ORACLE

Oracle Database 12*c* 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
  - Query Row Limits and Row Offsets
  - 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
  - Query Row Limits and Row Offsets
  - Extended Data Types

SQL Transration Framework

“SQL翻訳フレームワーク”

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)

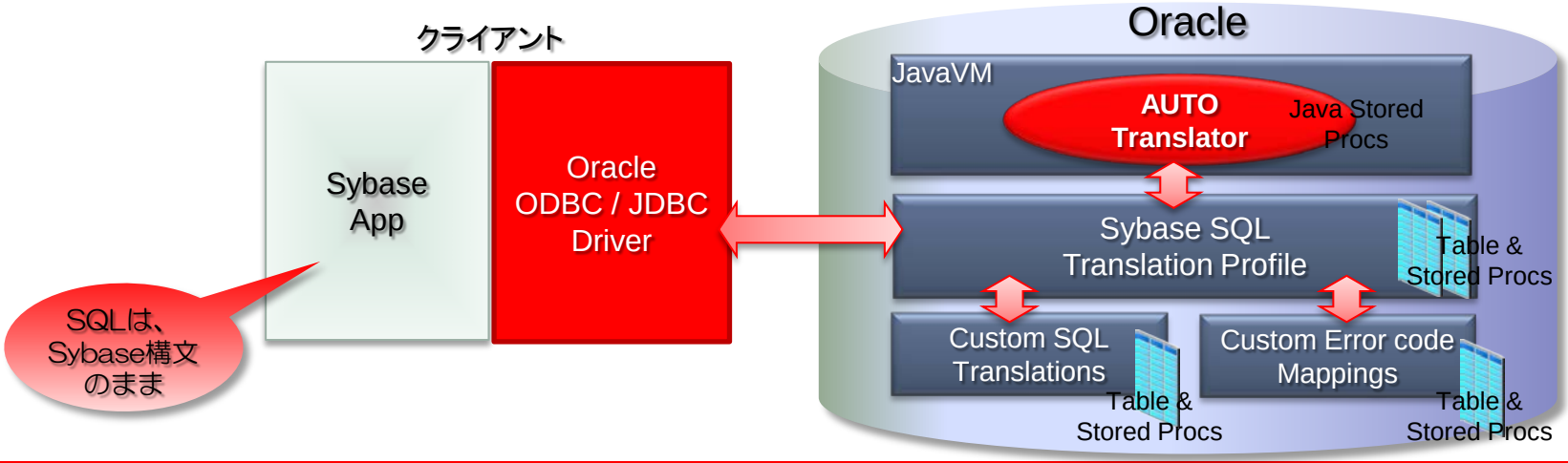
実行例

```
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

機能概要

- 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

利用イメージ

- サービスにプロファイル属性設定（srvctlコマンドの例）

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

- サービスにプロファイル属性設定（dbms_serviceパッケージの例）

```
declare
        params dbms_service.svc_parameter_array;
begin
        params('SQL_TRANSLATION_PROFILE') := 'miguser.sybase_profile';
        dbms_service.create_service(‘translator', ‘translator', params);
        dbms_service.start_service(‘translator');
end;
/
```

SQL Translation Framework

利用イメージ

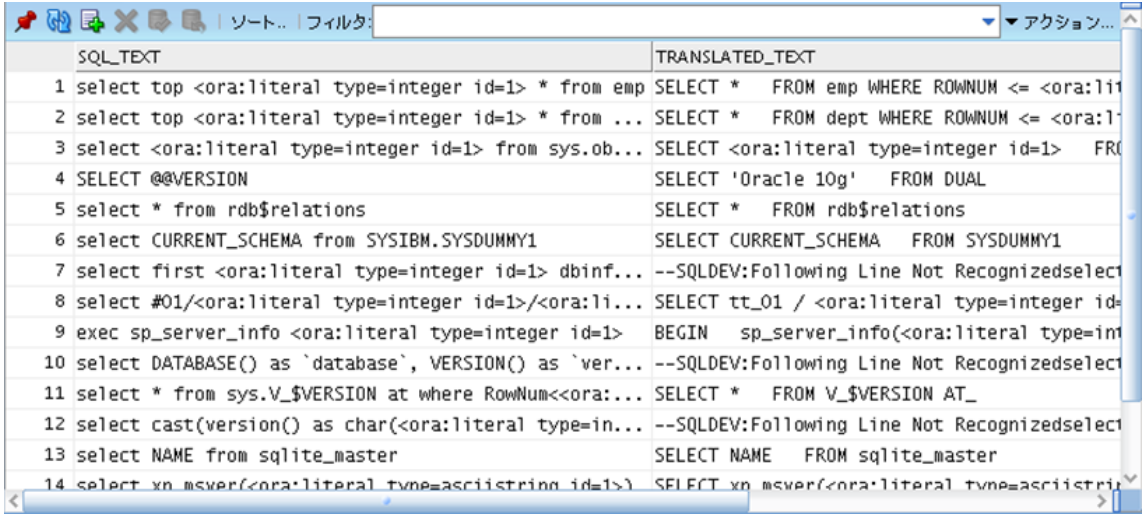
- サービスを利用した接続例（SQL*Plusの例）

