Adabas

概念および機能

バージョン 8.1.3

June 2008

Adabas

This document applies to Adabas Version 8.1.3 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

目次

1 概念および機能

このドキュメントは、Software AG の Adabas の技術の概要を説明しています。Adabas とは、"adaptable data base management system" の略で、適応性に優れたデータベース管理システムという意味です。本書は、Adabasのオペレーティング環境、設計、使用、およびオプションの拡張についての基本的な理解が必要な人に向けて、Adabas の概要を説明しています。

Adabas の概要	システムの概要を紹介し、サポートするオペレーティング環境について説明します。
Adabas 設計	Adabas で使用されている構造について説明し、またその機能と相互作用について説明します。
Adabas の使用	データベースへのアクセス、および保全性の維持について説明します。
Adabas ユーティリティ	Adabas ユーティリティについて簡単に説明します。
Adabas のライセンス	Adabas のライセンスの概念について説明します。
Adabas Security	Adabas で利用可能なセキュリティ機能について説明します。
オプションの拡張	Adabas の主要製品に機能を追加する、Software AG が提供するオプションの拡張について簡単に説明します。

2 Adabas の概要

Adabas は適応性に優れたデータベースで、データベースのパフォーマンスが重要な要素であるメインフレームプラットフォーム向けの高性能、マルチスレッド型のデータベース管理システムです。メインフレーム、ミッドレンジ、PCを含む複数の異なるプラットフォーム間で、優れた相互運用性、拡張性、移植性を発揮するデータベースです。

このchapterでは、次のトピックについて説明します。

動作の特徴

高い可用性

Adabas は、24 時間 365 日動作するように設計されています。 スペースはダイナミックに管理されます（「*Adabas のスペース管理*」を参照）。ファイルは、ロードおよびアンロード、バックアップおよびリストアが可能です。また、アクティブなデータベースを中断することなくシステムパフォーマンスを分析できます。

ストレージスペースの最適化

Adabas では、データを圧縮形式で格納し、必要なスペースを削減します。 最新のデータベースはギガバイト（1000 メガバイト）単位の容量を持ち、時にはテラバイト（1000 ギガバイト）にもなる場合もあります。したがって、データの保存には大容量のディスクスペースが必要となります。 また、必要なスペースが削減されると、入力／出力（I/O）システムの効率も改善されます。

パフォーマンス

パフォーマンスは Adabas の主要な要素です。Adabas には、パフォーマンスを向上させる多くの機能が含まれています。 例えば、数多くのセットアップパラメータを利用して、データベースの操作環境をチューニングできます。また、その多くが、データベースがアクティブでも変更が可能です。

フォルトトレランス

Adabas は、データベースまたはシステムの異常終了から自動的に復帰します。 Adabas データベースが開始するときは、必ず自動チェックが開始され、データベースが前回正常に終了したか、またはアクティブなトランザクションが中断されていないかを確認します。トランザクションが中断されていた場合、Adabas は未完了のトランザクションのすべての変更を自動的にリセットして、データベースの一貫性を保ちます。

オペレーティング環境

メインフレーム版の Adabas 8 は、次のオペレーティング環境をサポートします。

- SNI の BS2000
- IBM の z/OS、VSE、および z/VM

次の環境で Adabas を使用すると、メインフレーム Adabas を分散環境で使用することができます。

- OpenVMS 上で動作している Digital の VAX および Alpha AXP
- OS/400 上で動作している IBM の AS/400
- VS 上で動作している Wang
- OS/2 または Windows/NT 上の IBM PC および互換機
- 各種のサポート済み UNIX プラットフォーム

コミュニケーションインターフェイスとして、メインフレーム Adabas は、 Com-plete などの Software AG の TP モニタ、および TSO、 CICS、 IMS/DC、 AITM/DC、 TIAM、 UTM、および Shadow などのその他の一般的なモニタをサポートしています。

Software AG のマルチシステム処理ツール Entire Net-Work は、Adabas やその他のサービスタスクを使用したネットワーク全体における通信を可能にし、分散処理のメリットを受けることができます。

ネットワークアクセスメソッドのサポートはラインドライバの形で実装されています。 メインフレーム Entire Net-Work は、 VTAM、 IUCV、 DCAM、 CTCA（チャネル間アダプタ）、 TCP/IP、および XCF に対するドライバを提供します。

サポートされるデータモデル

Adabas は次の点でリレーショナル型の特徴を備えています。

- 行がデータの*レコード*を表し、列が*フィールド*を表す表形式でデータを格納する
- 個別の Adabas ファイルは、共通のフィールドにより論理的にリンクすることができる

Adabas は、次の点で真のリレーショナルデータベースとは異なります。

- 多くのデータの関係情報を物理的に格納するため、すべての関係情報を実行時に論理的に作成する真のリレーショナルデータベースと比較して求められる CPU リソースが少なくて済む
- フィールドの繰り返しグループのサポート

Adabas では、データの関係、管理、および実際の物理データからの取得を区別し、物理データを独立して格納します。 柔軟性に富んだアクセス方法が提供され、単純な検索や複雑な検索も迅速に効果的に行うことができます。 プログラムからデータが独立しているため、データベース構造が変更されても、再プログラムを行う必要性が最小限に抑えられます。

必要に応じて、論理データ関係を作成できます。 Adabas は、ユーザーの環境によって左右されるどのような表現やアクセス要件にも対応します。企業ユーザーは、個別にシステム内でのデータの表示方法を決定することができ、データベースや既存のプログラムを変更することなく、データ関係をダイナミックに変更することができます。

すべてのデータに対して単一のモデルが要求されるシステムとは異なり、Adabas では、アプリケーションで要求されるどのような構造も選択できます。 データモデルの観点からの選択により、さまざまなデータモデルを使用して同一のデータにアクセスできます。

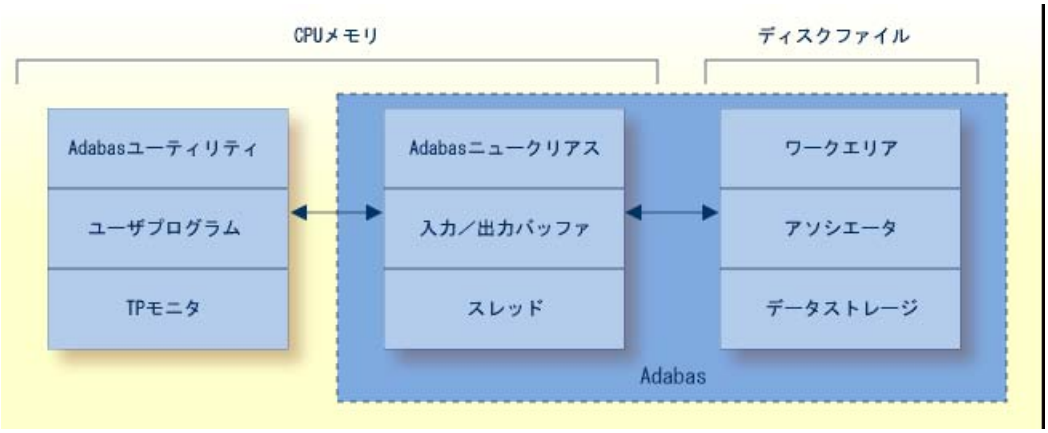
- ネストされた関係を含むリレーショナル型（テーブル内のテーブル）
- 構造型オブジェクトをサポートする実証済みの機能を備えたエンティティ関係
- 階層型、ネットワーク
- 地理情報
- テキスト

これらのデータモデルは、単一のビジネスソリューション内で組み合わせることが可能です。また、異なるソリューションでは、異なるデータモデルを使用して Adabas データを表示できます。

新しい要件が発生すると、Adabas は、データベースの再設計やアプリケーションシステムの再プログラムを行わずに、範囲や複雑さを拡張できます。例えば、フィールドやアクセスキーは、ファイルの再ロードや再構成を必要とせずに、いつでも Adabas ファイルに追加できます。

動作構造

次の図は、Adabas システムの動作構造を示しています。



ニュークリアス、I/O バッファおよびスレッド

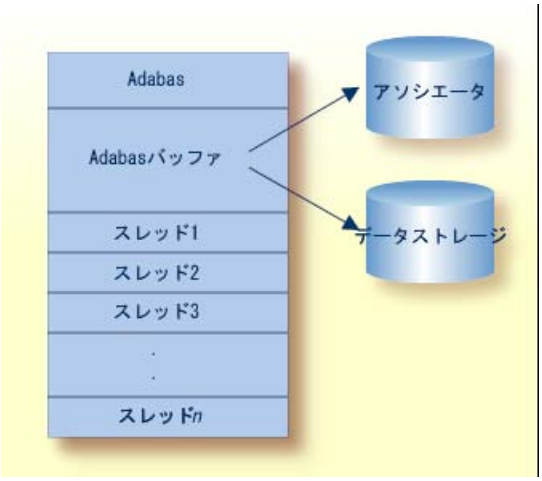
Adabas ニュークリアスおよび入力／出力（I/O）バッファは、開始時にメインメモリにロードされます。 ニュークリアスはプログラムの集まりを指し、Adabas を操作したり、すべての処理を調整したり、ユーザープログラムのステートメントを Adabas コマンドに変換したりします。 すべてのプログラムは、ニュークリアスを介して Adabas ファイルにアクセスします。 データアクセスや更新などのデータベースのアクティビティはすべて、Adabas ニュークリアスにより管理されます。 ほとんどの場合、1 つのニュークリアスが 1 つの物理データベースを管理します。

Note: 単一の物理データベースに対する複数のニュークリアスの実行についての情報は、単一のオペレーティングシステムイメージ上で動作する場合は、「*Adabas Parallel Services*」を、複数の z/OS イメージ上で動作する場合は「*Adabas Cluster Services*」を参照してください。

Adabas 入力／出力バッファエリアには、よく使用されるデータとデータ間の関係が含まれており、このエリアのサイズは Adabas セッションごとに変更することができます。このエリアでは、物理的入力／出力（I/O）アクティビティが最小化されるため、コンピュータの処理時間が節約されます。 このバッファエリアには、データベースから読み込まれたブロックとデータベースに書き込まれるブロックが格納されます。

- データベースから読み込まれるブロックのうち、最も頻繁にアクセスされるブロックは、バッファアルゴリズムによってメモリに保持されます。 データベース内のブロックが必要になると、バッファの内容がチェックされ、そのブロックがすでにメモリにあるかどうかが調べられます。これにより、不必要な読み込みが避けられます。
- 複数の更新が行われた場合、更新内容はブロックに蓄積された後、データベースに書き込まれます（フラッシュ）。

Adabas はマルチスレッド処理を提供し、スループットを最大化します。 I/O アクティビティにより、アクティブなスレッドでコマンドの処理が中断された場合は、自動的に別のスレッドに切り替えられます。 ユーザーは、最大250 個までの Adabas セッションで使用できるように、8 KB スレッドの数を設定できます。



データストレージ、アソシエータ、およびワーク

データストレージ、アソシエータおよびワークコンポーネントは、次に示すように物理的なディスクエリアです。

- データストレージには、未加工データが含まれます。通常、データは圧縮形式で格納されます。
- アソシエータには、データ関係についての情報が含まれます。
- ワークエリアには、データプロテクションエリア、および複合検索操作や分散トランザクション処理の中間結果を格納する一時ストレージが含まれます。

これらのデータベースコンポーネントの詳細は、「*Adabas 設計*」を参照してください。

ユーティリティ、ユーザープログラムおよび TP モニタ

ユーティリティ

ファイルのロードや削除などのデータベースサービスは、オンラインおよびバッチモードのユーティリティプログラムの統合セットにより処理されます。 ほとんどのユーティリティは、通常のデータベースアクティビティと並行して実行できるため、毎日の運用を中断することはありません。

Adabas ユーティリティは、初期設計およびロード操作、バックアップ／リストア／リカバリルーチン、データベース更新ルーチン、および監査／制御／チューニング処理の各機能を提供します。 各ユーティリティの簡単な説明については、「*Adabas ユーティリティ*」を参照してください。

ユーザープログラム

ニュークリアスは、次の言語で書かれたバッチまたはオンラインのユーザープログラムによりコールされます。

- Software AG の第 4 世代のアプリケーション開発環境である Natural、またはその他の第 4 世代言語
- アセンブラ、またはパワフルで柔軟な Adabas ダイレクトコールインターフェイスを使用する FORTRAN、COBOL、PL/I などの第 3 世代プログラミング言語（REXX/VSE インタプリタ言語もサポートされます） 各 Adabas コールは、パラメータリストを使用します。パラメータリストは、Adabas へ、または Adabas からの情報の転送に使用するユーザープログラムで定義されたバッファを識別します。

Note: Ada、COBOL、FORTRAN、および PL/I プログラム用のプリコンパイラである、Adabas Native SQL の詳細は、このマニュアルの「」を参照してください。 Adabas への SQL インターフェイスである、Adabas SQL Gateway の詳細は、「*Adabas SQL Gateway*」を参照してください。

VSAM、DL/I、IMS/DB、SESAM、または TOTAL データベース構造からのデータは、Adabas Bridge 製品を使用して Adabas に送信および格納できます。 変更されていない元のプログラムは、元のデータアクセスコマンドを使用して動作を続行しますが、Brigde 製品はデータアクセスコマンドをインターセプトして、Adabas ダイレクトコールに送信します。 Adabas Bridge の詳細は、「*Adabas Bridges*」を参照してください。

特別なユーザープログラム：トリガ、およびストアドプロシージャ

Adabas の基幹部分である Adabas トリガおよびストアドプロシージャ機能は、Natural とともに使用して（「*Natural アプリケーション開発環境*」を参照）、Adabas サーバー環境内で *トリガ*および*ストアドプロシージャ*を記述および管理できます。 *トリガ*は、体系的に使用するプログラムのことで、このプログラムはイベントに基づいて自動的に起動されます。例えば、参照の保全性を確保するために使用できます。 *ストアドプロシージャ*は、特別なユーザーコールの結果として Adabas により実行される、多くの異なるクライアントで使用されるプログラムです。これらのプログラムをサーバー上の Adabas ファイルに格納すると、サーバーとの間で送受信するデータのトラフィック量を削減できます。 Adabas トリガおよびストアドプロシージャの詳細は、「*トリガとストアドプロシージャの使用*」を参照してください。

TP モニタ、ADALINK、および Adabas API

ほとんどのシステムでは、Adabas に対する標準コールが許可されていません。アプリケーションプログラムから発行されたコールを Adabas で処理可能な形式に変換するアプリケーションプログラムインターフェイス（API）を提供します。 Adabas API は、すべてのサポートされるメインフレームプラットフォーム上で、バッチおよびオンライン処理で利用可能です。

オンライン処理は、TP モニタにより制御されます。TP モニタは、Adabas へのコミュニケーションインターフェイスとして機能します。サポートされる TP モニタは、*オペレーティング環境*」を参照してください。 Software AG では、個々の TP モニタに固有な Adabas API を提供しています。 *ADALINK* は個々の TP モニタに固有の API の一部を示す一般的な名称です。

バッチアプリケーションは、シングルユーザーとマルチユーザーの両モードでサポートされます（「*オペレーションのモード*」を参照）。 Adabas のバッチ API では標準的なコール規則が使用されているため、CALL メカニズムを使用することで、すべての主要なプログラミング言語からコールすることができます。 大部分のメインフレームのオペレーションシステムでは、バッチアプリケーションモジュールをバッチ API または ADAUSER のいずれともリンクすることが可能です。

バッチアプリケーションプログラムを、Adabas のバージョン非依存モジュール ADAUSER とリンクすることを強くお勧めします。 ADAUSER はオプションで Adabas API とリンクすることができます。 ADAUSER は、Adabas リリースとの上位互換性を提供し、API または、ユーザーとニュークリアス間のリージョン間コミュニケーションを処理するメカニズムへの将来の変更に対してある程度の独立性を保ちます（「*オペレーションのモード*」を参照）。

IBM の OpenEdition 上で実行しているクライアントは、Adabas にアクセスできます。 Adabas へのコールを含んでいる OpenEdition アプリケーションは、バッチ API または ADAUSER とリンクすることができます。

Adabas の実行

- セッションのタイプ
- ストレージエリア
- オペレーションのモード
- ADARUN の起動パラメータ

- セッションの制御

セッションのタイプ

Adabas では、3 つのタイプのセッションが識別されます。

- *Adabas セッション*は、ニュークリアスが起動したときに開始し、ニュークリアスが終了したときに終了します。 Adabas ニュークリアスは、Adabas 起動（ADARUN）、パラメータなどの、そのオペレーティングシステムに応じたジョブ制御により起動されます。
- ユーザーセッションとは、バッチモードプログラム、あるいは端末を使用する人を意味します。 ユーザーセッションは、Adabas セッション中にのみ存在できます。つまり、Adabas ニュークリアスがアクティブなときのみです。 これは、一連の Adabas コールから成り、オープンユーザーセッション（OP）コマンド（任意）で開始し、クローズユーザーセッション（CL）コマンドで終了させることもできます。
- ユーティリティセッションは、バッチで実行されるか、または **Adabas Online System** を使用してオンラインで実行されます。ユーティリティには、Adabasニュークリアスがアクティブであることが必要なものとそうでないものがあります。ユーティリティの実行には、ADARUN 起動パラメータも使用されます。

ストレージエリア

Adabasニュークリアスおよび各ユーザープログラムまたはAdabasユーティリティは、オペレーティングシステムで定義された固有のストレージエリアで実行されます。 次に示すように、ストレージエリアの名前は、オペレーティングシステムによって異なります。

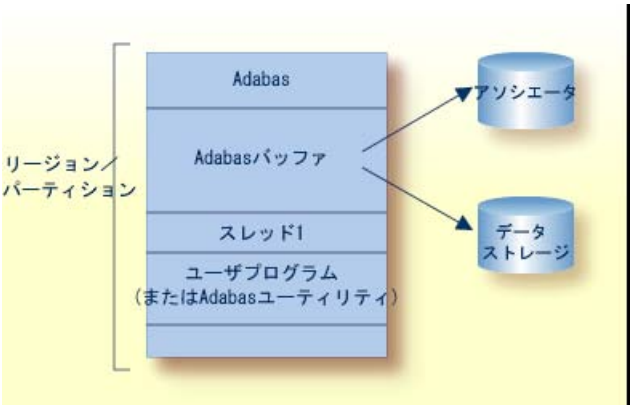
BS2000	タスク
z/OS	アドレススペース、データスペース、ハイパースペース、64 ビット仮想スペース
z/VM	仮想マシン
VSE	パーティション、アドレススペース、データスペース

説明の一貫性を保ち、また簡便にするため、Adabasのドキュメントでは、非VMエリア（タスク、アドレススペース、パーティションなど）を総称して*リージョン*と呼びます。 VM エリアは*仮想*マシンと呼びます。

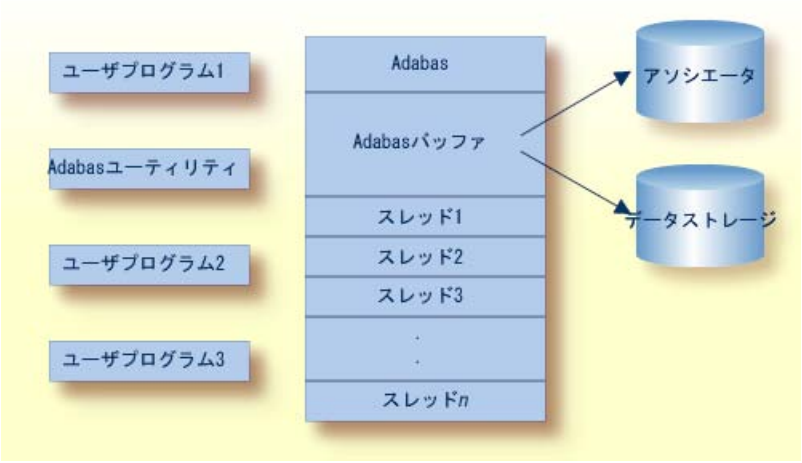
オペレーションのモード

Adabas では、シングルユーザーとマルチユーザーの 2 つのオペレーティングモードがサポートされます。

シングルユーザーモードは、ユーザープログラム（または Adabas ユーティリティ）が、Adabas ニュークリアスと同じパーティションまたはリージョンで実行された場合に有効になります。



マルチユーザーモードは、Adabas ニュークリアスが別のパーティションまたはリージョンに存在する場合に有効になります。これは最も効率的なモードであるため、推奨されるオペレーティングモードです。



マルチユーザーモードで Adabas を使用すると、リージョン間のコミュニケーションが Adabas によって処理されます。この場合、さまざまなオペレーティングシステムにより提供されるコミュニケーション機能が最大限に利用されます。

シングルユーザーモードでは、適切な Adabas ニュークリアス JCL をユーティリティまたはユーザープログラムのジョブコントロールに含める必要があります。

ADARUN の起動パラメータ

ADARUN コントロールステートメントを使って、Adabas 操作環境の定義を行い、Adabas を起動します。 ADARUN コントロールステートメントでも、Adabas ユーティリティを起動できます。 ADARUN

- すべてのデータベースI/Oおよび他のオペレーティングシステム依存の機能を実行する ADAIOR モジュールをロードします。
- ADARUN のパラメータステートメントを解釈した後、ADARUN パラメータの設定に従って必要な Adabas ニュークリアス、またはユーティリティモジュールのロードと変更を行います。
- Adabas に制御を渡します。

ADARUN ステートメントは、通常、一連のエントリから構成され、それぞれのエントリは1つ以上の ADARUN パラメータ設定を示します。このステートメントは、DDCARD（z/OS、z/VM、BS2000）データセットまたは CARD（VSE）データセットに指定します。

セッションの制御

Adabas では、Adabas、ユーザー、およびユーティリティセッションを監視および制御するいくつかの方法が提供されます。

- Adabas *オペレータコマンド*は、Adabas セッションまたはユーティリティ操作の実行中にオペレータコンソールから入力できます。
- ADADBS OPERCOM ユーティリティ機能は Adabas ニュークリアスにオペレータコマンドを発行できます。 次に、Adabas は、コマンド実行を確認するメッセージをオペレータに発行します。
- Adabas Online System（デモまたはフル機能版）を使用している場合、Adabas セッションがアクティブなときにメニューオプションやダイレクトコマンドを使用して、オペレータコマンドに対応した機能を実行させることもできます。

オペレータコマンドを使用して、Adabas やユーザーセッションの終了、ニュークリアスやユーティリティの表示、コマンドのログ記録、および Adabas のオペレーティングパラメータや状態の変更を行うことができます。

Adabas *ダイレクトコールコマンド*を使用して、ユーザーセッションのオープン／クローズを行うことも可能です。 ダイレクトコールコマンドの詳細は、「*ダイレクトコールインターフェイス*」を参照してください。

3 Adabas 設計

データベースシステムは、知識が豊富で経験に富んだユーザーのみが設計または使用することができる、複雑なデータ構造やデータ処理手順を伴うことが少なくありません。一方、Adabasの構造は非常にシンプルですが、操作の効率性、設計の容易さ、定義、およびデータベースの展開が格段に優れています。

このchapterでは、次のトピックについて説明します。

Adabas のエンティティ

Adabas では、*フィールド*が情報の論理的な最小単位（現在の給与など）で、この単位はユーザーにより定義され、参照されます。 *レコード*は、完結した情報のまとまりを構成する関連フィールドの集合を指します。例えば、従業員 1 人の全給与データがこれに該当します。 *ファイル*は、同じフォーマットを持つ関連レコードのグループを指します。ただし、例外もあるので、「*単一ファイル内の複数のレコードタイプ*」を参照してください。 データベースは、関連するファイルのグループを指します。

Adabas の制限

下の表は、各エンティティについて、メインフレーム Adabas がサポートする上限数を示しています。

エンティティ	最大値
データベース	65,535
データベースあたりのブロック数	2,147,483,646（4 バイト RABN 使用）
データベースあたりのファイル数	5,000 またはアソシエータのブロックサイズから 1 を引いた値のいずれかの少ない方
ファイルあたりのレコード数	4,294,967,294（4 バイト ISN 使用）
レコードあたりのフィールド数	926
非圧縮レコード長	オペレーティングシステムによって異なる
圧縮レコード長	データストレージブロックサイズ *スパンドレコード*は Adabas バージョン 8 以降でサポートされており、1 つの論理レコードを複数の物理レコードに分割します。分割後の各物理レコードは、1 個のデータストレージ（DS）ブロックより小さくなります。詳細は、「*スパンドレコードのサポート*」を参照してください。

Adabas のスペース管理

単一の Adabas データベースに割り当てられたディスクストレージスペースは、*論理* Adabas ファイルにセグメント化されます。 データベース内のスペース全体のうち、特定の部分が、各論理ファイルに割り当てられます。 ファイルのレコードを格納すると、スペースに空きがなくなる場合には、共通のフリースペースプールから追加のスペースが自動的に割り当てられます。このダイナミックなスペース割り当てと、解放されたスペースのダイナミックなリカバリを併用すると、特に操作しなくても Adabas データベースを長期間にわたって実行できます。

ディスクドライブ全体にわたってのデータベーススペースの分配は、スペースを複数の独立したデータセットに物理的にセグメント化することによって制御されます。 すべての物理データベーススペースが使用されて空きがなくなった場合、追加のデータセットがダイナミックに割り当てられるか、既存のデータセットのサイズが増加されるため、新しい物理ファイルは、データベース全体を再編成することなくロードされます。

データベースコンポーネント

データとアクセス構造の分離をサポートするため、Adabas ニュークリアスは次の 3 つのデータベースコンポーネントを使用します。

- 圧縮データ用のデータストレージ
- データ管理および取得用のアソシエータ
- 複合検索条件などの作業エリアであるワーク

このsectionでは、これらの各データベースコンポーネントについて説明します。

- データストレージ
- アソシエータ
- ワーク
- その他のコンポーネント

データストレージ

データストレージは、ブロックに分割されます。各ブロックは、3または4バイトの相対Adabasブロック番号（RABN）によりブロックの物理的な場所がコンポーネントの先頭から相対的に識別されます。 データストレージのブロックには、1つ以上の物理レコード、およびブロックのレコードを拡張するための予備スペースとしてパディングエリアが含まれています。

各物理レコードの先頭の 4 バイトに格納されている論理 ID には制御情報のみが含まれ、データブロックに格納されます。この内部シーケンス番号（ISN）により、各レコードは一意に識別され、変更されることはありません。 レコードが追加されると、既存の最も大きな ISN に 1 を加えた ISN が割り当てられます。 レコードが削除されると、そのレコードの ISN は再利用されます。これは Adabas に再利用を設定した場合のみ行われます。 ISN の再利用は、一部の検索に

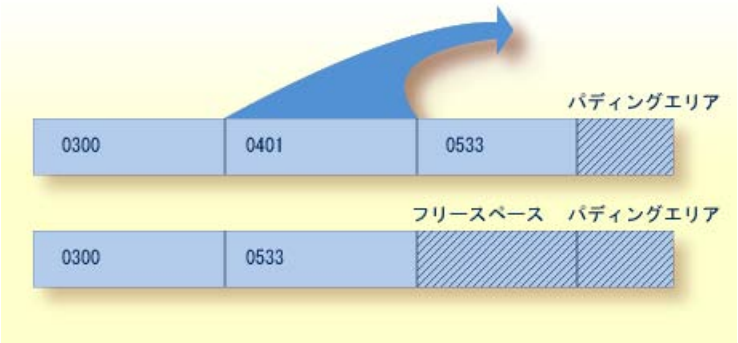
おいてシステムのオーバーヘッドを削減するため、頻繁に追加や削除が行われるレコードを含むファイルには ISN の再利用が推奨されます。

各ファイルごとに、各ブロックの 1～90 %（デフォルトは 10 %）がパディングエリアとして割り当てられます。この値は、予想される更新の量および種類によって決定されます。 この予約済みスペースにより、レコードは他のブロックに移行することなく拡張できるため、システムのオーバーヘッドの削減が可能になります。

- 空きスペースおよびスペースの再使用
- 圧縮

空きスペースおよびスペースの再使用

レコードが大きくなり、ブロックに収まらなくなった場合、新しい場所に移行されます。レコードが移行または削除されると、最後のレコードとパディングエリアの間に空きスペースができます。 次の図は、ISN 0401 のレコードが大きくなりブロックに収まらなくなったため、他のブロックに移行されて作成された空きスペースを示しています。



空きスペースを再利用するように Adabas を設定できます。 スペースを再利用すると、Adabas は検索中により少ない物理ブロックを読むだけで済み、コンピュータの処理時間が節約されます。 これは、すべてのファイルに対して推奨されます。

圧縮

データ圧縮により、必要なストレージの量が大幅に削減されます。また、1回の物理的な転送で送信できる情報量が増加するため、I/O 効率が向上します。

Adabas では、データレコードを圧縮形式で保持します。 圧縮には、数種類のオプションがサポートされています。

- デフォルトの圧縮
- 空値省略
- 固定フォーマット
- フォワードまたは接頭辞インデックス圧縮

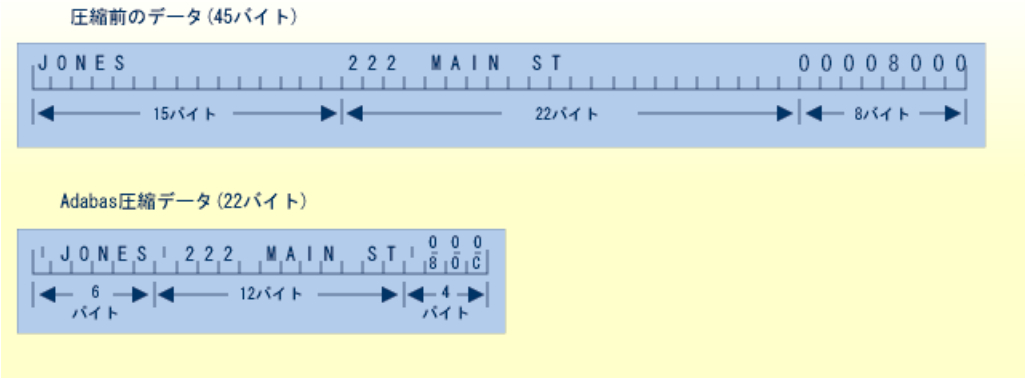
最初の3つのオプションは、フィールドレベルで定義および実行されます。空値省略および固定フォーマット圧縮はフィールドオプションとして追加されます。

4 番目のオプションのフォワードまたは接頭辞インデックス圧縮は、アソシエータのインバーテッドリストに含まれるディスクリプタ値を圧縮します。 このオプションは、ファイルまたはデータベースレベルで実装できます。この場合、ファイルごとに別々の設定を行うことができます。ファイルレベルの設定の方がデータベースレベルの設定よりも優先度が高くなります。フォワードインデックス圧縮オプションは、ADALOD ユーティリティを使用して設定されます。また、ADAORD ユーティリティを使用して変更できます。 この圧縮オプションについては、の「*インバーテッドリスト*」で詳細に説明されています。

フィールドオプションとして追加される空値省略および固定フォーマットオプションについては、「*データ圧縮オプション **FI** および **NU***」で説明されています。

デフォルトの圧縮は、英数字フィールドの末尾の空白、およびバイナリフィールドの先行のゼロを削除します。 フィールドの先頭の包括長さバイト（ILB）は、ILB 自身を含む、格納されている総バイト数を示しています。 したがって、20 文字長で定義されデフォルト圧縮が設定された "名前 "フィールドに "Susan" が入力された場合、格納されるサイズは、文字が 5 バイト、ILB が 1バイトの合計6バイトとなります。 また、レコードに含まれる空値フィールドは格納されず、空値フィールドは 1 バイトの空値フィールドカウンタ（EFC）に置き換えられます。 Adabas では、16 進数の 1 バイトに最大 63 個の連続した空値フィールドを格納できます。

多くの Adabas ファイルでは、未加工データが使用するスペースの 50～60 %のスペースしか必要ではありません。アソシエータに格納されるアクセス構造用に約25 %が追加されても、Adabas に必要なストレージは、データの圧縮を行わない従来のファイルストレージまたは DBMS よりも少ない容量で済みます。



アソシエータ

アソシエータは、データストレージ内のデータにアクセスするために必要な構造を保存するときに使用するまとまったユニットです。 アソシエータには次の要素が含まれます。

- データベース用の 2 つの*ジェネラルコントロールブロック（GCB）*。 GCB は、データベースの物理的な特性に関する情報を提供します。例えば、データベースID（DBID）、ロードされたファイル数、アソシエータ、データストレージおよびワークエクステントの数、アソシエータ、データストレージおよびワークデバイスのタイプ、システムファイル情報、データストレージスペーステーブル（DSST）のエクステント、データベースのバージョンインジケータなどが含まれます。
- 各ファイル用の個別の*ファイルコントロールブロック（FCB）*。 FCB は、データベースファイルの物理的な特性および関連する RABN を識別します。 その内容には、ファイル名、ファイル番号、現在のファイルステータス、ISN の再使用設定、スペースの再使用設定、MINISN および MAXISN 設定、最初の空き ISN、およびファイルに対する更新回数が含まれます。 さらに、最初の RABN、最後の RABN、および最初の未使用 RABN が FCB に格納されます。
- 各ファイル用のフィールド定義テーブル（FDT）、物理的にカップリングされたファイル用のカップリングリストなど、データベースを制御および維持するために必要なすべてのテーブル。 FDT の詳細は、「*レコードおよびフィールドの定義*」を参照してください。 物理的にカップリングされたファイルの詳細は、*カップリングファイル*」を参照してください。
- データベース内の各ファイルに含まれるディスクリプタ用のインバーテッドリスト、および各ファイル用のアドレスコンバータ。
- ファイルにスパンドレコードが使用されている場合はセカンダリアドレスコンバータ。

インバーテッドリスト

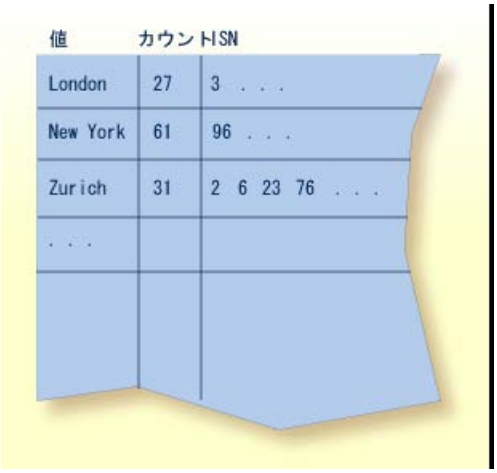
インバーテッドリストは、Adabas 検索コマンドの解決および論理順のレコード読み込みに使用されます。キーフィールドまたは *ディスクリプタ*として設計された Adabas ファイル内のフィールドごとに構築および維持されます（「*ディスクリプタオプション **DE**、**UQ**、および **XI***」を参照）。 ISN ではなく、ディスクリプタ値の順で並べられているため、*インバーテッドリスト*と呼ばれます。 このリストは、ノーマルインデックス（NI）および、14 個までのアッパーインデックス（UI）で構成されます。

特定のディスクリプタに対するインバーテッドリストのノーマルインデックス（NI）には各値に対するエントリが含まれます。 エントリには、値、値が含まれるレコードの数、およびそれらのレコードの ISN が含まれます。

検索の効率を向上するため、アッパーインデックス（UI）レベルが必要に応じて自動的に Adabas によって作成されます。各レベルは直下レベルのインデックスを管理します。 管理対象の NI と同様に、最初のレベルの UI には、各インデックスブロックの 1 つのディスクリプタに対するエントリのみが含まれます。その他のすべての UI レベルには、各インデックスブロックのすべてのディスクリプタのエントリが含まれます。 UI には最小サイズのスペースが必要です。最小サイズとは 2 ブロックです。

Note: Adabas ダイレクトアクセスメソッド（ADAM）機能を使用すると、インバーテッドリストにアクセスせずに、データストレージから直接レコードを取り出すことができます。 レコードが格納されているデータストレージブロック番号は、レコードの ADAM キーに基づきランダマイジングアルゴリズムを使用して計算されます。 ADAM は、アプリケーションプログラム、クエリおよびレポートライタの機能に対して完全に透過的に使用されます。詳細は、「*Adabas ダイレクトアクセスメソッド（ADAM）を使用したランダムアクセス*」を参照してください。

次の図は、顧客ファイル内のディスクリプタ CITY に対する一般的なノーマルインデックスを示しています。



この例では、CITY が Zurich であるレコードが 31 個あることを示しています（これらのレコードの ISN は 2、6、23、76 と続きます）。

フォワード（またはフロント、接頭辞）インデックス圧縮は、インデックス値から余分な接頭辞情報を削除します。1つのインデックスブロック内で、最初の値は完全な長さで保存されます。後続のすべての値については、前者と共通の接頭辞が圧縮されます。 インデックス値は、次のように表現されます。

```
<l,p,value>
```

ここでは次の内容を表しています。

p	前の値の接頭辞と等しいバイト数です。
l	p バイトを含む残りの値の排他的な長さです。

例

圧縮前	圧縮後
ABCDE	6 0 ABCDE
ABCDEF	2 5 F
ABCGGG	4 3 GGG
ABCGGH	2 5 H

インデックス値の類似点とファイルサイズに基づいて、インデックス値を圧縮するかどうかを判断します。

- インデックス値が類似するほど、圧縮の結果が良くなります。
- 小さいファイルは節約されるスペースの絶対量が小さくなるので、圧縮には向いていませんが、大きいファイルはインデックス圧縮に向いています。

