ORACLE

Oracle Database 12*c* Release 1

情報ライフサイクル管理

日本オラクル株式会社

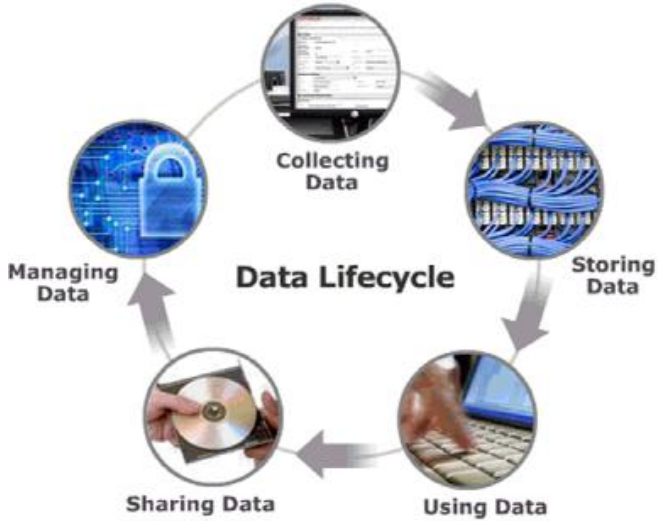
Program Agenda

- ILMの課題とソリューション
- ILMの新機能
- パーティションの新機能
- その他の新機能

ILMの課題とソリューション

11g以前

ILMの課題とソリューション



ストレージコスト削減

- 圧縮
  - Basic圧縮
  - OLTP圧縮
  - Hybrid Columnar圧縮
- 低コスト・ストレージへの移動
- 使われなくなったデータのパージ

性能改善

- パーティション（行移動）
- 高性能ストレージへの移動

安全なデータ

- 権限とビューの設定
- VPD／OLS の利用
- 監査／ファイングレイン監査の利用
- DB／Audit Vault の利用
- Flashback Data Archive の利用
- データの暗号化

11g以前

ILMの構築

データ区分

スコープ

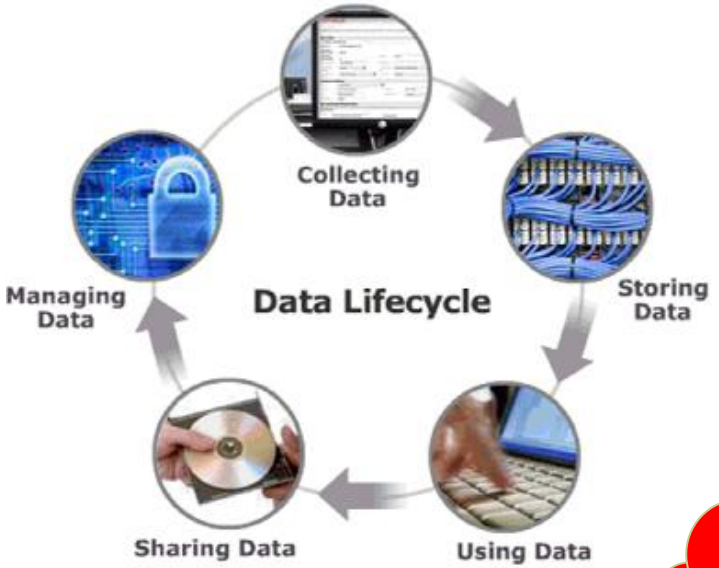
- 表
- 表パーティション

経過時間に応じて決められた処理

- ステージ① :1ヶ月経過後
  - OLTP圧縮
- ステージ② : 2ヶ月経過後
  - HCC圧縮 ＋ 低コスト・ストレージへ移動
- ステージ③ : 11ヶ月経過後
  - HCC圧縮 ＋ さらに低コスト・ストレージへ移動
- 最終ステージ : 6年経過後 ⇒ パージ

ステージ	ステージ⓪	ステージ①	ステージ②	ステージ③	最終ステージ
ストレージ層	高性能	高性能	低コスト	オンライン・アーカイブ	
圧縮属性	なし	OLTP圧縮	HCC圧縮 クエリー・モード	HCC圧縮 アーカイブ・モード	
期間	1カ月間	2か月間	11か月間	6年間	パージ

ILMの課題



安全なデータ

- 規制による要件の増加

データ増
VS
I/O性能

ストレージコスト削減

- 爆発的なデータ増
- 規制による要件の増加

性能改善

ILM ソリューション

- ストレージ・コスト削減
  - 実際のアクセス・パターンに応じて、**最適なストレージ層にデータを配置**することで、より効果的なストレージ管理
  - **圧縮の利用**による許容データ量の拡大
- 性能改善
  - 実際のアクセス・パターンに応じて、**圧縮レベルの使い分け**

適切なタイミングで実施

Automatic Data Optimization

- ユーザ定義のルール（ポリシー）に従って**圧縮**
- 表領域が逼迫したら、利用頻度の低いものから**移動**

In-Database Archiving

- 行単位での**アーカイブ属性**
- 行単位での**有効期間属性**

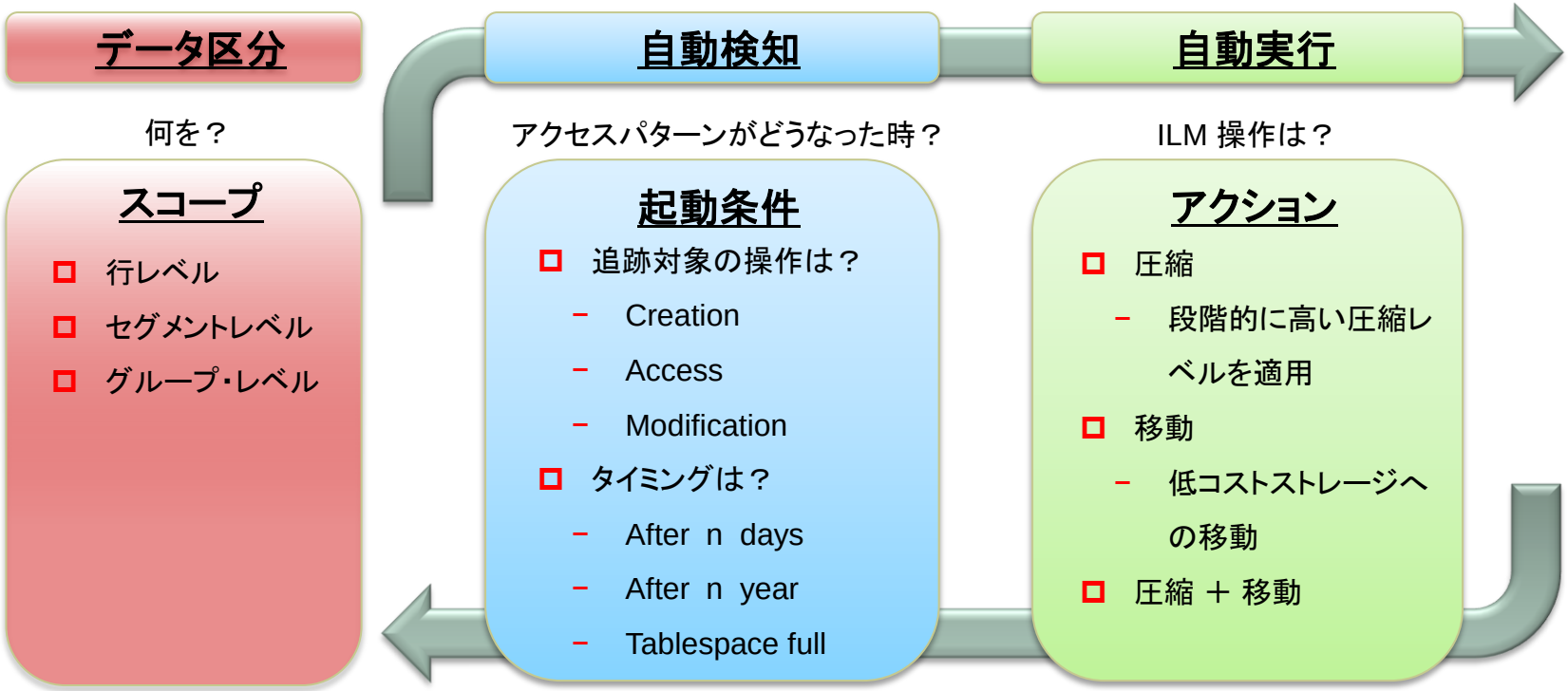
ILM アクションの自動化

時間経過と共にアクセス頻度が低下するデータ

ILMの構築

12c

Automatic Data Optimization ／ Heat Map



ILMの新機能
～データの移動・圧縮の自動化～

ILM の新機能　～データの移動・圧縮の自動化～

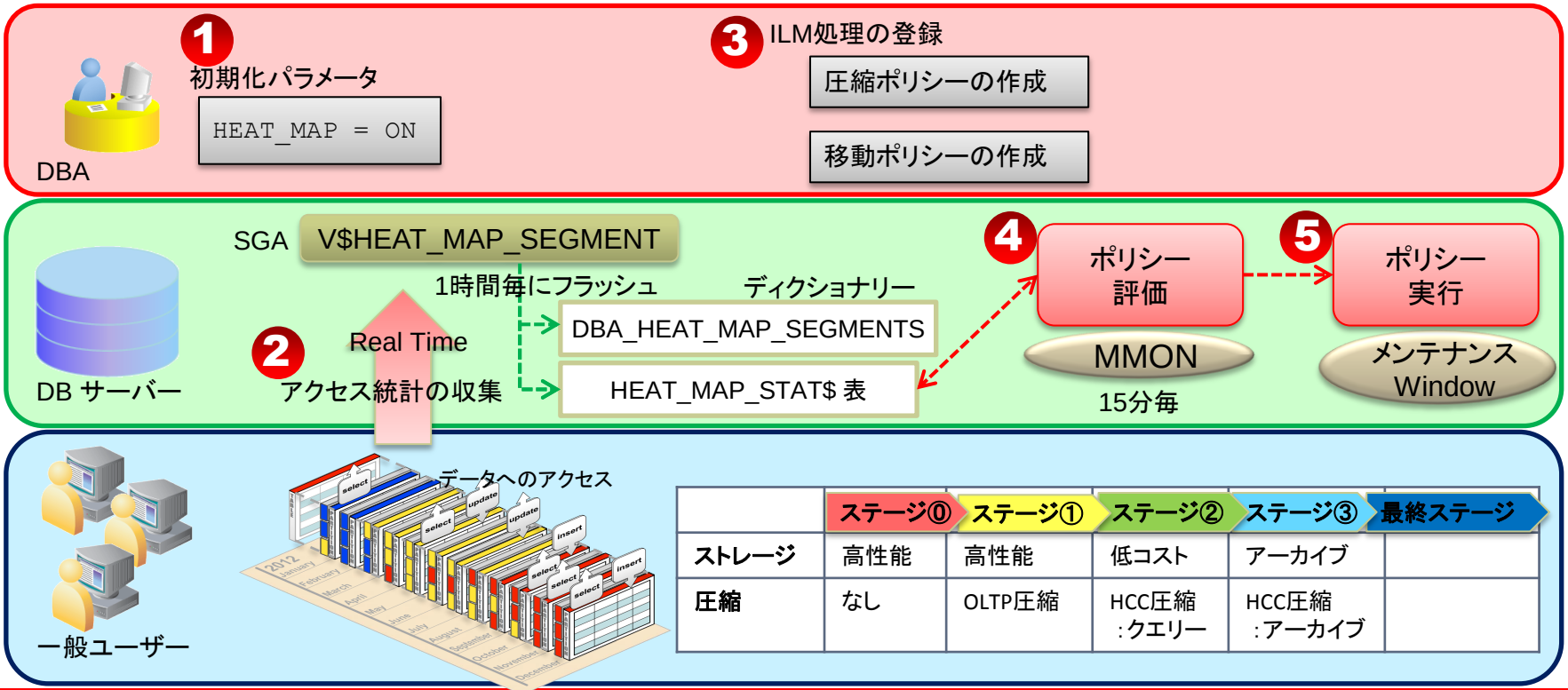
- **Automatic Data Optimization**
  - ILMアクション（圧縮／移動）と、そのアクションをどのタイミングで実行するかをポリシーとして設定し自動実行
- **Heat Map**
  - データへのアクセスパターン、アクセス頻度を記録・管理

