【A-1】

Oracle Linux
Summit2003 in Spring

Oracle on Linux startup講座
～Linuxを選択してみる～

ミラクル・リナックス株式会社
製品本部 マーケティング部
プロダクトマネジャー
佐藤　剛春

Agenda

- RDBMSって何？
  - RDBMSに求められること
- アーキテクチャ概要
  - 整合性の確保
  - 記憶領域管理
  - インスタンス
  - 内部動作
- 稼働環境
- 運用時の負荷の軽減
- Linuxを選択する理由
  - 高い信頼性・安全性
  - DB Serverとして高性能
  - File Serverとして高性能
  - スケーラビリティー向上
  - 進化するLinuxカーネル

RDBMSって何？

- Relational DataBase Management Systemの略称
- Relational Database
  - データを単純な2次元の表形式で表現
  - データ項目の列とレコードの行によって構成
  - 各表を関連付けて管理可能
- Database Management System
  - 複数のユーザによる、複数または同一データの使用を可能とする機能を提供
    - データ整合性を保全する機能(トランザクション管理)
    - データ破壊からの復旧機能(バックアップ・リカバリ)
    - 他

RDBMSに求められること

- データの種類が増加
- あらゆる情報へのアクセス要求
- より高度なサービスの要求

Internet

システムの追加？

部門での管理？

既存システムとの連携は？

RDBMSに求められること

- 情報の集約の重要性
  - 目的毎に構築されたシステムでは、
    - どこに最新のデータが存在するのか？
    - どんな形式で格納されているのか？
    - 類似データとの関連性はどうなっているのか？
    - 全社的な情報活用にはどのデータを使用すればよいのか？
    - システムの運用主体は利用部門？情報システム部門？

  といった把握が困難となり、情報のフラグメントと重複が発生する。

整合性の確保
～ロックの違い

	Oracle	Others
行レベルロック	Yes	Yes
マルチバージョニングのサポート	Yes	-
ページ/テーブル ロック	テーブル	両方
ロックエスカレーション	-	Yes
ロード中のデッドロック	-	Yes
分離レベルのトレードオフ	-	Yes
読み込みが書き込みをブロック	-	Yes
書き込みが読み込みをブロック	-	Yes
ダーティリード	-	Yes

行レベル・ロックとは

- あるユーザが更新中のデータに対して、別のユーザの更新を防ぐための排他制御が「ロック」です。
- 一般的なロックのレベル（単位）
  - 表
  - ページ
  - 行
- ロックの単位が大きくなると、他のデータにも不要なロックがかかってしまい、ロック解除待ちが多発するため、処理効率が大幅に低下します。

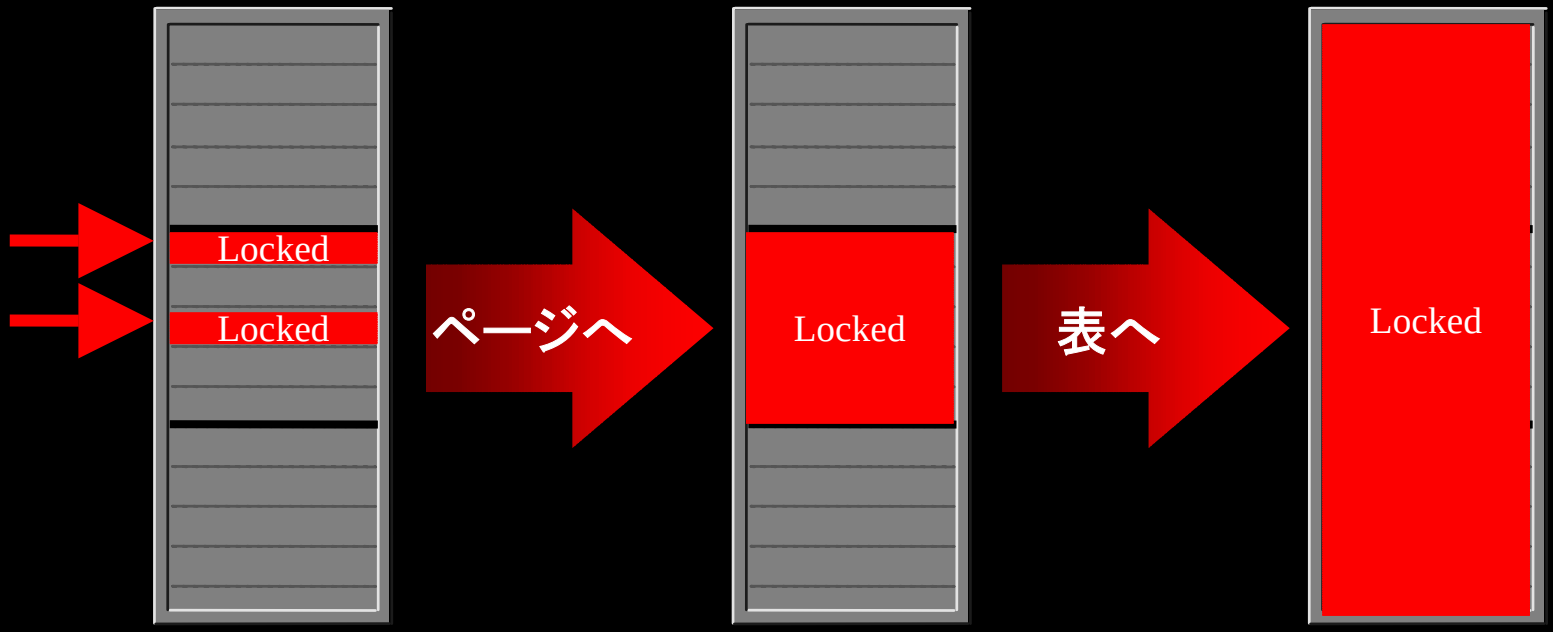
行レベル・ロック機能

データベース機能比較表

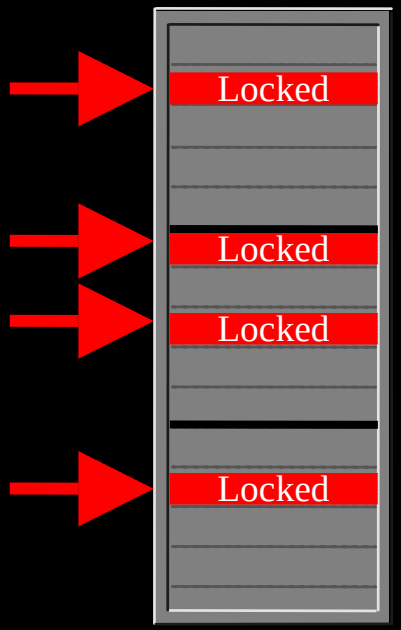
	他社製RDBMS	Oracle9*i* DB
行レベル・ロック	○	○

他社製RDBMS
不完全な行レベル・ロック

- ロックする行数がある一定の値を越えると、ロック・エスカレーションが発生し、ページ・レベル、表・レベルへ移行する。
- アプリケーションの同時実行性の低下



Oracleデータベース
完全な行レベル・ロック



- Oracleデータベースは常に行レベル・ロックをしています。
- Oracleの行レベル・ロッキングは、ロックの行数に制限がなくロックエスカレーションが発生しません。
- きめ細かい行レベル・ロッキングにより、ロックの競合が大幅に軽減されます。

行レベル・ロック機能

データベース機能比較表

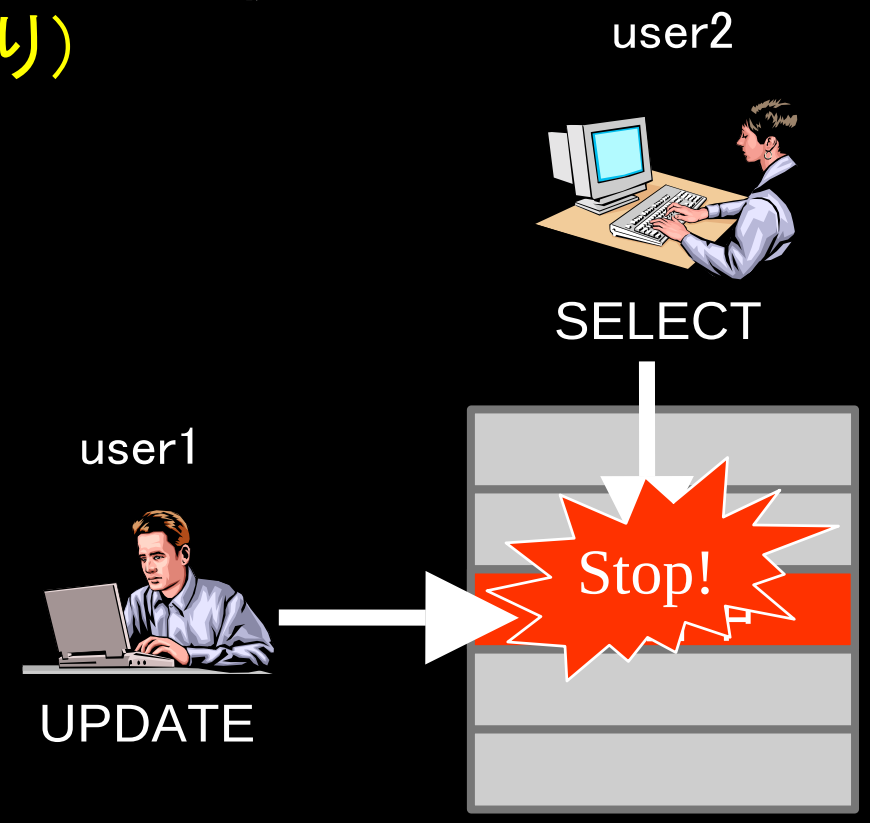
	他社製RDBMS	Oracle9*i* DB
行レベル・ロック	△	◎

トランザクション分離レベル

分離レベル（ANSI／ISO SQL92）	Oracle	他社RDBMS
UNCOMMITTED READ（未コミット読取り）	−	○
READ COMMITTED（コミット読取り）	◎	◎
REPEATABLE READ（繰返し可能読取り）	−	○
SERIALIZABLE（直列可能）	○	○

デフォルトは「**READ COMMITTED（コミット読み取り）**」
しかし、ベンダーによって実装方法が違います。

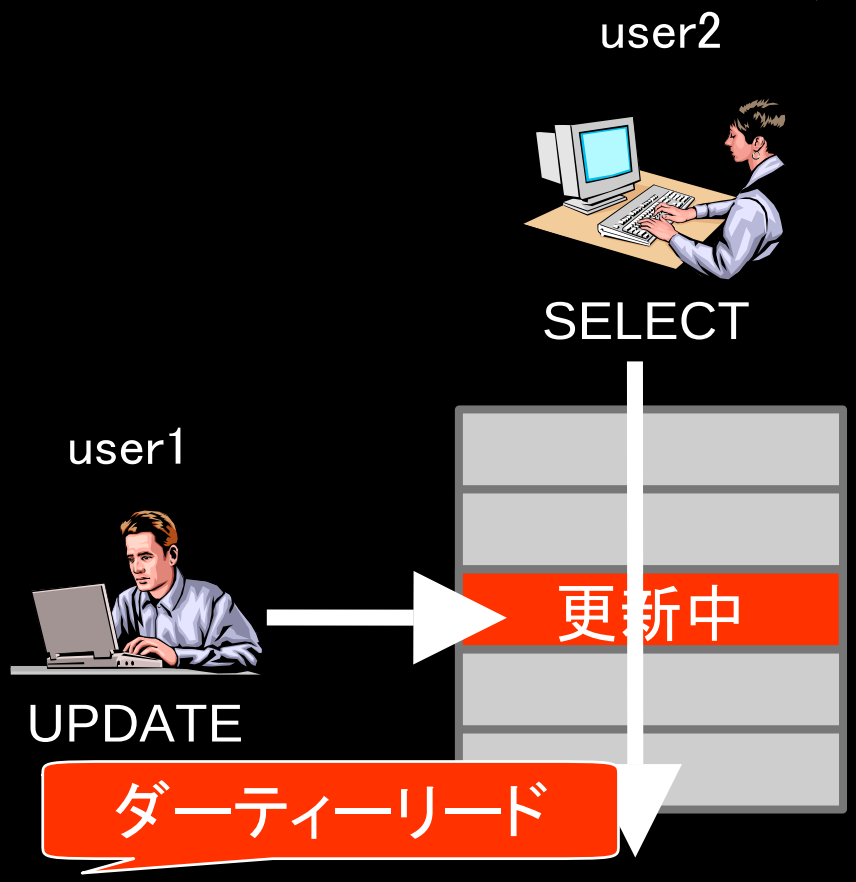
他社製RDBMS
ロックを使用したREAD COMMITTED（コミット読み取り）



