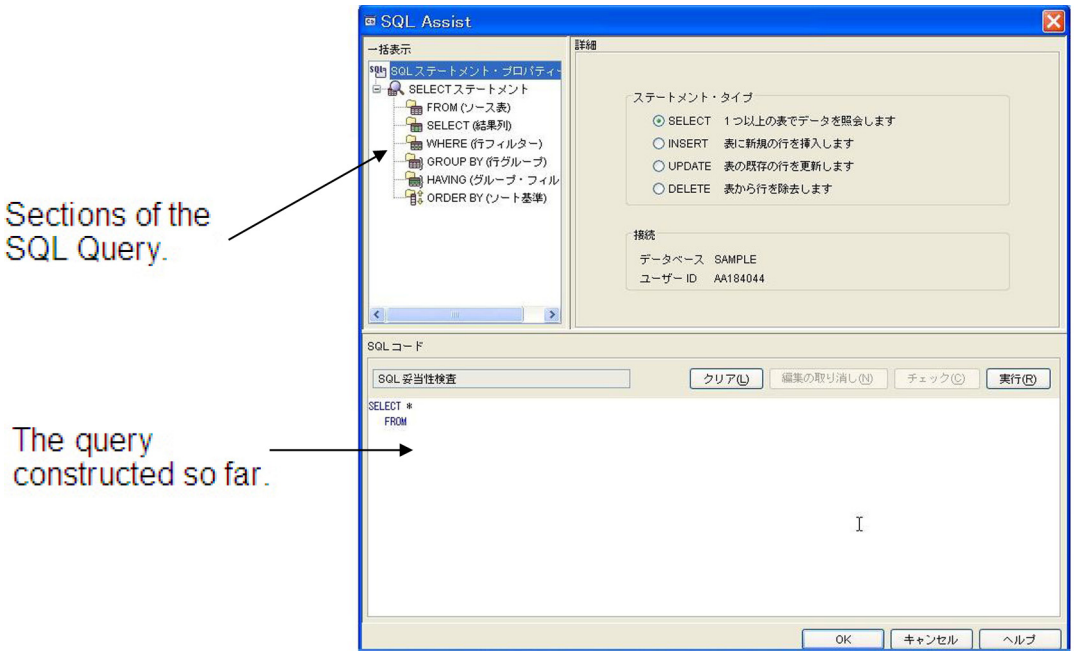


図 5.13 – SQL Assist ウィザード

5.5 「SQL 表示」ボタン

DB2 のほとんどの GUI ツールとウィザードでは、ツールやウィザードを使用してアクションを実行した後に、その結果として作成された実際のコマンドまたは SQL ステートメントを確認できるようになっています。結果を表示するには、操作中のツールに表示された「SQL 表示」ボタンをクリックします (図 5.14、図 5.15 を参照)。

図 5.14 – SQL 表示ボタン

図 5.15 – SQL 表示ボタンの出力

SQL ステートメントとコマンドを確認できるこの機能は、SQL 構文を学ぶ上でも、コマンドまたはステートメントを後で使用できるようにファイルに保存する上でも非常に重宝します。また、生成されたコマンドとステートメントを再利用してスクリプトを作成することもできます。

5.6 タスク・センター

「タスク・センター」GUI ツールは、タスクを作成する場合に使用します。タスクとは、DB2 コマンド、オペレーティング・システム・コマンド、またはスクリプトを実行する操作を 1 つにまとめたものです。タスクの失敗または成功に応じて、タスクに続くアクションを実行することができます。例えば、午前 3 時に重要なデータベースをバックアップするというタスクがあるとします。このバックアップ・タスクが成功した場合には DBA に E メールを送信してこの情報を通知し、失敗した場合にはタスク・センターを介して DBA に通知するといったアクションです。図 5.16 に、タスク・センターを示します。

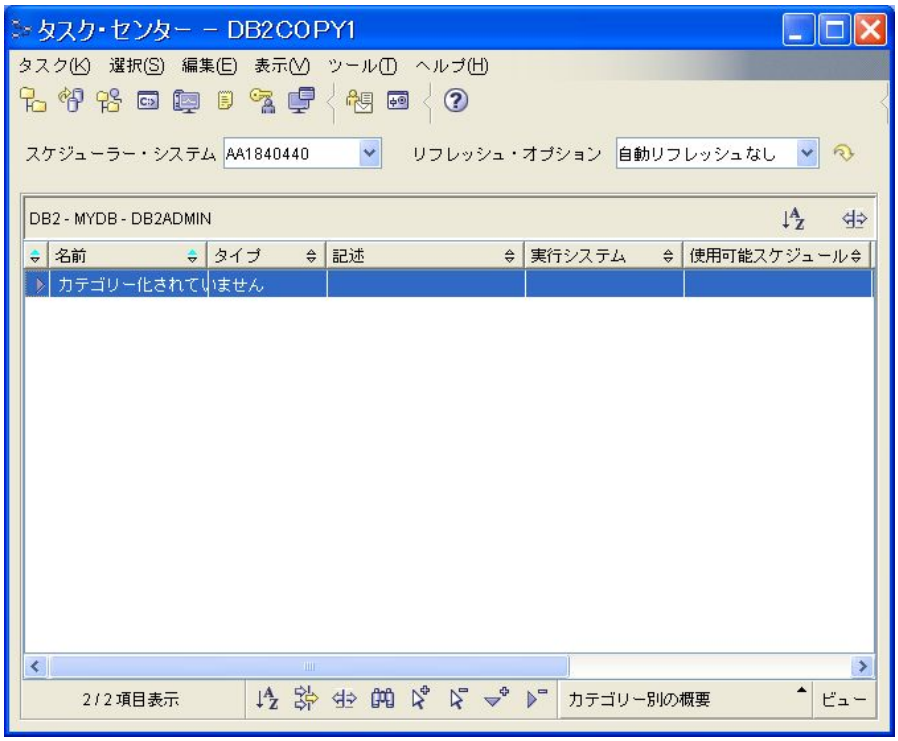


図 5.16 – タスク・センター

5.6.1 ツールカタログ・データベース

タスクとタスクのスケジューリングに関する詳細のすべては、ツールカタログ・データベースという別個の DB2 データベースに保管されます。タスクをスケジューリングするためには、あらかじめこのデータベースが作成されていなければなりません。ツールカタログ・データベースを作成するには、例えば以下のコマンドを使用します。

```
CREATE TOOLS CATALOG systools CREATE NEW DATABASE toolsdb
```

上記の例では、***systools*** がデータベースに含まれるすべての表のスキーマ名で、***toolsdb*** がデータベース名です。スキーマについての詳細は、第 8 章「データベース・オブジェクトの操作」で説明します。

5.6.1.1 タスク・センターの起動方法

タスク・センターを起動するには、コントロール・センターで「ツール」 -> 「タスク・センター」の順にクリックします (図 5.16 を参照)。または、Windows の「スタート」メニューを使用して、「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「汎用管理ツール」 -> 「タスク・センター」の順に選択するという方法でもこのツールを起動することができます。

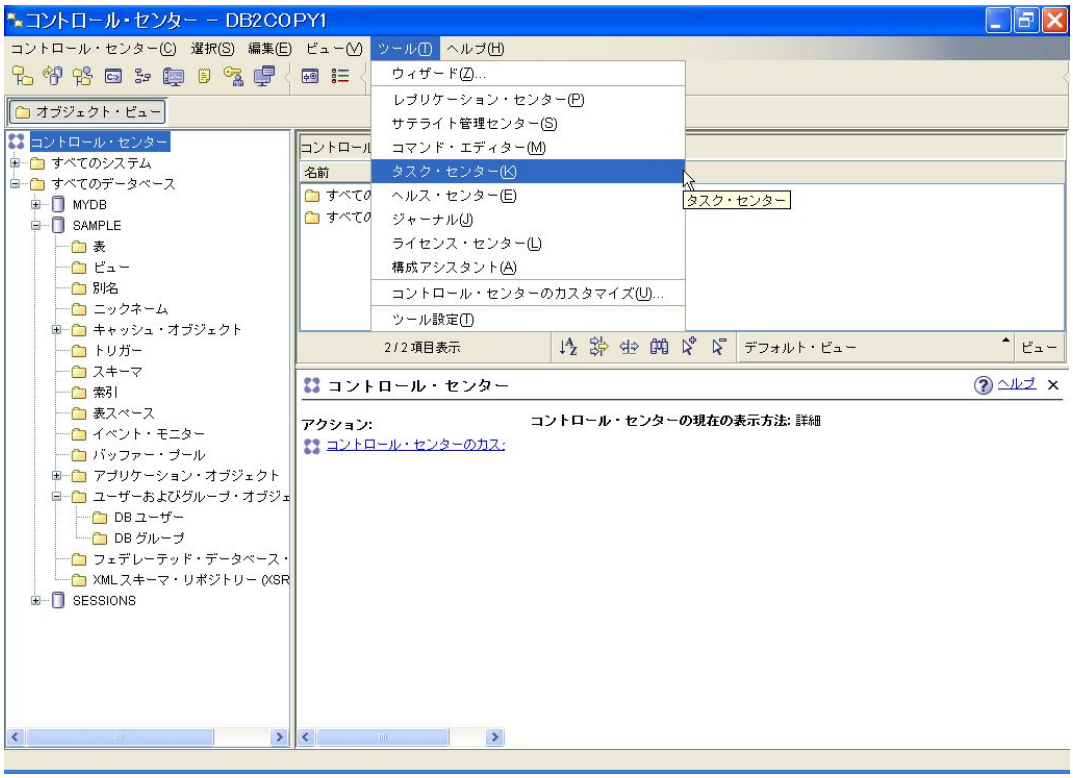


図 5.17 – タスク・センターの起動

5.6.1.2 タスク・センターでのスケジューリング

タスク・センターでは、DB2 GUI ツールで作成されたスクリプトであるかどうかに関係なく、あらゆるタイプのスクリプトをスケジューリングすることができます。タスクはスケジューリングされた時刻になると、ツールカタログ・データベースを作成したシステムから実行されます。タスク・センターをいろいろと調べてみてください。簡単にタスクを作成できることがわかるはずです。

5.7 ジャーナル

DB2 「ジャーナル」 GUI ツールは、DBA に対してアクティビティーのジャーナルをオンラインで提供します。図 5.18 にジャーナルを示します。表 5.2 には、ジャーナルから取得できる情報について説明します。

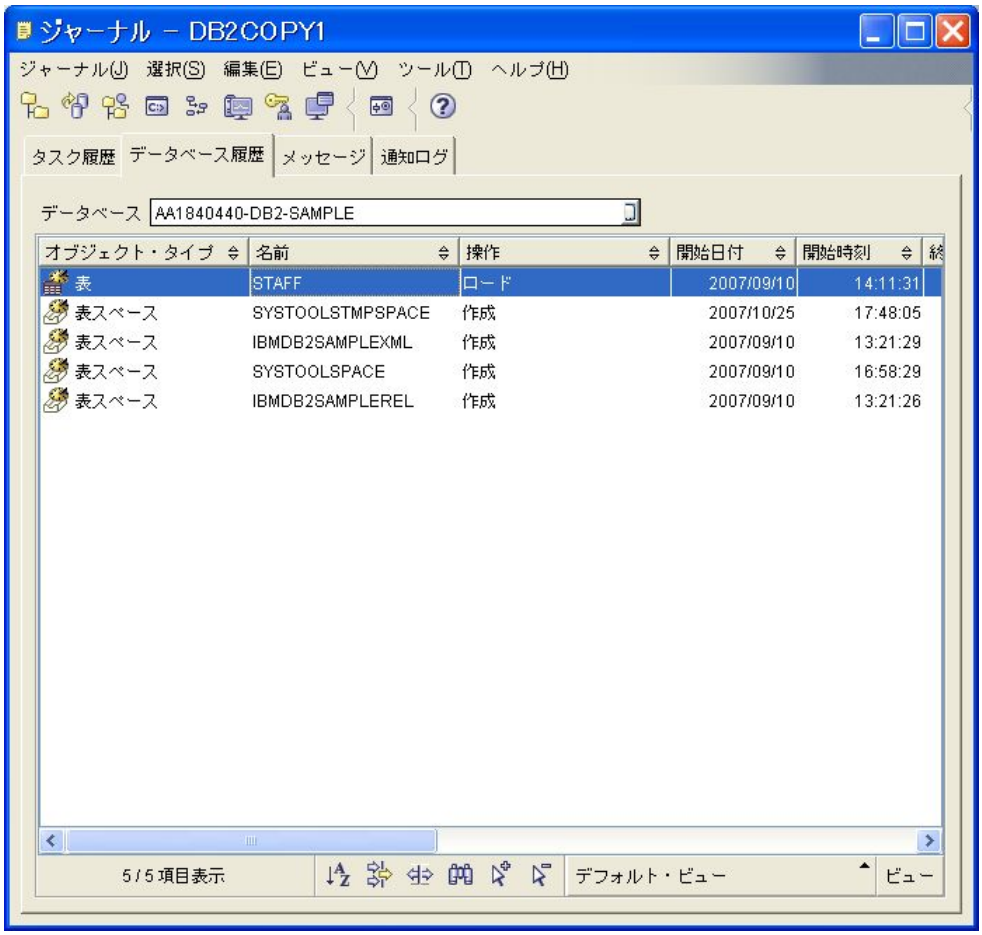


図 5.18 – ジャーナル

情報のタイプ	説明
「タスク履歴」	スケジューリングされたタスクのなかで試行されたすべてのタスクと、それぞれのタスクが成功したかどうかの状況
「データベース履歴」	実行されたデータベース・アクティビティー (バックアップ、リストア、REORG など) のレコード
「メッセージ」	DB2 ツールによって返されたメッセージの履歴。以前のエラー・メッセージを呼び起こして比較する場合、またはダイアログ・ボックスをよく読む前に閉じてしまったり、誤って閉じてしまった場合に役立ちます。

「通知ログ」	システム・レベルのメッセージを表示。重大なエラーはここに記録されます。

表 5.2 – ジャーナルに表示される情報

5.7.1 ジャーナルの起動方法

ジャーナルを起動するには、コントロール・センターで「ツール」 -> 「ジャーナル」の順にクリックします (図 5.19 を参照)。または、Windows の「スタート」メニューを使用して、「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「汎用管理ツール」 -> 「ジャーナル」の順に選択するという方法でもツールを起動することができます。

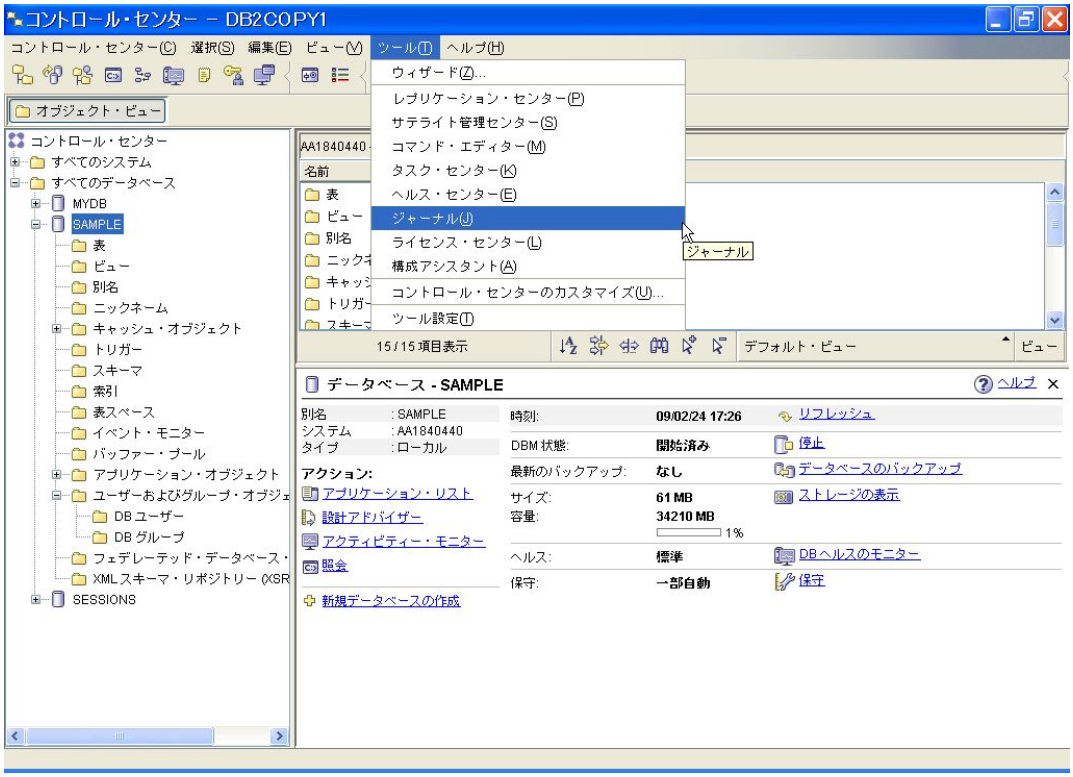


図 5.19 - ジャーナルの起動

5.8 ヘルス・モニター

ヘルス・モニターは DB2 エンジン内で実行されるデフォルト・エージェントで、データベースの状況に関するあらゆる側面 (メモリー、スペース管理、事前に定義された自動アクティビティーなど) をモニターします。DB2 の何らかの部分が、設定されたパラメーターの範囲から外れると、例外が発生して DBA に通知されます。アラート状態のタイプには以下の 3 つがあります。

- アテンション: 正常ではない状態を示します。
- 警告: 重大な問題ではなく、即時に対処する必要はありませんが、システムが最適な状態ではないことを示します。
- アラーム: 即時に対処する必要がある重大な問題があることを示します。

ヘルス・モニターのオン/オフは、データベース・マネージャー構成パラメーター HEALTH_MON で切り替えることができます。

5.8.1 ヘルス・センター

ヘルス・センターは、ヘルス・モニターを操作するためのグラフィカル・ツールです。ヘルス・センターには、システム上のヘルス・アラームがインスタンス、データベース、表スペース・レベル別に分類されて表示されます。図 5.20 に、ヘルス・センターを示します。

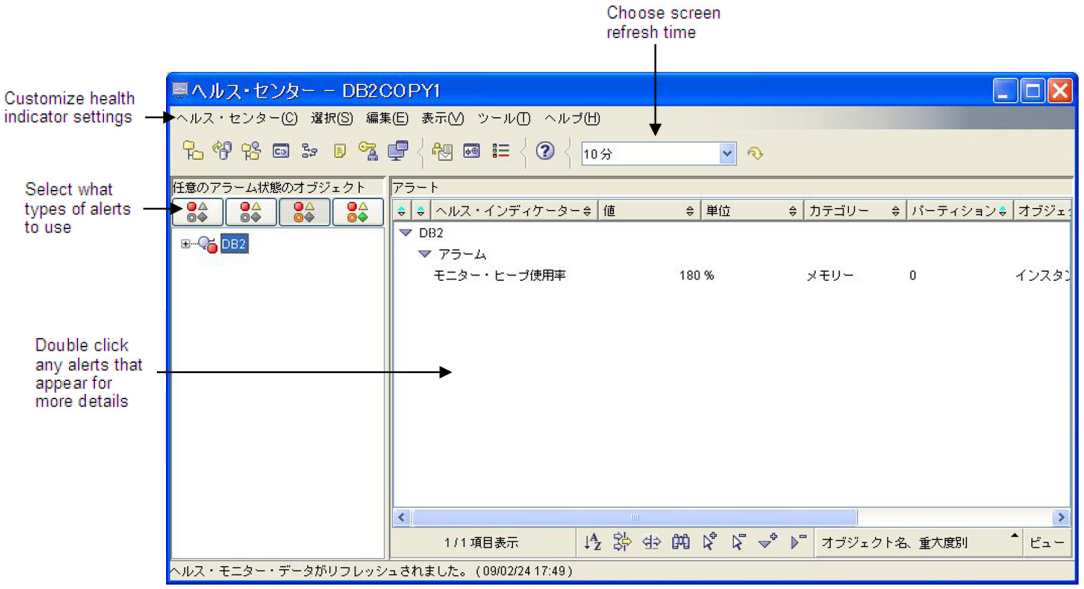


図 5.20 – ヘルス・センター

5.8.1.1 ヘルス・センターの起動方法

コントロール・センターからヘルス・センターを起動するには、「ツール」 -> 「ヘルス・センター」の順に選択します。この操作を図 5.21 に示します。または、「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「モニター・ツール」 -> 「ヘルス・センター」の順に選択してこのツールを起動することもできます。

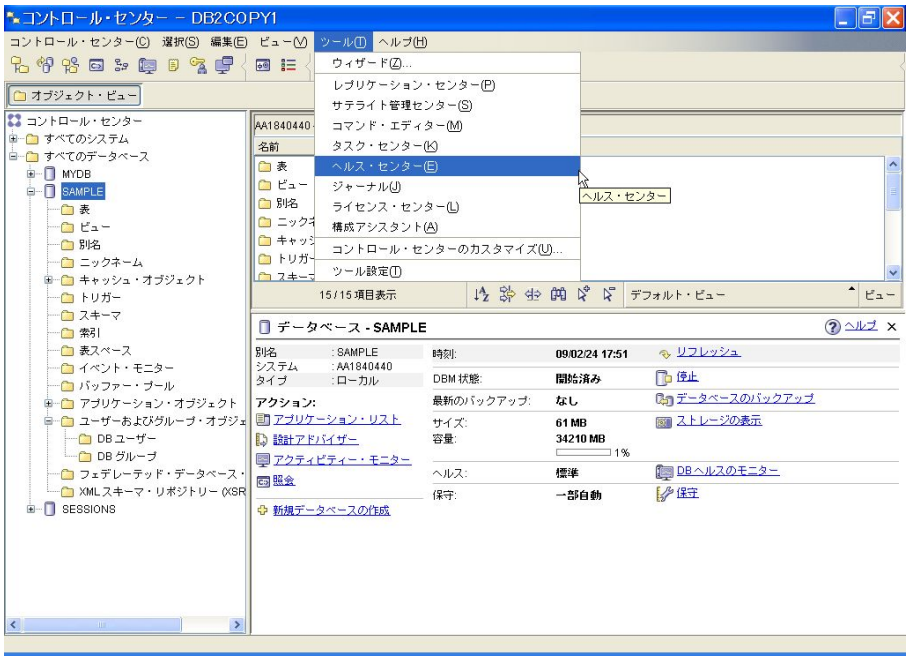


図 5.21 – ヘルス・センターの起動

5.8.1.2 ヘルス・アラート通知の構成方法

アラート通知を構成するには、ヘルス・センターを起動した後、「ヘルス・センター」メニューから「構成」 -> 「アラート通知」の順にクリックします (図 5.22 を参照)。アラート通知では、アラートが起動された場合に連絡する相手の名前を E メール・アドレスまたはページャー番号と併せて入力することができます。

図 5.21 – アラート通知

5.9 セルフ・チューニング・メモリー・マネージャー

DB2 9 で導入されたセルフ・チューニング・メモリー・マネージャー (STMM) はいくつかのオートノミック・コンピューティング・フィーチャーのうちの 1 つで、複数のメモリー構成パラメーターの値を自動的に設定することにより、メモリー構成作業を簡素化します。この機能を有効にすると、自動チューナーによって、データベースの複数のメモリー・コンシューマーに、使用可能なメモリー・リソースが動的に分配されます。メモリー・チューナーは、ワークロード特性の変化に対応して、メモリー構成パラメーターの値とバッファー・プールのサイズを調整し、パフォーマンスを最適化します。STMM を有効にするには、db cfg パラメーター SELF_TUNING_MEM を更新して ON にしてください。

自動保守や自動ストレージなどの他のオートノミック・コンピューティング・フィーチャーについては、本書の他の箇所に説明があります。

5.10 スクリプト

複数の DB2 コマンドまたは SQL ステートメントを繰り返し実行するスクリプト・ファイルを作成しておくことができると常に役立ちます。例えば、DBA が重要な表の行のカウントをチェックするために、ある特定のスクリプトを毎日実行しなければならない場合がこれに該当します。一般的なスクリプトの形式には以下の 2 つがあります。

- SQL スクリプト
- オペレーティング・システム (シェル) スクリプト

5.10.1 SQL スクリプト

SQL スクリプトには、クエリー・ステートメントとデータベース・コマンドを組み込みます。SQL スクリプトは比較的理解するのが簡単で、プラットフォームに共通です。ただし、変数やパラメーターはサポートされません。例えば script1.db2 という名前のファイルに、以下のリスト 5.1 に示すコマンドが保管されているとします。

```
CONNECT TO EXPRESS;
CREATE TABLE user1.mytable
        (   col1 INTEGER NOT NULL,
            col2 VARCHAR(40),
            col3 DECIMAL(9,2));
SELECT * FROM user1.mytable FETCH FIRST 10 ROWS ONLY;
COMMIT;
```

リスト 5.1 - script1.db2 ファイルに保管されているサンプル SQL スクリプト

上記のスクリプトでは、すべてのステートメントは SQL ステートメントであり、各ステートメントはステートメント区切り文字 (この場合はセミコロン) で分離されています。ファイル名には必ずしも拡張子「db2」を使う必要はないので、任意の拡張子を使用して構いません。

5.9.1.1 SQL スクリプトの実行方法

SQL スクリプトを実行するには、コマンド・エディター、Windows の DB2 コマンド・ウィンドウ、または Linux シェルのいずれかを使用することができます。リスト 5.1 のスクリプトを DB2 コマンド・ウィンドウまたは Linux シェルから実行する場合のコマンドは、以下のようになります。

```
db2 -t -v -f script1.db2 -z script1.log
```

あるいは以下のように記述することもできます。

```
db2 -tvf script1.db2 -z script1.log
```

このコマンドの内容は以下のとおりです。:

- **-t** ステートメントで使用するステートメント終了文字をデフォルト (セミコロン) に指定します。
- **-v** 冗長モードを指定します (db2 が実行中のコマンドをエコー出力することになります)。
- **-f** このフラグの後に指定されたファイル名を実行対象のスクリプト・ファイルとして指定します。
- **-z** 後で分析するため、画面出力にこのフラグの後に続くファイル名を追加するように指定します (これはオプションですが、推奨されます)。

-t フラグが使用されている場合、行区切り文字が指定されなければセミコロンがステートメントの区切り文字であるとみなされます。しかし、セミコロン以外の区切り文字が必要な場合もあります。例えば、SQL PL コードが含まれるスクリプトがあり、その SQL PL オブジェクト定義内で手続き型

のステートメントを終了するためにセミコロンが使用されている場合には、デフォルト (セミコロン) 以外のステートメント終了文字が必要となります。

一例として、以下のリスト 5.2 に示す functions.db2 というスクリプト・ファイルには、関数を作成するためのステートメントが含まれていますが、関数内の SELECT ステートメントは関数内で要求される構文に従う必要があるため、セミコロンで終了しなければなりません。そのため、CREATE FUNCTION ステートメントの終了文字としては、感嘆符 (!) を使用しています。セミコロンをステートメント終了文字として使用したとすると、スクリプト内で SELECT ステートメントとの間に競合が発生し、DB2 によってエラーがレポートされる結果となります。

```
CREATE FUNCTION f1()
   SELECT _   ;
   …
END!
```

リスト 5.2 - スクリプト･ファイル functions.db2 の内容

DB2 に異なるステートメント終了文字が使用されていることを通知するには、以下のように **-d** フラグの後に、使用する終了文字を続けます。:

```
db2 -td! -v -f functions.db2 -z functions.log
```

コマンド・ウィンドウまたは Linux シェルで使用できるその他のフラグを調べるには、以下のコマンドを実行します。

```
db2 list command options
```

5.10.2 オペレーティング・システム (シェル) スクリプト

オペレーティング・システム・スクリプトは SQL スクリプトに比べ、柔軟性と機能面が優れています。それは、オペレーティング・システム・スクリプトを使用すると、プログラミング・ロジックを追加することができるからです。この形式のスクリプトはプラットフォームに依存しますが、その一方、パラメーターと変数をサポートします。リスト 5.3 に単純な Windows オペレーティング・システム (シェル) スクリプトの一例を示します。

```
set DBPATH=C:
set DBNAME=PRODEXPR
set MEMORY=25
db2 CREATE DATABASE %DBNAME% ON %DBPATH% AUTOCONFIGURE USING
      MEM_PERCENT %MEMORY% APPLY DB AND DBM
db2 CONNECT TO %DBNAME% USER %1 USING %2
del schema.log triggers.log app_objects.log
db2 set schema user1
db2 -t -v -f schema.db2 -z schema.log
db2 -td@ -v -f triggers.db2 -z triggers.log
db2 -td@ -v -f functions.db2 -z functions.log
```

Listing 5.3 - オペレーティング・システム・スクリプト・ファイル create_database.bat の内容

上記のオペレーティング・システム・スクリプトをコマンドラインから実行するには、Windows で以下のコマンドを実行します。:

```
create_database.bat db2admin ibmdb2
```

ここで、***db2admin*** はユーザー ID であり、スクリプトの最初のパラメーターです。また、***ibmdb2*** はパスワードであり、スクリプトの 2 番目のパラメーターです。

ファイルがバッチ実行可能ファイルであることを通知するには、Windows では「bat」拡張子を使用します。Linux では、**chmod +x** のようなコマンドを使用してファイルのモードを変更し、そのファイルを実行可能ファイルとして指定する必要があります。その後は、上記に記載したのと同じ方法でスクリプトを実行することができます。

5.11 Windows Vista での考慮事項

Windows Vista のユーザー・アクセス制御 (UAC) 機能を使用すると、ご使用のユーザー ID がローカル管理者であっても、標準の権限を使用してアプリケーションが起動されます。このため、Vista で DB2 ツールまたはコマンドを起動すると、いずれも実行されるはずですが、アクセス権限に関連する問題が報告されることになります。この問題を回避するには、特に Vista ユーザー向けにインストール時に作成された「コマンド・ウィンドウ – 管理者」というショートカットを使用してください。このウィンドウでは、他のコマンドを実行したり、他のツールを起動したりすることができます (章の冒頭にある表 5.1 に示されたコマンドを使用)。また、Windows Vista の「スタート」メニューまたはいずれかの DB2 ショートカットから、起動する DB2 ツールを探し、それを右クリックし、「管理者として実行」オプションを選択する方法もあります。

DB2 拡張セキュリティーが有効になっている場合 (デフォルト。詳しくは第 10 章「データベース・セキュリティー」を参照)、コントロール・センターなどのグラフィカル・ツールを起動するためには、ご使用のユーザー ID が DB2ADMNS グループのメンバーであることも必要です。

5.12 まとめ

この章では、DB2 データ・サーバーの運用、構成、および管理に利用できる多様なツールを見てきました。

DB2 9.7 では、IBM Data Studio が主要な管理ツールとして導入されたことにより、データベースの管理作業および開発作業に新たな側面が加わりました。

さらにこの章では、コントロール・センター、SQL Assist ウィザード、タスク・センターおよびタスク・ジャーナル、ヘルス・エージェントおよびヘルス・モニターなどの、現在では非推奨となっているいくつかの GUI ツールについても説明しました。その一方で、コマンド行プロセッサーおよびコマンド・ウィンドウ・ツールは DB2 9.7 以降のバージョンでも引き続きアプリケーションを構成する一部であり、セルフ・チューニング・メモリー管理ツールも引き続きデータベース最適化プロセスの重要部分です。

データベース管理者のツールボックスの重要な要素として、DB2 のコマンドおよび機能を実行するために使用できるスクリプトがあります。この章では、SQL スクリプトとオペレーティング・シス

テム (シェル) スクリプトの両方、特にそれらを作成、保管、および実行する方法について詳細に考察しました。

最後に、DB2 ツールを Windows Vista で円滑に実行するための注意事項について学びました。

5.13 演習　スクリプトの利用

このセクションの演習では、DB2 でスクリプトを実際に扱います。

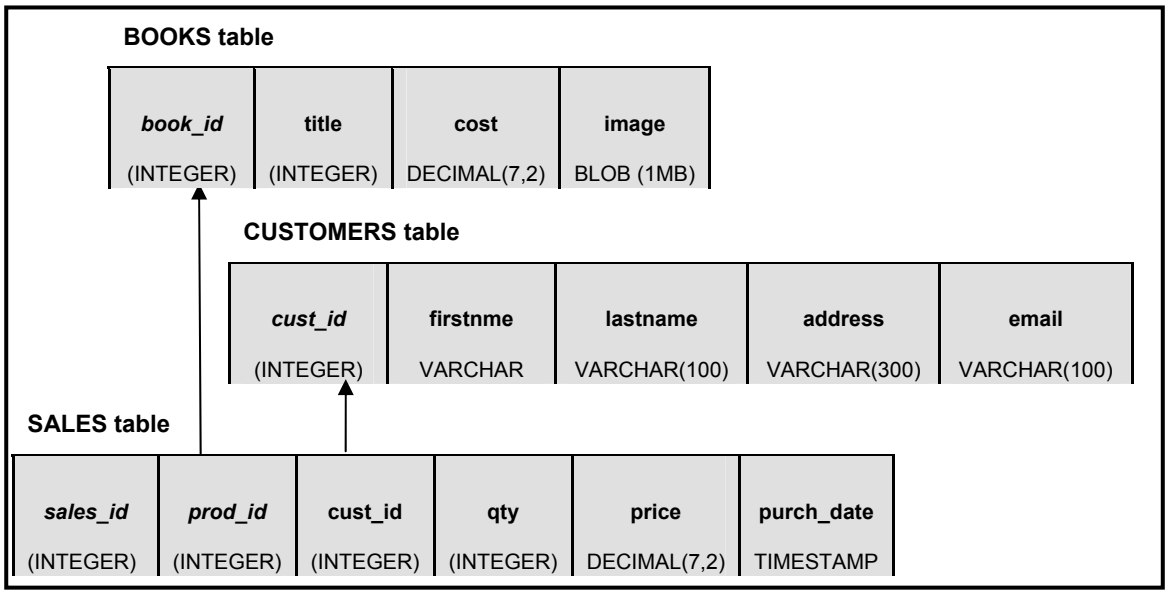
パート 1: スクリプトを使用して EXPRESS データベースにデータを追加する

このパートでは、コマンド・エディターと付属の 2 つのスクリプトを使用して EXPRESS データベース (前に作成したデータベース) にデータを追加します。

手順

1. EXPRESS データベースにいくつかの表とデータを追加します。ここでの作業で使用できるように、`Lab_Chpt5.db2` と `Lab_Chpt5.dat` という 2 つのスクリプトがあらかじめ用意されています。`Lab_Chpt5.db2` スクリプトには、表を作成するためのコマンドが含まれているので、このスクリプトは最初に実行する必要があります。`Lab_Chpt5.dat` スクリプトには、作成された表にデータを挿入するステートメントが含まれます。この 2 つのスクリプトは、本書に付属の `expressc_book_exercises_9.7.zip` ファイルにあります。スクリプトを実行するには、コマンド・エディターを開き、ツールバーのドロップダウン・リストから新しく作成したデータベースを選択します。このデータベースが一覧に表示されていない場合は、「追加」ボタンを使用してデータベースへの接続を追加してください。
2. コマンド・エディターで「選択」 -> 「オープン」メニューの順に選択し、スクリプトが保管されているフォルダーまでナビゲートします。`Lab_Chpt5.db2` ファイルを選択して「OK」ボタンをクリックします。このファイルの内容がコマンド・エディターの入力域に表示されます。「実行」ボタンをクリックしてスクリプトを実行します。スクリプトの実行中にエラーが発生しなかったことを確認してください。
3. 今度は `Lab_Chpt5.dat` ファイルに対して ステップ 2 を繰り返します。

作成した新規データベースは、極めて単純なインターネット書店を対象としたものです。***BOOKS*** 表にはこのインターネット書店が扱っている本に関するすべての情報が、***CUSTOMERS*** 表にはインターネット書店の顧客それぞれに関する情報が含まれます。そして ***SALES*** 表に含まれるのは販売データです。顧客が本を購入すると、***SALES*** 表にレコードが作成されます。以下の図に、この 3 つの表の設計と相互関係を示します。



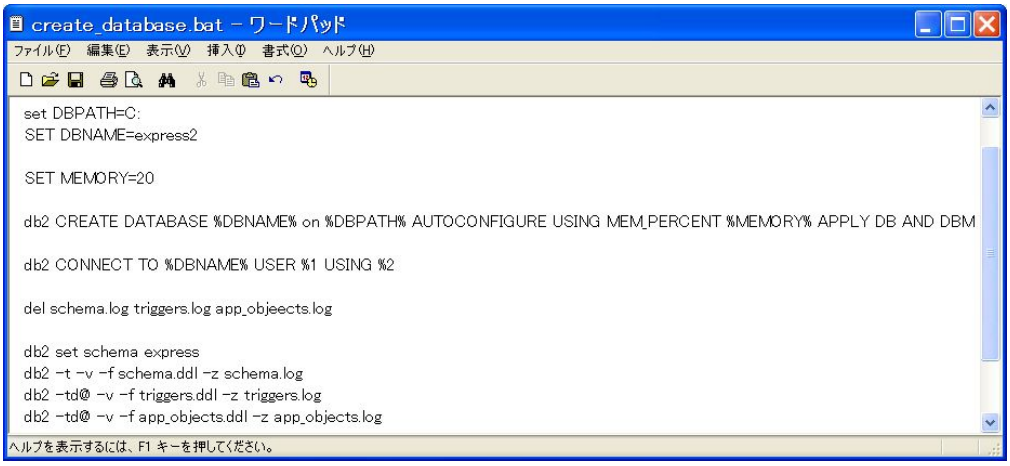
パート 2: EXPRESS データベースのインストール・スクリプトを作成する

スクリプトは、データベースの統計収集、バックアップ、データベースのデプロイメントなどといった反復タスクを実行するための強力なメカニズムです。オペレーティング・システム・スクリプトは、スクリプト・パラメーターをサポートするという利点があることから、柔軟性に優れたスクリプトとなっています。このパートでは、***EXPRESS*** データベースを ***EXPRESS2*** データベースとしてデプロイするオペレーティング・システム・スクリプトを作成します。このスクリプトで呼び出すのは、データベース・オブジェクトを対象に前に生成された SQL スクリプトです。スペースを節約するため、このクイックラボでは Windows プラットフォームに固有のスクリプトとコマンドのみを記載します。Linux で操作する場合には、以下の手順を適宜変更してください。

手順

1. テキスト･エディターを開いて、以下に示す内容を入力します。これらの行を入力する際にタイプミスをする人は数多くいるはずです。このようなエラーを犯し、それらを自分で修正することを学習できるように、この入力内容は意図的に、別個のファイルとして用意されていません。

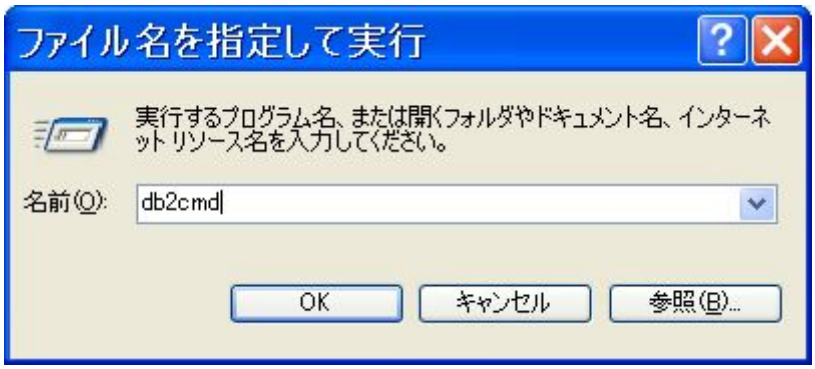
   `schema.ddl`、`triggers.ddl`、および `app_objects.ddl` ファイルの正しいパスを指定する必要があることにも留意してください。これらのファイルは、本書に付属の **`expressc_book_exercises_9.7.zip`** ファイルにも含まれています。



2. このスクリプト・ファイルに `create_database.bat` という名前を付けて、`C:\express` などのディレクトリーに保存します。

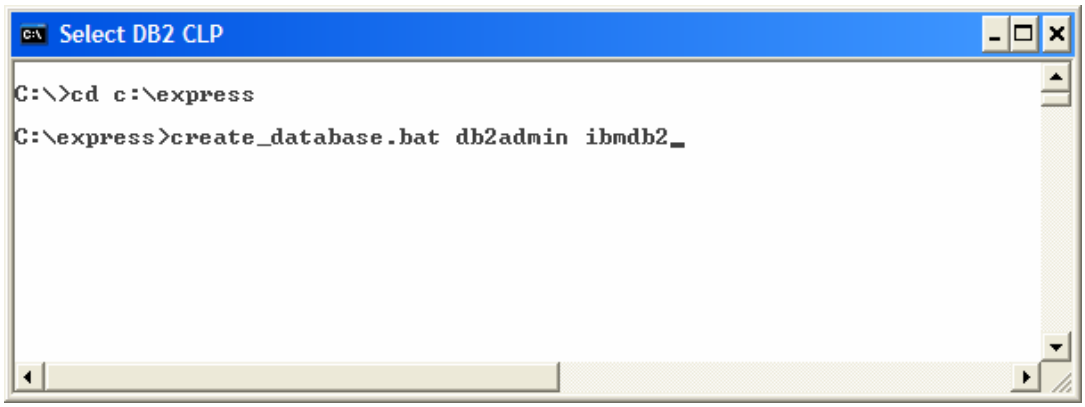
   注: ワードパッドを使用する場合は、「名前を付けて保存」ダイアログ・ウィンドウで MS-DOS 形式を選択するようにしてください。これ以外の形式で保存した場合、ワードパッドで表示できない文字が含まれてしまう可能性があり、スクリプトの実行の際に問題が生じることがあります。また以下の図に示すように、ファイル名は引用符で括るようにし、Windows によってファイル名に .TXT 拡張子が付けられることがないようにしてください。

3. DB2 を操作するスクリプトを実行するには、DB2 コマンドライン環境が必要です。DB2 コマンド・ウィンドウを開くため、「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1 (デフォルト)」 -> 「コマンド行ツール」 -> 「コマンド・ウィンドウ」の順に選択します。または以下に示すように、「スタート」 -> 「ファイル名を指定して実行」の順に選択し、**db2cmd** と入力して Enter キーを押すという方法もあります。



4. 以下のコマンドをコマンド・ウィンドウに入力して、スクリプトを実行します。

```
cd C:\express
create_database.bat db2admin ibmdb2
```



5. たった今作成したスクリプトが表示されるので、その内容を検討してください。各行で何が行われているかを理解できますか？

6. 以下の質問に答えてください。

   A. データベース接続が確立される場所はどこですか

   B. `%1` と `%2` は何を意味しますか？

   C. 以下のコード行が実行する内容と、その実行場所および目的は何ですか？

```
SET DBPATH=C:
```

D. 以下のコード行が実行する内容は何ですか？

```
del schema.log, triggers.log, app_objects.log
```

E. パラメーターを設定しないでスクリプトを呼び出した場合はどうなりますか？

F. 呼び出される SQL スクリプトに CONNECT TO ステートメントが含まれない理由は何ですか？ これらのスクリプトはどのようにしてデータベースに接続するのですか？

パート II – DB2 の学習:
データベース管理

6

第 6 章 – DB2 アーキテクチャー

この章では、DB2 アーキテクチャーについて以下の点から概説します。

- DB2 プロセス・モデル
- DB2 メモリー・モデル
- DB2 ストレージ・モデル

6.1 DB2 process model

図 6.1 に、DB2 プロセス・モデルを図解します。この図では、四角がプロセスを表し、楕円がスレッドを表しています。DB2 プロセスのメインとなるのは db2sysc というプロセスで、このプロセスは複数のスレッドから成っています。これらのスレッドのなかでもメインのスレッドは、プロセスと同じく db2sysc という名前のスレッドです。このスレッドが中心となって、他のスレッドが起動されます。リモート・アプリケーションから SQL CONNECT ステートメントを使用してサーバーに接続してくると、通信プロトコルのリモート・リスナーがこの要求を受け取り、DB2 コーディネーター・エージェント (db2agent) と通信を行います。DB2 エージェントは、DB2 に代わって操作を実行する小さなワーカー・エージェントのようなものです。ローカル・アプリケーション (つまり、DB2 と同じサーバーで実行されるアプリケーション) から接続される場合も、そのステップはリモート・アプリケーションの場合とほとんど同じで、db2tcpcm スレッドの代わりに db2ipccm エージェントが要求を処理する点が異なるぐらいです。並列処理が有効になっている場合をはじめ、db2agent が db2agntp スレッドのように機能する他のエージェントを起動する場合もあります。また、図中に示されている db2pfchr、db2loggr、db2dlock などのエージェントも、さまざまな目的で使用されます。表 6.1 では最もよく使われるプロセスについて、表 6.2 では最もよく使われるスレッドについて説明します。

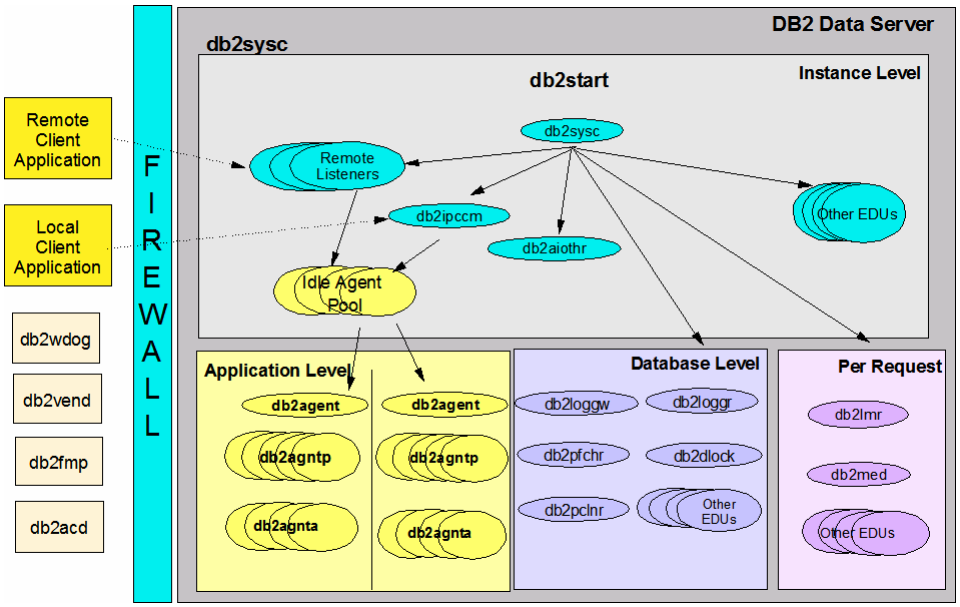


図 6.1 – DB2 プロセス・モデル

プロセス名	説明
db2sysc (Linux) db2syscs (Win)	メインの DB2 システム・コントローラーまたはエンジン。DB2 9.5 以降では、パーティション全体に対して、マルチスレッド化されたメイン・エンジン・プロセスが 1 つだけあります。エンジン・ディスパッチ可能単位 (EDU) はすべて、このプロセス内のスレッドです。このプロセスがなければ、データベース・サーバーは稼働することができません。
db2acd	オートノミック・コンピューティング・デーモン。クライアント・サイドの自動タスク (ヘルス・モニター、自動保守ユーティリティー、管理スケジューラーなど) を実行するために使用されます。このプロセスは以前、db2hmon と呼ばれていました。
db2wdog	DB2 ウォッチドッグ。ウォッチドッグは、メイン・エンジン・プロセス (db2sysc) の親プロセスです。db2sysc プロセスが異常終了した場合、このプロセスがリソースをクリーンアップします。
db2vend	DB2 9.5 で導入された fenced ベンダー・プロセス。すべてのサード・パーティー・ベンダー・コードはエンジン外部のこのプロセスの中で実行されます。サード・パーティー・ベンダー・アプリケーションは、DB2 を操作可能な IBM 以外のプログラムのことです。例えばログ・アーカイブの作成をサード・パーティー・ベンダー・コードで管理するには、ユーザー出口ルーチン・パラメーターがこのコードを指すように指定します。

db2fmp	ストアード・プロシージャーとユーザー定義関数の両方を対象に、ファイアウォール外側のサーバーでユーザー・コードを実行する fenced プロセス。このプロセスは、DB2 の以前のバージョンで使用されていた db2udf プロセスと db2dari プロセス両方の代わりとなります。

表 6.1 – 共通 DB2 プロセス

スレッド名	説明
db2sysc	システム・コントローラー・スレッド。このスレッドは、起動とシャットダウン、および実行中インスタンスの管理を行います。
db2tcpcm	TCP/IP 通信リスナー
db2agent	アプリケーションに代わってデータベース操作を実行するコーディネーター・エージェント (接続コンセントレーターが有効になっているかに応じて、少なくとも接続ごとに 1 つあります。)
db2agntp	INTRA_PARALLEL が YES に設定されている場合にアクティブになるサブエージェント。このスレッドはアプリケーションに代わってデータベース操作を実行します。db2agent が、db2agntp サブエージェント間での作業を調整します。
db2pfchr	DB2 非同期入出力データ・プリフェッチャー (NUM_IOSERVERS)
db2pclnr	DB2 非同期入出力データ・ライター (NUM_IOCLEANERS)

表 6.2 – 共通 DB2 スレッド

6.2 DB2 メモリー・モデル

DB2 メモリー・モデルは、インスタンス・レベル、データベース・レベル、そしてアプリケーションおよびエージェント・レベルのメモリー内にあるさまざまな領域からなります (図 6.2 を参照)。本書ではメモリー内の各領域について詳しく説明しませんが、その概要を簡単に説明します。

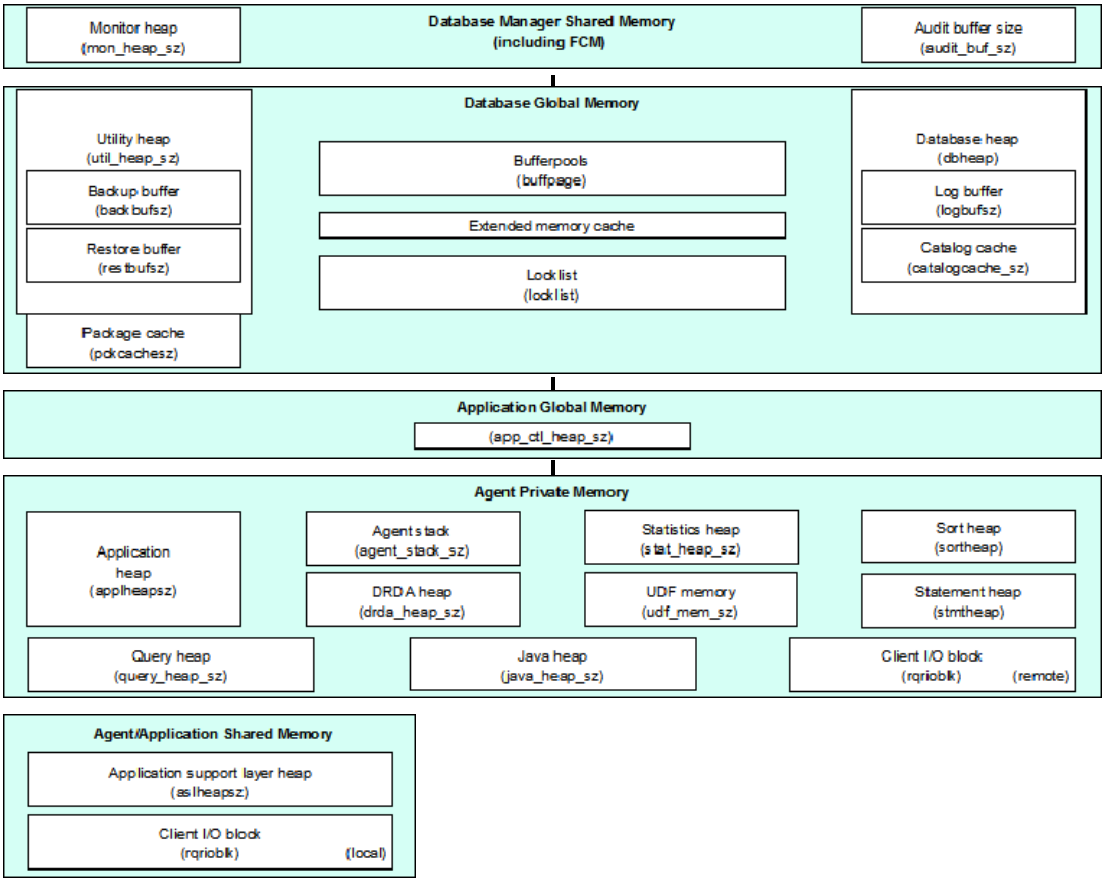


図 6.2 – DB2 メモリー・モデル

インスタンスが開始されると、データベース・マネージャーの共有メモリーが割り当てられます。通常、この共有メモリーが大きなスペースを占めることはありません。データベースに最初に接続するときには、データベース・グローバル・メモリーが割り当てられます。このブロックでは、特にクエリー・パフォーマンスの向上のために、バッファー・プールが最も重要な部分です。バッファー・プールのサイズによってデータベース・グローバル・メモリー全体のサイズが決まります。

エージェント専用メモリーとは、それぞれの DB2 エージェントが使用するメモリーのことです。接続コンセントレーターを使用しない場合、接続ごとに 1 つのエージェントが必要となります。エージェントに使用可能なメモリーは標準で 3 MB から 5 MB です。接続コンセントレーターを使用すると、複数の接続が 1 つのエージェントを使用できるようになるため、必要な物理メモリーが減ることになります。

6.3 DB2 ストレージ・モデル

このセクションでは、以下の概念を説明します。

- ページとエクステント
- バッファー・プール
- 表スペース

6.3.1 ページとエクステント

ページとは、DB2 内での最小ストレージ単位のことです。許容されるページのサイズは 4K、8K、16K、および 32K となっています。これらのページをグループ化したものが、エクステントです。DB2 で一度に 1 つのページを操作するとなると、パフォーマンス面でのコストがかかります。そのため、DB2 ではページではなくエクステントを処理するようになっています。ページ・サイズとエクステント・サイズは、バッファー・プールと表スペースを操作するときに定義されます。これについては、次のセクションで説明します。

6.3.2 バッファー・プール

バッファー・プールとは、表と索引データを対象とした実際のメモリー・キャッシュのことです。バッファー・プールは直接の順次入出力を減らしてパフォーマンスを向上させるとともに、非同期での読み取り (プリフェッチ) と書き込みを容易にします。つまり、DB2 は必要となるページを予測し、それらのページをディスクからバッファー・プールにプリフェッチして使用できるようにします。

バッファー・プールは、4K、8K、16K、および 32K ページのメモリー単位で割り当てられます。バッファー・プールはデータベースごとに少なくとも 1 つは必要です。さらに特定のページ・サイズの表スペースには、それに相当するバッファー・プールが少なくとも 1 つ必要となります。

6.3.2.1 バッファー・プールの作成方法

バッファー・プールを作成するには、**CREATE BUFFERPOOL** ステートメントを使用することができます。あるいは、コントロール・センターで特定データベース内のバッファー・プール・フォルダーを右クリックして「作成」を選択するという方法もあります (図 6.3 を参照)。

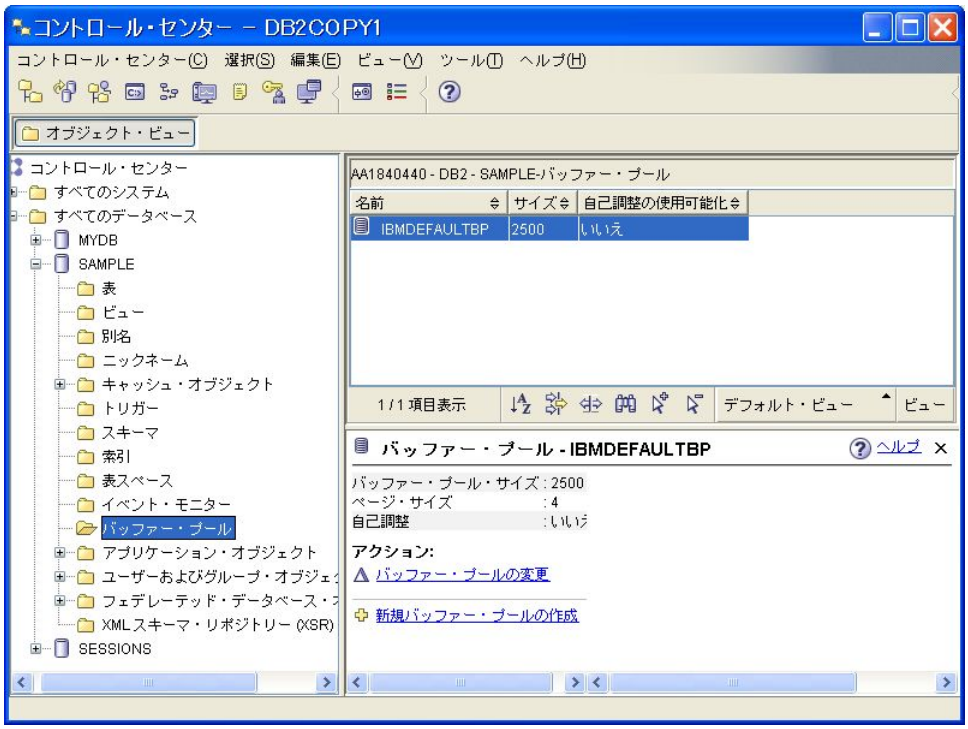


図 6.3 – バッファー・プールの作成

「作成」をクリックすると、「バッファー・プールの作成」ダイアログが表示されます (図 6.4 を参照)。

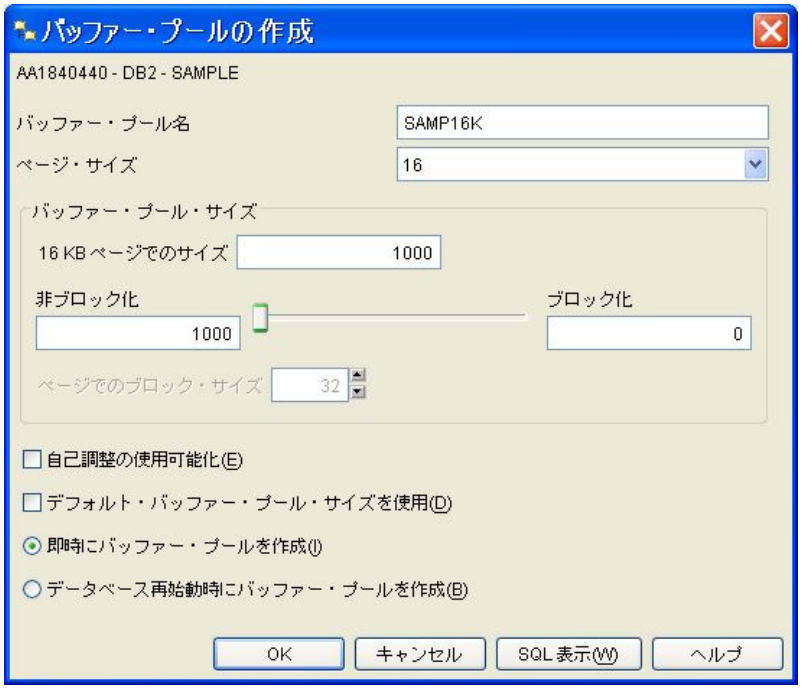


図 6.4 – バッファー・プールの作成ダイアログ・ボックス

図 6.4 の入力の内容は、ほとんどが一目瞭然です。「非ブロック化」フィールドと「ブロック化」フィールドでは、それぞれブロック・ベースでないページ数とブロック・ベースにするページ数を指定します。ブロック・ベースのバッファー・プールでは、ディスク上の連続するページがバッファー・プールのブロック・ベースの領域に同じく連続するページとして移されることになるため、これによってパフォーマンスが向上する可能性があります。ブロック・ベースに指定するページ数は、バッファー・プールのページ数の 98 パーセント以下でなければなりません。また、「ブロック化」フィールドの値を 0 に指定するとブロック入出力が無効になります。

バッファー・プールが作成されると、そのバッファー・プールはコントロール・センターに表示されます (図 6.5 を参照)。

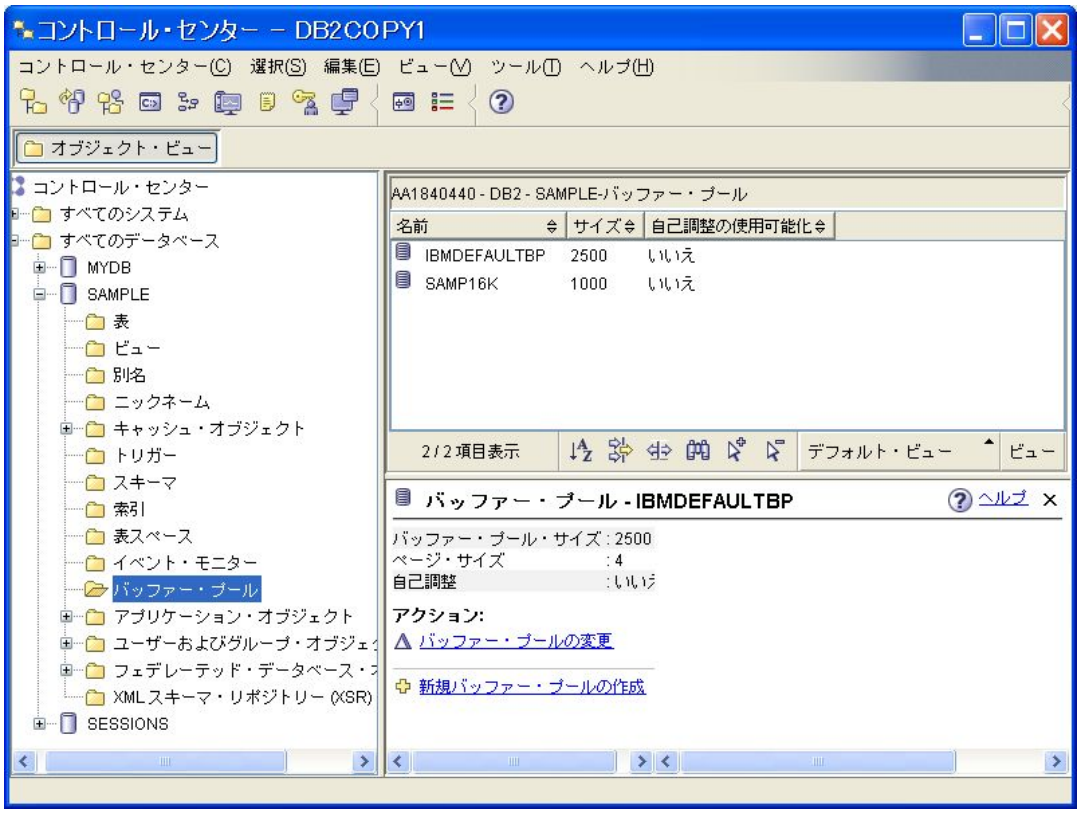


図 6.5 – コントロール・センターでバッファー・プール SAMP16K を作成

6.3.3 表スペース

表スペースとは、表と、システムの物理メモリー (バッファー・プール) およびコンテナー (ディスク) との論理インターフェースのことです。表スペースを作成するには、**CREATE TABLESPACE** ステートメントを使用します。このステートメントでは、以下の内容を指定することができます。

