FUJITSU

PowerRDBconnector クライアントパッケージ for NetCOBOL V3.1 L20
PowerRDBconnector サーバパッケージ for NetCOBOL V3.1 L20

PowerRDBconnector説明書 SQL Server編

Windows/Windows(64)

B1FW-5927-02Z0
2012年5月

まえがき

製品の呼び名について

本書に記載されている製品の名称を、以下のように略して表記します。オペレーティングシステムを総称して、OSと記載します。データベース製品を総称して、データベースと記載します。

- 以下の製品を、「PowerRDBconnector」と記載します。
  - PowerRDBconnector サーバパッケージ for NetCOBOL
  - PowerRDBconnector クライアントパッケージ for NetCOBOL
  - PowerRDBconnector サーバパッケージ for NetCOBOL (64bit)
- 以下の製品を、「Windows」または、「Windows XP」と記載します。
  - Microsoft(R) Windows(R) XP Professional Edition
- 以下の製品を、「Windows」または、「Windows Vista」と記載します。
  - Windows Vista(R) Business
  - Windows Vista(R) Enterprise
  - Windows Vista(R) Ultimate
- 以下の製品を、「Windows」または、「Windows 7」と記載します。
  - Windows(R) 7 Professional
  - Windows(R) 7 Enterprise
  - Windows(R) 7 Ultimate
  - Windows(R) 7 Professional(x64)
  - Windows(R) 7 Enterprise(x64)
  - Windows(R) 7 Ultimate(x64)
- 以下の製品を、「Windows」または、「Windows Server 2003」と記載します。
  - Microsoft(R) Windows Server(R) 2003, Standard Edition
  - Microsoft(R) Windows Server(R) 2003, Enterprise Edition
  - Microsoft(R) Windows Server(R) 2003 R2, Standard Edition
  - Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition
- 以下の製品を、「Windows」または、「Windows Server 2003(x64)」と記載します。
  - Microsoft(R) Windows Server(R) 2003, Standard x64 Edition
  - Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition
  - Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition
  - Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition
- 以下の製品を、「Windows」または、「Windows Server 2008」と記載します。
  - Microsoft(R) Windows Server(R) 2008 Foundation
  - Microsoft(R) Windows Server(R) 2008 Standard
  - Microsoft(R) Windows Server(R) 2008 Enterprise

・ 以下の製品を、「Windows」または、「Windows Server 2008(x64)」と記載します。

　－ Microsoft(R) Windows Server(R) 2008 Foundation (x64)

　－ Microsoft(R) Windows Server(R) 2008 Standard (x64)

　－ Microsoft(R) Windows Server(R) 2008 Enterprise (x64)

・ 以下の製品を、「Windows」または、「Windows Server 2008 R2」と記載します。

　－ Microsoft(R) Windows Server(R) 2008 R2 Foundation

　－ Microsoft(R) Windows Server(R) 2008 R2 Standard

　－ Microsoft(R) Windows Server(R) 2008 R2 Enterprise

・ 以下の製品を、「Solaris」と記載します。

　－ Oracle Solaris (TM) 9 オペレーティングシステム

　－ Oracle Solaris (TM) 10 オペレーティングシステム

・ 以下の製品を、「SQL Server」または、「SQL Server 2005」と記載します。

　－ Microsoft(R) SQL Server(R) 2005 Workgroup Edition

　－ Microsoft(R) SQL Server(R) 2005 Standard Edition

　－ Microsoft(R) SQL Server(R) 2005 Enterprise Edition

　－ Microsoft(R) SQL Server(R) 2005 Developer Edition

　－ Microsoft(R) SQL Server(R) 2005 Standard Edition x64 Extended

　－ Microsoft(R) SQL Server(R) 2005 Enterprise Edition x64 Extended

　－ Microsoft(R) SQL Server(R) 2005 Developer Edition x64 Extended

・ 以下の製品を、「SQL Server」または、「SQL Server 2008」と記載します。

　－ Microsoft(R) SQL Server(R) 2008 Workgroup(x64を含む)

　－ Microsoft(R) SQL Server(R) 2008 Standard(x64を含む)

　－ Microsoft(R) SQL Server(R) 2008 Enterprise(x64を含む)

　－ Microsoft(R) SQL Server(R) 2008 Developer(x64を含む)

・ 以下の製品を、「SQL Server」または、「SQL Server 2008 R2」と記載します。

　－ Microsoft(R) SQL Server(R) 2008 R2 Enterprise(x64を含む)

　－ Microsoft(R) SQL Server(R) 2008 R2 Standard(x64を含む)

　－ Microsoft(R) SQL Server(R) 2008 R2 Workgroup(x64を含む)

　－ Microsoft(R) SQL Server(R) 2008 R2 Developer(x64を含む)

・ 以下の製品を、「Oracle」または、「Oracle10g」と記載します。

　－ Oracle Database Enterprise Edition R10.1.0 / R10.2.0

　－ Oracle Database Standard Edition R10.1.0 / R10.2.0

　－ Oracle Database Standard Edition One R10.1.0 / R10.2.0

　－ Oracle Database Personal Edition R10.1.0 / R10.2.0

　－ Oracle Database Enterprise Edition R10.2.0 (x64)

　－ Oracle Database Standard Edition R10.2.0 (x64)

・ 以下の製品を、「Oracle」または、「Oracle11g」と記載します。

　－ Oracle Database Enterprise Edition R11.1.0 / R11.2.0

- Oracle Database Standard Edition R11.1.0 / R11.2.0
- Oracle Database Standard Edition One R11.1.0 / R11.2.0
- Oracle Database Personal Edition R11.1.0 / R11.2.0
- Oracle Database Enterprise Edition R11.1.0 (x64)/ R11.2.0 (x64)
- Oracle Database Standard Edition R11.1.0 (x64)/ R11.2.0 (x64)

・ 以下の製品を、「NetCOBOL」または、「NetCOBOL for Windows」と記載します。

- NetCOBOL Base Edition for Windows V7.0
- NetCOBOL Standard Edition for Windows V7.0
- NetCOBOL Professional Edition for Windows V7.0
- NetCOBOL 開発パッケージ for Windows V7.2/V8.0
- NetCOBOL Base Edition 開発パッケージ for Windows V7.2/V8.0
- NetCOBOL Standard Edition 開発パッケージ for Windows V7.2/V8.0
- NetCOBOL Professional Edition 開発パッケージ for Windows V7.2/V8.0
- NetCOBOL Base Edition サーバ運用パッケージ for Windows V7.0/V7.2/V8.0
- NetCOBOL Standard Edition サーバ運用パッケージ for Windows V7.0/V7.2/V8.0
- NetCOBOL クライアント運用パッケージ for Windows V7.0/V7.2/V8.0
- NetCOBOL Base Edition クライアント運用パッケージ for Windows V7.0/V7.2/V8.0
- NetCOBOL Standard Edition クライアント運用パッケージ for Windows V7.0/V7.2/V8.0
- NetCOBOL Base Edition 開発パッケージ V9.0/V10（注）
- NetCOBOL Standard Edition 開発パッケージ V9.0/V10（注）
- NetCOBOL Professional Edition 開発パッケージ V9.0/V10（注）
- NetCOBOL Base Edition サーバ運用パッケージ V9.0/V10（注）
- NetCOBOL Standard Edition サーバ運用パッケージ V9.0/V10（注）
- NetCOBOL クライアント運用パッケージ V9.0/V10（注）
- NetCOBOL Base Edition クライアント運用パッケージ V9.0/V10（注）
- NetCOBOL Standard Edition クライアント運用パッケージ V9.0/V10（注）
- NetCOBOL Base Edition サーバ運用パッケージ (64bit) V10（注）
- NetCOBOL Base Edition 開発パッケージ (64bit) V10（注）
- NetCOBOL Standard Edition サーバ運用パッケージ (64bit) V10（注）
- NetCOBOL Standard Edition 開発パッケージ (64bit) V10（注）

・ 以下の製品を、「NetCOBOL」または、「NetCOBOL for .NET」と記載します。

- NetCOBOL Base Edition for .NET V2.0
- NetCOBOL Standard Edition for .NET V2.0
- NetCOBOL Base Edition 開発パッケージ for .NET V2.1/V3.0/V3.1/V4.0/V4.1
- NetCOBOL Standard Edition 開発パッケージ for .NET V2.1/V3.0/V3.1/V4.0/V4.1
- NetCOBOL Enterprise Edition 開発パッケージ for .NET V3.0/V3.1/V4.0～V4.2
- NetCOBOL Base Edition サーバ運用パッケージ for .NET V2.0/V2.1/V3.0/V3.1/V4.0/V4.1
- NetCOBOL Standard Edition サーバ運用パッケージ for .NET V2.0/V2.1/V3.0/V3.1/V4.0/V4.1

－ NetCOBOL Enterprise Edition サーバ運用パッケージ for .NET V3.0/V3.1/ V4.0～V4.2

－ NetCOBOL Base Edition クライアント運用パッケージ for .NET V2.0/V2.1/V3.0/V3.1/V4.0/V4.1

－ NetCOBOL Standard Edition クライアント運用パッケージ for .NET V2.0/V2.1/V3.0/V3.1/V4.0/V4.1

（注）

製品名に、for Windows と付いていませんが、「NetCOBOL for .NET」と区別するときに、本書では、「NetCOBOL for Windows」と略して表記します。

PowerRDBconnector説明書の体系

PowerRDBconnectorのマニュアルは、以下の3冊から構成されています。

・「PowerRDBconnector説明書 SQL Server編」

・「PowerRDBconnector説明書 Oracle編」

・「PowerRDBconnector 動作環境ひな型作成ツール 操作手引書」

本書は、以下について説明しています。

・ PowerRDBconnector サーバパッケージ for NetCOBOL

・ PowerRDBconnector クライアントパッケージ for NetCOBOL

・ PowerRDBconnector サーバパッケージ for NetCOBOL(64bit)

・ SQL Serverとの接続

上記以外の説明は、対応するマニュアルを参照してください。

本書の目的

本書は、PowerRDBconnector の製品概要および環境設定から運用管理について説明しています。

本書の読者

本書は、PowerRDBconnector の導入、環境設定および運用管理を行う方を対象としています。

本書を読むにあたり、以下の事項について熟知している必要があります。

・ NetCOBOL

・ SQL Server

・ Windows

本書の構成

本書は、以下の章で構成しています。

第1章 導入前に考慮すること

本製品を使用する際に考慮および配慮が必要なことを説明します。

PowerRDBconnectorの導入を判断する前に、業務システムの機能範囲に適合するかを開発者が判断する上で、特に注意すべき観点を確認するときにお読みください。

第2章 PowerRDBconnector とは

PowerRDBconnector の特長と機能について説明しています。

業務システムにおける本製品の機能の適用方法や範囲を明確化するためにお読みください。

第3章 PowerRDBconnector の使用手引き

PowerRDBconnector の環境設定と使用方法について説明しています。

PowerRDBconnectorを使用し、業務アプリケーションの開発、保守を行うために必要な作業方法を説明しています。

第4章 COBOLアプリケーションの開発について

本製品を使用する際の注意事項を説明しています。

業務アプリケーションの開発時に、他のシステムからCOBOLアプリケーションを移行する場合に、特に注意すべきこと、およびCOBOLアプリケーション開発時の注意事項を確認するときにお読みください。

第5章 エラー時の対処

PowerRDBconnector使用時に発生したエラーの原因と対処、およびトレース機能について説明します。

エラーが発生したときにお読みください。

付録 A 他製品のファイル資源

他製品（ファイルシステム、データベース系）のファイル資源との関係、および代替方法について説明します。

開発者が、業務システムの機能・構成設計時にCSP/FXやASPのデータベース資産としてよく使用される論理ファイルとの違いについて確認するときにお読みください。

付録 B トラブルシューティング

PowerRDBconnectorの使用時に発生したトラブルの調査方法、およびトラブル事例について説明します。

トラブルが発生したときにお読みください。

付録 C 開発用サンプル情報

PowerRDBconnectorを使用する上で役立つサンプル集です。

開発者およびシステム管理者が、業務アプリケーションを開発するときに、参考にしてください。

付録 D データベースの相違点

PowerRDBconnectorからSQL Server、Oracleを使ったときの相違点について説明します。

開発者およびシステム管理者が、データベースを変更するときの機能差を確認するときにお読みください。

付録 E 32ビット動作と64ビット動作の相違点

PowerRDBconnectorの動作アーキテクチャーとして、32ビットと64ビットを使ったときの相違点について説明します。

開発者およびシステム管理者が、動作アーキテクチャーを変更するときの機能差を確認するときにお読みください。

付録 F リリース情報

PowerRDBconnectorのリリース情報について説明します。

開発者およびシステム管理者が、前版との機能差を確認するときにお読みください。

本書の読み方

本書は、PowerRDBconnector の使用方法について記載しています。

- NetCOBOLについては、NetCOBOLのマニュアルを参照してください。

・ SQL Serverについては、SQL Server Books Onlineを参照してください。

製品の仕様について

PowerRDBconnectorは、本書の記載範囲で検証しています。本書に記載していないPowerRDBconnector、オペレーティングシステムやデータベースの機能については、PowerRDBconnectorの動作保証はできません。

商標について

・ Microsoft、Windows、Windows Vista、Windows Server、SQL Server、Hyper-Vまたはその他のマイクロソフト製品の名称および製品名は、米国Microsoft Corporationの米国およびその他の国における登録商標または商標です。

・ Oracleは、Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

・ その他の会社名および製品名は、それぞれの会社の商標または登録商標です。

・ その他、本書に記載されているシステム名、製品名などには、必ずしも商標表示(TM、R)を付記していません。

マニュアルの変更点について

本書は、以下のマニュアルを元に作成されています。

・ PowerRDBconnector クライアントパッケージ for NetCOBOL V3.1L10

PowerRDBconnector説明書(SQL Server編)

・ PowerRDBconnector サーバパッケージ for NetCOBOL V3.1L10

PowerRDBconnector説明書(SQL Server編)

前版のマニュアルからの主な追加、変更点は、以下のとおりです。

前版のマニュアルからの主な追加、変更点

No	変更点	変更箇所
1	64ビット動作をサポートしました。	1.7 2.2.1.2 2.3.2 3.1.5 3.1.6 付録E
2	以下のOSをサポートしました。 ・ Windows Server 2008 R2 Foundation ・ Windows Server 2008 R2 Standard ・ Windows Server 2008 R2 Enterprise ・ Windows 7 Professional (x64を含む) ・ Windows 7 Enterprise (x64を含む) ・ Windows 7 Ultimate (x64を含む) 以下のOSの記述を追加しました。 ・ Windows Server 2008 Foundation(x64を含む)	2.3
3	以下のデータベースをサポートしました。 ・ SQL Server 2008 R2 Enterprise(x64を含む)	2.3

No	変更点	変更箇所
	・ SQL Server 2008 R2 Standard(x64を含む) ・ SQL Server 2008 R2 Workgroup(x64を含む) ・ SQL Server 2008 Developer(x64を含む)	
4	以下のNetCOBOLをサポートしました。 ・ NetCOBOL Base Edition 開発パッケージ for .NET V4.1 ・ NetCOBOL Standard Edition 開発パッケージ for .NET V4.1 ・ NetCOBOL Enterprise Edition 開発パッケージ for .NET V4.1/ V4.2 ・ NetCOBOL Base Edition サーバ運用パッケージ for .NET V4.1 ・ NetCOBOL Standard Edition サーバ運用パッケージ for .NET V4.1 ・ NetCOBOL Enterprise Edition サーバ運用パッケージ for .NET V4.1/ V4.2 ・ NetCOBOL Base Edition クライアント運用パッケージ for .NET V4.1 ・ NetCOBOL Standard Edition クライアント運用パッケージ for .NET V4.1	2.3
5	レコード長とテーブルの行長が一致していない場合のエラーについての記事を追加しました。	3.1.3.5
6	NetCOBOLでマルチセッションプログラミングが使用できるようになりました。	2.2.1.2 3.1.6
7	以下が未サポートであることを追記しました。 ・ リンクサーバー ・ Windows Azure、SQL Azure	2.2.1.3 2.3.2.4
8	ServerNameプロパティに、コロン(:)が記述できることを追加しました。	3.1.4
9	データベースの可変長項目をサポートしました。	3.1.3.5 3.7 4.1.2.8
10	英数字項目、日本語項目以外の項目に対応するデータベースの精度を追加しました。	3.1.3.5
11	COBOLと関連付かない項目を指定できるようにしました。	3.1.3.6 4.2.3.3
12	セッション変更サブルーチンをサポートしました。	3.1.5 3.1.6.1 3.2 4.2.4.2
13	ファイルのOPEN前でもトランザクション開始が可能になりました。	3.4 4.1.3.1
14	トランザクション時の注意事項に、以下を追加しました。 ・ タイムアウトエラーやデッドロックエラー発生時のトランザクション状態 ・ トランザクション取消し後のカーソル位置について	4.1.3.1
15	排他制御で、以下の注意事項を追加しました。 ・ レコードロックが、更新、削除時に解放されない。 ・ 解放されなかったレコードロックを解放する方法	3.5.2

No	変更点	変更箇所
16	イベントログにエラーが発生した項目番号が出力されるようになりました。	5.1.1.1
17	“JMP0320I-I/U”の補足情報FDBKの値の詳細を追加しました。	5.1.1.4
18	トレースへの文字列出力サブルーチンをサポートしました。	5.2.3.3
19	デッドロック出口の記述を追加しました。	2.2.1.2 4.2.2.4
20	トラブルシューティングを変更しました。 ・ 変更 — NULL値を含む列を、PowerRDBconnectorから読み込むことはできますか？ — 可変長属性の列があるテーブルの読込みや書込みができますか？ — FLOAT属性の列があるテーブルの読込みや書込みができますか？ — DATE属性やTIMESTAMP属性の列があるテーブルの読込みや書込みができますか？ ・ 追加 — 前バージョンのPowerRDBconnectorの環境は、そのまま新しいバージョンでも使用できますか？ — 32ビット動作のPowerRDBconnectorの環境を、64ビット動作の環境へそのまま移行できますか？ — 32ビット動作のPowerRDBconnectorで指定していたCOBOL初期化ファイルを64ビット動作のPowerRDBconnectorで指定すると、OPEN時にエラーが発生します。 — レコードキーを構成する項目数と、OPEN文の性能に依存関係はありますか？ — レコードを構成する項目数と、READ文やWRITE文の性能に依存関係はありますか？	付録B
21	XMROAUTHサブルーチンのプログラム原型の誤記を修正しました。	付録C
22	「付録D データベースの相違点」、「付録E 32ビット動作と64ビット動作の相違点」を追加しました。	付録D 付録E
23	リリース情報を変更しました。	付録F

高度な安全性が要求される用途への使用について

本製品は、一般事務用、パーソナル用、家庭用、通常の産業等の一般的用途を想定して開発・設計・製造されているものであり、原子力施設における核反応制御、航空機自動飛行制御、航空交通管制、大量輸送システムにおける運行制御、生命維持のための医療用機器、兵器システムにおけるミサイル発射制御など、極めて高度な安全性が要求され、仮に当該安全性が確保されない場合、直接生命・身体に対する重大な危険性を伴う用途（以下「ハイセイフティ用途」という）に使用されるよう開発・設計・製造されたものではありません。

お客さまは本製品を必要な安全性を確保する措置を施すことなくハイセイフティ用途に使用しないでください。また、お客さまがハイセイフティ用途に本製品を使用したことにより発生する、お客様または第三者からのいかなる請求または損害賠償に対しても富士通株式会社およびその関連会社は一切責任を負いかねます。

インターネットでの使用について

本製品はインターネットへのサービスを提供する用途を想定して設計・製造されておりません。インターネットに接続しない環境（ローカルネットワークまたはイントラネット内）で使用するか、インターネットに接続して使用する場合は、運用環境によりセキュリティ侵害対策を構築した上でご使用ください。

2012年5月

目　次

第1章 導入前に考慮すること

本章では、本製品を使用する際に考慮および配慮することを説明します。

この製品をはじめてお使いになる方は、必ずお読みください。

1.1 本製品を使用する前に

本製品は、データベースをファイルシステムと同様にREAD/WRITE文でアクセスするためのものですが、ファイルシステム系製品の機能互換を保証するものではありません。

本製品を使用する上で排他制御、データ構造、性能確保などの考慮が必要です。

例えば、ファイルアクセス系製品とは、データ値の扱いが異なります。データベースでは、文字データ型には文字のみ、数字データ型には数値のみが格納可能です。一方、COBOLではプログラミングの仕方で、数字データ型に文字が設定されることがあります。

このようなデータ操作はできないため、COBOLアプリケーションの修正が必要となります。

詳しくは、「第3章 PowerRDBconnectorの使用手引き」および「第4章 COBOLアプリケーションの開発について」を参照してください。

1.2 本製品で実現できる性能

本製品は、データベース製品が提供するSQLアクセスインターフェースを使用して、COBOLアプリケーションの入出力文を処理します。

そのため実現できる性能は、入出力文を直接サポートしているファイルシステム系製品に対して、入出力文で記述したCOBOLアプリケーションからアクセスした場合と同等の性能を保証するものではなく、入出力文の処理論理を変更せずに、SQLアクセスインターフェースでアクセスした場合と同等となります。

一般的に、SQLアクセスインターフェースを持つデータベース製品は、以下の傾向があります。

- アプリケーションで集合操作の処理を行うと、性能がよくなります。
- アプリケーションで、カーソルを宣言し、1レコードずつ処理すると、集合操作と比較してデータ件数に応じて性能が劣化します。

本製品では、COBOLのアクセス文に合わせ、カーソルを宣言し、1レコードずつ処理しています。このため、本製品を使用したアプリケーションの性能よりも、データベース製品へ直接SQL文でアクセスするアプリケーションの方が高性能となります。

データ検索では、COBOLの索引ファイルのレコードキーに対応するデータベースのインデックスが必要です。

データ更新を行うと、データベースではデータ保証のため、履歴管理されているトランザクションログファイルにも更新が行われます。そのため、履歴管理されないファイルシステム系製品と同等の更新性能は得られません。

データ更新の性能を向上させるには、トランザクションを適用して、複数の更新データをまとめて確定するなどの方法があります。性能向上については、「4.1.4 性能向上のポイント」を参照してください。

1.3 本製品で実現できる排他制御

データベースでは、排他制御の仕組みがファイルシステム系製品と大きく異なります。また、データベース製品によっても排他制御の仕組みが異なります。

このため、使用するデータベースで実現している排他制御の理解が必要です。

本製品の排他制御について、詳しくは「3.5 排他制御」を参照してください。

1.4 対象とする既存システム

本製品は、入出力文で記述した既存システムのCOBOLアプリケーションを、SQL文を使用したCOBOLアプリケーションに書き直さずに、活用することを目的としたものです。

主に、入出力文で記述した以下の既存システムのCOBOLアプリケーションを対象としています。

- 中小規模のシステム
- 既存システムのCOBOLアプリケーションの活用
- CSP/FX(FX-RDB)、ASP(RDB/6000、Symfoware6000)からの移行

1.5 データベース製品の使用における注意

データベース製品を使用するため、データベースを熟知した、高い技術力を持つ技術者が業務を設計してください。

1.6 インターネットでの使用について

本製品はインターネットへのサービスを提供する用途を想定して設計・製造されておりません。このため、以下の環境で使用してください。

- インターネットに接続しない環境（ローカルネットワークまたはイントラネット内）
- インターネットに接続して使用する場合は、VPN、ファイアウォールなど運用環境によりセキュリティ侵害対策を構築した環境

1.7 32ビット動作と64ビット動作について

アプリケーションを32ビット動作にする場合と、64ビット動作にする場合で、選択する製品が異なります。

表1.1 32ビット動作と64ビット動作について

アプリケーション動作	本製品のインストール媒体	使用するNetCOBOL	備考
32ビット動作	PowerRDBconnector	NetCOBOL for Windows	
		NetCOBOL for .NET	
64ビット動作	PowerRDBconnector (64bit)	NetCOBOL (64bit)	NetCOBOL for .NETは使用できません。

本製品は、32ビット動作と64ビット動作では機能が異なる部分がありますので注意してください。詳細は、「付録E 32ビット動作と64ビット動作の相違点」を参照してください。

第2章 PowerRDBconnectorとは

本章では、PowerRDBconnector の機能について説明します。

開発者およびシステム管理者が、業務システムを設計するときに、本製品の機能範囲を理解するためにお読みください。

2.1 PowerRDBconnectorの特長

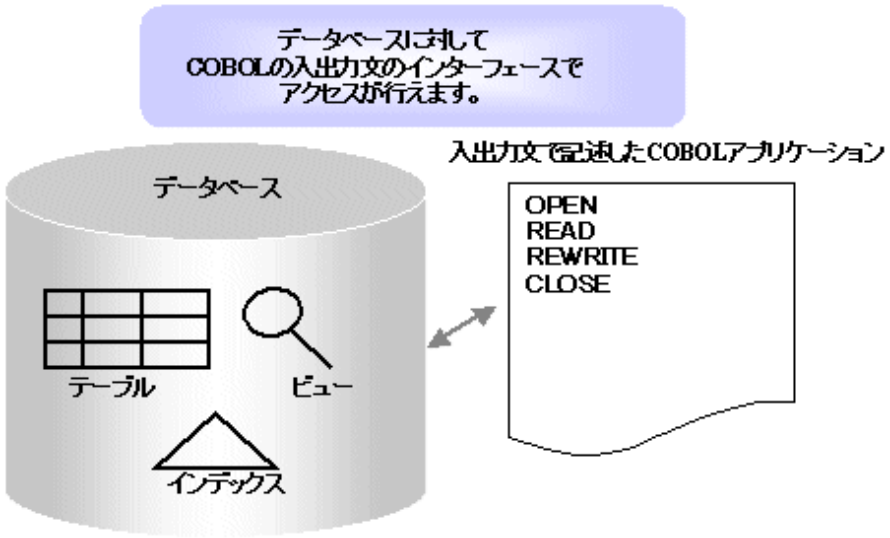
本製品は、データベースをファイルシステムと同様にREAD/WRITE文でアクセスするためのものです。

ここでは、PowerRDBconnector の特長を説明します。

2.1.1 入出力文でのデータベースアクセス

PowerRDBconnector は、COBOLアプリケーションのOPEN、START、READ、REWRITE、WRITE、DELETE、CLOSEといった入出力文で、一般の順ファイルや索引ファイルと同様に、データベースのテーブルまたはビューにアクセスできます。

図2.1 入出力文でのデータベースアクセス



2.1.2 2種類のプログラミングスタイル

本製品は、以下の2つのプログラミングスタイルが可能です。

- シングルセッションプログラミング
- マルチセッションプログラミング

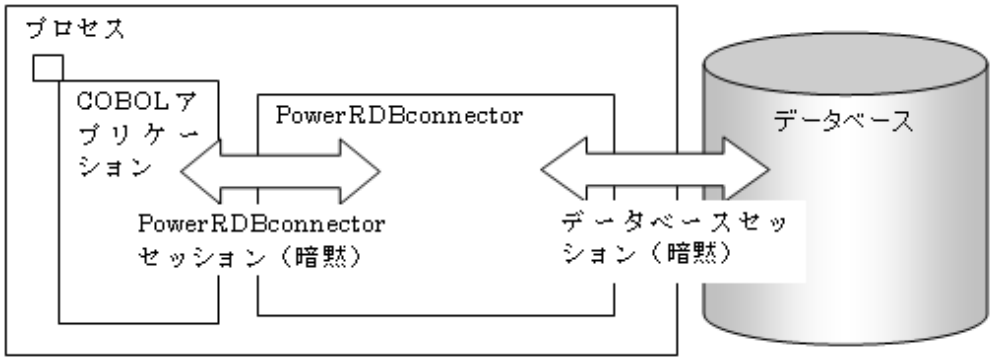
セッションは、「認証、排他制御、トランザクション制御を行う単位」のことです。

2.1.2.1 シングルセッションプログラミング

コンソールアプリケーションや、バッチアプリケーションなどのように、1プロセス上で1つのユーザーアプリケーションを動作させるプログラミングスタイルです。

シングルセッションプログラミングの場合、COBOLアプリケーションとPowerRDBconnectorの間には、暗黙のセッションが1つあるものとみなされ、データベースとのセッションも、プロセス上に1つだけ開設されます。

図2.2 シングルセッションプログラミングの概要



シングルセッションプログラミングとは、PowerRDBconnector V1.0からサポートされているプログラミングスタイルのことです。

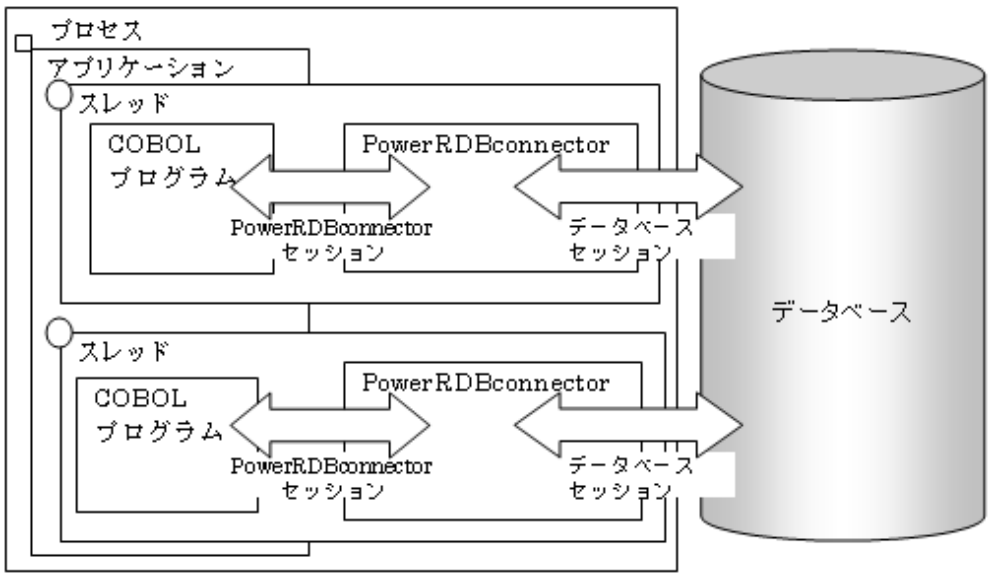
シングルセッションプログラミングの利点については、「表2.4 プログラミングスタイルごとの利点について」を参照してください。

2.1.2.2 マルチセッションプログラミング

マルチスレッドで動作するWebアプリケーションのように、1プロセス上でユーザーアプリケーションを複数動作させるため、アプリケーションでセッションの制御を行うプログラミングスタイルです。

マルチセッションプログラミングの場合、PowerRDBconnectorのセッションと、データベースのセッションは、1対1に対応付けられます。

図2.3 マルチセッションプログラミングの概要



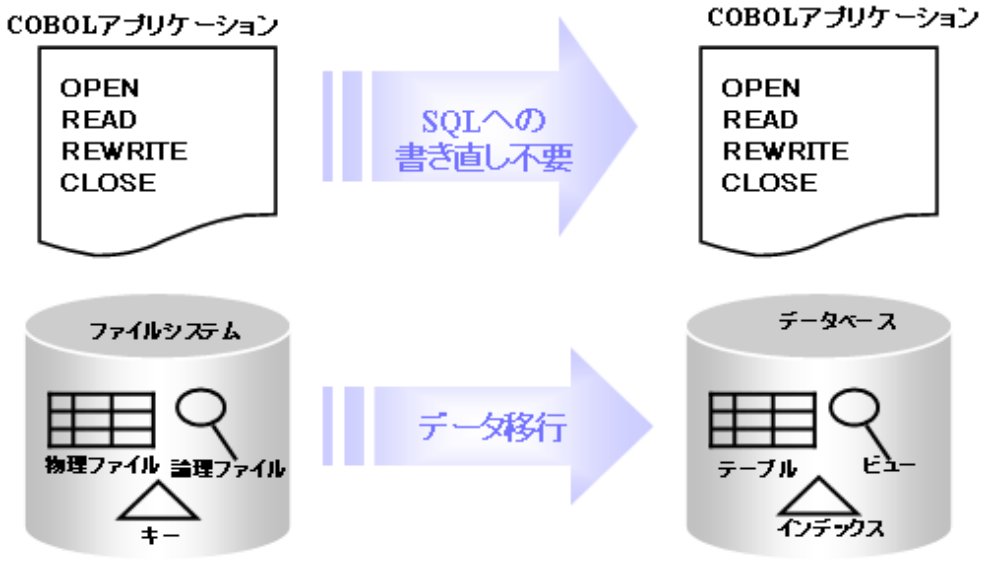
マルチセッションプログラミングの利点については、「表2.4 プログラミングスタイルごとの利点について」を参照してください。

2.1.3 既存COBOL資産の活用

入出力文で記述した従来のCOBOLアプリケーションを、SQL文による記述に設計変更せずに、データベース(テーブル、ビュー)にアクセスできるため、従来のCOBOLアプリケーションのソースコードや開発スキルを活用できます。

例えば、ファイルシステムのデータをデータベースに移行し、入出力文で記述したCOBOLアプリケーションを活用して、データベースにアクセスすることができます。

図2.4 既存COBOL資産の活用



なお、COBOLアプリケーションのプログラミングで考慮すべきポイントが、いくつかあります。詳しくは、「第4章 COBOLアプリケーションの開発について」を参照してください。

2.1.4 データベースを活用したシステム構築

データベースのバックアップ機能、リカバリー機能、クラスタシステムなどを活用して、一般のファイルシステムよりも堅牢性のある業務システムを構築できます。

また、業務に合わせ、COBOL以外の言語で作成したアプリケーションとデータベースを共有することができます。

2.2 機能

本節では、PowerRDBconnector の機能について説明します。

2.2.1 ファイルアクセス機能

COBOLアプリケーションからデータベースのテーブルまたはビューを、ファイル(順ファイル、索引ファイル)としてアクセスできます。

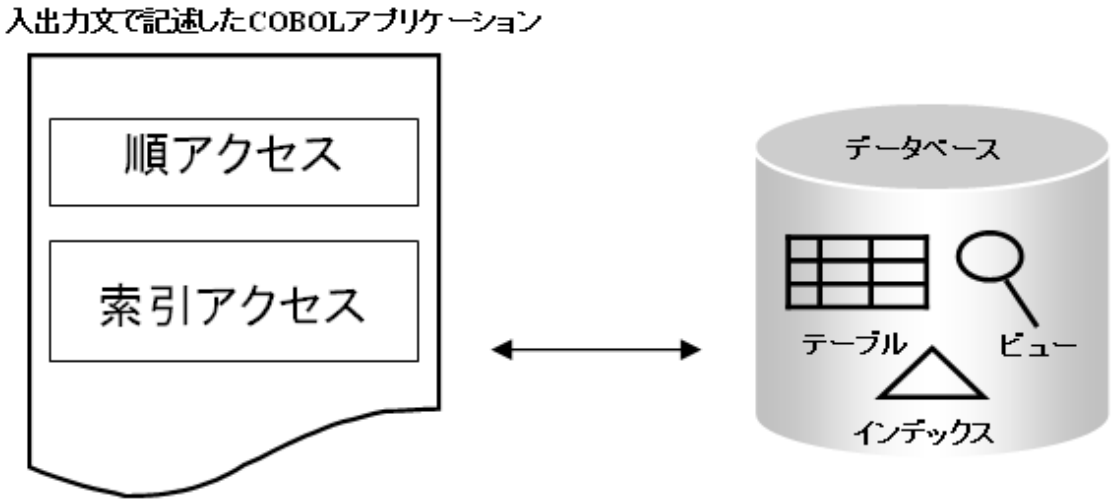
2.2.1.1 ファイルアクセス機能

以下にファイルアクセス機能を示します。

表2.1 ファイルアクセス機能

編成 ORGANIZATION	アクセスモード ACCESS MODE	データベースオブジェクト
SEQUENTIAL（順ファイル）	SEQUENTIAL(順呼出し)	テーブル、ビュー
INDEXED（索引ファイル）	SEQUENTIAL(順呼出し)	テーブル、ビュー、 インデックス
	RANDOM(乱呼出し)	
	DYNAMIC(動的呼出し)	

図2.5 ファイルアクセス機能



ビューへのファイルアクセスには、テーブルへのアクセスと比較していくつかの制約があります。詳細は、「4.2.3.2 ビューの使用について」を参照してください。

2.2.1.2 COBOLアプリケーションから利用できる機能範囲

PowerRDBconnectorを使用してデータベースにアクセスするために使用できるCOBOLの機能範囲を以下に示します。

表2.2 COBOLアプリケーションの機能範囲

項目			利用可否	
			32ビット	64ビット
ファイル	ファイル編成	レコード順ファイル	○	
		相対ファイル	×:エラー(注1)	
		索引ファイル	○	
レコード	レコード形式	固定長レコード形式	○	
		可変長レコード形式	×:エラー	
	レコードの最大長(バイト数)		8,060	
ファイル 管理記述項	SELECT句	OPTIONAL指定	×:無効(注2)	

項目			利用可否	
			32ビット	64ビット
	ASSIGN句 (注3)	ファイル識別名指定	○	
		ファイル識別名定数指定	○	
		データ名指定	○	
	FILE STATUS句	ファイル状態	○(注4)	
	LOCK MODE句	AUTOMATIC	×:無効(注5)	
		EXCLUSIVE		
		MANUAL		
	RECORD KEY句		○	
	ALTERNATE RECORD KEY句	指定できるキーの最大個数	16 (注6)	
	レコード キーの項目	英数字	○	
		日本語	○	
		符号なし外部10進数	○	
		符号付き外部10進数	○	
		符号なし内部10進数	○	
		符号付き内部10進数	○	
		符号なし2進数	○	
		符号付き2進数	○	
		キーの降順評価	×:指定不可	
	多重項目キー		○	
文	OPEN文		○	
	CLOSE文		○	
	READ文		○	
	WRITE文		○	
	REWRITE文		○	
	START文		○	
	DELETE文		○	
プログラミング	セッション	シングルセッション	○	
		マルチセッション	○(注7)	
トランザクション	操作(注8)	開始	○	
		確定	○	
		取消し	○	
	分離レベル		ReadCommitted	
排他制御(注9)	対象	テーブルロック	○	
		レコードロック	○	
	デッドロック出口(注10)		○	
1セッションで最大同時オープン数			128	
1システムでの最大同時開設セッション数			512	

項目	利用可否	
	32ビット	64ビット
1システムで多重実行可能な最大プロセス数	100 (注11)	200 (注11)
最大合計列名長(1つのテーブル/ビューに対する全列名の合計サイズ)	40,960文字(注12)	

○:使用可能

×:使用不可

エラー :エラー終了します。

無効 :指定した動作が行われません。

指定不可:サポートしていないため、指定する方法がありません。

(注1)相対ファイル

COBOLファイルシステムの相対ファイルなどを使用してください。

指定した場合、OPEN文実行時、以下のCOBOLエラーが発生します。

“JMP0001I-U ファイルのオープンに失敗しました.”

(注2)OPTIONAL指定

ファイルが存在しない場合、ファイルなしのエラーとなります。自動生成は行いません。

(注3)ASSIGN句

ASSIGN句はファイル識別名で行い、「3.1.5 COBOL初期化ファイル」に指定することを推奨します。なお、PowerRDBconnector ではファイル識別名を一意の名前で指定してください。

(注4)FILE STATUS句

入出力状態の詳細情報(FILE STATUS 02)は返却しません。指定した場合でも常に初期値が通知されます。

(注5)LOCK MODE句

エラーとなりませんが、LOCK MODE句で指定したロック機構は動作しません。

(注6)ALTERNATE RECORD KEY句

指定できるキーの最大個数は、RECORD KEY句(主キー)を含めて16個です。

(注7)マルチセッション

ASP.NETやInterstageなどと連携するときは、マルチセッションのプログラミングが必要です。

マルチセッションは、以下の環境で使用できます。

ー 以下のNetCOBOLを用いてコンパイルし、かつ

- NetCOBOL for .NET V3.0以降(32ビット動作)
- NetCOBOL for Windows V10.3以降(32ビット動作、64ビット動作)

ー セッションを制御するユーザーアプリケーションを作成した場合

(注8)トランザクション操作

トランザクション操作をするには、PowerRDBconnector が提供するトランザクションサブルーチンをCALL文で呼び出します。トランザクションは、セッション単位に開始、確定および取消しができます。

(注9)排他制御

テーブルロックまたはレコードロックを指定します。
レコードロックは、PowerRDBconnector動作環境ファイルの指定により行います。
テーブルロックは、COBOL初期化ファイルの指定と、PowerRDBconnector動作環境ファイルの指定により行います。テーブルロックの指定方法は、「3.1.4 PowerRDBconnector 動作環境ファイル」、「3.1.5 COBOL初期化ファイル」を参照してください。

(注10)デッドロック出口

NetCOBOL for Windows V10.3以降(32ビット動作、64ビット動作)で、デッドロックが発生した場合の例外処理を行えます。
詳細は、「デッドロック出口」を参照してください。

(注11)1システムで多重実行可能な最大プロセス数

同時実行できる最大多重実行数は、アプリケーションの処理内容(入出力文回数)、データベースおよび使用するハードウェア(CPU、ハードディスク能力、データベースに割り当てる実メモリなどのチューニングパラメーター)の要件に依存します。例えば、バッチプログラムの大量データ処理は、高負荷となるため、多重実行に耐えられません。

この値は、OSやデータベースのチューニングをせず、シングルセッションを用いて、常にデータベースにアクセスし続けるCOBOLアプリケーションを多重に実行して動作した値です。このため、この値は性能も含めて運用可能な多重度を保証するものではありません。

(注12)最大合計列名長

データベースのテーブルやビューを作成する際、列名の長さの合計が、半角文字で40,960文字、全角文字の場合は20,480文字以下にしてください。

2.2.1.3 参考:データベースの機能範囲

ここでは、データベースの機能範囲を入出力文で記述したCOBOLの機能範囲に置き換えて説明します。詳細は、データベースのマニュアルを参照してください。

表2.3 参考:データベースの機能範囲

項目			データベースの機能範囲
ファイル定義ツール			osql、sqlcmd、SQL Server Management Studio ユーティリティなど
ファイル	ファイルの最大サイズ(バイト)		32T
	ファイルの最大数		約20億
	別名		シノニム(注)
	他データベースへのリンク		リンクサーバー(注)
レコード	レコードの最大長(バイト)		8,060
	レコード内の最大項目数(列数)		1,024
	レコードの最大数(行数)		無制限
ファイル管理記述項	RECORD KEY句	指定できる項目(キーを構成する項目)の最大個数	16

項目			データベースの機能範囲
		指定できる項目総長の最大(バイト)	900
	ALTERNATE RECORD KEY句	指定できるキーの最大個数	249
		指定できる項目（キーを構成する項目）の最大個数	16
		指定できる項目総長の最大(バイト)	900
	多重項目キーに定義できる最大列数		16
トランザクション			○
リカバリー			○
バックアップ			○
1テーブルに定義できるインデックス数			249
1セッションの最大同時オープン数			特に記載なし
1システムの最大同時オープン数			特に記載なし

データベースの機能範囲について

データベースのマニュアルをもとに記載しています。詳しくは、データベースが提供している情報を参照してください。

（注）シノニム、リンクサーバー

SQL Serverで定義できますが、PowerRDBconnectorはサポートしていません。

2.2.2 プログラミングスタイル

ここでは、次の2種類のプログラミングスタイルについて説明します。

- シングルセッション
- マルチセッション

2.2.2.1 プロセスとスレッド

Windowsは、実行するプログラムごとにプロセスを割り当て、各プロセスに、メモリ上のプログラムのコード、データ、オープンされているファイル、動的に割り当てられたメモリ資源など、OSの資源を割り当てます。

Windowsは、CPUの実行時間をプロセス内に生成されたスレッド単位に割り当てます。

スレッドとは、CPUに実行時間を割り当てる最小単位です。つまり、1プロセスの内部に多数のスレッドを生成することで、複数のコード（プログラム）を同時に実行することができます。

1つのプロセス内のスレッドは、プロセスの仮想メモリ空間やグローバル変数などを共有します。このため、スレッド間のデータ交換などにかかるCPU負荷が、プロセス間のデータ交換に比べて非常に小さくなります。

Windowsは、プロセス内で複数スレッドを実行しない場合でも、CPU時間の割り当てはスレッド単位になるので、プロセスが起動されると、少なくとも1つのスレッドを起動するようになっています。

2.2.2.2 シングルスレッド（プロセス）プログラムとマルチスレッドプログラム

シングルスレッド（プロセス）プログラムとは、1つのプロセス内で、1つのスレッドを実行するプログラムのことです。

それに対し、マルチスレッドプログラムとは、1つのプロセス内で、複数のスレッド（マルチスレッド）で実行できるプログラムのことです。

2.2.2.3 プログラミングスタイルとは

PowerRDBconnectorを使用する場合、「表2.4 プログラミングスタイルごとの利点について」で示される2つのプログラミングスタイルが選択できます。

表2.4 プログラミングスタイルごとの利点について

プログラミングスタイル	利点	シングルスレッド(プロセス)での動作	マルチスレッドでの動作
シングルセッションプログラミング	・1プロセス上の全てのアクセスをデータベースと1つのセッションで実行します。 ・データベースとのセッションを意識せず、アプリケーションを作成できます。 ・既存COBOL資産のプログラミングスタイルを変更せずに開発できます。	○	×
マルチセッションプログラミング	・マルチスレッドで動作可能なアプリケーションを作成できます。 ・スレッド単位にセッションを保持できます。 ・ASP.NETモデルで動作できるアプリケーションが開発できます。	○ (注)	○

○:使用可能

×:使用不可

(注)

マルチセッションプログラミングで作成されたプログラムは、シングルスレッド(プロセス)でも動作可能です。
シングルスレッド(プロセス)プログラムを、マルチセッションで作成する必要はありません。

“マルチスレッド”と“マルチセッション”という表現は似ていますが、本書では以下のように使い分けています。

― プログラムやOS資源について関係がある場合には、プロセスやスレッド(マルチスレッド)という用語を使います。

― PowerRDBconnectorを使ったプログラミングのスタイルについては、セッションという用語を使います。

2.2.2.4 シングルセッション

1プロセス上の全てのアクセスをデータベースと1つのセッションで実行する方式です。

シングルセッションのプログラミングスタイルを使用する場合、以下を参照してください。

表2.5 シングルセッション使用時の各機能の説明/参照箇所

機能	説明部分
セッションの使用方法	セッションを意識する必要はありません。
コンパイル方法	「3.1.6 コンパイル方法」
データベース認証の使用方法	「3.3.2 データベース認証の使用方法(シングルセッション)」
トランザクションの使用方法	「3.4.2 トランザクションの使用方法(シングルセッション)」

2.2.2.5 マルチセッション

アプリケーションがセッションを複数開設して、セッションごとにデータベースへアクセスする方式です。

マルチセッションのプログラミングスタイルを使用する場合、以下を参照してください。

表2.6 マルチセッション使用時の各機能の説明/参照箇所

機能	説明部分
セッションの使用方法	「3.2 セッションの制御方法」
コンパイル方法	「3.1.6 コンパイル方法」
データベース認証の使用方法	「3.3.3 データベース認証の使用方法(マルチセッション)」
トランザクションの使用方法	「3.4.3 トランザクションの使用方法(マルチセッション)」

2.2.3 認証機能

SQL Serverでは、Windows認証か、データベース認証かのいずれかで認証し、データベースにアクセスすることができます。

PowerRDBconnectorは、どちらの認証方法も使用可能です。

- Windows認証の場合

  COBOLアプリケーションが動作するプロセスのユーザーIDでデータベースにアクセスできるよう、データベースを適切に設定してください。

- データベース認証の場合

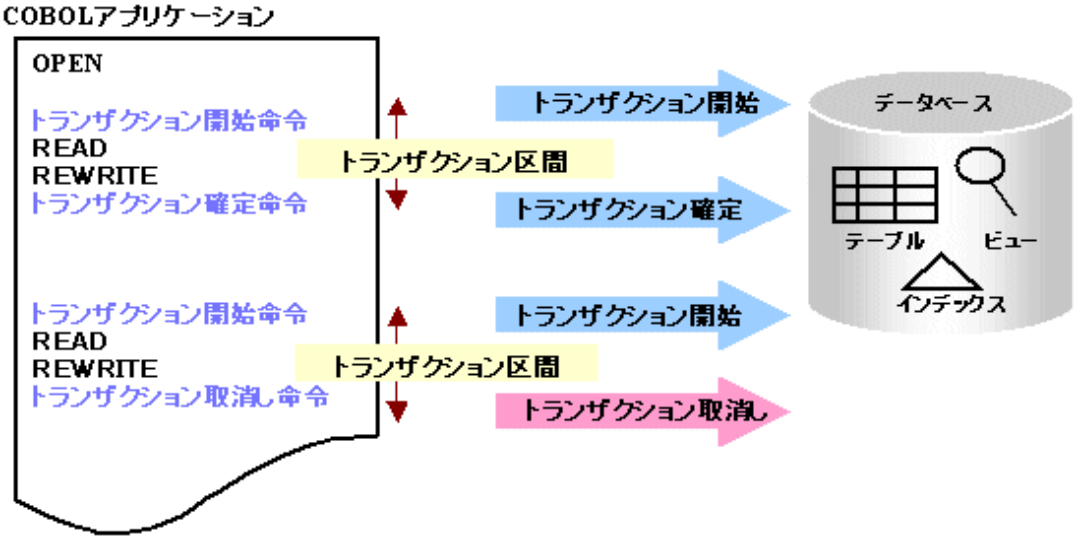
  COBOLアプリケーションの中から、認証情報登録サブルーチンを呼び出して、データベースが備えているデータベース認証を使用できます。

詳しくは「3.1.3.3 データベースにアクセスするための権限設定」と、「3.3 認証」を参照してください。

2.2.4 トランザクション機能

複数のファイルまたは、複数のレコード間のデータ整合性をセッション単位で保証するために、COBOLアプリケーションからトランザクション操作(開始、確定、取消し)を行うことができます。詳しくは、「3.4 トランザクション」を参照してください。

図2.6 トランザクション機能



2.2.5 排他制御

アプリケーションが多重動作するシステムにおいて、データの内容を保証するため、テーブルやレコードに対してセッション単位でロック(排他制御)を行います。

詳しくは、「3.5 排他制御」を参照してください。

2.3 システム構成

本節では、システムを構成するソフトウェアと運用環境について説明します。

2.3.1 必要なソフトウェア

PowerRDBconnectorを用いて、COBOLアプリケーションを開発・運用時に必要な製品を説明します。

2.3.1.1 PowerRDBconnector

インストールする本製品を、32ビット動作、64ビット動作および運用方法から以下のいずれかの製品を選択してください。

- 32ビット動作

  32ビットCOBOLアプリケーションには、以下のいずれかのPowerRDBconnectorが必要です。

  ー PowerRDBconnector クライアントパッケージ for NetCOBOL V3.1L20

  ー PowerRDBconnector サーバパッケージ for NetCOBOL V3.1L20

  (注)64bitが付いてない方の製品を使用してください。

- 64ビット動作

  64ビットCOBOLアプリケーションには、以下のPowerRDBconnectorが必要です。

  ー PowerRDBconnector サーバパッケージ for NetCOBOL (64bit) V3.1L20

  (注)64bitが付いている方の製品を使用してください。

2.3.1.2 OS

本製品をインストールするコンピュータ上に以下のいずれかの製品が必要です。

表2.7 PowerRDBconnectorでサポートするOS一覧

略称	製品名称	サーバパッケージ		クライアントパッケージ		サーバパッケージ(64bit)	
		開発	運用	開発	運用	開発	運用
Windows Server 2003	・Windows Server 2003, Standard Edition SP2 ・Windows Server 2003, Enterprise Edition SP2	○	○	○	△	×	×
	・Windows Server 2003, Standard x64 Edition SP2	○ WOW	○ WOW	○ WOW	△ WOW	×	×

略称	製品名称	サーバパッケージ		クライアントパッケージ		サーバパッケージ(64bit)	
		開発	運用	開発	運用	開発	運用
	・Windows Server 2003, Enterprise x64 Edition SP2	64 (注)	64 (注)	64 (注)	64 (注)		
Windows Server 2003 R2	・Windows Server 2003 R2, Standard Edition SP2 ・Windows Server 2003 R2, Enterprise Edition SP2	○	○	○	△	×	×
	・Windows Server 2003 R2, Standard x64 Edition SP2 ・Windows Server 2003 R2, Enterprise x64 Edition SP2	○ WOW 64 (注)	○ WOW 64 (注)	○ WOW 64 (注)	△ WOW 64 (注)	×	×
Windows Server 2008	・Windows Server 2008 Foundation SP2 ・Windows Server 2008 Standard SP2 ・Windows Server 2008 Enterprise SP2	○	○	○	△	×	×
	・Windows Server 2008 Foundation (x64) SP2 ・Windows Server 2008 Standard (x64) SP2 ・Windows Server 2008 Enterprise (x64) SP2	○ WOW 64 (注)	○ WOW 64 (注)	○ WOW 64 (注)	△ WOW 64 (注)	×	×
Windows Server 2008 R2	・Windows Server 2008 R2 Foundation SP2 ・Windows Server 2008 R2 Standard SP1 ・Windows Server 2008 R2 Enterprise SP1	○ WOW 64 (注)	○ WOW 64 (注)	○ WOW 64 (注)	△ WOW 64 (注)	○	○
Windows XP	・Windows XP Professional Edition SP3	○	×	○	○	×	×
Windows Vista	・Windows Vista Business SP 2 ・Windows Vista Enterprise SP2 ・Windows Vista Ultimate SP 2	○	×	○	○	×	×
Windows 7	・Windows 7 Professional SP 1 ・Windows 7 Enterprise SP 1 ・Windows 7 Ultimate SP 1	○	×	○	○	×	×
	・Windows 7 Professional (x64) SP1 ・Windows 7 Enterprise (x64) SP1 ・Windows 7 Ultimate (x64) SP1	○ WOW 64 (注)	×	○ WOW 64 (注)	○ WOW 64 (注)	○	×

○:サポートしています。

WOW64と記載時は、WOW64(Windows 32-bit On Windows 64-bit)サブシステム上での32ビット動作のみサポートしています。

×:未サポートです。

△:条件付きでサポートしています。

ターミナルサービス、ターミナルサーバー、Citrix XenAppやCitrix Presentation Server使用時のみサポートしています。ただしサポートしているクライアントOS動作は、各OSのドキュメントを参照してください。

(注) 64ビットOSで使用する場合の注意事項

ー 64ビットOSを使用するときは、64ビットのSQL Serverを使用してください。

データベース製品をネットワーク接続された別のサーバコンピュータにインストールする場合、データベースサーバコンピュータに、以下のいずれかの製品が必要です。

表2.8 PowerRDBconnectorでサポートする別サーバのOS一覧

略称	製品名称	サーバパッケージ	クライアントパッケージ	サーバパッケージ(64bit)
Windows Server 2003	・Windows Server 2003, Standard Edition SP2 ・Windows Server 2003, Enterprise Edition SP2	○	○	×
	・Windows Server 2003, Standard x64 Edition SP2 ・Windows Server 2003, Enterprise x64 Edition SP2	○(注1)	○(注1)	×
Windows Server 2003 R2	・Windows Server 2003 R2, Standard Edition SP2 ・Windows Server 2003 R2, Enterprise Edition SP2	○	○	×
	・Windows Server 2003 R2, Standard x64 Edition SP2 ・Windows Server 2003 R2, Enterprise x64 Edition SP2	○(注1)	○(注1)	×
Windows Server 2008	・Windows Server 2008 Foundation SP2 ・Windows Server 2008 Standard SP2 ・Windows Server 2008 Enterprise SP2	○	○	×
	・Windows Server 2008 Foundation (x64) SP2 ・Windows Server 2008 Standard (x64) SP2 ・Windows Server 2008 Enterprise (x64) SP2	○(注1)	○(注1)	×
Windows Server 2008 R2	・Windows Server 2008 R2 Foundation SP1	○(注1)	○(注1)	○

略称	製品名称	サーバパッケージ	クライアントパッケージ	サーバパッケージ(64bit)
	・Windows Server 2008 R2 Standard SP1 ・Windows Server 2008 R2 Enterprise SP1			
Solaris (注2)	・Solaris 9	○	○	×
	・Solaris 10	○	○	○

○:サポートしています。

×:未サポートです。

(注1) 64ビットOSで使用する場合の注意事項

ー 64ビットOSを使用するときは、64ビットのSQL Serverを使用してください。

(注2) Solarisの注意事項

Oracleのときのみ使用できます。

サーバコンピュータは、PRIMEPOWERシリーズ、富士通S series、SPARC Enterpriseに限ります。

2.3.1.3 COBOL

以下のいずれかの製品が必要です。(注)

表2.9 PowerRDBconnectorでサポートするNetCOBOL一覧

略称	製品名称	サーバパッケージ		クライアントパッケージ		サーバパッケージ(64bit)	
		開発	運用	開発	運用	開発	運用
NetCOBOL for Windows	・NetCOBOL Base Edition for Windows V7.0 ・NetCOBOL Standard Edition for Windows V7.0 ・NetCOBOL Professional Edition for Windows V7.0	○	×	○	×	×	×
	・NetCOBOL Base Edition クライアント運用パッケージ for Windows V7.0/V7.2/V8.0 ・NetCOBOL Standard Edition クライアント運用パッケージ for Windows V7.0//V8.0	×	△	×	○	×	×
	・NetCOBOL Base Edition サーバ運用パッケージ for Windows V7.0/V7.2/V8.0 ・NetCOBOL Standard Edition サーバ運用パッケージ for Windows V7.0/V7.2/V8.0	×	○	×	×	×	×
	・NetCOBOL 開発パッケージ for Windows V7.2/V8.0	×	×	○	×	×	×
	・NetCOBOL Base Edition 開発パッケージ for Windows V7.2/V8.0	○	×	○	×	×	×

略称	製品名称	サーバパッケージ		クライアントパッケージ		サーバパッケージ(64bit)	
		開発	運用	開発	運用	開発	運用
	・NetCOBOL Standard Edition 開発パッケージ for Windows V7.2/V8.0 ・NetCOBOL Professional Edition 開発パッケージ for Windows V7.2/V8.0						
NetCOBOL	・NetCOBOL Base Edition 開発パッケージ V9.0/V10 ・NetCOBOL Standard Edition 開発パッケージ V9.0/V10 ・NetCOBOL Professional Edition 開発パッケージ V9.0/V10	○	×	○	×	×	×
	・NetCOBOL Base Edition クライアント運用パッケージ V9.0/V10 ・NetCOBOL Standard Edition クライアント運用パッケージ V9.0/V10	×	△	×	○	×	×
	・NetCOBOL Base Edition サーバ運用パッケージ V9.0/V10 ・NetCOBOL Standard Edition サーバ運用パッケージ V9.0/V10	×	○	×	×	×	×
NetCOBOL for .NET	・NetCOBOL Base Edition for .NET V2.0 ・NetCOBOL Standard Edition for .NET V2.0	○	×	○	×	×	×
	・NetCOBOL Base Edition クライアント運用パッケージfor.NET V2.0～V4 ・NetCOBOL Standard Edition クライアント運用パッケージfor.NET V2.0～V4	×	△	×	○	×	×
	・NetCOBOL Base Edition サーバ運用パッケージ for .NET V2.0～V4 ・NetCOBOL Standard Edition サーバ運用パッケージ for .NET V2.0～V4	×	○	×	×	×	×
	・NetCOBOL Base Edition 開発パッケージ for .NET V2.1/V3.0/V3.1/V4 ・NetCOBOL Standard Edition 開発パッケージ for .NET V2.1/V3.0/V3.1/V4	○	×	○	×	×	×
	・NetCOBOL Enterprise Edition 開発パッケージ for .NET V3.0/V3.1/V4	○	×	×	×	×	×
	・NetCOBOL Enterprise Edition サーバ運用パッケージ for .NET V3.0/V3.1/V4	×	○	×	×	×	×
NetCOBOL (64bit)	・NetCOBOL Base Edition 開発パッケージ(64bit) V10.3	×	×	×	×	○	×