- user1があるデータを更新。同時に行をロック。
- user2が参照開始。
- ロックされている行があると待ち状態。

同時実行性の欠如

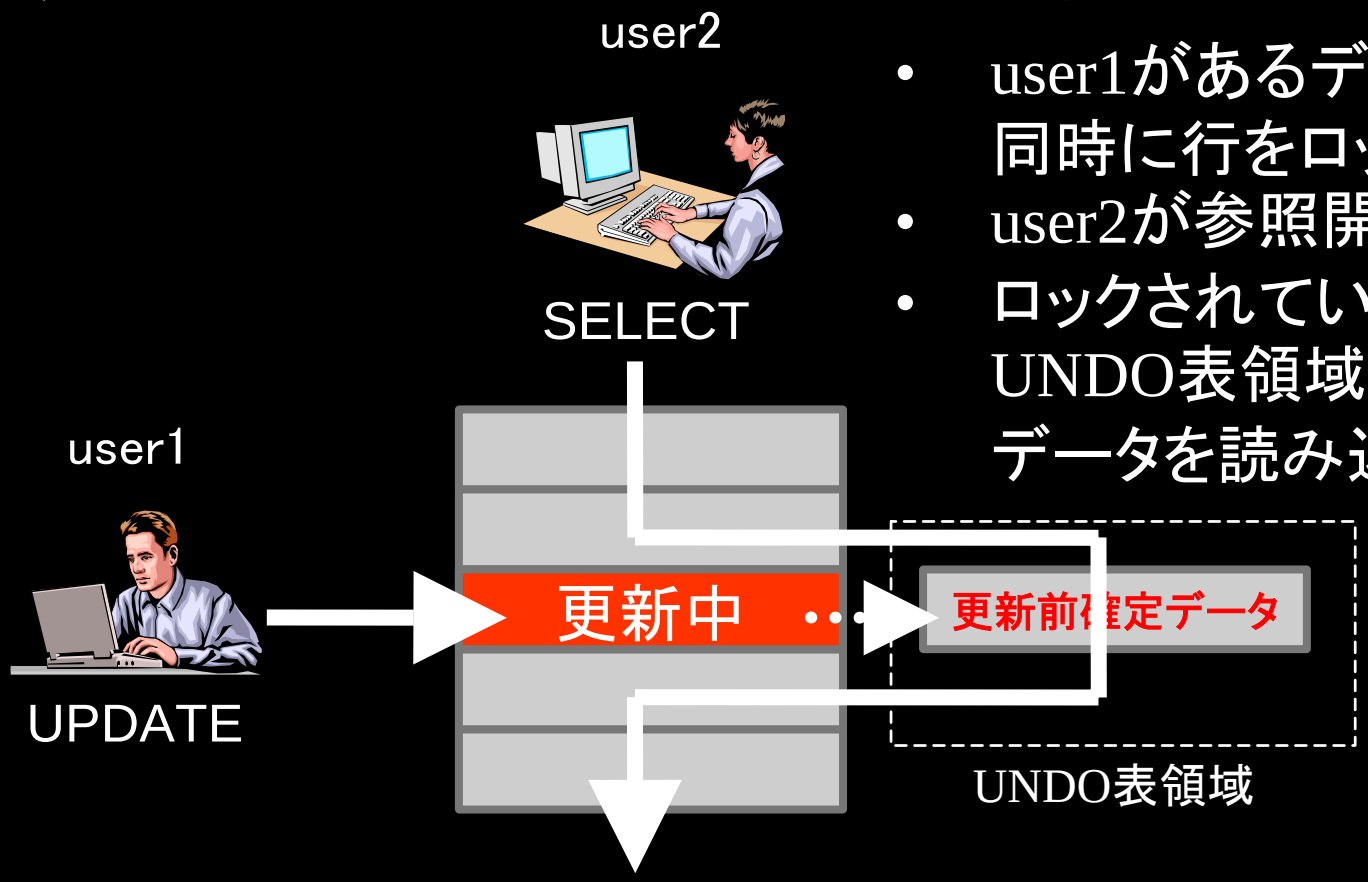
他社製RDBMS
READ UNCOMMITTED（非コミット読み取り）



- user1があるデータを更新。同時に行をロック。
- user2が参照開始。
- ロックされている行があると更新中のデータを読み込む

データの一貫性の欠如

Oracleデータベース
真のREAD COMMITTED（コミット読み取り）

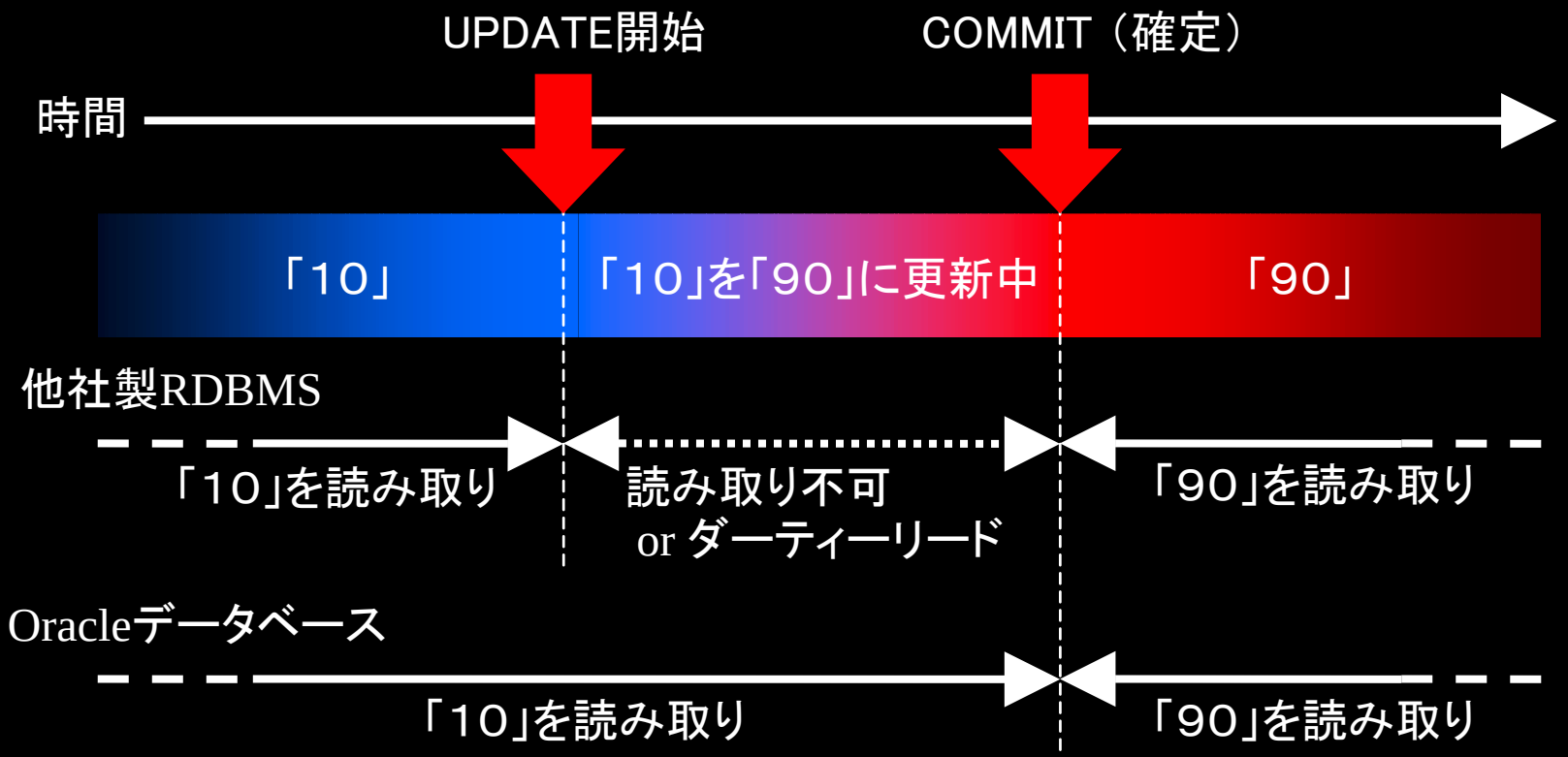


- user1があるデータを更新。同時に行をロック。
- user2が参照開始。
- ロックされている行があるとUNDO表領域にコピーされたデータを読み込む。

同時実行性と**データの一貫性**の両立

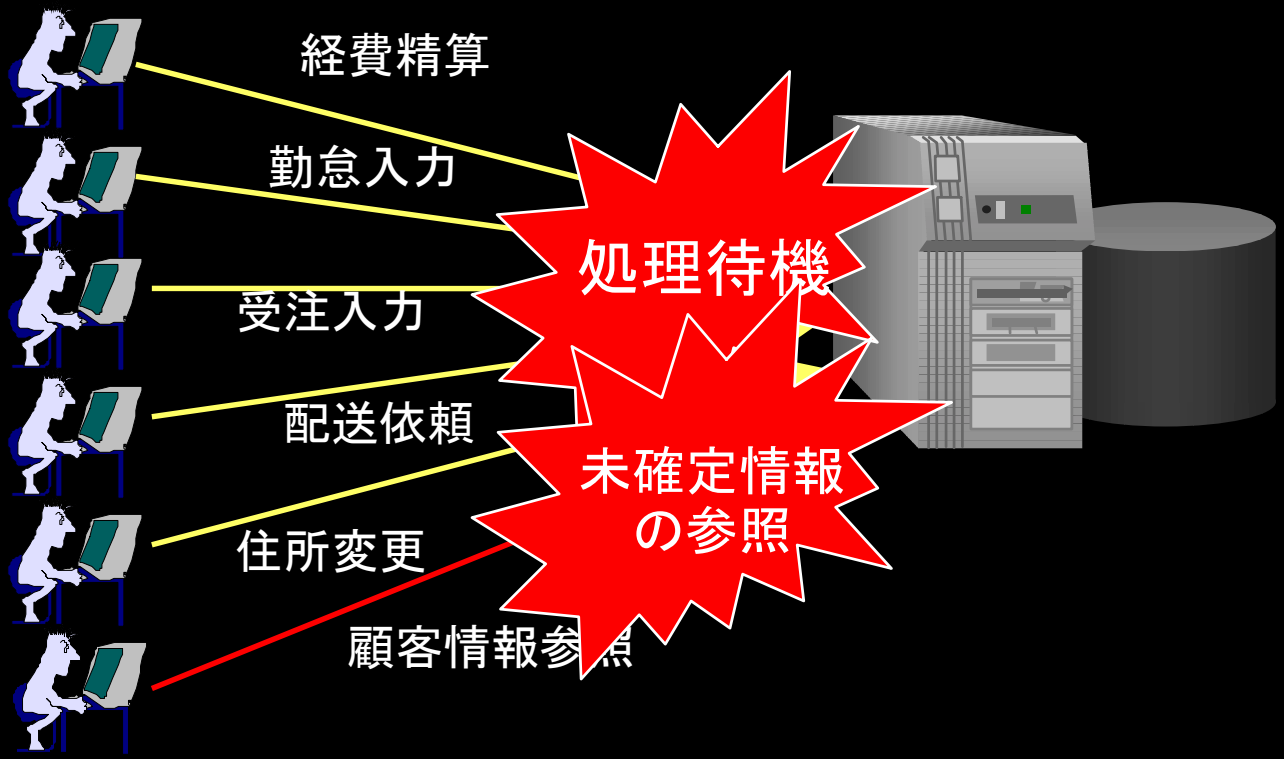
どのデータを読むのが正しいのか？

「10」を「90」に更新する場合。

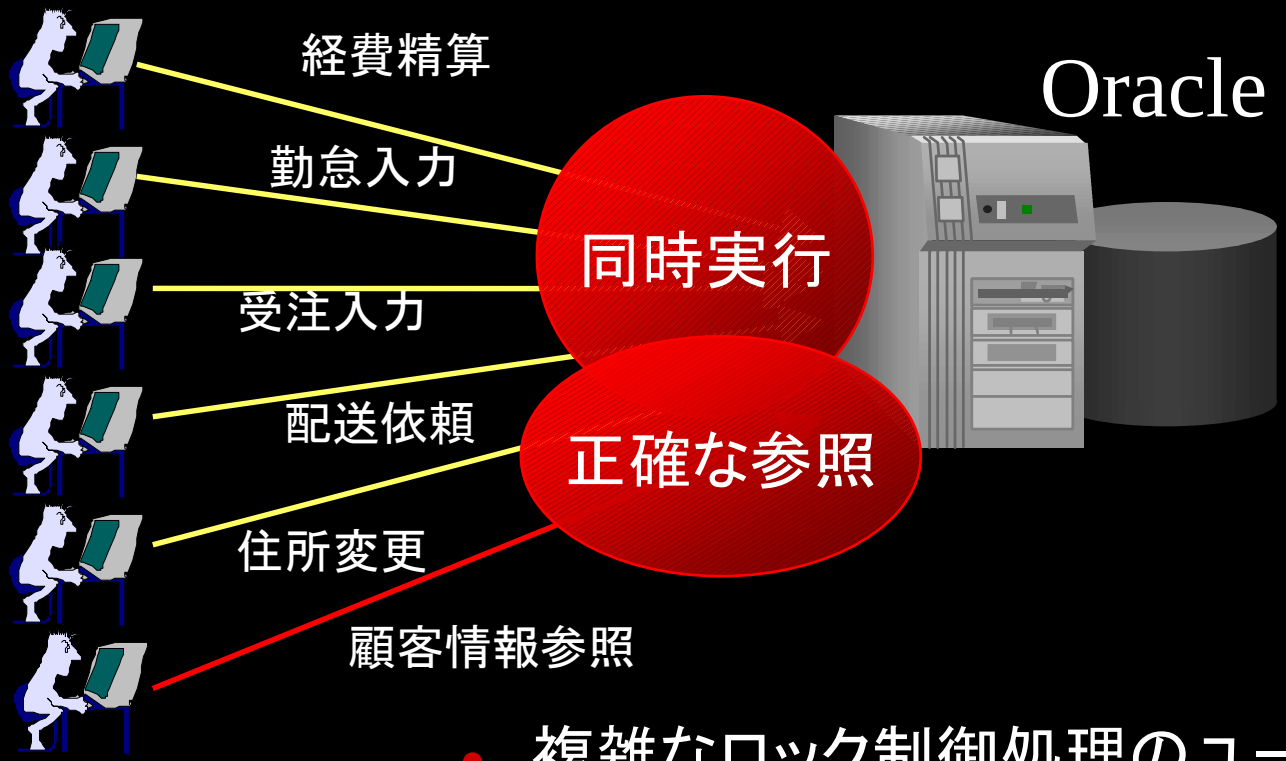


整合性の確保
なぜロックを使用しない読み取り一貫性が重要か？

他のRDBMSで、複数ユーザからのアクセスに対して、Oracleと同等のレコード制御や読み取り一貫性を実現しようとすると…



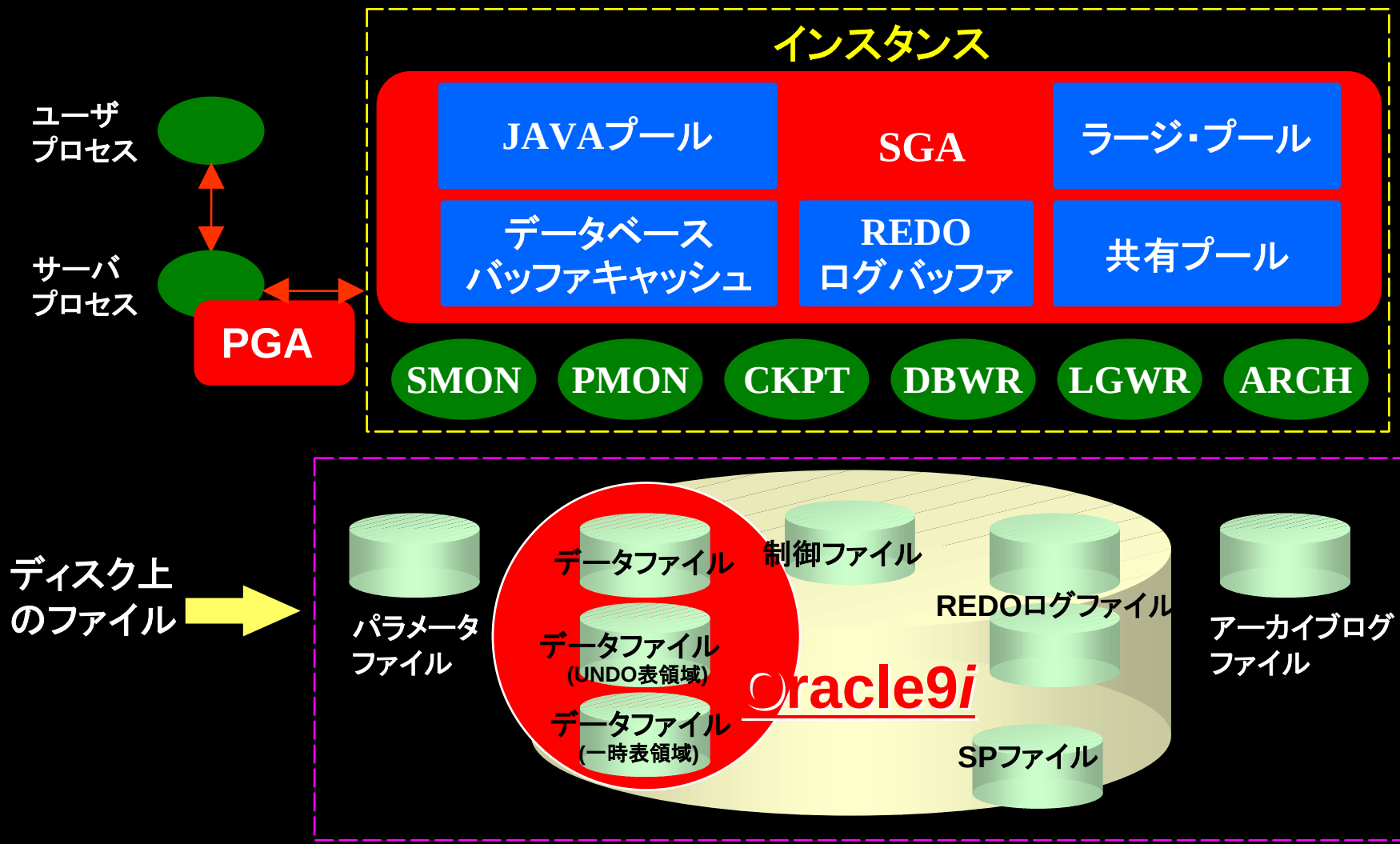
整合性の確保 – Oracleでは…



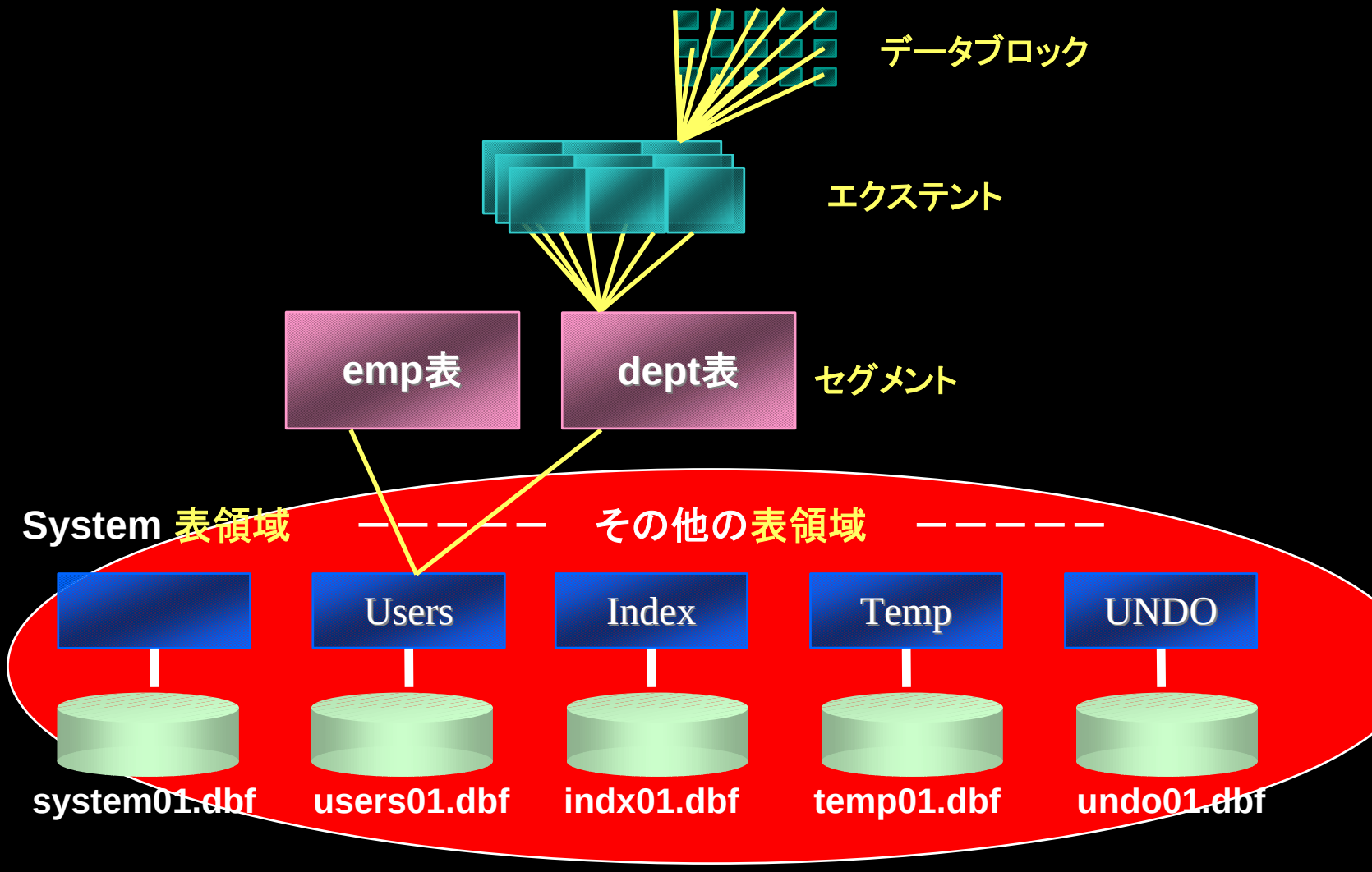
- 複雑なロック制御処理のユーザ開発
- 読み取り一貫性保証処理のユーザ開発

などは不要

アーキテクチャ概要

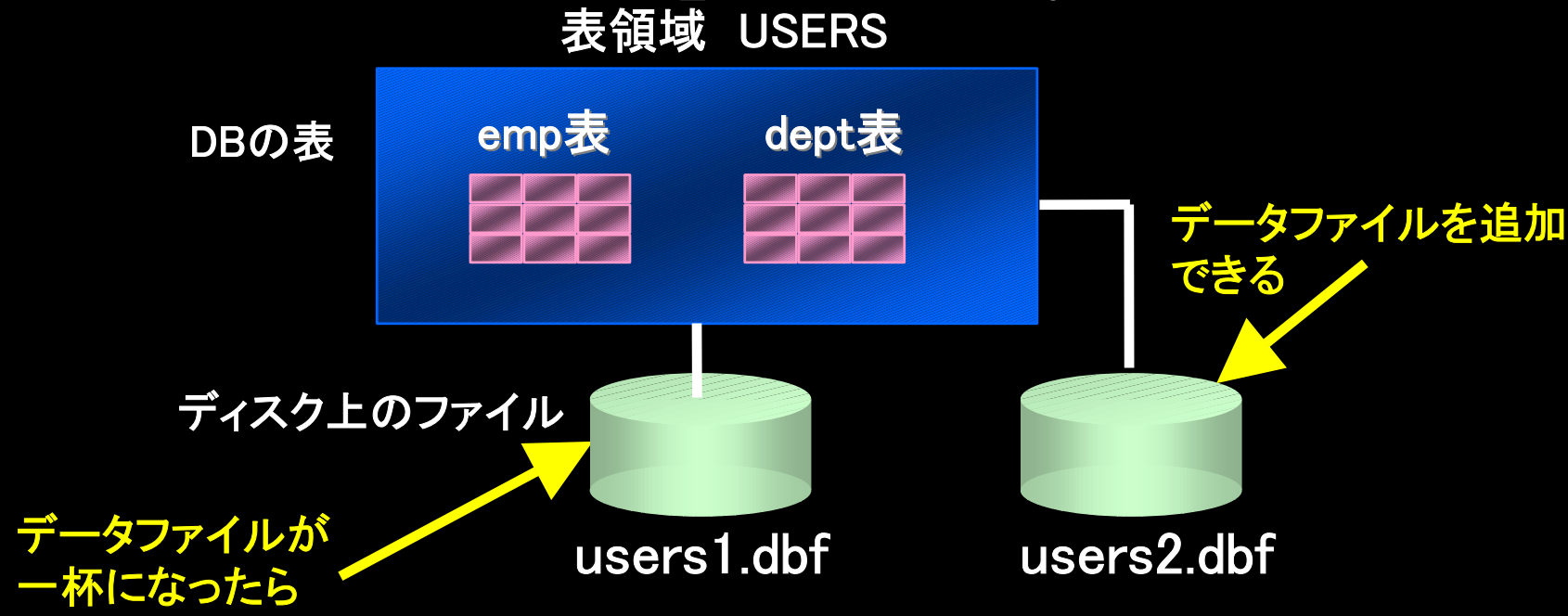

インスタンス
ユーザ
プロセス
サーバ
プロセス
PGA
JAVAプール
SGA
ラージ・プール
データベース
バッファキャッシュ
REDO
ログバッファ
共有プール
SMON
PMON
CKPT
DBWR
LGWR
ARCH
ディスク上
のファイル
パラメータ
ファイル
データファイル
データファイル
(UNDO表領域)
データファイル
(一時表領域)
制御ファイル
REDOログファイル
SPファイル
Oracle9i
アーカイブログ
ファイル

記憶領域管理

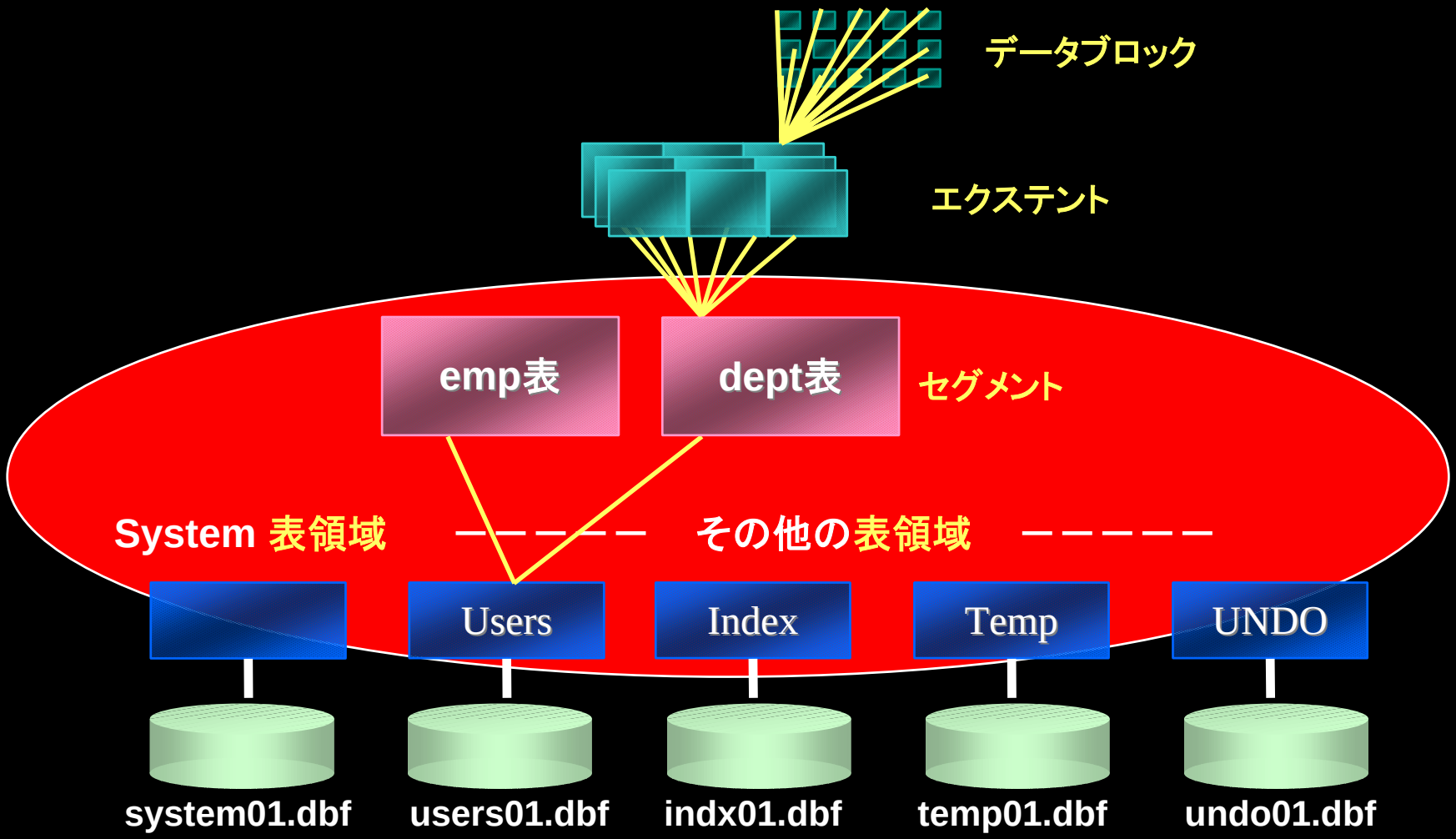

データブロック
エクステント
emp表
dept表
セグメント
System 表領域
その他の表領域
Users
Index
Temp
UNDO
system01.dbf
users01.dbf
indx01.dbf
temp01.dbf
undo01.dbf

表領域とディスク上のファイルの関係

- 表を入れる枠を作成し、表領域と呼ばれる領域
  と
  ディスク上のファイルを対応させる。

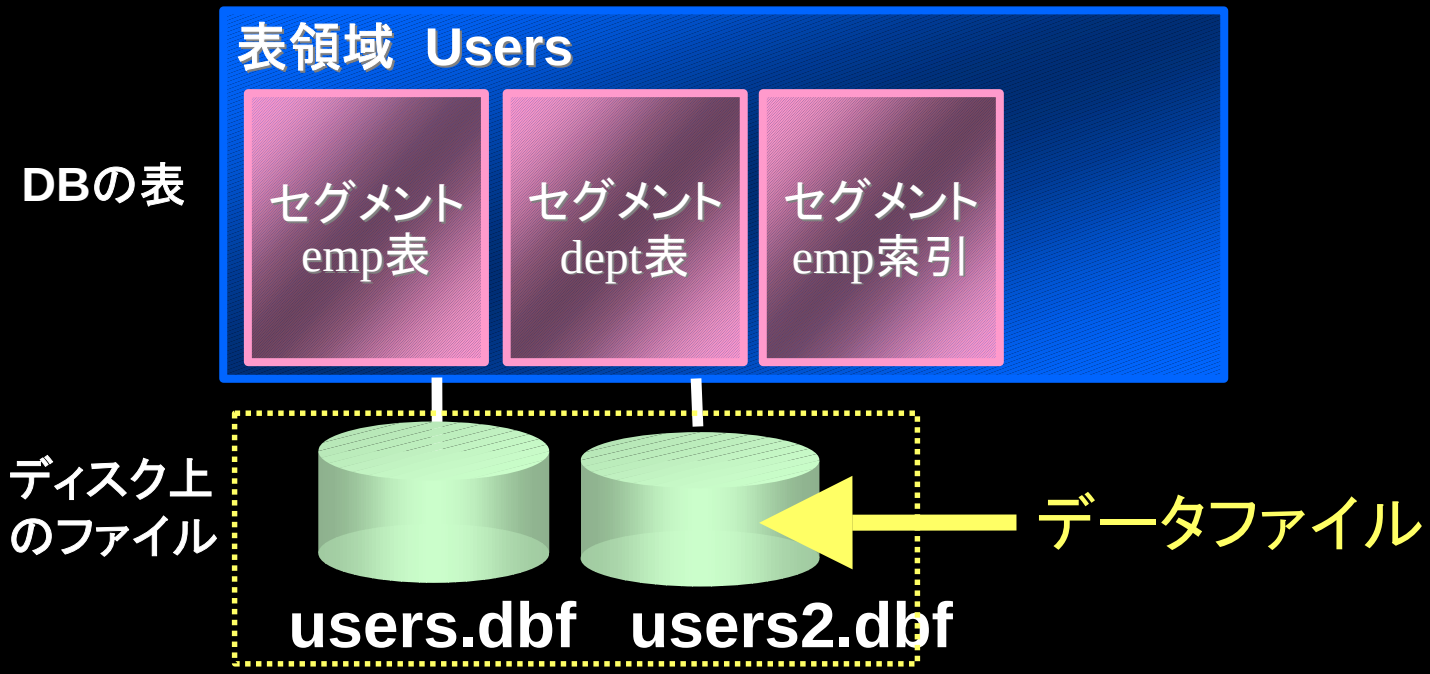


記憶領域管理

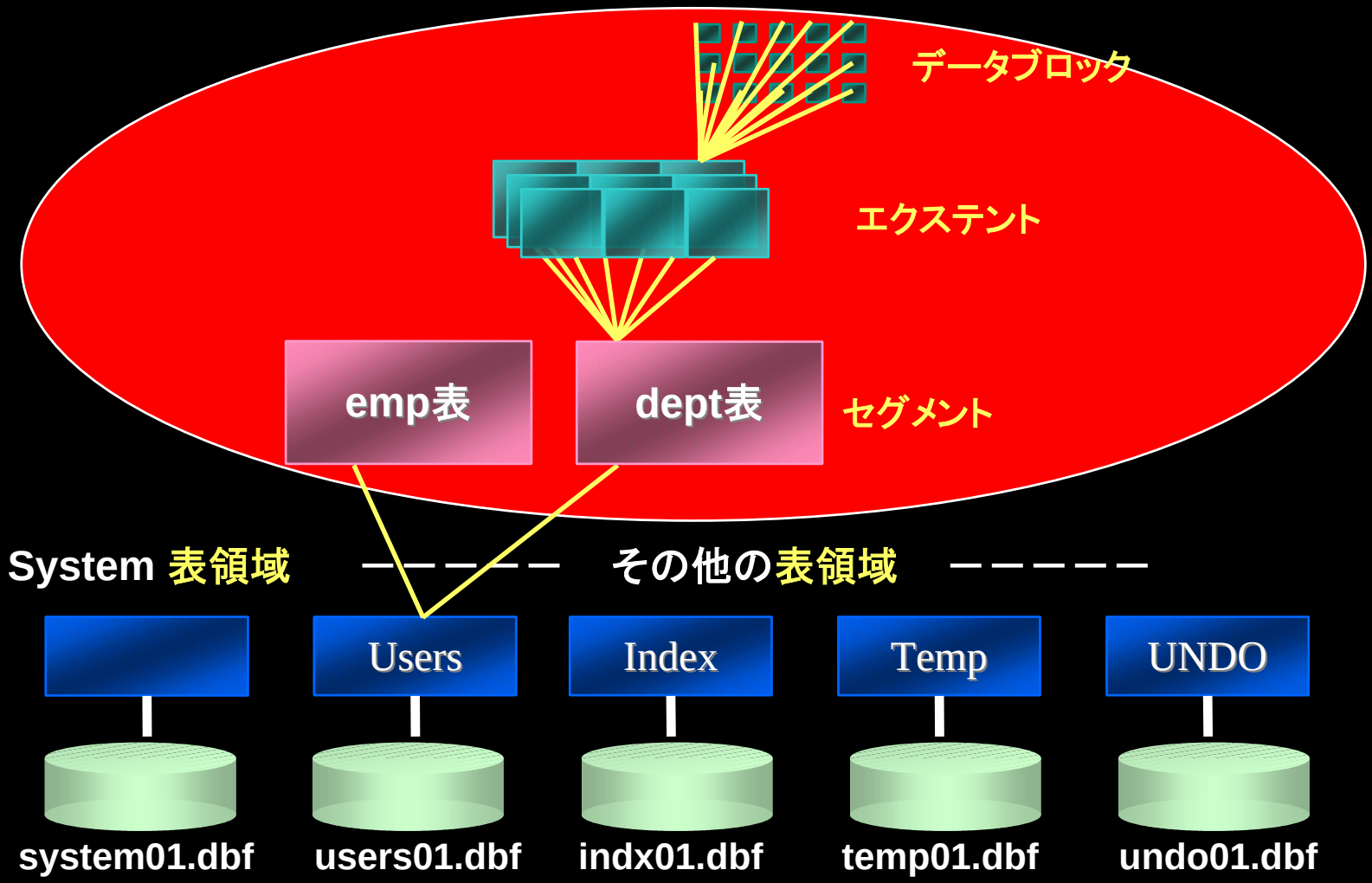

データブロック
エクステント
emp表
dept表
セグメント
System 表領域
その他の表領域
Users
Index
Temp
UNDO
system01.dbf
users01.dbf
indx01.dbf
temp01.dbf
undo01.dbf