```
SQL> connect miguser/miguser@sybase_service

接続されました。

SQL> alter session set events = '10601 trace name context forever, level 32';

セッションが変更されました。

SQL> select top 3 * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7369 SMITH      CLERK            7902 80-12-17        800                   20
      7499 ALLEN      SALESMAN         7698 81-02-20       1600        300         30
      7521 WARD       SALESMAN         7698 81-02-22       1250        500         30
```

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

Profile 機能

- SQL翻訳の手動登録も可能

```
exec dbms_sql_translator.register_sql_translation(
  ‘sybase_profile',
  'select row of (c1, c2) from sample_tab',
  'select c1, c2 from sample_tab‘
);
```

SQL Translation Framework

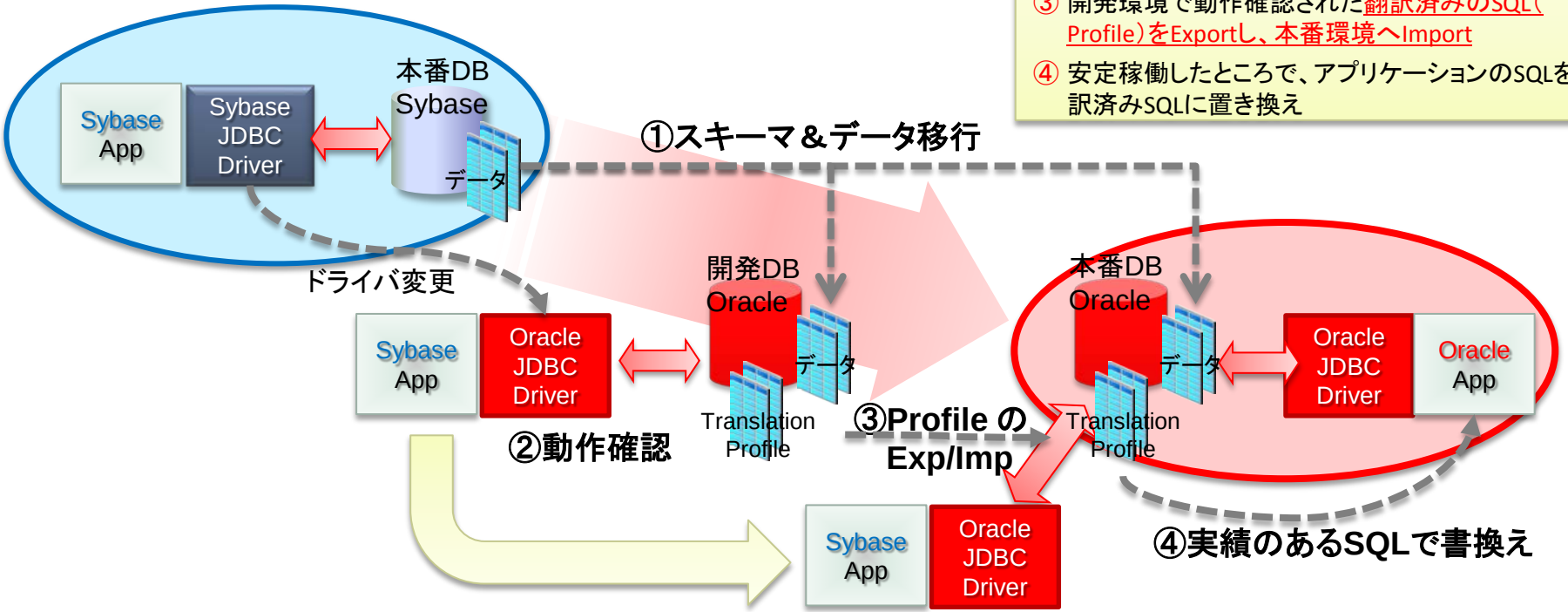
ユースケース

- SQL Translation Framework（以下STF）は、**SQLのレスポンスに影響**が出ることを考慮した利用検討が必要
- STF を使用した環境でアプリケーションの一連の動作確認ができれば、その結果として**実績のある翻訳済みSQL**を取得可能
- （例）STFを活用した完全移行までのステップ
  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

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

“暗黙的な文の結果”

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

他社DB（MySQL）の場合

- データの受け渡しで、明示的なパラメータが不要