略称	製品名称	サーバパッケージ		クライアントパッケージ		サーバパッケージ(64bit)	
		開発	運用	開発	運用	開発	運用
	・ NetCOBOL Standard Edition 開発パッケージ(64bit) V10.3						
	・ NetCOBOL Enterprise Edition 開発パッケージ(64bit) V10	×	×	×	×	○	×
	・ NetCOBOL Base Edition サーバ運用パッケージ(64bit) V10.3 ・ NetCOBOL Standard Edition サーバ運用パッケージ(64bit) V10.3	×	×	×	×	×	○
	・ NetCOBOL Enterprise Edition 運用パッケージ(64bit) V10	×	×	×	×	×	○

○:サポートしています。

×:未サポートです。

△:条件付きでサポートしています。

ターミナルサービス、ターミナルサーバー、Citrix XenAppやCitrix Presentation Server使用時のみサポートしています。

（注）

NetCOBOLに含まれているPowerCOBOLはサポートしていません。

2.3.1.4 データベース

PowerRDBconnectorをインストールするコンピュータ内か、ネットワーク接続されたWindowsのサーバコンピュータ内に、以下のいずれかの製品が必要です。

表2.10 PowerRDBconnectorでサポートするSQL Server一覧

略称	製品名称	サーバパッケージ		クライアントパッケージ		サーバパッケージ(64bit)	
		開発	運用	開発	運用	開発	運用
SQL Server 2005	・ SQL Server 2005 Enterprise Edition(32ビット) ・ SQL Server 2005 Standard Edition(32ビット) ・ SQL Server 2005 Workgroup Edition(32ビット)	○	○	○	○	×	×
	・ SQL Server 2005 Developer Edition(32ビット)	○	×	○	×	×	×
	・ SQL Server 2005 Enterprise Edition x64 Extended ・ SQL Server 2005 Standard Edition x64 Extended	○	○	○	○	×	×
	・ SQL Server 2005 Developer Edition x64 Extended	○	×	○	×	×	×
SQL Server 2008	・ SQL Server 2008 Enterprise (32ビット)	○	○	○	○	×	×

略称	製品名称	サーバパッケージ		クライアントパッケージ		サーバパッケージ (64bit)	
		開発	運用	開発	運用	開発	運用
	・SQL Server 2008 Standard (32ビット) ・SQL Server 2008 Workgroup (32ビット)						
	・SQL Server 2008 Developer (32ビット)	○	×	○	×	×	×
	・SQL Server 2008 Enterprise (x64) ・SQL Server 2008 Standard (x64) ・SQL Server 2008 Workgroup (x64)	○	○	○	○	×	×
	・SQL Server 2008 Developer (x64)	○	×	○	×	×	×
SQL Server 2008 R2	・SQL Server 2008 R2 Enterprise (32ビット) ・SQL Server 2008 R2 Standard (32ビット) ・SQL Server 2008 R2 Workgroup (32ビット)	○	○	○	○	×	×
	・SQL Server 2008 R2 Developer (32ビット)	○	×	○	×	×	×
	・SQL Server 2008 R2 Enterprise (x64) ・SQL Server 2008 R2 Standard (x64) ・SQL Server 2008 R2 Workgroup (x64)	○	○	○	○	○	○
	・SQL Server 2008 R2 Developer (x64)	○	×	○	×	○	×

○:サポートしています。

×:未サポートです。

2.3.2 運用形態

本製品は、NetCOBOLが動作するコンピュータにインストールし、データベースがインストールされているコンピュータにアクセスします。

- PowerRDBconnector クライアントパッケージ for NetCOBOL

  クライアントパッケージは、クライアントコンピュータ内で、クライアントサーバ形態またはスタンドアロン運用を行う場合に使用します。

  PowerRDBconnectorをインストールしたコンピュータ内のデータベースに対して、クライアントサーバ経由によらず直接アクセスするアプリケーション(例えばバッチ型のアプリケーションなど)は使用できません。

- PowerRDBconnector サーバパッケージ for NetCOBOL

  サーバパッケージは、サーバコンピュータ内で運用する場合に使用します。

2.3.2.1 開発時の形態

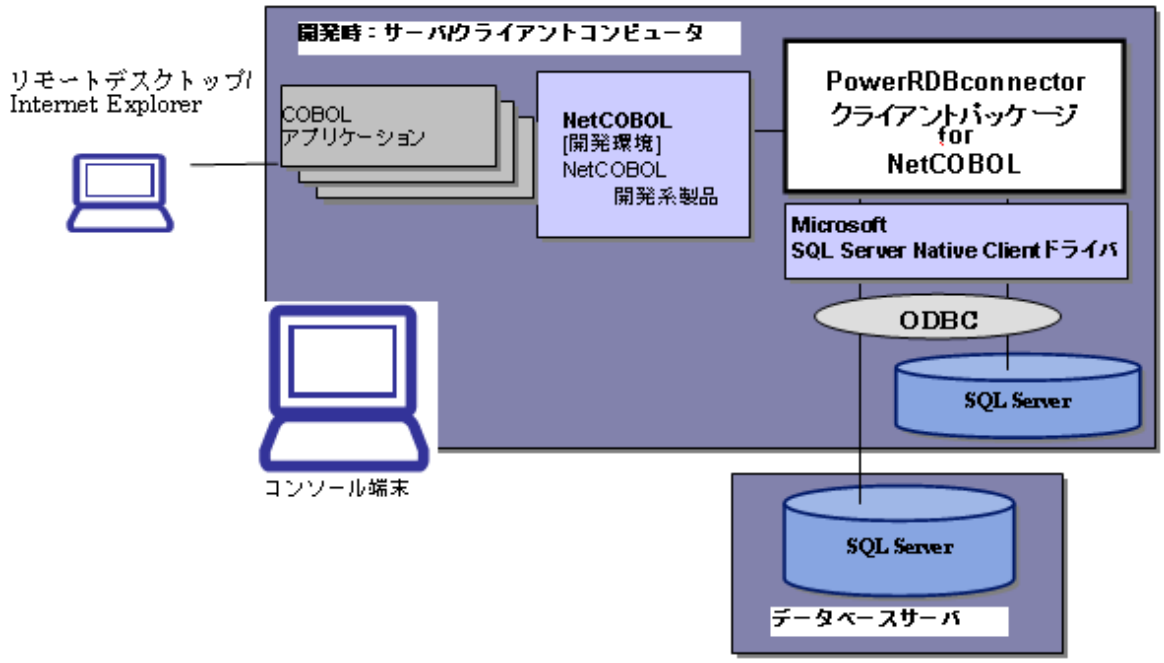
コンピュータは、サーバコンピュータおよびクライアントコンピュータが使用できます。NetCOBOLの開発系製品と同時に使用してください。

データベースは、PowerRDBconnectorをインストールしたコンピュータと同じコンピュータ内か、異なるWindowsのデータベースサーバ内に配置することができます。

【PowerRDBconnector クライアントパッケージ for NetCOBOL】

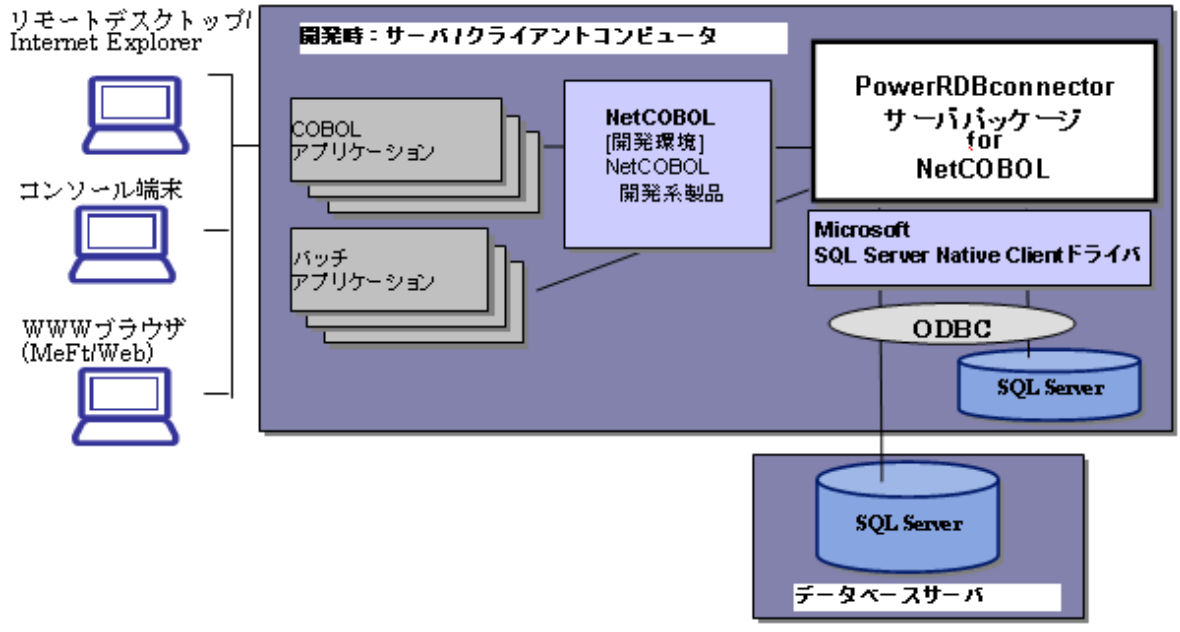
クライアントパッケージの場合、PowerRDBconnectorをインストールしたコンピュータ内のデータベースに対して、クライアントサーバ経由によらず直接アクセスするアプリケーション（例えばバッチ型のアプリケーションなど）の開発用途には使用できません。

図2.7 PowerRDBconnectorクライアントパッケージ for NetCOBOL 運用形態1



【PowerRDBconnector サーバパッケージ for NetCOBOL】

図2.8 PowerRDBconnectorサーバパッケージfor NetCOBOL 運用形態1



- 64ビットOSで32ビット動作を使用する場合の注意事項
  - ― 32ビットのNetCOBOLおよび、PowerRDBconnectorを使用してください。
  - ― 64ビットのSQL Serverを使用してください。
  - ― ASP.NETで使用するときは、IISの32ビット互換モードを使用してください。
- 64ビット動作を使用する場合の注意事項
  - ― 64ビットのNetCOBOLおよび、PowerRDBconnectorを使用してください。
  - ― 64ビットのSQL Serverを使用してください。

2.3.2.2 運用時の形態

【PowerRDBconnector クライアントパッケージ for NetCOBOL】

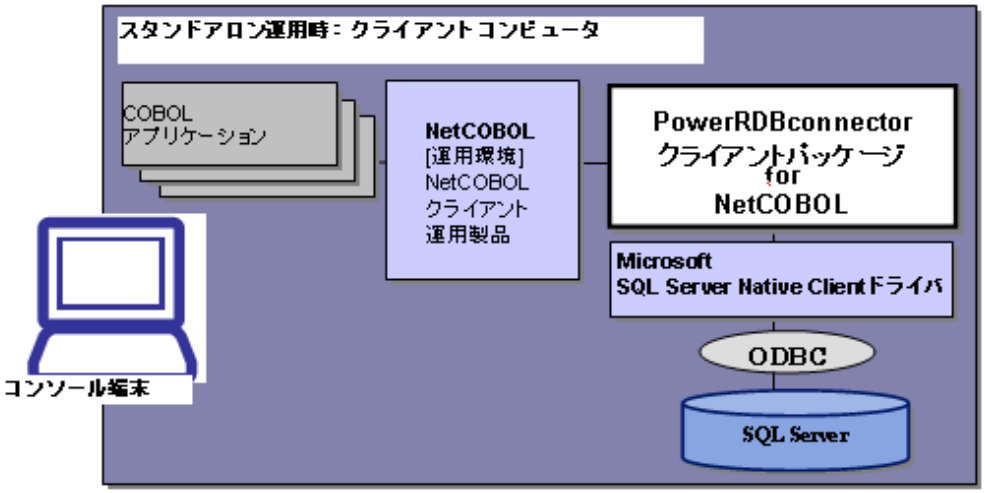
- スタンドアロン運用時

  コンピュータは、クライアントコンピュータが使用できます。NetCOBOLのクライアント運用系製品と同時に使用してください。

  PowerRDBconnectorをインストールしたコンピュータと同じコンピュータに、データベースを配置することができます。

  リモートデスクトップ接続や、WWWブラウザ経由のアクセスはできません。コンソール端末からアクセスできます。

  図2.9 PowerRDBconnectorクライアントパッケージ for NetCOBOL 運用形態2



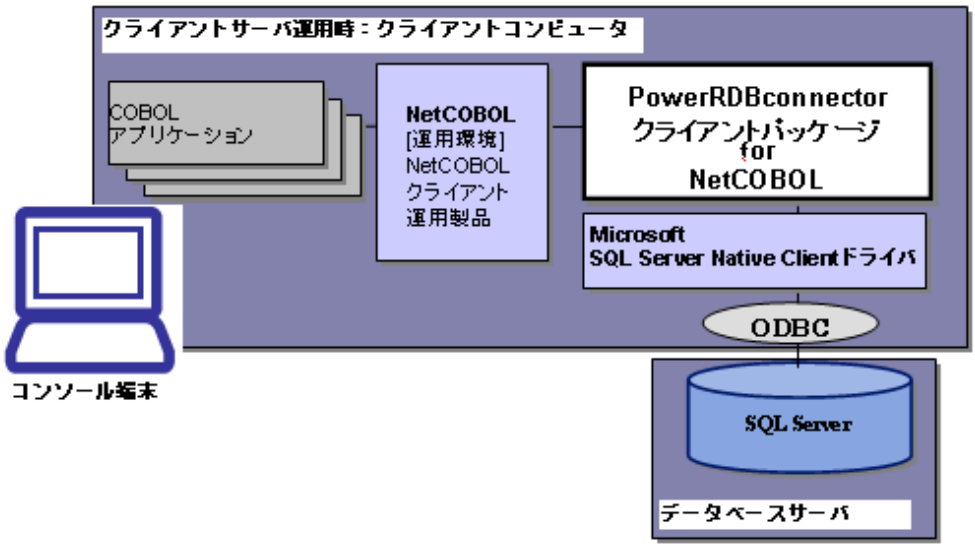
- クライアントサーバ運用時

  コンピュータは、クライアントコンピュータが使用できます。NetCOBOLのクライアント運用系製品と同時に使用してください。

  PowerRDBconnectorをインストールしたコンピュータとは異なるコンピュータで、かつWindowsのOSを持つサーバ内に、データベースを配置することができます。

リモートデスクトップ接続や、WWWブラウザ経由のアクセスはできません。コンソール端末からアクセスできます。

図2.10 PowerRDBconnectorクライアントパッケージ for NetCOBOL 運用形態3



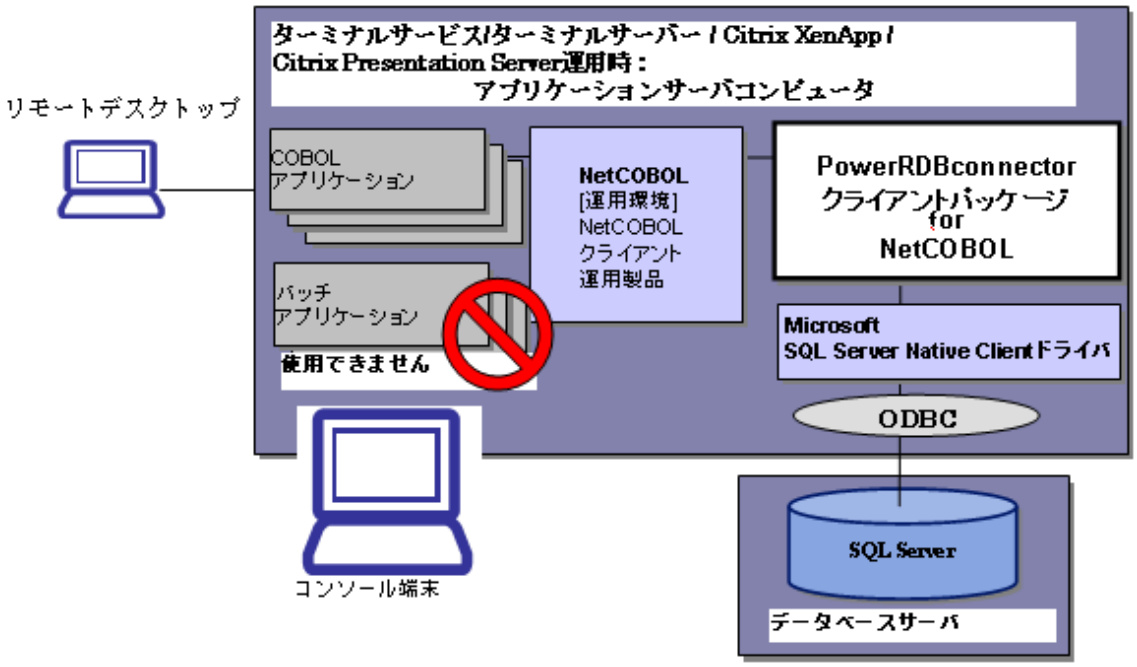
- ターミナルサービス/ターミナルサーバー運用時

  コンピュータは、アプリケーションサーバコンピュータが使用できます。NetCOBOLのクライアント運用系製品と同時に使用してください。

  データベースは、PowerRDBconnectorをインストールしたコンピュータとは異なるWindowsのデータベースサーバ内に配置することをお勧めします。

  ターミナルサービス、ターミナルサーバー、Citrix XenAppやCitrix Presentation Serverを使用したリモートデスクトップ接続や、コンソール端末からアクセスできます。アプリケーションサーバ内のバッチ型アプリケーションは使用できません。

図2.11 PowerRDBconnectorクライアントパッケージ for NetCOBOL 運用形態4



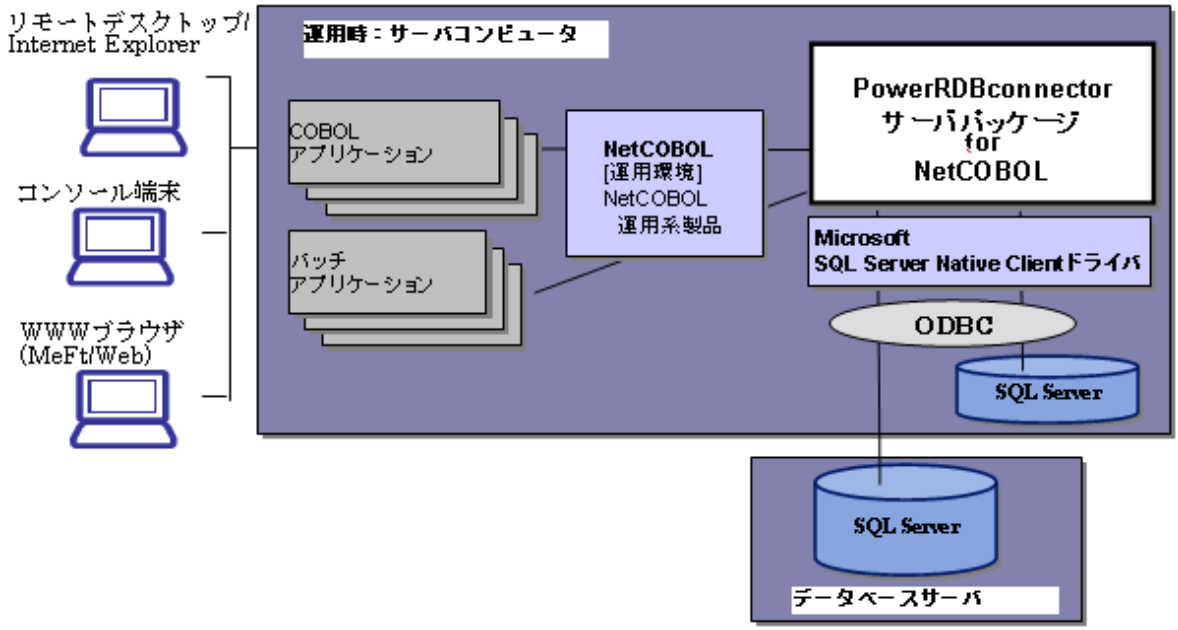
【PowerRDBconnector サーバパッケージ for NetCOBOL】

コンピュータは、サーバコンピュータが使用できます。NetCOBOLのサーバ運用系製品と同時に使用してください。

データベースは、PowerRDBconnectorをインストールしたコンピュータと同じコンピュータ内か、異なるWindowsのデータベースサーバ内に配置することができます。

ターミナルサービス、ターミナルサーバー、Citrix XenAppやCitrix Presentation Serverを使用したリモートデスクトップ接続や、WWWブラウザ経由および、サーバコンピュータ内のバッチ型のアプリケーションからアクセスできます。

図2.12 PowerRDBconnectorサーバパッケージ for NetCOBOL 運用形態2



- 64ビットOSで32ビット動作を使用する場合の注意事項
  - 32ビットのNetCOBOLおよび、PowerRDBconnectorを使用してください。
  - 64ビットのSQL Serverを使用してください。
  - ASP.NETで使用するときは、IISの32ビット互換モードを使用してください。
- 64ビット動作を使用する場合の注意事項
  - 64ビットのNetCOBOLおよび、PowerRDBconnectorを使用してください。
  - 64ビットのSQL Serverを使用してください。

2.3.2.3 サポートするOSとデータベースの組合せ

PowerRDBconnectorがサポートしているOSとデータベースの組合せを、以下に示します。

【PowerRDBconnector クライアントパッケージ for NetCOBOL】

図2.13 PowerRDBconnectorとデータベースを同じコンピュータ上にインストールする場合に、使用可能なOSとデータベースの組合せ（32ビット動作）

使用するSQL Server	PowerRDBconnectorクライアントパッケージ for NetCOBOLをインストールするOS								
	Windows				Windows Server				
	XP	Vista	7	7(x64)	2003	2003 (x64)	2008	2008 (x64)	2008 R2
2005	○	○	○	×	○	×	○	×	×
2005 (x64)	×	×	×	×	×	○	×	○	○
2008	○	○	○	×	○	×	○	×	×
2008 (x64)	×	×	×	×	×	○	×	○	○
2008 R2	○	○	○	×	○	×	○	×	×
2008 R2 (x64)	×	×	×	○	×	○	×	○	○

○:使用可能

×:使用不可能

図2.14 PowerRDBconnectorとデータベースを別々のコンピュータ上にインストールする場合に、使用可能なOSとデータベースの組合せ（32ビット動作）

別コンピュータの接続先		PowerRDBconnectorクライアントパッケージfor NetCOBOLをインストールするOS								
OS	SQL Server	Windows				Windows Server				
		XP	Vista	7	7(x64)	2003	2003 (x64)	2008	2008 (x64)	2008 R2
Windows サーバ	2005	○	○	○	×	○	×	○	×	×
	2005 (x64)	○	○	○	×	○	○	○	○	○
	2008	○	○	○	×	○	×	○	×	×
	2008 (x64)	○	○	○	×	○	○	○	○	○
	2008 R2	○	○	○	×	○	×	○	×	×
	2008 R2 (x64)	○	○	○	○	○	○	○	○	○

○:使用可能

×:使用不可能

【PowerRDBconnectorサーバパッケージ for NetCOBOL】

図2.15 PowerRDBconnectorとデータベースを同じコンピュータ上にインストールする場合に、使用可能なOSとデータベースの組合せ（32ビット動作）

使用するSQL Server	PowerRDBconnector サーバパッケージ for NetCOBOL をインストールするOS								
	Windows				Windows Server				
	XP	Vista	7	7(x64)	2003	2003 (x64)	2008	2008 (x64)	2008 R2
2005	○	○	○	×	○	×	○	×	×
2005 (x64)	×	×	×	×	×	○	×	○	○
2008	○	○	○	×	○	×	○	×	×
2008 (x64)	×	×	×	×	×	○	×	○	○
2008 R2	○	○	○	×	○	×	○	×	×
2008 R2 (x64)	×	×	×	○	×	○	×	○	○

○:使用可能

×:使用不可能

図2.16 PowerRDBconnectorとデータベースを別々のコンピュータ上にインストールする場合に、使用可能なOSとデータベースの組合せ（32ビット動作）

別コンピュータの接続先		PowerRDBconnectorサーバパッケージ for NetCOBOL をインストールするOS								
OS	SQL Server	Windows				Windows Server				
		XP	Vista	7	7(x64)	2003	2003 (x64)	2008	2008 (x64)	2008 R2
Windows サーバ	2005	○	○	○	×	○	×	○	×	×
	2005 (x64)	○	○	○	×	○	○	○	○	○
	2008	○	○	○	×	○	×	○	×	×
	2008 (x64)	○	○	○	×	○	○	○	○	○
	2008 R2	○	○	○	×	○	×	○	×	×
	2008 R2 (x64)	○	○	○	○	○	○	○	○	○

○:使用可能

×:使用不可能

図2.17 PowerRDBconnectorとデータベースを同じコンピュータ上にインストールする場合に、使用可能なOSとデータベースの組合せ（64ビット動作）

使用するSQL Server	PowerRDBconnectorサーバパッケージ for NetCOBOL (64bit)をインストールするOS								
	Windows				Windows Server				
	XP	Vista	7	7(x64)	2003	2003 (x64)	2008	2008 (x64)	2008 R2
2005	×	×	×	×	×	×	×	×	×
2005 (x64)	×	×	×	×	×	×	×	×	×
2008	×	×	×	×	×	×	×	×	×
2008 (x64)	×	×	×	×	×	×	×	×	×
2008 R2	×	×	×	×	×	×	×	×	×
2008 R2 (x64)	×	×	×	○	×	×	×	×	○

○:使用可能

×:使用不可能

図2.18 PowerRDBconnectorとデータベースを別々のコンピュータ上にインストールする場合に、使用可能なOSとデータベースの組合せ（64ビット動作）

別コンピュータの接続先		PowerRDBconnectorサーバパッケージ for NetCOBOL (64bit)をインストールするOS								
OS	SQL Server	Windows				Windows Server				
		XP	Vista	7	7(x64)	2003	2003 (x64)	2008	2008 (x64)	2008 R2
Windows サーバ	2005	×	×	×	×	×	×	×	×	×
	2005 (x64)	×	×	×	×	×	×	×	×	×
	2008	×	×	×	×	×	×	×	×	×
	2008 (x64)	×	×	×	×	×	×	×	×	×
	2008 R2	×	×	×	×	×	×	×	×	×
	2008 R2 (x64)	×	×	×	○	×	×	×	×	○

○:使用可能

×:使用不可能

2.3.2.4 未サポートの運用形態

以下の運用形態はサポートしていません。

- 32ビット動作を使用時、64ビットOS上で、64 ビットのCOBOLアプリケーションからのアクセス
- 64ビット動作を使用時
  - 32ビットのCOBOLアプリケーションからのアクセス
  - NetCOBOL for .NETのアプリケーションからのアクセス
  - PowerRDBconnector クライアントパッケージ for NetCOBOLの運用
- Server Coreインストールオプションでインストールしたwindows Server 2008、Windows Server 2008(x64)上での動作
- Windows Azureでの動作
- guestユーザーでの動作
- 1つのCOBOLアプリケーションから、2つ以上のデータベースサーバに対するアクセス
- SQL Azureへのアクセス
- PowerRDBconnectorをインストールしたWindowsコンピュータから、Windows以外のデータベースサーバに対するアクセス（リンクサーバー）

・ Hyper-V以外の仮想マシン上での動作

第3章 PowerRDBconnectorの使用手引き

本章では、PowerRDBconnector の使用方法について説明します。

開発者およびシステム管理者が、業務アプリケーションの設計時に、PowerRDBconnectorの知っておきたい機能やNetCOBOLアプリケーションの環境構築、およびアプリケーションの開発方法について説明します。

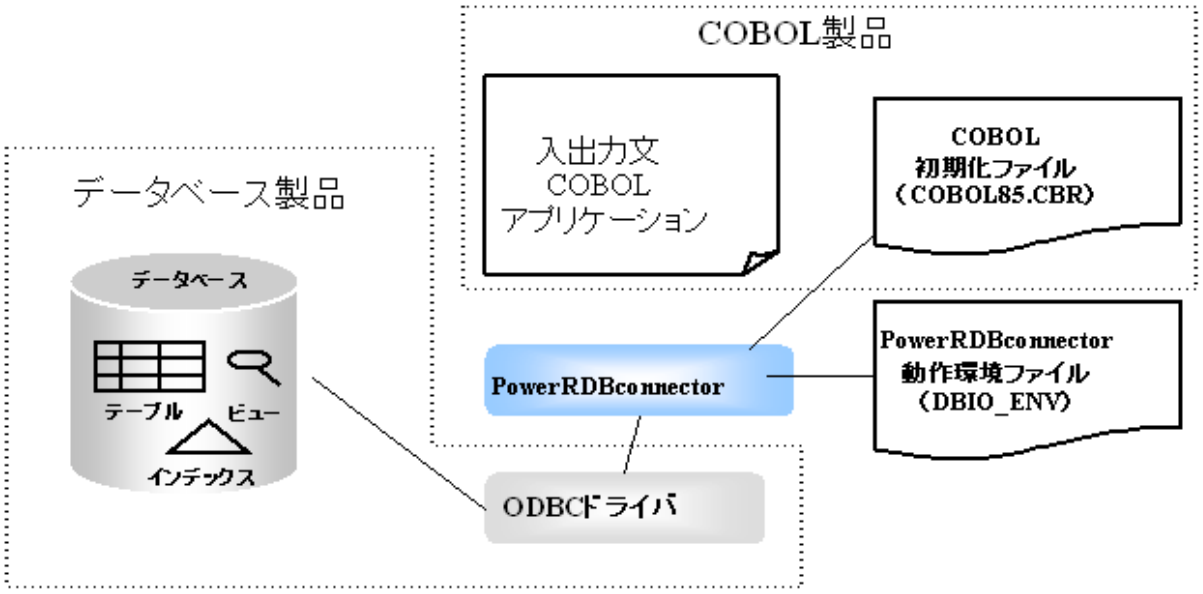
3.1 環境構築

本節では、PowerRDBconnector の環境設定方法について説明します。

3.1.1 PowerRDBconnectorを構成する要素

PowerRDBconnector を構成する要素は以下のようになります。

図3.1 PowerRDBconnector を構成する要素



各構成要素を以下に示します。

表3.1 PowerRDBconnector を構成する要素

PowerRDBconnectorを構成する要素	説明
COBOL初期化ファイル	COBOLアプリケーションの実行情報を記述したテキストファイル
PowerRDBconnector動作環境ファイル	PowerRDBconnector の動作環境を記述したテキストファイル
PowerRDBconnector	本ソフトウェア
ODBCドライバ	データベースにアクセスするインターフェース Microsoft SQL Server Native Client ドライバ

PowerRDBconnectorを構成する要素	説明
データベース	PowerRDBconnector がアクセスするSQL Server
テーブル	データファイルに相当するデータベースオブジェクト
ビュー	索引ファイルに相当するデータベースオブジェクト
インデックス	レコードキーに相当するデータベースオブジェクト

なお、動作環境ひな型作成ツールを使用することで、NetCOBOLのソースプログラムから、以下の定義情報のひな型を作成することができます。

- PowerRDBconnector動作環境ファイル
- COBOL初期化ファイル
- テーブル作成文(テーブル、ビュー、インデックスを定義するSQL文)

動作環境ひな型作成ツールについては、「PowerRDBconnector 動作環境ひな型作成ツール 操作手引書」を参照してください。

3.1.2 環境構築の手順

必要な環境構築の手順を説明します。

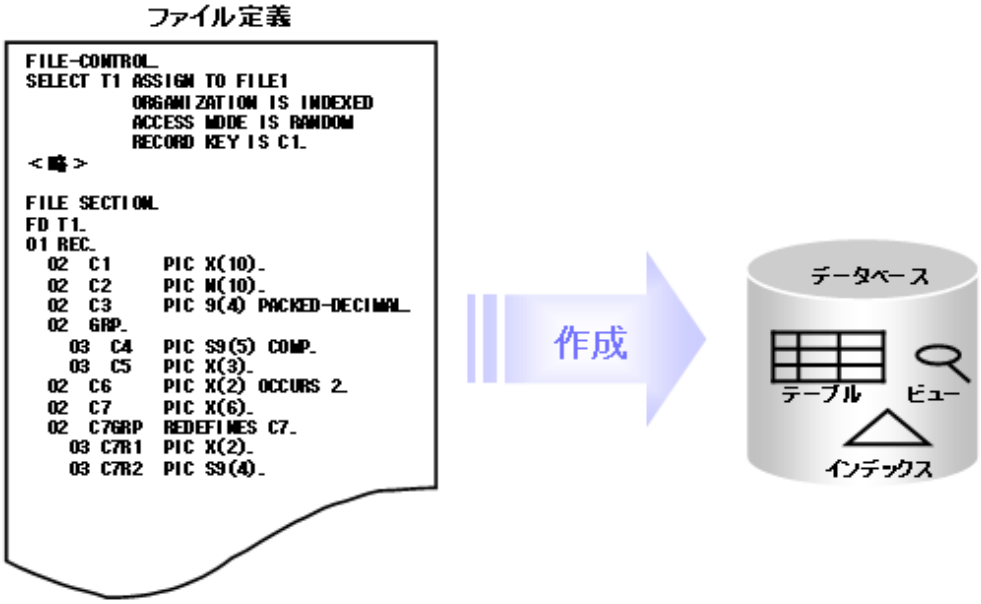
NetCOBOL製品のインストール	NetCOBOL製品をインストールします。 詳しくは、NetCOBOLのマニュアルを参照してください。
↓	
データベース製品のインストール	SQL Serverをインストールします。 また、PowerRDBconnectorをインストールするシステムに、Microsoft SQL Server Native Client ドライバをインストールしてください。 インストール方法については、SQL ServerのBooks Onlineを参照してください。なお、SQL Serverに最新パッチを適用してください。
↓	
PowerRDBconnectorのインストール	SQL Server用のPowerRDBconnector をインストールします。 詳しくは、製品CDに格納されているREADME.txtを参照してください。
↓	
環境の確認	PowerRDBconnector は、NetCOBOLおよびデータベースの実行環境が正しく設定されたプロセス上で動作できます。正しく設定されているか確認してください。 ・ NetCOBOL 詳しくは、NetCOBOLのマニュアルを参照してください。 ・ SQL Server 詳しくは、SQL ServerのBooks Onlineを参照してください。
↓	
データベースオブジェクトの作成	COBOLアプリケーションからアクセスするテーブルおよびビュー、インデックスを作成します。
↓	
PowerRDBconnector動作環境ファイルの作成	PowerRDBconnector 動作環境ファイルを作成します。
↓	
COBOL初期化ファイルの作成	COBOLアプリケーションの実行に必要なCOBOL初期化ファイルを作成します。

↓	
入出力文 COBOLアプリケーション の作成	COBOLアプリケーションを作成します。 COBOLアプリケーションの作成は、NetCOBOLのマニュアルを参照してください。入出力文によるファイルアクセスは、「第3章 PowerRDBconnectorの使用手引き」を参照してください。
↓	
COBOLアプリケーション のテスト	COBOLアプリケーションの動作を確認します。 COBOLアプリケーションが単体で正常に動作することを確認します。ファイルアクセスでエラーが通知される場合、「5.1 エラー時の対処」を参照して対処してください。
↓	
データベースのチューニング	データベースをチューニングします。 実際の運用に近い環境で動作確認を行い、SQL Serverのチューニングを行います。チューニングせずに運用を開始すると、最適な性能が得られず、タイムアウトが発生する場合があります。 SQL Serverのチューニング方法については、SQL ServerのBooks Online等を参照してください。
↓	
COBOLアプリケーション の運用	COBOLアプリケーションの運用を開始します。

3.1.3 データベースオブジェクトの作成

COBOLで定義されているファイルに対応するデータベースオブジェクト(テーブルまたはテーブルとビュー)を作成します。

図3.2 データベースオブジェクトの作成



3.1.3.1 データベースオブジェクトの作成手順

以下の手順を参考に作成します。正しく作成しないと動作できませんので、注意して作成してください。

データベースの作成	・ データベース テーブルおよびビューを格納するデータベースを作成します。 「3.1.3.2 データベースの作成」を参考に、データベースを作成します。 ・ データベースのアクセス権限 テーブルやビューへアクセスするために、COBOLアプリケーションを起動するユーザーにアクセス権限を設定します。 「3.1.3.3 データベースにアクセスするための権限設定」を参考に、ユーザーのアクセス権限を設定します。
↓	
SQL定義文の作成	・ テーブル名 テーブルまたはテーブルとビューで任意の名前で定義します。「表3.2 ファイルとテーブルまたはビューの対応」を参考にテーブルやビューを構成します。 ・ ファイル構成 レコードキーを定義するファイルは、インデックスまたはプライマリキーを定義します。「表3.2 ファイルとテーブルまたはビューの対応」を参考にデータベースオブジェクトを構成します。 ・ レコード構成 COBOLアプリケーションのファイル記述項（FD句）のレコード記述項に定義する基本項目からテーブルまたはビューの列を定義します。「3.1.3.5 列定義の対応」を参考に列定義（データ型、列名）を構成します。なお、各列には NOT NULL制約を定義してください。
↓	
SQL定義文の実行	作成したSQL定義文を実行します。 SQL Serverのosqlユーティリティ等で実行できます。

3.1.3.2 データベースの作成

COBOLアプリケーションからアクセスするテーブルおよびビューを格納するデータベースを作成します。なお、既存のデータベースを使用する場合には、新たにデータベースを作成する必要はありません。

データベースの作成は、SQL Serverが提供する以下のいずれかを使用します。

- SQL Server Management Studio
- Transact-SQL（CREATE DATABASEコマンド）

データベースのデータファイルおよびトランザクションログは、領域の自動拡張および自動圧縮が可能ですが、自動拡張や自動圧縮が動作するとCOBOLアプリケーションからのアクセス性能の低下やタイムアウトが発生する場合があります。このため、領域の自動拡張および自動圧縮は有効にしないことを推奨します。

データファイルおよびトランザクションログの最大サイズを設定、または自動拡張を無効にした場合、ファイルサイズの上限に達するとSQL Serverは領域不足を通知します。このため、データファイルのサイズを十分に割り当ててください。トランザクションログのファイルサイズは、最大でデータファイルの2倍必要になります。

領域不足が発生した場合には、SQL Server Management Studioや、Transact-SQL（ALTER DATABASEコマンド）で領域を拡張できます。

データベースの作成方法について、詳しくはSQL ServerのBooks Onlineを参照してください。

3.1.3.3 データベースにアクセスするための権限設定

COBOLアプリケーションからテーブルおよびビューへアクセスするための権限を、以下の手順で設定してください。

(1) ログインユーザーの追加

次のユーザーをSQL Serverのログインユーザーに追加してください。

- Windows認証使用時
  COBOLアプリケーションを起動するドメインまたはコンピュータのユーザー
- データベース認証使用時
  COBOLアプリケーションの認証情報登録サブルーチンに指定するユーザー

(2) テーブルおよびビューのアクセスに必要なシステム権限の設定

SQL Serverのログインユーザーには、テーブルおよびビューのアクセスに必要な権限を設定してください。

(3) "db_ddladmin"システム権限の設定

OUTPUTモードでOPEN文を実行するCOBOLアプリケーションを起動するユーザーについては、データベースロールに“db_ddladmin”を設定してください。

SQL Serverのログインユーザー追加および権限の設定は、SQL Serverのユーティリティを使用して行います。ログインユーザーの追加方法および権限の設定方法については、SQL ServerのBooks Onlineを参照してください。

COBOLアプリケーション中で認証情報登録サブルーチンを使用して、データベース認証を使用する場合は、「3.3.2 データベース認証の使用方法（シングルセッション）」、「3.3.3 データベース認証の使用方法（マルチセッション）」を参照してください。

3.1.3.4 ファイルとテーブルまたはビューの対応

COBOLアプリケーションで扱うファイルと、データベースで扱うテーブルまたはビューの対応を以下に示します。

表3.2 ファイルとテーブルまたはビューの対応

ファイル編成	データベースオブジェクト	補足
順ファイル	以下のいずれかを作成 ・ テーブルを作成 ・ テーブルとビューを作成	テーブルのみを作成する場合はテーブルの列名、テーブルとビューを作成する場合はビューの列名を「3.1.3.5 列定義の対応」に説明した列名にしてください。 テーブルとビューを作成する場合は、テーブルの列名は任意の名前で作成できます。
索引ファイル	以下のいずれかを作成 ・ インデックス付きテーブルを作成 ・ 上記のテーブルを導出元にしたビューを作成	テーブルのみを作成する場合はテーブルの列名、テーブルとビューを作成する場合はビューの列名を「3.1.3.5 列定義の対応」に説明した列名にしてください。 テーブルとビューを作成する場合は、テーブルの列名は任意の名前で作成できます。 RECORD KEY句にあたる列にインデックスを設定します。 UNIQUE制約は、RECORD KEY句のWITH DUPLICATEがない場合に指定してください。 インデックス付きテーブルを導出元にしたビューも索引ファイルとしてアクセスできます。

ビューに対するアクセス機能については、「4.2.3.2 ビューの使用について」を参照してください。

ビューの構成については、「【ファイル記述項からテーブルまたはビューの作成例】」を参照してください。

3.1.3.5 列定義の対応

COBOLのレコード記述項とデータベースの列定義の対応を以下に示します。

表3.3 COBOL定義とデータベース列定義の対応

COBOL定義					データベース定義	
項目の種類	USAGE句	PICTURE句			データ型	列名
		符号	精度(p)	位取り(s)		
数字項目	DISPLAY (外部10進)	なし	1～18	0～18	NUMERIC(p,s)	<文字列>**_UNSIGN_NUMERIC** または <文字列>**_UNUM**
		あり	1～18	0～18	NUMERIC(p,s)	<文字列>**_NUMERIC** または <文字列>**_NUM**
	COMP-3、 COMPUTATIONAL-3、 PACKED-DECIMAL (内部10進)	なし	1～18	0～18	DECIMAL(p,s)	<文字列>**_UNSIGN_DECIMAL** または <文字列>**_UDEC**
		あり	1～18	0～18	DECIMAL(p,s)	<文字列>**_DECIMAL** または <文字列>**_DEC**
整数項目	COMP、 COMPUTATIONAL、 BINARY (2進)	なし	1～4	---	SMALLINT	<文字列>**_UNSIGN_SMALLINT** または <文字列>**_USINT**
			5～9	---	INTEGER	<文字列>**_UNSIGN_INTEGER** または <文字列>**_UINT**
			10～18	---	BIGINT	<文字列>**_UNSIGN_BIGINT** または <文字列>**_UBINT**
		あり	1～4	---	SMALLINT	<文字列>**_SMALLINT** または <文字列>**_SINT**
			5～9	---	INTEGER	<文字列>**_INTEGER** または <文字列>**_INT**
			10～18	---	BIGINT	<文字列>**_BIGINT** または <文字列>**_BINT**
英数字項目	X(英数字)	---	p	---	CHAR(p) NCHAR(p)	<文字列>**_CHAR** または <文字列>**_CHR**
					VARCHAR(p) NVARCHAR(p)	<文字列>**_CHAR_<項目長>** または <文字列>**_CHR_<項目長>** または <文字列>**_CHAR**

COBOL定義					データベース定義	
項目の種類	USAGE句	PICTURE句			データ型	列名
		符号	精度(p)	位取り(s)		
						または <文字列>**_CHR**
日本語項目	N(日本語)	---	p	---	CHAR(p×2) NCHAR(p)	<文字列>**_NCHAR** または <文字列>**_NCHR**
					VARCHAR(p×2) NVARCHAR(p)	<文字列>**_NCHAR_<項目長>** または <文字列>**_NCHR_<項目長>** または <文字列>**_NCHAR** または <文字列>**_NCHR**
バイナリ項目	X(バイナリ)	---	p	---	BINARY(p)	<文字列>**_BINARY** または <文字列>**_BIN**
					VARBINARY(p)	<文字列>**_BINARY_<項目長>** または <文字列>**_BIN_<項目長>** または <文字列>**_BINARY** または <文字列>**_BIN**

精度(p) :精度は、全体桁数および項目の長さ(文字数)です。

位取り(s):位取りは、小数部の桁数です。

文字列:任意の文字列です。

項目長:COBOLでの項目長です。

注意

COBOLのレコード記述項と、データベースの列定義の対応には、以下の注意事項があります。

1. COBOLのレコード記述項のレコード長とテーブルの行長が一致しているかCOBOLがチェックします。必ず、レコード記述項に合わせてテーブルの列を定義してください。レコード長とテーブルの行長が一致していない場合、OPEN文実行時、以下のCOBOLエラーが発生します。

   －“JMP0310I-I/U ～ファイルでOPENエラーが発生しました. INV-LRECL”

2. 必ず、RECORD KEY句に定義したキーに対応するインデックスを定義してください。
3. COBOLのレコード記述項に定義したOCCURS句は、繰り返された基本項目が列になります。
4. COBOLのレコード記述項に定義したREDEFINES句は、再定義された最下位レベル番号の基本項目が列になります。
5. BINARYデータ型に対応させたX項目は、レコードキーにできません。
   BINARYデータ型に対応させたX項目は、1つの項目に、英数字項目と数字項目が混在して使用されているような場合に使用します。BINARYデータ型に対応させたX項目は、PowerRDBconnectorおよびデータベースでは変換/補正を行いません。このためCOBOLアプリケーションで項目に応じてデータ内容を自由に定義できます。
6. USAGE句がCOMP-5の場合は、エラーとはなりませんが、データが正しく評価されません。
7. 作成できるテーブルおよびビューの制限値(最大テーブル数、最大テーブルサイズ、最大行長、最大列数、最大列長、最大列名長など)は、データベースのマニュアルを参照してください。

8. 文字列の後方空白の扱いについては、「3.7.1 後方空白補正」を参照してください。

9. データベースの仕様で、項目で定義するデータベースの精度(p)には、以下の制限があります。

表3.4 データベースのデータの精度

項目の種類	データ型	精度
英数字項目	CHAR	8,000バイト
	VARCHAR	8,000バイト
日本語項目	NCHAR	8,000バイト
	NVARCHAR	8,000バイト
数字項目/整数項目	SMALLINT	2バイト
	INT	4バイト
	BIGINT	8バイト
	DECIMAL	38桁
	NUMERIC	38桁
バイナリ項目	BINARY	8,000バイト
	VARBINARY	8,000バイト

なお、unicodeはUTF-8で扱う場合、1文字が最大3バイトで表現されます。UTF-16で扱う場合、1文字が最大4バイトで表現されます。

10. 列名は、ASCIIコードまたはシフトJISコード(JIS X 0208-1990)の文字セットで記述してください。JIS X 0213:2004に対応した文字セット(JIS2004)では、4バイトの文字や、シフトJISコード(JIS X 0208-1990)に存在しない文字はエラーとなります。このためunicode固有文字や、JIS2004で追加された文字は使用しないでください。

11. JIS2004の文字コード系を使用時は、JIS2004の1文字につき、COBOLの英数字項目は4文字分、日本語項目は2文字分用意してください。対応するデータベースの項目型も、2文字分に対応したサイズを指定してください。

12. 可変長のデータ型について、COBOLのレコード記述項とデータベースの列定義の対応には、以下の注意事項があります。

- データベースの列名に定義する項目長は、COBOLのレコード記述項で定義する項目の長さと同じ値で定義してください。また、列名に定義する項目長は4桁で指定してください。項目長が4桁に満たない場合は先頭に0を詰めて指定してください。以下に例を示します。

  例

  - 項目長が10の場合

    <任意の文字列>_CHAR_0010

  - 項目長が1000の場合

    <任意の文字列>_CHAR_1000

  - 項目長が10で、列名に定義する項目長が4桁でない場合

    <任意の文字列>_CHAR_10

    以下のエラーになります。

    「EINVAL(22) stat = '9' stat = 0x34 stat = '9' stat = 0x34」

- データベースの列名に定義する項目長とCOBOLのレコード記述項で定義する項目の長さが一致しない場合、動作保証されません。アクセス時にエラーが発生しなくてもデータは正しく評価されません。

- データベースの列名に定義する項目長は、可変長項目を使用時に指定してください。可変長項目以外の項目に対してデータベースの列名に項目長を定義すると、エラーとなります。また、データベースの列名に定義する項目長を省略する場合は、COBOLのレコード記述項で定義する項目の長さとデータベースで定義する列定義の列長と同じ長さで定義してください。

ー COBOLのレコード記述項で定義する項目の長さとデータベースで定義する列定義の列長が一致している場合は、データベースの列名に定義する項目長は省略できます。

例

- データベースの列長よりCOBOLの項目長が短い場合

  以下の表では、データベースの定義がVARCHAR(2000)、COBOLの定義がX(100)とします。

表3.5 可変長項目の定義例（データベース列長＞COBOL項目長）

項目名	動作	動作内容
<任意の文字列>_CHAR_0100	○	データベースのVARCHAR(2000)と定義されている項目に対して、英数字の場合は先頭100文字分を、日本語の場合は先頭50文字分を対象に動作します。 ただし、データベースの可変長項目内に、後方空白を除いて、COBOLの項目長より長いデータが存在した場合、データ読込み時に右端が切断され、キーとして使用すると、正しく位置付けできません。このため、直接データベースのデータを入力／更新する場合、COBOLの項目長の範囲内で操作してください。
<任意の文字列>_CHAR_0010 または、 <任意の文字列>_CHAR_1000	×	列名に定義する項目長がCOBOLの定義と一致していない場合、エラーは発生しませんが、データの読み書きが正しく動作しません。
<任意の文字列>_CHAR	×	列名に項目長の定義がないため、COBOLの項目長は2000バイトであるとみなして動作します。そのため、レコード長の不一致となり、オープン時にエラーになります。

- データベースの列長とCOBOLの項目長が等しい場合

  以下の表では、データベースの定義がVARCHAR(2000)、COBOLの定義がX(2000)とします。

表3.6 可変長項目の定義例（データベース列長＝COBOL項目長）

項目名	動作	動作内容
<任意の文字列>_CHAR_2000	○	データベースのVARCHAR(2000)と定義されている項目に対して、英数字の場合は先頭2000文字分を、日本語の場合は先頭1000文字分を対象に動作します。 ※データベースの定義がCHAR(2000)と定義した場合と同様の動作になります。
<任意の文字列>_CHAR	○	

- データベースの列長よりCOBOLの項目長が長い場合

  以下の表では、データベースの定義がVARCHAR(1000)、COBOLの定義がX(2000)とします。

表3.7 可変長項目の定義例（データベース列長＜COBOL項目長）

項目名	動作	動作内容
<任意の文字列>_CHAR_1000	×	どのように定義してもレコード長の不一致となり、オープン時にエラーになります。
<任意の文字列>_CHAR_2000	×	
<任意の文字列>_CHAR	×	

【ファイル記述項からテーブルまたはビューの作成例】

以下に、ファイル記述項からテーブルまたはビューを作成する例について説明します。

1. ファイル記述項(FD句)のレコード記述項の例を以下に示します。

ファイル記述項（FD句）のレコード記述

```
FILE-CONTROL.
  SELECT T1 ASSIGN TO FILE1
            ORGANIZATION IS INDEXED
            ACCESS MODE IS RANDOM
            RECORD KEY IS C1.

＜略＞

FILE SECTION.
FD T1.
01     REC.
  02   C1      PIC X(10).
  02   C2      PIC N(10).
  02   C3      PIC 9(4)   PACKED-DECIMAL.
  02   GRP.
    03 C4      PIC S9(5)  COMP.
    03 C5      PIC X(3).
  02   C6      PIC X(2)   OCCURS  2.
  02   C7      PIC X(6).
  02   C7GRP   REDEFINES  C7.
    03 C7R1    PIC X(2).
    03 C7R2    PIC S9(4).
  02   C8      PIC S9(5)V9(2) PACKED-DECIMAL.
```

2. レコード記述項のレコード構造のレイアウトは以下のようになります。

   下線部のデータ項目が最下位レベル番号の基本項目になります。

レコード構造

C1	C2	C3	GRP		C6_1	C6_2	C7GRP X(6)		C8
			C4	C5			C7R1	C7R2	
X(10)	N(10)	9(4)PACK	S9(5)COMP	X(3)	X(2)	X(2)	X(2)	S9(4)	S9(5)V9(2) PACK

3. 「表3.2 ファイルとテーブルまたはビューの対応」と「3.1.3.5 列定義の対応」に基づいて、以下のSQL定義文(CREATE文)を作成します。

   以下の2通りで作成できます。

   ー テーブルのみを作成

   ー テーブルとビューを作成

索引ファイルで定義されているので、インデックスは必ず作成します。

テーブルのみを作成する場合

```
CREATE TABLE T1 ( ←(1)
 C1_CHAR      CHAR(10) NOT NULL,
 C2_NCHAR     CHAR(20) NOT NULL,
 C3_UNSIGN_DECIMAL   DECIMAL(4,0) NOT NULL,
 C4_INTEGER   INETGER NOT NULL,
 C5_CHAR      CHAR(3) NOT NULL,
 C6_1_CHAR    CHAR(2) NOT NULL, ←(4)
 C6_2_CHAR    CHAR(2) NOT NULL, ←(4)
 C7R1_CHAR    CHAR(2) NOT NULL,
 C7R2_NUMERIC NUMERIC(4,0) NOT NULL,
 C8_DECIMAL   DECIMAL(7,2) NOT NULL
)

CREATE UNIQUE INDEX T1_IX1
 ON T1 (C1_CHAR) ←(6)
```

テーブルとビューを作成する場合

```
CREATE TABLE T1 (
 C1    CHAR(10) NOT NULL,
 C2    CHAR(20) NOT NULL,
 C3    DECIMAL(4.0) NOT NULL,
 C4    INTGER NOT NULL,
 C5    CHAR(3) NOT NULL,
 C6_1  CHAR(2) NOT NULL, ←(4)
 C6_2  CHAR(2) NOT NULL, ←(4)
 C7R1  CHAR(2) NOT NULL,
 C7R2  NUMERIC(4,0) NOT NULL,
 C8    DECIMAL(7,2) NOT NULL
)

CREATE VIEW V1 (←(2)
 C1_CHAR,
 C2_NCHAR,
 C3_UNSIGN_DECIMAL,
 C4_INTEGER,
 C5_CHAR,
 C6_1_CHAR,
 C6_2_CHAR,
 C7R1_CHAR,
 C7R2_NUMERIC,
 C8_DECIMAL
) AS
 SELECT C1,C2,C3,C4,C5,C6_1,C6_2,C7R1,C7R2,
  C8 FROM T1

CREATE UNIQUE INDEX T1_IX1
 ON T1 (C1) ←(6)
```

COBOL初期化ファイルに指定する TableNameを求めます。

ー テーブルのみを作成する場合

(1)テーブル名 T1 が COBOL初期化ファイルに指定する TableName（FILE1=TableName=T1）になります。

ー テーブルとビューを作成する場合

(2)ビュー名 V1 が COBOL初期化ファイルに指定する TableName（FILE1=TableName=V1）になります。

(3)列名は、FD句のデータ項目名を使用しています。

(4)OCCURS句で定義している繰返し項目 C6 の列名は、"_"＋添字番号にしています。

(5)すべての列にNOT NULL制約を定義しています。

(6)レコードキーに対して、インデックスを定義しています。

4. 作成されたテーブルは以下のレコード構造になります。

テーブルまたはビューのレコード構造

C1	C2	C3	GRP		C6_1	C6_2	C7GRP X(6)		C8
			C4	C5			C7R1	C7R2	
X(10)	N(10)	9(4) PACK	S9(5)COMP	X(3)	X(2)	X(2)	X(2)	S9(4)	S9(5)V9(2) PACK
CHAR(10)	CHAR(20)	DECIMAL(4,0)	INTEGER	CHAR(3)	CHAR(2)	CHAR(2)	CHAR(2)	NUMERIC(4,0)	DECIMAL(7,2)

5. SQL定義文を実行します。

   osqlユーティリティなどを使用して実行してください。

3.1.3.6 COBOLと関連付けのない項目の指定方法

COBOLからアクセスできないデータベースの列を指定できます。

本指定により、キー指定なしの順アクセス実行時、アクセス順序をCOBOLとは別に指定できます。

例えば、データベースの列定義でCOBOLと関連付けのない項目に対し、オートナンバー型などを使用し、キー指定せずにシーケンシャルアクセスを実行した場合、データのアクセス順序をデータベースへのデータ格納順にすることができます。

COBOLと関連付けのない項目の定義方法を以下に示します。

データベースの列名	説明
<文字列>**_NOITEM**	COBOLと関連付けのない項目です。
<文字列>**_NOITEMK**	COBOLと関連付けのない項目です。本項目にはアクセスしませんが、キー指定せずに順アクセスを実行した場合、キーとして評価します。 キー指定の場合、_NOITEMと同様の扱いとなります。

文字列 :任意の文字列です。

注意

COBOLと関連付けのない項目について、以下の注意事項があります。

1. 他のサフィックスと同時に指定できません。

   同時に指定された場合は、最後に指定されたサフィックスが有効になります。

列名	動作内容
<任意の文字列>_NOITEM_CHAR	COBOLの型を表わすサフィックス(_CHAR)が有効になります。
<任意の文字列>_CHAR_NOITEM	_NOITEMが有効になります。(この項目は無視されます)
<任意の文字列>_CHAR_1000_NOITEMK	_NOITEMKが有効になります。 (この項目は無視されますが、キー指定なしの順アクセス時にキーとして使用します。)

2. レコード追加するファイルでCOBOLに関連付けない項目を定義する場合、関連付けない項目に該当するデータベースの列には、デフォルト値が設定されるようにデータベースを作成してください。

3.1.4 PowerRDBconnector 動作環境ファイル

PowerRDBconnector 動作環境ファイルは、ファイル名 "DBIO_ENV" で作成し、以下のプロパティを記述してください。

表3.8 PowerRDBconnectorの動作環境ファイルの書式

```
; PowerRDBconnectorの動作環境
ServerName=<データベースのサーバ名>
DataSourceName=<データベース名>
TimeOut=<タイムアウト時間>
Suppress=<後方空白補正>
DataCheck=<データチェック>
PrepareMode=<SQL文準備モード>
ErrorLog=<エラーログの出力先>
TraceMode=<トレースモード>
TraceSize=<トレースファイルのサイズ>
TraceLevel=<トレースのレベル>
```

- PowerRDBconnector動作環境ファイル（DBIO_ENV）の格納先は、以下のいずれかで設定します。

  1. 以下の環境変数にPowerRDBconnector動作環境ファイルのファイル名を含むフルパスで設定

     - 32ビット動作時

       環境変数名：DBIO_ENV

       設定例：DBIO_ENV=C:¥apl¥DBIO_ENV

     - 64ビット動作時

       環境変数名：DBIO_ENV_x64

       設定例：DBIO_ENV_x64=C:¥apl¥DBIO_ENV

     動作環境の設定内容は、サーバ一意の設定となります。

  2. 実行プログラムと同じディレクトリに格納

     動作環境の設定内容は、実行プログラム（EXEファイル）を格納するディレクトリで一意の設定となります。

     パス名には、JIS2004で追加された文字などのシフトJIS範囲外の文字を含めないでください。

  3. アプリケーション起動時のカレントパスに格納

     動作環境の設定内容は、実行プログラムを呼び出す業務アプリケーションや利用者の実行環境ごとに任意の設定となります。

  PowerRDBconnectorは、ファイルのオープン時に、PowerRDBconnector動作環境ファイル（DBIO_ENV）を、上記の順序で検索します。

- ASCIIコード、シフトJISコード（JIS X 0208-1990）またはUTF-8コードのテキストファイルで記述します。

  ただし、UTF-8の文字コードで作成する場合は、以下の点に注意してください。

ー BOM付きのUTF-8の文字コード系のテキストファイルとして作成してください。

ー PowerRDBconnector動作環境ファイルの内容はシフトJIS範囲内の文字で記述してください。

・ プロパティ名は行の先頭から始めてください。

・ プロパティ名と値は = (等号)でつなぎます。= の前後に空白、タブなどは記述できません。プロパティ名のみまたはプロパティ名と=だけで、値の指定のない記述はエラーとなります。

・ 大文字と小文字を区別します。

・ 行先頭に ; (セミコロン)文字がある行はコメント行とします。行途中の ; 文字はコメントとなりません。

・ アプリケーションを実行するユーザーからの読取り権限が必要です。

・ 32ビット動作時、ASP.NETで使用する場合、以下のいずれかの格納先を設定してください。

1. 環境変数DBIO_ENVにPowerRDBconnector動作環境ファイルのファイル名を含むフルパスで設定

   環境変数を変更した場合、Windowsの再起動が必要です。

2. Internet Information Service(IIS)のカレントパスに格納

   初期値は「x:¥Windows¥system32¥Inetsvr」または「x:¥Windows¥system32」です。カレントパスを変更する場合、アプリケーションで以下のように記述してください。なお、カレントパスは、セッション開設前に設定してください。

   (記述例)
   CLASS CLASS-ENVIRONMENT AS "System.Environment"
   PROPERTY PROP-CURRENTDIRECTORY AS "CurrentDirectory"
   SET PROP-CURRENTDIRECTORY OF CLASS-ENVIRONMENT TO "変更先パス名".