Automatic Data Optimization とは
概要

- **Automatic Data Optimization**（以下ADO）では、ILM処理（移動・圧縮）と起動条件をポリシーとして定義し、ILM処理を自動実行
  - 移動は表領域の利用率に応じて実施
  - 圧縮はアクセスパターンの変化に応じて実施
  - 実施条件は上記のデフォルトの記述方法に加え、PL/SQLファンクションを使ったカスタム条件の利用も可能
- **Heat Map**機能では、データへのアクセスパターンとそのアクセスが最後に発生した日時を記録
  - ADOポリシーが起動タイミングを自動検知するために利用

Automatic Data Optimization とは

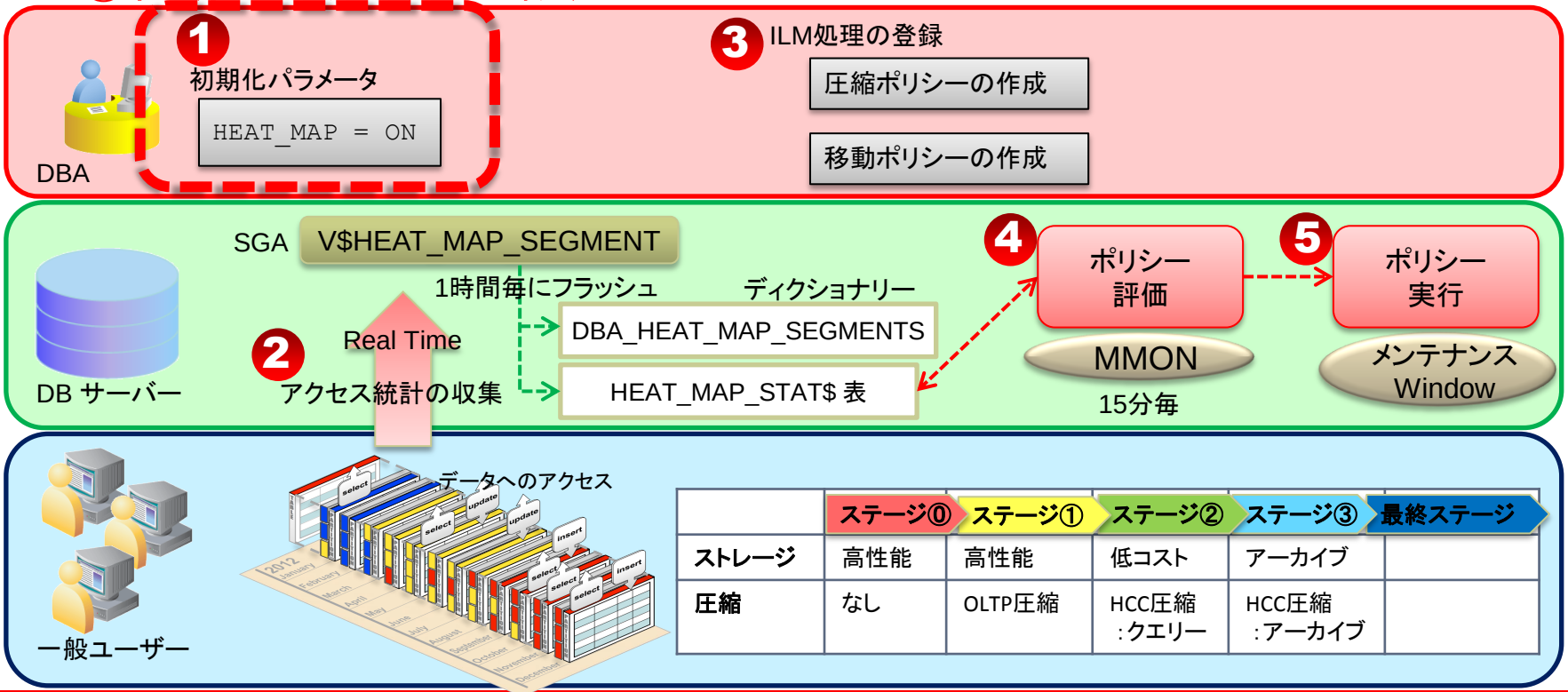
動作の仕組み



	ステージ⓪	ステージ①	ステージ②	ステージ③	最終ステージ
ストレージ	高性能	高性能	低コスト	アーカイブ	
圧縮	なし	OLTP圧縮	HCC圧縮 :クエリー	HCC圧縮 :アーカイブ	

Automatic Data Optimization とは

①初期化パラメータの設定



	ステージ⓪	ステージ①	ステージ②	ステージ③	最終ステージ
ストレージ	高性能	高性能	低コスト	アーカイブ	
圧縮	なし	OLTP圧縮	HCC圧縮 :クエリー	HCC圧縮 :アーカイブ	

Automatic Data Optimization とは

①初期化パラメータの設定

- Heat Map 情報の収集を有効にするためSYSTEMレベルで初期化パラメータを設定

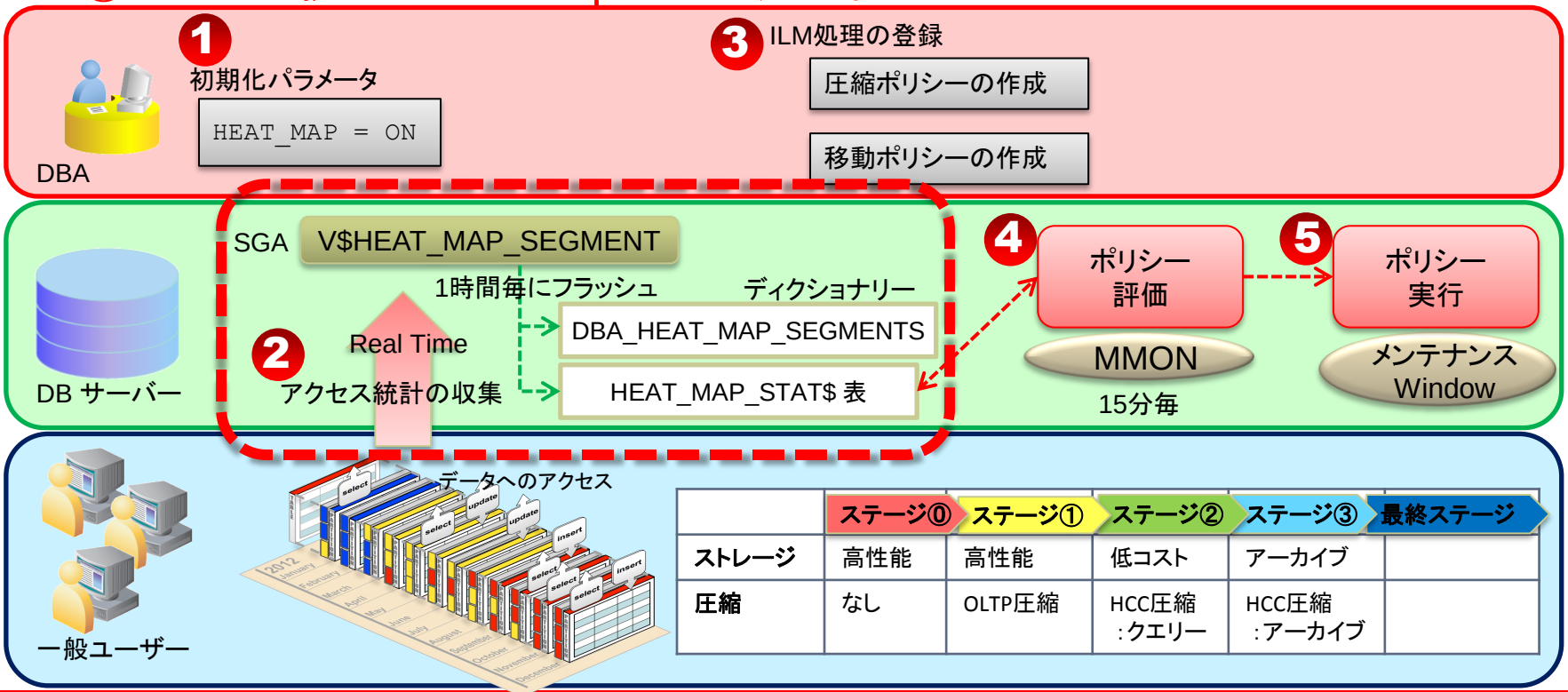
```
SQL> ALTER SYSTEM SET heat_map = ON;
```

- この設定により **SYSTEM および SYSAUX 表領域以外**に格納されているオブジェクトへのアクセス追跡を開始

- セッションレベルでも設定可能
  - この場合は、アクセス統計の収集（Heat Map機能）のみが有効化／無効化

Automatic Data Optimization とは

②アクセス統計（Heat Map）の自動収集

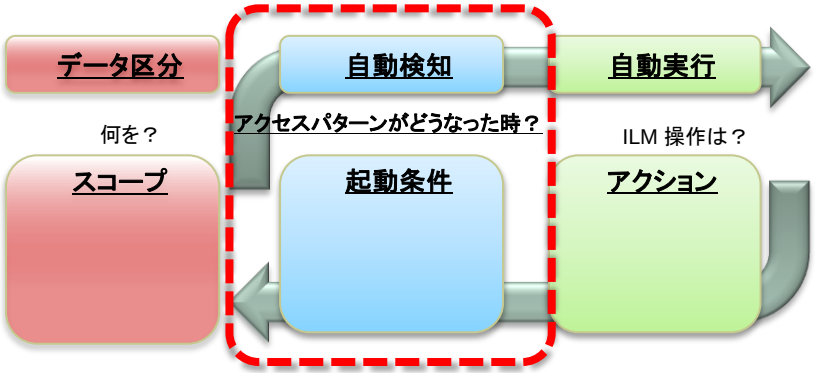


	ステージ⓪	ステージ①	ステージ②	ステージ③	最終ステージ
ストレージ	高性能	高性能	低コスト	アーカイブ	
圧縮	なし	OLTP圧縮	HCC圧縮 :クエリー	HCC圧縮 :アーカイブ	