- 表スペースのページ・サイズ (4KB、8KB、16KB、または 32KB)。このページ・サイズは、対応するバッファー・プールのページ・サイズと同じでなければなりません。
- この表スペースに関連付けられたバッファー・プールの名前
- エクステント・サイズ
- プリフェッチ・サイズ

6.3.3.1 表スペースのタイプ

表スペースには以下の 3 つのタイプがあります。

- REGULAR

  ユーザー表のための表スペースです。例えば、デフォルトで作成される USERSPACE1 表スペースは REGULAR 表スペースです。

- LARGE

  このタイプの表スペースは、オプションで LOB データを専用の表スペースに分離する場合に使用されます。また、pureXML サポートを使用して作成されたデータベースに、XML データ型の列を使用して XML データを保管する場合にも使用されます。LARGE 表スペースはデフォルトです。

- TEMPORARY

  TEMPORARY 表スペースは、さらに 2 つのタイプに分けられます

  - SYSTEM TEMPORARY
  - DB2 が内部操作 (ソートなど) に使用する表スペースです。例えば、データベースを作成するとデフォルトで作成される TEMPSPACE1 表スペースは、SYSTEM TEMPORARY 表スペースです。SYSTEM TEMPORARY 表スペースは少なくとも常に 1 つは存在している必要があります。
  - USER TEMPORARY
  - メモリー内の一時表の作成に使用される表スペースで、ユーザー定義の宣言済みグローバル一時表 (DGTT) と作成済みの一時表 (CGTT) がこのスペースで作成されます。これらの表スペースは SYSTEM TEMPORARY 表スペースと混同されがちです。USER TEMPORARY 表スペースを作成してからでないと、DGTT や CGTT を作成することはできません。

6.3.3.2 表スペースの管理

表スペースは、その管理方法によって分類することができ、その管理方法は **`CREATE TABLESPACE`** ステートメントで指定することができます。

システムによる管理

このタイプの表スペースは、SMS (System Managed Storage) として知られおり、オペレーティング・システムによってストレージが管理されます。SMS は管理が容易であり、ファイル・システム・ディレクトリーがコンテナーとなります。事前にスペースが割り当てられることはありませんが、ファイルは動的に拡張します。コンテナーを指定すると、これらのコンテナーが作成時の固定コンテナーとなるため、リダイレクト・リストアを使用しない限り、他のコンテナーを後から追加することができなくなります。SMS 表スペースを使用する場合には、表データ、索引、LOB データを別の表スペースに分散させることはできません。

データベースによる管理

このタイプの表スペースは、DMS (Database Managed Storage) として知られており、DB2 によってストレージが管理されます。スペースを管理する作業には、DBA の場合に比べて手動での操作が多くなります。コンテナーは事前に割り当てたファイルにすることも、ロー・デバイスにすることもできます。ロー・デバイスでは、OS キャッシュを使わずにデータが直接書き込まれます。

コンテナーは ALTER TABLESPACE ステートメントを使用して、追加、破棄、またはサイズ変更することが可能です。パフォーマンスの点からすると、表データ、索引、LOB データを別の表スペースに分割してパフォーマンスを向上させることができる DMS 表スペースが最適です。

自動ストレージによる管理

このタイプの表スペースは自動ストレージによって管理されるため、SMS 表スペースと同様に使いやすいという利点があるだけでなく、DMS 表スペースが持つ最適なパフォーマンスと柔軟性も兼ね備えています。そのため、DB2 9 からはこのタイプが表スペースのデフォルト・タイプとなっています。このタイプの表スペースでは、DB2 が表スペースを管理するために使用するストレージ・デバイスのストレージ･パスと論理グループをユーザーがあらかじめ指定します。コンテナー定義は明示的には指定されません。コンテナーはストレージ・パス全体で自動的に作成されます。既存のコンテナーの拡張と新規コンテナーの追加は、完全に DB2 によって管理されます。**CREATE DATABASE** コマンドでストレージ･パスが指定されない場合、データベース・パスがストレージ･パスとして使用されます。データベース・パスは、主なデータベース定義が存在している場所ですが、データベース・パスが指定されていない場合には、データベース・マネージャー構成パラメーターである DFTDBPATH からその情報が得られます。Windows の場合、データベース・パスとして指定できるのはパスではなく、ドライブのみとなります。

自動ストレージを使用できるようにするには、まず、自動ストレージを有効に設定して (これはデフォルトの振る舞いです) データベースを作成し、ストレージ・パスのセットを関連付ける必要があります。作成後に、必要に応じてストレージ・パスを再定義することも可能です。その場合には、データベース RESTORE 操作を使用します。ストレージ・パスを関連付けたデータベースが用意された後は、自動ストレージを使用する表スペースを作成することができます (これも同じくデフォルトの振る舞いです)。

自動ストレージによる管理は DMS 表スペースに非常によく似ていますが、操作は自動化されており、DB2 によって管理されています。この操作にはコンテナーの割り当ておよび配置、さらには自動サイズ変更が含まれます。

自動ストレージを使用した表スペースによる管理の例を見てみましょう。まず、以下の例に従って自動ストレージが有効に設定されたデータベースを作成します。

自動ストレージをデフォルトで有効にする場合:

```
CREATE DATABASE DB1
```

自動ストレージを明示的に指定する場合:

```
CREATE DATABASE DB1 AUTOMATIC STORAGE YES
```

自動ストレージをデフォルトで有効にし、ストレージ・パスを指定する場合、ストレージ·パスとしてディレクトリーを指定するには、あらかじめでそのディレクトリーを作成しておかなければなりません。

Windows の場合:

```
CREATE DATABASE DB1 ON C:/, C:/storagepath1, D:/storagepath2
```

リストの最初の項目がドライブであることに注意してください。この項目は Windows のパスを表すものではなく、ドライブのみを指定可能なデータベース・パスを表します。この項目はまた、ストレージ・パスの 1 つとしても使用されます。そのため、データベース・パスは C: であり、ストレージ・パスは C:、C:\storagepath1、および D:\storagepath2 からなります。後の 2 つのディレクトリーはあらかじめ作成しておく必要があります。

Linux の場合:

```
CREATE DATABASE DB1 ON /data/path1, /data/path2
```

自動ストレージを明示的に無効にする場合:

```
CREATE DATABASE DB1 AUTOMATIC STORAGE NO
```

次に、以下の例に従って自動ストレージが有効に設定された表スペースを作成します。

表スペースの自動ストレージも同じくデフォルトで有効にする場合:

```
CREATE TEMPORARY TABLESPACE TEMPTS
```

表スペースの自動ストレージを明示的に指定する場合:

```
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
```

自動ストレージを暗黙的に指定し、初期サイズを割り当て、サイズの増加分と最大許容サイズを指定する場合:

```
CREATE TABLESPACE TS1
    INITIALSIZE 500 K
    INCREASESIZE 100 K
    MAXSIZE 100 M
```

6.3.3.3 表スペース内でのデータの保管方法

デフォルトでは、DB2 は複数のコンテナーにストライピングされたディスク・エクステントに同時に書き込みを行います。例えば、DMS 表スペース上で 3 つのロー・デバイスによるコンテナーを使用し、エクステント・サイズが 8 の 4K の表スペースを作成したとすると、32K のデータ (4K x 8 ページ/エクステント) が 1 つのディスクに書き込まれてから、次のディスクへの書き込みが行われ

ることになります。この仕組みを図 6.6 に示します。表がエクステントを共有していないという点に注目してください。

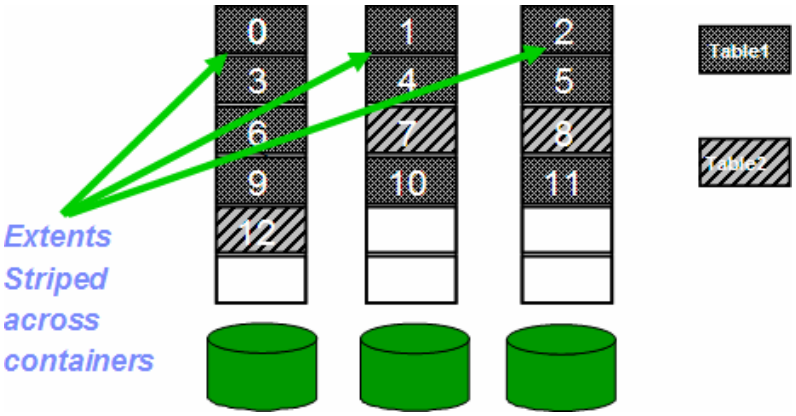


図 6.6 – 表スペースでのデータの書き込み

6.3.3.4 コントロール・センターでの表スペースの作成方法

コントロール・センターで表スペースを作成するには、特定のデータベース内にある「表スペース」フォルダーを右クリックし、「作成」を選択します (図 6.7 を参照)。これにより、「表スペースの作成ウィザード」が表示されます (図 6.8 を参照)。

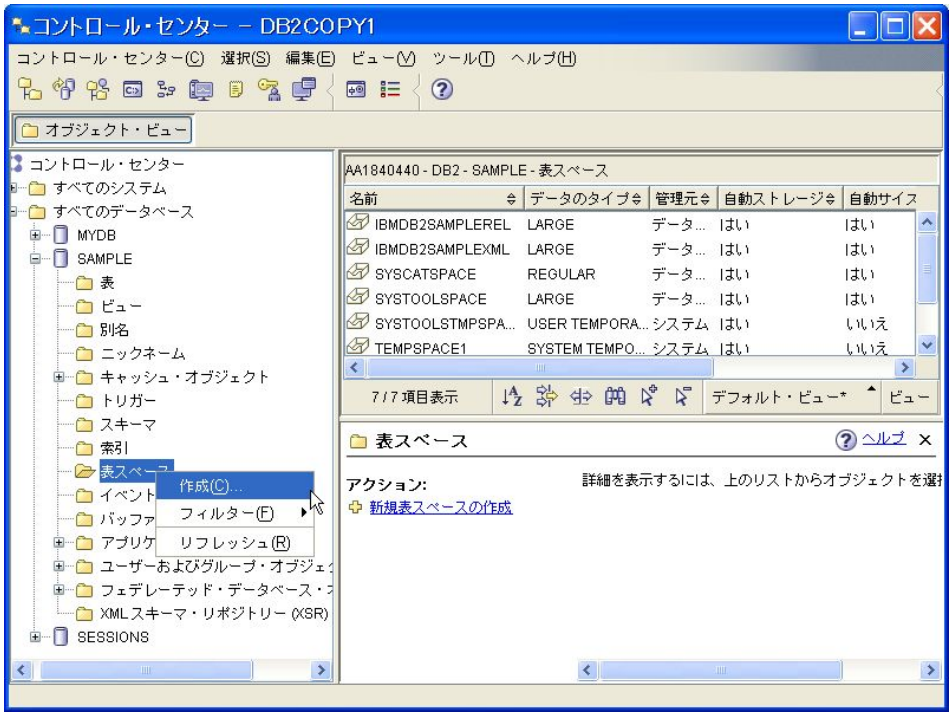


図 6.7 - コントロール・センターでの表スペースの作成

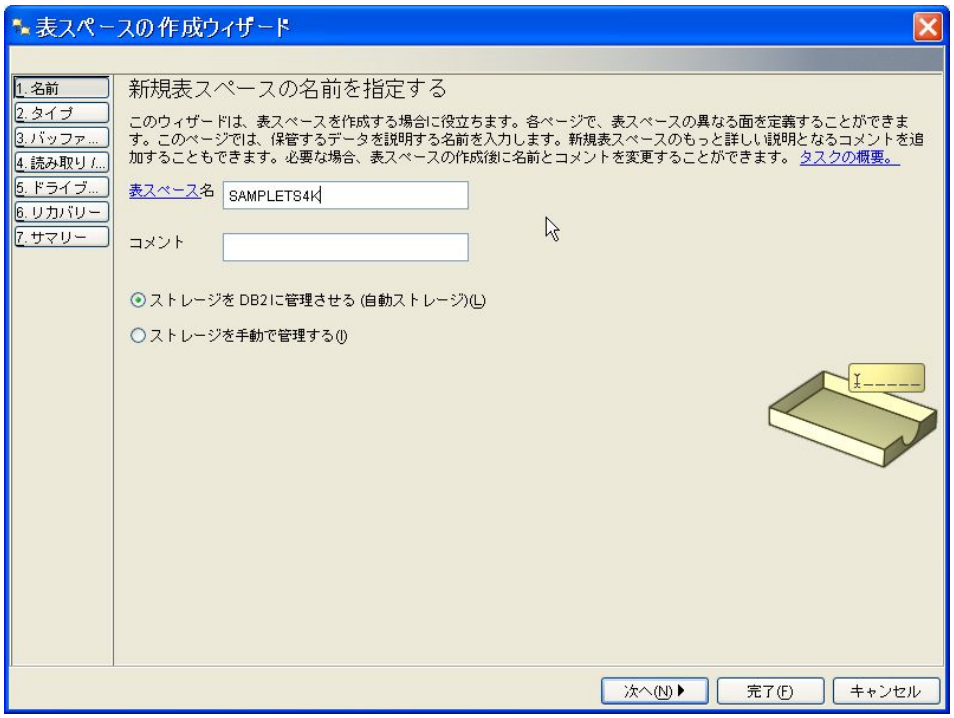


図 6.8 - 表スペースの作成ウィザード

図 6.8 に示すウィザードに従うと、表スペースを作成する手順が示されます。

6.4 まとめ

この章では、DB2 アーキテクチャーの 3 つの重要な特徴である、プロセス・モデル、メモリー・モデル、およびストレージ・モデルについて見てきました。プロセス・モデルについては、DB2 の稼働に不可欠な db2sysc を含め、共通のプロセスおよびスレッドを考察しました。

ストレージ・モデルについては、ページとエクステント、バッファー・プール (作成の詳細を含む)、および表スペースといった 3 つの重要な特徴を含め、詳細に検討しました。最後に、さまざまなタイプの表スペースと、その管理方法 (SMS、DMS、自動) およびコントロール・センターを使用した新規表スペースの作成方法について学びました。

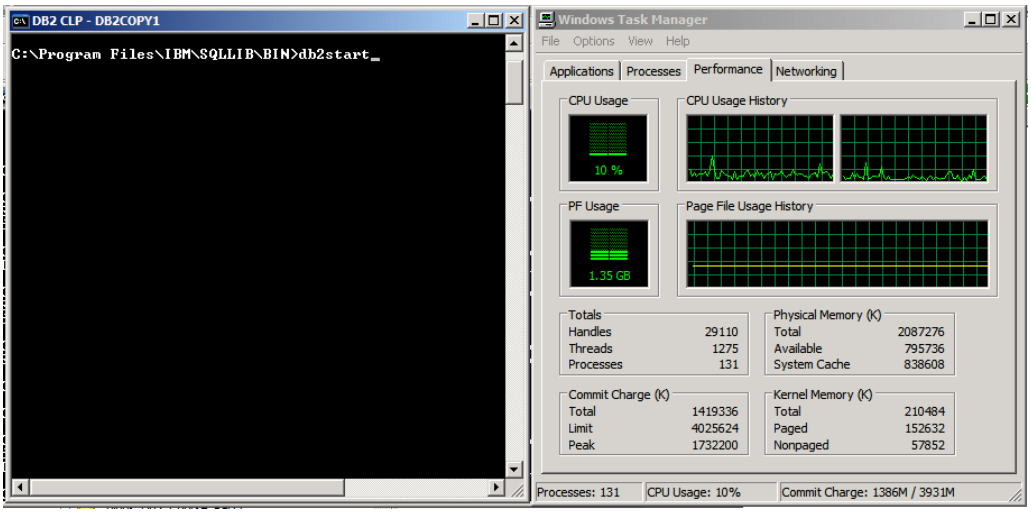
6.5 演習　DB2 アーキテクチャーの理解

この演習は、Windows での DB2 プロセス・モデル、メモリー・モデル、およびストレージ・モデルについての理解を助けることを目的としています。さまざまなプロセスおよびスレッドについての復習や、メモリー使用量のモニタリング、Windows の自動ストレージおよびストレージ・パスを使用するデータベースの作成の実践演習などを行います。ストレージ・パスはディスク (ドライブ) 間にまたがって作成するのが理想的ですが、コンピューターに複数のディスクが構成されていない場合があるため、この演習では C:\ ドライブのみを使用します。

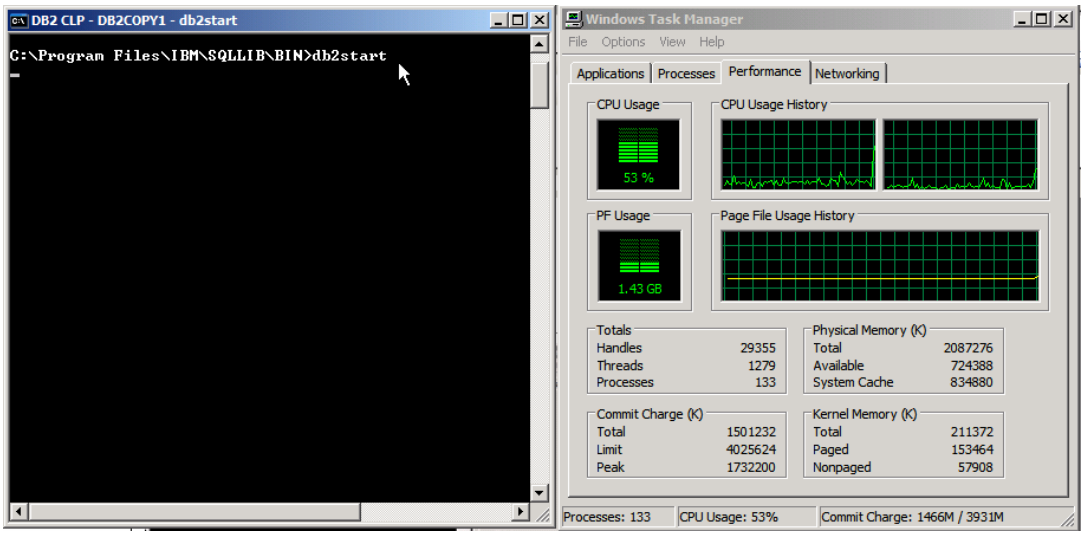
手順:

1. Windows のプロセスをいくつか見てみましょう。最初に、DB2 コマンド・ウィンドウを開き (「スタート」 -> 「ファイル名を指定して実行」 -> 「db2cmd」)、**db2stop force** コマンドを実行してインスタンスを確実に停止します。
2. Windows の「タスク マネージャ」を開き、「プロセス」タブを選択し、「イメージ名」列をクリックしてこの列をソートして、db2sysc.exe プロセスを探します。下図を参照してください。

3. ステップ 1 で DB2 インスタンスを停止したため、db2syscs.exe プロセスは見つからないはずです。
4. **db2start** コマンドで DB2 インスタンスを起動し、前のステップを繰り返します。今度は db2syscs.exe プロセスが見つかりましたか?
5. 次に CPU とメモリーの使用状況を見てみましょう。以下の手順に従ってください。
   A. アプリケーションをすべて終了し、システムで何も実行されていない状態にします。
   B. 新しい DB2 コマンド・ウィンドウを開き、**db2stop force** を実行します。
   C. 「タスク マネージャ」の「パフォーマンス」タブに移ります。
   D. 下図のように、「タスク マネージャ」と DB2 コマンド・ウィンドウを開いたままにして横に並べます。「物理メモリ」の利用可能メモリー量と「CPU 使用率」を書き留めてください。



E. **db2start** を実行し、すぐに一方で CPU 使用率とメモリーの使用量をモニターします。CPU 使用率で短いピークが見られ、利用可能なメモリーは 50 ～ 70MB 程度減少していることがわかります。これは DB2 インスタンスが使用しているメモリー量です。**db2stop force** をもう一度実行すると、メモリーは元の値に戻るはずです。



6. 前のステップを繰り返しますが、今回は SAMPLE データベースへ接続後に何が起きるかを観察します。DB2 コマンド・ウィンドウで以下のコマンドを実行してください。

```
db2start
db2 connect to sample
```

SAMPLE データベースに接続するとすぐに、利用可能な物理メモリー量が減少することがわかります。これは、データベースへの接続直後に、データベース・グローバル・メモリー (バッファー・プール、カタログ・キャッシュなど) が割り当てられるためです。

7. 前のステップを繰り返しますが、今回は任意のサイズのバッファー・プールを作成すると何が起きるかを観察します。バッファー・プールは、コンピューターの物理メモリーのサイズを超えないようにしてください。このサイズを超えてしまうと、DB2 はバッファー・プールを即座に割り当てなくなり、データベースがアクティブでなくなるまで作成を据え置きます。また、代わりに小さなシステム・バッファー・プールが使用され、DB2 は十分なメモリーを確保できるまでこのシステム・バッファー・プールを使用し続けます。例えば、約 160MB のバッファー・プールを作成するには、SAMPLE データベースに接続しているときに以下のコマンドを実行します。

```
db2 create bufferpool mybp immediate size 5000 pagesize 32k
```

8. 8. 自動ストレージを使用し、データベース・パスがドライブ C:\、C:\mystorage1、C:\mystorage2 であるデータベース ***mydb1*** を作成します。DB2 コマンド・ウィンドウで以下のコマンドを実行してください。

```
db2 create database mydb1 on C:\, C:\mystorage1, C:\mystorage2
```

まずディレクトリー `C:\`、`C:\mystorage1`、`C:\mystorage2` を作成しなければならないため、上記のコマンドを実行するとおそらくエラーになります。これらのディレクトリーを作成してから、もう一度試してください。

7

第 7 章 – DB2 クライアントとの接続

この章では、DB2 クライアントから DB2 サーバーへ TCP/IP で接続するために必要なセットアップについて説明します。DB2 サーバーにはクライアント・コンポーネントが付属しているため、DB2 サーバーがクライアントとして動作して別の DB2 サーバーに接続することも可能です。DB2 クライアントと接続する方法はいくつかありますが、この章では構成アシスタントを使用した最も簡単な方法だけを取り上げます。

注:

DB2 9.7 以降、構成アシスタントは非推奨となりましたが、製品には含まれており、使用することができます。

7.1 DB2 ディレクトリー

DB2 ディレクトリーとは、お使いのマシンから接続可能なデータベースに関する情報を保管するバイナリー・ファイルのことで、以下に説明する 4 つのディレクトリーがあります。

- システム・データベース・ディレクトリー
- ローカル・データベース・ディレクトリー
- ノード・ディレクトリー
- DCS ディレクトリー

「構成アシスタント」GUI ツールでは、上記のすべてのディレクトリーについて内容の表示および更新をすることができます。

7.1.1 システム・データベース・ディレクトリー

このディレクトリーはいわば、本の目次のようなものです。ここにはローカル・データベースであろうと、リモート・データベースであろうと、接続できるすべてのデータベースが表示されます。ローカル・データベースの場合には、ローカル・データベース・ディレクトリー へのポインターが含まれており、リモート・データベースの場合には、ノード・ディレクトリー 内のエントリーへのポインターが含まれています。このディレクトリーの内容を確認するには、以下のコマンドを実行します。

```
list db directory
```

7.1.2 ローカル・データベース・ディレクトリー

このディレクトリーには、マシン上の接続可能なデータベースに関する情報が含まれます。このディレクトリーの内容を確認するには、以下のコマンドを実行します。

```
list db directory on <drive/path>
```

7.1.3 ノード・ディレクトリー

このディレクトリーには、特定のリモート・データベースへの接続方法に関する情報が含まれます。例えば TCP/IP が使用されている場合、TCP/IP ノード・エントリーには、接続対象とする DB2 データベースが置かれているサーバーの IP アドレス、そしてこのデータベースが常駐するインスタンスのポートが含まれることになります。このディレクトリーの内容を確認するには、以下のコマンドを実行します。

```
list node directory
```

7.1.4 DCS ディレクトリー

このディレクトリーは、DB2 for z/OS (メインフレーム) または DB2 for i5/OS に接続するために DB2 Connect ソフトウェアをインストールしている場合にのみ表示されます。このディレクトリーの内容を確認するには、以下のコマンドを実行します。

```
list dcs directory
```

7.2 構成アシスタント

「構成アシスタント」GUI ツールを使用すると、DB2 クライアントと DB2 サーバーとの間の接続を簡単に構成することができます。Windows で構成アシスタントを起動するには、「スタート」 -> 「すべてのプログラム」 -> 「IBM DB2」 -> 「DB2COPY1」 -> 「セットアップ・ツール」 -> 「構成アシスタント」の順に選択します。

コマンドラインからこのツールを起動するには、**db2ca** を使用します。図 7.1 に、構成アシスタントを示します。

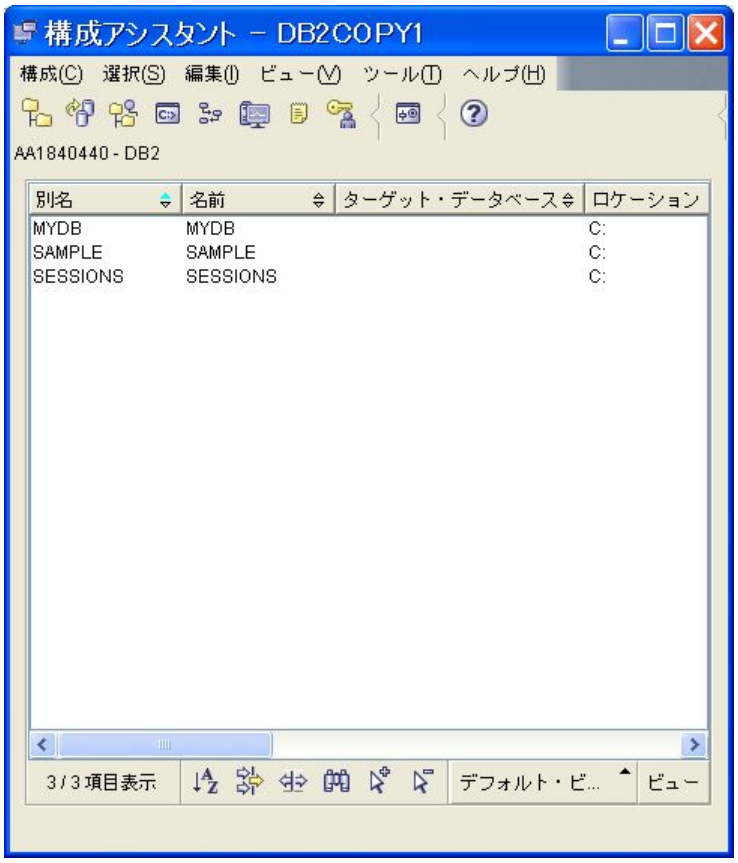


図 7.1 – 構成アシスタント

7.2.1 サーバー側で必要なセットアップ

サーバーでは以下の 2 つをセットアップする必要があります。

1) DB2COMM

この DB2 レジストリー変数は、クライアントからの要求をモニターする通信プロトコル・リスナーを指定します。一般に最もよく使用される通信プロトコルは TCP/IP です。このパラメーターを変更した後は、インスタンスを再起動する必要があります。構成アシスタントで DB2COMM の値を表示して変更するには、「構成」 -> 「DB2 レジストリー」の順に選択します (図 7.2 および図 7.3 を参照)。

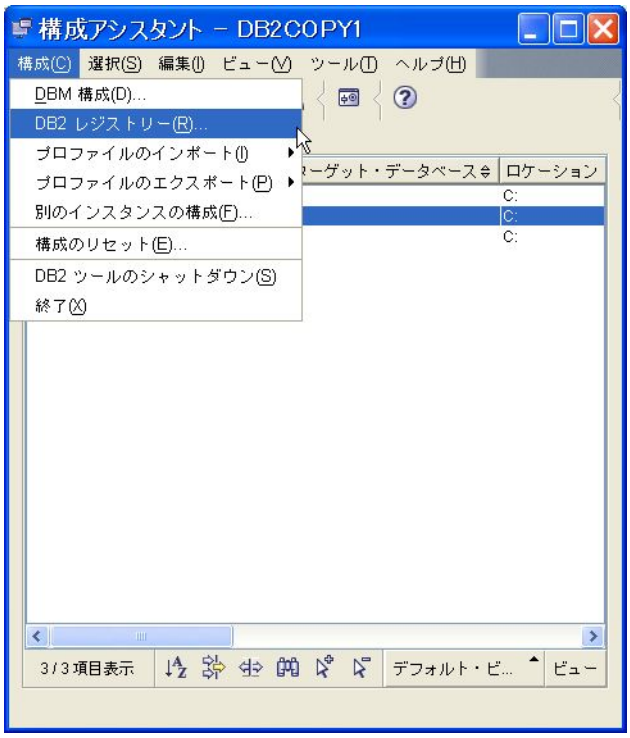


図 7.2 – DB2 レジストリーへのアクセス

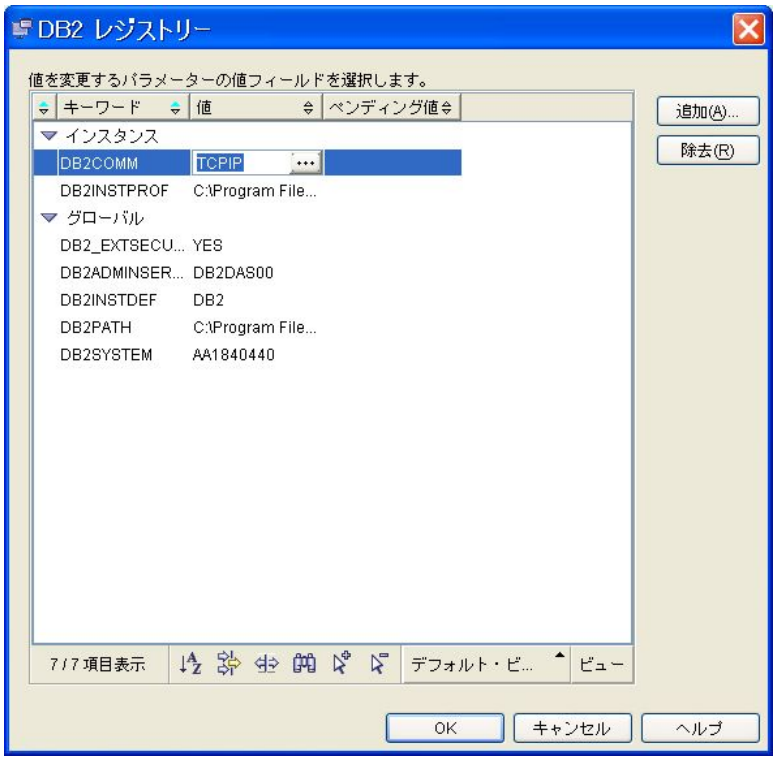


図 7.3 – DB2COMM DB2 Registry 変数の値の確認

2) SVCENAME

このデータベース・マネージャー構成パラメーターは、(TCP/IP サービス・ファイルで定義された) サービス名、またはこのインスタンスのデータベースにアクセスするときに使用するポート番号に設定する必要があります。このパラメーターを設定するには、構成アシスタントで、「構成」 -> 「DBM 構成」 の順に選択します (図 7.4 を参照)。

図 7.4 – 構成アシスタントでの DBM 構成の確認

「DBM 構成」ウィンドウが表示されたら、「通信」セクションにある SVCENAME を見つけてください。この値はストリングに変更することも、さらに必要な場合にはポート番号に変更することもできます。図 7.5 を参照してください。

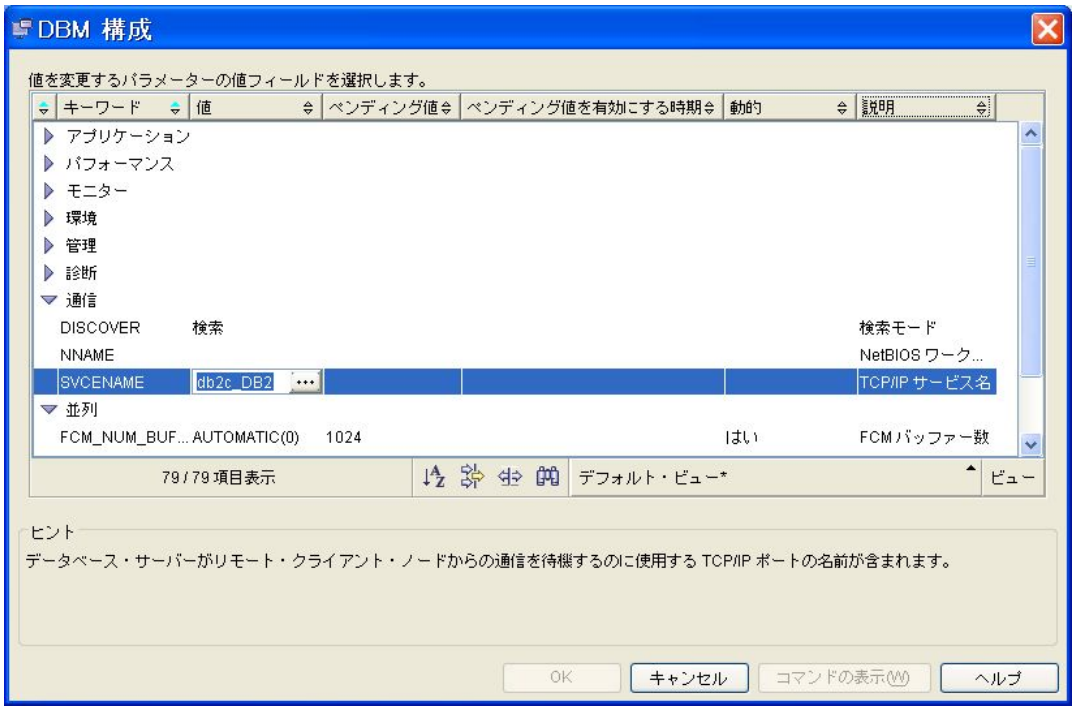


図 7.5 – SVCENAME DBM 構成パラメーターの確認

7.2.2 クライアント側で必要なセットアップ

クライアントで、以下の情報を事前に入手しておいてください。

- 接続先とするデータベースの名前
- データベースが常駐するサーバー側の DB2 インスタンスのポート番号。TCP/IP サービス・ファイル内に一致するエントリーがある場合には、サービス名を使用することもできます。
- データベースに接続するためのオペレーティング・システムのユーザー ID とパスワード。このユーザー ID はサーバーで定義済みのものでなければなりません。

以上の情報を DB2 クライアントから入力するには、構成アシスタントを使用します。まず、「選択」 -> 「データベースの追加 (ウィザードを使用)」の順にクリックして、データベースの追加ウィザードを起動します (図 7.6 を参照)。

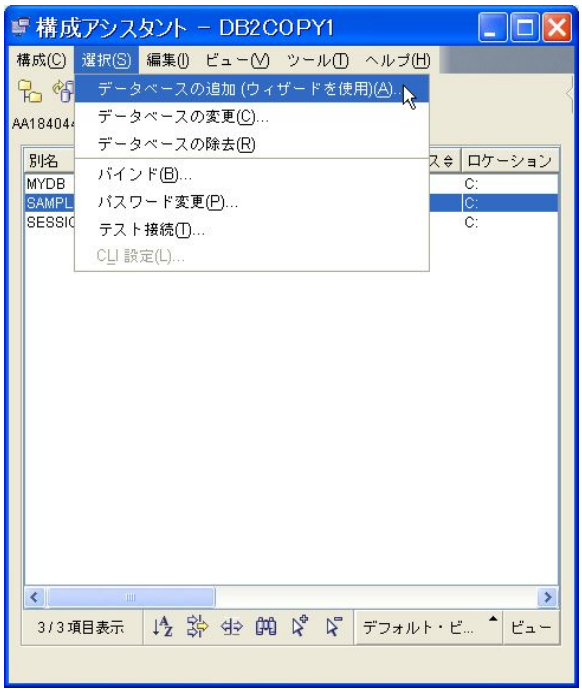


図 7.6 – データベースの追加ウィザードの起動

このウィザードは、構成アシスタントの空白部分で右クリックして「データベースの追加 (ウィザードを使用)」を選択するという方法でも起動することができます。図 7.7 にデータベースの追加ウィザードを示します。

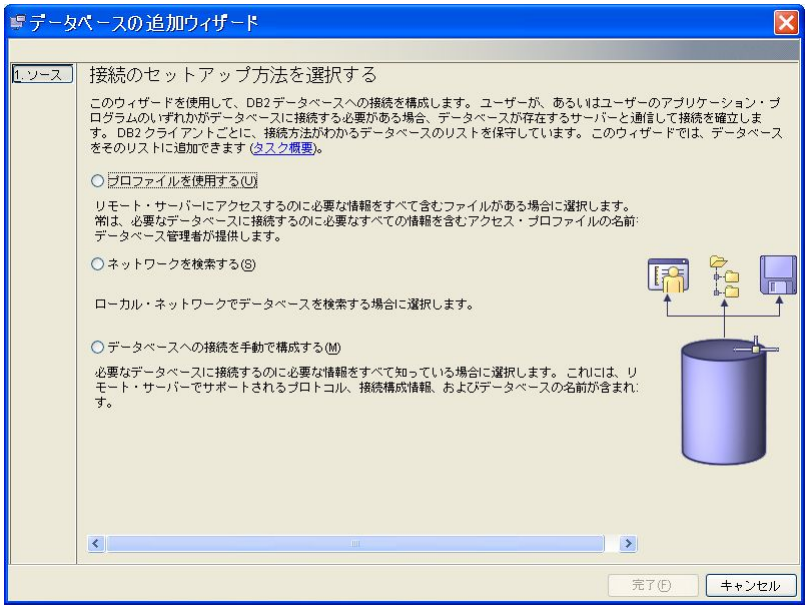


図 7.7 – データベースの追加ウィザード

データベースの追加ウィザードには、以下の 3 つのオプションがあります。

1.「プロファイルを使用する」

同じ 1 つの DB2 サーバーに接続する多数のクライアントを構成しなければならない場合があります。このような場合に便利な方法は、1 つのクライアントですべての構成を行い、これらの構成をプロファイル・ファイルに保管することです。このファイルを使用すれば、すべての情報を他のクライアントに直接ロードすることができます。図 7.7 で「プロファイルを使用する」を選択した場合には、既存のプロファイル・ファイルから情報をロードすることになります。詳細については、この後のクライアントおよびサーバー・プロファイルの作成方法についての節を参照してください。

2.「ネットワークを検索する」

ディスカバリーとしても知られるこのオプションは、DB2 にネットワークで特定のサーバー、インスタンス、データベースを検索するように指示します。このオプションが機能するには、データデータベースを検索する各 DB2 サーバーで、DAS が実行されている必要があります。このオプションには以下の 2 通りの検索方法があります。

- 「Search」

  ネットワーク全体を検索します。ネットワークが大規模で多数のハブがある場合には、すべてのシステムのデータを検索するのに時間がかかるため、この検索方法は推奨されません。

- 「Known」

  ネットワークで、指定されたアドレスにある既知のサーバーを検索します。

  上記の 2 つの方法を図 7.8 に図解します。

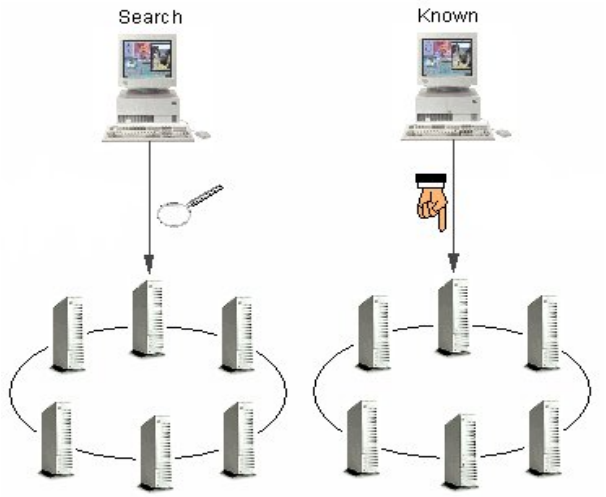


図 7.8 – Search および Known 検索 (ディスカバリー) の方法

管理者が、クライアントにはネットワークで機密情報を持つデータベースを検索させたくないという場合もあります。その場合には、DAS レベル、インスタンス・レベルまたはデータベース・レベルで防ぐことができます。図 7.9 にこの詳細を示します。

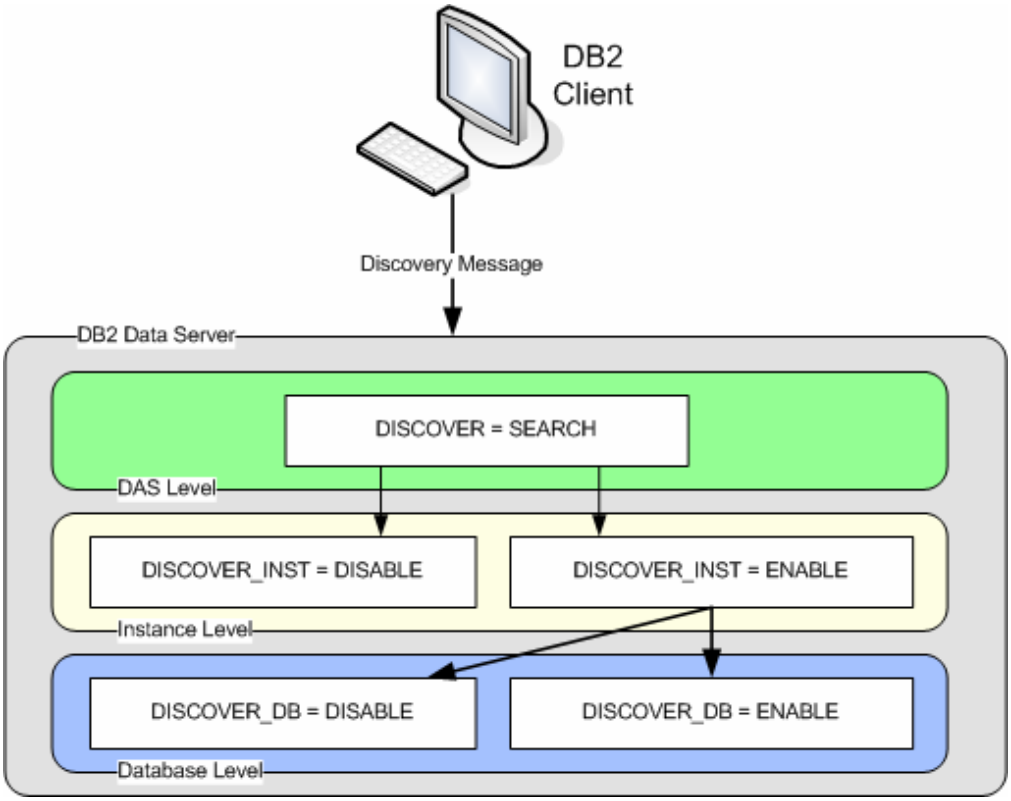


図 7.9 – ディスカバリーを許可するパラメーターの構成

図 7.9 には、ディスカバリーを有効または無効に設定できる各レベルが示されています。DAS レベルでは、DISCOVER パラメーターの値を SEARCH または KNOWN に指定することができます。インスタンス・レベルではデータベース・マネージャー構成パラメーター DISCOVER_INST を DISABLE または ENABLE のいずれかに設定することができます。そして、データベース・レベルでも DISCOVER_DB パラメーターを同じく DISABLE または ENABLE のいずれかに設定することができます。これらのパラメーターを設定することで、データベース・ディスカバリーを詳細に構成することができます。

3. 「データベースへの接続を手動で構成する」

このオプションを選択して、「構成アシスタント」に手動でホスト名、ポート番号、およびデータベース情報を入力すると、接続構成を実行するカタログ・コマンドが生成されます。構成アシスタントは、入力された情報が正しいかどうかはチェックしません。サーバーに接続できない場合には、情報が正しくないということになります。また、リモート・データベースに接続するために入力するユーザー ID とパスワードが正しいことも確認してください。デフォルトでは、接続対象の DB2 サーバーで認証が行われるため、そのサーバーに定義されたユーザー ID とパスワードを入力する必要があります。

7.2.3 クライアントおよびサーバー・プロファイルの作成

多数のサーバーあるいはクライアントを構成する場合には、それぞれを個別にセットアップする代わりに 1 つのサーバーまたはクライアントだけをセットアップし、そこからプロファイルをエクスポートして他のクライアント/サーバーにそのプロファイルを適用することができます。この方法は、環境のセットアップに要する管理作業の時間を大幅に節約してくれます。

構成アシスタントを使用してカスタマイズしたプロファイルを作成するには、「構成」メニューをクリックし、「プロファイルのエクスポート」 -> 「カスタマイズ」の順に選択します (図 7.10 を参照)。

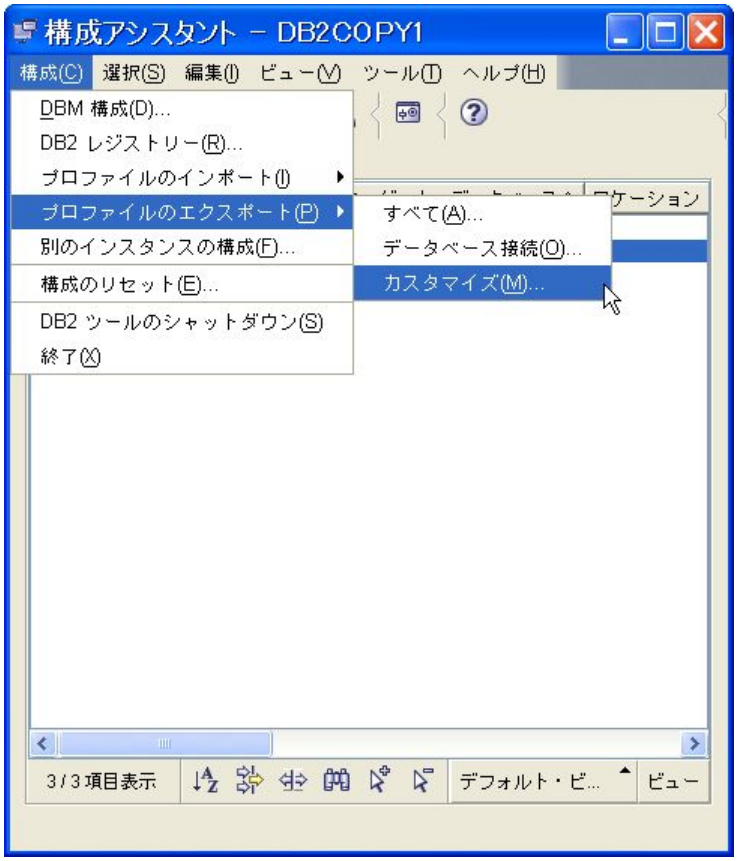


図 7.10 – プロファイルのエクスポート

図 7.11 に、プロファイルをエクスポートするための必須フィールドを示します。

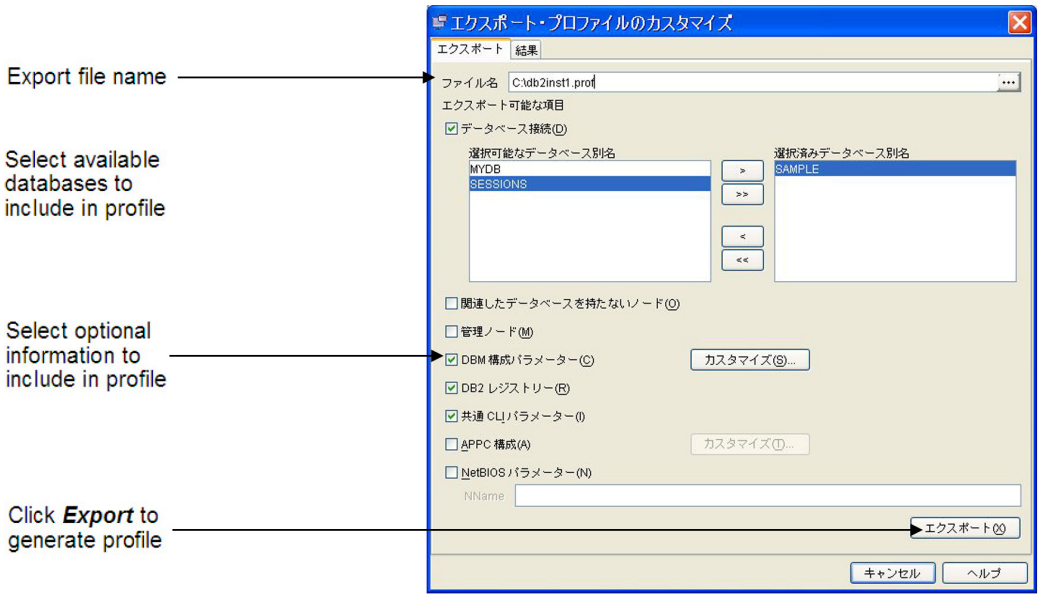


図 7.11 – エクスポート・プロファイルのカスタマイズダイアログ

「エクスポート・プロファイルのカスタマイズ」ダイアログで「エクスポート」をクリックしたときの結果の画面を図 7.12 に示します。

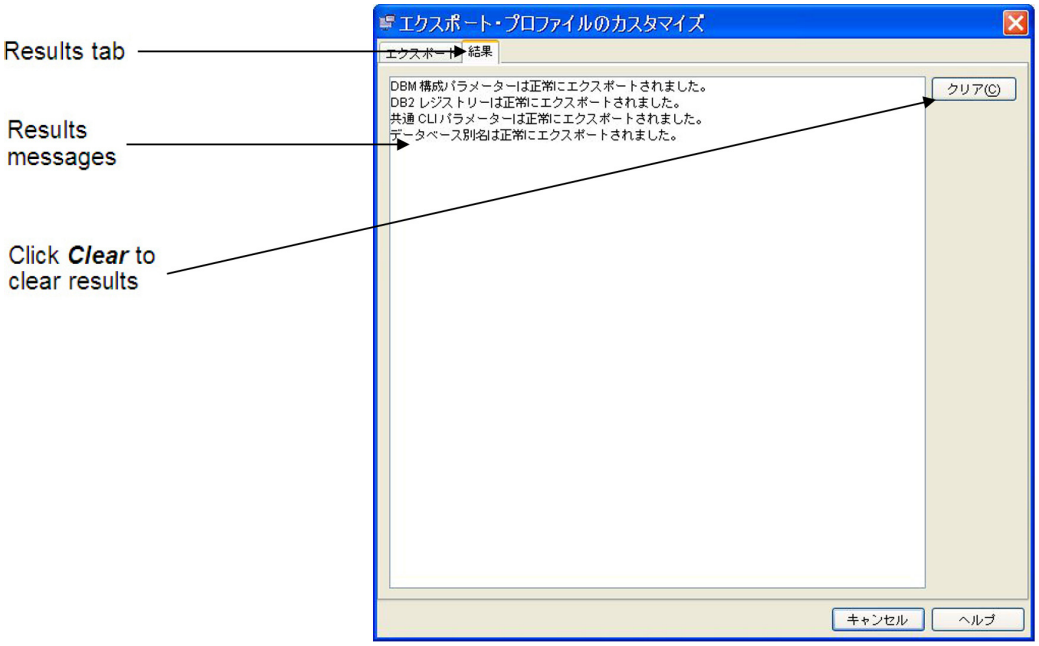


図 7.12 – プロファイルのエクスポート結果

カスタマイズしたプロファイルをインポートするには、構成アシスタントで「構成」メニューをクリックし、「インポート・プロファイル」 -> 「カスタマイズ」の順に選択します (図 7.13 を参照)。

図 7.13 – プロファイルのインポート

図 7.14 に、プロファイルをインポートするための必須フィールドを示します。

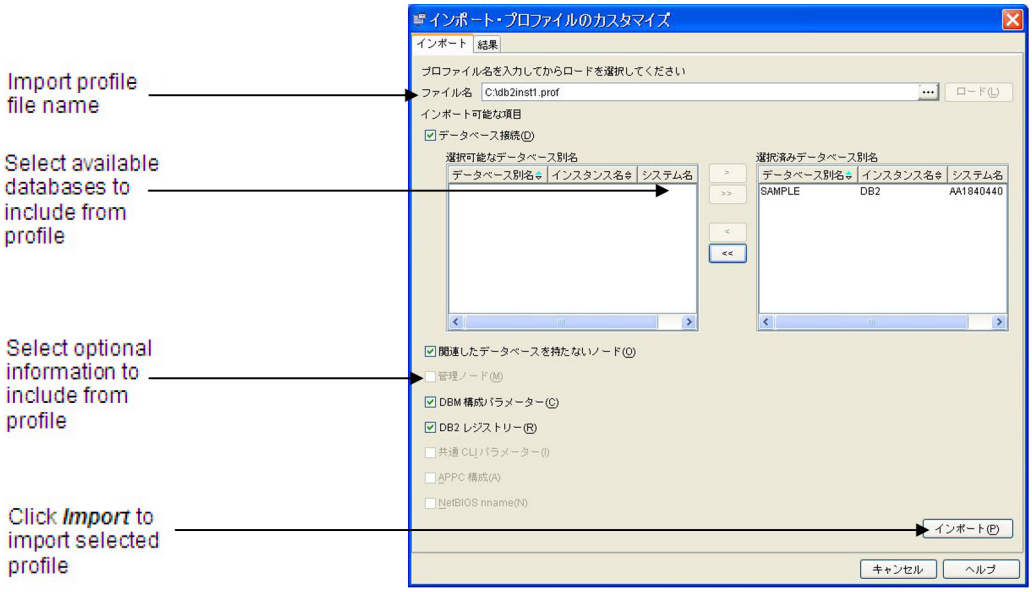


図 7.14 – インポート・プロファイルのカスタマイズ

7.3 まとめ

データ･クライアントからサーバーへの接続は、リレーショナル･データベース管理の重要な側面です。この章では、DB2 クライアントとの接続に関して見てきました。まずは、DB2 に関係するデータベース･ディレクトリーとノード･ディレクトリーについて、その目的と内容を説明しました。

次に、構成アシスタントの GUI を使用してクライアント・サーバー間の接続をセットアップする方法について、サーバー側とクライアント側のそれぞれで何を構成する必要があるのかを説明しました。

続いて、データベースの追加ウィザードを使用してクライアントからサーバーに接続を行う方法として、保管されているプロファイルを使用する方法、ネットワークを検索する方法 (ディスカバリーとしても知られています)、サーバー情報を手動で入力する方法についても見てきました。そして最後に、クライアントおよびサーバー･プロファイルを作成する方法について詳しく説明しました。

7.4 演習 「構成アシスタント」を使用する

構成アシスタントを使用すると、素早く簡単にリモート・データベース接続を構成することができます。この演習では、リモート DB2 サーバー (近くのワークステーション) に常駐するデータベースを「Search」モードと「Discover」モードの両方を使用してカタログします。いったんデータベースがカタログされると、ローカル・システム上のデータベースであるかのようにアクセスできるようになります。通信プロセスはすべて、DB2 によって行われます。

この演習では、読者がネットワーク内で作業していることを前提とします。そうでない場合には、自分のコンピューターをクライアント・マシン兼サーバー・マシンとして使用することで、以下の構成手順に従って独自のシステムに接続することができます。

手順

1. 近くのワークステーション (またはインストラクター) から以下の情報を入手します。
2. リモート・データベース情報:

   (PR) プロトコル __TCPIP___

   (IP) IP アドレスまたはホスト名 _____

   (PN) インスタンス・ポート番号 _____

   (DB) データベース名 __SAMPLE__

   ヒント

   Windows でホスト名を調べるには、コマンド・ウィンドウから **hostname** と入力します。

   Windows で IP アドレスを調べるには、コマンド・ウィンドウから **ipconfig** と入力します。
3. 構成アシスタントを開きます (ヒント: スタート・メニューからアクセスできます)。
4. 「選択」メニューを開き、「データベースの追加 (ウィザードを使用)」を選択します。

5. ウィザードの「ソース」ページで、「データベースへの接続を手動で構成する」オプションを選択します。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
6. ウィザードの「プロトコル」ページで、「TCP/IP」オプションを選択します。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
7. ウィザードの「TCP/IP」ページで、ステップ 1 で書き留めた完全ホスト名または IP アドレス、およびポート番号を入力します。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
8. 注: ローカル 側のサービス・ファイル内のエントリーに定義されたポート番号が、リモート・サーバー・インスタンスがリッスンしているポートに対応している場合は、「サービス名」オプションを使用することができます。このオプションを使用すると、DB2 はサーバーではなく、ローカル・マシン上でサービス・ファイルを検索します。このオプションを使用するには、ローカル・マシンのサービス・ファイルにエントリーを追加する必要があります。
9. ウィザードの「データベース」ページの「データベース名」フィールドに、ステップ 1 で書き留めた、リモート・サーバーで定義されたデータベース名を入力します。これと同じ値が、「データベース別名」フィールドに自動的に入力されることに注意してください。データベース別名は、ローカル・アプリケーションがこのデータベースに接続するために使用する名前です。SAMPLE という名前のローカル・データベースはすでに定義されており、DB2 では同じ名前を持つ別のデータベースのカタログを許可しないので、異なる別名を使用する必要があります。この例では、データベース別名を SAMPLE1 に変更します。必要に応じて、このデータベースに関するオプションのコメントを入力することもできます。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
10. ウィザードの「データ・ソース」ページでは、オプションでこの新しいデータベース (データ・ソース) を ODBC データ・ソースとして登録することができます。その場合、データベースは自動的に Windows ODBC Manager 内に登録されることになります。この例では、ODBC を使用しないので、「CLI/ODBC 用にこのデータベースを登録」のチェック・マークを外します。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
11. ウィザードの「ノード・オプション」ページで、リモート・データベースが位置するサーバーのオペレーティング・システムを指定します。このクイックラボのすべてのワークステーションは Microsoft Windows を使用するため、ドロップダウン・リストから「Windows」が確実に選択されるようにしてください。「インスタンス名」フィールドは DB2 に設定されているはずです。そうなっていない場合は、フィールドの値を DB2 に設定します。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
12. ウィザードの「システム・オプション」ページで、表示されたシステムおよびホスト名が正しいこと、そしてオペレーティング・システムの設定を確認します。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
13. ウィザードの「セキュリティー・オプション」ページでは、ユーザー認証を行うかどうかを指定し、使用する認証方式を選択することができます。ここでは「サーバーの DBM 構成の認証値を使用する」オプションを選択してください。このオプションを選択すると、リモート・インスタンスの構成ファイル内の `AUTHENTICATION` パラメーターで指定された

認証方式が使用されます。リモート・データベースをカタログしてウィザードを閉じるため、「完了」ボタンをクリックします。確認ボックスが表示されるので、「接続のテスト」ボタンをクリックしてデータベースに正常に接続できることを確認します。また、指定したユーザー名とパスワードが、リモート・サーバーで定義された有効なものであることも確認してください (サーバーの AUTHENTICATION パラメーターの値が SERVER に設定されている可能性が高いため)。テスト接続が成功した場合、リモート・データベースは正常にカタログされたことになります。成功しなかった場合には、ウィザードに戻って、指定した値がすべて正しいことを確認します (ウィザードの設定を再確認するには、「変更」ボタンをクリックします)。

14. コントロール・センターを開き、新しくカタログされたリモート・データベースの各表を表示してみてください。

15. 構成アシスタントに戻り、今度は「ネットワークを検索する」オプションを使用して別のデータベースをカタログします。手動で接続を構成したときと同じように、ウィザードを設定していきます。大規模なネットワークでは、ディスカバリーによって結果が戻されるまでに時間がかかることに注意してください。

8

第 8 章 – データベース・オブジェクトの操作

この章では、スキーマ、表、ビュー、索引、シーケンスなどのデータベース・オブジェクトについて説明します。一部の拡張データベース・アプリケーションのオブジェクト (トリガー、ユーザー定義関数 (UDF)、およびストアード・プロシージャーなど) については、第 14 章「SQL PL ストアード・プロシージャー」で説明します。

8.1 スキーマ

スキーマとは、データベース・オブジェクトのコレクション用の名前空間のことで、主に以下の目的で使用します。

- オブジェクトの所有権またはアプリケーションとの関係を示すため
- 関連するオブジェクトを論理的にグループ化するため

パブリック·シノニムを除くすべての DB2 データベース・オブジェクトには 2 つの部分からなる完全修飾名があります。スキーマは以下に示す完全修飾名の最初の部分です。

```
<schema_name>.<object_name>
```

完全修飾オブジェクト名はユニークでなければなりません。スキーマを指定しないでデータベースに接続してオブジェクトを作成または参照すると、DB2 はスキーマ名として、データベースとの接続に使用されたユーザー ID を使用します。例えば、ユーザー ***arfchong*** として **SAMPLE** データベースに接続し、以下の **CREATE TABLE** ステートメントを使用して ***artists*** 表を作成するとします。

```
CREATE TABLE artists …
```

この場合、作成される表の完全修飾名は ***arfchong.artists*** となります。

また **set schema** ステートメントを使用すると、セッションにスキーマを設定することができます。リスト 8.1 にその一例を示します。

```
connect to sample user arfchong using mypsw
select * from staff  ## This looks for arfchong.staff
set schema db2admin
select * from staff  ## This looks for db2admin.staff
```

リスト 8.1 - set schema ステートメントの使用例

スキーマの使用方法は「コンテスト・システム」を用いて説明することができます。例えば、ある企業がコンテストを開催しているとします。このコンテストの出場者は、独自の表を作成して SQL 操作を実行しなければなりません。すべての出場者は同じユーザー ID でデータベースに接続するようになっています。また、表を作成するために提供されているスクリプトも同じです。このスクリプトには、完全修飾名を持つオブジェクトは 1 つもありません。つまり、すべてのオブジェクトに、スキーマ名がないということです。出場者がコンテスト・システムにログオンすると、システムが接続後のタイムスタンプを基に、スキーマ名を生成します。このように、出場者 A は出場者 B と同じ名前で表を操作しますが、それぞれが持つスキーマは異なるため、2 人の操作が競合することはないというわけです。

8.2 パブリック・シノニム (パブリック別名)

パブリック・シノニム (パブリック別名) は、DB2 9.7 で新しく導入された概念です。パブリック・シノニムを使用すると、オブジェクトを参照する際にスキーマを指定する必要がなくなります。リスト 8.2 に一例を示します。

```
connect to sample user arfchong using mypsw
create public synonym raul for table arfchong.staff
select * from raul
select * from arfchong.raul  ## Error
connect to sample user db2admin using psw
select * from raul
```

リスト 8.2 – パブリック・シノニムの例

リスト 8.2 では、まずユーザー ***arfchong*** として接続し、表 ***arfchong.staff*** を参照するパブリック・シノニム ***raul*** を作成します。シノニム自体はスキーマを使用しません。スキーマを使おうとすると、エラーが返されることになります。リスト 8.2 の例で言うと、別のユーザー ***db2admin*** もパブリック・シノニムである ***raul*** を使用することができます。