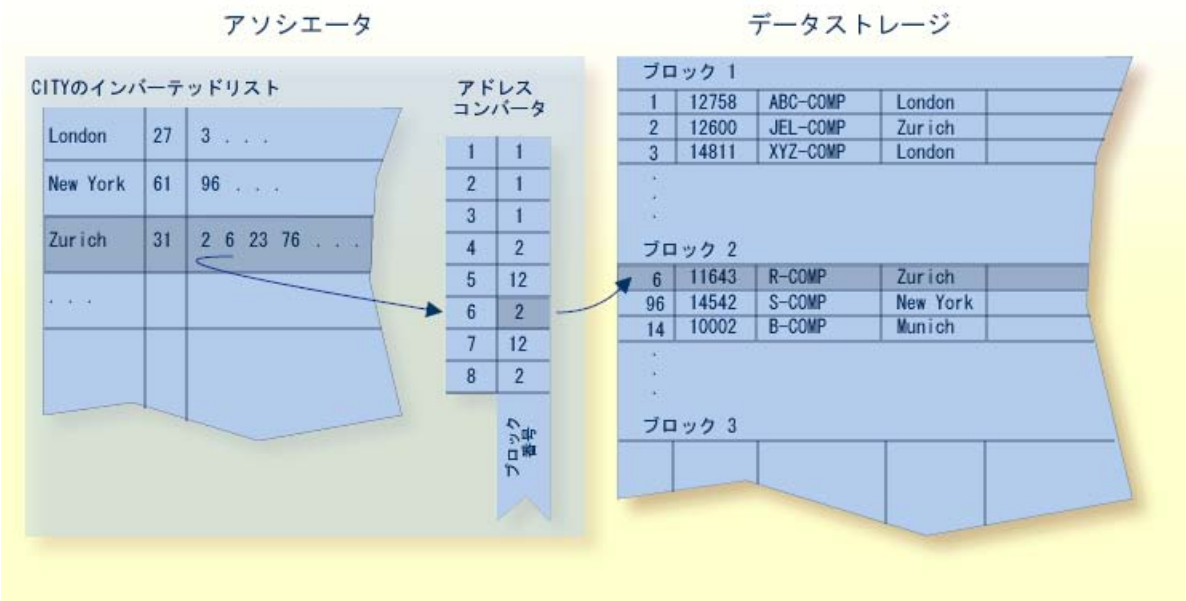
ファイルのインデックス値の圧縮率がよくない場合がありますが、それでもインデックスが圧縮されていれば、圧縮されていない場合よりも必要になるインデックスブロックが少なくなります。

アドレスコンバータ

アドレスコンバータはレコードの物理的な場所を特定します。 インデックスは、レコードの論理ID（ISN）を、レコードが格納されているデータストレージブロックの相対Adabasブロック番号にマッピングします。 スパンドレコードでは、セカンダリアドレスコンバータを使用して、セカンダリレコードが格納されているデータストレージブロックのRABNにセカンダリISNをマッピングします。 スパンドレコードの詳細は、「*スパンドレコード*」を参照してください。

アドレスコンバータには、ISNの順に並んだRABNが含まれています。 アドレスコンバータには、RABNのみが実際に格納されているため、ISNはその相対的な位置で識別されます。

次の図は、インバーテッドリスト、アドレスコンバータおよびデータストレージの関係を示しています。 例えば、ISNが6のレコードの物理的な場所を特定するには、AdabasはISNをアドレスコンバータのインデックスとして使用します。 アドレスコンバータの6番目のエントリは2です。 したがって、ISN 6は、このファイルのデータストレージ内の物理ブロック2にあることがわかります。



レコードが移動したり削除された場合、Adabasは自動的また透過的にアドレスコンバータを更新します。

レコードのISNは変わることがなく、また物理ブロックアドレスはアドレスコンバータのエントリにのみ格納されているため、レコード自身がデータストレージ内で移動するときに、アドレスコンバータの更新は1回だけで、レコードのアクセスパスの増やす必要はありません。

レコードに多くのディスクリプタが定義されていても、各ディスクリプタに対するインバーテッドリストにはISNが含まれているため変更する必要がありません。

この処理は、いかにAdabasが検索の複雑さに関係なく迅速かつ効率的に、データストレージ内にポインタ情報を格納せずに実行できるかを示しています。

ワーク

ワークワークエリアは、次の 4 つのパートに情報を格納します。

パート	格納するアイテム
1	自動再スタートと自動バックアウト用のルーチンで必要となるデータ保護情報を格納します。 詳細は、「*バックアウト、リカバリ、および再スタート*」を参照してください。
2	検索コマンドの中間結果（ISN リスト）
3	検索コマンドの最終結果（ISN リスト）
4	2 フェーズコミット処理に関連したデータ

その他のコンポーネント

- ソートエリアと一時エリア
- ログ

ソートエリアと一時エリア

さらに、特定の Adabas ユーティリティ（ADAINV と ADALOD）では、データのソートと中間ストレージ用に、2つのデータセット（ソートデータセットと中間データセット）が必要になります。 その他のユーティリティの特定の機能では、中間ストレージ用の中間データセットを必要とします。

中間データセットおよびソートデータセットのサイズは、実行するユーティリティの機能により異なります。 これらのデータセットは、ジョブの実行中に割り当てと解放を行うか、永久的なデータセットを割り当てて再利用することができます。

ログ

Adabas では、次のオプションログが使用されます。

- コマンドログ（CLOG）は、発行される各 Adabas コマンドのコントロールブロックの情報を記録します。 CLOG によって作成される監査記録は、デバッグおよびリソース使用状況の監視に使用できます。 シングル、デュアル、またはマルチ（2～8）データセットを使用できます（マルチデータセットの使用をお勧めします）。

  ADARUN の CLOGLAYOUT=8 パラメータを使用して作成された Adabas 8 コマンドログのタイムスタンプは、マシンタイム（GMT）で格納されます。一方、CLOGLAYOUT=5 タイムスタンプは常にローカルタイムで格納されます。 CLOGLAYOUT=8 を指定してコマンドログを作成した場合、LORECX レコードレイアウトには、CLOG レコードが書き込まれた時点のマシンタイムとローカルタイムの差が格納された時間差フィールドが含まれます。 このフィールドにより、コマンドログレコードのローカルタイムの計算が可能になります。

  タイムスタンプ形式の違いにより、異なる CLOGLAYOUT 設定を用いて作成されたコマンドログを混在させたり、マージしたりすることはお勧めしません。 クラスタ環境の Adabas

ニュークリアスでは、特に推奨されません。 詳細は、「*CLOGLAYOUT コマンドログレイアウト*」を参照してください。

- プロテクションログ（PLOG）は、データベースに変更が生じたときに、レコードやその他のエレメントの変更前イメージと変更後イメージを記録します。 PLOG 情報は、再スタート後のデータベースのリカバリに使われます（最新の完了したトランザクションまたはETまで）。シングル、デュアル、またはマルチ（2〜8）データセットを使用できます（マルチデータセットの使用をお勧めします）。
- リカバリログ（RLOG）は、Adabas Recovery Aid がリカバリジョブストリームを構成するために使用する追加情報を記録します。 詳細は、**ADARAI** ユーティリティに関する説明を参照してください。

データベースファイル

すべてのデータベースには、システムファイルとデータファイルが含まれます。 データファイルは、通常は必要な各レコード構造に対応して作成されます。つまり、識別されるすべての関連したフィールドのセットに対応して作成されます。

ファイルは、ADALODユーティリティを使用してデータベースにロードされます。ファイル番号はデータベース内で一意である必要があります。また、MAXFILES パラメータでデータベースに対して定義された最大数を超えることはできません。 チェックポイント、セキュリティ、トリガ、およびシステムファイルは、2バイトファイル番号を持つことができますが、5000を超えることはできません。物理的にカップリングされたファイルには、255を超える番号のファイルを含めることはできません。 ファイル番号は、ユーザーによってどのような順序にも割り当てることができます。

このsectionでは、異なるタイプのデータベースファイルについて説明します。

- システムファイル
- カップリングファイル
- パフォーマンス向上のためのファイルの構成

システムファイル

Adabas は特定のファイルを使用して、システム情報を格納します。 ADALOD ユーティリティの FILE パラメータを使用して、Adabas システムファイルを次のいずれかとして指定することができます。

チェックポイント	Adabas チェックポイントファイル
SECURITY	Adabas セキュリティファイル
SYSFILE	Adabas システムファイル
TRIGGER	Adabas トリガファイル

カップリングファイル

ファイルカップリングにより、単一の検索コマンドを使用して、別ファイル内の指定された値を含むレコードに関連付けられた（カップリングされた）あるファイルからレコードを選択できます。

- 物理カップリング
- 論理カップリングまたはソフトカップリング

物理カップリング

ファイル番号が255以下の2つのファイルは、同じフォーマットと長さの定義を持つ共通のディスクリプタ（「*ディスクリプタオプション DE、UQ、および XI*」を参照）が、両方のファイルに存在すれば、どのようなファイルでもカップリングできます。 1 つのファイルは、18 個まで他のファイルとカップリングが可能ですが、2 つのファイル間には同時に 1 個のカップリング関係だけが許されます。 自分自身とのカップリングは行うことはできません。

ファイルがカップリングされると、カップリングされた両方のファイルに対してカップリングリストがアソシエータ内に作成されます。 ファイルカップリングは、2つのカップリングリストが各カップリング関係に対して作成され、各リストは他のファイルへカップリングされる ISN を含むため、階層的ではなく双方向的と言えます。

物理カップリングリストが作成されると、両方のファイルのすべてのキーフィールドを検索条件内で使用することができます。

ファイルの更新頻度が高い場合、物理カップリングにより、オーバーヘッドの量が多くなる可能性があります。 いずれかのファイルのレコードが追加／削除されたり、カップリングの元になるディスクリプタの更新が起きると、カップリングリストも更新しなければなりません。

物理カップリングは、次の場合の情報取得システムに有効です。

- ファイルがほとんど変更されない
- カップリングリストの追加オーバーヘッドが、問い合わせ作成の容易さの向上と比較すると問題とならない

- ファイルが小さく、主に問い合わせに向いている

論理カップリングまたはソフトカップリング

検索条件に、ファイル間のリンケージに使用されるフィールドを指定して、複数ファイルの問い合わせを行うこともできます。これにより、Adabasがすべての必要な検索、読み込み、内部リストのマッチング操作を行います。

この方法は、物理的なファイルカップリングを必要としないことから、論理カップリングまたはソフトカップリングと呼ばれます。論理カップリングには読み込みコマンドが必要ですが、カップリングリストによるオーバーヘッドを避けることができるため、通常ではより効果的な方法です。

パフォーマンス向上のためのファイルの構成

Adabas データベースの構成がレコードタイプごとにファイルが 1 つずつ分かれる場合は、データベースに必要なアプリケーション機能はどのようなものでもサポートされ、問い合わせに対応するうえで操作方法が簡単になりますが、最良のパフォーマンスは得られません。

- Adabas ファイルの数が増えるにつれ、Adabas コールの数も増えます。 各 Adabas コールは解釈、整合性チェックが必要であり、マルチユーザーモードではスーパーバイザーコール（SVC）およびキューイングのオーバーヘッドが必要になります。
- 各ファイルから、少なくとも1回ずつインデックス、アドレスコンバータ、データストレージブロックにアクセスするために入力／出力（I/O）オペレーションが必要になるだけでなく、レコードタイプごとに 1 ファイルを持つ構造はバッファプールを必要とします。 十分なバッファスペースが用意されていない場合、将来の要求に対して必要となるブロックが上書きされてしまう可能性があります。

重要なプログラムによって使用される Adabas ファイルの数は次の方法で削減できます。

- マルチプルバリューフィールドおよびピリオディックグループの使用（「*フィールドレベル*」を参照）
- 複数の物理ファイルを 1 つの論理（拡張）ファイルにリンクする
- Adabas ファイル内に複数タイプのレコードを含める
- Adabas（マルチクライアント）ファイルに複数ユーザーのカテゴリのレコードを含める
- データの重複を制御してリソースの使用率を抑える

このsectionでは、次のトピックについて説明します。

- **拡張ファイル**
- **単一ファイル内の複数のレコードタイプ**
- **マルチクライアントファイル**

- データの冗長性の制御

拡張ファイル

同じタイプのレコードが大量に存在する場合は、複数の物理ファイルに分散させる必要があります。

アクセスするファイル数を減らすために、同一フォーマットのレコードをもった複数の物理ファイルをリンクして、1つの論理ファイルに結合することができます。このようなファイル構造のことを拡張ファイルと呼び、この拡張ファイルを構成する物理ファイルのことをコンポーネントファイルと呼びます。 拡張ファイルは、それぞれに異なった論理 ISN の範囲を持った 128 までのコンポーネントファイルによって構成されます。 1 つの拡張ファイルのレコード数は、4,294,967,294 までです。

Note: Adabas では、より大きいファイルサイズ、より多くの Adabas 物理ファイルやデータベースがサポートされるようになったので、拡張ファイルの必要性はほとんどなくなりました。

アプリケーションプログラムから論理ファイルがアクセスされても（ファイルのアドレスは拡張ファイルの基本コンポーネントの番号、あるいはアンカーファイル）、Adabas では基準フィールドとして定義したフィールドのデータに基づいて正しいファイルが選択されます。このフィールドのデータは1つのコンポーネントファイル内でのみ、レコードをユニークに識別する特性を持ちます。アプリケーションが拡張ファイルを更新するとき、Adabas は更新するコンポーネントファイルを決定するために書き込むレコードの基準フィールドのデータを検索します。 拡張ファイルデータを読み込むとき、Adabas は論理 ISN をキーとして使用し、正しいコンポーネントファイルを見つけます。

単一ファイル内の複数のレコードタイプ

単一の物理レコード内で複数のレコードタイプを定義できます。各レコードタイプは、ファイルに対して定義されたフィールドのサブセットで構成される論理レコードです。 指定されたタイプに属さないフィールドは空値省略されます。

レコードタイプは次の方法で Adabas により識別されます。

- 別のタイプとして区別できる値でレコードタイプフィールドを定義する
- タイプを区別できる既存のフィールドの値を使用する。例えば、2 つのタイプを区別するには、両方のタイプに共通なフィールドをゼロ値で1つのタイプとして識別し、同じフィールドをゼロ以外の値で他のタイプとして識別する方法があります。

マルチクライアントファイル

複数のユーザーまたはユーザーのグループが使用するレコードを、マルチクライアントとして定義された単一の Adabas 物理ファイルに格納することができます。 マルチクライアント機能では、各レコードに内部的なオーナー ID をつけることで、物理ファイルを複数の論理ファイルに分割します。

オーナー ID は、ユーザー ID に対して割り当てられます。 1 つのユーザー ID は、1 つのオーナー ID しか持つことができませんが、同じオーナー ID が複数のユーザーに所属していてもかまいません。 各ユーザーは、ユーザーのオーナー ID に関連付けられたレコードのサブセットだけにアクセスできます。

Note: RACF、CA-ACF2、または CA-Top Secret などの外部セキュリティパッケージがインストールされている場合でも、ユーザーは変わらず Natural ETID またはログオン ID で識別されます。

マルチクライアントファイルへのすべてのデータベース要求は、Adabas ニュークリアスにより処理されます。

データの冗長性の制御

物理的な冗長性を持たせると、ストレージの必要量が増えますが、パフォーマンスが向上し、複雑さを減らすことができます。 例えば、データベースの顧客注文ファイルに顧客と注文の情報を格納して、在庫ファイルに商品の詳細情報を格納し、請求書発行プログラムが顧客注文データに加えて商品の詳細情報を必要とする場合、商品の詳細情報のコピーを顧客注文ファイル内に格納すると、パフォーマンスが向上する可能性があります。

論理的な冗長性を持たせた場合も、ストレージの必要量が増えますが、複雑さを減らすことができます。 これは、あるファイルに格納する処理結果と別ファイルを関連付けることになるので、同じデータが、物理的に 2 か所に格納されるのではなく、別のファイルの内容として存在することになります。

物理的および論理的な冗長性により、更新プログラムの実行速度が遅くなります。 あるファイルを変更すると別ファイルのレコードにも影響を与える場合、別ファイルも更新する必要があるため、パフォーマンスを極度に低下させることがあります。 冗長性は、静的なデータ、またはほとんど更新されないデータのみで使用する必要があります。マルチプルバリューフィールド、ピリオディックグループ、および単一ファイル内の複数のレコードタイプを使用して、データの冗長性を制御できます。

レコードおよびフィールドの定義

Adabas では、物理ファイル内のレコード構造および各フィールドの内容が、アソシエータ内に格納されているフィールド定義テーブル（FDT）で示されます。 データベースファイルごとに 1 つの FDT があります。 FDT は、Adabas コマンド実行中にファイル内のすべてのフィールド（またはグループ）の論理構造と特徴を判断するために Adabas が使用します。

*スパンドレコード*は Adabas バージョン 8 以降でサポートされており、1 つの論理レコードを複数の物理レコードに分割します。分割後の各物理レコードは、1 個のデータストレージ（DS）ブロックより小さくなります。詳細は、「*スパンドレコードのサポート*」を参照してください。

このsectionでは、次のトピックについて説明します。

- レコード構造および FDT
- フィールドレベル
- フィールド名
- フィールド長およびデータフォーマット
- フィールドオプション
- 特殊なフィールドおよびディスクリプタフィールド

レコード構造および FDT

FDT には、ファイルのフィールドがレコードの物理的な順番に従って一覧表示されます。FDT の役割は、ファイルのレコードに対する簡易インデックスであり、ファイルのフィールド、サブフィールド、スーパーフィールド、およびディスクリプタ（照合ディスクリプタ、サブディスクリプタ、スーパーディスクリプタ、ハイパーディスクリプタ、およびフォネティックディスクリプタを含む）を定義します。 最小 1 個、最大 926 個のフィールド定義が指定できます。

各フィールドに関して、レベル、名前、長さ、フォーマット、オプション、特殊なフィールドおよびディスクリプタの属性についての情報が含まれます。

```
FIELD DESCRIPTION TABLE

       I      I        I        I              I
 LEVEL I NAME I LENGTH I FORMAT I  OPTIONS     I         PARENT OF
       I      I        I        I              I
 ------I------I--------I--------I--------------I----------------------------I
       I      I        I        I              I                            I
   1   I  AA  I    8   I    A   I DE,UQ        I                            I
   1   I  AB  I        I        I              I                            I
   2   I  AC  I   20   I    A   I NU           I                            I
   2   I  AE  I   20   I    A   I DE           I  SUPERDE,PHONDE            I
   2   I  AD  I   20   I    A   I NU           I                            I
   1   I  AF  I    1   I    A   I FI           I                            I
   1   I  AG  I    1   I    A   I FI           I                            I
```

```
    1   I  AH  I    6   I    U   I DE          I                          I
    1   I  A2  I        I        I             I                          I
    1   I  AO  I    6   I    A   I DE          I  SUBDE,SUPERDE           I
    1   I  AQ  I        I        I PE          I                          I
    2   I  AR  I    3   I    A   I NU          I  SUPERDE                 I
    2   I  AS  I    5   I    P   I NU          I  SUPERDE                 I
    1   I  A3  I        I        I             I                          I
    2   I  AU  I    2   I    U   I             I  SUPERDE                 I
    2   I  AV  I    2   I    U   I NU          I  SUPERDE                 I
```

FDT内のフィールドの順番により、レコードの構成およびアクセスの効率が決まります。フィールドの順番を決める際には、次の要因を考慮する必要があります。

- 頻繁にアクセスされるフィールドをFDTの先頭に置く必要があります。この方法を応用すれば、Adabasはフィールドのアクセス時にすべてのレコードを読み取る必要が無くなるため、CPUタイムが削減されます。
- 同時に頻繁にアクセスされるフィールドを、1つのグループフィールドに割り当てます。
- *常に*同時にアクセスされるフィールドは、単一のフィールドとして定義します。このように定義しておくと、圧縮や問い合わせ言語の使用が制限されることがありますが、内部処理が効率化されフォーマットバッファが短くなるため、処理時間が短縮されます。
- 必要に応じて、頻繁に空白になるフィールドは、FDT内で順番に並べて、デフォルト圧縮または空値省略を使用します。
- 数値フィールドは、最もよく使用されるフォーマットでロードします。

フィールドレベル

FDT内の2つまたはそれ以上の連続したフィールドに同時に頻繁にアクセスする場合、グループフィールドを定義して、これらのフィールドをまとめて参照することができます。フィールドのレベルとAdabasショートネーム以外に、グループフィールドに定義された属性はありません。このフィールドは、FDT内のメンバフィールドの直前にあります。メンバフィールドをグループフィールドに割り当てる際には、より高いフィールドレベル番号を使用します。Adabasは7個までのフィールドレベルをサポートしています。ユーザープログラムは、各メンバフィールドに個別にアクセスできます。グループフィールドを参照して、すべてのメンバフィールドに同時にアクセスすることもできます。

例えば、「*レコードおよびフィールドの定義*」セクションのフィールド定義テーブル（FDT）の図では、フィールドABがグループフィールドとして定義され、レベル1が割り当てられています。フィールドAC、AE、およびADはレベル2に割り当てられ、これらのフィールドがグループフィールドABに属していることを示しています。次のフィールドAFはレベル1に割り当てられ、このフィールドがABグループの一部ではないことを示しています。ユーザープログラムは、AC、AE、およびADの各フィールドに個別にアクセスできます。または、グループフィールドABを参照して、これらのフィールドに同時にアクセスできます。

グループフィールドは、その構成フィールドが複数の値を持つことができる場合（図のフィールドAQ）、*ピリオディックグループフィールド*として割り当てることができます。

フィールド名

フィールドは、2 文字の Adabas ショートネームにより Adabas に識別されます。このショートネームは最初の文字が英字、2 番目の文字は英数字でなければなりません。このうち E0 から E9 までは予約済みです。また特殊文字は使用できず、ファイル内でユニークである必要があります。 Adabas では、自動的にショートネームがフィールドに割り当てられますが、手動で割り当てることもできます。 Adabas では、内部でショートネームが使用され、このショートネームを使用して実際にフィールドにアクセスします。

フィールド長およびデータフォーマット

フィールドは、長さを固定または可変にでき、値には英数字、バイナリ、固定小数点、浮動小数点、パック／アンパック 10 進数、または文字列の各フォーマットを適用できます。

フィールドの長さ（単位はバイト）およびフォーマット（1 文字のコードで表現）により、コマンド処理中に Adabas が使用する標準（デフォルト）を定義します。この標準フォーマットは、フィールドの読み込み、更新時に、特にユーザーからの優先指定がない場合に使用されます。

フィールドの標準長がゼロの場合、そのフィールドは可変長フィールドとみなされます。 標準フォーマットはフィールドに必ず指定しなければなりません。指定したフォーマットに従って、フィールドに対するデフォルトの圧縮処理のタイプが決められます。

指定できるフィールド長の最大値はフォーマット値によって異なります。

フォーマット	フォーマットの説明	最大長
A	英数字（左づめ）：「*ロング英数字オプション **LA***」のロング英数字（LA）オプション、および「*ラージオブジェクトオプション **LB***」のラージオブジェクト（LB）オプションも参照してください。	253 バイト
B	2 進数（右詰め、符号なし、正の数）	126 バイト
F	固定小数点（右詰め、符号あり、一般的な形式で正の値、2 の補数形式で負の値）	4 バイト（常に 2 または 4 バイト）
G	浮動小数点（一般形式、符号あり）	8 バイト（常に 4 または 8 バイト）
P	パック 10進数（右詰め、符号あり）	15 バイト
U	アンパック 10進数（右詰め、符号あり）	29 バイト
W	ワイド文字（左づめ）：「*ロング英数字オプション **LA***」のロング英数字（LA）オプションも参照してください。	253 バイト

フィールドオプション

フィールドオプションは、2文字のコードで指定します。指定する順番は任意で、コンマで区切ります。

コード	オプション	参照セクション
DE	フィールドがディスクリプタ（キー）であることを示します。	*ディスクリプタオプション DE、UQ、および XI*
FI	フィールドは、固定ストレージ長です。値は、内部的な長さバイトなしで格納されます。圧縮はされません。定義したフィールド長よりもフィールドを長くすることはできません。	*データ圧縮オプション FI および NU*
LA	英数字またはワイド文字。可変長フィールドは、長さが16,381バイトまでの値を格納できます。	*ロング英数字オプション LA、LA および LB フィールドの違い*
LB	英数字フィールドには 2,147,483,643（約 2 GB）までのデータを格納できます。	*ラージオブジェクトオプション LB、LA および LB フィールドの違い*
MU	単一レコード内で、フィールドに最大約 65,534 個の値を含めることができます。	*MU および PE オプションおよびフィールドタイプ*
NB	LA および LB フィールドのどちらからも、末尾の空白を削除（圧縮）するべきではありません。このオプションの指定には、NU または NC の指定も必要とします。	*空白圧縮オプション NB*
NC	フィールドには、空値を含めることができます。SQL では、このフィールドには値がないと解釈されます。つまり、フィールドの値は未定義になり、カウントされません。	*SQL 互換性オプション NC および NN*
NN	NC オプションを定義したフィールドには、必ず定義した値が存在しなければなりません。SQL の空値を含むことできません。	*SQL 互換性オプション NC および NN*
NU	フィールドの空値は省略されます。	*データ圧縮オプション FI および NU*
NV	英数字またはワイド文字フィールドは、変換されずにレコードバッファ内で処理されます。	*エンコード変換オプション NV*
PE	このグループフィールドは、FDT 内の連続フィールド（1つ以上の MU フィールドを含めることができる）を定義します。これらのフィールドはレコード内で同じ順番で繰り返されます（最大約 65,534 回）。	*MU および PE オプションおよびフィールドタイプ*
UQ	フィールドはユニークディスクリプタとなります。つまり、ファイル内のレコードごとに、ディスクリプタの値を別々にする必要があります。	*ディスクリプタオプション DE、UQ、および XI*
XI	このフィールドでは、オカレンス（インデックス）番号が、ピリオディックグループ（PE）に設定されたユニークディスクリプタ（UQ）オプションから除外されます。	*ディスクリプタオプション DE、UQ、および XI*

ディスクリプタオプション DE、UQ、および XI

ディスクリプタは検索キーです。DEオプションは、フィールドがディスクリプタであることを示しています。 DE が指定されている場合のみ、UQ オプションを指定できます。UQ オプションは、DE フィールドが、ファイル中のレコードごとにそれぞれ異なる値（ユニークな値）を持つことを示します。UQフィールドがMUフィールドまたはピリオディックグループ内のフィールドでもある場合、同一レコード内で、そのフィールドの値が複数回出現する場合がありますが、異なるレコード間ではユニークである必要があります。 DE フィールドに応じて、アソシエータのインバーテッドリストにエントリが作成され、ディスクスペースが追加されたり、必要なオーバーヘッドが処理されたりします。

選択条件の中では、あらゆるフィールドを使用できます。 広範囲に使用されるフィールドが検索条件としてディスクリプタ（キー）に定義されると、選択処理が大幅に速くなります。これは、Adabas が、データストレージからレコードを読み込まずに、インバーテッドリストから直接、ディスクリプタの値にアクセスできるためです。

ディスクリプタフィールドは、検索コマンドでソートキーとして使用したり、論理的な順次読み込み処理（昇順または降順の値）を制御する方法として使用したり、ファイルカップリングの基準として使用したりすることができます。

ファイル中のいかなるフィールドもディスクリプタとして定義でき、その数には制限がありません。 マルチプルバリューフィールド、またはピリオディックグループ内のフィールドがディスクリプタとして定義された場合、レコードに対して複数のキーの値が生成されます。 キー検索は、ピリオディックグループ内の特定のオカレンスに限定される場合があります。

ディスクリプタフィールドが、ピリオディックグループ内のフィールド（PE フィールド）の一部である場合、グループインデックスは、インデックス内のディスクリプタ値の一部とみなされます。 これにより、値だけでなくグループインデックスも検索対象にすることができます。 デフォルトでは、任意の値と、レコードのあるオカレンスのグループインデックスの組み合わせは、同じ値と、別のレコードの別のグループインデックスの組み合わせとは異なるとみなされます。 グループインデックスが異なるため、これらの 2 つのオカレンスは "一意性" 条件に違反しません。 一意性条件からグループインデックスを除外したい場合は、XI オプションを使用します。 XI オプションは、ピリオディックグループ内のユニークディスクリプタに使用され、一意性の定義からオカレンス（インデックス）番号を除外します。

インバーテッドリストはディスクスペースおよび更新オーバーヘッドを必要とするため、ディスクリプタオプションは注意深く使用する必要があります。特に、ファイルが大きく、ディスクリプタとして見なされるフィールドが頻繁に更新される場合は注意が必要です。 例えば、ディスクリプタとして使用されているピリオディックグループのインバーテッドリストは、すべてのオカレンスを含むために非常に大きくなる可能性があります。

ディスクリプタは、ファイルの作成時に定義できます。または、Adabas ユーティリティを使用してファイルの作成後に定義することもできます。 ディスクリプタの定義はレコード構造から独立していて影響を受けないため、ディスクリプタは、データベースの再構築や再構成を必要とせずに、いつでも作成または削除が可能です。

ただし、ディスクリプタフィールドがレコード構造の最初に配列されず、論理的に物理レコードの終了を超えた場合、パフォーマンス上の理由から、そのレコードに対するインバーテッドリストエントリが生成されないことに注意してください。 このケースでインバーテッドリストエントリを生成するには、ファイルのアンロード（SHORT モード）、圧縮解除、そして再ロードが必要です。あるいは、アプリケーションプログラムを使用してフィールドをファイルの各レコードの最初に再配列します。

フィールドの一部をサブディスクリプタとして定義することができます。また、フィールドやその一部を組み合わせてスーパーディスクリプタとして定義することもできます。ユーザー指定のアルゴリズムに基づいて、*照合ディスクリプタ*や*ハイパーディスクリプタ*を定義したり、発音によるエンコードアルゴリズムに基づいて、*フォネティックディスクリプタ*を定義したりできます。フォネティックディスクリプタは、言語固有の要件に合わせてカスタマイズすることが可能です。 詳細は、「*特殊なフィールドおよびディスクリプタフィールド*」を参照してください。

データ圧縮オプション FI および NU

デフォルトのデータ圧縮については、「*圧縮*」セクションで説明されています。 フィールドレベルでは、追加の圧縮を指定できます（空値省略オプション）。または、すべての圧縮を無効にできます（固定ストレージオプション）。

空値省略（NU）はデフォルトの圧縮とは異なり、空値省略が指定されたディスクリプタフィールドを検索すると、ディスクリプタフィールドが空白のレコードは*返さません*。

固定フォーマット（FI）として定義されたフィールドには、長さバイトが含まれません。また、圧縮は行われません。このオプションを指定すると、1バイトフィールドまたは日常的に頻繁に使用されるフィールド（個人を特定する ID を含むフィールドなど）に必要なストレージスペースを節約できます。

エンコード変換オプション NV

英数字（A）またはワイド文字（W）フォーマットのフィールドに NV オプションを指定すると、そのフィールドはユーザーとの間では変換されず、レコードバッファ内で処理されます。

このフィールドはファイルエンコードの特性を持つため、デフォルトは空白です。

- A フィールドは常に EBCDIC の空白（X'40'）になります。
- W フィールドは常に W フォーマットのファイルエンコードの空白になります。

NV オプションは、変換しても意味のないデータ、またはアプリケーションで保管されているままのデータが必要とされるために変換すべきでないデータを含むフィールドに使用します。

NV フィールドのフィールド長は、ユーザーアーキテクチャでバイトスワップが採用されている場合、バイトスワップされます。

ロング英数字オプション LA

ロング英数字（LA）オプションは、長さがゼロの A または W フォーマットフィールドのように、可変長の英数字またはワイド文字フィールドにのみ指定できます。LA オプションでは、このような英数字またはワイド文字フィールドに最長16,381バイトの値を含めることができます。

LA オプションを含む英字またはワイド文字フィールドは、このオプションを含まない英字またはワイド文字フィールドと同様に圧縮されます。LA オプションを含むフィールドで実際に可能な最大長は、圧縮レコードを格納するブロックのサイズで制限されます。

Adabas 8（以降）では、**NB**（空白圧縮なし）オプションを LA フィールドに指定して空白圧縮を制御できます。

LA フィールドには、同時に LB フィールドオプションを定義することはできません。 LA および LB のどちらとしてフィールドを定義するかを決定する際に、「*LA および LB フィールドの違い*」を参考にしてください。

ラージオブジェクトオプション LB

フィールドにラージオブジェクト（LB）オプションを指定すると、ラージオブジェクトフィールドにすることができます。 LB フィールドには、2,147,483,643 バイト（約 2 GB）までのデータを格納できます。 現時点では、LB フィールド全体を格納し、取り出せるだけで、LB フィールドの一部を格納し、取り出すことはできません。

LB フィールドのフォーマットは "A"（英数字）であり、そのデフォルトフィールド長は現在、0 として定義されている必要があります。

LB フィールドは、次のものにすることは*できません*。

- ディスクリプタ、または特別なディスクリプタ（フォネティック、サブ、スーパー、またはハイパーディスクリプタ）の親
- FI または LA オプションとともに定義する。

  LA および LB のどちらとしてフィールドを定義するかを決定する際に、「*LA および LB フィールドの違い*」を参考にしてください。
- サーチバッファまたはフォーマットバッファのフォーマット選択条件で、指定されたフィールド

LB フィールドは次のいずれかです。

- MU、NB、NC、NN、NU、または NV オプションのいずれかで定義されたフィールド
- 単一グループまたは PE グループの一部にする。

LB フィールド定義に NB（非空白圧縮）フィールドオプションがあるかどうかは、文字を含む LB フィールドで末尾の空白が Adabas により削除されたかどうかを示します。

バイナリと文字の両データを含む LB フィールドがサポートされています。 NV および NB オプションの両方で定義された LB フィールドには、バイナリラージオブジェクトデータを格納できます。これは、Adabas ではバイナリ LB フィールドが変更されることがないためです。 格納されているのと同じ LB バイナリバイト文字列が、LB フィールドが読み込まれたときに取り出されます。 また、バイナリ値を含む LB フィールドは NV および NB オプションで定義されているため、Adabas では、何らかの文字コードページに従って LB フィールドバイナリ値が変換されることも、バイナリ値を含む LB フィールドで末尾の空白が削除されることもありません。

Note: バイナリ値を含む LB フィールドの定義には、フォーマット B は使用されません。これは、一部の環境ではフォーマット B が、バイト順の異なるバイトスワップを意味するためです。 バイトスワッピングは、バイナリ LB フィールドに適用されません。

次の表は、LB フィールドの FDT 定義の有効な例を示しています。

FDT の指定	説明
`1,L1,0,A,LB,NU`	フィールド L1 は、空値省略の文字型のラージオブジェクトフィールドです。
`1,L2,0,A,LB,NV,NB,NU,MU`	フィールド L2 は空値省略、マルチプルバリュー、バイナリフィールド型のラージオブジェクトフィールドです。

LB フィールドを扱うコマンドの対象は、*LOB ファイルグループの基本ファイル*にする必要があります。 *LOB ファイル*に対するユーザーコマンドは拒否されます。

LB フィールドの基本的な情報については、「*ラージオブジェクト（LB）フィールドの基本*」を参照してください。

LA および LB フィールドの違い

関連のある LA フィールドおよび LB フィールド機能を比較している次の表は、データベースのフィールドを定義する際にどちら使用するかを決定するのに役立ちます。

機能	LA フィールドの動作	LB フィールドの動作
フォーマットバッファ内のゼロフィールド長指定	対応するレコードバッファエリア内の 2 バイトが、LA フィールドの実際の長さを格納するために使用されます。	対応するレコードバッファエリア内の 4 バイトが、LB フィールドの実際の長さを格納するために使用されます。
データレコードストレージ	英数字フィールドおよびワイド文字フィールドが、圧縮レコード内に格納されます。 すべての長い値は、同じ圧縮レコード内に収まらなければなりません。シンプルデータレコードまたはスパンドデータレコードの最大長により、格納できる長い値の数および長	一部の LB フィールド値（253 バイト超）は、別のラージオブジェクトファイル（LOB ファイル）にオフラインで格納され、LOB ファイル内の LB フィールド値への参照のみがデータレコードに含まれます。これにより、通常のフィールド、または LA フィールドを使用するより、単一のデータレコードに長いオブジェクトを格納

機能	LA フィールドの動作	LB フィールドの動作
	さが制限されます。 複数の長い値がレコードに含まれると、問題となる場合があります。	できるようになります。 ただし、この動作により、LBフィールドの実行時およびファイルメンテナンスのパフォーマンスのオーバーヘッドが増加します。 小さなLBフィールド値（253バイト以下）は圧縮レコード内に直接格納されます。これにより、小さな値に対するパフォーマンスは向上しますが、同一の圧縮レコードに格納可能な小さな LB フィールドのオカレンス数が制限されます。
フォーマットバッファ内のアスタリスク（*）フィールド長表記	すべての長さの LA フィールドがサポートされます。	すべての長さの LB フィールドがサポートされます。
最大長が 16,381 バイト*以下*の格納オブジェクト	英数字またはワイド文字のLAフィールドを使用できます。 これにより、LBフィールドのオーバーヘッドを避けることができますが、単一レコードに格納できるフィールドの数が制限されます。	英数字の LB フィールドを使用できます。
最大長が 16,381 バイトを超える格納オブジェクト	サポートされません。	16,381 バイトを超えるオブジェクトがサポートされます。
シンプルデータレコードまたはスパンドデータレコードに収まらない非常に多くのラージオブジェクト	サポートされません。	複数のラージオブジェクトがサポートされます。

