
ORACLE®

ORACLE

Oracle Database 12*c* Release 1

Information Lifecycle Management

日本オラクル株式会社
磯部 光洋

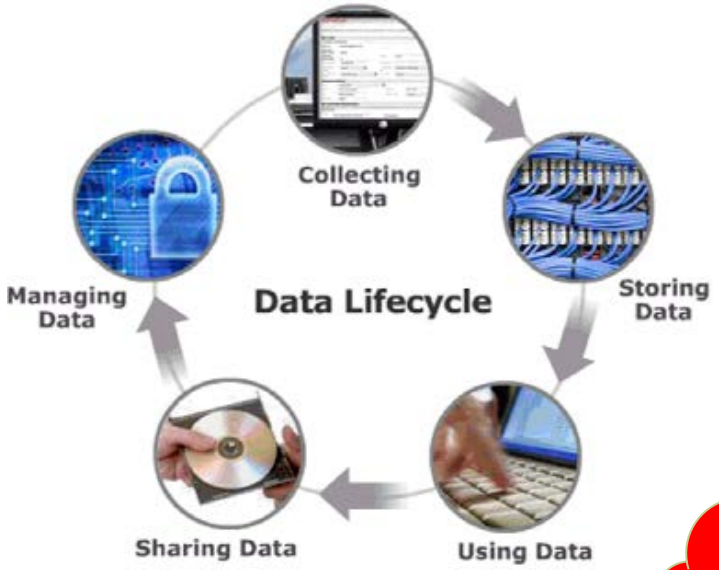
以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