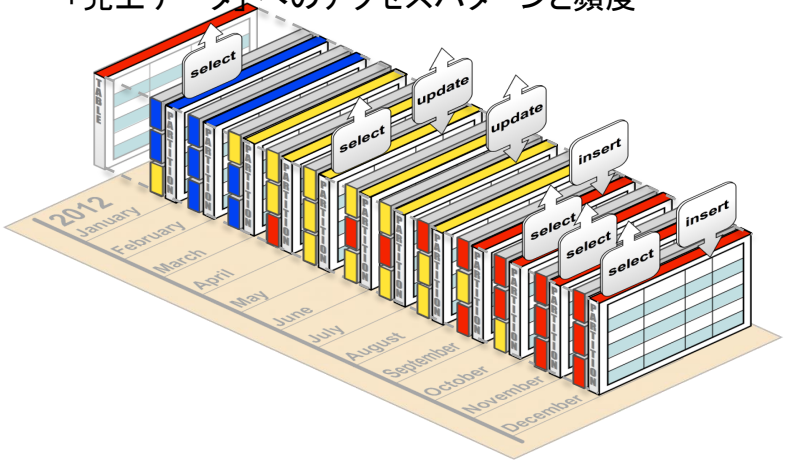
Automatic Data Optimization とは

②アクセス統計（Heat Map）の自動収集

- 初期化パラメータ HEAT_MAP=ON の設定レベルによる設定範囲の違い
  - SYSTEMレベル ⇒ :ADOの機能も同時に有効化／無効化
  - SESSIONレベル ⇒ :Heat Map機能のみ有効化／無効化



「売上データ」へのアクセスパターンと頻度



Automatic Data Optimization とは

②Heat Map情報のモニター：セグメント・レベル

View によるモニター

- DBA_HEAT_MAP_SEG_HISTOGRAM
- DBA_HEAT_MAP_SEGMENT
- V$HEAT_MAP_SEGMENT

```
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
INTERVAL_SALES              P2 28-MAR-12  NO  YES NO
INTERVAL_SALES              P3 28-MAR-12  NO  NO  YES
I_EMPNO                        07-dec-12  YES NO  YES
```

Automatic Data Optimization とは

②Heat Map情報のモニター：セグメント・レベル

- セグメント・レベルの最終アクセス日時

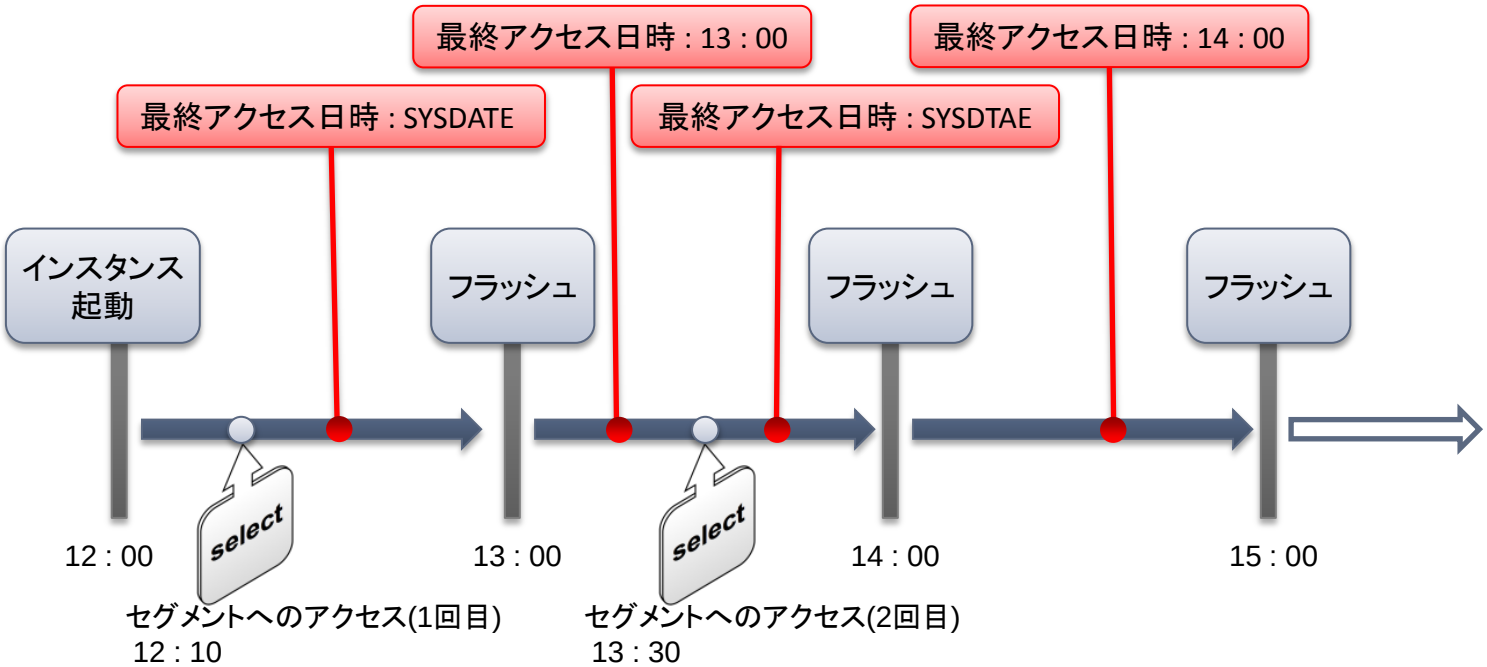
```
SQL> SELECT object_name,
  2       segment_write_time WRITE_T, segment_read_time READ_T,
  3       full_scan FTS_T, lookup_scan LKP_T
  4  FROM DBA_HEAT_MAP_SEGMENT;

OBJECT_NAME WRITE_T   READ_T    FTS_T     LKP_T
----------- --------- --------- --------- ---------
EMP                             07-dec-12
T1          07-dec-12           08-dec-12
EMPLOYEE    07-dec-12           08-dec-12 08-dec-12
I_EMPNO     07-dec-12                     08-dec-12
```

- 1時間毎にV$HEAT_MAP_SEGMENTの情報がDBA_HEAT_MAP_SEGMENTへフラッシュされるためDBA_HEAT_MAP_SEGMENTより戻される最終アクセス日時の値は、最終アクセス後にフラッシュが行われた日時となる
- DBA_HEAT_MAP_SEGMENTを問い合わせた時点で、最終アクセス後にフラッシュが実施されていない場合は問い合わせ時点の日時(SYSDATE)が戻される

Automatic Data Optimization とは

②Heat Map情報のモニター：セグメント・レベル

- DBA_HEAT_MAP_SEGMENT より戻される最終アクセス日時の例：



Automatic Data Optimization とは

②Heat Map情報のモニター：ブロック・レベル

PL/SQL関数によるモニター

- 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
```

Automatic Data Optimization とは

②Heat Map情報のモニター：エクステント・レベル

PL/SQL関数によるモニター

- DBMS_HEAT_MAP.EXTENT_HEAT_MAP

```
SQL> SELECT segment_name, block_id, blocks, max_writetime
  2  FROM table(dbms_heat_map.extent_heat_map
  3                                    ('SCOTT','EMPLOYEE'));

SEGMENT_    BLOCK_ID    BLOCKS MAX_WRITETIME
-------- ---------- ---------- ------------------
EMPLOYEE         136          8 12-DEC-12 12:58:46
EMPLOYEE         144          8 12-DEC-12 12:58:46
EMPLOYEE         256        128 12-DEC-12 12:58:47
EMPLOYEE         384        128 12-DEC-12 12:58:47
```

Automatic Data Optimization とは

②Heat Map情報のモニター：アクセス・ランキング（表領域・レベル）

```
SQL> SELECT TABLESPACE_NAME, SEGMENT_COUNT
  2          , ALLOCATED_BYTES, MAX_WRITETIME
  2          , MAX_READTIME, MAX_FTSTIME, MAX_LOOKUPTIME
  3  FROM DBA_HEATMAP_TOP_TABLESPACES;

TABLESPACE_NAME      SEGMENT_COUNT ALLOCATED_BYTES MAX_WRIT MAX_READ MAX_FTST MAX_LOOK
-------------------- ------------- --------------- -------- -------- -------- --------
HIGH_PERF_TBS                    3        33554432 13-06-04 13-06-04 13-06-04 13-05-31
USERS                           19        76349440 13-06-04 13-06-03 13-06-04 13-06-03
```

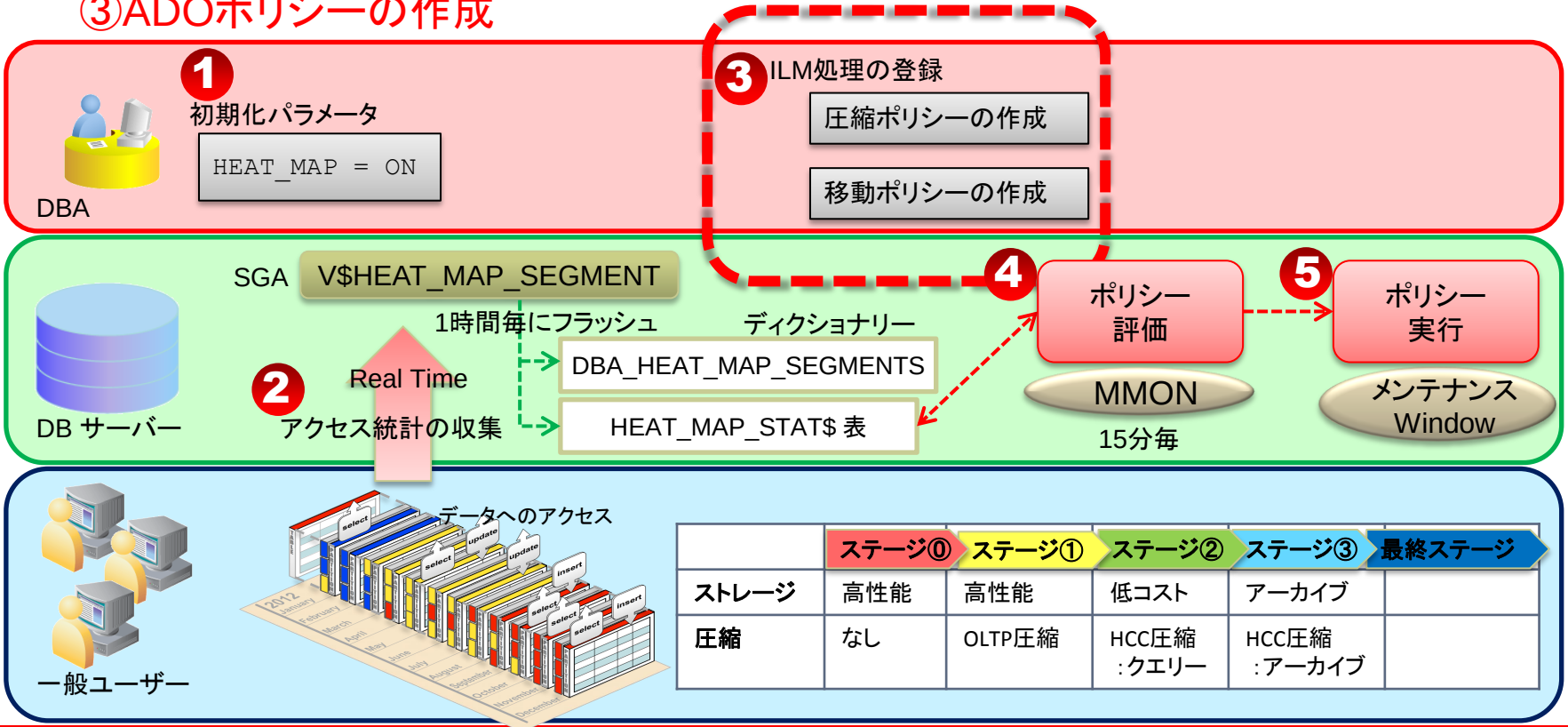
Automatic Data Optimization とは

②Heat Map情報のモニター：アクセス・ランキング（オブジェクト・レベル）

