

ORACLE®

手動の情報ライフサイクル管理はもう  
いらない！

Oracle Database 12cのILM新機能

株式会社システム・テクノロジー・アイ  
技術本部カスタマーサポート部  
代田 佳子



 #odddtky

日本オラクル、今年最大の技術トレーニング・イベント

**Oracle DBA &  
Developer Day 2013**

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント(確約)するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

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

# Program Agenda

- 情報ライフサイクル管理 (ILM) 概要
- 自動データ最適化 (ADO)
- データベース内アーカイブと時制有効性
- その他のILMに便利な機能

# 情報ライフサイクル管理 (ILM) 概要

# データのライフサイクル

データのライフサイクルにあわせたストレージ要件

Active

最新データ



最高パフォーマンス  
(小型で高速)

Less  
Active

更新頻度: 低い 更新頻度: なし  
検索頻度: 高い 検索頻度: 低い



低コスト  
(大容量)

Historical

長期分析用



オンライン  
アーカイブ  
(読取り専用)

Archive

長期分析用



オフライン  
アーカイブ  
(保存専用)

# データのライフサイクル

データのライフサイクルにあわせた圧縮要件

Active

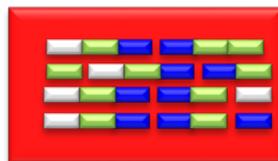
最新データ



2倍～4倍  
OLTP圧縮  
(拡張行圧縮)

Less  
Active

更新頻度:低い 更新頻度:なし  
検索頻度:高い 検索頻度:低い



10倍  
列圧縮(HCC)  
クエリーモード

Historical

長期分析用



15倍～50倍  
列圧縮(HCC)  
アーカイブモード

Archive

長期分析用

# 情報ライフサイクル管理 (ILM) としての解決策

時間経過とともにアクセス頻度が低下するデータに対するILMアクションの自動化

- 実際のアクセスパターンに応じた**最適なストレージ**にデータを配置
- **圧縮を活用**したストレージコストの削減
- **異なる圧縮レベル**によるパフォーマンスの向上

適切なタイミング  
で実施

## 自動データ最適化 (Automatic Data Optimization)

- ユーザー定義ルールに基づく**圧縮**
- 表領域の空き領域に対応した**データ移動**

## データベース内アーカイブ (In-Database Archiving)

- レコード単位で**アーカイブ** (行アーカイブ)

## 時制有効性 (Temporal Validity)

- レコード単位で**アクセス制限** (有効期間)

# 自動データ最適化(ADO)

# 自動データ最適化(ADO)とは

## データの圧縮と移動の自動化

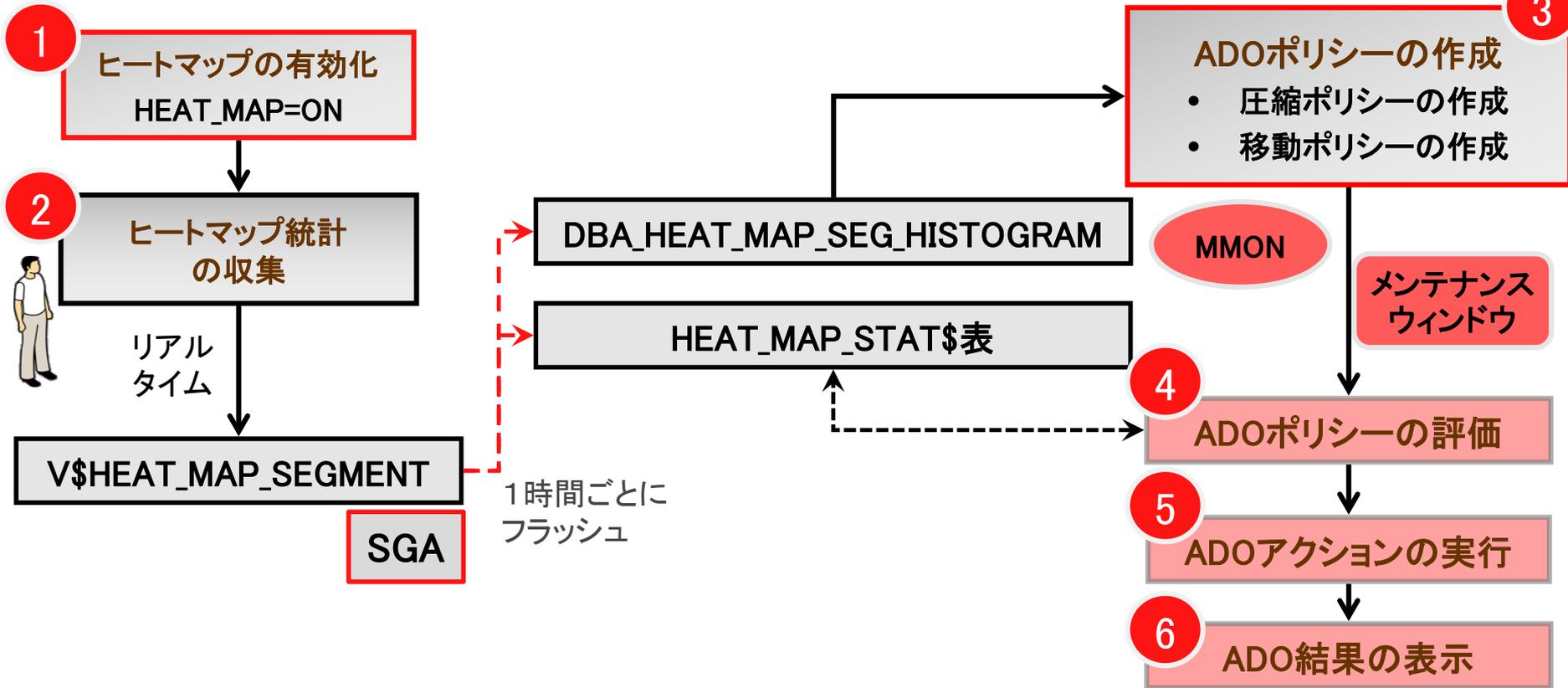
- 自動データ最適化 (Automatic Data Optimization : ADO)
  - ADOポリシーに従った**アクションの自動実行**
  - ADOポリシー=アクション(圧縮や移動)とタイミング(アクションの条件)
- ヒートマップ (Heat Map)
  - アクセスパターンやアクセス頻度の記録
  - ADOポリシーの**タイミングを自動検出**