上記の例でキーワード **public** を使用しない場合、作成されるシノニムはプライベート・シノニムになります。リスト 8.3 は上記と同じ例ですが、今度はプライベート・シノニムを使用しています。

```
connect to sample user arfchong using mypsw
create synonym raul for table arfchong.staff
select * from raul
select * from arfchong.raul  ## OK, it also works
connect to sample user db2admin using psw
select * from raul  ## Error, cannot find db2admin.raul
select * from arfchong.raul  ## OK, this works
```

リスト 8.3 – プライベート・シノニムの例

リスト 8.3 を見るとわかるように、シノニムはプライベートであるため、別のユーザーとして接続した場合、シノニムを参照するにはスキーマを指定しなければならなくなります。

8.3 表

表とは、関連するデータを論理的に列と行に配列したコレクションのことです。以下のリスト 8.4 には、**CREATE TABLE** ステートメントを使用して表を作成する例を示しています。

```
CREATE TABLE artists
 (artno           SMALLINT    not null,
  name            VARCHAR(50) with default 'abc',
  classification  CHAR(1)     not null,
  bio             CLOB(100K)  logged,
  picture         BLOB(2M)    not logged compact
 )
 IN mytbls1
```

リスト 8.4 - CREATE TABLE ステートメントの例

以降のセクションでは、この **CREATE TABLE** ステートメントの主要な部分について説明します。

8.3.1 データ型

DB2 インフォメーション･センターから抜粋した図 8.1 に、DB2 でサポートされるデータ型を記載します。

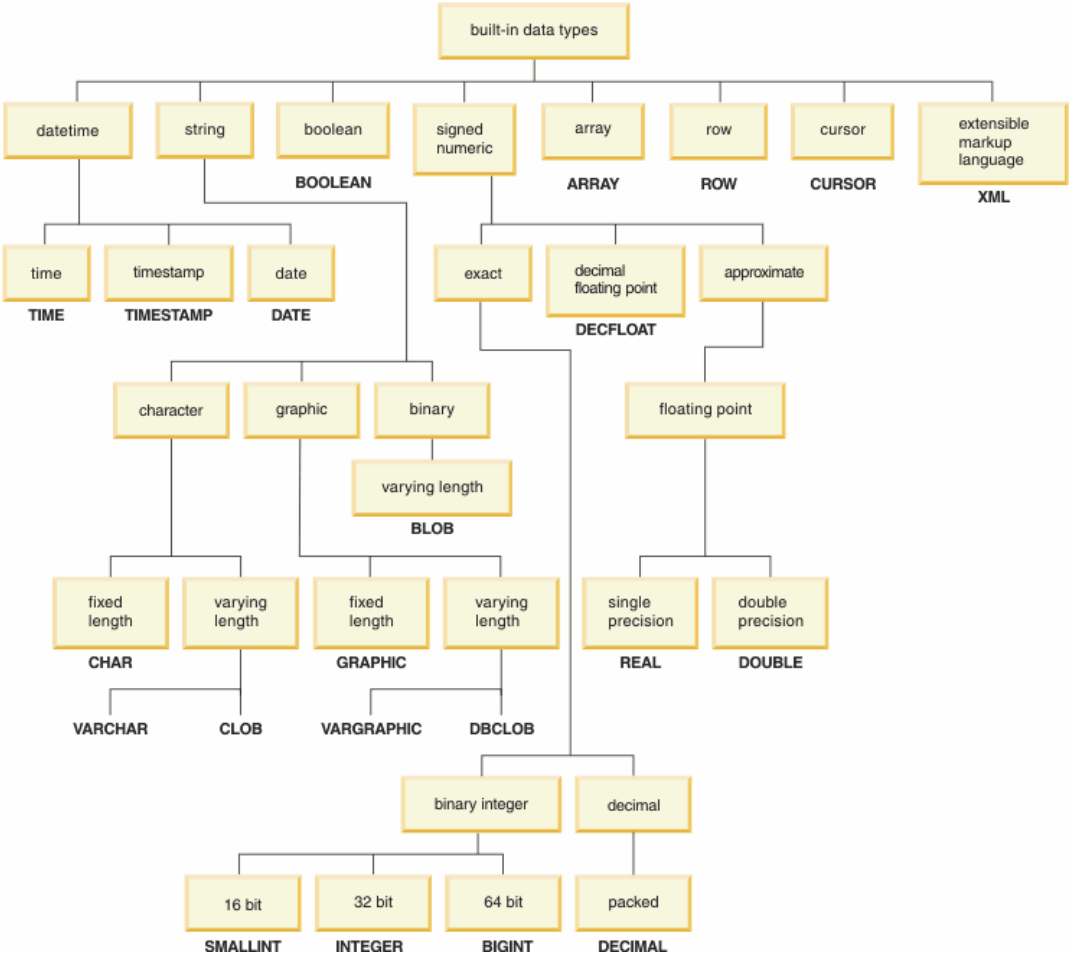


図 8.1 – DB2 の組み込みデータ型

図 8.1 に示されたデータ型については、DB2 マニュアルで詳しく説明しています。これらのデータ型のほとんどはリレーショナル・データベース管理システムの間で共通しているか、または非常によく似ているため、ここでは説明しません。その一方、初心者にとってすぐには理解しにくいラージ・オブジェクト (LOB) などのデータ型もあります。

ラージ・オブジェクト・データ型は、大規模な文字ストリングやバイナリー・ストリング、またはファイルを保管する場合に使用します (図 8.2 を参照)。

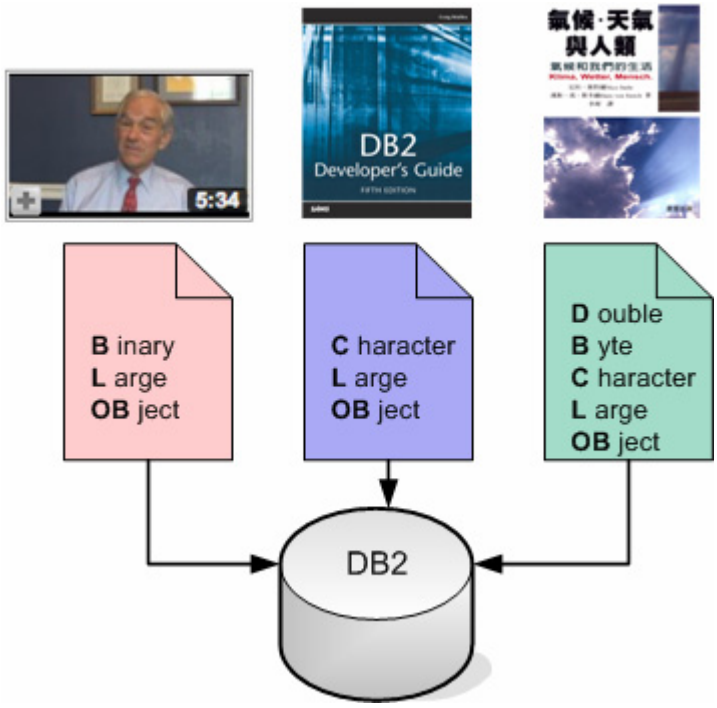


図 8.2 – LOB データ型

これらのラージ・オブジェクトのバイナリーは通常、簡潔にするために省略形で使用されます。バイナリー・ラージ・オブジェクトの省略形は BLOB、文字ラージ・オブジェクトの省略形は CLOB です。ダブルバイト文字ラージ・オブジェクトは DBCLOB としても知られています。

図 8.1 には DB2 9.7 で追加された以下の新しいデータ型が記載されています。

new in V9.7

- BOOLEAN
- ARRAY
- ROW
- CURSOR

これらのデータ型は Oracle データベース·サーバーで使われているデータ型の一部であり、現在 DB2 でサポートされているものです。Oracle データベース·サーバーで使われているデータ型に関しては、後ほどこの章で詳しく説明します。

8.3.1.1 ユーザー定義タイプ

DB2 では、ユーザー定義タイプ (UDT) を使用して独自のデータ型を定義できるようになっています。UDT は以下のように分類することができます。

- 特殊タイプ
- 構造化タイプ
- 参照タイプ

- 配列タイプ
- 行タイプ
- カーソル･タイプ

new in V9.7

参照タイプ、配列タイプ、行タイプ、そしてカーソル･タイプは、DB2 9.7 で新しく導入されたタイプであり、SQL PL ルーチンで使用することができます。ユーザー定義の特殊データ･タイプは組み込みデータ型をベースにしています。これらの UDT は、以下の場合に役立ちます。

- 値にコンテキストを設定しなければならない場合
- 強い型付けを使って DB2 にデータ型を定義させる必要がある場合

リスト 8.5 の SQL ステートメントで、特殊タイプの UDT を使用する方法と場合を説明します。

```
CREATE DISTINCT TYPE POUND AS INTEGER WITH COMPARISONS
CREATE DISTINCT TYPE KILOGRAM AS INTEGER WITH COMPARISONS
CREATE TABLE person
   (f_name    VARCHAR(30),
    weight_p  POUND NOT NULL,
    weight_k  KILOGRAM NOT NULL )
```

リスト 8.5 - 特殊データ･タイプの一例

上記の例では、POUND と KILOGRAM という 2 つの特殊タイプの UDT が作成されます。いずれも組み込みデータ型、INTEGER をベースに作成されています。この構文の一部として定義されている WITH COMPARISONS 節は、データ型と同じ名前のキャスト関数も作成するように指定します。

`person` 表はこの 2 つの新しい UDT をそれぞれ `weight_p` 列と `weight_k` 列で使用します。ここで、以下のステートメントを実行するとします。

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > weight_k
```

このステートメントでは異なるデータ型を持つ 2 つの列が比較されているため、エラー・メッセージを受け取ることになります。weight_p と weight_k はそれぞれ POUND データ型と KILOGRAM データ型を使用しますが、これらのデータ型は両方とも INTEGER データ型をベースに作成されたものであるため、UDT を作成することによって、このような比較は不可能になります。現実の世界ではポンドとキログラムを比較しても意味がありません。これがまさに、これらの UDT を作成した目的です。

次の例では、`weight_p` 列を整数と比較します。ただし、この 2 つのデータ型は異なるため、キャスト関数を使用しない限り、エラー・メッセージを受け取ることになります。

そのため、以下を見るとわかるように、このステートメントでは POUND() キャスト関数を使用しています。これで、比較が可能になります。前述のとおり、POUND() キャスト関数は CREATE DISTINCT TYPE ステートメントで WITH COMPARISONS 節を使用するときに UDT と併せて作成されたものです。

```
SELECT F_NAME FROM PERSON
```

```
WHERE weight_p > POUND(30)
```

new in V9.7

8.3.1.2 Oracle データベース･サーバーのデータ型

Oracle データベース･サーバーで使われている以下のデータ型は、現在では DB2 データ･サーバーでもサポートされています。NUMBER、VARCHAR2、TIMESTAMP(n)、DATE、BOOLEAN、INDEX BY、VARRAY、行タイプ、参照カーソル。これらを使用できるようにするには、以下のようにして DB2_COMPATIBILITY_VECTOR レジストリー変数が有効になるように最初に設定しておく必要があります。

```
db2set DB2_COMPATIBILITY_VECTOR=FF
db2stop
db2start
```

このレジスター変数が有効になるように設定すると、新しいデータベースでこれらのデータ型がサポートされるようになります。これらのデータ型のなかには、SQL PL を使用している状況でのみ使用できるものもあります。

注:

SQL Compatibility Feature (第 2 章 で説明) をサポートする DB2 のエディションをお使いの場合、PL/SQL を使用している場所でも、これらのデータ型を使用することができます。その場合、レジストリー変数 DB2_COMPATIBILITY_VECTOR に設定する値は、上記の例に示されている「FF」ではなく、「FFF」に設定してください。

new in V9.7

8.3.1.3 暗黙的キャストまたは弱い型付け

Ruby on Rails や PHP などの多くの動的言語では、暗黙的キャストが可能です。DB2 では強い型付けという要件があったため、今まで暗黙的キャストは問題となっていました。しかし DB2 9.7 では規則が緩和され、暗黙的キャスト、あるいは弱い型付けを使用することができます。これが何を意味するかと言うと、例えば以下のように、ストリングを数値型に割り当てたり、この 2 つを比較したりすることが可能になったということです。

```
create table t1 (col1 int)
select * from t1 where col1 = '42'
```

上記の例では、ストリング「42」を整数列 *col1* と比較することができます。

さらに、DB2 9.7 ではほとんどの場所で、型なしパラメーター・マーカーと型なし NULL を指定できるようになっています。以前は明示的に特定のデータ型にキャストする必要がありましたが、現在は、以下のようなステートメントも有効です。

```
select ?, NULL, myUDF(?, NULL) from t1
```

8.3.1.4 NULL 値

NULL 値は不明な状態を表します。CREATE TABLE ステートメントで、NOT NULL 節を使用して列を定義すると、列には確実に既知のデータ値が含まれることになります。また、NOT NULL が宣言されている列のデフォルト値を指定することもできます。以下のステートメントは、この振る舞いの一例です

```
CREATE TABLE staff (
     ID       SMALLINT NOT NULL,
     NAME     VARCHAR(9),
     DEPT     SMALLINT NOT NULL with default 10,
     JOB      CHAR(5),
     YEARS    SMALLINT,
     SALARY   DECIMAL(7,2),
     COMM     DECIMAL(7,2) with default 15
     )
```

この例では、***ID*** 列と ***DEPT*** 列が NOT NULL として定義されています。***DEPT*** 列には、値が提供されない場合のデフォルト値として 10 という値が指定されています。

8.3.2 ID 列

ID 列は数値の列で、挿入されたそれぞれの行に対してユニークな数値を自動的に生成します。表ごとに許容される ID 列の数は 1 つだけです。

ID 列の値は、この列がどのように定義されたかによって以下の 2 通りの方法で生成されます。

- **「Generated always」**： 値は常に DB2 によって生成されます。アプリケーションが明示的に値を指定することはできません。
- **「Generated by default」**： 値はアプリケーションによって明示的に指定することができます。値が指定されない場合には DB2 が値を生成しますが、DB2 では一意性を保証することができません。このオプションはデータの伝搬、および表のアンロードと再ロードを目的としています。

以下の例を見てください。

```
CREATE TABLE subscriber(
    subscriberID INTEGER GENERATED ALWAYS AS
        IDENTITY (START WITH 100 INCREMENT BY 100),
    firstname VARCHAR(50),
    lastname  VARCHAR(50) )
```

この例では、***subscriberID*** 列が INTEGER で、常に生成される ID 列として定義されています。生成されるのは 100 から始まる値で、その増分値は 100 です。

8.3.3 シーケンス・オブジェクト

シーケンス・オブジェクトは表とは独立していますが、ID 列と動作が似ているのでこのセクションで説明します。シーケンス・オブジェクトと ID 列との違いは、シーケンス・オブジェクトはデータベース全体でユニークな番号を生成しますが、ID 列は表内でユニークな番号を生成することです。以下のステートメントにシーケンス・オブジェクトの一例を示します。

```
CREATE TABLE t1 (salary int)

CREATE SEQUENCE myseq
   START WITH 10
   INCREMENT BY 1
   NO CYCLE

INSERT INTO t1 VALUES (nextval for myseq)

INSERT INTO t1 VALUES (nextval for myseq)

INSERT INTO t1 VALUES (nextval for myseq)

SELECT * FROM t1

SALARY
-----------
         10
         11
      12
  3 record(s) selected.

SELECT prevval for myseq FROM sysibm.sysdummy1

1
-----------
         12
  1 record(s) selected
```

リスト 8.6 - シーケンス・オブジェクトの一例

PREVVAL はシーケンスの現行の値を指定する一方、NEXTVAL はその次の値を指定します。上記の例では、SYSIBM.SYSDUMMY1 も使用しています。これは、1 つの列と 1 つの行からなるシステム・カタログ表です。この表は、クエリーが 1 つの値のみを出力として返すことを要求している場合に使用することができます。システム・カタログ表については、次のセクションで説明します。

8.3.4 システム・カタログ表

データベースにはそれぞれに固有のシステム・カタログ表とビューがあり、データベース・オブジェクトに関するメタデータ が保管されます。万一システム・カタログ表が破損した場合、データベースは使用不能と表示されることになります。システム・カタログ表は、通常のデータベース表と同じように照会することができますが、その際システム・カタログ表を識別するために使用するスキーマは、以下の 3 つです。

- SYSIBM: 基本表。DB2 用に最適化されます。
- SYSCAT: SYSIBM 表をベースとしたビュー。各用途に合わせて最適化されます。
- SYSSTAT: データベース統計

以下は、カタログ・ビューの例です。

- SYSCAT.TABLES
- SYSCAT.INDEXES
- SYSCAT.COLUMNS
- SYSCAT.FUNCTIONS
- SYSCAT.PROCEDURES

8.3.5 宣言済みグローバルー時表 (DGTT)

宣言済みグローバルー時表とは、メモリー内に作成される表のことで、アプリケーションで使用された後、アプリケーションの終了と同時に自動的にドロップされます。宣言済みグローバルー時表にはその表を作成したアプリケーションしかアクセスできないため、DB2 カタログ表の中に宣言済みグローバルー時表が存在することはありません。これらの表を使用すると、カタログ競合や行のロックが発生することなく、デフォルト・ロギング (ロギングはオプション) や権限チェックも行われないため、パフォーマンスが大幅に効率化されます。また、宣言済みグローバルー時表では索引がサポートされます。つまり、宣言済みグローバルー時表であらゆる標準索引を作成できるということです。さらに表に対して RUNSTATS を実行することもできます。

宣言済みグローバルー時表はユーザーー時表スペース内に置かれます。そのため、宣言済みグローバルー時表を作成する前に、ユーザーー時表スペースが定義されていなければなりません。リスト 8.7 のステートメントは、3 つの宣言済みー時表を作成する場合の例です。

```
CREATE USER TEMPORARY TABLESPACE apptemps
     MANAGED BY SYSTEM USING ('apptemps');

DECLARE GLOBAL TEMPORARY TABLE temployees
     LIKE employee NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempdept
      (deptid CHAR(6), deptname CHAR(20))
       ON COMMIT DELETE ROWS NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempprojects
     AS ( fullselect ) DEFINITION ONLY
     ON COMMIT PRESERVE ROWS NOT LOGGED
     WITH REPLACE IN TABLESPACE apptemps;
```

リスト 8.7 – DGTT を扱う

宣言済みグローバルー時表を作成するときのスキーマは SESSION になるため、DGTT を参照する際にはこのスキーマを指定する必要があります。作成した宣言済みグローバルー時表では、作成の際に使用したユーザー ID がすべての特権を持つことになります。またアプリケーションが宣言済みグローバルー時表を作成すると、アプリケーションそれぞれにその表の独自のコピーを持つことになります (図 8.3 を参照)。

図 8.3 – 宣言済みグローバルー時表のスコープ

リスト 8.8 に宣言済みグローバルー時表のスコープ制限を示します。このリストでは、ユーザーー時表スペースが作成済みであることを前提としています。

DB2 コマンド・ウィンドウ #1 より:

```
db2 connect to sample
db2 declare global temporary table mydgtt (col1 int, col2 varchar(10)) on
commit preserve rows
db2 insert into session.mydgtt values (1,'hello1'),(2,'hello2'),
(3,'hello3')
db2 select * from session.mydgtt

COL1        COL2
----------- ----------
          1 hello1
          2 hello2
          3 hello3
3 record(s) selected.
```

DB2 コマンド・ウィンドウ #2 より:

```
db2 connect to sample
db2 select * from session.mydgtt
SQL0204N  "SESSION.MYDGTT" is an undefined name.  SQLSTATE=42704
```

リスト 8.8 - DGTT のスコープを扱う

リスト 8.8 を見るとわかるように、2 番目のセッション (DB2 コマンド・ウィンドウ #2) で ***SESSION.MYDGTT*** を使おうとすると、このセッションでは DGTT が定義されていないため、エラーが返されます。DGTT 定義では ON COMMIT PRESERVE ROWS 節を使用することに注意してください。これはデフォルトでは、DB2 コマンド・ウィンドウでの操作は入力されたステートメントごとにコミットするためです。

new in V9.7

8.3.6 作成済みグローバル一時表 (CGTT)

DGTT で一時表を作成することはできても、その表の定義を異なる複数の接続またはセッションで共有することはできません。そのため、毎回セッションが確立されるたびに、DECLARE GLOBAL TEMPORARY TABLE ステートメントを実行しなければなりません。一方、作成済みグローバル一時表 (CGTT) では一時表の定義が DB2 カタログに恒久的に保管されるため、一時表を 1 度作成するだけで済みます。つまり、他の接続もその一時表を使えるため、一時表を再び作成する手間を省けることになります。表の構造はすぐに使用できますが、データは接続ごとにそれぞれに独立しているため、接続が閉じるとデータは消去されます。一例として、リスト 8.9 を見てください。ここでは、ユーザー一時表スペースがすでに作成されていることを前提とします。

DB2 コマンド・ウィンドウ #1 より:

```
db2 connect to sample
db2 create global temporary table mycgtt (col1 int, col2 varchar(10)) on
commit preserve rows
db2 insert into mycgtt values (1,'hello1'),(2,'hello2'), (3,'hello3')
db2 select * from mycgtt

COL1        COL2
----------- ----------
          1 hello1
          2 hello2
          3 hello3
3 record(s) selected.
```

DB2 コマンド・ウィンドウ #2 より:

```
db2 connect to sample
db2 select * from mycgtt

COL1        COL2
----------- ----------
  0 record(s) selected.
```

リスト 8.9 - CGTT のスコープを扱う

リスト 8.9 では、DB2 コマンド・ウィンドウ #2 (別のセッションまたは接続) のなかで CGTT をもう一度作成する必要はなく、単に CGTT を参照すればよいだけです。ただし、データは最初のセッションに特有のものであったため、行は返されていません。

8.4 ビュー

ビューとは、表内から抽出したデータを表示したものです。ビューのデータは別途保管されるのではなく、ビューを起動したときに取得されます。DB2 ではネストされたビュー (他のビューをベースに作成されたビュー) もサポートしています。ビューに関するすべての情報は、SYSCAT.VIEWS、SYSCAT.VIEWDEP、および SYSCAT.TABLES という DB2 カタログ・ビューに保持されます。リスト 8.10 に示すのは、ビューを作成し、そのビューを利用する方法の一例です。

```
CONNECT TO MYDB1;

CREATE VIEW MYVIEW1
  AS SELECT ARTNO, NAME, CLASSIFICATION
  FROM ARTISTS;

SELECT * FROM MYVIEW1;

Output:

ARTNO            NAME        CLASSIFICATION
------     -------------     --------------
  10       HUMAN             A
  20       MY PLANT          C
  30       THE STORE         E
  ...
```

リスト 8.10 – ビューを扱う

8.5 索引

索引とは、順序付けられたキーのセットのことで、それぞれのキーが表内の一行を指します。索引は一意性をもたらすとともに、パフォーマンスを向上させます。索引に定義できる特性には、例えば以下のものがあります。

- 索引の順序を昇順にするか、降順にするか
- 索引キーをユニークにするか、非ユニークにするか
- 索引に複数の列を使用できるようにするかどうか (複合索引と呼ばれます)
- 索引と物理データが同様の索引シーケンスでクラスター化されている場合、この 2 つをまとめてクラスター索引とするかどうか

以下はその一例です。

```
CREATE UNIQUE INDEX artno_ix ON artists (artno)
```

8.5.1 設計アドバイザー

設計アドバイザーは、特定の SQL のワークロードが最適となるようなデータベースを設計する際に、アドバイスを提供してくれる優れたツールです。設計アドバイザーは、索引、マテリアライズ照会表 (MQT)、多次元クラスタリング (MDC)、データベース・パーティション化機能などを設計する際にも使用することができます。設計アドバイザーをコントロール・センターから起動するには、データベースを右クリックして「設計アドバイザー」を選択します (図 8.4 を参照)。

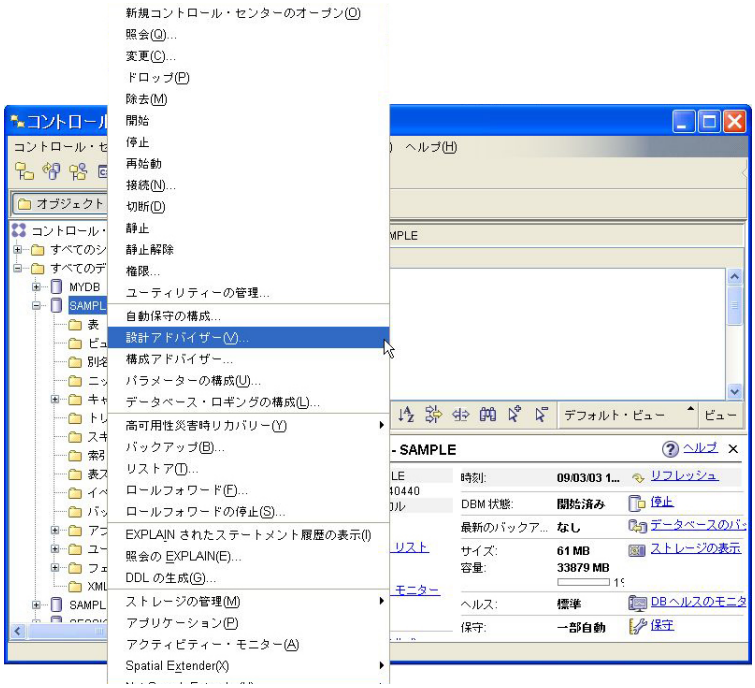


図 8.4 – コントロール・センターから設計アドバイザーの起動

図 8.7 に、設計アドバイザーを示します。このウィザードのステップに従うと、DB2 からの設計推奨案を得ることができます。

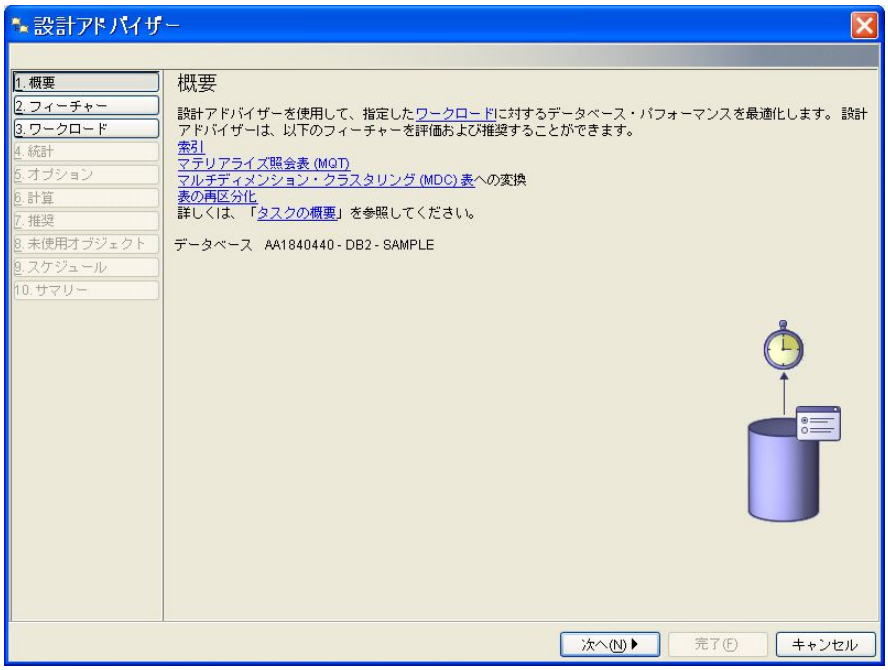


図 8.5 – 設計アドバイザー

8.6 参照整合性

データベースは参照整合性によって、表の相互関係を管理します。図 8.6 に示すように、表の間には親子タイプの関係を確立することができます。この図には DEPARTMENT と EMPLOYEE という 2 つの表があり、部門番号によって相互に関連付けられています。EMPLOYEE 表の WORKDEPT 列に含められる部門番号は、DEPARTMENT 表にすでに存在する部門番号のみです。そのため、この例では DEPARTMENT 表が親表で、EMPLOYEE 表がその子 (従属表) ということになります。この図には、関係を確立するために必要となる、EMPLOYEE 表に対する CREATE TABLE ステートメントも示されています。

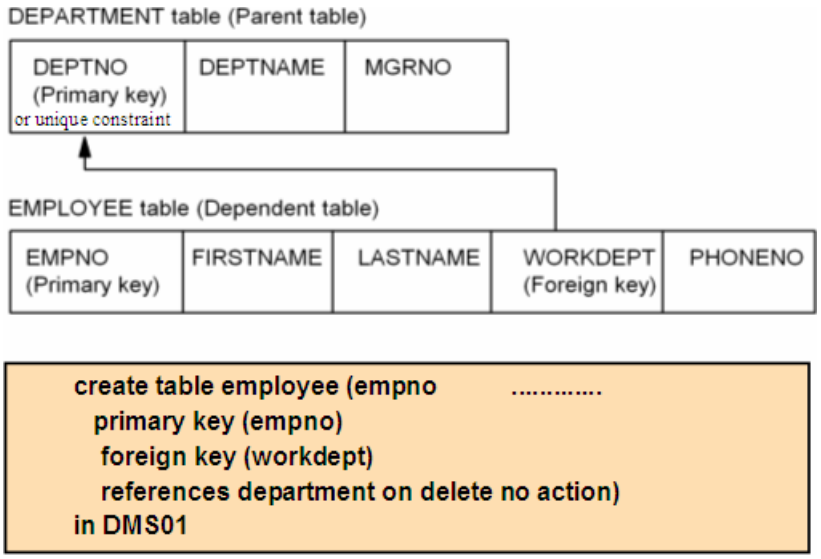


図 8.8 – 表間の参照整合性の例

参照整合性でよく使用される概念には表 8.1 に挙げたものがあります。

概念	説明
親表	親キーが存在する制御データ表
従属表	親表のデータに依存する表。この表には外部キーも含まれます。従属表に行が存在するためには、それに対応する行が親表に存在していなければなりません。
主キー	親表の親キーを定義します。NULL 値を含めることはできません。また、値はユニークでなければなりません。主キーは、表を構成する 1 つ以上の列からなります。
外部キー	親表の主キーを参照します。

表 8.1 – 参照整合性の重要概念

表内のデータは、参照整合性を持つ 1 つ以上の表に関連付けることができます。データ値に制約を設けて、特定のプロパティーまたはビジネス・ルールに準拠させるようにすることも可能です。例えば、表の列に個人の性別を保管する場合、男性には「M」の値、女性には「F」の値のみを許容するという制約を設定することができます。

new in V9.7

8.7 スキーマの進化

ビジネス・ニーズが変化したときには、それをサポートする情報技術 (IT) インフラストラクチャーとシステムも同じく変更しなければなりません。データベースの世界で言えば、新しい表の作成、既存の表のドロップまたは変更、トリガー・ロジックの変更などの作業が必要になってくるということです。これらの変更を行うのは簡単なように思えますが、実際には困難で複雑な作業になることがあります。そして長い保守期間が必要となり、複雑でリスクを伴うプロシージャーも実行しなければなりません。これまでビジネス・ニーズに応じたデータベースの変更が難しかった理由の 1 つは、DB2 ではすべてのオブジェクトが常に一貫している必要があったためです。そのため、オブジェクトを変更することによって、その従属オブジェクトにも影響するようなアクションは許可されていなかったか、あるいはそのようなアクションによって従属オブジェクトがドロップされる結果となっていました。図 8.7 にサンプル・シナリオを示します。

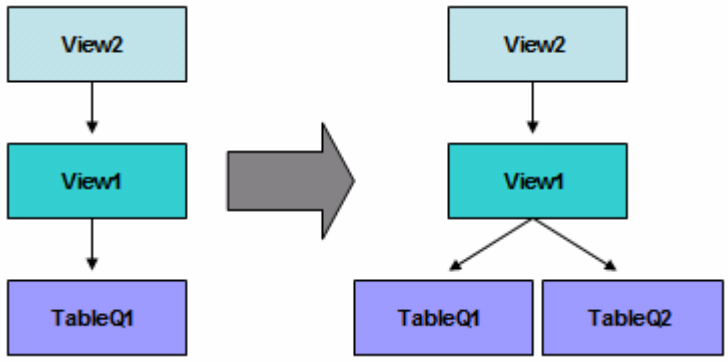


図 8.7 – スキーマの進化のサンプル・シナリオ

図 8.7 では、***View1*** は企業の第 1 四半期の財務情報が保管された ***TableQ1*** をベースとしていますが、例えばこのビューの定義を変更して、第 1 四半期、第 2 四半期の情報がそれぞれ保管された ***TableQ1*** と ***TableQ2*** の両方をベースにするようにしなければならないとします。この場合、通常は ***View1*** をドロップし、新しい定義を使用してビューを作成し直すことになりますが、***View2*** は ***View1*** に依存しています。DB2 9.7 より前の DB2 では、従属オブジェクト ***View2*** があるために、***View1*** のドロップは許可されません。そこで、まず ***View2*** をドロップしてから、***View1*** をドロップし、それから再びこの 2 つのビューを構成する必要がありました。DB2 9.7 では規則が緩和され、従属オブジェクトに影響する変更でも許可されるようになっています。従属オブジェクト (この例では ***View2***) を使用する前には、その有効性を再確認しなければなりませんが、この再確認は自動的に行われます。これは、**自動再確認**として知られる機能です。自動再確認の有効、無効を切り替え、再確認を行うタイミングを決定するには、db cfg パラメーター AUTO_REVAL を使用します。例えばこのパラメーターを DEFERRED_FORCE に設定すると、無効なオブジェクトまたは従属オブジェクトがアクセスされるまで、再確認は遅延されます。一方、まだ存在していない従属オブジェクトに対しては、CREATE ステートメントを使用することができます (ただし、警告が出されます)。

従属モデルに影響するその他の変更には、ビュー、関数、プロシージャー、トリガー、別名などを対象にした CREATE OR REPLACE 構文のようなフィーチャーの実装もあります。以下はその例です。

```
create or replace procedure p1 begin ... end
```

この構文では、オブジェクト (例えば procedure p1) が存在していない場合には、そのオブジェクトが作成されます。既存のオブジェクトがある場合には、それが置き換えられます。オブジェクトの依存性にとって重要なのは、この 2 番目の振る舞いです。そこで、p1 が置き換えられると、p1 に依存するオブジェクトの有効性が自動的に再確認されます。RENAME COLUMN などの新しいフィーチャーや、より多くのデータ型の変更をサポートするように拡張された ALTER COLUMN によるデータ型の変更でも、同じような再確認が行われます。

これに関連する、**ソフト無効化**として知られる概念により、ユーザーは他の実行中のトランザクションが使用しているオブジェクトでもドロップできるようになっています。すると、新しいトランザクションでは、ドロップされたオブジェクトへのアクセスが拒否されます。

8.8 まとめ

この章では DB2 のデータベース・オブジェクトを重点に、データベース・オブジェクトとは何か、そしてこれらのオブジェクトをどのように作成し、使用するかについて説明しました。まず始めに紹介したデータベース・オブジェクトはデータベース・スキーマです。そして、スキーマと新たに導入されたパブリック・シノニムとの違い、さらにパブリック・シノニムとプライベート・シノニムとの違いを説明しました。

次に、表とその要素について詳細に説明するなかで、データ型 (組み込みデータ型とユーザー定義データ型の両方)、ID 列、シーケンス・オブジェクト、グローバル一時表のそれぞれを取り上げました。続いて説明したのは、ビュー、索引、そして「設計アドバイザー」の GUI を使用して、表内のデータのアクセスしやすさ、取得しやすさを向上させる方法です。

最後に、表の相互関係を定義する参照整合性と、不必要に複雑な作業にすることなくデータ・オブジェクトを変更できるようにする新しい概念、スキーマの進化について検討しました。

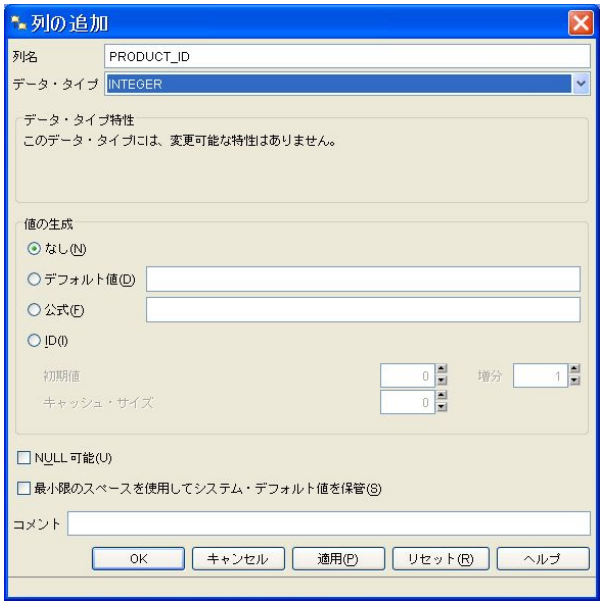
8.9 演習　新しい表を作成する

これまでは、**SAMPLE** データベースに含まれる既存の表を使って概念を説明してきました。しかし最終的には、データベース内に独自の表を作成する必要が出てきます。この演習では、表の作成ウィザードを使用して 2 つの新しい表を **SAMPLE** データベースに作成します。

手順

1. この章で説明した方法で表の作成ウィザードを起動します (「コントロール・センター」 -> 「すべてのデータベース」 -> 「SAMPLE」の順に選択し、「表」オブジェクトを右クリックして「作成」を選択します)。

2. 表の名前、列の定義、その他の制約を定義します。この表は、**SAMPLE** データベース内のプロジェクトで使用される事務用品に関する情報を保管するために使用するものです。事務用品を購入するたびに、この表に行が追加されます。表には以下の 6 つの列があります。
   - product_id: 購入する製品のユニークな ID
   - description: 製品の説明
   - quantity: 購入する数量
   - cost: 製品のコスト
   - image: 製品の写真 (入手可能な場合)
   - project_num: 製品を購入することとなったプロジェクト
3. ウィザードの最初のページでは、スキーマ名として現在ログオンしているユーザー ID を入力し、表の名前として ***SUPPLIES*** と指定します。また、オプションで説明を入力することもできます。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
4. このページから、表に列を追加することができます。列を追加するには「追加」ボタンをクリックします。



5. 「列名」に ***PRODUCT_ID*** と入力し、「データ・タイプ」は「**INTEGER**」を選択します。「NULL 可能」のチェック・マークを外し、「適用」ボタンをクリックして列を定義します。

6. 表の残りの列に対して上記のステップを繰り返します。それぞれの列には、以下の表に記載するオプションを使用してください。すべての列を追加 (「適用」ボタンをクリック) したら、「OK」ボタンをクリックします。すると、作成した列を要約したリストが表示されます。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。

列名	**属性**
product_id (完了)	INTEGER、NOT NULL
description	VARCHAR、length 40、NOT NULL
quantity	INTEGER, NOT NULL
cost	DECIMAL、Precision 7、Scale 2、NOT NULL
image	BLOB、1MB、NULLABLE、NOT LOGGED
project_num	CHAR, length 6, NOT NULL

注: NOT LOGGED オプションは、LOB 列を宣言する場合に指定することができます。サイズが 1GB を上回る列には、このオプションを必ず指定してください。また、サイズの大きな列が変更されるとログ・ファイルがすぐにいっぱいになってしまうため、このオプションは、10MB を上回る LOB 列にも一般的に推奨されます。NOT LOGGED を指定しているとしても、トランザクション中に行われた LOB ファイルへの変更は正常にロールバックすることができます。さらに、「NULL 可能」列として定義されているのは唯一、image 列だけであることにも注意してください。この列がなぜこのように定義されたかを考えてください。

7. この時点で、表を作成するために必要なすべての情報が指定されました。ウィザードの他のパネルはスキップすることで、これらのページのオプションにはデフォルト値を選択しています。キーと制約に関しては、表が作成されてからでも追加することができます。

8. 表に、quantity 列の値を制限するための制約を追加します。ウィザードの「制約」ページで、「追加」ボタンをクリックしてください。「制約名」には ***valid_quantities*** と入力し、「チェック条件」フィールドには ***quantity > 0*** と入力します。

   「OK」ボタンをクリックします。すると、ウィザードの「制約」ページで追加した制約の要約が表示されます。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。

9. ウィザードをこのまま続行すると、表のその他のパラメーターを変更することができます。あるいは、「サマリー」ページまでスキップすることも、単に「完了」ボタンをクリックして表を作成することもできます。

10. コントロール・センターのオブジェクト・ツリー・ペインに表示された **SAMPLE** データベースの下にある「表」フォルダーをクリックします。上記の手順で作成した表がリストに表示されているはずです。変更を表示するためには、「コントロール・センター」ビューをリフレッシュしなければならない場合もあります。

11. ここで、**SAMPLE** データベースの **STAFF** 表を使用して暗黙的なキャストのテストを行ってみましょう。そのためには、以下のとおりに実行します。

```
C:\>db2 describe table staff
```

ID 列は SMALLINT として定義されていることに注意してください。

```
C:\>db2 select * from staff where id = '350' --> '350' はストリングであることに注意してください。
C:\>db2 select * from staff where id =  350  --> 350 は数値であることに注意してください。
```

いずれの場合も、出力は以下のようになるはずです。

```
ID     NAME       DEPT   JOB   YEARS  SALARY    COMM
------ ---------- ------ ----- ------ --------- ---------
   350 Gafney         84 Clerk      5  43030.50    188.00
```

'350' をストリングとして使用した最初の SELCET ステートメントでは、DB2 は数値 (SMALLINT) への暗黙的なキャストを行っています。

9

第 9 章 – データ移動ユーティリティー

この章では、同じデータベース内、あるいは同じプラットフォームや異なるプラットフォームのデータベース間でデータを移動するために使用するツールやコマンドについて説明します。図 9.1 に、データ移動ユーティリティーの概要を示します。

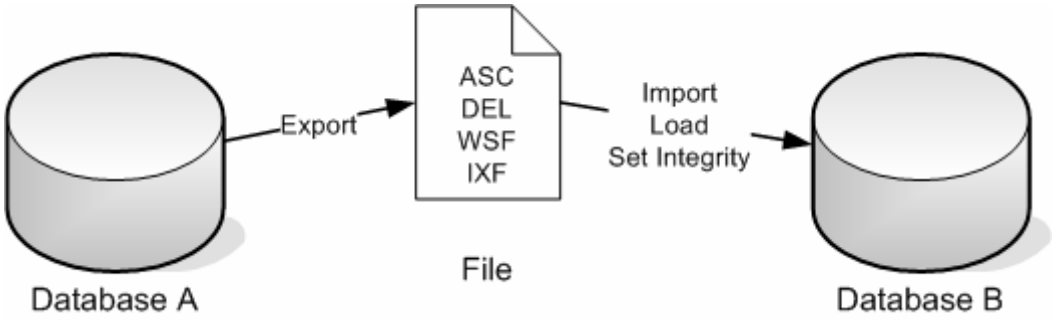


図 9.1 – データ移動ユーティリティー

図 9.1 には 2 つのデータベース、A と B があります。表のデータをファイルにエクスポートするには、エクスポート・ユーティリティーを使用します。エクスポートするファイルの形式は、以下のどの形式にすることもできます。

- ASC = ASCII
- DEL = 区切り文字付き ASCII
- WSF = ワークシート形式
- IXF = Integrated Exchange Format (統合交換フォーマット)

ASC および DEL ファイルはテキスト・ファイルで、任意のテキスト・エディターで開いて確認することができます。WSF は、データを Excel や Lotus® 1-2-3 などのスプレッドシートに移すことのできるファイル形式です。そして最後の IXF には、データだけでなく、対象とする表のデータ定義言語 (DDL) も含めることができます。表を再構成しなければならない場合には、IXF 形式でエクスポートされたファイルを使うと表を直接再構成することができるので便利ですが、IXF 以外のファイル形式では、表を直接再構成することはできません。

データをファイルにエクスポートした後は、インポート・ユーティリティーを使用してファイルのデータを別の表にインポートします。ファイル形式が ASC、DEL、WSF の場合、インポートする際には既存の表がなければなりませんが、IXF 形式の場合、その必要はありません。データを表にロードするには、ロード・ユーティリティーを使用することもできます。ロード・ユーティリティーは、

DB2 エンジンを介さずに直接データベース・ページにアクセスするため、高速な処理が可能になりますが、その一方で制約があるかどうかのチェックは行わないので、トリガーが起動されることはありません。ロード・ユーティリティーによってデータをロードした後は、データの整合性を保証するために SET INTEGRITY コマンドが使用されることがよくあります。

以降のセクションでは、エクスポート、インポート、ロードの各ユーティリティーについて詳しく説明します。

9.1 エクスポート・ユーティリティー

エクスポート・ユーティリティーは、前に説明したように表のデータをファイルに抽出するために使用します。実際に実行されているのは、SQL SELECT 操作です。以下は、***employee*** 表の 10 行を IXF 形式の `employee.ixf` ファイルにエクスポートする例です。

```
EXPORT TO employee.ixf OF IXF
   SELECT * FROM employee
   FETCH FIRST 10 ROWS ONLY
```

上記の例を試してみてください。employee 表は、前の章で作成した **SAMPLE** データベースに含まれているので、まずはこのデータベースに接続する必要があります。

GUI ツールを使用した操作をお望みの場合は、「コントロール・センター」からエクスポート・ユーティリティーを起動することもできます (図 9.2 を参照)。

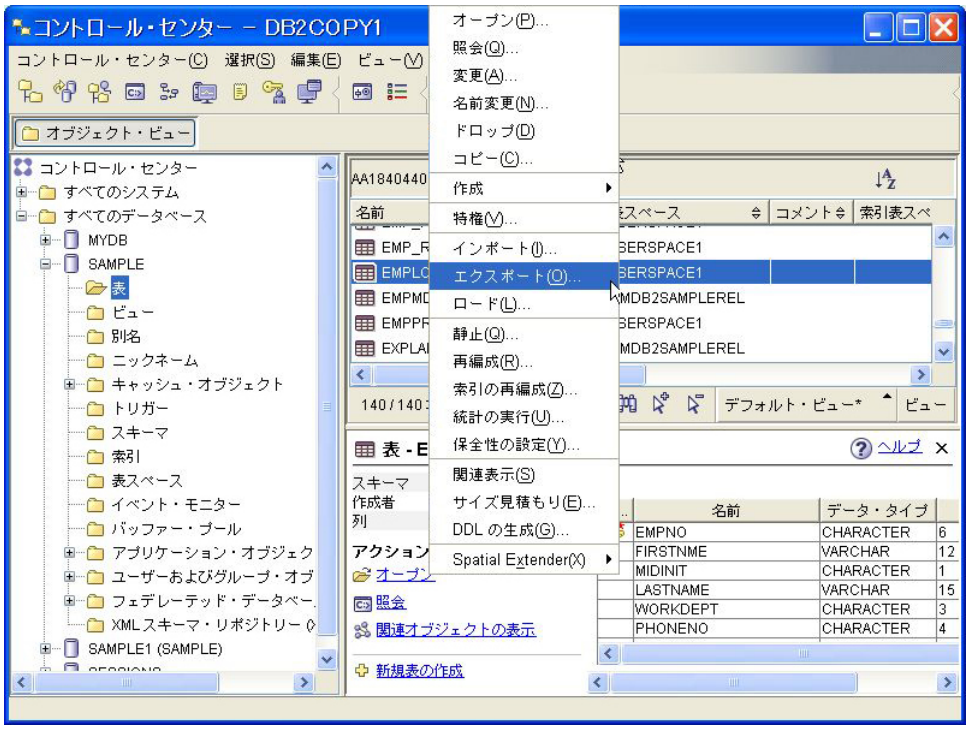


図 9.2 – 表のエクスポート・ダイアログの起動

上記の図に示されているように、まず employee 表を 1 回クリックして選択します。選択された状態の表を右クリックしてポップアップ・メニューを表示し、そこから「エクスポート」オプションを選択します。このオプションを選択するとウィザードが表示されます。このウィザードのステップに従いさえすれば操作は完了します。

9.2 インポート・ユーティリティー

インポート・ユーティリティーは、前に説明したようにデータをファイルから表にロードするために使用します。裏で実際に実行されているのは、SQL INSERT 操作です。INSERT 操作の実行中には、あらゆるトリガーがアクティブになり、すべての制約が即時に施行され、データベース・バッファー・プールが使用されます。以下に、IXF 形式の `employee.ixf` ファイルから、すべてのデータを ***`employee_copy`*** 表にロードする例を記載します。この例を実行してみることをお勧めしますが、そのためには前のセクションでエクスポート・ユーティリティーを実行済みである必要があります。

```
IMPORT FROM employee.ixf OF IXF
   REPLACE_CREATE
   INTO employee_copy
```

REPLACE_CREATE オプションは、インポート・ユーティリティーで使用できる数多くのオプションのうちの 1 つです。このオプションは、インポート・ユーティリティーが実行された時点で既に

employee_copy 表が存在している場合にはその表の内容を置き替え、まだ存在しない場合には新規に ***employee_copy*** 表を作成してデータをロードします。

コントロール・センターから操作する場合、インポート・ユーティリティーを起動するには、任意の表を選択してからその表を右クリックし、「インポート」を選択します (図 9.3 を参照)。

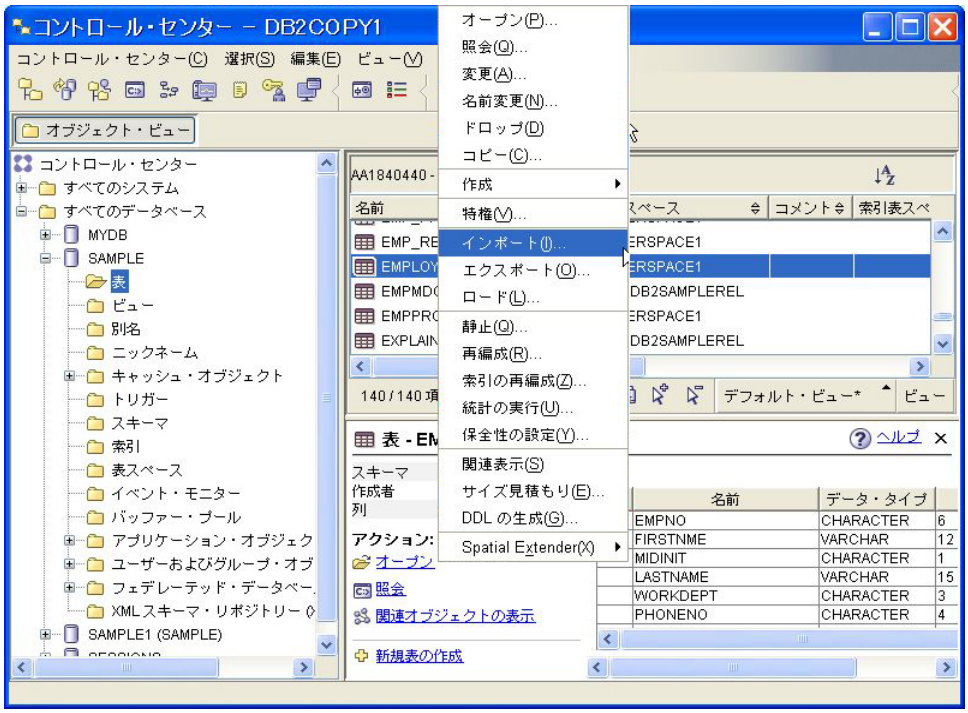


図 9.3 – インポート・ダイアログの起動

9.3 ロード・ユーティリティー

ロード・ユーティリティーはデータをファイルから表に素早くロードする方法となります。前にも説明したとおり、ロード・ユーティリティーは DB2 エンジンを介さないことから、トリガーはアクティブにならず、バッファー・プールも使用されません。また、制約は別のステップとしてしか施行することができません。その一方、ロード・ユーティリティーはディスク上のデータ・ページに直接アクセスする下位レベルのデータ・ローダーなので、その動作は IMPORT ユーティリティーよりも高速です。動作フェーズには、LOAD、BUILD、DELETE の 3 つがあります。

以下に、IXF 形式の `employee.ixf` ファイルから、すべてのデータを ***employee_copy*** 表にロードする例を記載します。REPLACE オプションは、ロード・ユーティリティーで使用できる数多くのオプションのうちの 1 つです。この例では、***employee_copy*** 表のすべてのコンテンツを置き換えるために、このオプションを使用しています。

```
LOAD FROM employee.ixf OF IXF
   REPLACE INTO employee_copy
```

上記のコマンドを実行すると、表が置かれている表スペースは CHECK PENDING 状態になるかもしれません。これは、SET INTEGRITY コマンドを実行してデータの整合性をチェックする必要があることを意味します。このコマンドの実行方法は以下のとおりです。

```
SET INTEGRITY FOR employee_copy
   ALL IMMEDIATE UNCHECKED
```

コントロール・センターを利用した方法をお望みならば、図 9.4 と図 9.5 に示すようにしてロード・ユーティリティーと SET INTEGRITY ユーティリティーをそれぞれ起動することができます。

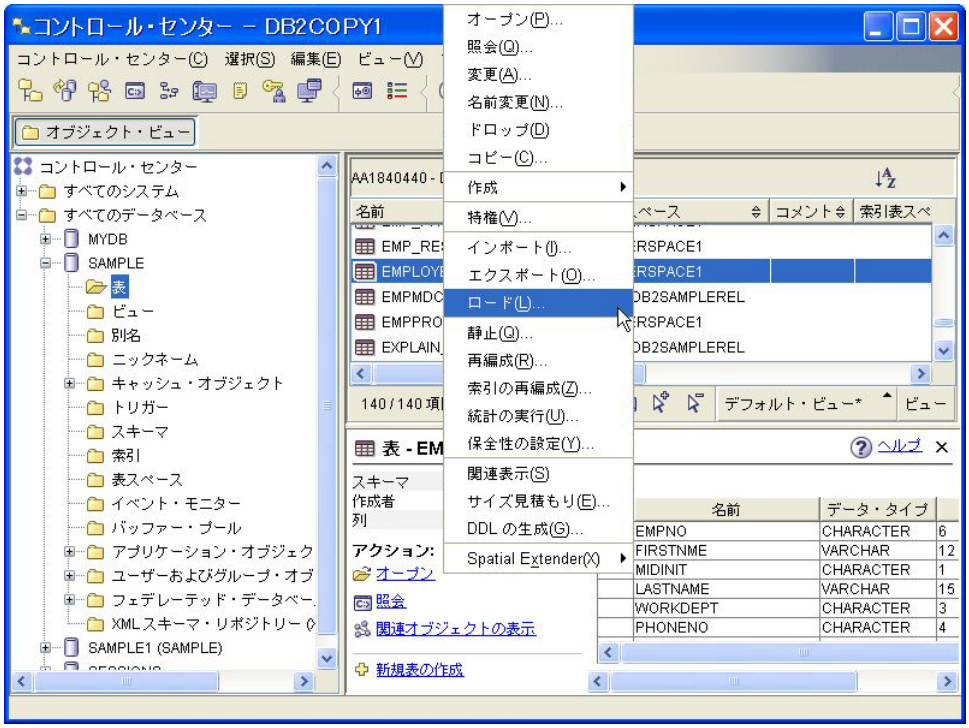


図 9.4 – ロード・ユーティリティーの起動

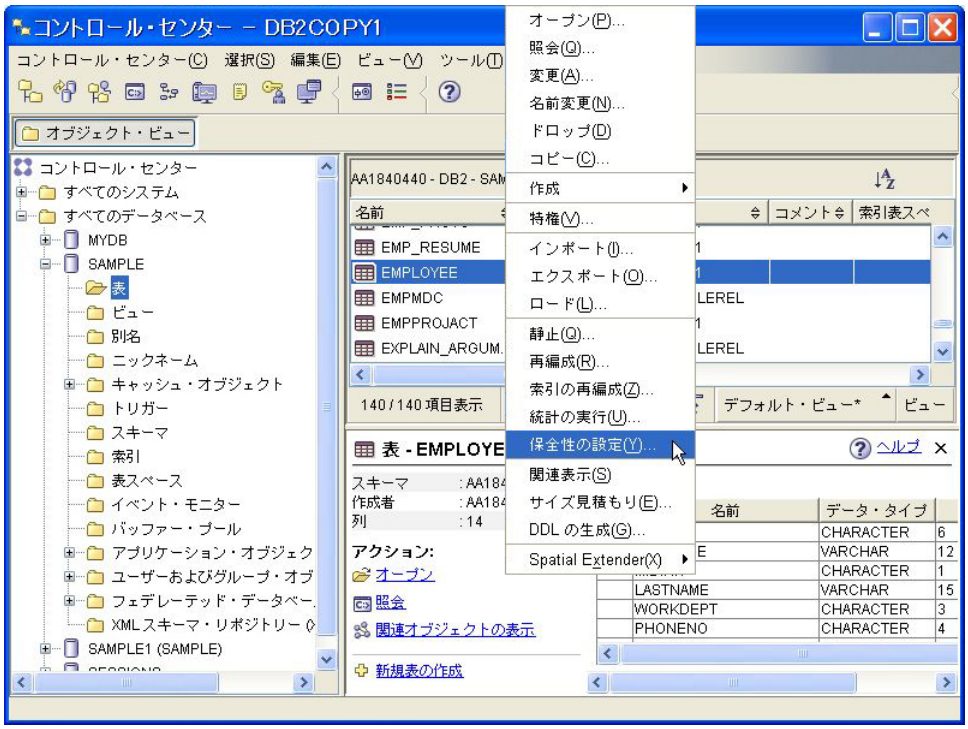


図 9.5 – 保全性の設定ウィザードの起動

9.4 db2move ユーティリティー

エクスポート、インポート、ロードの各ユーティリティーは、いずれも一度に 1 つの表しか処理しません。スクリプトを作成してデータベースに含まれる表ごとに上記のコマンドを生成することも可能ですが、そのようなスクリプトを作成する代わりに、**db2move** というユーティリティーを使用することができます。**db2move** ユーティリティーが処理できるのは IXF ファイルのみです。ファイル名は、**db2move** によって自動的に生成されます。以下の 2 つの例は SAMPLE データベースを使用した例で、それぞれ export オプションと import オプションを指定して **db2move** を実行する方法を示しています。

```
db2move sample export
db2move sample import
```

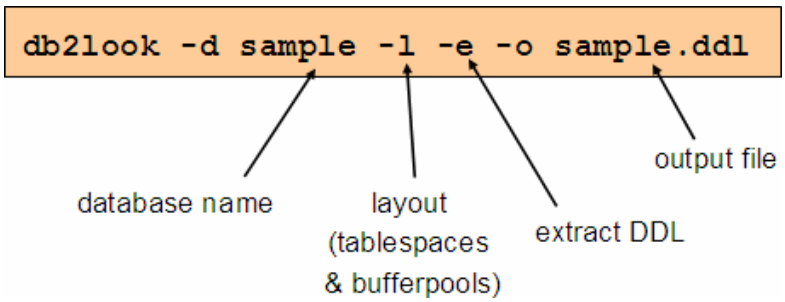
コントロール・センターからは **db2move** を実行することはできません。

9.5 db2look ユーティリティー

エクスポート、インポート、ロード、**db2move** の各ユーティリティーでは表から表へのデータ移動を同じデータベース内、または複数のデータベース間で実行できる一方、**db2look** ユーティリティーを使用すると、DDL ステートメントとデータベース統計、そしてデータベースの表スペースの特性を抽出し、後で別のシステムで実行できるようにスクリプト・ファイルに保管することができます。例えば、Linux 上で稼働している DB2 サーバーのデータベースを、Windows で稼働している DB2 サーバーへ複製しなければならないとします。この場合、まずは DB2 Linux サーバーで **db2look** ユーティリティーを実行して、データベースの構造を取得し、この構造をスクリプト・ファイルに保管します。このスクリプト・ファイルを DB2 Windows サーバーにコピーしてスクリプトを実行し、複製データベースの構築を開始すると、この時点でデータベースの構造が複製されます。次のステップでは、DB2 Linux サーバーで export オプションを指定して **db2move** ユーティリティーを実行します。これによって生成されたすべてのファイルを DB2 Windows サーバーにコピーし、さらに import または load オプションを指定して **db2move** を実行します。この作業が完了すると、異なるプラットフォームの別のサーバーへデータベースが完全に複製されています。

以上のシナリオが必要となるのは、Linux と Windows というように、それぞれ異なるプラットフォームに置かれたデータベースを操作する場合です。両方のサーバーが同じプラットフォーム上で稼働している場合には、バックアップ・コマンドとリストア・コマンドを使用したほうが、単純明快なプロセスになります。**バックアップ**および**リストア**・コマンドについては、以降の章で詳しく説明します。

以下の例は、**SAMPLE** データベースから表スペースおよびバッファー・プールのレイアウト、そして DDL ステートメントを抽出し、`sample.ddl` ファイルに保管する例です。以下のコマンドを実行して、出力テキスト・ファイル `sample.ddl` の内容を調べてみてください。



db2look コマンドのオプションはあまりにも数が多いので本書では説明しませんが、–h フラグを指定すると、使用可能なオプションの簡単な説明が表示されます。

