

Oracle8iをLinuxで使うためのテクニック

文：原 昭浩

Text : Akihiro Hara

LinuxでのOracleの使用

ここでは、ビジネスシステムでLinuxを使用する必然性、そのLinux上でOracleを使う必要性について述べ、さらにOracleアーキテクチャ、およびバックアップやチューニング方法について説明する。

さらに、RDBの特徴および設計についても簡単に触れたいと思う。

LinuxおよびOracleが採用される条件

Oracleが使われるような分野は、ビジネスシステムであることがほとんどである。ビジネスシステムにおける見積もりと設計について少し述べてみよう。

Linuxは、ご存じの方も多いように、Linus氏を中心にAlan Cox氏を始め、世界中の開発者コミュニティによってカーネルが作成され、バージョンアップされてきた。また、同様あるいは近い開発形態によって、X Window System、Sendmail、BIND、Apache、Samba、そしてGNOMEやKDEといったデスクトップ環境が作られ、さまざまなプロダクトが統合されて、Linuxディストリビューションとして今日流通している。

また、Linuxコミュニティは早くからビジネス用途でのLinuxの使用を推奨し、中心として使われてきたGPLのライセンス体系も、ソースの公開、改造、配布の自由などの基本的な制約を満たせば、その商用利用を禁じていなかった。

一昨年、IBMのApacheサポートを皮切りに、多くの大手企業がLinux上でのソフトウェアの開発提供やサポートの開始を表明した。登場が待たれていたLinux向

け商用RDBMSも、Oracle、Informix、Sybase、DB2、InterBaseなどの主要DBMSが対応、オープンライセンスで提供されるPostgreSQLやMySQLなども考慮すると、今や最も多種多様なDBMSが動作するプラットフォームとなっている。

システムを作る際には、必ず達成すべき目的があり、特にビジネスシステムでは投入可能な予算や達成すべき納期が決められている。また、拡張性や運用などのさまざまな制約条件がある。このようなビジネスシステムにおいてLinuxを採用するメリット、あるいは採用すべき理由は、次のような点が挙げられるだろう。

- ・OSのライセンスに制約されず、ハードウェア機器の構成やビジネスモデルの構築で柔軟性が高い
- ・汎用的なPCアーキテクチャで動作するので、ハードウェアの保守、拡張が安価で容易である
- ・単一サーバシステムとしては信頼性に定評がある
- ・サーバアプリケーションとしては、Web、メール、ファイルサーバ機能、JavaのJDK等々が提供され、必要なものがほぼ揃う
- ・Linux上のアプリケーションや運用手法は、小さな改造でSolarisベースの大型サーバにも移行可能

一方、現状では以下のような課題もあり、これらを加味したうえで採用を判断する必要がある。

- ・Linux、UNIXを運用できる人材は、

Windows系に比べれば少ない

- ・マルチクラスタ構成は実績も少なく、まだこれからの部分がある
- ・周辺機器のドライバなどは、用意されていない場合もありえるので事前の確認などの注意が必要
- ・GUIに関しては、まだ発展途上
- ・上記と同様であるが、GUIベースの開発ツール/デバッグツールもまだ少ない
- ・信頼性の高いファイルシステムは、まだ段階

これらの問題点も、将来的に解消される方向で進んでいるものもあるので、実際の導入検討時には、事前に情報を集めて調べる必要がある。

さて、Linux上で動くOracleを採用するとしたら、その理由は何だろうか？

- ・耐障害性の優れた機能および実績を持つDBエンジンを採用したい
- ・チューニング余地が大きい
- ・Oracleそのものの操作、機能はOSによらず同一なので、Oracle上でのPL/SQLアプリケーションが無改造で移行できる。運用もほぼ同一
- ・他のOracleとのDBリンクによる連携や、レプリケーションが容易
- ・クライアント側のアプリケーションや接続手順も、他のOracleと同様
- ・サードパーティ製のチューニングツールやバックアップツールの使用が可能

などが挙げられるだろう。一方で以下のような問題もある。

- ・メモリは、256Mバイト以上は最低でも必要と覚悟すべきである。

高可用性の要求や、将来的に商用UNIXへのアップグレードや連係を予定していれば、Oracle for Linuxは、検討すべき対象となる。

なお、2001年1月1日より、オラクルの価格体系が変更され、従来のWeb上での無制限ライセンスは、CPU単位の価格になり導入しやすくなった。

また、WebやJavaのプラットフォームとしてのLinuxには大きな可能性があり、PHPやRubyなどの新しいスクリプティング環境もLinuxで動いている。もともとPC畑で育ってきたという経歴からか、細かな操作の面で他のUNIX処理系より操作しやすい面も多分にある。これらもLinuxを採用する大きな動機のひとつだろう。