- ADOポリシー
- ヒートマップ

自動検出

自動アクション

# ADOとヒートマップによるILM管理



# 1. ヒートマップの有効化

## HEAT\_MAP初期化パラメータの設定

```
SQL> ALTER SYSTEM SET heat_map = ON;
```

- SYSTEMとSYS\_AUX以外の表領域でセグメントでのアクセス追跡
- インスタンスレベルで有効化
  - アクセス追跡(ヒートマップ統計の収集)とADOの有効化
- セッションレベルで有効化
  - アクセス統計の収集(ヒートマップ機能)のみ有効

ADOを使用するには、インスタンスレベルでヒートマップを有効化すること

## 2. ヒートマップ統計の収集

リアルタイムにアクセス統計(アクセスパターンやアクセス頻度)を収集

- セグメントレベル

- V\$HEAT\_MAP\_SEGMENT

ディクショナリに1時間に1度フラッシュ

- DBA\_HEAT\_MAP\_SEG\_HISTOGRAM

- DBA\_HEAT\_MAP\_SEGMENT

- ブロックレベル

- DBMS\_HEAT\_MAP.BLOCK\_HEAT\_MAP

- エクステントレベル

- DBMS\_HEAT\_MAP.EXTENT\_HEAT\_MAP

## 2. ヒートマップ統計の収集: セグメントレベル

### DBA\_HEAT\_MAP\_SEG\_HISTOGRAM

```
SQL> SELECT object_name, subobject_name, track_time,  
2          segment_write WRI, full_scan FTS, lookup_scan LKP  
3 FROM DBA_HEAT_MAP_SEG_HISTOGRAM;
```

OBJECT_NAME	SUBOBJECT_NAME	TRACK_TIME	WRI	FTS	LKP
-----	-----	-----	---	---	---
INTERVAL_SALES	P1	02-JAN-12	YES	YES	NO
<b>INTERVAL_SALES</b>	<b>P2</b>	<b>28-MAR-12</b>	<b>NO</b>	<b>YES</b>	<b>NO</b>
INTERVAL_SALES	P3	28-MAR-12	NO	NO	YES
I_EMPNO		07-dec-12	YES	NO	YES

表、表パーティション、索引、LOBセグメントに関するアクティビティ情報

## 2. ヒートマップ統計の収集: ブロックレベル

### DBMS\_HEAT\_MAP.BLOCK\_HEAT\_MAP

```
SQL> SELECT segment_name, tablespace_name, block_id, writetime
2 FROM TABLE(DBMS_HEAT_MAP.BLOCK_HEAT_MAP
3             ('SCOTT', 'EMPLOYEE', NULL, 8, 'ASC'));
```

SEGMENT_	TABLESPACE_NAME	BLOCK_ID	WRITETIME
EMPLOYEE	LOW_COST_STORE	196	12-DEC-12
EMPLOYEE	LOW_COST_STORE	197	12-DEC-12
EMPLOYEE	LOW_COST_STORE	198	12-DEC-12

ブロックの最新変更時間を戻すテーブル関数

# 3. ADOポリシーの作成

どのような条件でいつポリシーを適用するのか

- **圧縮**ポリシー: 指定した期間アクセスパターンがなかったら圧縮
- **移動**ポリシー: 表領域が満杯になったら移動

## 圧縮ポリシー

- 実行レベル
  - **処理対象** (表領域、グループ、行など)
- 圧縮タイプ
  - **圧縮レベル** (基本、拡張行、HCCなど)
- 追跡対象操作
  - **アクセスパターン** (作成、アクセスなど)
- 時期: 指定した操作になってからの**期間**

## 移動ポリシー

- 移動タイプ: **セグメント**レベルで移動
- 移動先: 指定した**表領域**に移動
- しきい値: ソース表領域**満杯の判定**

# 3. ADOポリシーの作成

## 圧縮ポリシー: 実行レベルと圧縮タイプ

- 実行レベル(処理対象)
  - **表領域**: 新規格納されるセグメントに継承(既存セグメントは対象外)
  - **グループ**: 対象オブジェクトに属する索引やLOBにも継承
  - **セグメント**: 表、パーティション、サブパーティション
  - **行**: ブロック単位で圧縮(拡張行圧縮時のみ)
- 圧縮タイプ(圧縮レベル)
  - **基本**: ROW STORE COMPRESS [BASIC]
  - **拡張行**: ROW STORE COMPRESS ADVANCED
  - **HCCクエリー**: COLUMN STORE COMPRESS FOR QUERY
  - **HCCアーカイブ**: COLUMN STORE COMPRESS FOR ARCHIVE