```
create procedure p()
begin
  SELECT deptno FROM dept;
  SELECT empno FROM emp;
end;
/
```

```
mysql> call p;

  +--------+
  | deptno |
  +--------+
  |      10|
  |      20|
  |      30|
  +--------+
  3 rows in set (0.00 sec)

  +--------+
  | empno  |
  +--------+
  |    7900|
  |    7902|
  |    7934|
  +--------+
  3 rows in set (0.00 sec)
```

Implicit Statement Results

Oracle11g の場合

- データの受け渡しには、必ずパラメータが必要

```
create or replace  procedure p
(
  c1 out sys_refcursor,
  c2 out sys_refcursor
) as
begin
  open c1 for select deptno from dept;
  open c2 for select empno from emp;
end;
/
```

```
SQL> variable c1 refcursor
SQL> variable c2 refcursor

SQL> exec p(:c1, :c2)

SQL> print c1

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

SQL> print c2

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

Implicit Statement Results

DB12c の場合

```
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

```
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
```

Implicit Statement Results

DB12c の場合

- 例）JDBC アプリケーションからの実行（12c）

```
create or replace  procedure foo 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;
/
```

```
String sql = "begin foo; end;";
 ...
Connection con = DriverManager.getConnection(jdbcURL, user,
password);
  try {
            Statement stmt = con.createStatement ();
            stmt.executeQuery (sql);

            while (stmt.getMoreResults())
            {
                  ResultSet rs = stmt.getResultSet();
                  System.out.println("ResultSet");
                  while (rs.next())
                 {
                       System.out.println(rs.getInt(1));
                  }
             }
       }
```

Enhanced SQL to PL/SQL Bind Handling

“SQLからPL/SQLへのバインド処理の改善”

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

“アイデンティティ列”

Identity Column

機能概要

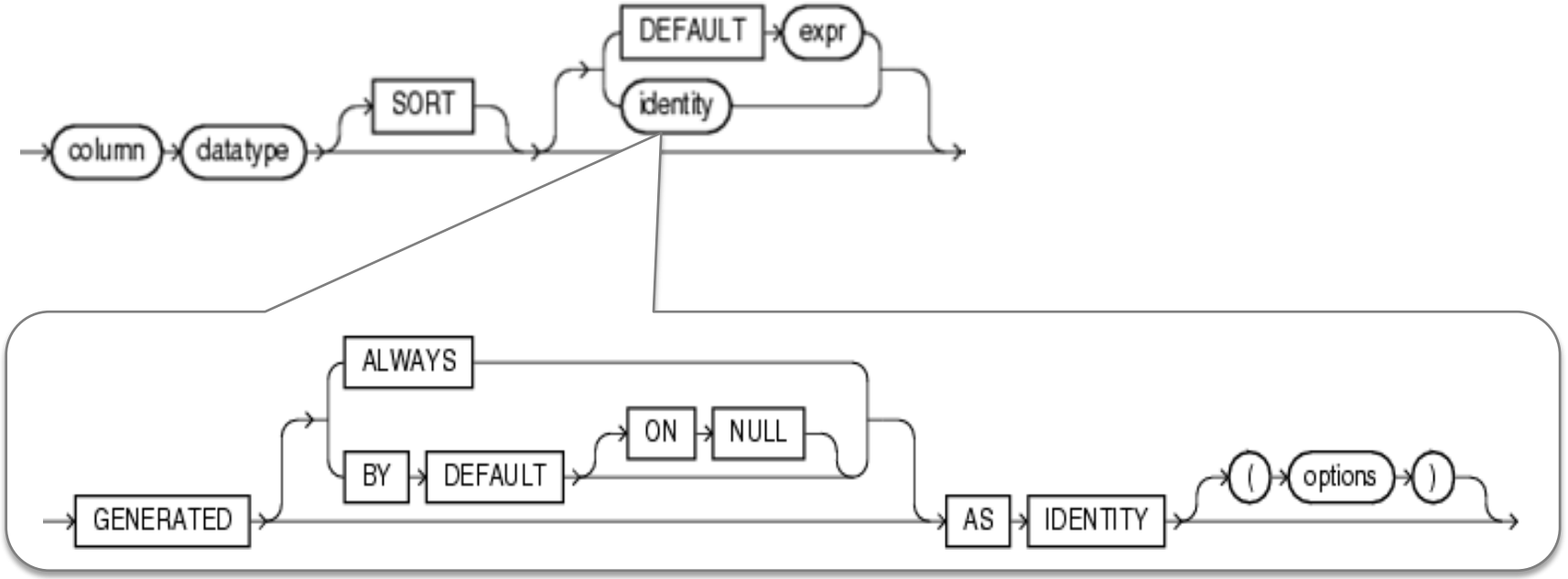
- **ANSI準拠のIDENTITY column を実装**
- 11gまでは、数値型の一意のIDをカラムとして定義する場合、事前に順序を作成し、カラムのデフォルト値としてその順序を指定
- Identity Column機能により、事前の順序作成は不要。