特に今までWindows NT & Oracleでやってきたソフトウェアエンジニアの方には、すぐに実務を行う機会がなくとも、Linux & Oracleの環境を触ってみることをお勧めする。インストールガイドやリリースノートは、日本オラクルのOTN-Japan (<http://otn.oracle.co.jp/>)などで入手できる。NTでOracleを触ってきた方は、とまどうこともあるだろうが、Linuxの入門書

を右手にトライすれば、比較的容易に慣れると思う。

Oracleのアーキテクチャについて

ここで、Oracleのアーキテクチャについて説明する(図1)。

Oracleは、表1のようなプロセスとファイルから構成されている。Windows NT上のOracleでは、これらのプロセスはすべて「サービス」に含まれ、表面的には見えないうが、Linux (UNIX) では、すべてプロセスとして見える(画面1)。

Oracleでは、実際のデータはすべて表領域に保存され、表領域は単一または複数の物理ファイルで構成されている。通常使われる表領域は以下のようなものである。

```
# ps -ef | grep oracle
oracle 454      1  0 Nov12 ?        00:00:00 ora_pmon_orcl
oracle 456      1  0 Nov12 ?        00:00:00 ora_dbw0_orcl
oracle 458      1  0 Nov12 ?        00:00:00 ora_lgwr_orcl
oracle 460      1  0 Nov12 ?        00:00:00 ora_ckpt_orcl
oracle 462      1  0 Nov12 ?        00:00:00 ora_smon_orcl
oracle 464      1  0 Nov12 ?        00:00:00 ora_reco_orcl
oracle 466      1  0 Nov12 ?        00:00:00 ora_snp0_orcl
oracle 468      1  0 Nov12 ?        00:00:00 ora_snp1_orcl
oracle 470      1  0 Nov12 ?        00:00:00 ora_snp2_orcl
oracle 472      1  0 Nov12 ?        00:00:00 ora_snp3_orcl
oracle 474      1  0 Nov12 ?        00:00:00 ora_s000_orcl
oracle 476      1  0 Nov12 ?        00:00:00 ora_d000_orcl
```

画面1 Linux上でのプロセス表示 (Oracle部分だけ)

- ・システム表領域
各種の管理情報を保存する表領域。表の属性、データディクショナリ、ユーザーなどの情報はここに保存される
- ・ロールバック領域
更新の履歴情報を保存する表領域。トランザクションのロールバック情報が保存される
- ・一時表領域
SQL実行の際のソートなどの作業に使われる表領域
- ・ユーザーオブジェクト格納用表領域
管理者が任意に設定できる表領域。実際のアプリケーションデータなどはここに保存される

さらに表領域以外のファイルとして、

プロセス	内容	説明
DBWR	DBライタ	データベースバッファ (SGA) 内の変更された内容を、データベースのファイルに書き込む
LGWR	ログライタ	REDOLOGバッファの内容を、REDOログファイルに書き込む
CKPT	チェックポイント	チェックポイントが発生したときに、DBWR プロセスにシグナルを送り、データファイルおよび制御ファイルのヘッダ情報を更新する
SMON	システムモニタ	Oracleのシステム全体を監視する。起動時のデータベースのチェックと回復、使用されていない一時セグメントのクリーンアップを行う
PMON	プロセスモニタ	ユーザープロセスに障害が発生した場合、回復処理を行う。キャッシュのクリーンアップなどを行いプロセスが使用していたリソースの解放を行う
ARCH	アーカイバ	オンラインREDOログファイルの内容をアーカイブファイルへコピーする
RECO	リカバラ	分散データベース環境の場合使用。分散トランザクションがある場合、ローカルのRECOプロセスは一定間隔でリモートDBに接続、ローカルデータベースのコミットリカバリを行う
Dnnn	ディスパッチャ	マルチスレッドサーバ構成でのみ使用。接続されたサーバプロセスから使用可能な共有サーバプロセスへ要求を割り当て、該当するサーバプロセスに応答する
Snnn	サーバ	接続しているユーザープロセスからの要求を、ユーザープロセスに代わって処理する。専用サーバ構成では、ひとつのサーバプロセスが、ひとつのユーザープロセスを処理する

表1 Oracleのプロセスとファイル

- ・制御ファイル
表領域やデータベース（インスタンス）名など、最も基本的な情報が保存されるファイル
- ・オンラインREDOログファイル
トランザクションにおいてデータベースの変更を保存し、ロールバックや障害からの回復に使用される。最低でも2つ必要で、3つ以上が推奨されている
- ・アーカイブファイル
オンラインREDOログの履歴を保存する。バックアップ取得時以降のオンラインREDOログおよびアーカイブファイルがあれば、データベースの完全回復が可能

また、Oracleが主記憶上で確保しようとするバッファ領域には、SGA（システムグローバル領域）とPGA（プロセスグローバル領域）があり、SGAの内容は以下のようになっている。