MU および PE オプションおよびフィールドタイプ

Adabas では、エレメンタリフィールドとマルチプルバリューフィールドの 2 つの基本フィールドタイプがサポートされています。 *エレメンタリ*フィールドの値は、レコードごとに 1 つだけです。 *マルチプルバリュー*（MU）フィールドは、1 レコードあたり 191 個（オカレンス）にするか、または最大約 65,534 個（オカレンス）にすることができます。 ファイル中で 191 個を超える MU フィールドまたは PE グループを使用することは、そのファイルが明示的に許可されている必要があります（デフォルトでは許可されません）。 この場合には、ADADBS MUPEX 機能または ADACMP COMPRESS `MUPEX` および `MUPECOUNT` パラメータを使用します。 それぞれのマルチプルバリューフィールドには、オカレンスの数を格納する*バイナリオカレンスカウンタ*（BOC）があります。

ピリオディック（PE）グループフィールドは、FDT 内の連続フィールドを定義します。これらのフィールドはレコード内でまとめて繰り返されます。 非ピリオディックグループフィールドのメンバ同様に PE メンバは、PE グループフィールドの直後にあり、PE フィールドより高いレ

ベル番号を持ち、個別にも、グループとしてもアクセス可能です。各PEには、オカレンスの数を格納する BOC があります。

ピリオディックグループの繰り返し回数は、1 レコードあたり 191 にするか、または最大で 65,534 程度にすることができます。また、ピリオディックグループは 1 つ以上のマルチプルバリューフィールドを含むことができます。 ファイル中で 191 個を超える MU フィールドまたは PE グループを使用することは、そのファイルが明示的に許可されている必要があります（デフォルトでは許可されません）。 この場合には、ADADBS MUPEX 機能または ADACMP COMPRESS MUPEX および MUPECOUNT パラメータを使用します。 使用されないオカレンスまたは値には、ストレージスペースは不要です。

したがって、Adabas では、次の 4 種類のフィールドがサポートされています。

	レコードごとに単一の値	レコードごとに複数の値
単一のフィールド	エレメンタリ	MU
マルチプルフィールド	グループ	PE

ファイル内の MU フィールドおよび PE グループのオカレンス数の上限は、実際にはデータストレージレコードの最大長（ADALOD MAXRECL パラメータ）によって決まり、デフォルトではデータストレージブロックのサイズから 4 を引いた数です。

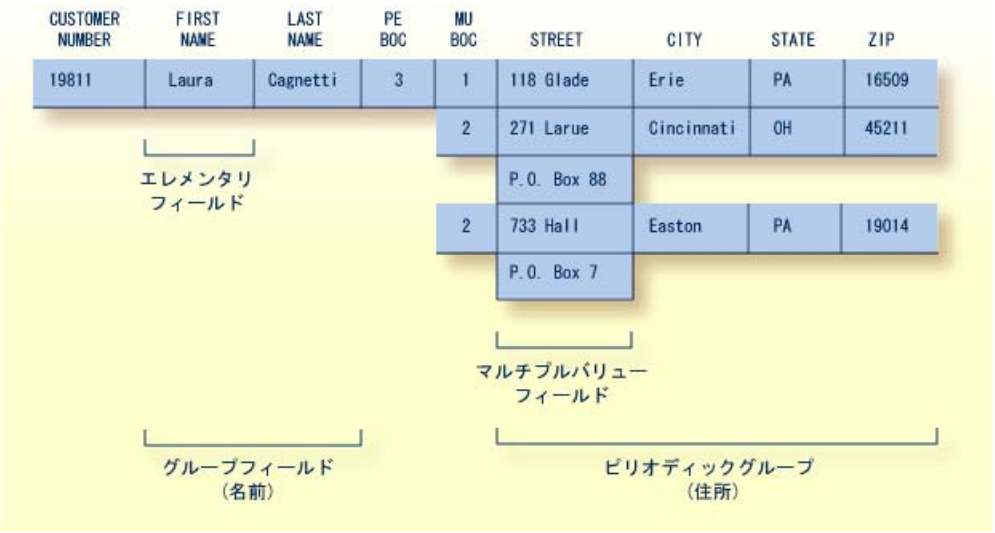
レコード中の各 MU フィールドまたは各 PE グループのオカレンス数は 191 個にするか、または最大約 65,534 個にすることができます。その場合は、ADADBS MUPEX 機能または、ADACMP COMPRESS MUPEX および MUPECOUNT パラメータを使用します。 ただし、実際のオカレンス数は、データストレージレコードの最大長（ADALOD MAXRECL パラメータ）によって制限されます。デフォルトではデータストレージブロックのサイズから 4 を引いた数です。また、デバイスのタイプ、およびファイルのタイプ（スパンされているかどうか）によっても制限されます。すべての MU フィールドおよび PE グループ、およびその他のフィールドは、1 つの圧縮レコードに収まらなければなりません。 スパンドレコード（Adabas 8 で導入）を使用すると、より多くの MU フィールドおよび PE グループを格納できます。

また、サブディスクリプタおよびスーパーディスクリプタを定義すると、レコードに含まれる MU フィールドまたは PE グループの数に影響する可能性があります。 例えば、スーパーディスクリプタが PE グループと 1 つまたは複数の MU フィールドの組み合わせとして作成されていて、オカレンス数が多い場合、パフォーマンスおよびリソースの問題が発生する可能性があります。

Note: 拡張 MU および拡張 PE フィールドの過度の使用は、パフォーマンスやリソースの問題を引き起こす場合があります。 これらの問題は、ワークストレージのオーバーフローを引き起こす可能性があり、その結果レスポンスコード 9 が返されます。 この問題が発生した場合は、データベースの ADARUN LP のサイズを増加させます。

すべての MU フィールドおよび PE グループ、およびその他のフィールドは、1 つの圧縮レコードに収まらなければなりません。 スパンドレコード（Adabas 8 で導入）を使用すると、より多くの MU フィールドおよび PE グループを格納できます。

次の図は、1 つのレコード構造内の 4 つのフィールドタイプを示しています。



PE フィールドは、別の PE グループ内でネストすることはできません。 上の図で示すように、PE グループ内での MU フィールドのネストは許可されますが、2 次元配列を導入する必要があり、プログラムが複雑になります。 また、データのアクセスと密接な関係があります。Adabas がピリオディックグループにアクセスすると、返される PE の各オカレンスに対して、MU フィールドの最初のオカレンスのみが返されます。

ピリオディックグループにはオカレンスの順番が変わらないという独特の特徴があり、この構造上の特性を活用する場合に、ピリオディックグループを使用します。 最初に 3 つのオカレンスが含まれているピリオディックグループは、第 1 または第 2 オカレンスが削除されると、それらのオカレンスが空値にセットされます。第 3 のオカレンスは、そのまま 3 番目の位置に残されます。 これとは対照的に、マルチプルバリューフィールドでは、先行する空値エントリが異なる方法で処理されます。 値の 1 つが削除された場合、マルチプルバリューフィールドの値の格納場所は、削除前と変わります。

拡張 MU または PE が制限されたファイルが作成された場合、MU フィールドまたは PE グループのオカレンスカウントをレコードバッファの 1 バイトフィールドに読み込んではいけません。これを行うと、Adabas はレスポンスコード 55、サブコード 9 を返します。 したがってフォーマットバッファで xxC エレメントを使用してオカレンスカウントを読み込む（FB='MUC.' または FB='MUC,1,B.' など）すべてのアプリケーションは、オカレンスカウントを 2 バイト以上のフィールドに読み込ませるように（FB='MUC,2,B.' または FB='MUC,4,B.' など）変更する必要があります。

空白圧縮オプション NB

NB オプションを **LA** および **LB** フィールドとともに使用すると、空白圧縮を制御できます。 NB オプションを指定すると、フィールドの末尾の空白が削除されなくなります。指定しない場合は、英数字またはワイド文字フィールドの値の格納時に末尾の空白が削除されます。 フィールドに NB オプションを指定した場合、そのフィールドに NU オプションまたは NC オプションも指定する必要があります。NB 処理には、NC または NU の使用も同時に必要となります。

Note: NB オプションを使用せずにフィールドを指定すると、格納したときと取り出したときのフィールドの長さが異なる可能性があります。 取り出したときの非 NB フィールドの長さは、空白圧縮のために、格納したときにそのフィールドに指定した長さよりも短くなる可能性が高くなります。これは、値が実際は文字列ではなく、空白を表す文字コードで終わるバイナリ値であった場合に問題となることがあります。 したがって、格納したときと取り出したときのフィールドの長さを同じにしたい場合は、NB オプションを使用します。

SQL 互換性オプション NC および NN

Adabas には特別なデータ定義オプションが含まれ、Software AG のメインフレーム Adabas SQL Gateway（ACE）などの SQL データベース問い合わせ言語でも、SQL 互換の空値を表現することができます。

NC（カウントなし）オプションを指定したフィールドには、SQL がフィールドに値がないと解釈できる空値を含めることができます。NC フィールドに空値が含まれるということは、*値*が入力されていないことを意味します。つまり、フィールド値は未定義になります。

この未定義の状態は、値が指定されていない非 NC フィールドに割り当てられた空値とは異なります。非 NC フィールドにおける空値は、フィールドの値が、そのフィールドのフォーマットに応じてゼロまたは空白であることを意味します。

NN（非空値）オプションは、NC が定義されたフィールドにのみ指定できます。 このオプションは、NC フィールドが常に定義済みの値を持つ必要があることを示します。SQL 空値を持つことはできません。 これにより、レコードの作成時または更新時に、フィールドが未定義のままになることを防ぎます。 ただし、フィールド値をゼロまたは空白にすることは可能です。

特殊なフィールドおよびディスクリプタフィールド

FDT は、フィールドが照合ディスクリプタ、サブフィールド、スーパーフィールド、サブディスクリプタ、スーパーディスクリプタ、ハイパーディスクリプタ、またはフォネティックディスクリプタの親フィールドであるかどうかを示します。 ファイル中のすべての特殊なフィールドおよびディスクリプタ（照合ディスクリプタ、サブディスクリプタ、サブフィールド、スーパーディスクリプタ、スーパーフィールド、フォネティックディスクリプタ、およびハイパーディスクリプタ）は、FDT の一部である特殊ディスクリプタテーブル（SDT）内で保持されます。

```
SPECIAL DESCRIPTOR TABLE

        I      I        I        I                       I                I
  TYPE  I NAME I LENGTH I FORMAT I       OPTIONS         I   STRUCTURE    I
        I      I        I        I                       I                I
 -------I------I--------I--------I-----------------------I----------------I
        I      I        I        I                       I                I
  SUPER I  H1  I    4   I    B   I DE,NU                 I AU (  1 -  2)  I
        I      I        I        I                       I AV (  1 -  2)  I
  SUB   I  S1  I    4   I    A   I DE                    I AO (  1 -  4)  I
  SUPER I  S2  I   26   I    A   I DE                    I AO (  1 -  6)  I
        I      I        I        I                       I AE (  1 - 20)  I
  SUPER I  S3  I   12   I    A   I DE,NU,PE              I AR (  1 -  3)  I
        I      I        I        I                       I AS (  1 -  9)  I
        I      I        I        I                       I                I
  PHON  I  PH  I        I        I                       I PH =PHON(AE)   I
        I      I        I        I                       I                I
  COL   I  Y1  I   20   I    W   I DE                    I CDX  8,PA      I
  COL   I  Y2  I   12   I    A   I DE,NU,PE              I CDX  1,AR      I
        I      I        I        I                       I                I
        I      I        I        I                       I                I
 ------------------------------------------------------------------------
```

このテーブルに含まれる情報としては、特殊フィールドおよびディスクリプタの名前、長さ、フォーマット、および指定オプションの他に、次のものがあります。

列	説明	
TYPE	COL	照合ディスクリプタ
	HYPER	ハイパーディスクリプタ
	PHON	フォネティックディスクリプタ
	SUB	サブフィールド／サブディスクリプタ
	SUPER	スーパーフィールド／スーパーディスクリプタ
STRUCTURE	サブ、スーパー、またはハイパーディスクリプタの構成フィールドおよびフィールドバイト。フォネティックディスクリプタは同等の英数字エレメンタリフィールドを示します。照合ディスクリプタは関連付けられた照合ディスクリプタユーザー出口と親フィールドの名前を示します。	

このsectionは、特殊なフィールドおよびディスクリプタについて説明します。

- 照合ディスクリプタ
- ハイパーディスクリプタ
- フォネティックディスクリプタ
- サブフィールド／スーパーフィールド
- サブディスクリプタ

- スーパーディスクリプタ

照合ディスクリプタ

英数字またはワイド文字フィールドは、照合ディスクリプタの親フィールドとして定義することができます。 照合ディスクリプタは、フィールド値をユーザー定義の特別な順序でソートするために使用します。 LF コマンドは、照合ディスクリプタフィールドの情報をレポートします。

照合ディスクリプタには、照合ディスクリプタユーザー出口（1～8）が割り当てられます。このユーザー出口は、照合ディスクリプタ値のエンコードおよび元のフィールド値へのデコードに使用されます。 ADARUN のパラメータ CDXnn は、照合ディスクリプタユーザー出口の指定に使用します。

ハイパーディスクリプタ

ハイパーディスクリプタオプションを使用すると、ユーザー指定のアルゴリズムに基づいてディスクリプタ値を生成できます。 単一の物理 Adabas データベースごとに 31 個までの異なるハイパーディスクリプタを定義できます。 使用するジョブの ADARUN ステートメントパラメータで適切な HEXnn を指定して各ハイパーディスクリプタに命名しなければなりません。

ハイパーディスクリプタを使用すると、あいまい一致検索が可能になります。例えば、検索条件と*完全に*一致するのではなく、*類似している*データを取り出すことができます。 ハイパーディスクリプタにより、マルチプルバリューインデックスが使用可能となります。つまり、1 つのデータフィールドに異なる複数の検索インデックスエントリを作成できるようになります。

ハイパーディスクリプタは n 個から構成されるスーパーディスクリプタ、派生キー、またはその他のキー構造を実装するために使用できます。 ハイパーディスクリプタを使用すると、厳密に正規化されたリレーショナル構造を基にしたアプリケーションよりも、簡単で柔軟なアプリケーションの開発が可能です。

ハイパーディスクリプタの応用分野として、名前の処理があります。 例えば、SCHROEDER という名前に対しては、それ自身のインデックス SCHROEDER だけが保存されますが、SCHRODER、SCHRADER などのさまざまな名前も、実質的にはインデックスとして保存されたことと同じになります。 したがって、物理的にはSCHROEDERのみがデータベースのデータエリアに保存されますが、このデータに対して複数の検索インデックスが存在します。 後で、SCHRODER という名前で検索が行われた場合、SCHROEDER のレコードが検索されます。

ハイパーディスクリプタのより高度な応用分野として指紋検索があります。この検索では、指紋の一般的な特徴によりあいまい一致アルゴリズムの基本を形成できます。つまり、元の指紋がデータベースに保存されますが、元の指紋とわずかに違う指紋も一致すると認識するアルゴリズムに基づいて、任意の数の検索インデックスをこの指紋に対して作成することができます。

フォネティックディスクリプタ

フォネティックディスクリプタを定義して、類似したフォネティック値を持つレコードの検索に使用できます。ディスクリプタのフォネティック値はフィールド値の先頭20バイトに基づいて内部アルゴリズムにより決定され、英字の値のみが考慮されます（数値、特殊文字および空白は無視されます）。

サブフィールド／スーパーフィールド

フィールドの一部（サブフィールド）またはフィールドの組み合わせ（スーパーフィールド）をエレメンタリフィールドとして定義することができます（「*MU および PE オプションおよびフィールドタイプ*」を参照）。サブフィールドとスーパーフィールドは、読み込みにのみ使用できます。これらのフィールドが変更されるのは、元のフィールドが更新された場合のみです。

サブディスクリプタ

サブディスクリプタは、単一のフィールドの一部であり、ディスクリプタとして使用されます。サブディスクリプタの元になるフィールドは、エレメンタリディスクリプタであってもなくてもかまいません（「*ディスクリプタオプション DE、UQ、および XI*」を参照）。検索条件が、英数字フィールドの先頭 n バイトまたは数値フィールドの最後の n バイトを対象とするとき、そのフィールドの関連バイトだけからサブディスクリプタを定義できます。サブディスクリプタは、ある範囲の複数の値ではなく、単一の値を指定するため、検索の効率を向上させることが可能です。

例えば、5バイトフィールドの最初の2バイトが地理的な地域を表していて、サブディスクリプタを使用せずに地域11のすべてのレコードを取得する場合、すべてのレコードに対して11000～11999の範囲で検索を行う必要があります。このフィールドの最初の2バイトからなるサブディスクリプタを定義すれば、サブディスクリプタが11であるすべてのレコードを検索できます。

スーパーディスクリプタ

スーパーディスクリプタは2～20個のフィールドのすべてまたはその一部を組み合わせます。スーパーディスクリプタの元になるフィールドは、エレメンタリディスクリプタであってもなくてもかまいません。検索条件にフィールドの組み合わせを必要とする場合、スーパーディスクリプタを使用する方が、複数のエレメンタリディスクリプタを組み合わせるよりも効率的です。

例えば、ある地域内の顧客の姓で検索を行う場合、5バイトの顧客番号フィールドの最初の2バイト（地理的な地域のインジケータ）と顧客の姓フィールド全体を組み合わせてスーパーディスクリプタを作成できます。

スーパーディスクリプタの定義については、ADACMPドキュメントの「*SUPDE：スーパーディスクリプタの定義*」を参照してください。

スパンドレコード

Adabas 8 では、レコードをデータベース内でスパンすることができます。 データベース内では、論理レコードは多くの物理レコードに分割されます。それぞれのレコードは単一のデータストレージ（DS）ブロックに格納されます。 分割された各物理レコードには、それぞれ ISN が割り当てられます。 最初の物理レコードは*プライマリレコード*と呼ばれ、圧縮レコードの先頭が含まれ、*プライマリ ISN* が割り当てられます。 残りの物理レコードは、*セカンダリレコード*と呼ばれ、論理レコードの残りのデータを含みます。 セカンダリレコードには、*セカンダリ ISN* が割り当てられます。 これらの ISN は、N2 コマンドの使用時に割り当てられるユーザー ISN、または L1 コマンドの I オプション使用時に使用される ISN に影響しません。 スパンドレコードでは、セカンダリアドレスコンバータを使用して、セカンダリレコードが格納されているデータストレージブロックの RABN にセカンダリ ISN をマッピングします。

スパンドレコードは、1 つのプライマリレコードと 1 つ以上のセカンダリレコードで構成されます。 ただし、スパンドレコードのセグメント数は制限されています。 Adabas ニュークリアスは、スパンドレコード内に最大 5 個の物理レコード（1 個のプライマリレコードと 4 個のセカンダリレコード）を許可しています。

スパンドレコードはアプリケーションプログラムで直接表示させることはできません。 アプリケーションは、常にプライマリ ISN を経由してスパンドレコードにアドレスします。

スパンドレコードは、拡張 **Adabas** ファイルおよびマルチクライアントファイルでもサポートされています。

Note: スパンドレコードのサポートは、ファイルに明示的に許可する必要があります。 ADADBS RECORDSPANNING 機能、または ADACMP COMPRESS の SPAN パラメータを使用してこれを行うことができます。 詳細は、ADADBS および ADACMP ユーティリティについての *Adabas ユーティリティ* のドキュメントを参照してください。

このsectionでは、次のトピックについて説明します。

- スパンドレコードの構造
- ファイル上でのスパンドレコードの許可
- セカンダリレコードのセグメンテーション
- パディングファクタ
- スパンドレコードの ISN の使用
- 関係する ADARUN パラメータ
- スパンドレコードに関するレポート

- スパンドレコードのセキュリティ保護

スパンドレコードの構造

スパンされた論理レコードは1つ以上の物理レコードで構成されます。この論理レコードには、1つのプライマリレコードおよび1つ以上のセカンダリレコードが含まれます。スパンドレコードを構成するレコード数は制限されています。Adabasニュークリアスは、スパンドレコード内に最大5個の物理レコード（1個のプライマリレコードと4個のセカンダリレコード）を許可しています。

スパンドレコードのプライマリレコードおよびセカンダリレコードはISNを使用して結合されます。各物理レコードのヘッダーには、現在のレコードのISN、プライマリレコードのISN、また次のセカンダリレコードのISNが含まれています。また、ヘッダーには、現在のレコードがプライマリレコードかセカンダリレコードかを示す情報も含まれています。

各物理レコードのヘッダーは、そのレコードの長さ情報も提供します。これはレコードがセグメント化されていても同様です。この場合、レコード長はセグメントの長さになります。

ファイル上でのスパンドレコードの許可

ファイル上のスパンドレコードは、ADACMP COMPRESSの SPAN パラメータ、ADADBSのRECORDSPANNING機能または同等なAdabas Online System機能を使用して明示的に要求した場合のみ許可されます。ADAREPデータベースレポート機能およびAdabas Online Systemレポート機能を使用すると、スパンドレコードを使用できるようにファイルが定義されているかどうかを確認できます。

ファイルのSPAN属性は、ADAULD UNLOAD機能内で保持されます。つまり、ファイルがアンロードされたり、削除されたり、再ロードされた場合に、スパンドレコードのサポートは変わらずに維持されます。

191を超えるMUまたはPEオカレンスが許可されているファイルにも同じ規則が適用されます。圧縮レコード内の191を超えるMUおよびPEオカレンスの識別の詳細については、「*圧縮レコード内の191を超えるMUおよびPEオカレンスの特定*」を参照してください。

セカンダリレコードのセグメンテーション

セカンダリレコードは、フィールド単位またはバイト単位のいずれかでセグメント化されます。パフォーマンス上の理由から、セグメンテーションは可能な限りフィールド単位に行われます。ただし、LB（ラージオブジェクト）タイプではないフィールドのサイズがデータストレージブロックサイズよりも大きい場合、このレコードはバイト単位で分割されます。フィールドのサイズがデータストレージブロックの残りの空きスペースより大きく、データストレージブロックサイズより小さい場合、このレコードはバイト単位ではなくフィールド単位で分割されます。各セカンダリレコードのヘッダーには、セグメントレコードがどのタイプであるかを示す情報が格納されています。

パディングファクタ

パディングファクタは、ブロックの領域をすべて使用するために、一般的にはスパンドレコードでは無視されます。 したがって、大抵の場合はレポート上でゼロが表示されます。 パディングファクタは、スパンドレコードの最後の短いセグメントのみに使用されます。

スパンドレコードの ISN の使用

プライマリレコードおよびセカンダリレコードは、アドレスコンバータ（AC）を使用して Adabas によりアドレスされます。ただし、プライマリアドレスコンバータはプライマリレコードの ISN のみを、対応するデータストレージブロックの RABN にマッピングします。 スパンドレコードでは、セカンダリアドレスコンバータを使用して、セカンダリレコードが格納されているデータストレージブロックの RABN にセカンダリ ISN をマッピングします。 したがって、各スパンドレコードのインデックスは 1 つだけになり、インデックス構造に何も影響を与えません。

ISN の範囲は、プライマリ ISN およびセカンダリ ISN 用に個別に維持されます。 ISN が格納または取り扱われる場合は、必ずそのアクションの対象がプライマリ ISN なのか、セカンダリ ISN なのかが区別されます。

すべてのコマンドは、プライマリレコードの ISN を使用して指定する必要があります。セカンダリレコードの ISN は内部的なもので、ユーザーが使用することはできません。 物理シーケンシャルコマンドは、データストレージ内のセカンダリレコードを自動的にスキップします。 セカンダリ ISN を指定した読み取りコマンドにはエラーが返されます（レスポンスコード 113）。

プライマリレコードの ISN は、TOPISN および MAXISN 値に含まれます。 セカンダリレコードの ISN は含まれません。 代わりに、セカンダリ ISN は、MINSEC および MAXSEC 値に含まれます。 スパンドレコードを含むファイルは MINISN 値を指定してロードできますが、MINISN はプライマリレコードの ISN のみを参照する必要があります（セカンダリレコードの ISN は参照できません）。

関係する ADARUN パラメータ

スパンドレコードを含むファイルをサポートするために、次の ADARUN パラメータ値を大きくする必要がある場合があります。

- 更新するスパンドレコードの数も増加するため、ユーザーごとのホールドキュー内の ISN 数（NISNHQ パラメータ）を大きくする必要がある場合があります。
- スパンドレコードのビフォーイメージとアフターイメージの両方を格納するスペースが必要なため、また、複数の更新スレッドの並行実行をサポートするため、Adabas ワークプール長（LWP）を大きくする必要がある場合もあります。 大きなディスクリプタバリューテーブルを格納するスペースが必要な場合もあります（PE グループのディスクリプタのオカレンス数は、最大 65,534 です）。

スパンドレコードに関するレポート

最大レコード長統計は、スパンされたファイルとは関連がありません。最大レコード長をレポートするユーティリティは、統計結果として"［N/A］"（該当なし）を表示するようになりました。 FCB の最大レコード長フィールドは、スパンドレコードを使用しているファイルの場合には、値が大きくなります。

スパンドレコードのセキュリティ保護

スパンドレコードを含んでいるファイルは、暗号化することができ、セキュリティバイバリューにより保護することができます。 プライマリレコードの ISN が参照されている場合、すべてのセカンダリのセグメントレコードも読み込まれる必要があります。したがって、処理時間が重要な要素となります。

4 Adabas の使用

一般的に、Adabas は他のデータベース管理システムが使用するデータ処理リソース（ディスクストレージ、CPU タイム、処理経過時間）の 10 %～50 %を使用します。 使用するハードウェア機能が少ないため、より少ないリソースを使用してより多くのデータを処理できます。 端末ワークステーションの台数が数千台に及び、テラバイト級のデータを処理する大規模なオンラインアプリケーションを実装しても、レスポンスタイムとコストは小規模システム並みに抑えることができます。

このchapterでは、次のトピックについて説明します。

プログラムからのデータベースへのアクセス

Adabas のアクセスはフィールドを基本にしています。ユーザープログラムは必要なフィールドのみにアクセスしたり、必要なフィールドのみを取得します。 ユーザープログラムのステートメントにより、自動的に Adabas 検索や取得処理を呼び出すことができます。

ダイレクトコールインターフェイス

Adabas には、データベース操作を行う強力で柔軟な一連のダイレクトコールコマンドが備わっています。 Natural または他の第 4 世代データベース言語を使用しないとき、Adabas ダイレクトコールコマンドが Adabas データベースとの直接インターフェイスを提供します。

コマンドは、次の機能に分類されます。

- データベース問い合わせ
- 読み込み（データストレージまたはアソシエータ）
- データベース更新
- 論理トランザクション処理
- 特殊コマンド

データベース問い合わせコマンド（Sx）

データベース問い合わせコマンド（S1/S4、S2、S5）は、指定された検索条件に応じて、指定されたレコードまたはレコードグループの ISN を検索して結果を返します。 その他のこの分類に属するコマンド（S8、S9）は、後から処理するための準備として、ISN結果リストをソートします。

S*x* コマンドの結果として返された ISN リストは、後でユーザーセッション中に取得できるように Adabas WORK データセットに格納できます。

これらのコマンドはほとんどの場合、実際にはデータベースを読み取りません。ISN はアソシエータのインバーテッドリストから直接読み込まれます。 オプションとして、ISN のレコードをホールド状態にして、レコードが解放されるまで他のプログラムによる更新を防止できます。また、必要に応じて、最初の ISN のレコードに含まれている追加フィールドの値を、データストレージから読み込むことができます。

読み込みコマンド（Lx）

L1～L6 コマンドは、データストレージから実際のレコードを読み込みます。 指定したコマンドとコマンドのオプションに応じて、レコードは個別に次のように読み込まれます。

- 格納された順番
- データベース問い合わせコマンドによって作成された ISN リストの順番
- ユーザー指定ディスクリプタに応じた論理的な順番

順次アクセスメソッドの詳細は、「*順次読み込み*」を参照してください。

ホールドオプションを使うと、専用のコマンドでレコードを解放するか、トランザクションが終了するまで、データベースレコードをロックできます。

L9 および LF コマンドは、アソシエータインバーテッドリストまたはフィールド定義テーブル（FDT）から直接情報を読み込みます。このコマンドが返す値は、指定したディスクリプタのインバーテッドリスト値、または指定したデータベースファイルのフィールド定義のいずれかです。

データベース更新（A1、E1、N1/N2）

データベース更新コマンド（A1、E1、および N1/N2）は、データベースレコードを変更、削除、または追加し、関連するアソシエータリストを適宜更新します。新しいレコードの ISN は、ユーザーまたは Adabas が割り当てます。

インバーテッドリストおよびアドレスコンバータは、Adabasが自動的に更新します。ユーザーからディスクリプタに新しい値が提供された場合、新しいレコードかレコード更新の一部として、Adabasはインバーテッドリストについてすべての必要なメンテナンスを行います。新しいレコードが追加されたか、レコードが削除された場合、アドレスコンバータが適切に更新されます。 これらの動作は、ユーザーには完全に透過的です。

論理トランザクション制御コマンド（ET/BT）

Adabas論理トランザクションは、実行中のデータベースオペレーションの論理的な開始（BT）と終了（ET）を定義します。 ユーザー操作または Adabas 自身により異常終了した場合、ユーザーはこれらのコマンドにより最後に失敗したトランザクションを再開できます。 ET/BT コマンドの処理内容は次のとおりです。

- トランザクションの開始と終了を定義します。
- トランザクションの正常終了を阻止する事態が発生した場合、トランザクション以前の状態に回復します。
- トランザクション中に書き込まれたプログラム固有のユーザーデータを読み取ります。

これらのコマンドを使用するプログラムをETロジックプログラムと呼びます。必須ではありませんが、ETロジックを使用することをお勧めします。 詳細は、「トランザクションロジック」を参照してください。

特殊コマンド

特殊コマンドは、Adabas データベース環境をメンテナンスするのに必要な維持管理の役割を多数行います。 このグループのコマンドは、次のことを行います。

- ユーザーセッションのオープン（OP）およびユーザーセッションのクローズ（CL）（ただしトランザクションは制御しません）
- データプロテクション情報、ユーザーデータおよびチェックポイント（C1、C3、C5）を書き込むコマンド
- レコードをホールド状態に設定するコマンド（HI）と レコードのホールド状態を解除するコマンド（RI）

また、RC コマンドは、ユーザーに対して現在割り当てられている 1 つ以上のコマンド ID を解放するか、1 つまたは全グローバルフォーマット ID を削除します。

RE コマンドは、C3、CL、または ET コマンドにより以前に Adabas システムファイルに格納されたユーザーデータを読みます。

複合検索

ほとんどの大規模データベースシステムでは、複合検索を実行すると処理時間が非常に長くなります。 Adabas は、この問題を解決できるように処理を効率化しました。 現在、使用されている多くの Adabas アプリケーションでは、最大 150 個の複合選択条件が動的に作成されます。 データは 5 千万件を超えるレコードを持つファイルから即時に取得されます。

複数ファイルの検索

問い合わせを解決するために、複数ファイルの検索が必要になることがよくあります。複数ファイルの検索は、複数検索コマンド、物理的にカップリングされたファイル、またはソフトファイルカップリングを使用して実行されます。カップリングファイルの詳細は、「*カップリングファイル*」を参照してください。

あるコマンドで取得された値が次のコマンドの検索値として使われる場合に、複数検索コマンドが使用されます。 この処理は 2 つのファイルに制限されません。

マルチインデックス検索

あいまい一致（*完全*一致ではなく*類似*一致に基づいたデータ取得など）は、ハイパーディスクリプタを使用して実装できます。 ハイパーディスクリプタにより、マルチプルバリューインデックスが使用可能となります。つまり、1つのデータフィールドに異なる複数の検索インデックスエントリを作成できるようになります。 パイパーディスクリプタの詳細は、「*ハイパーディスクリプタ*」を参照してください。

アクセスメソッド

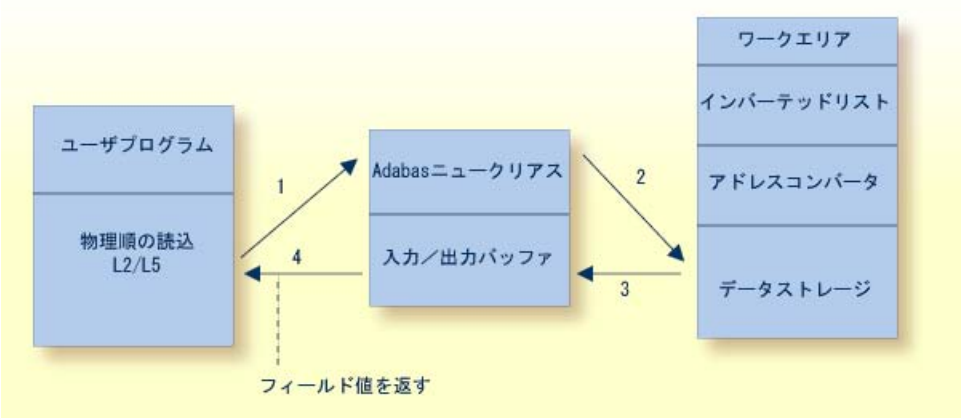
Adabas では、順次アクセスメソッドとランダムアクセスメソッドの両方がサポートされます。それぞれのコールでは、別々の Adabas アクセスパスおよびコンポーネントが使用されます。最も効率的な方法は、必要とする情報の種類および取得が必要なレコード数によって異なります。

- **順次アクセス**
- **ランダムアクセス**
- **Adabas ダイレクトアクセスメソッド（ADAM）を使用したランダムアクセス**

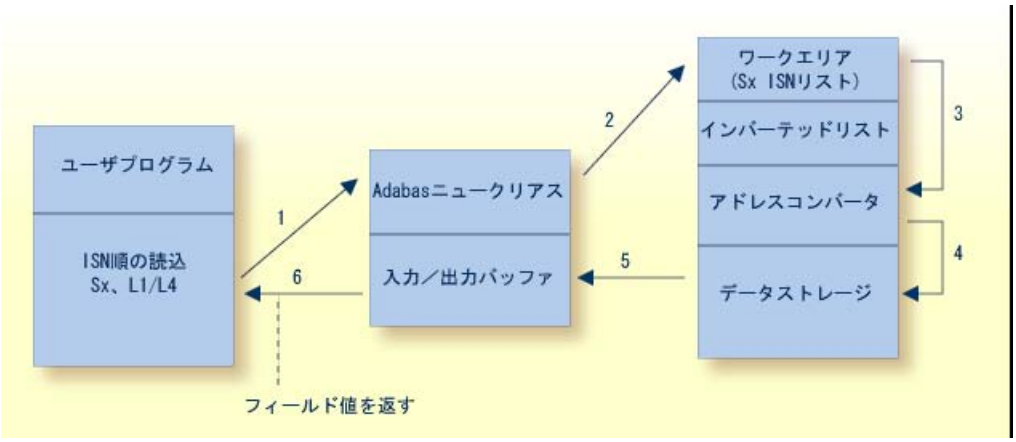
順次アクセス

*物理*順 にレコードを取得する場合、データストレージに格納された順番に従ってレコードが取得されます。 各レコード内のフィールドを、取得したい値を持つフィールドだけに制限できます。 また、開始 ISN を指定すると、指定された ISN により識別されたレコードの後に物理的に位置するレコードから順次読み込みが開始されます。

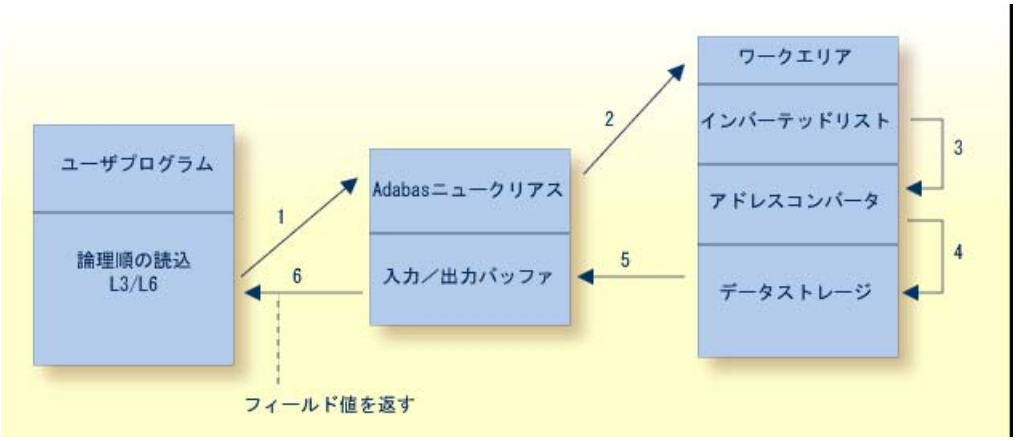
Adabas はアソシエータをバイパスしてデータストレージに直接アクセスし、最初のデータブロック（または指定された ISN 以降の最初のレコード）を読み込みます。最後のブロックが読み込まれるまでレコードが順番に読み込まれます。 物理順は、大量のレコードを処理する最も高速な方法です。



ISN 順にレコードを取得する場合、ISN 順に従ってレコードが取得されます。 Adabas はデータベース問い合わせコマンド（S*x*）を使用して、ISN リストを構築およびソートします。このリストは、GET NEXT オプションを指定して L1/L4 コマンドを使用して読み込むことができます。読み込み時に Adabas は、アドレスコンバータを使用して各 ISN の RABN を検出します。次にデータストレージからレコードを読み込んで返します。