```
SQL> SELECT OWNER, OBJECT_NAME, SEGMENT_COUNT
  2          , OBJECT_SIZE, MAX_WRITETIME, MAX_READTIME
  3          , MAX_FTSTIME, MAX_LOOKUPTIME
  4   FROM DBA_HEATMAP_TOP_OBJECTS;

OWNER      OBJECT_NAME   SEGMENT_COUNT OBJECT_SIZE MAX_WRIT MAX_READ MAX_FTST MAX_LOOK
---------- ------------ ------------- ----------- -------- -------- -------- --------
ILM        SALES3_PT                2          16 13-06-04 13-06-04
ILM        NOTMOVE                  1          15 13-06-04
ILM        SALES3_PT                2          16 13-06-04 13-06-04
MIGUSER    SALGRADE                 1       .0625 13-06-03
MIGUSER    EMP                      1       .0625 13-06-03
MIGUSER    DUMMY                    1       .0625 13-06-03
MIGUSER    DEPT                     1       .0625 13-06-03
```

Automatic Data Optimization とは

③ADOポリシーの作成

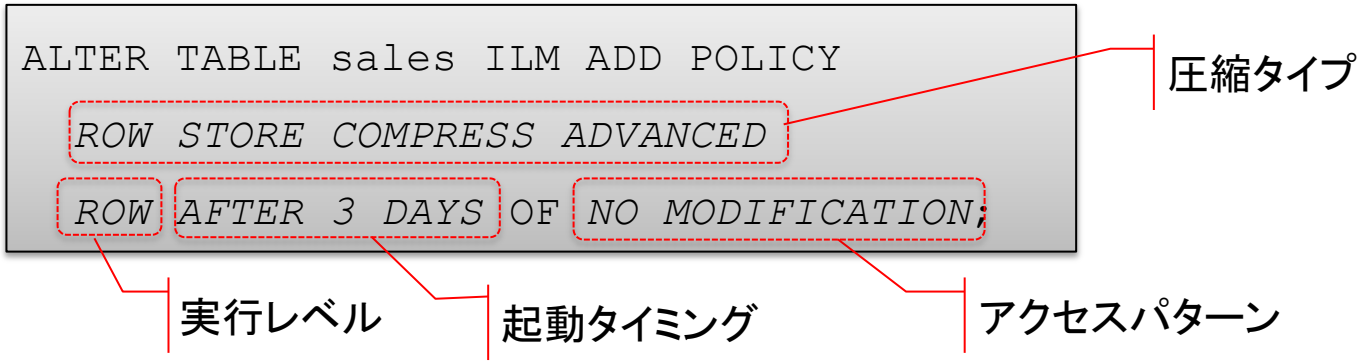


	ステージ⓪	ステージ①	ステージ②	ステージ③	最終ステージ
ストレージ	高性能	高性能	低コスト	アーカイブ	
圧縮	なし	OLTP圧縮	HCC圧縮 :クエリー	HCC圧縮 :アーカイブ	

Automatic Data Optimization とは

③ADOポリシーの作成

- **宣言型ポリシー設定: （圧縮）**



圧縮タイプ	実行レベル	アクセスパターン	起動タイミング
•圧縮タイプの指定	•処理対象（スコープ）	•アクセスの状態	•指定された「アクセスパターン」となってからの時間

Automatic Data Optimization とは

③ADOポリシーの作成

- **宣言型ポリシー設定: （圧縮）**

```
ALTER TABLE sales ILM ADD POLICY
  ROW STORE COMPRESS ADVANCED
  ROW AFTER 3 DAYS OF NO MODIFICATION;
```

圧縮タイプ

圧縮タイプ

- 圧縮タイプの指定
  - ROW STORE COMPRESS（Basic 圧縮）
  - ROW STORE COMPRESS ADVANCED（Advanced Row 圧縮）
  - COLUMN STORE COMPRESS FOR QUERY LOW/HIGH（HCC Query モード）
  - COLUMN STORE COMPRESS FOR ARCHIVE LOW/HIGH（HCC Archive モード）

Automatic Data Optimization とは

③ADOポリシーの作成

- **宣言型ポリシー設定: （圧縮）**

```
ALTER TABLE sales ILM ADD POLICY
  ROW STORE COMPRESS ADVANCED
  ROW AFTER 3 DAYS OF NO MODIFICATION;
```

実行レベル

- 処理対象（スコープ）の指定
  - Tablespace ---> 格納されるオブジェクトのデフォルトとして設定される
  - Group ---> 対象オブジェクトの索引やLOBも含めてILM 処理を適用する
  - Segment ---> 表／パーティション／サブパーティション
  - Row ---> ブロック単位の処理（Advanced Row圧縮のみ指定可能）

Automatic Data Optimization とは

③ADOポリシーの作成

- 宣言型ポリシー設定: （圧縮）

```
ALTER TABLE sales ILM ADD POLICY
  ROW STORE COMPRESS ADVANCED
  ROW AFTER 3 DAYS OF NO MODIFICATION;
```

アクセスパターン

アクセスパターン

- アクセスの状態
  - NO MODIFICATION ---> Insert/Updata/Delete が無い
  - NO ACCESS ---> Insert/Update/Delete/Select が無い
  - CREATION ---> セグメントの作成
  - ・・・

Automatic Data Optimization とは

③ADOポリシーの作成

- **宣言型ポリシー設定: （圧縮）**

```
ALTER TABLE sales ILM ADD POLICY
  ROW STORE COMPRESS ADVANCED
  ROW AFTER 3 DAYS OF NO MODIFICATION;
```

起動タイミング

起動タイミング

- 指定された「アクセスパターン」となってからの時間
  - n DAY[s]
  - n MONTH[s]
  - n YEAR[s]

Automatic Data Optimization とは

③ADOポリシーの作成

- **宣言型ポリシー設定: （移動）**

```
ALTER TABLE sales ILM ADD POLICY
  TIER TO Low_Cost_tbs;
```

移動タイプ

移動先

移動タイプ	移動先
•TIER TO のみ	•表領域を指定

Automatic Data Optimization とは

③ADOポリシーの作成

- 宣言型ポリシー設定: （カスタム・ポリシー）

```
ALTER TABLE emp ILM ADD POLICY
  ROW STORE COMPRESS ADVANCED
  SEGMENT ON custom_ilm_rules;
```

ユーザー定義のPL/SQL関数を利用

```
ALTER TABLE sales ILM ADD POLICY
  TIER TO users ON custom_ilm_rules;
```

PL/SQLを使って起動条件をカスタマイズ

Automatic Data Optimization とは

③ADOポリシーの作成

- **ポリシーの継承**
  - 表領域に設定されたポリシー
    - 格納される表・パーティションに継承（既存の表・パーティションには継承されない）
    - 表・パーティションに作成されているポリシーが優先
  - 表に設定されたポリシー
    - パーティションに継承（既存のパーティションにも継承される）
    - パーティションに作成されているポリシーが優先

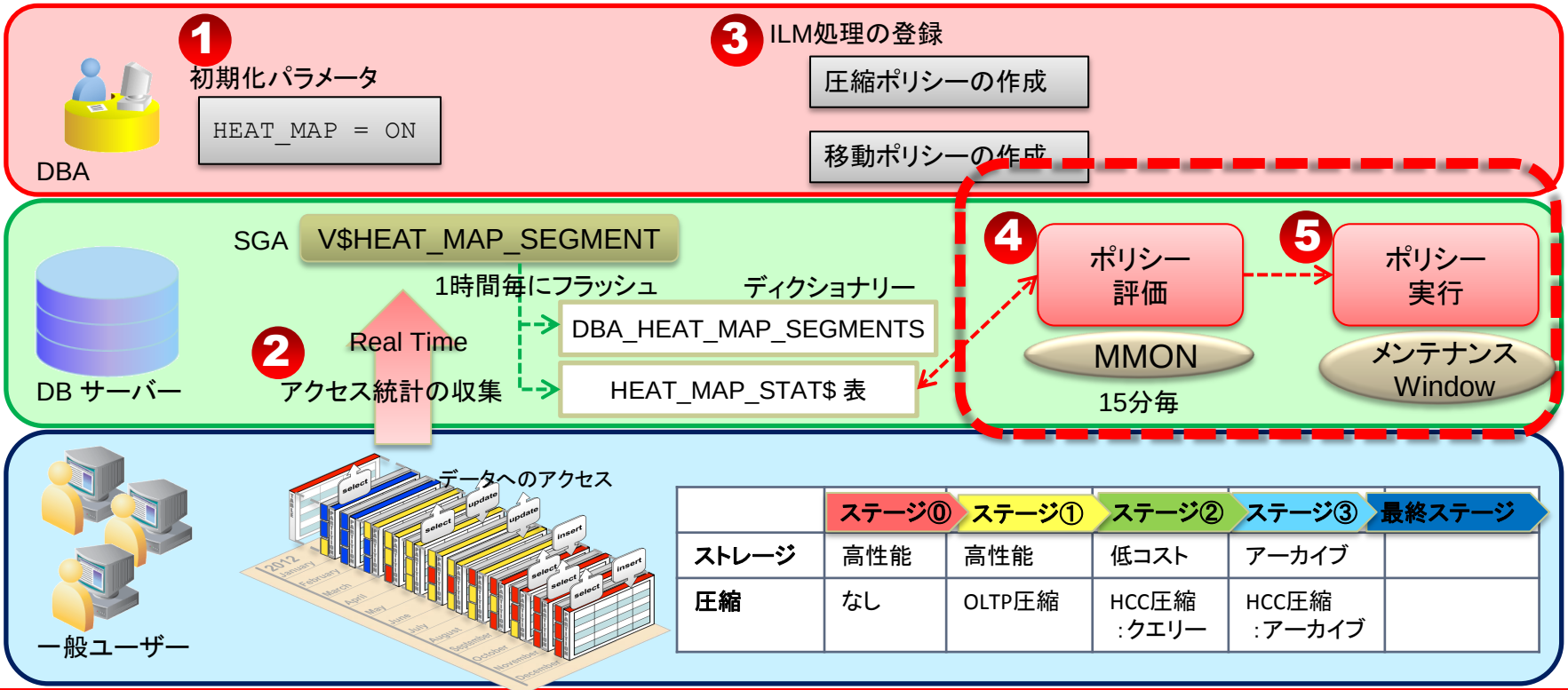
Automatic Data Optimization とは

設定可能な アクション と アクセスパターン の組合せ

Act (Scope) \ Acc Pattern	No modification	Creation	No Access	Tablespace fullness	Custom Policy
Compression (Row)	✔				
Compression (Segment)	✔		✔		✔
Compression (Group)	✔		✔		
Compression (Tablespace)	✔	✔	✔		
Storage Tiering (Segment)				✔	✔
Storage Tiering (Tablespace)				✔	

Automatic Data Optimization とは

④ポリシーの評価・実行



	ステージ⓪	ステージ①	ステージ②	ステージ③	最終ステージ
ストレージ	高性能	高性能	低コスト	アーカイブ	
圧縮	なし	OLTP圧縮	HCC圧縮 :クエリー	HCC圧縮 :アーカイブ	

Automatic Data Optimization とは

④ポリシーの評価・実行

- **自動実行**
  - メンテナンスWindow
- **強制実行**
  - DBMS_ILM.EXECUTE_ILM(