- ・共有プール

処理効率向上のために、SQLの実行計画、コンパイル済みのSQL、データディクショナリの一部などが格納される

- ・データベースバッファキャッシュ
データファイルからのデータが格納される。データ検索、更新などはバッファキャッシュ上で処理され、物理ファイルが更新される
- ・REDOログバッファ
データベースバッファキャッシュに対して行われた変更処理の内容が格納される

ここで書いたOracleのアーキテクチャは、Oracleでのバックアップ/リカバリ処理の設計や計画、チューニングなどを行うには必須である。逆に言うと、このあたりのしくみを理解しておけば、問題解決を素早く確実にできるだろう。

バックアップ/リカバリについて

Oracleのような商用RDBMSを選択する

理由のひとつは、その頑強さ、長い間の実績に裏付けされた耐障害性にある。

Linuxでは、ext2（Second Extended File System）というファイルシステムが一般的に使われている。これは、Linux専用に開発されたファイルシステムであり、最大256文字のファイル名、ファイルシステム全体で4Tバイトまで拡張可能といった特徴がある。

しかし、最新のサーバ向けファイルシステムと比較した場合、よく言われるのが「ジャーナル機能の欠如」である。ジャーナル機能とはファイルシステム自体がファイルシステム内のデータブロック更新を監視/記録しており、障害などが発生しても、可能な限りデータの欠損やリンクの切断などの事態を防ぐようなしくみである。

ext2には、そのようなしくみがないうえ、急な電源断などがあるとブート時に、fsck（ファイルシステムチェック）が実行される。最近の大容量ディスクではこの処理に長時間必要である。さらに最悪の場合は、

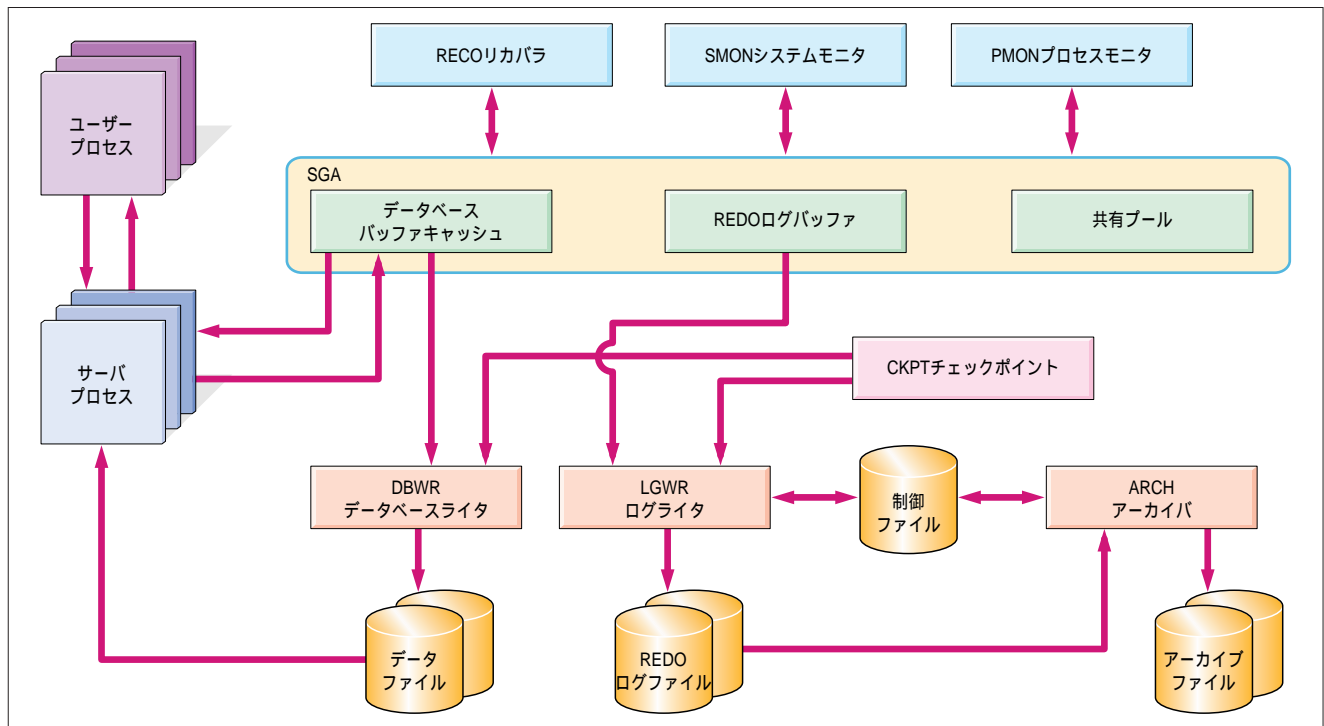


図1 Oracleのアーキテクチャ

ファイルの破壊などが起こる可能性もある。

もっとも上記のようなことがなくとも、データベースのバックアップは計画を立てて確実に実行する必要があるし、障害発生時に確実にデータを保護できるしくみを考えておく必要もある。