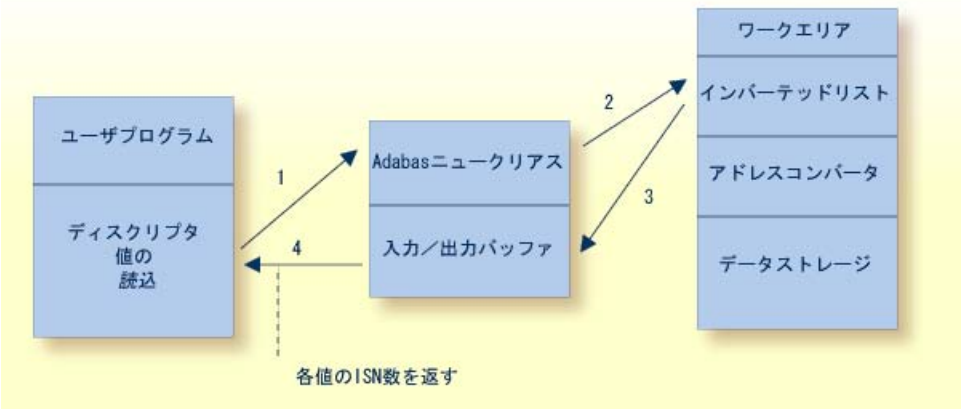


*論理*順にレコードを取得する場合、レコードはディスクリプタ値によって取得されます。Adabas はインバーテッドリスト内で値を検出し、アドレスコンバータを使用して、値に関連した ISN の RABN を検出します。次に、データストレージからレコードを取得します。



論理順の読み込みにより、指定されたディスクリプタの1つの値または値の範囲に関連したすべてのレコードが取得されます。 レコードはディスクリプタ値の昇順または降順にソートされて返され、各ディスクリプタ値内では ISN で昇順または降順にソートされます。 開始値および終了値を指定できます。 また、値が返されるフィールドを指定することができます。 論理的な読み込みは、特定のフィールドでソートされたレコードが必要な場合に便利です。

Adabasには、特別な読み込みコマンド（L9）が準備されており、ディスクリプタとして存在する値の範囲とその各値を含むレコードの数を求めるために使用します。 このようなデータ取得方法をヒストグラムと呼びます。 L9 コマンドは、データレコードへのアクセスを必要としません。アソシエータ内に格納されているインバーテッドリストのみにアクセスします。



ランダムアクセス

Adabas では、S1/S2/S4 コマンドを使用して*検索条件*を満たすレコードを選択します。見つかったレコードの数およびその ISN リストが返されます。 S1/S4 コマンドは、ISN を昇順で返します。S2 コマンドでは、返される ISN のソート順を指定できます。

検索条件を次のものから構成することができます。

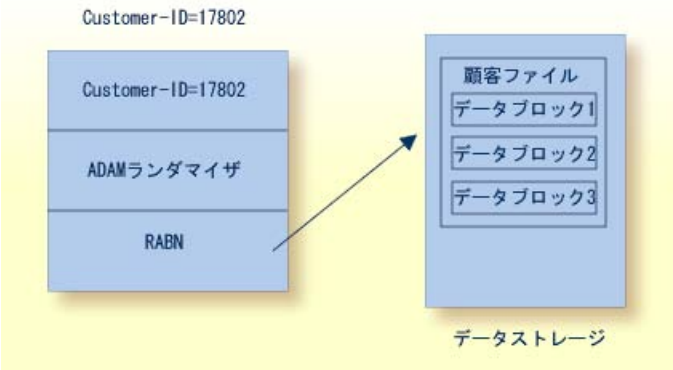
- 単一ファイル内の 1 つ以上のフィールド
- 2 個以上の物理的にカップリングされたファイル内のフィールド
- ソフトカップリング機能に基づく検索、読み込み、内部リストのマッチング

検索条件は、ディスクリプタとして定義されていない1つ以上のフィールドで構成してもかまいません。非ディスクリプタが使用されていると、Adabas はユーザーに返すレコードを判断するために、読み込み処理を実行します。検索条件内にディスクリプタだけが使用されている場合、Adabas はアソシエータのインバーテッドリストを使ってこの条件を処理するため、読み込み処理は実行しません。

Adabas ダイレクトアクセスメソッド（ADAM）を使用したランダムアクセス

Note: ADAM 機能は、メインフレームプラットフォームのみで利用できます。

Adabas ダイレクトアクセスメソッド（ADAM）は、ファイル上の特定のディスクリプタフィールドにおけるランダムアクセスのパフォーマンスを改善します。 ADAM はフィールド値を使用して、レコードストレージの相対ブロックアドレス（RABN）を計算します。 また、ADAM ディスクリプタは、選択条件内の他のディスクリプタと同様に使用されます。さらに、論理順次処理にも使用されます。 どのようなファイルに対しても、データベースへのロード時にこのオプションを選択できます。



トリガとストアドプロシージャの使用

Adabas のトリガおよびストアドプロシージャ機能は、Adabas に必要不可な部分ですが、この機能を使用するには、Natural（「*Natural アプリケーション開発環境*」を参照）が必要です。オンライントリガメンテナンス機能には、**Adabas Online System** アドオン製品を使用してアクセスできます。

プロシージャは、Natural の標準機能を使用して記述され、テストされた Natural サブプログラムです。

- *トリガ*は、特定の条件に合致した場合に、Adabasが自動的に実行するプロシージャです。一連の条件は、Adabas に送信されるコマンドごとに特定されます。条件の基準はターゲットファイル番号と、任意指定のコマンドタイプまたはフィールド（または両方）です。 コマンドタイプとは、検索、読み込み、格納、更新、および削除の各コマンドを指します。 フィールドは、コマンドの対応するフォーマットバッファ内にある必要があります。
- *ストアドプロシージャ*はAdabasによって実行されますが、ストアドプロシージャを使用する多くのアプリケーションからの専用のユーザーコールによって直接、呼び出されます。 複数のクライアントで使用されるプログラムをサーバー上のAdabasファイルに格納すると、サーバーとの間で送受信するデータのトラフィック量を削減できます。

トリガであるか、ストアドプロシージャであるかにかかわらず、同じタイプのパラメータがサブプログラムに渡されます。

Adabasのトリガとストアドプロシージャ機能により、両方のタイプのプロシージャの実装および維持が可能です。ストアドプロシージャはAdabasの機能として装備されており、アプリケーションに拡張性を提供します。 用途は次のとおりです。

- アプリケーションに対するさまざまなセキュリティおよび監査機能の実装
- データの検証と操作が可能な、一貫した一元的な環境の実現。検証と操作は、アプリケーションから手動で行うか、またはAdabasが自動的に実行します（トリガが定義されている場合）。

ユニバーサルエンコーディングサポート（UES）

Adabasには、異なるアーキテクチャ間で双方向にデータを変換する機能が装備されています。これにより、Web ベースアプリケーションから、または Windows 版 Natural などの PC ベースアプリケーションからの、TCP/IP プロトコルを介したメインフレームデータベースへのアクセスのサポートが可能になります。

データベースは、ADADEFによってアクティブ化されている場合、これらのUES機能を使用できます。

Adabas は、英数字（A）およびワイド文字（W）フォーマットのフィールドに対して文字データ変換を実行します。 Unicode、2 バイト文字セット（DBCS）、マルチバイト文字セット（MBCS）など、幅広い文字セットまたはコードページがサポートされます。

- データベースおよびファイルレベルの両方にエンコードキーを定義することにより英数字フィールドが拡張され、ワイド文字データがサポートされます。ファイルレベルのエンコードは、データベースエンコードよりも優先します。 エンコードは、データを保存するフォーマットを指定します。 これは、ローカルユーザーと交換するデータのデフォルトフォーマットとしても使用されます。ファイルエンコードは、EBCDIC グループに属している必要があります。
- ワイド文字フィールドは、英数字フィールドと同様に、エンコードのデフォルトはデータベースおよびファイルレベルの両方に定義され、ファイルエンコードは、データベースエンコードよりも優先します。 デフォルトエンコードは Unicode です。

さらに、Adabas は下位バイトファーストまたは IEEE 規格の浮動小数値などの数字データの変換をサポートします。

クライアントアプリケーションでは、ADALINK コードがデフォルトのアーキテクチャ（ASCII、バイトスワップ、および IEEE 規格浮動小数点など）を判別します。 クライアントアプリケーションは、セッションのオープン（OP コマンド）を使用して異なるエンコードまたはアーキテクチャを指定して、このデフォルト設定を無効にします。

多くのユーティリティが特殊なエンコードおよびアーキテクチャ設定に対応しています。

アーキテクチャおよびエンコード間の上位と下位方向の互換性を確保するために、ユーザーエンコードに対して同じまたはスーパーセットになる文字セットを持つファイルエンコードが選択される必要があります。 例えば、US-EBCDIC と ISO-8850-1 は同じ文字セットを持ちます。

照合ディスクリプタは、英数字またはワイド文字フォーマットのフィールドに対して定義できます。 文化的に正しい並び順（つまり辞書に掲載される順番）に従ってキーを生成するなどの目的で、照合ディスクリプタ出口がコールされると、ディスクリプタ値はアルゴリズムに従って取得されます。

データベースの完全性の維持

更新が競合する環境で、ユーザーまたは Adabas のセッションが中断されたときに、データの整合性を維持するための機能が Adabas には備わっています。

この機能はオンラインまたは従来のバッチ更新の両方で利用できます。 オンラインのトランザクションを基本とした処理では、データベースのトランザクションは完全性が保たれます。バッチモードの更新では、チェックポイントの書き込みおよび更新のバックアウト／再生成が行われるので、失敗した場合でも再スタートできるようになっています。

- トランザクションロジック
- 分散トランザクション処理
- 競合更新
- タイムアウトの制御
- バックアウト、リカバリ、および再スタート

- 拡張エラー処理およびメッセージバッファリング

トランザクションロジック

Adabas のデータ保護、リカバリ、およびユーザー再スタートは、*論理トランザクション*の概念に基づいています。論理トランザクションとは、データベース内の情報に論理的な不整合が生じないように、ユーザーの定義に従って、全体的に実行する必要がある処理の最小単位のことです。

論理トランザクションは 1 つ以上の Adabas コマンドで構成されます。これらのコマンドが実行されることによって、業務の論理ユニットを完了するために必要なデータベースの読み込みおよび更新が行われます。 論理トランザクションは、レコードをホールド状態にする最初のコマンドで始まり、ET（トランザクションの終了）、BT（トランザクションのバックアウト）、CL（クローズ）、または OP（オープン）コマンドが同じユーザーに発行されたときに終了します。

OP（オープン）または RE（ET データの読み込み）コマンドを使用して、C3、CL、または ET コマンドにより格納されたユーザー再スタートデータを取得できます。 このデータは、トランザクションから出力された各チェックポイントとともに、Adabas データプロテクションログにも書き込まれ、ADASEL ユーティリティを使用して読み込むことができます。

ET コマンドは、論理トランザクションの終了時点で発行しなければなりません。 ET コマンドの実行が正常に完了すると、後続のユーザーまたは Adabas セッションの中断に関係なく、トランザクション中に行われた更新処理のすべては物理的にデータベースに適用されます。

ET コマンドを正常に実行できない場合、そのトランザクションの実行中に発生した更新は、BT コマンドを手動で発行するか、または自動バックアウトルーチンの自動処理のいずれかによってバックアウトされます。

分散トランザクション処理

Adabas には、分散環境（並行して存在する複数のローカルまたはリモートのデータベースまたはシステムイメージ、またはその両方）において、グローバルデータベーストランザクションの実行をサポートするニュークリアス機能が組み込まれています。

2 フェーズコミットプロトコルでは、リソースマネージャ（RM）と呼ばれるグローバルトランザクションの処理に関与しているすべてのデータベース管理システム（DBMS）は、それぞれの処理対象のトランザクションを全体としてコミットまたはロールバックします。 第 1 フェーズでは、調整コンポーネント（トランザクションマネージャ、つまり TM）が、関連する RM がコミットできるように準備します。 第 1 フェーズが成功した場合のみ、TM は RM にコミットするように指示します（第 2 フェーズ）

このシナリオでは、Adabas は RM として機能します。 Adabas のアドオン製品 **Adabas Transaction Manager** がオペレーティングシステムイメージ内の調整役として機能します。また Entire Net-Work を利用して複数のシステムイメージにわたって調整することもできます。

また、この新しいプロトコルは、Adabas とその他の DBMS を統合します。 このプロトコルは既存のアプリケーションシステムおよび Natural に対して透過的です。

Adabas には、CICS により制御されるインターフェイスが含まれています。このインターフェイスは CICS リソースマネージャインターフェイス（RMI）に準拠しています。 このインターフェイスは適切な Adabas コマンドを発行して、2 フェーズコミットプロトコルと連携します。

競合更新

競合更新は、2 人以上のユーザー（マルチユーザーモード）が同一 Adabas ファイル（複数の場合もあり）を更新するときに発生します。 競合更新環境におけるデータ保全性を保証するために使用する Adabas 機能には、レコードのホールド／解放、リソースのデッドロック回避、および排他的制御更新が含まれます。

レコードのホールドと解放

Adabas のレコードホールド機能を使用すると、複数のユーザーは同時に同じレコードを更新することができなくなります。 S4（ホールド付き検索）コマンド、L4/L5/L6（ホールド付き読み込み）コマンド、ホールドオプションを指定した A1/E1（更新／削除）コマンド、N1/N2（ホールド付きレコード追加コマンド）、または HI（レコードのホールド）コマンドを使用して、1 つのユーザーが特定のレコードをホールド状態にできます（つまり、そのレコードの ISN がホールドキューに入ります）。

ホールド状態が要求されたレコードがすでに他のユーザーまたはユーティリティによりホールドされている場合、レコードホールドコマンドを発行したユーザーは、レコードが使用可能になるまで待機状態になります。使用可能になった時点で、Adabas はコマンドを実行します。 待機状態になることを回避する場合は、レスポンスコードを返すように要求できます。

ホールド状態のレコードは、レコードをホールドしないユーザーによるアクセス（検索および読み込み）が可能です。

ホールド状態のレコードは、ET または CL コマンドを発行することにより解放されます。 ET および BT コマンドでは、選択的にレコードを解放するオプションを利用できます。 CL コマンドは、ホールド状態にあるすべてのレコードをコマンド発行元ユーザーに解放します。

リソースのデッドロックの回避

2つのユーザーがそれぞれレコードをホールドし、一方のユーザーがホールドしたレコードをもう一方のユーザーがそれぞれ待機する状態になった場合に、リソースのデッドロックが発生します。 Adabas は、デッドロックの可能性を検知すると、一方のユーザーを待機状態に設定してから、もう一方のユーザーにレスポンスコードを返すことにより、このようなユーザーデッドロックを回避します。

排他制御更新

論理トランザクションコマンド（ET/BT）を使用するユーザーは、ET ロジックユーザーと呼ばれます。

ユーザーセッションの間、ユーザーは交互に 1 つ以上の Adabas ファイルの排他制御を要求することができます。 排他制御が要求されているファイルが、別のユーザーまたはユーティリティが更新するためにまだオープンしていなければ、ユーザーに排他制御が許可されて、排他更新（EXU）ユーザーとなります。 Adabas では、EXU ユーザーは非 ET ロジックユーザーとして扱われます。

Adabas は、EXU ユーザーに対して ISN をホールド状態にしません。 Adabas は、排他ファイル制御で更新されるファイルに対しては、ホールドロジック処理を無効にします。

ET コマンドを使用する代わりに、EXU ユーザーは参照ポイントとして動作するチェックポイントを要求できます。例えば、チェックポイント後に適用された更新を取り消すことができます。

タイムアウトの制御

Adabas では、次をタイムアウトすることができます。

- 指定されたリミットを超過したトランザクション
- 指定された期間、非アクティブなユーザー

トランザクションタイムリミット

Adabas では、ET ロジックユーザーに対してトランザクションタイムリミットを指定できます。タイムリミットは、ADARUN の TT パラメータで指定します。OP コマンドを使用して、特定のユーザーに対する優先設定が可能です。

トランザクションが指定されたリミットを超過すると、Adabas は BT（トランザクションのバックアウト）コマンドを生成し、トランザクション処理中に発生した更新情報をすべて削除して、ホールドされているすべてのレコードを解放します。 ユーザーはその後にバックアウトされたトランザクションを初めから繰り返すか、または別のトランザクションを開始することができます。

非アクティビティタイムリミット

すべてのユーザーが非アクティビティタイムリミットの対象となります。各ユーザータイプ、および各ユーザータイプ内の特定のユーザーに対して異なるリミットを設定できます。

ユーザーが指定されたリミットを超過した場合、ユーザーの状態によって処理が異なります。

- ET ロジックユーザーの場合、Adabas は現在のトランザクションをバックアウトして、ホールドされているすべてのレコードおよびコマンド ID を解放し、ユーザーのファイルリストを削除します。

- EXU ユーザーの場合、Adabas はユーザーのファイルリストを削除して、すべてのコマンド ID を解放します。 ユーザーは EXU ユーザー状態ではなくなり、アクセスオンリーユーザーとなります。
- アクセスオンリーユーザーの場合、Adabas はユーザーのファイルリストを削除します。

バックアウト、リカバリ、および再スタート

バックアウト リカバリ、および タイムアウト（「*タイムアウトの制御*」を参照）、Adabas がトランザクションを正常に完了できないと判断したときのプログラムエラー、Adabas、ハードウェアまたはオペレーティングシステムの障害、または電源の障害により、ユーザーまたは Adabas セッションが中断された場合、再スタートが必要です。

*ユーザーセッション*は、一連の Adabas コールから成り、オープン（OP）コマンドで開始し、クローズ（CL）コマンドで終了させることができます。 ユーザーとは、バッチモードプログラム、あるいは端末を使用する人を意味します。 各ユーザーの一意性は、ユーザー ID、マシン、アドレススペース、および端末 ID により確保されます。

*Adabas セッション*は、Adabas がアクティブになったときに開始され、Adabas が終了するまで継続します。 この間に、Adabasニュークリアスは、データベースに対して行われたすべての変更内容の正確な履歴として、一連のプロテクションエントリを作成します。 この一連のプロテクションエントリは、WORK データセット（パート 1）に書き込まれ、さらにプロテクションログへもブロックごとに書き出されます。 各ブロックにはニュークリアスのセッション番号、固有のブロック番号およびタイムスタンプが含まれています。

ユーザープログラムエラー

トランザクションを処理中のユーザープログラムは、トランザクションが正常に完了できないことを検知できます。この場合、BT（トランザクションのバックアウト）コマンドが使用されて、不完全なトランザクションが削除されるか、バックアウトされます。

ユーザープログラムエラーにより、データベースに論理的な誤りが生じた場合、ADASAV および ADARES ユーティリティを使用して影響を受けたファイルをリカバリする必要があります。

Adabas、ハードウェア、またはオペレーティングシステムの障害

Adabas ニュークリアスが異常終了する原因となる何らかの障害が発生すると、Adabas が再アクティブ化されるときに自動プロシージャが実行され、データベースが物理的、論理的に正常な状態になります。 部分的に実行された更新コマンドはすべてリセットされ、未完了トランザクションはすべてバックアウトされます。

この自動プロシージャは、データベースの修復、自動再スタート、および自動バックアウトの3段階で構成されています。

- 障害の発生時にバッファフラッシュが完了していた場合に、データベースを修復すると、データベースは障害発生直前の状態に復帰します。 つまり、データベース内のすべてのブロックは、ニュークリアスが正常に実行される状態となります。

- 自動再スタートは、システム障害の発生時に部分的に実行されていた1つの更新コマンドによる更新部分をバックアウトします。 これによりデータベース内部の整合性が解決され、物理的な保全性が保証されます。
- ET ロジックユーザーに対してのみ実行される自動バックアウトは、システム障害の発生時に部分的に実行されていたユーザートランザクションの更新をバックアウトします。 Adabas は、自動再スタートに続いて内部 BT（トランザクションのバックアウト）を実行し、最新のトランザクションがバックアウトされたことをユーザーに通知します。

自動バックアウトルーチンは、HALT コマンドで終了した ET セッションの終了時に実行されます。 また、自動バックアウトルーチンは、次の Adabas セッションで自動的に実行され、正常に完了しなかったトランザクションで実行されたすべての更新部分を削除します。

自動バックアウトの実行後は、データベースには論理的に完了したトランザクションの更新のみが反映されます。

Note: ET ユーザーは、BT（トランザクションのバックアウト）コマンドを発行して、いつでも手動で未完了のトランザクションのバックアウトを行うことができます。 「*データベースの完全性の維持*」を参照してください。

Adabas、ハードウェアまたはオペレーティングシステムの障害によりデータベースに物理的な障害が発生した場合、ADASAV および ADARES ユーティリティを使用してデータベースを再作成する必要が生じる場合があります。

電源障害

ハードウェアによっては、I/O 処理中に電源障害が発生すると、処理していた Adabas ブロックが破壊されることがあります。 このような損傷は自動再スタート中に検出することができないため、予期しないレスポンスコードやデータベース更新の喪失といった問題が後で発生する可能性があります。

ADARUN の IGNDIB=YES パラメータが設定されている場合、自動再スタートルーチンは、バッファフラッシュがセッションの中断が発生したときにアクティブだったかどうかをチェックします。 バッファフラッシュが処理中だった場合、自動再スタートは終了して、Adabas は可能性のある問題をユーザーに警告して、バッファフラッシュの処理中に更新していたファイルのリストを通知します。 DBA は、電源障害が発生していないかどうか調べる必要があります。

セッションの中断の原因によって、対処方法が異なります。

- *電源障害が原因の場合*、ADASAV および ADARES ユーティリティを使用して、影響を受けたファイルをリカバリすることを強くお勧めします。
- *原因が電源障害ではないことが確実*で、出力ハードウェア上の情報の保全性が保証できる場合は、データベースはすぐに再アクティブ化できます。 データベースのリカバリは必要ありません。

拡張エラー処理およびメッセージバッファリング

エラー処理およびメッセージバッファリング機能は、DBA の介入をほとんど必要とすることなく、特定タイプのエラーを自動的に分析および回復するため、24X7 オペレーションの実装に有効です。 また、追加情報も生成されるため、ユーザー独自によるエラー診断と、Software AG によるエラー診断が可能です。

ラップアラウンド方式のメッセージバッファは、コンソールや DDPRINT メッセージへのオンラインアクセスが不可能になった場合に、後から Adabas Online System で調査するために Adabas メッセージを収集します。 バッファは、問題分析とパフォーマンスチューニングに有効です。

このバッファリングのエラー処理機能は、オペレータコンソールまたは Adabas Online System から起動できます。

Adabas ニュークリアスの操作に不可欠なユーザー出口およびハイパー出口は、クリティカル（デフォルト）またはクリティカルではないとマークすることができます。

- *クリティカル*なユーザー出口については、エラー処理およびメッセージバッファリング機能による影響はありません。出口内の異常終了により、Adabas ニュークリアスも異常終了します。
- *クリティカルではない*ユーザー出口については、メッセージバッファリング機能がアクティブな Adabas ニュークリアスを維持し、場合によってはその出口の起動を止めて、出口エラー発生時のニュークリアスのダンプを取得し、システムログにメッセージを発行して DBA に問題を通知します。 DBA は次に診断情報を調べ、問題を解決します。次に、修正された出口をロードして再アクティブ化します。

拡張機能（プラグインルーチンまたは PIN）は、ニュークリアスが処理を継続できるようにしながらアベンドの原因を分析し、場合によっては原因を特定します。各 PIN サービスルーチンは、事前に定義された条件に遭遇すると、これを処理し、Adabas ニュークリアスに次のどちらかを許可します。

- 異なる方法で異常終了するときは、アクティブなままにする。
- エラーリカバリの補助として、拡張エラー診断を出力する。

異常発生時のレジスタが利用可能なため、PIN の実行中は、大部分の Adabas 機能が PIN に対して有効です。PIN は、ニュークリアスが処理を継続しても安全かどうかを判断し、適切なメッセージを出力して DBA に知らせます。

ニュークリアスが実行できるかどうかにより、PIN は、通常の処理を再開するために制御を Adabas ニュークリアスに移す（通常レスポンスコードを伴う）か、エラー処理およびメッセージバッファリング機能に制御を返して、Adabas ニュークリアスが異常終了できるようにします。

PIN を使用すると、特定のレスポンスコードまたは ABEND コードをデバッグしやすくするために、さまざまな状況に合わせてダンプをわかりやすい形式で出力させることもできます。

特別な PIN ルーチンのユーザー出口を使用して、レスポンスコードおよびアベンドについての詳細情報を得ることができます。 ユーザー出口では、特定のレスポンスコード、またはレスポンスコード／サブコードの組み合わせを監視するように指定することができます。 ユーザー出口を修正すると、それをリロードし、データベースをダウンさせることなく変更を有効にすることができます。

5 Adabas ユーティリティ

ファイルのロードや削除などのデータベースサービスは、オンラインおよびバッチモードのユーティリティの統合セットにより処理されます。ほとんどのユーティリティは、通常のデータベースアクティビティと並行して実行できるため、毎日の運用を中断することはありません。 詳しくは *Adabas* ユーティリティ のドキュメントを参照してください。

Adabas ユーティリティは、初期設計およびロード操作、バックアップ／リストア／リカバリルーチン、データベース更新ルーチン、および監査／制御／チューニング処理の各機能を提供します。

Note: このメニュー操作のインタラクティブな DBA ツールの詳細は、「*Adabas Online System*」を参照してください。

このchapterでは、次のトピックについて説明します。

初期設計およびロード操作

- ADACMP：圧縮／圧縮解除
- ADALOD：ローダー
- ADAULD：アンロード

ADACMP：圧縮／圧縮解除

ADACMP COMPRESS は、ADALOD を使用してデータベースにロードするデータレコードの編集および圧縮に使用します。また、ADACMP DECOMPRESS は、データ構造またはフィールド定義を変更するために、個別のファイルを圧縮解除するときに使用するか、または非 Adabas プログラムの入力用に使用します。

COMPRESS

入力

ADACMP に入力するデータは、シーケンシャルデータセット／ファイル内になければなりません。 インデックス付きのシーケンシャル入力や VSAM の入力は使用できません。 レコード長は固定、可変、あるいは不定長でもかまいません。 最大入力レコード長は、オペレーティングシステムによって決定されます。最大圧縮レコード長は、使用しているデータストレージのブロックサイズおよびそのファイルに設定された最大圧縮レコード長によって制限されます（ADALOD ユーティリティの MAXRECL パラメータを参照）。 入力レコードの形式は、ブロック形式か非ブロック形式のいずれかです。

パラメータ NUMREC=0 が指定されている場合は、入力データセットを省略できます。

COMPRESS への入力データの論理構造と特性は、フィールド定義ステートメントで記述します。FNDEF はフィールドまたはフィールドのグループを定義します。SUBFN および SUPFN はそれぞれサブフィールドとスーパーフィールドを定義します。COLDE、HYPDE、PHONDE、

SUBDE、および SUPDE は、さまざまなタイプのディスクリプタを定義します。 フィールド定義は、Adabas フィールド定義テーブル（FDT）の作成に使用されます。

デフォルトでは、入力データレコードはフィールド定義ステートメントの順で処理されます。 FORMAT パラメータにより、フィールドの処理順を変更したり、フィールドをスキップできます。

ユニバーサルエンコーディングサポート（UES）に対応するために、パラメータを使用して、入力および必要なファイルのデータアーキテクチャおよびユーザーエンコードの指定、および圧縮中に使用するユーザーエンコードの指定が可能です。

処理

ADACMP COMPRESS は、データレコードを編集および圧縮します。

編集では、フィールドごとにパック形式（P）なのかアンパック形式（U）なのかがチェックされ、フィールド値が数値であり、形式に従っているかどうかが検査されます。 誤ったデータが検出されると、そのデータを含むレコードはすべて ADACMP エラーデータセットに書き出され、圧縮データセットには書かれません。 ADACMP COMPRESS の処理中に実行する追加の編集を指定するために、Adabas ユーザー出口 6 が使用されます。 ユーザー出口の詳細は、*Adabas ユーザー出口* のドキュメントを参照してください。

圧縮では、英数字フィールドからの末尾の空白の除去、数値フィールドからの先行ゼロの除去、浮動小数点フォーマットフィールドからの末尾のゼロの除去、およびアンパック形式の数字フィールドのパック形式化が行われます。 固定（FI）オプションが指定されたフィールドは圧縮されません。また、レコードの終わりにある空白のフィールドは格納も圧縮もされません。 空値フィールドは、使用されているオプションによって処理方法が異なります。 SQL 空値処理がサポートされています。

ユニバーサルエンコーディングサポート（UES）パラメータが指定されている状態で圧縮が実行されると、ファイル圧縮時の指定エンコードに入力データが変換されます。

出力

処理対象のデータレコードは、ファイル定義情報とともに、可変長ブロック式のレコードフォーマットのシーケンシャルデータセットに書き込まれます。 このデータセット、または複数の ADACMP を実行した結果の複数データセットは、ADALOD ユーティリティへの入力として使用できます。 このデータセットは、レコードが 1 件もなくても、ADALOD への入力として使用できます。つまり、入力データセットにレコードが 1 つも入力されなくても、または編集処理でレコードがすべて排除されていてもかまいません。

ADACMP 処理レポートには、圧縮レコードに必要なデータストレージ内の概算スペース量がデバイスタイプ（DEVICE パラメータで指定）別に示されます。また、5 %～30 % の間のデータストレージパディングファクタに必要な概算スペース量も表示されます。 圧縮率は、圧縮ルーチンに実際に入力されたデータ量に基づいて計算されます。

DECOMPRESS

ADACMP DECOMPRESS は既存の Adabas ファイルのレコードを入力データとして受け付けますが、個別のファイルをアンロードせずに直接受け付けることも、ADAULD ユーティリティを使用してすでにアンロードされているファイルを受け付けることもできます。ファイルを直接、圧縮解除する場合、ファイルは、FDT 情報がない状態で圧縮解除処理の一部としてアンロードされるため、大きなファイルの圧縮解除時に時間を節約できます。

マルチクライアントファイルを直接、圧縮解除する場合、有効なオーナー ID（ETID パラメータ）が指定された場合にのみ、特定のユーザー向けのレコードに制限できます。

FDT で指定されたフォーマット以外のフォーマットにレコードを圧縮解除する場合は、FORMAT パラメータを使用できます。 これは、既存のファイルの FDT を変更するときに、特に便利です。

ユニバーサルエンコーディングサポート（UES）が使用されている場合、圧縮解除ファイルのエンコードの特性が、圧縮されたシーケンシャル入力のヘッダーに渡されます。 パラメータを指定することで、これらのエンコードの特性を上書きすることができます。

処理対象のデータレコードは、可変長ブロック式のレコードフォーマットのシーケンシャルデータセットに書き込まれます。 拒否されたデータレコードは、エラーデータセットに書き込まれます。

ADALOD：ローダー

ADALOD LOAD 機能は、ファイルをデータベースにロードします。 ADACMP または ADAULD ユーティリティで生成された圧縮レコードをこのユーティリティの入力に使用できます。 パラメータにより、ファイルインデックスが圧縮形式または非圧縮形式のどちらでロードされるを指定します。

ADALOD はデータストレージに圧縮レコードをロードし、ファイルのアドレスコンバータを作成し、ファイルのフィールド定義をフィールド定義テーブル（FDT）に格納します。 また、ADALOD はファイルに定義されている全ディスクリプタの値と、その値が存在する全レコードの ISN を抜き出し、中間データセットに書き出します。 その後、このデータセット内の情報は値／ISN の順番にソートされ、アソシエータのインバーテッドリストに書き込まれます。

ADALOD UPDATE 機能は、大量のレコードを Adabas ファイルに追加したり、Adabas ファイルから削除したりする際に使用できます。 UPDATE 機能を使用する場合、Adabas のレコードの追加／削除コマンドを繰り返し実行するのに比べて処理時間を大幅に節約することができます。 追加するレコードは、ADACMP または ADAULD ユーティリティで生成された圧縮レコードを使用します。 削除するレコードの ISN は、入力データセットに用意するか、またはコントロールステートメントに指定することができます。

1 回の ADALOD 実行中に、レコードの追加と、別のレコードの削除を行うことができます。

ADAULD：アンロード

ADAULDユーティリティは、Adabasファイルをデータベースまたはセーブテープからアンロードします。

Adabas ファイルは次の目的のためにデータベースからアンロードされます。

- 非Adabasプログラムによるデータの処理を可能にするため。この場合、ファイルをアンロードした後、ADACMP ユーティリティの DECOMPRESS 機能を用いてファイルを圧縮解除しなければなりません。
- 同じデータをもつ1つまたは複数のテストファイル作成のため。この手順では、ファイルをアンロードした後、別のファイル番号が付いたテストファイルとして再ロードする必要があります。
- フィールド定義テーブル（FDT）を変更するため。これにはファイルをアンロードし、圧縮解除し、変更したフィールド定義を使用して圧縮し、さらに再ロードしなければなりません。ADADBS ユーティリティを使ってフィールド定義をファイルに追加する場合は、ファイルを最初にアンロードする必要はありません。

レコードをアンロードする順序は、次のいずれかが可能です。

物理	レコードをデータストレージ内に物理的に位置している順序でアンロードします。
論理	レコードをユーザーが指定したディスクリプタの値の順序でアンロードします。
ISN	レコードを ISN の昇順にアンロードします。

アンロードされたレコードは圧縮形式で出力されます。出力レコードの形式は、ADACMPユーテイリティで作成されたレコードと同じです。

Adabasファイルは、次の目的のために、適切なデータベースまたはファイルセーブテープからアンロードできます。

- セーブテープからのファイルを、異なるテスト環境に含めるため
- 相互にブロックサイズの異なるセーブテープからデータベースにファイルを移動するため

バックアップ／リストア／リカバリルーチン

- ADAPLP：プロテクションログ／ワークの出力
- ADARAI：Recovery Aid
- ADARES：再スタート
- ADASAV：データベースまたはファイルの保存／リストア

- ADASEL：プロテクションデータの選択

ADAPLP：プロテクションログ／ワークの出力

ADAPLP ユーティリティは、Adabas WORK データセットまたは Adabas データプロテクションログ内のデータプロテクションレコードを出力します。 次の出力対象を選択することができます。

ALL	全プロテクションレコード（デフォルト）
ASSO	アソシエータプロテクションレコードのみ
DATA	データストレージプロテクションレコードのみ
C1	Adabas C1 コマンドによる発生レコード
C5	Adabas C5 コマンドによる発生レコード
EEKZ	ニュークリアスバッファフラッシュの終了時に書かれたレコード
ET	Adabas ET コマンドによる発生レコード
REPR	自動再スタートがインデックスの修復に使用する WORK データセットレコード
SAVO	オンライン SAVE（データベースまたはファイル）操作による発生レコード
VEKZ	更新コマンドの終了時に書かれたレコード

ファイル、ISN または RABN を指定して、出力されるプロテクションレコードの数をより削減できます。

ADARAI：Recovery Aid

Adabas Recovery Aid ユーティリティ ADARAI を使用すると、データベースのリカバリを自動化し、最適化できます。 詳細は、「*Adabas 再スタート／リカバリ*」を参照してください。

ADARAI は、Adabas と互換性があるすべてのテープ管理システムをサポートしています。

ADARAIユーティリティは、リカバリログファイル（RLOG）を準備します。このファイルは、リカバリジョブの制御ステートメントを構築するために必要なデータセット、ユーティリティのパラメータ、およびプロテクションログに関する情報を記録します。 ADARAI は、RLOG に含まれている情報をリスト化し、データベースをリカバリするためのジョブ制御ステートメントを作成します。また、ADARAI のロギングを無効にします。

情報は、世代ごとにRLOGに保存されます。 世代には、一連のADASAV SAVE/RESTORE（データベース）またはRESTORE GCBのオペレーション間のすべてのアクティビティが含まれます。第 1 世代には、最初の ADASAV SAVE/RESTORE（データベース）または RESTORE GCB オペレーションが含まれ、第 2 世代へ拡張されますが、第 1 世代には第 2 世代の情報は含まれません。

RLOG には、ADARAI PREPARE 手順の間に MINGENS パラメータで指定された最小数の世代が保持されます。 ただし、スペースが十分にある場合は、最大 32 世代が RLOG に保存されます。

Recovery Aid 機能を使用しているシステムには、リカバリログ（RLOG）データセットの DD/RLOGR1 が必要です。このデータセットには、最初に ADAFRM ユーティリティによるフォーマット、次に ADARAI ユーティリティによる定義が必要です。