```
owner                IN      VARCHAR2,
object_name          IN      VARCHAR2,
task_id              OUT     NUMBER,
subobject_name       IN      VARCHAR2 DEFAULT NULL,
policy_name          IN      VARCHAR2 DEFAULT ilm_all_policies,
execution_mode       IN      NUMBER DEFAULT ilm_execution_online)
```

Automatic Data Optimization とは

④ポリシーの評価・実行

- DBA_ILMRESULTS

```
SQL> SELECT task_id, policy_name, object_name, subobject_name,
  2         selected_for_execution, job_name
  3  FROM DBA_ILMEVALUATIONDETAILS
  4  WHERE object_name = 'SALES2_PT'
  5  ORDER BY task_id;

   TASK_ID POLICY_NAME     OBJECT_NAME     SUBOBJECT_NAME       SELECTED_FOR_EXECUTION          JOB_NAME
---------- --------------- --------------- -------------------- ------------------------------ ----------
      3580 P1734           SALES2_PT       SALES2_PT_Q1_2012    SELECTED FOR EXECUTION         ILMJOB4762
      3580 P1734           SALES2_PT       SALES2_PT_Q4_2012    PRECONDITION NOT SATISFIED
      3580 P1734           SALES2_PT       SALES2_PT_Q3_2012    PRECONDITION NOT SATISFIED
      3580 P1734           SALES2_PT       SALES2_PT_Q2_2012    PRECONDITION NOT SATISFIED
```

Automatic Data Optimization とは

④ポリシーの評価・実行

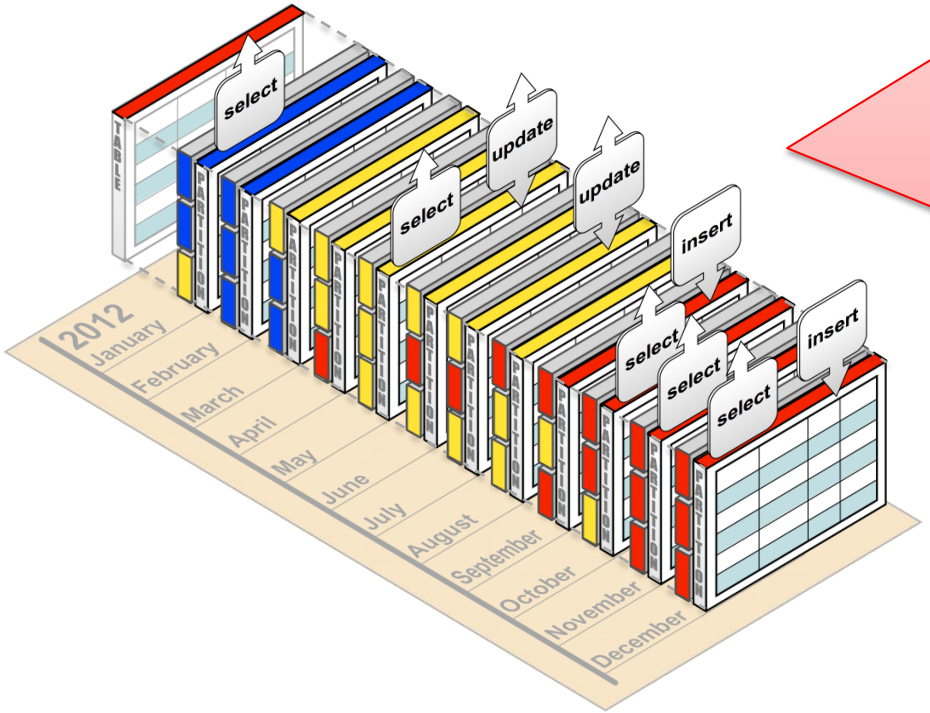
- DBA_ILMRESULTS

```
SQL> SSELECT task_id, job_name, job_state
  2  ,TRUNC(start_time) start_time
  3  ,TRUNC(completion_time) completion_time
  4  FROM DBA_ILMRESULTS
  5  WHERE job_name IN
  6    (SELECT job_name FROM DBA_ILMEVALUATIONDETAILS
  7       WHERE object_name = 'SALES2_PT')
  8  ORDER BY job_name;


   TASK_ID JOB_NAME      JOB_STATE                 START_TIME COMPLETION
---------- ------------- ------------------------- ---------- ----------
      3446 ILMJOB4566    COMPLETED SUCCESSFULLY 13-05-27   13-05-27
```

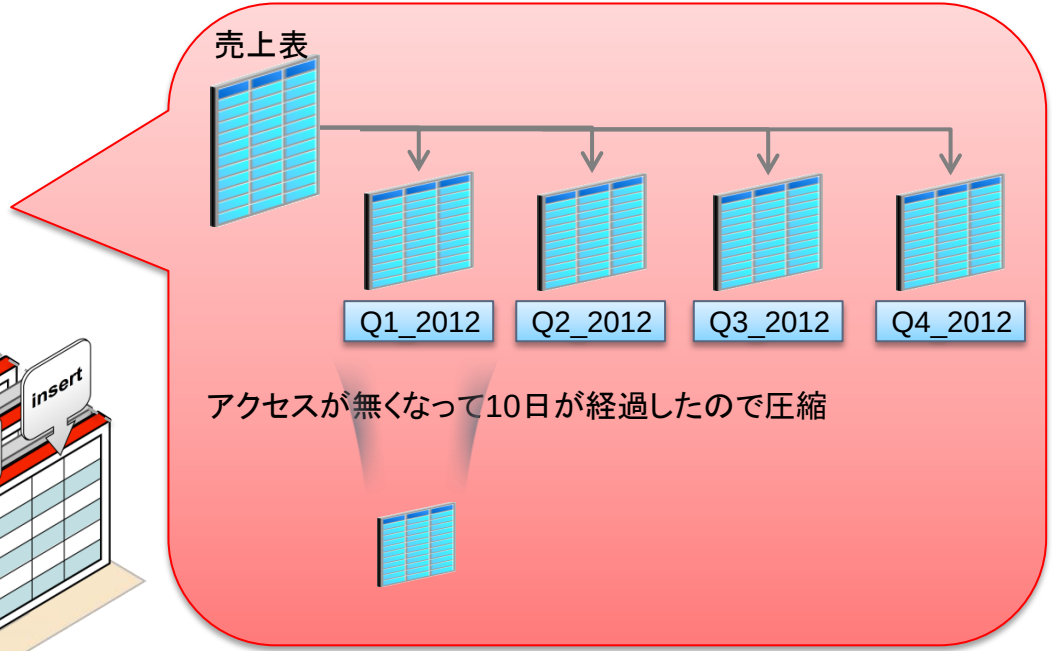
DEMONSTRATION

デモ・シナリオ

「売上データ」へのアクセスパターンと頻度



古いデータになるほどアクセス頻度が少なくなる



ILMの新機能
～データへのアクセス制御～

ILM の新機能　～データへのアクセス制御～

- **In-Database Archiving**
  - オンライン状態を維持したまま通常セッションからはアクセス制限
- **Temporal Validity**
  - 属性として有効期間を持つデータに対するアクセス制限

In-Database Archiving

機能概要

- 行単位にアーカイブ属性を設定
  - アーカイブ属性に設定された行は表示されない
  - アーカイブ属性の行を表示するには、セッションパラメータを設定

- アプリケーション性能を犠牲にすることなく、より長期間オンラインで保持
  - アーカイブ属性の行は、暗黙的にフィルタリング
  - アーカイブ属性のデータは、更新要求は無いはずなので、積極的に圧縮することが可能

  - ※参照:「Oracle Database VLDB and Partitioning Guide」

In-Database Archiving

- ORA_ARCHIVE_STATE カラムによって各行のアーカイブ属性を保持

```
SQL> CREATE TABLE emp
  2     (EMPNO NUMBER(7), FULLNAME VARCHAR2(40),
  3      JOB VARCHAR2(9), MGR NUMBER(7))
  4   ROW ARCHIVAL;
```

- ORA_ARCHIVE_SATE：Insertの際 0（デフォルト値）が設定される
- アーカイブ属性とするためには、ORA_ARCHIVE_STATE を 1 に更新

In-Database Archiving

- アーカイブ属性に更新

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

- 通常セッションでの表示（アクティブなデータのみ）

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

ORA_ARCHIVE_STATE FULLNAME
----------------- -----------------
                0 JEAN
```

アクティブ

In-Database Archiving

- アーカイブ属性データにアクセスするには

```
SQL> alter session set ROW ARCHIVAL VISIBILITY = ALL;
```