先にも解説したが、Oracleではバックアップファイル、オンラインREDOログとアーカイブログを使ってデータベースの回復を行う(図2)。

なるべく確実な回復を行うために、Oracle側でもいくつか手段が用意されているので、以下の指針でデータベースのファイル構造を設計、運用するとよいだろう。

(1) Oracleはアーカイブログモードで運用する

Oracleの運用モードには、アーカイブログモードと非アーカイブログモードの2つがある。データベースを作っただけ、あるいは初期データベースのデフォルト設定は、非アーカイブログモードであるが、何か特別な事情がない限りアーカイブログモードで運用するべきである。その移行手順を図3に示す。

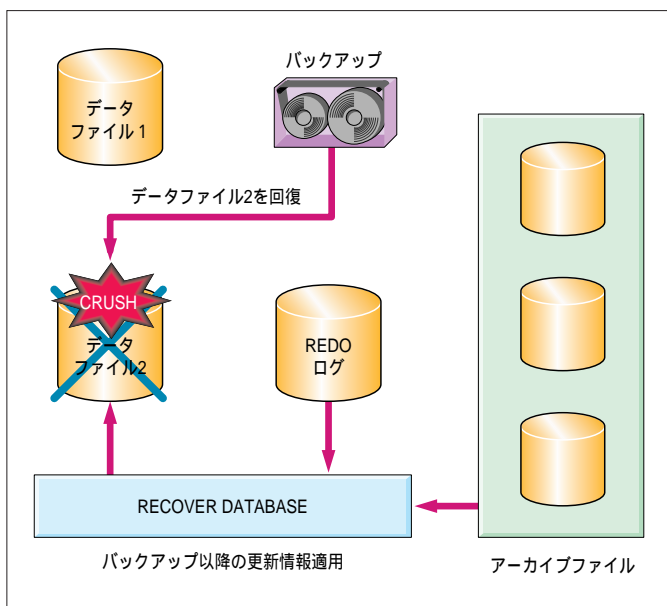


図2 データベース回復処理

ただし、アーカイブログモードにすると、データベースに更新が行われただけアーカイブファイルは増えていくため、定期的なバックアップと、その時のアーカイブファイルの削除を行う必要がある。このような処理は通常自動化しておくといだろう。もし、何かの理由でこの操作を行えないときは、非アーカイブログモードで運用するしかなくなる。

(2) 表領域が存在する物理ボリュームと、アーカイブログが置かれる物理ボリュームを別にする

Oracleのリカバリは以下のように行われる。

バックアップからファイルのリストアを行う

データベースをMOUNTモードで起動し、RECOVER DATABASEコマンドを発行する

RECOVER DATABASEコマンドは、自動的に必要なアーカイブファイルを適用するか、ユーザーに場所をリクエストする必要なログを適応し終わると、RECOVER DATABASEは終了するので、

管理者はデータベースの再起動を行う

アーカイブと実データを同じ物理ボリュームに割り当てると、その物理ボリュームが損傷した段階で、データベースを完全復活させる手段はなくなり、バックアップ取得時点までしか回復できなくなる。

これを別のボリュームに分けておけば、データ側が破損すれば、アーカイブとバックアップからデータベースを回復できるし、アーカイブ側が破損すれば、ただちにデータベースのバックアップを行い、アーカイブ側ディスクの交換を行えばよいということになる。

(3) 制御ファイルとオンラインREDOログファイルの二重化を必ず実施する

制御ファイルは小さいものであるが、Oracleの正常実行には必須のファイルである。また、更新処理にはオンラインREDOログファイルが必要である。これらのファイルはOracleの設定で複数のディスクにミラーできるので、それを忘れずに行っていくこと(図4)。

これらの処置を行えば、クラッシュ時に

アーカイブモードの変更。

No Archive Mode Archive Mode

(1) データベースをshutdownする

(2) 初期パラメータファイルを編集する

- ・ LOG_archive_start=TRUE (自動) or FALSE (手動)
- ・ アーカイブファイルの位置及び名前を設定を行う
LOG_ARCHIVE_DEST=ログファイルの位置のフルパス名
LOG_ARCHIVE_FORMAT=arch%s.arc アーカイブログファイル名の形式

(3) データベースをMOUNTモードで起動する

(4) ALTER DATABASE コマンドにより、アーカイブモードの変更を行う
SQLDBA>ALTER DATABASE ARCHIVELOG;
NOARCHIVELOG;

(5) データベースをオープンする

起動後に自動/手動を切り替えたい場合は、以下のコマンドを使用する
ALTER SYSTEM ARCHIVELOG START (自動に)
STOP (手動に)