# 3. ADOポリシーの作成

## 圧縮ポリシー: 追跡対象操作、時期

- 追跡対象操作 (アクセスパターン)
  - 変更なし (DMLなし): NO MODIFICATION
  - アクセスなし (SELECT、DMLなし): NO ACCESS
  - アクセス減少 (SELECT、DML減少): LOW ACCESS
  - 作成 (セグメント作成): CREATION
- 時期 (経過期間)
  - AFTER n **DAY[S]**
  - AFTER n **MONTH[S]**
  - AFTER n **YEAR[S]**

# 3. ADOポリシーの作成

## 圧縮ポリシー定義例

- 表領域レベル

```
SQL> ALTER TABLESPACE tbs1 DEFAULT ILM ADD POLICY
2          COLUMN STORE COMPRESS FOR QUERY LOW
3          SEGMENT AFTER 6 MONTHS OF LOW ACCESS;
```

- グループレベル、行レベル

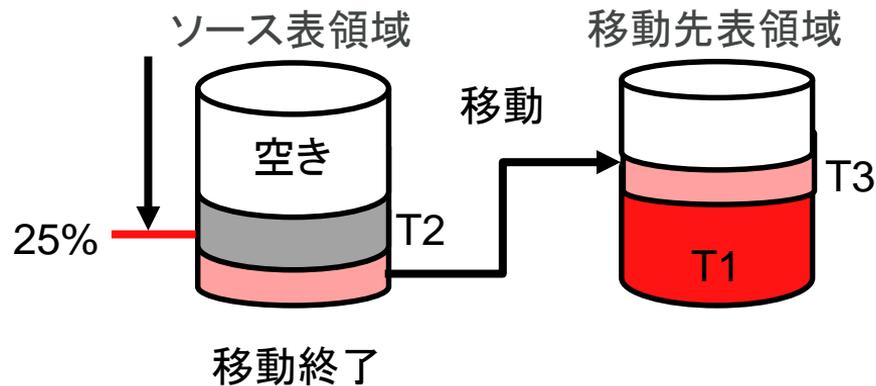
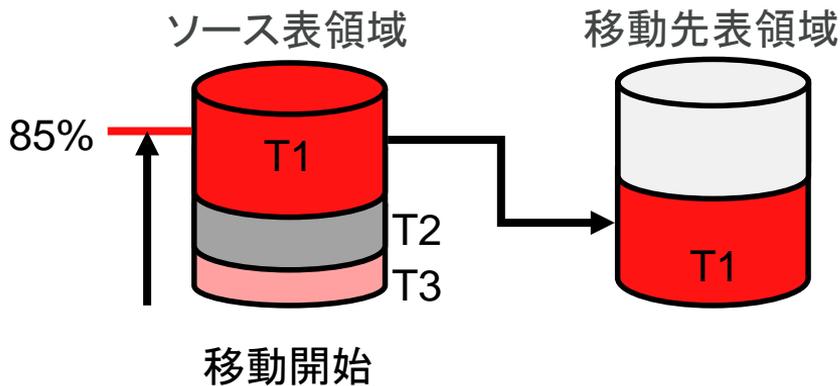
```
SQL> ALTER TABLE tab1 ILM ADD POLICY
2          COLUMN STORE COMPRESS FOR ARCHIVE HIGH
3          GROUP AFTER 5 YEARS OF NO MODIFICATION;
```

```
SQL> ALTER TABLE tab2 MODIFY PARTITION p1 ILM ADD POLICY
2          ROW STORE COMPRESS ADVANCED
3          ROW AFTER 30 DAYS OF NO ACCESS;
```

# 3. ADOポリシーの作成

## 移動ポリシー:しきい値

- 移動開始:使用率(USED)に達すると移動先表領域に移動
  - デフォルト:85%
- 移動終了:空き率(FREE)に達すると移動をやめる
  - デフォルト:25%



# 3. ADOポリシーの作成

## 移動ポリシー:しきい値の管理

- DBA\_ILMPARAMETERSビューで確認

```
SQL> SELECT * FROM DBA_ILMPARAMETERS;
```

NAME	VALUE
-----	-----
TBS PERCENT USED	85
TBS PERCENT FREE	25

- DBMS\_ILM\_ADMIN.CUSTOMIZE\_ILMにて変更

```
SQL> EXEC  
DBMS_ILM_ADMIN.CUSTOMIZE_ILM(DBMS_ILM_ADMIN.TBS_PERCENT_USED,90)  
SQL> EXEC  
DBMS_ILM_ADMIN.CUSTOMIZE_ILM(DBMS_ILM_ADMIN.TBS_PERCENT_FREE,15)
```

# 3. ADOポリシーの作成

## 移動ポリシー定義例

- 指定した表領域に移動

```
SQL> ALTER TABLE tab3 ILM ADD POLICY  
2          TIER TO tbs2  
3          READ ONLY;
```

- 移動後、移動先表領域をREAD ONLYに変更する

```
SQL> ALTER TABLE tab4 MODIFY PARTITION p1 ILM ADD POLICY  
2          TIER TO tablespace_tbs  
3          READ ONLY;
```