- アーカイブ属性データを表示

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

ORA_ARCHIVE_STATE FULLNAME
----------------- -----------------
                0 JEAN
                1 ADAM
                1 TOM
```

ILM の新機能　～データへのアクセス制御～

- **In-Database Archiving**
  - オンライン状態を維持したまま通常セッションからはアクセス制限
- **Temporal Validity**
  - 属性として有効期間を持つデータに対するアクセス制限

Temporal Validity

機能概要

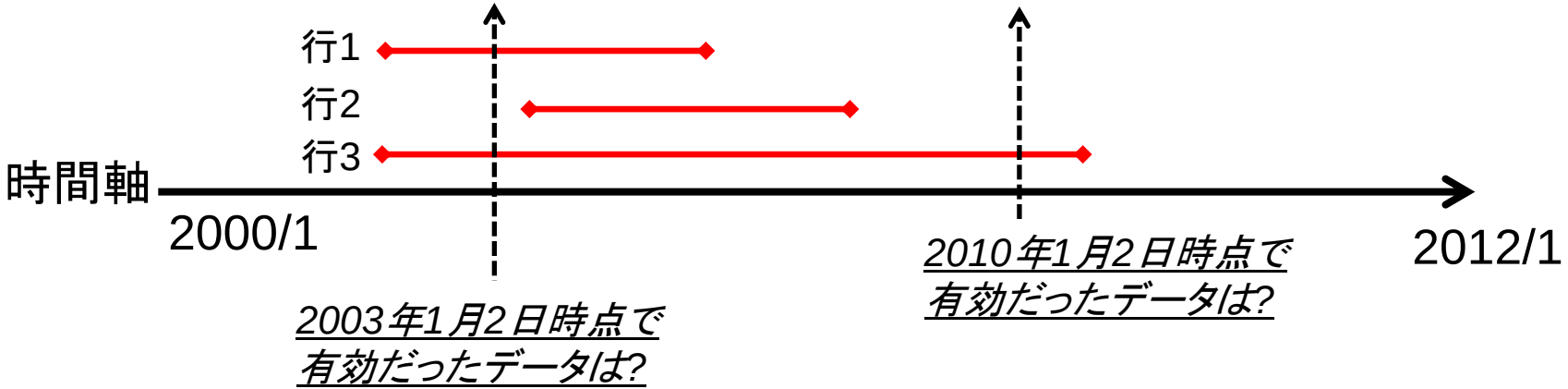
- 有効期間を持つデータ（クレジットカード、免許情報など）のハンドリングに有効
- 有効期間にあるデータだけが検索対象
  - 任意の時点で有効なデータ
  - 任意の期間で有効なデータ

アプリケーションの作成工数を軽減

Temporal Validity

機能概要

- Temporal Validityでは、有効期間を示す列を定義することにより、ある地点で有効なデータを検索することが可能
  - 2003年1月2日 ～ 2010年1月2日の間に有効なデータ: **行3のみ**
  - 2003年1月2日の時点有効なデータ: **行1、行3**



設定及び検索の方法

- 有効期間を示す列の設定
  - ユーザー定義の列に対して設定する方法
  - 隠し列として設定する方法

- 検索の方法
  - as of句を利用する

ユーザー定義の列に対して設定する方法

- 実際のテーブル定義を指定する方法
  - period for句で有効期間名及び、列名を指定する

```
CREATE TABLE emp_temp_model(
 empno              number,
 salary             number,
 deptid             number,
 name               varchar2(100),
user_time_start     timestamp,
user_time_end       timestamp,
     PERIOD FOR user_time(user_time_start, user_time_end));
```

隠し列として設定する方法

- ユーザーに見えない隠し列を使う
  - period for句で有効期間名(prefix)を指定する
  - 「有効期間名_start」と「有効期間名_end」という2つの隠し列が生成される
    - データ型はtimestamp型
    - 列名を指定して作成することはできない
  - これらの列に対する検索は通常通り可能

隠し列として設定する方法

設定例

```
SQL> CREATE TABLE emp(
  2  ename       char(20),
  3  dept        number,
  4    PERIOD FOR user_time);

表が作成されました。

SQL> desc emp
 名前                                      NULL?    型
 ----------------------------------------- -------- ----------------------------
 ENAME                                              CHAR(20)
 DEPT                                               NUMBER

SQL> INSERT INTO emp(ename, dept, user_time_start, user_time_end)
  2  VALUES('MORI', 20, sysdate - 500, sysdate + 500);

1行が作成されました。
```

検索方法

- **AS OF PERIOD FOR 有効期間名 ‘日付’**
  - ある有効期間名の中で、該当する日付で有効なデータを検索する

```
SELECT * FROM emp AS OF PERIOD FOR user_time '11-12-31';
```

- **VERSIONS PERIOD FOR ‘日付’ AND ‘日付2’**
  - ある有効期間名の中で、該当する期間の中で有効なデータを検索する

```
SELECT * FROM emp VERSIONS PERIOD FOR user_time
BETWEEN TO_DATE('2012-01-01','YYYY-MM-DD')
AND TO_DATE('2012-12-31','YYYY-MM-DD');
```

Temporal Validity vs Temporal History

- Temporal Validity

  （Valid-time）

  - 任意の時点／期間で有効なデータを対象にしたクエリー

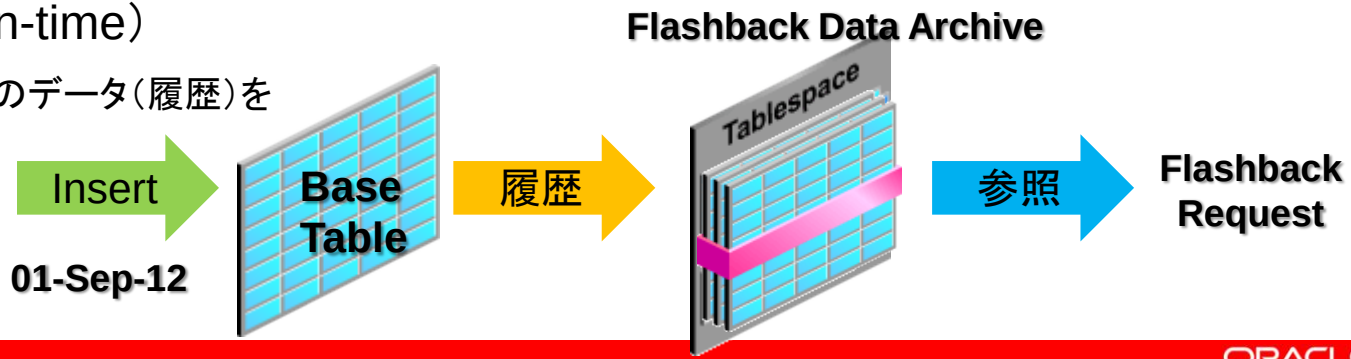
Emp ID	Job	Hire Date	Quit Date
100	Clerk	01-Apr-10	
200	Developer	01-Apr-11	
300	Sales	01-Sep-12	31-May-13

有効期間

- Temporal History

  （Transaction-time）

  - 任意の時点のデータ（履歴）を参照



制限事項

ILM 新機能 の制限事項

- ADO および Heat Map は、CDB/PDB では非サポート
- 処理対象が Row のADOポリシーは、Temporal Validity では非サポート
  - 期間の終端を示す列でパーティション化されているセグメント・レベルのポリシーの場合はサポート
- 処理対象が Row のADOポリシーは、In-Database Archiving では非サポート
  - ただし ORA_ARCHIVE_STATE 列でパーティション化されているセグメント・レベルのポリシーの場合はサポート
- カスタム・ポリシー（ユーザー定義の関数利用）は、表領域レベルでは非サポート
- ADOは、移動を実施する際、移動先表領域の空き容量をチェックしない
- ADOは、オブジェクト型を持つ表やマテリアライズド・ビューでは非サポート
- ADOのジョブ実行の同時実行性は、Oracle Scheduler の同時実行性に依存
  - ADOのジョブが2回以上失敗した場合、ジョブが無効とされるため、手動メンテナンスが必要
- ADOのポリシー（処理対象が Row 以外）は、メンテナンスWindow のオープン期間内のみで評価・実行
  - 期間外でも強制実行は可能
- 移動を行うADOポリシーでは、表や表パーティションのMOVE句と同じ制約を受けます
- Temporal Validity は、CDB/PDBでは非サポート

パーティションの新機能

パーティションの新機能

- 複数パーティションへの対応
- パーシャル索引
- グローバル索引メンテナンスの非同期実行
- インターバル・リファレンス・パーティショニング
- リファレンス・パーティションの CASCADE 処理
- オンラインでのパーティションの移動
- XMLIndex の対応拡張

複数パーティションへの対応

一つのコマンドで複数パーティションへをメンテナンス

11g 複数のパーティションのメンテナンスは
パーティション毎にコマンド実行が必要

(ex) 3つのパーティションの削除は
DROP PARTITION を**3回実行**

drop partition

drop partition

drop partition

パーティション表

- Partition P01
- Partition P02
- Partition P03
- Partition P04

12c 複数のパーティションのメンテナンスを
一つのコマンドで一括して実行可能

(ex) 3つのパーティションの削除は
DROP PARTITION を**1回実行**

拡張された処理
DROP / ADD / TRUNCATE / MERGE / SPLIT

drop partition

パーティション表

- Partition1
- Partition2
- Partition3
- Partition4

複数パーティションへの一括処理

実行コマンドの例

- ADD PARTITION

```
ALTER TABLE parttab ADD
PARTITION p11 VALUES LESS THAN (111),
PARTITION p12 VALUES LESS THAN (121),
PARTITION p13 VALUES LESS THAN (131);
```

- DROP PARTITION

```
ALTER TABLE parttab DROP PARTITION p02, p03, p04;
```

- TRUNCATE PARTITION

```
ALTER TABLE parttab TRUNCATE PARTITION p01, p02, p03;
```

複数パーティションへの一括処理

実行コマンドの例

- MERGE PARTITION