図3 アーカイブログモードへの移行手順

データが損失する可能性はかなり低くなる。

(4) RAIDでもバックアップは必須

最近、ときどき見かける意見に、RAIDだからバックアップは不要であるという話を聞くことがあるが、これは大きな誤解である。RAIDでもバックアップは必須である。それは以下のような理由による。

- ・ RAIDコントローラに障害が起これば、すべての情報が壊される
- ・ 実際のデータ損失の大半は、ミスオペなどの人為的要因。これはハード側では防げない

最初からRAIDを構成するようなシステムは、ハードウェアの冗長性を確保することによって、重要なデータの損失や、運用の停止といった事態を防ごうとしているはずである。このようなシステムでバックアップミスやミスオペレーションによるデータの損失は大きなものになることは容易に予想できるだろう。

このような点からも、定期的なバックアップオペレーションは必須である。

(5) バックアップメディアの保守

これについても特記しなければならない。私が知っている例では「システムに障害が発生し、そこからの回復を行おうとしたと

き、バックアップに使ったテープが読み込めずどうにもならなかった」という事例がある。このようにバックアップメディアの管理がまずいと、いざという時にたいへんなことになる。これを防ぐには以下の点に注意する。

- ・ ひとつのバックアップ対象に対して、2本以上の媒体を準備し、世代管理を行う(図5)
- ・ 媒体の寿命や繰り返し使用頻度の制限を確認し、一定期間以上使用したものは廃棄や定例オペレーションから除外する
- ・ 保管場所は、冷暗所で乾燥しているところが基本
- ・ バックアップ媒体は、信頼性の高いものを選択すること。値段が安いからといってノーブランド品などは選択しないことが望ましい

最近のハードディスクの大容量/低価格化に、正直なところバックアップ装置は追いついていないのが実状である。IDEで40Gバイトクラスのハードディスクが秋葉原だと2万以下で買えることが一般的になった現在、バックアップもそれに対応した構成が要求されるようになっている。ハードディスク自体を複数世代分用意してバックアップ装置として使うということも考えられる。

まだ、価格的には高いのだが、ニューテックなどからLinuxでも使えるDDS4、DLT、AITといったバックアップ装置も出てきつつあるので、これらも今後の提案に使えるだろう。

また、これらのバックアップ以外に、Oracleならばスタンバイデータベースという方式も利用できる。これはアーカイブログのデータを使うことで、主データベースの内容を、遠隔地のデータベースに反映させることを目的としたものである。これはマルチクラスタのように高価なハードウェアや特別なソフトウェアを必要とせず、障害時に素早く主データベースからスタンバイデータベースに切り替えることで、迅速なシステムの回復を可能とする。

性能を考慮したデータベース設計

データベースの場合は、レスポンスタイムが品質上の重要な指標になることが少なくない。レスポンスタイムをマークするためには、あらかじめそれを前提とした設計が必要となる。チューニングのテクニックも、それだけで厚い本が1冊書けるくらいだが、基本は以下の通りである。

(1) 十分なメモリの確保

OracleのSGAに必要なだけの空き実メモリの確保は必須である。またどの程度のバ

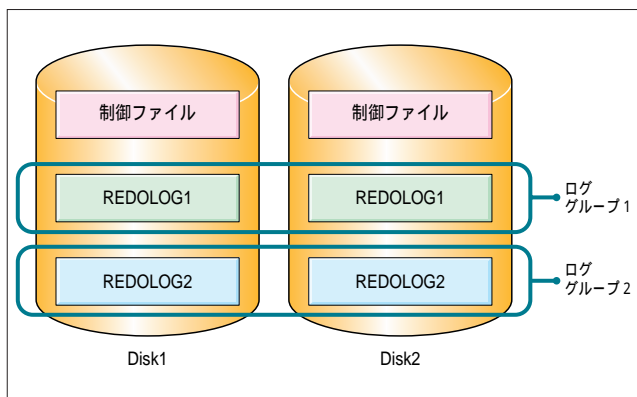


図4 制御ファイルとREDOログのミラー化

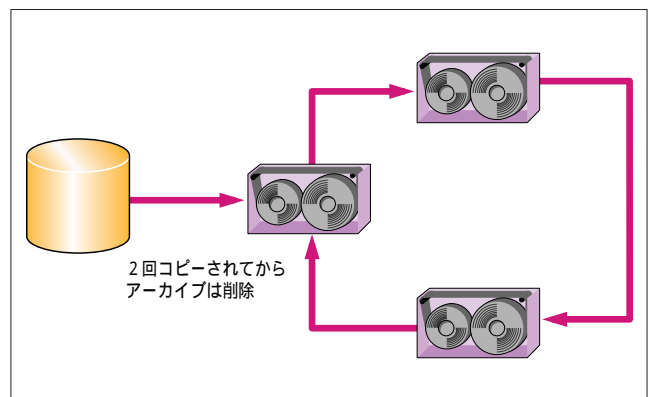


図5 バックアップ媒体の世代管理

ツファが必要となるかは、予想される検索や更新トランザクションの件数規模などから推察しなければならない。また、RDBMSのみならず、それを使用するアプリケーションのためのメモリ確保も必須となる。