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

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

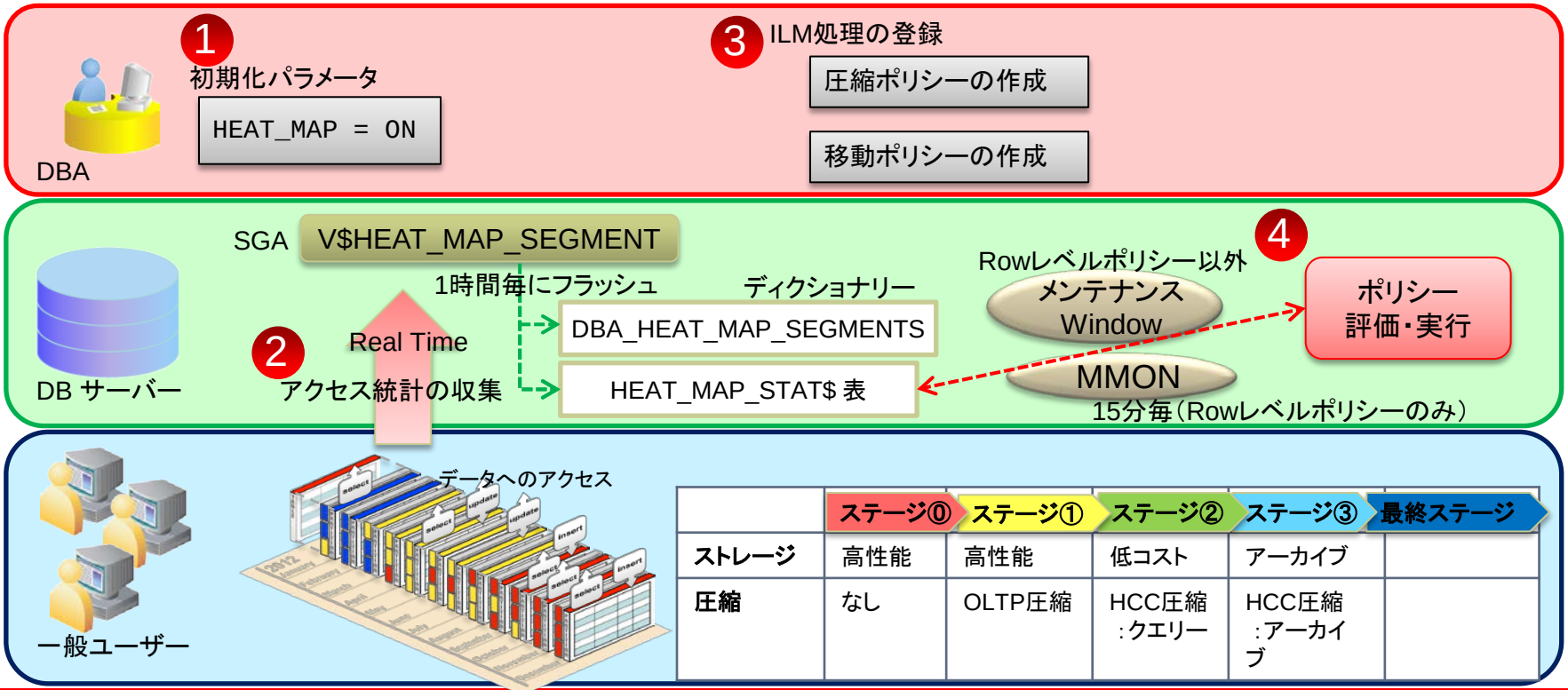
Automatic Data Optimization とは

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

1

DBA

1 初期化パラメータ

```
HEAT_MAP = ON
```

3 ILM処理の登録

- 圧縮ポリシーの作成
- 移動ポリシーの作成

DB サーバー

SGA V$HEAT_MAP_SEGMENT

1時間毎にフラッシュ

ディクショナリー

- DBA_HEAT_MAP_SEGMENTS
- HEAT_MAP_STAT$ 表

2 Real Time アクセス統計の収集

4

Rowレベルポリシー以外

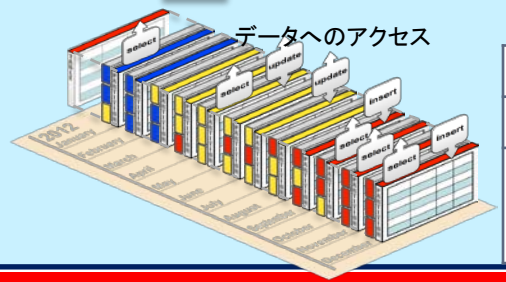
メンテナンス Window

MMON

15分毎（Rowレベルポリシーのみ）

ポリシー評価・実行

一般ユーザー



	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

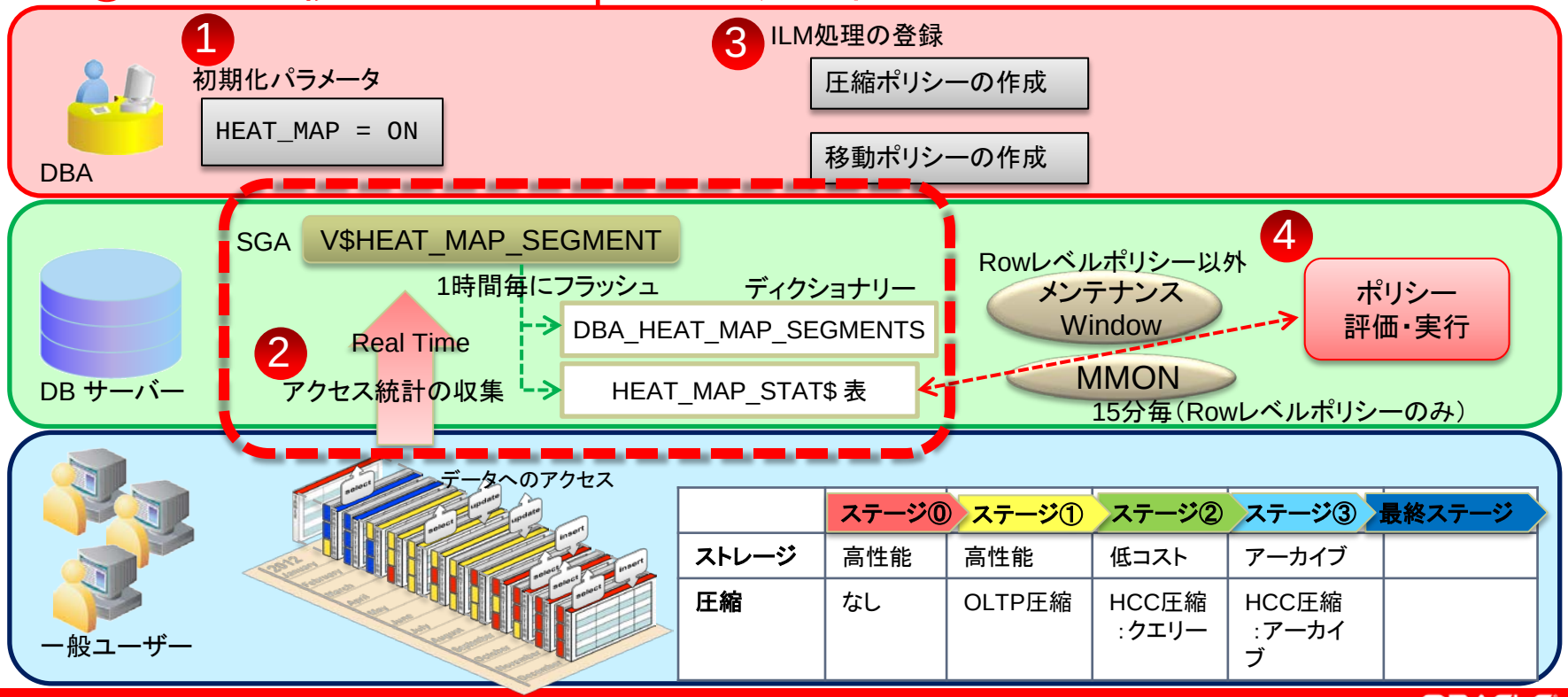
```
SQL> ALTER SYSTEM SET heat_map = ON;
```