# 3. ADOポリシーの作成

アクション(圧縮/移動)とアクセスパターンの組合せ

実行レベル		変更なし	アクセスなし	アクセス減少	作成	表領域が満杯	カスタムポリシー
圧縮	行	✓					
	セグメント	✓	✓	✓			✓
	グループ	✓	✓	✓			
	表領域	✓	✓	✓	✓		
移動	セグメント					✓	✓
	表領域					✓	

# 3. ADOポリシーの作成

カスタムポリシー: PL/SQLファンクションにて起動条件をカスタマイズ

- カスタムポリシー用ファンクション(TRUE/FALSEを戻す)

```
SQL> CREATE FUNCTION custom_ilm_rule(objid IN NUMBER)
2   RETURN BOOLEAN
3   ...
n   /
```

- カスタムポリシーを使用して起動条件を指定

```
SQL> ALTER TABLE EMP ILM ADD POLICY
2           ROW STORE COMPRESS ADVANCED
3           SEGMENT ON custom_ilm_rule;
```

## 4. ADOポリシーの評価

### MMONとメンテナンスウィンドウによる自動実行

- 自動実行
  - MMON: ROWレベルのポリシー(デフォルト15分毎)
  - メンテナンスWindow: ROWレベル以外
- 評価頻度の変更(分単位)

```
SQL> EXEC DBMS_ILM_ADMIN.CUSTOMIZE_ILM  
(DBMS_ILM_ADMIN.EXECUTION_INTERVAL, 5)
```

# 5. ADOアクションの実行

自動実行だけでなく即時実行やカスタマイズも可能

- 即時実行

```
SQL> VAR v_taskid NUMBER
SQL> EXEC DBMS_ILM.EXECUTE_ILM (owner =>'SCOTT',
object_name =>'EMPLOYEE', task_id =>:v_taskid)
```

- 評価対象オブジェクトのカスタマイズ

ポリシー評価

PREVIEW\_ILM

オブジェクトの追加/削除

ADD\_TO\_ILM  
REMOVE\_TO\_ILM

タスク実行

EXECUTE\_ILM\_TASK

## 6. ADO結果の表示

自動実行と手動実行のタスク結果を確認

- **DBA\_ILMTASKS**: ADO関連タスクの実行状態に関する情報
- **DBA\_ILMEVALUATIONDETAILS**: ADOポリシー評価の詳細
- **DBA\_ILMRESULTS**: 移動関連タスクの実行に関する情報

```
SQL> SELECT task_id, policy_name, selected_for_execution,  
2          job_name  
3 FROM DBA_ILMEVALUATIONDETAILS WHERE object_name='EMP';
```

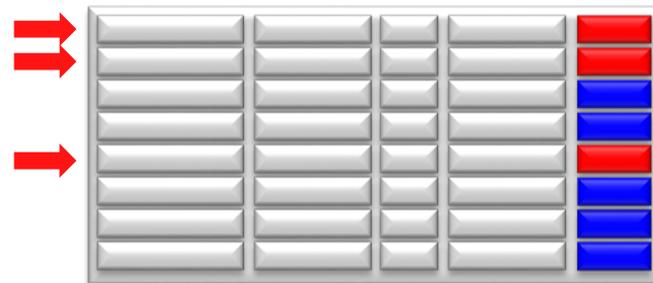
TASK_ID	POLICY_NAME	SELECTED_FOR_EXECUTION	JOB_NAME
18762	P281	POLICY DISABLED	
18542	P281	SELECTED FOR EXECUTION	ILMJOB5002
18862	P301	PRECONDITION NOT SATISFIED	

# データベース内アーカイブと 時制有効性

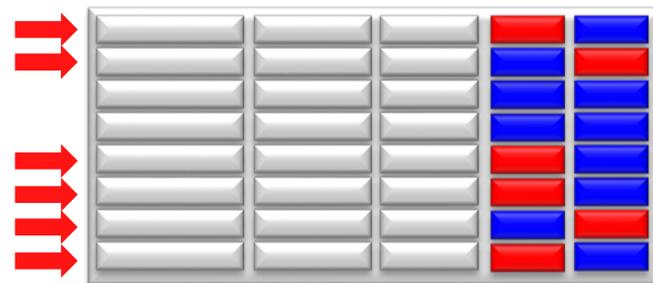
# データベース内アーカイブと時制有効性とは

## アクティブなデータにのみアクセス

- データベース内アーカイブ  
(In-Database Archiving)
  - **アクティブ**なデータにのみアクセス
  - **非アクティブ**なデータも表の中で保持する



- 時制有効性 (Temporal Validity)
  - **レコードに有効期限**の定義
  - 有効期限内のデータにのみアクセスさせる**フィルタ処理**



# データベース内アーカイブの有効化/無効化

## ROW ARCHIVAL句

- 有効化: ROW ARCHIVAL句
  - **ORA\_ARCHIVE\_STATE**列(アーカイブ属性列)が追加される

```
SQL> CREATE TABLE emp
2      (empno NUMBER(7), fullname VARCHAR2(40),
3      job VARCHAR2(9))
4      ROW ARCHIVAL;
```