   カレントパスはスレッドではなく、プロセスで1つです。COBOLプログラム内でカレントパスを変更すると、PowerRDBconnector動作環境ファイルを正しく読み込めないことがあるため、以下のいずれかの対処を行ってください。

   - IISの1プロセス内でカレントパスを変更しない場合

     PowerRDBconnector動作環境ファイルごとにIISのプロセスを分離してください。
     1つのIISのプロセス内で動作するスレッドでは、同じカレントパスを使用してください。詳細は、IISのマニュアルやヘルプを参照してください。

   - IISの1プロセス内でカレントパスを変更する場合

     カレントパスの変更により、異なるPowerRDBconnector動作環境ファイルを参照する場合は、カレントパスの変更からスレッド初回のOPEN文まで、スレッドのロックを行ってください。

・ MeFt/Webで使用する場合、PowerRDBconnector動作環境ファイルの格納先は、ファイル名を含むフルパスで以下のシステム環境変数に設定してください。

ー 32ビット動作時
  システム環境変数名:DBIO_ENV

ー 64ビット動作時
  システム環境変数名:DBIO_ENV_x64

システム環境変数に設定しない場合、利用者プログラムは正しく動作しません。

各プロパティの指定方法を以下に示します。

PowerRDBconnector 動作環境ファイルのプロパティ

表3.9 PowerRDBconnector 動作環境ファイルのプロパティ

プロパティ名	指定有無	値と意味
ServerName	**必須**	SQL Serverデータベースが存在するサーバのホスト名または、IPアドレスを半角255文字以内で指定します。記号として半角アンダーバー(_)、半角ハイフン(-)、半角円記号(¥)、コロン(:)、半角ドット(.)が使用できます。日本語の混在指定も可能です。半角円記号(¥)だけを指定することはできません。
DataSourceName	**必須**	データベース名を指定します。半角英数字64文字以内で指定します。先頭に数字は指定できません。 ※PowerRDBconnectorでは、ODBCデータソースアドミニストレータでODBC環境を設定する必要はありません。 ※DataSourceNameプロパティで指定する値は、SQL Serverデータベースのデータベース名であり、ODBCデータソースアドミニストレータでのデータソース名ではありません。
TimeOut	**選択**	データベースの処理完了待合わせ時間を秒単位で指定します。レコードロックの獲得を待合わせタイムアウト時間としても使用されます。 ・ **0** データベースの処理が完了するまで待ち続けます。 ・ **1～600** データベースの処理が完了するまで、指定時間だけ待合わせます。指定時間経過後、獲得できない場合、エラー復帰します。 省略した場合には、30(秒)が指定されたものとして動作します。 データベースの処理完了の待合わせについては、「3.6 タイムアウト機能」を参照してください。
Suppress	**選択(注意が必要)**	後方空白補正の有無を指定します。以下に指定値と意味を示します。 ・ **"ON/FULL"** 後方空白補正を行います。 日本語項目に対しては、READ時に全角空白を補正します。 ・ **"ON/HALF"** 後方空白補正を行います。 日本語項目に対しては、シフトJISで動作するCOBOLの場合はREAD時に全角空白を補正しますが、unicodeで動作するCOBOLの場合はREAD時に半角空白を補正します。 ・ **"ON"** 後方空白補正を行います。"ON/HALF"の指定と同じ意味です。 ・ **“OFF”** 後方空白補正を行いません。 省略した場合は、ON/HALFが指定されたものとみなします。 COBOL初期化ファイルでSuppressプロパティが指定された場合は、COBOL初期化ファイルの値が有効となります。 詳細は、「3.7.1 後方空白補正」を参照してください。
DataCheck	**選択(注意**	データのチェックを行うか否かを指定します。以下に指定値と意味を示します。 ・ **“NONE”**

プロパティ名	指定有無	値と意味
	が必要)	データチェックを行わず、データの補正を行います。 ・“C” 英数字項目と日本語項目のデータチェックを行います。 ・“U” 外部十進項目のデータチェックを行います。 ・“P” 内部十進項目のデータチェックを行います。 複数指定する場合は、”/”で区切って指定してください。 例:文字チェックと内部十進数チェックを行う場合 DataCheck=C/P 省略した場合は、C/U/Pが指定されたものとみなします。 データ内容のチェックを行わない場合、項目属性に違反するデータは補正されます。詳細は、「3.7.2 項目属性に違反するデータのチェックと補正」を参照してください。
PrepareMode	選択 (注意が必要)	SQL文準備モードを指定します。以下に指定値と意味を示します。 ・“OPEN” 使用する可能性のあるSQL文をOPEN時にすべて準備します。 ・“START” RECORD KEYやALTERNATE RECORD KEYを使用した初回のキー検索時(OPEN後の最初のSTART、乱READ、乱REWRITE、または、乱DELETE)に使用するSQL文を準備します。 省略した場合は、STARTが指定されたものとみなします。 OPENを指定すると、RECORD KEYやALTERNATE RECORD KEYのキー検索で使用するSQL文をOPEN文の延長で準備しておくことができますが、OPEN文の処理性能は遅くなります。これは、PowerRDBconnector V3.0までの動作となります。

トラブルが発生したとき、PowerRDBconnector内部調査用の情報を取得する場合に、以下のプロパティを指定してください。これらのプロパティを設定すると、性能が著しく低下する場合があります。通常の運用では設定しないでください。

表3.10 トラブル発生時に必要なPowerRDBconnector 動作環境ファイルのプロパティ

プロパティ名	指定有無	値と意味
ErrorLog	選択	エラーログおよびトレース情報の格納ディレクトリを指定します。 ErrorLogプロパティが指定されると、指定したディレクトリにログファイル(FJSVdbio.log)を作成し、エラーログを採取します。既に存在している場合は追記します。 エラーログの最大サイズは256Kバイトで2世代分のエラーログを保持します。なお、COBOLアプリケーションが異常終了するような場合にのみエラーログを採取するため、極端に大きくなることはありません。 エラーログは、PowerRDBconnectorの内部情報であり、内容は可変です。当社技術員からの依頼により採取してください。

プロパティ名	指定 有無	値と意味
		格納ディレクトリを指定するときの注意事項を以下に示します。 ・ ドライブ直下は指定できません。 ・ 半角英数字、半角記号の文字を使って、200バイト以内の値で指定してください。ただし、半角記号は ¥ | : * ? " < > / . を除きます。 ・ 全角文字を含むディレクトリは指定しないでください。指定すると格納先を保証できません。 ・ 実行プログラム（EXEファイル）を格納するディレクトリからの相対パスも指定可能です。相対指定は、".¥"で開始します。 例）ErrorLog=.¥ ただし、絶対パスに変換した場合に、200バイト以内になるように指定してください。 ・ 環境変数は指定できません。 ・ 存在するディレクトリで、書込み権限のあるディレクトリを指定してください。 ・ 存在しないディレクトリを指定した場合、エラーは通知されず、イベントログに警告のメッセージが通知されます。このとき、エラーログおよびトレース情報は採取されません。 ・ 書込み権限のないディレクトリを指定した場合、Windows 7（x64含む）、Windows Vista、Windows Server 2008（x64含む）、およびWindows Server 2008 R2では、ファイルの仮想化(リダイレクション)が生じることがあります。リダイレクションが生じた場合は、以下のイベントログに出力されます。 アプリケーションとサービス → Microsoft → Windows → UAC-FileVirtualization
TraceMode	**選択**	トレース情報の採取有無を指定します。以下に指定値と意味を示します。 ・ **“ON”** トレース情報を採取します。 ・ **“OFF”** トレース情報を採取しません。 トレースを採取する場合、本プロパティとErrorLogプロパティの2つを指定してください。 トレースは、ErrorLogプロパティで指定したログ格納ディレクトリにトレースファイル（FJSVdbio_trc.log）を作成し、トレース情報を採取します。ErrorLogプロパティまたはTraceModeプロパティの設定を省略した場合、トレース情報は採取されません。 TraceModeプロパティの指定に誤りがあった場合、エラーは通知されず、トレース情報は採取されません。このため、1度COBOLアプリケーションを実行し、トレース情報が採取されていることを確認してください。
TraceSize	**選択**	トレース情報の最大サイズをKバイト単位で指定します。 1から204800の値を指定します。 省略された場合、4,096（Kバイト）が指定されたものとして動作します。 なお、ErrorLogプロパティで指定されたディレクトリに2世代分のトレース情報を格納するため、TraceSizeで指定したサイズの2倍の空き容量(ディスク)が必要となります。 TraceModeプロパティを省略または値にOFFを設定した場合、TraceSizeプロパティの設定は無効です。

プロパティ名	指定 有無	値と意味
		TraceSizeプロパティの指定に誤りがあった場合、エラーは通知されません。204,800（Kバイト）が指定されたものとして動作します。ただし、数値以外の文字が指定された場合、エラーは通知されず、トレース情報は採取されません。 **注意** トレース情報は、COBOLアプリケーションを単体実行して採取してください。なお、COBOLアプリケーションが多重動作する環境では、最大でTraceSize×2までトレース情報を格納します。
TraceLevel	**選択 (注意が必要)**	トレースを採取する場合に、トレースの出力レベルを指定します。 以下に指定値と意味を示します。 ・ **0** アクセス情報のみ出力します。 ・ **1** 0に加え、列情報およびイベントログ内容を出力します。 ・ **2** 1に加え、エラー発生時のデータ内容を出力します。 ・ **3** 2に加え、アクセス時のキー値の内容および入出力データの内容を出力します。 省略した場合は、0が指定されたものとみなします。 トレース内容の詳細は、「5.2 トレース出力機能」を参照してください。

PowerRDBconnector 動作環境ファイル（DBIO_ENV）の記述例

SQL Serverを構築するサーバのホスト名が**snake**、データベース名が**pubs**の記述です。

PowerRDBconnector動作環境ファイル(DBIO_ENV) の記述例

```
; PowerRDBconnector動作環境ファイル
ServerName=snake.domain.com
DataSourceName=pubs
```

3.1.5 COBOL初期化ファイル

COBOL初期化ファイルには、ASSIGN句のファイル識別名とデータベースのテーブル、ビューとの関連付けを行います。

関連付けの指定方法として、次の方法があります。

- COBOL初期化ファイルに指定する方法
- 環境変数に指定する方法
- Webの場合には実行環境ユーティリティでファイル（web.config）に指定する方法

本マニュアルでは、PowerRDBconnectorを使用して、COBOLのファイル識別名とデータベースのテーブルやビューを関連付ける定義例を記載します。ファイル識別名以外の定義全般については、NetCOBOL のマニュアルを参照してください。

COBOL初期化ファイルの書式

```
ファイル識別名=TableName=<テーブル名またはビュー名>&SchemaName=<スキーマ名>&AccessMode=<アクセスモード>&TableLock=<テーブルロック>,ファイル識別名定数
```

- テーブルおよびビューごとにファイル識別名を記述します。

- ASCIIコード、シフトJISコード(JIS X 0208-1990)またはUTF-8コードのテキストファイルで記述します。

  ただし、UTF-8の文字コードで作成する場合は、以下の点に注意してください。

  - BOM付きのUTF-8の文字コード系のテキストファイルとして作成してください。
  - COBOL初期化ファイルの内容はシフトJIS範囲内の文字で記述してください。
  - スキーマ名、表名に、JIS X 0213:2004に対応した文字セット(JIS2004)で追加された文字などのシフトJIS範囲外の文字は、エラーとなることがあるため、使用しないでください。
  - UTF-8コードのCOBOL初期化ファイルは、NetCOBOL V10および NetCOBOL for .NET V4.0から使用できます。

- プロパティ名と値は = でつなぎます。= の前後に空白、タブなどは記述できません。

- プロパティ名=値 は & でつなぎます。& の前後に空白、タブなどは記述できません。

- 大文字・小文字を区別します。

- ファイル識別名定数の前は、, (カンマ)で区切ります。

各変数の指定方法を以下に示します。

COBOL初期化ファイル

表3.11 COBOL初期化ファイルのプロパティ

プロパティ名	指定有無	値と意味
ファイル識別名	必須	SELECT句(ファイル記述項)のASSIGN句に指定したファイル識別名を指定します。 COBOLのファイル名をデータベースのテーブルまたはビューに割り付けます。
TableName	必須	ファイル識別名に対するテーブル名またはビュー名を指定します。 以下の文字数以内で指定します。 ・ 半角英数字、半角記号(アンダーバー(_)、ハイフン(-)、円記号(¥))で128文字以内 ・ 全角で64文字以内 テーブル名、ビュー名には全角文字と半角文字を混在して指定できません。 また、先頭に数字は指定できません。 SQL Serverのシノニムは指定できません。
SchemaName	必須	ファイル識別名に対するテーブルの所有者名を指定します。

プロパティ名	指定有無	値と意味
		半角英数字、記号で128文字以内の値を指定します。指定可能な文字はSQL Serverの仕様に準じます。 また、先頭に数字は指定できません。
AccessMode	必須（32ビット） **選択（64ビット）**	ファイル識別名に対するアクセスモードを指定します。 以下に指定値と意味を示します。 ・ **"SEQUENTIAL"** 順呼出し ・ **"RANDOM"** 乱呼出し ・ **"DYNAMIC"** 動的呼出し 本モードはCOBOLアプリケーションのACCESS MODE句に指定する値と同意です。32ビット動作と64ビット動作で扱いが異なります。 ・ 32ビット動作時 本モードは必ず指定してください。入出力文COBOLアプリケーションのACCESS MODE句の指定と異なる場合は、動作保証できません。 ・ 64ビット動作時 本モードを指定する必要はありません。 本モードを指定しない場合、入出力文COBOLアプリケーションのACCESS MODE句の指定で動作します。 本モードを指定し、入出力文COBOLアプリケーションのACCESS MODE句の指定と異なる場合は、OPEN時にエラーとなります。
Truncate	選択（注意が必要）	OUTPUTモードでビューをオープンする場合、導出元テーブル名を指定します。 Truncateを使用する場合の注意事項については、「4.2.3.5 OUTPUTモードのオープンの使用について」を参照してください。
TableLock	選択	テーブルロックの使用有無を指定します。以下に指定値と意味を示します。 ・ **"ON"** テーブルロックを行います。 ・ **"OFF"** テーブルロックは行わず、レコードロックとなります。 省略した場合、OFF（レコードロック）を指定したものとして動作します。 テーブルロックについて、詳しくは「3.5.3 テーブルロック機能」を参照してください。
Suppress	選択	後方空白補正の有無を指定します。以下に指定値と意味を示します。 ・ **"ON/FULL"** 後方空白補正を行います。 日本語項目に対しては、READ時に全角空白を補正します。 ・ **"ON/HALF"**

プロパティ名	指定 有無	値と意味
		後方空白補正を行います。 日本語項目に対しては、シフトJISで動作するCOBOLの場合はREAD時に全角空白を補正しますが、unicodeで動作するCOBOLの場合はREAD時に半角空白を補正します。 ・ **"ON"** 後方空白補正を行います。"ON/HALF"の指定と同じ意味です。 ・ **"OFF"** 後方空白補正を行いません。 省略した場合は、PowerRDBconnector動作環境ファイルで指定されたSuppressプロパティが有効となります。 詳細は、「3.7.1 後方空白補正」を参照してください。
ファイル識別名定数	必須	識別値 **"RDM"** を指定します。
エントリ情報	選択	動的プログラム構造で、トランザクションサブルーチン、認証情報登録サブルーチンまたは、トレースへの文字出力サブルーチンを使用する場合は、エントリ情報を記述してください。エントリ情報はCOBOL初期化ファイル（COBOL85.CBR）に記述します。 **@CBR_ENTRYFILE=エントリ情報ファイル名** エントリ情報ファイルの内容は、以下の形式で記述します。 ・ 32ビット動作時 ー シングルセッション時 **[ENTRY]** **;トランザクションサブルーチン** **XMROTSTR=F3BWS1CB.DLL** **XMROTEND=F3BWS1CB.DLL** **XMROTCNL=F3BWS1CB.DLL** **XMROTRBK=F3BWS1CB.DLL** **; 認証情報登録サブルーチン** **XMROAUTH=F3BWS1SB.DLL** ー マルチセッション時 **;セッションサブルーチン** **COB_PRDB_START=F3BIEFNC.dll** **COB_PRDB_END=F3BIEFNC.dll** **COB_PRDB_CHG=F3BIEFNC.dll** **;トランザクションサブルーチン** **COB_PRDB_TRAN=F3BIEFNC.dll** **; 認証情報登録サブルーチン** **COB_PRDB_AUTH=F3BIEFNC.dll** ・ 64ビット動作時 ー シングルセッション時 **[ENTRY]** **;トランザクションサブルーチン** **XMROTSTR=F4ARS1CB_64.DLL** **XMROTEND=F4ARS1CB_64.DLL** **XMROTCNL=F4ARS1CB_64.DLL** **XMROTRBK=F4ARS1CB_64.DLL** **;認証情報登録サブルーチン** **XMROAUTH=F4ARS1SB_64.DLL**

プロパティ名	指定 有無	値と意味
		;トレース文字出力サブルーチン **XMROLOG=F4ARS1SB_64.DLL** ー マルチセッション時 **[ENTRY]** **;セッションサブルーチン** **COB_PRDB_START=F4AGEFNC.dll** **COB_PRDB_END=F4AGEFNC.dll** **COB_PRDB_CHG=F4AGEFNC.dll** **;トランザクションサブルーチン** **COB_PRDB_TRAN=F4AGEFNC.dll** **; 認証情報登録サブルーチン** **COB_PRDB_AUTH=F4AGEFNC.dll** **; トレース文字出力サブルーチン** **COB_PRDB_LOG=F4AGEFNC.dll** 詳しくは、NetCOBOLのマニュアルを参照してください。

COBOL初期化ファイル(COBOL85.CBR)の記述例

COBOLの索引ファイルのファイル識別名 **EMPLOYEE** をテーブル名 **employee** で所有者(スキーマ名) **dbo** に、COBOLの索引ファイルのファイル識別名 **CUSTOMER** をテーブル名 **customer** で所有者(スキーマ名) **dbo** に対応付け、エントリ情報ファイルに **ENTRY.ENT**を使用するための記述です。

(1)COBOL初期化ファイル(COBOL85.CBR)の記述例

```
EMPLOYEE=TableName=employee&SchemaName=dbo&AccessMode=RANDOM&Suppress=OFF,RDM
CUSTOMER=TableName=customer&SchemaName=dbo&AccessMode=RANDOM&Suppress=OFF,RDM
@CBR_ENTRYFILE=ENTRY.ENT
```

(2)エントリ情報ファイル(ENTRY.ENT)の記述例

表3.12 エントリ情報ファイルの記述例

	シングルセッション	マルチセッション
32ビット動作時	[ENTRY] ;トランザクションサブルーチン XMROTSTR=F3BWS1CB.DLL XMROTEND=F3BWS1CB.DLL XMROTCNL=F3BWS1CB.DLL XMROTRBK=F3BWS1CB.DLL ;認証情報登録サブルーチン XMROAUTH=F3BWS1SB.DLL	[ENTRY] ;セッションサブルーチン COB_PRDB_START=F3BIEFNC.dll COB_PRDB_END=F3BIEFNC.dll COB_PRDB_CHG=F3BIEFNC.dll ;トランザクションサブルーチン COB_PRDB_TRAN=F3BIEFNC.dll ; 認証情報登録サブルーチン COB_PRDB_AUTH=F3BIEFNC.dll
64ビット動作時	[ENTRY] ;トランザクションサブルーチン XMROTSTR=F4ARS1CB_64.DLL XMROTEND=F4ARS1CB_64.DLL XMROTCNL=F4ARS1CB_64.DLL XMROTRBK=F4ARS1CB_64.DLL ;認証情報登録サブルーチン XMROAUTH=F4ARS1SB_64.DLL ;トレース文字出力サブルーチン XMROLOG=F4ARS1SB_64.DLL	[ENTRY] ;セッションサブルーチン COB_PRDB_START=F4AGEFNC.dll COB_PRDB_END=F4AGEFNC.dll COB_PRDB_CHG=F4AGEFNC.dll ;トランザクションサブルーチン COB_PRDB_TRAN=F4AGEFNC.dll ;認証情報登録サブルーチン COB_PRDB_AUTH=F4AGEFNC.dll ;トレース文字出力サブルーチン COB_PRDB_LOG=F4AGEFNC.dll

注意

エントリ情報ファイルは、NetCOBOL for Windowsの場合のみ必要です。

3.1.6 コンパイル方法

ここでは、翻訳オプションおよび、リンクの方法を説明します。

PowerRDBconnectorを使用するCOBOLアプリケーションをコンパイルする場合、必要に応じた翻訳オプションを設定し、NetCOBOLでコンパイルしてください。その後、NetCOBOLのリンカを利用してPowerRDBconnectorのサブルーチンをリンクします。

3.1.6.1 NetCOBOL for Windowsでのコンパイル方法

NetCOBOL for Windowsは、使用するNetCOBOL製品によって、32ビット動作と64ビット動作が使用できます。NetCOBOL for Windows使用時のコンパイル方法について説明します。

[翻訳オプション]

PowerRDBconnectorに関連する翻訳オプションを説明します。各オプションについての詳細は、「NetCOBOLのマニュアル」を参照してください。

表3.13 NetCOBOL for Windowsの翻訳オプション

翻訳オプション名	機能	注意事項
ASCOMP5	2進項目の解釈の指定	NONE以外を指定しないでください。
DLOAD	プログラム構造の指定	DLOAD:動的プログラム構造 NODLOAD:動的リンク構造
RCS	実行時コード系の指定	SJIS:シフトJIS UCS2:unicode(注1) UTF16:unicode/JIS2004(注2)
THREAD	マルチスレッドプログラム作成の指定	SINGLE:シングルセッションで使用してください。 MULTI:マルチセッションで使用してください。
SHREXT	ファイル共用の指定	SHREXT:ファイル共用 NOSHREXT:ファイル共用なし

(注1)

NetCOBOL V10の場合、RCSオプションのUCS2はUTF16と同じ意味です。しかし本書では、NetCOBOL V10以外のときと区別するため、NetCOBOL V10のRCSオプションにunicodeを指定する場合はUTF16と記載します。

(注2)

NetCOBOL V10から、RCSオプションに、UTF16が指定可能です。

例)翻訳オプションの指定例を以下に2つ示します。

- COBOLソースの先頭に、「@OPTIONS ASCOMP5(NONE),DLOAD,RCS(SJIS)」と記述します。
- 翻訳オプションに、「/wc,"ASCOMP5(NONE),DLOAD,RCS(SJIS)"」と記述します。

[リンク方法]

- [動的プログラム構造の場合]

  NetCOBOLの翻訳時に以下の翻訳オプションを指定します。

  翻訳オプション:DLOAD

  リンク方法について、詳しくはNetCOBOLのマニュアルを参照してください。

  さらに実行時、COBOL初期化ファイル(COBOL85.CBR)にエントリ情報(@CBR_ENTRYFILE)を指定します。エントリ情報の指定方法については、「3.1.5 COBOL初期化ファイル」を参照してください。

  なお、本製品が提供するサブルーチンは、PowerRDBconnector が提供するシステムライブラリです。COBOLアプリケーションに静的リンクしないでください。

- [動的リンク構造の場合]

  NetCOBOLの翻訳時に以下の翻訳オプションを指定します。

  翻訳オプション:NODLOAD

  さらに、以下のインポートライブラリをリンクしてください。

表3.14 シングルセッションプログラミングのインポートライブラリ

	ファイル名	格納ディレクトリ
32ビット動作	F3BWS1CB.LIB F3BWS1SB.LIB	・ SQL Server 2005の場合 PowerRDBconnectorインストールディレクトリ¥SQLSV2005 ・ SQL Server 2008の場合 PowerRDBconnectorインストールディレクトリ¥SQLSV2008
64ビット動作	F4ARS1CB_64.LIB F4ARS1SB_64.LIB	・ SQL Server 2008の場合 PowerRDBconnectorインストールディレクトリ¥SQLSV2008

  リンク方法について、詳しくはNetCOBOLのマニュアルを参照してください。

  なお、本製品が提供するサブルーチンは、PowerRDBconnectorが提供するシステムライブラリです。COBOLアプリケーションに静的リンクしないでください。

[使用可能なサブルーチン]

NetCOBOL for Windowsから以下のサブルーチンを呼び出すことができます。

- シングルセッションプログラミングの場合

表3.15 NetCOBOL for Windowsで使用可能なサブルーチン一覧(シングルセッション)

機能	サブルーチン名
トランザクションの開始	XMROTSTR
トランザクションの確定	XMROTEND
トランザクションの取消し	XMROTCNL
テーブルロック解除時のトランザクションの取消し	XMROTRBK
認証情報の登録	XMROAUTH
トレースへの文字出力(64ビットのときのみ)	XMROLOG

上記サブルーチンの使い方は、「3.4.2 トランザクションの使用方法(シングルセッション)」、「3.3.2 データベース認証の使用方法(シングルセッション)」を参照してください。

- マルチセッションプログラミングの場合

表3.16 NetCOBOL for Windowsで使用可能なサブルーチン一覧(マルチセッション)

機能	サブルーチン名
トランザクションの開始	COB_PRDB_TRAN
トランザクションの確定	
トランザクションの取消し	
テーブルロック解除時のトランザクションの取消し	
セッションの開設	COB_PRDB_START
セッションの閉設	COB_PRDB_END
セッションの変更	COB_PRDB_CHG
認証情報の登録	COB_PRDB_AUTH
トレースへの文字出力(64ビットのときのみ)	COB_PRDB_LOG

上記サブルーチンの使い方は、「3.4.3 トランザクションの使用方法(マルチセッション)」、「3.3.3 データベース認証の使用方法(マルチセッション)」を参照してください。

注意

マルチセッションプログラミングの場合の注意事項を以下に示します。

ー シングルセッションプログラミングで使用可能なサブルーチンとマルチセッションプログラミングで使用可能なサブルーチンは、混在して使用できません。

3.1.6.2 NetCOBOL for .NETでのコンパイル方法

NetCOBOL for .NETは、32ビット動作時のみ使用できます。NetCOBOL for .NET使用時のコンパイル方法について説明します。

[翻訳オプション]

PowerRDBconnectorの動作に影響のある翻訳オプションについて以下に示します。各オプションの使用方法については、NetCOBOLのマニュアルを参照してください。

表3.17 NetCOBOL for .NETの翻訳オプション

翻訳オプション名	機能	注意事項
ASCOMP5	2進項目の解釈の指定	NONE以外を指定しないでください。
RCS	実行時コード系の指定	SJIS:シフトJIS UTF8-UCS2:unicode SJIS-UCS2:英数字項目がシフトJIS、日本語項目がunicode
SHREXT	ファイル共用の指定	SHREXT:ファイル共用 NOSHREXT:ファイル共用なし

翻訳オプション名	機能	注意事項
platform	生成されるファイルが動作するプラットフォームの指定	通常省略できますが、Windows Server 2003 (x64)、Windows Server 2008 (x64)、Windows Server 2008 R2で動作させる場合は、x86と指定してください。

例）翻訳オプションの指定例を以下に2つ示します。

- COBOLソースの先頭に、「@OPTIONS ASCOMP5(NONE),SHREXT,RCS(SJIS)」と記述します。
- 翻訳オプションに、「/wc:" ASCOMP5(NONE),SHREXT,RCS(SJIS)"」と記述します。

NetCOBOL for .NETでコンパイル時の注意事項について

- NetCOBOL for .NETには、マルチスレッドプログラムについての翻訳オプションはありません。

  NetCOBOL for .NETでコンパイルした場合、マルチスレッドプログラムとなります。

- マルチスレッドプログラムでも実行環境によって、ASP.NETのようにWebで動作させる場合にはマルチスレッドとして動作したり、コンソールアプリケーションのようにシングルスレッド（プロセス）プログラムとして動作したりします。マルチスレッドのCOBOLアプリケーションからPowerRDBconnectorを呼び出す場合、セッションを制御してください（マルチセッションプログラム）。

［使用可能なサブルーチン］

セッションを制御しないシングルセッションプログラムと、マルチセッションプログラムとでは、サブルーチンの使用方法が異なります。それぞれ使い方を示します。

NetCOBOL for .NETから以下のサブルーチンを呼び出すことができます。

- シングルセッションプログラミングの場合

  「表3.18 シングルセッションプログラミングで使用可能なサブルーチン一覧」に示すサブルーチンが使用できます。

  なお、NetCOBOL for .NETから以下のサブルーチンを呼び出す場合は、プログラム原型を定義してください。プログラム原型については、NetCOBOLのマニュアルを参照してください。

  プログラム原型のサンプルについては、「付録C 開発用サンプル情報」を参照してください。

  表3.18 シングルセッションプログラミングで使用可能なサブルーチン一覧

機能	サブルーチン名
トランザクションの開始	XMROTSTR
トランザクションの確定	XMROTEND
トランザクションの取消し	XMROTCNL
テーブルロック解除時のトランザクションの取消し	XMROTRBK
認証情報の登録	XMROAUTH

  上記サブルーチンの使い方は、「3.4.2 トランザクションの使用方法（シングルセッション）」、「3.3.2 データベース認証の使用方法（シングルセッション）」を参照してください。

- マルチセッションプログラミングの場合

  マルチセッションプログラミングの場合、「表3.19 マルチセッションプログラミングで使用可能なサブルーチン一覧」に示すサブルーチンが使用できます。

  なお、NetCOBOL for .NETから以下のサブルーチンを呼び出す場合は、プログラム原型の定義は必要ありません。

表3.19 マルチセッションプログラミングで使用可能なサブルーチン一覧

機能	サブルーチン名
トランザクションの開始	COB_PRDB_TRAN
トランザクションの確定	
トランザクションの取消し	
テーブルロック解除時のトランザクションの取消し	
セッションの開設	COB_PRDB_START
セッションの閉設	COB_PRDB_END
認証情報の登録	COB_PRDB_AUTH

上記サブルーチンの使い方は、「3.4.3 トランザクションの使用方法（マルチセッション）」、「3.3.3 データベース認証の使用方法（マルチセッション）」を参照してください。

マルチセッションプログラミングの場合の注意事項を以下に示します。

- シングルセッションプログラミングで使用可能なサブルーチンとマルチセッションプログラミングで使用可能なサブルーチンは、混在して使用できません。
- COB_PRDB_CHGサブルーチンは、NetCOBOL for .NETでは使用できません。

3.2 セッションの制御方法

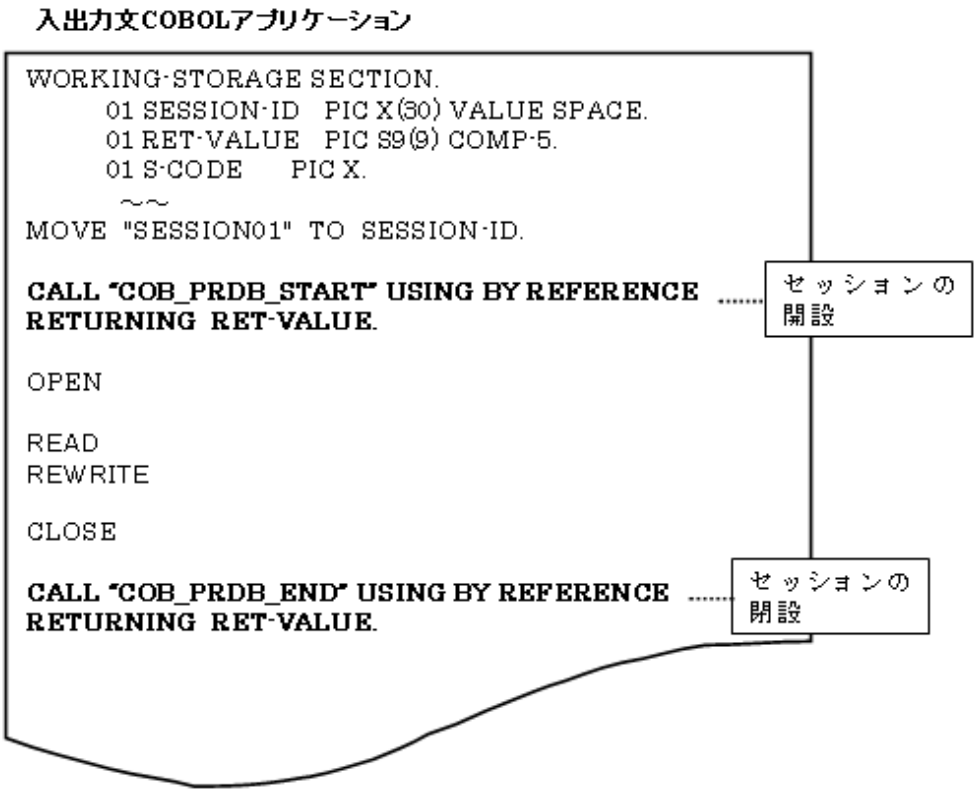
本節では、マルチセッションプログラミングスタイルで使用するセッションの制御方法について説明します。

3.2.1 セッションの制御方法

PowerRDBconnectorをマルチセッションプログラミングで使用する場合、アクセス文やサブルーチンが、どのセッションで動作するかを特定するために、セッションを指定します。

PowerRDBconnectorのセッションは、以下のように指定します。

図3.3 セッションサブルーチンの使用方法



COBOLアプリケーションから以下のサブルーチンをCALL文で呼び出すことで、セッションの制御ができます。

表3.20 セッション指定サブルーチンの一覧

機能	サブルーチン名
セッションの開設	COB_PRDB_START
セッションの閉設	COB_PRDB_END
セッションの変更	COB_PRDB_CHG

セッション開設後の初回OPEN文でデータベースに接続されます。セッション閉設時にデータベースから切り離されます。

使用例として「図3.4 セッションサブルーチンの使用例」、「3.2.3 セッションの使用例」を参照してください。

セッションサブルーチン使用時は、以下の点に注意してください。

- セッションサブルーチンを動的プログラム構造で使用する場合は、COBOL初期化ファイルにエントリ情報の記述が必要です。詳しくは、NetCOBOLのマニュアルを参照してください。

  例）エントリ情報ファイル（ENTRY.ENT）を指定する場合

  COBOL初期化ファイル（COBOL85.CBR)

  ```
  @CBR_ENTRYFILE=ENTRY.ENT
  ```

  エントリ情報ファイル（ENTRY.ENT）

  ・32ビット動作時

```
[ENTRY]
;セッションサブルーチン
COB_PRDB_START=F3BIEFNC.dll
COB_PRDB_END=F3BIEFNC.dll
COB_PRDB_CHG=F3BIEFNC.dll
```

・64ビット動作時

```
[ENTRY]
;セッションサブルーチン
COB_PRDB_START=F4AGEFNC.dll
COB_PRDB_END=F4AGEFNC.dll
COB_PRDB_CHG=F4AGEFNC.dll
```

セッションサブルーチンは、COBOLアプリケーションに静的リンクしないでください。詳しくは、「3.1.6 コンパイル方法」および「NetCOBOL 使用手引書」を参照してください。

- セッションを開設していないプログラムは、シングルセッションプログラムとして解釈されます。
- シングルセッションプログラミングの場合、セッションサブルーチンの呼出しは不要です。
- 1つのプログラムでシングルセッションとマルチセッションは混在できません。このため、マルチセッションのセッションを開設するプログラムは、セッションを閉設した後もシングルセッションプログラムとして動作できません。セッションを閉設した後もデータベースへのアクセスを継続する場合、再度セッションを開設する必要があります。
- セッションは、“セッションID”という任意の文字列で管理され、COBOLアプリケーションから、セッションサブルーチンのパラメーターとして指定します。
- PowerRDBconnectorへのセッションとデータベースへのセッションは、1対1に対応しています。
- 以下のNetCOBOLでのみ使用できます。それ以外のNetCOBOLでは使用できません。
  - NetCOBOL for .NET V3.0以降（32ビット動作）

    ただし、NetCOBOL for .NETでは、COB_PRDB_CHGサブルーチンは使用できません。
  - NetCOBOL for Windows V10.3以降（32ビット動作、64ビット動作）

3.2.2 セッションサブルーチンのインターフェース

セッションサブルーチンのインターフェースは、以下のとおりです。

3.2.2.1 セッション開設サブルーチンのインターフェース

- 機能

  PowerRDBconnectorへのセッションを開設します。

- 呼出し形式

```
CALL “COB_PRDB_START” USING BY REFERENCE セッションID RETURNING 復帰値.
```

- パラメーターのデータ定義

```
01 セッションID          PIC X(30).
01 復帰値                PIC S9(9) COMP-5.
```

※上記はすべて、必ずレベル番号01で記載してください。

- パラメーターの意味

  ー セッションID（必須）

  英数字項目で30バイト指定します。
  指定した文字列が30バイトに満たない場合、指定した文字列を含めた30バイトがセッションIDとして扱われます。

  文字の種類の制約はありません。

  なお、本パラメーターは必ず指定してください。

- 復帰値

  復帰値は、以下のとおりです。

表3.21 COB_PRDB_STARTサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常	なし
-1	エラー	あり
-2	内部矛盾	あり
-4	既に開設済みのセッションIDが指定されました。	なし
-200	以下のいずれかのエラーが発生しています。 ・ PowerRDBconnectorがインストールされていません。 ・ アクセスに必要なデータベース製品がインストールされていません。 ・ NetCOBOLの動作（32ビット動作、または64ビット動作）と異なっています。 ・ PATH環境変数にインストール先が正しく設定されていません。	なし
-201	使用できるセッション数の最大値を超えました。	なし

注意

本サブルーチンの注意事項を以下に示します。

- 本サブルーチンが実行された以降はマルチセッションプログラミングとして解釈されます。このため、以下のシングルセッションで使用するサブルーチンは使用できません。

  XMROTSTR、XMROTEND、XMROTCNL、XMROTRBK、XMROAUTH、XMROLOG

- 本サブルーチンではデータベースへの接続は行われません。データベースへの接続は初回オープン時に行われます。また、データベースへの接続はCOB_PRDB_ENDサブルーチンが呼び出されるまで保持されます。

3.2.2.2 セッション閉設サブルーチンのインターフェース

- 機能

  PowerRDBconnectorへのセッションを閉設します。

・ 呼出し形式

```
CALL “COB_PRDB_END” USING BY REFERENCE セッションID RETURNING 復帰値.
```

・ パラメーターのデータ定義

```
01 セッションID          PIC X(30).
01 復帰値                PIC S9(9) COMP-5.
```

※上記はすべて、必ずレベル番号01で記載してください。

・ パラメーターの意味

ー セッションID(必須)

英数字項目で30バイト指定します。

文字の種類の制約はありません。

セッション開設サブルーチンで指定したセッションIDを指定してください。

・ 復帰値

復帰値は、以下のとおりです。

表3.22 COB_PRDB_ENDサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常	なし
-1	エラー	あり
-2	内部矛盾	あり
-201	セッションIDの指定が正しくありません。	なし

注意

本サブルーチンの注意事項を以下に示します。

・ 本サブルーチンが実行されても、プロセスが終了するまではマルチセッションプログラミングとして解釈されます。このため、以下のシングルセッションで使用するサブルーチンは使用できません。

XMROTSTR、XMROTEND、XMROTCNL、XMROTRBK、XMROAUTH、XMROLOG

・ 本サブルーチンは、同一セッションでオープンしたすべてのファイルをクローズした後、呼び出してください。未クローズのファイルがある場合、本サブルーチンは、エラーとなります。

・ 同一セッションで本サブルーチンを複数回呼び出した場合、2回目以降の呼出しは無効となり正常終了します。

・ 指定されたセッションIDが一度も開設されていない場合、本サブルーチンはエラー終了します。

3.2.2.3 セッション変更サブルーチンのインターフェース

・ 機能

PowerRDBconnectorへのセッションを変更します。

- 呼出し形式

```
CALL “COB_PRDB_CHG” USING BY REFERENCE セッションID RETURNING 復帰値.
```

- パラメーターのデータ定義

```
01 セッションID          PIC X(30).
01 復帰値                PIC S9(9) COMP-5.
```

※上記はすべて、必ずレベル番号01で記載してください。

- パラメーターの意味

ー セッションID(必須)

英数字項目で30バイト指定します。

文字の種類の制約はありません。

セッション開設サブルーチンで指定したセッションIDを指定してください。

- 復帰値

復帰値は、以下のとおりです。

表3.23 COB_PRDB_CHGサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常	なし
-1	エラー	あり
-201	セッションIDの指定が正しくありません。	なし

注意

NetCOBOL for .NETでは、COB_PRDB_CHGサブルーチンは使用できません。

セッションサブルーチンでエラーが発生した場合は、エラー情報がWindowsのイベントログ(アプリケーションログ)へ出力されます。エラー情報は、「5.1.4 セッションサブルーチンのエラー情報」を参照してください。

図3.4 セッションサブルーチンの使用例

COBOLソース

```
DATA DIVISION.
WORKING-STORAGE  SECTION.
01 SESSION-ID   PIC X(30) VALUE SPACE.
01 RET-VALUE    PIC S9(9) COMP-5.
01 RET          PIC S9(9) COMP-5 VALUE 0.
＜略＞
PROCEDURE DIVISION.

MOVE  "SESSION01"  TO  SESSION-ID.
* セッションを開設する
CALL "COB_PRDB_START" USING BY REFERENCE
   SESSION-ID RETURNING  RET-VALUE.

OPEN I-O INFILE.

＜略＞
…
CLOSE INFILE.
* セッションを開設する。
CALL "COB_PRDB_END" USING BY REFERENCE
   SESSION-ID RETURNING  RET-VALUE.
```

データベースに接続されます。

データベースから切断されます。

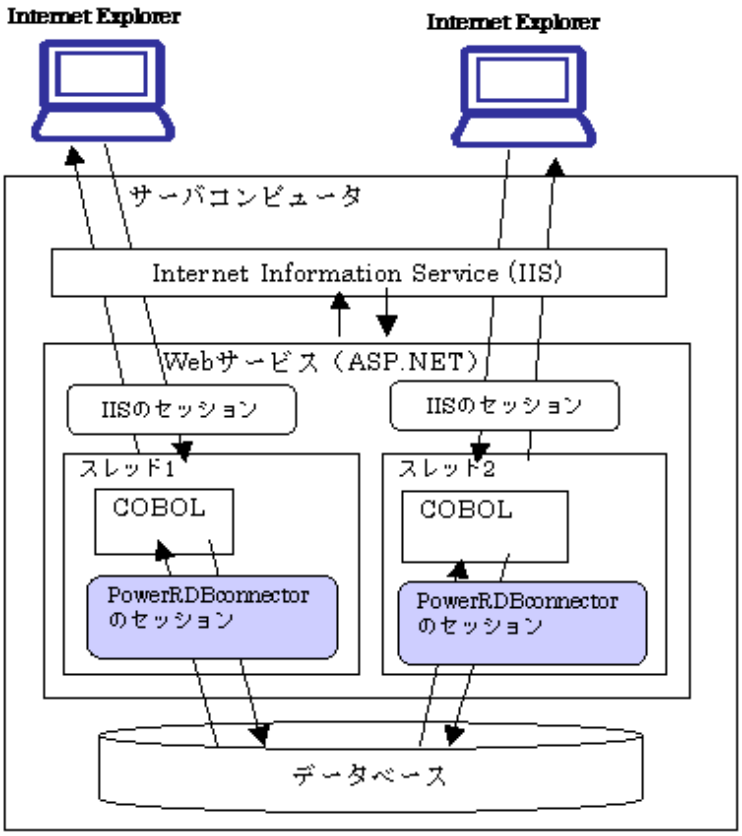
3.2.3 セッションの使用例

セッションが必要な使用例のモデルを以下に示します。

Internet Explorerの画面経由の使用例

Internet Explorerの画面に対応した、ASP.NETのプログラムを呼び出す使用例

図3.5 セッション使用例



この場合、PowerRDBconnectorのセッションと、IISのセッションとが1対1に対応するようにして、データベースにアクセスするCOBOLプログラムを開発します。

ASP.NETの場合、端末とIISのスレッドが固定化されないため、1回のアクセスのたびに、セッションの開設からセッションの閉設までを行ってください。

なお、上記使用例の運用には、PowerRDBconnector サーバパッケージが必要です。

3.3 認証

本節では、PowerRDBconnectorの認証機能について説明します。

PowerRDBconnectorは、SQL Serverが備えている2つの認証機能のどちらの方法でも使用可能です。

3.3.1 Windows認証の使用方法

SQL Serverがインストールされているコンピュータに、COBOLアプリケーションを起動するドメインまたはコンピュータのユーザーを登録することで、Windows認証が使用できます。

Windows認証を使用する場合、COBOLアプリケーションおよびPowerRDBconnectorへの設定は必要ありません。なおWindows認証は、シングルセッションプログラムとマルチセッションプログラムの両方で使用できます。

Windows認証の詳細については、SQL ServerのBooks Onlineを参照してください。

3.3.2 データベース認証の使用方法（シングルセッション）

シングルセッションプログラミングでの、データベース認証の使用方法について説明します。

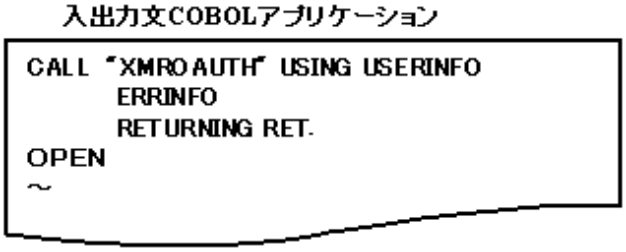
CALL文で認証情報登録サブルーチン（XMROAUTH）を呼び出しておくことで、OPEN文を発行したときにデータベース認証を行えます。

3.3.2.1 データベース認証の動作概要（シングルセッション）

データベース認証は、以下のように動作します。

- 認証情報登録サブルーチンは、指定された認証情報を内部的に記憶します。
- 実際のデータベースへの認証や接続は、プロセス初回または、全ファイルクローズ後の次のOPEN文実行時となります。
- 認証登録サブルーチンで登録したユーザー名やパスワードに誤りがあった場合、OPEN文でエラーとなります。
- データベースに接続するまでに、2回以上呼び出すと、後から指定された認証情報が有効となります。

図3.6 認証情報登録サブルーチンの使用方法



認証情報登録サブルーチンを使用する場合は、以下の点に注意してください。

- 認証情報登録サブルーチンを動的プログラム構造で使用する場合は、COBOL初期化ファイルにエントリ情報の記述が必要です。詳しくは、NetCOBOLのマニュアルを参照してください。

  例）エントリ情報ファイル（ENTRY.ENT）を指定する場合

  COBOL初期化ファイル（COBOL85.CBR)

  ```
  @CBR_ENTRYFILE=ENTRY.ENT
  ```

  エントリ情報ファイル（ENTRY.ENT）

  ・32ビット動作時

  ```
  [ENTRY]
  ; 認証情報登録サブルーチン
  XMROAUTH=F3BWS1SB.DLL
  ```

  ・64ビット動作時

  ```
  [ENTRY]
  ; 認証情報登録サブルーチン
  XMROAUTH=F4ARS1SB_64.DLL
  ```

  認証情報登録サブルーチンは、PowerRDBconnector が提供するシステムライブラリです。COBOLアプリケーションに静的リンクしないでください。詳しくは、「3.1.6 コンパイル方法」および「NetCOBOL 使用手引書」を参照してください。

- 認証情報登録サブルーチンは、初回のOPEN文より前に実行してください。既にOPEN文実行後に、認証情報登録サブルーチンを呼び出してもシーケンスエラーとなりません。
  次のプロセス初回のOPEN文実行時に、登録した認証情報を使用してデータベースに接続されます。
- 指定された認証情報は、そのままデータベースに渡されます。このため、認証情報の制約（文字数、使用可能文字など）はデータベースの仕様に準じます。
- 認証種別にデータベース認証を指定し、認証情報登録サブルーチンを実行した場合、以下の処理を行うまで、データベース認証が適用されます。

  ー プロセスの終了。

  ー 認証種別にWindows認証を指定した認証情報登録サブルーチンの再呼出し。
- データベース認証は、データベースのユーザー名とパスワードをプログラムで扱うため、セキュリティに注意してください。詳しくは、「4.2.5.2 セキュリティについて」を参照してください。

3.3.2.2 認証情報登録サブルーチンのインターフェース（シングルセッション）

認証情報登録サブルーチンのインターフェースは、以下のとおりです。

認証情報の登録

- 機能

  データベースに対する認証情報を登録します。

- 呼出し形式

```
CALL “XMROAUTH” USING 認証用登録情報 エラー情報 RETURNING 復帰値.
```

- パラメーターのデータ定義

```
01 認証用登録情報.
  02 認証種別          PIC 9(9) COMP-5.
  02 ユーザー名長      PIC 9(9) COMP-5.
  02 ユーザー名        PIC X(260).
  02 パスワード長      PIC 9(9) COMP-5.
  02 パスワード        PIC X(260).
01 エラー情報.
  02 終了情報          PIC S9(9) COMP-5.
  02 詳細情報          PIC S9(9) COMP-5.
  02 FILLER            PIC S9(9) VALUE 0.
01 復帰値              PIC S9(9) COMP-5 VALUE 0.
```

  ※上記は、必ずレベル番号01で記載してください。

  ※FILLERは、0を設定してください。

- パラメーターの意味

  ー 認証種別（必須）

  認証の種別を指定します。

  1：Windows認証

  2：データベース認証

－ ユーザー名長

データベース認証のユーザー名の長さを10進数のバイト長で指定します。1から256まで指定できます。データベース認証の場合、必要です。

－ ユーザー名

データベース認証のユーザー名を指定します。最大256バイトまで指定できます。データベース認証の場合、必要です。

－ パスワード長

データベース認証のパスワードの長さを10進数のバイト長で指定します。1から256まで指定できます。データベース認証の場合、必要です。

－ パスワード

データベース認証のパスワードを指定します。最大256バイトまで指定できます。データベース認証の場合、必要です。

－ 終了情報/詳細情報

終了情報および詳細情報については、「表5.8 認証情報登録サブルーチンのエラーコード」を参照してください。

・ 復帰値

復帰値は、以下のとおりです。

表3.24 XMROAUTHサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常(注)	なし
-1	エラー	なし

（注）

ユーザー名やパスワードの誤りがある場合、本サブルーチンは正常終了して、OPEN文でエラーが検出されます。

エラーが発生した場合には、エラー情報にエラーコードが通知され、認証情報の登録は行われません。
また、Windowsのイベントログ（アプリケーションログ）へエラー情報は出力されません。
エラー情報は、「5.1.3 認証情報登録サブルーチンのエラー情報」を参照してください。

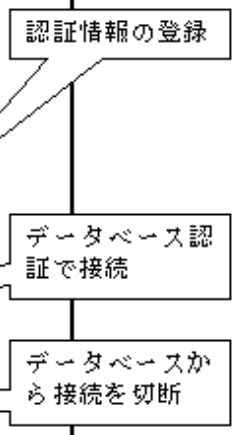
「図3.7 認証情報登録サブルーチンの使用例」に使用例を示します。

図3.7 認証情報登録サブルーチンの使用例

COBOLソース

```
DATA   DIVISION.
WORKING-STORAGE   SECTION.
01  USERINFO.
 02  AUTH        PIC 9(9) COMP-5 VALUE 0.
 02  USERNNO     PIC 9(9) COMP-5 VALUE 0.
 02  USERN       PIC X(260) VALUE SPACE.
 02  PASSWNO     PIC 9(9) COMP-5 VALUE 0.
 02  PASSW       PIC X(260) VALUE SPACE.
01  ERRINFO.
 02  ENDINFO     PIC S9(9) COMP-5 VALUE 0.
 02  DITINFO     PIC S9(9) COMP-5 VALUE 0.
 02  FILLER      PIC S9(9) VALUE 0.
01  RET          PIC S9(9) COMP-5 VALUE 0.
＜略＞
PROCEDURE DIVISION.
＊ データベース認証にする
MOVE  2  TO  AUTH.
MOVE  "SYSTEM100"  TO  USERN.
MOVE  9  TO  USERNNO.
MOVE  "SYSUSER"  TO  PASSW.
MOVE  7  TO  PASSWNO.
CALL  "XMROAUTH" USING USERINFO ERRINFO
          RETURNING RET.
MOVE SPACE TO USERN.
MOVE SPACE TO PASSW.

＜略＞
OPEN  I-O INFILE.
...
CLOSE INFILE.
＜略＞
```



3.3.3 データベース認証の使用方法（マルチセッション）

マルチセッションプログラミングでの、データベース認証の使用方法について説明します。

CALL文で認証情報登録サブルーチン（COB_PRDB_AUTH）を呼び出しておくことで、OPEN文を発行したときにデータベース認証を行えます。

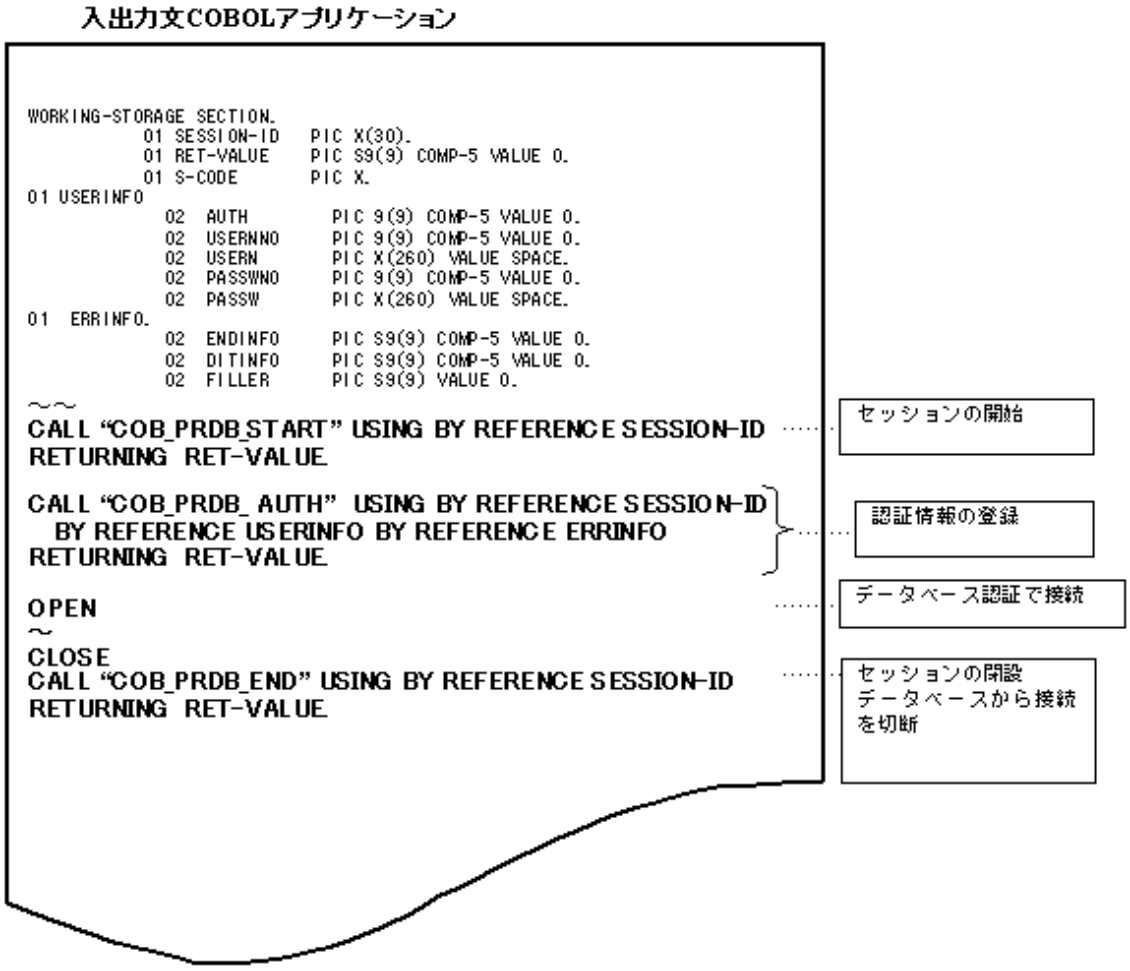
3.3.3.1 データベース認証の動作概要（マルチセッション）

データベース認証は、以下のように動作します。

- 認証情報登録サブルーチンは、指定された認証情報を内部的に記憶します。
- 実際のデータベースへの認証や接続は、セッション開始後、初回OPEN文実行時となります。認証に失敗すると、OPEN文でエラーとなります。

・ データベースに接続するまでに、2回以上呼び出すと、後から指定された認証情報が有効となります。

図3.8 認証情報登録サブルーチンの使用方法



認証情報登録サブルーチンを使用する場合は、以下の点に注意してください。

・ 認証情報登録サブルーチンを動的プログラム構造で使用する場合は、COBOL初期化ファイルにエントリ情報の記述が必要です。詳しくは、NetCOBOLのマニュアルを参照してください。

例）エントリ情報ファイル（ENTRY.ENT）を指定する場合

COBOL初期化ファイル（COBOL85.CBR)

```
@CBR_ENTRYFILE=ENTRY.ENT
```

エントリ情報ファイル（ENTRY.ENT）

・32ビット動作時

```
[ENTRY]
; 認証情報登録サブルーチン
COB_PRDB_AUTH=F3BIEFNC.dll
```

・64ビット動作時

```
[ENTRY]
;認証情報登録サブルーチン
COB_PRDB_AUTH=F4AGEFNC.dll
```

認証情報登録サブルーチンは、PowerRDBconnector が提供するシステムライブラリです。COBOLアプリケーションに静的リンクしないでください。詳しくは、「3.1.6 コンパイル方法」および「NetCOBOL 使用手引書」を参照してください。