- 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

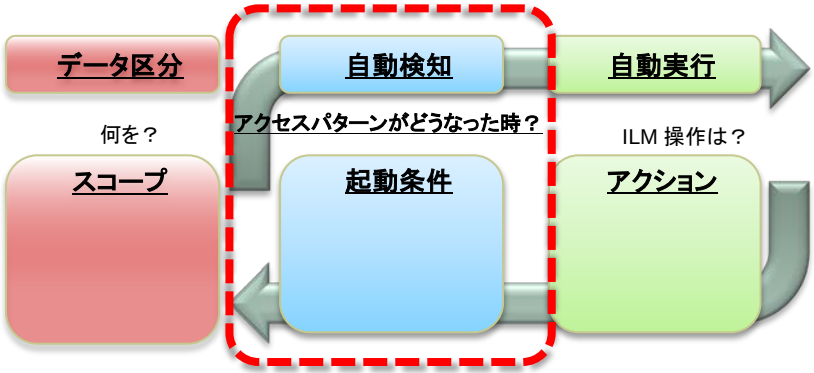


	ステージ⓪	ステージ①	ステージ②	ステージ③	最終ステージ
ストレージ	高性能	高性能	低コスト	アーカイブ	
圧縮	なし	OLTP圧縮	HCC圧縮 :クエリー	HCC圧縮 :アーカイブ	

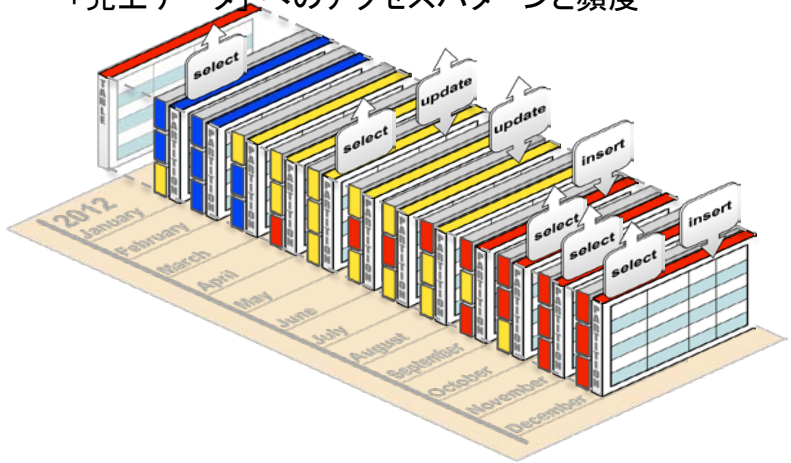
Automatic Data Optimization とは

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・レベルの最後のアクセス日時

```
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
```