```
db2look -h
```

db2look ユーティリティーは、コントロール・センターからも起動することができます (図 9.6 を参照)。

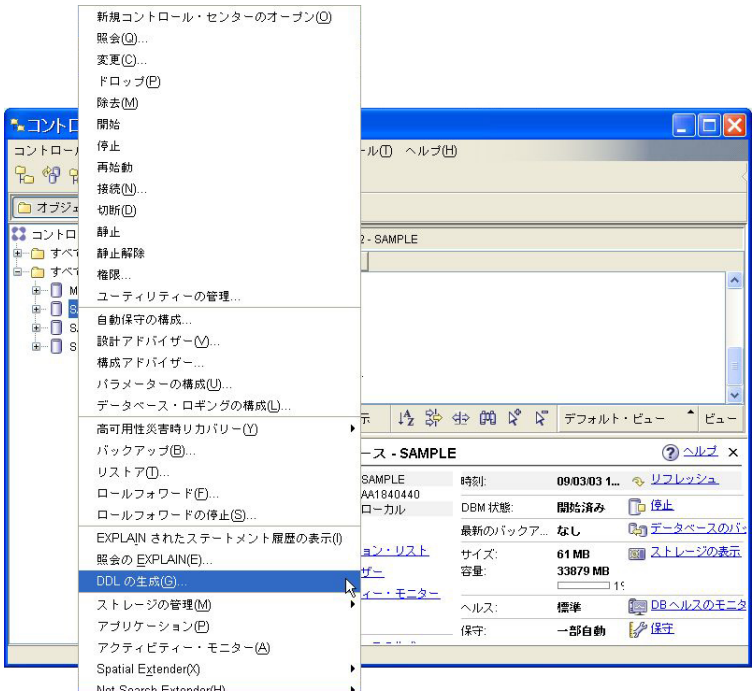


図 9.6 -コントロール・センターから DDL の生成ウィザードの起動

図 9.6 のように、DDL を取得するデータベースを選択して右クリックし、「DDL の生成」を選択します。すると、いくつかの抽出オプションが表示された「DDL の生成」ウィンドウが開きます (図 9.7 を参照)。

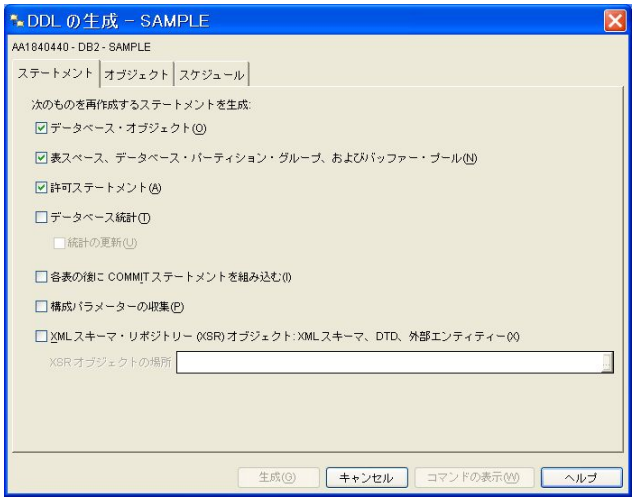


図 9.7 -DDL の生成ウィザード

9.6 まとめ

この章では、DB2 のさまざまなエクスポートおよびインポート機能について説明しました。まずは各種のエクスポート形式 (ASC、DEL、WSF、IXF) を紹介した上で、エクスポート･ユーティリティーについて詳しく説明しました。続いてインポート・ユーティリティーとロード・ユーティリティーを取り上げ、ロード・ユーティリティーを使用するときには SET INTEGRITY ステートメントが必要であることを説明しました。

db2move コマンドは、エクスポートおよびインポートによるデータ移動プロセスを単純化する手段となります。

さらに高度な db2look コマンドを使用すれば、データベースを作成するために必要なすべての要素を抽出して保管し、後で必要に応じてデータベースを複製することが可能になります。

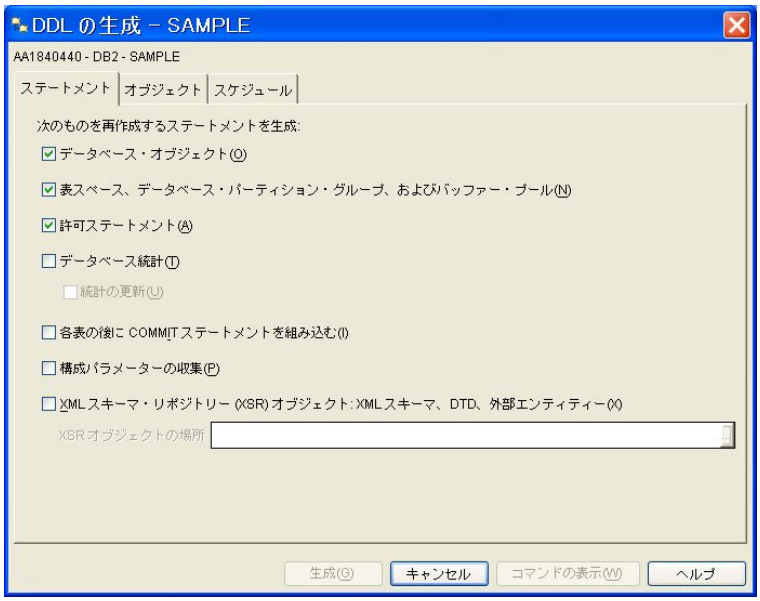
さらに高度な db2look コマンドを使用すれば、データベースに必要なすべての要素を抽出して保管しておくことができるので、後で必要に応じてデータベース全体を複製することが可能になります。

9.7 演習　EXPRESS データベースの DDL を抽出する

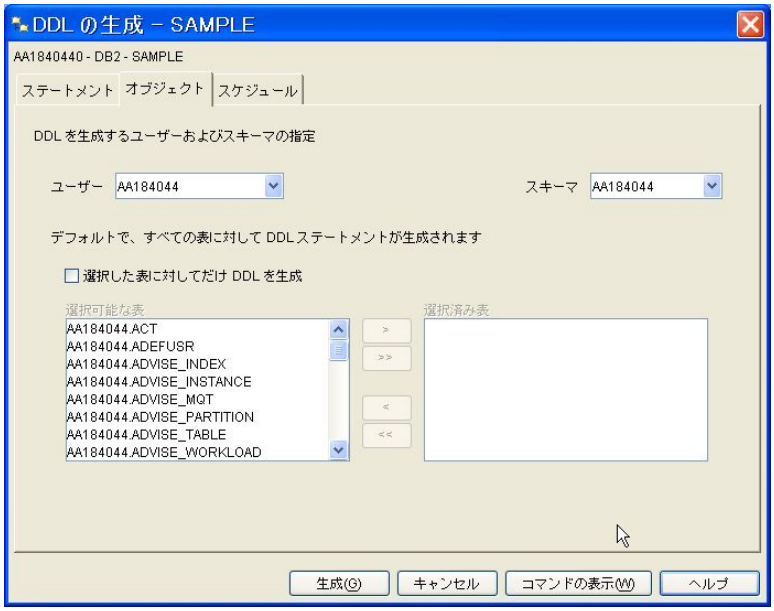
データベースを複製するときには、データベースの再作成プロセスをできるだけ簡単にして、しかも繰り返し実行できるようにしたいはずです。それには通常、SQL スクリプトを使用します。SQL スクリプトは、DB2 をインストールした直後から実行することができます。この演習では、「コントロール・センター」を使用して、(前に行った演習で作成した) ***EXPRESS*** データベースからオブジェクト定義を抽出します。

手順

1. コントロール・センターを開きます。
2. オブジェクト・ツリーで ***EXPRESS*** データベースを右クリックし、「DDL の生成」メニュー項目を選択します。この操作によって、「DDL の生成」ダイアログ・ウィンドウが起動します。
3. 「DDL の生成」ウィンドウで、以下の図に従って、生成する DDL のオプションを指定します。環境内に表スペースやバッファー・プールなどの追加オブジェクトを作成してある場合には、このウィンドウで該当するオブジェクトを選択することになりますが、こういった類いのオブジェクトはまだ作成していないため、ここではチェック・ボックスのチェック・マークを外してください。データベース統計の生成オプションを選択していない理由は、本番環境には開発環境とは異なる統計のセットが含まれる可能性が高いためです。同様に、構成パラメーターも異なる可能性が高いので生成対象として選択していません。お使いの環境で、後にデプロイされる通りにすべてを構成する場合には、そのためのオプションを選択しても構いません。



4. 「オブジェクト」タブに切り替えます。ここでは、DDL を生成するオブジェクトを具体的に選択することができます。この例では、すべてのオブジェクトの作成で使用してきたユーザーとスキーマを選択し、そのスキーマに含まれるすべてのオブジェクトの DDL を生成します。「生成」ボタンをクリックして DDL の生成を開始します。



5. 生成された DDL を確認します。前のステップを実行したことによって、選択したオブジェクトのすべての SQL ステートメントが含まれる単一のスクリプトが生成されているはずです。これからこのスクリプトを、論理グループ別に整理します。

6. ファイル・システムに `C:\express` というディレクトリーを作成し、DDL ファイルをこの新しいディレクトリーに `schema.ddl` という名前で保存します (「保管」ボタンをクリックします)。

7. 新しく保存したファイルをコマンド・エディターで開きます (ヒント:コマンド・エディターで「選択」 -> 「オープン」の順に選択します)。

8. 必要なのは表の DDL だけですが、ファイルを開くとわかるように、このファイルには他のデータベース・オブジェクトの DDL も含まれています。そこで、すべての **CREATE TRIGGER** ステートメントを別の新しいファイル、`triggers.ddl` に移します。ここでは 1 つのトリガーしか作成しませんが、通常はタイプ別にオブジェクトを分けるのがベスト・プラクティスです。

9. とりあえず、以下の種類のステートメントもすべて削除することをお薦めします。

   - CONNECT TO データベース・ステートメント
   - DISCONNECT ステートメント

   この時点で、スクリプトは以下の 2 つになっているはずです。

   `C:\express\schema.ddl` (表、ビュー、索引、制約の DDL が含まれています)

   `C:\express\triggers.ddl` (トリガーの DDL が含まれています)

10. デプロイメントに備えて以下のようにスクリプトを編集します。

    - 不要なコメントを削除します (-- CONNECT TO_ など)。
    - 関数とプロシージャーをそれぞれ別のファイルに分離します (関数とプロシージャーが大量にある場合に役立ちます)。さらに、機能やアプリケーション別に分類することもできます (`billing.ddl`、`math.ddl`、`stringfunc.ddl` など)。

11. トリガー、関数、プロシージャーの終わりを区切るために、特殊文字 (@) が使用されていることに気付いた方もいるのではないでしょうか。これは、**CREATE <object>** ステートメントの終わりには、オブジェクト内に含まれるプロシージャー・ステートメントの終わりとは異なる区切り文字が必要なためです。

10

第 10 章 – データベース・セキュリティー

この章では、DB2 でのセキュリティー対処法を説明します。図 10.1 に基本的な概要を示します。

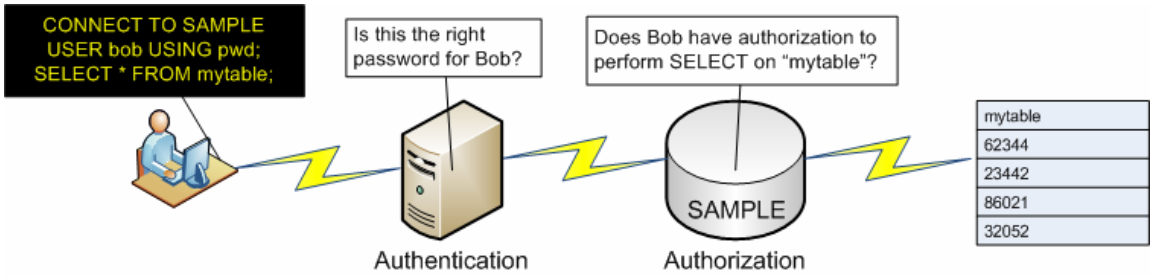


図 10.1 – DB2 セキュリティーの概要

図 10.1 に示されているように、DB2 セキュリティーは 2 段階で構成されます。

- **認証**

  これは、ユーザー ID の有効性を確認するためのプロセスです。認証は DB2 外部のセキュリティー機能によってセキュリティー・プラグインを使用して行われます。デフォルトのセキュリティー・プラグインは、オペレーティング・システムのセキュリティーを利用しますが、Kerberos や LDAP のプラグインを使用することも、カスタムで作成した独自の認証プラグインを使用することもできます。デフォルトの OS ベースの認証プラグインを利用する場合は、ユーザー ID とパスワードが (例えば、接続ステートメントの一部として) データベース・サーバーに渡されます。すると、データベース・サーバーが OS 認証を起動して、そのユーザー ID とパスワードの有効性を確認します。

- **許可**

  この段階では、認証済みのユーザーが要求している操作の実行が、そのユーザーに許可されているかどうかをチェックします。許可情報は、DB2 カタログおよび DBM 構成ファイルに保管されます。

図 10.1 の例では、ユーザー ***bob*** が以下のステートメントで **`SAMPLE`** データベースに接続します。

```
CONNECT TO sample USER bob USING pwd
```

bob と ***pwd*** の両方がオペレーティング・システムあるいは外部認証機能に渡されると、ユーザー名 ***bob*** がすでに定義されていること、入力されたパスワードが該当ユーザーと一致することが確認された上で、認証が承認されることになります。このプロセスが正常に完了した場合、オペレーティ

ング・システムはセキュリティー制御を DB2 に返します。図 10.1 では次に、ユーザー ***bob*** が以下のステートメントを実行します。

```
SELECT * FROM mytable
```

DB2 は許可チェックを実行してセキュリティー制御を引き継ぎ、ユーザー ***bob*** に ***mytable*** 表での SELECT 特権があることを確認します。許可チェックに失敗すると、DB2 はエラー・メッセージを返します。チェックに合格した場合には、このステートメントが ***mytable*** に対して実行されます。

10.1 認証

実際の認証はオペレーティング・システムによってデフォルトのセキュリティー･プラグインを利用して (または別の外部セキュリティー機能を利用して) 実行されますが、認証をどのレベルで行うかを決定するのは DB2 です。

DB2 サーバーで設定するデータベース構成パラメーター AUTHENTICATION に有効な値は複数あります。例えば、このパラメーターを SERVER (デフォルト) に設定すると、オペレーティング・システムや外部セキュリティー機能はサーバー上で認証を実行します。一方、AUTHENTICATION を CLIENT に設定すると、クライアント上でオペレーティング・システムや外部セキュリティー機能による認証が実行されます。図 10.2 を参照してください。

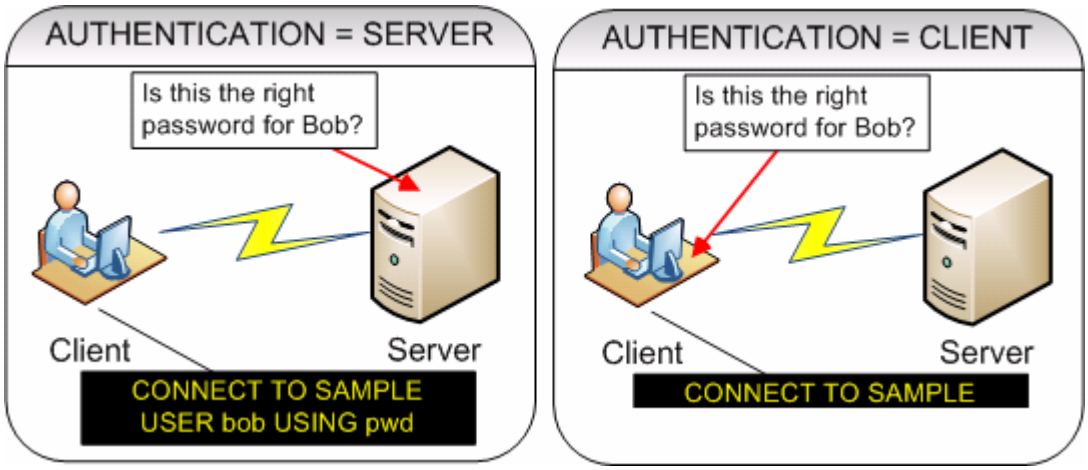


図 10.2 – 認証が行われる場所

AUTHENTICATION パラメーターは、表 10.1 に記載するどの値にでも設定することができます。

コマンド	説明
SERVER (デフォルト)	サーバー上で認証を行う
CLIENT	クライアント上で認証を行う
SERVER_ENCRYPT	ユーザー ID とパスワードを暗号化し、サーバー上で認証を行う
KERBEROS	Kerberos セキュリティー・メカニズムを使用して認証を行う
SQL_AUTHENTICATION_DATAENC	サーバー認証および接続にデータ暗号化を使用する
SQL_AUTHENTICATION_DATAENC_CMP	同上。ただし、データ暗号化は利用可能な場合にのみ使用する
GSSPLUGIN	外部 GSS API ベースのプラグイン・セキュリティー・メカニズムを使用して認証を行う

表 10.1 – AUTHENTICATION パラメーターに有効な値

10.2 許可

許可を構成する特権、権限、ロール、そしてラベル･ベースのアクセス制御 (LBAC) のクレデンシャルは、DB2 システム表に保管され、DB2 によって管理されます。

特権は、データベースに対して 1 つのタイプの操作 (CREATE、UPDATE、DELETE、INSERT など) を実行する許可をユーザーに与えます。

ロールは、複数の異なる特権をまとめてユーザー、グループ、他のロールなどに付与するために使用することができます。

権限は、事前定義されたロールのことで、このロールは複数の特権で構成されます。

ラベル・ベースのアクセス制御 (LBAC) のクレデンシャルに含まれるポリシーおよびラベルによって、特定のユーザーによるアクセスを、行および列レベルできめ細かく制御することができます。LBAC は DB2 Express-C には用意されていませんが、その詳細は第 2 章に記載されています。

ラベル・ベースのアクセス制御 (LBAC) のクレデンシャルには、ポリシーとラベルが含まれており、このポリシーとラベルによって、特定のユーザーによるアクセスを行および列レベルできめ細かく制御することができます。LBAC は DB2 Express-C ではサポートされませんが、その詳細は第 2 章に記載されています。

10.2.1 特権

図 10.3 に、DB2 での特権の一部を示します。

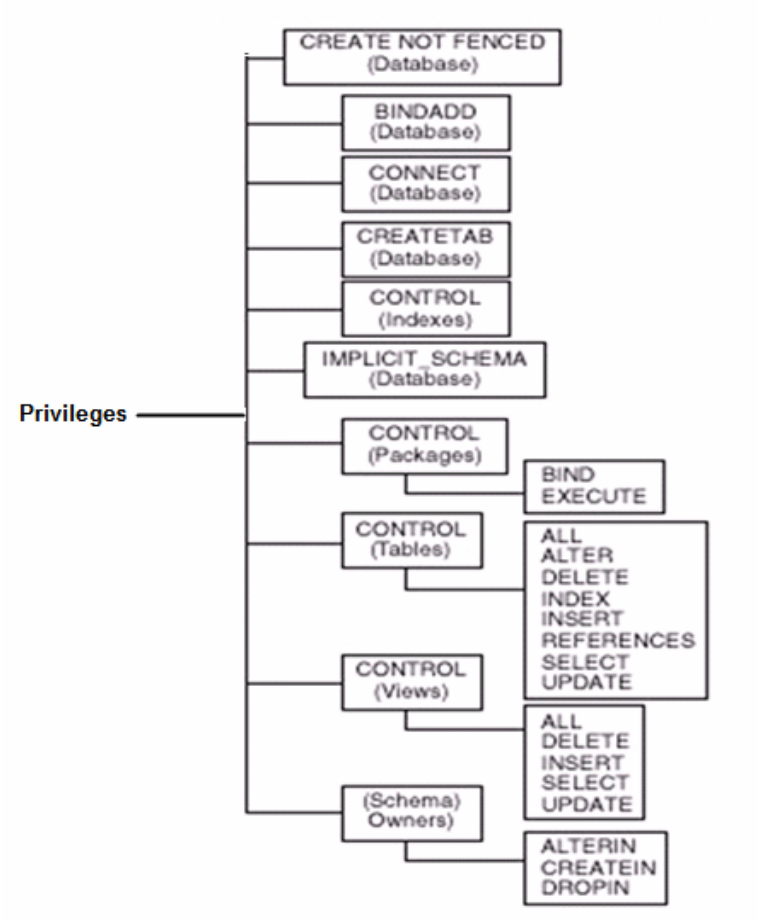


図 10.3 – DB2 の特権の一部

ユーザーまたはグループが CONTROL 特権を受け取るということは、そのユーザーまたはグループは、他のユーザーやグループに特権を付与することができるということを意味します。その他の特権についての詳細は、DB2 インフォメーション・センターを参照してください。

10.2.2 権限

権限は以下の 2 つのグループに分類されます。

- インスタンス･レベルの権限: インスタンス・レベルで有効な権限です (例えば、SYSADM)。
- データベース･レベルの権限: データベース・レベルでのみ有効な権限です (例えば、DBADM)。

10.2.2.1 インスタンス・レベルの権限

表 10.2 にインスタンス･レベルの権限を記載します。

権限	説明
SYSADM	インスタンス全体を管理する
SYSCTRL	データベース・マネージャー・インスタンスを管理する
SYSMAINT	インスタンス内のデータベースを保守する
SYSMON	インスタンスとそのデータベースを監視する

表 10.2 - インスタンス･レベルの権限

あるグループに SYSADM、SYSCTRL、SYSMAINT、あるいは SYSMON の権限を付与するためには、オペレーティング･システム・グループに DBM CFG パラメーターである SYSADM_GROUP、SYSCTRL_GROUP、SYSMAINT_GROUP、あるいは SYSMON_GROUP が付与する権限に応じて割り当てられます。

例えば、オペレーティング･システム・グループ ***myadmns*** に SYSADM 権限を付与するには、以下のコマンドを実行します。

```
update dbm cfg using SYSADM_GROUP myadmns
```

new in V9.7

それぞれの DB2 インスタンスには、インスタンスごとに固有の権限グループ定義があります。Windows では、上記のパラメーターはデフォルトで空白になっています。つまり、ローカル管理者グループが SYSADM 権限を持つことになります。DB2 9.7 では、DB2ADMNS グループ (拡張セキュリティーが有効な場合) とローカル・システム・アカウントも SYSADM 権限を持ちます。ローカル・システム・アカウントの許可 ID は SYSTEM です。Linux ではインスタンス所有者グループが、デフォルトの SYSADM グループとなります。

図 10.4 は、DB2 インフォメーション・センターから引用した図です。この図には、インスタンス・レベルごとの権限と、各権限で実行可能な機能が示されています。この図を見るとわかるように、SYSADM 権限には SYSCTRL 権限で実行可能なすべての機能と追加の機能が含まれ、SYSCTRL 権限には SYSMAINT 権限で実行可能なすべての機能と追加の機能が含まれるといった具合です。

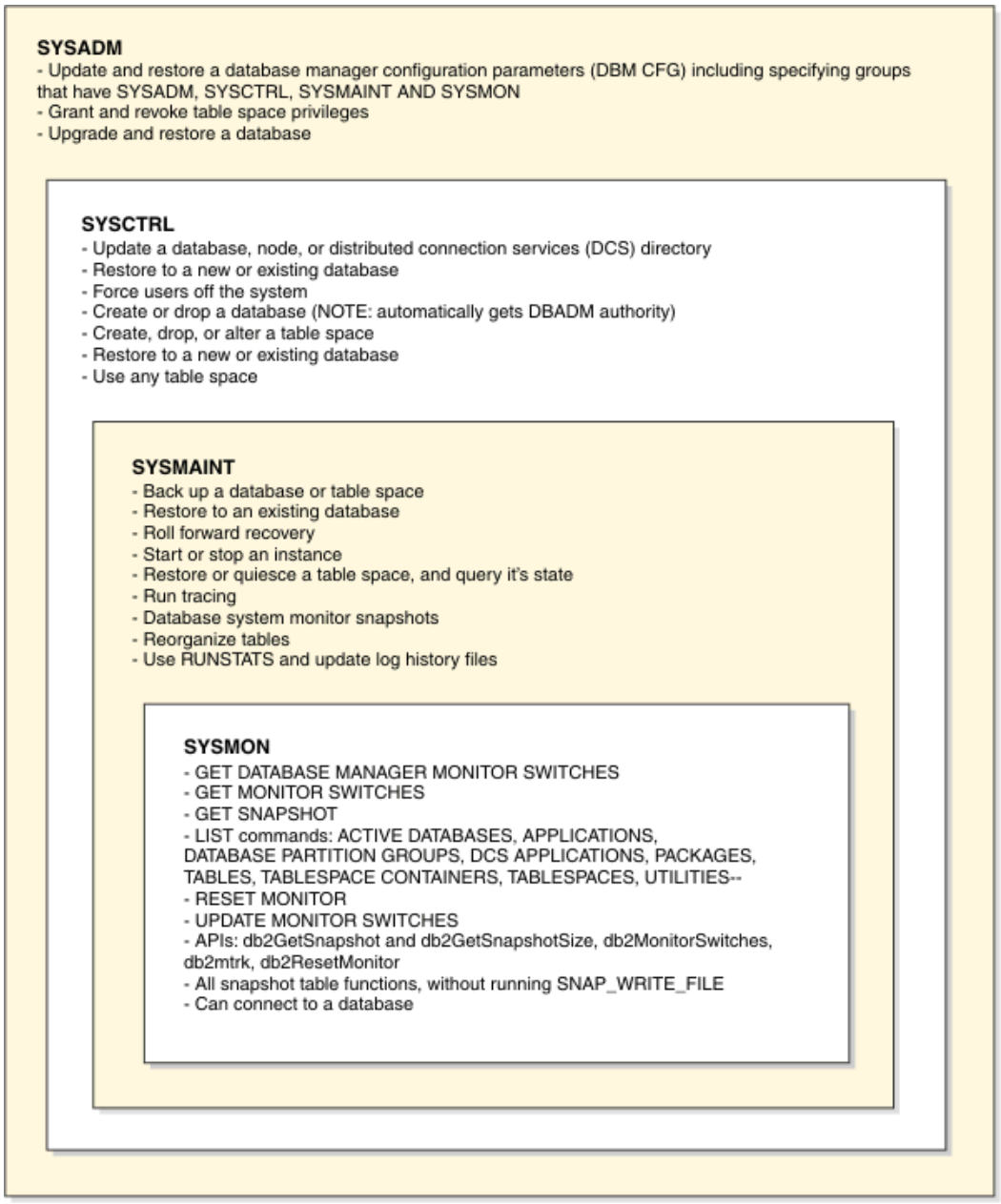


図 10.4 – インスタンス・レベルの権限とそれぞれの権限が持つ機能

10.2.2.2 データベース・レベルの権限

表 10.3 にデータベース・レベルの権限を記載します。

権限	説明
SECADM	データベース内のセキュリティーを管理する
DBADM	データベースを管理する
ACCESSCTRL (V9.7 より追加)	権限と特権を付与および解除する (SECADM、DBADM、ACCESSCTRL、および DATAACCESS 権限は除く。これらの権限は、SECADM 権限によって付与および解除する必要があることに注意)
DATAACCESS (V9.7 より追加)	データベース内のデータにアクセスできるようにする
SQLADM	SQL クエリーをモニターし、調整する
WLMADM	ワークロードを管理する
EXPLAIN	照会計画の説明責任があるユーザー (EXPLAIN 権限では、データ自体にアクセスすることはできない)

表 10.3 - データベース・レベルの権限

データベース・レベルの権限を付与するには、GRANT ステートメントを使用します。例えば、**SAMPLE** データベースに対する DBADM をユーザー ***bob*** に付与するには、以下のようにします。

```
connect to sample
grant DBADM on database to user bob
```

上記の例では、まずデータベース (この例では **sample** データベース) に接続してからでないと、ユーザーに DBADM を付与することはできません。DBADM 権限とその他のデータベース・レベルの権限を付与するには、SECADM 権限を持っている必要があります。

DBADM は表スペースを作成できないことに注意してください。表スペースはデータベース内のオブジェクトですが、表スペースが扱うコンテナー (ディスク) とバッファー・プール (メモリー) はシステムの物理リソースだからです。表スペースを作成できるのは SYSADM です。

DB2 インフォメーション・センターから抜粋した図 10.5 には、さまざまなデータベース・レベルの権限と、それぞれの権限を持っていると実行可能な機能を示してあります。

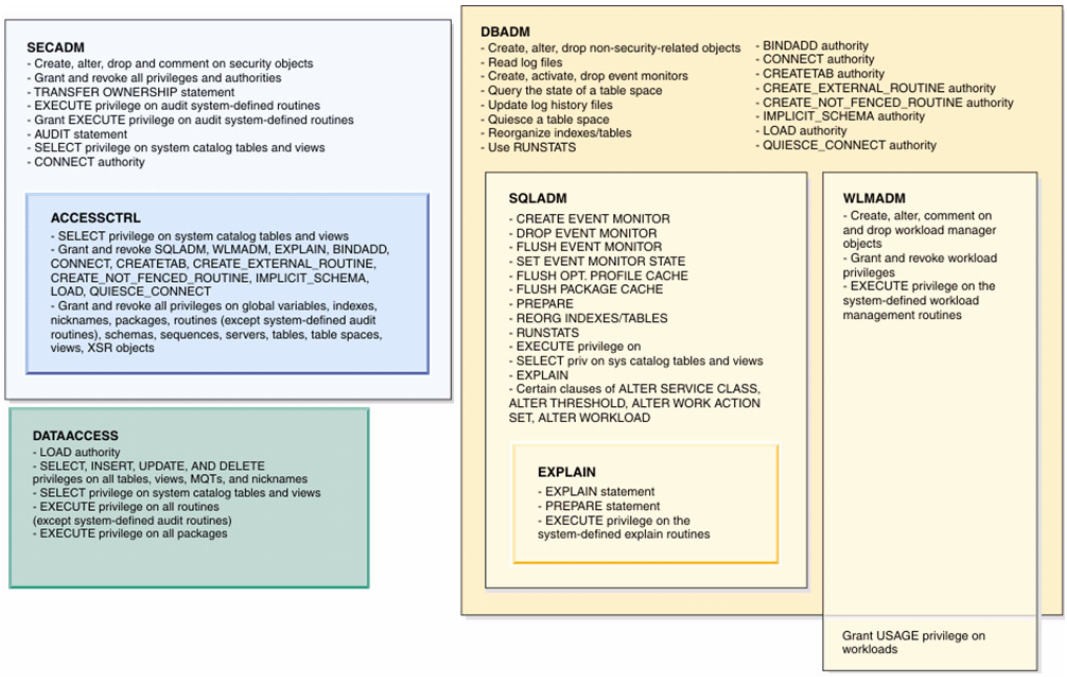


図 10.5 - データベース・レベルの権限とそれぞれの権限が持つ機能

new in V9.7

注:

DB2 9.7 ではデータの機密性およびガバナンス準拠を強化するため、許可モデルが新しくなっています。新しい許可モデルでは、システム管理者、データベース管理者、セキュリティー管理者の責任が明確に区別されます。

一般に、以前の DB2 バージョンと比べると、さまざまな権限でその機能の範囲が縮小されています。例えば SYSADM は以前、あらゆるデータベースのデータにアクセスできましたが、そのようなアクセス権はもうありません。また、DBADM も自身が管理するデータベースのデータにアクセスできなくなっています。その一方、SECADM が持つ機能は増え、例えばユーザー、ロール、グループに対して権限、特権を付与および解除できるようになりました。

また、新しい権限も作成され、システム・セキュリティーをさらに細分化して制御できるようになっています。ユーザーにはそのジョブを行うために必要とする以上の権限が与えないことで、データ露出の危険性も最小限になります。

10.2.2.3 DB2 9.7 より前のバージョンと同じように SYSADM と DBADM を機能させる方法

SYSADM を DB2 9.7 より前の DB2 バージョンと同じように機能させるには、以下の 2 つのケースが考えられます。

- SYSADM がデータベースの作成者であれば、SYSADM にはそのデータベースに対する DATAACCESS、ACCESSCTRL、SECADM、DBADM 権限が自動的に与えられます。したがって、SYSADM は DB2 9.7 より前のバージョンと同じ機能を実行することができます。

- SYSADM がデータベースの作成者でない場合、DB2 の前のバージョンと同じ機能 (SECADM を除く) を実行できるようにするためには、SECADM が SYSADM に、その特定のデータベースに対する DBADM 権限を DATAACCESS および ACCESSCTRL (これはデフォルトです) と併せて付与する必要があります。

SECADM については、以下のケースが考えられます。

- デフォルトの SECADM は、データベースの作成者です。
- SECADM 権限を持つユーザーが、SYSADM 権限を持つユーザーに SECADM 権限を付与すると、その SECADM 権限を与えられたユーザーは他のユーザーに SECADM 権限を付与できるようになります。
- SECADM 権限を持つユーザーがあるユーザーに DBADM 権限を付与すると、デフォルトにより、DBADM 権限を与えられたユーザーには DATAACCESS および ACCESSCTRL 権限も与えられます。

DB2 9.5 データベースからマイグレーションしている場合には、SYSADM と DBADM の機能は変更されません。DB2 はマイグレーションの時点で SYSADM グループに対し、自動的に DBADM、DATAACCESS、および ACCESSCTRL を付与するためです。DB2 はマイグレーションの時点で DBADM 権限を持つすべての許可 ID に対しても、自動的に DATAACCESS と ACCESSCTRL を付与します。さらに DB2 は、データベースの SECADM 権限を保持する USER タイプの許可 ID がない場合、マイグレーションを行っているユーザー ID に対して自動的に SECADM を付与します。これにより、SYSADM は DBADM および SECADM を付与または解除する暗黙の権限を失い、これらの権限の付与、解除は SECADM だけが実行できるようになります。

10.2.3 ロール

セキュリティー管理者はロールを使用して、複数のユーザーまたはグループにまとめて特権または権限を割り当てることができます。ロールはグループとよく似ていますが、ロールが定義されるのは DB2 内部です。このことから、ロールにはいくつかの利点があります。例えば、ロールに付与された特権および権限は、ユーザーがビューやトリガーなどのオブジェクトを作成するときに常に使用されます。これは、グループの場合には当てはまりません。その一方、ロールに SYSADM などのインスタンス・レベルの権限を割り当てることはできません。ロールに割り当てることができるのは特権およびデータベース・レベルの権限だけですが、グループにはすべての特権と権限を割り当てることができます。

ロールを操作するには、以下のいくつのステップが必要です。

1. まず、セキュリティー管理者 (SECADM) が以下のようなコマンドを使用してロールを作成します。

```
CREATE ROLE TESTER
```

2. 次に、DBADM がロールに特権または権限を付与します。例えば、**SAMPLE** データベースの **STAFF** および **DEPT** 表での SELECT 特権を ***TESTER*** ロールに付与するには、以下のコマンドを実行します。

```
GRANT SELECT ON TABLE STAFF TO ROLE TESTER
GRANT SELECT ON TABLE DEPT TO ROLE TESTER
```

3. セキュリティー管理者が ***TESTER*** ロールをユーザー RAUL とユーザー JIN に付与します。

```
GRANT ROLE TESTER TO USER RAUL, USER JIN
```

4. JIN が TEST 部門を離れることになった場合には、セキュリティー管理者がユーザー JIN の ***TESTER*** ロールを解除します。

```
REVOKE ROLE TESTER FROM USER JIN
```

10.3 グループ特権に関する考慮事項

ロールの代わりにグループを使用することにした場合は、以下の点を考慮してください。

- グループに付与された特権は、グループ・メンバーシップによって暗黙的に継承されて、そのグループのメンバーに付与されます。
- グループから削除されたユーザーは暗黙的なグループ特権を失いますが、ユーザーに対して明示的に付与された特権はそのまま残ります。明示的にユーザーに付与した特権を解除するには、明示的にユーザーからその特権を解除する必要があります。

10.4 The PUBLIC グループ

DB2 は、PUBLIC という名前の内部グループを定義しています。オペレーティング・システムまたはネットワーク認証サービスによって識別されるユーザーはすべて、暗黙的に PUBLIC グループのメンバーとなります。データベースを作成すると、以下の特権が自動的に PUBLIC に付与されます。

- CONNECT
- CREATETAB
- IMPLICIT SCHEMA
- BINDADD

セキュリティーを強化するために、以下のように PUBLIC グループのすべての特権を取り消すことをお薦めします。

```
REVOKE CONNECT ON DATABASE FROM PUBLIC
REVOKE CREATETAB ON DATABASE FROM PUBLIC
REVOKE IMPLICIT_SCHEMA ON DATABASE FROM PUBLIC
REVOKE BINDADD ON DATABASE FROM PUBLIC
```

10.5 GRANT および REVOKE ステートメント

GRANT および REVOKE ステートメントは SQL 標準の一部であり、ユーザー、グループ、またはロールの特権を付与または解除するために使用されます。このコマンドを実行するユーザーは少なくとも ACCESSCTRL を持っている必要があります。以下に、この 2 つのステートメントの例を記載します。

表 T1 の SELECT 特権を、ユーザー USER1 に付与する場合

```
GRANT SELECT ON TABLE T1 TO USER user1
```

表 T1 のすべての特権を、グループ GROUP1 に付与する場合

```
GRANT ALL ON TABLE T1 TO GROUP group1
```

表 T1 のすべての特権を、グループ GROUP1 から取り消す場合

```
REVOKE ALL ON TABLE T1 FROM GROUP group1
```

プロシージャー p1 の EXECUTE 特権を、ユーザー USER1 に付与する場合

```
GRANT EXECUTE ON PROCEDURE p1 TO USER user1
```

プロシージャー p1 の EXECUTE 特権を、ユーザー USER1 から取り消す場合

```
REVOKE EXECUTE ON PROCEDURE p1 FROM USER user1
```

10.6 許可および特権のチェック

許可と特権をチェックするには、コントロール・センターを使用するのが最も簡単です。図 10.6 に、コントロール・センターから ***EMPLOYEE*** 表の「表特権」ダイアログを起動する方法を示します。

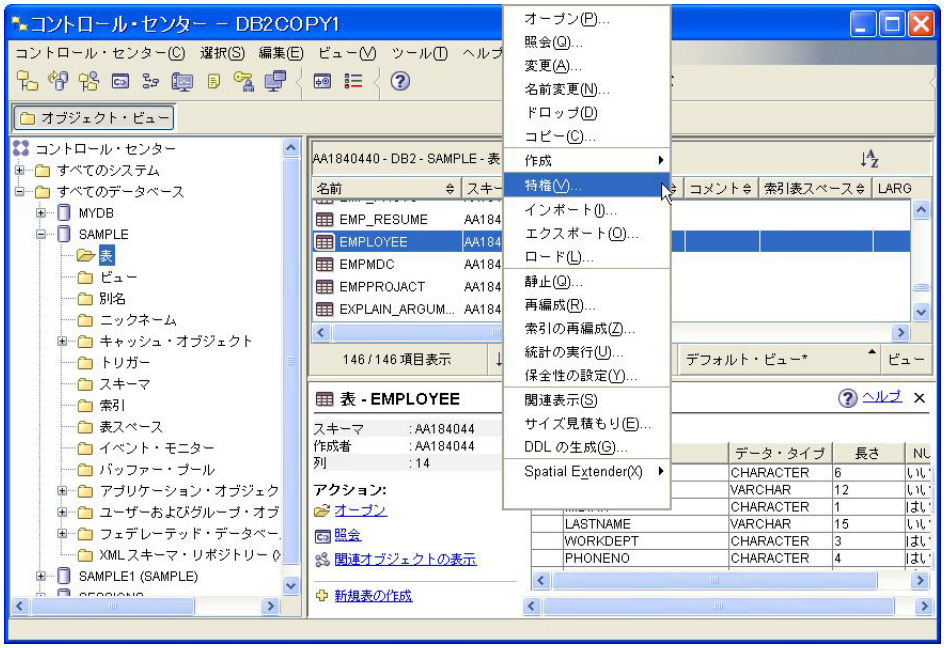


図 10.6 – 表特権ダイアログの起動

図 10.6 に示されているように、対象の表を選択して右クリックし、「特権」を選択します。このメニュー項目を選択すると、「表特権」ダイアログ・ボックスが表示されます。図 10.7 に、このダイアログ・ボックスを各フィールドと要素の説明と併せて示します。

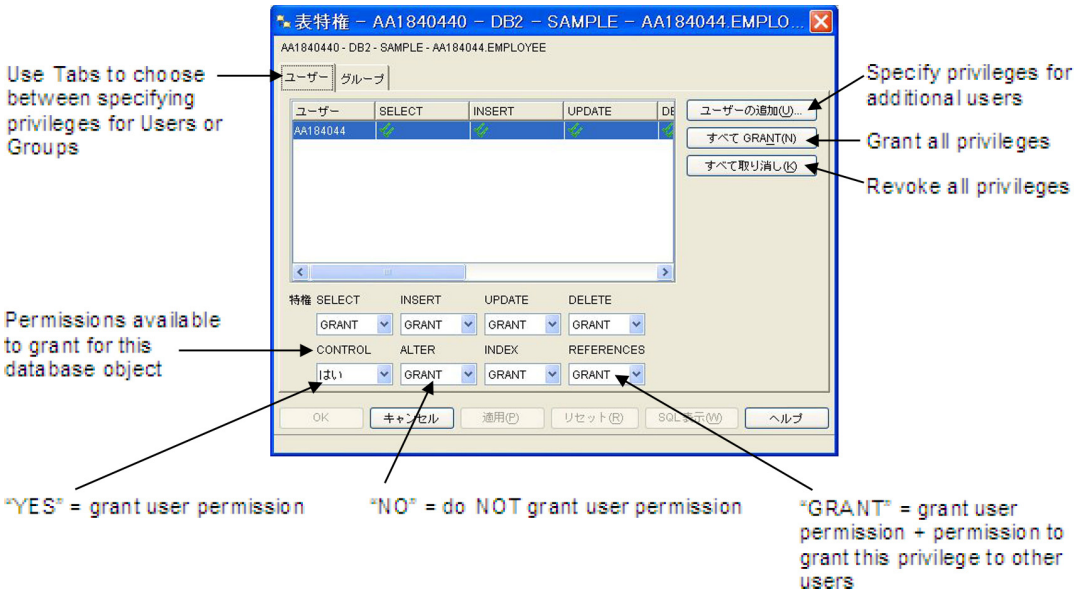


図 10.7 – 「表特権」ダイアログ・ボックス

許可と特権をチェックするには、許可情報が含まれる DB2 SYSCAT カタログ・ビューを照会するという方法もあります。例えば、ユーザー ***DB2ADMIN*** が表 T2 の SELECT 特権を持っているかどうか、そしてこの特権を持っているとしたら誰によって付与されたのかを調べるには、以下のクエリーを実行します。

```
SELECT grantor, grantee, selectauth
  FROM syscat.tabauth
 WHERE tabname = 'T2'

GRANTOR             GRANTEE               SELECTAUTH
--------------------------------------------------
ARFCHONG            DB2ADMIN                   Y
```

上記の例では、ユーザー ***ARFCHONG*** が、ユーザー ***DB2ADMIN*** に SELECT 特権を付与したことが示されています。

10.7 Windowsでの拡張セキュリティー

Windows オペレーティング・システムから DB2 のファイルやディレクトリー (例えば DB2 がインスタンス情報を保管するファイル、ディレクトリーなど) へのアクセスを防ぐため、DB2 ではインストール時にデフォルトで拡張セキュリティーが有効に設定されます。拡張セキュリティーでは以下の 2 つのグループが作成されます。

- DB2ADMNS: このグループおよびローカル管理者は、オペレーティング・システムからすべての DB2 オブジェクトにアクセスし、あらゆる操作を実行することができます。
- DB2USERS: このグループは、すべての DB2 オブジェクトに対し、オペレーティング・システムからの読み取りおよび実行アクセス権を持ちます。

new in V9.7

DB2 9.7 では、拡張セキュリティーが使用可能になっていて、データベース構成パラメーター SYSADM_GROUP が設定されていない場合、DB2ADMNS グループのメンバーには自動的に DB2 での SYSADM 権限が与えられます。

10.8 まとめ

DB2 のセキュリティーについて取り上げたこの章では、最初に認証と許可との違い、そしてそれぞれの重要性について包括的に説明し、それからインスタンスとデータベースにセキュリティーを提供する具体的な許可レベルを見てきました。

次に、ロールという新しい概念を紹介し、ロールをどのように使用すればセキュリティー面での利点が得られるか、またグループにセキュリティーを設定する場合の制約について説明しました。具体的には PUBLIC グループに焦点を当て、一般ユーザーをデータ・サーバーからブロックするために、PUBLIC グループに対するセキュリティーを強化する方法を提案しました。

さらに、GRANT および REVOKE ステートメントについても詳しく述べ、最後に「コントロール・センター」とシステム・カタログ表を使用して許可レベルと特権レベルを確認する方法を説明しました。

10.9 演習　ユーザー許可の付与、解除を行う

これまでは、インスタンス管理者アカウント (SYSADM) を使用して、すべてのデータベース・コマンドを実行してきました。このアカウントは、すべてのユーティリティー、データ、データベース・オブジェクトに対して幅広いアクセスをすることができます。そのため、偶発的あるいは故意のデータ損失を防ぐためには、このアカウントを保護することが非常に重要になります。そのためには、ほとんどの場合、異なるユーザー・アカウントまたはグループを作成して、それぞれに限られた数の許可を与えることになります。この演習では、新しいユーザー・アカウントを作成し、特定の特権を割り当てます。

パート 1 – 特権を扱う

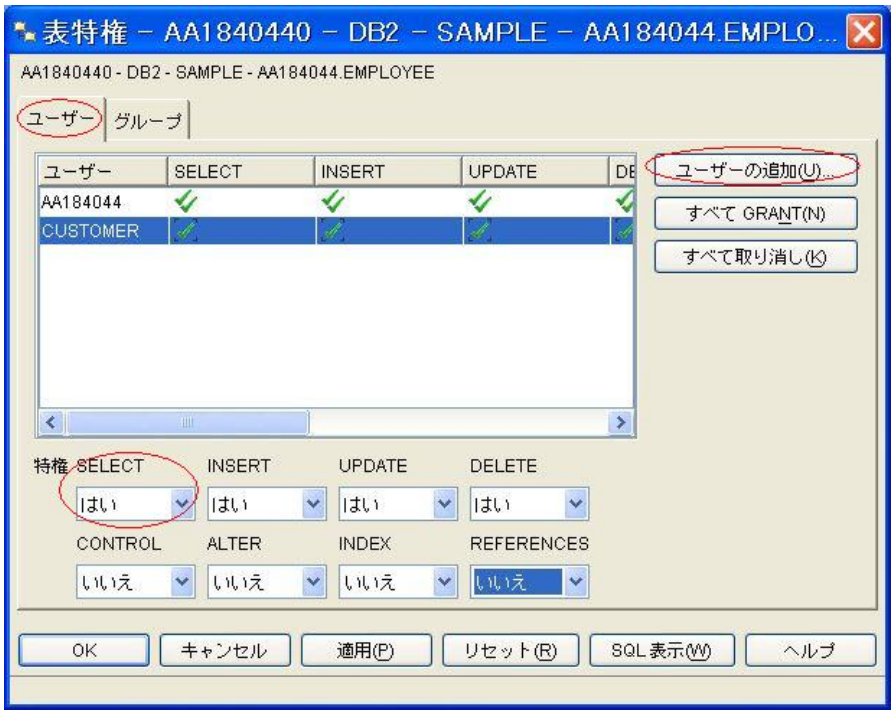
演習のこのパートでは、コントロール･センターを使用してユーザーに特権を付与したり、解除したりする方法を実践します。

手順

1. Windows デスクトップで「マイ コンピューター」アイコンを右クリックし、「管理」メニュー項目を選択して「コンピューターの管理」コンソールを開きます。
2. ウィンドウの左ペインに表示されたツリーで「システム ツール」セクションを展開し、「ローカル ユーザーとグループ」フォルダーを展開します。「ユーザー」フォルダーを右クリックして、「新しいユーザー」項目を選択します。
3. 「新しいユーザー」ダイアログ・ウィンドウで情報を入力します。「ユーザー名」フィールドには ***customer***、「フル ネーム」フィールドには ***Customer1***、「説明」フィールドには ***A typical bookstore customer*** と入力します。「パスワード」と「パスワードの確認入力」フィールドには、***ibmdb2ibm*** と入力してください。「ユーザーは次回ログオン時にパスワードの変更が必要」オプションのチェック・マークを外してから、「作成」ボタンをクリックして新規ユーザーを作成し、「閉じる」ボタンをクリックしてダイアログ・ウィンドウを閉じます。

4. コントロール・センターを開き、拡張ビューを選択します。拡張ビューに切り替えるには、「ツール」メニューから「コントロール・センターのカスタマイズ」を選択します。「詳細」オプションを選択して「OK」ボタンをクリックします。
5. コントロール・センターの左側のオブジェクト・ツリー・ペインで、オブジェクト・ツリーを「すべてのデータベース」 -> 「EXPRESS」 -> 「表」まで展開します。
6. 新しく作成したユーザーに、必要な特権を付与します。***EXPRESS*** データベースの表のリストで、***CUSTOMERS*** 表を右クリックし、「特権」項目を選択して「表特権」ダイアログ・ウィンドウを表示します。
7. 「ユーザーの追加」ボタンをクリックし、上記のステップで作成した「***CUSTOMER***」を選択します。「OK」ボタンをクリックして「ユーザーの追加」ダイアログ・ボックスを閉じます。

8. ユーザー ***CUSTOMER*** はユーザー・リストに追加されていますが、特権は 1 つも割り当てられていません。このユーザーに SELECT、INSERT、UPDATE、および DELETE 特権を付与するため、該当するそれぞれのドロップダウン・ボックスを「はい」に変更します。インターネット書店の顧客には、顧客自身のアカウント・データを表示、追加、更新、削除できるようにする必要がありますが、それ以外の許可はこのユーザーには必要ないので与えません。「OK」ボタンをクリックして「表特権」ダイアログ・ウィンドウを閉じ、変更内容を確定してください。



9. ***BOOKS*** 表と ***SALES*** 表に対して、ステップ 6 から 8 を繰り返します。***BOOKS*** 表については、SELECT 特権のみを付与してください。顧客には、インターネット書店の在庫データを変更させないようにする必要があるためです。***SALES*** 表の場合は SELECT および INSERT 特権を付与します。販売トランザクションを変更するためのアクセス権はインターネット書店の従業員だけが持たなければならないため、顧客には DELETE 特権も UPDATE 特権も与えてはなりません。

10. 上記で作成したユーザー ID、***customer*** を使用して、DB2 コマンド･ウィンドウで以下の内容を実行し、データベースに接続します。

```
db2 connect to express user customer using ibmdb2ibm
```

customers 表に SELECT を実行してデータを選択してみてください。データを選択できましたか？ 今度は ***SALES*** 表のデータを削除、または更新しようとして DELETE と UPDATE を実行すると、どうなるかを確認してください。

パート 2 – SYSADM、DBADM、SECADM 権限を扱う

演習のこのパートでは、SYSADM と DBADM 権限を割り当てる方法を実践し、これらの権限がどのように機能するかを学びます。

手順

1. パート 1 と同じステップに従って、***mysysadm*** という新規ユーザーを作成します。
2. Windows グループ ***mysysadmgrp*** を作成します。ユーザーを作成する場合と同じステップに従いますが、今回は「ユーザー」フォルダーを右クリックする代わりに、「グループ」フォルダーを右クリックして「新しいグループ」を選択します。「グループ名」フィールドには ***mysysadmgrp*** と入力します。「所属するメンバ」セクションに新規メンバーを追加するために「追加」をクリックし、***mysysadm*** と入力します。「名前の確認」ボタンをクリックし、メンバーを正しく入力したことを確認します。正しく入力されていたら「作成」をクリックし、続いて「閉じる」をクリックします。
3. ここまでのところで、ユーザー ***mysysadm*** と、この ***mysysadm*** が属する ***mysysadmgrp*** というグループを作成しました。この作業はすべて Windows オペレーティング・システムで行ったので、今度は DB2 に対し、***mysysadmgrp*** グループを SYSADM グループに設定するように指示するため、DB2 コマンド・ウィンドウから以下のコマンドを実行します。

   ```
   db2 update dbm cfg using SYSADM_GROUP mysysadmgrp
   ```

   SYSADM_GROUP は動的に有効になるパラメーターではないため、インスタンスを停止してから開始する必要があります。**db2stop** で force オプションを使用することで、**db2stop** の実行前にすべての接続が確実に切断されます。

   ```
   db2stop force
   db2start
   ```

4. DB2 コマンド・ウィンドウからユーザー ***mysysadm*** として SAMPLE データベースに接続し、STAFF 表で SELECT * ステートメントを実行します。注意する点として、必ず、この表を作成したときに使用したスキーマを使用してください。以下の例では、***arfchong*** をスキーマとして使用します。

   ```
   db2 connect to sample user mysysadm using ibmdb2ibm
   db2 select * from arfchong.staff
   ```

   この結果、以下のようなエラー・メッセージを受け取るはずです。SYSADM として実行しているはずなのに、なぜエラーが発生するのでしょうか。

   ```
   SQL0551N  "MYSYSADM" does not have the required authorization or
   privilege to perform operation "SELECT" on object "ARFCHONG.STAFF".
   SQLSTATE=42501
   ```

   DB2 9.7 から、SYSADM にはデフォルトで DBADM 権限が与えられません。そのため、このエラーを受け取ったわけです。

5. **SAMPLE** データベースを作成したときに使用した Windows ユーザーと同じユーザーとして、データベースに接続します。この例でこれに該当するユーザーは ***ARFCHONG*** です。次に、ユーザー ***mysysadm*** に DATAACCESS が含まれない DBADM 権限を付与し、***mysysadm*** としてもう一度、STAFF 表で SELECT を実行してみます。今度は上手くいくでしょうか。

```
db2 connect to sample user arfchong using ibmdb2ibm
db2 grant dbadm without dataaccess on database to user mysysadm
db2 connect to sample user mysysadm using ibmdb2ibm
db2 select * from arfchong.staff
```

この場合も同じ結果となり、**mysysadm** に DBADM 権限が付与された後でも同じエラー・メッセージを受け取ります。しかしこれは、WITHOUT DATAACCESS 節を組み込んだことから当然の結果です。WITHOUT DATAACCESS 節は DATAACCESS 権限が含まれていないことを意味するため、ユーザー **mysysadm** には相変わらずデータにアクセスする権限がありません。この例は、DBADM に対してデータへのアクセスを制限する方法を示しています。

6. 今度は ***mysysadm*** に DATAACCESS を付与して、もう一度 SELECT を実行してみます。

```
db2 connect to sample user arfchong using ibmdb2ibm
db2 grant DATAACCESS on database to user mysysadm
db2 connect to sample user mysysadm using ibmdb2ibm
db2 select * from arfchong.staff
```

今度は SELECT が正常に実行されるはずです。この演習では、DB2 9.7 から新しくなった SYSADM と DBADM の振る舞いを説明しました。この演習で説明しようとしている主な概念は、DB2 9.7 ではデータ・アクセスは、SYSADM および DBADM が実行できる操作とは区別されていることです。

7. SYSADM_GROUP の値を NULL にリセットして、ローカル管理者グループとローカル・システム・アカウントを SYSADM に戻します。

```
db2 update dbm cfg using sysadm_group NULL
db2stop force
db2start
```

11

第 11 章 – バックアップおよびリカバリー

この章では、DB2 データベースのロギング、BACKUP ユーティリティーを使用してデータベース全体またはその一部のコピーを作成する方法、そして RESTORE ユーティリティーを使用してデータをリカバリーする方法を説明します。

11.1 データベースのロギング

テキスト・エディターで作業しているときに文書を確実に保存しようと思ったら、毎回「保存」ボタンをクリックします。データベースの世界でこの操作に相当するのは、COMMIT ステートメントです。COMMIT ステートメントを実行するたびに、データに行った変更内容がすべて確実に、何らかの場所に保存されることになります。

同様に、テキスト文書を操作していると画面の右下隅に「自動保存中」という簡潔なメッセージが表示されることがありますが、データベースの世界でも同じことが起こります。データに対して実行する UPDATE、INSERT、DELETE などの操作は、その操作の実行中に一定の場所に保存されるからです。

この「一定の場所」とは、データベース・ログを指します。ディスクに保管されているデータベース・ログは、トランザクションのアクションを記録するために使用されます。システムやデータベースがクラッシュした場合、リカバリー中にはこれらのデータベース・ログを使用してコミット済みのトランザクションを再生し、再実行することになります。

図 11.1 に、ロギングに関するデータベース操作中の動作概要を図示します。

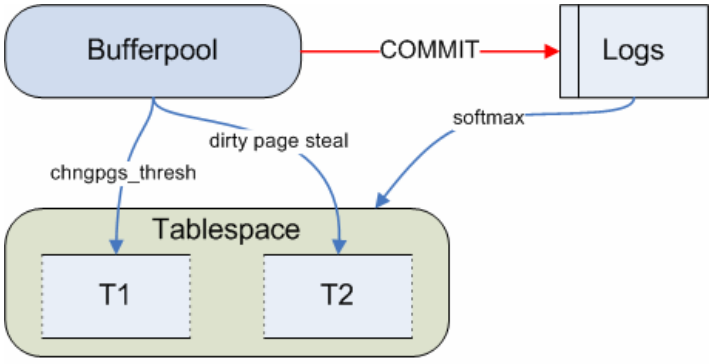


図 11.1 – データベース・ロギング

図 11.1 には、表スペースとログが示されています。どちらもディスク上に置かれていますが、この 2 つを同じディスク上に保持することはお薦めできません。例えば UPDATE 操作が実行されると、操作対象となる行のページがバッファー・プール (メモリー) に取り込まれます。バッファー・プール上では更新による変更が行われ、場合によってはすぐに、また場合によってはログ・バッファーがいっぱいになると、古い値と新しい値がログ・ファイルに保管されることになります。UPDATE に続いて COMMIT が実行された場合は、即座に古い値と新しい値がログ・ファイルに保管されます。このプロセスは、データベースで実行される他の多くの SQL 操作でも繰り返されます。CHNGPGS_THRES パラメーターに指定されたページ変更しきい値に達するなど、特定の条件が満たされない限り、バッファー・プール内のページは「外部化」されることも、表スペース・ディスクに書き込まれることもありません。CHNGPGS_THRESH パラメーターが指定するのは、バッファー・プール内に含まれる「ダーティー」ページ (変更が含まれるページ) のパーセンテージです。

パフォーマンスの観点からすると、COMMIT 操作ごとに 2 回書き込みを実行することは意味をなしません。2 回の書き込みとは、ログへの書き込みと、表スペース・ディスクへの書き込みです。そのため、表スペース・ディスクへのデータの「外部化」は、CHNGPGS_THRES しきい値のようなパラメーターに達した場合にのみ行わるようになっています。

11.2 ログのタイプ

ログには以下の 2 つのタイプがあります。

- **1 次ログ**

  1 次ログは事前に割り当てられたログです。使用可能な 1 次ログ・ファイルの数は、LOGPRIMARY データベース構成パラメーターで設定します。

- **2 次ログ**

  2 次ログは、DB2 の必要に応じて動的に割り当てられます。2 次ログ・ファイルの最大数は、LOGSECOND データベース構成パラメーターで設定します。ログを動的に割り当てるにはコストがかかるため、日常の操作には、1 次ログの割り当てでとどめるようにしてください。2 次ログ・ファイルは、データベースへのすべての接続が終了されると削除されます。

LOGSECOND の値を -1 に設定することで無制限にロギングすることもできます。しかしこうすると、ファイル・システムのスペースが不足する可能性があるので推奨されません。

11.3 ロギングのタイプ

ロギングには、循環ロギング (デフォルト) とアーカイブ・ロギングの 2 つのタイプがあります。

11.3.1 循環ロギング

循環ロギングはデフォルトのロギングです。データベース構成パラメーターの LOGARCHMETH1 および LOGARCHMETH2 は両方とも OFF に設定されると、循環ロギングが有効になります。この 2 つの構成パラメーターはログの保存方法を指定するものですが、ログをアーカイブに保存しない場合は、値を OFF に切り替えて循環ロギングを有効にします。図 11.2 に、循環ロギングの概略を示します。

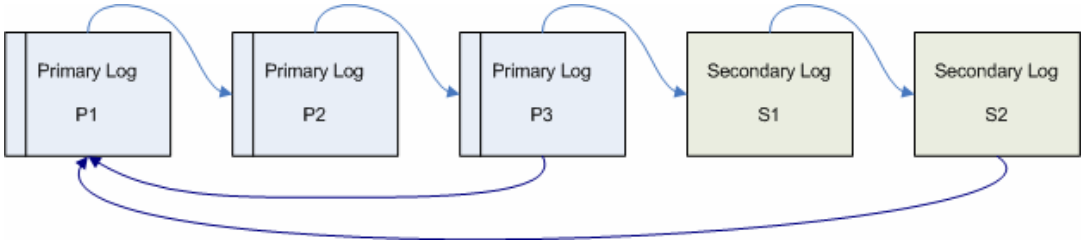


図 11.2 – 1 次ログおよび 2 次ログの操作

図 11.2 には 3 つの 1 次ログがあります。つまり、LOGPRIMARY パラメーターの値は 3 に設定されているということです。話を簡単にするために、この例では 1 つのトランザクションのみが実行されていることにします。トランザクションの実行に伴い、ログ・ファイル P1、続いてログ・ファイル P2 へとログが書き込まれていきます。コミットが行われ、続いて情報が表スペース・ディスクに外部化されると、P1 と P2 を上書きできるようになります。この情報はクラッシュ・リカバリーに必要なくなるためです (これについては、この章の後で詳しく説明します)。一方、トランザクションが長期間続いて P1、P2、P3 を使用し、それでもまだトランザクションがコミットも、外部化もされていないために、さらにログ・スペースが必要になると、2 次ログ (図中の S1) が動的に割り当てられます。さらにトランザクションが続いた場合には、LOGSECOND ログの最大数が割り当てられるまで、2 次ログが追加で割り当てられていきます。最大数を超えるログが必要になると、ログがフルの状態に達したことを通知するエラー・メッセージがユーザーに返されて、トランザクションがロールバックします。あるいは BLK_LOG_DSK_FUL 構成パラメーターを使用して、トランザクションを停止状態にさせて 5 分間隔でログへの書き込みを続けるように DB2 を構成することもできます。この方法では DBA に新しいスペースを探すための時間が与えられるため、トランザクションの継続が可能になります。

循環ロギングでは、クラッシュ・リカバリーによるリカバリーが可能ですが、データを特定の時点のデータでリカバリーしたい場合、利用可能な最新のデータは、最後にオフライン・バックアップを行った時点のデータになります。

11.3.2 アーカイブ・ロギング

ログ保存ロギングとしても知られるアーカイブ・ロギングでは、ログ・ファイルは上書きされず、オンラインまたはオフライン・ログとして保持されます。オンライン・アーカイブ・ログはクラッシュ・リカバリーに必要なアーカイブ・ログと一緒に保持される一方、オフライン・アーカイブ・ログは別のメディア (テープなど) に移されます。この移動を可能にするのは、USEREXIT ルーチンか、Tivoli Storage Manager、あるいは他のサード･パーティーのアーカイブ製品です。

アーカイブ・ロギングを有効にするには、データベース構成パラメーター LOGARCHMETH1 または LOGARCHMETH2 (あるいはその両方) を OFF 以外の値に設定します。また別の方法としては、構成パラメーター LOGRETAIN を RECOVERY に設定することでも有効にすることができます。この場合、LOGARCHMETH1 は自動的に LOGRETAIN に設定されます。ただし、LOGRETAIN パラメーターの使用は推奨されていません。このパラメーターが現在残されているのは、主に DB2 の古いバージョンとの互換性のためです。

アーカイブ・ロギングは通常、本番システムで使用されます。ログが保持されることから、最も古いログ・ファイルの時点に遡った状態へとデータベースをリカバリーすることができます。アーカイブ・ロギングにより、DBA は人的エラーからリカバリーすることができます。例えば、システムのユーザーが不用意に数日間続く誤ったトランザクションの実行を開始してしまった場合、問題が見つかったとしても、DBA はシステムを問題が起こった以前の状態にシステムをリストアすることができます。ただし、トランザクションを正しく再実行するために手動操作が必要になる場合もあります。

アーカイブ・ロギングは、ロールフォワード・リカバリーおよびオンライン・バックアップに必要です。図 11.3 に、アーカイブ・ロギングのプロセスを示します。

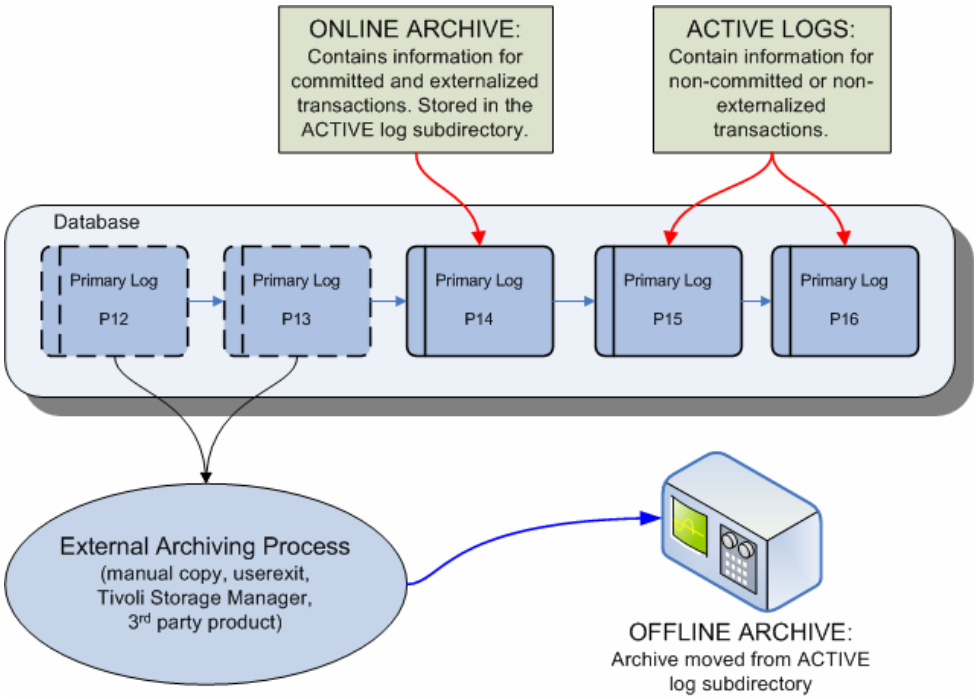


図 11.3 – アーカイブ・ロギング

11.4 「コントロール・センター」から行うデータベース・ロギング

データベース・ロギングを構成するには、コントロール・センターで対象のデータベースを右クリックし、「データベース・ロギングの構成」を選択します。図 11.4 を参照してください。

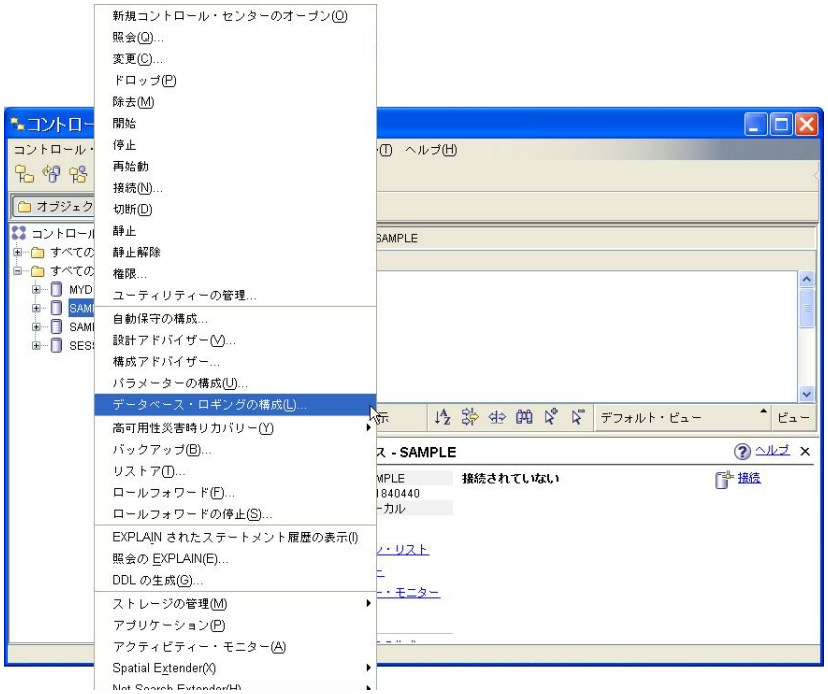


図 11.4 – コントロール・センターからのデータベース・ロギングの構成

図 11.5 に示すデータベース・ロギングの構成ウィザードでは、循環ロギングまたはアーカイブ・ロギングを選択することができます。

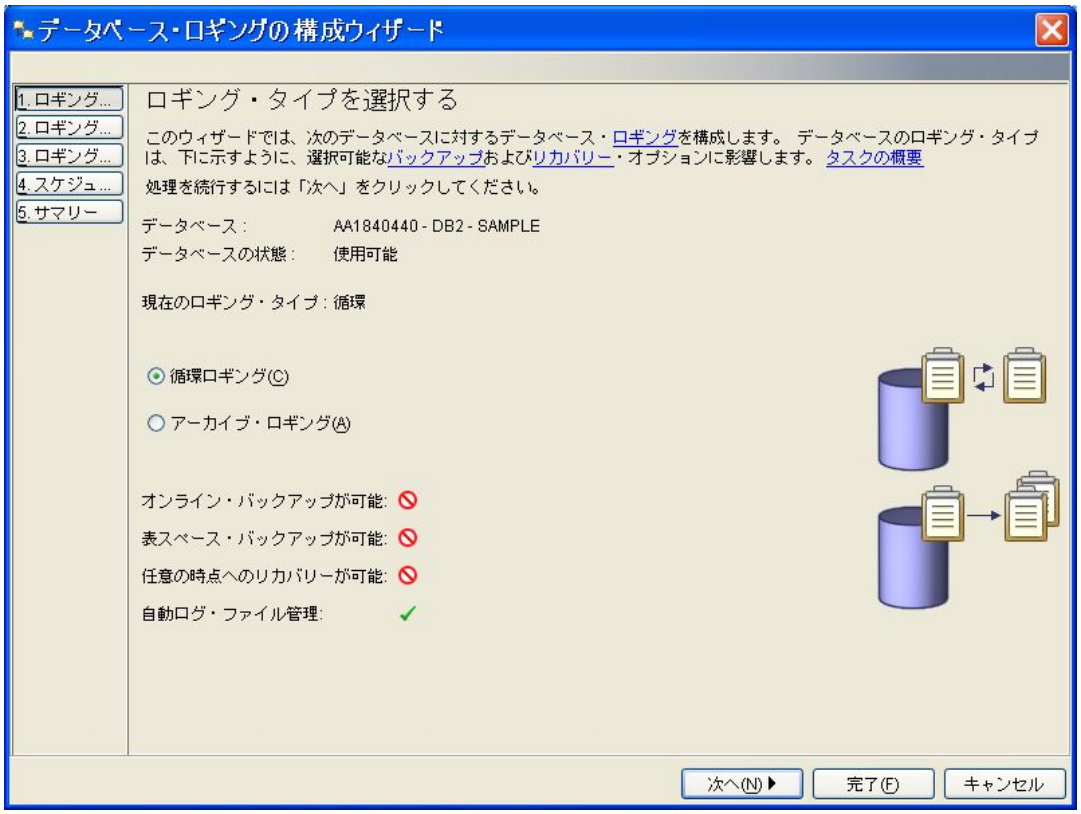


図 11.5 – データベース・ロギングの構成ウィザード

11.5 ロギング・パラメーター

ロギング関連の DB CFG パラメーターは多数あります。そのうちの主要なパラメーターを表 11.1 に記載します。

パラメーター	説明
logbufsz	ログ・レコードをディスクに書き込むまでにログ・レコードのバッファーとして使用するメモリーの量
logfilsz	構成する各ログのサイズ (4KB ページ単位)
logprimary	logfilsz のサイズで作成する 1 次ログ・ファイルの数
logsecond	必要に応じてリカバリー用に作成する 2 次ログ・ファイルの数
newlogpath	データベースのアクティブ・ログおよびオンライン・アーカイブ・ログが最初に作成されるのは、データベース・ディレクトリーの SQLOGDIR サブディレクトリー内です。この場所を変更するには、この構成パラメーターの値が別のディレクトリーまたはデバイスを指すように変更します。

mirrorlogpath	1 次ログ・パス上のログをディスク障害または不慮の削除から保護するため、1 次ログと同一のログ一式を 2 次 (ミラー) ログ・パスに維持するように指定するパラメーター
logarchmeth1 / logarchmeth2	アクティブ・ログ・パス以外の場所を指定してアーカイブ・ログ・ファイルを保管するためのパラメーター。この 2 つのパラメーターが両方とも指定されている場合、各ログ・ファイルは 2 度保存されます。つまり、異なる 2 つの場所のそれぞれに、アーカイブ・ログ・ファイルのコピーが保存されるということです。設定可能な値は OFF (循環ロギングが有効になります)、LOGRETAIN、USEREXIT、DISK、TSM、VENDOR です。
loghead	現在アクティブなログ・ファイルの名前
softmax	クラッシュ・リカバリーのコストを制限するパラメーター
overflowlogpath	ロールフォワード操作に必要なログ・ファイルを DB2 が検索する場所を指定します。ROLLFORWARD コマンドの OVERFLOW LOG PATH オプションと同様のパラメーター。
blk_log_dsk_ful	DB2 がアーカイブ・ログ・パスに新しいログ・ファイルを作成できない場合にディスク・フル・エラーが生成されないようにするためのパラメーター。エラーを生成する代わりに、DB2 はログ・ファイルを作成できるまで、5 分ごとにログ・ファイルの作成を試行します。非ブロック化された読み取り専用の SQL は続行します。
max_log	トランザクションごとのアクティブな最大ログ・スペースのパーセンテージ
num_log_span	アクティブな UOW のアクティブ・ログ・ファイル数
mincommit	ディスクに書き込む前にグループ化するコミット数

表 11.1 – ロギング・パラメーター

11.6 データベースのバックアップ

DB2 バックアップ・コマンドを実行すると、その時点でのデータベースのスナップショット・コピーを作成することができます。このコマンドの実行に使用できる最も単純な構文は以下のとおりです。