- 認証情報登録サブルーチンを使用する場合には、まずセッションの開設を行ってください。
- 認証情報登録サブルーチンは、初回のOPEN文より前に実行してください。OPEN文実行後に、認証情報登録サブルーチンを呼び出してもシーケンスエラーとなりません。
- 指定された認証情報は、そのままデータベースに渡されます。このため、認証情報の制約（文字数、使用可能文字など）はデータベースの仕様に準じます。
- 認証種別にデータベース認証を指定し、認証情報登録サブルーチンを実行した場合、以下の処理を行うまで、データベース認証が適用されます。

  ー セッションの閉設。

  ー 認証種別にWindows認証を指定した認証情報登録サブルーチンの再呼出し。
- データベース認証は、データベースのユーザー名とパスワードをプログラムで扱うため、セキュリティに注意してください。詳しくは、「4.2.5.2 セキュリティについて」を参照してください。

3.3.3.2 認証情報登録サブルーチンのインターフェース（マルチセッション）

認証情報登録サブルーチンのインターフェースは、以下のとおりです。

認証情報の登録

- 機能

  データベースに対する認証情報を登録します。

- 呼出し形式

```
CALL "COB_PRDB_AUTH" USING BY REFERENCE セッションID BY REFERENCE 認証用登録情報
     BY REFERENCE エラー情報 RETURNING 復帰値.
```

- パラメーターのデータ定義

```
01 セッションID          PIC X(30).
01 認証用登録情報.
   02 認証種別           PIC 9(9) COMP-5.
   02 ユーザー名長        PIC 9(9) COMP-5.
   02 ユーザー名          PIC X(260).
   02 パスワード長        PIC 9(9) COMP-5.
   02 パスワード          PIC X(260).
01 エラー情報.
   02 終了情報           PIC S9(9) COMP-5.
   02 詳細情報           PIC S9(9) COMP-5.
   02 FILLER             PIC S9(9) VALUE 0.
01 復帰値                PIC S9(9) COMP-5 VALUE 0.
```

※上記は、必ずレベル番号01で記載してください。

※FILLERは、0を設定してください。

・ パラメーターの意味

― セッションID（必須）

英数字項目で30バイト指定します。

文字の種類の制約はありません。

セッション開設サブルーチンで指定したセッションIDを指定してください。

― 認証種別（必須）

認証の種別を指定します。

1：Windows認証

2：データベース認証

― ユーザー名長

データベース認証のユーザー名の長さを10進数のバイト長で指定します。1から256まで指定できます。データベース認証の場合、必要です。

― ユーザー名

データベース認証のユーザー名を指定します。最大256バイトまで指定できます。データベース認証の場合、必要です。

― パスワード長

データベース認証のパスワードの長さを10進数のバイト長で指定します。1から256まで指定できます。データベース認証の場合、必要です。

― パスワード

データベース認証のパスワードを指定します。最大256バイトまで指定できます。データベース認証の場合、必要です。

― 終了情報/詳細情報

終了情報および詳細情報については、「表5.8 認証情報登録サブルーチンのエラーコード」を参照してください。

・ 復帰値

復帰値は、以下のとおりです。

表3.25 COB_PRDB_AUTHサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常(注1)	なし
-1	エラー	あり
-201	セッションIDの指定が正しくありません。(注2)	なし

（注1）

ユーザー名やパスワードの誤りがある場合、本サブルーチンは正常終了し、OPEN文でエラーが検出されます。

（注2）

セッションIDの指定に誤りがあった場合には、認証情報の登録は行われません。
この際、エラー情報にエラーコードは通知されず、Windowsのイベントログ（アプリケーションログ）へエラー情報は出力されません。

エラーが発生した場合には、エラー情報にエラーコードが通知され、認証情報の登録は行われません。
エラー情報は、「5.1.3 認証情報登録サブルーチンのエラー情報」を参照してください。

3.4 トランザクション

本節では、トランザクション機能の使い方について説明します。

3.4.1 トランザクション機能とは

トランザクション機能とは、アプリケーションのデータのやりとりを1単位（トランザクション単位）に分割し、その単位で処理を保証し、複数ファイルおよび複数レコード間の一貫性を保証する機能です。

トランザクション機能により、アプリケーションプログラムの実行中に何らかのエラーが発生したときに、トランザクション単位に処理を取り消すことができます。

アプリケーションプログラムが中断した場合には、実行中であったトランザクションを取り消し、アプリケーションプログラムがファイルに与える影響を局所化し、アプリケーションプログラムの再実行を容易に行うことができます。

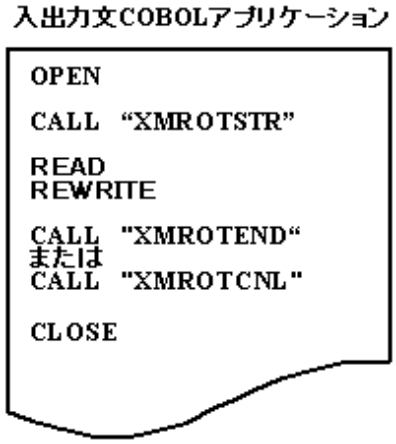
3.4.2 トランザクションの使用方法（シングルセッション）

3.4.2.1 トランザクションプログラム（シングルセッション）

PowerRDBconnectorは、“トランザクションの開始”、“トランザクションの確定”、および“トランザクションの取消し”の3種類のサブルーチンを、COBOLアプリケーションから呼び出すことで、トランザクション機能を実現します。

以下に、トランザクションサブルーチンの使用方法を示します。

図3.9 トランザクションサブルーチンの使用方法（シングルセッション）



トランザクションサブルーチンを以下に示します。

表3.26 トランザクションサブルーチンの一覧

トランザクション	サブルーチン名
開始	XMROTSTR
確定	XMROTEND
取消し	XMROTCNL
テーブルロック解除時の取消し※	XMROTRBK

※:トランザクションを使用せず、テーブルロック区間内で更新されたレコードを、テーブルロック解除時に、更新前に戻す場合に使用します。詳細は、「3.5.3.2 テーブルロック解除時のトランザクション取消し機能」を参照してください。

トランザクションは、以下のように動作します。

- トランザクションを開始していない場合は、データベースのオートコミット機能で動作します。
- トランザクション分離レベルは、ReadCommittedが使用されます。
- トランザクションは、セッション単位に開始、確定、取消しが行われます。

3.4.2.2 トランザクション使用時の注意事項（シングルセッション）

トランザクションの使用方法には、以下の注意事項があります。

- トランザクションサブルーチンを動的プログラム構造で使用する場合は、COBOL初期化ファイルにエントリ情報の記述が必要です。詳しくは、NetCOBOLのマニュアルを参照してください。

  例）エントリ情報ファイル（ENTRY.ENT）を指定する場合

  COBOL初期化ファイル（COBOL85.CBR)

  ```
  @CBR_ENTRYFILE=ENTRY.ENT
  ```

  エントリ情報ファイル（ENTRY.ENT）

  ― 32ビット動作時

  ```
  [ENTRY]
  ;トランザクションサブルーチン
  XMROTSTR=F3BWS1CB.DLL
  XMROTEND=F3BWS1CB.DLL
  XMROTCNL=F3BWS1CB.DLL
  XMROTRBK=F3BWS1CB.DLL
  ```

  ― 64ビット動作時

  ```
  [ENTRY]
  ;トランザクションサブルーチン
  XMROTSTR=F4ARS1CB_64.DLL
  XMROTEND=F4ARS1CB_64.DLL
  XMROTCNL=F4ARS1CB_64.DLL
  XMROTRBK=F4ARS1CB_64.DLL
  ```

  トランザクションサブルーチンは、PowerRDBconnector が提供するシステムライブラリです。COBOLアプリケーションに静的リンクしないでください。詳しくは、「3.1.6 コンパイル方法」および「NetCOBOL 使用手引書」を参照してください。

- トランザクション開始のタイミングについて、以下に示します。

  ― OPEN文や、他の入出力文(READ、WRITE、REWRITE、DELETE、START文)を発行前後によらず、呼び出すことができます。

  ― トランザクションが開始された状態で、再度トランザクション開始を行うとエラー終了します。

  ― トランザクション開始後に、OUTPUTモードのOPEN文を実行し、トランザクション取消しを行っても、OUTPUTモードのOPENによる初期化は、取消しできません。

- トランザクションの確定および取消しは、最終のCLOSE文を発行するより前に、呼び出してください。

  最終のCLOSE文とは、そのCLOSE文を実行すると、プロセス内でPowerRDBconnectorを使ってオープンしているファイルが1つもなくなるCLOSE文のことです。

・トランザクションの確定および取消し後に、トランザクションを開始する場合は、トランザクション開始用のサブルーチンを再度呼び出してください。

・テーブルロックを指定したファイルのオープン中は、トランザクション操作（開始、確定、取消し）が無効となります。テーブルロックについての詳細は、「3.5.3 テーブルロック機能」、「4.1.3.2 テーブルロック機能について」、「4.2.2.3 テーブルロックの排他制御について」を参照してください。

・トランザクションを開始し、トランザクションの確定および取消しを実行しなかった場合は、トランザクションが取り消されます。具体例を以下に示します。

ー トランザクションの確定および取消しを実行せずに最終のCLOSE文を実行した場合

ー タスクマネージャでCOBOLアプリケーションのプロセスを停止した場合

ー COBOLアプリケーションが、アプリケーションエラーで中断した場合

ー COBOLアプリケーション動作中に、データベースサービスを中断またはネットワークを中断した場合

3.4.2.3 トランザクションサブルーチンのインターフェース（シングルセッション）

トランザクションサブルーチンのインターフェースは、以下のとおりです。

(1)トランザクション開始

・機能

トランザクションの開始を宣言します。

トランザクション機能を使用するにあたって必ず発行してください。

・呼出し形式

CALL “XMROTSTR”

(2)トランザクション確定

・機能

現在のトランザクションを終了し、トランザクション中に更新された内容をすべて確定します。

・呼出し形式

CALL “XMROTEND”

(3)トランザクション取消し

・機能

現在のトランザクションを終了し、トランザクション中に更新された内容をすべて取り消します。

・呼出し形式

CALL “XMROTCNL”

(4)テーブルロック解除時のトランザクション取消し

・ 機能

テーブルロック解除時に、テーブルロック区間中に更新された内容をすべて取り消します。

・ 呼出し形式

CALL “XMROTRBK”

・ 注意

本サブルーチンの注意事項を以下に示します。

－ テーブルロックが有効な区間内で呼び出した場合のみ有効です。この範囲外で呼び出した場合には、無効となります。

－ テーブルロック対象のファイルすべてをクローズした後は、本サブルーチンの指定は無効となります。

詳細は、「3.5.3.2 テーブルロック解除時のトランザクション取消し機能」を参照してください。

トランザクションサブルーチン使用する場合の注意については、「4.2.1 トランザクションに関する注意事項」を参照してください。

3.4.3 トランザクションの使用方法（マルチセッション）

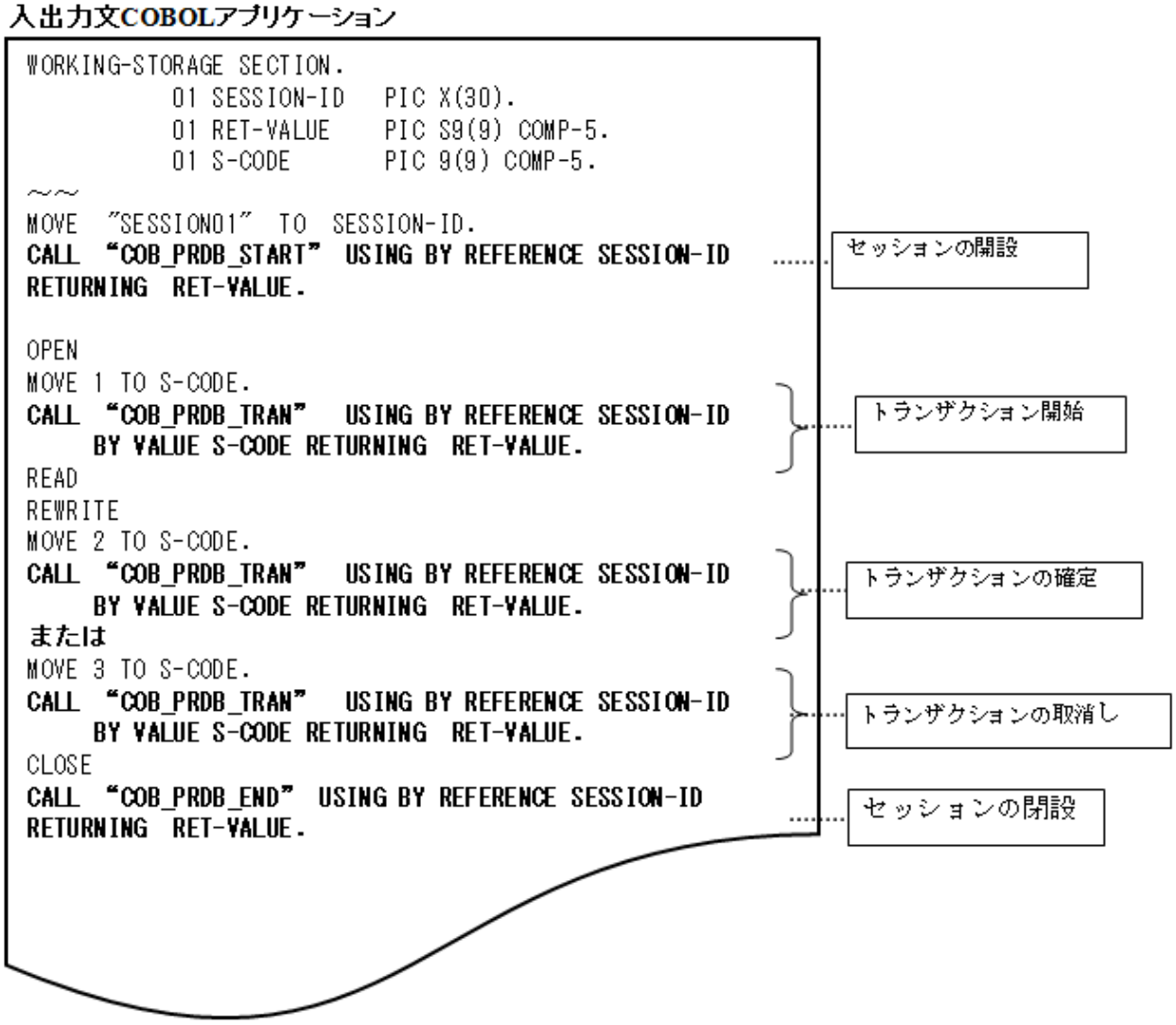
トランザクションサブルーチンのインターフェースは、以下のとおりです。

3.4.3.1 トランザクションプログラム（マルチセッション）

PowerRDBconnectorは、“トランザクションの開始”、“トランザクションの確定”、および“トランザクションの取消し”の3種類のサブルーチンを、COBOLアプリケーションから呼び出すことで、トランザクション機能を実現します。

以下に、トランザクションサブルーチンの使用方法を示します。

図3.10 トランザクションサブルーチンの使用方法（マルチセッション）



トランザクションサブルーチンを以下に示します。

表3.27 トランザクションサブルーチンの一覧

トランザクション	サブルーチン名	パラメーター値
開始	COB_PRDB_TRAN	1
確定		2
取消し		3
テーブルロック解除時の取消し※		4

※:トランザクションを使用せず、テーブルロック区間内で更新されたレコードを、テーブルロック解除時に、更新前に戻す場合に使用します。詳細は、「3.5.3.2 テーブルロック解除時のトランザクション取消し機能」を参照してください。

トランザクションは、以下のように動作します。

・トランザクションを開始していない場合は、データベースのオートコミット機能で動作します。

・トランザクション分離レベルは、ReadCommittedが使用されます。

・トランザクションは、セッション単位に開始、確定、取消しが行われます。

3.4.3.2 トランザクション使用時の注意事項（マルチセッション）

トランザクションの使用には、以下の注意事項があります。

・トランザクションサブルーチンを動的プログラム構造で使用する場合は、COBOL初期化ファイルにエントリ情報の記述が必要です。詳しくは、NetCOBOLのマニュアルを参照してください。

例）エントリ情報ファイル（ENTRY.ENT）を指定する場合

COBOL初期化ファイル（COBOL85.CBR)

```
@CBR_ENTRYFILE=ENTRY.ENT
```

エントリ情報ファイル（ENTRY.ENT）

― 32ビット動作時

```
[ENTRY]
;トランザクションサブルーチン
COB_PRDB_TRAN=F3BIEFNC.dll
```

― 64ビット動作時

```
[ENTRY]
;トランザクションサブルーチン
COB_PRDB_TRAN=F4AGEFNC.dll
```

・トランザクションサブルーチンは、PowerRDBconnector が提供するシステムライブラリです。COBOLアプリケーションに静的リンクしないでください。詳しくは、「3.1.6 コンパイル方法」および「NetCOBOL 使用手引書」を参照してください。

・トランザクションサブルーチンを使用する場合は、必ずセッションの開設を行ってください。

・トランザクション開始のタイミングについて、以下に示します。

― OPEN文や、他の入出力文(READ、WRITE、REWRITE、DELETE、START文)を発行前後によらず、呼び出すことができます。

― トランザクションが開始された状態で、再度トランザクション開始を行うとエラー終了します。

― トランザクション開始後に、OUTPUTモードのOPEN文を実行し、トランザクション取消しを行っても、OUTPUTモードのOPENによる初期化は、取消しできません。

・トランザクションの確定および取消しは、最終のCLOSE文を発行するより前に、呼び出してください。

最終のCLOSE文とは、そのCLOSE文を実行すると、セッション内でPowerRDBconnectorを使ってオープンしているファイルが1つもなくなるCLOSE文のことです。

・トランザクションの確定および取消し後に、トランザクションを開始する場合は、トランザクション開始を再度呼び出してください。

・テーブルロックを指定したファイルのオープン中は、トランザクション操作（開始、確定、取消し）が無効となります。

・トランザクションを開始し、トランザクションの確定および取消しを実行しなかった場合は、トランザクションが取り消されます。具体例を以下に示します。

　－ トランザクションの確定および取消しを実行せずに最終のCLOSE文を実行した場合

　－ タスクマネージャでCOBOLアプリケーションのプロセスを停止した場合

　－ COBOLアプリケーションが、アプリケーションエラーで中断した場合

　－ COBOLアプリケーション動作中に、データベースサービスを中断またはネットワークを中断した場合

・マルチセッションプログラムの注意事項は、「4.1.3.8 マルチセッションプログラミングについて」や、「4.2.4 マルチスレッド使用時の注意事項」を参照してください。

3.4.3.3 トランザクションサブルーチンのインターフェース（マルチセッション）

マルチセッションプログラミングを用いる場合のトランザクションサブルーチンのインターフェースは、以下のとおりです。

トランザクションの制御

・機能

　マルチセッションプログラミングの場合のトランザクション制御を行います。

・呼出し形式

```
CALL “COB_PRDB_TRAN” USING BY REFERENCE セッションID BY VALUE トランザクション種別 RETURNING 復帰値.
```

・パラメーターのデータ定義

```
01 セッションID          PIC X(30).
01 トランザクション種別  PIC 9(9) COMP-5.
01 復帰値                PIC S9(9) COMP-5.
```

　※上記はすべて、必ずレベル番号01で記載してください。

・パラメーターの意味

　－ セッションID（必須）

　　英数字項目で30バイト指定します。

　　文字の種類の制約はありません。

　　セッション開設サブルーチンで指定したセッションIDを指定してください。

　－ トランザクション種別（必須）

　　処理するトランザクション種別を指定します。

　　1:トランザクション開始

　　2:トランザクション確定

3:トランザクション取消し

4:テーブルロック解除時のトランザクション取消し

なお、本パラメーターは必ず指定してください。

- 復帰値

  復帰値は、以下のとおりです。

表3.28 COB_PRDB_TRANサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常	なし
-1	エラー	あり
-2	内部矛盾	あり
-3	トランザクション種別の指定に誤りがあります。	あり
-201	セッションIDの指定が正しくありません。	なし
-202	無効なトランザクション種別が指定されました。	なし

トランザクションサブルーチンでシーケンスエラーやデータベースのエラーが発生した場合は、イベントログ(アプリケーションログ)へエラー情報を出力後、COBOLアプリケーションは終了しません。エラー情報は、「5.1.2 トランザクションアクセス時のエラー情報」を参照してください。

- 注意

  トランザクションサブルーチンを使用する場合の注意については、「4.2.1 トランザクションに関する注意事項」を参照してください。

3.5 排他制御

本節では、PowerRDBconnectorの排他制御について説明します。

3.5.1 排他制御の種類

排他制御には、以下の2つの機能があります。

- レコードロック機能

  レコードロック機能とは、ある業務のSTART文で位置付けたレコードやREAD文で読み込んだレコードに対して、排他制御を行う機能です。

  複数の端末から同じファイルにアクセスするオンラインアプリケーションに適しています。詳しくは、「3.5.2 レコードロック機能」を参照してください。

- テーブルロック機能

  テーブルロック機能とは、ある業務で使用しているファイルのオープンからクローズまでの間に、他の業務からの該当ファイルへのオープンに対して、排他制御を行う機能です。
  COBOL初期化ファイルに、TableLockプロパティを指定することで使用できます。

  1つのプログラムが実行中ファイルを占有するバッチ型業務に適しています。

レコードロックを獲得しているテーブルに対して、他の業務からテーブルロックでOPEN文を実行すると、オープンは待ち合わせずに排他エラーで即時に復帰します。

テーブルロック機能を使用する場合、テーブルロック対象のファイルに対するオープンからクローズまでの間を、強制的に1つのトランザクション区間とすることで、テーブルロックを実現します。このため、オープンからクローズまでの区間でアプリケーションが終了した場合、この区間のすべての更新が取り消されることとなります。詳しくは、「3.5.3 テーブルロック機能」を参照してください。

注意

テーブルロック機能およびレコードロック機能の使用有無にかかわらず、以下の注意事項があります。

- OUTPUTモードでオープン直後から最初の入出力文までは、テーブルロックを使用した状態になります。
- OUTPUTモードでオープンしたファイルに対して、同一プロセス内で2重にオープンすることはできません。
  OUTPUTモードでオープンしたファイルに関するその他の注意については、「4.2.3.5 OUTPUTモードのオープンの使用について」を参照してください。
- 処理順序によっては、デッドロックが検出されることがあります。
- 排他制御は、セッション単位に行われます。

排他制御の注意については、「4.2.2 排他制御に関する注意事項」を参照してください。

3.5.2 レコードロック機能

I-Oモードでオープンした後、START文で位置付けたレコードや、READ文で読込んだレコードに対して、レコードロックが獲得され、他の業務からの位置付けや読込み操作を待ち合わせる機能です。

なお、INPUTモードでオープンした場合、レコードロックの待ち合わせは発生しません。

トランザクションを適用した場合、アクセスしたすべてのレコードに対して、レコードロックが獲得され、他の業務からの位置付け、読込み、更新や削除の操作を待ち合わせることができます。

アクセスして獲得されたレコードロックは、トランザクションの確定または取消しで解放されます。

レコードロックの注意事項については、「4.1.3.3 レコードロック機能について」を参照してください。

3.5.2.1 トランザクション未適用時のレコードロック

(1)レコードロック獲得

I-Oモードでオープンし、以下の操作を行ったレコードに対してレコードロックを獲得します。

- START文で位置付けたレコード(※)
- READ文で読み込んだレコード

※)

DYNAMICモードでオープンし、START FIRST文を実行した場合は位置付けされた先頭のレコードに対するレコードロックは獲得されません。

(2)レコードロック解放

獲得されたレコードロックは、以下の操作で解放されます。

表3.29 トランザクション未適用時のレコードロック解放タイミング

アクセスモード	レコードロックの解放タイミング
SEQUENTIAL	・ 次のレコードに位置付け（START）または読み込んだ（READ）場合（注） ・ EOFを検出した場合 ・ CLOSE文を実行した場合
RANDOM	・ 読込み（READ）、更新（REWRITE）、削除（DELETE）を行った場合 ・ CLOSE文を実行した場合
DYNAMIC	・ 次のレコードに位置付け（START）または読込んだ（READ）場合（注） ・ EOFを検出した場合 ・ CLOSE文を実行した場合

（注）更新(REWRITE)、削除(DELETE)を行ってもレコードロックは解放されません

PowerRDBconnectorでOracleにアクセス時や、以下の他製品にアクセス時、直前に読込み時に獲得したレコードロックは、更新（REWRITE）、削除（DELETE）を行うと、レコードロックは解放されていました。

- ASP(RDB/6000、Symfoware6000)
- SymfoWARE7000やPowerRW+

しかし、PowerRDBconnectorでSQL Serverにアクセス時は、解放されないため注意してください。

新たにレコードロックを獲得せずに、読込み時に獲得したレコードロックを解放したい場合、以下のようにしてください。

- アクセスしているテーブルに存在していないキー値を指定して、レコードに位置付け（START）を行って、“レコードなし”のエラーを発生させる。
- EOFを検出するように、読込み（READ）を行う。

ただし、上記は、どちらも次に読込む位置（カーソル位置）が不定となりますので、注意してください。

3.5.2.2 トランザクション適用時のレコードロック

(1)レコードロック獲得

I-Oモード、OUTPUTモード、またはEXTENDモードでオープンし、以下の操作を行ったすべてのレコードに対してレコードロックを獲得します。

- START文で位置付けたレコード
- READ文で読み込んだレコード
- トランザクション中にREWRITE文で更新したレコード
- トランザクション中にDELETE文で削除したレコード
- トランザクション中にWRITE文で追加したレコード

(2)レコードロック解放

獲得されたレコードロックは、以下の操作で解放されます。

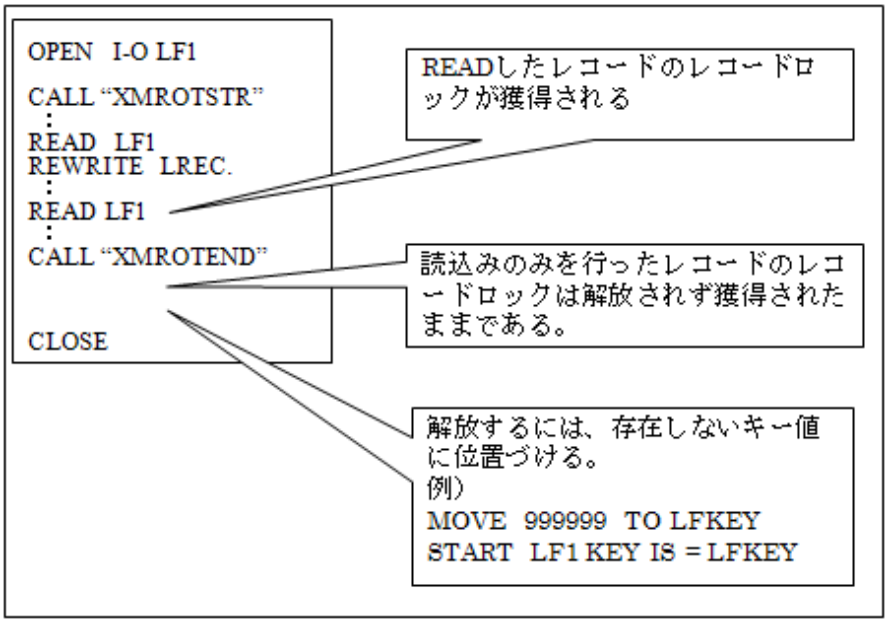
- トランザクションの確定または取消し

注意

トランザクション内で更新、削除を行わず、読み込みだけを行ったレコードについて

トランザクションを確定または取消しする直前に、位置付けまたは読み込んだレコードのレコードロックは解放されません。

図3.11 トランザクションを確定してもレコードロックが解放されない



3.5.2.3 レコードロックの獲得待合わせ

レコードロックを獲得しているレコードに対して、別のCOBOLアプリケーションから以下の操作を行うとレコードロック獲得の待合わせが発生します。

- レコードロックの獲得
- 乱呼出しのREWRITE文およびDELETE文

レコードロック獲得の待合わせは、PowerRDBconnector動作環境ファイルのTimeOutプロパティで指定できます。レコードロック獲得の待合わせについては、「3.6 タイムアウト機能」を参照してください。

DELETE文で削除したレコードにレコードロックが獲得され、WRITE文でレコードロック獲得の待合わせが発生する場合があります。詳しくは、「4.2.2.2 削除したレコードの待合せについて」を参照してください。

3.5.3 テーブルロック機能

テーブルロックは、オープン中に、他の業務からのオープンを防ぐ機能です。

3.5.3.1 テーブルロック機能の使用方法

COBOL初期化ファイルのTableLockプロパティにONを指定することにより、OPEN文でテーブルロックを行います。テーブルロック対象のすべてのテーブルをクローズすると、テーブルロックを解除します。

テーブルロックが有効なテーブルのOPEN文で強制的にトランザクションを開始してCLOSE文でトランザクションを確定することでテーブルロックを実現します。テーブルロックを指定したファイルのオープン中は、トランザクション操作（開始、確定、取消し）が無効となります。

強制的に開始するトランザクションの注意については、「4.2.2.3 テーブルロックの排他制御について」を参照してください。

INPUTモードでオープンすると、Share（共有モード）でテーブルロックを行います。INPUTモードでオープン後、同一テーブルに対して別のCOBOLアプリケーションからのオープンを防ぐことはできません。

INPUT以外のモードでオープンすると、Exclusive(占有モード)でテーブルロックを行います。INPUT以外のモードでオープンした後、同一テーブルに対して別のCOBOLアプリケーションからのオープンはできません。

[テーブルロック機能の使用方法]

COBOL初期化ファイルのTableLockプロパティにONを指定してください。

例）テーブルロックのファイル **EMPLOYEE** に対する記述例

```
; COBOL初期化ファイル
EMPLOYEE=TableName=employee&SchemaName=dbo
&AccessMode=RANDOM&TableLock=ON,RDM
```

テーブルロックの指定時は、ファイル単位に指定できるため、ファイルのロック動作は以下のようになり、テーブルロックとレコードロックが混在できます。

表3.30 テーブルロックの指定

TableLockの指定	ファイルのロック動作
指定あり	テーブルロック
指定なし	レコードロック

また、テーブルロックの注意事項については、「4.1.3.2 テーブルロック機能について」、「4.2.2.3 テーブルロックの排他制御について」、「4.2.3.5 OUTPUTモードのオープンの使用について」を参照してください。

[テーブルロック解除時のデータの扱いについて]

テーブルロックは、PowerRDBconnector内部で強制的にトランザクションを制御して実現しているため、テーブルロック解除時には、データは次のように扱われます。

表3.31 テーブルロック解除時のデータの扱い

テーブルロック解除のタイミング	テーブルロック区間内の更新データの扱い
テーブルロック対象のすべてのテーブルをクローズしたとき。	更新データが反映されます。
テーブルロック対象のすべてのテーブルをクローズせずにCOBOLアプリケーションを終了させたとき。	更新データが反映されます。

テーブルロック解除のタイミング	テーブルロック区間内の更新データの扱い
タスクマネージャでCOBOLアプリケーションのプロセスを停止した場合	更新データは反映されません。
COBOLアプリケーションが、アプリケーションエラーで中断した場合	更新データは反映されません。
COBOLアプリケーション動作中に、データベースサービスを中断またはネットワークを中断した場合	更新データは反映されません。

3.5.3.2 テーブルロック解除時のトランザクション取消し機能

テーブルロック機能は、テーブルロックが有効なテーブルのOPEN文で強制的にトランザクションを開始してCLOSE文でトランザクションを確定することで実現しています。

アプリケーションを強制的に中断した場合には、テーブルロック区間内で更新された内容が取り消されます。このような、強制的な中断と同じように処理したい場合（エラーを検出した場合など）は、テーブルロック解除時に、強制的にトランザクションを取り消し、更新されたすべての内容を取り消すことができます。

テーブルロックが有効な区間で、“XMROTRBK”または、”COB_PRDB_TRAN” (トランザクション種別：4)サブルーチンを呼び出すことで、テーブルロック解除時（CLOSE文または、“COB_PRDB_END”サブルーチン実行時）、強制的にトランザクションを取り消すことができます。

サブルーチン名

- シングルセッションプログラミングの場合

  CALL “XMROTRBK”

- マルチセッションプログラミングの場合

  CALL “COB_PRDB_TRAN” (トランザクション種別：4)

  サブルーチンの使用方法は、「3.4.2 トランザクションの使用方法（シングルセッション）」および「3.4.3 トランザクションの使用方法（マルチセッション）」を参照してください。

- 本サブルーチンは、テーブルロックが有効な区間内で呼び出した場合のみ有効です。この範囲外で呼び出した場合には、無効となります。
- シングルセッションプログラミングの場合とマルチセッションプログラミングの場合で更新を取り消すトランザクションの範囲が以下のように異なります。
  - ー シングルセッションプログラミングの場合（XMROTRBK）

    テーブルロック対象のファイルをすべてクローズするまでのすべての更新（他テーブルの更新も含む）を取り消します。
  - ー マルチセッションプログラミングの場合（COB_PRDB_TRAN/トランザクション種別：4）

    セッションを閉設するサブルーチン（COB_PRDB_END）を呼び出すまでのすべての更新（他テーブルの更新も含む）を取り消します。

テーブルロック対象のファイルをすべてクローズしても更新は取り消されません。セッションを閉設するサブルーチン（COB_PRDB_END）が呼び出された時点で更新が取り消されます。

3.6 タイムアウト機能

本節では、PowerRDBconnectorのタイムアウト機能について説明します。

入出力文によるデータベースへのアクセスで、処理完了の待合わせ時間を指定できます。指定した時間を超えて処理が完了しないと、タイムアウトを検出してエラー（FILE STATUS:90、iserrno:255）を通知します。

タイムアウト機能には、2つの用途があります。

- レコードロック獲得の待合わせ
- 高負荷な実行環境でのSQL Serverの処理完了の待合わせ

3.6.1 レコードロック獲得の待合わせ

COBOLアプリケーションが多重動作する環境では、レコードロックの獲得待ちになると、レコードロックが解放されるのを待合わせます。

このような場合に、レコードロックを獲得するまでの時間を指定できます。

レコードロック獲得の待合わせでタイムアウトが発生した場合、このときトランザクションを開始した状態であっても自動的にトランザクションが確定または取消しすることはありません。COBOLアプリケーションは、トランザクションの確定または取消しを行ってください。

なお、SQL Serverは、PowerRDBconnector動作環境ファイルのTimeOutプロパティの設定値よりも前にタイムアウトのエラーを通知する場合があります。この場合、SQL Serverの動作環境を見直してください。

3.6.2 処理完了待合わせ時間の設定

処理完了待合わせ時間の設定方法については、「3.1.4 PowerRDBconnector 動作環境ファイル」を参照してください。

例）レコードロック獲得待合わせ時間に60秒を指定するための記述例

PowerRDBconnector動作環境ファイル(DBIO_ENV)

```
; PowerRDBconnector動作環境ファイル
ServerName=snake
DataSourceName=pubs
TimeOut=60
```

3.6.3 高負荷な実行環境でのSQL Serverの処理完了の待合わせ

データベースへのアクセスで大量のデータを処理する場合や複数のアプリケーションが同時に動作する環境では、SQL Serverの負荷が高くなり、一時的に処理の完了を待合わせる場合があります。

例）OPEN文でタイムアウトが発生する例

インデックスが定義されておらず、かつレコード件数が大量にある表に対して、索引ファイルとしてOPEN文を実行した場合

レコードロック獲得の待合わせ以外でタイムアウトが発生する場合、SQL Serverの動作環境を確認し、必要に応じてSQL Serverをチューニングまたは SQL Serverに割り当てる実メモリを追加してください。それでもなお、タイムアウトのエラーが検出されるならば、PowerRDBconnector動作環境ファイルのTimeOutプロパティの値を大きくしてください。SQL Serverのチューニング方法については、SQL ServerのBooks Onlineを参照してください。

3.7 データ補正機能

本節は、アクセス時の例外的なデータに対する扱いについて説明します。データの表現形式についての説明であるため、COBOLアプリケーションがデータベースにアクセスでき、アクセスするデータ内容が明確になった際に、お読みください。

特に、ファイルシステムを用いたCOBOLアプリケーションからの移行や、データ表現が変わるような移行を行う場合に、お読みください。

3.7.1 後方空白補正

文字項目のデータで、設定したデータ長が項目長に満たない場合、COBOLランタイムは、指定したデータの後方に空白を設定します。COBOLアプリケーションで文字項目にデータを設定する場合に、半角空白や全角空白を意識して設定している場合と、空白を意識せず設定している場合とで、後方空白の扱いが異なります。

このどちらの場合でも、COBOLアプリケーションが想定したように動作できるようにするため、PowerRDBconnectorでは、後方空白補正機能を用意しています。

後方空白補正機能を使用した場合、以下のように後方空白が扱われます。

表3.32 後方空白補正機能を使用した場合の文字列データの後方空白の扱い

<table>
<tr><th rowspan="2">列名</th><th colspan="2">COBOLの入出力文</th></tr>
<tr><th>READ文</th><th>WRITE文 および REWRITE文</th></tr>
<tr><td>_CHAR
(英数字
項目)</td><td>半角空白を設定してアプリケーションへ通知します。</td><td rowspan="2">後方の半角空白と全角空白を削除し、データベースへ格納します。

その後、格納先のデータ型により、以下のように空白が設定されます。
・ 格納先のデータ型が固定長の場合
ー 格納先のデータ型がCHARの場合
データベースが半角空白を格納します。
ー 格納先のデータ型がNCHARの場合
データベースが以下のように後方に空白を設定します。
- データベース列長の不足分、半角空白を設定します。
- 全て空白の場合、データベースの列長分、半角空白を設定します。
・ 格納先のデータ型が可変長の場合
ー 空白以外の文字が存在する場合
半角空白も全角空白も後方に設定しません。</td></tr>
<tr><td>_NCHAR
(日本語
項目)</td><td>シフトJISで動作するCOBOLの場合
全角空白を設定してアプリケーションへ通知します。文字列が奇数バイトの場合、全角空白の設定前に、半角空白1文字を設定します。
注意)
Suppress プロパティに "ON/FULL" 、"ON/HALF"のどちらを指定しても、同じ動作となります。
unicodeで動作するCOBOLの場合
半角空白(Suppress プロパティに"ON/HALF"か"ON"を指定した場合)、または全角空白(Suppressプロパティに"ON/FULL"を指定した場合)を設定してアプリケーションへ通知します。文字列が奇数バイトの場合、全角空白の設定前に、半角空白1文字を設定します。</td></tr>
</table>

列名	COBOLの入出力文	
	READ文	WRITE文 および REWRITE文
		ー 全て空白の場合 半角空白を1文字設定します。

後方空白補正機能を使用しない場合は、PowerRDBconnectorが後方空白を設定したり削除したりすることはありませんが、データベースの列長に不足する文字列データを書き込んだときには、データベースによって列の後方に半角空白が設定されます。

後方空白の扱いを、COBOLランタイムと合わせるため、以下のとおりにSuppressプロパティを設定します。

- シフトJISのCOBOLアプリケーションの場合

  SuppressプロパティにOFF以外を指定します。

- NetCOBOL for WindowsでunicodeのCOBOLアプリケーションの場合

  SuppressプロパティにON/FULLを指定します。

- NetCOBOL for .NETでunicodeのCOBOLアプリケーションの場合

  SuppressプロパティにON/HALFまたはONを指定します。

[後方空白補正機能の使用方法]

- COBOLアプリケーション全体に指定する場合

  PowerRDBconnector動作環境ファイルのSuppressプロパティにON/FULL、ON/HALF、またはONを指定してください。

  例）PowerRDBconnector動作環境ファイル(DBIO_ENV)の記述例

```
; PowerRDBconnector動作環境ファイル
ServerName=snake.domain.com
DataSourceName=pubs
Suppress=ON/FULL
```

- 特定のファイルのみに指定する場合

  COBOL初期化ファイルのSuppressプロパティにON/FULL、ON/HALF、またはONを指定してください。

  例）ファイル EMPLOYEE に対するCOBOL初期化ファイル(COBOL85.CBR)の記述例

```
; COBOL初期化ファイル
EMPLOYEE=TableName=employee&SchemaName=dbo
&AccessMode=RANDOM&Suppress=ON/FULL,RDM
```

[後方空白補正に関する注意事項]

後方空白補正について、以下の注意が必要です。

- 日本語項目のデータで、設定したデータ長が項目長に満たない場合、COBOLランタイムは後方空白を以下のように扱います。
  - ー シフトJISのCOBOLアプリケーションの場合、データの後方に全角空白を設定します。
  - ー NetCOBOL for WindowsでunicodeのCOBOLアプリケーションの場合、データの後方に全角空白を設定します。
  - ー NetCOBOL for .NETでunicodeのCOBOLアプリケーションの場合、データの後方に半角空白を設定します。

・ 以下の場合は、後方空白補正を行わない設定(SuppressプロパティにOFFを指定)にしてください。

ー COBOLアプリケーション自身が、英数字項目や日本語項目の後方に空白を設定していて、この空白の種類(全角/半角)を変更されたくない場合

なお、後方空白補正を行わない場合は、以下の条件にすることが必要です。

ー シフトJISのCOBOLアプリケーションの場合

- 英数字項目(列名のサフィックスが_CHAR)は、データベースのCHAR、またはVARCHARのデータ型に対応させてください。
- 日本語項目(列名のサフィックスが_NCHAR)は、データベースのCHAR、NCHAR、VARCHAR、またはNVARCHARのデータ型に対応させてください。なお、NCHAR/NVARCHARのデータ型に対応させた場合、半角文字を格納することはできません。

ー NetCOBOL for .NETで、コード系にSJIS-UCS2(英数字項目がシフトJIS、日本語項目がunicode)を指定した場合

- 英数字項目(列名のサフィックスが_CHAR)は、データベースのCHAR、またはVARCHARのデータ型に対応させてください。
- 日本語項目(列名のサフィックスが_NCHAR)は、データベースのNCHAR、またはNVARCHARのデータ型に対応させてください。

これ以外の条件のCOBOLアプリケーションで、後方空白補正を行わない設定にすると、データ溢れが発生します。

・ PowerRDBconnector動作環境ファイルのSuppressプロパティの指定値は、COBOL初期化ファイルのSuppressプロパティの初期値となるため、PowerRDBconnector動作環境ファイルとCOBOL初期化ファイルにそれぞれ指定した場合、以下のように扱われます。

表3.33 Suppressプロパティの扱い

Suppressプロパティの指定		後方空白補正の有無
PowerRDBconnector動作環境ファイル	COBOL初期化ファイル	
ON/FULL	ON/FULL	有り(ON/FULL)
	ON/HALF(またはON)	有り(ON/HALF)
	OFF	無し
	指定なし	有り(ON/FULL)
ON/HALF (またはON)	ON/FULL	有り(ON/FULL)
	ON/HALF(またはON)	有り(ON/HALF)
	OFF	無し
	指定なし	有り(ON/HALF)
OFF	ON/FULL	有り(ON/FULL)
	ON/HALF(またはON)	有り(ON/HALF)
	OFF	無し
	指定なし	無し
指定なし(初期値ON/HALF)	ON/FULL	有り(ON/FULL)
	ON/HALF(またはON)	有り(ON/HALF)
	OFF	無し
	指定なし	有り(ON/HALF)

・ Suppressプロパティの初期値がON/HALFであるため、Suppressプロパティを指定していない場合は、無条件に後方空白補正が行われます。NetCOBOL for WindowsでunicodeのCOBOLアプリケーションの場合は、後方空白の扱いをCOBOLランタイムと合わせるため、SuppressプロパティをON/FULLに変更してください。

・ 後方空白補正機能を使用して格納したデータの扱いについて

「表3.32 後方空白補正機能を使用した場合の文字列データの後方空白の扱い」の“WRITE文およびREWRITE文”の欄にあるように、書込み時に、データベースの列長に満たない場合、半角空白がデータベースによって格納されます。この際、データ型がNCHARの列でも、全角空白ではなく半角空白が格納されます。

このデータは、以下のように扱われます。

― 後方空白補正を指定しアクセスすると、後方空白補正機能の扱いのとおりにデータが読み込まれます。

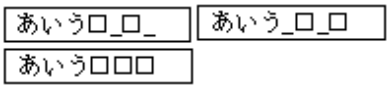
― 後方空白補正を指定せずにアクセスすると、日本語項目に半角空白が読み込まれるなど、期待する長さ以上の空白文字が読み込まれることになります。

― PowerRDBconnectorを使用しないアプリケーションの場合、期待する長さ以上の空白文字が読み込まれることになります。

・ 後方空白補正機能を使用しなかった場合の影響について

― 項目内のデータは項目長分すべてが評価されるため、キー値を指定する場合、空白以外の部分は一致しても、空白部分が全角空白と半角空白が混在していると、指定したキー値で思ったようにはアクセスできないことがあります。

例）以下は同じデータとして扱われません。
全角空白：□、半角空白：_ と示します。



・ 日本語項目にunicodeの文字コード系を使用する場合の注意事項

以下のCOBOLアプリケーションで、日本語項目の内容を文字比較すると、後方空白の扱いの違いから、想定外の比較結果(文字列が一致しない)となることがあります。

― NetCOBOL for Windowsで、実行時コード系がunicodeのCOBOLアプリケーションの場合で、かつ

― 後方空白補正(Suppressプロパティを指定しないか、ONまたはON/HALFを指定)を行った場合

現象を回避するには、SuppressプロパティをON/FULLに変更してください。

なお、PowerRDBconnectorを使用しない場合でも、日本語項目の後方空白の扱いを一致させないと、想定外の比較結果となります。詳細は、NetCOBOLのマニュアルを参照してください。

3.7.2 項目属性に違反するデータのチェックと補正

COBOLアプリケーションでは、レコード内の同じ位置のデータに対して、複数の属性を再定義したり、または複数のアプリケーションで異なった属性で定義したりすることが可能なため、項目属性に違反するデータが生じることがあります。

項目属性で規定されたデータとは、以下のデータです。

a. 英数字項目と日本語項目
データがすべて文字コードに準拠したコードから生成されています。

b. 外部十進項目
データが以下の形式で構成されています。

― ゾーン部：3

ー 数字部:0～9

ー 符号部:符号なしの場合は3、符号付の場合は4,5

c. 内部十進項目
データが以下の形式で構成されています。

ー 数字部:0～9

ー 符号部:符号なしの場合は0xF、符号付の場合は0xC、0xD

PowerRDBconnector経由では、データベースに異常なデータが設定されないよう、項目属性に違反するデータの扱いを選択することができます。この機能は、PowerRDBconnector動作環境ファイルのDataCheckプロパティにデータチェックの有無を設定することで使用できます。

- DataCheckプロパティに、データチェックを行う指定があった場合(DataCheck=C/U/Pなど)、データのチェックを行い、項目属性に違反する場合には、データ例外のエラーとします。詳細は、「3.7.2.1 データチェック」を参照してください。

- DataCheckプロパティに、データチェックを行わないと指定された場合(DataCheck=NONEなど)、データのチェックは行わず、補正可能なデータ部分をデータベース定義に応じて補正します。詳細は、「3.7.2.2 データ補正」を参照してください。

[データのチェックと補正の使用方法]

- データチェックを、すべての属性に対して行う場合、以下のいずれかを行ってください。

  ー PowerRDBconnector動作環境ファイルにDataCheckプロパティを記述しないでください。

  ー PowerRDBconnector動作環境ファイルにDataCheckプロパティを以下のように指定してください。

  PowerRDBconnector動作環境ファイル(DBIO_ENV)の記述例

```
; PowerRDBconnector動作環境ファイル
ServerName=snake.domain.com
DataSourceName=pubs
DataCheck=C/U/P
```

- データチェックを、特定の属性に対して行う場合、DataCheckプロパティを以下のように指定してください。以下の例では、データチェックを、英数字項目と日本語項目、および外部十進項目に対して行い、内部十進項目に対しては、データの補正を行います。

  PowerRDBconnector動作環境ファイル(DBIO_ENV) の記述例

```
; PowerRDBconnector動作環境ファイル
ServerName=snake.domain.com
DataSourceName=pubs
DataCheck=C/U
```

上記機能は、“英数字項目と日本語項目”、“外部十進項目”、および“内部十進項目”のそれぞれで指定できます。DataCheckプロパティの指定方法については、「3.1.4 PowerRDBconnector 動作環境ファイル」および、「表3.34 DataCheckプロパティの扱い」を参照してください。

[データのチェックと補正の注意事項]

DataCheckプロパティについて、以下の注意事項があります。

- DataCheckプロパティにデータチェックを行う指定がある場合、またはDataCheckプロパティを指定しない場合、違反している項目属性は、データ例外のエラーとなります。

・ データのチェックまたは補正は、以下の処理で動作します。

ー WRITE文またはREWRITE文実行時、データをデータベースに書き込む場合

ー キー指定でアクセス時、キー値を解釈する場合

・ PowerRDBconnector動作環境ファイルのDataCheckプロパティの指定値と、データのチェックする属性およびデータの補正する属性を以下に示します。

表3.34 DataCheckプロパティの扱い

DataCheckプロパティ		指定なし	C/U/P	C/U	C/P	U/P	C	U	P	NONE
データチェック	英数字項目と日本語項目	○	○	○	○	×	○	×	×	×
	外部十進項目	○	○	○	×	○	×	○	×	×
	内部十進項目	○	○	×	○	○	×	×	○	×
データ補正	英数字項目と日本語項目	×	×	×	×	○	×	○	○	○
	外部十進項目	×	×	×	○	×	○	×	○	○
	内部十進項目	×	×	○	×	×	○	○	×	○

○:対象

×:非対象

3.7.2.1 データチェック

DataCheckプロパティを指定しない、またはDataCheckプロパティに項目種類（DataCheck=C/U/P）を指定した場合、指定した属性のデータチェックは以下のように行われます。

a. 英数字項目と日本語項目
DataCheckプロパティに”C”を指定した場合、以下のデータチェックが行われます。

ー 項目データがすべて0x00となっている場合は、データ例外のエラーとなります。

ー データの途中に0x00が存在する場合、エラーとならず、そのままデータベースにデータを渡します。0x00の位置による動作は、「表3.38 データ途中に0x00が存在する場合の動作」を参照してください。

b. 外部十進項目
DataCheckプロパティに”U”を指定した場合、以下のデータチェックが行われます。

ー 項目定義の符号属性と、データの符号部が一致しているかチェックされます。

例）属性が符号なしであるが、データの符号部が4、5の場合

ー 符号部に異常なデータがないかチェックされます。

ー ゾーン部に異常なデータがないかチェックされます。

例）数値“123”のデータが、“0x313233”であるはずが“0x313203”などの場合

表3.35 外部十進のデータチェック

列情報	データ		結果	備考
	データ内容	COBOLのデータ定義例		
_UNSIGNあり	符号部が0x3* （符号なしデータ）	PIC 9(n)	正常	*は0～9 nは整数
	符号部が0x5* （符号ありの負の値）	PIC S9(n)	エラー	*は0～9 nは整数
	符号部が0x4* （符号ありの正の値）	PIC S9(n)	エラー	*は0～9 nは整数
_UNSIGNなし	符号部が0x3* （符号なしデータ）	PIC 9(n)	エラー	*は0～9 nは整数
	符号部が0x5* （符号ありの負の値）	PIC S9(n)	正常	*は0～9 nは整数
	符号部が0x4* （符号ありの正の値）	PIC S9(n)	正常	*は0～9 nは整数
_UNSIGNあり _UNSIGNなし	すべて空白	－	エラー	
	LOW-VALUE	－	エラー	
	HIGH-VALUE	－	エラー	
	0x00または0x20を含みます	－	エラー	
	ゾーン部が異常	－	エラー	
	数字部が異常	－	エラー	

c. 内部十進項目
DataCheckプロパティに”P”を指定した場合、以下のデータチェックが行われます。

－ 項目定義の符号属性と、データの符号部が一致しているかチェックされます。
例）属性が符号なしであるが、データの符号部がCの場合

－ 符号部に異常なデータがないかチェックされます。

表3.36 内部十進のデータチェック

列情報	データ		結果	備考
	データ内容	COBOLのデータ定義例		
_UNSIGNあり	符号部が0x*F （符号なしの値）	PIC 9(n) COMP	正常	*は0～9 nは整数
	符号部が0x*C （符号ありの正の値）	PIC S9(n) COMP	エラー	*は0～9 nは整数
	符号部が0x*D （符号ありの負の値）	PIC S9(n) COMP	エラー	*は0～9 nは整数
_UNSIGNなし	符号部が0x*F （符号なしの値）	PIC 9(n) COMP	エラー	*は0～9 nは整数
	符号部が0x*D （符号ありの負の値）	PIC S9(n) COMP	正常	*は0～9 nは整数
	符号部が0x*C （符号ありの正の値）	PIC S9(n) COMP	正常	*は0～9 nは整数

列情報	データ		結果	備考
	データ内容	COBOLのデータ定義例		
_UNSIGNあり _UNSIGNなし	数字部が0～9以外	―	エラー	

3.7.2.2 データ補正

以下のような場合には、データ内容と属性が矛盾することがあります。

- あるレコード内の同じ位置のデータに対して、複数の属性で再定義した場合
- 複数のアプリケーションから異なった属性で定義した場合

COBOLアプリケーションの数が多い、または影響する部分がどこかわからないなど、データ内容に応じた正しい属性で定義しなおすことができないときがあります。

この場合、データ補正を行うことで、COBOLアプリケーションの対応を少なくすることができます。

DataCheckプロパティに項目種類を指定しない場合、指定した属性のデータチェックを行わず、COBOLアプリケーションからデータベースへデータを渡すとき（REWRITE文、WRITE文およびキー指定時）、以下のように補正が行われます。

a. 英数字項目と日本語項目
DataCheckプロパティに”C”を指定しない場合、以下のように補正が行われます。

― 項目データがすべて0x00となっている場合、半角空白(0x20)に置き換えて正常なデータとして扱います。
なお、Suppressプロパティの指定によらず、半角空白(0x20)に置き換えます。

表3.37 文字データの補正

列名	COBOL文字コード指定	0x00と置き換える文字コード
_CHAR (英数字項目)	unicode	半角空白(0x20)
	シフトJIS	半角空白(0x20)
_NCHAR (日本語項目)	unicode	半角空白(0x20)
	シフトJIS	半角空白(0x20)

― データの途中に0x00が存在する場合、エラーとならず、そのままデータベースにデータを渡します。このため、コード系と0x00の位置によって以下のように動作します。

表3.38 データ途中に0x00が存在する場合の動作

COBOLの文字コード系	列名	動作
シフトJIS	-	0x00以降のデータが欠如します。
unicode	_CHAR (英数字項目)	0x0000 以降が欠如します。
		0x00nn の下位バイト以降が欠如します。
		0xnn00 以降が欠如します。 (nn:任意の16進数)
	_NCHAR (日本語項目)	0x0000 以降が欠如します。
		0x00nnや0xnn00のデータがあっても、データの欠如は発生しません。

COBOLの文字コード系	列名	動作
		(nn:00以外の任意の16進数)

b. 外部十進項目

DataCheckプロパティに”U”を指定しない場合、以下のように補正が行われます。

- HIGH-VALUE、LOW-VALUE、すべてが半角空白の場合は数値の0として扱います。
- 項目定義の符号属性とデータの符号に不一致があった場合、正の値として扱います。
- 符号部に異常があった場合、正の値として扱います。
- ゾーン部に異常があった場合、ゾーン部を補正し正常なデータとして扱います。

表3.39 外部十進のデータ補正

列情報	データ内容	COBOL 定義例	補正内容	備考
_UNSIGNあり	符号部が0x3* （符号なしデータ）	PIC 9(n)	正常	*は0～9 nは整数
	符号部が0x5* （符号ありの負の値）	PIC S9(n)	符号なしの正の値に補正します。	*は0～9 nは整数
	符号部が0x4* （符号ありの正の値）	PIC S9(n)	符号なしの正の値に補正します。	*は0～9 nは整数
_UNSIGNなし	符号部が0x3* （符号なしデータ）	PIC 9(n)	正の値に補正します。	*は0～9 nは整数
	符号部が0x5* （符号ありの負の値）	PIC S9(n)	正常	*は0～9 nは整数
	符号部が0x4* （符号ありの正の値）	PIC S9(n)	正常	*は0～9 nは整数
_UNSIGNあり _UNSIGNなし	すべて半角空白	－	数値の0に置換します。	
	LOW-VALUE	－	数値の0に置換します。	
	HIGH-VALUE （キー値の指定時）	－	数値の0に置換します。	
	HIGH-VALUE （書込み時）	－	エラーとなります。	
	0x00または0x20を含みます。	－	異常部分を数値の0に置換します。（注1）	
	ゾーン部が不正	－	ゾーン部に0x3*を設定します。	*は0～9
	数字部が異常	－	エラーとなります。	

（注1)

最終バイトが0x00、0x20の場合、符号無の場合は 0x30、符号付の場合は0x40として置換します。

例）

- 前方に空白ありの場合 ’ 123’⇒‘00123’として扱います。
- 後方に空白ありの場合 ’123 ’⇒‘12300’として扱います。
- 混在した場合 ’1 2 3’⇒‘10203’として扱います。

c. 内部十進項目
DataCheckプロパティに”P”を指定しない場合、以下のように補正が行われます。

ー 符号部に異常なデータがあった場合、正の値として扱います。

注意

数字部に異常があった場合、データ補正ができないため、データベースアクセス時にエラーとなります。

表3.40 内部十進のデータ補正

列情報	データ内容	COBOLのデータ定義例	補正内容	備考
_UNSIGNあり	符号部が0x*F （符号なしデータ）	PIC 9(n) COMP	正常	*は0～9 nは整数
	符号部が0x*D （符号ありの負の値）	PIC S9(n) COMP	正の値に補正します。	*は0～9 nは整数
	符号部が0x*C （符号ありの正の値）	PIC S9(n) COMP	正の値に補正します。	*は0～9 nは整数
_UNSIGNなし	符号部が0x*F （符号なしデータ）	PIC 9(n) COMP	正の値に補正します。	*は0～9 nは整数
	符号部が0x*D （符号ありの負の値）	PIC S9(n) COMP	正常	*は0～9 nは整数
	符号部が0x*C （符号ありの正の値）	PIC S9(n) COMP	正常	*は0～9 nは整数
_UNSIGNあり _UNSIGNなし	数字部が0～9以外	ー	エラー	

第4章 COBOLアプリケーションの開発について

本章では、COBOLアプリケーションの開発で考慮が必要なことを説明します。

開発者およびシステム管理者が、他のシステムからCOBOLアプリケーションを移行する場合に注意すべきこと、およびCOBOLアプリケーション開発時の注意事項を確認するときにお読みください。

4.1 開発のポイント

本節では、PowerRDBconnectorを使用したCOBOLアプリケーションの開発について説明します。

COBOLアプリケーションのファイルアクセス機能とデータベースで備える機能では、インターフェースやデータ属性など、機能の相違点を考慮してCOBOLアプリケーションを開発しなければなりません。特に、既存のCOBOLアプリケーションを活用して開発する場合には、本章を参照してください。

PowerRDBconnectorで使用できるCOBOLアプリケーションのファイルアクセス機能は、NetCOBOLが提供する機能と一部異なりますので、事前に「2.2.1.2 COBOLアプリケーションから利用できる機能範囲」を参照して確認してください。