ADARES：再スタート

ADARES ユーティリティは、データベースリカバリに関する次の機能を実行します。

- BACKOUT は、2 つのチェックポイント間に適用されたすべての更新を取り消します。 使用されるチェックポイントは、通常は非同期チェックポイントコマンド（C1）の結果生じたものですが、同期チェックポイントの場合もあります。 データベース全体をバックアウト処理の対象にすることもできます。または、バックアウト対象を選択したファイルに限定することもできます。
- CLCOPYは、ディスクからシーケンシャルデータセットにコマンドログデータセットをコピーします。 この機能は、Adabasセッションでデュアルまたはマルチコマンドロギングが有効な場合にのみ必要となります。
- COPYは、Adabasのシーケンシャルプロテクションログデータセットをコピーします。 シーケンシャルプロテクションログデータセットが作成されたAdabasセッションが異常終了した場合、この機能を実行する必要があります。
- MERGE CLOGは、それぞれのニュークリアスCLCOPYの実行により生じたコマンドログデータセットを、ニュークリアスのクラスタ用の単体のコマンドログに手動でマージします。
- PLCOPY は、ディスクからシーケンシャルデータセットにプロテクションログデータセットをコピーします。 この機能は、Adabasセッションでデュアルまたはマルチプロテクションロギングをが有効な場合にのみ必要となります。
- REGENERATE は、2 つのユーザー指定チェックポイント間に行われたすべての更新を再適用します。 指定されたチェックポイントは、通常は非同期チェックポイントコマンド（C1）の結果生じたものですが、同期チェックポイントの場合もあります。 REGENERATE 機能は、処理対象をすべてのファイルにすることも、1つ以上のファイルに限定することも可能です。この機能は、データベース（または 1 つ以上のファイル）が、ADASAV ユーティリティの RESTORE または RESTONL 機能を使用して以前の状態にリストアされた後に、最も多く使用されます。
- REPAIR は、いかなる理由であれ使用不可能となったデータストレージ内の1つ以上のブロックを修復します。 データベースの最新のセーブテープ、およびそれ以降に作成されたすべてのプロテクションログテープが、この機能への入力として使用されます。

システム障害から復帰するために必要な時間を最小限に抑えるため、ADARES の BACKOUT、BACKOUT DPLOG または MPLOG、および REGENERATE 機能を、同時に複数のコマンドがアクティブな元の更新環境をシミュレートする複数のスレッドで実行できます。

ADASAV：データベースまたはファイルの保存／リストア

ADASAV ユーティリティは、データベースまたは 1 つ以上のファイルの内容を、順次データセットに保存、またはシーケンシャルデータセットからリストアします。 ADASAV の実行が必要な頻度は、データベース内のファイルの数やサイズ、および更新の量とタイプによって決まります。 データベースの規模が大きい場合、ADASAV 機能はデータベースが構成されているいくつかのディスクパックに並行して実行することができます。

Adabas Delta Save Facility で使用するには、専用の ADASAV 機能を使います。 詳細については、*Adabas Delta Save 機能*のドキュメントを参照してください。

RESTONL 機能は、Adabas ニュークリアスがアクティブ（つまりオンライン）であった期間に作成された 1 つ以上の SAVE データセットからリストアします。RESTORE 機能は、Adabas ニュークリアスが*非アクティブ*（つまりオフライン）であった期間に作成された 1 つ以上の SAVE データセットからリストアします。

RESTONL および RESTORE には副機能（GCB、FILES、FMOVE）があります。

- 副機能を指定しないと、RESTONL および RESTORE はデータベース全体をリストアします。
- GCB 副機能を指定すると、RESTONL および RESTORE 機能は、ジェネラルコントロールブロック、データベースのアソシエータ RABN 3～30、および指定ファイルをリストアします。
- FILES 副機能を指定すると、RESTONL および RESTORE は、1 つ以上のファイルを既存のデータベースの元の RABN にリストアします。
- FMOVE 副機能を指定すると、RESTONL および RESTORE は、既存データベースのエクステントサイズを変更できるフリースペースに 1 つ以上のファイルをリストアします。

オンライン SAVE 中に変更が生じると、RESTONL 機能に続き、RESTPLOG 機能が自動的に実行されます。 RESTPLOG は、オンライン SAVE 処理中に生じたためにオンライン SAVE に含まれていない更新を適用します。

プロテクションログ（PLOG）の更新が完全にリストアされる前に終了した RESTONL や RESTONL FILES 機能の後でも、RESTPLOG は実行されます。 RESTPLOG は、RESTONL 機能が正常に実行されなかったために適用されなかったデータベース更新を適用します。

データベースまたは 1 つ以上のファイルをセーブするための SAVE 機能は、Adabas ニュークリアスがアクティブなとき（オンライン）にも、アクティブでないとき（オフライン）にも実行できます。 リカバリエイドのオプションがアクティブな場合は、SAVE データベース処理が新しく RLOG 生成を開始します。

ADASEL：プロテクションデータの選択

ADASEL ユーティリティは、Adabas シーケンシャル（SIBA）、デュアル、またはマルチ（PLOG）プロテクションログの中の情報を選択します。 ADASEL は、情報を圧縮解除して出力データセット（DD/DRUCK）またはユーザー指定の出力データセットに書き込みます。

プロテクションログの中には、任意の Adabas セッション中にデータベースに適用されたすべての更新の情報が入っています。 ADASEL で選択した情報は、監査に使用したり、Natural や非 Adabas プログラムへの入力として使用することができます。

新規のレコード、更新されたレコード、および削除されたレコードについて、ビフォーイメージ、アフターイメージ、または両方のイメージを選択できます。 また、Adabas C5 コマンドを用いて、データプロテクションログに書き込まれたデータを選択することもできます。

データベース更新ルーチン

ADACDC：変更データの取得

ADACDCは、すべてのデータベース変更情報が含まれるシーケンシャルファイルを生成する、一定間隔で実行される非同期型の大量更新機能です。 この機能は、オープンシステムやデータウェアハウスソリューションでは重要です。

ADACDCは、データの最新のステータスに影響しないように、シーケンシャルファイル内の未加工データを処理します。 ADACDC ユーティリティは、次のように処理を行います。

- 1 つ以上のシーケンシャルプロテクションログを入力として使用します。
- 入力プロテクションログがカバーしている期間のすべてのデータベースの変更のデルタを出力として作成します。

*変更のデルタ*とは、ファイルの各 ISN に対するこの期間中の最新の変更を意味し、プライマリ出力ファイルに出力されます。

プライマリ出力ファイルはデータウェアハウスのデータ作成プロシージャの入力として使用され、データベース全体のコピーではなくデータベース変更のデルタがデータウェアハウスデータ

ベースに適用されます。 このことによって、データウエアハウス更新の時間削減と頻度の増加を実現でき、格納される情報の精度を高めることができます。

ADACNV：データベース変換

Adabas バージョン間でデータベースを移動する場合に、オペレーティングシステム依存および非依存のデータベースシステム構造に必要な変換をすべて実行するために ADACNV ユーティリティを*使用する必要があります*。

ADACNV ユーティリティは、Adabas データベースを下位バージョンから上位バージョンに変換（CONVERT）します。また、このユーティリティは、Adabas データベースを上位バージョンから下位バージョンに逆変換（REVERT）します。 これらの機能には、いくつかの制約が適用されます。

データベースの完全性を確保するために、最初に ADACNV は、変更するブロックを中間ストレージ、すなわちシーケンシャルデータセット DD/FILEA に書き込みます。 すべての変更するブロックがDD/FILEAに書き込まれると、非復帰点に到達して、変更するブロックがデータベースに書き込まれます。 ADACNV が非復帰点以降に異常終了した場合は、RESTART パラメータを指定し ADACNV を実行させると、DD/FILEA の内容を読み取ってデータベースに書き込みます。

TEST パラメータは、変換または逆変換できるか調べるためのパラメータで、変更をデータベースに書き込みません。

ADADBS：データベースサービス

すべてのADADBS機能はAdabasOnlineSystem（AOS）を使用して実行することもできます。Adabas Recovery Aid がアクティブな場合、AOS はリカバリ操作に必要なチェックポイントを書き込むため、ファイル変更操作には AOS の使用をお勧めします。

ADADBS は、さまざまな機能を提供します。1 回のユーティリティの実行中には、それらの機能をいくつ実行してもかまいません。

データベース機能

ADD 機能は、アソシエータまたはデータストレージに新しいデータセットを追加します。データセットはそれぞれ最大 99 個まで追加できます。 ただし、最大数は、最初のアソシエータデータセット（DDASSOR1）のブロックサイズに依存するため、実際の最大数はこれより小さくなります。

DECREASE 機能は、現在アソシエータまたはデータストレージに使用されている最終のデータセットのサイズを縮小します。 解放するスペースは、フリースペーステーブル内で利用可能でなければなりません。

DECREASE 機能は、指定された物理エクステントスペースをすべて*割り当て解除するわけではありません*。スペースの割り当てを解除するには、DECREASE機能を使用してデータベースを

縮小し、ADASAV SAVE を使用して保存します。次に、ADAFRM を使用してデータセットを再フォーマットし、ADASAV を使用してデータベースをリストアします。

INCREASE 機能は、現在アソシエータまたはデータストレージに使用されている最終のデータセットのサイズを拡張します。 この機能は、アソシエータに対しては何回でも実行できます。データストレージスペーステーブル（DSST）が最大数（99）に達すると、すべてのデータストレージエクステントは、ADAORD ユーティリティの REORASSO または REORDB 機能を使用して単体のエクステントに結合する必要があります。

RENAME 機能は、ファイルまたはデータベースに割り当てられた名前を変更します。ファイルを指定していない場合、またはファイル番号 0 のファイルを指定している場合は、データベースの名前が変更されます。

TRANSACTIONS 機能は、更新トランザクション処理を中断および再開します。つまり、リカバリ可能な開始ポイントにできるデータベースの静止状態にします。

ファイル機能

ALLOCATE/DEALLOCATE 機能は、特定のサイズの論理エクステント（アドレスコンバータ、データストレージ、ノーマルインデックス、アッパーインデックス）をそれぞれ割り当ておよび割り当て解除するために使用します。 1 回の ADADBS の実行で、1 つのエクステントについてのみ割り当て、または割り当てを解除が可能です。

CHANGE 機能は、Adabas フィールドの標準長を変更しますが、データストレージ内のレコードは変更されません。したがって、Adabas の定義に基づく新しい標準長とレコード内の実際のバイト数が違うため、ユーザーは、結果が正しくならないフィールドを参照しないようにする必要があります。

DELETE 機能はデータベースから Adabas ファイルを削除します。 カップリングされたファイルは削除できません。 Adabas 拡張ファイルを指定すると、拡張ファイル全体（アンカーおよびコンポーネントファイル）が削除されます。 削除処理により、ファイルに割り当てられたすべての論理エクステントが割り当て解除され、新しいファイルまたは既存のファイルの新しいエクステントが使用できるようにスペースを解放します。

DSREUSE 機能は、指定されたファイルに対して、レコードの削除により空きとなったデータストレージブロックが再使用されるかどうかを決定します。 ブロックの再使用は、本来 ADALOD FILE 機能でデータベース内にファイルをロードするか、あるいは ADADEF DEFINE 機能でシステムファイルが定義される際に決定されるものです。 両方の場合で、デフォルトのブロック再使用は YES に設定されています。

ユニバーサルエンコーディング（UES）をサポートするために、ENCODEF 機能を使用して、すでにロードされているファイルのフィールドに対するエンコードを定義できます。

- 英数字フィールドには EBCDIC ファイルエンコードが適用されます。
- ワイド文字フィールドにはユーザーエンコードが適用されます。ワイド文字フィールドのファイルエンコードは、この機能では変更できません。

ISNREUSE 機能は、指定されたファイルに対して、Adabas が新しいレコードに削除されたレコードの ISN を再使用するかどうかを決定します。 再使用しない場合は、次の番号の未使用の ISN が割り当てられます。

システムファイル以外の Adabas ファイルが指定された場合、MODFCB 機能は、アソシエータまたはデータストレージのファイルパディングファクタ、データストレージ、ノーマルインデックス、アッパーインデックスの2次論理エクステント割り当て最大サイズ、許容最大圧縮レコード長などのパラメータを変更します。また、特別な E1 コマンドの発行によるファイル更新操作をユーザープログラムに許可するかどうかのパラメータを変更します。

NEWFIELD 機能は、1 つ以上のフィールドを指定されたシステムファイル以外の Adabas ファイルに追加します。 新フィールド定義がフィールド定義テーブル（FDT）の末尾に追加されます。NEWFIELDは、新しいフィールドの実際のデータストレージデータを指定するためには使用できません。データは、後で Adabas の追加または更新コマンド、または Natural コマンドを使用して指定できます。

ONLINVERT 機能により、オンラインアプリケーションがアクティブなときにファイルのインバートが可能になります。これによりファイルに継続的にアクセスできるようになります。1回の実行で 1 つのファイルに 1 つのディスクリプタを追加できます。

ONLREORFASSO（アソシエータのリオーダ）、ONLREORFDATA（データストレージのリオーダ）、および ONLREORFILE（アソシエータとデータストレージ両方のリオーダ）機能により、オンラインアプリケーションがアクティブなときにファイルリストのリオーダが可能になります。これによりファイルに継続的にアクセスできるようになります。 ファイルは、既存のエクステント内でリオーダされます。これにより、空きスペースが回復され、処理の必要性に応じてデータレコードのソート順が変更されるため、I/O のパフォーマンスが向上します。

REFRESH 機能は、ファイルにロードされたレコードを "0" 件の状態にセットし、アドレスコンバータ、データストレージ、ノーマルインデックス、およびアッパーインデックスに割り当てられた第1エクステントを空の状態にセットし、また他のエクステントの割り当てを解除します。

RELEASE機能は、ディスクリプタをディスクリプタ状態から解除します。このディスクリプタに対するインバーテッドリストにより占有されているアソシエータ内のすべてのスペースは解放されます。ただし、この機能により解放されたスペースは、リオーダまたはADALOD UPDATEを実行すれば再利用できます。 データストレージには何の変更も行われません。

RENAME機能は、ファイルまたはデータベースに割り当てられた名前を変更します。ファイルを指定していない場合、またはファイル番号0のファイルを指定している場合は、データベースの名前が変更されます。

RENUMBER 機能は、システムファイル以外の Adabas ファイルの番号を変更します。 新しい番号がすでに別のファイルに割り当てられている場合、RENUMBER 機能は実行されません。

UNCOUPLE 機能は、2 つのファイルの間のカップリング関係を解消します。

その他の機能

CVOLSER機能は、ボリュームシリアル番号で指定されたディスクボリューム上に含まれているAdabas ファイルエクステントを出力します。

DELCP 機能は、指定された日付まで（その日付を含む）に記録されたチェックポイント情報を削除します。指定された日付以降に記録されたチェックポイントは削除されません。 ADADBS DELCPの実行後、残りのレコードには、新しいISNが割り当てられます。このISNには、チェックポイントレコードの削除時に使用されたISN も含まれます。 より低いISN が割り当てられますが、チェックポイントの時間的な順序は保持されます。

OPERCOM機能は、Adabasニュークリアスにオペレータコマンドを発行します。 Adabasは、コマンド実行を確認できるように、オペレータにメッセージを発行します。クラスタ環境では、OPERCOM コマンドのほとんどの発行先は、クラスタ内の別のニュークリアス、または実行対象のクラスタ内のすべてのニュークリアスになります。

PRIORITY機能は、ユーザーのAdabasプライオリティの設定および変更を行います。ユーザーのプライオリティは、0（最低）から255（最高、デフォルト）までの範囲があります。 このプライオリティ値は、リージョン間コミュニケーションメカニズムにより、オペレーティングシステムプライオリティに追加されます。 プライオリティが設定または変更されるユーザーは、Adabas コントロールブロック（OP コマンド、アディション1フィールド）で提供されたユーザー ID と同じユーザー ID で識別されます。

RECOVER 機能は、フリースペーステーブル（FST）を再構築することで、割り当てられたスペースを復元します。 RECOVERは、使用可能スペースの合計からファイルエクステントおよびDSST エクステントを差し引きます。

REFRESHSTATS 機能は、現セッションで Adabas ニュークリアスに保守された、統計の値をリセットします。 次のパラメータを使用して、機能の対象を特定の統計値グループに制限することができます。

- ALL（デフォルト）は、CMDUSAGE、COUNTERS、FILEUSAGE、POOLUSAGE および THREADUSAGE の値をすべてリセットします。
- CMDUSAGEは、L*x*、S*x*、またはA1などのAdabasダイレクトコールコマンドのカウンタをリセットします。
- COUNTERSパラメータは、ローカルまたはリモートコール、物理または論理コール、フォーマット変換、フォーマット上書き、自動再スタート、プロテクションログスイッチ、バッファフラッシュ、およびコマンドスローバックに対するカウンタフィールドをリセットします。
- FILEUSAGE は、各ファイルに対するコマンドのカウントをリセットします。
- POOLUSAGE パラメータは、ワークプール、コマンドキュー、ユーザーキューなどのニュークリアスプールの最大値をリセットします。
- THREADUSAGE は、各 Adabas スレッドに対するコマンドのカウントをリセットします。

Adabasは、各Adabasユーティリティが使用しているファイルのリストをデータ保全ブロック（DIB）に保持します。 DDIBオペレータコマンド（またはAdabas Online System）を実行する

と、このブロックの内容が表示されるので、どのジョブがどのファイルを使用しているかを判断することができます。ユーティリティは正常終了時には、DIBからエントリを削除します。ユーティリティが異常終了した場合（例えば、オペレータによるジョブのキャンセル）、そのユーティリティが使用しているファイルの状態は使用中のままです。 RESETDIB 機能は、このようなすべてのファイルを解放して、指定されたジョブまたは特定のユーティリティの実行、またはその両方に対する DIB のエントリをリセットします。

ADADEF：データベースの定義

ADADEF は、次の目的で使用されます。

- チェックポイントファイルが含まれる、新しいデータベースを定義する（DEFINE 機能）
- 新しいデータベースに対するデフォルトのデータベースエンコードを設定するため、または既存のデータベースに対するデフォルトのデータベースエンコードを変更する（MODIFY機能）
- 既存のデータベースに対して、新しいワークファイルを定義する（NEWWORK 機能）

データベースは、名前、ID、デバイスタイプおよびサイズが指定されたコンポーネント（アソシエータ、データストレージおよびワーク）、デフォルトエンコードにより定義されます。

Adabasは特定のファイルを使用して、システム情報を格納します。チェックポイントファイルは、チェックポイントデータの格納に使用されます。また、Adabas CL および ET コマンドに提供されるユーザーデータも格納されます。 このファイルは必須であり、ADADEF DEFINE（データベース）機能で必ず指定してください。

データベースコンポーネント（アソシエータ、データストレージ、およびワーク）を ADADEF で定義できるように、それぞれのコンポーネントをADAFRMユーティリティでフォーマットしておく必要があります。

ADAFRM：データセットのフォーマット

ADAFRM ユーティリティは、Adabas ダイレクトアクセス（DASD）データセット、つまり、アソシエータデータセット、データストレージデータセット、および WORK データセットおよび中間ストレージ（中間、ソート、リカバリログ、およびデュアルまたはマルチコマンド／プロテクションログ）データセットをフォーマットします。

ADAFRMを使用するフォーマットは、次の2つの基本操作で構成されます。まず、指定されたトラック／シリンダ上でブロック（RABN）を作成します。次に、作成したブロックをバイナリの 0（空値）で埋めます。

すべての新しいデータセットは、フォーマットしないと Adabas ニュークリアスまたは Adabas ユーティリティにより使用できません。 ADADBS INCREASE または ADD 機能を使用してデータセットを増やした場合には、新しい RABN もフォーマットしなければなりません。

ADAFRMはまた、既存のアソシエータ、データストレージ、またはワークのブロックをバイナリの 0（空値）にリセットする機能も備えています。

同じジョブの中で複数の ADAFRM 機能（ASSOFRM や DATAFRM など）を実行することができます。 しかし、各機能は個別のステートメントで分けて指定しなければなりません。

ADAINV：インバート

ADAINV は、次の目的で使用されます。

- ディスクリプタを作成する（INVERT 機能）
- 2 つのファイルをカップリングする（COUPLE 機能）

INVERT 機能

- フィールド定義テーブル（FDT）を変更し、指定したフィールドがディスクリプタであることを示します。
- フィールドのすべての値と対応する ISN リストをインバーテッドリストに追加します。

新規に定義したディスクリプタは、他のディスクリプタと同様に使用できます。 この機能はサブディスクリプタ、スーパーディスクリプタ、フォネティックディスクリプタ、ハイパーディスクリプタ、または照合ディスクリプタの作成にも使用することができます。

COUPLE機能は、2つのファイルについての共通のディスクリプタを追加します（インバーテッドリストを更新します）。2つのファイル間に同一のフォーマットおよび長さで定義してある共通のディスクリプタが存在する場合、その2つのファイルはカップリングすることができます。1 つのファイルは、18 個まで他のファイルとカップリングが可能ですが、2 つのファイル間には同時に 1 個のカップリング関係だけが許されます。 自分自身とのカップリングは行うことはできません。

Note: ファイル番号が 255 以下のファイルだけがカップリングできます。

カップリングしているファイルのディスクリプタのインバーテッドリストのいずれかに変更が行われると、他のファイルは自動的に変更されます。 カップリングのベースとなるディスクリプタを更新する際や、カップリングがされているいずれかのファイルにレコードを追加したり削除したりする際に、カップリングリストの更新に必要となるオーバーヘッドについて DBA は十分に考慮すべきです。 例えば、カップリングのベースとして使用されているフィールドに大量の空値が含まれていて、NU（空値省略）オプションが指定されていない場合、実行時間が非常に長くなり、カップリングリストを格納するためのディスクスペースが大量に必要となります。

中断された ADAINV オペレーションを再開させる場合には、その前にファイルをリストアしておく必要はありません。

ADAORD：リオーダ

ADAORD ユーティリティでは、3 種類の機能を使用できますが、1 回の ADAORD 実行で実行できる機能は 1 つだけです。

リオーダ機能

REORASSO 機能は、すべてのファイルの全アソシエータブロックを物理的にリオーダします。REORFASSO は、単一ファイルのアソシエータをリオーダします。 この機能は、アソシエータのスペースフラグメントを解消し、アドレスコンバータ、ノーマルインデックス、アッパーインデックス、およびデータストレージスペーステーブル（DSST）の各コンポーネントの複数のエクステントをコンポーネントごとに 1 つの論理エクステントに結合します。

REORDATA 機能は、データベース内の全ファイルに対してデータストレージをリオーダします。REORFDATA は、1 つのファイルについてデータストレージをリオーダします。 この機能は、空ブロックしか持たないエクステントを圧縮し、またファイル削除によって発生するデータストレージ内のフラグメントを削除します。

REORDB 機能は、1 回の ADAORD 実行で REORASSO と REORDATA 機能の両方を実行します。REORFILE 機能は、1 回の ADAORD 実行で REORFASSO と REORFDATA 機能の両方を実行します。 レコードはディスクリプタ、ISN による論理的な順番、またはレコードが現在格納されている順にリオーダできます。

再構築機能

RESTRUCTURE 機能は、異なる物理デバイスにデータベースや指定されたファイルを再配置するために使用します。

RESTRUCTUREDB 機能は、データベース全体を 1 つのシーケンシャルデータセットにアンロードします。RESTRUCTUREF は、1 つ以上のファイルを 1 つのシーケンシャルデータセットにアンロードします。 このデータセットは、STORE 機能の入力として使用できます。

格納機能

STORE 機能は、RESTRUCTURE 機能または REORDB 機能の出力を使用して、1 つ以上のファイルを既存データベースにロードします。

ADAZAP：物理データベースブロックの修正

ADAZAPユーティリティは物理データベースブロックの修正に使用されます。 用途は次のとおりです。

- データベース更新の監査証跡を提供する各 VER および REP 用のチェックポイントの書き込み。 SYNP 3F チェックポイントは、Adabas Online System および ADAREP の両方により出力されます。また、ADARES はこれを無視します。
- 標準 Adabas ユーティリティ規則に従ってエラーを処理する。

ADAZAP の実行時には、次の点に注意が必要です。

- ADAZAP を実行する前に、現在のセーブテープを用意してください。 ADAZAP の実行中にエラーが発生した場合は、影響を受けたファイルまたはデータベースをリストアする必要があります。
- マスタコードは、その使用を制御できる認可された人のみが使用できます。マスタコードは、文書による請求を受け付けた後に、Software AG から配布されます。

監査／制御／チューニング処理

ADAACK：アドレスコンバータのチェック

ADAACK は、診断以外の目的では使用できません。 診断する内容は次のとおりです。

- アドレスコンバータが、指定されたファイルおよび ISN の範囲内にあるかどうかをチェックします。 ADAICK と組み合わせて使用します。
- 各アドレスコンバータエレメントを参照して、データストレージRABNがファイルコントロールブロックで指定されたデータストレージエクステントの使用部分内にあるかどうかをチェックします。
- 指定した ISN 範囲内に存在する、各データストレージブロック内の各レコードの ISN を参照して、その ISN に対するアドレスコンバータエレメントに正しいデータストレージ RABN が含まれているかどうかをチェックします。

ADADCK：データストレージのチェック

ADADCK は、診断以外の目的では使用できません。 ADADCK は、データベース内の特定のファイル（複数ファイルも可）のデータストレージおよびデータストレージスペーステーブル（DSST）をチェックします。

ADADCKは、各使用データストレージブロックをファイルコントロールブロック内のデータストレージエクステントに従って読み込み、次のチェックを行います。

- ブロック長が許容範囲内かどうか（4 ブロック長の物理ブロックサイズ）。
- データストレージブロック内のすべてのレコード長の合計に4を加えるとブロック長と等しくなるかどうか
- ファイルの最大圧縮レコード長を超えるレコード長、または長さが0のレコードがあるかどうか。
- 1 つのブロック内に重複した ISN があるかどうか。
- 関連するDSSTエレメントが正しい値を持っているかどうか。正しい値を持っていない場合、DSST は修復が必要です（ADADCK の REPAIR パラメータについての説明を参照）。

ADAICK：インデックスおよびアドレスコンバータのチェック

ADAICK は、診断以外の目的では使用できません。 このユーティリティはアソシエータの物理構造をチェックします。このチェックでは、ディスクリプタ値の構成と、ジェネラルコントロールブロック（GCB）およびファイルコントロールブロック（FCB）に定義されたアソシエータエクステントに従って、インデックスの整合性チェックが行われます。

ADAICK は次の機能を実行できます。

- 特定のファイルのインデックスやアドレスコンバータのチェック
- データベース内のすべてのアソシエータまたはデータストレージブロックの内容の出力／ダンプ
- GCB、FCB、および FDT の内容の、形式付き出力／ダンプの作成

ADAMER：ADAM 見積り

ADAMER ユーティリティは、ADAM ディスクリプタを使ってレコードを検索し、読み込むのに必要なデータストレージのアクセス回数を示す統計表を作成します。 この情報は次のことを判断するのに使用します。

- ADAM ディスクリプタを使ってレコードを取得するために必要なアクセス回数が、標準の Adabas アクセス方法より少ないかどうかを判断します。
- ADAM ディスクリプタをランダマイジングして、最適レコード分布を行うのに必要なデータストレージスペースを決定します。

ADAMER の入力データは、ADACMP または ADAULD ユーティリティによって生成されたファイルの圧縮レコードを含むデータセットです。

ADAM ディスクリプタとして使用するフィールドは、ADAMDE パラメータで指定します。 マルチプルバリューフィールドやピリオディックグループ内のフィールドは使用できません。 ランダマイジングのベースとして、ディスクリプタの代わりにレコードに割り当てられる ISN を使用することもできます（ADAMDE=ISN）。

ADAMディスクリプタは各レコードでユニークな値をもたなければなりません。これは、ADAMディスクリプタに重複した値が存在すると、ADALOD ユーティリティの ADAM オプションでファイルを正常にロードできないからです。 ADAMER ユーティリティには、ユニーク（UQ）に定義されたディスクリプタフィールドが必要ですが、このユーティリティは値がユニークかどうかのチェックを行いません。ディスクリプタ値がユニークかどうかのチェックは、ADALOD ユーティリティが ADAM ファイルとしてファイルをロードするときに行います。

BITRANGE パラメータを用いて、ランダマイジングアルゴリズムの入力として使用する前に、各ADAMディスクリプタ値からビット数を切り捨てるように指定できます。こうすれば、同じ値で始まるADAMディスクリプタ値（例えば、40643210、40643220、40643344）をもつレコードをデータストレージ内の同一物理ブロックにロードできます。この手法により、ADAMディスクリプタをレコードの読み込み順の制御に使用すると、ファイルの順次読み込みを最適化でき、チェック桁のような意味のない情報を除くこともできます。

ADAREP：レポート

ADAREP ユーティリティは、データベースまたは適切なセーブテープの現在の物理レイアウトおよび論理的な内容に関する情報を提供するステータスレポートを作成します。

このレポートには次の情報が含まれます。

- データベース概要：データベース名、データベース番号、作成日時、ファイルのステータス、および現在のログ番号。
- アソシエータ、データストレージ、およびワーク用スペースの現在のリソース：現在使用されているスペースおよび割り当て済みの未使用スペースの量と位置。
- ファイル情報の概要と詳細：ISN、エクステント、パディングファクタ、使用済み／未使用のアソシエータおよびデータストレージ用スペース、およびファイルオプションのファイル別の概要。またオプションとして、すべての概要情報、MINISN/MAXISN 設定、詳細なスペース情報、作成日および最終使用日時、フィールド定義テーブル（FDT）の内容、および一般または拡張チェックポイントファイル情報のファイル別の詳細。
- チェックポイント情報：一般および拡張チェックポイントファイルの情報。
- 物理構造：デバイスタイプ、VOLSER 番号、ファイル番号（該当する場合）、および使用状況（AC、NI/UI、Data Storage、DSST、および未使用）を含む、アソシエータ／データストレージ RABN の情報。

セーブテープレポートは、セーブテープの内容を調べるのに使用します。

ADAVAL：データベースの整合性チェック

ADAVAL（整合性チェック）ユーティリティは、Adabas データベース内の任意のファイルまたは全ファイルの整合性チェックを行いますが、チェックポイントとセキュリティファイルはチェック対象ではありません。

ADAVAL はデータストレージのレコードのディスクリプタ値とアソシエータに格納された対応する値との整合性をチェックします。したがって、アソシエータとデータストレージの同期がとれていることと、アソシエータ内に値の不足がないことが保証されます。

ADAVALを実行する前に、ADAICKユーティリティを使用して、インバーテッドリストの整合性をチェックする必要があります。

ADAPRI：選択した Adabas ブロックの出力

ADAPRI ユーティリティは、アソシエータ、データストレージ、ワーク、中間、ソート、デュアルまたはマルチコマンドログ（CLOG）、デュアルまたはマルチデータプロテクションログ（PLOG）、リカバリログ（RLOG）、またはデルタセーブイメージ（DSIM）の各データセットに格納されているブロック（またはブロック範囲）の内容を出力します。

6 Adabas のライセンス

次のトピックについて説明します。

ライセンスファイル

Adabas がインストールされているすべてのメインフレームプラットフォームには、有効なライセンスファイルをインストールする必要があります。各製品のライセンスは、インストールテープに州力されています。 必要があれば、電子メールで配布することも可能です。

ライセンスファイルを受け取ったら、このファイルをインストールで使用する前に、メインフレームホストに FTP で送信する必要があります。

ライセンスファイルは XML ドキュメント（エンコーディングは US-ASCII）として提供されます。このドキュメントは、PCのブラウザツールまたはテキストエディタを使用して表示できます。 ドキュメントには、ライセンス情報、デジタル署名、ライセンスキーを表すテキストが含まれています。 製品ライセンスには、環境情報が含まれています。

Important: ライセンスファイルは、メインフレーム上であっても ASCII 形式のままにしてください。 これは変更しないでください。 ライセンスファイルを変更すると、デジタル署名が無効になり、ライセンスのチェックに失敗します。 チェックに失敗すると、製品を起動できなくなります。 チェックに失敗した場合は、Software AG 技術サポートに連絡してください。

ライセンスファイルのインストール

Adabas のインストール中に、ライセンスファイルはロードされ、オペレーティングシステムおよびユーザー要求によっては、Adabas ニュークリアスによってロードされるオブジェクトモジュールに変換される場合があります。 インストール方法はオペレーティングシステムにより異なります。プラットフォームに固有のインストール手順の該当する手順を参照してください。

製品ライセンスのチェック

Adabas ニュークリアスを指定した MODE=MULTI（マルチユーザーモード）で起動するたびに、ライセンスオブジェクトモジュールまたはライセンスファイルのライセンス情報が確認され、ライセンスキーの有効性がチェックされます。 このチェックに失敗すると、製品を起動できません。 ただし、CPU ID が定義されていなかったり CPU の容量を超えたなど特定の失敗については、システムコンソールにメッセージが表示されるだけで、特に通知されることもなく、Adabas ニュークリアスが起動されます。 ライセンスのチェックに失敗した場合は、Software AG 技術サポートにお問い合わせください。

製品ライセンスのチェックに関する FAQ

1. なぜ **Adabas** のメインフレームでのライセンスチェック機能が導入されたのですか。
   有効で十分な製品ライセンスを持つメインフレームマシンでのみ Adabas を実行していただくためです。 これにより、弊社はソフトウェア製品を制御しやすくなります。 他のプラットフォーム（UNIX、Windows）では、以前から製品ライセンスが導入されています。
2. 製品ライセンスにはどのような情報が含まれていますか。
   製品ライセンスは、次の項目を含む XML 形式の US-ASCII テキストのシーケンシャルファイルです。
   - Software AG ヘッダー
   - 顧客情報（名前、ID）
   - 暗号化ライセンスキー
   - ライセンスの有効期限（または無期限）
   - 製品情報（製品コード、バージョン、製品名）
   - 環境情報（オペレーティングシステムの種類、CPU ID、システム名、容量）

   Caution: ライセンスキーファイルを変更すると、デジタル署名が無効になり、ライセンスキーのチェックに失敗します。 チェックに失敗すると、製品を起動できなくなります。 チェックに失敗した場合は、Software AG 技術サポートに連絡してください。

3. **CPU ID** とは何ですか。
   IBM では、CPU ID を "中央演算処理複合ノード記述子のシーケンス番号" として定義しています。これは、ユニークな16進で示されたマシンのシリアル番号です（マシンのモデル番号は含まれません）。

   z/OS

z/OS システムでは、次のコマンドを入力して CPU に関する情報をオペレータコンソールに表示できます。

```
D M=CPU
```

例えば、このコマンドからのコンソール出力に次の行が含まれているとします。

```
EE174I 16.38.50 DISPLAY M 951
PROCESSOR STATUS
ID  CPU                  SERIAL
00  +                     0FA10E2096
01  +                     0FA10E2096
CPC ND = 002096.S07.IBM.83.0000000 7A20K
CPC SI = 2096.V03.IBM.83.000000000007A20K
....
```

この例では、CPU ID は 上記の例で強調表示されているように "7A20K" です。

z/VSE

z/VSE システムでは、次のオペレータコマンドを入力して CPU に関する情報をオペレータコンソールに表示できます。

```
sir
```

例えば、このコマンドからの出力に次の行が含まれているとします。

```
...
AR 0015 PROCESSOR = IBM 2096-V03 83 (7A20K83)  LPAR = DAEX      No. = 0007
....
```

例では CPU ID が強調表示されています。

BS2000

BS2000/OSD システムでは、CPU ID（8 バイトの 16 進数）はマシンのシリアル番号、プロセッサ ID、およびマシンのモデル番号で構成されています。 CPU ID の 2 番目のバイトのプロセッサ ID は、ライセンスチェックでは無視されます。 次の BS2000/OSD コマンドを入力して、CPU に関する情報を表示できます。

```
/SHOW-SYSTEM-INFORMATION INFORMATION=*CPU-ID-LIST
```

例えば、このコマンドからの出力に次の行が含まれているとします。

```
...
%CONFIGURATION              = 7.500- S140-20A      «
%CPU-ID-LIST :  ADR   0     = 1D02301375000000 «
...
```

これらの行には、最も関連する情報が表示されます。最初の値はマシンの種類を"7.500-S140-20A" として識別します。2 番目の値は物理 CPU ID を "1D02301375"（末尾のゼロは省略されます）として識別します。

z/VM

z/VM システムでは、次のコマンドを入力して CPU ID を決定できます。

```
q cpu
```

このコマンドからの出力が次のようになるとします。

```
CPUID = FF07A20K20968000
```

最初のバイトは常に "FF" で、次に CPU-ID（例では強調表示）となります。

4. **Adabas Online System** などの **Adabas** アドオン製品に製品ライセンスはありますか。
   いいえ。 ライセンスファイルは、マルチユーザーモード（ADARUN MODE=MULTI）で稼動している標準 Adabas 製品に対して提供されています。 現時点では、アドオン製品にライセンスを導入する予定はありません。
5. 製品ライセンスファイルはどのようにインストールしたらいいですか。
   製品ライセンスファイルは個別のカスタマインストールテープ、または別に送信される電子メールの添付ファイルで提供されます。 ライセンスファイルは、インストール処理中に RECFM=F または FB および LRECL=80 を指定した Adabas ソースディレクトリ（BS2000 システムの場合、これは SAM ファイルです）にコピーされます。 オペレーティングシステムとユーザー要求によっては、ライセンスファイルはアセンブラ入力ファイルに変換される場合があります。ファイルが変換されると、リンク可能なモジュールにアセンブルされ、Adabas ニュークリアスによってロードされます。 環境ごとにライセンステキストファイルをアセンブラ入力ファイルおよびリンク可能なロードモジュールに変換できるサンプルジョブが提供されています。 次の表に、これらのサンプルジョブを示します。