表領域とセグメントの関係

- 表領域は、複数のセグメントから構成される。



記憶領域管理


データブロック
エクステント
emp表
dept表
セグメント
System 表領域
その他の表領域
Users
Index
Temp
UNDO
system01.dbf
users01.dbf
indx01.dbf
temp01.dbf
undo01.dbf

セグメントとその構成要素

セグメント

一つの表や索引などのオブジェクトが使用する領域

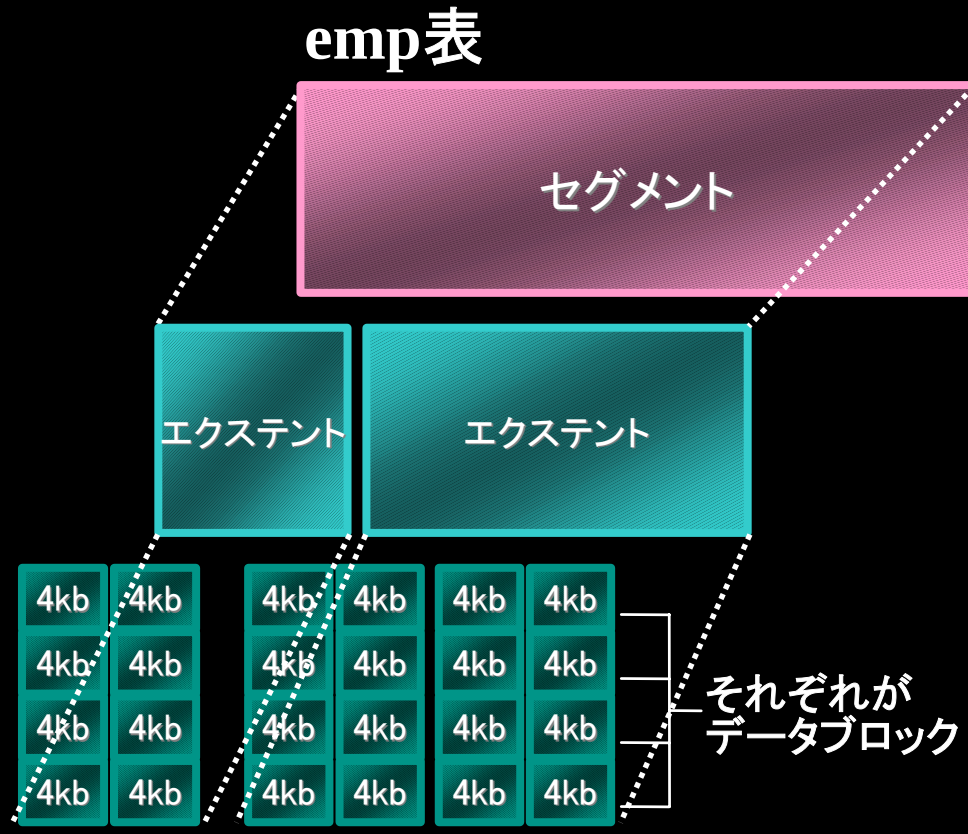
エクステント

セグメントの中の連続したデータブロックで構成される領域

データブロック

Oracleが一度にデータをI/O(読み書き)する最小の単位

OLTP系、DWH系でサイズを変える



System表領域

- 必須の表領域
  - データディクショナリ（データベース全体の情報）
  - ビュー、シノニム、順序などの情報
  - パッケージ、ストアドプロシージャ、ストアドファンクション
  - データベーストリガー
  - などが格納されている
- ユーザーのテーブルやインデックスを格納することも可能ですが、好ましくありません

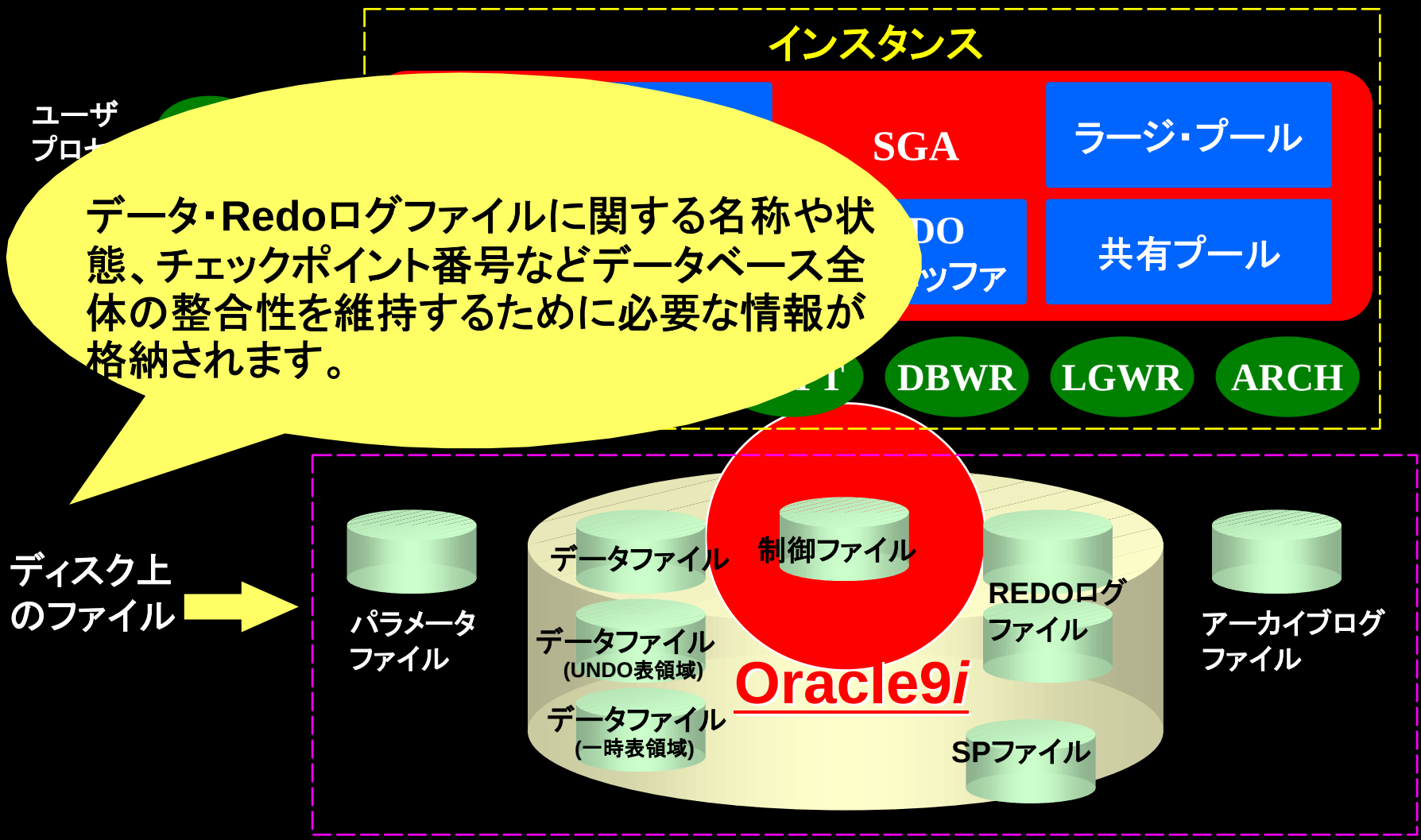
Undo表領域

- ロールバックデータの格納用の表領域
  - Oracle8iまでは「ロールバックセグメント」
  - ロールバックデータの管理
    - ロックを使用しない読み取り一貫性
    - トランザクションのロールバック
    - データベースのリカバリ
  - Oracle9iから自動管理
  - データの保持期間を設定
    - フラッシュバッククエリに使用

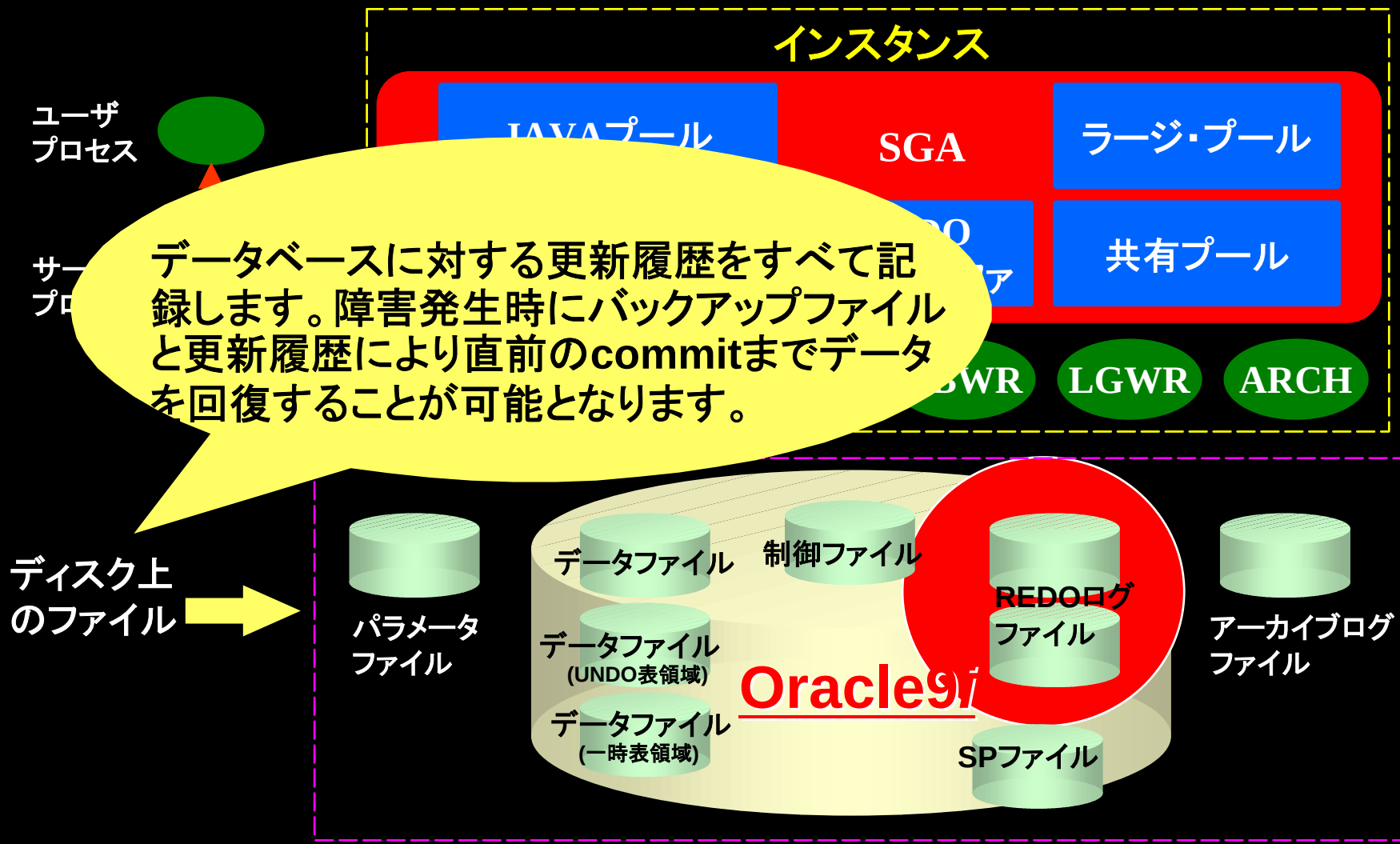
一時表領域

- ソート処理の時に使用する領域
  - DatabaseのDefault一時表領域の利用(Oracle9i)
  - Oracleのソート処理
    - 最初はメモリ(sort_area)を使用してソートする
    - メモリ領域でソートしきれない場合、一時表領域を使用する

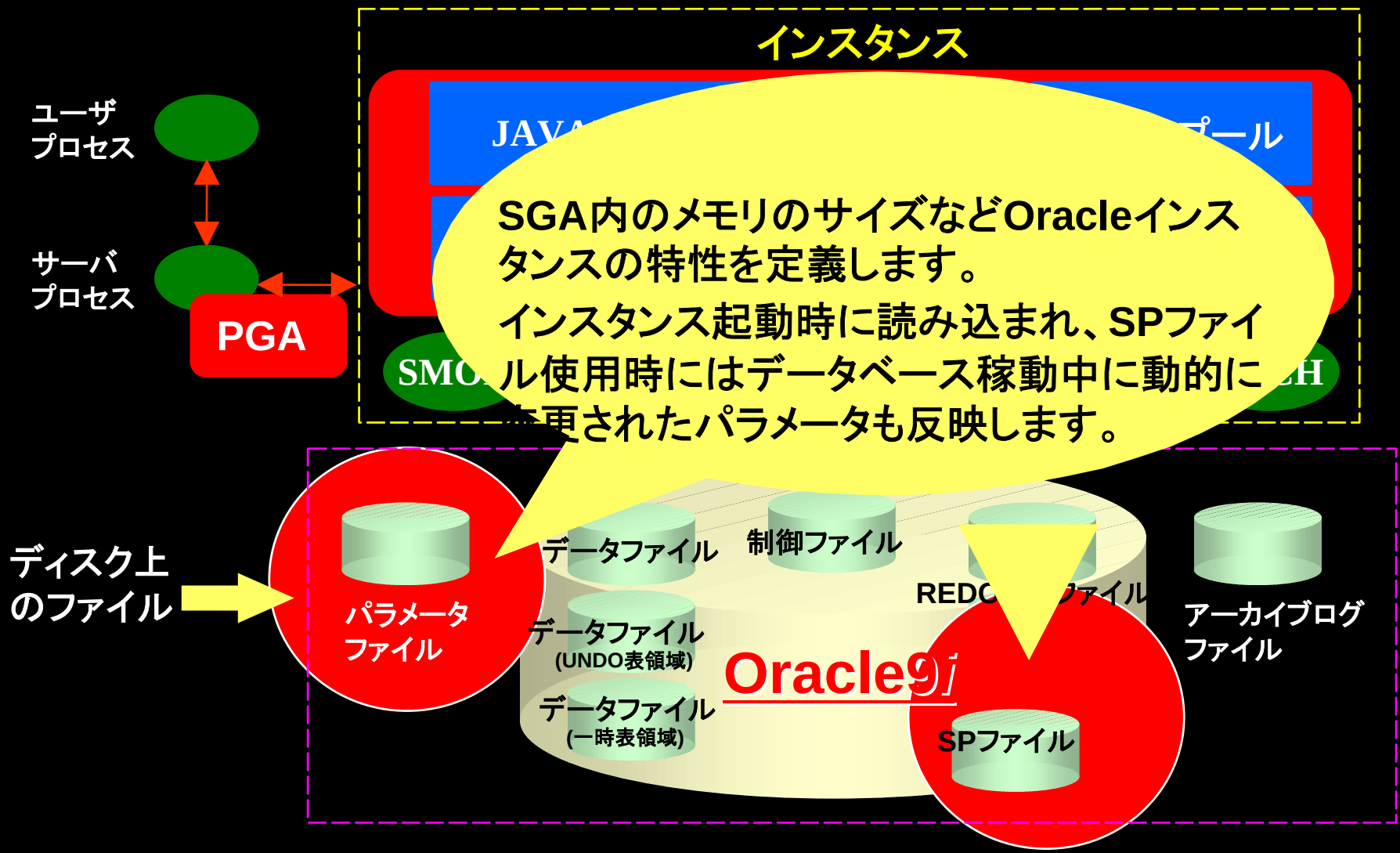
制御ファイル


インスタンス
SGA
ラージ・プール
共有プール
DBWR
LGWR
ARCH
データ・Redoログファイルに関する名称や状態、チェックポイント番号などデータベース全体の整合性を維持するために必要な情報が格納されます。
ディスク上のファイル
パラメータファイル
データファイル
制御ファイル
REDOログファイル
アーカイブログファイル
データファイル
(UNDO表領域)
Oracle9i
データファイル
(一時表領域)
SPファイル

REDOログファイル


インスタンス
ユーザ
プロセス
SGA
ラージ・プール
共有プール
LGWR
ARCH
データベースに対する更新履歴をすべて記録します。障害発生時にバックアップファイルと更新履歴により直前のcommitまでデータを回復することが可能となります。
ディスク上
のファイル
パラメータ
ファイル
データファイル
制御ファイル
REDOログ
ファイル
アーカイブログ
ファイル
データファイル
(UNDO表領域)
Oracle9i
データファイル
(一時表領域)
SPファイル

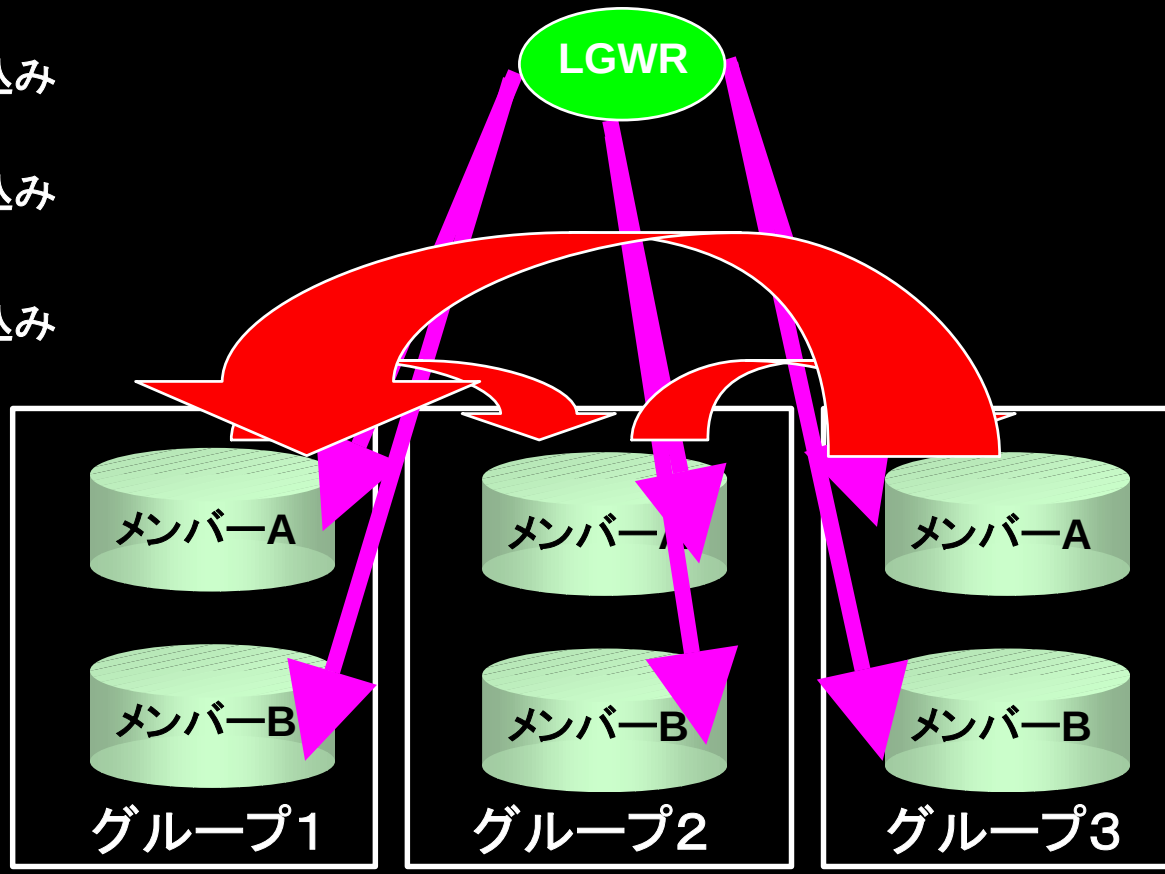
パラメータファイル


インスタンス
ユーザ
プロセス
サーバ
プロセス
PGA
SGA内のメモリのサイズなどOracleインスタンスの特性を定義します。
インスタンス起動時に読み込まれ、SPファイル使用時にはデータベース稼動中に動的に変更されたパラメータも反映します。
ディスク上
のファイル
パラメータ
ファイル
データファイル
制御ファイル
REDO ログファイル
アーカイブログ
ファイル
データファイル
(UNDO表領域)
データファイル
(一時表領域)
Oracle9i
SPファイル