4.1.1 環境設定のポイント

環境設定時のポイントを説明します。

4.1.1.1 ファイル識別名について

COBOL初期化ファイルに指定するファイル識別名は、一意の名前で指定してください。

以下のような入出力文COBOLアプリケーションでは、ASSIGN句に指定するファイル識別名に同じ名前を使用することが多いため、既存のCOBOLアプリケーションを活用する場合には、ファイル識別名が一意になるように変更してください。

- 複数のSELECT句に同一ファイル名で定義し、かつ
- 異なったアクセスモードで定義しているアプリケーション。

4.1.1.2 データベースの照合順序について

キー値およびレコードの並び順は、データベースの照合順序に依存します。照合順序の変更が必要な場合には、データベースを構築する前に行ってください。

データベースの照合順序については、SQL ServerのBooks Onlineを参照してください。

4.1.2 データベース作成のポイント

データベースの作成時のポイントを説明します。

4.1.2.1 混在項目を使用する場合

混在項目をデータベースの1つの列に対応付けることはできません。混在項目をデータ型毎に分割してCOBOLのレコード記述項を変更してください。変更したレコード記述項に合わせてデータベースの列を定義してください。

4.1.2.2 OCCURS句、REDEFINES句を使用する場合

最下位レベル番号の基本項目に合わせてデータベースの列を定義してください。

4.1.2.3 ビューについて

COBOLアプリケーションからビューへアクセスする場合、テーブルのアクセスに比べていくつかの制限事項があります。必要に応じてデータベースの構成を変更して対処してください。ビューに関する注意については、「4.2.3.2 ビューの使用について」を参照してください。

4.1.2.4 列の定義について

COBOLのレコード記述項の定義とデータベースの列定義が正しく対応されていないと、COBOLアプリケーションは正しく動作しません。

NetCOBOLは、COBOLのレコード記述項のレコード長と、データベースで定義された列長が一致しているか、OPEN文でチェックします。レコード記述項に合わせて列と、列名のサフィックスを定義してください。ただし、COBOLの項目と関連付けないデータベースの列を定義する場合を除きます。

レコード長が一致しない場合、NetCOBOLが以下の実行時エラーを通知します。

- “JMP0310I-I/U ～ファイルでOPENエラーが発生しました. INV-LRECL”

「3.1.3.5 列定義の対応」を参照して、COBOLのレコード記述項とデータベースの定義を確認してください。

COBOLの項目と関連付けないデータベースの列を定義する場合には、「3.1.3.6 COBOLと関連付けのない項目の指定方法」を参照してください。

上記以外のNetCOBOLの実行時エラーについては、「5.1.1.4 COBOLアプリケーション終了時のメッセージ一覧」を参照してください。

4.1.2.5 データベースで扱えないデータ値との整合性について

データベースには、データ型で定義した値のみを格納できます。

- 数字型へは、数値のみ格納できます。
- 文字型へは、文字のみ格納できます。

例えば、以下のようにして、COBOLの数字項目に文字データを格納することはできません。

```
FD TPFILE.
01 REC.
  02  TIME.
      03  TIME-HH   PIC 9(2).
      03  TIME-MM   PIC 9(2).
      03  TIME-SS   PIC 9(4).

MOVE BLANK TO TIME.  <-- 項目 TIME は、英数字項目と評価され、空白が設定されます。
WRITE REC.           <-- 数字項目に文字データが含まれるため、WRITE文はエラーとなります。
```

基本項目のデータ型でデータを入力してください。

なお、PowerRDBconnectorでは、上記の不整合なデータに対して、エラーにするか補正するかが選択できます。詳細については、「3.7 データ補正機能」を参照してください。

4.1.2.6 インデックスの作成について

RECORD KEY句を使用してアクセスするテーブルには、必ず、RECORD KEY句で定義したキーに対応するインデックスをCREATE INDEX文で定義してください。インデックスを定義しないとデータ更新できません。

インデックスには、UNIQUE制約またはプライマリー制約を推奨します。UNIQUE制約またはプライマリー制約を設定しないと、キー値が重複することがあります。キー値が重複した場合、注意が必要です。キー値が重複した場合の注意については、「4.2.3.1 キーに重複した値がある索引ファイルについて」を参照してください。

4.1.2.7 文字コードについて

COBOLアプリケーションとデータベースで扱う文字のコード系が一致しない場合、アプリケーションとデータベースの文字コードを同一にしてください。コード系が一致しない場合でも、PowerRDBconnectorは特殊な変換は行いません。

ユーザー定義文字は、シフトJISのコード系の範囲およびunicodeのコード系の範囲内で使用できます。

PowerRDBconnectorは、unicodeコード系の場合に、UTF-8とUCS2(UTF-16)との間でコード変換を行う場合があります。文字コード変換の注意については、「4.2.3.4 文字コード変換について」を参照してください。

SQL Serverの文字コード系と、COBOLアプリケーションの文字コードの対応について以下に示します。

表4.1 COBOLアプリケーションの実行時コード系とSQL Serverの文字コード系について

COBOLアプリケーションの文字コード系			SQL Serverの文字コード系	
			シフトJIS	UCS2(UTF-16)
種別	文字コード系	USAGE句	CHAR/VARCHAR	NCHAR/NVARCHAR
NetCOBOL for .NET	SJIS系	X(英数字)	○	○
		N(日本語)	○	○
	UTF8系	X(英数字)	○(注)	○
		N(日本語)	○(注)	○
	混在系	X(英数字)	○	○
		N(日本語)	○(注)	○
NetCOBOL for Windows	SJIS系	X(英数字)	○	○
		N(日本語)	○	○
	UTF16系	X(英数字)	○(注)	○
		N(日本語)	○(注)	○

(注)シフトJISの範囲でしかデータが入りません。シフトJIS範囲外のデータが入った場合、文字化けすることがあります。

文字コード系の表記は、NetCOBOL for .NETまたはNetCOBOL for Windowsをコンパイル時に以下のオプションを選択した場合です。

SJIS系:RCSオプションにSJISを指定したCOBOLアプリケーション

UTF8系:RCSオプションにUTF8-UCS2を指定したCOBOLアプリケーション

混在系:RCSオプションにSJIS-UCS2を指定したCOBOLアプリケーション

UTF16系:RCSオプションにUCS2またはUTF16を指定したCOBOLアプリケーション

上の表は、シフトJISのCOBOLアプリケーションから、unicodeのデータベースにアクセスできるということを示しています。

COBOLランタイムが補正する空白文字種別の違いがありますので、シフトJISのときと全く同じCOBOLアプリケーションが動作できるとは限りません。詳細は、「[後方空白補正に関する注意事項]」を参照してください。

【JIS2004の文字コードについて】

JIS X 0213:2004に対応したWindowsの文字セット(JIS2004)でサポートされた1文字が4バイトの文字(JIS2004固有文字)を扱う場合について以下に示します。

- JIS2004固有文字が使える環境

  以下の環境でJIS2004固有文字は使用できます。

  ー Windows Server 2008(x64、R2を含む)、Windows Vista、Windows 7(x64を含む)の場合

  ー NetCOBOL V10を使用する場合

  上記の場合、COBOLのUSAGE句には、1文字に、JIS2004以外の1文字を使用する2倍の定義を行ってください。なおデータベースの定義も、2倍の定義に対応した長さを指定してください。

- JIS2004固有文字が使えない環境

  以下の環境ではJIS2004固有文字は使用できません。

  ー NetCOBOL V10以外のNetCOBOLを使用する場合

  上記の場合、JIS2004で追加された文字(JIS2004の文字セットでは、4バイトの文字や、シフトJISコード(JIS X 0208-1990)に存在しない文字)をレコード内のデータに使用すると、エラーや、文字化けが生じることがあります。

  また、パス名、スキーマ名、表名、列名に使用すると、エラーとなります。このため、JIS2004で追加された文字は、使わないでください。

  ー SQL Server 2005の場合は、データベースの問題により、JIS2004固有文字は正しく検索できません。

4.1.2.8 ユーティリティを使用する場合

COBOLアプリケーション以外のアプリケーションやユーティリティとテーブルを共用する際は、COBOLで扱えるデータが格納されるようにしてください。

- 符号なし数字項目および符号なし整数項目にマイナス値が格納されてしまうことがあります。

  符号なし数字項目および符号なし整数項目でマイナス値は扱えません。

  マイナス値の格納を制限したい場合は、データベースの機能(CHECK制約)で可能です。

- 整数項目の精度(p)より大きい整数値が格納されてしまうことがあります。

  整数値の範囲を制限したい場合は、データベースの機能(CHECK制約)で可能です。

- NULL値が格納されてしまうことがあります。

  NULL値は扱えません。NULL値が格納されないように制限したい場合は、データベースの機能(NOT NULL制約)でNULL値の書込みを制限できます。

・ 2進項目の精度(p)より大きい整数値が格納されてしまうことがあります。

SMALLINT、INTEGER、BIGINTには以下の整数値が格納できます。整数値の範囲を制限したい場合は、データベースの機能(CHECK制約)で可能です。

ー SMALLINT

-2^15 (-32,768) から2^15 - 1 (32,767) までの整数値

ー INTEGER

-2^31 (-2,147,483,648) から2^31 - 1 (2,147,483,647) までの整数値

ー BIGINT

-2^63 (-9,223,372,036,854,775,808) から2^63 -1 (9,223,372,036,854,775,807) までの整数値

・ データ溢れが発生することがあります。

文字データをデータベースに格納する場合、データ溢れが発生することがありますので、以下の注意が必要です。データ溢れが発生すると、COBOLのFILE STATUSに90を通知します。

ー X項目とCHAR またはNCHAR を対応させている場合

CHARまたはNCHARに全角文字が格納されていると、データ溢れが発生します。

例えば、以下の場合に発生します。

- X(2) のX項目に対して、CHAR(2) の全角文字'A'をREADするとX項目をunicode（UTF-8）で扱うNetCOBOL for .NETで3バイトが必要となり、データ溢れが発生します。
- X(4) のX項目に対して、NCHAR(4) の全角文字'ABCD'をREADすると8バイトが必要となり、データ溢れが発生します。

ー N項目とCHAR を対応させている場合

N項目をunicode(UCS2)で扱うNetCOBOL for .NETでデータ溢れが発生します。

例えば、N(2) のN項目に対して、CHAR(4) の半角文字'ABCD'をUCS2でREADすると8バイトが必要となり、データ溢れが発生します。

CHAR(p) は、半角文字だけであれば、p文字格納できますが、全角文字を含むと、p文字は格納できません。

NCHAR(p) は、全角文字だけでp文字格納できます。

ー 可変長項目を使用している場合

可変長項目に、COBOLの項目長より長いデータが存在した場合、データ溢れが発生します。

データ溢れが発生すると、以下のようになります。

- READ文実行時、COBOLの項目長より長いデータは切り捨てられます。
- キー項目が可変長項目に対応していた場合、キー指定のREWRITE文、DELETE文で、キー値の位置付け時にエラーとなります。

4.1.3 COBOLアプリケーション作成のポイント

COBOLアプリケーション作成時のポイントを説明します。

4.1.3.1 トランザクションについて

トランザクションを使用する場合、以下の点に注意して、トランザクションサブルーチンを使用してください。

- シングルセッションでトランザクションサブルーチンを使用した場合、サブルーチン内でエラーが発生すると、プログラムが強制的に終了されます。エラーが発生しても、エラーの対処を行って、プログラムを継続したい場合は、マルチセッションでトランザクションサブルーチンを使用してください。
- I/Oオープンしたファイルに対して、トランザクション区間内で、READ文を実行すると、更新の有無によらずレコードロックを獲得します。このため、トランザクション区間内で大量のレコードをアクセスすると、大量にレコードロックを獲得してしまい、共用度が低下します。トランザクション区間内でアクセスするレコードは、できるだけ少なくしてください。
- トランザクション開始は、OPEN文の前でも後でも実行できます。
- トランザクション終了は、CLOSE文の前に実行してください。

  トランザクション終了前に、CLOSE文を実行した場合、以下のように動作します。

  ー 最終のCLOSE文を実行した場合

  CLOSE文の実行時に、トランザクションは取り消されます。

  以降はトランザクション区間外となるため、トランザクション開始が実行できます。

  ー 最終でないCLOSE文を実行した場合

  トランザクション区間は継続します。

  最終のCLOSE文とは、そのCLOSE文を実行すると、セッション内でPowerRDBconnectorを使ってオープンしているファイルが1つもなくなるCLOSE文のことです。
- レコードロックを獲得できず、タイムアウトエラーやデッドロックエラーが発生しても、PowerRDBconnectorはトランザクションが継続している状態になっています。トランザクションを取り消す場合には、トランザクション取消しサブルーチンを実行してください。

  ただし、デッドロックエラーが発生した場合、SQL Serverのトランザクションは自動的にロールバックされます。このため、データの更新は取り消されます。詳細は、「4.2.2.4 デッドロック状態について」を参照してください。
- 次の場合、トランザクション終了または取消しを行っても、ロックを獲得したままのレコードが残ることがあります。

  ー レコードロックを獲得中に、トランザクション開始を行った場合

  ー I/Oオープンしたファイルに対して、トランザクション区間内で、READ文を実行し、そのままトランザクション終了または取消しを行った場合
- トランザクション取消し後、カーソル位置は不定となります。このため、取消し後に再度アクセスを行う場合は、再度START文を実行するなどして、カーソル位置を指定し直してください。

4.1.3.2 テーブルロック機能について

テーブルロックを指定したファイルをオープンすると、強制的にトランザクションを開始するため、COBOLアプリケーションのトランザクション制御で不具合が生じないか、以下の動作について確認してください。

- レコードの更新、削除、および追加処理で、データベースからトランザクション確定ができないエラー（データベースのサービス停止など）が通知されると、オープン直後の状態に戻ります。
- テーブルロックを指定したファイルのオープン中は、トランザクションの操作（開始、確定、取消し）が無効となります。トランザクションサブルーチンのCALL文は何もせず、正常に復帰します。テーブルロックが有効なファイルが同じセッション内で全てクローズされた時点でデータを確定します。
- テーブルロック対象のファイルとレコードロック対象のファイルが混在使用されている場合、テーブルロック対象ファイルのCLOSE文の実行タイミングによっては、レコードロック対象のファイルの複数レコードもロックされます。詳しくは、「4.2.2.3 テーブルロックの排他制御について」を参照してください。
- テーブルロック区間で、“XMROTRBK”または、”COB_PRDB_TRAN” (トランザクション種別:4)サブルーチンを呼び出していると、テーブルロック区間内で更新されたデータは取り消されます。詳しくは、「3.5.3.2 テーブルロック解除時のトランザクション取消し機能」を参照してください。

4.1.3.3 レコードロック機能について

多重動作するCOBOLアプリケーションで、I-Oモードでオープンし、START文やREAD文でレコードを操作する場合、以下のようにCOBOLアプリケーションで対処してください。

- トランザクションを適用しない場合

  ー DYNAMICアクセスモードの場合、START FIRST文では、位置付けされた先頭のレコードのレコードロックは獲得されません。READ文でレコードを読み込んでレコードロックを獲得してください。

  ー SEQUENTIALアクセスモードやDYNAMICアクセスモードの場合、READ文で読み込んだレコードに対して、REWRITE文(更新)やDELETE文(削除)を行っても、レコードロックは解除されません。業務アプリケーションで、画面の入力待ちを行うためなど、レコードロックを解除したい場合、以下の方法で、レコードロックを解除してください。

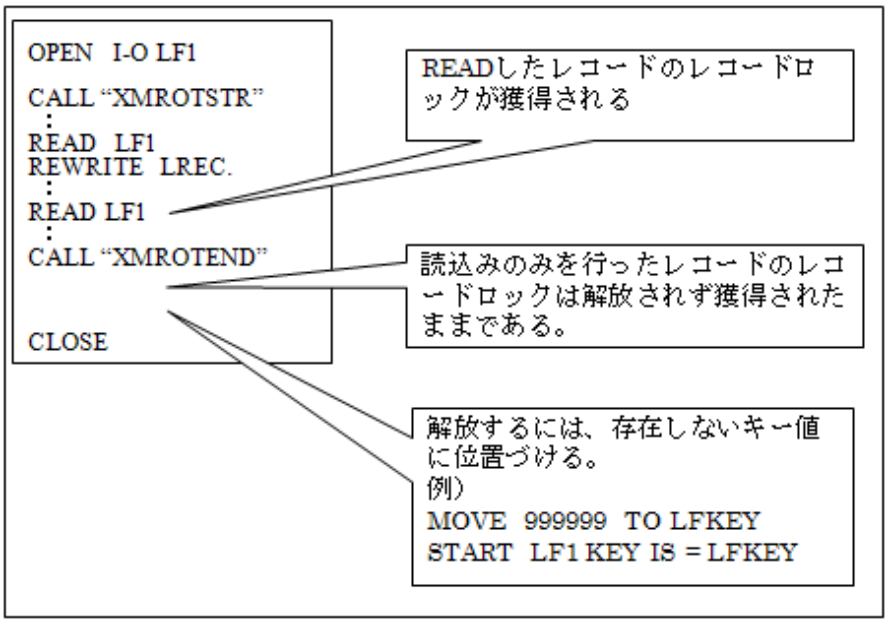
  - 別レコードをREAD文で読み込む。ただしこの場合、読み込んだレコードのレコードロックは獲得されます。
  - DYNAMICアクセスモードでSTART FIRST文を実行する。
  - アクセスしているテーブルに存在していないキー値を指定して、レコードに位置付け(START)を行って、“レコードなし”のエラーを発生させる。
  - EOFを検出するように、読込み(READ)を行う。

  ただし、上記は、どちらも次に読込む位置(カーソル位置)が変わりますので、注意してください。

- トランザクションを適用する場合

  トランザクションを確定または取消しする直前に、位置付けまたは読込んだレコードのレコードロックは解放されません。解放するには、トランザクションを確定、または取消し後、つまりトランザクションを適用しない状態となっている間に、上記の方法でレコードロックを解放してください。

図4.1 トランザクション確定後に解放されないロックを解放する



4.1.3.4 OUTPUTモードのオープンについて

OUTPUTモードでオープンしているファイルは、同じCOBOLアプリケーション内または別のCOBOLアプリケーションでオープンすることができません。OUTPUTモードでオープンするファイルは、他のCOBOLアプリケーションと共用しないようにしてください。OUTPUTモードでオープンするファイルは、テーブルロックを行ってください。

ビュー表に対してOUTPUTモードでオープンする場合は、COBOL初期化ファイルのTruncateプロパティに導出元表名を指定してください。

OUTPUTモードでオープンする場合の注意について、詳しくは「4.2.3.5 OUTPUTモードのオープンの使用について」を参照してください。

4.1.3.5 レコードの削除方法について

レコードを削除するには、必ずDELETE文を使用してください。レコードの先頭にバイナリデータの‘FF’を書き込んでレコード削除することはできません。

4.1.3.6 LOW-VALUE、HIGH-VALUEについて

表意定数のLOW-VALUE、HIGH-VALUEを使用した全ての入出力文操作はできません。

- LOW-VALUE、HIGH-VALUEの値を使用したWRITE文およびREWRITE文など
- LOW-VALUE、HIGH-VALUEの値をキー値としたSTART文およびREAD文など

論理的にレコードを位置付けるには、以下の代替方法があります。

- LOW-VALUE :START ファイル名 FIRST RECORD KEY IS [データ名-1]…]
- HIGH-VALUE:START ファイル名 FIRST
- RECORD KEY IS [データ名-1]…] WITH REVERSED ORDER

データ補正の使用有無によって、LOW-VALUE、HIGH-VALUEを指定した場合に、エラーまたは、数値の0に補正されます。詳細は、「3.7.2 項目属性に違反するデータのチェックと補正」、「表3.35 外部十進のデータチェック」、および「表3.36 内部十進のデータチェック」を参照してください。

4.1.3.7 使用メモリ量について

PowerRDBconnectorで使用するメモリ量の概算値を以下に示します。なお、計算式には、NetCOBOLやデータベース内部で獲得されるメモリ容量は含まれません。

- プロセス終了まで保持するメモリ容量

  513×(セッション数、シングルセッションプログラミングの場合は1)(単位:Kバイト)

- 使用スタック域の容量(各スレッド単位)

  64 (単位:Kバイト)

- 1つのファイルをオープンする単位でCLOSE文まで保持するメモリ容量

  (データベースで定義している列名長の合計)+(レコード長×16)+(項目数×68)+(キー項目数×24)+
  {項目数×(COBOLに記述している最大の列名長+10)+200}(単位:バイト)

  ※複合キーの場合、キー項目数とは構成している基本のキー項目の総和です。

- 1つのファイルをオープンする単位でOPEN文実行時に必要なメモリ容量

  (レコード長×3)+(項目数×72)+64,000(単位:バイト)

項目数、列名長およびキー項目の数が多いほどメモリ使用量が増加するため、必要以上にキー項目の数を増やすなどを行わないよう注意してください。

4.1.3.8 マルチセッションプログラミングについて

マルチセッションプログラミングを使用する場合、セッションサブルーチンを組み込む必要があります。このためシングルセッションプログラミングで記述されたCOBOLアプリケーションをそのままマルチセッションプログラミングでは使用できません。以下の注意点を考慮し、必要な箇所を修正してください。

- マルチセッションプログラミングでは、セッションサブルーチンを使用してください。セッションサブルーチンについては、「3.2 セッションの制御方法」を参照してください。

- トランザクションサブルーチンや認証情報登録サブルーチンの名称および記述方法を修正してください。トランザクションサブルーチンについては、「3.4.3 トランザクションの使用方法（マルチセッション）」を、認証情報登録サブルーチンについては、「3.3.3 データベース認証の使用方法（マルチセッション）」を参照してください。

- トランザクションサブルーチンでエラーが発生した場合、シングルセッションプログラミングでは強制的にプログラムが終了しますが、マルチセッションプログラミングでは、エラー通知されます。このためマルチセッションプログラミングで、トランザクションサブルーチンを使用するときには、エラー処理が必要です。トランザクションサブルーチンのエラーについては、「5.1.2 トランザクションアクセス時のエラー情報」を参照してください。

4.1.4 性能向上のポイント

運用により性能向上させるためのポイントを説明します。

4.1.4.1 インデックスの定義

検索性能を向上させるためインデックスを作成し、索引ファイルによるファイルアクセスで最適な性能が得られます。

4.1.4.2 データ更新時のトランザクション適用

データ更新性能を向上させるには、トランザクションを適用して更新データをまとめて確定する考慮が必要です。

データベースでは、データ更新を行うと更新されたデータは、データファイルに加えて、トランザクションログファイルへも書込み処理が行われます。

データ更新性能を向上させるには、書込み回数を減少させてください。トランザクションを適用すると、トランザクション中のデータ更新の書込み回数が最小限に抑えられ、更新されたデータは確定することで、まとめて書き込まれます。

例えば、COBOLアプリケーションで、トランザクション開始後に、WRITE文を1万回実行する毎に確定させるなどの方法があります。

テーブルロック機能を使用すると、強制的にトランザクションが適用されるため、データ更新はクローズ時に確定しますので、書込み回数を減少させることができます。

なお、トランザクション操作を行う際の注意について、「4.2.2.3 テーブルロックの排他制御について」を参照してください。

4.1.4.3 トランザクションログファイルのディスク配置

トランザクションログファイルは、データファイルと別の物理ディスクに配置することを推奨します。

SQL Serverは、データ更新を行うと更新されたデータをデータファイルに書込み、トランザクションログファイルへも書き込みます。

データファイルと別の物理ディスクに配置しないと書込みが1つの物理ディスクに集中するため、ディスクへの書込みが頻繁に行われ、性能が低下する場合があります。また、データファイルの物理ディスクにハード障害が発生した場合、トランザクションログファイルも破壊され、バックアップした時点のデータでしか復旧できなくなります。

4.1.4.4 データベース・ユーティリティの活用

順アクセスの性能(特にWRITE)は、一般にファイルシステム系製品と比較して、処理時間がかかります。大量のデータコピーを行う場合は、PowerRDBconnectorを使用せずにデータベース製品で提供されているユーティリティを活用してください。

データコピーを前提に設計されたデータベース製品のユーティリティは、そのデータベース製品で最も高速にデータコピーできるように設計されています。

SQL Serverでは、以下のユーティリティでデータを高速にコピーできます。

- SQL Server Integration Services

  データベースをコピーします。

- bcp ユーティリティ

  テーブルやビューをコピーします。

4.2 注意事項

本節では、「4.1 開発のポイント」で説明した開発手法が適用できない場合の注意事項を説明します。

4.2.1 トランザクションに関する注意事項

ここでは、トランザクションに関する注意事項を説明します。

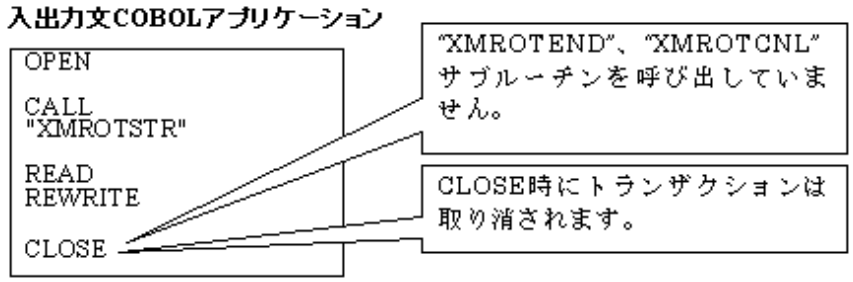
4.2.1.1 トランザクションの確定、取消しを実行せずに終了した場合

COBOLアプリケーションではトランザクションの確定または取消し、およびファイルのクローズを行ってください。

トランザクションサブルーチンを使用するCOBOLアプリケーションにおいて、トランザクションの確定または取消し、およびファイルをクローズせずにCOBOLアプリケーションが終了した場合、トランザクションは以下のようになります。

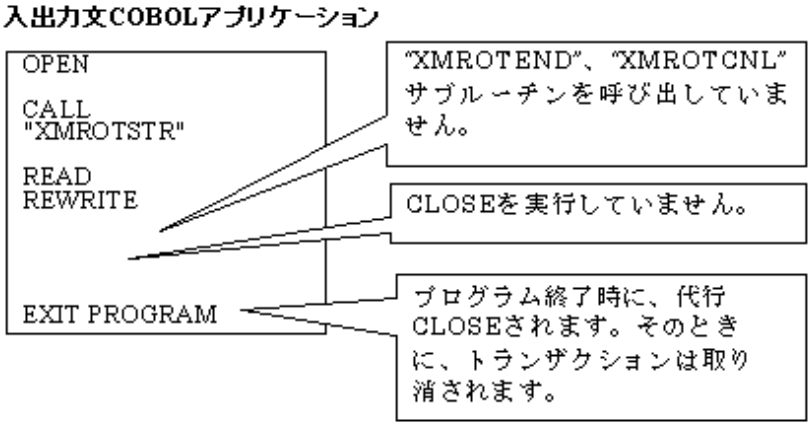
・ トランザクションの確定または取消しを行わずに全てのファイルをクローズした場合、トランザクションは取り消されます。

図4.2 トランザクションの確定、取消しを実行しない場合のトランザクション処理



・ トランザクションを終了（確定または取消し）せず、またはファイルをクローズせずにアプリケーションが終了した場合、トランザクションは取り消されます。

図4.3 トランザクションの確定、取消し、およびCLOSEを実行しない場合のトランザクション処理



・ タスクマネージャなどでCOBOLアプリケーションを強制終了した場合、データベースにより、トランザクションは取り消されます。

・ COBOLアプリケーションが、トランザクションを終了（確定または取消し）する前に、アプリケーションエラーで中断した場合、データベースによりトランザクションは取り消されます。

・ COBOLアプリケーション動作中に、データベースサービスを中断またはネットワークを中断した場合、トランザクションは取り消されます。

図4.4 プログラム中断した場合のトランザクション処理

4.2.1.2 テーブルロック機能とトランザクションについて

テーブルロックを指定したファイルをオープンすると、強制的にトランザクションを開始するため、テーブルロックを指定したファイルのオープン中は、トランザクションの操作（開始、確定、取消し）が無効となります。トランザクションサブルーチンのCALL文は何もせず、正常に復帰します。テーブルロックが有効なファイルが全てクローズされた時点でデータを確定します。

このため、できるだけテーブルロックとトランザクションを同時に使わないようお勧めします。同時に使った場合には、トランザクションは以下のようになります。

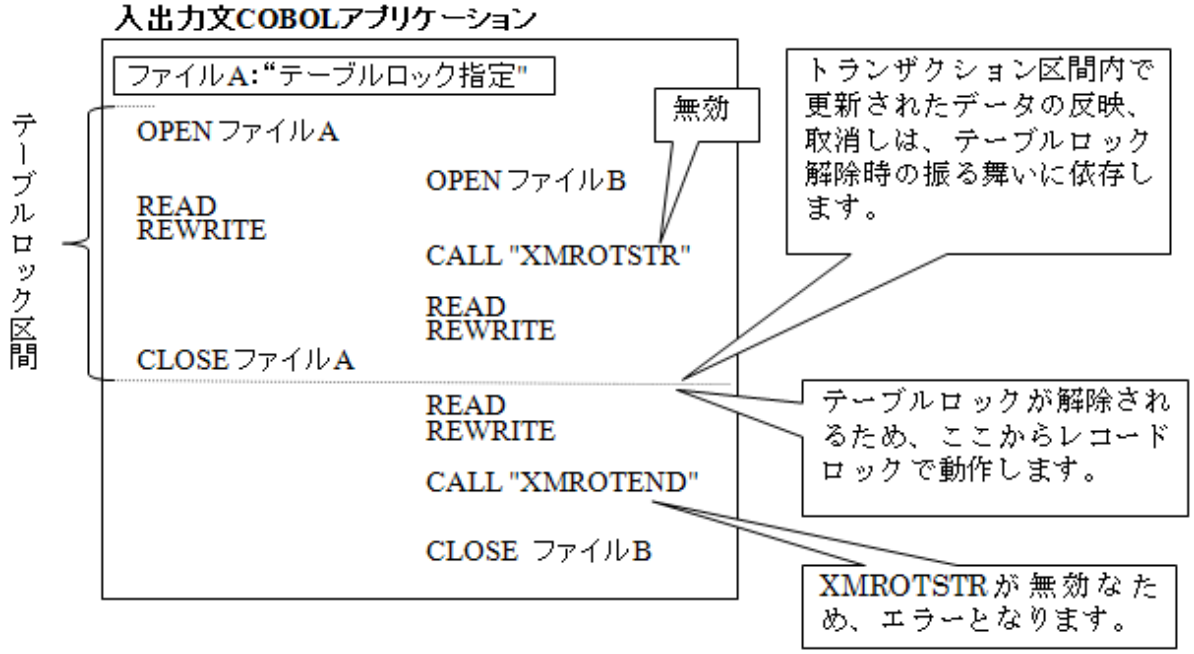
(1)テーブルロック開始後にトランザクションの開始が行われる場合

テーブルロック区間に含まれるトランザクション区間内で更新されたデータの反映や、取消しは、テーブルロック解除時の更新データの扱いと同じに扱われます。

このため、テーブルロック解除時のトランザクション取消し機能が実行されると、トランザクション区間内の更新も取り消されます。テーブルロックについては、「3.5.3 テーブルロック機能」を参照してください。

テーブルロックが解除されたとき、トランザクション区間が続いているようなシーケンスであっても、トランザクション開始処理は無効であるため、トランザクションとはなっておらず、レコードロックで動作します。

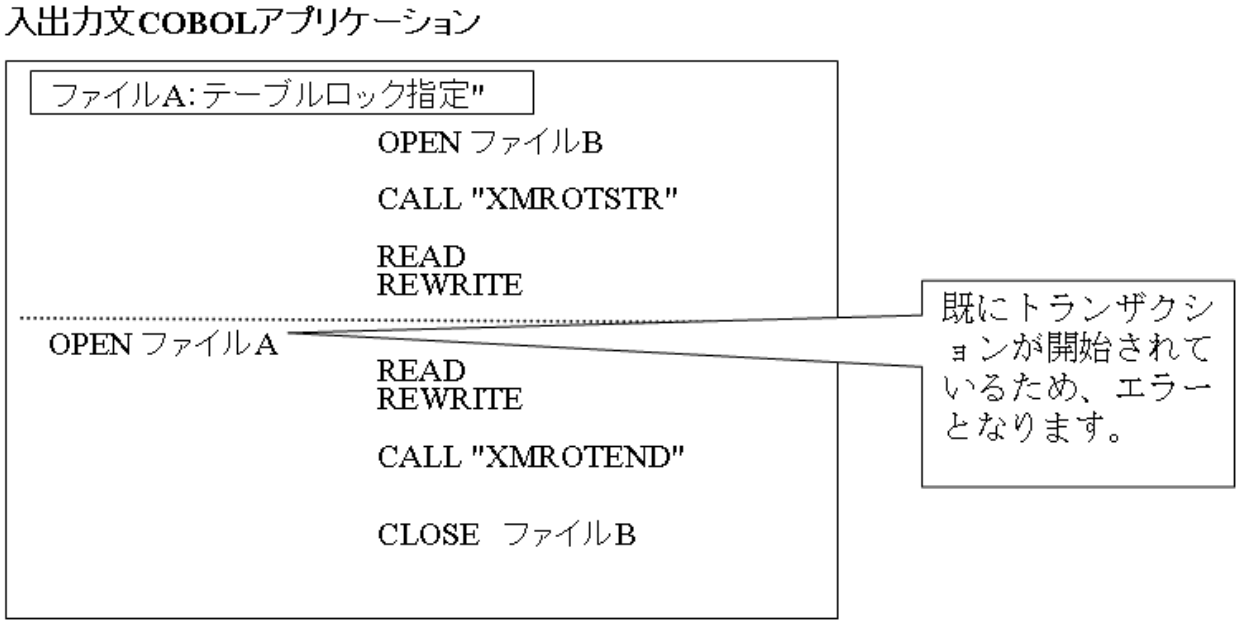
図4.5 テーブルロック開始後にトランザクションが開始された場合のトランザクション処理



(2)テーブルロック開始前にトランザクションの開始が行われる場合

テーブルロックを開始しようとするとき、既にトランザクションが開始しているため、エラーとなります。

図4.6 テーブルロック開始前にトランザクションが開始された場合のトランザクション処理



4.2.2 排他制御に関する注意事項

ここでは、排他制御に関する注意事項を説明します。

4.2.2.1 COBOLアプリケーションを終了した場合

- テーブルロックで強制的に開始されたトランザクションは確定されます。
  トランザクションを取消す場合は、トランザクションサブルーチンのテーブルロック解除時の取消しを実行してください。
- テーブルロックまたはレコードロックの解放が遅れる場合があります。
  このため、直後に実行するCOBOLアプリケーションでタイムアウトまたは排他エラーが発生する場合があります。しばらくしてからCOBOLアプリケーションを再実行してください。

4.2.2.2 削除したレコードの待合せについて

UNIQUE制約のインデックスを持つテーブルに対して、以下の条件を満たす場合には、READ文、REWRITE文およびDELETE文だけでなく、WRITE文でもレコードロックの待合せが発生します。

- レコードロックを獲得し、かつ、レコードを削除する場合
- レコードロックを解放していない状態にて、他のセッションやアプリケーションから削除したレコードと同一のキー値に対し、レコードの書き込みを行う場合

4.2.2.3 テーブルロックの排他制御について

テーブルロックを使用するセッションは、ファイルのOPEN文で強制的にトランザクションを開始してCLOSE文でトランザクションを確定することでテーブルロックを実現します。テーブルロックの排他制御で動作するトランザクションについて、以下の注意があります。

- トランザクションは、テーブルロックが有効なファイルが全てクローズされるまで確定しません。(「参考例2」を参照してください。)
- 排他モードが変化しない場合があります。
  例えば、1つのアプリケーションで、ファイルAをオープン中に、ファイルBをI-Oオープン(Exclusive)した後、一旦クローズしてINPUTオープン(Share)しても、先に排他モードが強いExclusive(占有モード)でロックされているため、ファイルBは、Exclusive(占有モード)のままとなります。(「参考例3」を参照してください。)
- テーブルロックを指定したファイルのオープン中に、テーブルロックを指定しないファイルを操作する場合、テーブルロックの強制的なトランザクションに従って操作されます。
  (「参考例4」を参照してください。)

参考として、1つのアプリケーションが1つまたは2つのファイルを使用した入出力文操作順序の組合せによるトランザクション区間の遷移を示します。

トランザクション区間は以下の表記です。

図4.7 参考例1)1ファイルの単純な入出力操作の場合

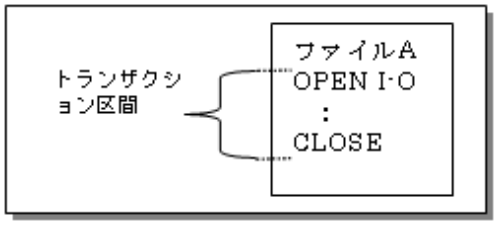
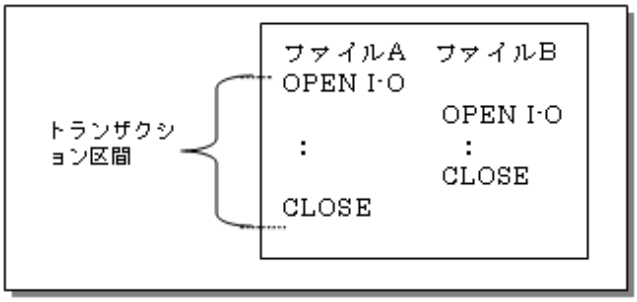
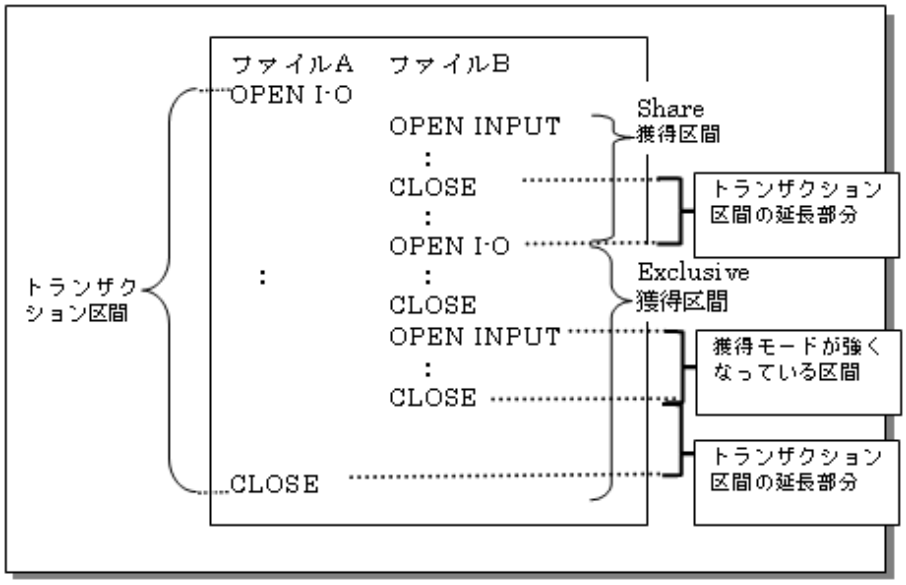


図4.8 参考例2)トランザクション区間がファイルAの入出力操作に依存する場合



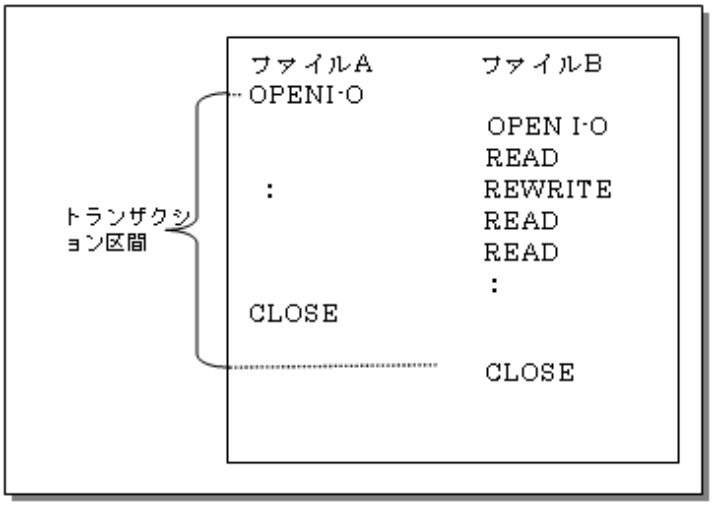
ファイルAのオープン中にファイルBがオープンされるため、テーブルロックはテーブルロック対象の全てのファイルがクローズするまで区間が延長されます。

図4.9 参考例3)排他モードが変化しない場合



1つのアプリケーションで、ファイルAをオープン中に、ファイルBをI-Oオープン(Exclusive)した後、INPUTオープン(Share)しても、先に排他モードが強いExclusive(占有モード)でロックされているため、ファイルBは、Exclusive(占有モード)のままとなります。

図4.10 参考例4)テーブルロック対象のファイルAと、テーブルロック対象のファイルBが混在する場合



ファイルAのオープン中にファイルBがオープンされるため、テーブルロックはテーブルロック対象の全てのファイルがクローズするまでトランザクション区間が延長されます。

4.2.2.4 デッドロック状態について

テーブルロックまたはレコードロックを使用する場合、処理順序によっては、他のセッションや他のアプリケーションとたすき掛けとなり、デッドロック状態となることがあります。

- テーブルロックの例

  以下の処理順序でデッドロックが発生します。

  1. 先行アプリケーションがAテーブルをテーブルロックし、Bテーブルをテーブルロックしません。
  2. 後行アプリケーションがAテーブルをテーブルロックしないで、Bテーブルをテーブルロックします。
  3. 先行アプリケーションがBテーブルに対して、以下のいずれかの操作を行います。
     WRITE/REWRITE/DELETE/OUTPUT OPEN
  4. 後行アプリケーションがAテーブルに対して、以下のいずれかの操作を行います。
     WRITE/REWRITE/DELETE/OUTPUT OPEN

- レコードロックの例

  複数プロセスで、同じテーブルのレコードをREADする場合、トランザクションが継続している間は、I-Oオープンで READした行はロックされたままとなっているため、別のレコードをREADすると、デッドロックが発生することがあります。

- マルチセッションプログラムの例

  マルチセッションプログラムの場合、1つのプログラムでも異なるセッションでアクセスするため、テーブルロックやレコードロックによるデッドロックが発生することがあります。

デッドロック出口

デッドロック状態は、NetCOBOLのデッドロック出口の記述を行うことで、例外処理を作成できます。

以下のエラーが発生した場合をデッドロックと判断し、デッドロック出口へ分岐する契機となります。

- デッドロックと判断するエラー
  - FILE STATUS = 92
  - iserrno = 107

・ デッドロック出口の記述例

```
*******************************************************
*     デッドロック出口（異常）処理
*******************************************************
 DECLARATIVES.
 DEAD-LOCK-PROCEDURE SECTION.
     USE FOR DEAD-LOCK.
 DEAD-LOCK-START.
      :
     GO TO ～.
 END DECLARATIVES.
```

なお、PowerRDBconnectorのデッドロック検出によるデッドロック出口への分岐は、NetCOBOL V10.3以降サポートしています。

デッドロック出口の詳細は、NetCOBOLのマニュアルを参照してください。

注意

デッドロックの注意事項について

・ デッドロックを少なくするために

デッドロックの発生頻度を少なくするため、以下の対処を検討してください。

― 1つのアプリケーション内で、獲得するレコードロックをできるだけ少なくする。

― 同時動作するアプリケーションでは、獲得するレコードは、できるだけ同じ順番で獲得する。

― 獲得したレコードは、できるだけ早く解放する。

・ デッドロック発生後の処理について

デッドロックが発生し、すぐに同じ処理を繰り返した場合、再びデッドロックとなることがあります。このため、デッドロック発生後、以下のいずれかの対処を検討してください。

― デッドロックとなったアプリケーションを終了させ、競合したレコードを獲得しているアプリケーションの状況を確認した後、再実行させます。

― デッドロックとなったアプリケーションで、レコードを解放した後、ある程度の時間を待ち合せた後、再実行させます。

・ トランザクションサブルーチンの使用について

― エラーを検出したトランザクション内の更新は、SQL Serverで自動的にロールバックされ、トランザクション内で獲得していたレコードロックは自動的に解放されます。しかしトランザクション開始サブルーチンが実行された状態であるため、アプリケーションの処理を継続させる場合、トランザクション取消しサブルーチンを実行してから処理を継続させてください。

4.2.3 COBOLアプリケーションに関する注意事項

ここでは、COBOLアプリケーションに関する注意事項を説明します。

4.2.3.1 キーに重複した値がある索引ファイルについて

索引ファイルのキーに重複した値がある場合は、以下の注意が必要です。

・ キーに重複した値があるレコードの並び順は格納順になりません。

データベースの制御上の順番となるためです。

- キーを指定した更新や削除は行わないでください。

  キーを指定して更新または削除を行うと、データベースから検索された同一キーのレコードのうち、最初のレコードが更新または削除されます。この場合、対象のレコードは、データベースの並び順に依存し、利用者からは特定できないためです。

- キーに重複した値があるレコードに対し、操作中に読取り方向を変更しないでください。

  読取り方向を変更すると、データベースへ再度検索を行います。このとき、重複したキーのレコードがデータベースの並び順で検索されます。このため、読取り方向変更前に読み取った順序の逆順に読み取れない場合があります。

4.2.3.2 ビューの使用について

ビューを使用する場合、以下の注意が必要です。

- 結合ビューおよびUNION結合ビューは、更新できません。
- OUTPUTモードでオープンできません。ただし、COBOL初期化ファイルのTruncateプロパティを使用することで可能となります。
- 関数、副問合せなどが定義されたビューには、アクセスできません。
- ビューを構成するテーブルの列属性を変更した場合、ビューを1度削除して再作成してください。

4.2.3.3 レコードの格納位置について

WRITE文でレコードを追加した場合、レコードはテーブルの最後に格納されるとは限りません。データベースの仕様に依存します。順編成ファイルに対して、書込み順どおりに読み込まれない場合があります。

キー指定なしの順アクセス実行時、書込み順どおりに読み込みたい場合、COBOLと関連付けのない項目にキーを指定してください。詳細は、「3.1.3.6 COBOLと関連付けのない項目の指定方法」を参照してください。

4.2.3.4 文字コード変換について

データベースに格納される文字コードは、データベースのテーブル定義または設定で異なります。特殊文字、ユーザー定義文字などの文字セットに注意してください。

- PowerRDBconnectorは、Interstage Charset Managerの文字コード変換プログラムは使用しません。
- unicodeとシフトJISの文字セット間のコード変換は、PowerRDBconnectorでは行っていません。

4.2.3.5 OUTPUTモードのオープンの使用について

OUTPUTモードでテーブルまたはビューをオープンする場合、以下の注意事項があります。

- OUTPUTモードでOPEN文を実行するCOBOLアプリケーションを起動するユーザーについては、データベースロールに“db_ddladmin”を設定してください。
- OUTPUTモードでオープンしたファイルに対して、2重にオープンすることはできません。
- OUTPUTモードでオープン直後から最初の入出力文までは、テーブルロックを使用した状態になります。
- COBOL初期化ファイルのTruncateプロパティを使用することで、ビューをOUTPUTモードでオープンすることができます。
- テーブルロック対象のテーブルをOUTPUTモードでオープンした場合、トランザクションの適用有無にかかわらず、ファイルをクローズせずにCOBOLアプリケーションを終了するとデータは初期化され、COBOLアプリケーション終了直前までにWRITE文が成功したレコードを格納します。
- レコードロック対象のテーブルをOUTPUTモードでオープンした後にトランザクションを開始後、1度もトランザクションを確定せずに、トランザクションを取消しまたはCOBOLアプリケーションを終了した場合、テーブルのデータは初期化されず、オープン前のデータに戻ります。

4.2.4 マルチスレッド使用時の注意事項

ここでは、マルチセッションプログラミングを使用したマルチスレッドプログラムの開発についての注意事項を説明します。

4.2.4.1 マルチスレッドプログラムについて

マルチスレッドプログラムについての主な注意点を以下に示します。

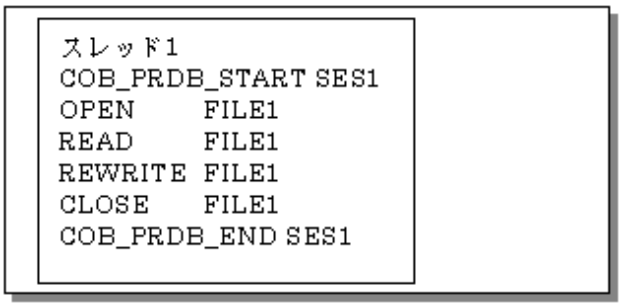
- マルチスレッドプログラムは、多重動作時のプログラムのデバッグが、プロセスプログラムより難しくなります。
- プロセスプログラムは、エラーが発生したプロセスだけが影響を受け、他のプロセスには影響がおよびません。しかしマルチスレッドプログラムは、1つのスレッドでエラーが発生すると、同じプロセス内の他のスレッドにも影響がおよぶことがあります。
  例えば、マルチスレッドプログラムが多重動作している場合、1つのスレッドで、プログラムを中断してしまうと、同じプロセスの他のスレッドも中断されてしまいます。
  このため、特にエラー処理を注意して開発してください。
- マルチスレッドプログラムでは、共有資源(静的な変数、カレントパスなど)を各スレッドが操作する場合は、共有資源に矛盾が生じないように、スレッドをロックする必要があります。
- マルチスレッドプログラミングの場合、スレッドのスタック域がアプリケーションで規定された値を超えないように考慮する必要があります。PowerRDBconnectorは、スタック域として64KBを使用します。

4.2.4.2 スレッドとセッションの関係について

マルチセッションプログラミングを用いて、マルチスレッド環境で動作させた場合の、スレッドとセッションの関係について以下に示します。なお、例では、次の記号を使用しています。

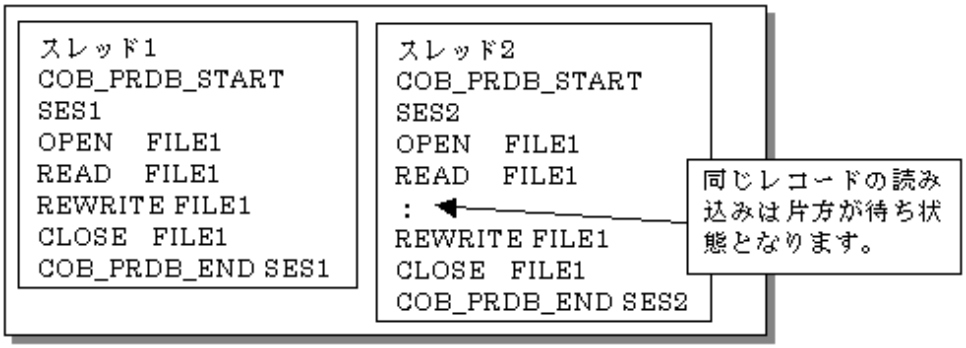
- COB_PRDB_START SES1:COB_PRDB_STARTサブルーチンを使用し、SES1というセッションを開設します。
- COB_PRDB_END SES1:COB_PRDB_ENDサブルーチンを使用し、SES1というセッションを閉設します。
- COB_PRDB_CHG SES1:COB_PRDB_CHGサブルーチンを使用し、SES1というセッションを指定します。

図4.11 例1 1つのプログラムをシングルスレッドで動作させる場合



コンソールアプリケーションや、シングルセッションプログラムを単純にマルチセッションプログラムに変更した場合、上図のようになります。

図4.12 例2 同じプログラムを複数のスレッドで動作させる場合



例1のプログラムが複数のスレッドで動作した場合、上図のようになります。

マルチスレッドプログラミングで、スレッド間で同じCOBOL言語のファイル結合子を共有する方法は、翻訳オプションにSHREXTを指定した上で以下のいずれかの指定をしてください。

- スレッド間共有外部ファイル(EXTERNAL句)
- 静的に定義したファイル(STATIC)
- オブジェクト内に定義したファイル(OBJECT)

以下に、スレッド間共有外部ファイル(EXTERNAL句)を使った例を示します。その他の場合は、「NetCOBOLのマニュアル」を参照してください。

例3 スレッド間でファイルを共有する場合

OPEN文、CLOSE文およびアクセス文が異なるプログラムの場合

図4.13 アクセスするスレッドが1つの場合

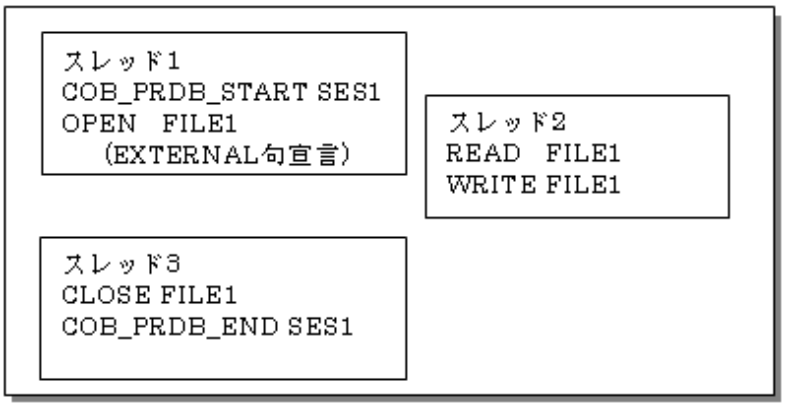
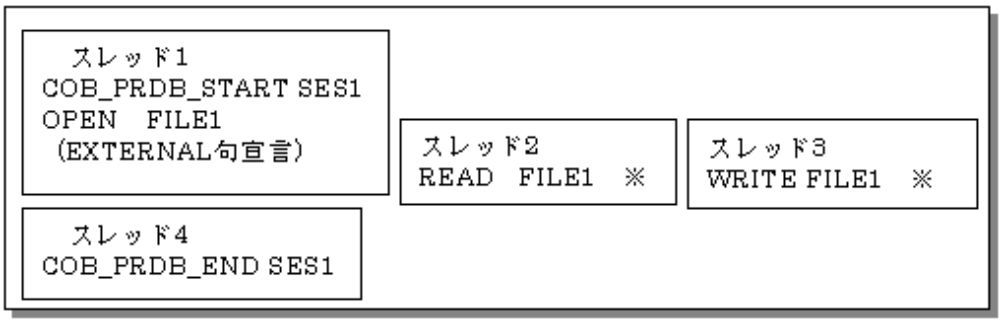


図4.14 アクセスするスレッドが複数の場合



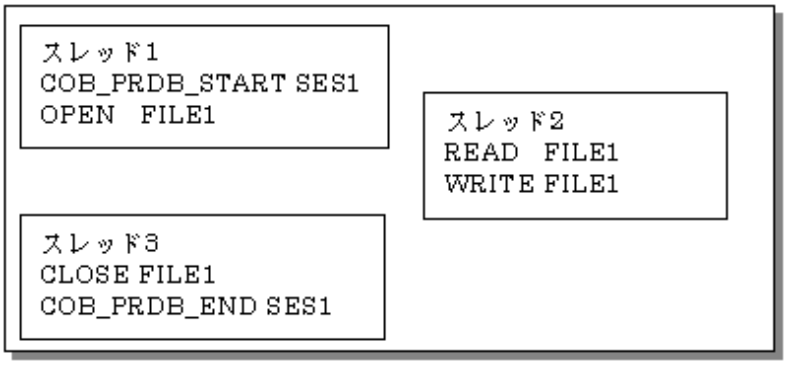
※)この場合、スレッド2のREAD文とスレッド3のWRITE文は同時に実行されず、先に実行した文の終了を待って次の文が実行されます。

例4 スレッド間でファイルを共有する場合(使用できない例)

1つのセッションを使用し、OPEN文、CLOSE文およびアクセス文が異なるプログラムの場合

EXTERNAL句の宣言がないため、同じファイルに対してスレッドをまたがってアクセスできません。

図4.15 スレッド間でファイルを共有する場合(使用できない例)



例5 セッションの開設/閉設と、アクセスするプログラムが異なる場合

負荷の高いデータベースとのコネクション回数を削減させるため、実際にアクセスする部分ではデータベースとのコネクションを切断することがないような場合の例を示します。

図4.16 セッションが1つの場合

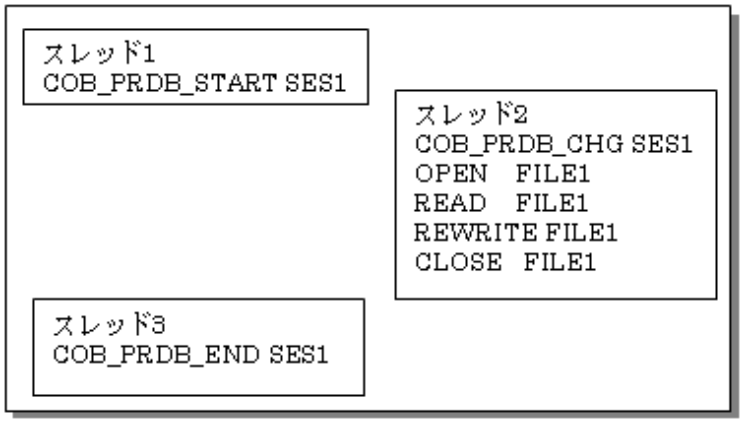
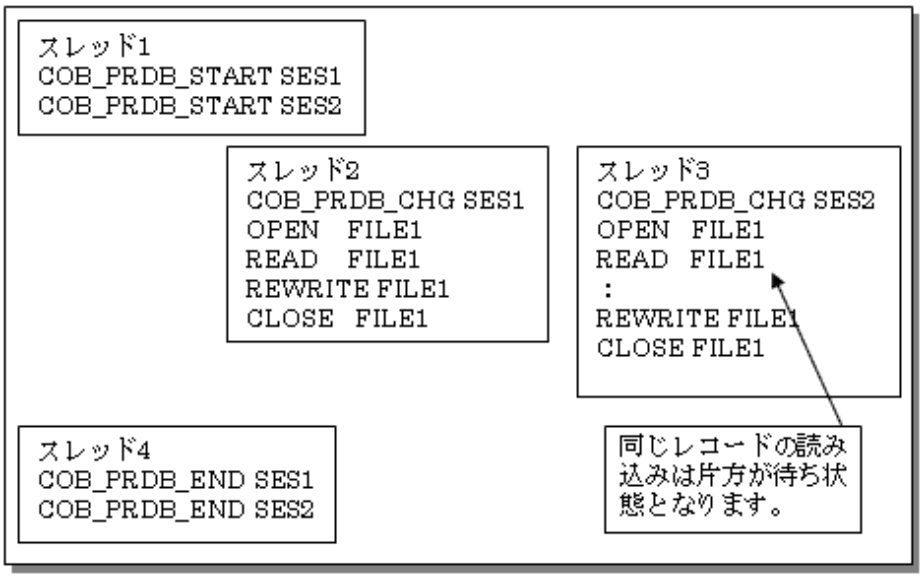