サンプルジョブ	説明
ADALICA(J)	ライセンスファイルをライブラリにロードしこれをアセンブルする BS/2000 環境用のライセンス変換サンプルジョブ。 Adabas 8 ニュークリアスを起動するには、ライセンスファイルが必要です。 ライセンスファイルはオブジェクト ADALIC(R) または DDLIB/BLSLIB チェーンのライブラリにアセンブルされるか、ライセンスがリンク名 DDLIC から読み込まれます。 どちらにも当てはまらない場合は、リンク名が DDLIC に見つからないため、DMS0A4F エラーを出力してニュークリアスが終了します。
ASMLICAL.X	ライセンスファイルがライブラリにロードされている場合は、z/VSE 環境のライセンス変換サンプルジョブ。
ASMLICAM	z/OS 環境および z/VM 環境のライセンス変換サンプルジョブ。
ASMLICAV.X	ライセンスファイルがデータセットにロードされている場合は、z/VSE 環境のライセンス変換サンプルジョブ。

サポートされている各環境にライセンスファイルをインストールする方法の詳細情報については、このドキュメントセットに記載されている環境別のインストール手順を参照してください。

6. 製品ライセンスファイルはどのようにして読みむことができますか。
   - 製品ライセンスファイルは、XML エディタ（ファイルタイプ .xml）または PC ベースのテキストエディタを使用して、PC で読むことができます。
   - 変換処理中に、変換プログラムによりライセンスファイルのリストが出力されます。
7. 製品ライセンスはいつチェックされますか。
   製品ライセンスは、Adabas ニュークリアスの初期化時に常にチェックされます。 ニュークリアスの初期化を行った後の Adabas セッション中は、ライセンスはチェックされません。
8. 製品ライセンスのどの項目がチェックされますか。
   - 暗号化ライセンスキー
   - ライセンスの有効期限（ある場合）
   - オペレーティングシステム（z/OS、z/VSE、z/VM または BS2000）
   - 製品コード（ADA）
   - 製品バージョン（*v.r* または *v.r.s*）
   - マシン CPU ID
   - マシンの容量（z/OS および z/VM の場合のみ）
9. 製品ライセンスが無効または不十分であったりインストールされていない場合はどうなりますか。
   次の 2 つの処理が行われます。
   a. 次のいずれかの場合に発生した問題を示すエラーメッセージを出力して Adabas セッションが終了します。
      - 製品ライセンスモジュールがないか変更されている場合

- オペレーティングシステム、製品コードまたは製品バージョンが無効である場合
- ライセンスの有効期限が切れている場合

b. セッションは開始されますが、次のいずれかの場合に、システムコンソールに警告メッセージが表示されます。

- ライセンスにマシンの CPU ID が定義されていないか、マシンの容量が製品ライセンスに指定された値より大きい場合 注：これは、Adabas の今後のリリースで変更される予定で、セッションは終了するようになります。
- 製品ライセンスの有効期限が 30 日以内になっている場合

10. 提供されたライセンスファイルが環境に不十分な場合、どのようにして新しい製品ライセンスファイルを取得したらよいのでしょうか。
契約に合った正しい製品ライセンスファイルを取得するには、Software AG 営業部門に連絡してください。 製品ライセンスファイルは、ASCII 形式または変換されたアセンブラ入力フォーマットのいずれかで、電子メールで送信されます。

11. **1** つの **Adabas** ニュークリアスを別のマシンで使用する場合、ライセンスファイルをどのように扱えばよいでしょうか。
1 つのライセンスファイルに、複数の CPU ID を定義できます。 すべての CPU ID がライセンスファイルに定義されていることを確認してください。 1 つのライセンスモジュールしか持つことはできませんが、これをすべての Adabas ニュークリアスで使用できます。

7 Adabas Security

Adabasは、次の機能を提供して、Adabasデータベースファイルへの未認可のアクセスやAdabasデータベースファイルの未許可の更新を防止します。

- データの機密保護機能を提供する Adabas データ暗号化（サイファリング）
- ファイルのレコードへのアクセスを制御する、Adabas マルチクライアントファイル
- Adabas Security および関連するセキュリティユーティリティ ADASCR（ファイル、フィールド、およびフィールド値レベルでアクセス／更新を行うユーザーを限定する Adabas アドオンユーティリティ）
- RACF、CA-ACF2、CA-Top Secret などの System AuthorizationFacility（SAF）に基づく標準セキュリティパッケージを使用してデータベース／ユーティリティ、コマンド、またはファイルの各レベルにおける Adabas のリソースを制御する Adabas アドオン製品、Adabas SAF Security（AAF または ADASAF） ADASAF は現時点では z/OS オペレーティングシステムのみに対応しています。 この製品は、Adabasを中枢セキュリティリポジトリに統合して、そのリポジトリの資産から最大限のメリットを引き出します。

Note: 今後の Adabas のリリースでは、Adabas SAF Security のサポートを、Adabas がサポートするすべてのオペレーティングシステムに拡張することが計画されています。

セキュリティは、パスワードと権限レベルを比較することで実現されます。

このchapterでは、次のトピックについて説明します。

データの暗号化

データの暗号化は、Adabasに欠かすことができない機能です。この機能には、オプションや追加のモジュールは必要ありません。 データは、データベース内に配置される前に暗号化されます。

ユーザーは、レコードの格納時にサイファキーを指定する必要があります。 このキーは格納されませんが、データの要求や復号を行う場合に必要になります。 データは暗号化されるため、システムへの未認可アクセスによってデータが危険にされらされる可能性が最小限に抑えられます。

サイファコードに対する最大限の制御を維持するには、Adabasユーザー出口プログラムを作成して、現在有効なサイファコードをユーザーアプリケーションに入力します。これにより、コードをユーザーに知らせる必要性が無くなります。また、違うサイファコードで暗号化されたファイルを追加することにより、発生する可能性があるファイルの破壊を防止します。

マルチクライアントファイル

マルチクライアントファイルも、オプションや特別なモジュールを必要としない、Adabas に欠かすことができない機能です。

マルチクライアントとして定義された単体の Adabas 物理ファイルは、複数のユーザーまたは複数のユーザーグループが使用するレコードを格納できます。 マルチクライアント機能では、各レコードに内部的なオーナー ID をつけることで、物理ファイルを複数の論理ファイルに分割します。

オーナー ID は、ユーザー ID に対して割り当てられます。 1 つのユーザー ID は、1 つのオーナー ID しか持つことができませんが、同じオーナー ID が複数のユーザーに所属していてもかまいません。 各ユーザーは、ユーザーのオーナー ID に関連付けられたレコードのサブセットだけにアクセスできます。

Note: RACF や CA-Top Secret などの外部セキュリティパッケージがインストールされている場合でも、ユーザーは変わらず Natural ETID またはログオン ID で識別されます。

マルチクライアントファイルへのすべてのデータベース要求は、Adabas ニュークリアスにより処理されます。

Adabas Security および ADASCR

アクセス／更新の制御は、Adabas Security、および Adabas Security 機能を定義および制御する関連セキュリティユーティリティ ADASCR によってのみ実施できます。

Adabas Security は、アクセス／更新と値の 2 のレベルの保護機能を提供します。

アクセス／更新レベルの保護

アクセス／更新レベルの保護により、各ファイルベースで基本レベルのセキュリティが適用されます。 アクセス／更新保護は、一部のファイルにのみ定義することができます。 この保護により、ファイルまたはファイル内のフィールドの使用は、適切なアクセス／更新プロファイル定義、およびこのファイルのユーザーが指定したパスワードを持つユーザーに制限されます。

0～14 の範囲のアクセス／更新権限の値は、各ユーザーに対して定義され、そのユーザーのパスワードに関連付けられます。また、保護された各ファイル（および必要に応じて選択された 1 つ以上のフィールド）には、ユーザーの権限レベルと同じ範囲でアクセス／更新保護のしきい値が設定されています。 指定されたファイル（フィールドも適用可）の保護レベル以上の権限値を持つユーザーのみが、ファイルまたフィールドに関する操作タイプ（アクセスまたは更新）の実行が許可されます。 アクセス／更新権限レベルが 0 の場合は、保護レベル 0 の保護されていな

いファイルまたはフィールド、または保護パスワードが定義されていないファイルまたはフィールドのみにアクセスが制限されます。

値レベルの保護

値レベルの保護は、特定のフィールドでアクセスまたは更新されるタイプおよび範囲の値に適用されます。 制限は、ユーザーのパスワード（値レベルの保護を使用するフィールドを持つファイルはパスワード保護が必要）に従って適用されます。また、特定の値または値の範囲に対して制限したり、許可または拒否条件とすることもできます。

SAF ベースのパッケージへの Adabas インターフェイス

ユーザーまたはユーザーグループが使用できるリソースの厳格な制御を提供するために、z/OS および互換サイトでは Authorization Facility（SAF）が使用されます。 IBM の RACF および Computer Associates の ACF2 または Top Secret などの互換セキュリティパッケージを使用して、システム管理者は次の機能を実行できます。

- ユーザー ID やパスワードなどのユーザー識別情報の維持
- ユーザーが使用できるデータセット、ストレージボリューム、トランザクションおよびレポートを決定するプロファイルの確立

一般的に、セキュリティパッケージにより、システム管理者はユーザーによるシステムリソースへのアクセスに対する認可が可能になります。 次に、セキュリティパッケージはすべてのユーザーおよびユーザーのリソース使用状況を監視して、未認可のアクセスまたは変更が行われないようにします。 未認可ユーザーがシステムや特定のシステムリソースを使用しようとすると、これらの行動は記録されてレポートされます。

単体ユーザーまたはユーザーグループに適用されるユーザープロファイルは、ユーザーに使用を許可するシステムハードウェアおよびソフトウェアリソースを定義します。 リソースプロファイルは、1つ以上のデバイス、ボリューム、プログラムに対してアクセス／更新権限を定義します（特定の機能を実行するために同時に使用する必要があるリソースは、同一プロファイルに同時に定義できます）。

ユーザーがシステムにログオンすると、セキュリティパッケージはユーザーのログオン ID を使用してそのユーザーのプロファイルを特定します。 ユーザーがタスクを実行したり情報へアクセスしようとすると、セキュリティパッケージは毎回リソースプロファイル内の情報を使用して、アクセスを許可または拒否します。 このプロファイルコンセプトを使用すると、セキュリティパッケージにより、ログオン ID 単体による認可の制限を超えて、すべてのシステムリソースにわたる幅広い制御が可能になります。

その結果得られるセキュリティリポジトリ、およびそれを管理するインフラストラクチャは、非常に優れた資産となります。 企業が所有する重要情報の量は常に増加しており、また同時にそのデータを参照するユーザーの数も増加しています。 これらの常に増加し続けるアクセスを制

御するという課題には、柔軟で簡単に実装でき、なによりも企業の資産を守るソリューションが必要です。

Adabas SAF Security（ADASAF）

Adabas SAF Security（ADASAF）は、Adabas のリソースを中枢セキュリティリポジトリに統合することにより、SAF ベースのセキュリティパッケージの適用範囲を拡大します。 ADASAF は次の機能を可能にします。

- すべてのリソースに対応する単一の制御および監査システム
- Adabas データに対する業界標準の保護
- セキュリティリポジトリの投資回収率の最大化

ADASAF の動作は、個別のニュークリアスごとに調整が可能です。そのため、実装に対して優れた柔軟性を持ちます。 ADASAF は次の要素で構成されます。

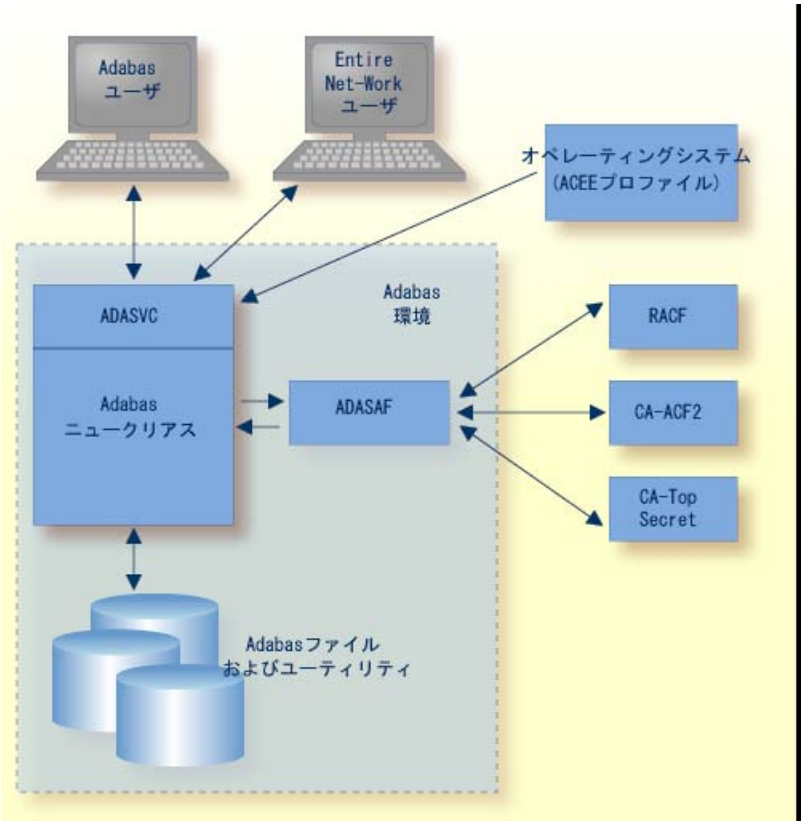
- 保護された各 Adabas アドレススペース内で動作しているサーバー
- Adabas SVC とリンクされたルーター拡張機能
- オンラインの管理および監視システム（デモおよびフル機能版両方の Adabas Online System（AOS）からアクセスできる、Natural で記述されたアプリケーション）
- Adabas のエラー処理機能とのインターフェイスとなるプラグインルーチン PINSAF この機能は、初期化時に自動的にアクティブ化され、問題の診断を支援します。

ADASAF により、次の Adabas のリソースを保護できます。

リソース	保護
データベースニュークリアス	Adabas ニュークリアスを起動できるユーザーを制御します。
Adabas ユーティリティ	ユーティリティまたはデータベース ID 別に、ユーティリティを実行できるユーザーを制御します。例えば、あるユーザーまたはグループは ADAREP を実行できますが、特定のデータベースに対して ADASAV を実行できません。
データベースファイル	データベースファイルにアクセスできるユーザーを制御します。
データベースコマンド	アクセス（READ/FIND）コマンドおよび更新（STORE/UPDATE/DELETE）コマンドを使用できるユーザーを制御します。 パフォーマンスを最適化するため、ADASAF は、RC などのファイルを指定しないコマンドは無視します。
実稼動環境データ	実稼動またはテスト環境で操作できるユーザーを制御します。 このようなクロスレベルのチェックは、例えば、不注意で違うデータベース ID に対してカタログ化されたアプリケーションプログラムによる破壊を防ぐために使用できます。
トランザクションデータ	ET データを格納または取得できるユーザーを制御します。

リソース	保護
Adabas オペレータコマンド	システムコンソールから発行可能な Adabas オペレータコマンドを制御します。
ファイルパスワードおよびサイファコード	セキュリティリポジトリ内に保持されているか、またはユーザー出口により提供されるパスワードおよびサイファコードをダイナミックに適用します。 これにより、アプリケーションがセキュリティデータを管理する必要がなくなります。また、クライアントからデータベースに機密情報を送信する必要がなくなります。
Adabas Basic Services	リソースプロファイルの定義に従って、選択レベル（メイン機能のみ、またはメイン機能と副機能）で Adabas Basic Services を保護します。また、これらのプロファイルへのユーザーのアクセスを制御します。

次の図では、データベースユーザーと Adabas の間のすべてのトラフィックが、Adabas ルーターによって制御されています。 ADASAF がインストールされると、ADASAF ルーターが Adabas ルーターの代わりに機能して、Adabas へのすべてのアクセスを制御します。



システムへのログオンには、一元管理されているセキュリティのログオン ID が使用されます。オペレーティングシステムまたは TP モニタを介して、インストールされた外部セキュリティパッケージは、ログオン ID の認証をチェックします。 リモートワークステーションまたは IBM 以外のプラットフォームからのコールでは、リモートログオンプロシージャが使われ、ログオン ID が ADASAF に渡されます。 ルーターにはセキュリティ出口があり、ユーザーの ACEE からログオン ID を抽出します。

*1 ユーザー1 定義*のアプローチで、完全で柔軟な制御が維持されます。一方、従来のホストベースのセキュリティシステムおよびインフラストラクチャ資産は強化されて破棄されることはありません。

関連するセキュリティオプション

- Adabas Online System セキュリティ
- Natural Security
- SAF リポジトリを使用した Software AG 製品の保護
- Entire Security SAF Gateway
- Entire Net-Work SAF Security （NETSAF）

Adabas Online System セキュリティ

Adabas に付属の Adabas Online System（AOS）のデモバージョンには、Adabas のオンライン機能へのアクセスを制限するセキュリティ機能が含まれています。 AOS Security は前提条件として Natural Security を必要とします。 詳細は、『*Adabas Security* Manual』を参照してください。

Natural Security

Natural Security システムは、Adabas/Natural ユーザーに広範囲なセキュリティ機能を提供します。 このシステムは、AOS Security には必須で、他の Adabas の機能でも使用が推奨されます。詳しくは *Natural Security* のドキュメントを参照してください。

SAF リポジトリを使用した Software AG 製品の保護

Adabas SAF Security または **ADASAF** は、Software AG のセキュリティ製品の 1 つです。次の SAF 中枢セキュリティリポジトリの効果を強化します。

製品	保護対象
Adabas SAF Security	Adabas
Adabas SQL Server SAF Security	Adabas SQL Server（ESQ）
Entire Net-Work SAF Security	Entire Net-Work バージョン 5.6 以降
EntireX Security	EntireX、Entire Broker、Broker Services
Natural SAF Security	Natural

Entire Security SAF Gateway

Entire Security SAF Gateway は、z/OS 上にインストールできます。 SAF 互換のセキュリティシステムを使用すると、次の製品の保護に使用できます。

- メインフレーム対応 Adabas（バージョン 6.2 以前）
- z/OS 環境で動作する Adabas SQL Server
- Entire Broker を使用した Natural RPC
- メインフレーム対応 Natural
- z/OS、UNIX、および Windows 環境で動作する Entire Broker（EntireX 以前）（EntireX Communicator に組み込まれているセキュリティ機能は、EntireX Broker および Broker Services も保護するようになりました）。
- Windows および UNIX アプリケーション用の API 機能

SAF Gateway は、クライアント／サーバー型、ピアツーピア型、および標準のアプリケーションシステムを保護します。 このソフトウェアは、SAF ベースのセキュリティシステムで作成された定義を使用して、クライアント、サーバーおよびピア間のコミュニケーションが保護される特定の点に実装されます。

SAF Gateway は、どのような実装でも最低 2 つのコンポーネントで構成されます。

- SAF Gateway 開始タスクと呼ばれる主要コンポーネントは、それ自身の z/OS アドレススペースで、SAF ベースのセキュリティシステムへのゲートウェイとして動作します。 Software AG ネットワーク内のノードとして、このコンポーネントは SAF ベースの処理を保護対象の製品に設定します。 SAF Gateway 開始タスクは、既存の Adabas データベースとの組み合わせで動作できます。
- 2つ目のコンポーネントは、保護される対象により異なり、前述した各種の分散およびメインフレームのシナリオに該当します。 このコンポーネントは、メインフレーム Natural などのアプリケーションソフトウェア自身の内部に配置することができます。配布されるアプリケーションは、クライアント／サーバーを認証することで保護されます。また、異なるコンピュータ間のコミュニケーションを保護します。

Windows および UNIX アプリケーション用の API 機能は、すでに次のアプリケーションを保護するために使用されています。

- Windows NT および UNIX 版の Web サービス
- Windows 版の Visual Basic アプリケーション
- Windows 版の PowerBuilder アプリケーション
- Windows 版の Delphi アプリケーション
- Windows および UNIX 版の C および C++ アプリケーション

Entire Net-Work SAF Security（NETSAF）

Entire Net-Work SAF Security（NETSAF）は、Entire Net-Work バージョン 5.6 以降を実行している z/OS 環境用の個別のオプション製品です。 この製品を使用すると、Entire Net-Work クライアントは、Adabas、EntireX Communicator、および Entire System Server などの SAF 保護データソース（ターゲット）にアクセスできるようになります。

NETSAF は、個別のリンクごとにアクティブ化できます。 いくつかのノードの 1 つが外部と通信する場合、このノード、および外部リンクに対してのみセキュリティがアクティブになります。

Entire Net-Work を保護するには、SAF リポジトリ内でリソースプロファイルを定義する必要があります。 リソースプロファイルは、ホストターゲットごとに定義されます。 Adabasのリソースプロファイルは、ファイルレベルで定義できます。 コマンドタイプにより、認証の成功に必要なアクセスレベルが決定されます。有効なアクセスレベルは、READ、UPDATE、および CONTROL です。 CONTROL は、例えば AOS コマンドに適用されます。

受信する要求のアクセスポイントの検証は、SAF ベースの中枢セキュリティリポジトリに対して行われます。メインフレームクライアントからのすべてのアクセスは、同一のセキュリティプロファイルに対して検証されます。

セキュリティチェックは、信頼済みユーザーの ID に基づいています。このユーザー ID は中枢セキュリティリポジトリ内に存在する必要があります。 場合によっては、ユーザー ID はコール元のホーム環境で認証されるか、または Entire Net-Work 構成などにより解決されます。 コールが中間ゲートウェイノードを介してルーティングされた場合、ユーザー ID は失われる可能性があります。

8 オプションの製品拡張

このchapterで説明されるアドオン製品は、機能または製品に対する個別の購入契約を交わしたAdabas のお客様が使用できます。

このchapterでは、次のトピックについて説明します。

Adabas Bridges

Adabas Bridge テクノロジにより、DL/I（および IMS/DB）および VSAM アプリケーション開発環境への効果的なアクセスが可能になります。 エミュレーションは、アプリケーションプログラムの修正が必要なく、従来の変換に付随する遅延や労力を避けることができます。

Note: ソリューションは、TOTAL および SESAM でも使用できます。

Adabas Bridge テクノロジには次の特徴があります。

- ユーザーアプリケーションの透過性。既存のアプリケーションプログラムを変更したり、ネイティブの VSAM または DL/I コールを使用するサードパーティ製アプリケーションソフトウェアを変更したりする必要はありません。
- バッチおよびオンライン処理環境、RPG、COBOL、PL/I、FORTRAN、およびアセンブラプログラム言語のサポート
- データおよびアプリケーションの保全性

Adabas Bridge for VSAM

Adabas Bridge for VSAM（AVB）により、VSAM 環境内のデータにアクセスするために作成されたアプリケーションソフトウェアで、Adabas 環境のデータにアクセスできるようになります。 AVB はバッチモードでもオンライン（CICS 配下）でも動作し、z/OS および VSE オペレーティング環境で使用できます。

また、AVB は Adabas ファイルのみ、VSAM ファイルのみ、または Adabas および VSAM ファイル両方（混合環境）の環境で実行可能です。 混合環境で動作可能であるということは、必要性およびリソースに合わせて移行計画を調整できることを意味します。 必要に応じて VSAM から Adabas にファイルを移行できます。1 つのアプリケーション、1 つのファイルを個別に移行することも可能です。

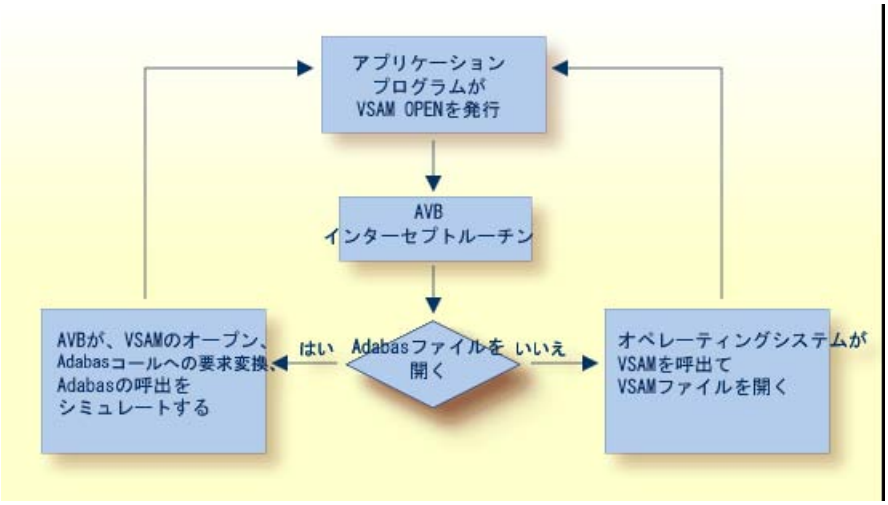
AVB は、透過テーブルを使用して、VSAM ファイルの名前および構造を対応する Adabas ファイルの番号および構造とマッピングします。 VSAM ファイルが Adabas に移行されて AVB 透過テーブルで定義されると、Adabas へのブリッジが可能となります。 VSAM ファイルがブリッジされると、AVB は VSAM ファイルへの各要求を Adabas コールに変換して、VSAM ファイルの代わりに Adabas ファイルがアクセスされます。

AVB がアクティブな場合、各ファイルの OPEN および CLOSE 要求をインターセプトして、Adabas ファイルに対する要求を処理するかどうかを判定する一連のチェックを実行します。

Adabas ファイルに対する要求を処理しない場合、AVB は、参照する VSAM ファイルを開くことができるように、要求をオペレーティングシステムに渡します。

AVB が、ブリッジされたファイルに対する OPEN または CLOSE 要求を検出した場合、この要求を Adabas コマンドに変換して Adabas をコールし、対応する Adabas ファイルを開いたり閉じたりします。 OPEN の実行後は、VSAM ファイルに対するすべての読み込みおよび更新要求は、直接 AVB に渡されます。

AVB は、VSAM コントロールブロックを割り当てて、標準の VSAM ファイル要求から結果が返されたときと同様に、アプリケーションが結果を処理できるように必要な情報を挿入します。 Adabas コールの後で、AVB は標準の VSAM コントロールブロックおよびワークエリアを使用して、結果をアプリケーションに返します。



VSAM に対する Adabas の優位点

以前に VSAM ファイル構造のみが使用されていた環境で Adabas を使用できるようになると、次の利点を得ることができます。

- Adabasの強力なインデックス機能により、アプリケーションを拡張できます。効果的なビューやパスを使用した、データの問い合わせ、取得および操作が可能になります。
- Natural や SQL などのプログラミング言語を使用してアプリケーションを拡張できます。
- アプリケーションプログラムがデータ構造から独立しているため、メンテナンスのコスト削減、プログラマの生産性の向上が図れます。
- 自動再スタート／リカバリにより、ハードウェアまたはソフトウェアの障害時に、データベースの物理的な保全性を確保できます。
- データの圧縮により、必要なオンラインストレージの容量が飛躍的に削減され、また1回の物理 I/O で多くの情報を転送できるようになります。
- パスワード保護と、ファイルおよびフィールドレベル、さらにデータ値に基づくレコードレベルの保護によりセキュリティが向上します。

- ユーザー指定の暗号化処理用キーなどの暗号化オプションを利用できます。

移行後は、使用しているアプリケーションプログラムのデータの見え方は以前と変わりませんが、新しい Adabas ファイルを構成して上記で要約した利点を最大限に活用できます。

Adabas にファイルを認識させるテーブルはアプリケーションの外部にあり、アプリケーションプログラムを再リンクすることなく変更できます。 このような特性があるため、ファイルまたはセキュリティ情報を変更するとき、またはアプリケーションのステータスをテストから実稼動に移行するときに特に便利です。

Adabas Bridge for DL/I（および IMS/DB）

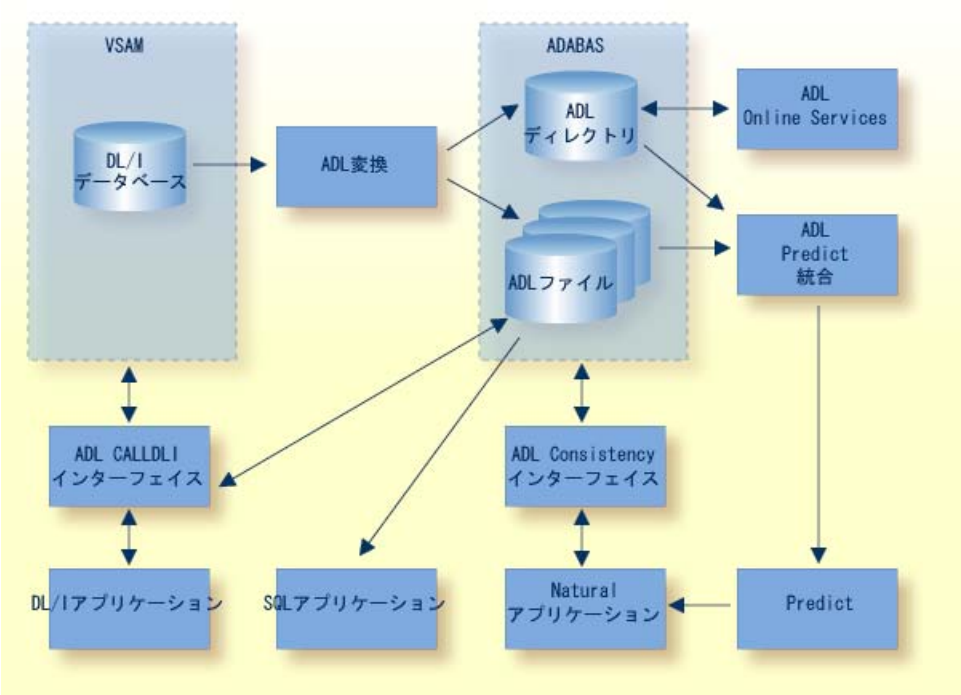
Adabas Bridge for DL/I（ADL）は、DL/I または IMS/DB データベースを Adabas へ移行するためのツールです。 DL/I という用語は、IMS/VS および DL/I DOS/VS を示す一般的な名称として使用されています。 ADL はバッチモードでもオンライン（CICS または IMS/DC 配下）でも動作し、z/OS および VSE オペレーティング環境で使用できます。

DL/I アプリケーションは変更せずに続けて実行できます。Adabas SQL Gateway が使用可能な場合、移行されたデータには Natural および SQL アプリケーションによりアクセスできます。ADL は、Adabas データベースに対して標準の DL/I アプリケーションを実行するために使用できます。

機能の構成

ADL は次の 6 個の主要な機能で構成されます。

- DL/I データベースを ADL ファイルと呼ばれる Adabas ファイルに自動的に変換する変換ユーティリティの集合。ADL ファイルは、ネイティブの Adabas ファイルにはないこれらのファイルの性質が特徴付けられています。
- 変換処理の結果として、DL/I データベース定義（DBD）および関連する Adabas ファイルレイアウトが、ADL ディレクトリと呼ばれる Adabas ファイルに格納されます。ADL ディレクトリには、ADL エラーメッセージやその他の情報も格納されます。
- ADL ディレクトリの内容のレポートなどの、多くのオンラインサービスを提供する Natural で記述されたメニュー型のアプリケーション。
- ADL ディレクトリのデータを使用して ADL ファイルの Predict 定義を生成する、特別な一連の統合プログラム。Predict 定義は、Natural ビューを生成するために使用されます。
- DL/I アプリケーションが、元の DL/I データベースへのアクセスと同じ方法で ADL ファイルにアクセスする（および混在モードで両方のデータベースに同時にアクセスする）ためのコールインターフェイス。 このインターフェイスは、アセンブラ、COBOL、PL/I、RPG、FORTRAN、および DL/I 用の Natural をサポートします。 EXEC DLI インターフェイスを使用するプログラム用に、特別なプリコンパイラが用意されています。
- Natural アプリケーションまたは Adabas ダイレクトコールを使用しているプログラム用の ADL ファイルへのアクセスを可能にする整合インターフェイス。 このインターフェイスは、現行の DL/I アプリケーションにとって重要なデータの階層構造を維持します。



DL/I に対する Adabas の優位点

以前に DL/I ファイル構造のみが使用されていた環境で Adabas を使用できるようになると、次の利点を得ることができます。

- データは、Software AG の第 4 世代言語、Natural で操作できるようになります。
- データは、Adabas によりフィールドレベルで自動的に圧縮されます。
- 削除されたデータレコードは、すぐにストレージから解放されます。これは、削除されたデータレコードにフラグを立てるだけで、そのレコードが使用しているストレージが解放されない DL/I とは対照的です。 Adabas では、解放されたスペースは、新しいレコードによってすぐに再使用が可能です。削除としてマークされたレコードの維持は不要です。また、データベースの再構成の必要性が少なくなります。
- 元のアクセスメソッドに関係なく、すべての変換された DBD は完全な HIDAM 機能を備えています。
- データのアンロードおよび再ロードをせずに、フィールド長を長くすることができます。
- DBD をアンロードおよび再ロードせずに、DBD の最後にセグメントを追加できます。
- オンラインおよびバッチモードでトレース機能を使用できます。
- バッチモードの CALLDLI テストプログラムを使用できます。
- DL/I とは異なり Adabas は CICS リージョン／パーティション内で実行しないため、オンラインのシステムリソースを削減できます。
- VSE 配下で、シンボリックチェックポイント機能を使用できます。
- HD データベースを z/OS 上で使用できます。

Adabas Caching Facility

Adabas Caching Facility は、システムパフォーマンスの向上を支援し、拡張メモリ、データスペース、ハイパースペース、z/OS バージョン 1.2 以降の環境の 64 ビット仮想ストレージ内の Adabas バッファプールを増強して ESA 機能を完全に使用できるようにします。

Adabas Caching Facility は、データベースへのチャネル実行プログラム（EXCP、BS2000 では UPAM SVC）の読み込み回数を削減することにより、Adabas バッファマネージャを増強します。 これにより、貴重な仮想メモリリソースを独占することなく、オペレーティングシステムに存在する機能を使用できるようになります。

Note: データベースの保全性を維持するために、常に書き込み EXCP が発行されます。

Adabas Caching Facility は機能的には Adabas バッファマネージャと同等ですが、次の機能が追加されています。

- 必要性が生じたときにすぐにアクセスできるように、ユーザー指定の RABN（ブロック）をキャッシュまたは確保することが可能です。これは、アクティブなバッファプール内に RABN を保持するだけのアクティビティがない場合でも可能です。 RABN を確保しておくと、Adabas ニュークリアスがこれらの RABN を再読み込みする必要が生じた場合でも、それにかかる I/O レスポンスタイムを削減できます。
- Adabas WORK データセットパート 2 および 3 をキャッシュして、大量の複雑な問い合わせを処理する環境におけるパフォーマンスを向上できます。 Adabas WORK パート 2 および 3 は、複雑な問い合わせの ISN リストの解決および維持に使用される中間ワークエリアとして機能します。 これらの複雑な問い合わせに使用する WORK パート 2 および 3 への EXCP の読み取りと書き込みの回数を減らすことにより、飛躍的に処理時間を短縮して、パフォーマンスを大幅に向上できます。
- ファイルまたはファイルの範囲を指定して、すべての関連する RABN をキャッシュできます。必要に応じて、アソシエータまたはデータストレージブロックのみをキャッシュすることも可能です。 ファイルは、サービスのクラスを割り当てて優先順位をつけることができます。サービスのクラスによって、使用可能最大キャッシュスペースのうち、指定されたファイルが使用できる割合と、いつファイルの RABN ブロックをキャッシュから削除するかが決まります。
- 次に示す内容を変更すると、オペレータコマンドを使用して、変化するデータベース環境に対してダイナミックに対応させることができます。
  - RABN 範囲、ファイル、またはファイル範囲ごとにキャッシュする RABN
  - RABN 範囲、ファイル、またはファイル範囲ごとに有効化または無効化する RABN
  - Adabas Caching Facility により使用されるシステムリソースを確保および解放するタイミング
- 先読みキャッシングオプションを使用して、連続する Adabas コマンド（例えば、論理的読み込み、物理的読み込み、ヒストグラム、非ディスクリプタを使用した検索）を処理する場合、1 つの EXCP が発行され、ディスクデバイスの単一トラックに存在するすべての連続する ASSO

または DATA ブロック、またはその両方が読み込みまれます。 ブロックはキャッシュに保持されて、ニュークリアスが次のブロックを順次要求するときに、すぐに利用できます。 この機能により、3380 の ASSO の物理読み込み I/O の数を最大 1/18 まで削減して、パフォーマンスを向上させることができます。

Adabas RABN は、Adabas バッファプールとキャッシュの両方に同じものが保持されるわけではないので、データベースの保全性は維持されます。 Adabas RABN は、Adabas バッファプールまたはキャッシュエリアのどちらかに存在することはありますが、両方に存在することはありません。すべてのデータベース更新、およびその結果のすべてのバッファフラッシュは、Adabas バッファプールから行われます。 他のキャッシュシステムとは異なり、この非冗長キャッシュメカニズムは、貴重なシステムリソースの使用を節約します。

オンラインのキャッシュメンテナンスアプリケーションである Cache Services を使用するには、Adabas Online System のデモ版またはフル機能版が必要です。 詳細は、*Adabas Caching Facility* のドキュメントを参照してください。

Adabas Cluster Services

Adabas Cluster Services により、マルチニュークリアス、マルチスレッド並列処理が実装され、IBM 並列シスプレックス（systems complex）環境の Adabas を最適化します。 シスプレックスクラスタ内の Adabas ニュークリアスは、Sysplex Timer®（IBM）により同期された複数の z/OS イメージに分散させることができます。 1 つ以上の Adabas ニュークリアスを 1 つの z/OS イメージ内で実行できます。