```
ALTER TABLE parttab MERGE PARTITIONS p11, p12, p13
INTO PARTITION p10;
```

- SPLIT PARTITION

```
ALTER TABLE parttab SPLIT p10 INTO
( PARTITION p11 VALUES LESS THAN (111),
  PARTITION p12 VALUES LESS THAN (121),
  PARTITION p13 VALUES LESS THAN (131),
  PARTITION p14 );
```

パーシャル索引

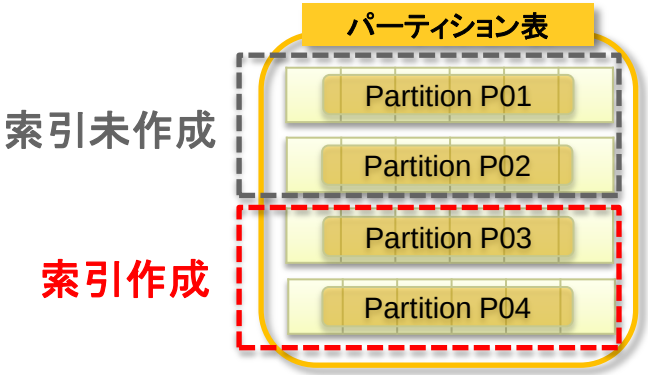
特定のパーティション/サブパーティションに対する索引の作成

- パーシャル索引
  - 特定のパーティションのみに作成する索引

アクセス頻度が低い
パーティション

未使用だが削除不可な
パーティション

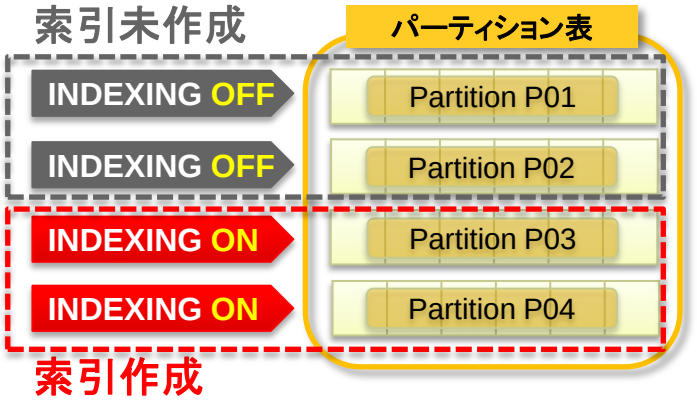
◆ 索引付けを行わない
- DISK領域の削減
- 統計情報取得時間削減



パーシャル索引

パーシャル索引とは

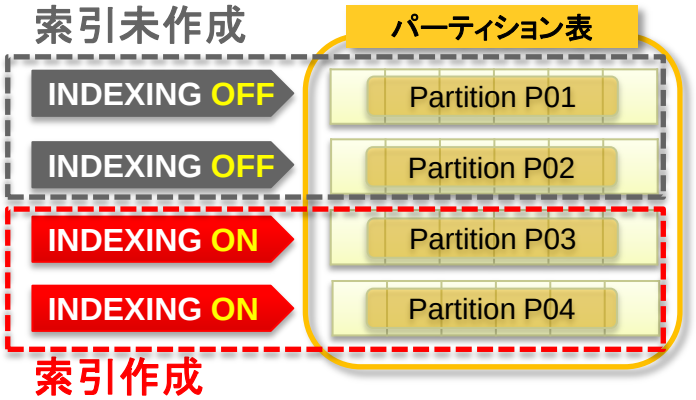
- パーシャル索引
  - パーティション/サブパーティション毎に設定した INDEXING 属性のON / OFF で制御
  - 索引作成時に INDEXING PARTIAL を指定
  - ローカル索引・グローバル索引の両方で使用可能
  - 索引作成後に、INDEXING 属性を変更することも可能
    - 動的な索引付け・排除が可能



パーシャル索引

パーシャル索引の作成方法

- パーシャル索引の作成
  - パーティション表作成時に INDEXING を指定
  - 索引作成時に INDEXING PARTIAL を指定



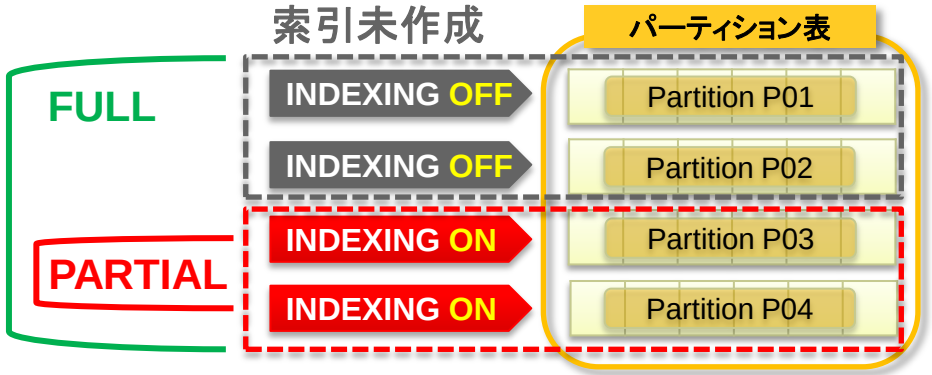

```
CREATE TABLE parttab (c1 number, c2 number)
PARTITION BY RANGE(c1)
(PARTITION p01 VALUES LESS THAN (10001) INDEXING OFF,
 PARTITION p02 VALUES LESS THAN (20001) INDEXING OFF,
 PARTITION p03 VALUES LESS THAN (30001) INDEXING ON,
 PARTITION p04 VALUES LESS THAN (maxvalue));
```

```
CREATE INDEX parttab_ind1 ON parttab(c1) LOCAL INDEXING PARTIAL;
CREATE INDEX parttab_ind2 ON parttab(c2) GLOBAL INDEXING PARTIAL;
```

パーシャル索引

パーシャル索引作成時の指定による違い

- 索引作成時の INDEXING PARTIAL と FULL の違い
  - INDEXING FULL (デフォルト)
    - パーティションの INDEXING 属性に関わらず、すべてのパーティションに索引付けをする
  - INDEXING PARTIAL
    - パーティション/サブパーティションの INDEXING 属性が ON のものにのみ索引付けをする




```
CREATE INDEX parttab_ind1 ON parttab(c1) LOCAL INDEXING PARTIAL;
CREATE INDEX parttab_ind2 ON parttab(c2) LOCAL INDEXING FULL;
```

パーシャル索引

パーティション/サブパーティションの指定と索引の指定

- パーティション表、索引での INDEXING 属性指定による索引の作成状況の違い

索引での指定	パーティション/サブパーティションでの指定		
	INDEXING OFF	INDEXING ON	INDEXING 未指定
INDEXING PARTIAL	✕	●	▲
INDEXING FULL	●	●	●
INDEXING 指定なし	●	●	●

✕ 索引未作成
● 索引作成
▲ デフォルト値に依存

パーシャル索引

単一レベル・パーティションでの INDEXING 属性の指定

表での指定	パーティションでの指定	パーティションに設定される内容
有	有	パーティションに指定された内容で設定
	なし	表で指定された内容で設定
なし	有	パーティションで指定された内容で設定
	なし	デフォルト構成で設定

パーシャル索引

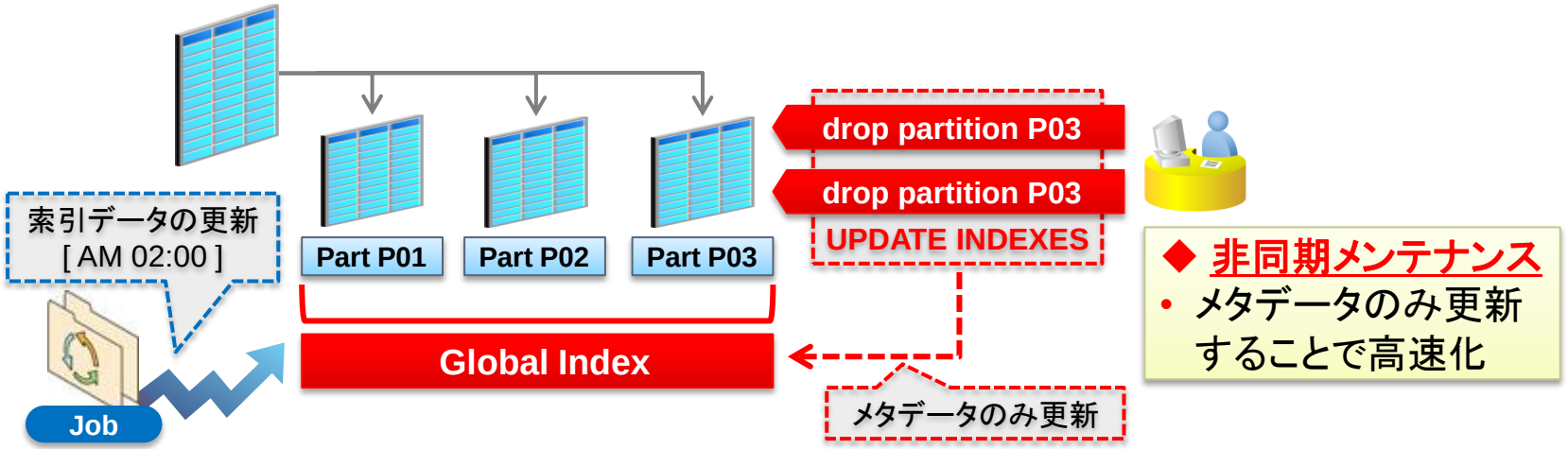
コンポジット・パーティションでの INDEXING 属性の指定

表での指定	パーティションでの指定	サブパーティションでの指定	パーティションに設定される内容	サブパーティションに設定される内容
有	有	有	パーティションに指定された内容で設定	同左
		なし	パーティションで指定された内容で設定	同左
	なし	有	**サブパーティションで指定された内容で設定**	**同左**
		なし	表で指定された内容で設定	同左
なし	有	有	パーティションで指定された内容で設定	同左
		なし	パーティションで指定された内容で設定	同左
	なし	有	**サブパーティションで指定された内容で設定**	**同左**
		なし	デフォルト構成で設定	同左

グローバル索引メンテナンスの非同期実行

DROP/TRUNCATE PARTITION でのグローバル索引のメンテナンス

- DROP/TRUNCATE PARTITION 文で UPDATE INDEXES を指定した場合のグローバル索引のメンテナンス処理を非同期に実行
  - SYS.PMO_DEFERRED_GIDX_MAINT_JOB のジョブが索引データを更新
  - UPDATE INDEXES 未指定時は、11g と同様に索引が UNUSABLE となる



グローバル索引メンテナンスの非同期実行

DROP/TRUNCATE PARTITION でのグローバル索引のメンテナンス

- 自動メンテナンス
  - ジョブにより毎日 AM2時にメンテナンス処理を実行
  - ジョブのスケジュールの手動での変更は可能
    - ジョブの削除は非推奨

- 対象外のケース：11g と同様の動作となる
  - オブジェクト型を含む表
  - ドメイン索引を含む表
  - SYS スキーマの表
    - ユーザが新規に作成した表も含む

グローバル索引メンテナンスの非同期実行

DROP/TRUNCATE PARTITION でのグローバル索引のメンテナンス

- 手動でのメンテナンス
  - 一括メンテナンス (DB全体)

```
SQL> EXECUTE DBMS_PART.CLEANUP_GIDX;
```

  - スキーマ単位でのメンテナンス

```
SQL> EXECUTE DBMS_PART.CLEANUP_GIDX ('SCOTT');
```

  - 表単位でのメンテナンス

```
SQL> EXECUTE DBMS_PART.CLEANUP_GIDX ('SCOTT', 'PARTTAB');
```

  - 索引単位でのメンテナンス

```
SQL> ALTER INDEX parttab_ind_global COALESCE CLEANUP;
```

  - ジョブの実行

```
SQL> EXECUTE DBMS_SCHEDULER.RUN_JOB('PMO_DEFERRED_GIDX_MAINT_JOB');
```

グローバル索引メンテナンスの非同期実行

DROP/TRUNCATE PARTITION でのグローバル索引のメンテナンス

- メンテナンス必要性の判断
  - *_INDEXES の ORPHANED_ENTRIES 列から確認可能

実行例

```
SQL> select table_name, index_name,partitioned,status,orphaned_entries
       from user_indexes;

TABLE_NAME INDEX_NAME          PARTITIONED STATUS   ORPHANED_ENTRIES
---------- ------------------- ----------- -------- ----------------
PARTTAB    PARTTAB_IND_GLOBAL NO          VALID    NO

SQL> ALTER TABLE parttab TRUNCATE PARTITION p01 UPDATE GLOBAL INDEXES;

SQL> select table_name, index_name,partitioned,status,orphaned_entries
       from user_indexes;