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）

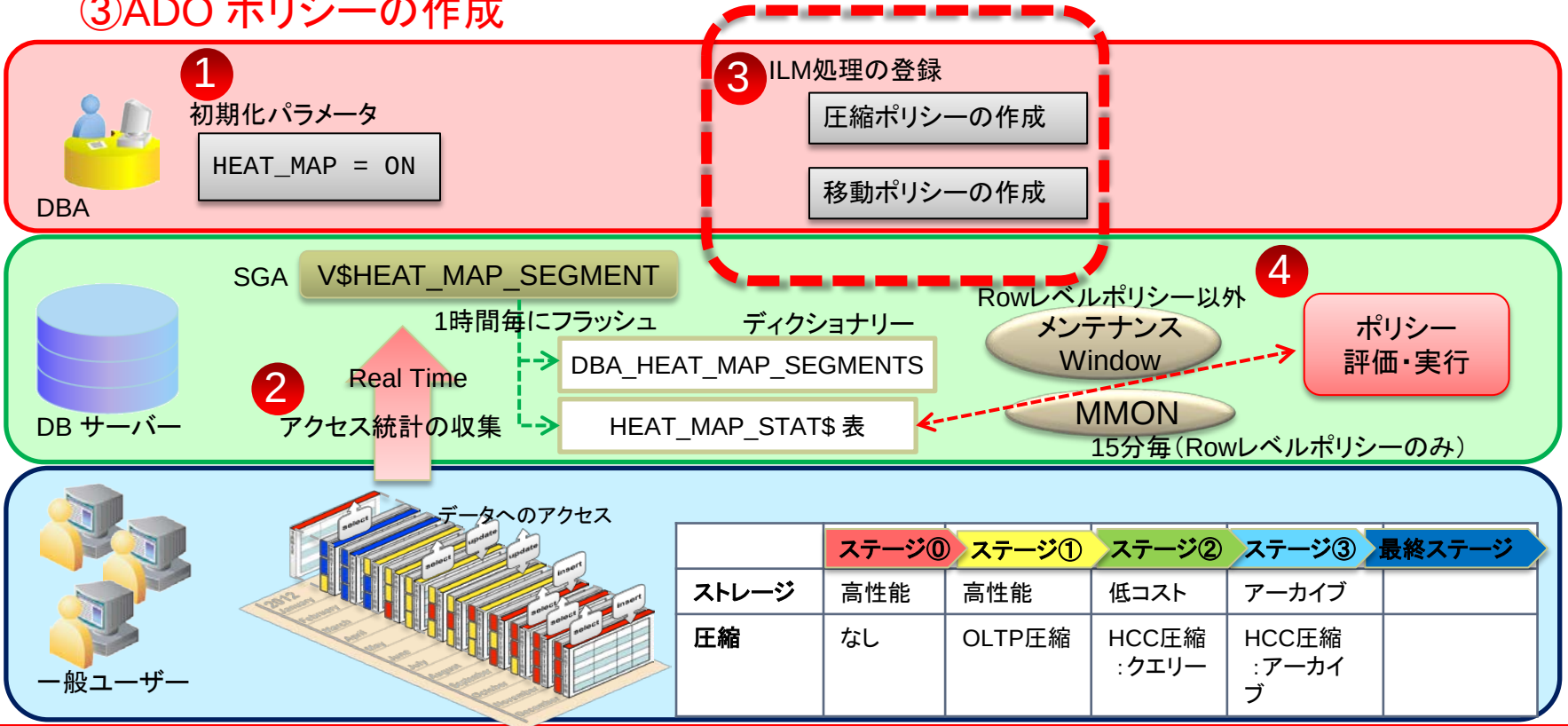
```
SQL> SELECT OWNER, OBJECT_NAME, SEGMENT_COUNT
  2          , OBJECT_SIZE, MAX_WRITETIME, MAX_READTIME
  3          , MAX_FTSTIME, MAX_LOOKUPTIME
  4  FROM DBA_HEATMAP_TOP_OBJECTS;

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

3

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）**

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

圧縮タイプ

実行レベル

起動タイミング

アクセスパターン

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

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

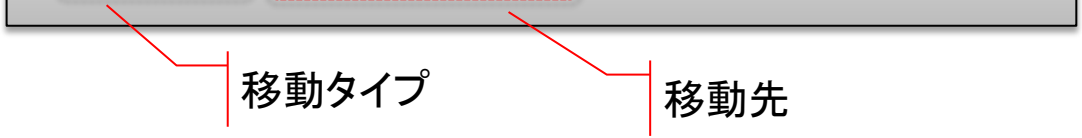
- 指定された「アクセスパターン」となってからの時間
  - n DAY[s]
  - n MONTH[s]
  - n YEAR[s]

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
• 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;
```

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

ユーザー定義のPL/SQL関数を利用

PL/SQLを使って起動条件をカスタマイズ

Automatic Data Optimization とは

4

④ポリシーの評価・実行

1 初期化パラメータ

HEAT_MAP = ON

DBA

3 ILM処理の登録

圧縮ポリシーの作成

移動ポリシーの作成

SGA V$HEAT_MAP_SEGMENT

1時間毎にフラッシュ

ディクショナリー

DBA_HEAT_MAP_SEGMENTS

HEAT_MAP_STAT$ 表

2 Real Time アクセス統計の収集

DB サーバー

Rowレベルポリシー以外

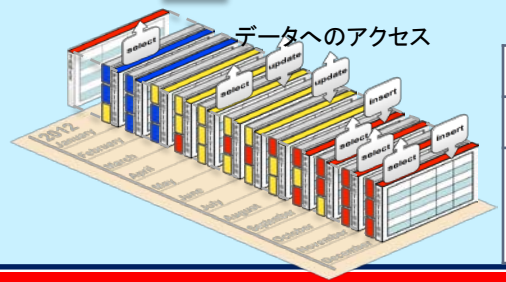
メンテナンス Window

MMON

15分毎（Rowレベルポリシーのみ）

4 ポリシー評価・実行

一般ユーザー



	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

④ポリシーの評価・実行

- **自動実行**
  - MMON（15分毎）　　：処理対象が Row のポリシーのみ
  - メンテナンスWindow　　：処理対象が Row のポリシー以外
- **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)

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

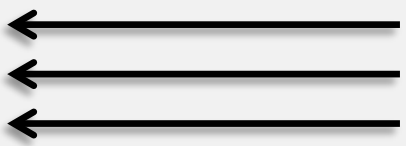
- アーカイブ属性データにアクセスするには

```
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
```