Adabas Cluster Services は、z/OS イメージ間および各シスプレックスニュークリアスクラスタ内の関連する Adabas ニュークリアス間の相互コミュニケーション能力およびデータ保全性を保証するソフトウェアコンポーネントで構成されています。 最大 32 個のクラスタ化されたニュークリアスでそれぞれ構成されるシスプレックスクラスタの数は、シスプレックス内の複数の z/OS イメージでは制限がありません。

並列処理によりスループットが向上するだけでなく、Adabas Cluster Services では、意図的であれ突発的であれシステムが停止状態になっても、データベースの可用性が高くなります。特定のオペレーティングシステムイメージまたはクラスタニュークリアスにメンテナンスが必要になった場合、またはこれらが予期せずにダウンした場合でも、データベースは引き続き利用可能な状態が維持されます。

1 つ以上のオペレーティングシステムイメージが含まれるクラスタ環境をサポートするため、限られた Software AG Entire Net-Work ライブラリが Adabas Cluster Services の一部として含まれています（「*Entire Net-Work マルチシステム処理ツール*」を参照）。 Entire Net-Work は、Adabas および Adabas Cluster Services のコマンドを、z/OS イメージ間で相互に送信するために使用されます。 このプログラムは、シスプレックスクラスタ内のニュークリアス間のコミュニケーションメカニズムを提供します。 Adabas Cluster Services に対応するために、Entire Net-Work には変更が加えられていません。

クラスタ化されたニュークリアスを監視および制御するために、ADACOMモジュールが使用されます。 各クラスタについて、そのクラスタに参加しているニュークリアスを含む各 z/OS イメージ、またはクラスタデータベースにアクセスするユーザーを含む各 z/OS イメージ内で ADACOM モジュールを実行する必要があります。

Adabas Cluster Services SVC コンポーネントの SVCCLU は、Adabas SVC にあらかじめリンクされ、ローカルおよびリモートのニュークリアスにコマンドをルーティングするために使用されます。 CSA スペースは、ローカルおよびリモートのアクティブなニュークリアス、および現在アクティブなユーザーの情報を維持するために使用されます。

シスプレックスキャッシュ構造は、セッション中に更新された ASSO/DATA ブロックを保持するために使用されます。 このキャッシュ構造により、ニュークリアス、ユーザー、および z/OS イメージが同期され、データの保全性が確保されます。また、複数のニュークリアスの再スタートおよびリカバリが処理されます。

Cluster Services と他の Adabas 製品の連携

Adabas Online System は、シスプレックスクラスタ内のすべてのニュークリアスと通信します。

Adabas Caching Facility は、クラスタ化されたニュークリアスをサポートし、そのクラスタのパフォーマンスを向上させます。

Entire Net-Work を使用する利点

Adabas Cluster Services 環境では、さまざまなネットワークノード上のユーザーは、Entire Net-Work により複数の z/OS イメージにまたがる論理データベースの問い合わせが可能になります。ユーザーは、従来の単一ノードデータベースにアクセスするように、クラスタデータベースにアクセスします。

既存のアプリケーションを変更しなくても、Adabas に要求を送ることができます。要求はシステムにより自動的に処理されます。処理の流れはアプリケーションに対して透過的です。

Entire Net-Work は、オペレーティングシステムインターフェイス用、およびリージョン間コミュニケーション用の Adabas に依存するサービスルーチンを使用して互換性を維持します。Entire Net-Work を実行するためのジョブ制御ステートメントは、Adabas を実行するためのジョブ制御ステートメントと非常に似ています。 例えば、EXEC ステートメントは Entire Net-Work 用の ADARUN プログラムを実行します。これは、Adabas においても同様です。また、Entire Net-Work 用の ADARUN パラメータは、Adabas パラメータのサブセットです。

ターゲットまたはサービスがネットワークとのコミュニケーションを確立していても、またはコミュニケーションが終了していても、ステータス情報はすべてのノードにブロードキャストされるため、一元的な場所でデータベースまたはターゲットのパラメータファイルを維持したり、それらのファイルを参照する必要はありません。

各ノードでは単一の Entire Net-Work タスクのみが許可されているため、すべての必要な情報を 1 か所で維持することによるネットワークトポロジーに対する制御が行われます。 これにより、

ネットワークの運用およびメンテナンスの混乱が回避できます。 ただし、追加のルーターをインストールすることにより、必要に応じて複数の Entire Net-Work タスクを実行することもできます。

各 Entire Net-Work ノードは、要求キューとアタッチドバッファプールのみを 1 つずつ維持するため、バッファストレージの使用を節約できます。 特定のコマンドに不要なすべてのバッファは、転送から除外されます。さらに、レコードバッファおよび ISN バッファの中で、実際にデータが存在する部分のみが、データベース応答でユーザーに返されます。

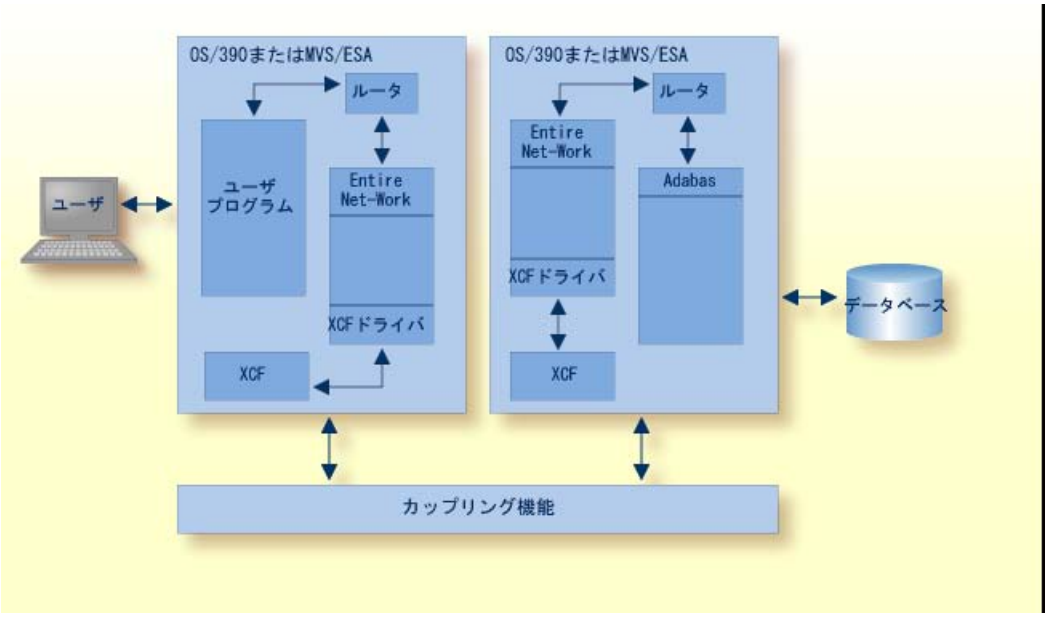
Entire Net-Work のバッファサイズのサポートは、Adabas のバッファサイズサポートに相当するため、Adabas で有効なすべてのサイズのバッファをリモートノードに転送できます。

Entire Net-Work XCF Option

実際のネットワークデータトラフィックは、IBM のシステム間カップリング機能（XCF）へのインターフェイスである Entire Net-Work XCF Option により制御されます。XCF は、システム上の認証済みアプリケーションが同一のシステムまたは他のシステム上のアプリケーションと通信できるようにします。 XCF は、シスプレックス内の 1 つまたは複数の z/OS イメージに存在するグループのメンバ間でデータおよびステータス情報を送信します。

各 Entire Net-Work ノードに Entire Net-Work XCF Option をインストールすると、パフォーマンスが高くなり、シスプレックス内の異なる CPU 上の z/OS イメージ間の透過的なコミュニケーションを可能にします。他のノードへの複数接続がサポートされ、ラインドライバのモジュラー設計により、新しいアクセスメソッドのサポートをシステムに簡単に追加できます。

メンバは、マルチシステムアプリケーションの特定の機能（1 つ以上のモジュール／ルーチン）です。メンバは XCF に定義され、マルチシステムアプリケーションによりグループに割り当てられます。 メンバは、シスプレックス内の z/OS イメージ上に存在し、XCF サービスを使用して、同一グループ内の他のメンバと通信（データの送受信）できます。 XCF ラインドライバを実行している各 Entire Net-Work ノードは、Entire Net-Work 接続用に特別にセットアップされたグループ内の異なるメンバとして認識されます。



Adabas Delta Save

Adabas Delta Save（DSF）は、Adabas データベースの変更部分（デルタ）のみをバックアップすることにより、ADASAV ユーティリティの処理を著しく向上させます。作成される保存出力の量を削減し、保存操作の時間を短縮します。これにより、データベースの可用性が向上します。 より頻繁に保存操作を実行できるため、データベースのリカバリに要する時間も削減できます。

Adabas Delta Save の特徴は次のとおりです。

- データベースの可用性を低下させることなく、頻繁な保存が可能
- 24 時間無休のオペレーションの強化
- データベースをアクティブに保ちながら、完全にオフラインで保存
- リカバリにおける REGENERATE 時間の短縮

Adabas Delta Save は、最後の保存操作以降に変更された（デルタ部分）アソシエータおよびデータストレージブロックのみを保存することにより、上記の目的を達成しました。 この操作の出力先は、*デルタセーブテープ*と呼ばれます。 より少ない量の出力がデルタセーブテープに書き込まれるため、セカンダリ（テープ、カセットなど）ストレージへの接続の必要性が減少します。

Adabas Delta Save には、次の機能が備わっています。

- 変更されたデータベースブロック（RABN）のログの保持

- データベースのオンライン中における、必要に応じた中間デルタセーブテープの作成およびマージ
- 最新のデータベースセーブテープとデルタセーブテープの統合による、最新のフルセーブテープの作成
- 最新のフルセーブテープおよびその後のデルタセーブテープからのデータベースのリストア

Adabas Delta Save は、高い可用性が求められる 1 つ以上の大容量で頻繁に更新されるデータベースを持つ Adabas サイト用に設計されています。 1 日に変更されるデータ量がデータベースの合計サイズと比較して非常に少ない場合、特に有益です。

DSF を使用するには、デモ版またはフル機能版の Adabas Online System が必要です。 詳細については、『*Adabas Delta Save Facility* マニュアル』を参照してください。

Adabas Fastpath

Adabas Fastpath では、レスポンスタイムの迅速化と、リソース使用量の節約が実現されています。これは、参照データの場所をできるだけユーザーに近い場所に設定すること、オーバーヘッドを減らすこと、さらには要求をローカル（同一のリージョンまたはパーティション内）で処理することによって達成しています。

FASTPATH は、アプリケーション処理内で Adabas の問い合わせを解決できるため、データベースとの問い合わせの送受信に必要なオペレーティングシステムのオーバーヘッドを回避できます。 コマンドキュー処理、フォーマットプール処理、バッファプールスキャン、および圧縮解除などのデータベースアクティビティも回避できます。

FASTPATH は、問い合わせサンプラを使用して、次の問い合わせを効果的に識別します。

- 最も多く発行される共通のダイレクトアクセスの問い合わせ。この問い合わせでは、クライアントは ISN、検索値などの検索対象データを識別します。
- 順次アクセスの問い合わせ。この問い合わせでは、クライアントは、シーケンスまたは検索条件に関連した、一連のデータ項目を識別します。

問い合わせサンプラは、シャットダウン時にインタラクティブに、最適化可能な問い合わせの正確なタイプおよび相対的な使用頻度をレポートします。

問い合わせの各タイプに対して、FASTPATH は最も頻繁に使用されるデータを認識および保持しするアルゴリズムを使用して、あまり使用されないデータを破棄または上書きします。 オペレーティングシステム内のすべてのクライアントが使用できる使用量に制限がある場合は、FASTPATH は頻繁に使用されるデータ問い合わせの結果を保持して、その問い合わせが繰り返された場合は、クライアントの処理内で解決できるようにします。 保持された結果は、過去の問い合わせの実績から得られた結果を反映した共通知識ベースとして蓄積されます。 知識ベースは継続的に更新され、保持されている最も使用頻度が低いデータが破棄または上書きされる点で、ダイナミックです。

DBMS に付属の FASTPATH コンポーネントは、使用頻度の高いデータが変更されると、知識ベースに返される結果に反映します。 FASTPATH のデータは常に DBMS のデータと一致しています。

問い合わせが DBMS に渡される前に、FASTPATH の最適化機能は知識ベース内で問い合わせを解決しようとします。 知識ベース内で問い合わせが解決する場合には、問い合わせは処理間コミュニケーションまたは DBMS のアクティビティは発生しないため、処理は高速になります。FASTPATH は、Adabas プリフェッチ先読みロジックをダイナミックに適用することにより、シーケンスを最適化して、DBMS アクティビティを削減します。1 回の DBMS へのアクセスで、最大 256 個のデータ項目を取得できます。

FASTPATH による最適化はクライアント処理で発生しますが、アプリケーションシステムの変更は不要です。異なる最適化プロファイルを、1 日の異なる時刻に自動的に適用することができます。 FASTPATH バッファは、一度起動すると、特に操作することなくアクティブな状態が保持されます。FASTPATH は、DBMS の起動およびシャットダウンに自動的に対処します。

詳細については、*Adabas Fastpath* のドキュメントを参照してください。

Adabas Native SQL

Adabas Native SQL は、Software AG の高水準の記述式データ操作言語（DML）です。Ada、COBOL、FORTRAN、および PL/I で記述されたアプリケーションから Adabas ファイルにアクセスするために使用します。

データベースへのアクセスは、アプリケーションプログラム内に組み込まれた SQL と似た構文で指定します。 Adabas Native SQL プリコンパイラは、次に SQL ステートメントを透過的な Adabas ネイティブコールに変換します。

Software AG の Adabas、Natural、Predict、および Software AG Editor は、Adabas Native SQL の前提条件です。Adabas Native SQL は、Natural ユーザービューの概念と Predict データディクショナリシステムを全面的に活用し、すべての Adabas の機能を利用します。

Natural フィールド指定を使用すると、Adabas Native SQL は、自動的に正しいフィールドセット、フィールド名、フィールドシーケンス、レコード構造、フィールドフィーマットおよび長さを持った Ada、COBOL、FORTRAN、または PL/I データ宣言を作成します。

Adabas Native SQL は、Predict に含まれているファイルおよびレコードレイアウトの情報を使用して、生成された Ada、COBOL、FORTRAN、または PL/I プログラムがデータベースにアクセスするために必要なデータ構造を作成します。 次に、Adabas Native SQL がプログラムを処理する過程で、プログラムがアクセスするファイルやフィールドの名前などのアクティブなクロスリファレンス（Xref）情報を Predict 内に記録します。

これらの機能は、データベースにアクセスするプログラム内に正しくないデータ宣言を書き込むリスクを排除するのに役立ちます。 さらに、これらの機能は、どのプログラムがデータベース

から読み取るか、どのプログラムがデータベースを更新するかを示す総合的な記録をデータディクショナリ内に作成し、DBA に効果的な管理ツールを提供します。

詳細は、*Adabas Native SQL* のドキュメントを参照してください。

Adabas Online System

Note: Adabas には、デモバージョンの Adabas Online System が収録されており、その機能を実行することができます。また、限られた他のサービスにアクセスできます。

Adabas Online System（AOS）は、Adabas に対するオンラインのデータベース管理ツールとして使用します。同様な機能は、バッチ方式のユーティリティのセットを使用して実現できます。

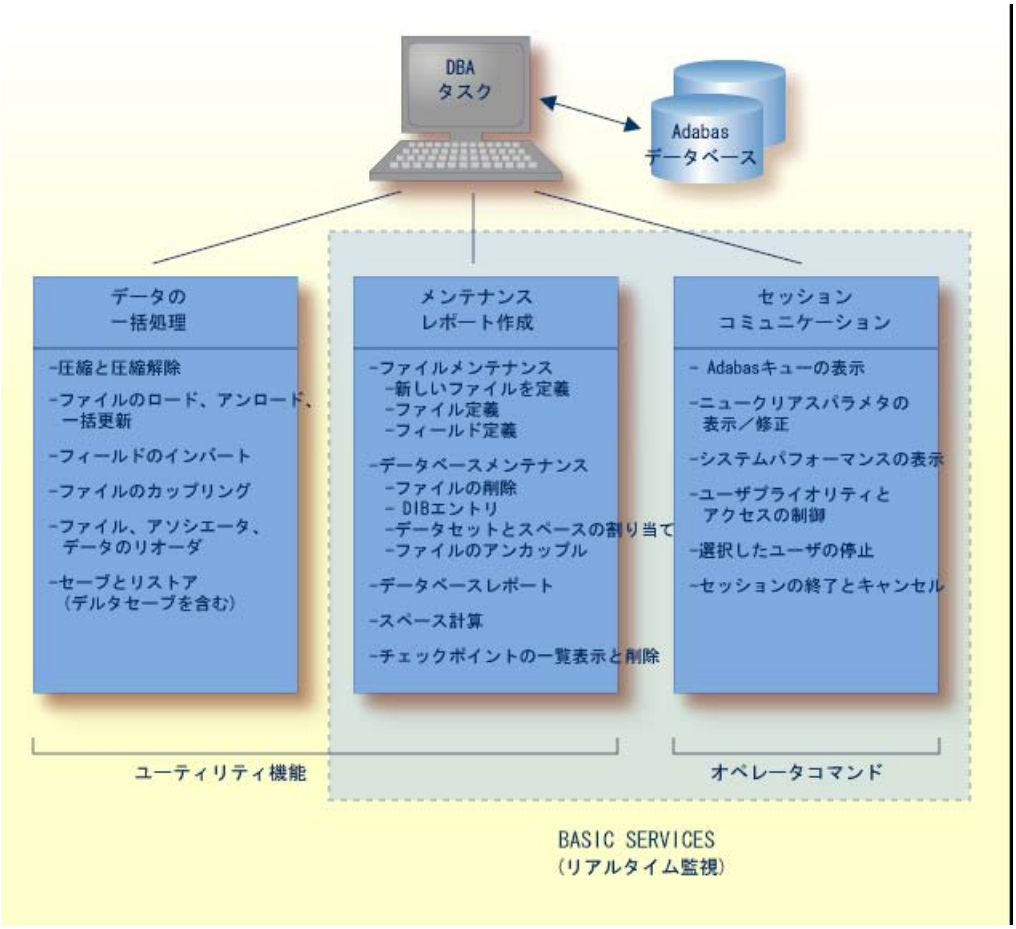
AOS は、オンラインの Adabas データベースの解析および制御に使用する一連のサービスを提供するインタラクティブなメニュー型のシステムです。 これらのサービスにより、データベース管理者（DBA）は次の操作が可能になります。

- Adabas ユーザー統計の表示、特定ユーザーまたは全ユーザーのアクセスおよび操作の監視および制御
- Adabas フィールドおよびファイルの表示および変更、フィールドの追加、ファイルスペースの割り当ておよび割り当て解除、ファイルおよびデータベースレイアウトの変更、フィールドディスクリプタの表示および削除
- ユーティリティユーザーのみへのファイル使用の制限、ファイルアクセスの完全なロックおよびロック解除

ニュークリアスクラスタ環境および SAF Security もサポートされます。 さらに、AOS は、ADARUN パラメータをダイナミックに変更する機能も備えています。

AOS は、Software AG の第 4 世代アプリケーション開発言語である Natural で記述されています。 AOS セキュリティ機能は、Software AG の Natural Security がインストールされている場合のみ使用できます。

AOS には、Adabas オペレータコマンドおよびユーティリティに相当する機能が含まれています。



基本サービスにより、DBA は Adabas セッションがアクティブなときに、Adabas データベースのさまざまな局面をインタラクティブに監視および変更できます。 メニューオプションまたはダイレクトコマンドを使用すると、DBAはリソースのステータスおよびユーザーキューの表示、スペース割り当ての表示および変更、ファイルおよびデータベースパラメータの変更、オンラインでの新規ファイルの定義、および選択したユーザーまたは現在の Adabas セッションの停止が可能です。

AOS は、Adabas からの個別のデータセット／ライブラリとして提供されます。 Adabas から受け取った配布セットの中に最初から、AOS 機能がデモ版として収録されています。 フル機能版にするには、AOS ロードライブラリの内容を Adabas ロードライブラリにコピーするか、AOS ロードライブラリを Adabas ニュークリアス JCL の STEPLIB 内で連結させる必要があります。 さらに、AOS ユーザー（デモ版またはフル機能版）および Predict ユーザーは、Adabas ロードライブラリにあるロードモジュール AOSASM を Natural ニュークリアスにリンクしてください。

詳細は、*Adabas Online System* のドキュメントを参照してください。

Adabas Parallel Services

Adabas Parallel Services（旧バージョンの ADASMP）は、マルチニュークリアス、マルチスレッド並行処理を実装し、単一オペレーティングシステムイメージ上の複数エンジンプロセッサ環境の Adabas を最適化します。

Adabas Parallel Services クラスタ内の最大 31 個の Adabas ニュークリアスが、オペレーティングシステムにより同期している複数のエンジンに分散されます。

Note: 以前の製品である Adabas support for Multiprocessing（ADASMP）は、単一のニュークリアスの更新、およびマルチニュークリアスの読み込み機能を提供してきました。 Adabas Parallel Services から、マルチニュークリアスの更新機能もサポートされるようになりました。

クラスタ内のすべてのニュークリアスは、1つの物理データベースに同時にアクセスします。物理データベースは、1セットのアソシエータとデータストレージのデータセットから成り、単一のデータベース ID 番号（DBID）で識別されます。

ニュークリアスはお互いに通信や協調を行い、ユーザーの業務を処理します。圧縮、圧縮解除、フォーマットバッファの変換、ソート、取得、検索、および更新操作は、すべて並行して行うことができます。

並列処理によりスループットが向上するだけでなく、Adabas Parallel Services では、意図的であれ突発的であれシステムが停止状態になっても、データベースの可用性が高くなります。特定のクラスタニュークリアスにメンテナンスが必要になった場合、またはこれらが予期せずにダウンした場合でも、データベースは引き続き利用可能な状態が維持されます。

同一または異なるルーターまたは SVC 配下の同一のオペレーティングシステムで操作できる Adabas Parallel Services クラスタの数は無制限です。つまり、個別に処理可能なデータベースの数には制限がありませんが、それぞれのデータベースの Adabas Parallel Services クラスタのニュークリアスは最大 31 個までです。

アプリケーションから参照するデータベースターゲットは1つだけなので、インターフェイスの変更は不要です。 アプリケーションは、変わらずに目的のデータベースと通信して、変更せずに Adabas Parallel Services クラスタのニュークリアスと通信します。

詳細は、*Adabas Parallel Services* のドキュメントを参照してください。

Adabas Review

Adabas Review（旧バージョンの Review Database）は、Adabas 環境およびその環境内で実行しているアプリケーションのパフォーマンスの監視を可能にする一連の監視、計算、およびレポートツールを提供します。

Adabas の使用に関して得られた情報は、最小のリソースで最大のパフォーマンスを得るためのアプリケーションプログラムの調整に役立ちます。

Adabas アドレススペースで Adabas Review が実行しているローカルモードに加え、Adabas Review はハブモードを提供します。このモードでは、クライアント／サーバー型のアプローチを使用して、Adabas のパフォーマンス情報を収集します。

- Adabas Review インターフェイス（クライアント）は、各 Adabas ニュークリアス内に存在します。
- Adabas Review ハブ（サーバー）は、それ自身のアドレススペース、パーティションまたはリージョン内に存在します。

Cluster Services 統計の収集がサポートされています。ファイルレベル（CF）のキャッシング統計および Cluster Services のロックが、ユーザー定義の間隔で長期間にわたって監視されます。統計は、Review History ファイルに書き出され、Review のレポート機能を使用して取得または表示できます。

詳細は、*Adabas Review* のマニュアルを参照してください。

ハブサーバー

Adabas Review ハブは、ユーザーに対するデータ収集機能およびレポートインターフェイスです。ハブは、Adabas データベースを監視するためのデータ集積およびレポート機能を担当します。監視するデータには、アプリケーションに関連する使用情報、コマンド、最小コマンドレスポンスタイム（CMDRESP）、I/O のアクティビティ、およびバッファ効率が含まれます。

インタラクティブなレポート機能により、問題を迅速に特定することができます。レポートには、Adabas のアクティビティの詳細および概要データが含まれます。各データベースに特有な情報も入手できます。

実績のある Adabas および Review コンポーネントは、一元的なサーバー（ハブ）として組み合わされることにより、次の利点を得ることができます。

- 単一のハブは、複数の Adabas ニュークリアスおよび Adabas Parallel Services または Adabas Cluster Services により管理される Adabas ニュークリアスクラスタから情報を収集できます。つまり、企業全体に分散した Adabas ニュークリアスをサポートするために必要な Adabas Review ニュークリアスの数を最小限に抑えることができ、リソース要件を最低限に抑えることができるため、パフォーマンスを向上できます。
- 各 Adabas ニュークリアスのアドレススペース、パーティション、またはリージョンを Review サブタスクの監視対象から除外することにより、Adabas メインタスクのパフォーマンスが向上します。 また、将来の Adabas のリリースが Adabas Review の機能に与える影響を最小限に抑えることができます。

ハブの構成要素は次のとおりです。

- ADAREV。受信 Review データコールおよび要求を管理統括する論理モジュールです。
- REVHUB。Adabas Review の環境を確立および維持するモジュールです。
- RAOSAUTO などの Review ニュークリアスおよびサブシステム、自動起動レポートパラメータ生成ルーチン、および履歴データ保存ルーチンである RAOSHIST。

インターフェイスクライアント

Adabas Review インターフェイスは、Review データを構築し、そのデータを Adabas ニュークリアスから Adabas Review ハブに送信します。 監視対象の各 Adabas ニュークリアスと、1 つの Adabas Review インターフェイスが統合されます。

インターフェイスは、既存のリージョン間コミュニケーション処理（ADALNK、ADASVC、および ADAMPM）を利用します。このコミュニケーション処理は、サポートされる複数のプラットフォーム全体にわたって整合性が保たれています。

すべてのサポートされるプラットフォームおよびシステムが正しくネットワーク化されていれば、Adabas Review は、マルチプラットフォーム、マルチオペレーティングシステムの Adabas データベース環境をサポートします。

インターフェイスの構成要素は次のとおりです。

- ADALOG。Adabas コマンドロギングモジュールです。
- RAOSDAEX。Adabas コマンドログレコードに存在しない追加情報を取得する Adabas Review コマンドログ拡張モジュールです。
- ADARVU。RAOSDAEX の環境の調節、および Review データを Adabas Review ハブに送信するための Adabas API 要件を処理します。

インターフェイスコール

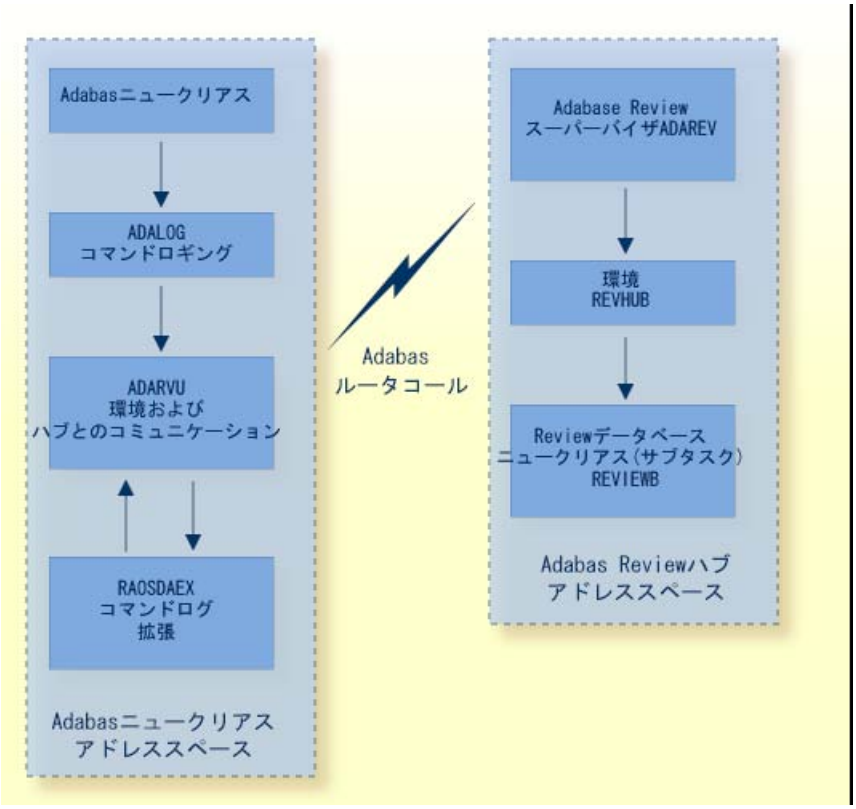
ADARVU モジュールは、Adabas ニュークリアスからのコールを Adabas Review ハブに発行しますが、パフォーマンスを落とさないために、ハブの処理の完了やハブからの応答を待機しません。これは、ADARVU からハブに渡される Review データが常に正しいことを前提にしているからです。

ただし、ADARVU は初期化ステップを実行して、Adabas ニュークリアスによるコマンド処理に先立ってハブが必ずアクティブであるようにします。 ハブがアクティブではない場合、ADARVU は、WTO またはユーザー出口、またはその両方を使用してユーザーに通知します。ユーザー出口が使用されている場合、ハブがアクティブになるまで待機することも可能です。または、初期化を続行して、ハブがアクティブであるときのみ、ハブをコールすることもできます。

コールのハブ側では、クロスメモリポストコールが除外されるため、Adabas クライアントとのアクティブなコミュニケーションのオーバーヘッドが削減され、パフォーマンスが向上します。これにより、ハブは受動的なデータ収集機能の状態を保ちます。

クライアント／サーバー環境の例

次の図は、 クライアント／サーバーアーキテクチャにおける Adabas Review インターフェイスの主なコンポーネント（Adabas ニュークリアスアドレススペース）、および Adabas Review ハブ（Adabas Review ハブアドレススペース）を示しています。



Adabas SQL Gateway

Adabas SQL Gateway は、CONNX のサブセットです。独特のクライアント／サーバー接続プログラムツールセットであり、コンピュータを使って、リアルタイムでインタラクティブに多くのデータベースを操作できるようになります。CONNXデータアクセスエンジンは、データベースへのアクセスを提供するだけではなく、複数のデータベースを企業全体にわたるリレーショナルデータソースとして表現するという点で独特です。 CONNX は、ODBC SQL Level 2 との互換性、強化されたセキュリティ、メタデータ管理、強化されたSQL機能、ビュー、異種データベース間の結合、双方向のデータ変換の各機能を提供し、データの読み込み／書き込みアクセスを可能にします。 このようなテクノロジは、データウェアハウス、データの統合、アプリケーションの統合、電子商取引、データの移行に使用され、また、レポートの目的でも使用されます。このテクノロジは、異なる種類のデータソースを使用している企業内でも利用されています。このような企業では、Web上でのデータの利用を必要としているか、古いアプリケーションでミッションクリティカルな情報を格納しています。

CONNX には次のコンポーネントが含まれます。

- CONNX Data Dictionary（CDD）
- CONNX ODBC Driver

  CONNX OLE RPC Server（CONNX および VSAM には実装されない）

- CONNX Host Data Server（RMS、VSAM［CICS/C++ TCP/IP リスナ／サーバーとして実装］、C-ISAM、Rdb、および DBMS）
- CONNX JDBC Driver（シンクライアント）
- CONNX JDBC Server
- CONNX JDBC Router

多くの種類のデータベースに加え、CONNX は、IBM z/OS、Windows 2000/XP/2003、AIX、Solaris、Linux、HP-UX などのプラットフォーム上で動作する Adabas をサポートします。

Adabas SQL Gateway の詳細については、*Adabas SQL Gateway* のドキュメントを参照してください。

Adabas SQL Server

Software AG は、標準データベース問い合わせ言語の SQL に ANSI/ISO 規格を実装して、Adabas SQL Server を実現しました。 Adabas SQL Server には、Adabas への SQL インターフェイスや、および SQL ステートメントをダイナミックに実行したり、カタログから情報を取得するインタラクティブ機能が装備されています。

このサーバーは、組み込み済みのスタティックおよびダイナミックな SQL、およびインタラクティブな SQL や SQL2 拡張をサポートします。 複雑な Adabas データ構造が、標準 SQL で処理が可能な一連の 2 次元データビューに自動的に正規化されます。

Adabas SQL Server は、次のステートメントを実行して Adabas データにアクセスし、データを操作します。

- Natural アプリケーションプログラムに組み込まれたステートメント。
- 第 3 世代ホスト言語の C、COBOL、または PL/I に組み込まれたステートメント。
- 直接的でインタラクティブなインターフェイスを使用したステートメント。

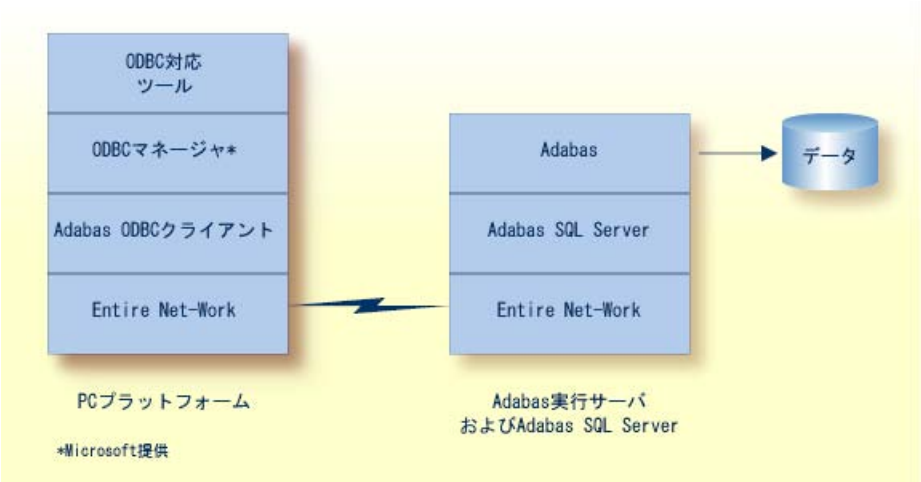
現在、Adabas SQL Server には、C、COBOL、PL/I に組み込まれた SQL ステートメント用のプリコンパイラが装備されています。プリコンパイラは、プログラムソースをスキャンして、SQL ステートメントをホスト言語ステートメントに置き換えます。 Adabas SQL Server のモジュラー設計により、どのホスト言語を選択しても、機能が変わることはありません。

標準の SQL で搭載されていない特定の拡張機能を使用すると Adabas の機能を最大限に利用できるので、アプリケーションプログラムのコンパイル時には、ANSI 互換モード、DB2 互換モード、または Adabas SQL Server モードの 3 つの SQL モードから 1 つを選択する必要があります。

リモートおよびローカルのクライアントは、Entire Net-Work をクライアント／サーバー要求の転送プロトコルとして使用して、サーバーと通信します。 Adabas ODBC Client を使用すると、ODBC ドライバにより、ODBC 互換のデスクトップツールを使用した Adabas SQL Server へのアクセスが可能になります。 同じく ODBC 互換の Entire Access には、SQL 対応の共通のアプ

リケーションプログラミングインターフェイス（API）が装備されています。この API は、Adabas SQL Server のクライアント／サーバーソリューションに相当する、ローカルおよびリモートのデータベースへのアクセスに使用されます。

Adabas SQL Server は、Adabas 自身が利用可能な大部分のハードウェアおよびオペレーティングシステム環境で使用できます。 Adabas SQL Server の主要な機能は、プラットフォームが異なっても同一になります。



詳細は、*Adabas SQL Server* のドキュメントを参照してください。

Adabas Statistics Facility

データベース管理者（DBA）は、定期的にデータベースの状態（ディスクおよびメモリの利用率など）をチェックして、現状の傾向に基づいて、今後のディスクスペース要件をどのように確保するのかなどの長期計画を立てます。

Adabas に対して、DBA は ADAREP（データベースレポート）ユーティリティ、ニュークリアスエンドセッションプロトコル、および Adabas Online System を使用して作成されたアドホック問い合わせを使用して、個別のデータベースおよびファイルのチェックが可能です。 通常、このチェックには長時間の処理を必要とします。

Adabas Statistics Facility（ASF）は、次のプログラムよって構成される自動化環境を提供します。

- アクティブなニュークリアスセッション中にデータベースステータス情報を収集する*格納プログラム*。 このプログラムは通常、数週間または数ヵ月の長期間にわたって定期的な間隔（例えば1日に1回）で実行するようにスケジュールされ、統計的な評価が可能なデータを収集します。 格納プログラムは、ASF オンラインメニューシステム内のコマンドを使用して、DBA によりアドホックベースで開始できます。

- 格納プログラムにより収集された統計情報を解釈し、概要評価レポートを画面またはハードコピープリンタに出力する*評価プログラム*。 レポートは、Entire Connection を使用して PC にダウンロードすることも可能です。

データベース情報は、ニュークリアスセッションの開始時、終了時、および途中に収集できます。 数週間または数ヵ月にわたり蓄積されるニュークリアスの開始および終了データにより、長期的なデータベース規模の推移を知ることができ、将来のデータベース要件を予測することが可能です。 メインメモリやプールの使用率などのニュークリアスのパフォーマンスデータにより、DBA は Adabas ニュークリアスのパラメータを分析および調節できます。