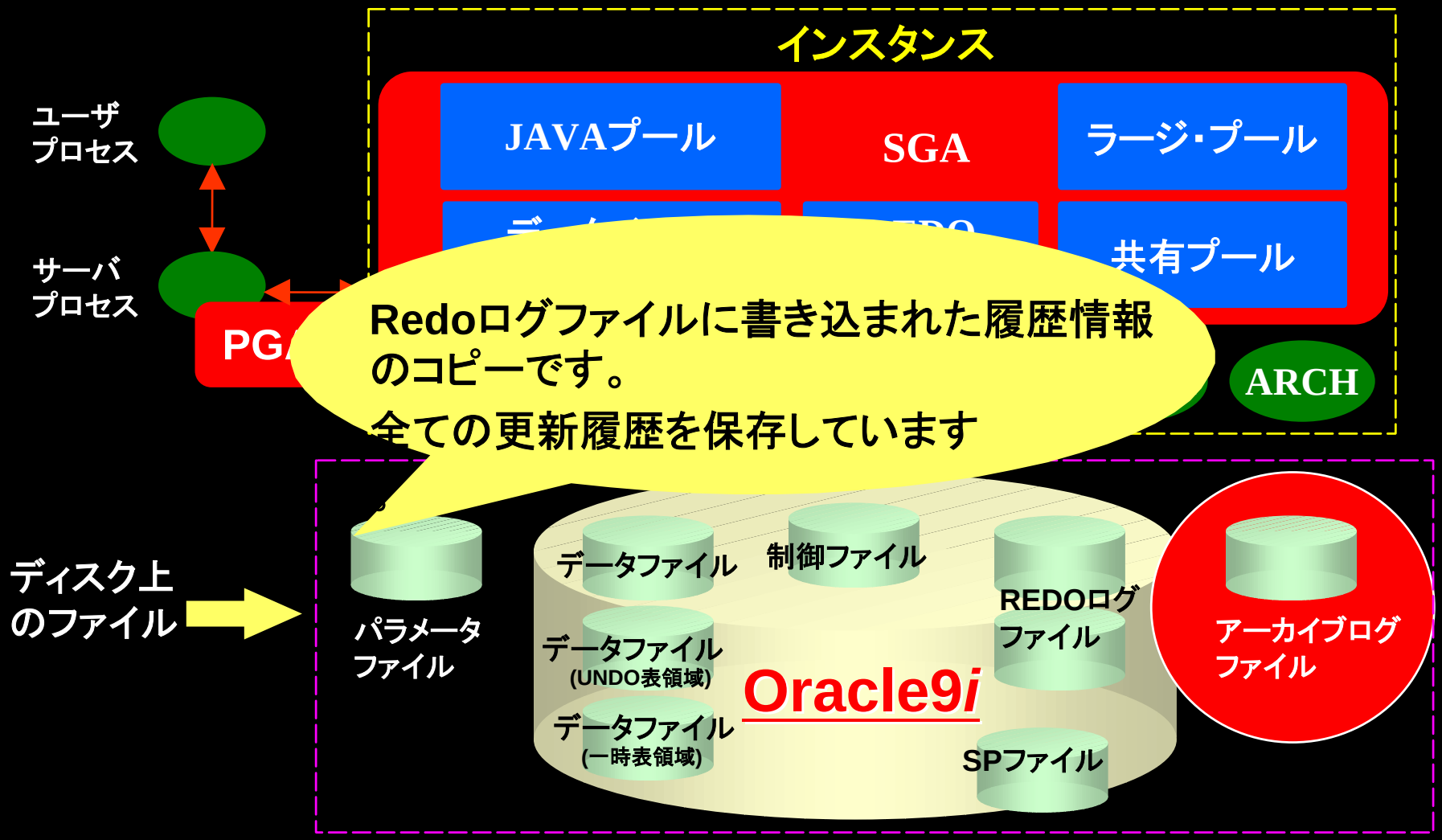
REDOログファイル

ログ書込みの動作

1. グループ1のメンバーに書込み
   → ログスイッチが発生
2. グループ2のメンバーに書込み
   → ログスイッチが発生
3. グループ3のメンバーに書込み
   → ログスイッチが発生
4. グループ1のメンバーに書込み

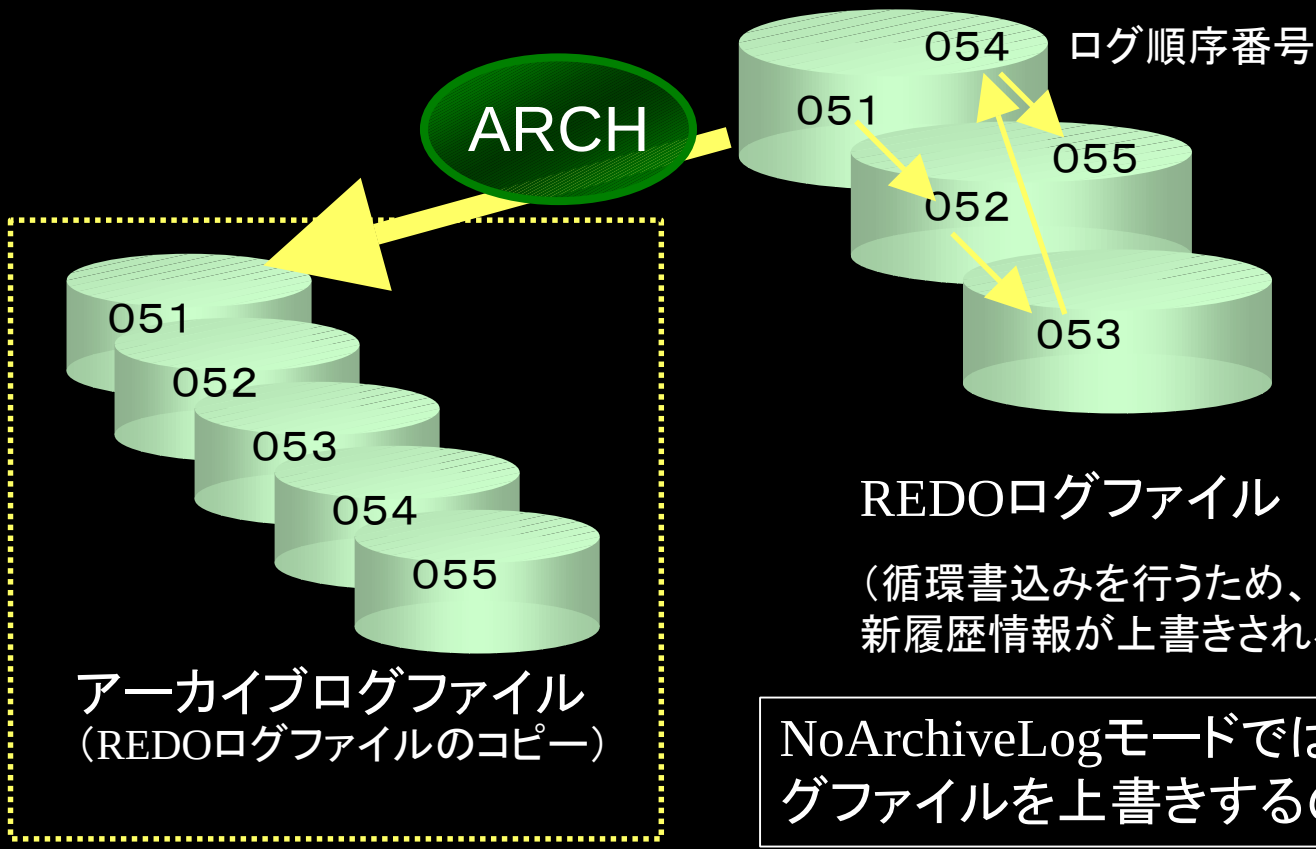


アーカイブログファイル



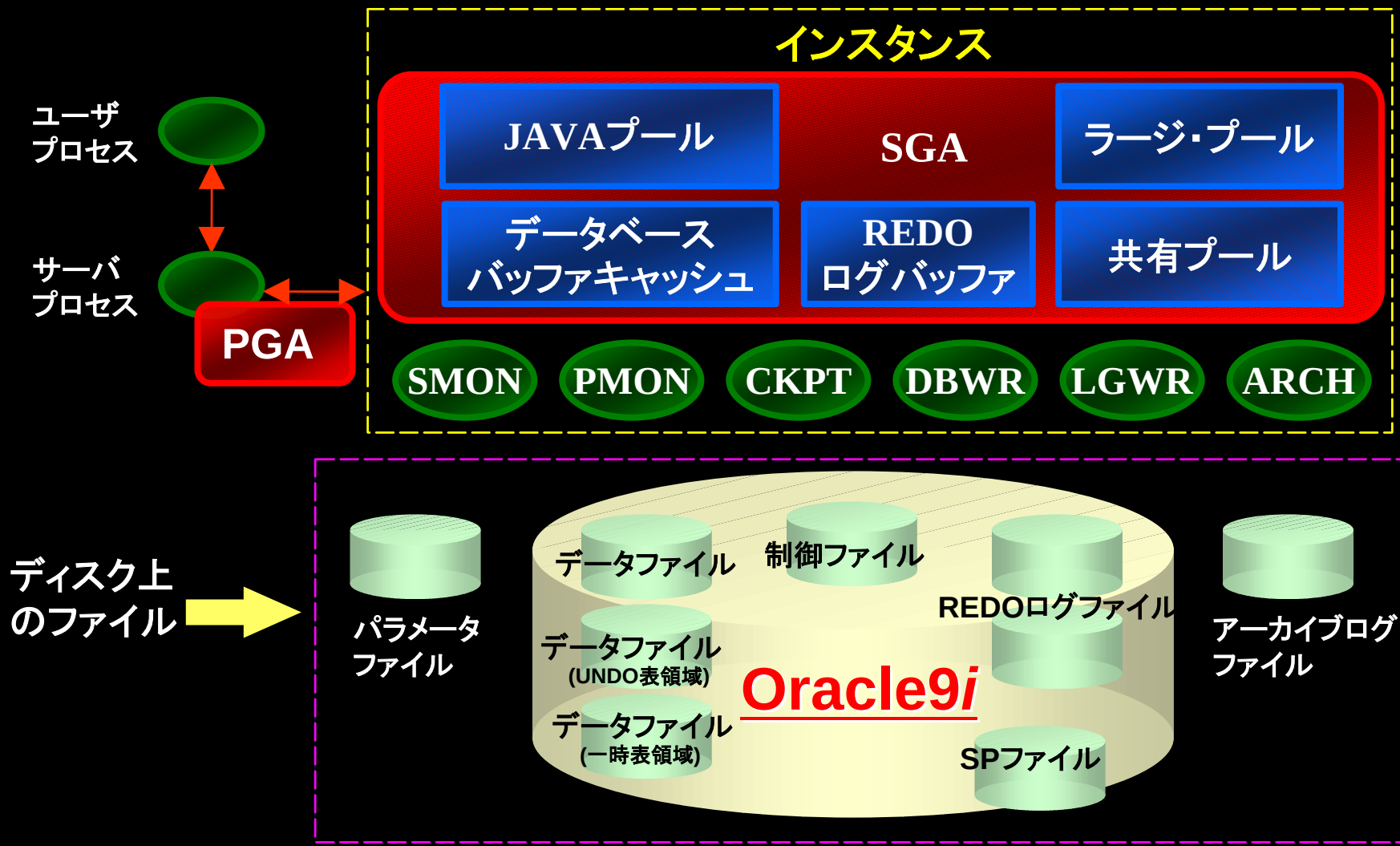
アーカイブログファイル

ArchiveLogモードでの運用



NoArchiveLogモードでは、Redoログファイルを上書きするのみ。

アーキテクチャ概要


インスタンス
ユーザ
プロセス
サーバ
プロセス
PGA
JAVAプール
SGA
ラージ・プール
データベース
バッファキャッシュ
REDO
ログバッファ
共有プール
SMON
PMON
CKPT
DBWR
LGWR
ARCH
ディスク上
のファイル
パラメータ
ファイル
データファイル
制御ファイル
REDOログファイル
データファイル
(UNDO表領域)
Oracle9i
データファイル
(一時表領域)
SPファイル
アーカイブログ
ファイル

SGA(System Global Area)

Javaプール/ラージプール/共有プール

ディクショナリ情報/解析済みSQL文などを保持するメモリ

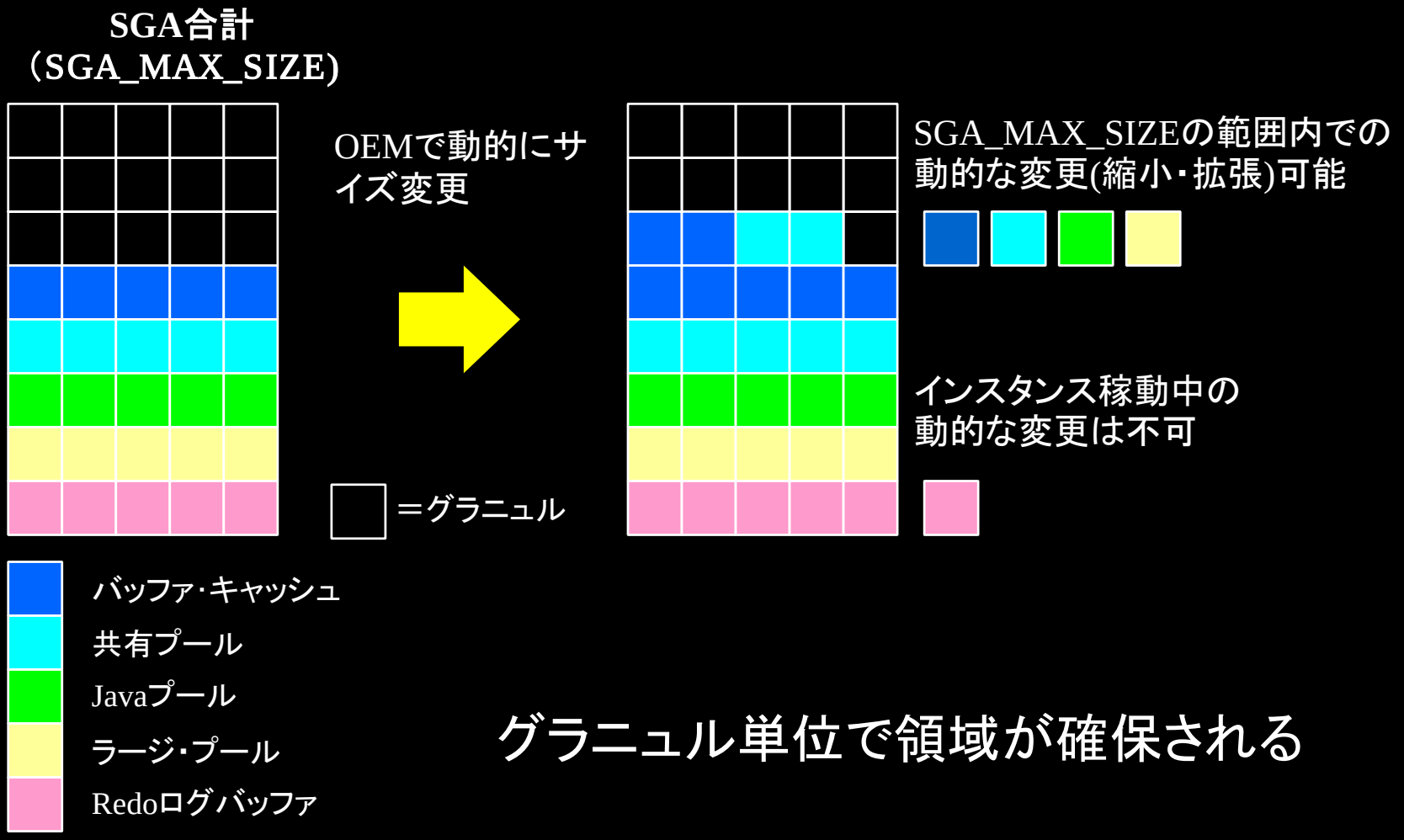
データベースバッファキャッシュ

データファイルから読み込んだデータを保持するメモリ

Redoログバッファ

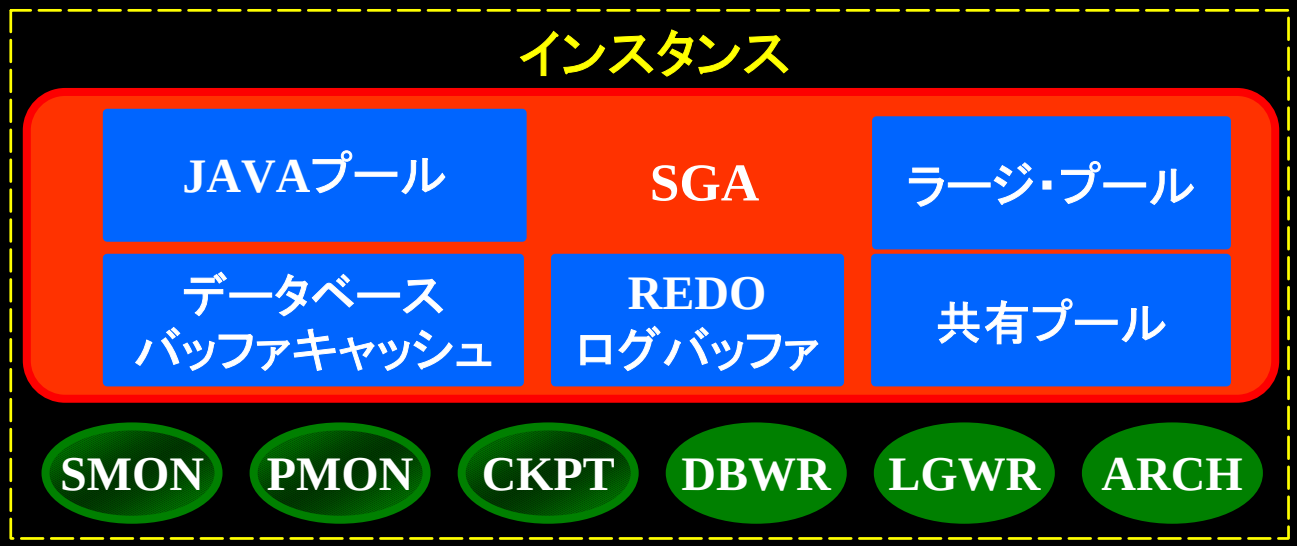
データベースに対する更新履歴の一時格納メモリ

SGA(System Global Area)