アクティブ
非アクティブ
非アクティブ

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

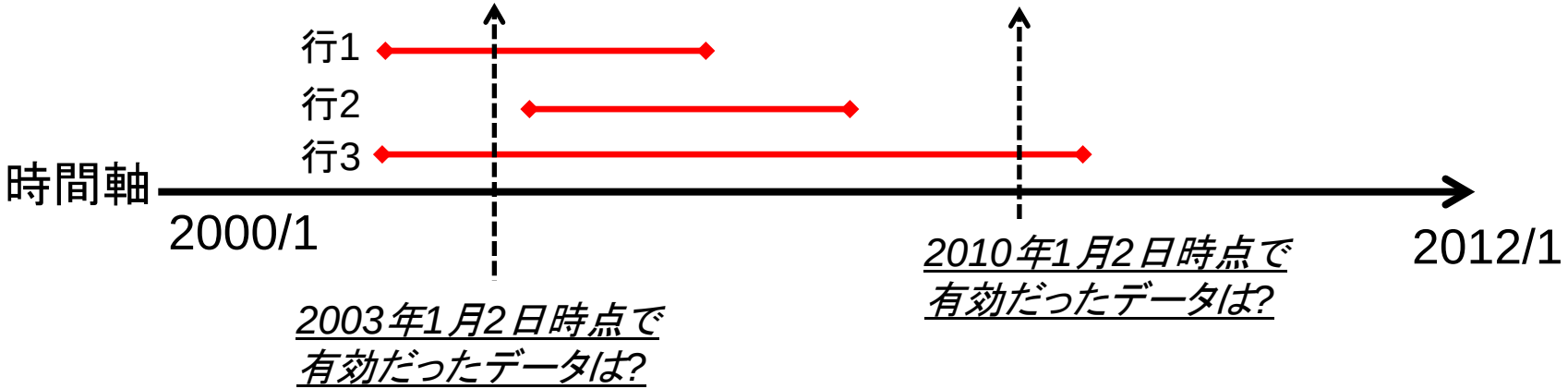
- 有効期間を持つデータ（クレジットカード、免許情報など）のハンドリングに有効
- 有効期間にあるデータだけが検索対象
  - 任意の時点で有効なデータ
  - 任意の期間で有効なデータ

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**



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 timestam p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

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');
```

ORACLE

Oracle Database 12c Release 1 CoreTech Seminar

Migration

日本オラクル株式会社
磯部 光洋

Program Agenda

- Migration 概要
- 新機能詳細
  - SQL Translation Framework
  - Implicit Statement Results
  - Enhanced SQL to PL/SQL Bind Handling
  - Identity Columns
  - Extended Data Types

Migration 概要

Migration 概要

- **他社データベースからOracleデータベースへの移行**をより容易にする機能
- **アプリケーションの移行**を容易にする機能追加／機能拡張がメイン
  - SQL Translation Framework
  - Implicit Statement Results
  - Enhanced SQL to PL/SQL Bind Handling
  - Identity Columns
  - Extended Data Types

新機能詳細

SQL Translation Framework

機能概要

- **他社データベース用に作成された SQL を、そのまま Oracle データベースに向けて実行可能**
- Oracleは他社構文のSQL を受取り、Oracleが解釈可能なSQLに翻訳＆実行
- 下記DBのSQL構文に対応
  - SQL Server
  - Sybase Adaptive Server(ASE)

実行例

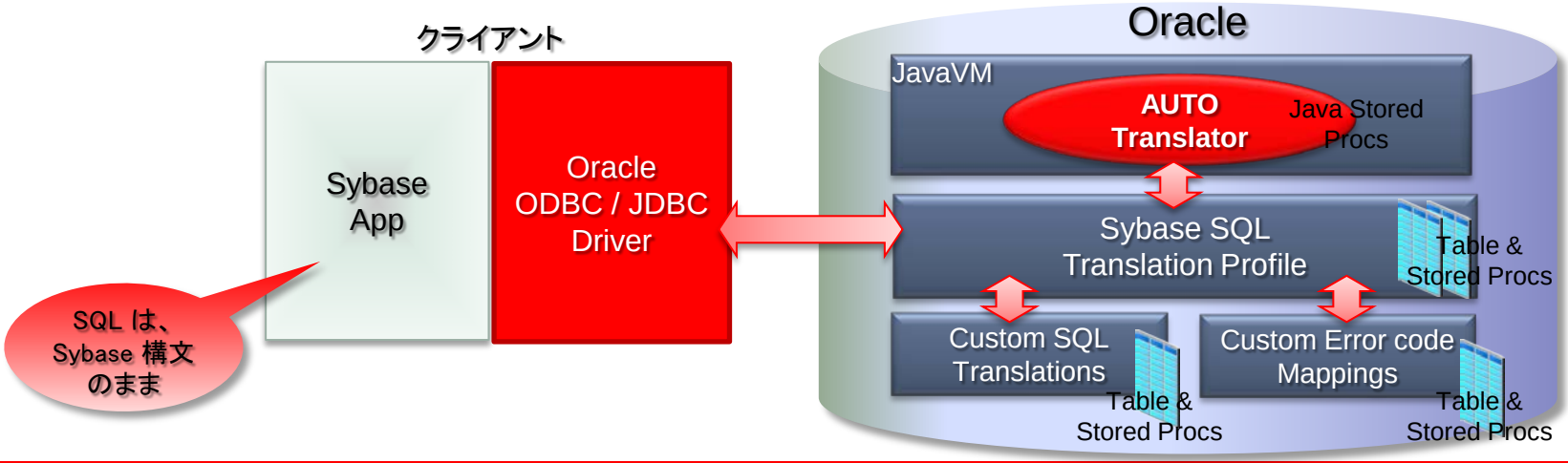
```
SQL> select top 3 * from emp;