図4.17 セッションが複数の場合



4.2.4.3 セッションとトランザクションの関係について

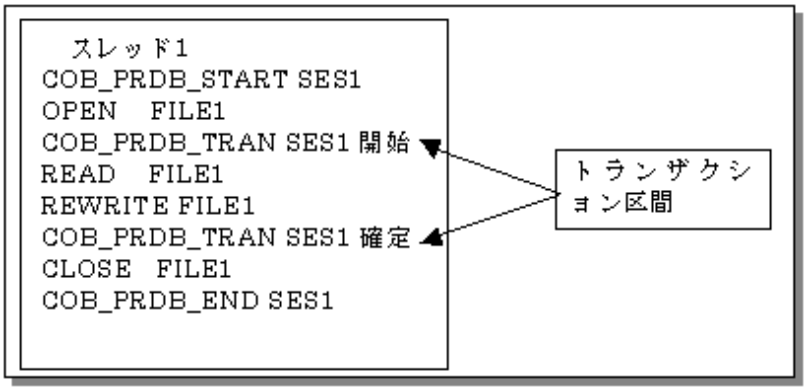
マルチセッションプログラミングを用いて、マルチスレッド環境で動作した場合の、スレッドとセッションおよびトランザクションの関係について以下に示します。なお、例では、次の記号を使用しています。

- COB_PRDB_START SES1：COB_PRDB_STARTサブルーチンを使用し、SES1というセッションを開設します。
- COB_PRDB_END SES1：COB_PRDB_ENDサブルーチンを使用し、SES1というセッションを閉設します。
- COB_PRDB_TRAN SES1：COB_PRDB_TRANサブルーチンを使用し、SES1というセッションを指定します。
- COB_PRDB_CHG SES1：COB_PRDB_CHGサブルーチンを使用し、SES1というセッションを指定します。

例1 トランザクション区間が1つのスレッドに閉じる場合

単純なCOBOLプログラムの場合、データベースにアクセスする部分は1つのスレッドで行われます。この場合の例を示します。

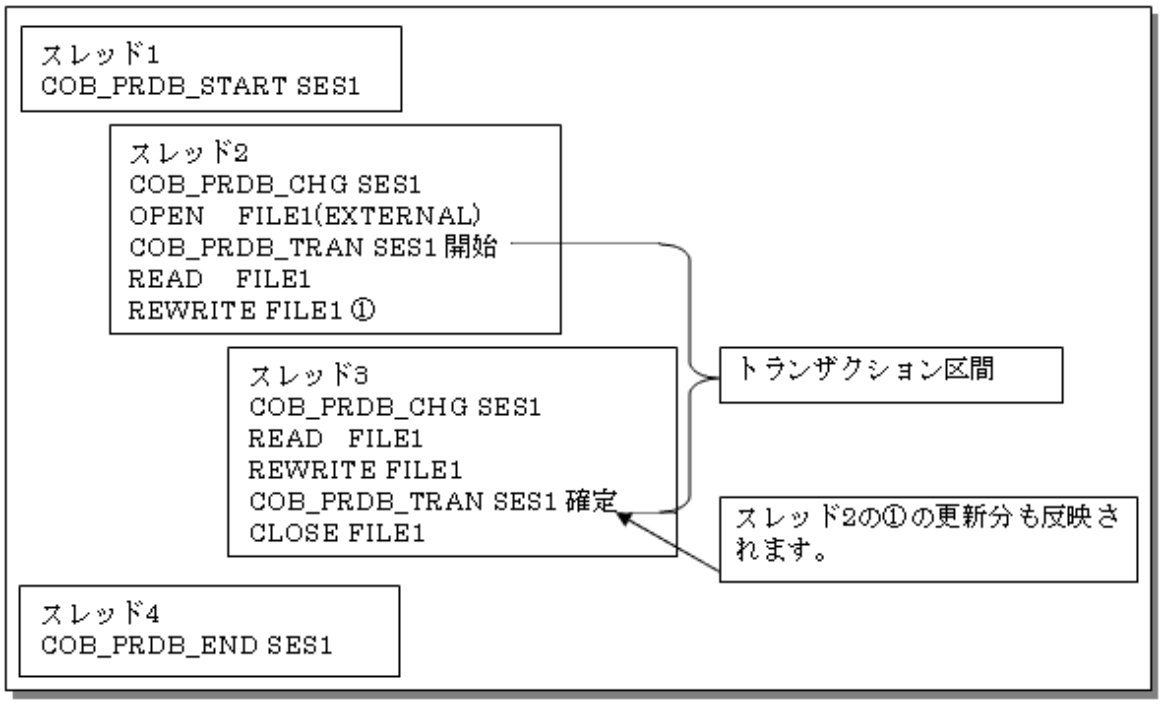
図4.18 トランザクション区間が1つのスレッドに閉じる場合



例2 トランザクション区間が複数のスレッドにまたがる場合

トランザクションは、セッション単位で制御されるため、複数のスレッドにまたがることがあります。この場合の例を示します。

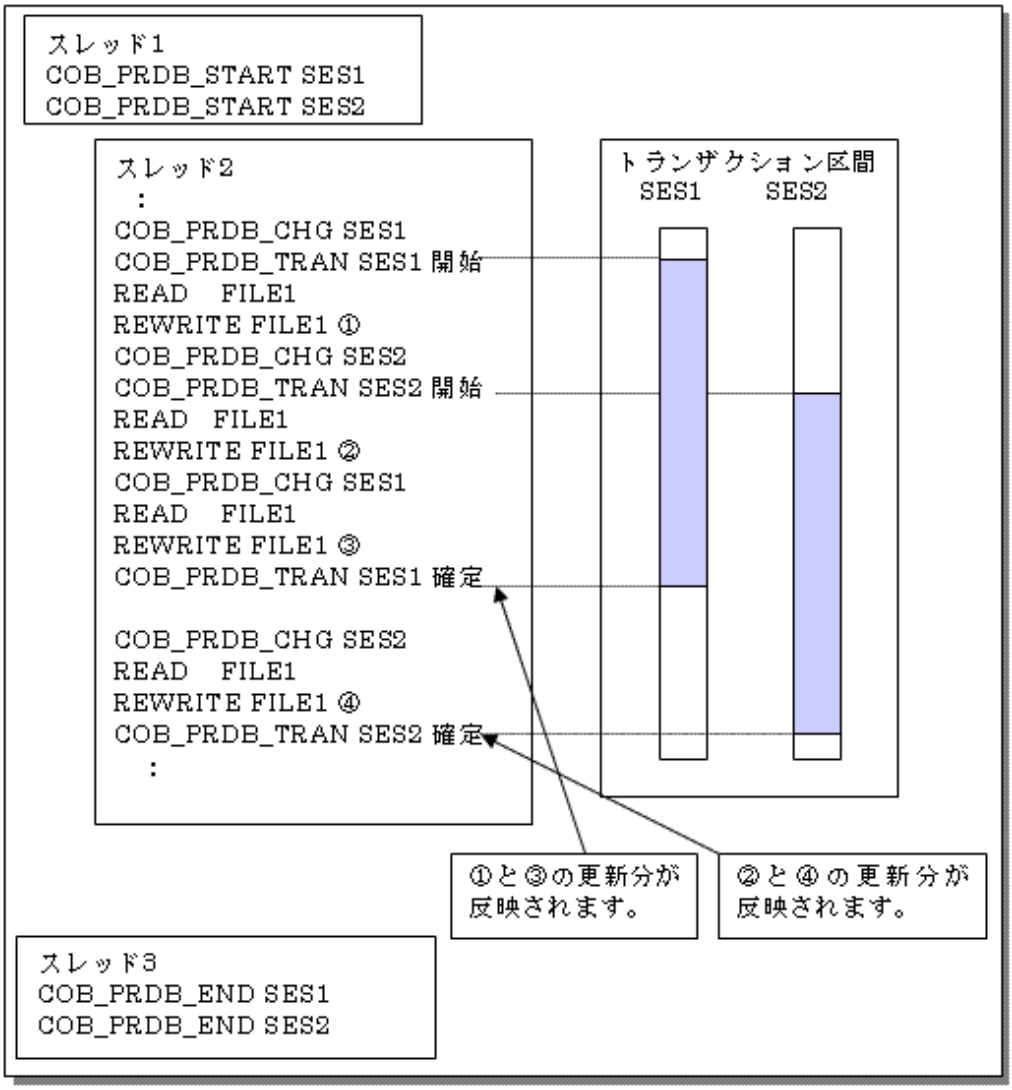
図4.19 トランザクション区間が複数のスレッドにまたがる場合



例3 複数のセッションを使用したトランザクション区間の場合

トランザクションは、セッション単位で制御されるため、1つのプログラムで複数のセッションを使用することができます。この場合の例を示します。

図4.20 複数のセッションを使用したトランザクション区間の場合



4.2.5 リモートのデータベースアクセスに関する注意事項

ここでは、リモートのデータベースに対してCOBOLアプリケーションからアクセスする場合についての注意事項を説明します。

4.2.5.1 大量のレコードアクセスについて

リモートデータベースにCOBOLアプリケーションから、アクセスする場合、レコード1件毎にネットワークを通じたアクセスが発生します。このため大量のレコードアクセスを行うと、以下の影響が発生します。

- ネットワーク負荷の増加
- レコードアクセス性能の劣化

上記のため、大量のレコードアクセスを行うバッチ型のアプリケーションは、データベースを格納したコンピュータ上で実行してください。

4.2.5.2 セキュリティについて

リモートデータベースにアクセスする場合、データ内容がネットワーク上で通信されます。このため、ネットワーク上のセキュリティの確保が必要です。

データベース認証を行う場合は、COBOLアプリケーション内で、ユーザー名、パスワードの管理についてのセキュリティを確保してください。

例）

- 認証情報登録サブルーチンに使用する、ユーザー名、パスワードは、COBOLアプリケーション実行時に、ユーザーから入力してもらう。
- 認証情報登録サブルーチンの使用後は、ユーザー名、パスワードの領域を、空白データなどで初期化する。

ネットワーク上は、データベース、Windowsやその他の製品が備えているセキュリティ機能を使用して、セキュリティの確保を行ってください。

第5章 エラー時の対処

本章では、PowerRDBconnector使用時に発生したエラーの原因と対処、およびトレース機能について説明します。エラーが発生したときにお読みください。

5.1 エラー時の対処

本節では、エラー発生時の対処について説明します。

エラー発生時は、エラー情報をアプリケーションログに出力します。アプリケーションログは、Windowsのイベントビューアで参照できます。

ファイルアクセスでエラーが発生した場合は、COBOLアプリケーションにFILE STATUSを通知します。なお、FILE STATUSの入出力状態値がエラーでもPowerRDBconnector のエラー情報が出力されていない場合は、NetCOBOLでエラーを検出した可能性があります。その場合は、NetCOBOLのマニュアルを参照してください。

以下の手順を参考に対処してください。

COBOLアプリケーションのアクセスでエラー発生	アプリケーションの状態を確認します。 FILE STATUSの入出力状態値を確認します。 使用しているデータベース、テーブル、ビューを特定します。
↓	
アプリケーションログの確認	アプリケーションログを確認します。 エラー情報(エラーコードとエラー詳細コード)から原因を調査します。「5.1.1.3 ファイルアクセス時のエラーコード一覧」、「5.1.2.2 トランザクションアクセス時のエラーコード一覧」、「5.1.3.2 認証情報登録サブルーチンのエラーコード一覧」、および「5.1.4.2 セッションサブルーチンのエラーコード一覧」を参考に調査してください。 エラー情報にメッセージが含まれる場合は、データベースの実行環境を調査してください。
↓	
データベースの確認	データベースの実行環境を確認します。 アプリケーションログに出力されているメッセージを参考に、使用しているファイル(テーブル、ビューおよびインデックス)が使用可能な状態か、SQL Serverの保守機能(SQLプロファイラ等)を使用して動作状況を調査します。SQL ServerのBooks Onlineを参考に対処してください。
↓	
エラーログ、トレース情報の採取と調査	エラーが再現する場合、エラーログおよびトレース情報を採取します。 トレース情報を調査し、COBOLアプリケーションを調査してください。トレース情報については、「5.2 トレース出力機能」を参照してください。 エラーログは、内部情報です。当社技術員からの依頼により採取してください。

5.1.1 ファイルアクセス時のエラー情報

5.1.1.1 ファイルアクセス時のイベントログ情報

ファイルアクセス時にエラーが発生すると、以下の形式でエラー情報が出力されます。

表5.1 ファイルアクセス時のイベントログ情報

<table>
<tr><td>出力先</td><td colspan="2">アプリケーションログ</td></tr>
<tr><td>ソース</td><td colspan="2">・ 32ビット動作時
FSP_RDBCONNECTOR_FJSVDBIO
・ 64ビット動作時
FSP_RDBCONNECTOR_FJSVDBIO64</td></tr>
<tr><td>出力形式</td><td colspan="2">File Access Error
PID = <process id> TID=<thread id>
function = <function name> iserrno = <error code>
isstat1 = <error code1> isstat2 = <error code2>
isstat3 = <error code3> isstat4 = <error code4>
TableName = <table name> ColName = <column name> Para = <parameter name>
ItemNo=<item no>
<database message></td></tr>
<tr><td></td><td><process id></td><td>プロセスID</td></tr>
<tr><td></td><td><thread id></td><td>スレッドID</td></tr>
<tr><td></td><td><function name></td><td>エラーが発生した関数名</td></tr>
<tr><td></td><td><error code></td><td>エラーコード</td></tr>
<tr><td></td><td><error code1></td><td>エラー詳細コード1</td></tr>
<tr><td></td><td><error code2></td><td>エラー詳細コード2</td></tr>
<tr><td></td><td><error code3></td><td>エラー詳細コード3</td></tr>
<tr><td></td><td><error code4></td><td>エラー詳細コード4</td></tr>
<tr><td></td><td><table name></td><td>エラーが発生したテーブル名またはビュー名を出力します。※
PowerRDBconnector動作環境ファイルおよびCOBOL初期化ファイルが正しく設定されていない場合には通知されません。</td></tr>
<tr><td></td><td><column name></td><td>エラーが発生した列名を出力します。※</td></tr>
<tr><td></td><td><parameter name></td><td>エラーが発生したPowerRDBconnector動作環境ファイルのプロパティ名を出力します。※</td></tr>
<tr><td></td><td><item no></td><td>エラーが発生した項目番号を出力します。※
項目番号とは、COBOLの項目定義順に対応する番号です。</td></tr>
<tr><td></td><td><database message></td><td>SQL Serverから通知されたメッセージを出力します。
SQL Serverでエラーが検出された場合に出力します。SQL ServerのBooks Onlineを参照して調査してください。
SQL Serverでエラーが発生していない場合には、メッセージは出力されません。</td></tr>
</table>

出力例	File Access Error PID = 712 TID = 1544 function = iswrite iserrno = 22 isstat1 = '9' isstat2 = 0x35 isstat3 = '9' isstat4 = 0x35 TableName = TPFILE

※:エラー内容によっては、出力されない情報があります。

5.1.1.2 イベントログに出力されないエラー情報

以下のFILE STATUSが通知された場合は、アプリケーションログにエラー情報は出力されません。

表5.2 アプリケーションログに出力されないエラー情報

FILE STATUS	iserrno	意味
10	110	EOFが検出されました。
22	100	キー値に重複データが検出されました。
23	111	目的のレコードが見つかりません。

5.1.1.3 ファイルアクセス時のエラーコード一覧

FILE STATUS句を指定したアプリケーションについて、ファイルアクセス時のエラーコードを以下に示します。ただし、NetCOBOLでエラーを検出するコードについては、iserrno、isstatは通知されません。

表5.3 ファイルアクセスエラーコード一覧

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
10	110	'1'	0x30	'1'	0x30	・EOFが検出されました。	・対処は特にありません。 ※エラー情報はアプリケーションログに出力されません。
22	100	'2'	0x32	'2'	0x32	・キー値に重複データが検出されました。	・重複するキー値を扱う場合はテーブルのUNIQUE属性を外してください。
						・WRITE文またはREWRITE文の実行時に、同じキー値がテーブルに存在していました。	・WRITE文、REWRITE文で重複するキー値を指定しないでください。 ※エラー情報はアプリケーションログに出力されません。
23	111	'2'	0x33	'2'	0x33	・目的のレコードがありません。 ・指定されたキー値がテーブルに存在していません。	・入出力文COBOLアプリケーションのSTART文またはキー付きREAD文、REWRITE文、DELETE文に指定しているキー値が

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
							テーブルにあるか確認してください。 ※エラー情報はアプリケーションログに出力されません。
30	—	—	—	—	—	・メモリ領域獲得に失敗しました。	・メモリを増設してください。もしくは、仮想メモリ(スワップ)を増やしてください。
						・ファイルが多すぎます。 ・ハンドルに誤りがあります。	・アプリケーションを見直してください。
35	2	‘0’	0x00	‘0’	0x00	・OPEN文で指定されたテーブルが存在しません。 ・COBOL初期化ファイルのSchemaName、TableNameプロパティに存在しないスキーマ名、テーブル名が指定されています。	・COBOL初期化ファイルに指定したスキーマ名、テーブル名を確認し、適切なスキーマ名、テーブル名に修正してください。
						・他のプロセスまたは他のセッションでOUTPUTオープンされています。	・対象テーブルが他のプロセスまたは他のセッションでOUTPUTオープンされていないか確認してください。
37	—	—	—	—	—	・ファイル属性に一致しないアクセスがありました。	・当社技術員に連絡してください。
39	—	—	—	—	—	・ファイル属性に一致しないため、ファイルのオープンができません。 ・ファイル編成が一致しません。	・COBOL初期化ファイルで指定したファイル属性と、ファイル属性とが一致するようにしてください。
						・索引ファイルのキー項目に対する属性(オフセット、大きさ、WITH DUPLICATES等)が一致しません。	・索引ファイルのキーの属性が、ファイルと一致するようにしてください。
90	5	‘0’	0x00	‘0’	0x00	・RANDOMアクセスモードでキー指定がありません。 ・DYNAMICアクセスモードでキー指定がありません。	・COBOL初期化ファイルと、COBOLソース中のアクセスモードの指定が同じであるか確認してください。
						・その他の内部エラーが発生しました。	・当社技術員に連絡してください。
90	5	‘0’	0x00	‘1’	0x00	・未サポートのコード系が指定されました。	・COBOLの翻訳オプションを確認してください。
90	5	‘9’	0x02	‘9’	0x02	・入出力エラーが発生しました。	・当社技術員に連絡してください。
90	5	‘9’	0x03	‘9’	0x03	・入出力エラーが発生しました。	・当社技術員に連絡してください。

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
90	6	‘0’	0x00	‘0’	0x00	・ データベースのサービスが起動されていません。	・ データベースのサービスが起動されているか確認してください。
						・ データベースにアクセスするための権限がありません。	・ データベースにアクセスするための権限があるか確認してください。
						・ データベースにログイン情報が登録されていません。	・ データベースにWindows認証のログイン情報が登録されているか確認してください。
						・ PowerRDBconnector動作環境ファイルのServerNameプロパティで指定されたサーバが見つかりません。 ・ PowerRDBconnector動作環境ファイルのDataSourceNameプロパティで指定されたデータベースが見つかりません。	・ サーバ環境または動作環境の設定を確認してください。
90	9	‘0’	0x00	‘0’	0x00	・ 可変長レコード形式が指定されました。	・ 固定長レコード形式になっているか確認してください。
90	12	‘0’	0x00	‘0’	0x00	・ メモリが不足しています。	・ OS、データベースのメモリ設定環境を見直し、適切に対処してください。
90	13	‘0’	0x00	‘0’	0x00	・ データベースまたはスキーマ、テーブルに対するアクセス権限がありません。	・ データベースまたはスキーマ、テーブルの権限設定を確認してください。
						・ OUTPUTオープンするユーザーのデータベースロールにdb_ddladminが設定されていません。	・ ユーザーのデータベースロールを確認してください。
90	14	‘9’	0x00	‘9’	0x00	・ ファイルディスクリプタが正しくありません。	・ 当社技術員に連絡してください。
90	22	‘0’	0x00	‘0’	0x00	・ 1テーブル内の列名長の合計（最大合計列名長）が半角40，960文字または全角20，480文字を超えました。	・ テーブルおよびビューを作成する際、列名の長さの合計が、半角40，960文字または全角20，480文字以下で定義してください。
						・ サポートしていないデータベースへ接続しようとしました。 ・ インストール時に選択したデータベースと異なるデータベースに接続しようとしました。	・ 接続先データベースを確認してください。 ・ インストール時のデータベース種別を、環境変数"RDBCONNECTOR"(32ビット)、または"RDBCONNECTOR64"(64ビット)で確認してください。

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
90	22	‘9’	0x34	‘9’	0x34	・COBOLのデータ型とテーブルのデータ型で一致していないものがあります。 ・テーブルの列名に誤りがあります。 ・未サポートのデータ型の列が存在します。	・テーブル定義を確認し、「3.1.3.5 列定義の対応」を参照して適切なテーブル定義に修正してください。
						・列名の項目長が4桁で指定されていません。	・列名の項目長は4桁固定で指定してください。
90	22	‘9’	0x35	‘9’	0x35	・符号なしのデータ型で定義される列に符号付きのデータを書き込もうとしました。 ・未サポートのデータ型のデータを検出しました。 ・文字コード変換に失敗しました。	・テーブルの列属性およびテーブルに格納されているデータの確認をしてください。テーブルの列属性については、「3.1.3.5 列定義の対応」を参照してください。 ・項目属性に違反するデータの扱いについては、「3.7.2 項目属性に違反するデータのチェックと補正」を参照してください。
						・使用できないキー値が入力されました。	・キー値に指定するデータを確認してください。
						・データ長がCOBOLまたはテーブルの定義長を超えています。	・COBOLまたはテーブルの定義長を確認してください。
90	101	‘9’	0x00	‘9’	0x00	・ファイルディスクリプタが正しくありません。	・当社技術員に連絡してください。
90	101	‘9’	0x00	‘9’	0x32	・ファイルディスクリプタが正しくありません。	・当社技術員に連絡してください。
90	102	‘9’	0x00	‘1’	0x00	・COBOL初期化ファイルの読込みに失敗しました。	・環境設定が正しいか確認してください。
90	102	‘9’	0x00	‘2’	0x00	・COBOL初期化ファイルの設定値の文字列長が規定の長さを超えています。	・COBOL初期化ファイルの指定内容を見直して、適切に対処してください。COBOL初期化ファイルについては、「3.1.5 COBOL初期化ファイル」を参照してください。
90	102	‘9’	0x00	‘3’	0x00	・COBOL初期化ファイルのプロパティ名に誤りがあります。 ・COBOL初期化ファイルのAccessMode 、SchemaName、TableNameプロパティに値が指定されていません。	
90	102	‘9’	0x00	‘4’	0x00	・COBOL初期化ファイルのプロパティに指定した値に誤りがあります。 ・AccessModeが指定されていません。	

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
						・スキーマ名またはテーブル名が指定されていません。 ・COBOL初期化ファイルのTableNameプロパティに全角文字と半角文字が混在したテーブル名が指定されています。	
90	102	‘9’	0x00	‘5’	0x00	・COBOL初期化ファイルのアクセスモードとCOBOLのアクセスモードに不整合があります。	・COBOLのアクセスモードとCOBOL初期化ファイルのアクセスモードが正しいか確認してください。
90	103	‘9’	0x00	‘9’	0x00	・指定したキーの構成とアクセスするインデックスの構成が一致しません。 ・SQL Serverでサポートされているインデックスの構成列数の上限値(16)を超えています。 ・キーの構成にサポートしていないデータ型が含まれています。	・指定するキー構成およびテーブルのインデックス構成を見直し、適切なキーを設定してください。
90	104	‘9’	0x02	‘9’	0x02	・1プロセス(シングルスレッド)で可能な最大オープン数128を超えました。	・最大オープン数が128を超えないようにCOBOLアプリケーションの内容を確認してください。
90	116	‘9’	0x00	‘9’	0x00	・メモリが不足しています。	・OS、データベースのメモリ設定環境を見直し、適切に対処してください。
90	255	‘0’	0x00	‘0’	0x00	・製品が正しくインストールされていません。	・本製品が正しくインストールされているか確認してください。
						・その他のエラーが発生しました。	・当社技術員に連絡してください。
90	255	‘0’	0x00	‘1’	0x00	・ファイルDSNがWindowsのシステムフォルダから削除されました。 以下にファイルDSN名を示します。 － 32ビット時 F3BWSDSN.dsn － 64ビット時 F4ARSDSN.dsn	・Windowsのシステムフォルダにファイル DSNが存在するか確認してください。
						・PowerRDBconnector動作環境ファイル名が誤っています。	・PowerRDBconnector 動作環境ファイルの設定を確認してください。 PowerRDBconnector 動作

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
						・ PowerRDBconnector動作環境ファイルの読込みに失敗しました。	環境ファイルについては、「3.1.4 PowerRDBconnector動作環境ファイル」を参照してください。
90	255	‘0’	0x00	‘2’	0x00	・ PowerRDBconnector動作環境ファイルの設定値の文字列長が規定の長さを超えています。	
90	255	‘0’	0x00	‘3’	0x00	・ PowerRDBconnector動作環境ファイルのキーワードに誤りがあります。	
90	255	‘0’	0x00	‘4’	0x00	・ PowerRDBconnector動作環境ファイルで指定された値に誤りがあります。	
						・ COBOL初期化ファイルの設定内容に誤りがあります。 ・ Truncateで指定されたテーブルが見つかりません。	・ COBOL初期化ファイルの設定内容を確認してください。
						・ インストール時のデータベース種別に誤りがあります。	・ インストール時のデータベース種別を、環境変数"RDBCONNECTOR"（32ビット）、または"RDBCONNECTOR64"（64ビット）で確認し、誤っていた場合には、本製品をアンインストール後、正しいデータベース種別を指定してインストールしてください。
90	255	‘0’	0x00	‘5’	0x00	・ PowerRDBconnector動作環境ファイルが、シフトJISまたはUTF-8以外のコード系で作成されています。	・ PowerRDBconnector 動作環境ファイルは、シフトJISまたはUTF-8のコード系で作成してください。
90	255	‘0’	0x00	‘6’	0x00	・ データベースでエラーが発生しました。 ・ COBOLのデータ型とデータベースの列定義が異なっています。 ・ RECORD KEY句にあたる列にインデックスまたはプライマリーキーが定義されていません。 ・ ビューをOUTPUTモードでオープンしました	・ SQL Serverの動作環境を確認してください。 ・ RECORD KEY句にあたる列にインデックスまたはプライマリーキーを定義してください。 ・ ビューをOUTPUTモードでオープンしていないか確認してください。
						・ PowerRDBconnector動作環境ファイルのTimeOutプロパティに指定した時間を超えて、タイムアウトを検出しました。	・ レコードロックの獲得状態を確認してください。レコードロックについては、「3.5.2 レコードロック機能」を参照してください。

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
							・ SQL Serverの動作環境を確認してください。
90	255	‘0’	0x00	‘8’	0x00	・ セッション開設は既に実行されています。 ・ シングルセッションプログラミングの処理が動作中に、セッション開設が実行されました。 ・ マルチセッションのプログラムが動作中のプロセスで、シングルセッションのプログラムを動作させようとしました。	・ セッション開設およびセッション閉設のシーケンスを確認してください。
90	255	‘0’	0x00	‘9’	0x00	・ オープン中のファイルが残っている状態で、セッション閉設が実行されました。	・ ファイルを全てクローズ後にセッション閉設を行ってください。
90	255	‘0’	0x00	‘3’	0x11	・ トランザクションが開始されていません。	・ トランザクションサブルーチンの呼出し順序が正しいか見直してください。
90	255	‘0’	0x00	‘4’	0x0C	・ メモリが不足しています。	・ OS、データベースのメモリ設定が適切か見直してください。
90	255	‘0’	0x00	‘4’	0x06	・ トランザクションサブルーチン実行時にデータベースでエラーが発生しました。	・ SQL Serverのエラー情報を確認してください。
90	255	‘0’	0x00	‘1’	0x01	・ サブルーチン実行時のパラメーターに誤りがあります。	・ パラメーターが正しいか確認してください。
90	255	‘0’	0x00	‘1’	0x02	・ パラメーターで指定した文字列長が正しくありません。	・ 指定した文字列長が256バイト以内か確認してください。
92	11	‘9’	0x02	‘9’	0x02	・ OUTPUTモードでオープンしたファイルを、再度オープンしました。 ・ オープンしたファイルを、OUTPUTモードで再度オープンしました。	・ COBOLアプリケーションの内容を確認してください。
92	11	‘9’	0x03	‘9’	0x03	・ 他のプロセスまたは他のセッションがテーブルロックでオープンしています。	・ 他のプロセスまたは他のセッションがテーブルロックでオープンしていないか確認してください。
92	107	‘0’	0x00	‘0’	0x00	・ デッドロックが発生しました。	・ COBOLアプリケーションの内容を確認してください。
92	—	—	—	—	—	・ 既にオープンされているファイルに対してオープンが発行されました。 ・ ファイルがクローズされていません。	・ オープン、クローズの順番を調査してください。
93	—	—	—	—	—	・ ファイルが有効ではありません。	・ 当社技術員に連絡してください。

FILE STATUS	error code					意味	対処
	iserrno 10進数	isstat					
		1	2	3	4		
94	—	—	—	—	—	・ ファイル属性に一致しないため、ファイルのオープンができません。 ・ ファイル編成が一致しません。	・ COBOL初期化ファイルで指定したファイル属性と、ファイル属性とが一致するようにしてください。
						・ 索引ファイルのキー項目に対する属性(オフセット、大きさ、WITH DUPLICATES等)が一致しません。	・ 索引ファイルのキーの属性が、ファイルと一致するようにしてください。
						・ ファイルをオープンするためのメモリが不足しています。	・ メモリを増設してください。もしくは、仮想メモリ(スワップ)を増やしてください。
						・ 指定したファイル名に誤りがあります。	・ ファイル名に無効な文字が含まれていないか確認してください。

5.1.1.4 COBOLアプリケーション終了時のメッセージ一覧

FILE STATUS句を指定しないアプリケーションで、ファイルアクセス時にエラーが発生した場合など、COBOLアプリケーションが中断するレベルのエラーが発生した場合、COBOLランタイムシステムがメッセージを出力します。以下に代表的なメッセージを示します。なお、出力形式、対処など詳細については「NetCOBOLのマニュアル」の実行時エラーを参照してください。

表5.4 COBOLの実行時エラーメッセージの例

メッセージ	$3	エラー内容
JMP0310I-I/U '$2'ファイルで'$1'エラーが発生しました.'$3' $1:OPENまたはCLOSE $2:アクセス名またはファイル名	ACC-METHOD	ファイルのアクセス方法が誤っています。
	ERFLD=xxxx	入出力ファイルのアクセスエラーが発生しました。 xxxx:16進表記 xxxx:ファイルアクセス時のエラーコード PowerRDBconnector使用時は、iserrnoの値が設定されます。
	REC-MODE	レコード形式に誤りがあります。
	INV-KEYDUP	キーの重複可否(DUPLICATES)に誤りがあります。
	INV-KEYLEN	割り当てられたファイルのキー長がプログラムでの定義と矛盾します。
	INV-LRECL	割り当てられたファイルのレコード長がプログラムでの定義と矛盾します。
	INV-KEYRCS	割り当てられたファイルのキーのコード系がプログラムの動作コード系と矛盾します。
	KEY-ATTR	割り当てられたファイルのキー属性がプログラムでの定義と矛盾します。

メッセージ	$3	エラー内容
JMP0320I-I/U '$2'ファイルに対する'$1'文の実行で,入出力エラーが発生しました. $1:COBOL文 $2:ファイル名またはアクセス名	FDBK=xxxx	入出力ファイルのアクセスエラーが発生しました。 xxxx:16進表記 PowerRDBconnector使用時は、以下の値が設定されます。 ・ 先頭2桁:iserrno値の16進表現値 ・ 後方2桁:isstat2の値
	INV-LEN	WRITE/REWRITE 文でレコード長が正しくありません
JMP0321I-U '$1'ファイルに対するREAD文の実行で,ファイル終了条件が発生しました. $1:ファイル名またはアクセス名	ー	ファイルの終端(EOF)まで読み込みが終わりました。
JMP0323I-U '$2'ファイルに対する'$1'文の実行で,重複キーによる誤りが発生しました.$3 $1:COBOL文 $2:ファイル名またはアクセス名	エラーアドレス	キーの値が重複しています。
JMP0324I-U '$2'ファイルに対する'$1'文の実行で,求めるレコードが見つかりません.$3 $1:COBOL文 $2:ファイル名またはアクセス名	エラーアドレス	求めるレコードが見つかりません。
JMP0327I-U '$2'ファイルに対する'$1'文の実行で,無効キー条件が発生しました.'$3' $1:COBOL文 $2:ファイル名またはアクセス名	エラーアドレス	キーが無効です。
JMP0330I-I/U '$2'ファイルの'$1'文で,実行順序の誤りが発生しました.'$3' $1:COBOL文 $2:ファイル名またはアクセス名	AT-END	ファイル終了条件発生後、さらにREAD文が実行されました。
	DUPL-OPEN	既に開かれたファイルに対してOPEN文が実行されました。
	NO-READ	直前が成功したREAD文でありません。
	NOT-OPENED	開かれていないファイルに対して入出力文が実行されました。
	OPEN-MODE	OPENモードが正しくありません。
	POS-ERROR	ファイル位置指示子が不定です。

5.1.1.5 データベースのエラー発生時のイベントログ情報

SQL Serverでエラーが発生した場合は、イベントログの最終行に"[Microsoft]"という見出しで始まるSQL Serverのエラー情報が出力されます。

SQL Serverのエラー情報については、SQL Serverのマニュアル等を参照してください。

以下に出力例を示します。

・ 接続エラーの場合(1)

function = thr_isopen2 iserrno = 6

isstat1 = 0x0 isstat2 = 0x0 isstat3 = 0x0 isstat4 = 0x0

TableName = M_FILE02

[Microsoft][ODBC SQL Server Driver][SQL Server]このログインで要求されたデータベース "COBOL1" を開けません。ログインに失敗しました。

・ 接続エラーの場合(2)

function = thr_isbuild iserrno = 6

isstat1 = '0' isstat2 = 0x0 isstat3 = '0' isstat4 = 0x0

TableName = PF1

[Microsoft][SQL Native Client]TCP プロバイダ : タイムアウト エラー [258].

・ アクセス権限エラーの場合

function = iswrite iserrno = 13

isstat1 = 0x0 isstat2 = 0x0 isstat3 = 0x0 isstat4 = 0x0

TableName = M_FILE02

[Microsoft][ODBC SQL Server Driver][SQL Server]INSERT 権限がオブジェクト 'M_FILE02'、データベース 'COBOL'、スキーマ 'dbo' で拒否されました。

5.1.2 トランザクションアクセス時のエラー情報

5.1.2.1 トランザクションアクセス時のイベントログ情報

シングルセッションプログラミングでトランザクションサブルーチンなどのサブルーチンでエラーが発生した場合は、FILE STATUSを通知せず、COBOLアプリケーションが終了します。この場合には、トランザクションサブルーチンの呼出し順番などに問題がないか見直してください。
以下のエラー情報が出力されます。

トランザクションアクセス時のエラーは、以下の形式です。

表5.5 トランザクションアクセス時のイベントログ情報

出力先	アプリケーションログ
ソース	・ 32ビット動作時 FSP_RDBCONNECTOR_FJSVDBIO ・ 64ビット動作時 FSP_RDBCONNECTOR_FJSVDBIO64

<table>
<tr><td>出力形式</td><td colspan="2">
<pre>Transaction Access Error
PID = <process id> TID = <thread id>
command name = <command name>
error code = <error code>
error code1 = <error code1>
<database message></pre>
</td></tr>
<tr><td></td><td><process id></td><td>プロセスID</td></tr>
<tr><td></td><td><thread id></td><td>スレッドID</td></tr>
<tr><td></td><td><command name></td><td>エラーが発生したトランザクション機能名
以下に機能名とサブルーチンの対応を示します。
<table>
<tr><td rowspan="2">出力される機能名</td><td colspan="3">対応するサブルーチン名</td></tr>
<tr><td>シングルセッションプログラミング</td><td colspan="2">マルチセッションプログラミング</td></tr>
<tr><td></td><td></td><td>サブルーチン名</td><td>トランザクション種別</td></tr>
<tr><td>XMROTSTR</td><td>XMROTSTR
サブルーチン</td><td rowspan="3">COB_PRD_TRAN
サブルーチン</td><td>1</td></tr>
<tr><td>XMROTEND</td><td>XMROTEND
サブルーチン</td><td>2</td></tr>
<tr><td>XMROTCNL</td><td>XMROTCNL
サブルーチン</td><td>3</td></tr>
</table>
</td></tr>
<tr><td></td><td><error code></td><td>終了情報</td></tr>
<tr><td></td><td><error code1></td><td>詳細情報</td></tr>
<tr><td></td><td><database message></td><td>SQL Serverから通知されたメッセージを出力します。
SQL Serverでエラーが検出された場合に出力します。SQL ServerのBooks Onlineを参照して調査してください。
SQL Serverでエラーが発生していない場合には、メッセージは出力されません。</td></tr>
<tr><td>出力例</td><td colspan="2">
<pre>Transaction Access Error
PID = 1284 TID = 1532
command name = XMROTSTR
error code = 92
error code1 = 17430</pre>
</td></tr>
</table>

5.1.2.2 トランザクションアクセス時のエラーコード一覧

トランザクションアクセス時のエラーコードを以下に示します。

表5.6 トランザクションのエラーコード

Command			error code		意味	対処
STR	END	CNL	終了情報 10進数	詳細情報 10進数		
○	○	○	0	0	正常終了	なし
○	○	○	91	0	パラメーターエラー COB_PRDB_TRANサブルーチンに指定するトランザクション種別の指定に誤りがあります。	COB_PRDB_TRANサブルーチンの呼出し時のパラメーターを確認してください。

○	—	—	92	17430	シーケンスエラー トランザクション開始が既に実行されています。	トランザクション開始と、終了または取消しサブルーチンのシーケンスを確認してください。
—	○	○	93	17430	シーケンスエラー トランザクション開始が実行されていません。	トランザクションサブルーチンの呼出し順序が正しいか見直してください。
○	○	○	94	12	メモリが不足しています。	OS、データベースのメモリ設定が適切か見直してください。
○	○	○	94	6	データベースでエラーが発生しました。	イベントログに出力されたメッセージからSQL Serverの Books Onlineを参照して調査してください。

Command種別

STR：XMROTSTRサブルーチン/COB_PRDB_TRANサブルーチン(トランザクション種別=1)

END：XMROTENDサブルーチン/COB_PRDB_TRANサブルーチン(トランザクション種別=2)

CNL：XMROTCNLサブルーチン/COB_PRDB_TRANサブルーチン(トランザクション種別=3)

通知の有無

○：通知します。

－：通知しません。

5.1.3 認証情報登録サブルーチンのエラー情報

5.1.3.1 認証情報登録サブルーチンのイベントログ情報

認証情報登録サブルーチン実行時のエラーは、以下の形式です。

表5.7 認証情報登録サブルーチンのイベントログ情報

出力先	アプリケーションログ	
ソース	・ 32ビット動作時 FSP_RDBCONNECTOR_FJSVDBIO ・ 64ビット動作時 FSP_RDBCONNECTOR_FJSVDBIO64	
出力形式	Authentication Error PID = <process id> TID = <thread id> command name = <command name> error code = <error code> error code1 = <error code1>	
	<process id>	プロセスID
	<thread id>	スレッドID
	<command name>	認証情報登録機能名”XMROAUTH”
		マルチセッションプログラミング使用時でもシングルセッションプログラミングと同じ名前で出力されます。
	<error code>	終了情報

	<error code1>	詳細情報
出力例	Authentication Error PID = 1284 TID = 1532 command name = XMROAUTH error code = 91 error code1 = 0	

5.1.3.2 認証情報登録サブルーチンのエラーコード一覧

認証情報登録サブルーチンのエラーコードを以下に示します。

表5.8 認証情報登録サブルーチンのエラーコード

Command	error code		意味	対処
XMROAUTH COB_PRDB_AUTH	終了情報 10進数	詳細情報 10進数		
○	0	0	正常終了	なし
○	91	0	・ 認証種別に誤りがあります。 ・ ユーザー名長、パスワード長に誤りがあります。	・ パラメーターの内容が正しいか確認してください。

通知の有無

○:通知します。

ー:通知しません。

5.1.4 セッションサブルーチンのエラー情報

以下のエラー情報が出力されます。

5.1.4.1 セッションサブルーチンのイベントログ情報

セッションサブルーチン実行時のエラーは、以下の形式です。

表5.9 セッションサブルーチンのイベントログ情報

出力先	アプリケーションログ	
ソース	・ 32ビット動作時 FSP_RDBCONNECTOR_FJSVDBIO ・ 64ビット動作時 FSP_RDBCONNECTOR_FJSVDBIO64	
出力形式	File Access Error PID = <process id> TID=<thread id> function = <function name> iserrno = <error code> isstat1 = <error code1> isstat2 = <error code2> isstat3 = <error code3> isstat4 = <error code4> <error message>	
	<process id>	プロセスID

	<thread id>	スレッドID
	<function name>	機能名
		COB_PRDB_START サブルーチン: prdb_start COB_PRDB_END サブルーチン : prdb_end
	<error code>	エラーコード
	<error code1>	エラー詳細コード1
	<error code2>	エラー詳細コード2
	<error code3>	エラー詳細コード3
	<error code4>	エラー詳細コード4
	<error message>	詳細情報
出力例	File Access Error PID = 1148 TID = 1888 function = prdb_end iserrno = 255 isstat1 = '0' isstat2 = 0x0 isstat3 = '9' isstat4 = 0x0 OPEN FILE EXIST	

5.1.4.2 セッションサブルーチンのエラーコード一覧

セッションサブルーチン使用時のエラーコードを以下に示します。

表5.10 セッションサブルーチン使用時のエラーコード

Command		error code					意味	対処
STR	END	iserrno 10進数	isstat					
			1	2	3	4		
○	○	255	'0'	0x00	'8'	0x00	・セッション開設およびセッション閉設のシーケンスエラーです。 ・セッション開設は既に実行されています。 ・シングルセッションプログラミングの処理が動作中に、セッション開設が実行されました。	・セッション開設およびセッション閉設のシーケンスを確認してください。
							・1つのセッションIDに対して、複数スレッドから同時にアクセスされました。	・1つのセッションIDに対して、複数スレッドから同時にアクセスされていないか確認してください。
—	○	255	'0'	0x00	'9'	0x00	・オープン中のファイルが残っている状態で、セッション閉設が実行されました。	・ファイルを全てクローズ後にセッション閉設を行ってください。
○	○	255	'0'	0x00	'99'	0x00	・内部矛盾が発生しました。	・当社技術員に連絡してください。

Command種別

STR：COB_PRDB_STARTサブルーチン

END:COB_PRDB_ENDサブルーチン

通知の有無

○:通知します。

ー:通知しません。

5.2 トレース出力機能

本節では、PowerRDBconnectorのトレース機能について説明します。

COBOLアプリケーションのトラブル調査をする際に、PowerRDBconnectorを用いたデータベースへのアクセス履歴をトレースファイル(FJSVdbio_trc.log)に採取し、調査できます。

5.2.1 トレース機能の使用方法

PowerRDBconnector動作環境ファイルの以下のプロパティを指定してください。

- ErrorLogプロパティにトレース情報の出力先ディレクトリを指定してください。相対指定も可能です。
- TraceModeプロパティにONを指定してください。
- 必要に応じて、TraceSizeプロパティに、トレース情報の最大サイズを指定してください。
- 必要に応じて、TraceLevelプロパティに、トレース情報のレベルを指定してください。

例)トレース機能を使用する際の記述例

PowerRDBconnector動作環境ファイル(DBIO_ENV) の記述例

```
; PowerRDBconnector動作環境ファイル
ServerName=snake.domain.com
DataSourceName=pubs
   ～
ErrorLog=D:¥log
TraceMode=ON
TraceSize=4096
TraceLevel=0
```

トレース機能の使用時の注意事項を以下に示します。

- トレース機能を使用すると、性能が著しく低下する場合があります。通常の運用では設定しないでください。
- COBOLアプリケーションが動作中に、PowerRDBconnector動作環境ファイルのトレース関連のプロパティを変更しても、ただちに有効とはなりません。有効となるのは、次にCOBOLアプリケーションを実行したときからです。

PowerRDBconnector動作環境ファイルの各プロパティの詳細については、「3.1.4 PowerRDBconnector 動作環境ファイル」を参照してください。

5.2.2 トレースの種類

PowerRDBconnectorのトレース情報に出力する情報の種類を、PowerRDBconnector動作環境ファイルにTraceLevelプロパティで指定することができます。以下に指定する値と出力するトレース情報の種類を示します。

表5.11 トレースの種類

TraceLevel プロパティ 指定値	出力情報				用途例
	アクセス情報	列情報、イベントログ内容	エラー時のデータ内容	キー値、入出力データの内容	
0	○	×	×	×	アクセス履歴や、エラー前後のアクセス状況を調査する場合
1	○	○	×	×	オープン時に項目不一致などのエラー原因を調査する場合
2	○	○	○	×	エラー発生時のキー値の内容を調査する場合
3	○	○	○	○	アクセスしたデータ内容を詳細に調査する場合

○:出力します。

×:出力しません。

トレースの種類についての注意事項を以下に示します。

- TraceLevelプロパティで指定する数値が大きくなればなるほど出力する情報量が多くなり、性能が遅くなります。このため用途に応じて最小の値を指定してください。

5.2.3 トレース情報の内容

PowerRDBconnectorで出力するトレースファイル(FJSVdbio_trc.log)の内容を以下に示します。

5.2.3.1 トレース情報の形式

トレースファイル(FJSVdbio_trc.log)の形式を以下に示します。

<< TRACE DATETIME=YYYY/MM/DD-HH:MM:SS.SSS PPPPPPP >> 1

<< Windows:メジャーVer/マイナーVer Windows製品名 >> 2

<< PowerRDBconnectorのモジュール情報 >> 3

<< TraceLevel=トレースレベル, LOG_FILE=パス名 >> 4

5	6	7	8	9	10	11	12	13	14

5～14の項目は、個々をセミコロン(;)で区切られています。

表5.12 トレース情報の説明

No	TraceLevel				出力項目	内容
	0	1	2	3		
1	○	○	○	○	採取日時	トレース採取の日時と、パフォーマンスカウンタを出力します。
2	○	○	○	○	VL	Windowsのバージョンレベルを出力します。
3	○	○	○	○	モジュール情報	PowerRDBconnectorのモジュール情報を出力します。

<table>
<tr><th rowspan="2">No</th><th colspan="4">TraceLevel</th><th rowspan="2">出力項目</th><th rowspan="2">内容</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>モジュール名
バージョン情報
モジュールの更新日時</td></tr>
<tr><td>4</td><td>○</td><td>○</td><td>○</td><td>○</td><td>トレース情報</td><td>トレース情報として、TraceLevelとトレース出力先のパス名を出力します。</td></tr>
<tr><td>5</td><td>○</td><td>○</td><td>○</td><td>○</td><td>日時</td><td>トレースの出力日時（YYYY/MM/DD-HH:MM:SS.SSS）</td></tr>
<tr><td>6</td><td>○</td><td>○</td><td>○</td><td>○</td><td>パフォーマンスカウンタ</td><td>トレース出力時のパフォーマンスカウンタ</td></tr>
<tr><td>7</td><td>○</td><td>○</td><td>○</td><td>○</td><td>プロセスID</td><td>対象のプロセスID</td></tr>
<tr><td>8</td><td>○</td><td>○</td><td>○</td><td>○</td><td>スレッドID</td><td>対象のスレッドID</td></tr>
<tr><td>9</td><td>○</td><td>○</td><td>○</td><td>○</td><td>実行単位ID</td><td>セッション単位の識別子</td></tr>
<tr><td>10</td><td>○</td><td>○</td><td>○</td><td>○</td><td>COBOL命令</td><td>COBOL命令を次の形式で出力します。
マルチセッションプログラミングの場合
<table><tr><th>COBOL命令</th><th>出力形式</th></tr><tr><td>OPEN文</td><td>OPEN</td></tr><tr><td>OUTPUTモードのOPEN文</td><td>OPEN(OUTPUT)</td></tr><tr><td>CLOSE文</td><td>CLOSE</td></tr><tr><td>START文</td><td>START</td></tr><tr><td>順READ文</td><td>READ(SEQ)</td></tr><tr><td>乱READ文</td><td>READ(RAN)</td></tr><tr><td>WRITE文</td><td>WRITE</td></tr><tr><td>順REWRITE文</td><td>REWRITE(SEQ)</td></tr><tr><td>乱REWRITE文</td><td>REWRITE(RAN)</td></tr><tr><td>順DELETE文</td><td>DELETE(SEQ)</td></tr><tr><td>乱DELELE文</td><td>DELETE(RAN)</td></tr><tr><td>COBOL命令以外</td><td>----------（ハイフン）</td></tr></table>シングルセッションプログラミングの場合
<table><tr><th>COBOL命令</th><th>出力形式</th></tr><tr><td>全て</td><td>----------（ハイフン）</td></tr></table></td></tr>
<tr><td>11</td><td>○</td><td>○</td><td>○</td><td>○</td><td>PowerRDBconnector情報</td><td>PowerRDBconnectorの製品保守用の情報を出力します。</td></tr>
<tr><td>12</td><td>○</td><td>○</td><td>○</td><td>○</td><td>内部関数名</td><td>COBOL命令に対応した内部関数名が出力されます。詳細は「表5.14 プログラミングスタイルによる関数対応表」を参照してください。</td></tr>
<tr><td>13</td><td>○</td><td>○</td><td>○</td><td>○</td><td>ロック状態</td><td>ロックの状態を次の文字列で出力します。
<table><tr><th>ロックの状態</th><th>出力形式</th></tr><tr><td>テーブルロック状態</td><td>TBLLOCK</td></tr><tr><td>レコードロック状態</td><td>RECLOCK</td></tr><tr><td>全て</td><td>----------（ハイフン）</td></tr></table>上記の後に、製品保守用の情報として“(n)”を出力します。
（n：数値）</td></tr>
<tr><td rowspan="2">14</td><td colspan="4">詳細情報</td><td></td><td>詳細情報として次の内容を出力します。</td></tr>
<tr><td>○</td><td>○</td><td>○</td><td>○</td><td>開始終了情報</td><td>PowerRDBconnectorの内部関数開始時に”Start”を出力します。</td></tr>
</table>

<table>
<tr><th rowspan="2">No</th><th colspan="4">TraceLevel</th><th rowspan="2">出力項目</th><th rowspan="2">内容</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>PowerRDBconnectorの内部関数終了時に、正常の場合は”Normal End”を、エラーの場合は”ABNormal End”を出力します。
"Start"と"Normal End"または"ABNormal End"が対で出力されます。
"Start"のみ出力されている場合は、処理中であることを意味します。</td></tr>
<tr><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>データベースのVL</td><td>セッション開設時、データベースのバージョンレベルを出力します。</td></tr>
<tr><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>内部動作モード</td><td>PowerRDBconnectorの製品保守用の情報を出力します。</td></tr>
<tr><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>テーブル名</td><td>OPEN文とCLOSE文の延長で出力します。</td></tr>
<tr><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>COBOL初期化ファイル情報</td><td>OPEN文に対応するCOBOL初期化ファイルの情報を出力します。</td></tr>
<tr><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>アクセスモード</td><td>COBOLのアクセスモードを次の形式で出力します。ただし、64ビットモジュールのみ出力されます。
<table><tr><th>アクセスモード</th><th>出力形式</th></tr><tr><td>SEQUENTIAL</td><td>AccessMode=SEQUENTIAL(0x0001)</td></tr><tr><td>DYNAMIC</td><td>AccessMode=DYNAMIC(0x0002)</td></tr><tr><td>RANDOM</td><td>AccessMode=RANDOM(0x0003)</td></tr></table></td></tr>
<tr><td></td><td>－</td><td>○</td><td>○</td><td>○</td><td>列情報</td><td>データベースの列情報を取得時に、列情報を出力します。未サポートのデータ型の場合は”UNDEFINE(データ型番号)”と出力します。
列情報の形式を、以下に示します。
列名、データ型名、データサイズ
（文字項目の場合は、データ長。数字項目の場合は、桁数-精度）</td></tr>
<tr><td></td><td>－</td><td>○</td><td>○</td><td>○</td><td>エラー情報</td><td>エラー発生時、エラーログおよびイベントログに出力する内容を出力します。</td></tr>
<tr><td></td><td>－</td><td>－</td><td>○</td><td>○</td><td>エラーダンプ</td><td>エラー発生時、要因となるデータを16進ダンプの形式で出力します。
（注）</td></tr>
<tr><td></td><td>－</td><td>－</td><td>－</td><td>○</td><td>ダンプ</td><td>キー値と入出力データを16進ダンプ形式で出力します。
（注）</td></tr>
</table>

○:出力します。

×:出力しません。

（注）

トレースデータの出力コード系について
トレース情報に出力するエラーダンプや、ダンプに含まれる文字データは、以下のコード系での16進ダンプの形式で出力されます。

表5.13 トレースデータの出力コード系について

	コンパイル時の実行時の文字コード系の指定		出力コード系	備考
	NetCOBOL for Windows	NetCOBOL for .NET		
REWRITE/WRITE時の書き込みデータや、キーアクセス時のキー値の場合	SJIS	SJIS	シフトJIS	COBOLから通知されたデータではなく、データベースへ通知するために変換したデータが出力されます。
	UCS2(UTF-16)	UTF8-UCS2	UCS2(UTF-16)	
	―	SJIS-UCS2	UTF-8	
READ時の読み込みデータの場合	SJIS	SJIS	シフトJIS	データベースから読み込んだデータではなく、COBOLへ通知するために変換したデータが出力されます。
	UCS2(UTF-16)	UTF8-UCS2	・英数字項目 UTF-8 ・日本語項目 UCS2	
	―	SJIS-UCS2	・英数字項目 シフトJIS ・日本語項目 UCS2	

5.2.3.2 トレースの出力例

トレースファイル（FJSVdbio_trc.log）の出力例を以下に示します。

なお、以下では説明の都合で、トレース出力結果に適宜空白を埋め込んでいます。

```
<< TRACE DATETIME=2011/08/08-09:52:58.016 14318180 >>
<< Windows:6/0/10 Windows Server 2008, Enterprise Edition (x64) Service Pack 1 >>
       ～～中略～～
<< TraceLevel=1, LOG_FILE=E:¥Log¥FJSVdbio_trc.log >>
2011/08/08 09:53:05.780; 45～68; [0～F4];[0～24];[0～B20]; -----～; [prdb.c ～],0000; prdb_start()     ～; -～(0) : Start.
2011/08/08 09:53:05.780; 45～03; [0～F4];[0～24];[0～B20]; -----～; [prdb.c ～],0000; prdb_start()     ～; -～(0) : Normal End.
2011/08/08 09:53:08.372; 45～08; [0～F4];[0～24];[0～B20]; OPEN ～; [prdb.c ～],0000; prdb_isopen()    ～; -～(0) : Start.
2011/08/08 09:53:08.372; 45～30; [0～F4];[0～24];[0～B20]; OPEN ～; [sub_CwX～],0000; sub_thr_isopen() ～; -～(0) : Start.
2011/08/08 09:53:08.372; 45～98; [0～F4];[0～24];[0～B20]; OPEN ～; [CwXisam～],0001; cw_getProfile()  ～; -～(0) : PrepareMode=1
2011/08/08 09:53:08.372; 45～75; [0～F4];[0～24];[0～B20]; OPEN ～; [sub_CwX～],0001; cw_FileOpenInfo()～; -～(0) : TableName=M_FILE02
2011/08/08 09:53:08.372; 45～78; [0～F4];[0～24];[0～B20]; OPEN ～; [sub_CwX～],0001; cw_FileOpenInfo()～; -～(0) : AccessMode=DYNAMIC
```

表5.14 プログラミングスタイルによる関数対応表

COBOL命令	内部関数名		備考
	シングルセッション	マルチセッション	
OPEN	thr_isbuild() _isbuild()	thr_isbuild() prdb_isbuild() sub_thr_isbuild()	テーブル名が出力されます。 OUTPUTモードの場合に出力されます。
	thr_isopen2() _isopen()	thr_isopen2() prdb_isopen() sub_thr_isopen()	テーブル名が出力されます。 OUTPUTモード以外の場合に出力されます。

COBOL命令	内部関数名		備考
	シングルセッション	マルチセッション	
	isindexinfo() _isindexinfo()	isindexinfo() prdb_isindexinfo() sub_isindexinfo()	キー項目毎に呼び出されます。
	isaddindex() _isaddindex()	isaddindex() prdb_isaddindex() sub_isaddindex()	キー項目毎に呼び出されます。 OUTPUTモードの場合に出力されます。
CLOSE	isclose() _isclose()	isclose() prdb_isclose() sub_isclose()	—
READ	isread() _isread()	isread() prdb_isread() sub_isread()	—
WRITE	iswrite() _iswrite()	iswrite() prdb_iswrite() sub_iswrite()	—
REWRITE	isrewcurr() _isrewcurr() isrewrite() _isrewrite()	isrewcurr() prdb_isrewcurr() sub_isrewcurr() isrewrite() prdb_isrewrite() sub_isrewrite()	—
DELETE	isdelete() _isdelete() isdelcurr() _isdelcurr()	isdelete() prdb_isdelete() sub_isdelete() isdelcurr() prdb_isdelcurr() sub_isdelcurr()	—
START	isstart() _isstart()	isstart() prdb_isstart() sub_isstart()	—

COBOL命令に対応して複数行の情報が出力されます。以下にCOBOL命令毎のパターン例を示します。

図5.1 マルチセッションプログラミング時のトレース出力パターン例

COBOL命令	トレース情報(TraceLevel=3の場合)	内容
セッション開始 COB_PRDB_START	; ---------- : ～ : prdb_start() ; -------(0) : Start. ; ---------- : ～ : prdb_start() ; -------(0) : Normal End.	
セッション終了 COB_PRDB_END	; ---------- : ～ : prdb_end() ; -------(0) : Start. ; ---------- : ～ : prdb_end() ; -------(0) : Normal End.	
OPEN (INPUT)	; OPEN : ～ : prdb_isopen() ; -------(0) : Start. ; OPEN : ～ : sub_thr_isopen() ; -------(0) : Start. ; OPEN : ～ : openDBODBC() ; -------(0) : Server Version= xxxxxx ; OPEN : ～ : openTBODBC() ; -------(0) : TableName=FILE02 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_NUM_UNSIGN_SMALLINT, SMALLINT(5),5-0 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_DATA_CHAR,CHAR(1),8 ; OPEN : ～ : sub_thr_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End.	●接続先データベースのバージョン情報が出力されます。(Server Version) ●オープンしたファイル名が出力されます。(TableName=) ●列情報が出力されます。 ●本例ではテーブルロックを行ったことが分かります。(TBLLOCK)
OPEN (OUTPUT)	; OPEN : ～ : prdb_isbuild() ; -------(0) : Start. ; OPEN : ～ : sub_thr_isbuild() ; -------(0) : Start. ; OPEN : ～ : openDBODBC() ; -------(0) : Server Version= xxxxxx ; OPEN : ～ : openTBODBC() ; -------(0) : TableName=FILE02 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_NUM_UNSIGN_SMALLINT, SMALLINT(5),5-0 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_DATA_CHAR,CHAR(1),8 ; OPEN : ～ : sub_thr_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End.	●接続先データベースのバージョン情報が出力されます。(Server Version) ●オープンしたファイル名が出力されます。(TableName=) ●列情報が出力されます。 ●本例ではテーブルロックを行ったことが分かります。(TBLLOCK)