メモリが不十分では、いろいろなチューニングテクニックを駆使してもその効果は小さいし、今日のたいのシステムでは、その人件費を考慮すればハードのコストは安いものである。

SGAの状況を見るには、svrmgrlでshow sgaコマンドを使う(画面2)。Linux側のメモリ資源は、vmstat、free、topコマンドで見ることができる。V\$SYSSTATビューで動的な監視も可能である。

(2) アクセスの分散

これはバックアップでもいえることであるが、特定テーブルやあるいはテーブル内の特定の場所にアクセスが集中するようだと、そこが大きなボトルネックになる。なるべく分散するような設計を心がけるべきである。

さらに論理的にそのような集中が避けられない場合は、Oracleなら、逆キーインデ

ックス、パーティショニングでパーティションごとに別のディスクを配置するなどの処置を考える必要がある(図6)。

このような集中を監視するには、V\$FILESTATというビューの監視が有効である。

(3) SQLの見直し

パフォーマンス問題の大半は、作成されているSQL文に起因されていると言われていいる。SQLの処理は基本的に集合論の処理であり、検索条件に一致する物を抽出するには、内部的に何らかのソーティング処理が入る場合が多い。この場合、そのような並び替えの件数やデータ量が多くなれば、必然的にレスポンスタイムは悪化する。このあたりはかなりテクニカルな要素が大きくなるが、where句における結合条件やインデックスの使用などでうまく回避する手段を考えるべきである。

(4) インデックス

テーブルを検索する際に、必要な検索条件にインデックスが張られていれば、全件を検索しなくとも数回のディスクアクセスで必要な行を検索することができる。

Oracleは基本的にはバイナリツリーによるインデックスを作成する。特にテーブル同士を参照するのに使う外部キーなどにはインデックスを張るべきである。ただし、更新/追加時にはインデックスの修正も必要になるから、逆に性能悪化の原因になる。インデックスが必要な列は厳選するべきだ。

また、インデックスを張る列が決まり切った値を使うとき(男女の区分け、学年など)は、Oracle8iならビットマップインデックスが有効である。これは比較的件数が少ないテーブルのインデックスとして用いても有効である。

(5) ブロックのパラメータ

OracleはPCTUSED、PCTFREEといったパラメータで、表領域やテーブルの作成時にブロックの使用率を設定することができる。たとえばある行の挿入が行われた後のブロックの状態を図7に示す。ここでこの行の更新が行われて、行のサイズが増大したとする。ブロックに空きがあれば、それは同一ブロック内に格納可能である。しかし、もし十分な空きがなければ、1つの行が複数のブロックにまたがって配置されることになり、これには複数回のディスクアクセスが発生する。このような行を連鎖行と呼ぶ。

```
[oracle@miracle1 oracle]$ svrmgrl

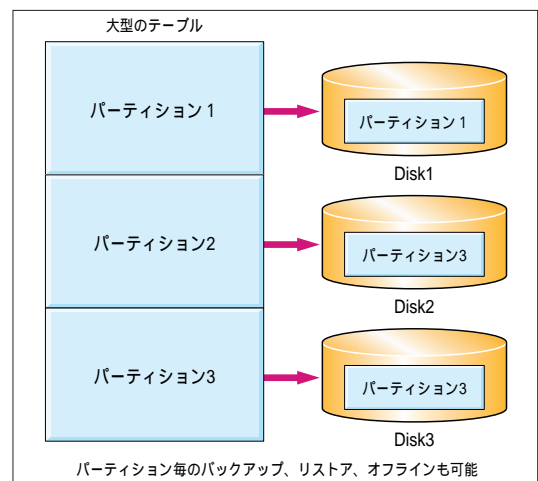
Oracle Server Manager Release 3.1.6.0.0 - Production

Copyright (c) 1997, 1999, Oracle Corporation. All Rights Reserved.

Oracle8i Enterprise Edition Release 8.1.6.1.0 - Production
With the Partitioning option
JServer Release 8.1.6.0.0 - Production

SVRMGR> connect internal
接続されました。
SVRMGR> show sga
システム・グローバル領域合計      56012784バイト
Fixed Size                          69616バイト
Variable Size                       38993920バイト
Database Buffers                    16777216バイト
Redo Buffers                         172032バイト
SVRMGR>
```