EMPNO ENAME      JOB         MGR HIREDATE             SAL  COMM DEPTNO
----- ---------- --------- ----- ----------------- ----- ----- ------
 7369 SMITH      CLERK      7902 17-DEC-80           800           20
 7499 ALLEN      SALESMAN   7698 20-FEB-81          1600   300     30
 7521 WARD       SALESMAN   7698 22-FEB-81          1250   500     30
```

SQL Translation Framework

ユースケース

- Oracle は受信した SQL の翻訳結果が Profile に未登録の場合、AUTO Translator で Oracle が解釈可能なSQLに翻訳（Hard Parse の場合のみ）
- 翻訳された SQL を実行し、結果をクライアントに返す
- 翻訳された SQL は、翻訳前の SQL と共に Profile に登録（デフォルト動作）



SQL Translation Framework

利用手順

- サービスとして登録＆起動（DBMS_SERVICESパッケージでも作成可能）

```
$ srvctl add service –d orcl -s translator –sql_translation_profile
 ¥ miguser.sybase_profile
$ srvctl start service –db orcl –service translator
```

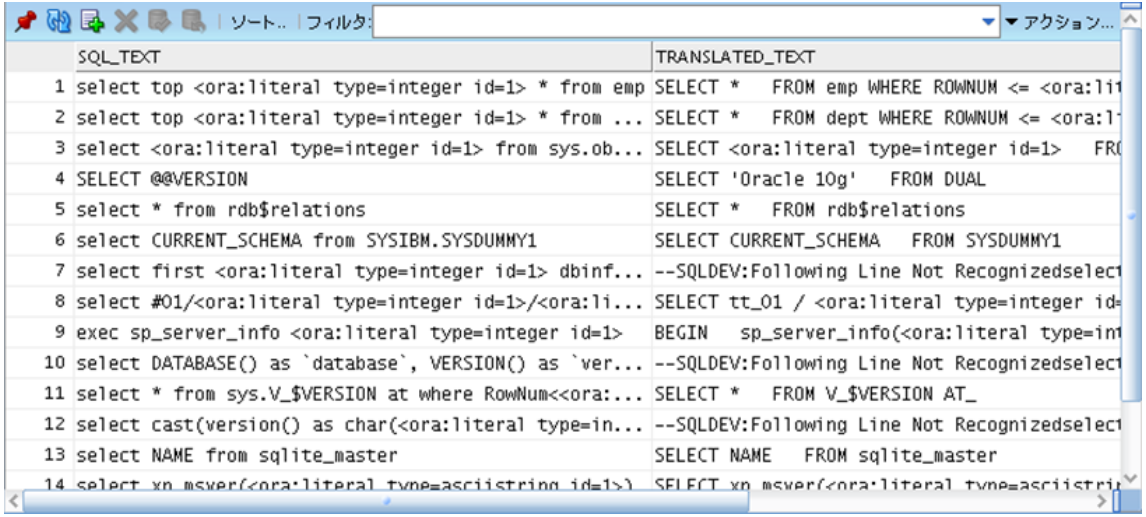
- サービスを利用した接続例（JDBCの例）