TABLE_NAME INDEX_NAME          PARTITIONED STATUS   ORPHANED_ENTRIES
---------- ------------------- ----------- -------- ----------------
PARTTAB    PARTTAB_IND_GLOBAL NO          VALID    YES
```

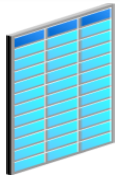
メンテナンス不要

メンテナンスが必要

インターバル・リファレンス・パーティショニング

リファレンス・パーティションの親表をインターバル・パーティションで作成可

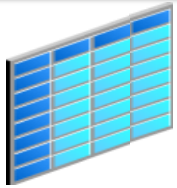
◆ 親表
Orders



表名	列名
Ordes	order_id
	customer_id
	order_date

パーティション・キー

◆ 子表
OrderItems



表名	列名
OrderItems	order_id
	line_item_id
	product_id
	quantity

パーティション・キー不要

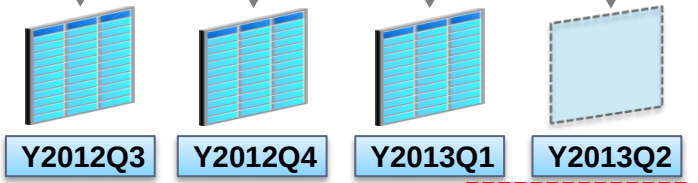
参照整合性制約
orderitems_fk

◆ リファレンス・パーティション

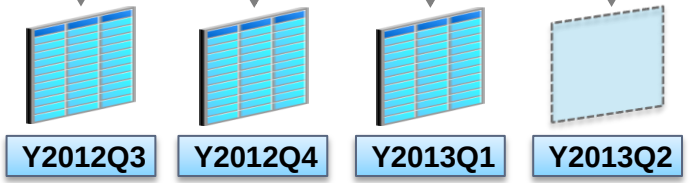
12c

- インターバル・パーティションを親表とした構成をサポート

自動的に追加



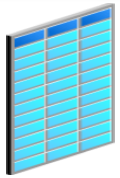
自動的に追加



リファレンス・パーティションの Cascade 処理

1. CASCADE での TRUNCATE PARTITION

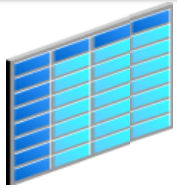
◆ 親表
Orders



表名	列名
Ordes	order_id
	customer_id
	order_date

ON DELETE CASCADE ＋ 参照整合性制約 orderitems_fk

◆ 子表
OrderItems



表名	列名
OrderItems	order_id
	line_item_id
	product_id
	quantity

◆ リファレンス・パーティション

12c
- 親表への TRUNCATE 時に
  子表への TRUNCATE を自動実行

TRUNCATE

CASCADE

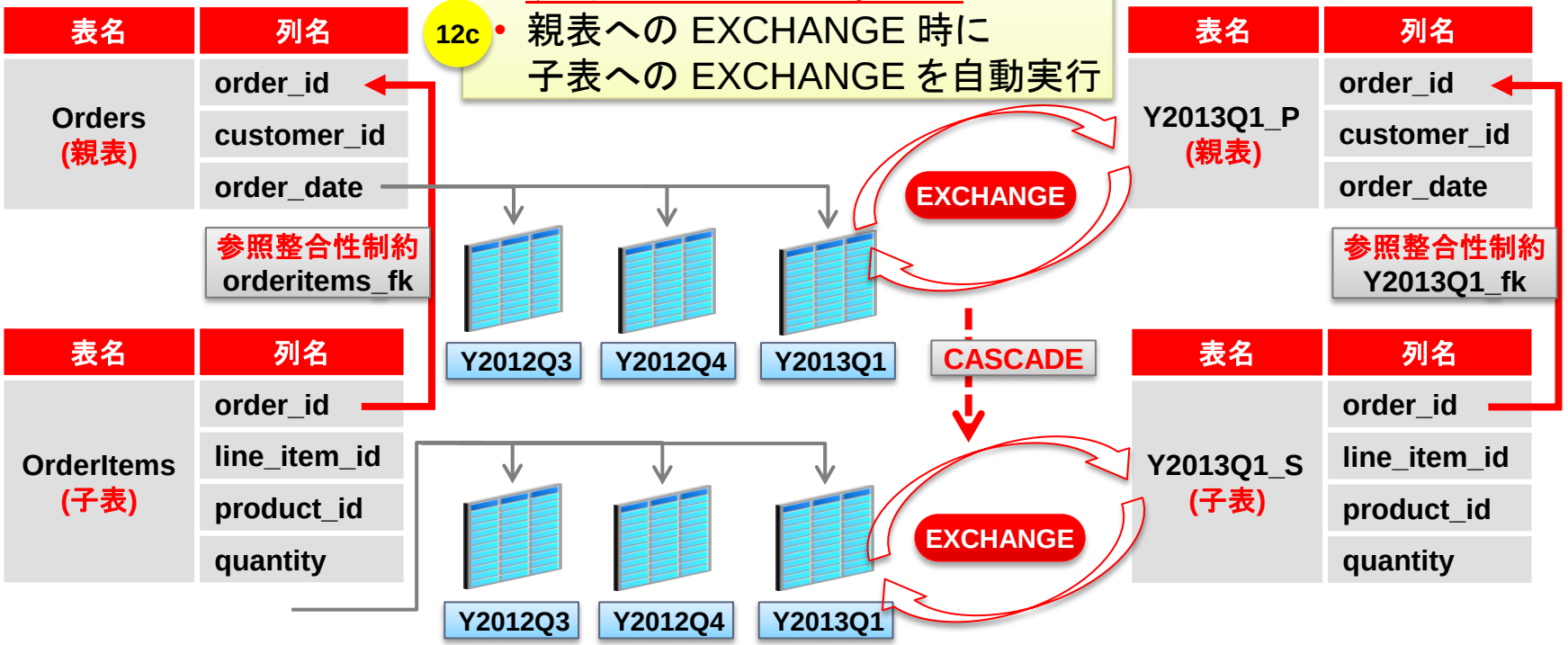
TRUNCATE

リファレンス・パーティションの Cascade 処理

2. CASCADE での EXCHANGE PARTITION

◆ リファレンス・パーティション

12c • 親表への EXCHANGE 時に
子表への EXCHANGE を自動実行



リファレンス・パーティションの Cascade 処理

CASCADE による TRUNCATE / EXCHANGE PARTITION

- CASCADE オプションの追加
  - リファレンス・パーティション / リファレンス・インターバル・パーティションに有効
  - 親表に対する処理を子表にも自動実行
  - 対象となる処理
    - TRUNCATE PARTITION / SUBPARTITION
      - 親表での TRUNCATE 時に子表の対象データも同時に TRUNCATE
      - 子表の参照整合性制約に ON DELETE CASCADE オプションの指定が必要
    - EXCHANGE PARTITION / SUBPARTITION
      - 親表での EXCHANGE 時に子表の対象データも同時に EXCHANGE
      - EXCHANGE 対象の表も親表・子表で構成しておく必要がある

オンラインでのパーティションの移動

ALTER TABLE MOVE PARTITION/SUBPARTITION の ONLINE 化

- DML 実行中の ALTER TABLE .. MOVE PARTITION / SUBPARTITION の実行が可能に

実行例

```
SQL> ALTER TABLE parttab MOVE PARTITION p01 TABLESPACE newtbs ONLINE;
SQL> ALTER TABLE parttab MOVE SUBPARTITION p01_s01 TABLESPACE newtbs ONLINE;
```

- 制限：ONLINE で実行できないケース
  - 以下の表に対する処理は実行不可
    - SYS スキーマの表
    - 索引構成表
    - オブジェクト型の列を持つヒープ構成表
    - ビットマップ索引 / ビットマップ結合索引 / ドメイン索引 が作成されている表
  - データベースでサプリメンタル・ロギングが設定されている場合

XMLIndex の対応拡張

XMLIndex 作成可能なパーティショニング・メソッドの追加

- XMLType 表 / XMLType 列を含む表のパーティション化
  - ハッシュ / インターバル・パーティションにも XMLIndex の作成が可能に

パーティショニング・メソッド	11g	12c
レンジ	●	●
リスト	●	●
ハッシュ	×	●
インターバル	×	●

◆ XMLType 表 / XMLType 列を含む表のパーティショニング メソッドによる XMLIndex 作成の可否

その他の新機能

その他の新機能

- OLTP表圧縮の機能拡張
- ネットワーク圧縮

OLTP表圧縮の機能拡張

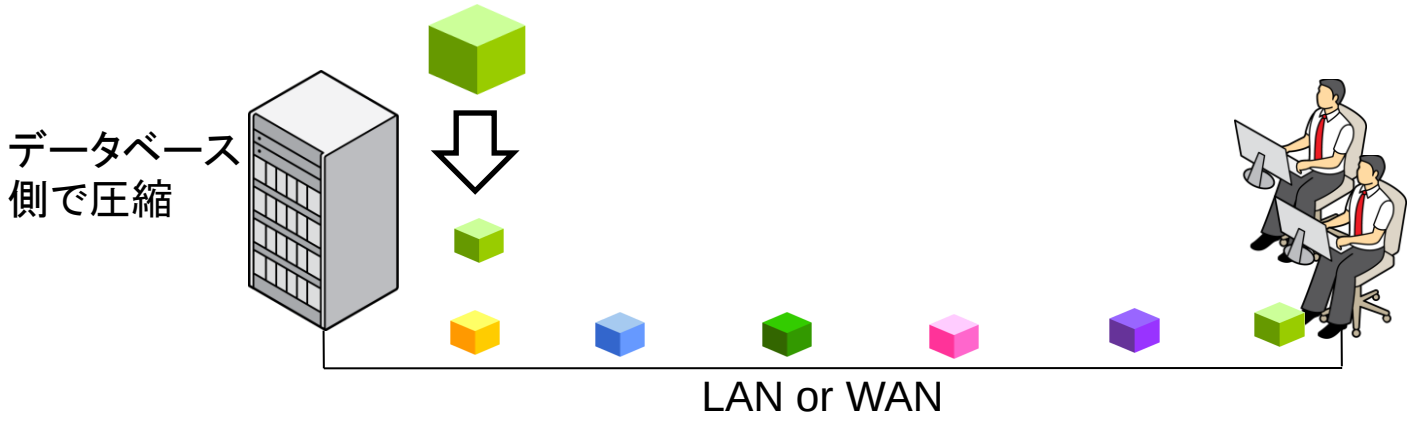
1000列までの表に対するサポート

- Oracle Database 12cでは、1000列までの表に対して、Advanced Row 圧縮が可能
  - Oracle Database 11g R2までは、254列までの表が対象
    - 255以上の列が含まれる表は圧縮不可
    - OLTP表圧縮を利用したくても、この上限値により適用できないケースも

より広範な表の圧縮が可能となり、ストレージコストの削減に寄与

ネットワーク圧縮

- Oracle Netのレイヤーでクライアント間での通信データを圧縮する
  - ネットワークボトルネックが発生した場合の帯域向上
  - 通信データを圧縮することにより、ネットワーク帯域の有効活用が可能



ネットワーク圧縮

設定方法

- ネットワーク圧縮を有効化するには、サーバーおよびクライアントのsqlnet.oraファイルにいくつかの項目を手動で追加する
  - 設定を行うパラメータ
    - SQLNET.COMPRESSION=[ON/OFF/AUTO]
      - ネットワーク圧縮の有効化を設定する。デフォルトはOFF
    - SQLNET.COMPRESSION_LEVELS=[HIGH/LOW]
      - 圧縮のレベルを設定する。デフォルトはLOW
    - SQLNET.COMPRESSION_THRESHOLD=1024(bytes)
      - ネットワーク圧縮が行われるデータサイズを設定する。このサイズ以下のデータは圧縮が行われずに送信される。デフォルトは1024(bytes)

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®