COBOL命令	トレース情報(TraceLevel=3の場合)	内容
OPEN (I-O)	; OPEN : ～ : prdb_isopen() ; -------(0) : Start. ; OPEN : ～ : sub_thr_isopen() ; -------(0) : Start. ; OPEN : ～ : openDBODBC() ; -------(0) : Server Version= xxxxxx ; OPEN : ～ : openTBODBC() ; -------(0) : TableName=FILE02 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_NUM_UNSIGN_SMALLINT, SMALLINT(5),5-0 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_DATA_CHAR,CHAR(1),8 ; OPEN : ～ : sub_thr_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End.	●接続先データベースのバージョン情報が出力されます。 (Server Version) ●オープンしたファイル名が出力されます。 (TableName=) ●列情報が出力されます。 ●本例ではテーブルロックを行ったことが分かります。(TBLLOCK)
OPEN (EXTEND)	; OPEN : ～ : prdb_isopen() ; -------(0) : Start. ; OPEN : ～ : sub_thr_isopen() ; -------(0) : Start. ; OPEN : ～ : openDBODBC() ; -------(0) : Server Version= xxxxxx ; OPEN : ～ : openTBODBC() ; -------(0) : TableName=FILE02 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_NUM_UNSIGN_SMALLINT, SMALLINT(5),5-0 ; OPEN : ～ : openTBODBC() ; -------(0) : REC_DATA_CHAR,CHAR(1),8 ; OPEN : ～ : sub_thr_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isopen() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Start. ; OPEN : ～ : sub_isindexinfo() ; TBLLOCK(7) : Normal End. ; OPEN : ～ : prdb_isindexinfo() ; TBLLOCK(7) : Normal End.	●接続先データベースのバージョン情報が出力されます。 (Server Version) ●オープンしたファイル名が出力されます。 (TableName=) ●列情報が出力されます。 ●本例ではテーブルロックを行ったことが分かります。(TBLLOCK)
CLOSE	; CLOSE : ～ : prdb_isclose() ; TBLLOCK(7) : Start. ; CLOSE : ～ : sub_isclose() ; TBLLOCK(7) : Start. ; CLOSE : ～ : closeDBODBC() ; TBLLOCK(7) : TableName=FILE02 ; CLOSE : ～ : sub_isclose() ; -------(0) : Normal End. ; CLOSE : ～ : prdb_isclose() ; -------(0) : Normal End.	●クローズしたファイル名が出力されます。 (TableName=)
START	; START : ～ : prdb_isstart() ; TBLLOCK(7) : Start. ; START : ～ : sub_isstart() ; TBLLOCK(7) : Start. ; START : ～ : _RandomFind() ; TBLLOCK(7) : ; START : ～ : +0 +4 +8 +C ; START : ～ : +000000 02C10200 00000000 00000000 00000000 .... ; START : ～ : +000010 00000000 000000 .... ; START : ～ : sub_isstart() ; TBLLOCK(7) : Normal End. ; START : ～ : prdb_isstart() ; TBLLOCK(7) : Normal End.	●キー値のダンプ情報が出力されます。

COBOL命令	トレース情報(TraceLevel=3の場合)	内容
順READ	; READ(SEQ) : ～ : prdb_isread() ; TBLLOCK(7) : Start. ; READ(SEQ) : ～ : sub_isread() ; TBLLOCK(7) : Start. ; READ(SEQ) : ～ : getRecordODBC() ; TBLLOCK(7) : ; READ(SEQ) : ～ : +0 +4 +8 +C ; READ(SEQ) : ～ : +000000 00016162 63202020 2020 ..abc ; READ(SEQ) : ～ : sub_isread() ; TBLLOCK(7) : Normal End. ; READ(SEQ) : ～ : prdb_isread() ; TBLLOCK(7) : Normal End.	●データベースから読込まれたデータダンプ情報が出力されます。
乱READ	; READ(RAN) : ～ : prdb_isstart() ; TBLLOCK(7) : Start. ; READ(RAN) : ～ : sub_isstart() ; TBLLOCK(7) : Start. ; READ(RAN) : ～ : _RandomFind() ; TBLLOCK(7) : ; READ(RAN) : ～ : +0 +4 +8 +C ; READ(RAN) : ～ : +000000 02C10200 00000000 00000000 00000000 .... ; READ(RAN) : ～ : +000010 00000000 000000 ..... ; READ(RAN) : ～ : sub_isstart() ; TBLLOCK(7) : Normal End. ; READ(RAN) : ～ : prdb_isstart() ; TBLLOCK(7) : Normal End. ; READ(RAN) : ～ : prdb_isread() ; TBLLOCK(7) : Start. ; READ(RAN) : ～ : sub_isread() ; TBLLOCK(7) : Start. ; READ(RAN) : ～ : getRecordODBC() ; TBLLOCK(7) : ; READ(RAN) : ～ : +0 +4 +8 +C ; READ(RAN) : ～ : +000000 00017171 71202020 2020 ..qqq ; READ(RAN) : ～ : sub_isread() ; TBLLOCK(7) : Normal End. ; READ(RAN) : ～ : prdb_isread() ; TBLLOCK(7) : Normal End.	●キー値のダンプ情報が出力されます。 ●データベースから読込まれたデータダンプ情報が出力されます。
順DELETE	; DELETE(SEQ) : ～ : prdb_isdelcurr() ; TBLLOCK(7) : Start. ; DELETE(SEQ) : ～ : sub_isdelcurr() ; TBLLOCK(7) : Start. ; DELETE(SEQ) : ～ : sub_isdelcurr() ; TBLLOCK(7) : Normal End. ; DELETE(SEQ) : ～ : prdb_isdelcurr() ; TBLLOCK(7) : Normal End.	
乱DELETE	; DELETE(RAN) : ～ : prdb_isdelete() ; TBLLOCK(7) : Start. ; DELETE(RAN) : ～ : sub_isdelete() ; TBLLOCK(7) : Start. ; DELETE(RAN) : ～ : _RandomFind() ; TBLLOCK(7) : ; DELETE(RAN) : ～ : +0 +4 +8 +C ; DELETE(RAN) : ～ :+000000 02C10400 00000000 00000000 00000000 ... . ; DELETE(RAN) : ～ :+000010 00000000 000000 ..... ; DELETE(RAN) : ～ : sub_isdelete() ; TBLLOCK(7) : Normal End. ; DELETE(RAN) : ～ : prdb_isdelete() ; TBLLOCK(7) : Normal End.	●キー値のダンプ情報が出力されます。
WRITE	; WRITE : ～ : prdb_iswrite() ; TBLLOCK(7) : Start. ; WRITE : ～ : sub_iswrite() ; TBLLOCK(7) : Start. ; WRITE : ～ : insertRecordODBC() ; TBLLOCK(7) : ; WRITE : ～ : +0 +4 +8 +C ; WRITE : ～ : +000000 02C10200 00000000 00000000 00000000 .... ; WRITE : ～ : +000010 00000000 0000 ..... ; WRITE : ～ : insertRecordODBC() ; TBLLOCK(7)- : ; WRITE : ～ : +0 +4 +8 +C ; WRITE : ～ : +000000 61626300 00000000 abc..... ; WRITE : ～ : sub_iswrite() ; TBLLOCK(7) : Normal End. ; WRITE : ～ : prdb_iswrite() ; TBLLOCK(7) : Normal End.	●キー値のダンプ情報が出力されます。 ●データベースへ書き込んだデータダンプ情報が出力されます。

COBOL命令	トレース情報(TraceLevel=3の場合)	内容
順REWRITE	; REWRITE(SEQ): ～ : prdb_isrewcurr() ; TBLLOCK(7) : Start. ; REWRITE(SEQ): ～ : sub_isrewcurr() ; TBLLOCK(7) : Start. ; REWRITE(SEQ): ～ : updateRecordODBC() ; TBLLOCK(7) : ; REWRITE(SEQ): ～ : +0 +4 +8 +C ; REWRITE(SEQ): ～ : +000000 02C10200 00000000 00000000 00000000 .... ; REWRITE(SEQ): ～ : +000010 00000000 0000 ..... ; REWRITE(SEQ): ～ : updateRecordODBC() ; TBLLOCK(7) : ; REWRITE(SEQ): ～ : +0 +4 +8 +C ; REWRITE(SEQ): ～ : +000000 71717100 00000000 qqq..... ; REWRITE(SEQ): ～ : sub_isrewcurr() ; TBLLOCK(7) : Normal End. ; REWRITE(SEQ): ～ : prdb_isrewcurr() ; TBLLOCK(7) : Normal End.	●キー値のダンプ情報が出力されます。 ●データベースへ書込みを行ったデータダンプ情報が出力されます。
乱REWRITE	; REWRITE(RAN): ～ : prdb_isrewrite() ; TBLLOCK(7) : Start. ; REWRITE(RAN): ～ : sub_isrewrite() ; TBLLOCK(7) : Start. ; REWRITE(RAN): ～ : _RandomFind() ; TBLLOCK(7) : ; REWRITE(RAN): ～ : +0 +4 +8 +C ; REWRITE(RAN): ～ : +000000 02C10400 00000000 00000000 00000000 .... ; REWRITE(RAN): ～ : +000010 00000000 000000 ..... ; REWRITE(RAN): ～ : updateRecordODBC() ; TBLLOCK(7) : ; REWRITE(RAN): ～ : +0 +4 +8 +C ; REWRITE(RAN): ～ : +000000 02C10400 00000000 00000000 00000000 .... ; REWRITE(RAN): ～ : +000010 00000000 0000 ..... ; REWRITE(RAN): ～ : updateRecordODBC() ; TBLLOCK(7) : ; REWRITE(RAN): ～ : +0 +4 +8 +C ; REWRITE(RAN): ～ : +000000 74747400 00000000 ttt..... ; REWRITE(RAN): ～ : sub_isrewrite() ; TBLLOCK(7) : Normal End. ; REWRITE(RAN): ～ : prdb_isrewrite() ; TBLLOCK(7) : Normal End.	●キー値のダンプ情報が出力されます。 ●データベースへ書込みを行ったデータダンプ情報が出力されます。
トランザクション開始	; ---------- : ～ : prdb_sub() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTSTR() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTSTR() ; ------(0) : Normal End. ; ---------- : ～ : prdb_sub() ; ------(0) : Normal End.	―
トランザクション確定	; ---------- : ～ : prdb_sub() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTEND() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTEND() ; ------(0) : Normal End. ; ---------- : ～ : prdb_sub() ; ------(0) : Normal End.	―
トランザクション取消	; ---------- : ～ : prdb_sub() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTCNL() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTCNL() ; ------(0) : Normal End. ; ---------- : ～ : prdb_sub() ; ------(0) : Normal End.	―
テーブルロック解除時のトランザクションの取消	; ---------- : ～ : prdb_sub() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTRBK() ; ------(0) : Start. ; ---------- : ～ : sub_XMROTRBK() ; ------(0) : Normal End. ; ---------- : ～ : prdb_sub() ; ------(0) : Normal End.	―
認証情報の登録	; ---------- : ～ : prdb_sub() ; ------(0) : Start. ; ---------- : ～ : sub_XMROAUTH() ; ------(0) : Start. ; ---------- : ～ : sub_XMROAUTH() ; ------(0) : Normal End. ; ---------- : ～ : prdb_sub() ; ------(0) : Normal End.	―

注意

セッションの変更サブルーチンのトレース出力について

セッションの変更サブルーチン(COB_PRDB_CHG)を実行しても、トレース出力されません。トレースに、変更したセッションIDの情報を出力したい場合には、トレースへの文字列出力サブルーチンを使用してください。詳細は、「5.2.3.3 トレースへの文字列の出力」を参照してください。

5.2.3.3 トレースへの文字列の出力

PowerRDBconnectorのトレースファイルに、任意の文字列を出力することができます。これにより、次のような場合に、トラブル調査をより容易に行うことができます。

- どのプログラムからのアクセスかを特定する文字列を出力し、多重実行時のトレースファイルから、問題点を調査する。
- マルチセッションプログラミングの場合、アクセスしているセッション名を出力し、どのセッションで問題が発生しているかを調査する。

XMROLOGサブルーチン(シングルセッション)か、COB_PRDB_LOGサブルーチン(マルチセッション)を実行してトレースファイルに、任意の文字列を出力します。

注意

トレースへの文字列の出力サブルーチンのサポートについて

XMROLOGサブルーチン(シングルセッション)と、COB_PRDB_LOGサブルーチン(マルチセッション)は、64ビット動作のみサポートしています。32ビット動作ではサポートしていないため、注意してください。

5.2.3.3.1 トレースへの文字列出力サブルーチンのインターフェース(シングルセッション)

トレースへの文字列出力サブルーチンのインターフェースは、以下のとおりです。

- 機能

  トレースファイルに任意文字列を出力します。

- 呼出し形式

```
CALL  "XMROLOG" USING 出力情報 エラー情報 RETURNING 復帰値.
```

- パラメーターのデータ定義

```
01 出力情報.
  02 文字列長          PIC S9(9) COMP-5.
  02 出力文字列        PIC X(260).

01 エラー情報.
  02 終了情報          PIC S9(9) COMP-5.
  02 詳細情報          PIC S9(9) COMP-5.
  02 FILLER            PIC S9(9) VALUE 0.
01 復帰値              PIC S9(9) COMP-5 VALUE 0.
```

  ※上記は、必ずレベル番号01で記載してください。

  ※FILLERは、0を設定してください。

- パラメーターの意味

  ー 文字列長

  トレースファイルに出力する文字列長を、256バイト以内で指定します。

  ー 出力文字列

  トレースファイルに出力する文字列を英数字で指定します。
  文字列長に指定した長さよりも長い文字列を指定した場合、文字列長に指定した長さ分だけが出力されます。

  ー 終了情報/詳細情報

  終了情報および詳細情報を、以下に示します。

表5.15 トレースへの文字列出力サブルーチン(シングルセッション)のエラーコード

error code		意味	対処
終了情報 10進数	詳細情報 10進数		
0	0	・ 正常終了	・ なし

error code		意味	対処
終了情報 10進数	詳細情報 10進数		
91	0	・ 文字列長の指定に誤りがあります。	・ 文字列長が256バイト以内か確認してください。
		・ コード変換でエラーが発生しました。	・ シフトJISにコード変換できない文字があるか確認してください。

・ 復帰値

復帰値は、以下のとおりです。

表5.16 XMROLOGサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常	なし
-1	エラー	あり

・ 使用例

使用例を示します。

```
DATA              DIVISION.
WORKING-STORAGE   SECTION.
01 出力情報.
    02 文字列長     PIC S9(9) COMP-5.
    02 出力文字列   PIC X(260).
01  エラー情報.
    02 終了情報     PIC S9(9) COMP-5.
    02 詳細情報     PIC S9(9) COMP-5.
    02 FILLER       PIC S9(9) VALUE 0.
01 復帰値           PIC S9(9) COMP-5 VALUE 0.
PROCEDURE DIVISION.

OPEN  I-O INFILE.
* 任意ログを出力する。
MOVE  "COBOL-SESSION=USER01" TO 出力文字列.
MOVE 20 TO 文字列長
CALL  "XMROLOG" USING 出力情報 エラー情報 RETURNING 復帰値.
…

CLOSE INFILE.
```

図5.2 出力結果例

```
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isopen()     … : Start.
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isopen()     … : End.
… [00～03A8];[00～0C88];[00～13A4420]; … sub_XMROLOG ()   … : COBOL-SESSION=USER01
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isclose()    … : Start.
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isclose()    … : End.
```

5.2.3.3.2 トレースへの文字列出力サブルーチンのインターフェース（マルチセッション）

トレースへの文字列出力サブルーチンのインターフェースは、以下のとおりです。

・ 機能

トレースファイルに任意文字列を出力します。

・ 呼出し形式

```
CALL "COB_PRDB_LOG" USING BY REFERENCE セッションID BY REFERENCE 出力文字情報
     BY REFERENCE エラー情報 RETURNING 復帰値.
```

・ パラメーターのデータ定義

```
01 セッションID          PIC X(30).
01 出力文字情報.
  02 文字列長            PIC S9(9) COMP-5
  02 出力文字列          PIC X(260).

01 エラー情報.
  02 終了情報            PIC S9(9) COMP-5.
  02 詳細情報            PIC S9(9) COMP-5.
  02 FILLER              PIC S9(9) VALUE 0.
01 復帰値                PIC S9(9) COMP-5 VALUE 0.
```

※上記は、必ずレベル番号01で記載してください。

※FILLERは、0を設定してください。

・ パラメーターの意味

－ セッションID（必須）

英数字項目で30バイト指定します。
セッション開設サブルーチンで指定したセッションIDを指定してください。

文字の種類の制約はありません。なお、本パラメーターは必ず指定してください。

－ 文字列長

トレースファイルに出力する文字列長を、256バイト以内で指定します。

－ 出力文字列

トレースファイルに出力する文字列を英数字で指定します。
文字列長に指定した長さよりも長い文字列を指定した場合、文字列長に指定した長さ分だけが出力されます。

－ 終了情報/詳細情報

終了情報および詳細情報を、以下に示します。

表5.17 トレースへの文字列出力サブルーチン(マルチセッション）のエラーコード

error code		意味	対処
終了情報 10進数	詳細情報 10進数		
0	0	・ 正常終了	・ なし
91	0	・ 文字列長の指定に誤りがあります。	・ 文字列長が256バイト以内か確認してください。
		・ コード変換でエラーが発生しました。	・ シフトJISにコード変換できない文字があるか確認してください。

・ 復帰値

復帰値は、以下のとおりです。

表5.18 COB_PRDB_LOGサブルーチンの復帰値

復帰値	意味	イベントログ出力の有無
0	正常	なし
-1	エラー	あり
-201	セッションIDの指定が正しくありません。	なし

・ 使用例

マルチセッションを使用した場合、トレースファイルよりCOBOLアプリケーションのセッションIDとプログラムのスレッドIDを関連付けすることができます。

```
DATA              DIVISION.
WORKING-STORAGE   SECTION.
01  SESSION-ID      PIC X(30).
01  RET-VALUE       PIC S9(9) COMP-5.
01  セッション情報.
  02  文字列長        PIC 9(9) COMP-5.
  02  出力文字列      PIC X(260).
01 エラー情報.
  02 終了情報 PIC S9(9) COMP-5.
  02 詳細情報 PIC S9(9) COMP-5.
  02 FILLER PIC S9(9) VALUE 0.
01 復帰値 PIC S9(9) COMP-5 VALUE 0.
*
PROCEDURE DIVISION.
* セッション開始
MOVE "SESSION-01" TO SESSION-ID.
CALL "COB_PRDB_START" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.

OPEN I-O INFILE.

* セッションIDを出力する。
MOVE "COBOL-SESSION=SESSION-01" TO 出力文字列.
MOVE 24 TO 文字列長.
CALL "COB_PRDB_LOG" USING BY REFERENCE SESSION-ID
                          BY REFERENCE セッション情報
                          BY REFERENCE エラー情報 RETURNING 復帰値.
…
CLOSE INFILE.

* セッション終了
CALL "COB_PRDB_END" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.

* セッション開始
MOVE "SESSION-02" TO SESSION-ID.
CALL "COB_PRDB_START" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.

OPEN I-O INFILE.
* セッションIDを出力する。
MOVE "COBOL-SESSION=SESSION-02" TO 出力文字列.
MOVE 24 TO 文字列長.
CALL "COB_PRDB_LOG" USING BY REFERENCE SESSION-ID
                          BY REFERENCE セッション情報
                          BY REFERENCE エラー情報 RETURNING 復帰値.
```

```
…
CLOSE INFILE.

* セッション終了
CALL "COB_PRDB_END" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.
```

図5.3 出力結果例

```
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isopen()     … : Start.
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isopen()     … : End.
… [00～03A8];[00～0C88];[00～13A4420]; … sub_XMROLOG ()   … : COBOL-SESSION=USER01
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isclose()    … : Start.
… [00～03A8];[00～0C88];[00～13A4420]; … sub_isclose()    … : End.
```

5.2.3.3.3 トレースへの文字列出力サブルーチンの注意事項

トレースへの文字列出力サブルーチンの注意事項を、以下に示します。

- 本サブルーチンは、PowerRDBconnector動作環境ファイルのTraceLevelプロパティの値によらず、TraceModeプロパティがONのときだけ、動作します。
  TraceModeプロパティがOFFの場合、本サブルーチンは正常終了しますが、動作しません。
- トレースファイルのコード系は、シフトJISです。このため、COBOLアプリケーションが、unicodeで動作していた場合でも、本サブルーチンで出力される文字列はシフトJISに変換されて出力されます。シフトJISにない文字を出力しようとした場合、本サブルーチンはエラーとならず、？のように文字化けして出力します。
- 本サブルーチンによりトレースファイルに文字列を出力時、トレースファイルの書込みでエラーが発生しても、本サブルーチンはエラー終了されず、トレース情報の採取のみ停止します。
- 本サブルーチンは、以下の区間でのみ動作します。

表5.19 トレースへの文字列出力サブルーチンの有効区間について

プログラミングスタイル	サブルーチン名	有効区間
シングルセッション	XMROLOG	プロセス初回OPEN後～プロセス終了
マルチセッション	COB_PRDB_LOG	スレッド初回OPEN後～セッション閉設

  上記区間外では、本サブルーチンは正常終了しますが、動作しません。

- 本サブルーチンで大量の文字列を出力すると、トレースファイルの負荷が増加するため、性能劣化します。調査に必要なときのみ、本サブルーチンを実行するようにしてください。

付録A　他製品のファイル資源

本章では、他製品（ファイルシステム、データベース系）のファイル資源および代替方法について説明します。

業務システムの機能・構成設計時に、CSP/FXやASPのデータベース資産としてよく使用される、論理ファイルの使用について、開発者が確認する場合にお読みください。

A.1　論理ファイル

論理ファイルとは、物理ファイルを射影する仮想的なファイルで、以下の製品に存在します。
物理ファイルとは、物理データが格納されたデータファイルです。詳しくは、該当する製品のマニュアルを参照してください。

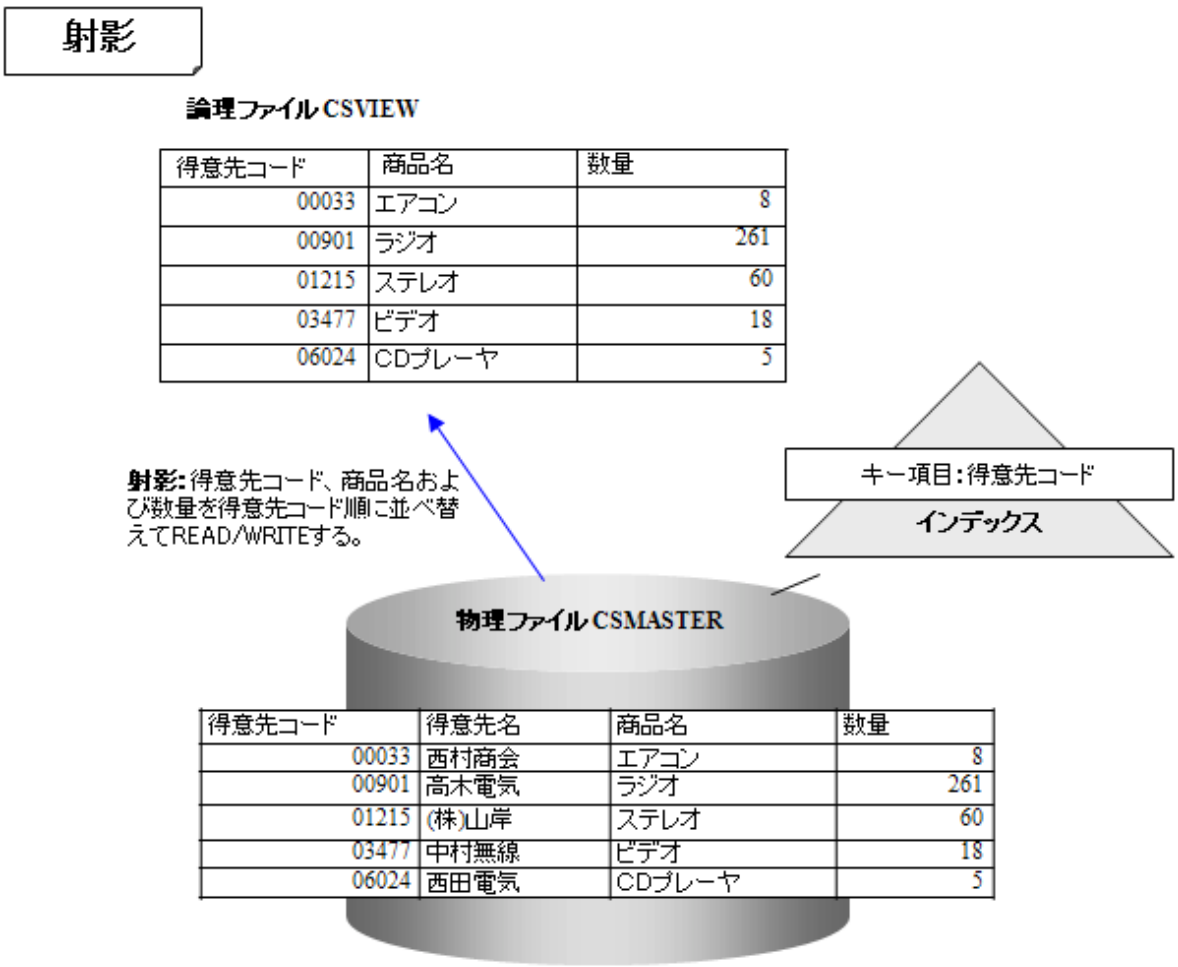
- 「CSP/FX FX-RDB」
- 「ASP RDB/6000、Symfoware6000」

論理ファイルで定義された単純論理ファイルは、ビューを定義することで上記製品の論理ファイルと同等の機能を実現することが可能です。

なお、複数のテーブルで構成された結合ビューに対する更新、およびビューへのOUTPUTモードのオープンはできません。

単純論理ファイル

単一ビューで作成することで、上記製品の論理ファイルと同等の機能が実現できます。



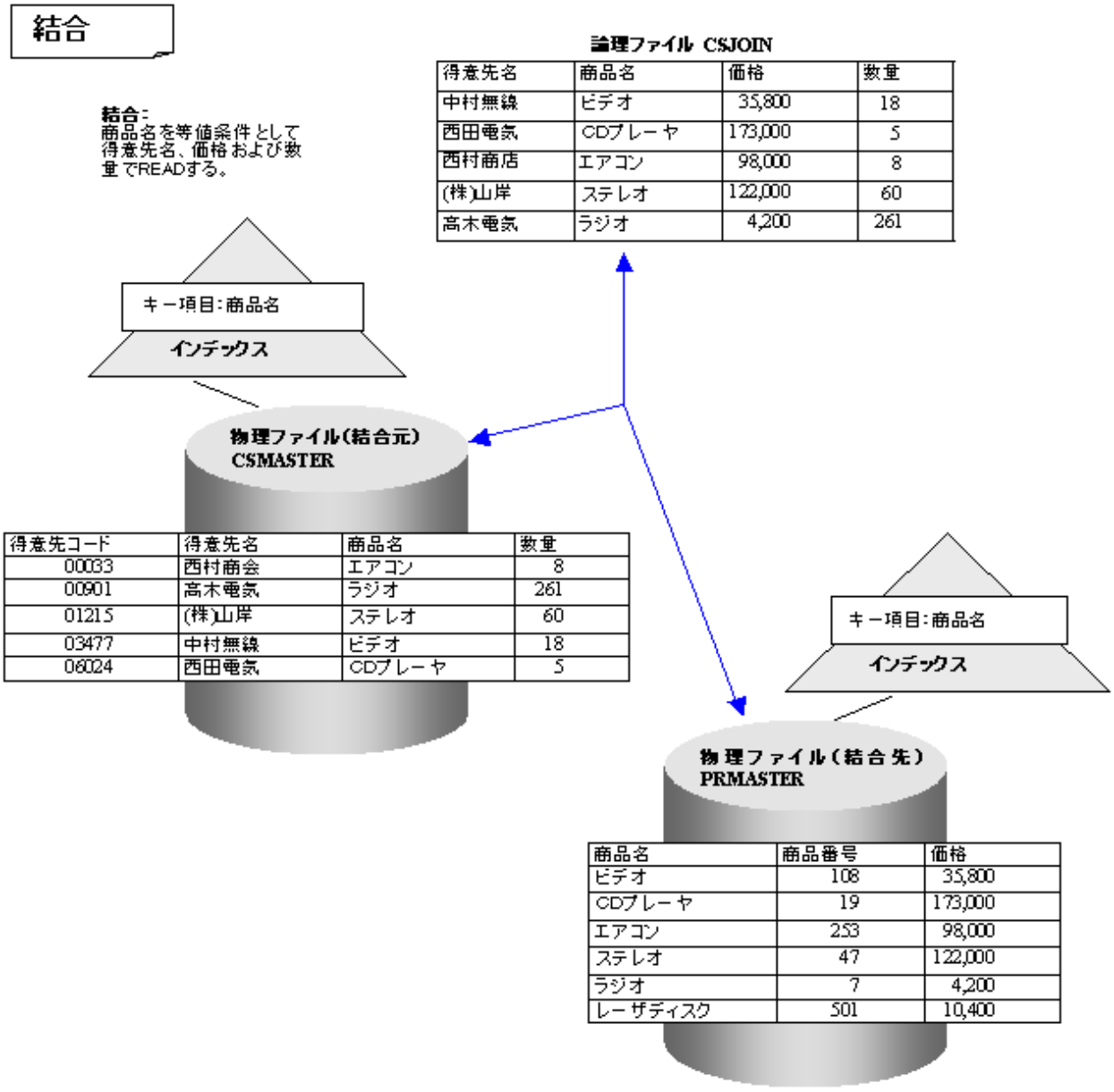
SQL定義文の例

```
CREATE VIEW CSVIEW
(得意先コード_DECIMAL, 商品名_CHAR, 数量_INTEGER)
AS SELECT 得意先コード, 商品名, 数量
FROM CSMASTER

CREATE UNIQUE INDEX CSVIEWIX ON CSMASTER (得意先コード)
```

結合論理ファイル

結合ビューを作成することで参照できます。



SQL定義文の例

```
CREATE VIEW CSJOIN
(CS_NAME_CHAR ,P_NAME_CHAR, PRICE_DECIMAL, QUANTITY_INTEGER)
AS SELECT 得意先名, 商品名, 価格, 数量
FROM CSMASTER, PRMASTER
```

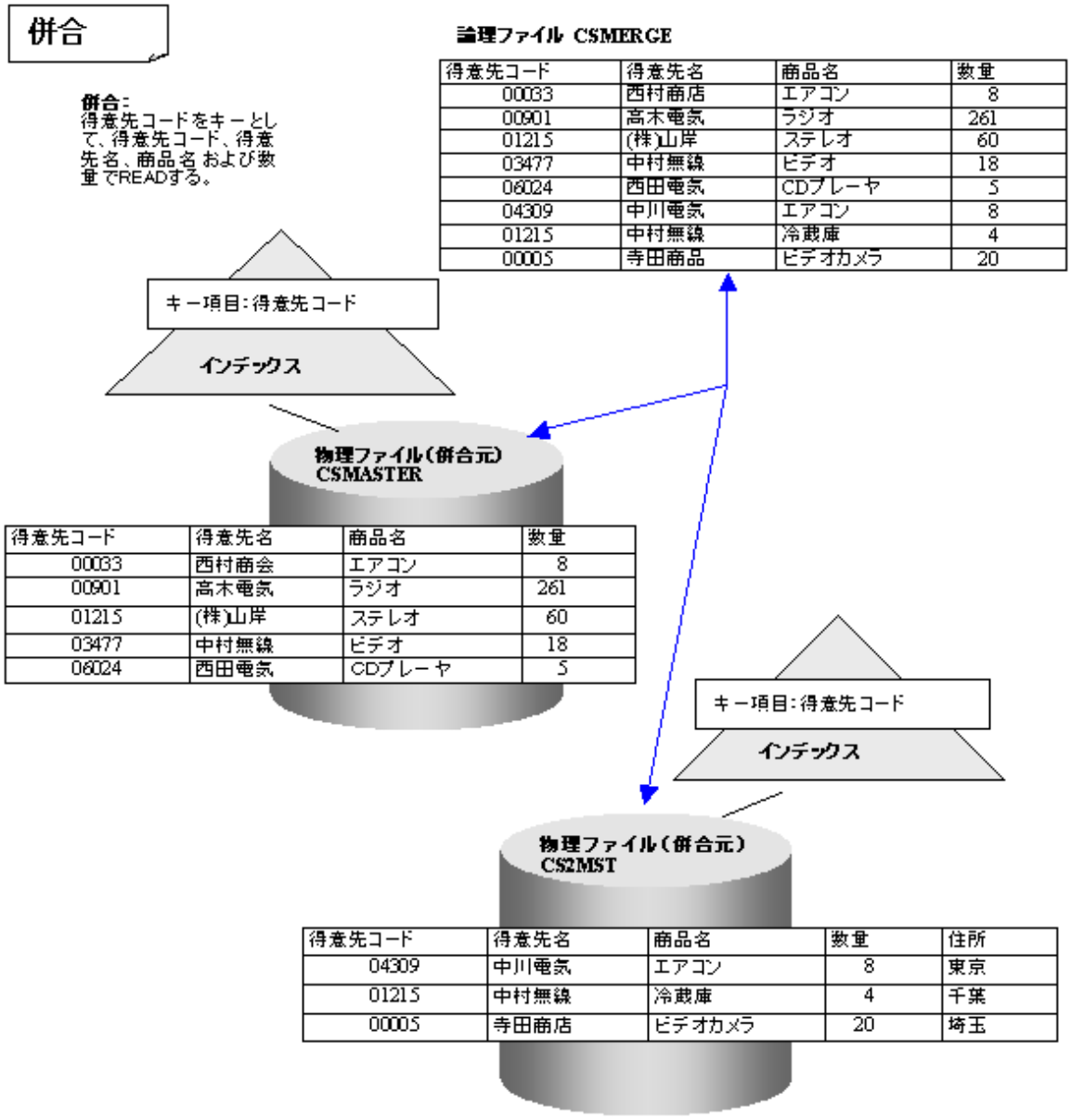
WHERE CSMASTER.商品名 = PRMASTER.商品名~

CREATE UNIQUE INDEX CSJOINIX ON CSMASTER (商品名)

CREATE UNIQUE INDEX CSJOINIX2 ON PRMASTER (商品名)

併合論理ファイル

UNION結合ビューで作成することで参照できます。



SQL定義文の例

CREATE VIEW CSMERGE
(CS_CODE_DECIMAL,CS_NAME_CHAR,PRODUCT_NAME_CHAR, QUANTITY_INTEGER)
AS
SELECT 得意先コード, 得意先名, 商品名, 数量
FROM CSMASTER~
UNION ALL

```
SELECT 得意先コード, 得意先名, 商品名, 数量
FROM CS2MST

CREATE UNIQUE INDEX CSMERGEIX ON CS2MST (得意先コード)
```

付録B　トラブルシューティング

本章では、PowerRDBconnectorでトラブルが発生した場合の対処方法と、トラブル事例について説明します。

B.1　トラブルへの対処方法

本節では、PowerRDBconnector 使用時にトラブルが発生した場合の調査方法について示します。

- COBOLのアクセス命令がエラーとなる場合
  アプリケーションに通知されるエラーコードを元に原因を取り除いてください。
  エラーコードに対する対処方法については、「5.1 エラー時の対処」を参照してください。
  PowerRDBconnectorのモジュールでアプリケーションエラーが発生した場合には、イベントログ、再現する場合は、トレースを採取し、当社技術員に連絡してください。
- COBOLのアクセス性能を調査する場合
  PowerRDBconnectorには性能値を算出する機能はありません。COBOLアプリケーションで、時刻を採取するなどの対処を行って調査してください。
- COBOLのアクセス文からの応答がない場合
  イベントログを確認して、なんらかのエラーが発生していないか確認してください。
  どのアクセス文から応答がないかは、COBOLアプリケーションのトレース機能などを用いて調査してください。
- COBOLのアクセス文の処理結果が正しくない場合
  COBOLアプリケーションのトレース機能や、PowerRDBconnectorのトレース機能を用いて、正しいシーケンスで動作しているか、アクセスしたデータ内容が正しいのかを調査してください。PowerRDBconnectorのトレース機能については、「5.2 トレース出力機能」を参照してください。

B.2　事例

本節では、よくある質問や、陥りやすいトラブルとその解決方法について示します。

B.2.1　製品仕様について

1. **1つのプログラムで複数のデータベースをアクセスすることはできますか？**

   できません。1つのプログラムでアクセスできるデータソースは1つだけです。プログラムごとにアクセスするデータソースを分けることはでき、PowerRDBconnector動作環境ファイル(DBIO_ENVファイル)で指定します。詳細は、「3.1.4 PowerRDBconnector 動作環境ファイル」を参照してください。

2. **サポートされるファイル編成を教えてください。**

   SEQUENTIAL(順編成)とINDEX(索引編成)へのアクセスがサポートされています。
   RELATIVE(相対編成)は、COBOLファイルシステム等をご使用ください。
   なお、順編成ファイルで格納（WRITE）した順番にレコードを読み込む場合は、”_NOITEMK”のサフィックスを使用してください。詳細は、「3.1.3.6 COBOLと関連付けのない項目の指定方法」を参照してください。

3. **テーブル全体を排他することができますか？**

   テーブルロック機能を使ってください。詳細は、「3.5.3 テーブルロック機能」を参照してください。

4. **COBOLソースのデータ項目名を、PowerRDBconnectorで規定された項目名（例：item_CHAR）に合わせる必要がありますか？**

   合わせる必要はありません。PowerRDBconnectorでは、対象データベースの列名により、対応するCOBOLデータ型に合わせた処理を行っています。COBOLソース内のデータ定義の属性と順序が、データベースの列名と順序に一致していればよく、COBOLソース内に記述したデータ項目名と列名を一致させる必要はありません。

5. **テーブルの列名のサフィックスに属性を付加せずに、PowerRDBconnectorからアクセスすることはできますか？**

   できません。
   ビューを定義し、ビュー側の列名のサフィックスに属性(例：item_CHAR)を付加し、ビュー側をアクセスするようにすることで、読込みが可能となります。

6. **X項目やN項目における、PowerRDBconnectorでの後方空白の変換規則を教えてください。**

   後方空白の変換規則については、「3.7.1 後方空白補正」を参照してください。

7. **X項目をシフトJIS、N項目をunicode(UCS2)で扱うことはできますか？**

   NetCOBOL for Windowsではできません。
   NetCOBOL for .NETでは可能です。
   COBOL翻訳時の実行時コード系は、RCS(SJIS-UCS2)を指定してください。詳細は、「3.1.6 コンパイル方法」を参照してください。

8. **NULL値を含む列を、PowerRDBconnectorから読み込むことはできますか？**

   NULL値の読み書きはできません。

   NULL可能列を含むテーブルは、以下のいずれかの方法でNULL可能列をアクセス対象から除外してアクセスしてください。

   - NULL可能列を除外したビューを作成して、ビューにアクセスする。
   - NULL可能列をCOBOLと関連付けのない項目に指定する。詳細は、「3.1.3.6 COBOLと関連付けのない項目の指定方法」を参照してください。

9. **可変長属性の列があるテーブルの読込みや書込みができますか？**

   可変長属性の列への読込みや書込みができます。詳細は、「3.1.3.5 列定義の対応」を参照してください。

10. **FLOAT属性の列があるテーブルの読込みや書込みができますか？**

    FLOAT属性の列を含むテーブルへの読込みや書込みはできません。

    FLOAT属性の列を含むテーブルは、以下のいずれかの方法でFLOAT属性の列をアクセス対象から除外してアクセスしてください。

    - FLOAT属性の列を除外したビューを作成して、ビューにアクセスする。
    - FLOAT属性の列をCOBOLと関連付けのない項目に指定する。詳細は、「3.1.3.6 COBOLと関連付けのない項目の指定方法」を参照してください。

11. **DATE属性やTIMESTAMP属性の列があるテーブルの読込みや書込みができますか？**

    DATE属性やTIMESTAMP属性の列を含むテーブルへの読込みや書込みはできません。

    DATE属性やTIMESTAMP属性の列を含むテーブルは、以下のいずれかの方法でDATE属性とTIMESTAMP属性の列をアクセス対象から除外してアクセスしてください。

    - DATE属性やTIMESTAMP属性の列を除外したビューを作成して、ビューにアクセスする。
    - DATE属性やTIMESTAMP属性の列をCOBOLと関連付けのない項目に指定する。詳細は、「3.1.3.6 COBOLと関連付けのない項目の指定方法」を参照してください。

12. **COBOLアプリケーションでREDEFINES句が使用できますか？**

    X項目をX項目で再定義している場合のみ使用できます。使用する場合、最小単位のX項目に対して、データベースのCHAR型と対応させます。

13. **COBOLアプリケーションでOCCURS句が使用できますか？**

    使用できます。データベースのテーブルには、COBOLの繰り返し項目数分の列の定義が必要です。

14. **1つのアプリケーションから異なる複数のサーバ内のデータベースにアクセスできますか？**

    1つのアプリケーションからは、特定のサーバ内のデータベースにしかアクセスできません。

    動作するプロセスが異なる場合、PowerRDBconnector動作環境ファイルを分けることで異なるサーバで同じ種類のデータベースにアクセスできます。

15. **JIS2004の文字コード系の文字が使えますか？**

    以下の環境で使用できます。

    - Windows 7、Windows Vista、Windows Server 2008 R2、Windows Server 2008、Windows Server 2008(x64)の場合

ー SQL Server 2005、SQL Server 2008の場合

ただし、SQL Server 2005の場合、データベースの問題により、JIS2004固有文字は正しく検索できません。

ー NetCOBOL V10を使用する場合

PowerRDBconnector動作環境ファイルや、COBOL初期化ファイルなどのパス名、スキーマ名、表名、列名には使用できません。
詳細は、「4.1.2.7 文字コードについて」を参照してください。

16. **前バージョンのPowerRDBconnectorの環境は、そのまま新しいバージョンでも使用できますか？**

前バージョンで使用していた以下の環境は、そのまま新しいバージョンでも使用できます。

ー PowerRDBconnector動作環境ファイル（DBIO_ENV）

ー COBOL初期化ファイル

ー データベースの列定義

ただし、新しいバージョンで追加されたプロパティなどは、初期値で動作しますので注意してください。

17. **32ビット動作のPowerRDBconnectorの環境を、64ビット動作の環境へそのまま移行できますか？**

PowerRDBconnectorを32ビットから64ビットに変更するときは、以下の点に注意してください。

ー NetCOBOLも64ビット版に変更してください。その際、32ビットのアプリケーションはそのままでは動作しないので、再翻訳が必要です。

詳細は、「NetCOBOLのマニュアル」を参照してください。

ー 環境変数を使用していた場合

PowerRDBconnector動作環境ファイルのパス名が設定されていた環境変数“DBIO_ENV”を“DBIO_ENV_x64”に変更してください。

なお、PowerRDBconnector動作環境ファイル（DBIO_ENV）の名前は変わりません。

ー 32ビット版のデータベースを使用していた場合

32ビット版のデータベースを、64ビット版のデータベースに変更する必要があります。同時に、PowerRDBconnector動作環境ファイル（DBIO_ENV）の接続先データベースを64ビット版のデータベースに変更します。

なお、64ビット版のPowerRDBconnectorを使用して32ビット版のデータベースへ接続した場合は以下のエラーとなります。

- FILE STATUS：90
- iserrno：22
- isstat1：'0'、isstat2：0x00、isstat3：'0' 、isstat4：0x00

ー COBOL初期化ファイルのAccessModeプロパティの指定が、COBOLプログラム内のAccessMode句と異なっていた場合

64ビット版のPowerRDBconnectorでは、エラーとなります。
COBOL初期化ファイルのAccessModeプロパティの指定とCOBOLプログラム内のAccessMode句を一致させてください。

なお、32ビット動作では、AccessModeプロパティとCOBOLプログラム内のAccessMode句が一致していない場合、動作保証されません。このため、64ビット動作に移行したCOBOLアプリケーションは32ビット動作のときと動作が異なる場合があります。

ー インポートライブラリを使用している場合

インポートライブラリ名が異なります。COBOLアプリケーションを再翻訳するときに指定するインポートライブラリ名を64ビット版のインポートライブラリに変更してください。

64ビット版のインポートライブラリ名については、「表E.5 インポートライブラリの相違点」を参照してください。

ー 動的プログラム構造の場合

COBOL初期化ファイルのエントリ情報に記述するDLL名が異なります。COBOL初期化ファイルのエントリ情報を64ビット版のDLL名に変更してください。

64ビット版のDLL名については、「表E.4 エントリ情報の相違点」を参照してください。

B.2.2 トラブル事例

1. **OPEN文やSTART文などがエラーになります。効率的なデバッグ方法を教えてください。**

   PowerRDBconnectorが、COBOLアプリケーションにエラーを通知する場合、イベントログにエラー詳細(iserrno、isstat1～4、TableNameなど)を記録します。
   COBOLアプリケーションのデバッグ中にイベントビューアを起動しておき、エラー発生時にイベントビューアのアプリケーションログを参照し、「5.1.1 ファイルアクセス時のエラー情報」のエラーコード一覧と突き合わせることで、デバッグ作業を効率化できます。

2. **FILE STATUS 90が通知されます。原因を教えてください。**

   以下の場合などにFILE STATUS 90が通知されます。
   なお、イベントログに詳細なエラー情報が出力されます。詳細情報については、「5.1.1 ファイルアクセス時のエラー情報」を参照してください。

   - COBOL初期化ファイルの設定に誤りがあります。
   - カレントパスにCOBOL初期化ファイルが存在しません。
   - PowerRDBconnector動作環境ファイル(DBIO_ENVファイル)の設定に誤りがあります。
   - カレントパスにPowerRDBconnector動作環境ファイル(DBIO_ENVファイル)が存在しません。
   - COBOL初期化ファイルのTableNameプロパティで指定したテーブル名に全角文字と半角文字が混在しています。
   - スキーマ名やテーブル名に使用できない文字が含まれています。
   - COBOLアプリケーションで指定されたキーに対して、索引(プライマリキー、インデックス)がテーブルに設定されていません。
   - ディスクへの入出力エラーが発生しています。
   - 排他エラーが発生しています。
   - データベースのサービスが未起動です。
   - データベースに対するアクセス権限がありません。
   - 製品ライセンスがありません。
   - 1アプリケーションで129以上オープンしようとしています。

3. **外部十進項目に' '(空白)が格納できません。原因を教えてください。**

   データベースの数値項目には、文字を書き込むことはできません。十進項目に数値以外の値を書き込まないように、アプリケーションの修正を行ってください。またはデータ補正機能を使用してください。データ補正機能の詳細については、「3.7.2 項目属性に違反するデータのチェックと補正」を参照してください。

4. **OUTPUTオープンができません。対処方法を教えてください。**

   db_ddladminグループ(またはsysadmin)権限が付与されていることを確認してください。

5. **データ溢れが発生します。原因を教えてください。**

   NetCOBOLの文字コードとデータベースの文字コードを一致させてください。特に、COBOLのX項目と、データベースのCHAR、NCHARを対応させている場合、文字コード系を一致させていないと、データ溢れが発生します。

6. **START文で HIGH-VALUEや LOW-VALUEへの位置づけができません。対処方法を教えてください。**

   HIGH-VALUEやLOW-VALUEへの位置づけはできません。START FIRSTまたはSTART FIRST WITH REVERSED ORDERに置き換えてください。なお、データ補正機能を使用することで、位置づけ可能になる場合があります。詳しくは、「3.7.2 項目属性に違反するデータのチェックと補正」を参照してください。

7. **Windowsのサービスでアプリケーションを起動しますが、PowerRDBconnectorのアクセスでエラーが発生します。対処方法を教えてください。**

   以下の原因が考えられます。

   - データベース製品のサービス起動完了前にアプリケーションのサービスが起動されると、データベースに接続できません。サービス起動順番の依存関係を見直してください。

ー サービスとして起動されたアプリケーションは暗黙では"SYSTEM"ユーザーとなりますが、このユーザー権限ではデータベースに接続できません。他のユーザー権限でサービスを起動するよう設定してください。

8. **翻訳時にNetCOBOLのエントリ情報ファイル(ENTRY.ENT)にトランザクション処理用DLL名を記載する必要がありますが、DLLがどこに格納されているのかわかりません。**

   トランザクション処理用のDLLは、PowerRDBconnectorのインストールディレクトリ配下に格納されます。PowerRDBconnectorのインストール時に、このディレクトリに対するPATH変数が設定されるため、トランザクション処理用DLL名を、エントリ情報ファイルにフルパスで定義する必要はありません。詳細は、「3.1.6 コンパイル方法」を参照してください。

9. **同一ファイルを親プログラムと子プログラムでそれぞれオープンし、親プログラムでレコードロックした後、子プログラムから同一レコードを読み込むことができてしまいます。理由を教えてください。**

   シングルセッションプログラミングの場合、排他制御はプロセス単位に行われます。このため、同一プロセスの親プログラムと子プログラム間でのレコードロックの獲得待合せは発生しません。詳細は、「3.5 排他制御」を参照してください。

10. **START文やREAD文でFILE STATUS=90、iserrno=22、isstat1='9'、isstat2=0x35、isstat3='9'、isstat4=0x35のエラーが発生します。原因を教えてください。**

    表の列名に設定したサフィックスと、START文やREAD文に渡したデータの型が一致していない場合に発生します。例として、以下の場合に発生します。

    ー 表の列名のサフィックスに「_UNSIGN_NUMERIC」を指定しているにも関わらず、START文で符号付きデータを使用した、またはデータベースの該当列にマイナス値のデータが格納されていました。

    ー 表の列名のサフィックスに「_NUMERIC」を指定しているにも関わらず、符号無しデータを使用しました。

11. **I-O OPENのREAD文を実行した後、REWRITEやDELETE文を実行してもレコードロックが解除されません。どのようにレコードロックを解除するのでしょうか？**

    START FIRST文を発行することでレコードロックが解除できます。画面の入力待ちに入る場合などにレコードロックを解除したい場合、START FIRST文で解除してください。

    なお、下記製品はI-O OPEN(トランザクション未使用)でREAD文を実行した後、REWRITEやDELETEを実行することでレコードロックが解除される動作であったため、これらの製品から移行した場合には、START FIRST文でレコードロックを解除するように、COBOLプログラムの修正が必要になります。

    ー SymfoWARE7000 for Windows NT

    ー PowerRW+

    ー PowerRW+ for NetCOBOL

    ー RDB/6000

    ー Symfoware6000

12. **複数の実表を結合したビュー表に対して、アクセスできません。**

    結合表への更新はできません。詳細は、「4.2.3.2 ビューの使用について」を参照してください。

13. **NetCOBOLのアプリケーションからN項目に対してSTARTを発行するとエラーとなります。対処方法を教えてください。**

    NetCOBOLの文字コードとデータベースの文字コードを一致させてください。

14. **トリガを設定しているテーブルにアクセスすると、START文やREAD文がエラーになる場合があります。トリガを設定しないと正常に動作します。トリガを設定しているテーブルにはアクセスできないのでしょうか？**

    イベントログに以下のエラーメッセージが出力されている場合には、トリガを修正することで回避できます。

    [Microsoft][ODBC SQL Server Driver]ほかの実行結果のために接続できません
    または
    [Microsoft][ODBC SQL Server Driver]カーソルの状態が正しくありません。

    トリガ内でSQL文を実行する前に以下の文を1文追加してください。
    SET NOCOUNT ON

15. **データ例外のエラー発生時の調査方法を教えてください。**

    以下のいずれかの方法で調査してください。

ー エラーが発生した際のレコード内容を、別ファイルに出力してデータ内容を調査してください。

ー エラー発生時のデータ内容を出力するようトレース情報を採取して、データ内容を調査してください。

16. **排他エラーが発生したとき、獲得しているアプリケーション(プロセス)の調査方法を教えてください。**

SQL ServerのSQLプロファイラで、以下のように指定しトレースを採取し、調査してください。

ー 「使用できるイベントクラス」から「TSQL」、「ストアドプロシジャ」、「ロック」を選択してください。

ー 「データ列」に、EventClass、TextData、ApplicationName、ClientProcessID、SPID、Mode BinaryData(レコードを識別できるためのシステムID)を追加してください。

ー 「フィルタ」に、ApplicationName Like "PowerRDBconnector"を設定し、「システムIDを除外」のプロパティを指定してください。

なお、セッションIDを特定する場合は、トレース出力機能を使用してください。

17. **64bitのWindowsOSでCOBOLアプリケーションを実行すると、「ハンドルされていない例外」のエラーが発生し、動作しません。**

NetCOBOLのコンパイルオプション(/platform)を指定し作成されたアプリケーションだけ、64bitのWindowsOSで動作します。アプリケーション作成時のコンパイルオプションを確認してください。

18. **OPEN文を実行すると、"クエリタイムアウトが時間切れになりました"というエラーが発生します。**

対応するインデックスが定義されておらず、かつレコード件数が大量にある表に対して、索引ファイルとしてOPEN文、START文やキー指定のアクセス命令を実行した場合に、SQL Serverの処理に、TimeOutプロパティで指定した時間以上の時間がかかっているためです。インデックスを定義するか、TimeOutプロパティに大きな値を指定してください。

19. **OPEN文を実行すると、FILE STATUSに90が発生していますが、イベントログにはエラー情報が出力されていません。**

トレース情報を採取して、アクセス対象表の項目構成と、COBOLプログラムを突き合わせて、COBOLプログラムと、表またはビュー表のレコード長が異なっているか確認してください。

固定少数点を使用する場合は、COBOLとデータベース間で、桁数の表現が違うことに注意してください。

ー COBOLのPIC S9(9)V9(2) PACKED-DECIMAL.は、DECIMAL(9,2)ではなく、DECIMAL(11,2)となります。

20. **ASP.NETで使用し、COB_PRDB_STARTサブルーチンを実行すると、復帰値が-1、iserrno=255、isstat1='0'、isstat2=0x0、isstat3='1'、isstat4=0x0、"DBIO_ENV file open error"のエラーが発生します。**

セッション開設時に、PowerRDBconnector動作環境ファイルが見つからないため、エラーが発生しています。カレントパスの設定を確認してください。

21. **64bitのWindowsOSで、ASP.NETを使用したCOBOLアプリケーションを実行すると以下のエラーが発生して動作しません。**
JMP0097I-U ランタイムシステムが正しくインストールされていません.
'NOT-INSTALLED' Fujitsu.COBOL.Runtime.Subroutines.PRDBconnector
説明: 現在の Web 要求を実行中に、ハンドルされていない例外が発生しました。

IISで動作させるアプリケーションが32ビット互換モードで動作していないために発生しています。以下の手順で、IISで動作させるアプリケーションを32ビット互換モードで動作させてください。

1. 次のコマンドを入力して 32 ビット互換モードを有効にします。

   cscript %SYSTEMDRIVE%¥inetpub¥adminscripts¥adsutil.vbs
   SET W3SVC/AppPools/Enable32bitAppOnWin64 1

2. 次のコマンドを入力して ASP.NET 2.0 (32ビット版) をインストールし、スクリプト マップをIIS ルート下にインストールします。

   %SYSTEMROOT%¥Microsoft.NET¥Framework¥v2.0.xxxxx¥aspnet_regiis.exe -i

3. インターネット インフォメーション サービス マネージャのWeb サービス拡張の一覧で、ASP.NET version 2.0.xxxxx (32 ビット版) の状態が [許可] に設定されていることを確認します。

ただし、この場合、IISで動作させるアプリケーションはすべて32ビット互換モードで動作します。

22. **インストール時に指定したデータベース種別を調べる方法を教えてください。**

[システムのプロパティ]の[詳細設定]の画面にある[環境変数]を表示し、システム環境変数"RDBCONNECTOR"(32ビット)または、"RDBCONNECTOR64"(64ビット)の値により、以下のようにインストールしたデータベース種別がわかります。

ー SQLSV2005:SQL Server 2005

- SQLSV2008：SQL Server 2008
- ORA10g：Oracle10g
- ORA11g：Oracle11g

23. **COBOL初期化ファイルには、テーブル名を正しく指定していますが、OPEN文でエラーとなります。調査方法を教えてください。**

    COBOL初期化ファイルがUTF-8のコード系で作成されている場合、テーブル名にシフトJIS範囲外の文字が指定されている可能性があります。PowerRDBconnectorのトレース機能を使用して、以下のように確認してください。

    - シフトJIS範囲内の文字でテーブル名が指定されている場合

      トレースログの出力結果に、以下のような文字化けを起こさない情報が出力されています。

      cw_FileOpenInfo() ; ------- : TableName=M_FILE02&SchemaName=RDBCTEST&AccessMode=DYNAMIC

    - シフトJIS範囲外の文字でテーブル名が指定されている場合

      トレースログの出力結果に、以下のような文字化けを起こす情報が出力されています。

      cw_FileOpenInfo() ; ------- : TableName=**?????**&SchemaName=RDBCTEST&AccessMode=DYNAMIC

    PowerRDBconnectorのトレース機能については、「5.2 トレース出力機能」を参照してください。

24. **Windowsファイアウォールを有効にすると、エラーが発生します。**

    Windowsファイアウォールを有効にするとSQL Serverの既定ポートが許可されていません。そのため、クライアントサーバ形態でデータベースに接続を行うと、以下の接続エラーが発生します。

    エラー時のイベントログ内容

    iserrno = 6 isstat1 = '0' isstat2 = 0x0 isstat3 = '0' isstat4 = 0x0

    [Microsoft][SQL Native Client]名前付きパイプのプロバイダ : SQL Server への接続を開けませんでした [1326].