グラニュル単位で領域が確保される

バックグラウンド・プロセス



SMON

システムを監視し、起動時にインスタンス回復などを行う

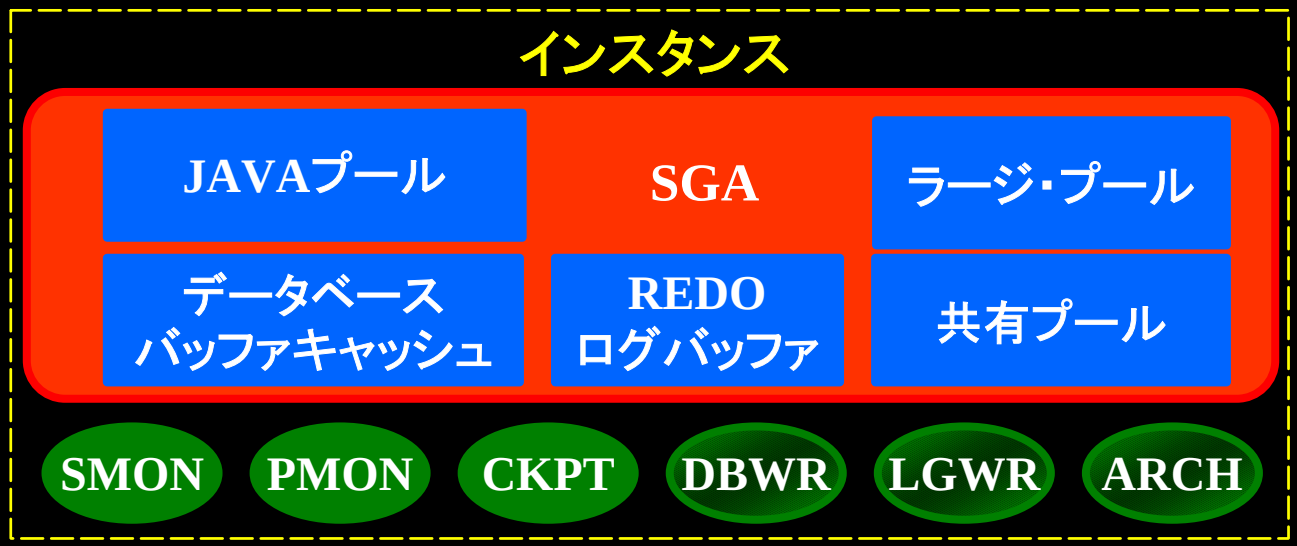
PMON

プロセス群を監視し、異常終了した接続をクリーンアップ

CKPT

チェックポイント発生時にDBWRへシグナルを送る

バックグラウンド・プロセス



DBWR

データベースバッファキャッシュ上の更新済みデータをデータファイルに書込む

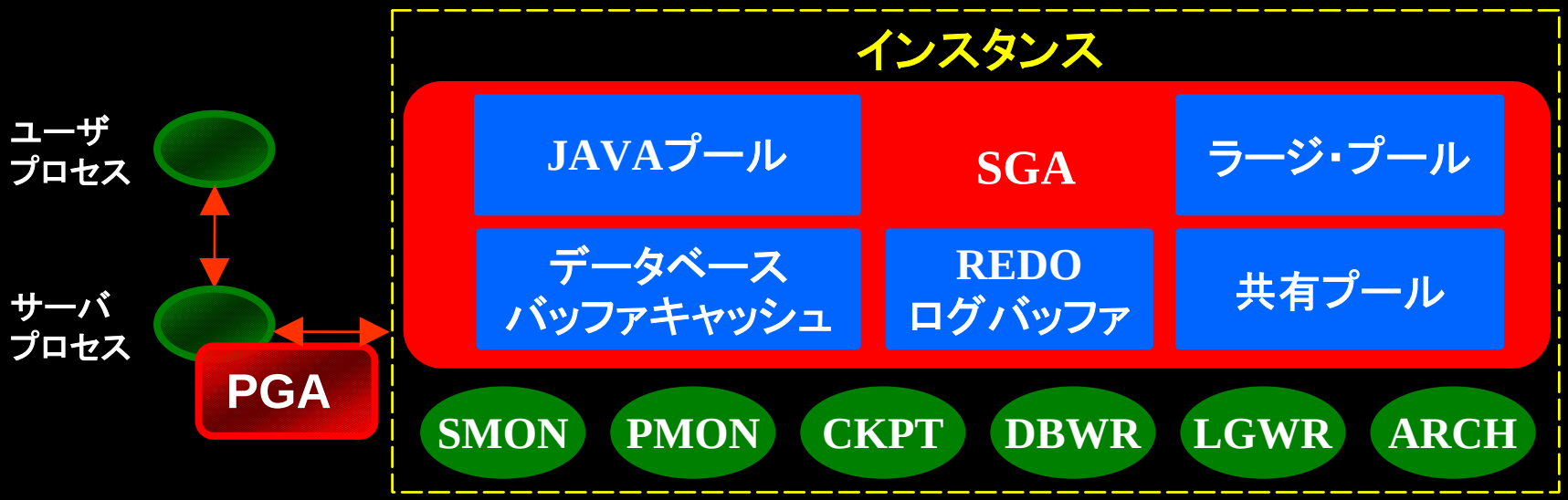
LGWR

Redoログバッファ上の更新記録をRedoログファイルに書込む

ARCH

アーカイブログファイルにRedoログファイルのコピーを書込む

ユーザプロセスとサーバプロセス

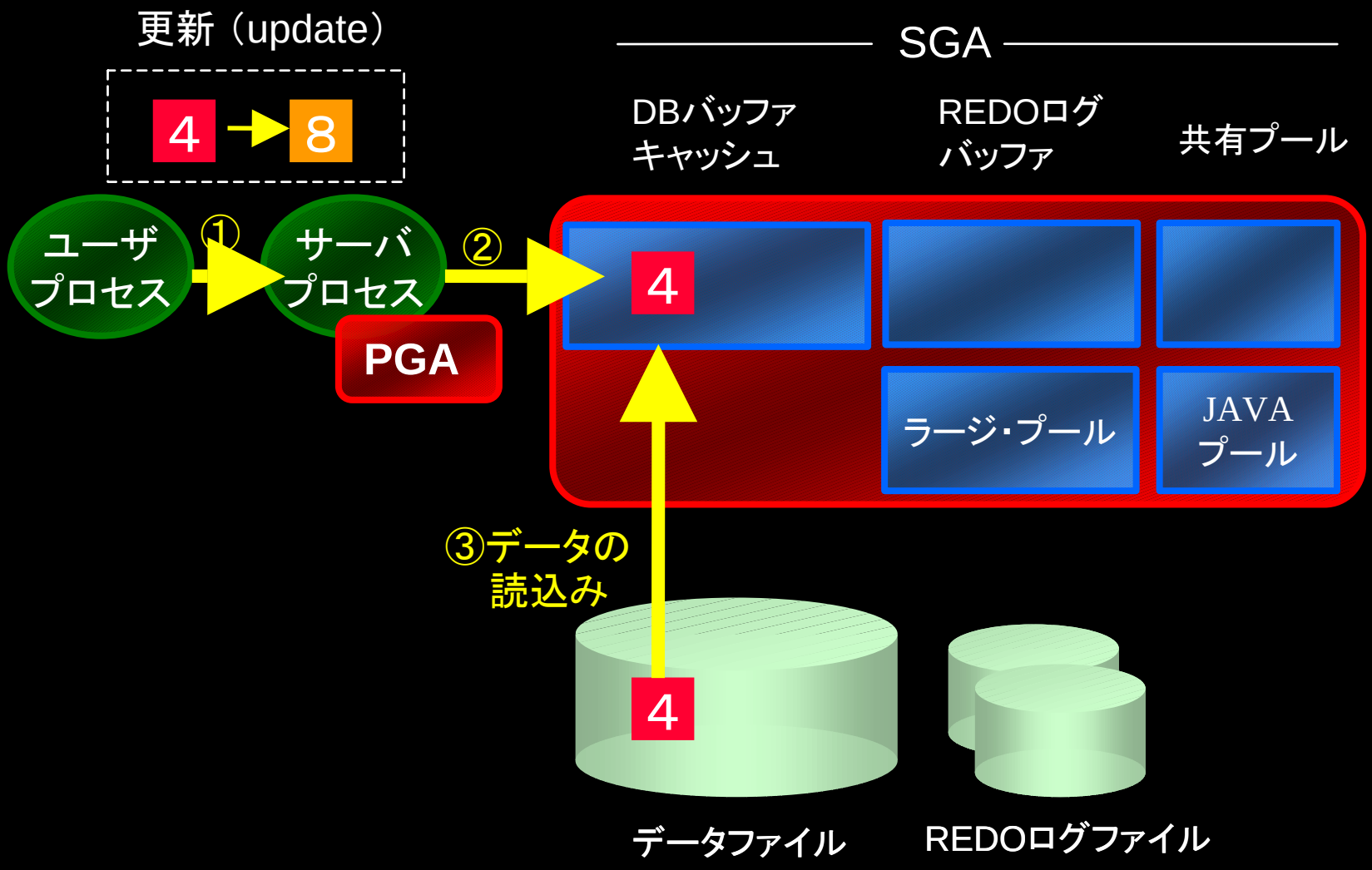


サーバプロセスはインスタンスに含まれず、インスタンスとユーザプロセスの間の命令のやり取りを行い、基本的に一つの接続ごとに一つ必要となる

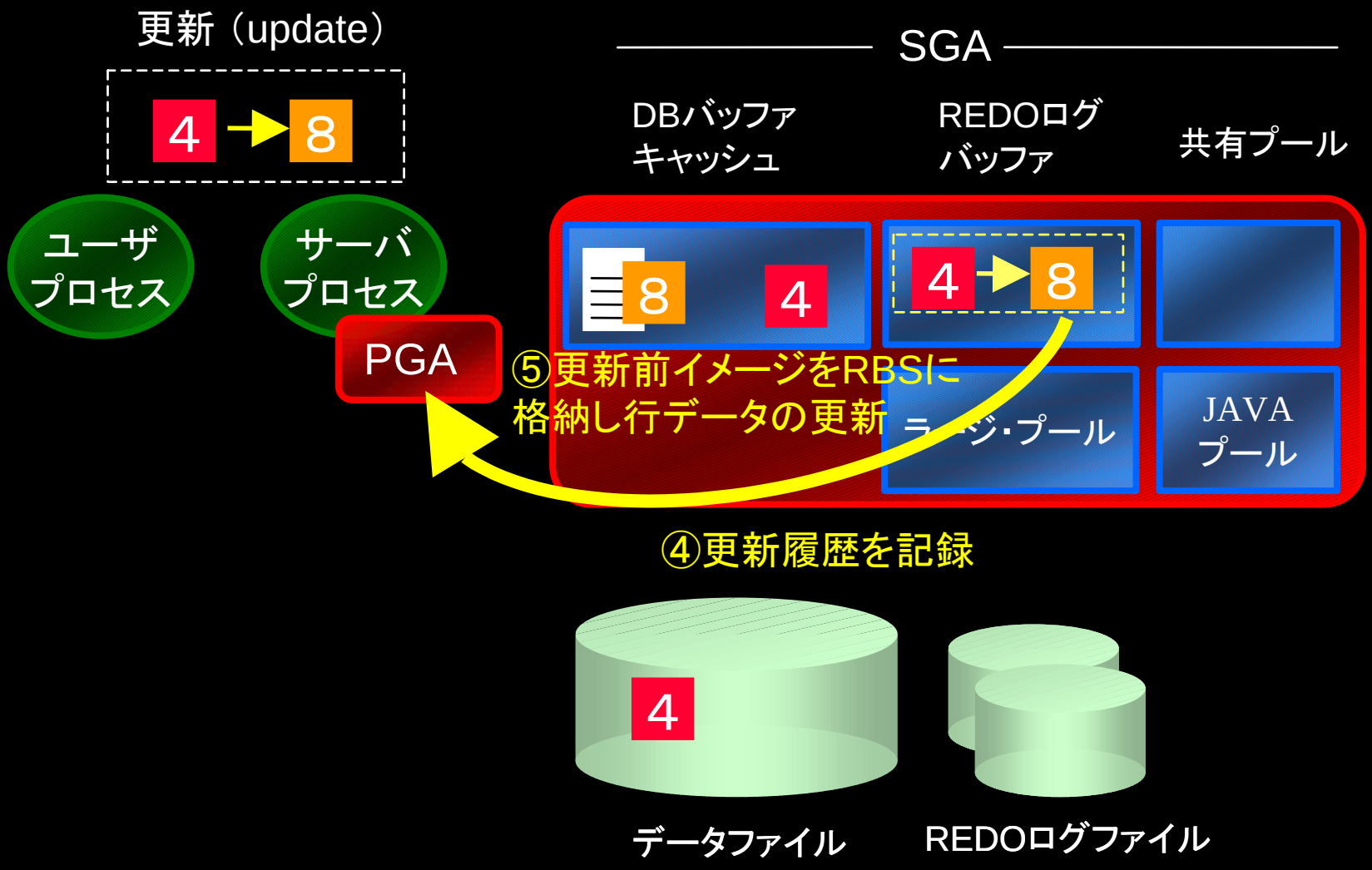
PGA(Program Global Area)