- 無効化: NO ROW ARCHIVAL句
  - **ORA\_ARCHIVE\_STATE**列が削除される

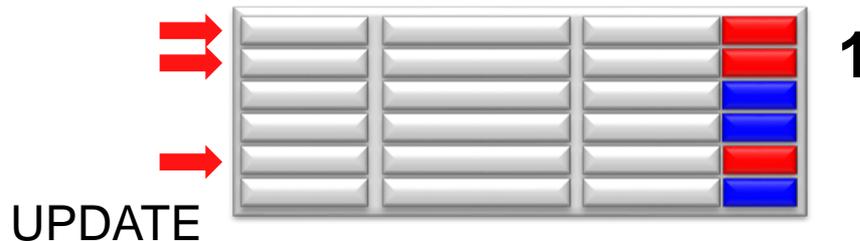
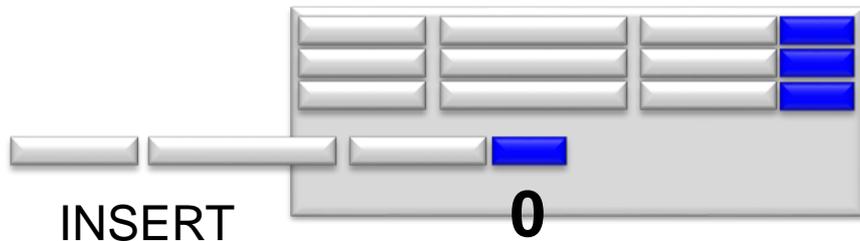
```
SQL> ALTER TABLE emp NO ROW ARCHIVAL;
```

# データベース内アーカイブの変更

## ORA\_ARCHIVE\_STATE列の変更

- DBMS\_ILM.ARCHIVESTATENAME関数
  - 値「0」= **アクティブ**なレコード (INSERT時のデフォルト)
  - 値「1」= **非アクティブ**なレコード (明示的にUPDATE)

```
SQL> UPDATE emp
2      SET ORA_ARCHIVE_STATE = DBMS_ILM.ARCHIVESTATENAME(1)
3  WHERE empno < 100;
```



# データベース内アーカイブの表示制御

## ROW ARCHIVAL VISIBILITYセッションパラメータ

- アクティブな行のみ表示 (デフォルト=ACTIVE)
- すべての行を表示 (ALL)

```
SQL> ALTER SESSION SET ROW ARCHIVAL VISIBILITY = ALL;
```

```
SQL> SELECT ORA_ARCHIVE_STATE, fullname FROM emp;
```

```
ORA_ARCHIVE_STATE FULLNAME
```

```
-----
```

```
0 JEAN
```

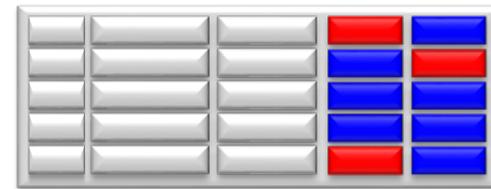
```
1 ADAM ←
```

```
1 JIM ←
```

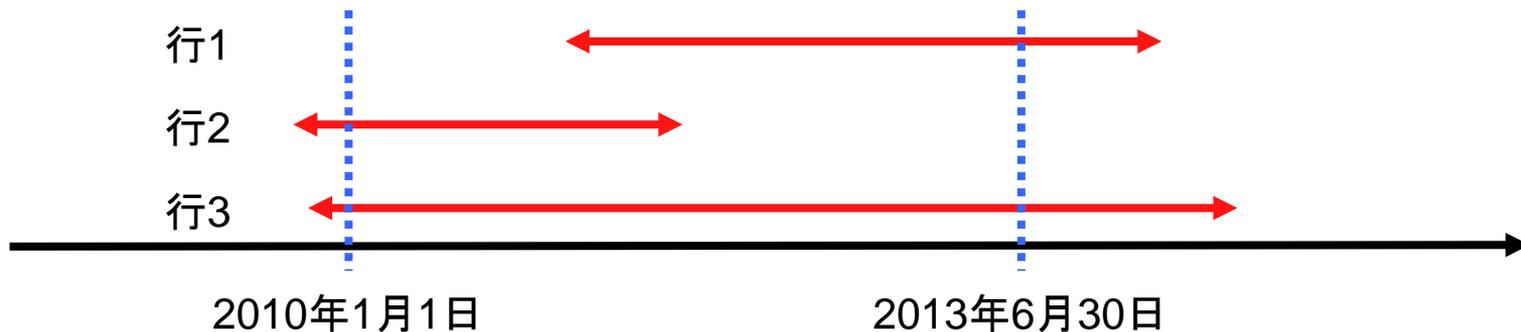
ALLにより非アクティブでも表示

# 時制有効性

## 有効期間の定義



- 有効期間列: 有効期間を決定する**開始列**と**終了列**
- 有効期間(任意の時点、任意の期間)にあるデータのみが対象となる
  - 2010年1月1日時点で有効なデータ => 行2、行3
  - 2013年6月30日時点で有効なデータ => 行1、行3
  - 2010年1月1日から2013年6月30日の間で有効なデータ => 行1、行2、行3



# 時制有効性の有効化/無効化

## ユーザー定義列を使用したPERIOD FOR句

- 有効化: **既存の日付型** (DATE、TIMESTAMP) **の列** を利用

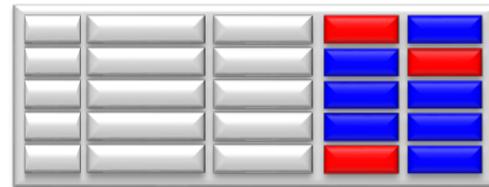
```
SQL> CREATE TABLE emp
  2   (empno NUMBER, name VARCHAR2(100), salary NUMBER,
  3     user_time_start DATE, user_time_end DATE,
  4     PERIOD FOR user_time (user_time_start, user_time_end));
```