```
BACKUP DATABASE <dbname> [ TO <path> ]
```

大部分のコマンドとユーティリティーは、オンラインでもオフラインでも実行することができます。オンラインでは、コマンドの実行中に他のユーザーがデータベースに接続して操作を実行する可能性があります。一方オフラインでは、操作を実行している間、他のユーザーは誰もデータベースに接続しません。オンライン操作を可能にするには、コマンド構文にキーワード、ONLINE を追加します。このキーワードを追加しないと、コマンドはデフォルトでオフラインでの実行を前提とします。

例えば、**SAMPLE** データベースをパス C:\BACKUPS にバックアップするとします。この場合、DB2「コマンド・ウィンドウ」または Linux シェルで以下のコマンドを実行することができます。

```
db2 BACKUP DB sample TO C:\BACKUPS
```

C:\BACKUPS ディレクトリーは、コマンドを実行する前に存在していなければならないことに注意してください。また、上記のコマンドを実行するときには、データベースへの接続が一切ないことを確認する必要があります。接続があるときにはオフライン・バックアップの実行は不可能なため、エラー・メッセージを受け取ることになります。

インスタンス内のデータベースへの接続があるかどうかを調べるには、以下のコマンドを DB2 コマンド・ウィンドウまたは Linux シェルから実行します。

```
db2 list applications
```

インスタンス内のすべてのデータベースへのすべての接続を強制的に切断するには、以下のコマンドを DB2 コマンド・ウィンドウまたは Linux シェルから実行します。

```
db2 force applications all
```

多数のユーザーがいる本番環境では、上記最後のコマンドを実行すると、怒った他の作業者からの電話が殺到するおそれがあるため、実行は控えたほうがいいかもしれません。もう 1 つの注意事項として、このコマンドは非同期で実行されるので、このコマンドを実行した直後にバックアップ・コマンドを実行しようとしても上手くいかない可能性があります。バックアップ・コマンドを実行してエラー・メッセージを受け取った場合には、数秒待ってからもう一度実行してください。

バックアップ・コマンドの実行が正常に完了すると、バックアップ・データベース・イメージが含まれる新しいファイルが作成されます。このファイルの名前は、図 11.6 に示す規則に従って設定されます。

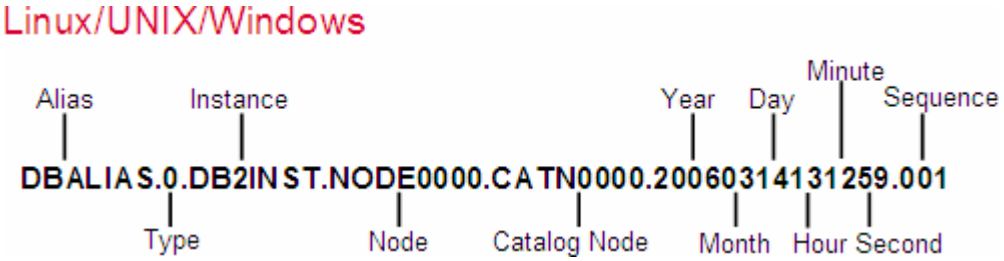


図 11.6 – バックアップ・イメージの命名規則

タイプ「0」は、バックアップがフル・バックアップであるという意味です。タイプ「3」は、表スペース・バックアップであることを意味しています。ノードは、パーティション化されていないデータベースの場合、NODE0000 に固定されます。DPF のフィーチャーを備えた DB2 Enterprise Edition を除き、すべての DB2 エディションはこれに該当します。カタログ・ノードも同じく CATN0000 に固定されます。詳細については、DB2 マニュアルを参照してください。

複数のバックアップを作成して同じパスに保管するときには、それぞれのバックアップ・イメージを区別するためにファイル名の最後に付けられたタイムスタンプが使用されます。次のセクションで説明するように、RESTORE コマンドでは、このタイムスタンプを使用して特定のバックアップからリストアすることが可能です。

11.7 データベースのリカバリー

データベースのリカバリーとは、バックアップやログからデータベースをリストアすることを意味します。バックアップだけを使用してリストアする場合、そのバックアップが作成された時点の状態にデータベースが再作成されることになります。

バックアップを作成する前にアーカイブ・ロギングが有効になっていた場合には、バックアップ・イメージを使用してリストアするだけでなく、さらにログからもリストアすることができます。次のセクションで説明するようにロールフォワード・リカバリーを使用すると、バックアップからリストアし、それからログの最後、または特定の時点にログを適用 (ロールフォワード) することができます。

このセクションでは「リカバリー」という言葉を頻繁に使用しますが、リカバリーに使用するコマンドの名前は「RESTORE」であることに注意してください。

11.7.1 リカバリーのタイプ

リカバリーには以下の 3 つのタイプがあります。

- **クラッシュまたは再始動リカバリー**

  例えば、デスクトップ・コンピューターで DB2 データベースに対して重要なトランザクションを実行している途中で、停電が発生したり、あるいは誰かが誤って電源コードを抜いてしまったりしたとします。この場合、データベースにはどのような影響があるでしょうか。

  コンピューターを再起動し、DB2 を再始動すると、クラッシュ・リカバリーが自動的に実行されます。クラッシュ・リカバリーでは、DB2 が自動的に RESTART DATABASE コマンドを実行し、アクティブ・ログをベースにトランザクションを読み取って再実行または取り消します。このコマンドが完了すると、データベースが整合状態にあることが保証されます。つまり、コミットされた内容は保存され、コミットされていない内容はロールバックするということです。

- **バージョンまたはイメージ・リカバリー**

  このタイプのリカバリーは、バックアップ・イメージのみからのリストアを意味します。そのため、データベースはバックアップが作成された時点の状態になります。バックアップが作成された後にデータベースで行われたトランザクションは失われます。

- **ロールフォワード・リカバリー**

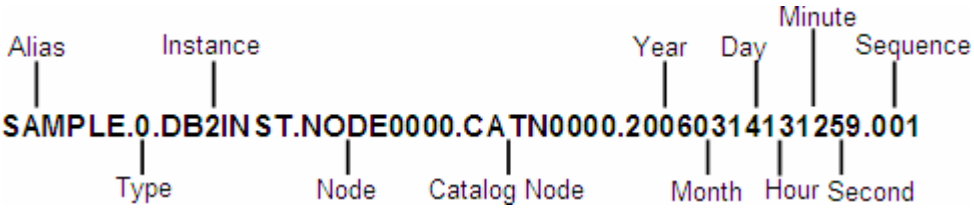
  このタイプのリカバリーでは、RESTORE コマンドによりバックアップ・イメージからリストアするだけでなく、ROLLFORWARD コマンドを実行してバックアップをベースにさらにログを適用するため、特定の時点の状態までデータベースをリカバリーさせることができます。このタイプのリカバリーは、データ損失を最小限に抑えることになります。

11.7.2 データベースのリストア

バックアップ・イメージからデータベースをリカバリーするには、RESTORE コマンドを使用します。このコマンドに使用できる最も単純な構文は以下のとおりです。

```
RESTORE DATABASE <dbname> [from <path>] [taken at <timestamp>]
```

例えば、以下の名前を持つ sample データベースのバックアップ・イメージ・ファイルがあるとします。



この場合のコマンドは以下のようになります。

```
RESTORE DB sample FROM <path> TAKEN AT 20060314131259
```

11.8 BACKUP および RESTORE によるその他の操作

以下に、BACKUP コマンドと RESTORE コマンドを使用して実行できるその他の操作を記載します。詳細については、DB2 マニュアルを参照してください。

- 32-bit のインスタンスにあるデータベースをバックアップし、64-bit のインスタンスにリストアする
- 既存のデータベースに上書きリストアする
- リストアする際に、バックアップ・イメージに指定されたディスク数とは異なる数のディスクがあるシステムへのリダイレクト・リストアを利用する
- データベース全体でなく、表スペースごとにバックアップまたはリストアする
- デルタ・バックアップまたはインクリメンタル・バックアップを実行する。デルタ・バックアップではバックアップ間の変更のみを記録する一方、インクリメンタル・バックアップではすべての変更を記録してバックアップ・イメージのそれぞれで変更を累積していきます。
- フラッシュ・コピーからバックアップする (適切なハードウェアが必要です)
- ドロップされた表をリカバリーする (対象となる表でこのオプションが有効にされている場合)
- バックアップを作成したプラットフォーム (例えば Windows) とは異なるプラットフォーム (例えば Linux) でリストアする。このシナリオでは、db2look と db2move を使用します。UNIX オペレーティング・システムをサポートする DB2 エディションの場合、UNIX オペレーティング・システム内には UNIX プラットフォーム間でのバックアップおよびリストアが可能なプラットフォームが何種類かあることに注意してください。

11.9 まとめ

この章では DB2 でのロギング機能について説明しました。取り上げた内容は、ログの 2 つのタイプ (1 次ログと 2 次ログ)、ロギングの 2 つのタイプ (循環ロギングとアーカイブ・ロギング)、そしてロギングに関連するさまざまなパラメーターです。ロギング・タイプについては、それぞれのタイプをいつ、どのような目的で使用するか、そして「コントロール・センター」から設定する方法を説明しました。

また、DB2 コマンド行を使用してバックアップおよびリストア操作を実行する方法を検討し、データベース・リストアについてはクラッシュ、バージョン、およびロールフォワードという 3 つのタイプについて詳しく説明しました。

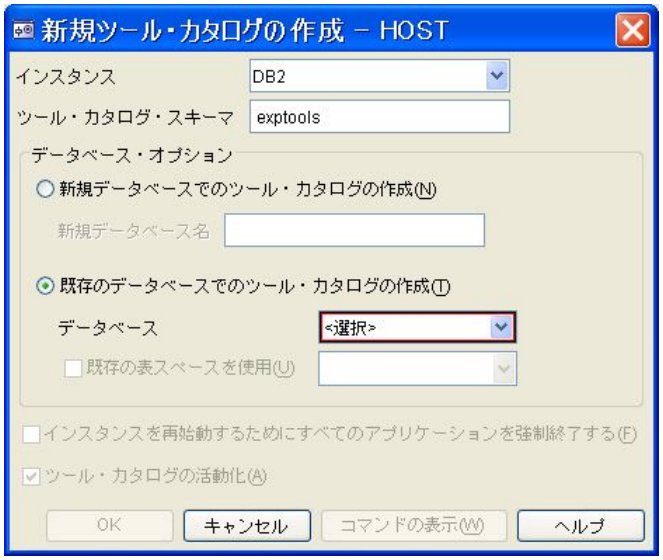
11.10 演習　バックアップをスケジュールする

DB2 では複数のデータベース保守アクティビティーを自動化することができますが、特定のアクティビティーが発生するタイミングをカスタマイズする必要がある場合もあります。この演習では、***EXPRESS*** データベースの夜間バックアップ・スケジュールをカスタマイズします。

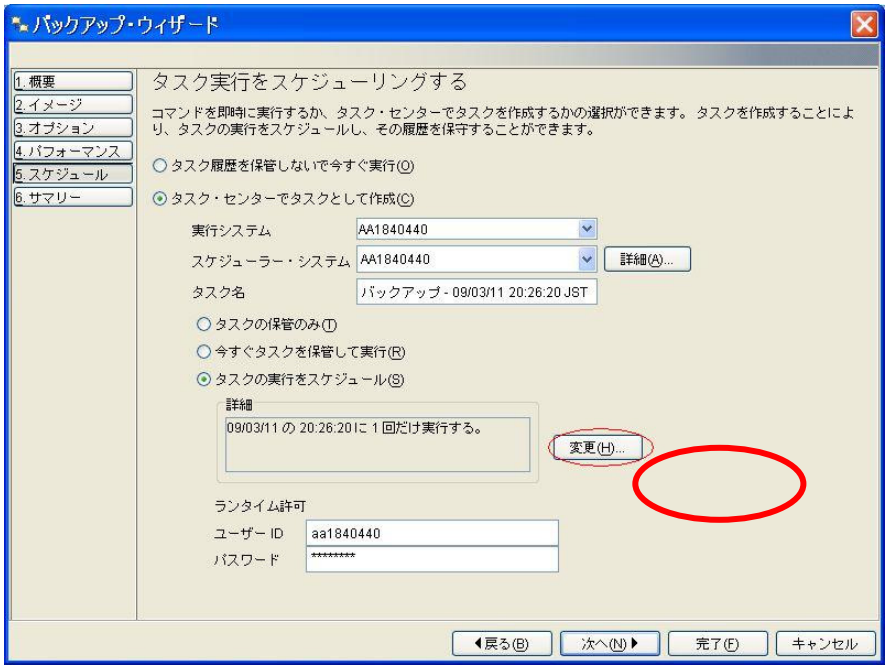
手順

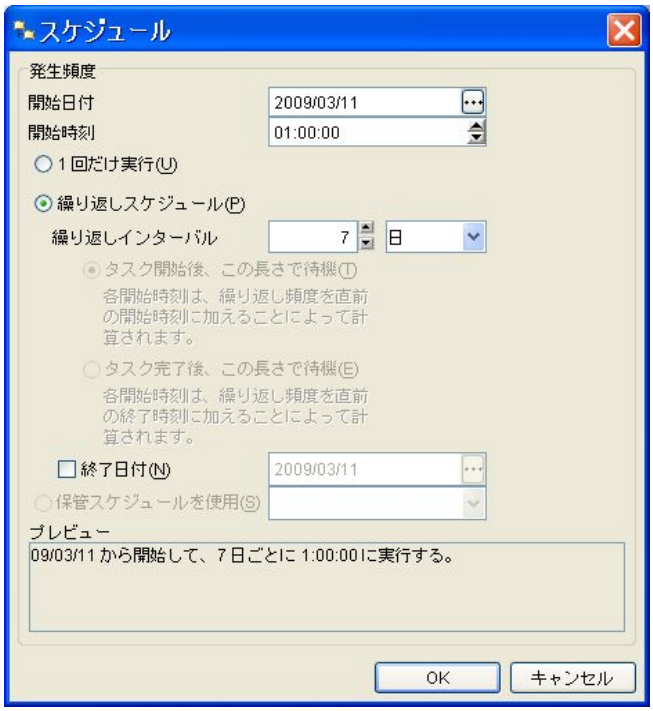
1. コントロール・センターのオブジェクト・ツリーで、「コントロール・センター」 -> 「すべてのデータベース」までナビゲートします。***EXPRESS*** データベースを右クリックして、「バックアップ」を選択します。この操作によって、バックアップ・ウィザードが起動します。
2. ウィザードの「概要」ページに、データベースの現在の状態 (最後にバックアップされた時間、ロギング方式など) が要約されます。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
3. ウィザードの「イメージ」ページで、バックアップ・イメージの保存先を選択します。通常は、既存のデータベースが保管されているドライブとは別の物理ドライブを選択します。ここでは、ファイル・システムに `C:\db2backup` という新しいフォルダーを作成して、このフォルダーをバックアップ・ロケーションとして指定することにします。ウィザードの「メディア・タイプ」ドロップダウン・リストから「ファイル・システム」を選択し、「追加」ボタンをクリックします。新しく作成したフォルダーを選択して、「OK」ボタンをクリックします。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
4. 「オプション」ページおよび「パフォーマンス」ページで詳細な設定をすることもできますが、DB2 は最適なデータベース・バックアップを自動的に実行するため、通常はデフォルト・オプションで十分です。詳細の検討が終わったら、「スケジュール」ページまでナビゲートしてください。
5. 「スケジュール」ページでスケジューラーが有効になっていない場合は、スケジューラーを使用可能にするオプションを選択します。ツールカタログを作成するシステムを選択し、新しいツールカタログを作成します。ツールカタログのスキーマを指定して、既存の ***EXPRESS*** データベースでツールカタログを作成するように選択します。ツールカタログは、スケジュールされたすべてのタスクに関するメタデータを保持します。「OK」ボタンをク

リックして先に進みます。ツールカタログの作成が完了したら、「次へ」ボタンをクリックして、ウィザードの次のページに進んでください。



6. 「スケジュール」ページで、タスクの実行スケジュールを作成するオプションを選択します。ここでは毎日午前 1 時にバックアップが開始されるようにスケジュールしてください。「次へ」ボタンをクリックして、次のページに進みます。





7. 「サマリー」ページでは、スケジュール済みタスクが作成される前にその内容を確認することができます。変更内容を確認したら、「完了」ボタンをクリックしてタスクを作成します。

8. タスク・センターを起動して、新しく作成したバックアップ・タスクを確認、または変更します。.

12

第 12 章 – 保守タスク

この章では、データベースを良好な状態に維持しておくために必要となるタスクについて説明します。DB2 では全体的な方向性として、このような保守タスクの大部分を自動化しています。そのため、現行のあらゆる DB2 エディションの例に漏れず、DB2 Express-C エディションにもそのような自動化機能が組み込まれています。この自己管理機能は、フルタイムの DBA を雇ってデータ・サーバーを管理できない中小企業にとっては大きなメリットをもたらします。一方 DBA を雇っている場合でも、この自己管理機能があるおかげで、DBA は企業の収益につながる高度な作業に多くの時間を割けるようになります。

12.1 REORG、RUNSTATS、REBIND

DB2 での主な保守タスクには、図 12.1 に示すように REORG、RUNSTATS、REBIND の 3 つがあります。

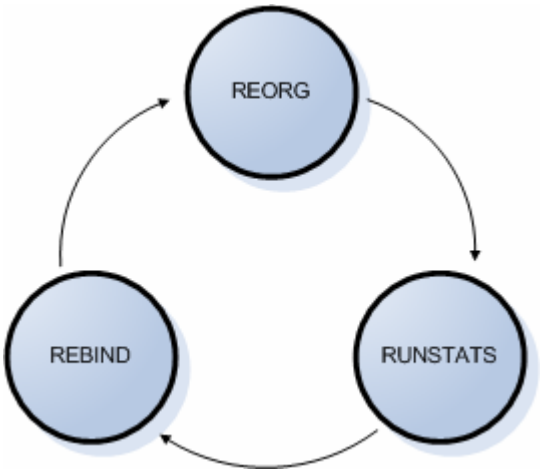


図 12.1 – 保守タスク: REORG、RUNSTATS、REBIND

図 12.1 に示されている保守タスクは循環的に実行されます。REORG を実行したら、次に RUNSTATS、続いて REBIND を実行することが推奨されています。データベースの表は一定の時間が経過すると、UPDATE、DELETE、および INSERT 操作によって変更されていきます。変更に伴って、保守タスクのサイクルが REORG から再度開始されることになります。

12.1.1 REORG コマンド

データベースで INSERT、UPDATE、DELETE 操作の実行を重ねるにつれ、データは次第にデータベース・ページ全体で断片化されていきます。REORG は、無駄になっているスペースを再利用してデータを再編成し、効率的にデータを取得できるようにするためのコマンドです。そのため、頻繁に変更される表には、この REORG コマンドが極めて役立ちます。REORG は表だけでなく索引に対しても実行することができます。また、オンラインとオフラインのどちらでも実行することができます。

オフラインでの REORG は高速で効率的ですが、表にアクセスすることはできません。一方、オンラインで REORG を実行する場合、表にはアクセスできますが、システム・リソースを大量に消費する可能性があるため、オンラインでの REORG が最適なのは小規模な表の場合です。

構文

```
REORG TABLE <tablename>
```

例:

```
REORG TABLE employee
```

REORG を実行する前に REORGCHK コマンドを使用すると、表または索引を修正する必要があるかどうかを判断することができます。

12.1.2 RUNSTATS コマンド

DB2 オプティマイザーは DB2 の「頭脳」として、データを配置および取得するのに最も効率的なアクセス・パスを見つけます。このオプティマイザーはシステムのコストを認識し、データベースのパフォーマンスを最大にするために、カタログ表に保管されたデータベース・オブジェクトの統計を利用します。カタログ表には例えば、表内の列数、行数、表に使用できる索引の数とタイプなどに関する統計が保管されます。

統計情報は動的には更新されません。これは意図的なもので、データベースに対する操作が実行されるたびに DB2 に統計を更新させるとしたら、データベースのパフォーマンス全体に悪影響が出るためです。代わりに DB2 には、データベース・オブジェクトの統計を更新する RUNSTATS コマンドが用意されています。データベースの統計を最新の状態に維持するためには、このコマンドが必須です。DB2 オプティマイザーは、表の行数が 100 万行の場合と 1 行の場合とでは、アクセス・パスを大幅に変更することができます。データベース統計が最新であれば、DB2 はより適切なアクセス・プランを選択できるのです。統計の収集頻度は、表内のデータが変更される頻度に応じて決める必要があります。

構文:

```
RUNSTATS ON TABLE <schema.tablename>
```

例:

```
RUNSTATS ON TABLE myschema.employee
```

12.1.3 BIND / REBIND

RUNSTATS コマンドが正常に完了しても、すべてのクエリーが最新の統計を使用するわけではありません。静的 SQL アクセス・プランは最初に BIND コマンドが実行された時点で決定されるため、RUNSTATS コマンドの完了時に使用される統計は、現行の統計とは異なる可能性があります。この概念を理解する助けとなるよう、図 12.2 に図解してあります。

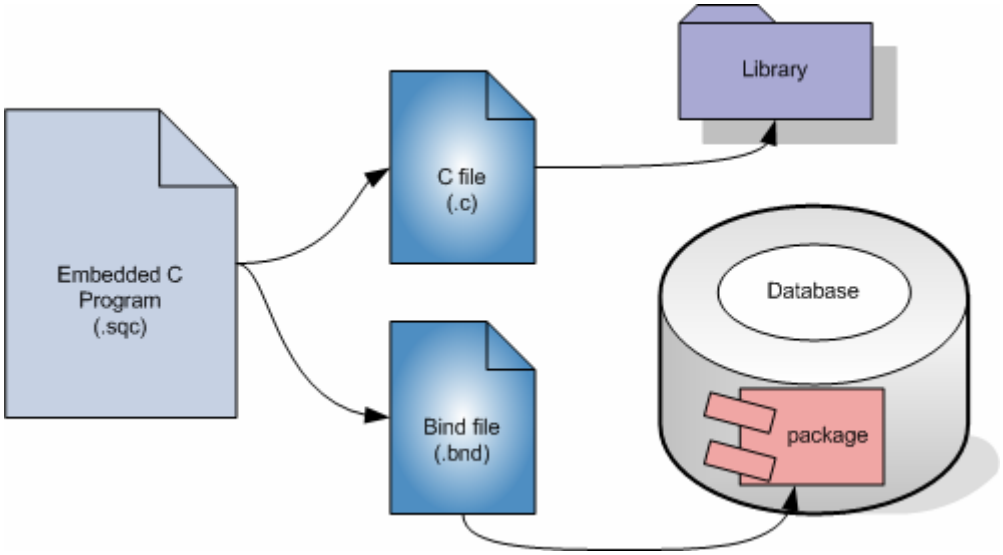


図 12.2 – 静的 SQL バインド・プロセス

図 12.2 では、まず、組み込み C プログラム (「sqc」拡張子を持つファイルとして保管) がプリコンパイルされます。プリコンパイルが完了すると、2 つのファイルが生成されます。1 つは、すべての SQL がコメント化された C コードが含まれる「.c」ファイル、そしてもう 1 つは、すべての SQL ステートメントが含まれる「.bnd」ファイルです。「.c」拡張子を持つ C ファイルは C コンパイラーによって通常と同じようにコンパイルされ、図の右上にある「ライブラリー」が作成されます。「.bnd」ファイルも同様にバインドされ、このバインディングによって生成されたパッケージがデータベースに保管されます。バインディングは SQL ステートメントのコンパイルに相当し、その時点で使用可能な統計に基づいて最適なアクセス・プランが決定され、パッケージに保管されます。

ここで、この組み込み C プログラムの SQL で使用する表に 100 万行が挿入されるとしたらどうなるでしょう。行の挿入後に RUNSTATS が実行されると統計は更新されますが、パッケージが自動的に更新されることはないため、この最新の統計に基づいてアクセス・パスが再計算されることもありません。そこで、**db2rbind** コマンドを使用して既存のすべてのパッケージを再バインドし、最新の統計が考慮されるようにます。

構文:

```
db2rbind database_alias -l <logfile>
```

例:

SAMPLE データベースのすべてのパッケージを再バインドし、出力ログを `mylog.txt` ファイルに保管するには、以下のコマンドを実行します。

```
db2rbind sample -l mylog.txt
```

12.1.4 コントロール・センターからの保守タスク

REORG および RUNSTATS は、コントロール・センターから実行することができます。図 12.3 にその方法を示します。

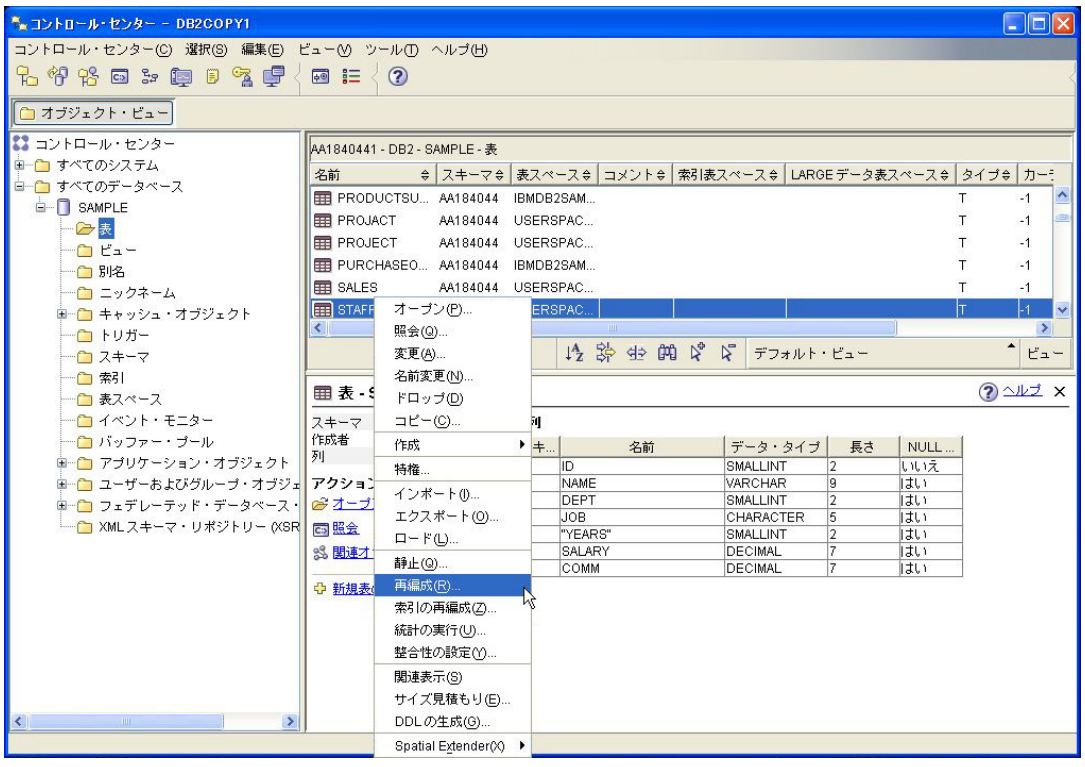


図 12.3 – コントロール・センターから実行する REORG および RUNSTATS

操作対象の表を選択したら、その表を右クリックして「再編成」 (REORG の場合) または「統計の実行」 (RUNSTATS の場合) を選択します。

12.1.4.1 データベース運用状態ビュー

データベースを選択すると、コントロール・センターの右下にあるデータベース運用状態ビューにデータベースに関する情報 (データベースのサイズ、最新のバックアップが行われた日時、自動保守が設定されているかどうかなど) が表示されます。このビューを見れば、データベースに保守が必要かどうかを素早く識別することができます。図 12.4 に、このビューに表示される情報を示します。

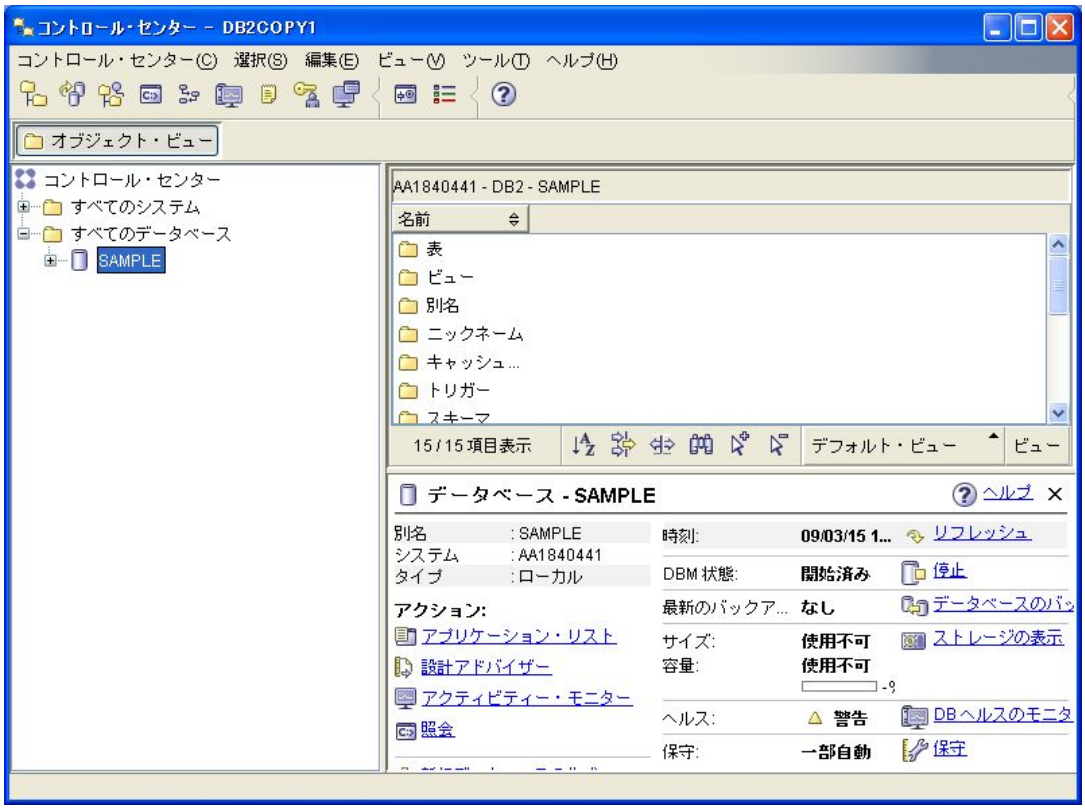


図 12.4 – コントロール・センターのデータベース運用状態ビュー

12.2 保守の選択肢

保守タスクを実行する方法には、以下の 3 つの選択肢があります。

- 手動による保守
- 必要に応じて手動で保守アクティビティーを実行します。
- スクリプトを作成して実行することによる保守
- 保守コマンドを含めたスクリプトを作成し、定期的にスクリプトが実行されるようにスケジュールすることができます。
- 自動保守
- DB2 が自動的に保守 (REORG、RUNSTATS、BACKUP) を行うようにします。

このセクションでは、自動保守の方法に焦点を絞って説明します。

自動保守の内容は以下のとおりです。

- 保守タスクを実行することによる影響が最小限になるような保守時間枠をユーザーが指定します。例えば、システムのアクティビティーが最も少なくなるのが日曜日の午前 2 時から午前 4 時までの間だとすれば、この時間帯を保守時間枠にします。
- オンライン操作とオフライン操作それぞれを対象とした 2 つの保守時間枠があります。
- DB2 は保守時間枠のなかで、保守が必要になったときだけ保守操作を自動的に実行します。

「コントロール・センター」から「自動保守の構成」ウィザードを起動するには、図 12.5 に示すように操作します。

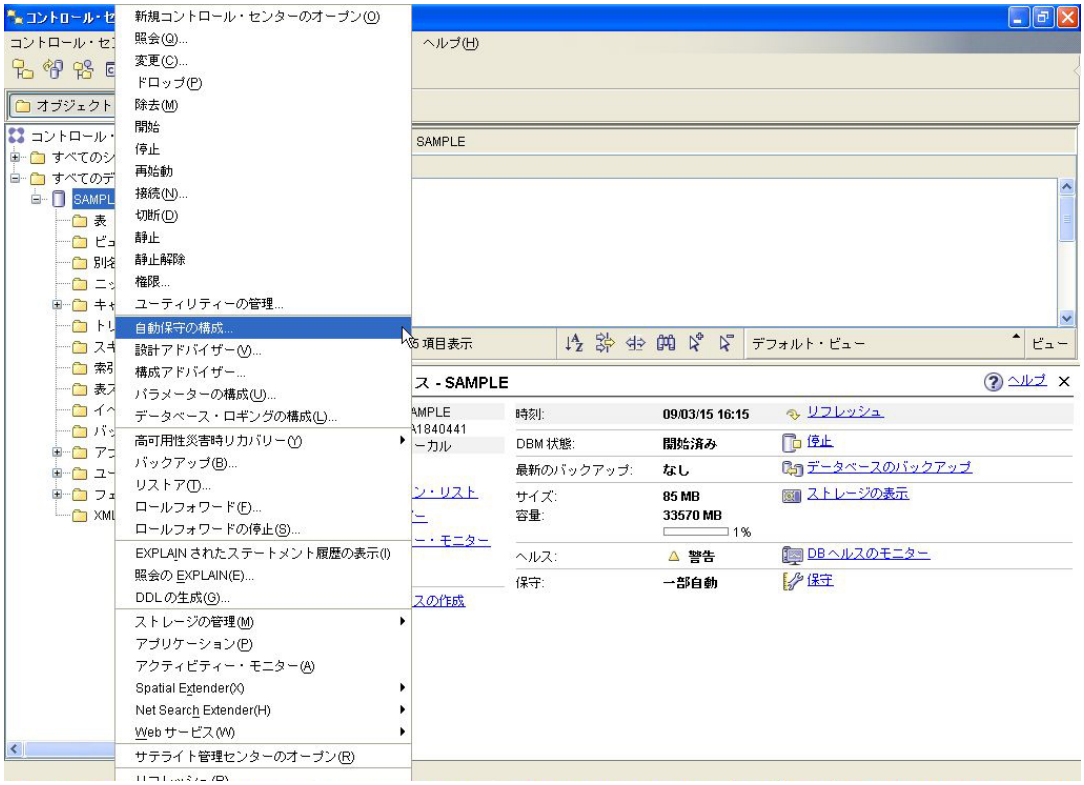


図 12.5 – 自動保守の構成ウィザードの起動

図 12.6 に「自動保守の構成」ウィザードを示します。

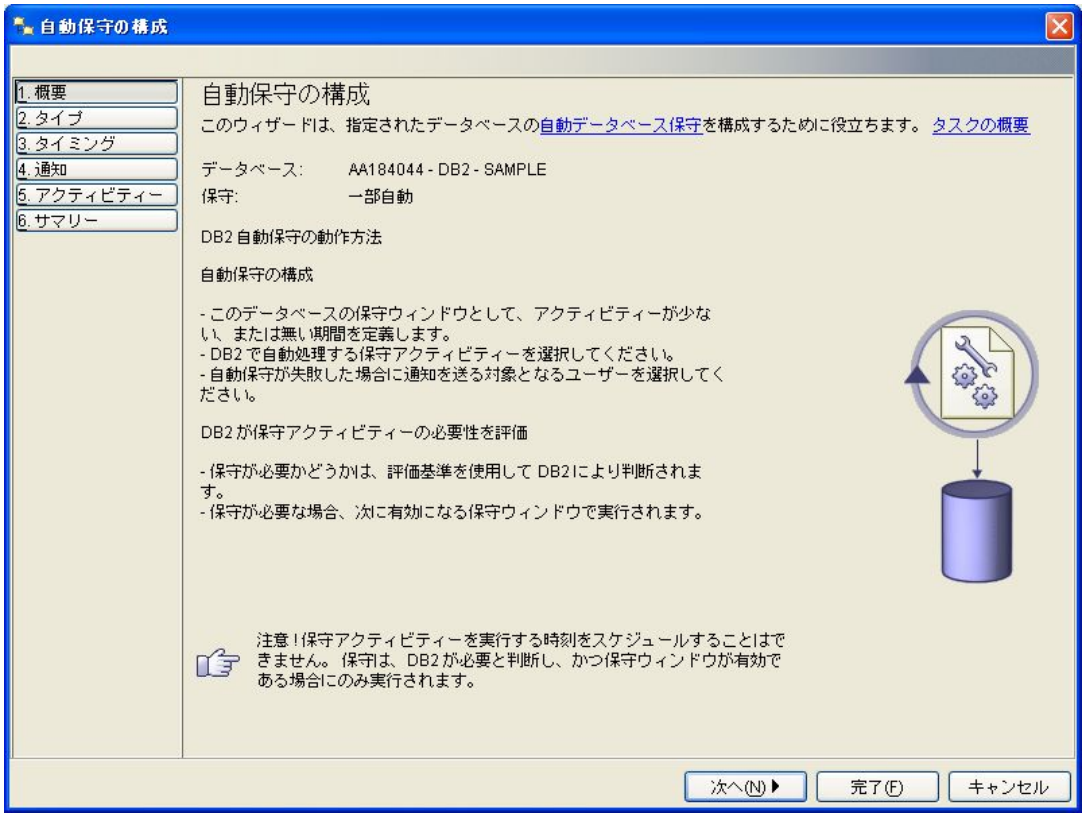


図 12.6 – 自動保守の構成ウィザード

12.3 まとめ

この章では、REORG、RUNSTATS、REBIND サイクルの役割をはじめとする、データベースの保守の重要性を説明してきました。REORG コマンドはその名のとおり、データの再構成を行い、断片化されているデータをなくすことで、データの取得を迅速に行えるようにします。RUNSTATS は、DB2 の最適化ツールが使用する統計情報を更新することで、データ操作時のパフォーマンスを向上させます。BIND / REBIND プロセスは、最新のアクセス･パスを使ってデータベース･パッケージを更新します。

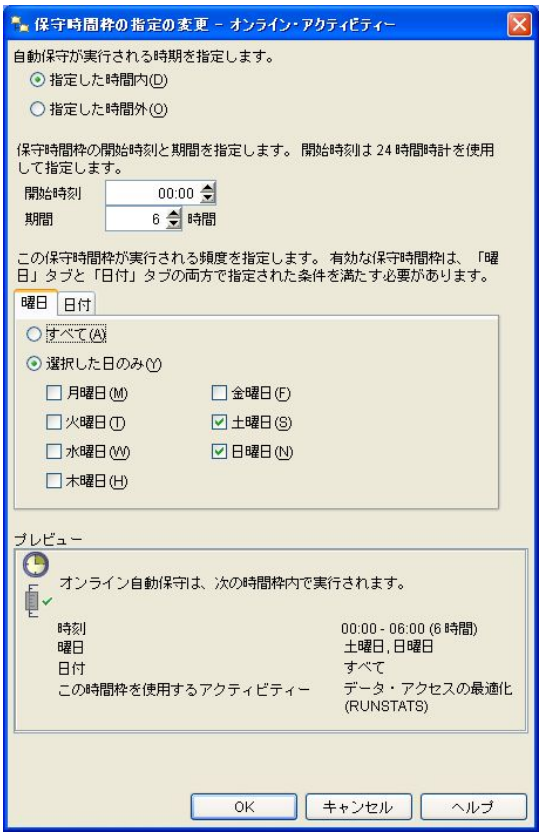
本章ではまた、DB2 コントロール･センターのグラフィカル･ツールを使用して、保守作業を手動で行う方法、スクリプトを使用して行う方法、自動で行う方法についても説明しました。

12.4 演習　自動保守を構成する

この演習では、DB2 **SAMPLE** データベースの自動保守を構成します。

手順

1. コントロール・センターのオブジェクト・ツリーで **SAMPLE** データベースを右クリックし、「自動保守の構成」メニュー項目を選択します。これにより、「自動保守の構成」ウィザードが起動します。
2. ウィザードの「概要」ページには、現在の自動保守設定が表示されます。自動保守オプションを選択してデータベースを作成した場合、自動保守はすでに構成済みになっていますが、自動保守オプションを再構成する場合にも、このウィザードを使用することができます。「次へ」ボタンをクリックして、ウィザードの次のページに進みます。
3. ウィザードの「タイプ」ページでは、すべての自動化を使用不可にするか、または自動化設定を変更するかを選択します。現在の自動化設定を変更するオプションを選択してください。「次へ」をクリックします。
4. ウィザードの「タイミング」ページで、保守時間枠を指定します。以下に示すように、土曜日と日曜日の深夜 0 時から午前 6 時までの間、データベースがオフラインになるように構成します。オフライン保守時間枠プレビュー・ペインの横にある「変更」ボタンをクリックして、目的の時間を選択してください。必要な情報を指定し終わったら、「OK」ボタンをクリックしてウィザードに戻ります。オンライン時間枠の設定は変更しません (つまり、オンライン保守をいつでも行えるようにします)。「次へ」ボタンをクリックします。



5. ウィザードの「通知」ページでは、自動保守アクティビティーが失敗した場合の連絡先を設定することができます。ここではこのステップをスキップして、「次へ」ボタンをクリックします。

6. ウィザードの「アクティビティー」ページでは、特定のアクティビティーを個々に自動化したり、自動化しないようにしたりといった選択をすることができます。また、特定のアクティビティーを通知するように設定することもできます。この例では、すべての「自動化」チェック・ボックスにチェック・マークが付いていること、そして「通知」チェック・ボックスのチェック・マークが外れていることを確認してください。「次へ」ボタンをクリックします。

7. ウィザードの次のページに進む前に、データベースのバックアップ・ロケーションを構成する必要があります。バックアップを保管するロケーションは、できればディスク障害に備えて別の物理ドライブにしてください。「アクティビティー」ページで、「データベースのバックアップ」オプションを選択してから「設定の構成」ボタンをクリックします。

8. 「設定の構成」ダイアログ・ウィンドウの「バックアップ基準」タブで、「データベースのリカバリー可能性とパフォーマンスとの間で平衡を取る」オプションを選択します。「バックアップ・ロケーション」タブでは、既存のバックアップ・ロケーションを選択して「変更」ボタンをクリックし、バックアップを実行する別のロケーションを指定します (そのドライブに十分な空きがあることを確認してください)。「バックアップ・モード」タブで、「オフライン・バックアップ」が選択されていることを確認します。「OK」ボタンをクリックして、「設定の構成」ダイアログ・ウィンドウを閉じます。「次へ」ボタンをクリックします。

17. 「自動保存の構成」ウィザードの「サマリー」ページに、選択された内容が要約されて表示されます。「完了」ボタンをクリックして、変更内容を確定して実装します。

13

第 13 章 – 並行性およびロック

この章では、複数のユーザーが同時に 1 つのデータベースにアクセスし、ユーザーが互いに干渉することなく、それぞれの操作の整合性を維持できるようにするための方法を説明します。そのために理解しておかなければならないのは、以下に説明するトランザクション、並行性、ロックの概念です。

13.1 トランザクション

トランザクション (作業単位) は 1 つ以上の SQL ステートメントで構成されます。SQL ステートメントは、まとめて 1 つの単位として実行されます。つまり、トランザクションを構成するステートメントが 1 つでも失敗すると、トランザクション全体が失敗し、その時点までに実行されたすべてのステートメントがロールバックされます。トランザクションは COMMIT ステートメントで終了します。これは、新しいトランザクションが開始するという意味でもあります。図 13.1 に、トランザクションの一例を記載します。

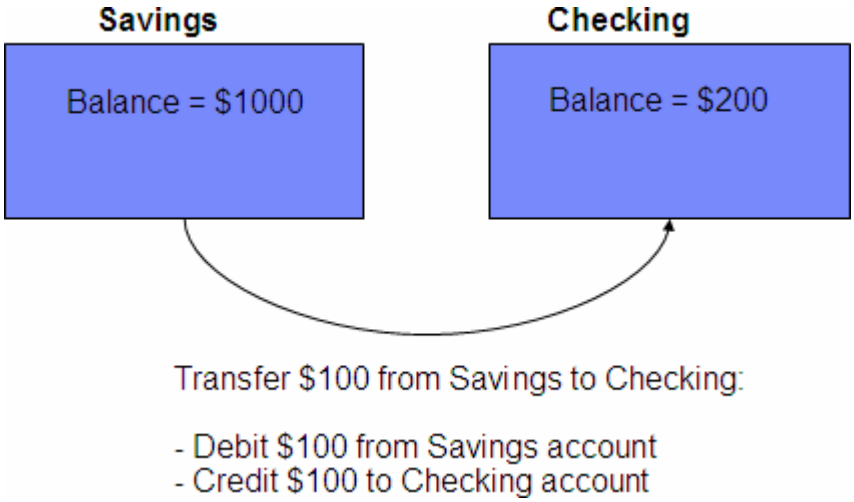


図 13.1 – トランザクションの例

図 13.1 の例では、普通預金口座から当座預金口座に 100 ドルを送金するという前提となっています。この図に示すタスクを実現するには、以下のイベント・シーケンスが必要です。

- 普通預金口座から 100 ドルを引き出します。
- 100 ドルを当座預金口座に預金します。

上記のイベント・シーケンスが 1 つの作業単位、つまりトランザクションとして処理されないとすると、例えば普通預金口座からの引き出しの後、当座預金口座に預金する前に停電が起こった場合にどうなるかを考えてみてください。当然、100 ドルの損失となります。

13.2 並行性

並行性とは、複数のユーザーが同じ 1 つのデータベース・オブジェクトを同時に操作することを意味します。DB2 はマルチユーザー・データベースとして設計されています。そのため、データの保全性と整合性を確実にするメカニズムによって、データへのアクセスは適切かつユーザーにはわからないところで調整されます。例として、図 13.2 を見てください。

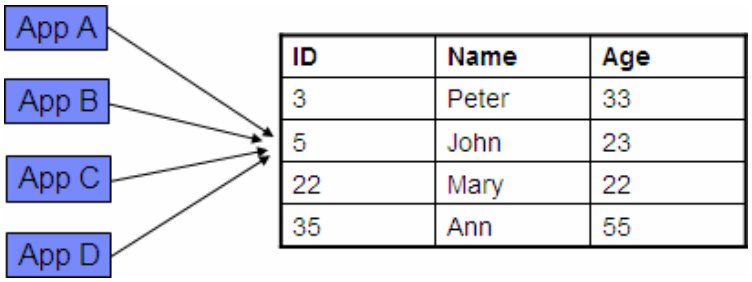


図 13.2 – 並行性の例と並行性制御の必要性

図 13.2 には 4 つのアプリケーション (App A、App B、App C、App D) があり、このすべてが表内の同じ 1 つの行 (2 行目) にアクセスしようとしています。並行性制御が何も行われていなければ、すべてのアプリケーションがこの行に対する操作を実行します。この場合、例えば 4 つのアプリケーションがそれぞれに異なる値で 2 行目の Age 列を更新すると、最後に更新を行ったアプリケーションが「勝者」になります。この例から、一貫した結果を保証するためには、ある種の並行性制御が必要であることがわかるはずです。この並行性制御は、ロックをベースとして行われます。

ロックの概念と並行性の概念は、切り離して考えることはできません。ロックは 1 つの操作が完了するまで、一時的に他のアプリケーションが操作を実行できなくなるようにします。システム内でロックが増えれば増えるほど、並行性の可能性は低くなります。一方、システム内で発生するロックが少なければ、その分、並行性の可能性は高くなります。

ロックはトランザクションをサポートするために必要に応じて自動的に取得され、トランザクションが (COMMIT または ROLLBACK コマンドによって) 終了すると解放します。ロックを取得できるのは、表または行に対してです。ロックには 2 つの基本タイプがあります。

- 共用ロック (S ロック) – アプリケーションが行を読み取るときに、他のアプリケーションがその行を更新できないようにする場合に取得されます。
- 排他ロック (X ロック) – アプリケーションが行を更新、挿入、または削除するときに取得されます。

ここで、図 13.3 を見てください。図 13.2 に似ていますが、この図にはロックが示されています。

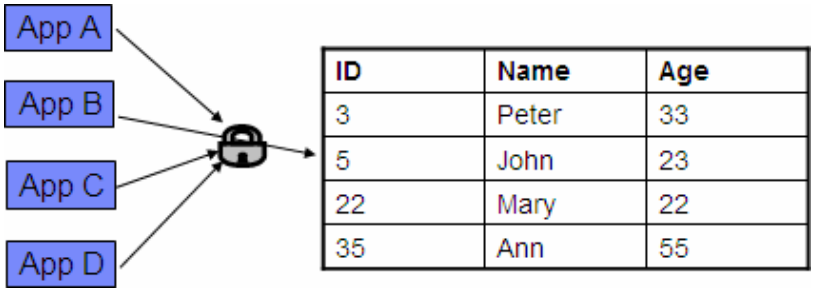


図 13.3 – 並行性の例とロックの必要性

図 13.2 で、例えば App B が最初に 2 行目にアクセスして UPDATE を実行するとします。この場合、App B はこの行で X ロックを保持します。App A、App C、App D が 2 行目にアクセスしようとしても、この X ロックのために UPDATE を実行することはできません。このような制御によって、データの整合性と保全性を維持することが可能になります。

13.3 並行性制御が行われない場合の問題

何らかの並行性制御が行われない場合には、以下の問題が発生するおそれがあります。

- 更新喪失
- 非コミット読み取り
- 反復不能読み取り
- 幻像読み取り

13.3.1 更新喪失

更新喪失という問題は、このセクションで前述した、最後の更新を実行したアプリケーションが「勝者」になるという問題と同様です。

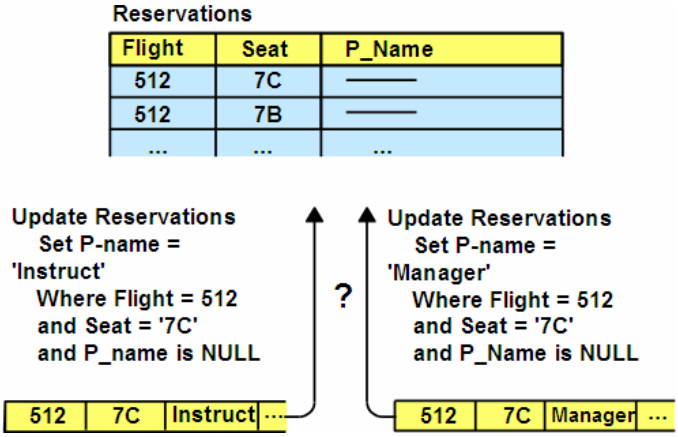


図 13.4 – 更新喪失

図 13.4 では、左側の App1 と右側の App2 の 2 つのアプリケーションが同じ行を更新しようとしています。イベント・シーケンスは以下のとおりです。

1. App1 が行を更新します。
2. App2 が同じ行を更新します。
3. App1 がコミットします。
4. App2 がコミットします。

この場合、App1 の更新は App2 が更新した時点で失われてしまいます。これが、「更新喪失」です。

13.3.2 非コミット読み取り

「ダーティー読み取り」とも呼ばれる非コミット読み取りは、まだコミットされていない情報の読み取りがアプリケーションに許可されるため、情報が必ずしも正確ではないという問題です。

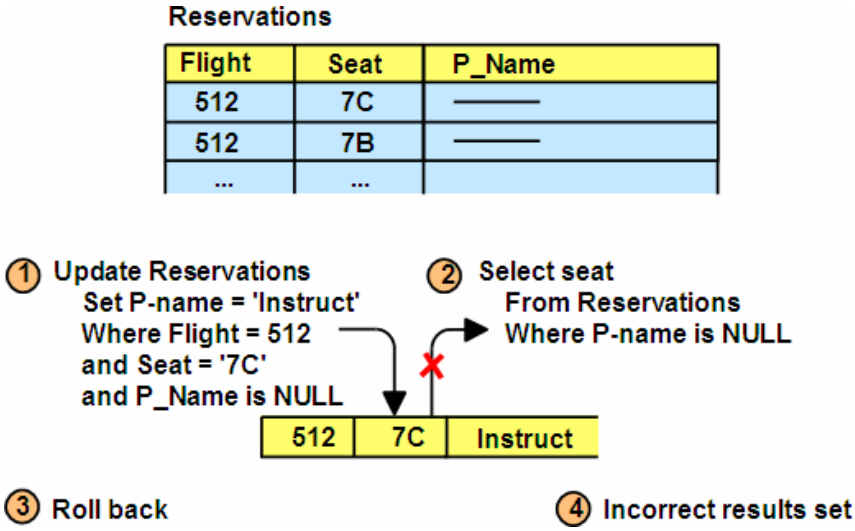


図 13.5 – 非コミット読み取り

図 13.5 のイベント・シーケンスは以下のとおりです。

1. App1 が行を更新します。
2. App2 が更新された行から新しい値を読み取ります。
3. App1 が行に対する変更をロールバックします。

App2 が読み取るのはコミットされていないデータ、つまり無効なデータです。そのため、この問題は「非コミット読み取り」と呼ばれます。”

13.3.3 反復不能読み取り

反復不能読み取りとは、同じ操作のなかで同じ読み取りを行っても、同じ結果にならないことを意味します。

FLIGHT	SEAT	NAME	DESTINATION	ORIGIN
512	7B	____	DENVER	DALLAS
....				
....				
814	8A	____	SAN JOSE	DENVER
....				
134	1C	____	HONOLULU	SAN JOSE
....				....

図 13.6 – 反復不能読み取り

図 13.6 で例えば、Dallas から Honolulu へのフライトを予約するとします。イベント・シーケンスは以下のようになります。

1. App1 が、図 13.6 の情報が含まれるカーソル (結果セットとも呼ばれます) を開きます。
2. App2 がカーソルを構成する行のうちの 1 つ (例えば、宛先が「San Jose」の行) を削除します。
3. App2 が変更をコミットします。
4. App1 がカーソルを閉じてから再び開きます。

この場合、App1 が再びカーソルを開いたときに読み取るデータは、最初にカーソルを開いたときのデータとは異なります。そのため、データ・セットを再現することができません。これが、この問題が「反復不能読み取り」と呼ばれる理由です。

13.3.4 幻像読み取り

幻像読み取りの問題は、反復不能読み取りの問題と似ていますが、後続のフェッチで取得する行数が減る代わりに増えるという点が異なります。図 13.7 は、この問題の一例を示しています。

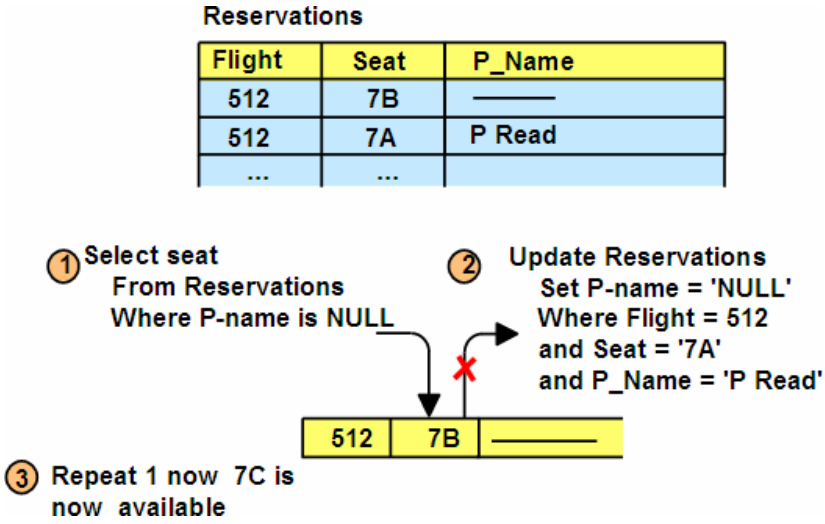


図 13.7 – 幻像読み取り

図 13.7 のイベント・シーケンスは以下のとおりです。

1. App1 がカーソルを開きます。
2. App2 が、カーソルの一部となる行をデータベースに追加します。
3. App2 が変更をコミットします。
4. App1 がカーソルを閉じてから再び開きます。

この場合、App1 が再びカーソルを開いたときに読み取るデータは、最初にカーソルを開いたときのデータとは異なり、行数が増えています。そのため、この問題は「幻像読み取り」と呼ばれます。

13.4 分離レベル

分離レベルは、ロック・ポリシーのことだと考えてください。つまり、どの分離レベルを選択するかによって、アプリケーションでのデータベース・ロックの振る舞いが変わるということです。

DB2 では以下の保護レベルそれぞれに応じてデータを分離します。:

- 非コミット読み取り (UR)
- カーソル固定 (CS)
- 読み取り固定 (RS)
- 反復可能読み取り (RR)

13.4.1 非コミット読み取り

非コミット読み取りは、ダーティー読み取りとしても知られています。これは最も低い分離レベルで、並行性は最大になります。読み取り操作では、別のアプリケーションが表のドロップまたは変更を試行しない限り、行のロックは取得されません。更新操作については、カーソル固定の分離レベルの適用時と同様に機能します。

この分離レベルでも発生する可能性のある問題:

- 非コミット読み取り
- 反復不能読み取り
- 幻像読み取り

この分離レベルで防げる問題:

- 更新喪失

13.4.2 カーソル固定

カーソル固定は、デフォルトの分離レベルです。最小必要限度のロックを行うこの分離レベルでは、基本的にカーソルの「現在」の行がロックされます。行が読み取られるだけの場合、ロックは新しい行がフェッチされるか、または作業単位が終了するまで保持されます。行の更新時には、作業単位が終了するまでロックが保持されます。

この分離レベルでも発生する可能性のある問題

- 反復不能読み取り
- 幻像読み取り

この分離レベルで防げる問題:

- 更新喪失
- 非コミット読み取り

13.4.2.1 Currently Committed

DB2 9.7 より前のバージョンでは、カーソル固定分離レベルを使用すると、書き込み側 (UPDATE 操作) が操作対象とする行には、読み取り側 (SELECT 操作) がアクセスできなくなります。これは、書き込み側が行に対して変更を行っているため、読み取り側が最終的にコミットされた値を見るには、更新が終了するまで待機しなければならないというロジックです。DB2 9.7 では新規データベースのカーソル固定分離レベルに、新しいデフォルトの振る舞いが導入されています。この新しい振る舞いは、**currently committed (CC)** セマンティクスを使用して実装されます。CC では、書き込み側が同じ行に対する読み取り側のアクセスを防ぐことはありません。これまでも、この振る舞いは分離レベルを非コミット読み取り (UR) にすることで可能でしたが、現在の振る舞いとの違いは、UR では読み取り側が**コミットされていない** 値を取得するのに対し、CC では読み取り側が**現在**

コミット済み の値を取得するという点です。現在コミット済みの値とは、書き込み操作の開始前にコミットされた値のことです。

例えば、T1 表の内容が以下のようになっているとします。