詳細は、*Adabas Statistics Facility* のドキュメントを参照してください。

データ収集プログラム

ASF は、*格納プログラム*と呼ばれるデータ収集プログラムを使用して、アクティブなニュークリアスセッションの開始時、終了時、および途中でデータベースステータス情報を収集します。このプログラムは通常、数週間または数ヵ月の長期間にわたって定期的な間隔（1 日に 1 回など）でバッチプログラムとして実行するようにスケジュールされ、統計的に評価が可能なデータを収集します。 格納プログラムは、ASF 内のコマンドを使用して、DBA によりアドホックベースで開始できます。 オンラインメニューシステム

DBA は、それぞれ監視するデータベースおよびファイルのセットごとに異なる格納プロファイルを定義します。各プロファイルは、格納プログラムの実行時に入力として指定されます。 異なる格納プロファイルを持つ複数の格納プロファイルを同時に実行できます。

Adabas データベースおよびファイルの監視には、条件によりコールされる約 170 個のデータフィールドが使用されます。 データフィールドは、ディスクおよびバッファの使用率、スレッドの使用率、データベース負荷、ADARUN パラメータ、プールの使用率、および特定の Adabas コマンドの使用頻度などの Adabas データベースの状態を表します。 すべてのデータフィールドは、格納プロファイルで指定された各データベースおよびファイルに対して格納されます。

数週間または数ヵ月にわたり蓄積されるニュークリアスの開始および終了データにより、長期的なデータベース規模の推移を知ることができ、将来のデータベース要件を予測することが可能です。 メインメモリの使用率やプール使用率などのニュークリアスパフォーマンスデータは、Adabas ニュークリアスのパラメータを調節する際の参考になります。

データ評価プログラム

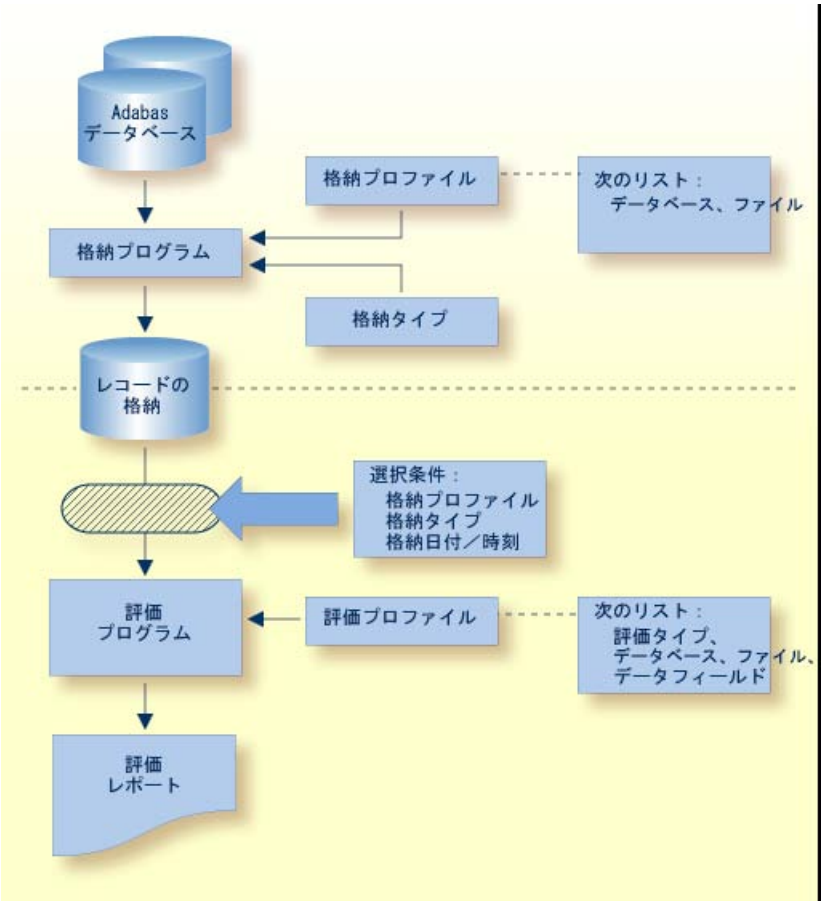
ASF は、*評価プログラム*と呼ばれる一連のプログラムを使用して、格納プログラムにより収集された統計を評価し、評価レポートと呼ばれるサマリレポートを作成します。評価レポートは、オンラインでの表示、出力、および PC へのダウンロードが可能です。

各評価レポートに対して、DBA はオンラインメニューシステムを使用して、次の項目を指定する評価プロファイルを定義します。

- データを評価するデータベースおよびファイル（これらに対するデータは格納プログラムで収集されます）
- 指定されたデータベースおよびファイルに対して、レポートの分析対象となる1つ以上のデータフィールド
- 指定されたデータフィールドの計測単位
- 指定されたデータフィールドのクリティカルレベルを表す、上限および下限値
- レポートの見出しのフォーマットを決定する、レポートタイプ（01～10 の 1 つ）

主なレポートタイプは次のとおりです。

- 全般的な評価：過去および現在のデータベースステータスの分析 生成される統計テーブルは、さまざまなデータベースおよびファイルのステータスの概要を提供します。 出力テーブルの各列で、最大値および最小値が表示されます。また、値の合計や平均値などの統計量も表示されます。
- トレンド評価：日、週、または月単位の間隔で、指定された日付までの予測を示す予測統計テーブルです。
- クリティカルレポート：指定されたデータフィールドが指定されたクリティカル限界値に達したか限界値を超えたデータベースおよびファイルのレポートです。
- クリティカルトレンドレポート：一定の期間内に、指定されたデータフィールドがクリティカル限界値に達するか限界値を超えることが予想されるデータベースおよびファイルのレポートです。予測は現在のトレンドに基づいています。



Adabas Text Retrieval

Adabas Text Retrievalは、フォーマット付き、およびフォーマットなし（テキスト）データの両方に同時にアクセスするアプリケーションの開発を可能にするAdabasの拡張機能です。 Adabas Text Retrievalは、インデックス情報を管理しますが、データの内容は管理しないため、ドキュメントの内容は、Adabas、シーケンシャルファイル、CD-ROM、PCなど、どこにでも格納できます。Adabas Text Retrievalのコールインターフェイスは、Software AGの第4世代アプリケーション開発環境Natural、またはCOBOLやPL/Iなどのすべての第3世代言語内に組み込むことができます。

ドキュメントは、Adabas Text Retrievalにより管理されるフォーマットなしテキスト、またはAdabas自身により管理されるフォーマット付きフィールドとして指定された章で構成されます。 フォーマットなしの章は、パラグラフおよび文章で構成されていて、個別に検索することが可能です。

入力されたテキストの文字列の一部分は、Adabas Text Retrieval文字テーブルで定義された文字と照合することにより、認識またはトークン化されます。この認識またはトークン化には、文字

列の文脈に基づく文字認識アルゴリズムが使用されるか、以前に認識された文字をソートまたは制限する変換テーブルが使用されます。

ドキュメントのインデックスエントリは、フォーマットなしのデータがインバートされるときに作成されます。 テキスト全体をインバートできますが、インバートは、制御されているシソーラス、または禁止ワードリスト上の語句を無視することにより制限することもできます。

検索は、語句、語句の一部（左右または中の省略が可能）、発音、同義語、統合シソーラス関係（下位語／上位語）、近接演算子（語順指定近接、語順不同近接、文内、パラグラフ内）、関係演算子、ブール演算子、以前の問い合わせの参照（絞り込み）、またはソート（昇順または降順）を基準に実行できます。 検索処理は構造には依存しません。つまり、フォーマットなしテキストとフォーマット付きフィールドのどのような組み合わせも検索可能です。 検索結果を強調表示させることができます。

Natural のユーザーは、完全なドキュメント管理サービスを提供する Natural Document Management を使用して Adabas Text Retrieval の機能を拡張することができます。

Adabas Transaction Manager

Adabas Transaction Manager は、Adabas に分散トランザクションのサポートを導入します。トランザクションは、1 つ以上のオペレーティングシステム（Entire Net-Work で接続）上の複数の Adabas データベースに分散させることが可能です。 トランザクションは、DB2、IMS などの Adabas 以外の DBMS にも分散させることができます。 トランザクションが Adabas 以外の DBMS に分散されている場合、Adabas Transaction Manager は、IBM の CICS Syncpoint Manager や IBM の Recoverable Resource Management Services などの、他のトランザクション調整ツールと相互運用するように構成する必要があります。 Adabas Transaction Manager はいつでも実行中のトランザクション、状態が不明なトランザクション、参加中のデータベースなどを明らかにできます。

Adabas Transaction Manager（ATM）は、次の役割を担う Adabas のアドオン製品です。

- グローバルトランザクションに参加している Adabas データベースへの変更を調整する。
- トランザクションマネージャからの2フェーズコミット指示を処理する（最初のリリースではこの機能は利用できません）。このトランザクションマネージャは、IBM の RRMS や CICS Syncpoint Manager などのグローバルトランザクションの調整処理を上位レベルで制御し、単一のオペレーティングシステムイメージで、トランザクションがデータベース（Adabas と Adabas 以外のものを含む）を包括的に処理できるようにします。
- 1 つ以上のシステムイメージ上の Adabas データベースを変更するグローバルトランザクションの調整における主要な役割を担う。 この役割とは、Entire Net-Work 内の複数のシステムイメージのコンポーネント間のコミュニケーションを行うことです。

各 Adabas Transaction Manager インスタンス（オペレーティングシステムイメージごとに1つ）は、特別な Adabas ニュークリアスとして、そのアドレススペース内で実行します。 各 Adabas Transaction Manager は、分散システム内および調整対象データベース内の他の ATM を認識し

ており、連携して動作しています。 Adabas Transaction Manager は、いつでも調整しているグローバルトランザクションのステータスを明らかにできます。

Adabas Transaction Manager は、企業目標に対するな基本的な 2 つのニーズに対応します。

- 主幹のクリティカルなビジネスシステム内で、幅広い商業用途向けの業務に耐える強固な企業目標を実現するニーズ
- 大量のデータを複数のコンピュータおよび組織全体に分散させて、Adabas ユーザーがより均等に管理できるようにするニーズ

Adabas Transaction Manager には、Natural に基いた、Adabas Online System から利用できるオンライン管理システムが含まれます。

詳細については、*Adabas Transaction Manager* のドキュメントを参照してください。

Adabas Vista

Adabas Vista により、ビジネスアプリケーションを再構築することなく、データを個別に管理されるファイルに分割できます。物理データモデルが分割されて、幅広く多くのコンピュータに分散されても、ビジネスアプリケーションは引き続き 1 つの（単純な）Adabas ファイルエンティティを参照します。

データは、複数の Adabas データベースサービスにわたって分割できます。 大容量のファイルが複数のデータベースに分割された場合、処理負荷は、実質的に各コンピュータサービスに分散されます。 コンピュータに複数の CPU エンジンが搭載されている場合、CPU エンジンの並行処理機能をより一層活用できます。

Adabas Vista パーティションは、完全に独立した Adabas ファイルです。

- 複数のパーティションを同一にする必要はありません。 すべてのパーティションが、Adabas Vista で使用されるすべてのビューをサポートしている場合、異なる複数の物理レイアウト（FDT）を持つファイルを操作できます。もちろん、すべてのパーティションに共通な Adabas ソースフィールドは、各 FDT で同一に定義する必要があります。
- パーティションは個別にメンテナンスできます。 パーティションは、必要に応じて個別にサイズの変更、整理、およびリストアが可能です。 すべてのパーティションを、同一の物理的な制約上で操作する必要はありません。 例えば、一部の選択したパーティションを大きくして、他のパーティションを中サイズにすることができます。また、パーティションに応じて ASSO スペースを調節することも可能です。

Adabas Vista がすべてのパーティションに対して単一ファイルイメージを提供するアプリケーションを選択できます。 また、アプリケーションを混合アクセスモードに設定して、プログラムが単一ファイルイメージを使用していても、実際のファイル番号を使用して直接パーティションにアクセスできるようにすることも可能です。

ファイルは通常、場所や日付などの全体的な情報となるキーフィールド（分割条件）に基づいて、分割されます。 ただし、分割条件を使用しないでファイルを分割することも可能です。

アプリケーションは通常、いくつかのキーフィールドに基づいた検索データを使用してファイルにアクセスします。 Adabas Vista は、明示的あるいは黙示的に、アクセスが分割条件に基づいていることを検出し、検索引数を検証して、特定の必要なパーティションにアクセスを限定することにより、処理オーバーヘッドを最小化します。 これは、集中アクセスと呼ばれます。

Adabas Vista のパーティション停止対応機能により、パーティションが使用不能になった場合の対処方法を制御できます。 パーティションの停止を検知するように設定でき、ビジネスアプリケーションは、ユーザーに基づいて停止したパーティションを無視することができます。 例えば、分割条件が場所であり、特定の場所のデータのみがその場所のユーザーにとって重要な場合、ユーザー自身の場所でパーティションを停止すると、ユーザーは操作が中断され、他の場所ではパーティションが停止しても操作を続行できるように、パーティション停止を設定できます。 これにより、データに対する全体的な可用性が大幅に向上し、ビジネスの効率を飛躍的に強化することができます。

Adabas Vista パーティション制限機能により、パーティションを意識せずにデータを利用できます。 この機能を使用して、安全上またはパフォーマンス上の理由から、役割、場所またはその他の業務上の定義に応じて特定のユーザーのみにデータの使用を制限することが可能です。

Adabas Vista の統合機能により、単一のファイルイメージを、以前は無関係だった複数のファイルと強制的に関連付けることができます。 このファイルは、個別のファイルのままですが、統一されたビューをサポートします。

Adabas Vista は、サポートされるバージョンの Adabas とともに、IBM メインフレーム環境（z/OS、z/VM、VSE）で使用できます。

Adabas Vista は、第 3 世代言語プログラムおよび Natural からの Adabas コールをサポートします。 Natural 環境では、オンラインサービスオプションを使用できます。

Adabas Vista は、スタブ（クライアント）部分とサーバー部分で構成されています。 大部分の処理は、次の目的のためにサーバーではなくクライアント処理内で行われます

- CPU 使用率の最小化
- データベースサービス上でのパーティション化に関連するオーバーヘッドの影響の最小化
- できる限り多くの並行処理 CPU エンジンまたは複数のコンピュータへの負荷の分散

詳細については、*Adabas Vista* のドキュメントを参照してください。

Event Replicator for Adabas

Event Replicator for Adabas は、Software AG の製品ファミリで構成されています。 基本的な製品は、Adabas データベース内のデータの変更の監視、および別のアプリケーションへの変更済みデータのレプリケートを行うために使用されます。 詳細は、次のトピックを参照してください。

- イベントレプリケーションの概要
- Event Replicator Target Adapter
- Event Replicator Administration
- Entire Net-Work Administration

Event Replicator for Adabas の詳細については、*Event Replicator for Adabas* のドキュメントを参照してください。

イベントレプリケーションの概要

Software AG の Event Replicator for Adabas は、特定の Adabas ファイルに対してデータの変更状況を監視することができます。 監視対象ファイルの 1 つが変更（削除、格納、または更新）された場合は、Event Replicator は常に変更されたすべてのレコードを抽出して、メッセージングシステム（EntireX Communicator または IBM MQSeries など）を介して 1 つ以上のターゲットアプリケーションに渡します。 レプリケートされる一連のファイルは、1つ以上のサブスクリプション内で定義されます。

Note: このドキュメントで使用されている *MQSeries* という用語は、*WebSphere MQ* という製品と同じものを指します。

Event Replicator は、通常の Adabas の処理に与える影響を最小限に抑えながら、Adabas データの変更をターゲットアプリケーションに渡すことが必要な組織には欠かすことができないツールです。 Event Replicator には、次の基本的な機能が含まれます。

- ほぼリアルタイムのレプリケーション
- 非同期レプリケーション
- 渡されるデータの整合性および順序を保証
- コミットされた更新のみのレプリケーション

Event Replicator を使用すると、1つ以上のサブスクリプション内の定義に従って、Adabas ファイル全体または特定のレコードのセットを目的の場所にレプリケートできます。 データのレプリケーションは非同期で行われるため、Adabas データベースは、レプリケーションの実行中も通常に動作できます。 事前定義されたレプリケート対象ファイルの、コミットされた Adabas の変更点のみがトランザクションレベルでレプリケートされます。

Event Replicator for Adabas の詳細については、*Event Replicator for Adabas* のドキュメントを参照してください。

Event Replicator Target Adapter

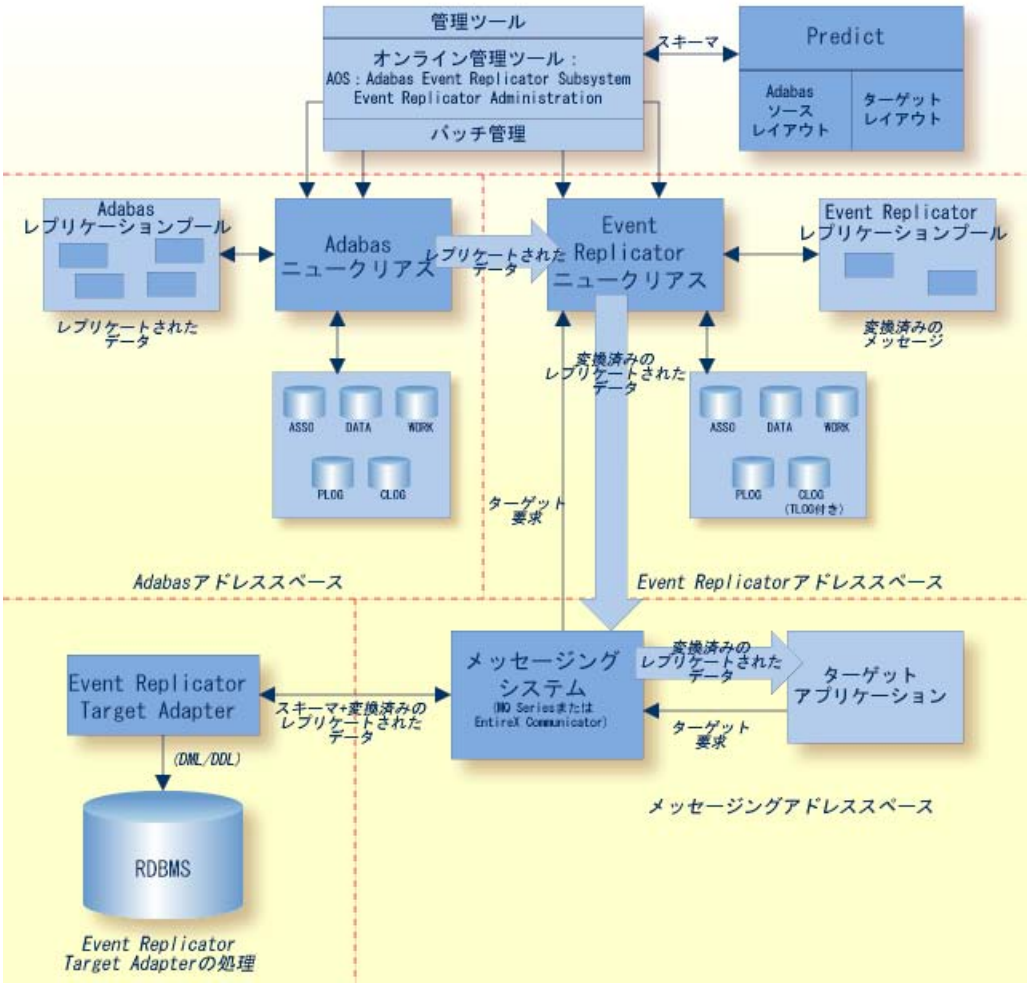
Event Replicator Target Adapter は、レプリケートされたデータを変換し、リレーショナルデータベース（Oracle、DB2、Microsoft SQL Server、MySQL、Sybase など）に適用するために使用します。

Event Replicator Target Adapter は、次のものを使用する必要があります。

- グローバルフォーマットバッファフィールドテーブルが生成されている Event Replicator for Adabas サブスクリプション。フィールドテーブルが生成されていない場合、Event Replicator Target Adapter は動作しません。
- "SAGTARG "クラスタイプが指定されている、少なくとも 1 つの Event Replicator for Adabas 宛先定義。 この宛先は、サブスクリプションにより使用されなければなりません。

この要件を満たして、サブスクリプション定義および1つ以上の関連する宛先定義が作成されていて、これらがアクティブな場合、Event Replicator for Adabas は、レプリケートされたデータをマッピングするスキーマを自動的に作成します。 Event Replicator Target Adapter は、スキーマを使用してレプリケートされたデータを変換し、リレーショナルデータベースに適用します。Event Replicator Target Adapter は、テーブルが存在しない場合は、ダイナミックにテーブルを作成して、挿入、更新および削除処理を使用して、Adabas データが含まれたテーブルをデータベースに追加します。これらの処理は、Adabas ファイルがレプリケートされたときに、ほぼリアルタイムで行われます。

Event Replicator for Adabas および Event Replicator Target Adapter の高レベルな処理を次の図に示します。



適切なEvent Replicatorサブスクリプションおよび宛先定義がアクティブになり、Event Replicator Target Adapter が起動すると、Event Replicator Target Adapter の通常の処理により、サブスクリプションに従ってデータが処理され、レプリケートされたデータが変換されてリレーショナルデータベースに適用されます。 また、手動で次の要求を Event Replicator Target Adapter に送ることができます。

- リレーショナルデータベーステーブルを追加するための初期状態要求の開始
- リレーショナルデータベーステーブル内のデータのクリア
- リレーショナルデータベーステーブルおよびそのデータの削除

Event Replicator Target Adapter の詳細については、*Event Replicator for Adabas* のドキュメントを参照してください。

Event Replicator Administration

Event Replicator Administration は、Web ベースのグラフィカルユーザーインターフェイス（GUI）です。Event Replicatorの管理タスクを実行する際に使用します。 このGUIは、Software AG の製品を集中的に管理する System Management Hub（SMH）と通信する標準の Web ブラウザを使用します。 SMH はユーザーセッションを処理して、ユーザーとブラウザの相互作用を、Event Replicator 管理タスクを実装した Event Replicator 用に作成された特定のエージェントへの要求に変換します。 次に、SMH はエージェントの応答を HTML ページとしてブラウザに転送します。

Note: System Management Hub を実行しているブラウザのウィンドウを閉じるか、別の URL に移動すると、Event Replicator Administration とのセッションが終了します。

System Management Hub の Event Replicator Administration エリアには、次の 2 種類の管理が含まれています。

- Adabas データベース管理
- Event Replicator データベース管理

これらの管理エリアを使用して、Event Replicator for Adabas が使用している Adabas および Event Replicator データベースを管理できます。

エリア	説明
Adabas データベース	このエリアを使用して、Event Replicator for Adabas が使用する Adabas データベースの登録および登録解除を行います。
Replicator	このエリアを使用して Event Replicator for Adabas が使用する、Event Replicator データベースの登録および登録解除を行います。また、各 Event Replicator データベースに関連付けられた Replicator システムファイル内のレプリケーション定義のメンテナンスを行います。

Event Replicator Administration の詳細については、*Event Replicator for Adabas* のドキュメントを参照してください。

Entire Net-Work Administration

Event Replicator Administration を使用してレプリケーション定義を保守する場合、いくつかの Software AG のミドルウェアコンポーネントのインストールが必要です。 Entire Net-Work Client または Entire Net-Work のバージョン 7.2 以降のクライアント側へのインストールが推奨されます。 Event Replicator Administration は、Entire Net-Work 5.9（このバージョンの Entire Net-Work TCP/IP Option のシンプル接続ラインドライバが含まれます）の機能限定版である Entire Net-Work Administration、および Entire Net-Work Client に付属しています。

上記の Entire Net-Work 製品を使用する場合、Event Replicator Administration を使用して保守を行う Event Replicator Server および Adabas データベースは、UES が有効でなければなりません。 Event Replicator Administration と Event Replicator Server および Adabas データベース間

のコミュニケーションを確立する方法として、Windows 版の Entire Net-Work 2.6 および Entire Net-Work を使用する方法があります。 5.8.1.

Event Replicator Administration および Event Replicator 要件の詳細については、*Event Replicator for Adabas* のドキュメントを参照してください。

Entire Net-Work マルチシステム処理ツール

Software AG のマルチシステム処理ツールである Entire Net-Work により、ネットワーク全体にわたる範囲において Adabas およびその他のサービスタスクとの通信が可能になり、分散処理のメリットを受けられます。 この柔軟性により、次の操作が可能になります。

- ネットワークが構成されたシステム上で、データベースの場所に左右されずに Adabas データベースアプリケーションを実行する
- さまざまなネットワークノード上に存在するコンポーネントを使用して、分散された Adabas データベースを操作する
- 特定のタイプのタスクの実行に最も適切なネットワークノード上で、他のネットワークシステムからこれらのサービスへのアクセスを制限しないで、そのタスクを実行する
- Entire System Server（旧バージョンの Natural Process）にアクセスして、リモートシステムでオペレーティングシステム依存の機能を実行する
- クライアント／サーバーアプリケーションを実装するために、Entire Broker にアクセスする

Mainframe Entire Net-Work は、BS2000、z/OS、VSE、z/VM および富士通の MSP をサポートします。 この Entire Net-Work は、オペレーティングシステムおよびハードウェアアーキテクチャが異なっている可能性がある、異なる物理または仮想マシン上で実行しているクライアントとサーバープログラム間の透過的な接続を提供します。

Entire Net-Work は、ミッドレンジプラットフォームの OpenVMS、UNIX および AS/400、およびワークステーションプラットフォームの OS/2、Windows、および Windows NT でも使用できます。

最も低いレベルでは、Entire Net-Work はリモートシステム上のターゲットまたサーバーに向けられたメッセージを受け取り、適切な送信先に届けます。 これらの要求に対する応答は、要求の発行元のクライアントアプリケーションに返されます。その際、アプリケーションへの変更は不要です。

操作方法、およびサーバーの場所や操作上の特徴は、ユーザーおよびクライアントアプリケーションに対して完全に透過的です。 サーバーおよびアプリケーションは、Entire Net-Work がインストールされ、通信可能なシステム内の任意のノードに配置できます。 ネットワークターゲットおよびサーバーのユーザービューは、ユーザーのローカルノード上にある場合と同様に表示されます。 遠隔処理の遅延により、一部のトランザクションのタイミングが変わる可能性がある点に注意してください。

Entire Net-Work は、アプリケーションをプラットフォーム特有の構文要件から解放し、ユーザーが潜在的なネットワークの特性に制限されることを防ぎます。 また、（回線ダウン時）ダイナミックな再構築および再ルーティング機能を備え、ネットワークパスの最適化に寄与し、またネットワークレベルの統計情報を生成します。

Entire Net-Work は、クライアント／サーバー機能を必要とする関連ホストまたはワークステーションシステムにインストールされています。 1 つのシステムは、Entire Net-Work コントロールモジュール、コントロールモジュールサービスルーチン、および必要なすべてのラインドライバで構成されています。 Entire Net-Work がインストールされたシステムは、ネットワークのノードとなります。 各ノードから近接する他ノードへのリンクは、名前およびドライバのタイプによって決まります。

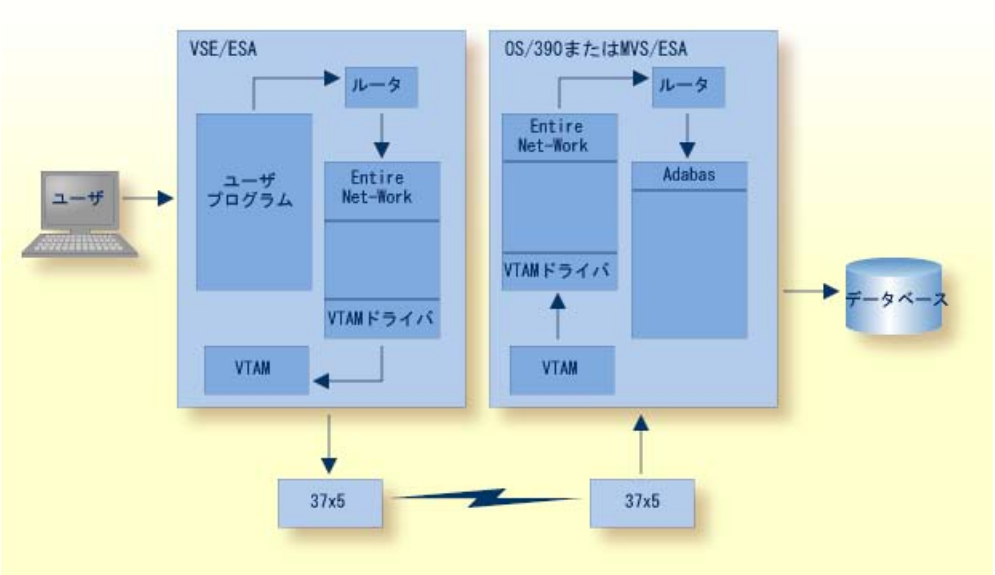
各Entire Net-Work ノードには、受信要求用の要求キューが存在します。 このキューは、Adabas で使用するコマンドキューと類似しています。このキューにより、ノードはローカルで実行中のユーザー／クライアントプログラムからのAdabas コールを受信できるようになります。Adabas コールはEntire Net-Work によりキューが解除され、要求されたサービスが存在するノードに転送されます。

Entire Net-Work の各ローカルノードは、すべてのアクティブなネットワークサービスの情報を常に得ています。したがって、ユーザーの要求を実行できるか、または拒否すべきかを判断できます。 要求が実行可能な場合はメッセージが転送されます。実行が不可能な場合は、Entire Net-Work はコール元のユーザーに直ちに通知します。これは、Adabas ルーターがローカルのデータベース要求に対して行う処理と同等です。

実際のネットワークデータトラフィックは、Entire Net-Work ラインドライバにより制御されます。このラインドライバは、VTAM、IUCV、DCAM、XCF、および TCP/IP などのサポートされるコミュニケーションアクセスメソッドへのインターフェイス、またはチャネル間アダプタ（CTCA）などのハードウェアデバイスへの直接的なインターフェイスとなります。 各 Entire Net-Work ノードには、そのノードでアクティブなアクセスメソッドに必要なラインドライバのみが含まれます。さらに、各ラインドライバは、他のノードへの複数の接続をサポートします。このラインドライバのモジュラー設計により、新しいアクセスメソッドをシステムへ簡単に追加できます。

Entire Net-Work XTI インターフェイスにより、ユーザーは独自のクライアント／サーバーアプリケーションの作成が可能になります。アプリケーションは、主にAdabas の構造の影響を受けない C で記述します。 XTI は、移植性に優れたアプリケーションを作成するための、国際的に認められた仕様です。 理論上では、XTI仕様に準拠して作成されたアプリケーションは、XTI実装をサポートしている他のどのようなプラットフォームへも簡単に移植が可能です。

Entire Net-Work XTI の実装は、同一のコンピュータ内および異なるコンピュータで実行しているプログラム間のコミュニケーションをサポートします。 Entire Net-Work は、アプリケーションプログラマからは、転送プロバイダとして見えます。



詳細については、*Entire Net-Work* のドキュメントを参照してください。

Entire Transaction Propagator

Entire Transaction Propagator（ETP）により、Adabas ユーザーは単一データベースまたは分散ネットワーク内のデータベースファイルを複製、またはレプリケートできるようになります。コピーは、ネットワーク全体にわたって分散させることができ、各ユーザーの場所から迅速で効率的にファイルにアクセスできるようになります。

分散データベースの概念は、操作の効率および柔軟性を提供しながら、同時にほぼ無制限のデータ容量を実現できます。 このようなネットワーク化されたデータベース構造は、特定の部署が必要とするデータベースデータの一部分をローカルシステム上に配置し、同時に共通データベースリソースとして企業全体で利用できることを意味します。

分散データベースに特性は、データを最も必要とする場所に、そのデータの複製を配置できる点にあります。このような特性があるために、コピーされた複数のデータファイルを、データベースネットワーク全体に配置できますが、ユーザーからはこのコピーは論理的に1つのファイルとして見えます。

通常は、レプリケートされたファイルは、変更が発生するごとに、すべてのファイルコピーにおけるデータの保全性を確保するための複雑な制御処理を必要とします。書き込みトランザクションより読み込みトランザクションの比率が高い分散システムでは、このような厳密な制御を必要としない場合があります。 ETP では、ファイルのレプリケーションに厳密性の低い制御プロセ

スが採用されていますが、実質的には、ファイルのレプリケーションによる長所をすべてを実現しています。 マスタ／レプリケートの制御システムを使用して、ETP はすべてのレプリケートコピーをユーザー指定の間隔でマスターコピーと再同期します。

Natural アプリケーション開発環境

Software AG の先進の第 4 世代アプリケーション開発環境である Natural、および分析や設計、コード生成、およびリポジトリ機能を有する Software AG の主要なアプリケーション開発用製品ファミリを使用することで、Adabas の利用レベルが向上します。

Natural から Adabas への直接アクセス、または Entire Access コールを使用した Adabas SQL Server 経由のアクセスが可能です。

FDT では Adabas ファイル内の物理レコードを定義するのに対し、Natural プログラムでは、ファイルにアクセスするときの物理ファイルの論理ビューを定義し使用します。 ビューは、データ定義モジュールとユーザービューの 2 つのレベルを持つことができます。

データ定義モジュール（DDM）は、Adabas FDT と非常に似ている Natural モジュールです。 DDMは、一連のフィールドおよびその属性（タイプ、フォーマット、長さなど）で構成され、レポートフォーマット、編集マスクなどの追加的な仕様を含めることができます。

DDMには、FDTに定義したフィールドすべて、またはそのフィールドのサブセットを含めることができます。 各 Adabas ファイルには、少なくとも 1 つの DDM が必要です。 例えば、Employees という Adabas ファイルには、Employees という DDM が必要です。 Natural ステートメントの READ EMPLOYEES BY NAME は、実際には物理ファイルではなく DDM です。 DDM は、Natural ステートメントを Adabas ファイルにリンクします。

1 つの Adabas ファイルに、複数の DDM を定義できます。 複数の DDM を定義することで、ファイル内のフィールドへのアクセスを制限できます。 例えば、役職者が使用するプログラム用のDDMには、制限情報を持つフィールドを含めて、一般ユーザーが使用するプログラム用のDDMにはこれらのフィールドを含めないようにできます。 ワークステーショングループでは、データベース管理者はグループ用の標準 DDM セットを定義できます。

既存の Natural DDM から、新しい Adabas FDT を作成できます。 逆に、FDT が作成または変更されると、Adabas は DDM を自動的に生成または上書きすることができます。

Note: Adabas ファイルからフィールドを削除する場合は、そのフィールドを参照する Natural プログラムから該当フィールドを削除する必要があります。

Natural のユーザービューには通常、DDM 内のフィールドのサブセットが含まれます。 ユーザービューは、Data Area Editor 内、またはプログラムまたはルーチン内で定義できます。 ユーザービューがDDMを参照するとき、フォーマットと長さはDDMで定義されてるため、新たに定義する必要はありません。 DDM またユーザービューでは、FDT 内のフィールドとは異なる順序でフィールドを定義できます。

Adabas のアクセスはフィールドを基本にしています。Natural プログラムは必要なフィールドのみにアクセスしたり、必要なフィールドのみを取得します。 Natural ステートメントからは、自動的に Adabas 検索や取得操作を呼び出すことができます。

Adabas では、さまざまな順次アクセスメソッドとランダムアクセスメソッドがサポートされます。 Natural ステートメントごとに、別々の Adabas アクセスパスおよびコンポーネントを使用します。最も効率的な方法は、必要とする情報の種類および取得が必要なレコード数によって異なります。

詳細については、*Natural* のドキュメントを参照してください。

Predict データディクショナリシステム

Predict（Adabas データディクショナリシステム）により、データディクショナリの設定と更新ができます。 データディクショナリは、標準の Adabas ファイル内に格納されるため、Natural から直接アクセスできます。

データディクショナリには、データの定義、構造、使用法についての情報が含まれます。 実際のデータ自身ではなく、データまたはメタデータに関する情報が格納されています。ディクショナリは、データの定義をすべて含む情報の格納場所であり、データ属性、特性、データ源、使用法および他のデータとの相互関係を含んでいます。 ディクショナリは、データをより有用にするための情報を収集します。

データディクショナリにより、DBA は組織のデータリソースのより高度な管理や制御が可能になります。 より高度なデータディクショナリのユーザーは、プロジェクト管理やシステム設計においてこれが有効なツールであることを認めています。

オンラインモードでもバッチモードでも、ディクショナリに対してデータベースに関する情報を格納できます。 Adabas ディクショナリ内のデータの記述には、ファイル、各ファイルに定義されたフィールド、ファイル間の関係に関する情報が含まれます。 使用法に関する記述には、そのデータを使用するシステム、プログラム、モジュールおよびレポートに関する情報の他、データの所有者と使用するユーザーに関する情報が含まれます。 ディクショナリエントリは、次のような情報から成ります。

- ネットワーク構造
- Adabas データベース
- ファイル、フィールド、および相互の関係
- 所有者と使用者
- システム、プログラム、モジュールおよびレポート
- フィールド整合性チェック（処理ルール）

標準のデータディクショナリレポートは、次の情報を出力できます。

- データディクショナリの全内容
- フィールド情報、ファイル情報、相互の関連情報
- ファイルごとのフィールド情報

詳細については、*Predict* のドキュメントを参照してください。

索引

シンボル

A

B

C

D

E

F

I

N

O

P

R

S

T

U

V

W

X

あ

い

え

お

か

き

く

け

こ

さ

し

す

せ

む

ゆ

ら

り

る

れ

ろ

わ