- 無効化: PERIOD FORのDROP
  - ユーザー定義列はそのまま残る

```
SQL> ALTER TABLE emp DROP (PERIOD FOR user_time);
```

# 時制有効性の有効化/無効化

暗黙生成列(非表示列)を使用したPERIOD FOR句



- 有効化:有効期間名のみを指定
  - 有効期間名\_STARTと有効期間名\_END列が自動作成
  - TIMESTAMP(6) WITH TIME ZONEデータ型の非表示列

```
SQL> CREATE TABLE emp
  2   (empno NUMBER, name VARCHAR2(100), salary NUMBER,
  3   PERIOD FOR emp_time);
```

- 無効化:PERIOD FORのDROP
  - 有効期間列(非表示列)も同時削除される

```
SQL> ALTER TABLE emp DROP (PERIOD FOR emp_time);
```

# 時制有効性のデータ検索

## PERIOD FOR句を使用したフラッシュバッククエリー

- AS OF PERIOD FOR *有効期間名 日付*
  - 指定した日付時点で有効なデータを検索

```
SQL> SELECT * FROM emp AS OF PERIOD FOR user_time
2     TO_DATE('2010-01-01','YYYY-MM-DD');
```

- VERSIONS PERIOD FOR *有効期間名 開始日付 AND 終了日付*
  - 指定した開始日付から終了日付の間で有効なデータを検索

```
SQL> SELECT * FROM emp VERSIONS PERIOD FOR user_time
2     BETWEEN TO_DATE('2010-01-01','YYYY-MM-DD')
3     AND TO_DATE('2013-06-30','YYYY-MM-DD');
```

# 時制有効性のデータ検索

## DBMS\_FLASHBACK\_ARCHIVEを使用した表示の制御

- **ENABLE\_AT\_VALID\_TIME** プロシージャにてセッションレベルの制御
  - **ALL**: すべてのデータを表示

```
SQL> EXEC DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('ALL')
```

- **CURRENT**: 現在の日付で有効なデータのみを表示

```
SQL> EXEC DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('CURRENT')
```

- **ASOF**: 指定した時点で有効なデータのみを表示

```
SQL> EXEC DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME('ASOF',  
  (TO_TIMESTAMP('2010-09-10 17.50.20')))
```

# その他のILMに便利な機能

# オンラインでのデータファイル移動

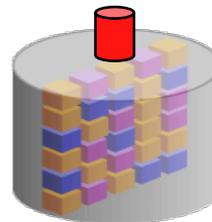
## MOVE DATAFILE句によるデータファイル移動

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '/disk2/myexample01.dbf';
```

- **KEEP**句: 元ファイルの保存も可能(OMFを除く)

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '+DiskGroup2' KEEP;
```

データファイルの移動中もユーザー操作、メンテナンス操作が可能

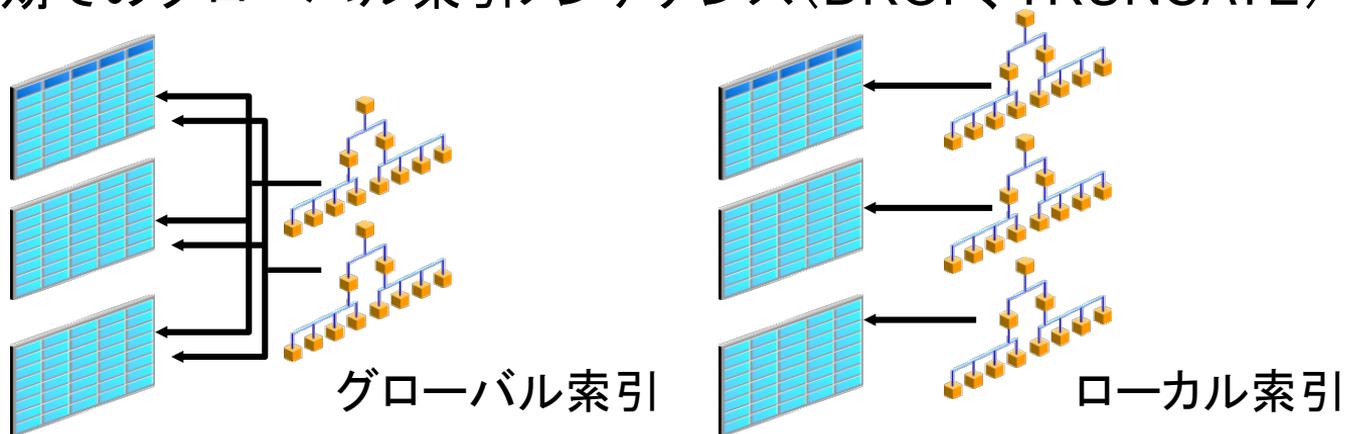


1. OFFLINE
2. 物理的にコピー
3. RENAME
4. ONLINE

# パーティショニング関連操作の改善

## パーティションメンテナンス操作 (PMOP) の効率化

- オンラインでのパーティション移動
- 複数パーティションでのパーティションメンテナンス操作
- パーティション表の部分索引 (INDEXING属性)
- 非同期でのグローバル索引メンテナンス (DROP、TRUNCATE)