```
FIRSTNAME     LASTNAME
Raul          Chong
Jin           Xie
```

ここで、アプリケーション AppA が以下のステートメントを実行しますが、コミットはしません。

```
update T1 set lastname = 'Smith' where firstname = 'Raul'
```

続いてアプリケーション AppB が以下のステートメントを実行します。

```
select lastname from T1 where firstname = 'Raul' with CS
```

DB2 9.7 より前の DB2 では、2 番目に実行されたステートメントは停止状態となります。なぜなら、このステートメントは AppA (書き込み側) の更新ステートメントが保持する排他ロックが解放されるまで待機するためです。

一方、DB2 9.7 で currently committed を有効にしている場合 (新規データベースの場合、デフォルトで有効になります)、ステートメントは現在コミット済みの値である ***Chong*** を返します。

CS はデフォルトですが、上記のステートメントでは明確にするため「with CS」を含めていることに注意してください。この節については、本章の後のほうで説明します。

今度は AppB が以下のステートメントを試行したとします。

```
select lastname from T1 where firstname = 'Raul' with UR
```

上記では UR 分離レベルが使用されているため、結果はコミットされていない値、***Smith*** となります。

この例から、CC を使用すると、アプリケーションの並行性が向上し、書き込み側が更新中の行に、読み取り側がアクセスできるようになることがわかります。

DB2 9.7 より前のバージョンで競合の原因となっていたもう 1 つのシナリオは、読み取り側によって、書き込み側が行にアクセスできないという場合です。このシナリオが 1 つの理由となって、読み取り操作にも、共用ロック (S ロック) を確実に解放するための COMMIT ステートメントが推奨されていました。しかし CC を使用すれば読み取り側が書き込み側をブロックすることがなくなるため、この問題は解消されます。

コミットされない INSERT 操作に関しては、読み取り操作はデフォルトでスキップします。つまり、挿入が実行された行は結果セットに表示されないということです。DELETE コマンドについても、読み取り操作は該当する行を無視しますが、その振る舞いは DB2 レジストリー変数 DB2_SKIPDELETED の値に依存します。BIND および PREPARE コマンドには、他にも CC のデフォルトの振る舞いを変更可能なレジストリー変数とプロパティーがあります。

currently committed とは、現在コミット済みの情報のみを示すことを意味するため、コミットされない INSERT または DELETE 操作は無視されると覚えておいてください。

前述のとおり、新規データベースではデフォルトで CC が有効に設定されます。CC を無効にしたい場合、または DB2 9.7 より前に作成されて DB2 9.7 にアップグレードしたデータベースに対して有効にしたい場合は、データベース構成値を CUR_COMMIT に更新します。以下は、**SAMPLE** データベースで CC を無効にする例です。

```
db2 update db cfg for sample using CUR_COMMIT off
db2stop
db2start
```

13.4.3 読み取り固定

読み取り固定では、作業単位のなかでアプリケーションが取得するすべての行がロックされます。特定のカーソルに対し、この分離レベルは結果セットを構成するすべての行をロックします。例えば、10,000 行からなる表があり、クエリーによってそのうちの 10 行が返された場合、その 10 行がロックされることになります。読み取り固定では、中程度のロックを使用します。

この分離レベルでも発生する可能性のある問題:

- 幻像読み取り

この分離レベルで防げる問題:

- 更新喪失
- 非コミット読み取り
- 反復不能読み取り

13.4.4 反復可能読み取り

反復可能読み取りは、最も高い分離レベルです。このレベルでは最大限のロックを行うため、並行性は最小となります。ロックは、結果セットを作成するために処理されたすべての行で保持されます。つまり、最終的に結果セットに含まれるとは限らない行でも同じくロックされるということです。作業単位が完了するまでは、他のアプリケーションは結果セットに影響を及ぼす可能性のある行を更新、削除、挿入することはできません。反復可能読み取りは、アプリケーションが作業単位のなかで同じクエリーを実行すると、常に同じ結果が得られることを保証します。

この分離レベルでも発生する可能性のある問題:

- なし

この分離レベルで防げる問題:

- 更新喪失
- 非コミット読み取り
- 反復不能読み取り
- 幻像読み取り

13.4.5 分離レベルの比較

図 13.8 に、フェッチを行う場合の分離レベルを比較します。この図を見ると、非コミット読み取り (UR) 分離レベルではロックが使用されないことがわかります。カーソル固定 (CS) 分離レベルでは各行をフェッチするときにロックを取得しますが、そのロックは次の行をフェッチすると同時に解放されます。読み取り固定 (RS) や反復可能読み取り (RR) の分離レベルの場合、フェッチされたすべての行はロックされ、トランザクションの終了時（コミット・ポイント）までロックは解放されません。

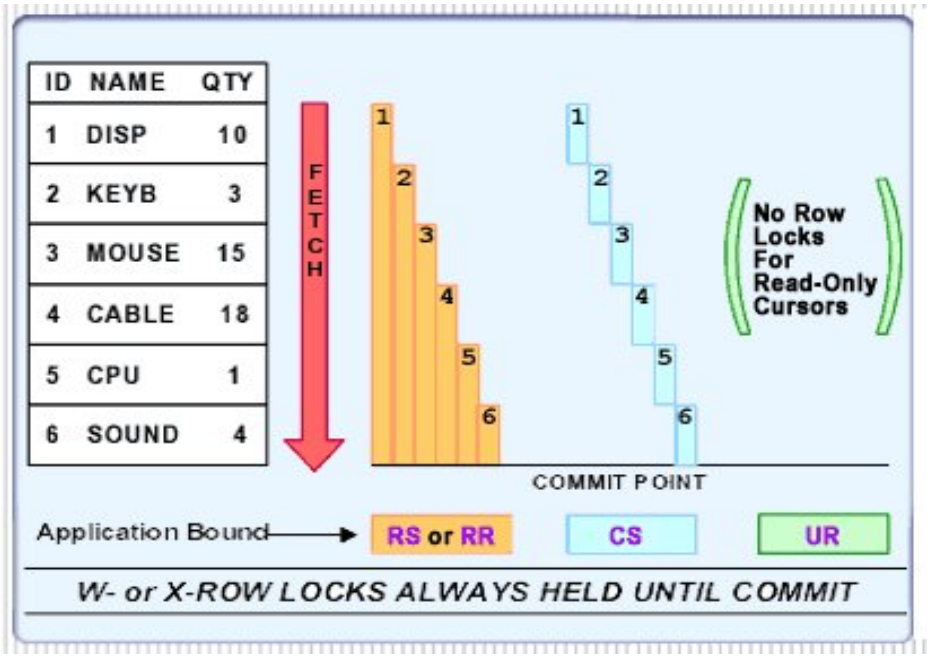


図 13.8 – フェッチの場合の分離レベルの比較

13.4.6 分離レベルの設定

分離レベルは、以下のさまざまな単位ごとに指定することができます。

- セッション (アプリケーション)
- 接続
- ステートメント

分離レベルは通常、セッション (すなわちアプリケーション) 単位で定義されます。アプリケーションに分離レベルが指定されていない場合、分離レベルはカーソル固定にデフォルト設定されます。例として、表 13.1 に .NET または JDBC プログラムに設定可能な分離レベルと、それぞれのプロパティーに対応する DB2 の分離レベルを記載します。

DB2	.NET	JDBC
非コミット読み取り (UR)	ReadUncommitted	TRANSACTION_READ_UNCOMMITTED
カーソル固定 (CS)	ReadCommitted	TRANSACTION_READ_COMMITTED
読み取り固定 (RS)	RepeatableRead	TRANSACTION_REPEATABLE_READ
反復可能読み取り (RR)	Serializable	TRANSACTION_SERIALIZABLE

表 13.1 – 分離レベル用語の対応表

分離レベルのステートメントは、以下のように WITH {isolation level} 節を使用して設定することができます。

```
SELECT ... WITH {UR | CS | RS | RR}
```

サンプル・シナリオ:

アプリケーションで、表に含まれる行数の「大まか」なカウントを取得する必要があるとします。最優先されるのはパフォーマンスです。この場合、以下の SQL ステートメントを例外とするカーソル固定の分離レベルが必要となります。

```
SELECT COUNT(*) FROM tab1 WITH UR
```

分離レベルは、組み込み SQL の場合にはバインド時に設定され、動的 SQL の場合には実行時に設定されます。

どの分離レベルを使用するかは、アプリケーション次第です。アプリケーションに上記の例と同じく正確なカウントが必要でないのであれば、UR 分離を選択します。アプリケーションが、操作対象のデータを極めて厳格に管理しなければならない場合には、RR 分離を選択してください。

new in V9.7

currently committed セマンティクスをバインド時または準備時に使用するには、以下の構文を使用します。

バインド時:

```
>--+-------------------------------------------------------------+-->
   '--CONCURRENTACCESSRESOLUTION--+--USE CURRENTLY COMMITTED--+--'
                                  '--WAIT FOR OUTCOME---------'
```

準備時:

```
concurrent-access-resolution:
|-+-USE CURRENTLY COMMITTED-+-------------------------|
  '-WAIT FOR OUTCOME--------'
```

IBM Data Server Driver for JDBC and SQLJ を使用した JDBC アプリケーションでは、concurrentAccessResolution プロパティーを使用して currently committed を有効にすることができます。

13.5 ロック・エスカレーション

DB2 が行うロックは例外なく、ある程度のメモリーを消費します。複数の行にロックを使用するよりも、表全体で 1 つのロックを使用したほうが適切であると、オプティマイザーが判断すると、ロック・エスカレーションが発生します。図 13.9 に、その場合の状態を示します。

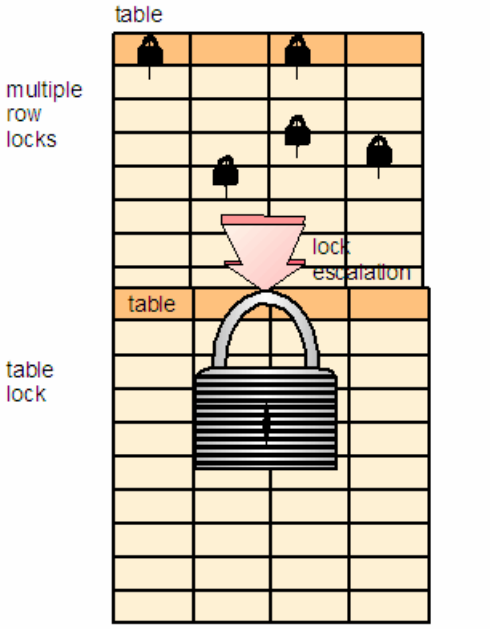


図 13.9 – ロック・エスカレーション

ロック・エスカレーションに関係する主なデータベース構成パラメーターは、以下の 2 つです。

- **LOCKLIST** – 接続されたすべてのアプリケーションのロックを管理するために確保するメモリーの量 (4K のページ数)。
- **MAXLOCKS** – ロック・リスト全体のなかで、単一のアプリケーションが使用できる最大パーセンテージ。

この 2 つのパラメーターはどちらもデフォルトで AUTOMATIC に設定されます。これは、セルフチューニング・メモリー・マネージャー (STMM) によってサイズが変更されることを意味します。STMM を有効に設定せずに、値を自分で設定すると、設定した値がロック・エスカレーションの発生するタイミングに影響することになります。したがって、例えば LOCKLIST が 200K で、MAXLOCKS が 22% の場合には、単一のアプリケーションに 44K を上回るロック・メモリーが必要になると、ロック・エスカレーションが発生します (200K * 22% = 44K)。デフォルト設定でロック・エスカレーションが頻繁に発生する場合には、LOCKLIST と MAXLOCKS の値を増やしてください。ロック・エスカレーションが発生すると並行性が減少するため、パフォーマンスが低下する可能性があります。ロック・エスカレーションが発生しているかどうかを判断するには、DB2 診断ログ・ファイル (`db2diag.log`) を使用します。このファイルについての詳細は、付録 A を参照してください。

13.6 ロックのモニター

ロックの使用状況は、DB2 アプリケーションのロック・スナップショットを使用してモニターすることができます。ロックに対してスナップショットをオンにするには、以下のコマンドを実行します。

```
UPDATE MONITOR SWITCHES USING LOCK ON
```

スイッチがオンになると、モニター情報の収集が開始されます。特定の時点でのロックに関するレポートを取得するには、以下のコマンドを実行します。

```
GET SNAPSHOT FOR LOCKS FOR APPLICATION AGENTID <handle>
```

図 13.9 は、アプリケーションのロック・スナップショットの出力例です。

```
           Application Lock Snapshot

Snapshot timestamp                         = 11-05-2002
00:09:08.672586

Application handle                         = 9
Application ID                             = *LOCAL.DB2.00B9C5050843
Sequence number                            = 0001
Application name                           = db2bp.exe
Authorization ID                           = ADMINISTRATOR
Application status                         = UOW Waiting
Status change time                         = Not Collected
Application code page                      = 1252
Locks held                                 = 4
Total wait time (ms)                       = 0

List Of Locks
 Lock Name                   = 0x05000700048001000000000052
 Lock Attributes             = 0x00000000
 Release Flags               = 0x40000000
 Lock Count                  = 255
 Hold Count                  = 0
 Lock Object Name            = 98308
 Object Type                 = Row
 Tablespace Name             = TEST4K
       Table Schema                = ADMINISTRATOR
```

図 13.9 – アプリケーションのロック・スナップショット

new in V9.7

注:

DB2 9.7 では現在、データベース・モニター機能をシステム・モニターおよびスナップショット技術から切り離し、ワークロード管理表の機能や IBM Data Studio ツールと同じく、SQL を内部メモリーにアクセスさせるための取り組みが行われているところです。詳細については DB2 の公式資料を参照してください。

13.7 ロック待機

例えば複数のアプリケーションが同じオブジェクトに対して操作を実行する必要がある場合、そのうち一方のアプリケーションは必要なロックを取得するまで待機しなければならない可能性があります。デフォルトでは、アプリケーションは無期限に待機します。アプリケーションがロックを待機する時間は、データベース構成パラメーター LOCKTIMEOUT で制御します。このパラメーターのデフォルト値は、-1 (無期限待機) です。

CURRENT LOCK TIMEOUT レジスターを使用すると、特定の接続に対してロック待機を設定することができます。このレジスターのデフォルト値は、LOCKTIMEOUT です。この値を変更するには SET LOCK TIMEOUT ステートメントを使用します。このレジスターの値を特定の接続に対して設定すると、トランザクション全体でその設定が維持されます。

以下はステートメントの一例です。

```
SET LOCK TIMEOUT=WAIT n
```

13.8 デッドロックの原因と検出

同じデータベースに接続された複数のアプリケーションが無期限のリソース待機状態になると、デッドロックが発生します。それぞれのアプリケーションが互いに他のアプリケーションが必要とするリソースを保持し続けるため、この待機状態が解消されることはありません。ほとんどのデッドロックは、アプリケーション設計の問題です。図 13.10 は、デッドロックを説明しています。

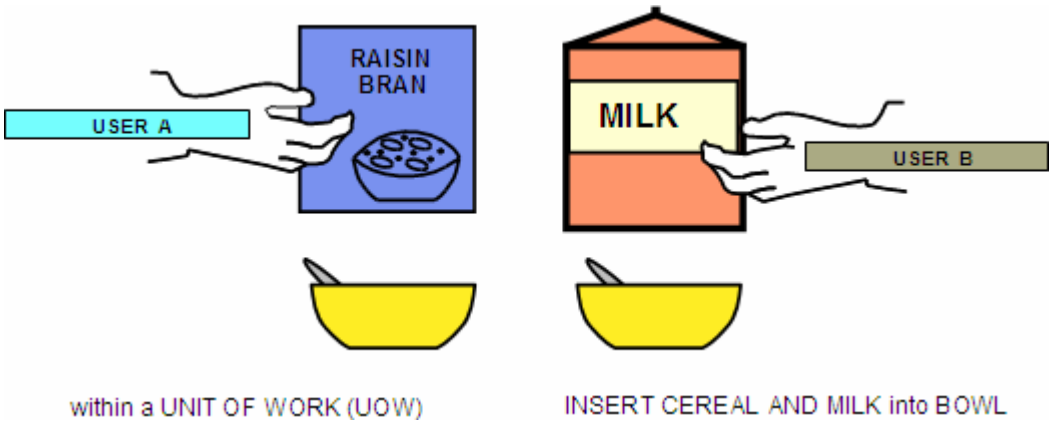


図 13.10 – デッドロックのシナリオ

図 13.10 では、ユーザー A がシリアルを持っています。ユーザー A はミルクを手に入れるまではシリアルを手放しません。一方ユーザー B はミルクを持っており、シリアルを手に入れてからでないと、そのミルクを手放すことはありません。このため、デッドロック状態が発生します。

new in V9.7

DB2 9.7 では currently committed を使用することにより、アプリケーションは現在コミット済みの値にアクセスし、他のアプリケーションがロックを解放するまで待機する必要がなくなります。そのため、デッドロックの発生が大幅に減少します。

DB2 でのデッドロック状態をシミュレートするには、以下のステップに従ってください。

1. Currently Committed を OFF にします。

```
db2 update db cfg for sample using cur_commit off
db2stop force
db2start
```

2. 同じデータベースに接続する 2 つの異なるアプリケーションを表す 2 つの DB2 「コマンド・ウィンドウ」を開きます (ここでは、それぞれを「CLP1」、「CLP2」と呼びます)。

3. CLP1 から以下のコマンドを実行します。

```
db2 connect to sample
db2 +c update employee set firstnme = 'Mary' where empno = '000050'
```

上記ではまず、**SAMPLE** データベースに接続し、次に employee 表の「empno = 50000」の行に対して更新ステートメントを実行しています。ステートメントに含まれる「+c」は、DB2 「コマンド・ウィンドウ」に自動的にステートメントをコミットさせないように指定するためのオプションです。このオプションは、ロックを保持するために意図的に設定しています。

4. CLP2 から以下のコマンドを実行します。

```
db2 connect to sample
db2 +c update employee set firstnme = 'Tom' where empno = '000030'
```

2 つ目のアプリケーションを表す CLP2 ウィンドウでも、同じく SAMPLE データベースに接続していますが、この場合の更新対象は、employee 表の別の行です。

5. CLP1 から以下のコマンドを実行します。

```
db2 +c select firstnme from employee where empno = '000030'
```

Enter キーを押して上記の SELECT ステートメントを実行すると、SELECT ステートメントが停止しているように見えるかもしれません。しかし実際に停止しているわけではなく、CLP2 がステップ 4 で取得したこの行の排他ロックの解放を待機しているためです。この時点で LOCKTIMEOUT がデフォルト値の -1 に設定されているとしたら、CLP1 アプリケーションは永遠に待機状態となります。

6. CLP2 から以下のコマンドを実行します。

```
db2 +c select firstnme from employee where empno = '000050'
```

上記の SELECT ステートメントを実行することで、デッドロックの状態がもたらされます。CLP1 がステップ 2 で取得したこの行の排他ロックが解放されるのを待機していることから、この SELECT ステートメントも同じく停止しているように見えるはずです。

上記のデッドロックのシナリオでは、DB2 がデータベース構成パラメーター DLCHKTIME をチェックすることになります。このパラメーターが設定するのは、デッドロックをチェックする間隔です。例えば、このパラメーターが 10 秒に設定されている場合、DB2 は 10 秒ごとにデッドロックが発生しているかどうかをチェックします。デッドロックが実際に発生していると、DB2 は内部アルゴリズムを使用し、2 つのトランザクションのうちのどちらをロールバックし、どちらを続行させるかを決定します。

多数のデッドロックが発生する場合には、既存のトランザクションを再検討して、作成し直せるかどうかを調べてください。

13.9 並行性およびロックのベスト・プラクティス

並行性を可能な限り最適な状態にするためには、以下に記載するヒントに従ってください。

1. アプリケーション・ロジックで許容される場合には必ず、currently committed (CC) を有効にします。
2. トランザクションはできる限り短くしてください。それには、アプリケーションのロジックが許す限り頻繁に COMMIT ステートメントを実行します (読み取り専用トランザクションの場合でも同じです)。
3. トランザクション情報は、必要な場合にのみログに記録します。
4. 素早くデータをパージ (除去) します。そのためには、以下のコマンドを実行するか、

```
ALTER TABLE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
```

DB2 9.7 では、以下の TRUNCATE コマンドを使用することもできます。

```
TRUNCATE <table name>
```

5. 以下のように、バッチ/グループごとにデータ変更を実行します。

```
DELETE FROM (
    SELECT * FROM tedwas.t1 WHERE c1 = … FETCH FIRST 3000 ROWS ONLY)
```

6. DB2 データ移動ツールの並行性フィーチャーを使用します。
7. データベース・レベルで LOCKTIMEOUT パラメーターを設定します (推奨される値の範囲は 30 秒から 120 秒です)。デフォルトの -1 のままにしないでください。セッション・ベースのロック・タイムアウトを使用することも可能です。

8. 必要以上のデータを取得しないでください。例えば、SELECT ステートメントで FETCH FIRST n ROWS ONLY 節を使用します。

注:

並行性およびロックのベスト・プラクティスについての詳細は、
http://www.ibm.com/developerworks/data/bestpractices/ で公開されているベスト・プラクティスに関する資料を参照してください。

13.10 まとめ

この章ではトランザクション制御、同時ユーザー・アクセス、およびロックのレベルという観点から、データ保全性の維持について説明しました。並行性にはさまざまなレベルがあり、そのすべてのレベルに、データへのアクセス方法および管理方法に影響を与える可能性のある問題があります。

この章ではさらに、これらの問題に対処する上で分離レベルの設定が果たす役割を詳しく検討し、分離レベルを操作することによって、アプリケーションとデータに必要な最大限の柔軟性を実現する方法について説明しました。

次に、ロック・エスカレーション、ロック待機、ロックのモニター、そしてデータベースで発生するデッドロックの原因、検出、対処方法を説明しました。

そして最後に、可能な限り最適な並行性を実現するためのベスト・プラクティスを紹介しました。

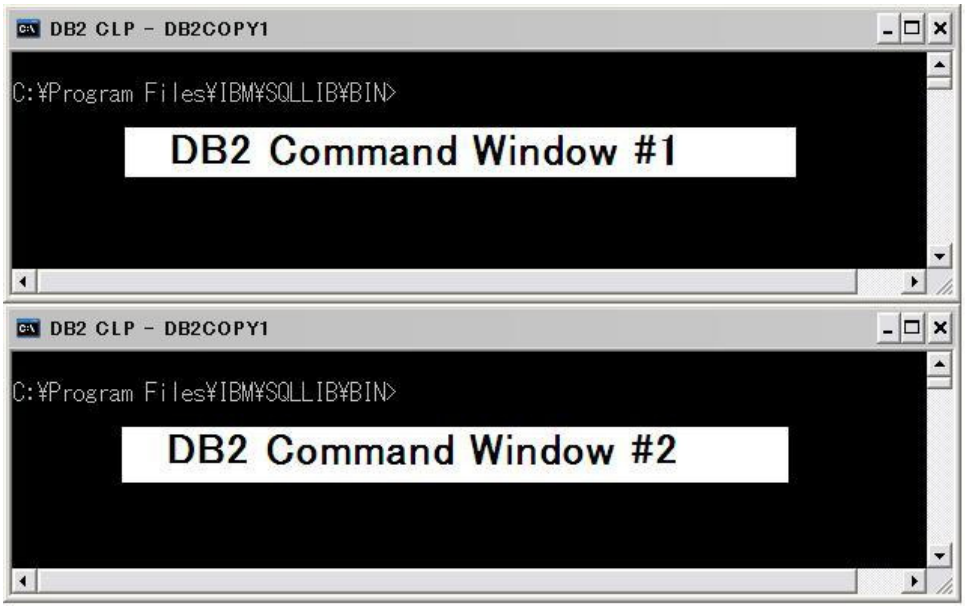
13.11 演習　並行性及びロックのテスト

以下の演習では、この章で説明した並行性およびロックの概念を DB2 コマンド・ウィンドウを使用して実践します。このツールがデフォルトで使用するロックの分離レベルは CS です。SQL ステートメントの実行後、コマンド・ウィンドウは自動的にコミットを実行します (自動コミットとしても知られています)。この演習では説明のために、`+c` フラグを使用して自動コミットを無効に設定し、SQL ステートメントの後に `WITH <isolation level>` 節を使用してデフォルトの CS 分離レベルの指定を変更します。

パート 1: 幻像読み取りの問題および RR 分離のテスト

手順:

1. 以下の図に示すように、2 つの DB2 コマンド・ウィンドウを開きます。上の方のウィンドウを「DB2 コマンド・ウィンドウ #1」と呼び、下の方のウィンドウを「DB2 コマンド・ウィンドウ #2」と呼ぶことにします。



2. DB2 コマンド・ウィンドウ #1 から以下を実行します。

```
db2 connect to sample
db2 +c select * from staff
```

この結果、35 レコードが返されるはずです。

3. DB2 コマンド・ウィンドウ #2 から以下を実行します。

```
db2 connect to sample
db2 +c insert into staff (id,name) values (400, 'test')
```

4. DB2 コマンド・ウィンドウ #1 から以下を実行します。

```
db2 +c select * from staff
```

これでもやはり、35 レコードが返されるはずです。

5. DB2 コマンド・ウィンドウ #2 から以下を実行します。

```
db2 commit
```

6. DB2 コマンド・ウィンドウ #1 から以下を実行します。

```
db2 +c select * from staff
```

今度は、36 レコードが返されるはずです。

DB2 コマンド・ウィンドウ #1 のアプリケーションがカーソルまたは結果セットを開くと (`select * from staff`)、35 レコードが返されました。DB2「コマンド・ウィンドウ」

#2 のアプリケーションが新しいレコードを挿入した後でも (ただしコミットはしていません)、DB2 コマンド・ウィンドウ #1 のアプリケーションが同じトランザクション (このウィンドウでは **commit** ステートメントを実行していないため) のなかで同じカーソルを開くと、やはりレコード数は 35 のままです。

次に、DB2 コマンド・ウィンドウ #2 のアプリケーションが挿入をコミットします。ここで DB2 コマンド・ウィンドウ #1 のアプリケーションから改めてカーソルを開くと、結果セットは 1 行多い (幻像読み取り) 36 レコードを返します。この例は、同じトランザクション内で同じカーソルを開くと、前より多くの行が返されるという幻像読み取りの問題を説明するものです。ここで使用している分離レベルは CS であり、この章で説明したように、分離レベルが CS の場合、幻像読み取りを防ぐことはできません。

7. 次のステップに進む前に、挿入されたレコードをクリーンアップします。

   DB2 コマンド・ウィンドウ #1 から以下を実行します。

   ```
   db2 rollback
   ```

   DB2 コマンド・ウィンドウ #2 からは、以下を実行します。

   ```
   db2 delete from staff where id = 400
   db2 select * from staff
   ```

   この結果、再び 35 レコードが返されるようになります。.

8. ここで、RR 分離レベルによってこの幻像読み取りの問題を防ぐことができるかどうかを調べます。

   DB2 コマンド・ウィンドウ #1 から以下を実行します。

   ```
   db2 connect to sample
   db2 +c select * from staff with RR
   ```

   この結果、35 レコードが返されます。

   DB2 コマンド・ウィンドウ #2 から以下を実行します。

   ```
   db2 connect to sample
   db2 +c insert into staff (id,name) values (400, 'test')
   ```

   このステートメントは停止状態になりますが、これは正常です。

   WITH RR 節が DB2「コマンド・ウィンドウ」#1 の SELECT ステートメントに追加されたことにより、結果セットが変更されることになる行での INSERT 操作を RR 分離レベルが防いでいるというわけです。このように、RR 分離レベルでは幻像読み取りの問題を防ぐことができます。

9. 演習のパート 2 に進む前にクリーンアップをしておきます。

   DB2 コマンド・ウィンドウ #2 から以下を実行します。

   ```
   Ctrl-C (中断するため)
   ```
   ウィンドウを閉じます。

   DB2 コマンド・ウィンドウ #1 から以下を実行します。

   ```
   db2 rollback
   ```
   ウィンドウを閉じます。

パート 2: Currently Committed (CC) と非コミット読み取り (UR) レベルのテスト

手順 1: Currently Committed を有効にせずに分離 CS の振る舞いを分析する

1. DB2 コマンド・ウィンドウを開き、以下のステートメントを実行します。

   ```
   db2 connect to sample
   db2 select * from staff
   ```

   STAFF 表の内容を調べて、値が「10」となっている ***ID*** を探します。これに対応する ***NAME*** 列の値は ***Sanders*** です。ウィンドウを閉じます。

2. currently committed は、新規データベースではデフォルトで有効に設定されます。sample データベースで実際に有効に設定されていることを確認します。

   ```
   db2 get db cfg for sample
   ```

   出力の終りのほうで、以下の行を探します。
   ```
   Currently Committed                      (CUR_COMMIT) = ON
   ```

   値が ON となっている場合は、CS 分離レベルの振る舞いを分析するために、まず値を OFF に変更して DB2 9.7 より前の状態にします。
   ```
   db2 update db cfg for sample using CUR_COMMIT off
   db2 force applications all
   ```
   `force` オプションを追加すると、確実に接続が存在しない状態になります (CUR_COMMIT の変更は、次の接続で有効になります)。

   CUR_COMMIT が無効になっていることを確認します。以下のように設定されていなければなりません。
   ```
   Currently Committed                      (CUR_COMMIT) = DISABLED
   ```

3. パート 1 と同じように 2 つの DB2 コマンド・ウィンドウを開き、一方のウィンドウをもう一方のウィンドウの上に配置します。これから、currently committed が有効でない状態で更新操作 (書き込み側) と選択操作 (読み取り側) が同じ行を対象とする場合に CS 分離レベ

ルがどのように機能するかを見ていきます。ステートメントの後に「WITH CS」を指定する必要はないことに注意してください (これがデフォルトであるためです)。

DB2 コマンド・ウィンドウ #1 (書き込み側) から以下を実行します。

```
db2 connect to sample
db2 +c update staff set name = 'Chong' where id = 10
```

DB2 コマンド・ウィンドウ #2 (書き込み側) から以下を実行します。

```
db2 connect to sample
db2 +c select * from staff
```

この SELECT は、DB2 コマンド・ウィンドウ #1 アプリケーションが排他ロック (X ロック) を解放するのを待機するために停止状態になります。このように、DB2 9.7 より前の CS のデフォルトでの振る舞いでは、並行性のレベルが低くなります。

DB2 コマンド・ウィンドウ #2 から以下を実行します。

```
Ctrl-C (中断するため)
```

ウィンドウを閉じます。

DB2 コマンド・ウィンドウ #1 から以下を実行します。

```
db2 rollback
```

ウィンドウを閉じます。

手順 2: Currently Committed を有効にして分離 CS の振る舞いを分析する

1. Currently Committed を ON にします。

   DB2 コマンド・ウィンドウを開き、以下のステートメントを実行します。

   ```
   db2 update db cfg for sample using CUR_COMMIT on
   db2 force applications all
   ```

   ウィンドウを閉じます。

2. パート 1 のときと同様に 2 つの DB2 コマンド・ウィンドウを上下に並べて開きます。

   DB2 コマンド・ウィンドウ #1 から以下を実行します。

   ```
   db2 connect to sample
   db2 +c update staff set name = 'Chong' where id = 10
   ```

   DB2 コマンド・ウィンドウ #2 から以下を実行します。

   ```
   db2 connect to sample
   ```

```
db2 +c select * from staff
```

この SELECT ステートメントは正常に実行され、停止状態にはなりません。表示される値は現在コミット済みの値である ***Sanders*** です。

DB2 コマンド・ウィンドウ #1 から以下を実行します。

```
db2 rollback
```

ウィンドウを閉じます。

DB2 コマンド・ウィンドウ #2 から以下を実行します。

```
db2 rollback
```

ウィンドウを閉じます。

手順 3: 分離 UR の振る舞いの分析

1. パート 1 のときと同様に 2 つの DB2 コマンド・ウィンドウを上下に並べて開き、以下を実行します。

   DB2 コマンド・ウィンドウ #1 から以下を実行します。

   ```
   db2 connect to sample
   db2 +c update staff set name = 'Chong' where id = 10
   ```

   DB2 コマンド・ウィンドウ #2 から以下を実行します。

   ```
   db2 connect to sample
   db2 +c select * from staff with UR
   ```

   この SELECT ステートメントは正常に実行されますが、表示される値がコミットされてない ***Chong*** であることに注意してください。

   DB2 コマンド・ウィンドウ #1 から以下を実行します。

   ```
   db2 rollback
   ```

   ウィンドウを閉じます。

   DB2 コマンド・ウィンドウ #2 から以下を実行します。

   ```
   db2 rollback
   ```

   ウィンドウを閉じます。

パート III – DB2 の学習:
アプリケーション開発

14

第 14 章 – DB2 アプリケーション開発の概要

IBM DB2 は、リレーショナル・データと XML データ両方の管理に利用できる強力なデータ・サーバー・ソフトウェアです。DB2 はデータベース管理者だけなく、データベース開発者にも柔軟性をもたらします。プログラムの開発にどの言語を使用しているかに関わらず、DB2 は、アプリケーションの一部としてデータベースを操作するために必要なドライバー、アダプター、そして拡張機能を提供します。さらに DB2 Express-C を使用すれば、データベースにコストを掛けずにアプリケーションを開発することができます。このエディションには、データベースのサイズに関する制限もなく、DB2 の他のバージョンと同じレベルのプログラミング言語のサポートが備わっています。DB2 Express-C で開発したアプリケーションは、何の変更も加えることなく、他のどの DB2 エディションでも実行することができます。

注:

このセクションで説明するのは、DB2 アプリケーション開発の概要のみです。現在、**Community Book Series** の一部として作成が進められている 25 を超える無料のオンライン・ブックのなかに、DB2 アプリケーション開発を専門に取り上げた 1 冊があります。このシリーズでは、DB2 以外にも Java、PHP、Ruby on Rails、Python、Perl、Web 2.0、SOA、Eclipse、オープンソース開発、クラウド・コンピューティングなどをテーマにしたオンライン・ブック、さらに pureQuery、Data Studio、InfoSphere Data Architect などの IBM 技術を詳しく説明しているオンライン・ブックを揃えています。これらのオンライン・ブックは、2009 年 10 月に公開される予定です。

14.1 DB2 アプリケーション開発: 全体像

DB2 がもたらす柔軟性により、サーバー・サイドではデータベース開発者がストアード・プロシージャーやユーザー定義関数などの開発用フィーチャーを利用し、クライアント・サイドではアプリケーション開発者が任意のプログラミング言語を使ってアプリケーションを開発することができます。この柔軟性を図 14.1 に示します。

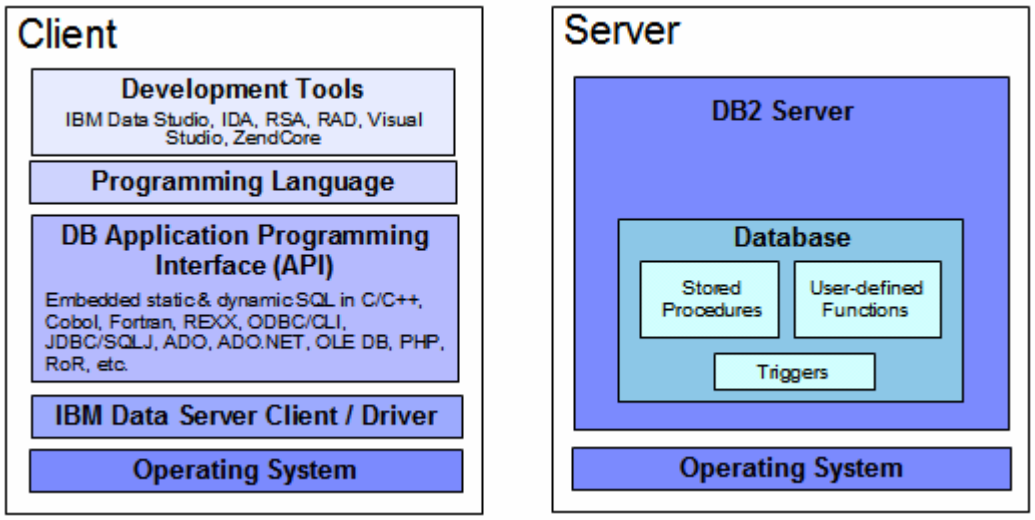


図 14.1 – データベース開発者とアプリケーション開発者両方のための DB2

図 14.1 の左側は、アプリケーション・プログラマーがプログラムを開発して実行するクライアント・マシンを表します。このクライアント・マシンには、オペレーティング・システムの他、開発するアプリケーションのタイプによっては IBM Data Server Client がインストールされる場合もあります。IBM Data Server Client は、JDBC ドライバーや ODBC/CLI ドライバーなどの必要な接続ドライバーが組み込まれたクライアントです。これらのドライバーは、IBM DB2 Express-C の Web サイト (www.ibm.com/db2/express) にアクセスして別途ダウンロードすることもできます。

アプリケーションの開発は、IBM Data Studio や InfoSphere Data Architect (IDA)、Rational Software Architect (RSA)、Rational Application Developer (RAD) などのプログラミング・ツールを使用して、お望みのプログラミング言語を選択して行うことができます。IBM Data Server Client にはこれらのプログラミング言語に対応した API　ライブラリーも組み込まれています。そのため DB2 サーバーに接続すると、プログラムのすべての命令が、これらの API　によって DB2 が理解する SQL または XQuery ステートメントに適切に変換されます。表 1.1 に、上述のツールについて簡単に説明します。

ツール名	説明
IBM Data Studio	ユーザーがデータ・サーバーを管理し、ストアード・プロシージャー、関数、Data Web サービスを開発できるようにする Eclipse ベースのツール。IBM Data Studio については、本書の前のセクションを参照
InfoSphere Data Architect (IDA)	データのモデリング・ツール。データベースの論理設計および物理設計を支援する
Rational Software Architect (RSA)	UML 図の作成を支援する Eclipse ベースのソフトウェア・エンジニアリング用ツール。

Rational Application Developer (RAD)	ソフトウェア開発者が迅速にアプリケーションを開発できるようにする Eclipse ベースのアプリケーション開発ツール
Visual Studio	Windows プラットフォームで Microsoft の技術を利用してアプリケーションを開発するための IDE
ZendCore	無料の PHP アプリケーション開発用 IDE (旧称 ZendCore for IBM)

表 14.1 – DB2 でのアプリケーション開発を支援するツール

図 14.1 の右側に示されているのは、1 つのデータベースが含まれる DB2 サーバーです。このデータベースには、ストアード・プロシージャー、ユーザー定義関数、トリガーが保管されます。これらのオブジェクトについては、以降のセクションで詳しく説明します。

14.2 サーバー・サイドの開発

DB2 でのサーバー・サイドの開発とは、アプリケーション・オブジェクトを開発し、DB2 データベースに保管することを意味します。このセクションでは、以下のアプリケーション・オブジェクトについて簡単に説明します。

- ストアード・プロシージャー
- ユーザー定義関数 (UDF = User-Defined Function)
- トリガー

14.2.1 ストアード・プロシージャー

ストアード・プロシージャー は、SQL ステートメントとビジネス・ロジックをカプセル化することができるデータベース・アプリケーション・オブジェクトです。アプリケーション・ロジックの部分をデータベースに保持することで、アプリケーションとデータベースとの間のネットワーク・トラフィックの量が減り、その結果パフォーマンスが向上します。さらに、ストアード・プロシージャーは、他のアプリケーションが同じストアード・プロシージャーを再利用できるようにコードを一ヶ所にまとめて保管する場所にもなります。ストアード・プロシージャーを呼び出すには、**CALL** ステートメントを使用します。DB2 でストアード・プロシージャーを開発するには、SQL PL、Java、C/C++、OLE、COBOL といった複数の言語を使用することができます。以下に示すのは、DB2 の「コマンド・ウィンドウ」または Linux シェルから SQL PL でストアード・プロシージャーを作成して呼び出す単純な例です。

```
db2 create procedure P1 begin end
db2 call P1
```

上記の procedure P1 は空のストアード・プロシージャーで、何も実行していません。この例から、ストアード・プロシージャーを簡単に作成できることがわかるはずです。複雑なロジックを使用してストアード・プロシージャーを開発する場合は、デバッガーが組み込まれた IBM Data Studio を使用することをお薦めします。

14.2.2 ユーザー定義関数

ユーザー定義関数 (UDF = User-Defined Function) とは、ユーザーが独自のロジックで SQL 言語を拡張できるデータベース・アプリケーション・オブジェクトのことです。ユーザー定義関数は必ず 1 つ以上の値を返しますが、通常は関数に組み込まれたビジネス・ロジックの結果を返します。関数を呼び出すには、その関数を SQL ステートメント内に含めるか、あるいは **values** 関数を使用します。DB2 で UDF を開発するには、SQL PL、Java、C/C++、OLE、CLR などの複数の言語を使用することができます。

以下に示すのは、DB2 の「コマンド・ウィンドウ」または Linux シェルから SQL PL で UDF を作成して呼び出す単純な例です。

```
db2 create function F1() returns integer begin return 1000; end
db2 values F1
```

上記の例で、function F1 は整数値 1000 を返す関数です。この関数を呼び出すには **VALUES** ステートメントを使用することができます。ストアード・プロシージャーの場合と同じく、関数を作成するには IBM Data Studio を使用することをお薦めします。

14.2.3 トリガー

トリガー とは、表またはビューで自動的に操作を実行するオブジェクトのことです。オブジェクトにトリガーが定義されている場合、そのオブジェクトでアクションを起動すると、定義されたトリガーが起動されます。一般に、トリガーはアプリケーション・オブジェクトであるとはみなされないため、データベース開発者がトリガーをコーディングすることは通常ありません。しかし、データベース管理者はトリガーをコーディングするため、このセクションにはトリガーの項目を含めました。以下に示すのは、トリガーの一例です。

```
create trigger myvalidate no cascade before insert on T1
      referencing NEW as N
      for each row
      begin atomic
         set (N.myxmlcol) = XMLVALIDATE(N.myxmlcol
            according to xmlschema id myxmlschema);
      end
```

上記のトリガーは、表 T1 で INSERT 操作が実行される前に起動されます。トリガーは値 (XML 文書) を挿入しますが、指定のスキーマでこの XML 文書の妥当性検査を行うために、XMLVALIDATE 関数を呼び出します。XML と XML スキーマについては、第 15 章「*DB2 pureXML*」で詳しく説明します。

14.3 クライアント・サイドの開発

その言葉が示すように、クライアント・サイドの開発では、アプリケーション開発者がクライアント上のプログラムをコーディングし、それから DB2 に用意されたアプリケーション・プログラム・インターフェース (API = Application Program Interface) を使用して DB2 データベースに接続し、アクセスを行います。このセクションで取り上げる内容は以下のとおりです。

- 組み込み SQL
- 静的 SQL と動的 SQL の違い
- CLI と ODBC
- JDBC、SQLJ、および pureQuery
- OLE DB
- ADO.NET
- PHP
- Ruby on Rails
- Perl
- Python

14.3.1 組み込み SQL

組み込み SQL アプリケーション とは、ホスト言語 (C、C++、COBOL など) に SQL を組み込んだアプリケーションのことです。次のセクションで説明するように、組み込み SQL アプリケーションには静的 SQL または動的 SQL を組み込むことができます。図 14.2 に、組み込み SQL アプリケーションがどのように構築されるかを示します。

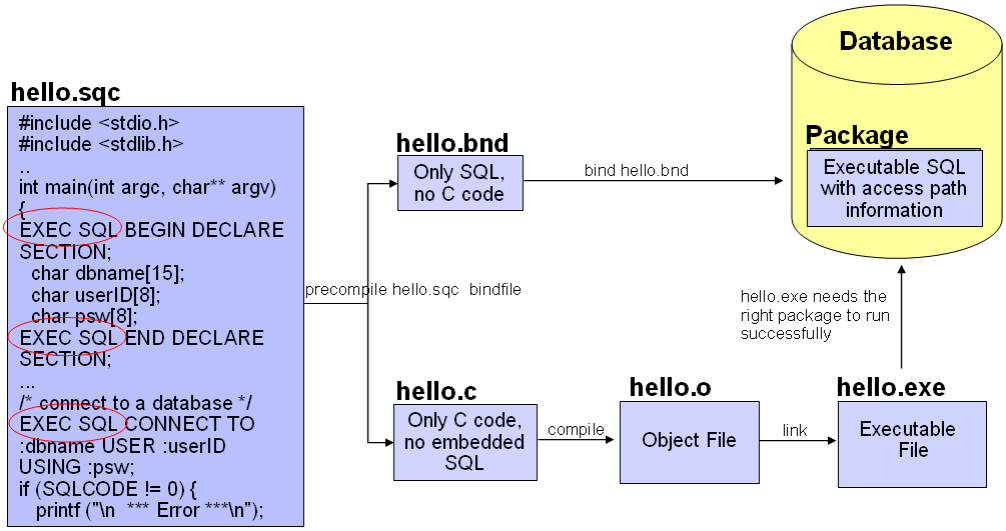


図 14.2 – 組み込み SQL アプリケーションの構築方法

この図では、hello.sqc という C プログラムに組み込み SQL が含まれています。C 言語対応の組み込み SQL API は、EXEC SQL (図 14.2 で強調表示) を使用して、プリコンパイル・プロセスが組み込み SQL ステートメントと実際の C コードを区別できるようにします。また、hello.sqc のリストに示されているコロンで始まる変数、***:dbname***、***:userID***、***:psw*** にも注目してください。これらの変数は、ホスト変数と呼ばれます。ホスト変数はホスト言語からの変数で、組み込み SQL ステートメントのなかで参照されます。

bindfile オプションを指定して **precompile** コマンド (別名 **prep** コマンド) を実行すると、2 つのファイルが生成されます。1 つは、SQL ステートメントだけが含まれる hello.bnd バインド・ファイル、そしてもう 1 つは C コードだけが含まれる hello.c ファイルです。**bind** コマンドを使用してバインド・ファイルをコンパイルすることにより、データベースに保管する**パッケージ**を取得することができます。パッケージには、コンパイル済み/実行可能 SQL と、DB2 がデータを取得するために辿るアクセス・パスが組み込まれます。**bind** コマンドを実行するには、データベースに接続されていなければなりません。一方、図の下のほうに示されている hello.c ファイルは、通常の C プログラムと同じようにコンパイルされてリンクされます。その結果生成される hello.exe 実行可能ファイルは、データベースに保管されたパッケージと一致していなければ、正常に実行されません。

14.3.2 静的 SQL と動的 SQL の違い

静的 SQL ステートメントとは、プリコンパイル時に SQL 構造が完全に既知であるステートメントのことです。以下はその一例です。

```
SELECT lastname, salary FROM employee
```

上記の例では、ステートメントで参照されている列名 (**lastname**、**salary**) と表の名前 (**employee**) は、プリコンパイルの時点で完全にわかっています。以下の例も、同じく静的 SQL ステートメントです。

```
SELECT lastname, salary FROM employee WHERE firstnme = :fname
```

この 2 番目の例では、ホスト変数 ***:fname*** を組み込み SQL ステートメントの一部として使用しています。ホスト変数の値は実行時まで不明ですが、そのデータ型はプログラムから判断することができます。また、その他すべてのオブジェクト (列名、表の名前) もすべて事前にわかっていいます。DB2 はこれらのホスト変数に予測する値を使用して事前にアクセス・プランを計算することから、この場合も同じく静的 SQL であるとみなされます。

静的に実行される SQL ステートメントは、アプリケーションを実行する前にプリコンパイル、バインド、およびコンパイルしてください。静的 SQL は、統計が大幅に変わることのないデータベースで使用するのに最適です。次は、以下の例を見てください。

```
SELECT ?, ? FROM ?
```

この例では、ステートメントで参照される列と表の名前は、実行時までわかりません。したがって、アクセス・プランは実行時になって初めて、その時点で使用可能な統計を使用して計算されます。このようなタイプのステートメントは、**動的 SQL** ステートメントとみなされます。

一部のプログラミング API (JDBC や ODBC など) は、SQL ステートメントに既知のオブジェクトが含まれるかどうかに関わらず、常に動的 SQL を使用します。例えば、**SELECT lastname, salary FROM employee** というステートメントでは、列と表のすべての名前は事前に既知となっています。しかし、このようなステートメントでも、JDBC や ODBC を使用する場合にはプリコンパイルしません。ステートメントのアクセス・プランは常に実行時に計算されます。

一般に、SQL ステートメントを動的なステートメントとして処理するには以下の 2 つのステートメントを使用します。

- **PREPARE**: このステートメントは、SQL ステートメントを作成またはコンパイルして、データを取得するために使用するアクセス・プランを計算します。
- **EXECUTE**: このステートメントは SQL を実行します。

あるいは、**EXECUTE IMMEDIATELY** という 1 つのステートメントで **PREPARE** と **EXECUTE** の両方を実行することもできます。

リスト 14.1 に、PREPARE と EXECUTE が含まれる C 言語への組み込み動的 SQL ステートメントの例を記載します。

```
strcpy(hVStmtDyn, "SELECT name FROM emp WHERE dept = ?");
PREPARE StmtDyn FROM :hVStmtDyn;
EXECUTE StmtDyn USING 1;
EXECUTE StmtDyn USING 2;
```

リスト 14.1 – PREPARE と EXECUTE を使用した、C 言語への組み込み動的 SQL ステートメント

リスト 14.2 の例は、リスト 14.1 の実行内容と同じですが、ここでは EXECUTE IMMEDIATELY ステートメントを使用しています。

```
EXECUTE IMMEDIATELY SELECT name from EMP where dept = 1
EXECUTE IMMEDIATELY SELECT name from EMP where dept = 2
```

リスト 14.2 – EXECUTE IMMEDIATELY を使用した、C 言語への組み込み動的 SQL ステートメント

PHP や Ruby on Rails をはじめ、SQL を動的に実行する多くの動的プログラミング言語では、プログラマーは以下のように、SQL ステートメントは同じで、フィールド値だけが異なるコードを作成する傾向があります。

```
SELECT lastname, salary FROM employee where firstnme = 'Raul'

SELECT lastname, salary FROM employee where firstnme = 'Jin'
```

...

上記の 2 つのステートメントは、**firstnme** 列の値を除けば、まったく同じです。DB2 は、この 2 つの動的 SQL ステートメントを別個のものとみなすため、実行時にそれぞれのステートメントを単独で作成して実行します。同じステートメントを何度も作成すると、そのオーバーヘッドによってパフォーマンスが劣化しかねません。そのため、DB2 9.7 より前のバージョンで推奨されていたのは、以下のようにステートメントをコーディングすることです。