サーバプロセス毎に存在し、処理結果のソート用の領域や接続情報などを保持
データベース全体で使用するPGAのサイズの指定が可能。

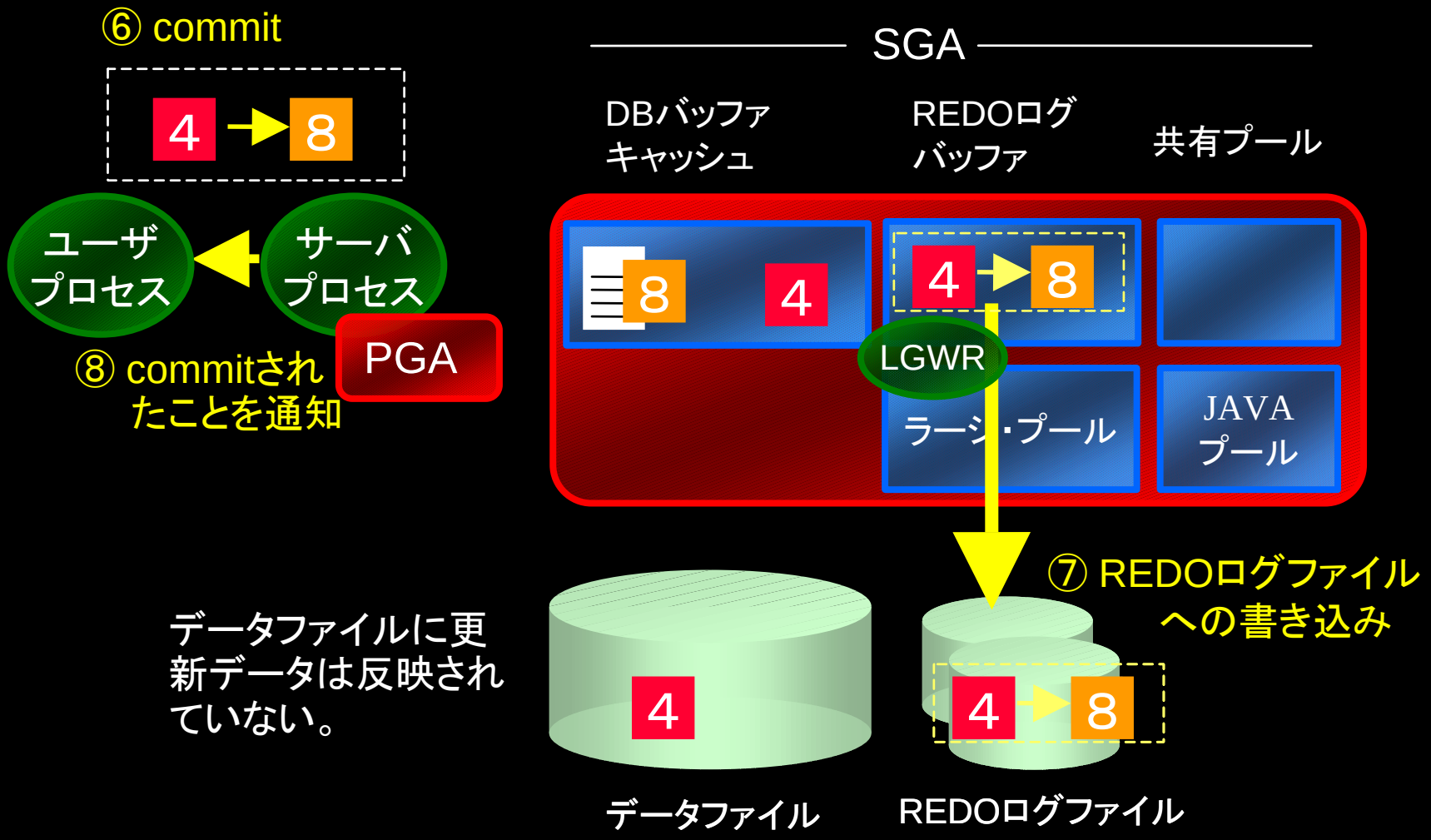
データ更新時の動作（読込み）



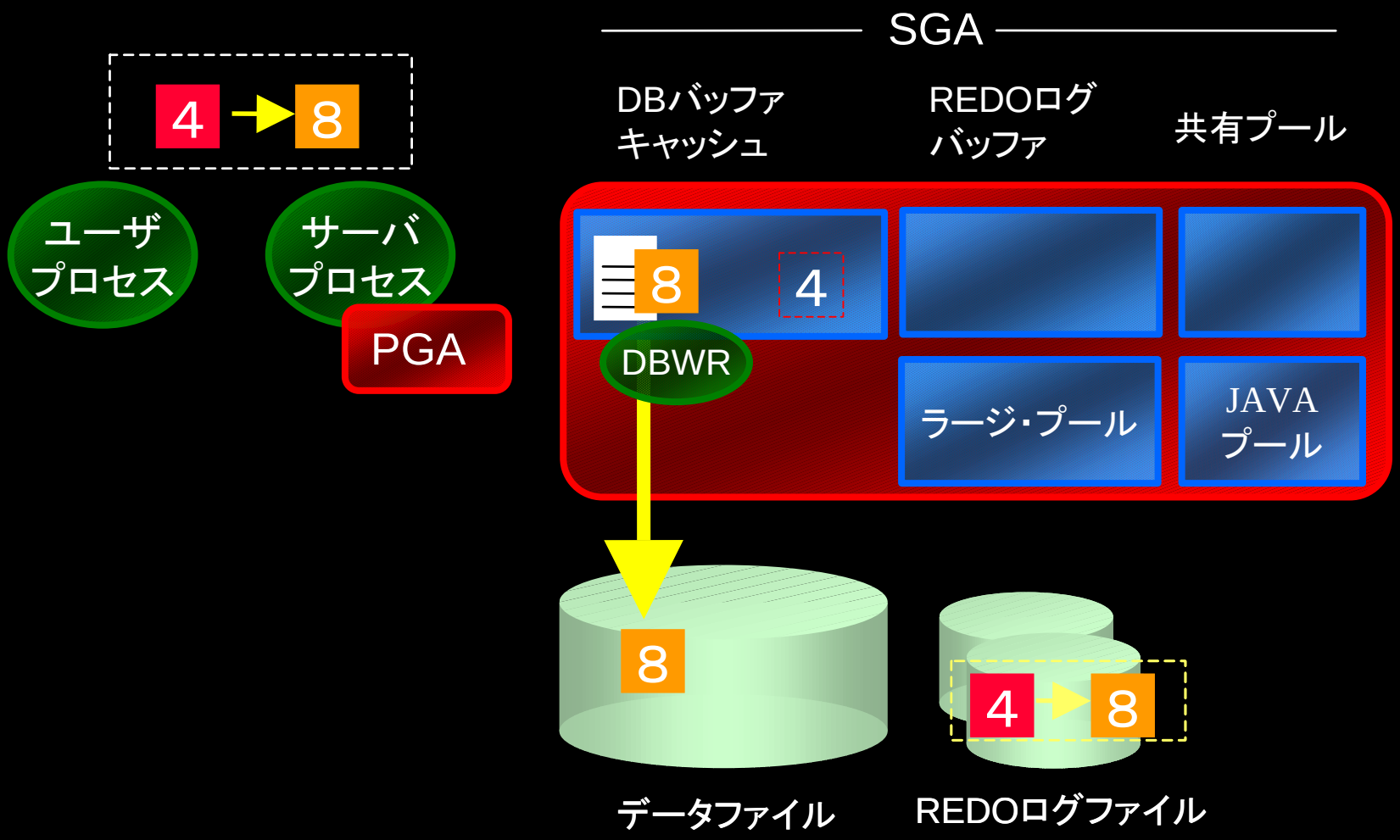
データ更新時の動作（更新履歴の記録）



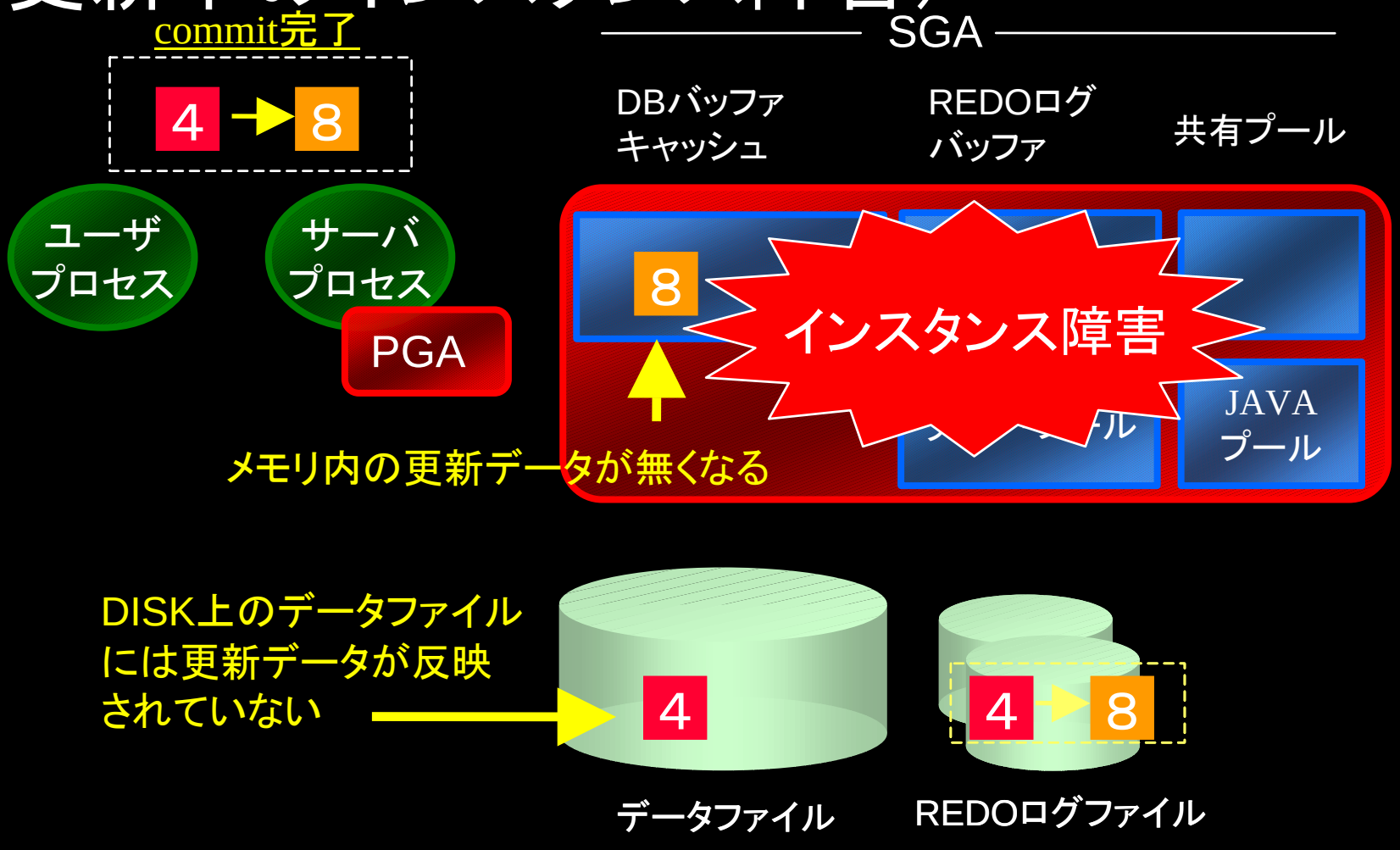
データ更新時の動作（commit）



データ更新時の動作（遅延書込み）

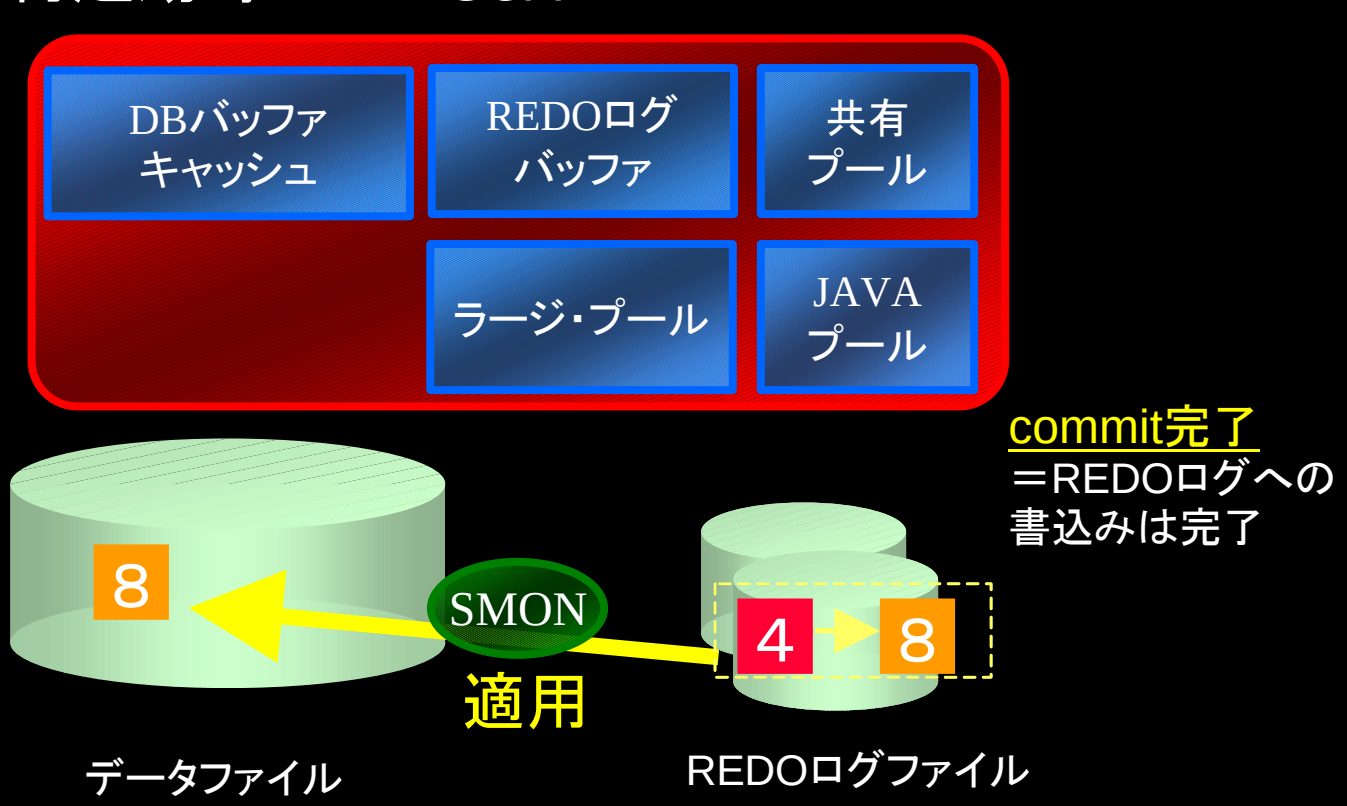


データ更新時の動作（更新中のインスタンス障害）

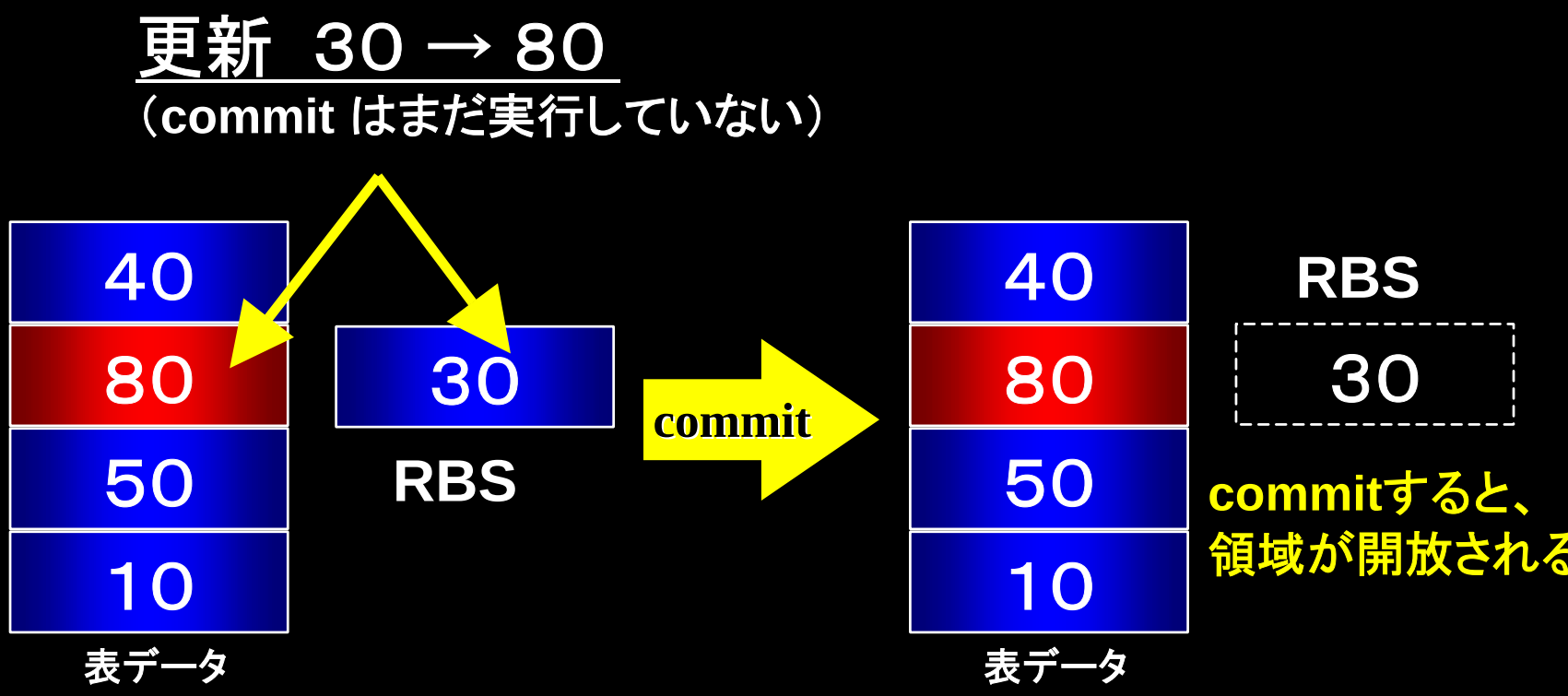


データ更新時の動作（更新中のインスタンス障害）

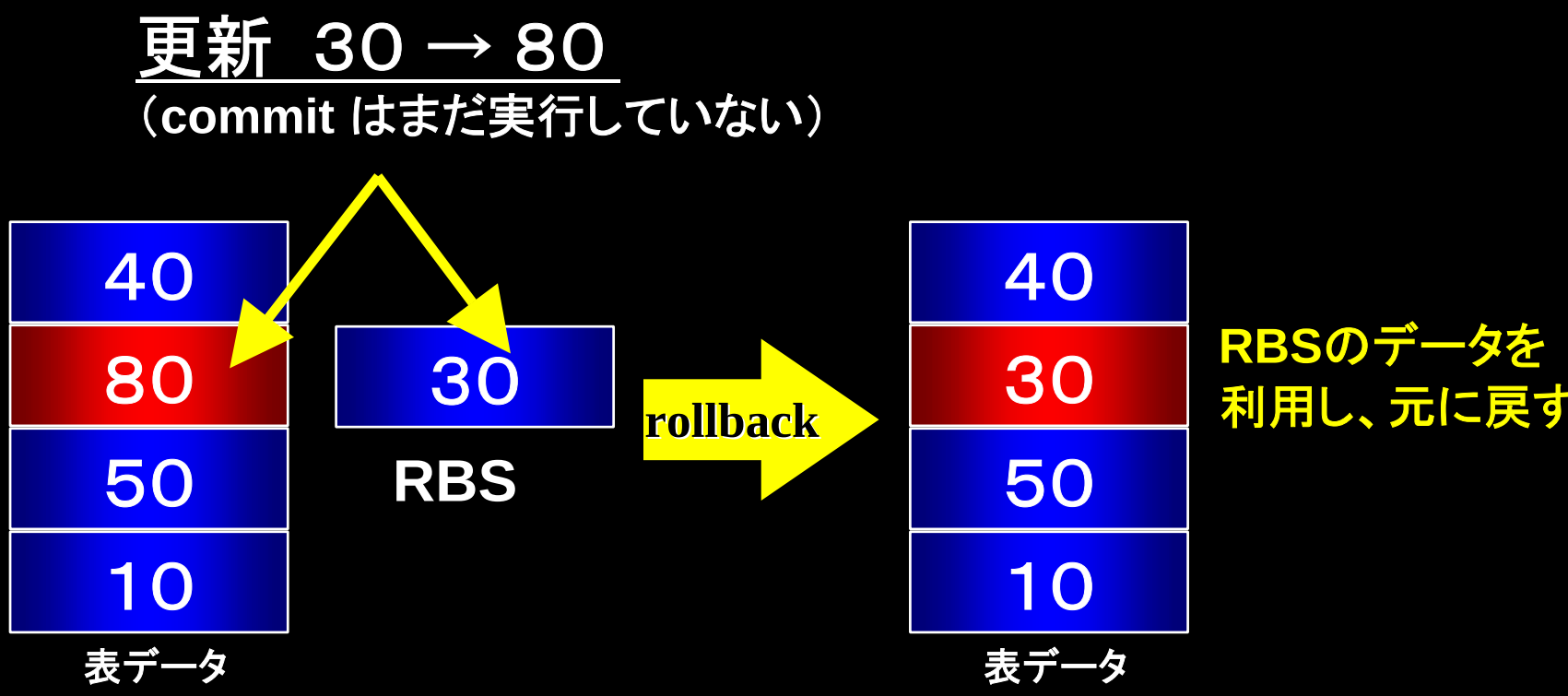
インスタンス再起動時



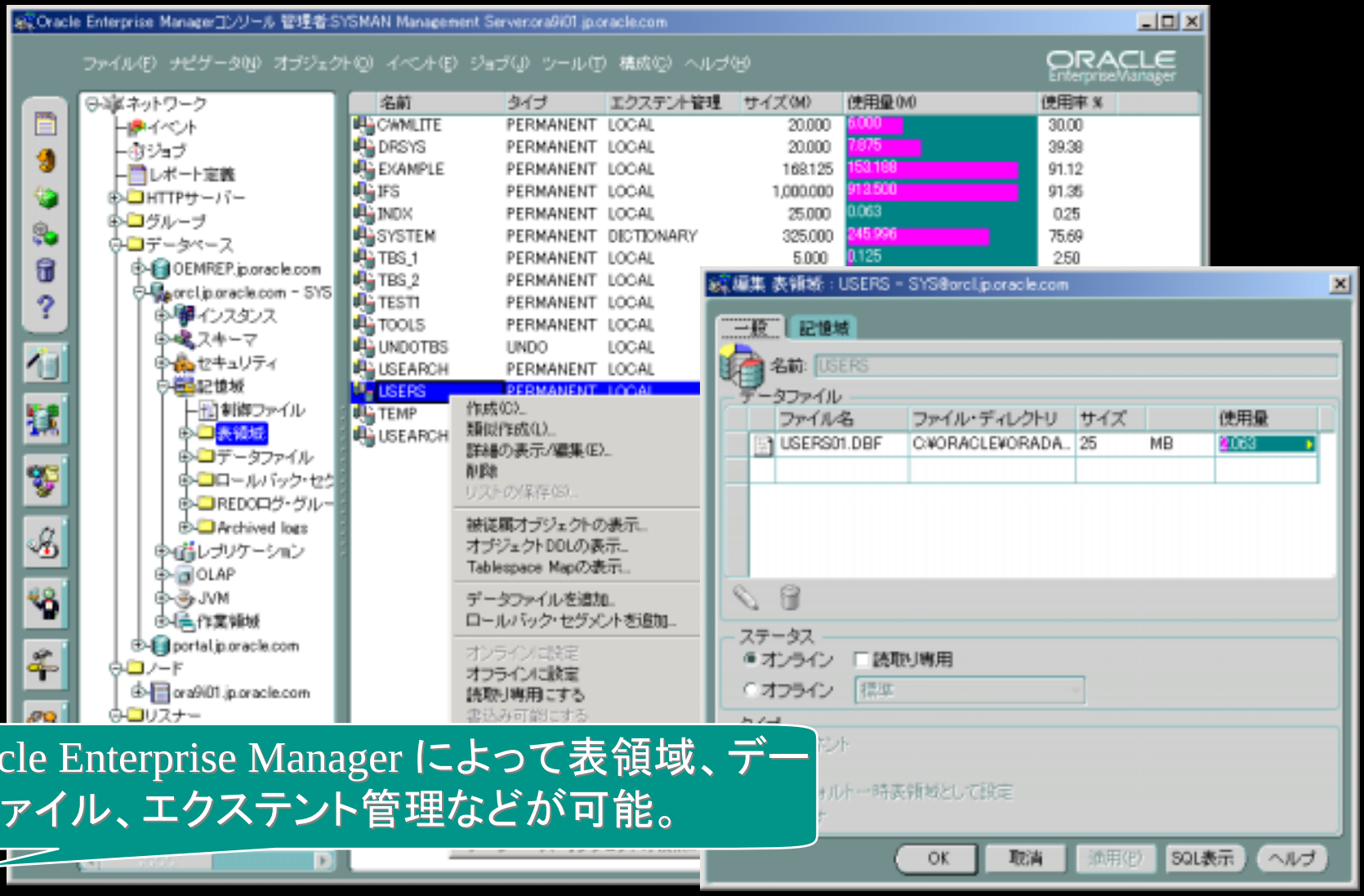
ロールバックセグメント（commit）



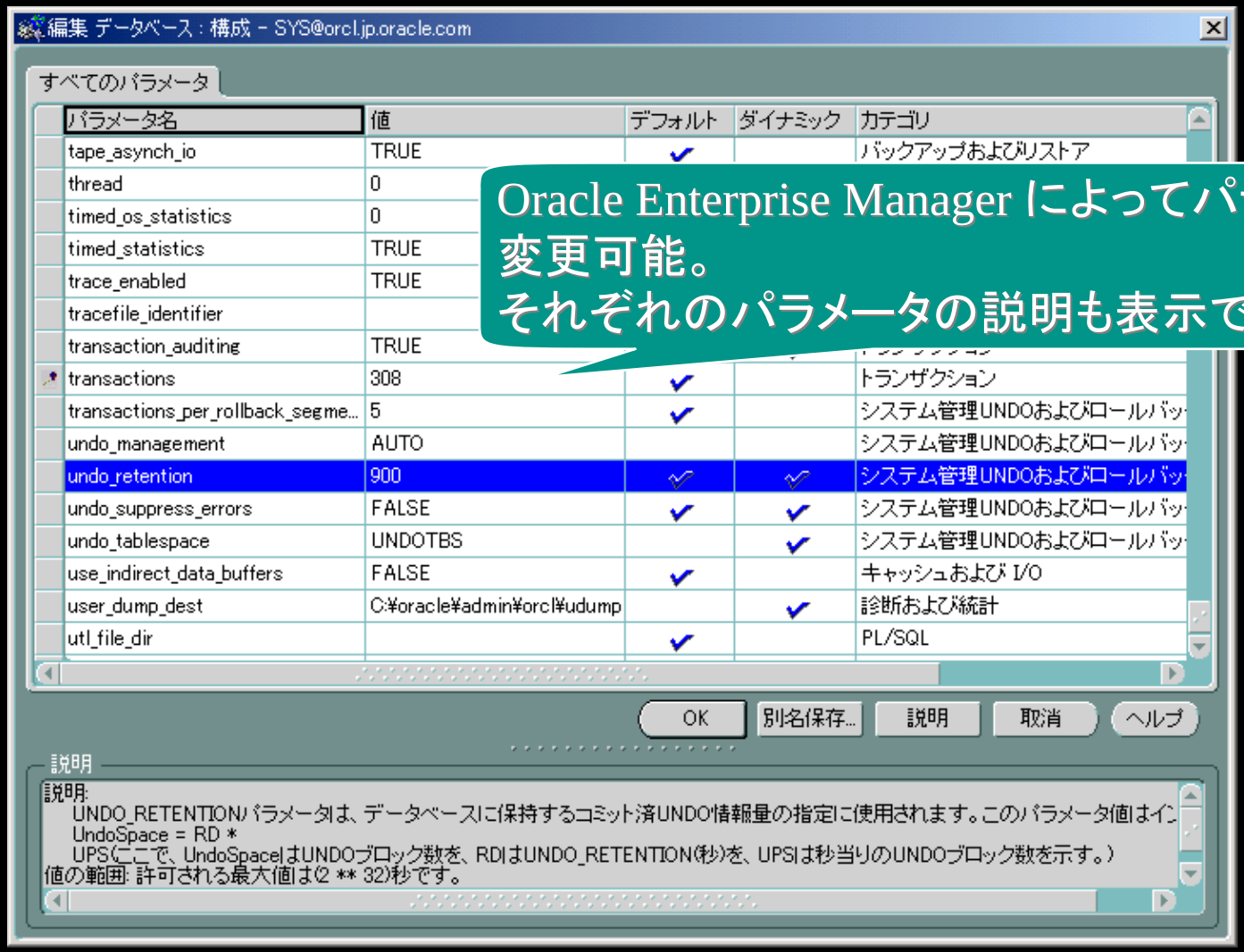
ロールバックセグメント（rollback）



参考:GUIによる記憶領域管理

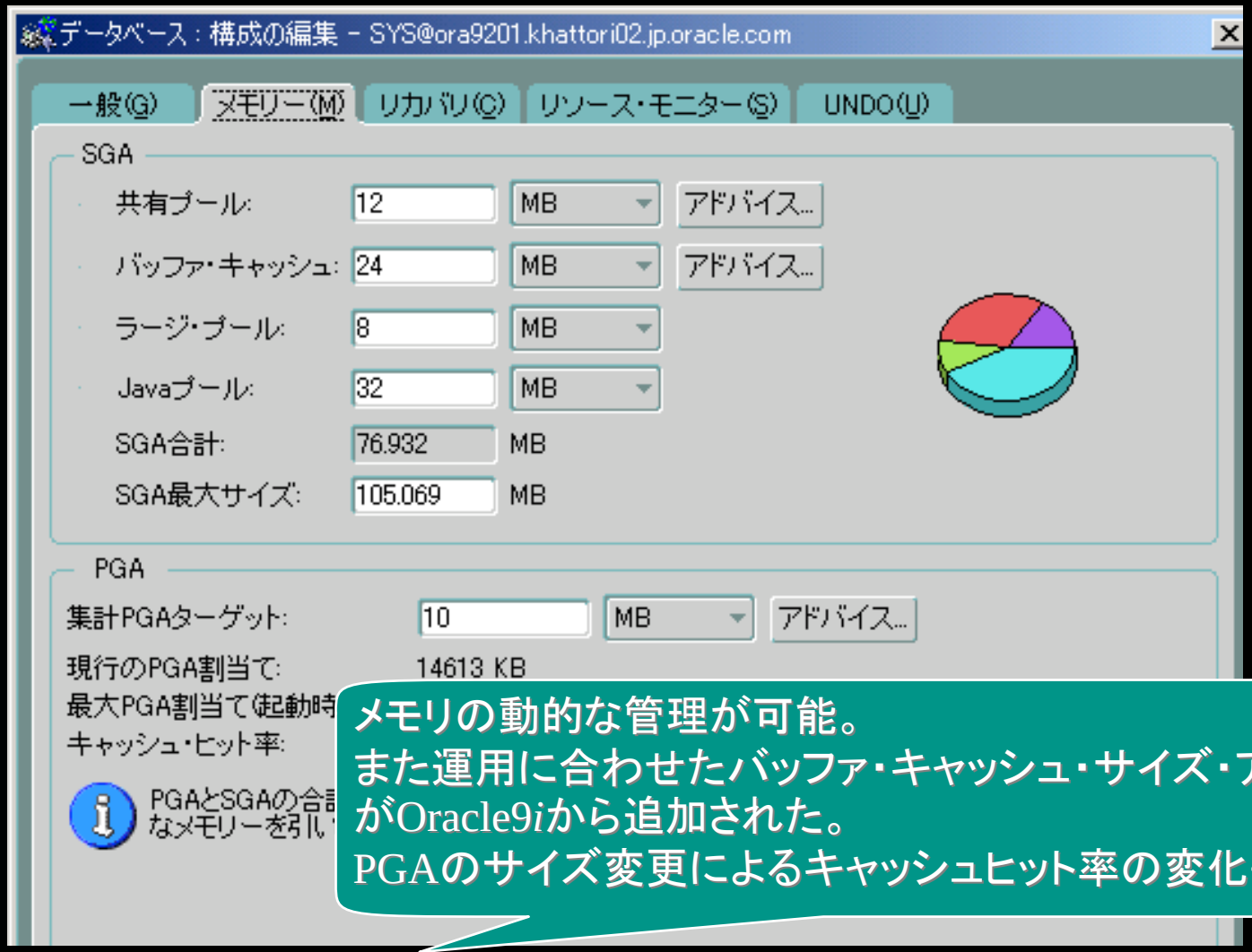


参考：GUIによるパラメータ変更



Oracle Enterprise Manager によってパラメータを変更可能。
それぞれのパラメータの説明も表示できる。

参考：GUIによるメモリ管理

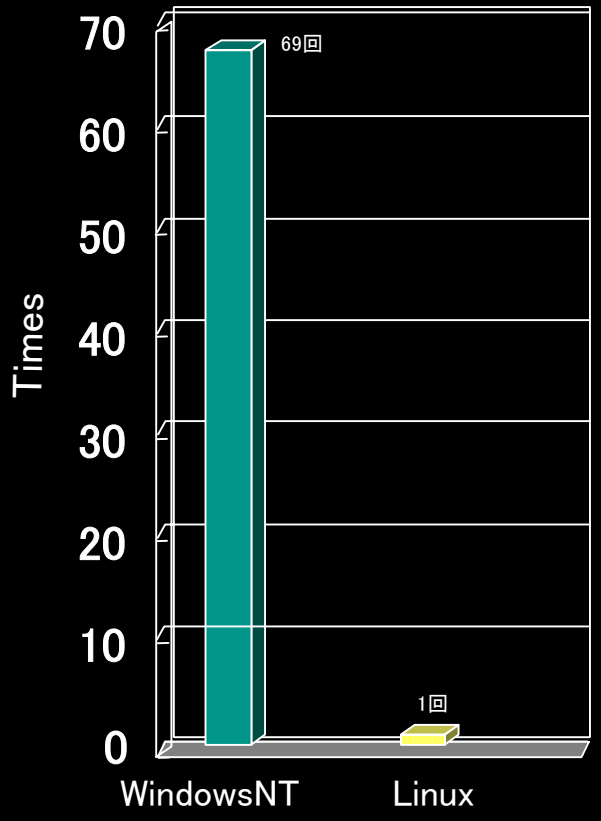


Linuxのメリット

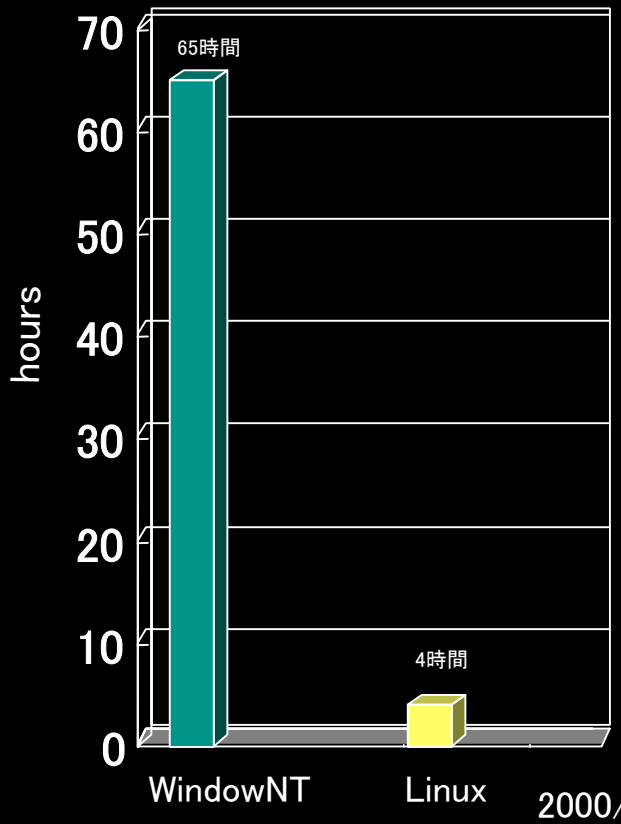
- 高い信頼性
  - 連続運転に強い（メモリ・リーク等による再起動は不要）
  - 障害原因の究明可能
- 高い性能
  - －Linuxカーネルの急速な進化
- 圧倒的に安い導入コスト