画面2 SGAのメモリサイズをsvrmgrlで見る



もうひとつのケースとしては、行全体が入るブロックを探してそこに格納し、元の行があった場所には、その移行先の情報が格納される。これを移行と呼ぶ。これが増えるとパフォーマンスに影響を与えてくる。これらはanalyzeコマンドかV\$SYSSTATで監視可能である。解消にはエクスポート/インポートを使う。

また、たいていのRDBMSはクエリに実行計画を最適化する機構（オプティマイザ）を持っている。このオプティマイザは、さまざまな統計情報などから、インデックスを使うか否かや結合条件などを決めていく。この統計情報を収集するには、analyzeコマンドを使う。

```
analyze index ( index_name )
compute statistics
```

ここでは紙面の都合もあり、あまり詳しく書けなかったが、これらをキーワードにしてOTN-Japanなどからドキュメント入手すれば、かなりの情報が得られるはずである、また、チューニングに関する書籍も多く出版されている。

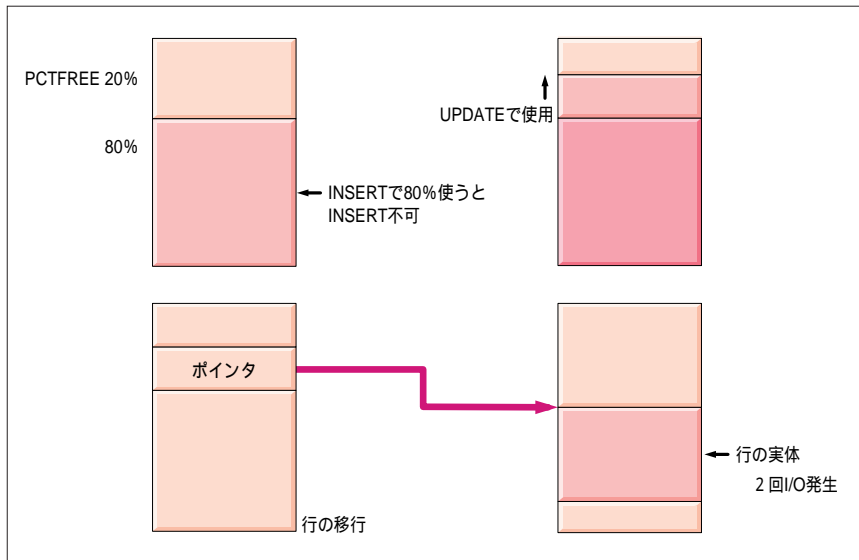


図7 データベースブロックの状態

RDBMSの設計

RDBMSの設計に関してより上流部分について解説しよう。RDBにはいくつかの重要な特徴がある。その中でも特筆すべきは、データ構造を表形式で表現しているということである。その表は以下の性質がある（図8）。

- ・テーブル内に同一の行（Row）は存在しない
- ・テーブル内に列（Column）は、互いに独立しデータ型を持つ
- ・行の順番に意味はない
- ・列の順番に意味はない

これらはテーブル設計の大原則になる。このような構造を作り出すために使われるのが、「正規化」という手法である。

システムに必要な要件や業務分析がまとまってくると、それらの現実の世界を、システムに翻訳するモデル化が必要となってくる。このような分析の一助としてもモデルは使われる。分析と改善が行われた概念モデルを元に、論理モデル設計が行われ、

論理モデルを元に、物理設計（実際のデータベースのスキーマやプロセス）が行われる。

概念設計の段階で、さまざまな事象（オブジェクト、エンティティ）が抽出されてくるが、これらの事象は、何らかの属性を持ち、それらは互いに関連あるものとしてまとめられ、データベースに格納される必要がある。

たとえば、図9のような出荷伝票を考えてみよう。出荷伝票には次のような情報がある。

顧客名
顧客連絡先
納品商品名
納品個数
納入先

最低限書かれているのはこれだけだ。実際には出荷と納品時にロット構成が違って

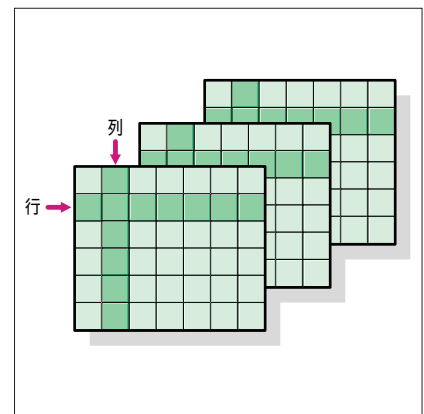


図8 RDBMSのテーブル

× 株式会社殿向け出荷	
商品 A	3個
商品 B	2個
届け先住所	
東京都渋谷区代々木xx-yy-zz	
連絡先電話番号	
03-5351-0000	

図9 出荷伝票の例

いたり、中間業者や輸送途中の高めなど、業界や商品、商習慣によりさまざまな条件が考えられる。本質的には納品と出荷は分けて考えるべきであるが、ここでは単純に上記の構成を考えてみる。