```
  serverURL = “jdbc:oracle:thin:@192.168.63.100:1521/translator”;
  username = “scott”;
  password = “tiger”;
. . .
try {
  conn = DriverManager.getConnection(serverURL, username, password);
. . .
}
. . .
```

SQL Translation Framework

Profile 機能

- AUTO Translator による翻訳結果が登録されているデータベース表
- 以前に登録されたSQLの場合、Hard Parseの際に**Profile にある翻訳結果を利用し**、**AUTO Translator のオーバーヘッドを回避**
- Oracle Data PumpによるExport/Importが可能

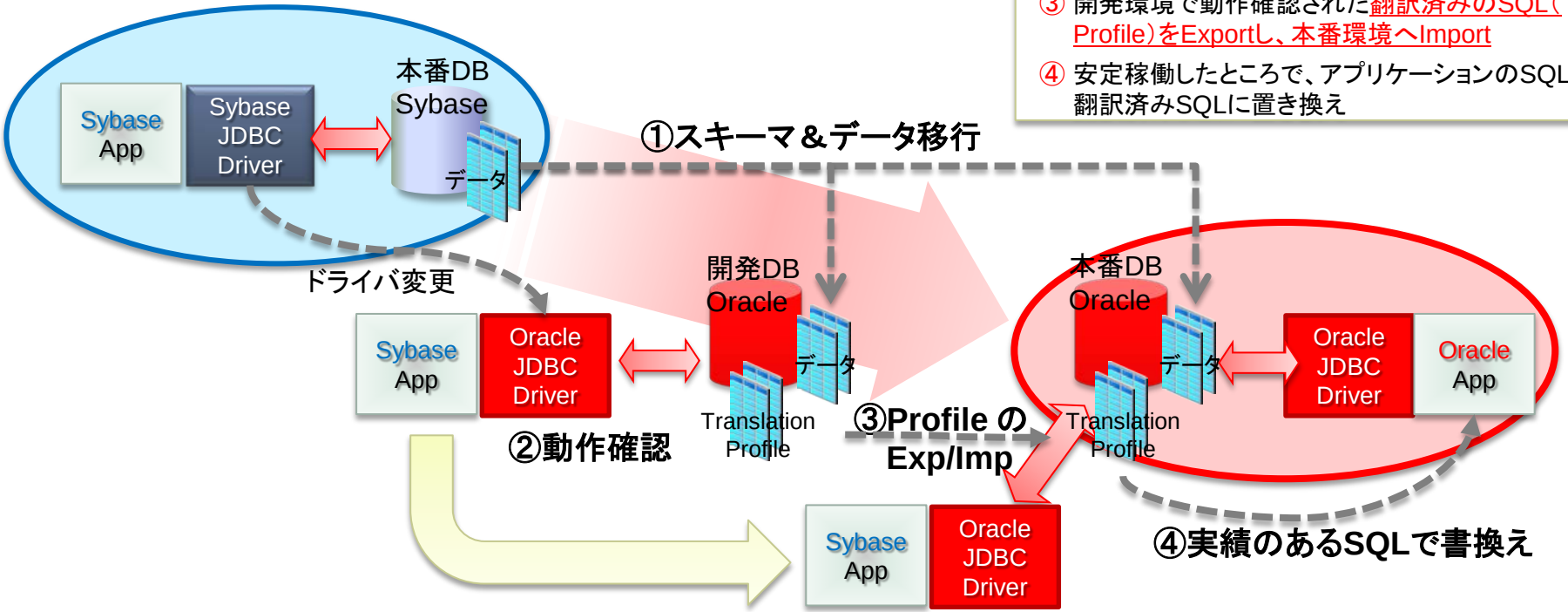


	SQL_TEXT	TRANSLATED_TEXT
1	select top <ora:literal type=integer id=1> * from emp	SELECT * FROM emp WHERE ROWNUM <= <ora:lit
2	select top <ora:literal type=integer id=1> * from ...	SELECT * FROM dept WHERE ROWNUM <= <ora:l
3	select <ora:literal type=integer id=1> from sys.ob...	SELECT <ora:literal type=integer id=1> FR
4	SELECT @@VERSION	SELECT 'Oracle 10g' FROM DUAL
5	select * from rdb$relations	SELECT * FROM rdb$relations
6	select CURRENT_SCHEMA from SYSIBM.SYSDUMMY1	SELECT CURRENT_SCHEMA FROM SYSDUMMY1
7	select first <ora:literal type=integer id=1> dbinf...	--SQLDEV:Following Line Not Recognizedselect
8	select #01/<ora:literal type=integer id=1>/<ora:li...	SELECT tt_01 / <ora:literal type=integer id=
9	exec sp_server_info <ora:literal type=integer id=1>	BEGIN sp_server_info(<ora:literal type=int
10	select DATABASE() as `database`, VERSION() as `ver...	--SQLDEV:Following Line Not Recognizedselect
11	select * from sys.V_$VERSION at where RowNum<<ora:...	SELECT * FROM V_$VERSION AT_
12	select cast(version() as char(<ora:literal type=in...	--SQLDEV:Following Line Not Recognizedselect
13	select NAME from sqlite_master	SELECT NAME FROM sqlite_master
14	select xp_msver(<ora:literal type=asciistring id=1>)	SELECT xp_msver(<ora:literal type=asciistri

SQL Translation Framework

ユースケース

① 開発環境へ、スキーマ定義およびデータを移行。
② 開発環境で、既存アプリケーションの一連の動作を確認
③ 開発環境で動作確認された翻訳済みのSQL（Profile）をExportし、本番環境へImport
④ 安定稼働したところで、アプリケーションのSQLを翻訳済みSQLに置き換え



Implicit Statement Results

機能概要

- 11g まで
  - Oracle PL/SQLでは、Select文による出力結果は、**INTO句やカーソル**を使用するなどして**明示的な出力結果の受け渡し**が必要
  - Sybase ASE や MS SQLServer、IBM DB2、MySQLは、Oracleの様に明示的な出力結果の受け渡しを記述しない方法が可能
    - **この場合、Select文のコール元に結果を返す**
- 12c では
  - Implicit Result Sets機能により、MySQL や IBM DB2、MS SQLServer などと同様に、Select文の結果を暗黙的にコール元に返すことが可能

Implicit Statement Results

機能概要

- Implicit Result Sets（12c）

プロシージャ定義