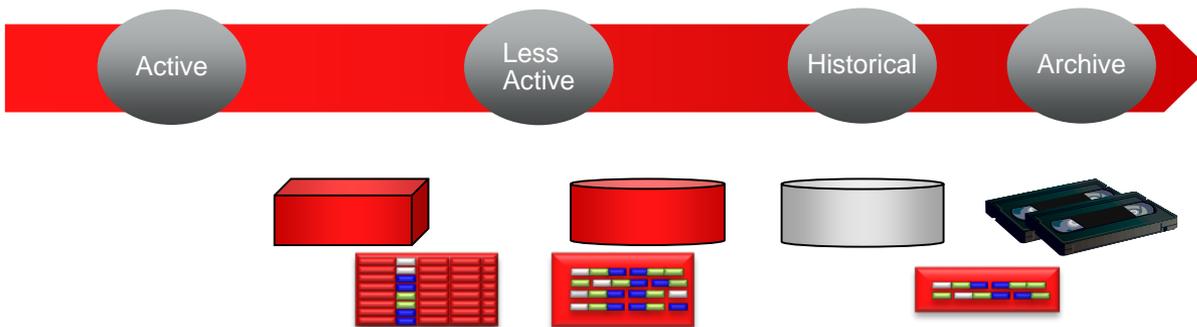


# パーティショニング関連操作の改善

## オンラインでのパーティション移動

```
SQL> ALTER TABLE orders MOVE PARTITION ord_p1  
2 ROW STORE COMPRESS ADVANCED  
3 UPDATE INDEXES ONLINE;
```

- 移動と同時に**圧縮**も可能
- **グローバル索引とローカル索引**を同時にメンテナンス可能

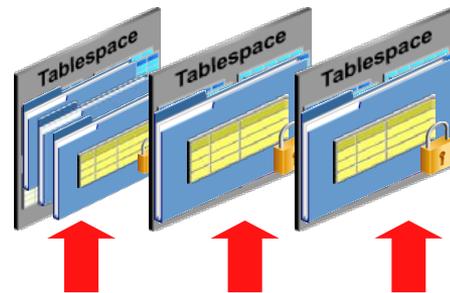


# パーティショニング関連操作の改善

## 複数パーティションでのパーティションメンテナンス操作

- 複数のパーティションに対するメンテナンスは1つのコマンドで一括操作
  - DROP
  - ADD
  - TRUNCATE
  - MERGE
  - SPLIT

```
SQL> ALTER TABLE orders
2  MERGE PARTITIONS p10,p11,p12
3  INTO PARTITION p10;
```



DROP PARTITION S p1,p2,p3

# パーティショニング関連操作の改善

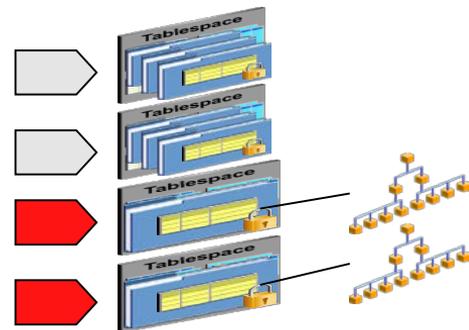
## パーティション表の部分索引(INDEXING属性)

```
CREATE TABLE part1(  
  ...  
  PARTITION BY RANGE (col1)  
  (PARTITION ord_p1 VALUES LESS THAN(100) INDEXING OFF,  
  PARTITION ord_p2 VALUES LESS THAN(200) INDEXING OFF,  
  PARTITION ord_p3 VALUES LESS THAN(300),  
  PARTITION ord_p4 VALUES LESS THAN(maxvalue));
```

```
SQL> CREATE INDEX ind1 ON part1(col1)  
  2 LOCAL INDEXING PARTIAL;
```

```
SQL> CREATE INDEX ind2 ON part1(col2)  
  2 GLOBAL INDEXING PARTIAL;
```

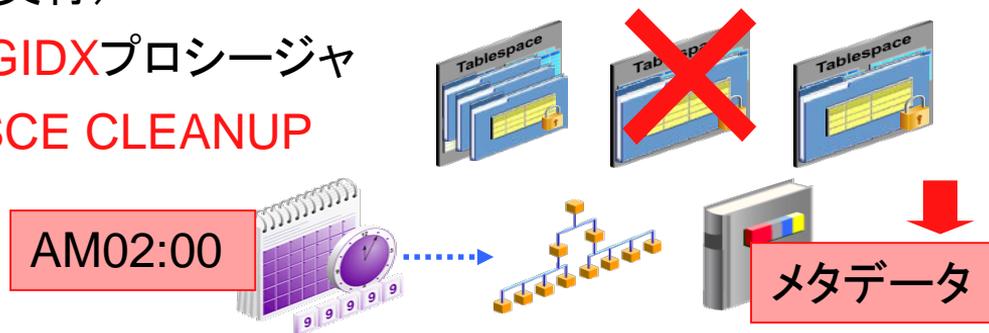
- ディスク領域の削減
- 統計取得時間の短縮



# パーティショニング関連操作の改善

## 非同期でのグローバル索引メンテナンス(DROP、TRUNCATE)

- **UPDATE INDEXES**句を指定したパーティションへの**DROP**と**TRUNCATE**  
=ローカル索引とグローバル索引の同時メンテナンス
  - コマンド実行時は**メタデータのみ更新**(操作の高速化)
- クリーンアップは後で実行(非同期)
  - **PMO\_DEFERRED\_GIDX\_MAINT\_JOB**スケジューラジョブ  
(デフォルト:毎日午前2時に実行)
  - **DBMS\_PART.CLEANUP\_GIDX**プロシージャ
  - **ALTER INDEX ... COALESCE CLEANUP**