```
SELECT lastname, salary FROM employee where firstnme = ?
```

上記のステートメントで使用している疑問符 (?) は、**パラメーター・マーカー** として知られています。パラメーター・マーカーを使用すると、プログラムはステートメントを 1 度だけ作成し、その後は毎回パラメーター・マーカーに異なる値を指定して **EXECUTE** ステートメントを実行します。

new in V9.7

DB2 9.7 では、DB2 に**ステートメント・コンセントレーター** という技術が導入されました。ステートメント・コンセントレーターは、フィールド値だけが異なるすべてのステートメントを、パラメーター・マーカーを使用して自動的に 1 つのステートメントにまとめます。その上で、異なる値を使用して EXECUTE ステートメントを実行するという機能です。しかも、ステートメント・コンセントレーターにはひとまとめにしてはならないステートメントを判別できるだけのインテリジェンスが備わっています。例えば、DB2 オプティマイザーに影響を与えるために意図的に節を追加しているステートメントについては、ひとまとめにして扱うことはしません。

パフォーマンスに関しては、通常は静的 SQL のほうが動的 SQL に勝っています。それは、静的 SQL のアクセス・プランは実行時ではなく、プリコンパイル時に計算されるためです。ただし、INSERT や DELETE といったアクティビティーが盛んに実行される環境では、プリコンパイル時に計算された統計が最新の状態を反映した統計ではなくなってしまったために、静的 SQL のアクセス・プランが最適なものでない可能性もあります。最新の統計を収集するために RUNSTATS コマンドが頻繁に実行されているとすれば、そのような環境には動的 SQL のほうが適しています。

注:

多くのユーザーは組み込み SQL は静的でしかないと考えていますが、実際には静的にも動的にもすることができます。

14.3.3 CLI と ODBC

当初、X/Open Company および SQL Access Group によって開発された **Call Level Interface (CLI)** は、RDBMS ベンダーとは関係なく移植できる C/C++ アプリケーションを開発することを目的とした、呼び出し可能 SQL インターフェースの仕様です。この X/Open Call Level Interface の草案を基盤として Microsoft は **Open Database Connectivity (ODBC)** を開発し、その後、ISO CLI International Standard が X/Open Call Level Interface 仕様の大部分を承認しました。DB2 CLI は、ODBC と SQL/CLI 国際標準の両方をベースとしています (図 14.3 を参照)。

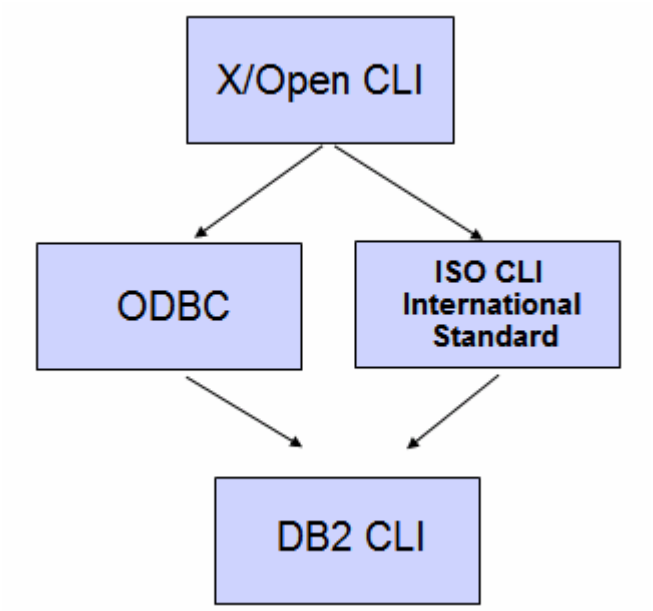


図 14.3 - ODBC および ISO CLI 国際標準をベースとした DB2 CLI

DB2 CLI は ODBC 3.51 に準拠していることから、ODBC Driver Manager でロードした場合には ODBC ドライバーとして使用することができます。図 14.4 から、DB2 CLI が ODBC をサポートすることが理解できるはずです。

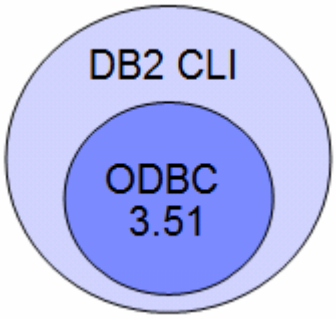


図 14.4 – ODBC 3.51 準拠の DB2 CLI

CLI/ODBC には以下の特徴があります。

- さまざまな RDBMS ベンダーの間でコードを容易に移植することができます。
- 組み込み SQL とは異なり、プリコンパイラーもホスト変数も不要です。
- 動的 SQL を実行します。
- 汎用されています。

CLI/ODBC アプリケーションを**実行** するために必要なのは、DB2 CLI ドライバーだけです。このドライバーは以下のクライアントおよびドライバーのいずれかからインストールされます。これらの

クライアントおよびドライバーは、www.ibm.com/db2/express から無料でダウンロードして利用することができます。

- IBM Data Server Client
- IBM Data Server Runtime Client
- IBM Data Server Driver for ODBC and CLI

CLI/ODBC アプリケーションを**開発** するには、DB2 CLI ドライバーに加え、適切なライブラリーがいくつか必要になります。これらのライブラリーは、IBM Data Server Client にあります。

アプリケーションで DB2 CLI ドライバーをセットアップする方法をよく理解できるように、以下に一例を記載します。図 14.5 に示されているのは、それぞれインドネシア、ブラジル、カナダにある 3 つの異なるマシンです。

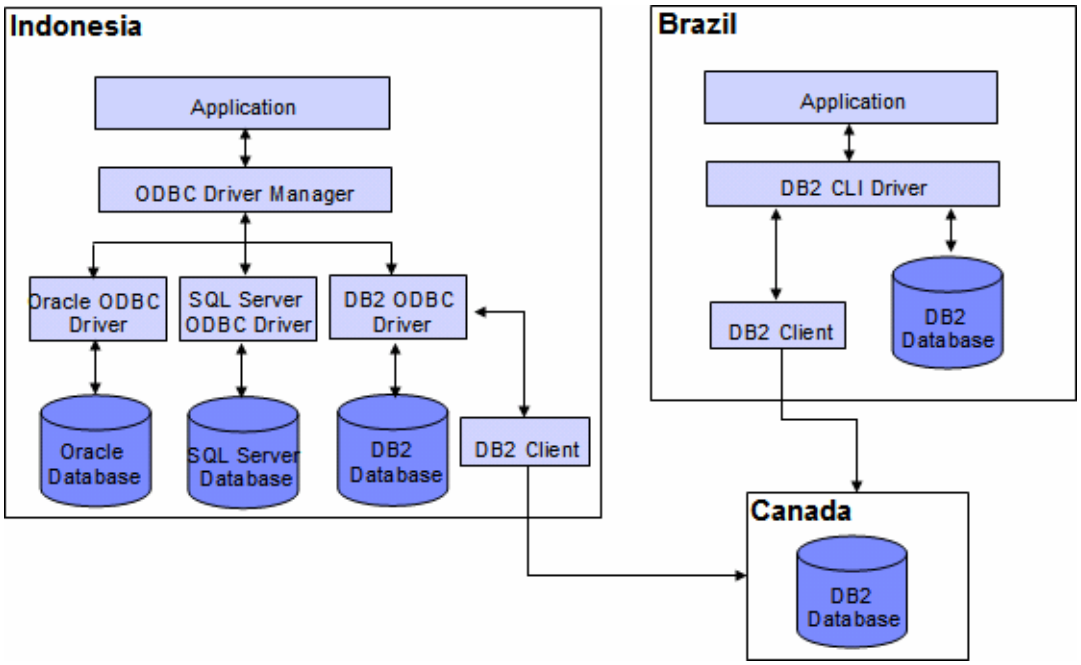


図 14.5 – DB2 CLI/ODBC サンプル・シナリオ

上記の図は、2 つの事例を示しています。

図の左側では、インドネシアにあるマシンが ODBC アプリケーション を実行しています。このアプリケーションは、Oracle、Microsoft SQL Server、または DB2 のいずれの RDBMS とも連動することができます。どのデータベースにアクセスしているかに応じて、ODBC Driver Manager が適切な ODBC ドライバーをロードします。アプリケーションがカナダの DB2 にアクセスする場合には、リモート接続用のコンポーネントを備えた DB2 クライアントを介して接続する必要があります。

一方、右側のブラジルにあるマシンでは、CLI アプリケーション が実行されています。これが CLI アプリケーションである理由は、ODBC では使用できない特定の関数を使用する場合があるため、そして DB2 データベースにのみ有効なアプリケーションであるためです。この CLI アプリケーショ

ンは DB2 CLI ドライバーを介して、ブラジルのローカル DB2 データベースに接続することができます。カナダのリモート・データベースに接続する必要がある場合には、さらに DB2 クライアントを介して接続します。

このセクションでは最後に、CLI/ODBC アプリケーションと C 言語への組み込み SQL による動的アプリケーションとの違いを明確にしておきます。図 14.6 に、その違いを示します。

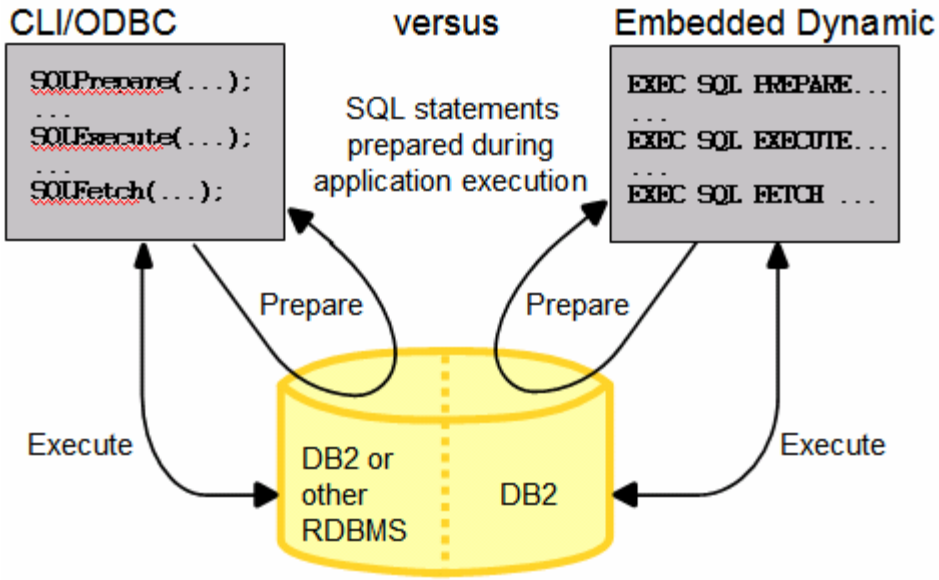


図 14.6 – CLI/ODBC アプリケーションと C 言語への組み込み SQL による動的アプリケーションとの違い

図 14.6 を見るとわかるように、CLI/ODBC アプリケーションと C 言語への組み込み SQL による動的アプリケーションとの違いは唯一、CLI/ODBC の場合、コードは移植可能であり、接続ストリングを変更するだけで他の RDBMS にもアクセスできる一方、C 言語への組み込み SQL による動的バージョンの場合には DB2 に固有の要素をコーディングすることになる可能性がある点です。当然のことながら、**PREPARE** と **EXECUTE** のそれぞれの関数を呼び出す方法も変わってきます。

14.3.4 JDBC、SQLJ、および pureQuery

Java Database Connectivity (JDBC) は、データベースの操作およびアクセス手段を標準化する Java プログラミング API です。JDBC では、さまざまな RDBMS ベンダーの間でコードを移植するのは簡単です。通常、コードに必要となる変更は、ロードする JDBC ドライバーと、接続ストリングだけです。JDBC が使用するのは動的 SQL だけであることも、高い人気を集めています。

SQLJ は、SQL を Java プログラムに組み込むための標準です。主として静的 SQL で使用されますが、JDBC と相互運用することも可能です (図 14.7 を参照)。SQLJ プログラムは通常、JDBC プログラムに比べ簡潔で、パフォーマンスにも優れていますが、汎用されるには至っていません。SQLJ プログラムをコンパイルするには、あらかじめプリプロセッサー (SQLJ トランスレーター) を使用して実行する必要があります

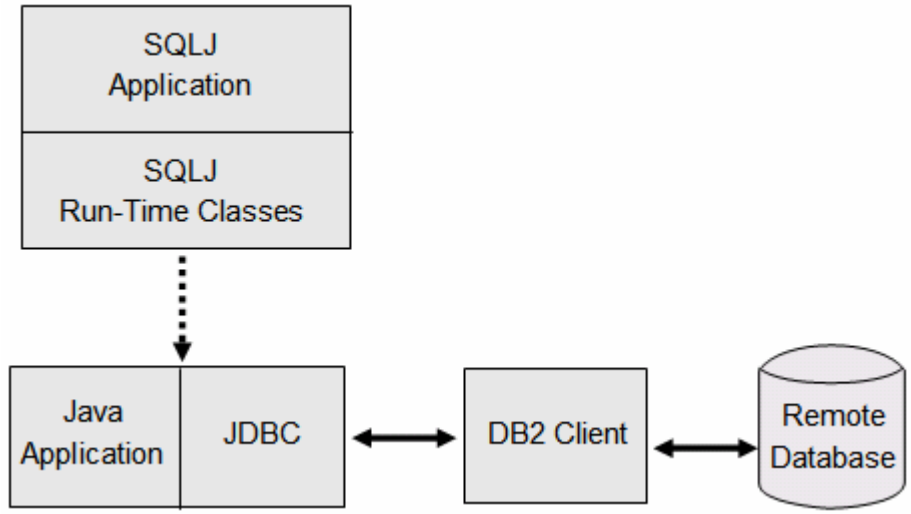


図 14.7 – SQLJ アプリケーションと JDBC アプリケーションとの関係

図 14.7 の DB2 クライアントが必要であるかどうかは、使用する JDBC ドライバーのタイプによって決まります (このセクションで追って説明します)。

pureQuery は、リレーショナル・データをオブジェクトとして管理するための IBM による Eclipse ベースのプラグインです。2007 年から使用できるようになった pureQuery では、オブジェクト指向のコードとリレーショナル・データベース・オブジェクトとの間のオブジェクト・リレーショナル・マッピング (ORM = Object-Relational Mapping) を確立するコードを自動的に生成することができます。それにはまず、Optim Development Studio (ODS) で Java プロジェクトを作成し、DB2 データベースに接続し、そして ODS にすべてのデータベース・オブジェクトを検出させます。ODS GUI を使って表を選択し、それから pureQuery コードの生成を選択すると、ベースとなるリレーショナル表に含まれるあらゆるエンティティーが Java オブジェクトに変換されます。コードが生成されると、関連する SQL ステートメントと、これらのステートメントをカプセル化する親 Java オブジェクトが作成されます。このようにして生成された Java オブジェクトとそこに含まれる SQL ステーメントは、さらにカスタマイズすることができます。pureQuery を使用する場合、SQL を静的モードと動的モードのどちらで実行するかを実行時に決定することができます。pureQuery は Java と .NET の両方をサポートします。

14.3.4.1 JDBC および SQLJ ドライバー

JDBC ドライバーにはいくつかのタイプ (タイプ 1、2、3、4 など) がありますが、そのうちタイプ 1 と 3 は、通常は使用されません。DB2 でのこの 2 つのタイプのサポートは廃止が予定されています。タイプ 2 には、この後説明するように 2 つのドライバーがありますが、そのうちの 1 つも廃止される予定です。

DB2 では、タイプ 2 とタイプ 4 がサポートされています (表 14.2 を参照)。タイプ 2 のドライバーは DB2 クライアントを使用してデータベースとの通信を確立するため、DB2 クライアントがインストールされている必要があります。タイプ 4 は純粋な Java クライアントなので DB2 クライアントがインストールされている必要はありませんが、このドライバーは JDBC アプリケーションが実行されているマシンにインストールしなければなりません。

ドライバー・タイプ	ドライバー名	パッケージ名	サポート対象	Java SDK の最小レベル要件
タイプ 2	DB2 JDBC Type 2 Driver for Linux, UNIX and Windows (**廃止予定***)	db2java.zip	JDBC 1.2 および JDBC 2.0	1.4.2
タイプ 2 およびタイプ 4	IBM Data Server Driver for JDBC and SQLJ	db2jcc.jar および sqlj.zip	JDBC 3.0 準拠	1.4.2
		db2jcc4.jar および sqlj4.zip	JDBC 4.0 以前	6

表 14.2 – DB2 JDBC および SQLJ ドライバー

* 廃止予定とは、現在はまだサポートされていますが、今後の拡張は行われないことを意味します。

前述の説明と表 14.2 の記載どおり、タイプ 2 には 2 種類のドライバーが用意されていますが、db2java.zip というファイル名の DB2 JDBC Type 2 Driver for Linux, UNIX and Windows は廃止される予定です。

DB2 サーバーをインストールすると、DB2 クライアント、または JDBC 3.0 準拠の IBM Data Server Driver for JDBC and SQLJ (db2jcc.jar および sqlj.zip ファイル) が自動的にクラス・パスに追加されます。

14.3.5 OLE DB

Object Linking and Embedding, Database (OLE DB) は、多種多様なソースに保管されたデータにアクセスするための一連のインターフェースです。OLE DB は ODBC に代わるものとして設計されましたが、さらに広範なソースをサポートするように拡張されており、オブジェクト指向データベースやスプレッドシートなどの非リレーショナル・データベースもサポートします。OLE DB の実装には、Component Object Model (COM) 技術が使用されます。

OLE DB コンシューマーが DB2 データベースにアクセスするには、IBM OLE DB Provider for DB2 を使用します。このプロバイダーには、以下の特徴があります。

- プロバイダー名: IBMDADB2
- レベル 0 の OLE DB プロバイダー仕様 (および、いくつかのレベル 1 インターフェース) をサポートします。
- バージョン 2.7 以降の Microsoft OLE DB 仕様に準拠します。
- IBM Data Server Client と Microsoft Data Access Components (MDAC) をインストールする必要があります。
- IBMDADB2 が明示的に指定されない場合、デフォルトで Microsoft の OLE DB ドライバー (MSDASQL) が使用されます。MSDASQL では、OLE DB を使用するクライアントが ODBC

ドライバーを使用して Microsoft SQL サーバー以外のデータ・ソースにアクセスすることができますが、OLE DB ドライバーの完全な機能は保証されません。

14.3.6 ADO.NET

.NET Framework は、Component Object Model (COM) 技術の代わりとなる Microsoft の技術です。.NET Framework では、40 種類以上のプログラミング言語で .NET アプリケーションのコードを作成することができます。そのうち最もよく使われている言語は、C# と Visual Basic .NET です。

..NET Framework クラス・ライブラリーには、NET アプリケーションを作成するために使用するビルディング・ブロックが用意されています。オペレーティング・システムとアプリケーション・サービスへのインターフェースを提供するこのクラス・ライブラリーは、言語にとらわれません。.NET アプリケーションはどの言語で作成されていても、バイト・コードの一種である中間言語 (IL = Intermediate Language) にコンパイルされます。

.NET Framework の中核となっているのは、共通言語ランタイム (CLR = Common Language Runtime) です。この CLR が、IL コードをオンザフライでコンパイルしてから実行します。コンパイルした IL コードを実行する際に、CLR はオブジェクトを起動し、そのオブジェクトのセキュリティー上の権限を検証してからメモリーを割り当て、それからオブジェクトを実行します。そして実行が完了するとオブジェクトのメモリーをクリーンアップします。

Java が機能する仕組みを例にとると、Java で作成されたプログラムは、最小限の変更を加えるか、またはまったく変更することなく、さまざまなプラットフォームで実行可能です。つまり、1 つの言語で、複数のプラットフォームに対応するということです。.NET では、サポートされている 40 の言語のどれを使って作成したプログラムでも、最小限の変更または変更なしで同じ 1 つのプラットフォーム、つまり Windows で実行可能です。つまり Java とは反対に、言語は複数でも 1 つのプラットフォームでいずれも動作するということです。

.NET Framework でのデータ・アクセスをサポートしているのは、**ADO.NET** です。ADO.NET は接続型アクセスと非接続型アクセスの両方をサポートします。非接続型アクセスの場合、ADO.NET で最も重要なコンポーネントとなるのは `DataSet` クラスです。このクラスのインスタンスが、アプリケーションのメモリー内に常駐するデータベース・キャッシュとして機能します。

接続型アクセスと非接続型アクセスのいずれにしても、アプリケーションはデータベースにアクセスする手段として、**データ・プロバイダー** を使用します。各種のデータベース製品には、それぞれに固有の .NET データ・プロバイダーが組み込まれています。DB2 for Windows も、その 1 つです。

.NET データ・プロバイダーには、以下に挙げる基本クラスが実装されています。

- Connection: データベース接続を確立および管理します。
- Command: データベースに対して SQL ステートメントを実行します。
- DataReader: データベースのデータを読み取り、結果セットのデータを返します。
- DataAdapter: DataSet インスタンスをデータベースに接続します。DataSet は DataAdapter インスタンスを使用することにより、データベース表のデータに対する読み取りおよび書き込み操作を実行します。

表 14.3 に、DB2 と連動可能な 3 つのデータ・プロバイダーを記載します。

データ・プロバイダー	特徴
ODBC .NET データ・プロバイダー (非推奨)	▪ DB2 CLI ドライバーを使用して DB2 データ・ソースに対する ODBC 呼び出しを行う ▪ キーワード・サポートおよび制約事項は、DB2 CLI ドライバーと同じ ▪ .NET Framework バージョン 1.1、2.0、または 3.0 で使用可能
OLE DB .NET データ・プロバイダー (非推奨)	▪ IBM DB2 OLE DB ドライバー (IBMDADB2) を使用 ▪ キーワード・サポートおよび制約事項は、DB2 OLE DB ドライバーと同じ ▪ .NET Framework バージョン 1.1、2.0、または 3.0 でのみ使用可能
DB2 .NET データ・プロバイダー (推奨)	▪ ADO.NET インターフェースの DB2 サポートを拡張 ▪ DB2 管理対象プロバイダーは、標準 ADO.NET クラスおよびメソッド一式と同じものを実装 ▪ IBM.DATA.DB2 名前空間のもとに定義 ▪ 以下のいずれかをダウンロードすることによって入手可能 – Data Server Driver for ODBC, CLI, and .NET – IBM Data Server Runtime Client – DB2 Data Server

表 14.3 – ADO.NET のデータ・プロバイダー

14.3.7 PHP

PHP Hypertext Preprocessor (PHP) は、Web アプリケーション開発用に設計された、プラットフォームに依存しないオープンソースのスクリプト言語です。HTML 内での組み込みが可能な PHP は、一般に、PHP コードを使用して Web ページを出力する Web サーバー上で実行されます。

PHP はモジュール式言語です。つまり、拡張機能を使用することによって、使用可能な機能をカスタマイズすることができます。なかでもとりわけよく使用されている PHP 拡張機能は、データベースへのアクセスに使用されるものです。IBM では以下の 2 つの拡張機能によって DB2 データベースへのアクセスをサポートしています。

- **ibm_db2**: ibm_db2 拡張機能は、データベース・メタデータへの広範なアクセスに加え、データベースに対して作成、読み取り、更新、書き込み操作を行うための手続き型アプリケーション・プログラミング・インターフェースを提供します。この拡張機能は、PHP 4 または PHP 5 のいずれかを使用してコンパイルすることができます。
- **pdo_ibm**: pdo_ibm は、PHP Data Objects (PDO) 拡張機能のドライバーです。PHP 5.1 で導入された標準オブジェクト指向データベース・インターフェースを介して、DB2 データベースにアクセスできるようにします。このドライバーは、DB2 ライブラリーに対して直接コンパイルすることができます。

PHP 拡張機能およびドライバーは、http://pecl.php.net/ にアクセスして PECL リポジトリーから無料で入手することができます。また、IBM Data Server Client にも付属しています。ibm_db2 と pdo_ibm はどちらも IBM DB2 CLI レイヤーをベースとします。

さらに IBM では、PHP と DB2 Express-C を使用して開発するための環境として用意された無料のツールキット、ZendCore をサポートするために、Zend Technologies Inc. とも提携しています。ZendCore バンドルには、PHP ライブラリー、Apache Web サーバー、および DB2 Express-C が含まれています。ZendCore をダウンロードするには、http://www.ibm.com/software/data/info/zendcore にアクセスしてください。

14.3.8 Ruby on Rails

Ruby はオープンソースのオブジェクト指向言語です。この Ruby を使用して作成された Web フレームワークが、Rails です。Ruby on Rails (RoR) は、バック･エンドにデータベースを配した Web ベースのアプリケーションを開発するには理想的な手段となります。モデル・ビュー・コントローラー (MVC = Model, View, Controller) アーキテクチャーをベースとしたこの最新技術は、アジャイル・ソフトウェア開発の原則に従います。

Rails には特殊なファイル・フォーマットも、統合開発環境 (IDE = Integrated Development Environment) も必要ありません。つまり、テキスト・エディターがあれば作業を始められるということです。その一方、Rails サポートを備えた各種の IDE も使用できるようになっています。その一例は、Eclipse 対応の Rails 環境である RadRails です。RadRails についての詳細を調べるには、http://www.radrails.org/ にアクセスしてください。

DB2 でサポートされているのは、Ruby 1.8.5 以降、および Ruby on Rails 1.2.1 以降です。IBM_DB gem には、DB2 での操作を可能にする IBM_DB Ruby ドライバーおよび Rails アダプターが組み込まれています。CLI レイヤーをベースとするこの gem は、IBM Data Server Client と併せてインストールする必要があります。IBM_DB ドライバーおよびアダプターをインストールするには、Ruby gem を使用することも、または Rails プラグインとしてインストールすることもできます。

14.3.9 Perl

多くのオペレーティング・システムに無料で用意されている Perl は、よく使われているインタープリター方式のプログラミング言語です。Perl は動的 SQL を使用することから、アプリケーションのプロトタイプを作成するには最適な手段となります。

Perl では、各種データベースにアクセスするためのモジュールとして、Database Interface (DBI) と呼ばれる標準モジュールを用意しています。DBI モジュールは、http://www.perl.com から入手する

ことができます。このモジュールはさまざまなデータベース・ベンダーのドライバーと「対話」します。DB2 の場合、対話の対象となるドライバーは DBD::DB2 ドラバーです。このドライバーは http://www.ibm.com/software/data/db2/perl から入手することができます。

14.3.10 Python

Python はスクリプトの作成によく使われる動的言語です。Python はコードの読みやすさに重点を置き、手続き型プログラミング、オブジェクト指向プログラミング、アスペクト指向プログラミング、メタ・プログラミング、関数型プログラミングなど、さまざまなプログラミング・パラダイムをサポートしています。Python は、迅速なアプリケーション開発を実現するのに理想的な言語です。

表 14.4 に、Python アプリケーションから DB2 データベースにアクセスするために使用できる拡張機能を記載します。

拡張機能	説明
ibm_db	IBM によって定義 高度な機能を対象とした最善のサポートを提供 SQL クエリーの実行、ストアード・プロシージャーの呼び出し、pureXML の使用、メタデータ情報へのアクセスを可能にする
ibm_db_dbi	Python データベース API 仕様バージョン 2.0 を実装 ibm_db API がサポートする高度な機能の一部は提供しない Python データベース API 仕様バージョン 2.0 をサポートするドライバーを使用したアプリケーションの場合には、容易に ibm_db に切り替えることが可能。ibm_db と ibm_db_dbi の API が一緒にパッケージ化されている
ibm_db_sa	よく使用されているオープンソースの Python SQL ツールキットである SQLAlchemy とオブジェクト・リレーショナル・マッパー (ORM = Object-to-Relational Mapper) をサポート

表 14.4 – IBM Data Server - Python 拡張機能

14.4 XML と DB2 pureXML

Extensible Markup Language (XML) は、Web 2.0 のツールと手法がベースとする技術であるだけでなく、サービス指向アーキテクチャー (SOA = **Service Oriented Architecture)** のベースともなっています。XML の重要性をいち早く認識した IBM では、DB2 での XML 文書により優れたストレージ・サポートを提供するための技術、pureXML の実現に向けて多大な投資を行いました。

2006 年に導入された DB2 9 は、ハイブリッド・データ・サーバーです。つまりリレーショナル・データだけでなく、階層型データもネイティブに保管することができます。それ以前のバージョンの DB2 や、市場に出回っているその他のデータ・サーバーでも XML 文書を保管することは可能でしたが、DB2 9 で用いられている保管方法は、パフォーマンスと柔軟性両方の面で改善されています。DB2 9 の pureXML 技術では、XML 文書は構文解析後の階層ツリーという形で DB2 内部に保管されるため、XML 文書の操作が大幅に強化されます。それ以降の DB2 9.5 や DB2 9.7 などの DB2

のリリースでは、さらに pureXML のサポートが改善されています。pureXML については、この技術に話題を絞った第 15 章「*DB2 pureXML*」で詳しく説明します。

14.5 Web サービス

単純に定義すると、Web サービスとは、ネットワークを介して呼び出すことのできる機能のことです。ネットワークを介して呼び出すということは、どのプログラミング言語で開発されたのか、どのオペレーティング・システムで動作するのか、そしてどこで実行されるのかについての知識はまったく必要ないということです。Web サービスでは、XML をベースとした拡張可能な業界標準プロトコルを使用して、アプリケーション間でデータを交換することができます。その仕組みを図 14.8 に示します。

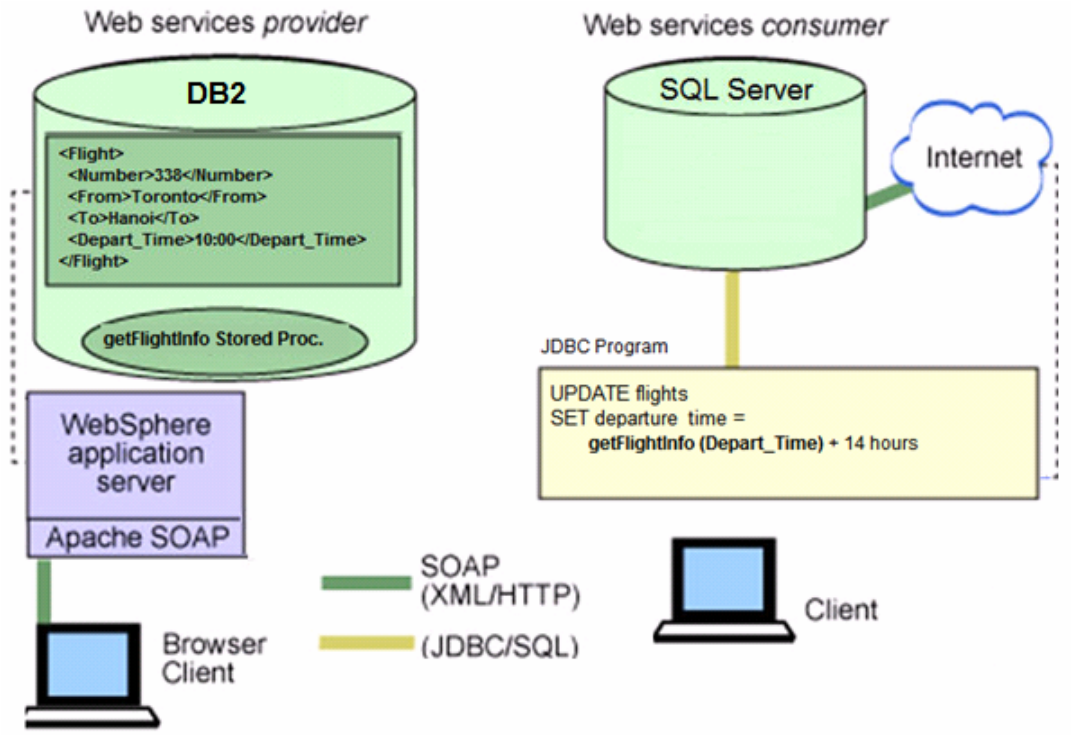


図 14.8 – 一例としての Web サービスの仕組み

この図の左側は、架空の航空会社 Air Atlantis のシステムを表しています。この会社では Linux 上で DB2 を使用し、フライト情報を XML フォーマットで DB2 データベースに保管しています。図の右側にあるのは、別の架空の航空会社 Air Discovery のシステムです。こちらの会社では Windows 上で動作する SQL サーバーを使用しています。ここで、例えば Air Atlantis と Air Discovery が連携協定を結び、両社のフライトを調整するために、スケジュールと価格に関する情報を共有しなければならなくなったとします。それぞれの会社が使用しているオペレーティング・システムは異なり (Linux と Windows)、データ・サーバーも異なることから (DB2 と SQL サーバー)、この 2 つの航空会社で情報を共有するのは難しいかもしれません。Air Atlantis がトロントから北京へのフライト・

スケジュールを変更した場合、Air Discovery が北京から上海までの乗継フライトのスケジュールを自動的に調整するにはどうしたらよいでしょうか。その答えは、Web サービスにあります。この場合、Air Atlantis は、DB2 データベースに保管されたフライト情報を使用したストアード・プロシージャー (***getFlightInfo*** ストアード・プロシージャー) の出力を返す **Data Web サービス** を作成することによって、そのフライト情報の一部を公開することができます。**Data** Web サービスは、データベース情報をベースとした Web サービスです。Data Web サービスを WebSphere Application Server などのアプリケーション・サーバーにデプロイすれば、Air Discovery のようなクライアントやパートナーは、ブラウザーを使用して Air Atlantis のフライト情報にごく簡単にアクセスすることができます。この例では、Web サービスを開発して使用できるようにした Air Atlantis が Web サービス・プロバイダーとして振る舞い、その Web サービスを使用する Air Discovery が Web サービス・コンシューマーとして振る舞います。

Air Discovery に関しては、この会社独自の JDBC アプリケーションから Web サービスを呼び出し、自社の SQL サーバー・データベースに保管されたデータを使用した計算を実行することも可能です。例えば、トロントから北京までの平均飛行時間が 12 時間だとすると、Air Discovery が北京発上海行きの乗継フライトの出発時刻を計算するには、Air Atlantis のトロント発のフライトの出発時刻に 12 時間を足し、さらにバッファーとして数時間を足します。バッファーとして計算する時間が、Air Discovery のシステムで SQL サーバー・データベースに保管されているとすれば、JDBC アプリケーションでは以下のような単純な式を使用することになります。

Air Discovery Flight Departure time (Beijing to Shanghai)	=	Air Atlantis Flight Departure time (Toronto to Beijing)	+	Air Atlantis Flight Duration (Toronto to Beijing) 12 hours average	+	Air Atlantis Flight Buffer time (Toronto to Beijing) 2 hours

Air Atlantis がそのフライト出発時刻を変更すると、この情報は、Air Discovery システムが Web サービスを呼び出したときに自動的に伝達されます。

14.6 管理 API

DB2 には、開発者が独自のユーティリティーやツールを作成するために使用できる多数の管理 API が用意されています。例えば、**sqlecrea API** を呼び出すことで、データベースを作成することができます。また、インスタンスを開始するには **db2InstanceStart API** を、データを表にインポートするには **db2Import API** を使用することができます。DB2 インフォメーション・センターには、これらの管理 API を網羅したリストが記載されています。DB2 インフォメーション・センターの URL にアクセスして、「参照情報」セクションを参照してください。

14.7 その他の開発

DB2 Express-C のユーザーのなかには、MS Excel や MS Access などのサード・パーティーの製品を利用して、DB2 に接続する単純なフォームを作成するユーザーもいます。そこで、このセクションではサード・パーティーの製品と DB2 Express-C を連動させる方法を説明します。

DB2 Express-C は Mac OS X でも使用できるので、Mac でネイティブに DB2 を使用してデータベース・アプリケーションを開発することもできます。これは、Mac プラットフォームを採用している RoR コミュニティーにとって特に魅力的な利点となるはずです。

14.7.1 Microsoft Access および Microsoft Excel との連動

Microsoft Excel と Microsoft Access は、レポートの生成やフォームの作成、そしてビジネス情報をデータに追加する単純なアプリケーションを開発する際によく使われているツールです。DB2 はこの 2 つのツールと極めて容易にやりとりをします。DBA が企業のデータをセキュア DB2 サーバーに保管すれば、Access や Excel を使用する一般のユーザーがこのデータにアクセスしてレポートを生成することができます。この仕組みを図 14.9 に示します。

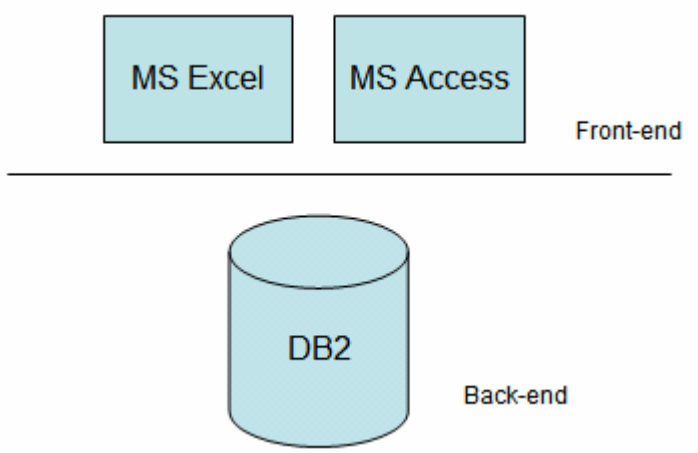


図 14.9 – Excel、Access、DB2 の連動

この図には、フロントエンドのアプリケーションの開発には Excel および Access を使用できる一方、データ・セキュリティー、信頼性、パフォーマンスについては、そのアプリケーションのバックエンドである DB2 が処理するという仕組みが示されています。このように、すべてのデータを DB2 に集めることによって、データ・ストレージ・モデルが単純化されます。

Excel の場合、DB2 データにアクセスするのに最も簡単な方法は、OLE DB ドライバー、例えば IBM OLE DB Provider for DB2 を使用することです。このドライバーは、DB2 Express-C の Web サイト (www.ibm.com/db2/express) からダウンロードすることができる無料の IBM Data Server Client をインストールすると、そのなかに組み込まれています。インストールが完了したら、MS Excel のメニューから、該当する OLE DB Provider で使用するデータ・ソースを選択してください。それには、「データ」→「外部データの取り込み」→「データの取り込み」の順に選択します。その後のステップは、記事「*IBM® DB2® Universal Database™ and the Microsoft® Excel Application Developer… for Beginners*」[1] に記載されているとおりです。詳細は、「参考文献」セクションを参照してください。

Microsoft Access の場合には、以下のいずれかがインストールされている必要もあります。

- IBM Data Server Client
- IBM Data Server Driver for ODBC, CLI and .Net

- IBM Data Server Driver for ODBC and CLI

IBM Data Server Driver for ODBC, CLI and .Net と IBM Data Server Driver for ODBC and CLI はいずれも、IBM DB2 ODBC ドライバーとして知られています。DB2 ODBC ドライバーは DB2 CLI ドライバーと同じですが、Access から DB2 に接続する場合に使用するドライバーです。ドライバーのインストールが完了したら、Access 2007 プロジェクトを作成し、「テーブル・ツール」リボンの「外部データ」タブ内にある「ODBC データベース」オプションを選択します。ここから先のステップは、記事「DB2 9 and Microsoft Access 2007 Part 1: Getting the Data...」[2] を参照してください。Microsoft Access で**リンク・テーブル** を使用すると、データは Access 2007 のユーザーに使用可能になりますが、データが置かれるのは DB2 データ・サーバーです。

Access 2007 より前のバージョンでのセットアップは多少異なりますが、その場合のセットアップについては記事「Use Microsoft Access to interact with your DB2 data」[3] に説明されています。詳細は、「参考文献」セクションを参照してください。

14.8 開発ツール

現在開発者たちの間で最もよく使用されている統合開発環境 (IDE = Integrated Development Environments) には、Microsoft Visual Studio と Eclipse の 2 つがあります。どちらも DB2 と相性の良い IDE です。

Microsoft Visual Studio 用に、DB2 では Visual Studio アドインを用意しています。このアドインをインストールすると Visual Studio のメニューに IBM のツールが追加されるため、開発者が DB2 データベースを操作する際に他のツールに切り替えなくても済むようになります。Visual Studio アドインは、IBM DB2 Express-C の Web サイト (www.ibm.com/db2/express) からダウンロードすることができます。

Eclipse に関しては、IBM では IBM Data Studio をリリースしています。これは無料の Eclipse ベースのツールで、このツールを使用して SQL および XQuery スクリプト、ストアード・プロシージャー、UDF、Web サービスを開発することができます。IBM Data Studio は Eclipse プラットフォームをベースとしたツールなので、開発者たちの多くはこれまでに蓄積した Eclipse の知識を活用することができます。

14.9 サンプル・プログラム

DB2 をデータ・サーバーとして使用して各種の言語でプログラミングを行う方法については、DB2 サーバー・インストールの SQLLIB\samples ディレクトリーに用意されている付属のサンプル・アプリケーションを調べると参考になります。図 14.10 に、Windows プラットフォームで実行可能な DB2 に付属のサンプル・プログラムを示します。

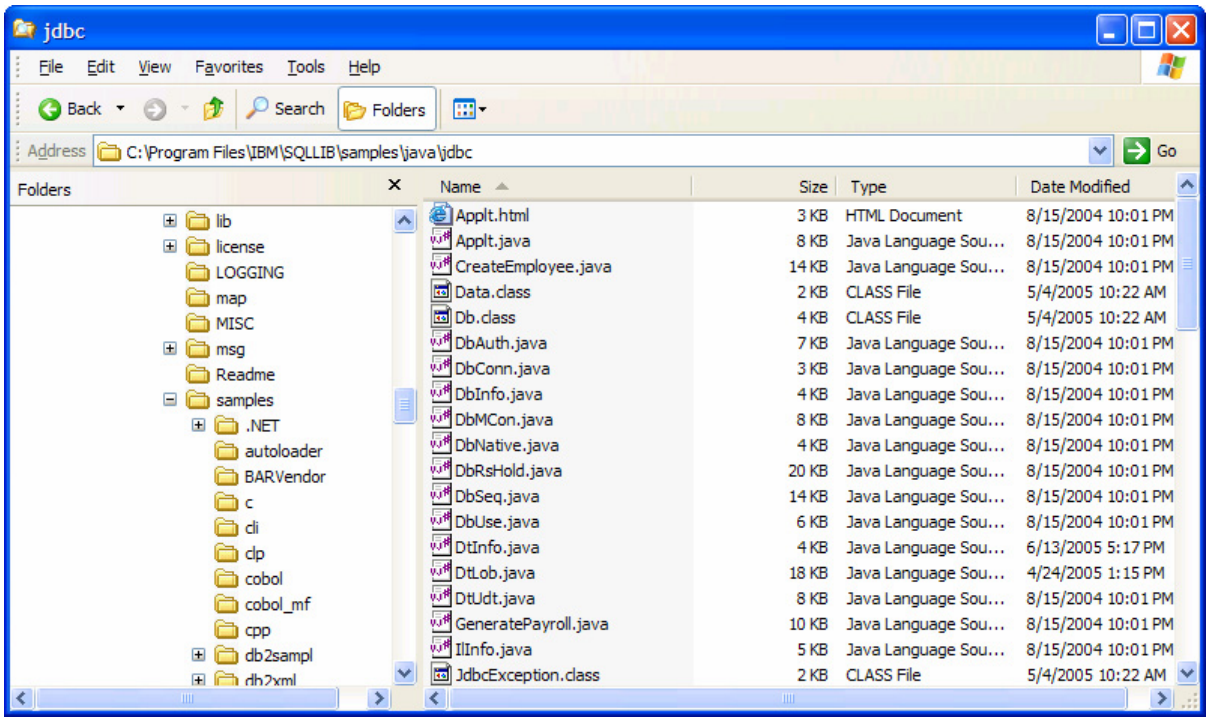


図 14.10 - DB2 に付属のサンプル・プログラム

14.10 まとめ

この章では、データベース・アプリケーションをプログラミングする際に DB2 によってもたらされる柔軟性について見てきました。この柔軟性はサーバー上のデータベース内部にもたらされることも、DB2 データ・サーバーに接続するクライアント・サイドのアプリケーションを介してもたらされることもあります。

サーバー・サイドのアプリケーションについてのセクションでは、ストアード・プロシージャー、ユーザー定義関数、トリガーを取り上げました。

クライアント・サイドに関しては、DB2 アプリケーション開発で使用できる多数のプログラミング・インターフェースとプログラミングの方法を説明し、データベース・サーバーとしての DB2 が持つ驚くべき柔軟性と性能を再度明らかにしました。

15

第 15 章 – DB2 pureXML

この章では、XML ネイティブ・ストレージをサポートするために DB2 9 で導入された新しい技術、pureXML について説明します。この章で説明するサンプルと概念の多くは、IBM Redbook「DB2 9: pureXML overview and fast start」から抜粋したものです。この本についての詳細は、「参考文献」セクションを参照してください。図 15.1 に、DB2 の「全体像」のなかでこの章が説明している部分を示します。

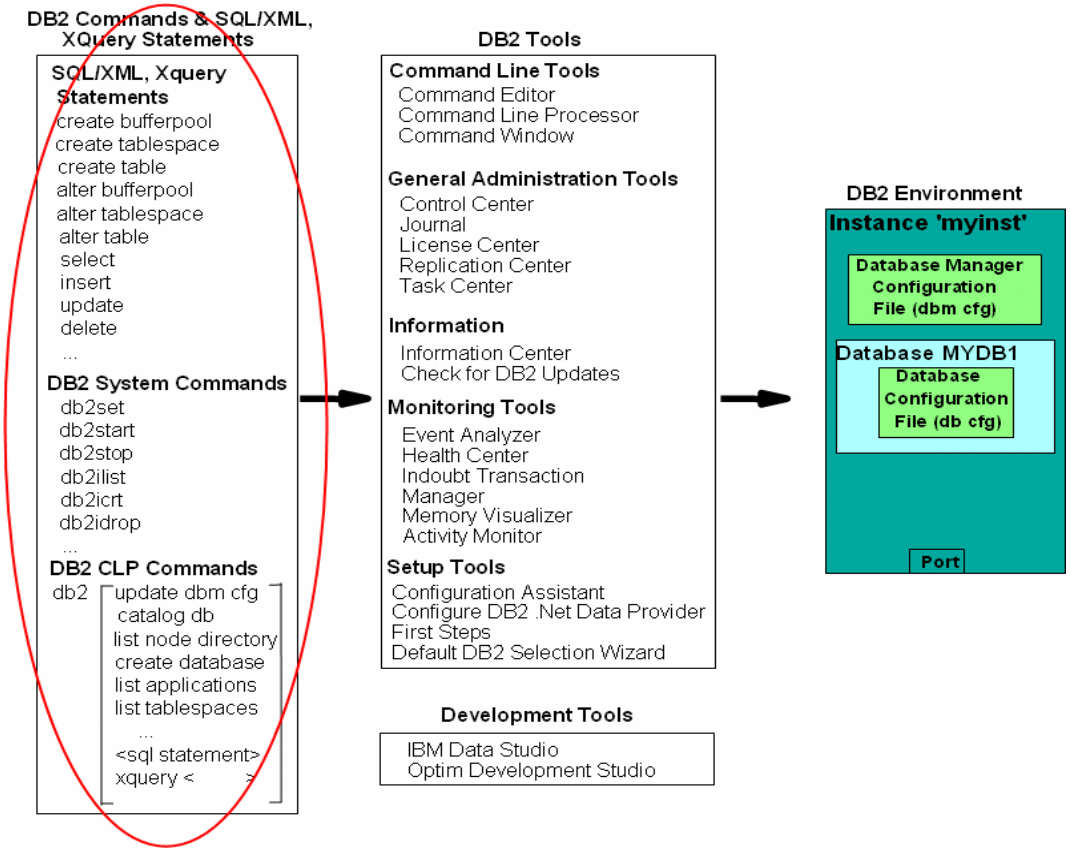


図 15.1 – DB2 の全体像: DB2 コマンド、SQL/XML および XQuery

15.1 データベースでの XML の使用

XML 文書は、テキスト・ファイルや、XML リポジトリー、データベースなどに保管することができますが、多くの企業では XML 文書をデータベースに保管することにしています。それには 2 つの主な理由があります。

- 大量の XML データを管理することは、データベースが対応するのに適した問題であるため。XML は他のデータと同様で、全体的なフォーマットが異なるというだけにすぎません。そのためリレーショナル・データをデータベースに保管する理由と同じ理由が XML データにも当てはまります。それは、データベースによって実現する効率的な検索およびデータの取得、データの永続性に対する強固なサポート、バックアップとリカバリー、トランザクション・サポート、そしてパフォーマンスとスケーラビリティーです。
- 統合が可能になるため。リレーショナル・データと XML 文書を併せて保管することにより、新しい XML データを既存のリレーショナル・データと統合し、1 つのクエリーのなかで SQL に XPath または XQuery を組み合わせることが可能になります。さらに、リレーショナル・データを XML として、あるいは XML をリレーショナル・データとしてパブリッシュすることもできます。この統合により、データベースは Web アプリケーション、SOA、Web サービスのサポートを強化することができます。

15.2 XML データベース

XML データを保管するデータベースには、以下の 2 つのタイプがあります。

- XML 対応データベース
- ネイティブ XML データベース

15.2.1 XML 対応データベース

XML 対応データベースでは、リレーショナル・モデルをその中核的データ・ストレージ・モデルとして使用して XML を保管します。これには XML (階層型) データ・モデルとリレーショナル・データ・モデルとのマッピングを使用するか、あるいは XML データを文字ラージ・オブジェクトとして保管する必要があります。XML 対応データベースは「古い」技術であると考えられますが、多くのデータベース・ベンダーでは今でもこのタイプのデータベースを使用しています。図 15.2 に、XML 対応データベースで XML を保管する 2 つの方法について詳しく説明します。

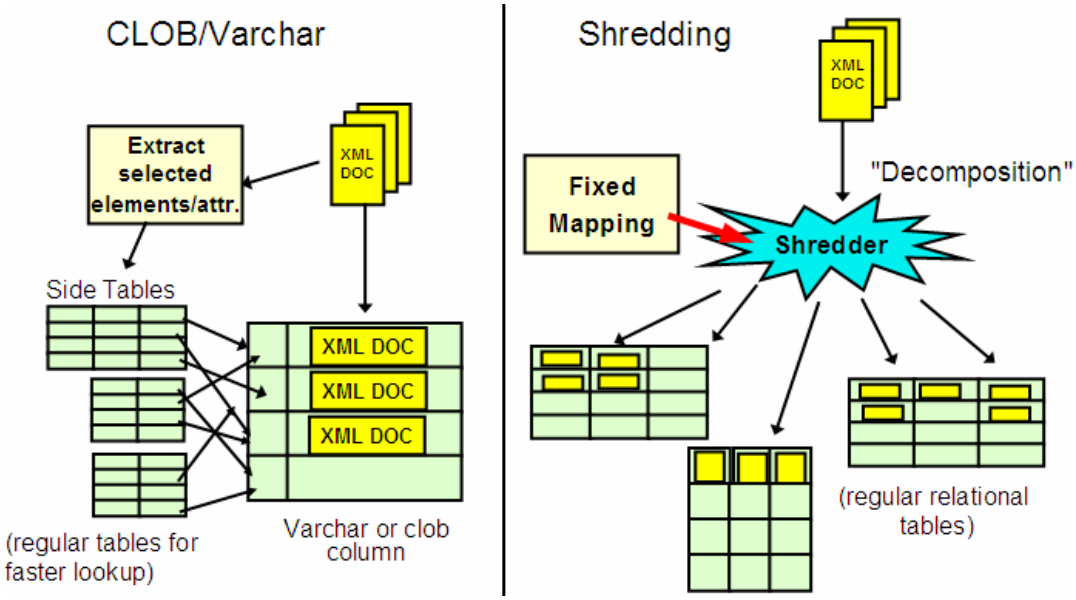


図 15.2 – XML 対応データベースで XML を保管する方法

図 15.2 の左側で図解しているのは、**CLOB および varchar 方式** を使用してデータベースに XML 文書を保管する場合です。この方式では、XML 文書は構文解析されていないストリングとして、データベースの CLOB 列または varchar 列に保管されます。XML 文書がストリングとして保管されている場合、XML 文書の一部を取得しようとすると、プログラムがストリング全体を取得し、それを構文解析して、要求されている部分を見つけることになります。構文解析とは、メモリー内に XML 文書ツリーを組み立て、このツリーをナビゲートできるようにすることだと考えてください。この方式ではメモリーをかなり使用し、柔軟性はあまりありません。

XML 対応データベースで使用できるもう 1 つの方法は、**シュレッド** または**ディコンポジション** と呼ばれる方式で、この方式を説明しているのが図 15.2 の右側です。この方式では、XML 文書全体が小さなパーツに断片化され、これらのパーツが表に保管されます。つまりこの方式では文字通り、階層型モデルに基づく XML 文書を強制的にリレーショナル・モデルに変換するということです。この方式には柔軟性はありません。XML 文書が変更された場合、その変更を該当する表に伝播させるのは容易なことではなく、その他多くの表を作成しなければならないこともあるからです。さらに、これはパフォーマンスに優れた方式でもありません。元の XML 文書に戻すには、手間のかかる SQL 結合操作を実行する必要がありますが、関与する表が多ければ、かかる手間はさらに大きなものになります。

15.2.2 ネイティブ XML データベース

階層型 XML データ・モデルを使用するネイティブ XML データベースでは、内部で XML を保管して処理します。つまり、保管フォーマットと処理フォーマットは同じであるため、リレーショナル・モデルとのマッピングはなく、XML 文書が構文解析されていないストリング (CLOB または varchar) として保管されることもありません。XPath または XQuery ステートメントが使用される場合、これらのステートメントはエンジンによってネイティブに処理され、SQL には変換されませ

ん。これが、「ネイティブ」XML データベースと呼ばれる由縁です。現在、このサポートを提供できる商用データ・サーバーは DB2 の他にはありません。

15.3 DB2 での XML

図 *15.3* に、リレーショナル・データと階層データ (XML 文書) の両方がどのように DB2 ハイブリッド・データベースに保管されるかを説明します。この図には、`dept` という名前の表を作成するために使用した **`CREATE TABLE`** ステートメントも示されています。

```
CREATE TABLE dept (deptID CHAR(8),…, deptdoc XML);
```

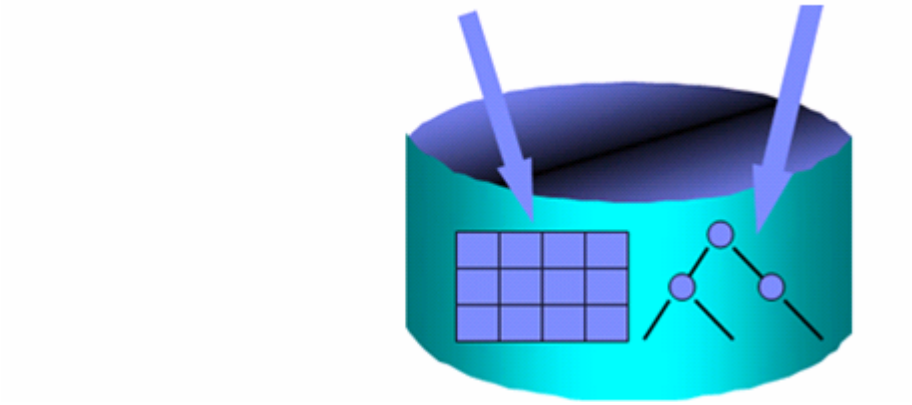


図 15.3 - DB2 での XML

この表定義では、`deptdoc` 列の新しいデータ型として XML を使用していることに注目してください。図の左側の矢印は、リレーショナル・フォーマット (表) に保管されたリレーショナル列 `deptID` を示しています。一方、XML 列 `deptdoc` は構文解析された階層フォーマットで保管されます。

図 15.4 に示すように、DB2 9 では現在、4 つの方法でデータにアクセスすることができます。

- SQL を使用してリレーショナル・データにアクセス
- SQL と XML 拡張機能 (SQL/XML) を使用して XML データにアクセス
- XQuery を使用して XML データにアクセス
- XQuery を使用してリレーショナル・データにアクセス

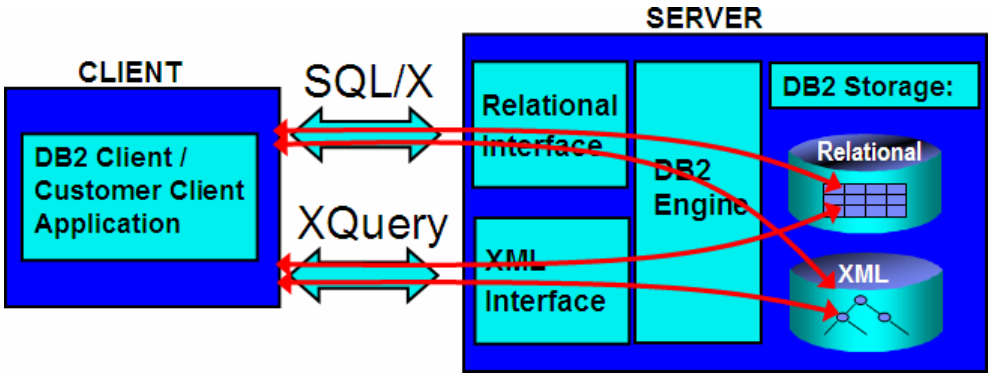


図 15.4 – DB2 のデータにアクセスする 4 つの方法

したがって、バックグラウンドに応じて説明すると、SQL を使い慣れている人にとっての DB2 は、XML もサポートする国際的レベルの RDBMS であり、XML を使い慣れている人にとっての DB2 は、SQL もサポートする国際的レベルの XML リポジトリーであると言えます。

IBM ではこの技術を表す言葉として、ネイティブ XML ではなく、pureXML という言葉を使用していることに注意してください。今でも CLOB/varchar またはシュレッドという古い手法で XML 文書を保管している他のベンダーは、その古い技術を「ネイティブ XML」と呼んでいます。混乱を避けるため、IBM では pureXML という新しい用語を使用することに決め、他のデータベースまたは XML のベンダーがこの名前で別の技術を意味することがないように、pureXML を商標登録することにしました。pureXML サポートは、Unicode として作成されたデータベースにも、Unicode 以外のデータベースにも提供されます。

15.3.1 pureXML 技術の利点

pureXML 技術は、以下のさまざまな利点をもたらします。

1. 表の列に新しい XML データ型を使用して XML 文書を保管することで、今までのリレーショナル・データベースへの投資をシームレスに利用することができます。
2. コードの複雑さを緩和することができます。例として、図 15.5 に PHP スクリプトの作成に pureXML を使用した場合と使用しない場合を記載します。pureXML を使用した場合 (左側の小さいほうの枠)、コードの行数は少なくなります。これは、コードが単純化されることを意味するだけでなく、構文解析および保守の対象となるコードの行数が少なくなることから、全体的なパフォーマンスが改善されることも意味します。

```
<?php
  $conn = db2_connect($dbname, $dbuser, $dbpass);

  /* Insert Customer Documents */

  $stmt = db2_prepare($conn, "VALUES (NEXT VALUE FOR
Cid)");
  db2_execute($stmt);
  list($Cid) = db2_fetch_array($stmt);

  $fileContents = file_get_contents
("customers/c1.xml");

  $stmt = db2_prepare($conn, "INSERT INTO xmlcustomer
(Cid, Info) VALUES (?, ?)");
  if(!db2_execute($stmt, array($Cid, $fileContents)))
{
    echo db2_stmt_errormsg($stmt);
  }

  /* Insert Product Documents */

  $fileContents = file_get_contents
("products/p1.xml");
  $dom = simplexml_load_string($fileContents);

  $prodID = (string) $dom["pid"];

  $stmt = db2_prepare($conn, "INSERT INTO xmlproduct
(Pid, Description) VALUES (?, ?)");
  if(!db2_execute($stmt, array($prodID,
```

```
  db2_execute($stmt);
  list($Cid) = db2_fetch_array($stmt);

  $fileContents = file_get_contents
("customers/c1.xml");
  $dom = simplexml_load_string($fileContents);

  $custName = (string) $dom->name;
  $custCountry = (string) $dom->addr["country"];
  $custStreet = (string) $dom->addr->street;
  $custCity = (string) $dom->addr->city;
  $custProvince = (string) $dom->addr->("prov-state");
  $custZip = (string) $dom->addr->("pcode-zip");
  $custPhone = (string) $dom->phone;

  $stmt = db2_prepare($conn, "INSERT INTO sqlcustomer
(Cid, Name, Country, Street, City, Province, Zip,
Phone, Info) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");
  if(!db2_execute($stmt, array($Cid, $custName,
$custCountry, $custStreet, $custCity, $custProvince,
$custZip, $custPhone, $fileContents) )) {
    echo db2_stmt_errormsg($stmt);
  }

  /* Insert Product Documents */

  $fileContents = file_get_contents
("products/p1.xml");
  $dom = simplexml_load_string($fileContents);

  $prodID = (string) $dom["pid"];
```

図 15.5 – pureXML を使用した場合と使用しない場合のコードの複雑さの比較

3. XML および pureXML 技術を使用することで、スキーマを容易に変更できるようになります。図 15.6 は、この柔軟性の向上を示す一例です。この図では、***Employee*** 表と ***Department*** 表で構成されるデータベースがあるという設定となっています。例えばマネージャーから、従業員ごとに 1 つの電話番号 (自宅の電話番号) だけを保管するのではなく、もう 1 つの電話番号 (携帯電話の番号) も保管するように指示されたとします。非 XML データベースの場合、***Employee*** 表に列を追加し、この新しい列に携帯電話の番号を保管するという方法を考えるでしょうが、この方法はリレーショナル・データベースの正規化規則に反しています。正規化規則に従うとしたら、***Phone*** という新しい表を別途作成し、この表にすべての電話番号情報を移さなければなりません。そうすれば、携帯電話の番号もこの表に追加することができます。けれども新しい ***Phone*** 表を作成するとなると、既存の大量のデータを移動しなければならないだけでなく、アプリケーションのすべての SQL もこの新しい表を指すように変更しなければならないため、手間がかかります。

   一方、図の左側には XML を使用して携帯電話の番号を追加する方法を示しています。***Christine*** という従業員は自宅の電話番号だけでなく、携帯電話の番号も持っているとすれば、新しいタグを追加することで、この情報を追加することができます。携帯電話の番号を持っていない従業員 ***Michael*** のデータについては、何もする必要はありません。

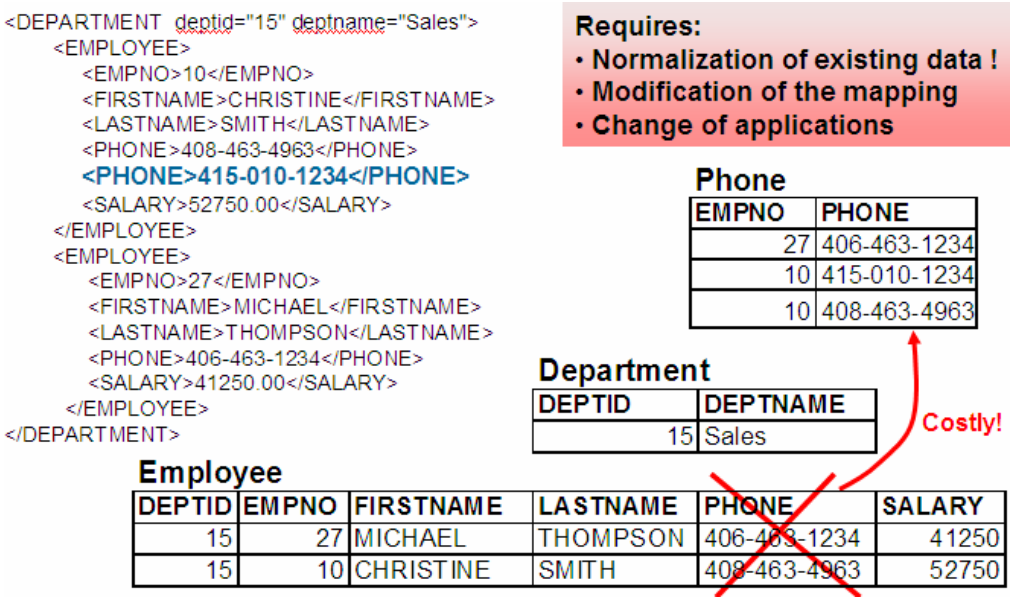


図 15.6 – XML の使用による柔軟性の向上

4. XML アプリケーションのパフォーマンスを改善することができます。pureXML 技術を使用して行ったテストでは、XML アプリケーションのパフォーマンスが大幅に向上することが示されました。表 15.1 に、古い技術から pureXML に切り換えた企業でのテスト結果を記載します。2 列目に記載されているのが、DB2 以外のリレーショナル・データベースを使用した以前の方法で XML を操作した場合のテスト結果です。3 列目に、DB2 pureXML を使用した場合の結果が示されています。

タスク	別のリレーショナル DB	pureXML を使用した DB2
検索および取得を行うビジネス・プロセスの開発	CLOB: 8 時間 シュレッド: 2 時間	30 分
I/O コードの相対行数	100	35 (65% の削減)
スキーマへのフィールド追加	1 週間	5 分
照会	24 ～ 36 時間	20 秒 ～ 10 分

表 15.1 – pureXML 技術の使用によるパフォーマンスの向上

15.3.2 XPath の基本

XPath は、XML 文書を照会するために使用できる言語です。リスト 15.1 に記載する XML 文書は、構文解析して階層 (「ノード」または「リーフ」とも呼ばれます) フォーマットで表すと図 15.7 のようになります。この後、この構文解析後の階層フォーマットを使用して、XPath がどのように機能するかを説明します。

```
<dept bldg="101">
    <employee id="901">
          <name>John Doe</name>
<phone>408 555 1212</phone>
<office>344</office>
</employee>
<employee id="902">
<name>Peter Pan</name>
<phone>408 555 9918</phone>
<office>216</office>
</employee>
</dept>
```

リスト 15.1 - XML 文書

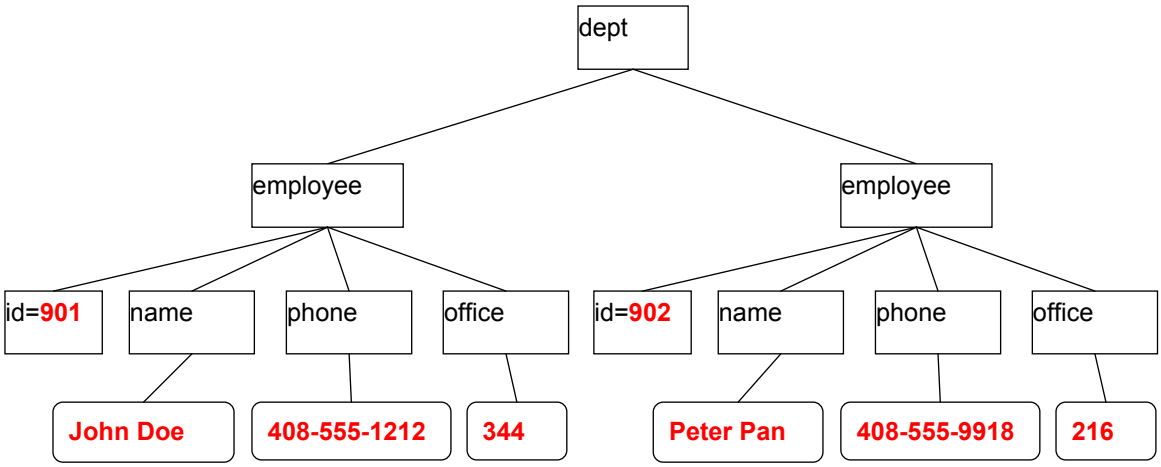


図 15.7 – リスト 15.1 の XML 文書を構文解析した後の階層による表現

XPath を短時間で習得するには、MS-DOS または Linux/UNIX の `change directory (cd)` コマンドと比較してください。`cd` コマンドを使用してディレクトリー・ツリーをトラバースする場合は、以下のようにします。

```
cd /directory1/directory2/…
```

XPath でも同様に、XML 文書内のある要素から別の要素に移動するにはスラッシュを使用します。例えば XPath でリスト 15.1 の文書から従業員全員の名前を取得するには、以下のクエリーを実行します。

```
/dept/employee/name
```

15.3.2.1 XPath 式

XPath 式では、完全修飾パスを使用して要素や属性を指定します。属性を指定するには、「@」記号を使用します。要素の値 (テキスト・ノード) のみを取得するには、**text()** 関数を使用します。表 15.2 に、リスト 15.1 の XML 文書を使用した XPath クエリーとそれぞれに対応する結果を記載します。

XPath	結果
/dept/@bldg	101
/dept/employee/@id	901 902
/dept/employee/name	<name>Peter Pan</name> <name>John Doe</name>
/dept/employee/name/text()	Peter Pan John Doe

表 15.2 – XPath 式の例

15.3.2.2 XPath ワイルドカード

XPath で使用するワイルドカードには、主に以下の 2 つがあります。

- 「*」はすべてのタグ名と一致します。
- 「//」は「下層またはそれ自身」を表すワイルドカードです。

表 15.3 に、リスト 15.1 の XML 文書を使用した例をさらに記載します。

XPath	結果
/dept/employee/*/text()	John Doe 408 555 1212 344 Peter Pan 408 555 9918 216
/dept/*/@id	901 902
//name/text()	Peter Pan John Doe
/dept**//**phone	<phone>408 555 1212</phone> <phone>408 555 9918</phone>

表 15.3 – XPath ワイルドカードの例

15.3.2.3 XPath 述部

述部は大括弧 [ ] で囲みます。述部はいわば、SQL での WHERE 節に相当するものです。例えば `[@id="902"]` とある場合、「id 属性が 902」であることを条件としていると解釈することができます。1 つの XPath 式に複数の述部が含まれる場合もあります。定位置述部を指定する場合には、[n] を使用します。これは、n 番目の子を選択するという意味です。例えば `employee[2]` は、2 番目の従業員を選択することを意味します。表 15.4 に、リスト 15.1 の XML 文書を使用した例をさらに記載します。

XPath	結果
/dept/employee[@id="902"]/name	<name>Peter Pan</name>
/dept[@bldg="101"]/employee[office >"300"]/name	<name>John Doe</name>
//employee[office="344" OR office="216"]/@id	901 902
/dept/employee[2]/@id	902

表 15.4 – XPath 述部の例

15.3.2.4 親軸

MS-DOS や Linux/UNIX と同様に、現行のコンテキストを参照していることを示すには、式のなかで「.」(ドット) を使用します。親コンテキストを参照する場合に使用するのは「..」(2 つのドット) です。表 15.5 に、リスト 15.1 の XML 文書を使用した例をさらに記載します。

XPath	結果
/dept/employee/name[../@id="902"]	<name>Peter Pan</name>
/dept/employee/office[.>"300"]	<office>344</office>
/dept/employee[office > "300"]/office	<office>344</office>
/dept/employee[name="John Doe"]/../@bldg	101
/dept/employee/name[.="John Doe"]/../../@bldg	101

表 15.5 – XPath 親軸

15.3.3 XQuery の基本

XQuery は、XML を対象に作成された問い合わせ言語です。XQuery ではパス式によって XML 階層構造をナビゲートすることができます。実のところ、XPath は XQuery のサブセットであるため、上記で説明した XPath の内容はすべて、XQuery にも当てはまります。XQuery は型付けされたデータと型付けされていないデータの両方をサポートします。XML 文書は欠落しているデータ、または不明なデータを省略することから、XQuery には NULL 値がありません。XQuery 式と XPath 式は大/小文字を区別し、XQuery は XML データのシーケンスを返します。

XQuery は FLWOR 式をサポートします。SQL に例えるなら、FLWOR 式は SELECT-FROM-WHERE 式に相当します。次のセクションで、FLWOR について詳しく説明します。

15.3.3.1 XQuery: FLWOR 式

FLWOR は、以下の略語です。

- FOR: シーケンスを繰り返し処理し、変数を項目にバインド
- LET: 変数をシーケンスにバインド
- WHERE: 繰り返し処理の項目を排除
- ORDER: 繰り返し処理の項目を再配列
- RETURN: 照会結果を構成

FLWOR 式は、XML 文書を操作して別の式を返せるようにする式です。例えば、以下の定義を持つ表があるとします。