```
create or replace  procedure p as
  c1 sys_refcursor;
  c2 sys_refcursor;
begin
  open c1 for select deptno from dept;
  dbms_sql.return_result(c1);

  open c2 for select empno from emp;
  dbms_sql.return_result(c2);
end;
/
```

※dbms_sql.return_result() は、引数の文カーソルの実行結果を返すプロシジャー

コマンド・ツール からの実行例

```
SQL> exec p

PL/SQLプロシージャが正常に完了しました。

ResultSet #1

    DEPTNO
----------
        10
        20
        30

ResultSet #2

     EMPNO
----------
      7900
      7902
      7934
```

Enhanced SQL to PL/SQL Bind Handling

機能概要：1

- 11gまでは引数や戻り値のデータ型にSQLデータ型以外を使用不可
- 12c では、以下の型のパラメータを持つPL/SQL関数を起動可能
  - Boolean
  - パッケージ仕様部で宣言されたレコード
  - パッケージ仕様部で宣言されたコレクション

```
DECLARE
  fruits pkg.names;
  dyn_stmt VARCHAR2(3000);
BEGIN
  fruits := pkg.names('apple','banana','cherry');
  dyn_stmt := 'BEGIN print_names(:x); END;';
  EXECUTE IMMEDIATE dyn_stmt USING fruits;
END;
```

```
CREATE OR REPLACE PACKAGE pkg
  AUTHID CURRENT_USER AS
    TYPE names IS TABLE OF VARCHAR2(10);
    PROCEDURE print_names (x names);
END pkg;
/
CREATE OR REPLACE PACKAGE BODY pkg AS
  PROCEDURE print_names (x names) IS
  BEGIN
    FOR i IN x.FIRST .. x.LAST LOOP
      DBMS_OUTPUT.PUT_LINE(x(i));
    END LOOP;
  END;
END pkg;
/
```

Enhanced SQL to PL/SQL Bind Handling

機能概要：2

- 11g の Table 演算子はPL/SQL 表には非対応
- 12c では、PL/SQL 表にも対応し、通常の表と同様に**PL/SQL 表にSQL でアクセス**が可能

```
CREATE OR REPLACE PACKAGE pkg AS
  TYPE rec IS RECORD(f1 NUMBER, f2 VARCHAR2(30));
  TYPE mytab IS TABLE OF rec INDEX BY pls_integer;
END;
 /

DECLARE
   v1 pkg.mytab;
   v2 pkg.rec;
   c1 sys_refcursor;
BEGIN
   open c1 FOR SELECT * FROM TABLE(v1);
   fetch c1 INTO v2;
END;
 /
```

Identity Column

機能概要

- **ANSI準拠のIDENTITY column を実装**
- 11gまでは、数値型の一意のIDをカラムとして定義する場合、事前に順序を作成し、カラムのデフォルト値としてその順序を指定
- Identity Column機能により、事前の順序作成は不要。

```
SQL> create table t1
  2  (c1 number
  3       GENERATED BY DEFAULT ON NULL AS IDENTITY,
  4   c2 varchar2(10));

Table created.

SQL> insert into t1(c2) values('abc');

1 row created.

SQL> insert into t1(c1, c2) values(null, 'xyz');

1 row created.
```

```
SQL> select c1, c2 from t1;

        C1 C2
---------- ----------
         1 abc
         2 xyz
```

Extended Data Types

機能概要

- Varchar2、NVarchar2、RAW型において、最大長が32,767バイトまで定義可能
- 他社DBからの移行を完全カバー
- 利用するには下記の2つの初期化パラメータを設定する必要があります。

```
compatible = ’ 12.0.0.0.0’
max_string_size =  ‘EXTENDED’
```

  - ただし、EXTENDEDからSTANDARD（従来通りの最大データ長）の変更不可

- 参考：他社DBの同データ型最大長
  - M社：　　8,000バイト
  - I社：　　32,672バイト

Extended Data Types

設定手順

- DBを停止
- DBをUPGRADE モードで再起動
- 下記の2つの初期化パラメータを設定

```
compatible =  ‘12.0.0.0’
max_string_size =  ‘EXTENDED’
```

- AS SYSDBA で接続し、SQLスクリプト（$ORACLE_HOME/rdbms/admin/utl32k.sql）を実行
- DBをNORMAL モードで再起動

※参照：マニュアル「Oracle® Database Reference 12c Release 1 (12.1) 『MAX_STRING_SIZE』」

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®