    このような場合、Windowsファイアウォールのポート番号設定を確認し、データベースのポートを許可してください。

25. **NetCOBOL for Windows の実行コード系がunicode のCOBOLアプリケーションで、日本語項目の文字判定が正しく行えません。**

    PowerRDBconnectorとCOBOLランタイム間で、日本語項目の後方空白の半角/全角の扱いが一致していない可能性があります。後方空白の扱いの詳細は、「3.7.1 後方空白補正」を参照してください。

    なお、PowerRDBconnector のトレース機能を使用して、日本語項目に設定されている空白の種類を確認できます。

    PowerRDBconnectorのトレース機能については、「5.2 トレース出力機能」を参照してください。

    またNetCOBOL for Windowsでunicodeを使用する場合の詳細は、NetCOBOLのマニュアルを参照してください。

26. **OPEN文でFILE STATUS=90、iserrno=22、isstat1='9'、isstat2=0x34、isstat3='9'、isstat4=0x34のエラーが発生します。原因を教えてください。**

    表の列名に、PowerRDBconnectorで既定しているサフィックス文字が付加されていない場合に発生します。

    列名の定義方法については、「3.1.3.5 列定義の対応」を参照してください。

27. **OPEN文でFILE STATUS=90、iserrno=6、isstat1='0'、isstat2=0x00、isstat3='0'、isstat4=0x00のエラーが発生します。原因を教えてください。**

    Microsoft SQL Server Native Client ドライバがインストールされていない可能性があります。

    SQL Serverのメディアから、Microsoft SQL Server Native Client ドライバをインストールしてください。

28. **START後に順READを行うと1レコードしか読み込めません。**

    COBOL初期化ファイルのAccessModeに、DYNAMICと定義すべきところを誤ってRANDOMと定義している可能性があります。

29. **32ビット版のPowerRDBconnectorで使用していたCOBOL初期化ファイルを、64ビット版のPowerRDBconnectorで使用すると、OPEN時にエラーが発生します。**

    COBOL初期化ファイルのAccessModeプロパティの指定とCOBOLプログラム内のAccessMode句が一致していないと、OPEN時にエラーが発生します。

AccessModeプロパティの指定をCOBOLプログラム内のAccessMode句と同じ設定にしてください。

なお、32ビット動作では、AccessModeプロパティとCOBOLプログラム内のAccessMode句が一致していない場合、動作保証されません。このため、64ビット動作に移行したCOBOLアプリケーションは32ビット動作のときと動作が異なる場合があります。

B.2.3 性能

1. **レコードの項目数と、OPEN文の性能に依存関係はありますか？**

   項目数が多くなるに従い、OPEN処理の性能が遅くなります。PowerRDBconnectorでは、OPEN文の延長でデータベースからカラム情報を取得しているためです。

2. **OPEN文が遅かったり、OPEN文を発行したときに他のアプリケーションの動作が遅くなったりする場合があります。**

   PowerRDBconnector動作環境ファイル内でPrepareModeプロパティにOPENが指定されていると、OPEN文の延長で使用する可能性のあるSQL文をすべて準備します。特に、下記の条件でSQL文の準備に時間がかかるとともに、データベース内で大量のメモリ獲得が発生し、メモリ負荷が発生する場合があります。

   ー RECORD KEYやALTERNATE RECORD KEYを構成する項目(列)数の多い表に対してOPEN文を発行した場合

   PowerRDBconnector動作環境ファイル内で、PrepareModeプロパティにSTARTを指定し、SQL文の準備を初回のキー検索(START、乱READ)時に行うよう変更することで、改善できる場合があります。

3. **OPEN、START、READ、REWRITE、WRITE性能が極めて遅いのですが、性能のチューニング方法を教えてください。**

   PowerRDBconnectorは、以下の設定を行うことで最適な性能が得られます（この設定を行わないと、性能が大幅に劣化します）。詳しくは、「4.1.4 性能向上のポイント」を参照してください。

   ー インデックスを設定してください。

   ー インデックスを定期的にメンテナンスしてください。

   ー クラスタ化インデックスに変更することで、性能を改善できる場合があります。

   ー RECORD KEY句にあたる列にインデックスを設定しても、他のインデックス(クラスタ化インデックス)を使用したアクセスが行われてしまい、最適な性能とならない場合があります。この場合、クラスタ化インデックスを非クラスタ化インデックスに変更することで、改善できる場合があります。

     - プライマリキー用のインデックスは、暗黙でクラスタ化インデックスとして作成されるため、プライマリキー用のインデックスを使用したアクセスが行われてしまう場合があります。

4. **I-OモードでOPENした場合のREAD性能が遅いのですが、性能のチューニング方法を教えてください。**

   PowerRDBconnectorが提供するレコードロック機能(暗黙)でなく、テーブルロック機能を使用すると、READ文のレコードロック処理を迂回することができ、READ文の性能を数倍程度、高速化することができます。詳しくは、「4.1.4 性能向上のポイント」を参照してください。

5. **ASP(RDB/6000、Symfoware6000)のSETPFコマンドをCOBOLプログラムに書き換えたところ、性能が遅くなりました。**

   ASP(RDB/6000、Symfoware6000)のSETPFコマンドは複数データを一括処理することで性能を確保しています。これをCOBOLプログラムに書き換えると大量にREADやWRITEを発行することとなるため、性能が遅くなります。大量データのロードを行うときには、データベース製品のユーティリティの活用を検討してください。データベース製品のユーティリティについては、「4.1.4.4 データベース・ユーティリティの活用」を参考にしてください。

6. **レコードキーを構成する項目数と、OPEN文の性能に依存関係はありますか？**

   PowerRDBconnector動作環境ファイル内でPrepareModeプロパティに“OPEN”が指定されていると、OPEN文の延長で使用する可能性のあるSQL文をすべて準備します。
   このため、レコードキーを構成する項目数が多くなるに従い、OPEN処理の性能が遅くなります。

   OPEN文の延長でPowerRDBconnectorが準備するSQL文の数が多くなるためです。

   ただし、PrepareModeプロパティに“START”を指定、またはPrepareModeプロパティを省略すると、OPEN文を実行した後の最初のSTART文、乱READ文、乱REWRITE文、または乱DELETE文の性能が遅くなります。

7. **レコードを構成する項目数と、READ文やWRITE文の性能に依存関係はありますか？**

   レコードを構成する項目数が多くなるに従い、READ文やWRITE文の性能が遅くなります。

   項目ごとに内容のチェックや、データ補正などの処理が行われるためです。

動作環境にもよりますが、項目数を半分（50%）にすると、性能が約40%向上する場合もあります。

付録C　開発用サンプル情報

本章では、PowerRDBconnectorを使用する際に役立つサンプルを示します。

注）サンプル使用時の注意事項

各サンプルが複数ページにまたがることがありますので、カット＆ペーストして使用する際には、ページ番号までコピーされる点に注意してください。

C.1　NetCOBOL for .NETのプログラム原型

本節では、NetCOBOL for .NETで使用するサブルーチンのプログラム原型のサンプルを示します。使い方の詳細は、NetCOBOLのマニュアルを参照してください。

C.1.1　XMROTSTRサブルーチン

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID.          XMROTSTR AS "XMROTSTR"
000030                      IS PROTOTYPE CUSTOM-ATTRIBUTE IS PTRN.
000040
000050 ENVIRONMENT              DIVISION.
000060 CONFIGURATION             SECTION.
000070 SPECIAL-NAMES.
000080    CONSOLE        IS      CONSL
000090    CUSTOM-ATTRIBUTE PTRN
000100        CLASS DLLIMPORT USING "F3BWS1CB.DLL"
000110          PROPERTY P-CALLINGCONVENTION IS STDCALL OF E-CALLINGCONVENTION
000120          PROPERTY CHARSET IS ANSI OF E-CHARSET.
000130 REPOSITORY.
000140    CLASS DLLIMPORT AS "System.Runtime.InteropServices.DllImportAttribute"
000150    ENUM E-CALLINGCONVENTION
000160       AS "System.Runtime.InteropServices.CallingConvention"
000170    PROPERTY P-CALLINGCONVENTION AS "CallingConvention"
000180    PROPERTY STDCALL AS "StdCall"
000190    PROPERTY CHARSET AS "CharSet"
000200    ENUM E-CHARSET AS "System.Runtime.InteropServices.CharSet"
000210    PROPERTY ANSI AS "Ansi"
000220    CLASS SYS-STRING AS "System.String".
000230 END PROGRAM XMROTSTR.
```

C.1.2　XMROTENDサブルーチン

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID.          XMROTEND AS "XMROTEND"
000030                      IS PROTOTYPE CUSTOM-ATTRIBUTE IS PTRN.
000040
000050 ENVIRONMENT              DIVISION.
000060 CONFIGURATION             SECTION.
000070 SPECIAL-NAMES.
000080    CONSOLE        IS      CONSL
000090    CUSTOM-ATTRIBUTE PTRN
000100        CLASS DLLIMPORT USING "F3BWS1CB.DLL"
000110          PROPERTY P-CALLINGCONVENTION IS STDCALL OF E-CALLINGCONVENTION
000120          PROPERTY CHARSET IS ANSI OF E-CHARSET.
000130 REPOSITORY.
000140    CLASS DLLIMPORT AS "System.Runtime.InteropServices.DllImportAttribute"
000150    ENUM E-CALLINGCONVENTION
```

```
000160      AS "System.Runtime.InteropServices.CallingConvention"
000170   PROPERTY P-CALLINGCONVENTION AS "CallingConvention"
000180   PROPERTY STDCALL AS "StdCall"
000190   PROPERTY CHARSET AS "CharSet"
000200   ENUM E-CHARSET AS "System.Runtime.InteropServices.CharSet"
000210   PROPERTY ANSI AS "Ansi"
000220   CLASS SYS-STRING AS "System.String".
000230 END PROGRAM XMROTEND.
```

C.1.3 XMROTCNLサブルーチン

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID.          XMROTCNL AS "XMROTCNL"
000030                  IS PROTOTYPE CUSTOM-ATTRIBUTE IS PTRN.
000040
000050 ENVIRONMENT             DIVISION.
000060 CONFIGURATION           SECTION.
000070 SPECIAL-NAMES.
000080   CONSOLE       IS      CONSL
000090   CUSTOM-ATTRIBUTE PTRN
000100       CLASS DLLIMPORT USING "F3BWS1CB.DLL"
000110         PROPERTY P-CALLINGCONVENTION IS STDCALL OF E-CALLINGCONVENTION
000120         PROPERTY CHARSET IS ANSI OF E-CHARSET.
000130 REPOSITORY.
000140   CLASS DLLIMPORT AS "System.Runtime.InteropServices.DllImportAttribute"
000150   ENUM E-CALLINGCONVENTION
000160     AS "System.Runtime.InteropServices.CallingConvention"
000170   PROPERTY P-CALLINGCONVENTION AS "CallingConvention"
000180   PROPERTY STDCALL AS "StdCall"
000190   PROPERTY CHARSET AS "CharSet"
000200   ENUM E-CHARSET AS "System.Runtime.InteropServices.CharSet"
000210   PROPERTY ANSI AS "Ansi"
000220   CLASS SYS-STRING AS "System.String".
000230 END PROGRAM XMROTCNL.
```

C.1.4 XMROTRBKサブルーチン

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID.          XMROTRBK AS "XMROTRBK"
000030                  IS PROTOTYPE CUSTOM-ATTRIBUTE IS PTRN.
000040
000050 ENVIRONMENT             DIVISION.
000060 CONFIGURATION           SECTION.
000070 SPECIAL-NAMES.
000080   CONSOLE       IS      CONSL
000090   CUSTOM-ATTRIBUTE PTRN
000100       CLASS DLLIMPORT USING "F3BWS1CB.DLL"
000110         PROPERTY P-CALLINGCONVENTION IS STDCALL OF E-CALLINGCONVENTION
000120         PROPERTY CHARSET IS ANSI OF E-CHARSET.
000130 REPOSITORY.
000140   CLASS DLLIMPORT AS "System.Runtime.InteropServices.DllImportAttribute"
000150   ENUM E-CALLINGCONVENTION
000160     AS "System.Runtime.InteropServices.CallingConvention"
000170   PROPERTY P-CALLINGCONVENTION AS "CallingConvention"
000180   PROPERTY STDCALL AS "StdCall"
000190   PROPERTY CHARSET AS "CharSet"
000200   ENUM E-CHARSET AS "System.Runtime.InteropServices.CharSet"
```

```
000210    PROPERTY ANSI AS "Ansi"
000220    CLASS SYS-STRING AS "System.String".
000230 END PROGRAM XMROTRBK.
```

C.1.5 XMROAUTHサブルーチン

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID.            XMROAUTH AS "XMROAUTH"
000030                        IS PROTOTYPE CUSTOM-ATTRIBUTE IS PTRN.
000040
000050 ENVIRONMENT              DIVISION.
000060 CONFIGURATION             SECTION.
000070 SPECIAL-NAMES.
000080    CONSOLE         IS       CONSL
000090    CUSTOM-ATTRIBUTE PTRN
000100        CLASS DLLIMPORT USING "F3BWS1SB.DLL"
000110          PROPERTY P-CALLINGCONVENTION IS STDCALL OF E-CALLINGCONVENTION
000120          PROPERTY CHARSET IS ANSI OF E-CHARSET.
000130 REPOSITORY.
000140    CLASS DLLIMPORT AS "System.Runtime.InteropServices.DllImportAttribute"
000150    ENUM E-CALLINGCONVENTION
000160       AS "System.Runtime.InteropServices.CallingConvention"
000170    PROPERTY P-CALLINGCONVENTION AS "CallingConvention"
000180    PROPERTY STDCALL AS "StdCall"
000190    PROPERTY CHARSET AS "CharSet"
000200    ENUM E-CHARSET AS "System.Runtime.InteropServices.CharSet"
000210    PROPERTY ANSI AS "Ansi"
000220    CLASS SYS-STRING AS "System.String".
000230 DATA DIVISION.
000240
000250 LINKAGE SECTION.
000260 01 USERINFO.
000270 02 AUTH PIC 9(9) COMP-5.
000280 02 USERNNO PIC 9(9) COMP-5.
000290 02 USERN PIC X(260).
000300 02 PASSWNO PIC 9(9) COMP-5.
000310 02 PASSW PIC X(260).
000320 01 ERRINFO.
000330 02 ENDINFO PIC S9(9) COMP-5.
000340 02 DITINFO PIC S9(9) COMP-5.
000350 02 FILLER PIC S9(9).
000360 01 RET-VALUE PIC S9(9) COMP-5.
000370 PROCEDURE DIVISION USING BY REFERENCE USERINFO ERRINFO RETURNING RET-
VALUE.
000380 END PROGRAM XMROAUTH.
```

C.2 マニュアル内で使用した図表に対するテキスト形式の雛型

本節は、本マニュアル内の図表のうち、カット&ペーストで活用できるテキスト形式の雛型を載せています。

各雛型の記載形式の詳細は、本マニュアル内の説明箇所を参照してください。

C.2.1 PowerRDBconnector動作環境ファイルのサンプル

以下のサンプルは、必要に応じて、< >内に値を設定したり、コメントを解除したりして使用してください。

詳細は、「3.1.4 PowerRDBconnector 動作環境ファイル」を参照してください。

```
; PowerRDBconnectorの動作環境
ServerName=<データベースのサーバ名>
DataSourceName=<データベース名>
; TimeOut=<タイムアウト時間>
; Suppress=<後方空白補正>
; DataCheck=<データチェック>
; PrepareMode=<SQL文準備モード>
; ErrorLog=<エラーログの出力先>
; TraceMode=<トレースモード>
; TraceSize=<トレースファイルのサイズ>
; TraceLevel=<トレースのレベル>
```

C.2.2 COBOL初期化ファイルのサンプル

以下のサンプルは、必要に応じて名称などを追加、変更して使用してください。

詳細は、「3.1.5 COBOL初期化ファイル」を参照してください。

```
EMPLOYEE=TableName=employee&SchemaName=dbo&AccessMode=RANDOM&Suppress=OFF,RDM
CUSTOMER=TableName=customer&SchemaName=dbo&AccessMode=RANDOM&Suppress=OFF,RDM
@CBR_ENTRYFILE=ENTRY.ENT
```

C.2.3 エントリ情報ファイルのサンプル

以下のサンプルは、必要に応じて名称などを追加、変更して使用してください。

詳細は、「3.1.5 COBOL初期化ファイル」を参照してください。

- 32ビット動作時
  - シングルセッション時

```
[ENTRY]
;トランザクションサブルーチン
XMROTSTR=F3BWS1CB.DLL
XMROTEND=F3BWS1CB.DLL
XMROTCNL=F3BWS1CB.DLL
XMROTRBK=F3BWS1CB.DLL
; 認証情報登録サブルーチン
XMROAUTH=F3BWS1SB.DLL
```

  - マルチセッション時

```
[ENTRY]
;セッションサブルーチン
```

```
COB_PRDB_START=F3BIEFNC.dll
COB_PRDB_END=F3BIEFNC.dll
COB_PRDB_CHG=F3BIEFNC.dll
;トランザクションサブルーチン
COB_PRDB_TRAN=F3BIEFNC.dll
; 認証情報登録サブルーチン
COB_PRDB_AUTH=F3BIEFNC.dll
```

・ 64ビット動作時

ー シングルセッション時

```
[ENTRY]
;トランザクションサブルーチン
XMROTSTR=F4ARS1CB_64.DLL
XMROTEND=F4ARS1CB_64.DLL
XMROTCNL=F4ARS1CB_64.DLL
XMROTRBK=F4ARS1CB_64.DLL
; 認証情報登録サブルーチン
XMROAUTH=F4ARS1SB_64.DLL
; トレース文字出力サブルーチン
XMROLOG=F4ARS1SB_64.DLL
```

ー マルチセッション時

```
[ENTRY]
;セッションサブルーチン
COB_PRDB_START=F4AGEFNC.dll
COB_PRDB_END=F4AGEFNC.dll
COB_PRDB_CHG=F4AGEFNC.dll
;トランザクションサブルーチン
COB_PRDB_TRAN=F4AGEFNC.dll
; 認証情報登録サブルーチン
COB_PRDB_AUTH=F4AGEFNC.dll
; トレース文字出力サブルーチン
COB_PRDB_LOG=F4AGEFNC.dll
```

C.2.4　セッションサブルーチンのサンプル

以下のサンプルは、必要に応じて名称などを追加、変更して使用してください。

詳細は、「3.2 セッションの制御方法」を参照してください。

```
DATA DIVISION.
WORKING-STORAGE  SECTION.
01 SESSION-ID   PIC X(30) VALUE SPACE.
01 RET-VALUE    PIC S9(9) COMP-5.
01 RET          PIC S9(9) COMP-5 VALUE 0.

＜略＞

PROCEDURE DIVISION.

MOVE "SESSION01" TO SESSION-ID.
* セッションを開設する

CALL "COB_PRDB_START" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.


OPEN I-O INFILE.


＜略＞


CLOSE INFILE.

* セッションを閉設する。

CALL "COB_PRDB_END" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.
```

C.2.5　認証情報登録サブルーチンのサンプル

以下のサンプルは、必要に応じて名称などを追加、変更して使用してください。

シングルセッションの場合

詳細は、「3.3.2 データベース認証の使用方法(シングルセッション)」を参照してください。

```
DATA   DIVISION.
WORKING-STORAGE   SECTION.
01 USERINFO.
 02 AUTH        PIC 9(9) COMP-5 VALUE 0.
 02 USERNNO     PIC 9(9) COMP-5 VALUE 0.
 02 USERN       PIC X(260) VALUE SPACE.
 02 PASSWNO     PIC 9(9) COMP-5 VALUE 0.
 02 PASSW       PIC X(260) VALUE SPACE.
01 ERRINFO.
 02 ENDINFO     PIC S9(9) COMP-5 VALUE 0.
 02 DITINFO     PIC S9(9) COMP-5 VALUE 0.
 02 FILLER      PIC S9(9) VALUE 0.
01 RET          PIC S9(9) COMP-5 VALUE 0.
＜略＞
PROCEDURE DIVISION.
* データベース認証にする
MOVE 2 TO AUTH.
MOVE "SYSTEM100" TO USERN.
MOVE 9 TO USERNNO.
MOVE "SYSUSER" TO PASSW.
```

```
MOVE 7 TO PASSWNO.
CALL "XMROAUTH" USING USERINFO ERRINFO RETURNING RET.
MOVE SPACE TO USERN.
MOVE SPACE TO PASSW.

＜略＞
```

マルチセッションの場合

詳細は、「3.3.3 データベース認証の使用方法(マルチセッション)」を参照してください。

```
WORKING-STORAGE SECTION.
   01 SESSION-ID   PIC X(30).
   01 RET-VALUE    PIC S9(9) COMP-5.
   01 S-CODE       PIC X.
01 USERINFO.
   02 AUTH        PIC 9(9) COMP-5 VALUE 0.
   02 USERNNO     PIC 9(9) COMP-5 VALUE 0.
   02 USERN       PIC X(260) VALUE SPACE.
   02 PASSWNO     PIC 9(9) COMP-5 VALUE 0.
   02 PASSW       PIC X(260) VALUE SPACE.
01 ERRINFO.
   02 ENDINFO     PIC S9(9) COMP-5 VALUE 0.
   02 DITINFO     PIC S9(9) COMP-5 VALUE 0.
   02 FILLER      PIC S9(9) VALUE 0.
～～
* セッションの開始
CALL "COB_PRDB_START" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.

* 認証情報の登録
MOVE 2 TO AUTH.
MOVE "SYSTEM100" TO USERN.
MOVE 9 TO USERNNO.
MOVE "SYSUSER" TO PASSW.
MOVE 7 TO PASSWNO.
CALL "COB_PRDB_AUTH" USING BY REFERENCE SESSION-ID
   BY REFERENCE USERINFO BY REFERENCE ERRINFO RETURNING RET-VALUE.
MOVE SPACE TO USERN.
MOVE SPACE TO PASSW.

＜略＞

OPEN I-O FILE1.
＜略＞

CLOSE FILE1.

* セッションの閉設
CALL "COB_PRDB_END" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.
```

C.2.6 トランザクションサブルーチンのサンプル

以下のサンプルは、必要に応じて名称などを追加、変更して使用してください。

マルチセッションの場合

詳細は、「3.4.3 トランザクションの使用方法(マルチセッション)」を参照してください。

```
WORKING-STORAGE SECTION.
   01 SESSION-ID   PIC X(30).
   01 RET-VALUE    PIC S9(9) COMP-5.
```

```
   01 S-CODE       PIC 9(9) COMP-5.
～～
MOVE "SESSION01" TO SESSION-ID.
* セッションの開設
CALL  “COB_PRDB_START”  USING BY REFERENCE SESSION-ID
RETURNING RET-VALUE.

＜略＞


OPEN I-O FILE1.

MOVE 1 TO S-CODE.
* トランザクションの開始
CALL "COB_PRDB_TRAN" USING BY REFERENCE SESSION-ID
     BY VALUE S-CODE RETURNING RET-VALUE.
＜略＞


MOVE 2 TO S-CODE.
* トランザクションの確定
CALL "COB_PRDB_TRAN" USING BY REFERENCE SESSION-ID
     BY VALUE S-CODE RETURNING RET-VALUE.

または


MOVE 3 TO S-CODE.
* トランザクションの取消し
CALL "COB_PRDB_TRAN" USING BY REFERENCE SESSION-ID
     BY VALUE S-CODE RETURNING  RET-VALUE.
＜略＞


CLOSE FILE1.


* セッションの閉設
CALL "COB_PRDB_END" USING BY REFERENCE SESSION-ID RETURNING RET-VALUE.
```

付録D　データベースの相違点

本章では、PowerRDBconnectorからSQL ServerとOracleにアクセスする場合の相違点について説明します。

データベースが持つ固有の機能やツールなどの相違点については、データベースそれぞれのマニュアルを参照してください。

D.1 機能の相違点

表D.1 ファイルアクセス機能

編成 ORGANIZATION	アクセスモード ACCESS MODE	SQL Server	Oracle
SEQUENTIAL（順ファイル）	SEQUENTIAL(順呼出し)	テーブル ビュー	テーブル ビュー シノニム
INDEXED（索引ファイル）	SEQUENTIAL(順呼出し) RANDOM(乱呼出し) DYNAMIC(動的呼出し)	テーブル ビュー インデックス	テーブル ビュー インデックス シノニム

表D.2 COBOLアプリケーションの機能範囲

項目			SQL Server	Oracle
レコード	レコードの最大長(バイト数)		8,060	32,760
ファイル 管理記述項	ALTERNATE RECORD KEY句	指定できるキーの最大個数	16 (注1)	99 (注1)
排他制御(注2)	対象	テーブルロック	○	○
		レコードロック	○	○
	ロック解放タイミング 非トランザクション時、REWRITE文での解放		×	○
	トランザクション開始をまたがるロック		×	○
	トランザクション終了/取消しをまたがるロック		○	×
最大合計列名長（1つのテーブル/ビューに対する全列名の合計サイズ）			40,960(注3)	－

○:使用可能

×:使用不可

－:該当項目なし

（注1）ALTERNATE RECORD KEY句

指定できるキーの最大個数は、RECORD KEY句（主キー）を含めた数になります。

（注2）排他制御

テーブルロックまたはレコードロックを指定します。
レコードロックは、PowerRDBconnector動作環境ファイルの指定により行います。
テーブルロックは、COBOL初期化ファイルの指定と、PowerRDBconnector動作環境ファイルの指定により行います。

（注3）最大合計列名長（SQL Serverのみ）

テーブルおよびビュー作成時、列名の長さの合計が、半角文字で40,960文字、全角文字の場合は20,480文字以下にしてください。

ここでは、COBOLの入出力文で記述する機能範囲をデータベースの機能に置き換えて説明します。詳細は、データベースのマニュアルを参照してください。

表D.3 データベースの機能範囲

項目			SQL Server	Oracle
仮想マシン動作	Hyper-V		○	×
64ビットOS	64ビット動作		○	○
	32ビット動作		○	○ Oracleクライアント(32ビット) が必要
	DBMS（32ビット版）		△	△
	DBMS（64ビット版）		○	○
リモートデスクトップ接続形態	ターミナルサービス、ターミナルサーバー		○	○
	上記以外		○	×
ファイル定義ツール			osql、sqlcmd、SQL Server Management Studioなど	SQL*Plusなど
ファイル	ファイルの最大サイズ(バイト)		32T	無制限
	ファイルの最大数		約20億	無制限
	別名		△ シノニム	○ シノニム
	他データベースへのリンク		△ リンクサーバー	△ データベース・リンク
レコード	レコードの最大長(バイト)		8,060	2,000,000 (注1)
	レコード内の最大項目数（列数）		1,024	1,000
	レコードの最大数（行数）		無制限	無制限
ファイル管理記述項	RECORD KEY句	指定できる項目の最大個数	16	32
		指定できる項目総長の最大(バイト)	900	設定依存(注2)
	ALTERNATE RECORD KEY句	指定できるキーの最大個数	無制限	無制限
		指定できる項目の最大個数	16	32
		指定できる項目総長の最大(バイト)	900	設定依存
	多重項目キーに定義できる最大列数		16	32
1テーブルに定義できるインデックス数			249	無制限
1セッションで可能な最大オープン数			不明(注3)	設定依存(注3)

○:使用可能

△:データベースではサポートしていますが、PowerRDBconnectorでは未サポートです。

×:使用不可

（注1）レコードの最大長(バイト)

Oracleのサポートする行長は32,760バイトより大きいですが、PowerRDBconnectorのサポートする最大レコード長（32,760バイト）に制限されます。

（注2）キー指定できる項目総長の最大(バイト)

Oracleのデータ・ブロックのサイズ（初期化パラメーター DB_BLOCK_SIZE）に依存します。

（注3）1セッションで可能な最大オープン数

SQL Server：SQL Serverの技術資料を参照してください。

Oracle：Oracleの初期化パラメーター OPEN_CURSORS に依存します。

D.2 インストール時の相違点

表D.4 インストール時の相違点

分類			機能	SQL Server	Oracle
セットアップ	インストール時のデータベース種別の選択		32ビット	SQL Server 2005 SQL Server 2008	Oracle Database 10g Oracle Database 11g
			64ビット	SQL Server 2008	Oracle Database 11g
	インストール時に設定される項目	RDBCONNECTOR環境変数	インストールディレクトリ¥DB種毎のディレクトリ DB種毎のディレクトリは以下の通り。		
			32ビット	・SQL Server 2005 SQLSV2005 ・SQL Server 2008 SQLSV2008	・Oracle 10g ORA10g ・Oracle 11g ORA11g
			64ビット	・SQL Server 2008 SQLSV2008	・Oracle 11g ORA11g
データベース製品のインストール	インストール時の設定			特になし	Oracle Call Interface（OCI）
	エンジン以外のインストール		32ビット	Microsoft SQL Server Native Clientドライバ	クライアント/サーバ形態または、64ビットデータベース使用時 ・クライアント(32ビット)製品 ・Oracle Client (Oracle Net Services)
			64ビット		クライアント/サーバ形態使用時 ・クライアント製品 ・Oracle Client (Oracle Net Services)

D.3 列定義の相違点

表D.5 COBOL定義とデータベース列定義の対応

COBOL定義			データベース定義		
項目の種類	USAGE句	符号	列名	SQL Serverのデータ型	Oracleのデータ型
数字項目	DISPLAY（外部10進）	なし	<文字列>**_UNSIGN_NUMERIC** または <文字列>**_UNUM**	NUMERIC（p，s）	NUMBER(p,s)
		あり	<文字列>**_NUMERIC** または <文字列>**_NUM**	NUMERIC（p，s）	NUMBER(p,s)
	COMP-3、COMPUTATIONAL-3、PACKED-DECIMAL（内部10進）	なし	<文字列>**_UNSIGN_DECIMAL** または <文字列>**_UDEC**	DECIMAL（p，s）	NUMBER(p,s)
		あり	<文字列>**_DECIMAL** または <文字列>**_DEC**	DECIMAL（p，s）	NUMBER(p,s)
整数項目	COMP、COMPUTATIONAL、BINARY（2進）	なし	<文字列>**_UNSIGN_SMALLINT** または <文字列>**_USINT**	SMALLINT	NUMBER(p,0)
			<文字列>**_UNSIGN_INTEGER** または <文字列>**_UINT**	INTEGER	NUMBER(p,0)
			<文字列>**_UNSIGN_BIGINT** または <文字列>**_UBINT**	BIGINT	NUMBER(p,0)
		あり	<文字列>**_SMALLINT** または <文字列>**_SINT**	SMALLINT	NUMBER(p,0)
			<文字列>**_INTEGER** または <文字列>**_INT**	INTEGER	NUMBER(p,0)
			<文字列>**_BIGINT** または <文字列>**_BINT**	BIGINT	NUMBER(p,0)
英数字項目	X（英数字）	-	<文字列>**_CHAR** または <文字列>**_CHR**	CHAR(p)	CHAR(p)
				NCHAR(p)	NCHAR(p)
			<文字列>**_CHAR_<項目長>** または <文字列>**_CHR_<項目長>** または <文字列>**_CHAR** または <文字列>**_CHR**	VARCHAR(p)	VARCHAR2(p)
				NVARCHAR(p)	NVARCHAR2(p)
日本語項目	N（日本語）	-	<文字列>**_NCHAR** または <文字列>**_NCHR**	CHAR（p×2）	・シフトJIS CHAR(p×2) ・unicode CHAR(p×3)
				NCHAR（p）	NCHAR(p)

COBOL定義			データベース定義		
項目の種類	USAGE句	符号	列名	SQL Serverのデータ型	Oracleのデータ型
			<文字列>**_NCHAR_<項目長>** または <文字列>**_NCHR_<項目長>** または <文字列>**_NCHAR** または <文字列>**_NCHR**	VARCHAR(p×2)	・シフトJIS VARCHAR2(p×2) ・unicode VARCHAR2(p×3)
				NVARCHAR(p)	NVARCHAR2(p)
バイナリ項目	X(バイナリ)	-	<文字列>**_BINARY** または <文字列>**_BIN**	BINARY(p)	RAW(p)
			<文字列>**_BINARY_<項目長>** または <文字列>**_BIN_<項目長>** または <文字列>**_BINARY** または <文字列>**_BIN**	VARBINARY(p)	-
			<文字列>**_BINARY_<項目長>** または <文字列>**_BIN_<項目長>**	-	BLOB

精度(p) :精度は、全体桁数および項目の長さ(文字数)です。

位取り(s):位取りは、小数部の桁数です。

文字列:任意の文字列です。

項目長:COBOLでの項目長です。

表D.6 データベースの文字列データの精度

項目の種類	SQL Server		Oracle	
	データ型	サイズ	データ型	サイズ
英数字項目	CHAR	8,000バイト	CHAR	2,000バイト
	VARCHAR	8,000バイト	VARCHAR2	4,000バイト
日本語項目	NCHAR	8,000バイト	NCHAR	2,000バイト
	NVARCHAR	8,000バイト	NVARCHAR2	4,000バイト
数字項目/整数項目	SMALLINT	2バイト	NUMBER	38桁
	INT	4バイト		
	BIGINT	8バイト		
	DECIMAL	38桁		
	NUMERIC	38桁		
バイナリ項目	BINARY	8,000バイト	RAW	2,000バイト
	VARBINARY	8,000バイト	BLOB	2ギガバイト

D.4 コード系の相違点

表D.7 コード系の相違点

相違点	SQL Server	Oracle		
使用可能なコード系	シフトJIS / UCS2(UTF-16)	コード系	CHAR	NCHAR
		シフトJIS系	JA16SJIS	AL16UTF16
		UTF-8系	AL32UTF8	UTF8
		UTF-16系	AL32UTF8	AL16UTF16
文字コードの設定	COBOLとDBのコード系を同一にしてください。(推奨)			
	COBOLがunicodeで、DBがシフトJISの場合、シフトJISの範囲しかデータが入りません。シフトJIS範囲外のデータが入った場合、文字化けすることがあります。	COBOLとDBとで、文字コードが異なっていた場合、データベースアクセス時にエラーが発生することがあります。		
PowerRDBconnector内でのコード変換	変換しません。	・ NetCOBOL for .NETでSJIS-UCS2(X項目がシフトJIS、N項目がunicode)のコード系指定時、X項目のデータを、シフトJISとunicode(UTF-8)の間で変換します。 このため、X項目内の日本語で、変換エラーが生じることがあります。 ・ COBOLの文字コード系が、unicode系(UTF8-UCS2、UCS2またはUTF16)の場合、N項目のデータをUTF-8間で変換します。		

表D.8 COBOLアプリケーションの実行時コード系とDBのコード系ついて

COBOLアプリケーションのコード系			SQL Server		Oracle					
			シフトJIS	UTF-16	シフトJIS		UTF-8		UTF-16	
種別	コード系	USAGE句			JA16SJIS (CHAR)	AL16UTF16 (NCHAR)	AL32UTF8 (CHAR)	UTF8 (NCHAR)	AL32UTF8 (CHAR)	AL16UTF16 (NCHAR)
NetCOBOL for .NET	SJIS系	X(英数字)	○	○	○	○	△(注3)	△(注3)	△(注3)	△(注3)
		N(日本語)	○	○	○	○	△(注3)	△(注3)	△(注3)	△(注3)
	UTF8系	X(英数字)	○(注1)	○	×(注2)	○	○	○	○	○
		N(日本語)	○(注1)	○	×(注2)	○	○	○	○	○
	混在系	X(英数字)	○	○	×(注2)	○	○	○	○	○
		N(日本語)	○(注1)	○	×(注2)	○	○	○	○	○
NetCOBOL for Windows	SJIS系	X(英数字)	○	○	○	○	△(注3)	△(注3)	△(注3)	△(注3)
		N(日本語)	○	○	○	○	△(注3)	△(注3)	△(注3)	△(注3)
	UTF16系	X(英数字)	○(注1)	○	×(注2)	○	○	○	○	○
		N(日本語)	○(注1)	○	×(注2)	○	○	○	○	○

○:使用できます。

△:後方空白や文字列の表現サイズの違いから、検索の項目として使用できない場合があります。

×:使用できません。

文字コード系の表記は、NetCOBOL for .NETまたはNetCOBOL for Windowsをコンパイル時に以下のオプションを選択した場合です。

SJIS系:RCSオプションにSJISを指定したCOBOLアプリケーション

UTF8系:RCSオプションにUTF8-UCS2を指定したCOBOLアプリケーション

混在系:RCSオプションにSJIS-UCS2を指定したCOBOLアプリケーション

UTF16系:RCSオプションにUCS2またはUTF16を指定したCOBOLアプリケーション

(注1)

シフトJISの範囲でしかデータが入りません。範囲外の場合、文字化けすることがあります。

(注2)

WRITE文は正常動作しますが、READ文で以下のエラーが発生します。

FILE STATUS=90、iserrno = 255 isstat1='0' isstat2=0x0 isstat3='6' isstat4=0x0

ORA-01406:フェッチされた列の値は切り捨てられました。

(注3)

シフトJISのCOBOLアプリケーションからUTF-8系、UTF-16系のデータベースにアクセスする場合

日本語の文字はNCHARの列データ内でしか扱えません。CHARの列データに格納した場合、後方空白の文字サイズの違いにより、正しくキー検索できなかったり、FILE STATUSが23や90のエラーが発生したりすることがあります。

D.5 PowerRDBconnector 動作環境ファイルの相違点

表D.9 PowerRDBconnector 動作環境ファイルのプロパティの相違点

プロパティ名		設定内容の相違点	
SQL Server	**Oracle**	**SQL Server**	**Oracle**
ServerName	**ServerName**	サーバのホスト名 または IPアドレス	データベースに接続する名前(データベース名) または Oracle のTNSエントリー名
DataSourceName	－	SQL Serverデータベースのデータベース名	なし
－	**ProviderName**	なし	OCI
－	**TableLock**	なし	ONまたは指定なし
TimeOut	－	レコードロック待合せ時間	なし
－	**RecordLock**	なし	レコードロック待合せ時間
－	**CheckLock**	なし	OFF 排他制御をデータベース任せにする場合のみ

－:該当プロパティなし

D.6 COBOL初期化ファイルの相違点

表D.10 COBOL初期化ファイルのプロパティの相違点

プロパティ名		設定内容の相違点	
SQL Server	**Oracle**	**SQL Server**	**Oracle**
TableName	**TableName**	テーブル名またはビュー名に指定できる文字数を以下に示します。 ・ 半角:128文字以内	テーブル名またはビュー名に指定できる文字数を以下に示します。 ・ 半角:30文字以内 ・ 全角

プロパティ名		設定内容の相違点	
SQL Server	Oracle	SQL Server	Oracle
		・ 全角:64文字以内	ー シフトJISの場合 15文字以内 ー unicodeの場合 10文字以内
Truncate	**Truncate**	OUTPUTモードでビューをオープンする場合、導出元テーブル名を指定します。	OUTPUTモードで、高速にオープンする場合に指定します。

D.7 コンパイル時の相違点

インポートライブラリの格納位置が異なります。

表D.11 シングルセッションプログラミングのインポートライブラリの相違点

動作	ファイル名	格納ディレクトリ	
		SQL Server	Oracle
32ビット	F3BWS1CB.LIB F3BWS1SB.LIB	PowerRDBconnectorをインストールしたディレクトリ配下の以下のディレクトリに格納されます。 ・ SQL Server 2005:SQLSV2005 ・ SQL Server 2008:SQLSV2008	PowerRDBconnectorをインストールしたディレクトリ配下の以下のディレクトリに格納されます。 ・ Oracle10g:ORA10g ・ Oracle11g:ORA11g
64ビット	F4ARS1CB_64.LIB F4ARS1SB_64.LIB	PowerRDBconnectorをインストールしたディレクトリ配下の以下のディレクトリに格納されます。 ・ SQL Server 2008:SQLSV2008	PowerRDBconnectorをインストールしたディレクトリ配下の以下のディレクトリに格納されます。 ・ Oracle11g:ORA11g

D.8 COBOLアプリケーションの相違点

D.8.1 トランザクションサブルーチンの相違点

表D.12 トランザクションサブルーチンの相違点

項目	SQL Server	Oracle
トランザクション開始サブルーチンのシーケンス	・ 既にトランザクションが開始されていた場合は、エラーとなります。	・ 既にトランザクションが開始されていた場合は、エラーとなります。
	・ レコードロック獲得中に、トランザクション開始を行ってもエラーになりません。	・ レコードロック獲得中に、トランザクション開始を行うとエラーになります。
デッドロックエラー発生後の動作	・ トランザクション確定を行うとエラーになります。	・ トランザクション確定を行ってもエラーになりません。

D.8.2 排他制御の相違点

表D.13 トランザクション未適用時のレコードロックの相違点

項目			SQL Server	Oracle
実現方法			・ 更新可能カーソルによる1レコードのロック	・ 強制的なトランザクション 1つのプログラムで、複数のファイルを更新している場合は、すべてのファイルでレコードロック解放タイミングの条件を満たした時点で、レコードロックは解放されます。
			―	・ 複数ファイルのレコードロックが重なると、一方のファイルのレコードロックのみを解放することはできません。
レコードロックの獲得 I-Oモードのオープン			・ START文で位置付けたレコード (注)	―
			・ READ文で読み込んだレコード	・ READ文で読み込んだレコード
レコードロックの解放	アクセスモード	SEQUENTIAL	・ START文を実行した場合	・ START文を実行した場合
			・ 次のレコードを読み込んだ（READ）場合	・ 次のレコードを読み込んだ(READ)場合
			―	・ 更新(REWRITE)、削除(DELETE)を行った場合
			・ EOFを検出した場合	・ EOFを検出した場合
			・ CLOSE文を実行した場合	・ CLOSE文を実行した場合
		RANDOM	・ 読込み（READ）、更新（REWRITE）、削除（DELETE）を行った場合	・ 読込み(READ)、更新(REWRITE)、削除(DELETE)を行った場合
			・ CLOSE文を実行した場合	・ CLOSE文を実行した場合
		DYNAMIC	・ START文を実行した場合	・ START文を実行した場合
			・ 次のレコードを読み込んだ(READ)場合	・ 次のレコードを読み込んだ(READ)場合
			―	・ 更新(REWRITE)、削除(DELETE)を行った場合
			・ EOFを検出した場合	・ EOFを検出した場合
			・ CLOSE文を実行した場合	・ CLOSE文を実行した場合

―：対応項目なし

（注）

DYNAMICモードでオープンし、START FIRST文を実行した場合は位置付けされた先頭のレコードに対するレコードロックは獲得されません。

表D.14 トランザクション適用時のレコードロックの相違点

項目	SQL Server	Oracle
レコードロックの獲得 I-Oモード、OUTPUTモード、またはEXTENDモードのオープン	・ START文で位置付けたレコード	―
	・ READ文で読み込んだレコード	・ READ文で読み込んだレコード
	・ トランザクション中にREWRITE文で更新したレコード	・ トランザクション中にREWRITE文で更新したレコード
	・ トランザクション中にDELETE文で削除したレコード	・ トランザクション中にDELETE文で削除したレコード

	・トランザクション中にWRITE文で追加したレコード	・トランザクション中にWRITE文で追加したレコード
レコードロックの解放	・トランザクションの確定または取消し(注)	・トランザクションの確定または取消し
デッドロック発生時	・エラーが発生した場合、SQL Serverのトランザクションは自動的にロールバックされますが、PowerRDBconnectorのトランザクションは継続している状態になっています。 この場合、以下のいずれかの対処を行ってください。 ー アプリケーションを終了する。 ー トランザクション取消しサブルーチンを実行する。	・エラーが発生したトランザクションは、継続されています。

ー:対応項目なし

(注)

トランザクションの確定または取消し直前に、位置付けまたは読み込んだレコードのレコードロックは解放されません。

D.8.3 可変長項目の相違点

表D.15 可変長項目の相違点について

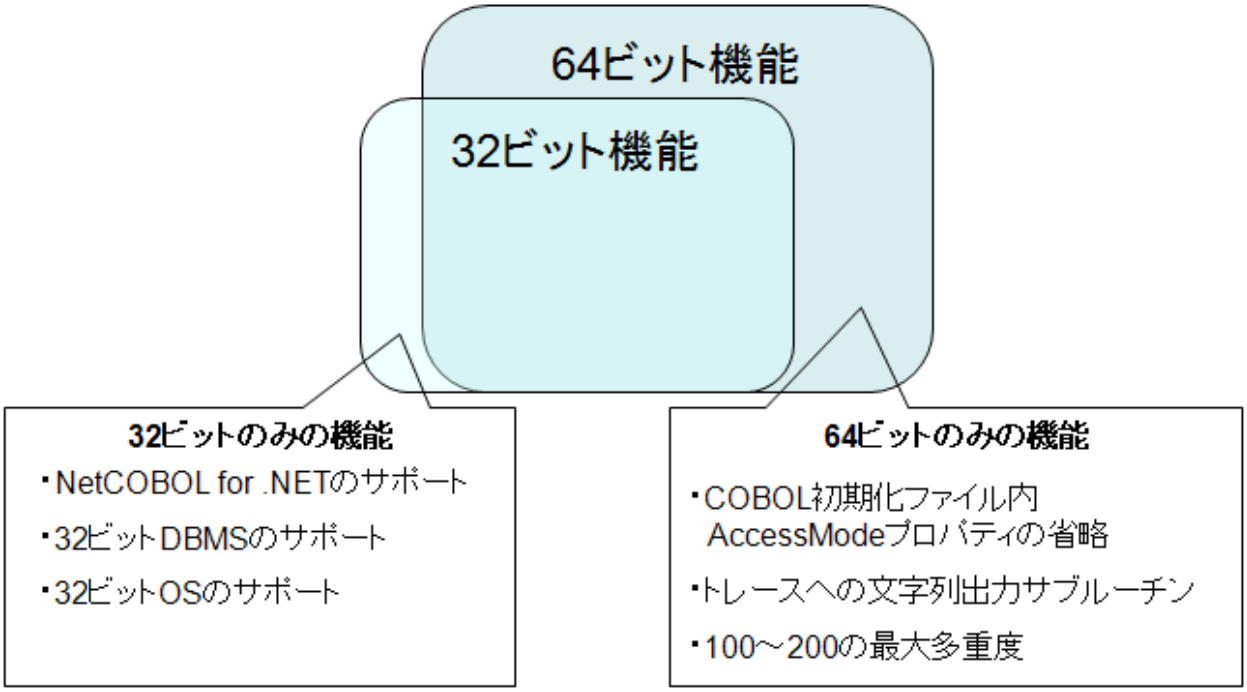
項目	SQL Server	Oracle
データベース項目長がCOBOL項目長より大きい場合	・データを書き込むときは、COBOL項目長の長さでデータが格納されます。読み込むときは、COBOL項目長の長さだけ読み取ります。	・エラーとなります。

付録E　32ビット動作と64ビット動作の相違点

本章では、PowerRDBconnectorを32ビットのアプリケーションで使用する場合と、64ビットのアプリケーションで使用する場合との相違点について説明します。

NetCOBOL、データベース、OSおよびIISなどの詳細な相違点については、それぞれのマニュアルを参照してください。

図E.1 32ビット動作と64ビット動作の相違点



E.1　動作環境の相違点

E.1.1　必要なソフトウェアの相違点

表E.1 必要なソフトウェアの相違点

分類	32ビット動作使用時	64ビット動作使用時
PowerRDBconnector	32ビット版 製品名に(64bit)が付いていない版	64ビット版 製品名に(64bit)が付いている版
OS	Windows(32ビット) Windows（x64）(*1)	Windows 7（x64）（開発時のみ） Windows Server 2008 R2
データベース	データベース(32ビット) データベース(x64) (*1)	データベース(x64)
	SQL Server 2005 SQL Server 2008 SQL Server 2008 R2 Oracle Database R10.1.0 Oracle Database R10.2.0 Oracle Database R11.1.0 Oracle Database R11.2.0	SQL Server 2008 R2 Oracle Database R11.2.0

分類	32ビット動作使用時	64ビット動作使用時
NetCOBOL	NetCOBOL (32ビット)	NetCOBOL (64bit)
	NetCOBOL for .NET (*1)	使用できません。

*1:PowerRDBconnectorはWOW64サブシステム上で32ビット動作します。OS、データベースおよびNetCOBOLはそれぞれの環境に注意してください。

E.1.2 インストールした動作環境の相違点

表E.2 インストールした動作環境に追加される環境変数の相違点

分類	32ビット動作使用時	64ビット動作使用時
追加される環境変数名	RDBCONNECTOR	RDBCONNECTOR64
Path環境変数に追加されるパス名	%RDBCONNECTOR%	%RDBCONNECTOR64%

E.2 PowerRDBconnector機能の相違点

E.2.1 COBOLアプリケーションの機能範囲の相違点

表E.3 COBOLアプリケーションの機能範囲の相違点

項目		32ビット動作使用時	64ビット動作使用時
PowerRDBconnector動作環境ファイル(DBIO_ENV)の格納先	環境変数の指定先	環境変数DBIO_ENVで指定可能	環境変数DBIO_ENV_x64で指定可能
	実行プログラムと同じディレクトリ	○	○
	アプリケーション起動時のカレントパス	○	○
COBOL初期化ファイルの記述	AccessModeプロパティの指定	必須	省略可能
	AccessModeプロパティの指定値と、COBOLのACCESS MODE句の指定が異なっていた場合	異なっていても、エラーになりません。ただし、動作保証されません。	異なっているとエラーになります。
	エントリ情報	DLL名が異なります。詳細は、「表E.4 エントリ情報の相違点」を参照してください。	
イベントログ情報のソース名		FSP_RDBCONNECTOR_FJSVDBIO	FSP_RDBCONNECTOR_FJSVDBIO64
1システムで多重実行可能な最大プロセス数		100(*1)	200(*1)
トレースへの文字列出力サブルーチン		×	○

○:使用可能

×:使用不可

(*1)1システムで多重実行可能な最大プロセス数

同時実行できる最大多重実行数は、アプリケーションの処理内容(入出力文回数)、データベースおよび使用するハードウェア(CPU、ハードディスク能力、データベースに割り当てる実メモリなどのチューニングパラメーター)の要件に依存します。例えば、バッチプログラムの大量データ処理は、高負荷となるため、多重実行に耐えられません。

この値は、OSやデータベースのチューニングをせず、シングルセッションを用いて、常にデータベースにアクセスし続けるCOBOLアプリケーションを多重に実行して動作した値です。このため、この値まで、性能も含め運用可能な多重度を保証するものはありません。

E.2.2 エントリ情報の相違点

エントリ情報に記述するサブルーチンのDLL名が異なります。

表E.4 エントリ情報の相違点

	サブルーチン名	32ビット	64ビット
シングルセッション	XMROTSTR	F3BWS1CB.DLL	F4ARS1CB_64.DLL
	XMROTEND	F3BWS1CB.DLL	F4ARS1CB_64.DLL
	XMROTCNL	F3BWS1CB.DLL	F4ARS1CB_64.DLL
	XMROTRBK	F3BWS1CB.DLL	F4ARS1CB_64.DLL
	XMROAUTH	F3BWS1SB.DLL	F4ARS1SB_64.DLL
	XMROLOG	なし	F4ARS1SB_64.DLL
マルチセッション	COB_PRDB_START	F3BIEFNC.dll	F4AGEFNC.dll
	COB_PRDB_END	F3BIEFNC.dll	F4AGEFNC.dll
	COB_PRDB_CHG	F3BIEFNC.dll	F4AGEFNC.dll
	COB_PRDB_TRAN	F3BIEFNC.dll	F4AGEFNC.dll
	COB_PRDB_AUTH	F3BIEFNC.dll	F4AGEFNC.dll
	COB_PRDB_LOG	なし	F4AGEFNC.dll

E.3 コンパイル時の相違点

E.3.1 インポートライブラリ

インポートライブラリのファイル名が異なります。

表E.5 インポートライブラリの相違点

	32ビット	64ビット
インポートライブラリ名	F3BWS1CB.LIB F3BWS1SB.LIB	F4ARS1CB_64.LIB F4ARS1SB_64.LIB

付録F　リリース情報

本章では、PowerRDBconnectorのリリース情報を示します。

F.1　リリース情報

製品名	VL	リリース情報
PowerRDBconnector for NetCOBOL	V1.0L10A	SQL Server 2000をサポートしました。
PowerRDBconnector for NetCOBOL	V1.0L20	・ Oracle9i, Oracle10g Release 1をサポートしました。 ・ インストール先を可変にしました。
PowerRDBconnector クライアントパッケージ for NetCOBOL	V1.0L20	・ クライアント運用をサポートしました。 ・ データベース認証をサポートしました。 ・ Oracleのビュー表を、OUTPUTオープン可能としました。 ・ ServerNameプロパティの制限を255文字まで緩和しました。
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V2.0L10	・ Windows Server 2003 Release 2をサポートしました。 ・ Windows Server 2003(x64)をサポートしました。 ・ Oracle10g Release2をサポートしました。 ・ SQL Server 2005をサポートしました。 ・ NetCOBOL for .NET V3.0をサポートしました。 ・ マルチスレッドプログラミング用の以下のサブルーチンを追加しました。 　－ COB_PRDB_STARTサブルーチン 　－ COB_PRDB_ENDサブルーチン 　－ COB_PRDB_TRANサブルーチン 　－ COB_PRDB_AUTHサブルーチン ・ データベース認証をサポートしました。 ・ SQL Serverのビュー表に対し、OUTPUTモードでのオープンをサポートしました。 ・ テーブルロック解除時に強制トランザクションを取り消す機能(XMROTRBKサブルーチン)をサポートしました。 ・ Oracleでバイナリ項目をサポートしました。 ・ PowerRDBconnector動作環境ファイルに以下のプロパティを追加しました。 　－ Suppressプロパティ:後方空白補正 　－ DataCheckプロパティ:データチェック 　－ TraceLevelプロパティ:トレースのレベル 　－ COBOL初期化ファイルに以下のプロパティを追加しました。 　－ Suppressプロパティ:後方空白補正 ・ TableNameプロパティ、ServerNameプロパティに記述できる記号を増やしました。

製品名	VL	リリース情報
		・ ServerNameプロパティに指定できる値を255文字にしました。 ・ SJIS-UCS2(英数字項目がシフトJIS、日本語項目がunicode)のコード系をサポートしました。 ・ ErrorLogプロパティの指定に誤りがあった場合のメッセージ通知機能を追加しました。 ・ 環境誤りのエラーコードを細分化し、isstat3を変更しました。 ・ エラー情報に、列名やPowerRDBconnector動作環境ファイルのプロパティ名を出力する機能を追加しました。 ・ データ補正機能を追加しました。 ・ トレース出力機能を追加しました。
PowerRDBconnector クライアントパッケージ for NetCOBOL	V2.0L20	・ インストール時に選択したデータベース以外にアクセスできないよう変更しました。 ・ Windows Vistaをサポートしました。 ・ NetCOBOL for .NET V3.1をサポートしました。 ・ NetCOBOL V9.0をサポートしました。 ・ CLOSE時のエラーは無視するように修正しました。 ・ ALTERNATE KEYに対応する項目を更新後、キー順読込みするカーソル位置をCOBOL標準となるよう修正しました。 ・ Visual C++ 6.0のランタイムモジュールを使用しないように変更しました。
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V3.0L10	サーバパッケージのみのリリース情報を以下に示します。 ・ Windows Vistaをサポートしました。 ・ NetCOBOL for .NET V3.1をサポートしました。 ・ NetCOBOL V9.0をサポートしました。 ・ CLOSE時のエラーは無視するように修正しました。 ・ ALTERNATE KEYに対応する項目を更新後、キー順読込みするカーソル位置をCOBOL標準となるよう修正しました。 ・ Visual C++ 6.0のランタイムモジュールを使用しないように変更しました。 サーバパッケージ、クライアントパッケージ共通のリリース情報を以下に示します。 ・ JIS2004の文字コード系のデータをサポートしました。 ・ Windows Server 2008、Windows Server 2008(x64)をサポートしました。(注) ・ NetCOBOL V10をサポートしました。 ・ Oracle11gをサポートしました。 ・ UTF-8で作成されたPowerRDBconnector動作環境ファイル、COBOL初期化ファイルをサポートしました。

製品名	VL	リリース情報
		(注) Windows Server 2008のサポートについて Windows Server 2008 、Windows Server 2008(x64) 上 の PowerRDBconnectorからアクセス可能なデータベースは、SQL Server 2005のみです。
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V3.1L10	・ SQL Server 2008をサポートしました。 ・ Windows Server 2008 、Windows Server 2008(x64) 上 の PowerRDBconnectorからOracleへのアクセスをサポートしました。 ・ Solaris上のOracleへのアクセス形態をサポートしました。 ・ NetCOBOL for .NET V4.0をサポートしました。 ・ 動作環境ひな型作成ツールを新規提供しました。 ・ Windows OSから通知されたエラー情報の文字列をそのままイベントログ情報に出力するようにしました。 ・ PowerRDBconnector動作環境ファイルに以下のプロパティを追加しました。 － PrepareModeプロパティ:SQL文準備モード ・ PowerRDBconnector動作環境ファイルのSuppressプロパティに、指定できる値を追加しました。 － ON/HALF － ON/FULL
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V3.1L20	32ビット動作、64ビット動作共通のリリース情報を以下に示します。 ・ Windows Server 2008 R2、Windows 7をサポートしました。 ・ Oracle 11g R2/SQL Server 2008 R2をサポートしました。 ・ NetCOBOL for .NET V4.1、V4.2をサポートしました。 ・ セッション切換えサブルーチンをサポートしました。 ・ データベースの可変長項目をサポートしました。 ・ COBOLと関連付けないデータベース列をサポートしました。 ・ トランザクション開始のタイミングを、OPEN文の前後によらず実行できるようにしました。 ・ トランザクション区間内で、SQL Serverにアクセス中、デッドロック発生後、トランザクション取消しサブルーチンが正常終了するようにしました。 64ビット動作のみのリリース情報を以下に示します。 ・ サーバパッケージのみ、64ビットネイティブ動作をサポートしました。 ・ PowerRDBconnector動作環境ファイルを、COBOL初期化ファイルの格納場所と同じ場所に配置できるようにしました。 ・ COBOL初期化ファイルのアクセスモードを省略可能にしました。 ・ トレースファイルへ文字列を出力するサブルーチンをサポートしました。

F.2 非互換について

製品名	VL	直前のVLとの非互換
PowerRDBconnector for NetCOBOL	V1.0L20	・ インストール先を可変にしたため、サブルーチン使用時のリンク先の指定が変更になりました。 ・ 列名に付加された文字列(_UNSIGN_NUMERIC等)を、列名の後方から検索するように修正しました。
PowerRDBconnector クライアントパッケージ for NetCOBOL	V1.0L20	特にありません。
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V2.0L10	・ PowerRDBconnector動作環境ファイルに、プロパティ名と＝のみ記述した場合、初期値動作でなく、エラーとなります。 ・ PowerRDBconnector動作環境ファイルのErrorLogプロパティに、ドライブ名のみを指定すると、オープン処理でエラーとなります。
PowerRDBconnector クライアントパッケージ for NetCOBOL	V2.0L20	・ インストール時に選択したデータベース以外にはアクセスできなくなります。 ・ CLOSE時のエラーは通知されず、正常終了します。 ・ ALTERNATE KEYに対応する項目を更新後、キー値の重複データを順に読込む位置が、次のように変更されます。 ー 前VLまでは変更後のキー位置まで読み飛ばされます。 ー COBOL標準であるキー変更前の位置の次から読み飛ばしがなく正しく読み込まれます。
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V3.0L10	サーバパッケージの前版（V2.0L10）からの非互換点を以下に示します。 ・ インストール時に選択したデータベース以外にはアクセスできなくなります。 ・ CLOSE時のエラーは通知されず、正常終了します。 ・ ALTERNATE KEYに対応する項目を更新後、キー値の重複データを順に読込む位置が、次のように変更されます。 ー 前VLまでは変更後のキー位置まで読み飛ばされます。 ー COBOL標準であるキー変更前の位置の次から読み飛ばしがなく正しく読み込まれます。 クライアントパッケージの前版（V2.0L20）からの非互換点を以下に示します。 ・ Windows Vistaで、SQL Server 2000やOracle9iのデータベースに対応したPowerRDBconnectorがインストールできなくなりました。
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V3.1L10	・ 前版(V3.0L10)までは、使用するSQL文をOPEN時にすべて準備していましたが、初回のキー検索時(OPEN後の最初のSTART、乱READ、乱REWRITE、または、乱DELETE)にSQL文を準備するように変更しました。この変更により、OPEN文の処理性能が向上します。 ー 業務の起動時など、まとめて複数ファイルのOPENを行っている処理の性能が改善されます。 一方、初回のキー検索の処理性能が遅くなる場合があります。 なお、PowerRDBconnector動作環境ファイルのPrepareModeプロパティにOPENを指定することで、前版までの動作に戻すことができます。

製品名	VL	直前のVLとの非互換
		・ Windows 2000、Oracle 9iおよびSQL Server 2000が、未サポートになりました。
PowerRDBconnector サーバパッケージ for NetCOBOL PowerRDBconnector クライアントパッケージ for NetCOBOL	V3.1L20	・ 前版(V3.1L10)までは、OPEN文後にしかトランザクション開始できませんでした。V3.1L20では、OPEN文の前後によらず、トランザクション開始が実行できるようになりました。 ・ SQL Serverへのアクセスでデッドロックエラーが発生しても、COBOLアプリケーションを継続して実行できるようになりました。

索　引

[ら]