高い信頼性・安全性

サーバー障害回数/1年



サーバーダウンタイム/1年



2000/01 Bloor Resaerch

http://gnet.dhs.org/stories/bloor.php3

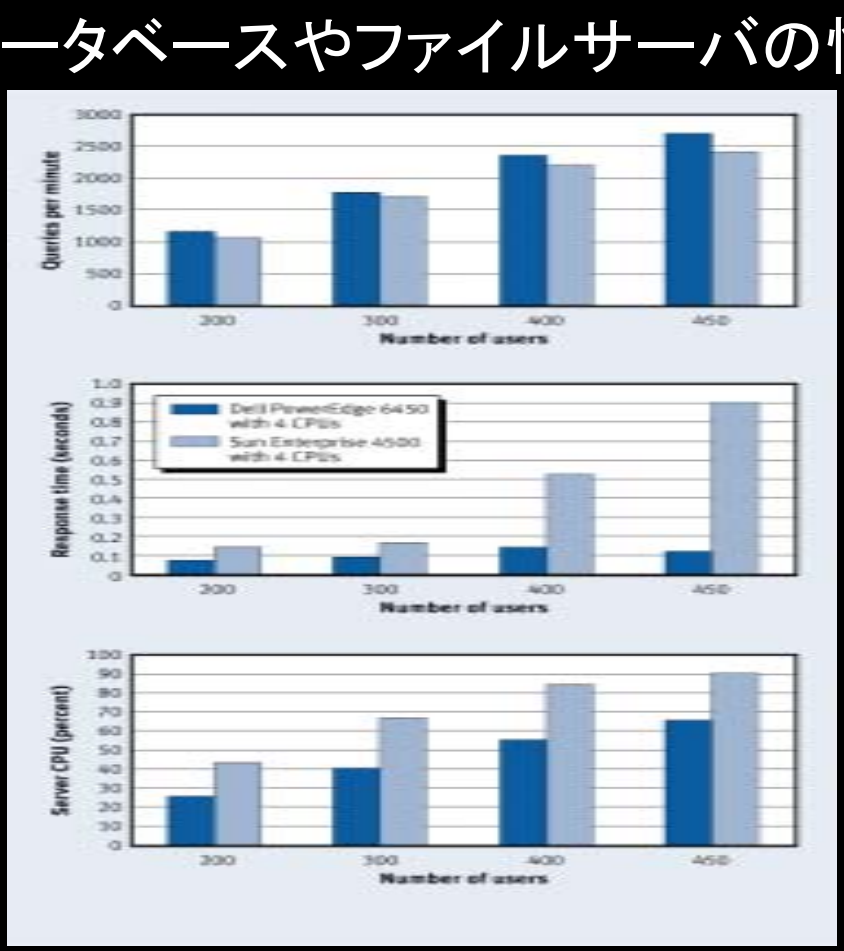
高い信頼性・安全性　～Linuxの特徴～

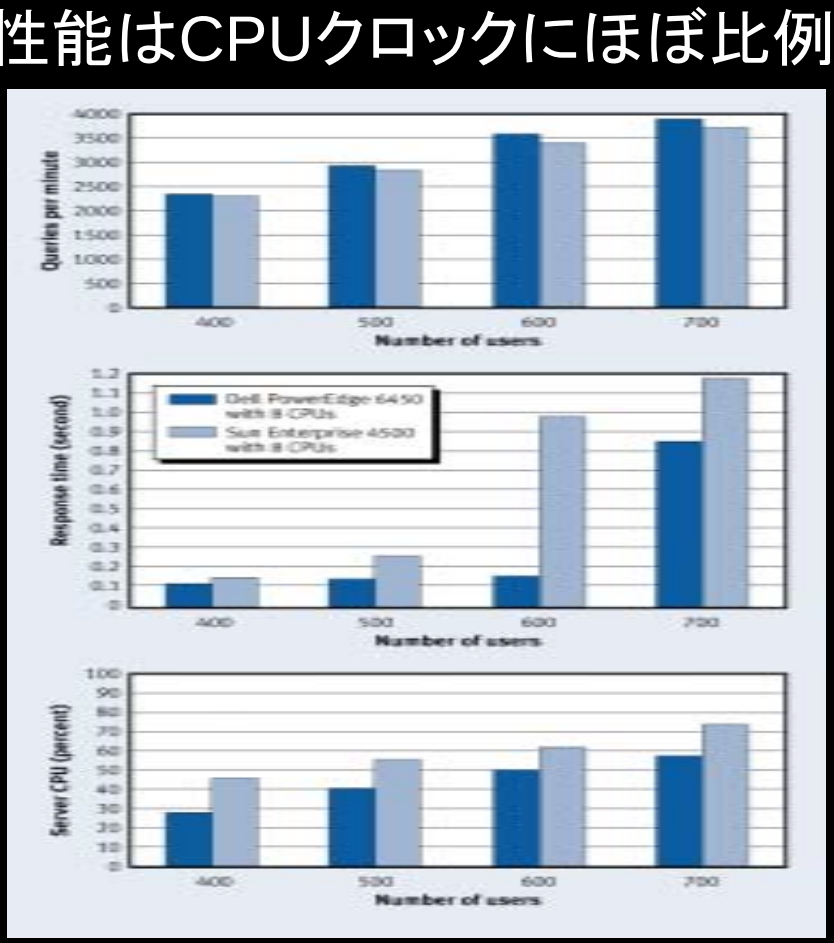
- UNIXクローン

- マルチタスク
  - 様々な処理を平行して行うことが可能
- マルチユーザ
  - 複数ユーザの同時使用が可能
- 進化の早さ
  - セキュリティーフィックス
  - exec-shield等、新しい試み

データベースプラットフォームとして高い性能
～INTEL vs SPARC on Oracle～

データベースやファイルサーバの性能はCPUクロックにほぼ比例





データベースプラットフォームとして高い性能
～Linux vs Windows～

最新データベースベンチマーク結果
TPC-Cの結果(2002/09/16現在)

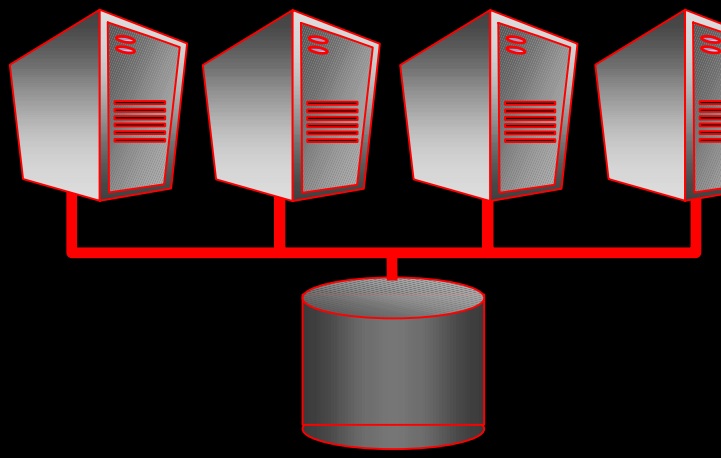
Oracle9*i*DB Release2
on Linuxでの結果

TPC-C スループット	138,362
価格 / tpmC	17.21 US $

Oracle9*i*DB Release2
on Windowsでの結果

TPC-C スループット	**137,260**
価格 / tpmC	18.46 US $

http://www.tpc.org

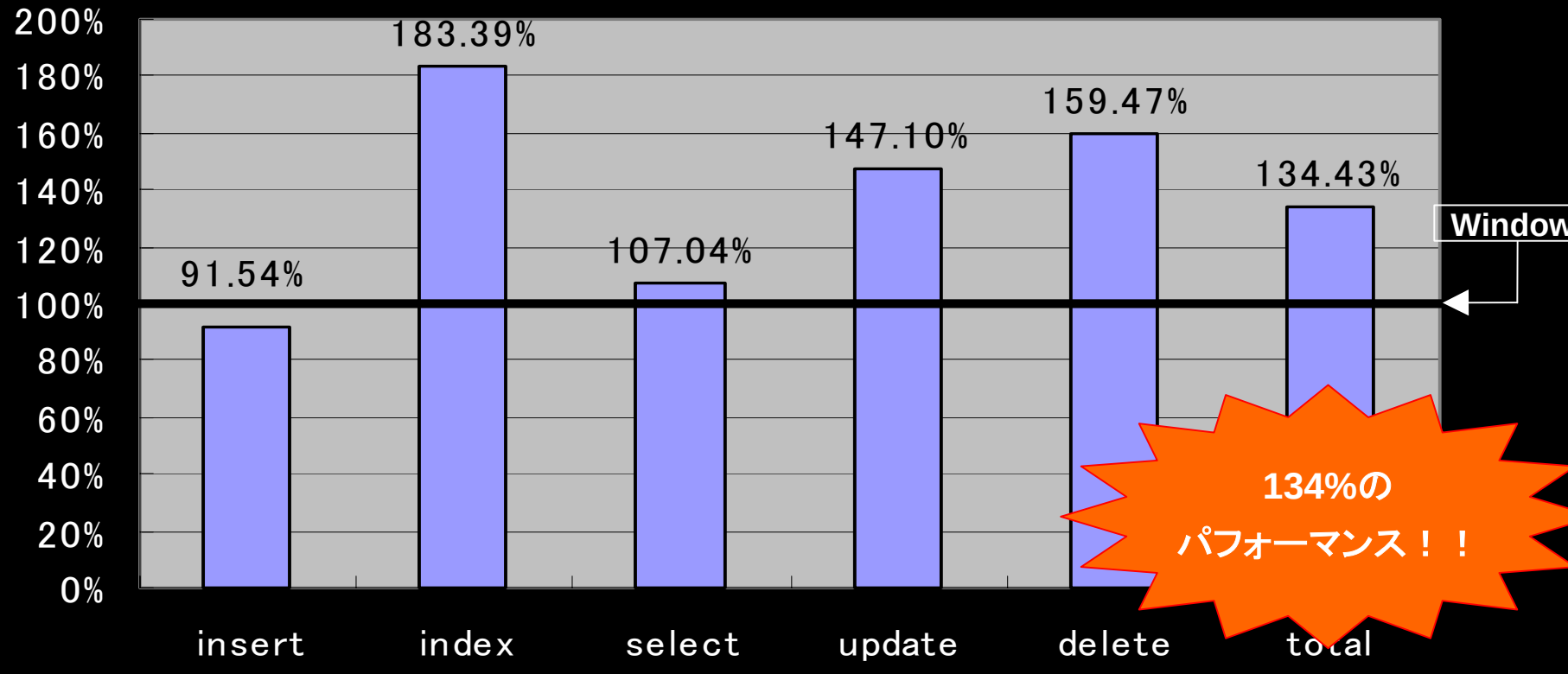


ハードウェア：
HP DL580 4CPU x 8ノード

データベースプラットフォームとしても高い性能
～Linux vs Windows～

Oracle9i DB(Linux vs Windows)

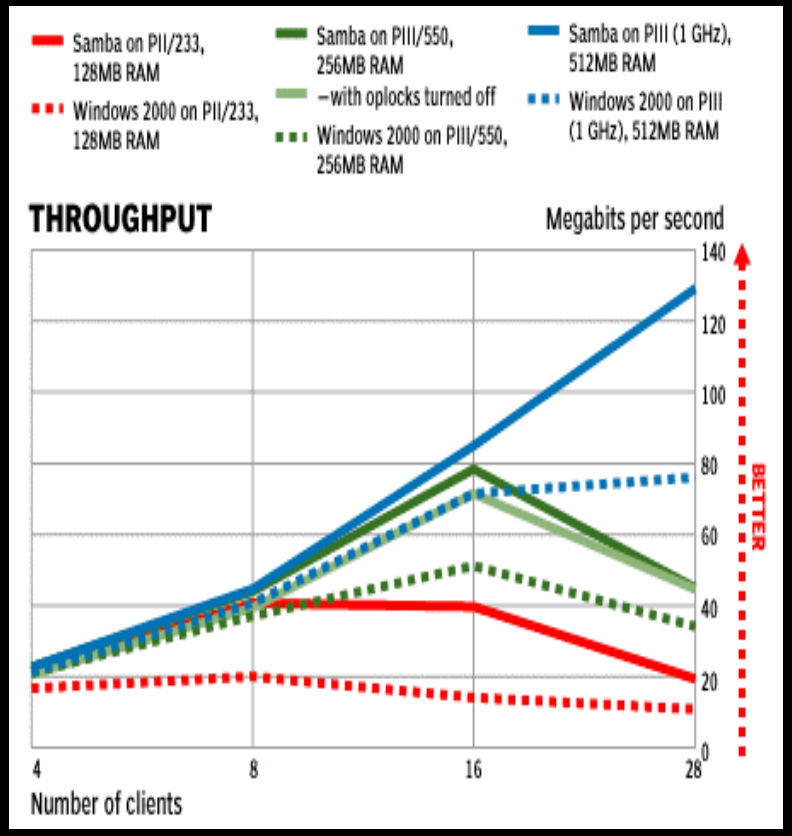
Windowsの処理速度に対する
MIRACLE LINUXの処理速度の割合

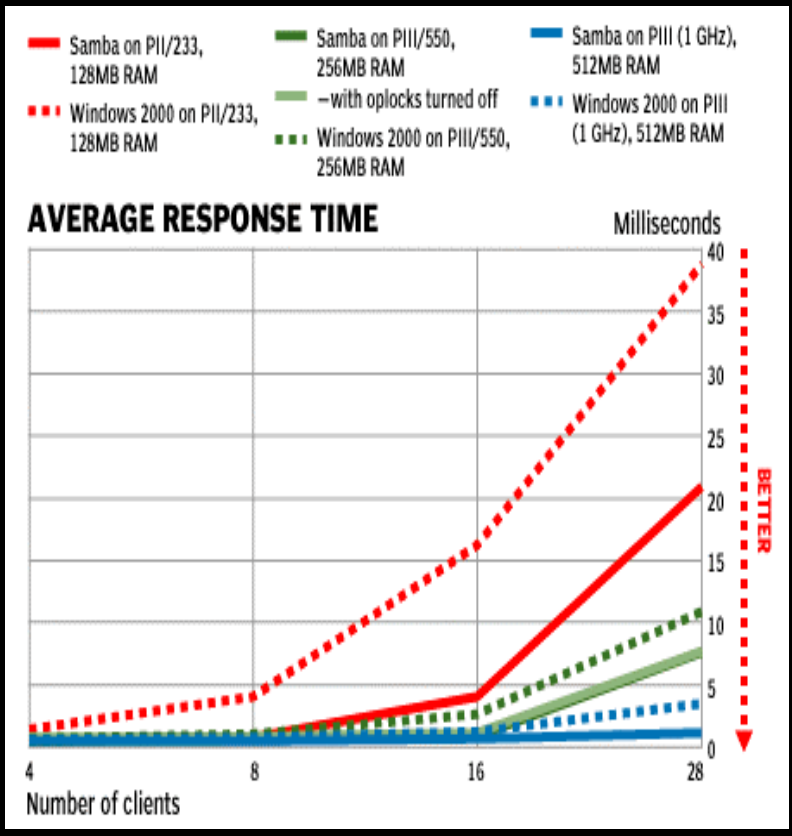


ファイルサーバ性能向上

Windows 2000よりLinux+Sambaは高性能

http://www.pcmag.com/article/0,2997,s%253D1474%2526a%253D16554,00.asp

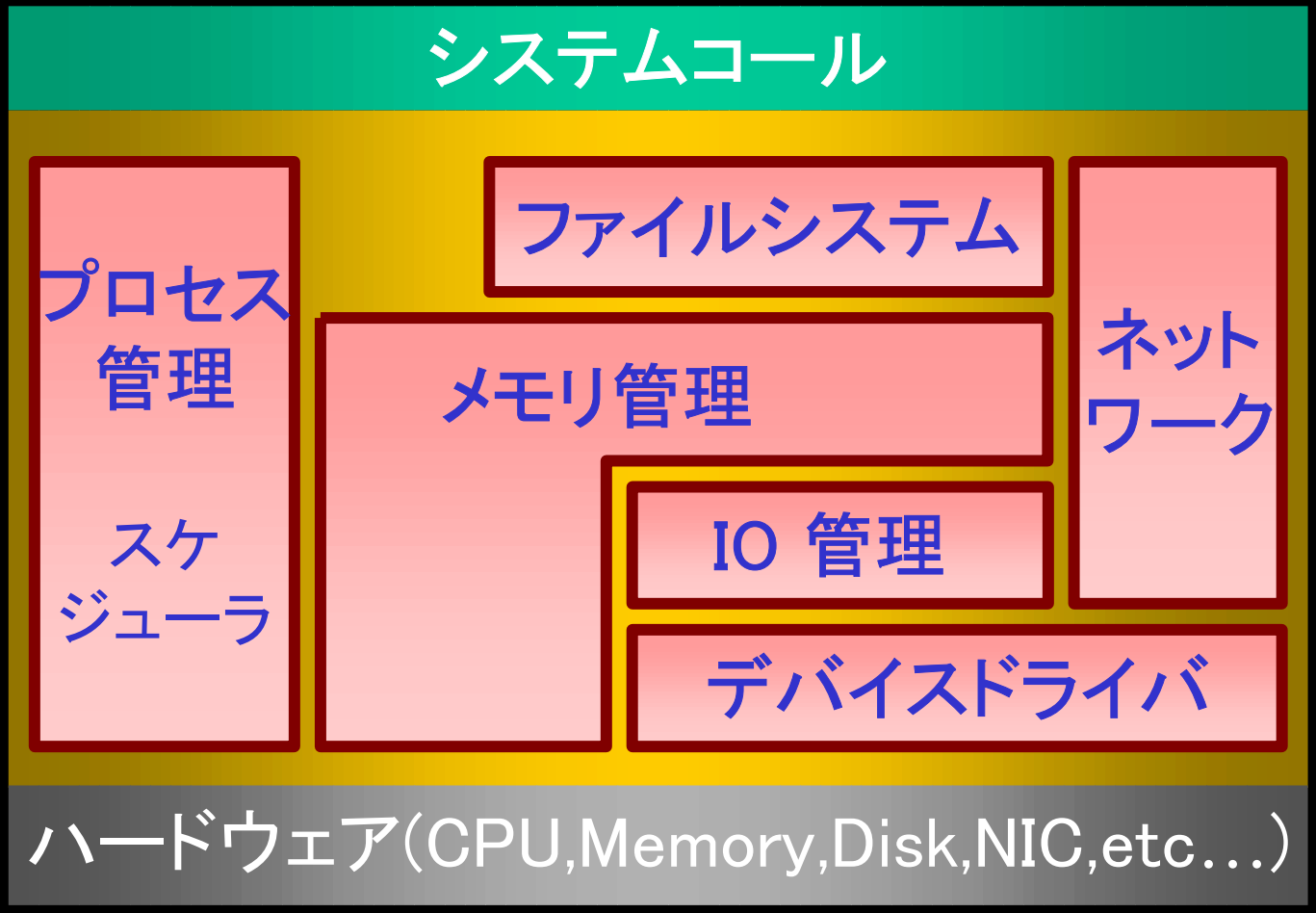




Linuxのコア：カーネル

- カーネルとは
  - ソフトウェアの中で最もハードウェアと密接に関わっている部分
  - ハードウェアを直接コントロールしているソフトウェア

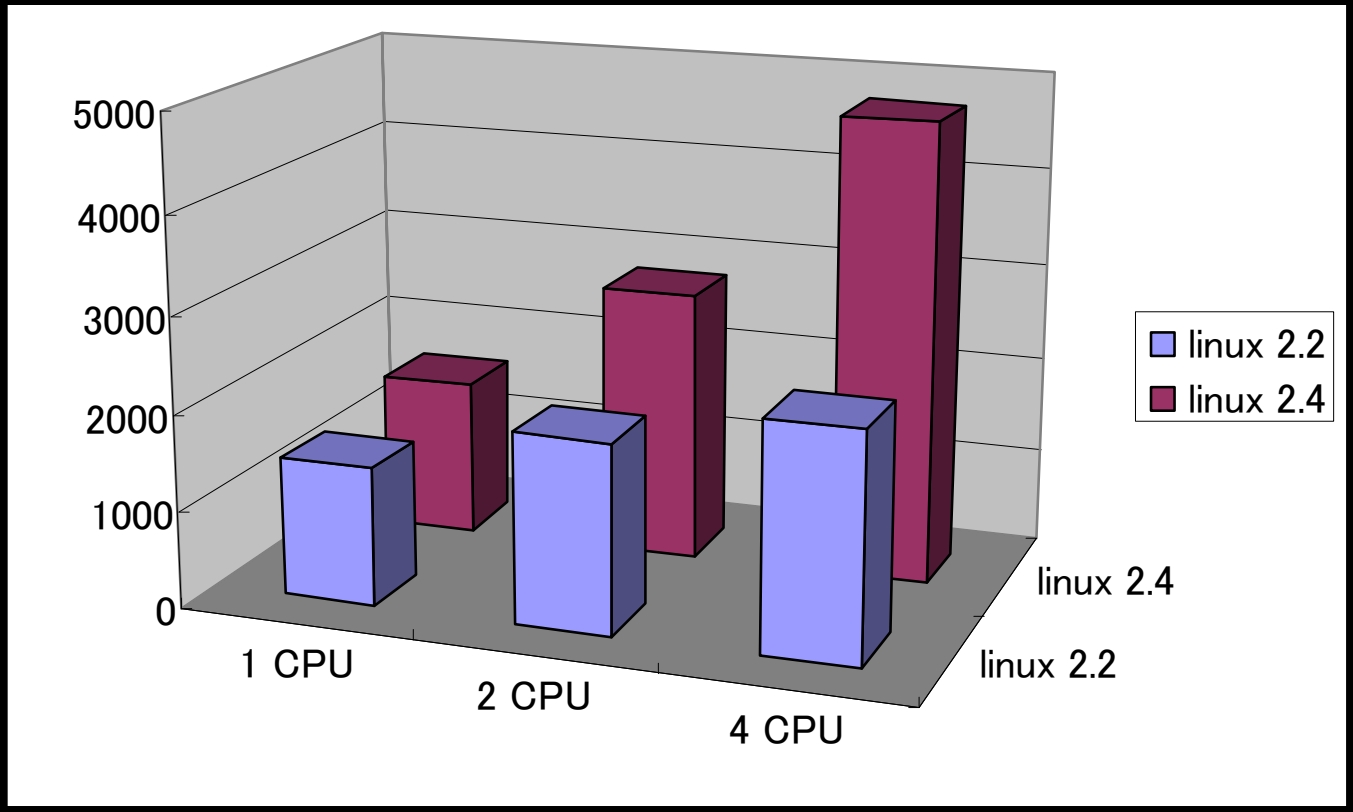
Linuxカーネルの構造



進化するLinuxカーネル

～Linux 2.2 と2.4 のスケーラビリティ～

- WebBench で測定



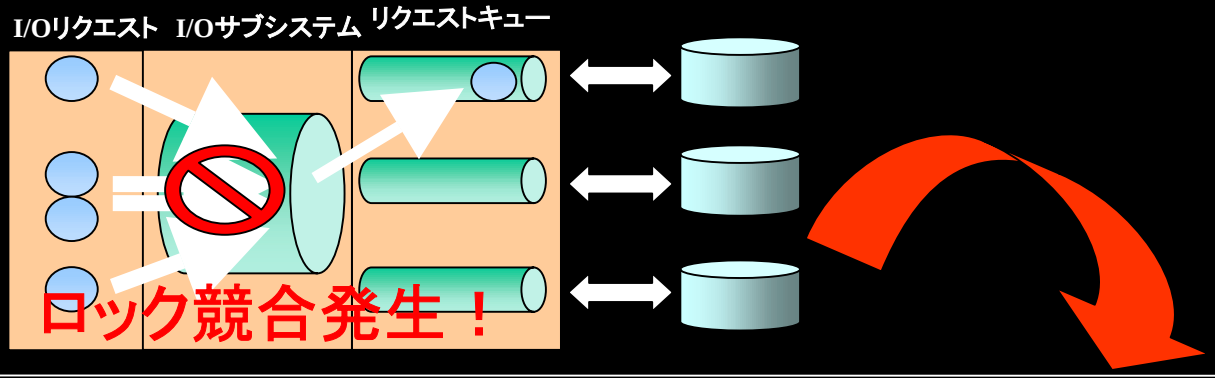
進化するLinuxカーネル
～更に機能強化されたカーネル～

- エンタープライズ領域で要求されるスケーラビリティを実現
- Oracle 9iR2 と組み合わせることによって、より大規模な DBシステムを構築可能
  - 非同期I/Oサポート
  - VLM（Very Large Memory)機能
  - O(1) プロセス スケジューラの導入
  - I/Oリクエストロックの細分化