```
CREATE TABLE dept(deptID CHAR(8),deptdoc XML);
```

この表の ***deptdoc*** 列には、リスト 15.2 の XML 文書が挿入されます。

```
<dept bldg="101">
    <employee id="901">
        <name>John Doe</name>
        <phone>408 555 1212</phone>
        <office>344</office>
    </employee>
    <employee id="902">
        <name>Peter Pan</name>
        <phone>408 555 9918</phone>
        <office>216</office>
    </employee>
</dept>
```

リスト 15.2 - XML 文書の例

この場合、FLWOR 式を使用したリスト 15.3 の XQuery ステートメントを実行することができます。

```
xquery
for $d in db2-fn:xmlcolumn('dept.deptdoc')/dept
let $emp := $d//employee/name
where $d/@bldg > 95
order by $d/@bldg
return
   <EmpList>
   {$d/@bldg, $emp}
   </EmpList>
```

リスト 15.3 – FLWOR 式を使用した XQuery ステートメントの例

上記のステートメントによって、リスト 15.4 の出力が返されます。

```
<EmpList bldg="101">
 <name>
        John Doe
 </name>
 <name>
        Peter Pan
 </name>
</EmpList>
```

リスト 15.4 – リスト 15.3 の XQuery ステートメントを実行した後の出力

15.3.4 XML 文書の挿入

XML 文書を DB2 データベースに挿入するには、SQL の INSERT ステートメント、または IMPORT ユーティリティーを使用することができます。XQuery は、文書を挿入する目的では使用できません。現時点ではまだ、標準で定義されていないためです。

例として、リスト 15.5 に記載する table_creation.txt という名前のスクリプトを調べてみましょう。このスクリプトを実行するには、DB2 コマンド・ウィンドウまたは Linux シェルから以下のステートメントを使用します。

```
db2 -tvf table_creation.txt
```

```
-- (1)
drop database mydb
;

-- (2)
create database mydb using codeset UTF-8 territory US
;

-- (3)
connect to mydb
;

-- (4)
create table items (
  id          int primary key not null,
  brandname   varchar(30),
  itemname    varchar(30),
  sku         int,
  srp         decimal(7,2),
  comments    xml
);
```

```
-- (5)
create table clients(
  id          int primary key not null,
  name        varchar(50),
  status      varchar(10),
  contact     xml
);

-- (6)
insert into clients values (77, 'John Smith', 'Gold',
   '<addr>111 Main St., Dallas, TX, 00112</addr>')
;

-- (7)
IMPORT FROM "D:\Raul\clients.del" of del xml from "D:\Raul" INSERT INTO
CLIENTS (ID, NAME, STATUS, CONTACT)
;

-- (8)
IMPORT FROM "D:\Raul\items.del" of del xml from "D:\Raul" INSERT INTO
ITEMS (ID, BRANDNAME, ITEMNAME, SKU, SRP, COMMENTS)
;
```

リスト 15.5 – table_creation.txt ファイルの内容

このスクリプト・ファイルおよびその関連ファイルは、本書に付属の **Expressc_book_exercises_9.7.zip** という圧縮ファイルに含まれています。以下に、リスト 15.5 のスクリプトのそれぞれの行について説明します。

1. データベース ***mydb*** をドロップします。これは通常、クリーンアップを行うためにスクリプト・ファイルで行われることです。***mydb*** が存在していない場合にはエラー・メッセージを受け取りますが、このエラーは無視して構いません。
2. UTF-8 コードセットを使用して ***mydb*** データベースを作成します。この行で作成しているのは Unicode のデータベースです。pureXML は、Unicode のデータベースでも、Unicode 以外のデータベースでもサポートされます。
3. 新しく作成した ***mydb*** データベースに接続します。これは、データベース内にオブジェクトを作成するためです。
4. ***items*** という名前の表を作成します。この表の最後の列 (***comments*** 列) は、新しい XML データ型を使用した XML 列として定義されることに注意してください。
5. ***clients*** という名前の表を作成します。この表の最後の列 (***contact*** 列) も、新しい XML データ型を使用した XML 列として定義されます。

6. この行の SQL INSERT ステートメントを使用して、XML 文書を XML 列に挿入することができます。INSERT ステートメントでは、XML 文書を引用符で囲んだストリングとして渡します。

7. IMPORT コマンドを使用すると、複数の XML 文書をリレーショナル・データと併せてデータベースに挿入 (インポート) することができます。(7) の行では、`clients.del` ファイル (区切り ASCII ファイル) のデータをインポートする際に、この `clients.del` ファイルが参照する XML データの配置場所 (例えば、`D:\Raul`) も指定しています。

`clients.del` ファイルについてはこの後さらに詳しく調べますが、その前に、まずは `D:\Raul` ディレクトリーの中を見てください。その内容は、図 15.8 のとおりです。

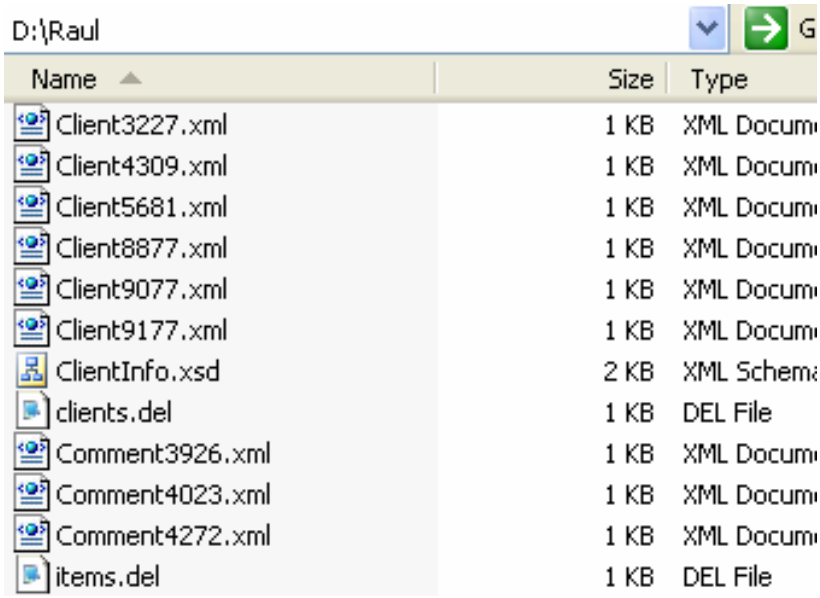


図 15.8 – XML 文書が含まれる D:\Raul ディレクトリーの内容

リスト 15.6 に、`clients.del` ファイルの内容を記載します。

```
3227,Ella Kimpton,Gold,<XDS FIL='Client3227.xml' />,
8877,Chris Bontempo,Gold,<XDS FIL='Client8877.xml'/>,
9077,Lisa Hansen,Silver,<XDS FIL='Client9077.xml' />
9177,Rita Gomez,Standard,<XDS FIL='Client9177.xml'/>,
5681,Paula Lipenski,Standard,<XDS FIL='Client5681.xml' />,
4309,Tina Wang,Standard,<XDS FIL='Client4309.xml'/>
```

リスト 15.6 – clients.del ファイルの内容

`clients.del` ファイルでは、「`XDS FIL=`」を使用して特定の XML 文書ファイルを指しています。

図 15.9 に、上記のスクリプトを実行した後のコントロール・センターを示します。

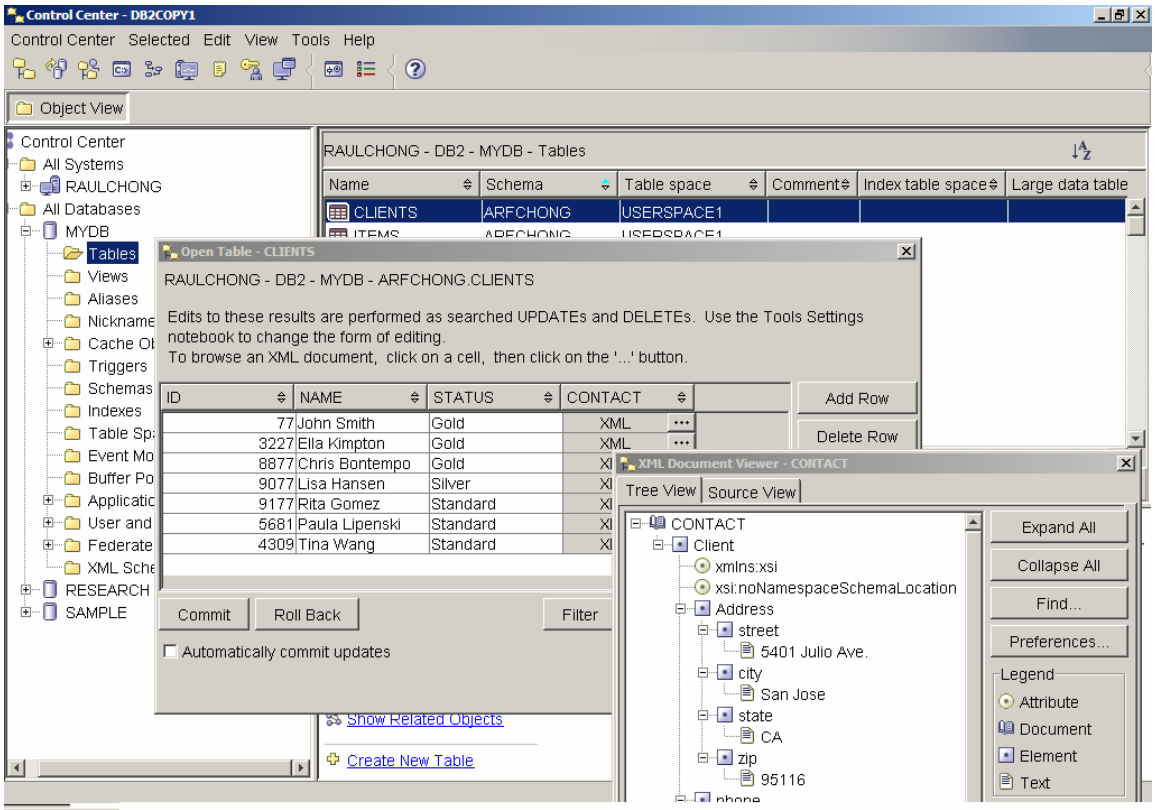


図 15.9 - table_creation.txt スクリプトを実行した後のコントロール・センター

上記の図に示されている **CLIENTS** 表の内容に注目してください。最後の **CONTACT** 列は、XML 列です。3 つのドットが付いたボタンをクリックすると、XML 文書の内容を表示する別のウィンドウが開きます。それが、図 15.9 の右下隅に示されているウィンドウです。

15.3.5 XML データの照会

DB2 に保管された XML データを照会するには、以下の 2 つの方法があります。

- XML 拡張機能を備えた SQL を使用 (SQL/XML)
- XQuery を使用

いずれの方法にしても、DB2 は国際 XML 標準に従います。

15.3.5.1 SQL/XML による XML データの照会

行と列を処理するには、通常の SQL ステートメントを使用することができます。ただし、SQL ステートメントは XML 文書全体の処理には使用できますが、文書の一部だけを取得したい場合には役に立ちません。その場合には、XML 拡張機能を備えた SQL (SQL/XML) を使用する必要があります。

表 15.6 に、SQL 2006 標準で使用できる SQL/XML 関数をいくつか抜粋して説明します。

関数名	説明
XMLPARSE	文字またはラージ・オブジェクトのバイナリー・データを構文解析し、XML 値を生成する
XMLSERIALIZE	XML 値を文字またはラージ・オブジェクトのバイナリー・データに変換する
XMLVALIDATE	XML スキーマに照らし合わせて XML 値の妥当性検査を行い、XML データに型のアノテーションを付ける
XMLEXISTS	XQuery が結果 (つまり、1 つ以上の項目のシーケンス) を返すかどうかを判断する
XMLQUERY	XQuery を実行し、結果シーケンスを返す
XMLTABLE	XQuery を実行し、結果シーケンスをリレーショナル表として返す (可能な場合)
XMLCAST	XML 型に、または XML 型からキャストする

表 15.6 – SQL/XML 関数

以下で説明する例は、前に作成した ***mydb*** データベースでテストすることができます。

例 1

特定の郵便番号の住所に住むすべてのクライアントの名前を探したいとします。郵便番号を含め、顧客の住所は **clients** 表の XML 列に保管されています。この場合、XMLEXISTS を使用して、まず XML 列でターゲットの郵便番号を検索し、その検索結果に応じて戻りの結果セットを絞り込むことができます。そのために必要なクエリーは、リスト 15.7 のとおりです。

```
SELECT name FROM clients
 WHERE xmlexists(
     '$c/Client/Address[zip="95116"]'
     passing clients.contact as "c"
     )
```

リスト 15.7 – XMLEXISTS の使用例

リスト 15.7 の最初の行は、**clients** 表の **name** 列にある情報を取得するように指定する SQL 節です。

WHERE 節では XMLEXISTS 関数が呼び出されています。この関数に指定した XPath 式では、DB2 に **zip** 要素にナビゲートし、95116 の値を見つけるように指示しています。

$c/Client/Address 節が指定しているのは、DB2 が **zip** 要素を検出できる XML 文書階層内のパスです。ドル記号 ($) は変数を指定するために使用されます。したがって、「c」は変数です。この変数は次の行、`passing clients.contact as "c"` で定義されます。ここで、**clients** は表の名前、**contact** は XML データ型が定義されている列の名前です。つまり、ここでは XML 文書を変数「c」に渡していることになります。

DB2 は **contact** 列に含まれる XML データを検査してから、ルートである **Client** ノードから **Address** ノードまでナビゲートし、さらに **zip** ノードまでナビゲートして顧客がターゲットの郵便番号の住所に住んでいるかどうかを最終的に判断します。XMLEXISTS 関数が「true」に評価されると、DB2 はその行に関連付けられたクライアントの名前を返します。

DB2 9.5 以降、上記のクエリーはリスト 15.8 のように単純化できるようになっています。

```
SELECT name FROM clients
 WHERE xmlexists(
     '$CONTACT/Client/Address[zip="95116"]'
     )
```

リスト 15.8 – リスト 15.7 に記載したクエリーの簡易バージョン

DB2 により、XML 列と同じ名前の変数が自動的に作成されます。上記の例では、***CONTACT*** 変数が DB2 によって自動的に作成されます。この名前は、XML 列である **CONTACT** と同じです。

例 2

ステータスが「Gold」の顧客の E メール・アドレスをリストアップするレポートを作成しなければならないとします。この問題を解決するには、例えばリスト 15.9 に記載するクエリーを実行するという方法が考えられます。

```
SELECT xmlquery('$c/Client/email' passing contact as "c")
  FROM  clients
  WHERE status = 'Gold'
```

リスト 15.9 – XMLQUERY の使用例

最初の行で、XML 文書の要素である (リレーショナル列ではありません) E メール・アドレスを返すように指定します。前の例と同じく、「$c」は変数で、この変数には XML 文書が含まれます。この例で使用しているのは、XMLQUERY 関数です。この関数は SELECT の後に続けて使用することができます。一方、XMLEXISTS を使用できるのは WHERE 節の後です。

例 3

例えば、XML データを表形式で表示したいという場合もあります。それには、リスト 15.10 に記載する XMLTABLE 関数を使用します。

```
SELECT t.comment#, i.itemname, t.customerID, Message
  FROM items i,
  xmltable('$c/Comments/Comment' passing i.comments as "c"
      columns Comment# integer path 'CommentID',
      CustomerID integer path 'CustomerID',
      Message varchar(100) path 'Message') AS t
```

リスト 15.10 – XMLTABLE の使用例

最初の行で、結果セットに含める列を指定します。「t」変数で始まる列は、XML 要素の値をベースとします。

3 番目の行で XMLTABLE 関数を呼び出し、ターゲット・データ (**i.comments**) が含まれる DB2 XML 列を指定し、次に列の XML 文書内で該当する要素が位置する場所へのパスを指定します。

4 行目から 6 行目までの **columns** 節によって特定される XML 要素が、1 行目に指定された SQL 結果セットに列を出力するためにマッピングされます。このマッピングには、XML 要素の値をどのデータ型に変換するかの指定も含まれます。この例では、すべての XML データは従来の SQL データ型に変換されます。

例 4

今度は、SQL/XML の XMLQUERY 関数のなかに XQuery の FLWOR 式を組み込む単純な例を見てみます。この例をリスト 15.11 に記載します。

```
SELECT name, xmlquery(
    'for $e in $c/Client/email[1] return $e'
    passing contact as "c"
    )
  FROM clients
  WHERE status = 'Gold'
```

リスト 15.11 – XMLQUERY および FLWOR の使用例

最初の行で、顧客の名前と XMLQUERY 関数による出力を結果セットに含めるように指定します。2 番目の行で、*Client* 要素の最初のサブ要素である *email* を返すように指定します。3 番目の行で XML データのソース (**contact** 列) を特定し、4 番目の行で、この列に **clients** 表のデータが入力されることを指定します。そして 5 番目の行で、**Gold** 顧客のみが対象であることを指定します。

例 5

リスト 15.12 でも、XQuery の FLWOR 式を取る XMLQUERY 関数の使い方を示していますが、この例は XML だけでなく、HTML も返すことに注目してください。

```
SELECT xmlquery('for $e in $c/Client/email[1]/text()
    return <p>{$e}</p>'
    passing contact as "c")
  FROM clients
  WHERE status = 'Gold'
```

リスト 15.12 – XML と HTML を返す例

XQuery の return 節によって、XML 出力を必要に応じて変換することができます。最初の行で **text()** 関数を使用しているということは、ここで対象としているのは、該当する顧客の最初の E メール・アドレスのテキスト表現だけであるということです。2 番目の行が、この情報を HTML のパラグラフ・タグで囲むように指定しています。

例 6

以下の例では、XMLELEMENT 関数を使用して一連の item 要素を作成しています。各要素には ID、商標名、SKU (最小在庫管理単位) に対応するサブ要素があり、これらの値が **items** 表の対応するそ

れぞれの列から取得されます。基本的に、リレーショナル・データを XML データに変換するには XMLELEMENT 関数を使用することができます。リスト 15.13 を参照してください。

```
SELECT
  xmlelement (name "item", itemname),
  xmlelement (name "id", id),
  xmlelement (name "brand", brandname),
  xmlelement (name "sku", sku)
  FROM items
  WHERE srp < 100
```

リスト 15.13 – XMLELEMENT の使用例

リスト 15.13 のクエリーによって、リスト 15.14 の出力が返されます。

```
<item>
       <id>4272</id>
       <brand>Classy</brand>
       <sku>981140</sku>
</item>
…
<item>
       <id>1193</id>
       <brand>Natural</brand
       <sku>557813</sku>
</item>
```

リスト 15.14 – リスト 15.13 のクエリーによる出力

15.3.5.2 XQuery による XML データの照会

前のセクションでは、XML 拡張機能を備えた SQL を使用して XML データを照会する方法を説明しました。記載した例では、SQL が常に主要な照会手法として使われており、XPath または XQuery は SQL のなかに組み込まれて使用されていました。このセクションで説明するのは、XQuery を使用して XML データを照会する方法です。今回は XQuery が主要な照会手法となり、場合によっては (**db2-fn:sqlquery** 関数を使用して) XQuery のなかに SQL を組み込んで使用します。XQuery を使用する際にはいくつかの関数を呼び出し、さらに FLWOR 式も使用します。

例 1

以下の例は、顧客の連絡先データを返す単純な XQuery です。この例では、**CONTACT** が XML 列の名前で、**CLIENTS** が表の名前です。

```
       xquery db2-fn:xmlcolumn('CLIENTS.CONTACT')
```

XQuery 式の前には必ず **xquery** コマンドを付けなければなりません。これは、DB2 に XQuery パーサーを使用する必要があることを認識させるためです。このコマンドで始まらない場合、DB2 は SQL 式を実行しようとしていると判断します。**db2-fn:xmlcolumn** 関数は、パラメーターとして指定された列から XML 文書を取得する関数です。この関数は列に含まれる全データを取得することから、以下の SQL ステートメントに相当します。

```
SELECT contact FROM clients
```

例 2

リスト 15.15 に記載する例では、FLWOR 式を使用してクライアントのファックス・データを取得します。

```
xquery
  for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
  return $y
```

リスト 15.15 – FLWOR 式を使用した XQuery

最初の行は XQuery パーサーを呼び出します。2 番目の行は、DB2 に **CLIENTS.CONTACT** 列に含まれる fax サブ要素を繰り返し処理するように指示します。各 fax 要素は変数 $y にバインドされます。3 番目の行で、繰り返し処理ごとに値「$y」を返すように指定しています。

リスト 15.16 に、このクエリーの出力を記載します (この出力では名前空間が省略されています。名前空間は複数行にわたる場合があるので、省略しないと出力が読みにくくなるためです)。

```
<fax>4081112222</fax>
<fax>5559998888</fax>
```

リスト 15.16 – リスト 15.15 のクエリーによる出力

例 3

リスト 15.17 の例は、XML データを照会し、その結果を HTML として返します。

```
xquery
  <ul> {
    for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/Address
    order by $y/zip
    return <li>{$y}</li>
  }
  </ul>
```

リスト 15.17 – HTML を返す、FLWOR 式を使用した XQuery ステートメントの例

リスト 15.18 は、上記のステートメントによって返される HTML の一例です。

```
<ul>
<li>
<address>
    <street>9407 Los Gatos Blvd.</street>
```

```
    <city>Los Gatos</city>
    <state>ca</state>
    <zip>95302</zip>
</address>
</li>
<address>
<street>4209 El Camino Real</street>
    <city>Mountain View</city>
    <state>CA</state>
    <zip>95302</zip>
</address>
</li>
...
</ul>
```

リスト 15.18 – リスト 15.17 のクエリーによる出力

例 4

以下の例に、**db2-fn:sqlquery** 関数を使用して XQuery のなかに SQL を組み込む方法を示します。**db2-fn:sqlquery** 関数は SQL クエリーを実行し、選択された XML データのみを返します。**db2-fn:sqlquery** に渡す SQL クエリーが返すのは、XML データのみです。この XML データを、XQuery でさらに処理することができます。リスト 15.19 を参照してください。

```
xquery
  for $y in
  db2-fn:sqlquery(
    'select comments from items where srp > 100'
  )/Comments/Comment
  where $y/ResponseRequested='Yes'
  return (
    <action>
      {$y/ProductID
       $y/CustomerID
       $y/Message}
    </action>
  )
```

リスト 15.19 – XQuery に SQL が組み込まれた db2-fn:sqlquery 関数の例

上記の SQL クエリーは、**srp** 列が 100 よりも大きい値を持つことを条件に行をフィルタリングします。そしてフィルタリングされた行から、XML 列である **comments** 列を選択します。続いてサブ要素を処理するために XQuery (または XPath) が適用されます。

注:

SQL は大/小文字を区別しないため、DB2 はデフォルトで、表と列のすべての名前を大文字で保管します。一方、XQuery は大/小文字を区別します。上記の関数は XQuery インターフェース関数なので、すべての表および列の名前は大文字で渡さなければなりません。オブジェクト名を小文字で渡した場合、オブジェクト名が定義されていないというエラーが発生することになります。

15.3.6 SQL/XML による結合

このセクションでは、それぞれ別個の表にある 2 つの XML 列、または XML 列とリレーショナル列の結合操作を行う方法を説明します。ここに記載する例では、リスト 15.20 のステートメントによって 2 つの表が作成されていることを前提とします。

```
CREATE TABLE dept (unitID CHAR(8), deptdoc XML)
CREATE TABLE unit (unitID  CHAR(8) primary key not null,
                   name    CHAR(20),
                   manager VARCHAR(20),
                   ...
                   )
```

リスト 15.20 – 結合操作の例で使用する表の DDL

結合操作を行うには、2 つの方法があります。最初の方法は、リスト 15.21 に記載するとおりです。

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE XMLEXISTS (
    '$e//employee[name = $m]'
    passing d.deptdoc as "e", u.manager as "m")
```

リスト 15.21 – SQL/XML で結合を行うための最初の方法

結合操作は、上記リストのステートメントの 4 行目に示されています。この操作で結合しているのは、**dept** 表の **deptdoc** XML 列のサブ要素である **name** 要素と、**unit** 表の **manager** リレーショナル列です。

リスト 15.22 に、もう 1 つの結合方法を記載します。

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE u.manager = XMLCAST(
    XMLQUERY('$e//employee/name '
      passing d.deptdoc as "e")
    AS char(20))
```

リスト 15.22 – SQL/XML で結合を行うための 2 番目の方法

この 2 番目の方法では、リレーショナル列が結合操作の式の左側にあります。このようにリレーショナル列が式の左側にある場合、XML 索引の代わりにリレーショナル索引を使用することができます。

15.3.7 XQuery による結合

以下の表がすでに作成されているとします。

```
CREATE TABLE dept(unitID CHAR(8), deptdoc XML)
CREATE TABLE project(projectDoc XML)
```

リスト 15.23 は、SQL/XML を使用して結合操作を行う場合の一例です。

```
SELECT XMLQUERY (
  '$d/dept/employee' passing d.deptdoc as "d")
  FROM dept d, project p
  WHERE XMLEXISTS (
    '$e/dept[@deptID=$p/project/deptID]'
      passing d.deptdoc as "e", p.projectDoc as "p")
```

リスト 15.23 – SQL/XML による結合

XQuery を使用して結合操作を行う場合は、リスト 15.24 のようになります。

```
xquery
  for $dept in db2-fn:xmlcolumn("DEPT.DEPTDOC")/dept
  for $proj in db2-fn:xmlcolumn("PROJECT.PROJECTDOC")/project
  where $dept/@deptID = $proj/deptID
  return $dept/employee
```

リスト 15.24 – XQuery による結合

この 2 番目の方法のほうが、解釈しやすくなります。このリストでは、***$dept*** 変数に **dept** 表の XML 列 **deptdoc** に保管されている XML 文書が入り、***$proj*** 変数には **project** 表の XML 列 **projectdoc** に保管されている XML 文書が入ります。続いて 4 番目の行が最初の XML 文書の属性と 2 番目の XML 文書の要素との間で結合操作を行います。

15.3.8 更新および削除操作

XML データでの更新および削除操作は、以下のいずれかの方法で行うことができます。

- SQL の UPDATE および DELETE ステートメントを使用
- transform 式を使用

最初の UPDATE および DELETE ステートメントを使用する方法では、更新または削除は文書レベルで行われます。つまり、XML 文書全体が更新後の文書に置き換えられるということです。例えば、リスト 15.25 の UPDATE ステートメントの場合、変更するのは <state> 要素のみだとしても、実際には XML 文書ごと置き換えられます。

```
UPDATE clients SET contact=(
  xmlparse(document
    '<Client>
       <address>
         <street>5401 Julio ave.</street>
```

```
          <city>San Jose</city>
          <state>CA</state>
          <zip>95116</zip>
        </address>
        <phone>
          <work>4084633000</work>
          <home>4081111111</home>
          <cell>4082222222</cell>
        </phone>
        <fax>4087776666</fax>
        <email>newemail@someplace.com</email>
      </Client>')
  )
  WHERE id = 3227
```

リスト 15.25 – SQL UPDATE の例

transform 式を使用する 2 番目の方法では、サブ文書を更新できるため、遥かに効率的です。この式では、XML 文書内のノードを置換、挿入、削除、または名前変更することができます。さらに、ノード自体を置換することなく、ノードの値を変更することもできます。要素または属性の値を変更するというのは、非常に一般的なタイプの更新です。このサポートは、DB2 9.5 で追加されました。

transform 式は XQuery 言語の一部であるため、例えば FLWOR 式のなかや、SQL/XML ステートメントの XMLQUERY 関数のなかでなど、通常 XQuery を使用するような場所で使用することができます。最も一般的な使用方法としては、SQL UPDATE ステートメントのなかで、XML 列の XML 文書を変更するために使うというものです。

リスト 15.26 に、transform 式の構文を記載します。

```
>>-transform--| copy clause |--| modify clause |--| return clause |-><

copy clause

         .-,-------------------------------------.
         V                                       |
|--copy----$VariableName--:=--CopySourceExpression-+------------|

modify clause

|--modify--ModifyExpression-------------------------------------|

return clause

|--return--ReturnExpression-------------------------------------|
```

リスト 15.26 - transformt 式の構文

copy 節を使用して、変数に処理対象の XML 文書を割り当てます。

modify 節では、insert、delete、rename、または replace 式を呼び出すことができます。これらの式を使用することによって、XML 文書を更新することができます。

例えば、以下の使用方法があります。

- 新しいノードを文書に追加するには、insert 式を使用します。
- XML 文書からノードを削除するには、delete 式を使用します。
- XML 文書内の要素または属性の名前を変更するには、rename 式を使用します。
- 既存のノードを新しいノードまたはノードのシーケンスに置き換えるには、replace 式を使用します。この式に含める置換値は、要素または属性の値を変更するためにのみ使用することができます。

return 節が、transform 式の結果を返します。

リスト 15.27 は、transform 式を使用した UPDATE ステートメントの一例です。

```
(1)-- UPDATE customers
(2)-- SET contactinfo = xmlquery( 'declare default element namespace
(3)--                              "http://posample.org";
(4)--    transform
(5)--    copy $newinfo := $c
(6)--          modify do insert <email2>my2email.gm.com</email2>
(7)--                    as last into $newinfo/customerinfo
(8)--    return $newinfo' passing contactinfo as "c")
(9)-- WHERE id = 100
```

リスト 15.27 – transform 式を使用した UPDATE

上記の例で、(1)、(2)、および (9) の行は SQL UPDATE 構文の一部です。(2) の行で XMLQUERY 関数が呼び出されることによって、行 (4) の transform 式が呼び出されます。(4) から (8) の行までの transform 式のブロックによって、***email2*** 要素が含まれる新しいノードが XML 文書に挿入されます。ビューによる XML 文書の要素の更新はサポートされていないことに注意してください。

表から XML 文書全体を削除するのは、SQL/XML で SELECT ステートメントを使用する場合と同様に単純で、SQL の DELETE ステートメントを使用して、必要な WHERE 述部を指定するだけのことです。

15.3.9 XML 索引の作成

XML 文書では、索引を要素、属性、あるいは値 (テキスト・ノード) を対象に作成することができます。以降に記載する例では、以下の表が作成されていることを前提とします。

```
CREATE TABLE customer(info XML)
```

また、この表に保管されている文書のうちの 1 つは、リスト 15.28 に記載する XML 文書であるという前提です。

```
<customerinfo Cid="1004">
   <name>Matt Foreman</name>
   <addr country="Canada">
       <street>1596 Baseline</street>
       <city>Toronto</city>
       <state>Ontario</state>
       <pcode>M3Z-5H9</pcode>
   </addr>
   <phone type="work">905-555-4789</phone>
   <phone type="home">416-555-3376</phone>
   <assistant>
      <name>Peter Smith</name>
      <phone type="home">416-555-3426</phone>
   </assistant>
</customerinfo>
```

リスト 15.28 – XML 索引に関する例で使用する XML 文書

リスト 15.29 に記載するステートメントは、特定の ***Cid*** 属性で索引を作成します。

```
CREATE UNIQUE INDEX idx1 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/@Cid'
  AS sql DOUBLE
```

リスト 15.29 – Cid 属性の索引

リスト 15.30 に記載するステートメントは、特定の ***name*** 要素で索引を作成します。

```
CREATE INDEX idx2 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/name'
  AS sql VARCHAR(40)
```

リスト 15.30 – name 要素の索引

リスト 15.31 に記載するステートメントは、すべての ***name*** 要素で索引を作成します。

```
CREATE INDEX idx3 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//name'
  AS sql VARCHAR(40);
```

リスト 15.31 – すべての name 要素の索引

リスト 15.32 に記載するステートメントは、すべての text ノード (すべての値) で索引を作成しますが、この方法は推奨されません。更新、削除、挿入操作に際して索引を管理するのに手間がかかり、しかも索引が大きくなりすぎてしまうためです。

```
CREATE INDEX idx4 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//text()'
  AS sql VARCHAR(40);
```

リスト 15.32 – すべての text ノードの索引 (非推奨)

15.4 XML スキーマの操作

DB2 で XML 文書をデータベースに挿入できるのは、その文書が整形式である場合です。整形式でなければ、文書を挿入しようするとエラーを受け取ることになります。その一方、DB2 では XML 文書の妥当性検査を強制しません。XML 文書の妥当性検査を行いたい場合には、このセクションで説明するいくつかの方法を使用することができます。

15.4.1 XML スキーマの登録

XML スキーマは、DB2 データベースでは XML スキーマ・リポジトリーと呼ばれる場所に保管されます。XML スキーマをリポジトリーに追加するには、REGISTER XMLSCHEMA コマンドを使用します。

例えば、ある XML 文書が `order.xml` というファイルに保管されているとします (図 15.10 を参照)。

```
<?xml version="1.0" encoding="UTF-8"?>
  <po:PurchaseOrder xmlns:po="http://www.test.com/po">
     <Header>
          <Id>1</Id>
          <date>2004-01-29</date>
          <description>purchase order</description>
          <value>20</value>
          <status>shipped</status>
     </Header>
     <Items>
          <Item>
               <ItemDescription color="red" weight="5">
                        <Name>Widget C</Name>
                        <SKU>1</SKU>
                        <Price>30</Price>
                        <Comment>no comment</Comment>
               </ItemDescription>
               <NumberOrdered>1</NumberOrdered>
          </Item>
     </Items>
     <Customer type="regualar">
           <Name>Manoj K Sardana</Name>
           <Address>ring road, bangalore</Address>
           <Phone>918051055109</Phone>
           <email>msardana@in.ibm.com</email>
     </Customer>
  </po:PurchaseOrder>
```

図 15.10 – XML 文書が保管されている order.xml ファイル

ここで、`order.xsd` というファイルに XML スキーマ文書が保管されているとします (図 15.11 を参照)。

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.test.com/po"
            xmlns:po="http://www.test.com/po"
            xmlns:head="http://www.test.com/header"
            xmlns:prod = "http://www.test.com/product"
            xmlns:cust = "http://www.test.com/customer">

<xsd:import namespace="http://www.test.com/product" schemaLocation="product.xsd" />
<xsd:import namespace="http://www.test.com/customer" schemaLocation="customer.xsd" />
<xsd:import namespace="http://www.test.com/header" schemaLocation="header.xsd" />
<xsd:complexType name="itemType">
<xsd:sequence>
  <xsd:element name="Item" minOccurs="1" maxOccurs="unbounded" >
    <xsd:complexType>
       <xsd:sequence>
         <xsd:element name="ItemDescription" type="prod:prodType" />
         <xsd:element name="NumberOrdered" type="xsd:integer" />
       </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="potype">
<xsd:sequence>
  <xsd:element name="Header"   type="head:headerType"  />
  <xsd:element name="Items"    type="po:itemType" />
  <xsd:element name="Customer" type="cust:customerType" />
</xsd:sequence>
</xsd:complexType>

<xsd:element name="PurchaseOrder" type="po:potype" />

</xsd:schema>
```

図 15.11 – XML スキーマが保管されている order.xsd ファイル

上記の XML スキーマ文書では、以下の箇所が円で囲まれて強調表示されています。

- <xsd:schema _.>: これが XML スキーマ文書であることを示します。
- <xsd:import _>: この XML スキーマに組み込む他の xsd ファイル (他の XML スキーマ) をインポートします。
- minOccurs="1": これは XML スキーマの「規則」の一例で、ここでは **Item** 要素は少なくとも 1 回は発生しなければならないことを規定しています。つまり、少なくとも 1 つの **Item** 要素がなければならないということです。

この XML スキーマをデータベースに登録するには、リスト 15.33 のようなスクリプトを使用します。このスクリプトには、その内容がわかるようにコメントを組み込んでいます。

```
-- データベースに接続
CONNECT TO SAMPLE;

-- メイン XML スキーマを登録
REGISTER XMLSCHEMA http://www.test.com/order FROM D:\example3\order.xsd AS
order;

-- XML スキーマ文書をメイン・スキーマに追加
ADD XMLSCHEMA DOCUMENT TO order ADD http://www.test.com/header FROM
D:\example3\header.xsd;

-- XML スキーマ文書をメイン・スキーマに追加
ADD XMLSCHEMA DOCUMENT TO order ADD http://www.test.com/product FROM
D:\example3\product.xsd;

-- XML スキーマ文書をメイン・スキーマに追加
ADD XMLSCHEMA DOCUMENT TO order ADD http://www.test.com/customer FROM
D:\example3\customer.xsd;

-- スキーマの登録を完了
COMPLETE XMLSCHEMA order;
```

リスト 15.33 – XML スキーマの登録手順を説明するスクリプトの例

後でこの情報を見直す際には、SELECT ステートメントを使用して Catalog 表からこの情報を選択します (リスト 15.34 を参照)

```
SELECT CAST(OBJECTSCHEMA AS VARCHAR(15)), CAST(OBJECTNAME AS VARCHAR(15))
  FROM syscat.xsrobjects
  WHERE OBJECTNAME='ORDER';
```

リスト 15.34 – DB2 Catalog 表から XML スキーマ情報を取得する方法

15.4.2 XML スキーマによる妥当性検査

XML スキーマを DB2 に登録した後は、以下の 2 つの方法で XML 文書の妥当性検査を行えます。

- INSERT のなかで XMLVALIDATE 関数を使用
- BEFORE トリガーを使用

図 15.12 に記載する例では、図 15.10 の XML 文書の妥当性検査を図 15.11 の XML スキーマに対して行っています。

```
DROP TABLE t1;
CREATE TABLE t1 (po xml);

INSERT INTO t1 VALUES(xmlvalidate(xmlparse(document('<?xml version="1.0" encoding="UTF-8"?>
        <po:PurchaseOrder xmlns:po="http://www.test.com/po">
          <Header>
            <Id>1</Id>
            <date>2004-01-29</date>
            <description>purchase order</description>
            <value>20</value>
            <status>shipped</status>
          </Header>
          <Items>
            <Item>
              <ItemDescription color="red" weight="5">
                    <Name>Widget C</Name>
                    <SKU>1</SKU>
                    <Price>30</Price>
                    <Comment>no comment</Comment>
              </ItemDescription>
              <NumberOrdered>1</NumberOrdered>
            </Item>
          </Items>
          <Customer type="regular">
            <Name>Manoj K Sardana</Name>
            <Address>ring road, bangalore</Address>
            <Phone>918051055109</Phone>
            <email>msardana@in.ibm.com</email>
          </Customer>
        </po:PurchaseOrder>')) ACCORDING TO XMLSCHEMA ID order));
```

図 15.12 – XMLVALIDATE を使用した XML スキーマによる妥当性検査

CHECK 制約で「IS VALIDATED」述部を使用することで、XML 文書が妥当性検査済みであるかどうかをテストすることができます。

1 つの列内の XML 文書の妥当性検査を異なる XML スキーマに対して行うことも可能です。これは、XML スキーマをバージョン 1 からバージョン 2 に容易にマイグレーションする上で重要な機能となります。また同じ XML 列内に、妥当性検査をまったく行わない XML 文書があることもあります。

信頼できるソースと信頼できないソースの両方から文書を受け取る場合、スキーマによる妥当性検査は後者にのみ必要となるので、このように同じ列に妥当性検査の対象となる文書と対象でない文書を混在させられると便利です。

15.4.3 その他の XML サポート

現在、小規模な XML 文書を基本表に埋め込めるようになっています。つまり、XML データをリレーショナル・データと同じ場所に保管し、通常のリレーショナル・データと同じ圧縮メカニズムを利用できるようになったということです。大規模な XML 文書については、別の内部オブジェクトに保管して圧縮することができます。

DB2 は XML スキーマの進化にも対応します。これは、XML スキーマが変更されたとしても、UPDATE XMLSCHEMA コマンドを使用して簡単に XML スキーマを更新できることを意味します。ただし、XML スキーマにかなり大幅な変更が加えられている場合には、エラーが発生する可能性があります。

DB2 では XML の分解、つまり「シュレッド」もサポートされています。シュレッドはデータベースに XML を保管するための「古い」手法で、他のベンダーではこの手法を使って XML を保管しています。DB2 は必要に応じてこの手法を引き続きサポートしますが、推奨されるのは pureXML です。DB2 がサポートしている XML Extender も、この古い XML 保管手法を使用していますが、このエクステンダーの拡張は今後予定されていません。

DB2 9.7 では、pureXML に伴うあらゆる利点がデータベース・パーティションにも拡張されています。データベース・パーティションはデータウェアハウスに一般的に使用されている手法です。Database Partitioning Feature (DPF) は、DB2 Enterprise Edition で提供されます。

15.6 まとめ

この章では、XML および pureXML 技術について紹介しました。Web 2.0 のツールと手法、そして SOA により、XML 文書が使用される機会は急激に増えています。XML 文書を DB2 データベースに保管すれば、DB2 ならではのセキュリティー、パフォーマンス、そしてコーディングの柔軟性を、pureXML を介して利用できるようになります。pureXML は、XML 文書を構文解析後の階層フォーマットの XML 文書をツリーとして保管できするようにする技術です。しかも、この処理は XML 文書をデータベースに挿入する時点で行われます。クエリーを実行する際に、XML 文書を構文解析してツリーを構築する必要はありません。XML 文書のツリーはすでに構築されて、データベースに保管されているからです。その上、pureXML 技術は Xquery を理解するネイティブ XML エンジンを使用するため、他の RDBMS 製品のように XQuery を SQL にマッピングする必要もありません。

この章では、SQL/XML および XQuery を使用して XML 文書を挿入、削除、更新、照会する方法を説明し、XML 索引、XML スキーマ、さらに圧縮や XML スキーマの進化への対応などのフィーチャーについても取り上げました。

15.7 演習　データベースでのXMLの使用

この章全体をとおして、SQL/XML および XQuery 構文の例をいくつか見てきました。また、DB2 コマンド・エディターと IBM Data Studio についても紹介しました。この演習で、SQL/XML と XQuery の知識をテストすると同時に、これらのツールを実際に使用してみてください。ここで使用する **mydb** データベースは、この章で説明した **table_creation.txt** スクリプト・ファイル (リスト 15.5) を使って作成します。

手順

1. この章で説明した手順に従って、**mydb** データベースを作成し、XML データをロードします。table_creation.txt ファイルは、第 2 章のフォルダー内に置かれた付属ファイル、**Expressc_book_exercises_9.7.zip** に含まれています。DB2 コマンド・ウィンドウまたは Linux シェルから、以下のコマンドによって table_creation.txt スクリプト・ファイルを実行します。

   ```
   db2 -tvf table_creation.txt
   ```

2. いずれかの手順でスクリプトの実行に失敗した場合は、エラー・メッセージを調べて問題を突き止めます。スクリプトの実行中に発生する問題の典型的な原因は、ファイルのパスにあります。ファイルが別のディレクトリーに置かれている場合は、ファイルのパスを変更しなければなりません。任意の時点でデータベースをドロップし、始めからやり直すことができます。その場合には、DB2 コマンド・ウィンドウまたは Linux シェルから以下のコマンドを実行します。

   ```
   db2 drop database mydb
   ```

2. データベースをドロップしようとしたときに、データベースの接続がアクティブであったためにエラーが発生した場合は、以下のコマンドを最初に実行します。

   ```
   db2 force applications all
   ```

3. スクリプトの実行が正常に完了したら、DB2「コントロール・センター」または IBM Data Studio を使用して、**items** 表と **clients** 表が作成されていること、それぞれの表に 4 つの行、7 つの行が含まれていることを確認します。

4. **mydb** データベースが作成され、2 つの表がロードされたら、データベースに接続してリスト 15.7 から 15.19 までのクエリーを実行します。

付録 A – トラブルシューティング

この付録では、DB2 の操作中に発生する可能性のある問題を、どのようにトラブルシューティングするかについて説明します。図 A.1 に、問題が発生した場合に実行するアクションの概要を示します。

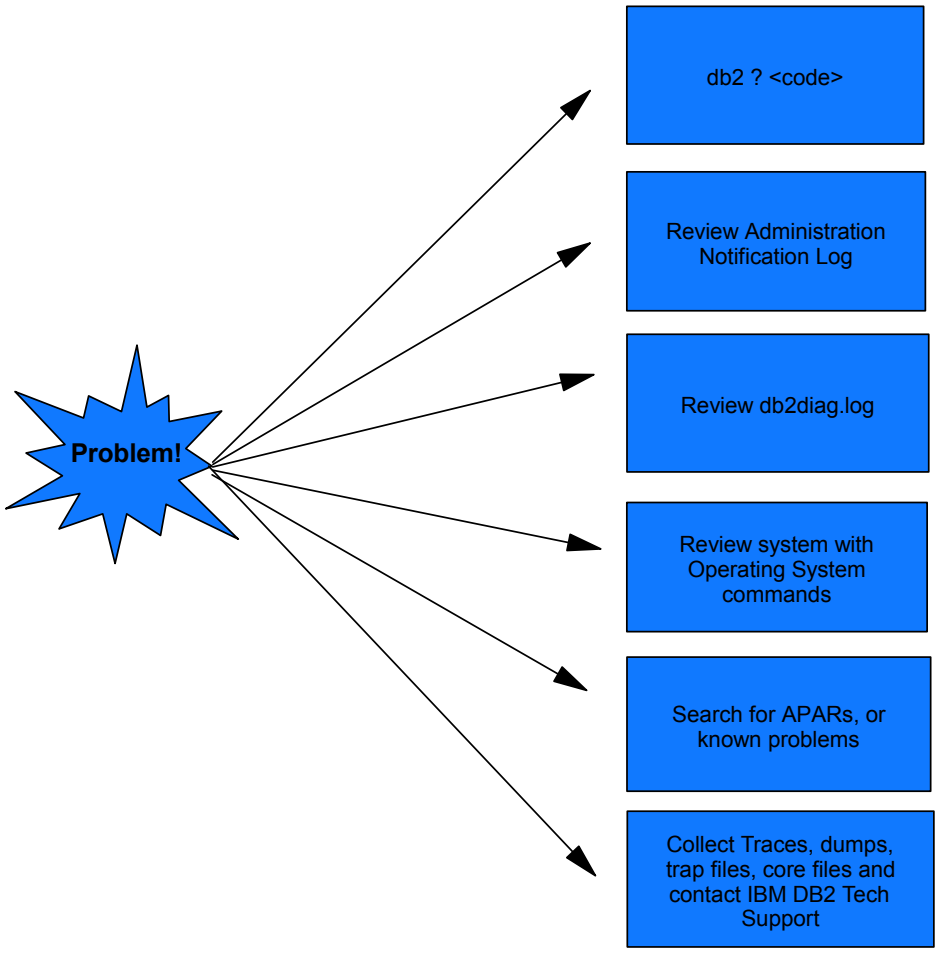


図 A.1 – トラブルシューティングの概要

A.1 エラー・コードの詳細確認

受け取ったエラー・コードについての詳細を表示するには、コマンド・エディターの入力域に、疑問符を先頭に付けたエラー・コードを入力し、「実行」ボタンをクリックします。図 A.2 に、一例を示します。

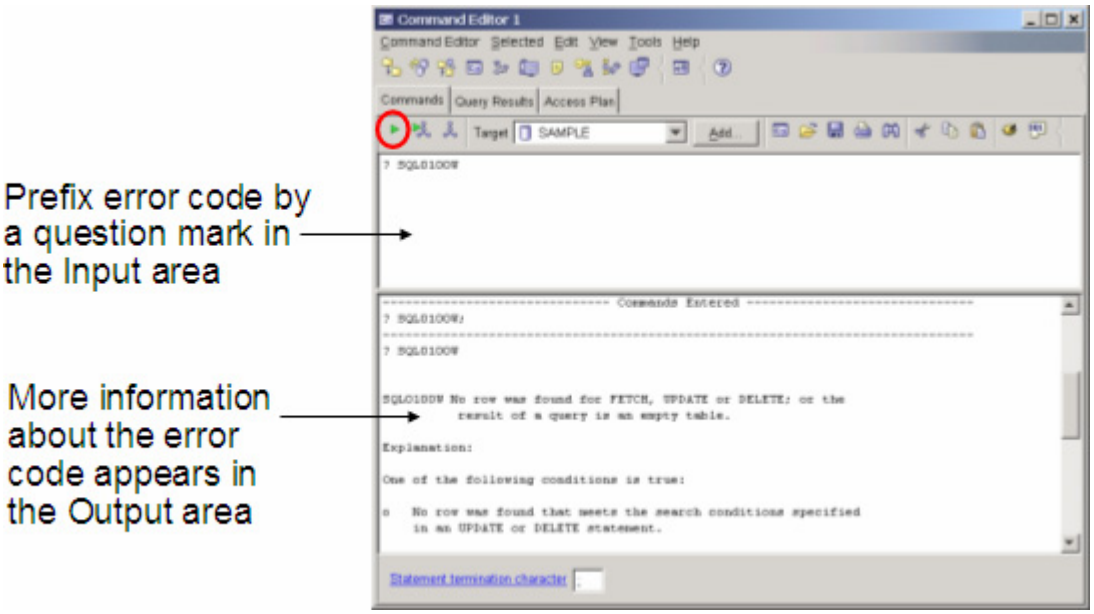


図 A.2 – DB2 エラー・コードの詳細確認

疑問符 (?) によって DB2 ヘルプ・コマンドが呼び出されます。例えば SQL エラー・コード「-104」を受け取った場合、ヘルプ・コマンドを呼び出すには以下の方法があります。これらの例はいずれも同じ結果が得られます。

```
db2 ? SQL0104N
db2 ? SQL104N
db2 ? SQL-0104
db2 ? SQL-104
db2 ? SQL-104N
```

A.2 SQLCODE および SQLSTATE

SQLCODE は、各 SQL ステートメントが実行されるたびに受け取るコードです。以下に、値の意味を要約します。

- SQLCODE = 0; コマンドは成功しました。
- SQLCODE > 0; コマンドは成功しましたが、警告が返されました。
- SQLCODE < 0; コマンドは失敗し、エラーが返されました。

SQLSTATE は、ISO/ANSI SQL92 標準規格に準拠した 5 文字のストリングです。最初の 2 文字は、SQLSTATE クラス・コードとして知られています。

- クラス・コードが 00 の場合、コマンドが成功したことを意味します。
- クラス・コードが 01 の場合、警告を意味します。
- クラス・コードが 02 の場合、条件が見つからなかったことを意味します。
- その他すべてのクラス・コードはエラーとみなされます。

A.3 DB2 管理用通知ログ

DB2 管理用通知ログは、障害点でのエラーに関する診断情報を提供します。Linux および UNIX プラットフォームの場合、管理用通知ログは <instance name>.nfy という名前のテキスト・ファイルです (例えば、「db2inst.nfy」)。Windows では、管理用通知メッセージはすべて、Windows イベント・ログに書き込まれます。

管理者は DBM 構成パラメーター **notifylevel** で、記録する情報のレベルを指定することができます。指定できる値は以下のとおりです。

- 0 -- 管理用通知メッセージを取り込まない (推奨されません)
- 1 -- 致命的エラー、またはリカバリー不能エラー
- 2 -- 即時アクションが必要なエラー
- 3 -- 重要な情報。ただし即時アクションは不要 (デフォルト)
- 4 -- 情報メッセージ

A.4 db2diag.log

db2diag.log は、DB2 管理用通知ログよりも詳細な情報を提供します。このログを使用するのは通常、IBM DB2 技術サポートまたは熟練した DBA のみです。db2diag.log には以下の情報が含まれます。

- エラーをレポートしている DB2 コードのロケーション
- アプリケーション ID。この ID により、サーバーとクライアント両方の db2diag.log で、アプリケーションに関するエントリーを照合することができます。
- エラーの原因を説明する診断メッセージ (「DIA」で始まるメッセージ)
- 利用可能なすべてのサポート・データ (SQLCA データ構造、追加ダンプやトラップ・ファイルのロケーションへのポインターなど)

Windows (Vista 以外) では、db2diag.log はデフォルトで以下のディレクトリーに置かれます。

```
C:\Documents and Settings\All Users\Application
Data\IBM\DB2\DB2COPY1\<instance name>
```

Windows Vista では、db2diag.log はデフォルトで以下のディレクトリーに置かれます。

```
C:\ProgramData\IBM\DB2\DB2COPY1\<instance name>
```

Linux/UNIX では、db2diag.log はデフォルトで以下のディレクトリーに置かれます。

```
/home/<instance_owner>/sqllib/db2dump
```

診断テキストの詳細度は、dbm cfg 構成パラメーター DIAGLEVEL によって決まります。このパラメーターの値の範囲は 0 から 4 で、0 が最低の詳細度、4 が最高の詳細度となります。デフォルト・レベルは 3 です。

A.5 CLI トレース

CLI、Java、PHP、および Ruby on Rails アプリケーションの場合、CLI トレース機能を有効にしてアプリケーションのトラブルシューティングを行うことができます。この機能を有効にするには、アプリケーションを実行しているサーバーの db2cli.ini ファイルを変更します。db2cli.ini ファイルに含まれる標準的なエントリーを以下のリスト A.1 に示します。

```
[common]
trace=0
tracerefreshinterval=300
tracepathname=/path/to/writeable/directory
traceflush=1
```

リスト A.1 – CLI トレースを有効にするための db2cli.ini ファイルに含まれるエントリー

詳細なトレース (db2trc) も使用できますが、一般的にこれが役立つのは DB2 技術サポートのみです。

A.6 DB2 の欠陥と修正

場合によっては、DB2 の欠陥によって問題が発生することがあります。IBM では定期的に、欠陥に対処するコード修正が含まれる fix pack (APAR) をリリースしています。fix pack の資料には、その fix pack に含まれるすべての修正のリストが含まれています。新しいアプリケーションを開発するときには、最新の修正が確実に適用されるようにするため、最新の fix pack を使用することをお薦めします。現行バージョンと fix pack のレベルを確認するには、コントロール・センターの**「ヘルプ」**メニューから**「情報」**を選択するか、コマンド・ウィンドウで **`db2level`** と入力します。ただし DB2 Express-C では、fix pack および正式な IBM DB2 技術サポートは提供されませんのでご注意ください。DB2 Express-C での修正は、fix pack を適用する形ではなく、イメージそのものに組み込まれる形で提供されます。

B

付録 B – 参考文献およびその他のリソース

B.1 参考文献

[1] ZIKOPOULOS, P. 著、「*IBM® DB2® Universal Database™ and the Microsoft® Excel Application Developer… for Beginners*」、dbazine.com article,、2005 年 4 月、
http://www.dbazine.com/db2/db2-disarticles/zikopoulos15

[2] ZIKOPOULOS, P. 著、「*DB2 9 and Microsoft Access 2007 Part 1: Getting the Data...,*」、Database Journal article、2008 年 5 月、
http://www.databasejournal.com/features/db2/article.php/3741221

[3] BHOGAL, K. 著、「*Use Microsoft Access to interact with your DB2 data*」、developerWorks article、2006.年 5 月、http://www.ibm.com/developerworks/db2/library/techarticle/dm-0605bhogal/

[4] SARACCO, C. 他著、IBM Redbooks、「*DB2 9: pureXML overview and fast start*」

2006 年 7 月、http://www.redbooks.ibm.com/abstracts/sg247298.html

B.2 Web サイト:

1. DB2 Express-C Web サイト: www.ibm.com/db2/express

   この Web サイトから、DB2 Express-C サーバーのイメージ、DB2 クライアント、DB2 ドライバー、マニュアルをダウンロードしてください。また、チームのブログにアクセスしたり、メーリング・リストにサインアップすることもできます。

2. DB2 Express フォーラム:
   www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19

   マニュアルを調べても問題を解決できない場合には、フォーラムに質問を投稿してください。

3. DB2 インフォメーション・センター
   http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

   インフォメーション・センターからオンライン・マニュアルにアクセスすることができます。ここが、最新の情報源です。DB2 の各バージョンには、対応するインフォメーション・センターがあります。

   - DB2 9.1: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp
   - DB2 9.5: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp
   - DB2 9.7: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

4. developerWorks: http://www.ibm.com/developerworks/db2

   開発者と DBA にとって貴重なリソースとなるこの Web サイトからは、最新の記事、チュートリアルなどに無料でアクセスできます。

5. alphaWorks: http://www.alphaworks.ibm.com/

   この Web サイトから、これから公開される予定の IBM の新しい技術に直接アクセスすることができます。ここでは、IBM Research による最新技術を調べることができます。

6. planetDB2: www.planetDB2.com

   多数の寄稿者からの DB2 に関するブログ投稿を集めるブログ・アグリゲーターです。

7. DB2 技術サポート: http://www.ibm.com/software/data/db2/support/db2_9/

   欠陥および問題のレポート (APAR) や、その他の技術情報を調べられます。

8. ChannelDB2: http://www.ChannelDB2.com/

   ChannelDB2 は、DB2 コミュニティーのためのソーシャル・ネットワークです。ここには、Linux、UNIX、Windows、z/OS、i5/OS 対応 DB2 関連のビデオ、デモ、ポッドキャスト、ブログ、ディスカッション、リソースなどが揃っています。

B.3 書籍

1. 無料の Redbooks: 「DB2 Express-C: The Developer Handbook for XML, PHP, C/C++, Java, and .NET」

   Whei-Jen Chen、John Chun、Naomi Ngan、Rakesh Ranjan、Manoj K. Sardana 共著、

   2006 年 8 月、SG24-7301-00

   http://www.redbooks.ibm.com/abstracts/sg247301.html?Open

2. 無料の Redbooks: 「DB2 pureXML Guide」

   Whei-Jen Chen、Art Sammartino、Dobromir Goutev、Felicity Hendricks、Ippei Komi、Ming-Pang Wei、Rav Ahuja、Matthias Nicola 共著、

   2007 年 8 月

   http://www.redbooks.ibm.com/abstracts/sg247315.html?Open

3. 無料の Redbooks: 「Developing PHP Applications for IBM Data Servers」

   Whei-Jen Chen、Holger Kirstein、Daniel Krook、Kiran H Nair、Piotr Pietrzak

   2006 年 5 月、SG24-7218-00

   http://www.redbooks.ibm.com/abstracts/sg247218.html?Open

4. 『Understanding DB2: Learning Visually with Examples V9.5』

   Raul F. Chong その他による共著、2008 年 1 月

   ISBN-10: 0131580183

5. 『DB2® SQL PL: Essential Guide for DB2® UDB on Linux™, UNIX®, Windows™, i5/OS™, and z/OS®』(第 2 版)

   Zamil Janmohamed、Clara Liu、Drew Bradstock、Raul Chong、Michael Gao、Fraser McArthur、Paul Yip 共著

   ISBN: 0-13-100772-6

6. 「DB2 9: pureXML overview and fast start」Cynthia M. Saracco、Don Chamberlin、Rav Ahuja 共著、2006 年 6 月 SG24-7298

   http://www.redbooks.ibm.com/abstracts/sg247298.html?Open

7. 『Information on Demand: Introduction to DB2 9 New Features』

Paul Zikopoulos、George Baklarz、Chris Eaton、Leon Katsnelson 共著

ISBN-10: 0071487832

ISBN-13: 978-0071487832

B.4 連絡先 E メール・アドレス

DB2 Express-C メールボックス (管理に関する質問の場合): db2x@ca.ibm.com

DB2 on Campus プログラム メールボックス: db2univ@ca.ibm.com

DB2 9.7 は至って簡単に使い始めることができます。本書は以下の目的で作成されています。

- **無料のエディション、Express-C を使用して DB2 を理解する**
- **DB2 のアーキテクチャー、ツール、セキュリティーを理解する**
- **DB2 データベースを管理する方法を学ぶ**
- **SQL、XQuery、ストアード・プロシージャーを作成する**
- **DB2 のデータベース・アプリケーションを開発する**
- **実践演習を使用して学習する**

IBM の DB2 Express-C は、リレーショナル・データと XML データの両方を容易に管理するための DB2 データ・サーバーの無償のエディションです。無償ということは、つまり DB2 Express-C を無料でダウンロードし、無料でアプリケーションを作成し、無料で本番環境にデプロイできるということです。また、独自のソリューションと統合して配布するとしても料金は発生しません。さらに、DB2 ではデータベースのサイズ、データベースの数、ユーザーの数に関しても一切、制限を設けていません。

DB2 Express-C は Windows および Linux システムで稼働し、C/C++、Java、.NET、PHP、Perl、Ruby などの多種多様なプログラミング言語に対応したアプリケーション・ドライバーを提供します。オプションで、サポートおよび追加機能を利用できる低価格のサブスクリプションを購入することもできます。さらにスケーラビリティーを高めたり、高度な機能を使用する必要が出てきた場合は、DB2 Express-C で作成したアプリケーションを、Workgroup や Enterprise などの他の DB2 エディションにシームレスにデプロイすることができます。

この無料の DB2 エディションは、開発者やコンサルタント、ISV、DBA、そして学生など、データベース・アプリケーションを開発、テスト、デプロイ、または配布しようとしている誰にとっても理想的な選択肢です。ますます大きく成長している DB2 Express-C ユーザー・コミュニティーに今すぐ参加して、DB2 Express-C を試してみてください。そして、次世代のアプリケーションを作成し、革新的ソリューションを実現する方法の探求を始めてください。

DB2 Express-C の詳細を調べたり、DB2 Express-C をダウンロードしたりするには、以下のサイトにアクセスしてください。

ibm.com/db2/express

DB2 コミュニティーと交流したり、DB2 関連のビデオおよびブログを見たりするには、以下のサイトにアクセスしてください。

ChannelDB2.com

本書の翻訳ならびに発行を支えてくださった以下の方々に感謝します。
・早稲田大学　理工学術院　教授　　村岡 洋一様
・早稲田大学　理工学術院　　福盛 秀雄様
三宅 佑治様
佐藤 博志様
芳賀 優樹様
・早稲田情報技術研究所　社長　　加藤 浩一様

［訳者・監訳者一覧］

日本アイ・ビー・エム株式会社

ソフトウェア事業　　大沼 啓希
平野 一路
ISV & デベロッパー事業推進 テクノロジー・チーム

グローバル・テクノロジー・サービス　　竹尾　さつみ

DB2 Express-C入門 V9.7対応
2010年3月31日 第二版発行
日本アイ・ビー・エム株式会社