ここから読みとれるのは、出荷という業務を行うのに管理したい情報は、

- ・顧客に関する情報
- ・商品に関する情報
- ・出荷そのものの情報

この3つであるということである(図10)。

上記のうちで、出荷そのものに関する情報以外の、顧客、商品などの情報は、企業システムの他の部分でも有効に使えることは、すぐに想像できるはずである。また、このようなデータはアプリケーションにも本質的に依存せず、その属性は若干の変更はあってもビジネスの中で汎用的に存在するはずである。これがデータモデルであり、データを中心とするDOA(Data Oriented Approach)の考え方である。

正規化は「ひとつの事実は1カ所だけに存在させる」というDOAにおける鉄則である。たとえば出荷のシステムと売り上げ計算のシステムで、製品情報を別々に管理するケースを想定してみよう。何か製品が追加、削除するたびに2カ所でも変更が必要になり、メンテナンスが煩雑となり運用コストの増大やミス増加に結びつくのは

容易に想像できると思う。

この正規化は、以下のようなプロセスを経て実現可能である。

第1正規化 繰り返しの除去。いわゆるオカレンスを切り離すRDBMSでは通常、細目1、2、……といった持ち方は行わない

第2正規化 識別子の一部の属性に従属する属性の分離

第3正規化 識別子以外の属性に従属する属性の分離

このように比較的、機械的な手順でこれらを達成できる(実際には、よりやっかいな分析が必要なケースも少なくないのであるが)。

また、これらの分析技法は、いわばRDBMSのテーブルで管理すべき対象の切り出し作業であり、実際にはデータモデルからスキーマへの転換には、RDBMSのアーキテクチャに合わせた設計や、コードの付与などが要求されるものの、比較的親和性が高く、普及している。この点に関して、今は広く書籍があるので、詳しくはそれらを参照することをお勧めする。ここではこのようなものがあるという紹介に留めておく。

一方で、RDBMSを中心としたDOAの考え方は、オブジェクト指向を中心とする分析技法に移行しつつある。オブジェクト

指向は、データとそれに対する処理をひとつのオブジェクトとしてまとめている。図11は最近普及しつつある、UMLの記法の例である。ここではオブジェクトの名前や属性のほか、オブジェクトで実行できるメソッドが記述されている。これはJavaなどでシステムを作るときに整合性がよいとされている。

しかし、オブジェクト指向をベースとするオブジェクトDBMSは、一部では採用されているものの、あまり広く普及しているとはいえない状況である。これは筆者の考えであるが、

- ・データとプロセスの寿命の差が大きい
データと比較するとアプリケーションは、高い頻度で変更される
- ・DWHなどの分析、ロード/アンロード、バックアップや一部移行などデータそのものを加工するケースが多い
- ・SQLというまがりなりにも統一的なベースが存在する
- ・チューニングスキルの蓄積がある

といった状況があるものと思う。

このように、さまざまな設計手法の理解と取得も、今後のソフトウェアエンジニアにとっては大きなテーマになると思う。これはLinuxのうえでも同じことであろう。

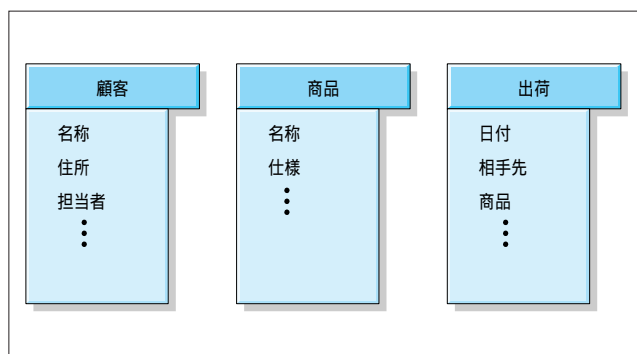


図10 各データベースのテーブル

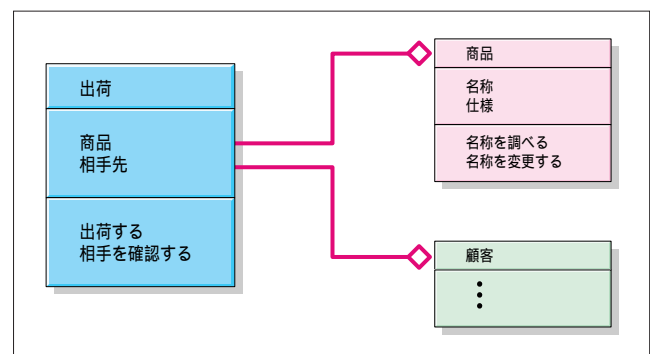


図11 UMLの記述例