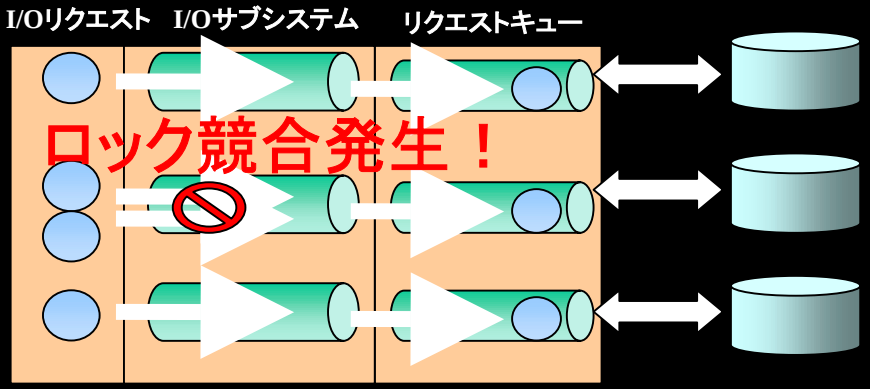
OSスケーラビリティ向上

I/Oリクエストロックの細分化

カーネルがI/Oリクエストを制御するためのロックを細分化することにより、多数のデバイスを使用するような大規模システムにおいて、スケーラビリティが向上します

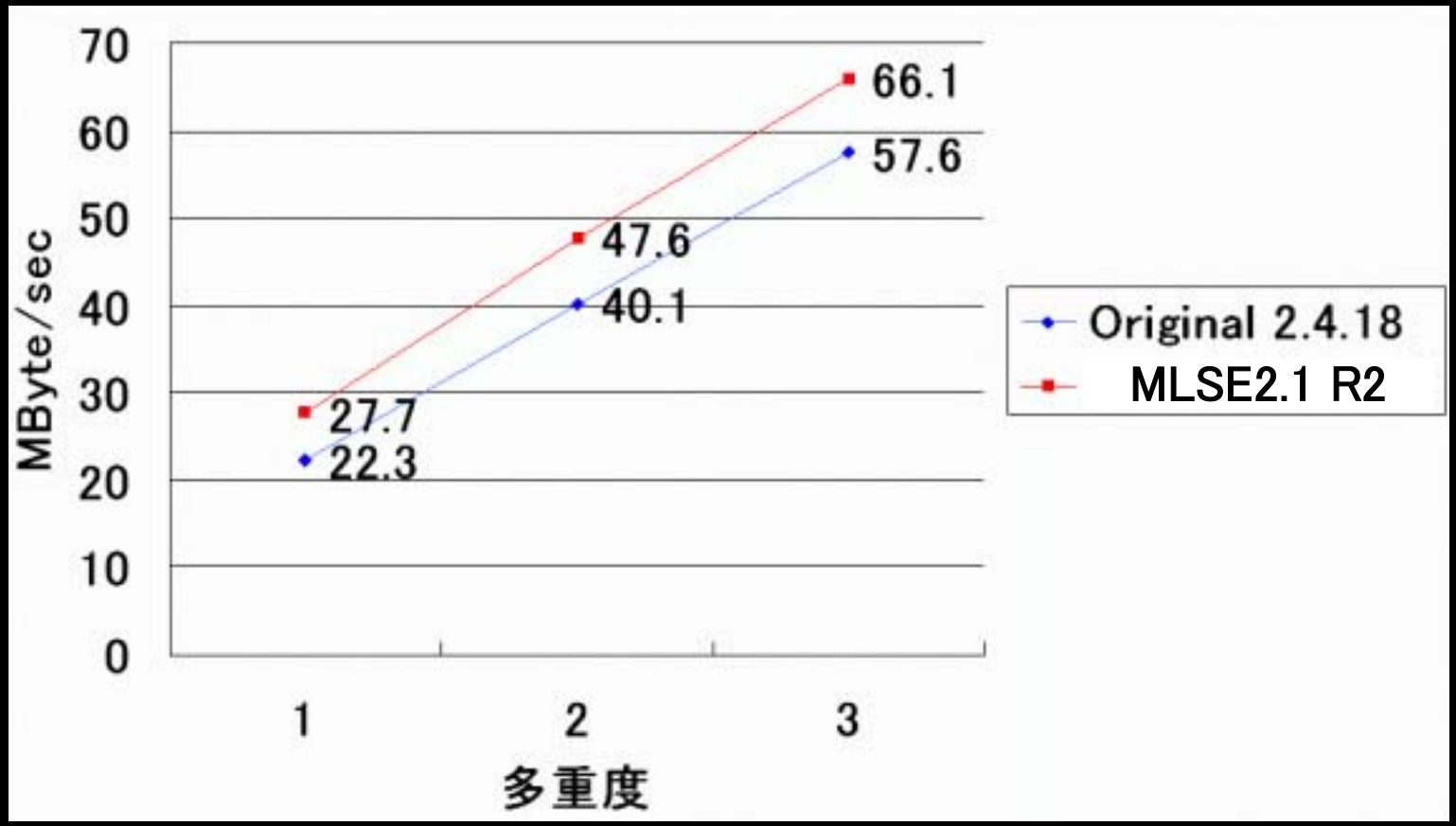


ロック競合頻度の減少によってI/O性能が向上

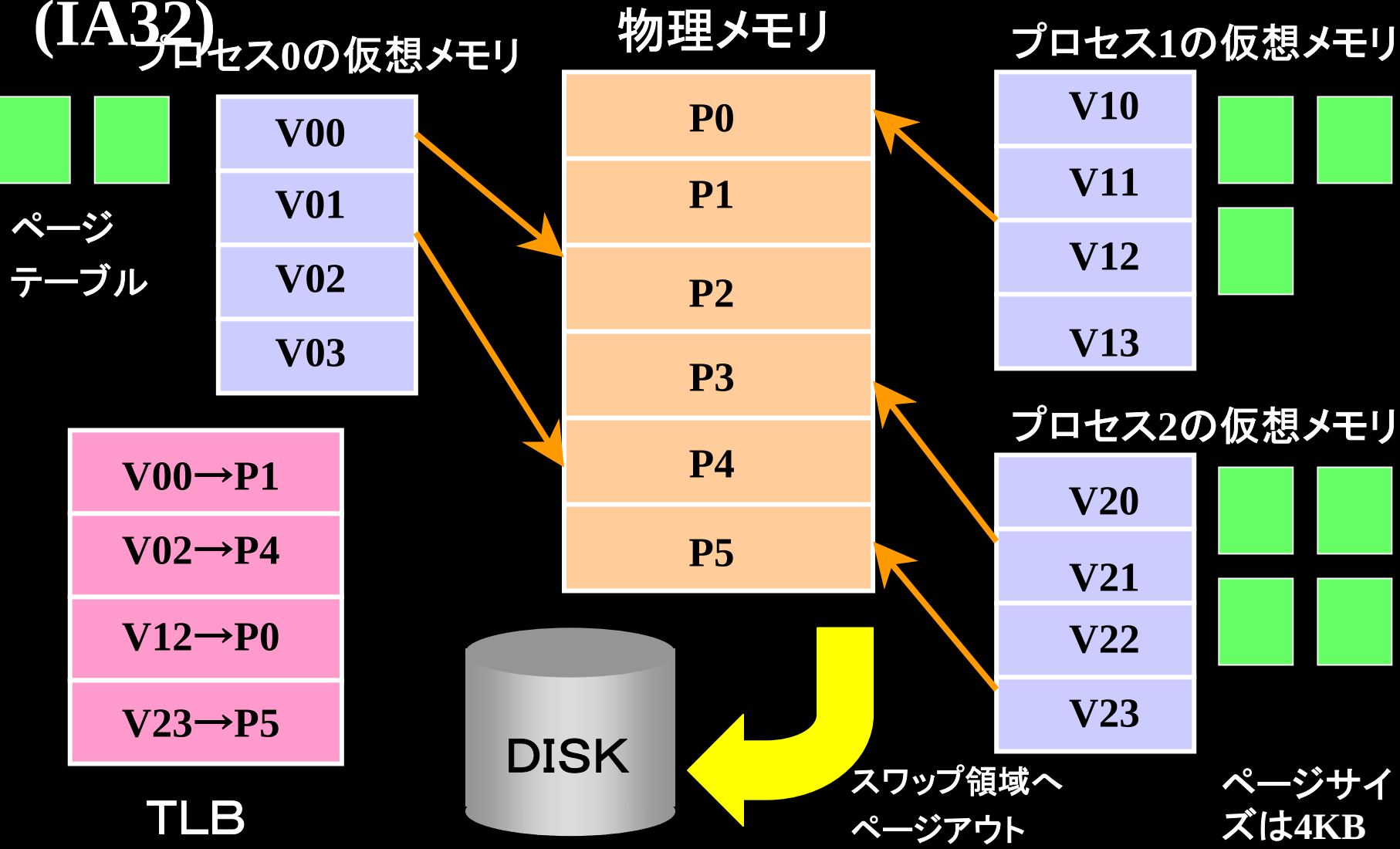


I/Oリクエストロックの細分化

- 効果測定結果



仮想アドレスから物理アドレスへのマッピング (IA32)



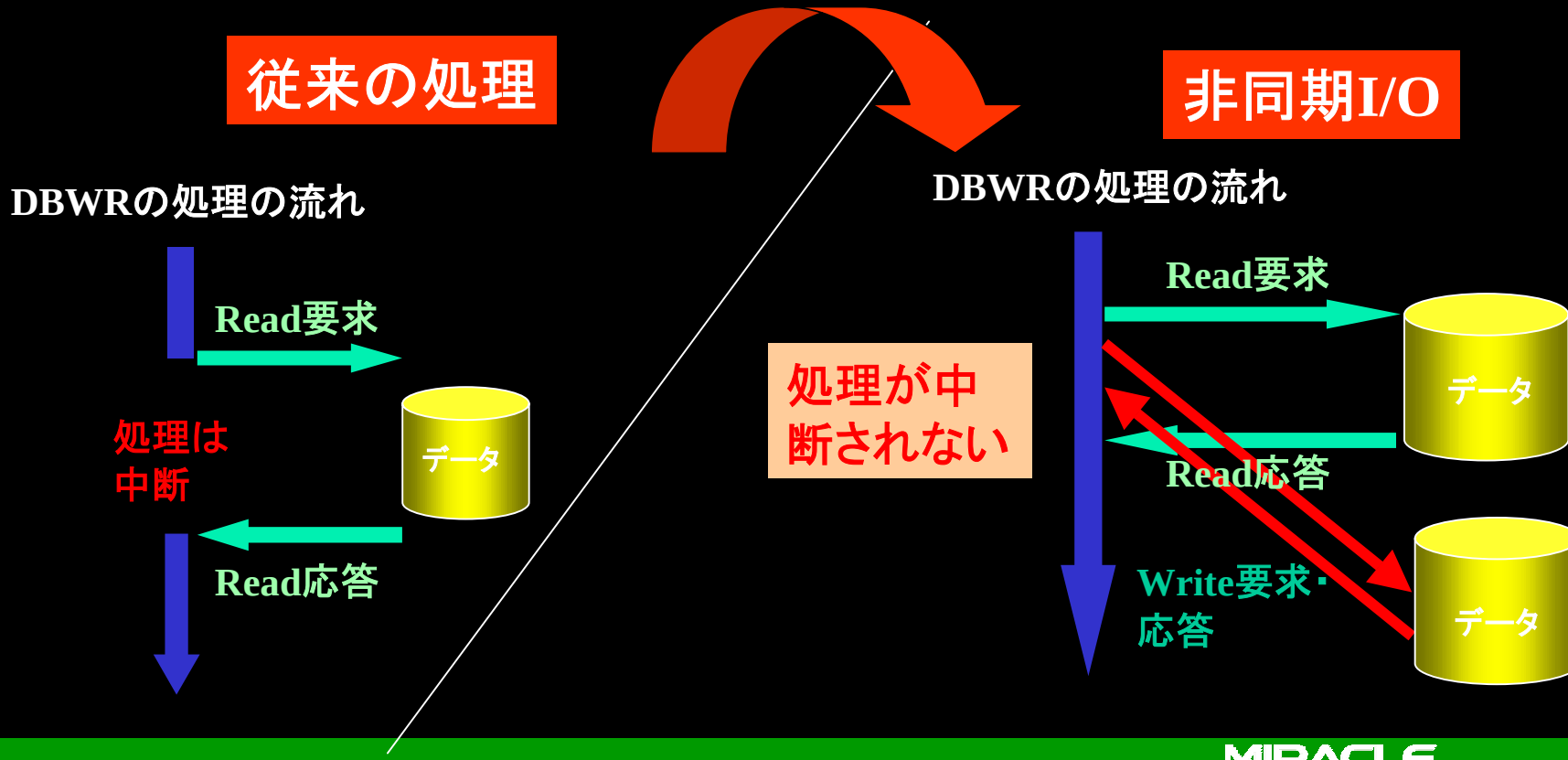
Oracle VLM 機能

- 大規模メモリを Oracle のデータベースバッファキャッシュとしてフルに利用可能
  - 従来ではSGAのサイズは最大約1.7GBという制限
  - MLSE2.1R2+Oracle9iR2 ではメモリファイルシステム使って物理メモリをフルに利用可能

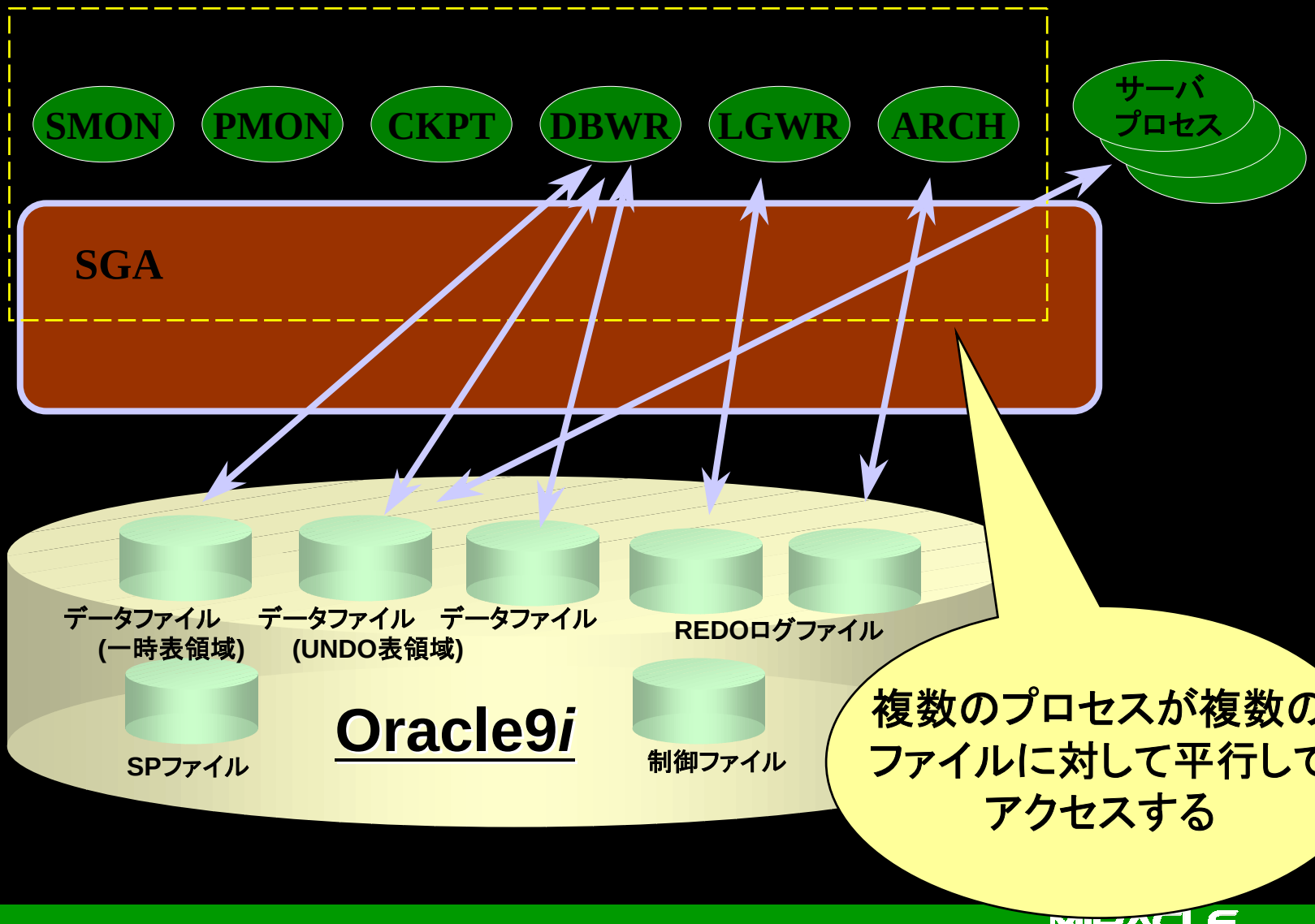
64GBメモリの
システムの場合

大規模データベースパフォーマンス向上

- 非同期 I/O（Oracle DBWRのスループット向上）
  - オラクルデータベース ライタプロセスのI/O待ちが軽減され、データベースのスループットが向上。大規模・高負荷なデータベースシステムが構築可能になる。



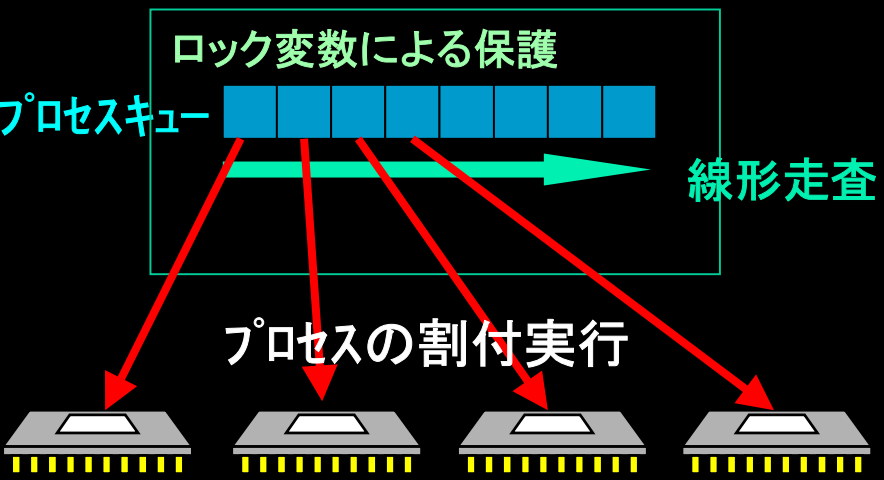
Oracleデータベースのアーキテクチャ



OSパフォーマンス向上（1）

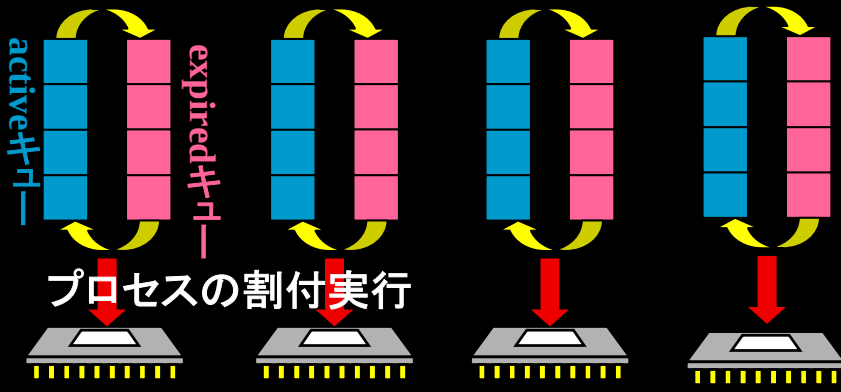
- プロセススケジューラの改善
  - 同時に多数の要求が来た際のプロセス処理が改善され、OS としての処理速度が著しく向上。
  - 複数の CPU を搭載した大規模データベース／アプリケーションサーバ システムの性能が向上。

従来の処理



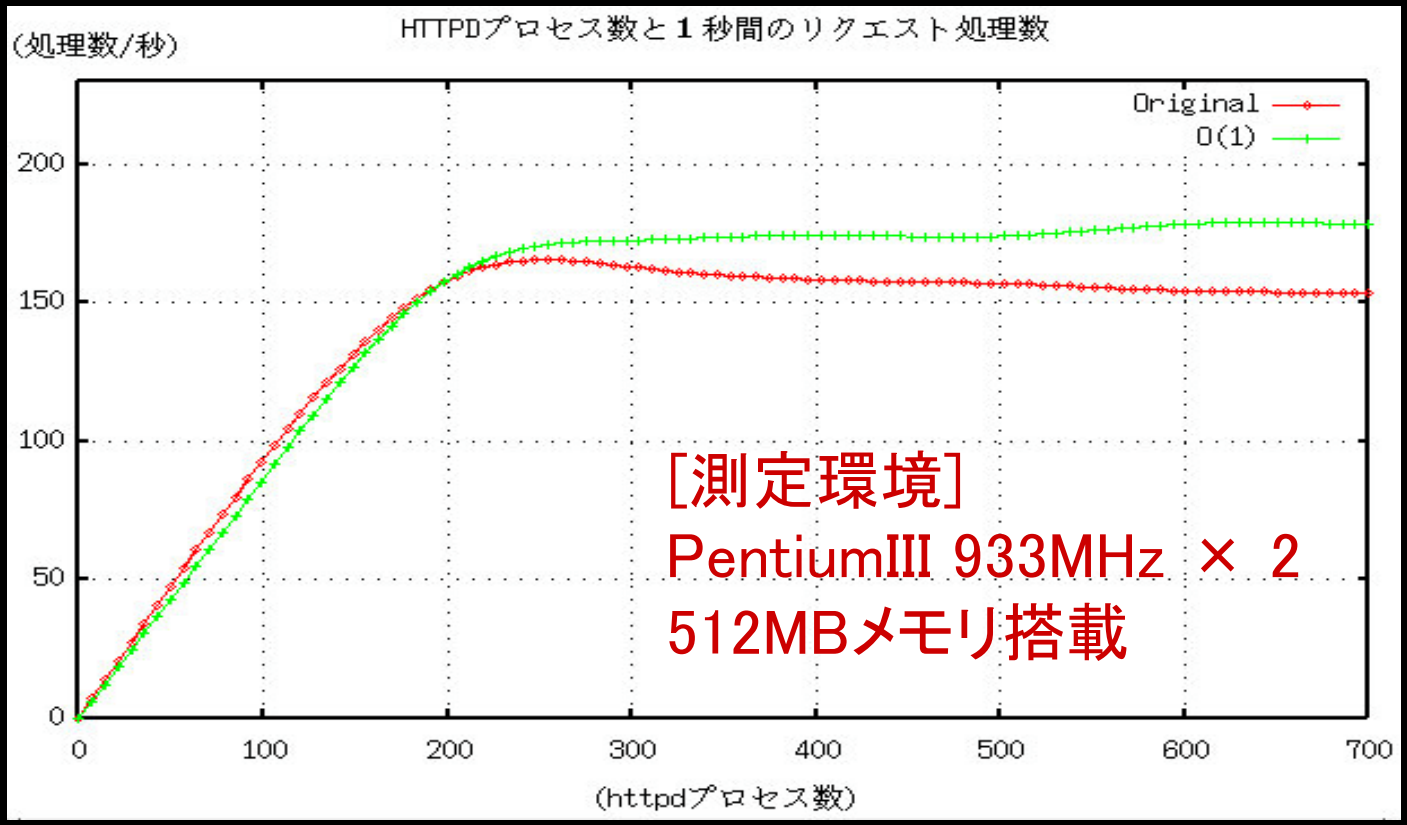
スケジューラ改善

O(1) スケジューラの導入



OSパフォーマンス向上（2）

O(1)スケジューラ測定結果



プロセス数の増加によるボトルネックが抑えられ、
パフォーマンスが向上

【お問い合わせ先】

info@miraclelinux.com

http://www.miraclelinux.com