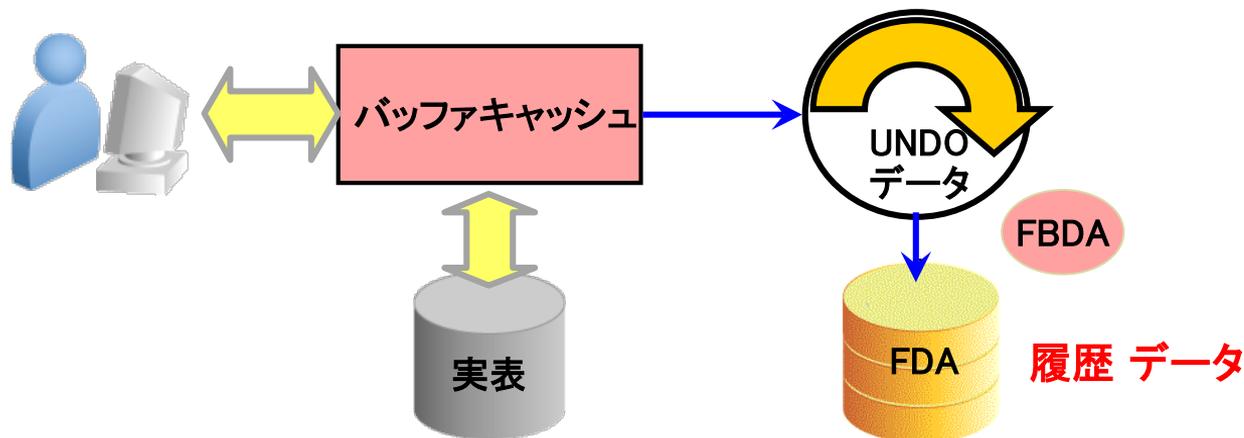


ORACLE

# フラッシュバックデータアーカイブの改善

## フラッシュバックデータアーカイブ管理の効率化

- ユーザーコンテキスト (USERENV) の追跡
- 履歴表の最適化 (圧縮と重複除外)



# フラッシュバックデータアーカイブの改善

## ユーザーコンテキスト (USERENV) の追跡

- 履歴データに属性データも保存

```
SQL> EXEC DBMS_FLASHBACK_ARCHIVE.SET_CONTEXT_LEVEL('TYPICAL')
```

- NONE: 収集なし (デフォルト)
- TYPICAL: ユーザーID、クライアント識別子、サービス名、ホスト名など
- ALL: すべてのユーザー属性

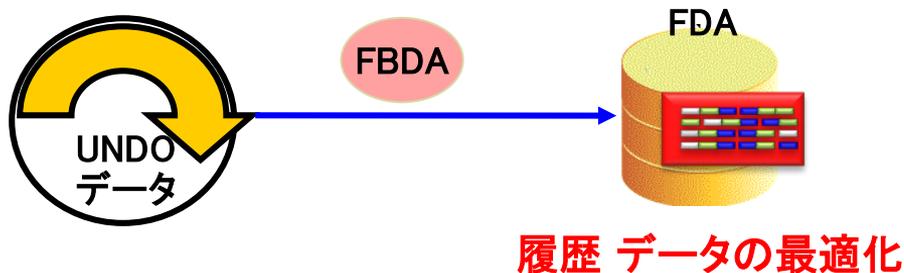
```
SQL> SELECT DBMS_FLASHBACK_ARCHIVE.GET_SYS_CONTEXT
2          (VERSIONS_XID, 'USERENV', 'SESSION_USER'),
3          VERSIONS_XID, VERSIONS_STARTTIME, VERSIONS_ENDTIME,
4          employee_id, salary
5 FROM hr.employees
6 VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE ;
```

# フラッシュバックデータアーカイブの改善

## 履歴表の最適化(圧縮と重複除外)

- デフォルト:最適化なし(NO OPTIMIZE DATA)
- **OPTIMIZE DATA**句にて最適化(圧縮と重複除外)の有効化

```
SQL> CREATE FLASHBACK ARCHIVE fla1 TABLESPACE tbs1  
2      OPTIMIZE DATA RETENTION 5 YEAR;
```



# 参考マニュアル

[http://docs.oracle.com/cd/E49329\\_01/index.htm](http://docs.oracle.com/cd/E49329_01/index.htm)

- 『Oracle Database **VLDBおよびパーティショニングガイド**』
  - 自動データ最適化(ADO)、ヒートマップ
  - データベース内アーカイブ(行アーカイブ)
  - 時制有効性(時間的な有効性)
  - パーティショニング関連操作の改善
- 『Oracle Database **開発ガイド**』
  - フラッシュバックデータアーカイブの改善
- 『Oracle Database **管理者ガイド**』
  - オンラインでのデータファイル移動

**Hardware and Software**

**ORACLE®**

**Engineered to Work Together**

ORACLE®