```
SQL> create table t1
  2  (c1 number GENERATED BY DEFAULT ON NULL AS IDENTITY,
  3   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
```

Identity Column

構文 (1/2)

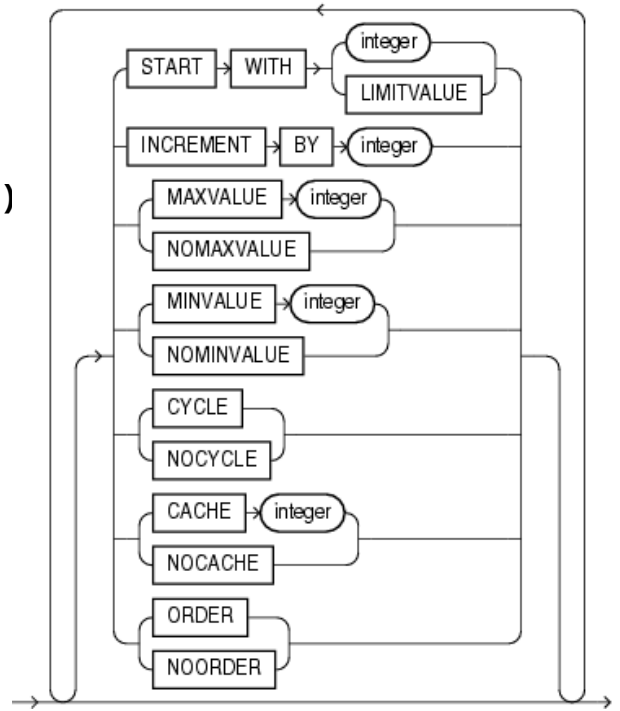


Identity Column

構文 (2/2)

options::=

[START WITH (<sequence generator start value> | LIMIT VALUE)
| INCREMENT BY <sequence generator increment>
| (MAXVALUE <sequence generator max value> | NO MAXVALUE)
| (MINVALUE <sequence generator min value> | NO MINVALUE)
| (CYCLE | NO CYCLE)
| (CACHE integer | NOCACHE)
| (ORDER | NOORDER)]+



Identity Column 設定例(1)

ALLWAYS の例

```
SQL> create table id_test(
  2  id number GENERATED ALWAYS AS IDENTITY,
  3  ename      varchar2(30)
  4  );
```

```
SQL> select * from id_test;

        ID ENAME
---------- ----------
         1 user2
```

```
SQL> insert into id_test(ID, ENAME) values(100, 'user1');
insert into id_test(ID, ENAME) values(100, 'user1')
                    *
行1でエラーが発生しました。:
ORA-32795: GENERATED ALWAYSで作成されたアイデンティティ列には挿入できません

SQL> insert into id_test(ENAME) values('user2');

1行が作成されました。
```

Identity Column 設定例(2)

BY DEFAULT の例

```
SQL> create table id_test2(
  2  ID number GENERATED BY DEFAULT AS IDENTITY,
  3  ENAME      varchar2(30)
  4  );
```

```
SQL> insert into id_test2(ID, ENAME) values(100, 'user1');

1行が作成されました。

SQL> insert into id_test2(ENAME) values('user2');

1行が作成されました。
```

```
SQL> select * from id_test2;

        ID ENAME
---------- ----------
       100 user1
         1 user2
```

Identity Column 制限事項

- 表に対して1つのみ定義可能
- 設定できるカラムのデータ型は数値型
- Identity Columnを設定したカラムにはDEFAULT句は指定不可
- Identity Columnを設定したカラムには暗黙的に NOT NULL 制約と NOT DEFERRABLE 制約が付加されるため、競合する制約は定義不可
- Identity Columnを暗号化する場合、暗号化アルゴリズムが推測されやすくなるため、強力な暗号化アルゴリズムの設定を推奨
- CTAS では Identity Column は継承されない

Query Row Limits and Row Offsets

“問合せの行制限と行オフセットのネイティブサポート”

Query Row Limits and Row Offsets

機能概要

- ANSI SQLの**Fetch First構文**が可能
  - 問合せの出力結果を制限

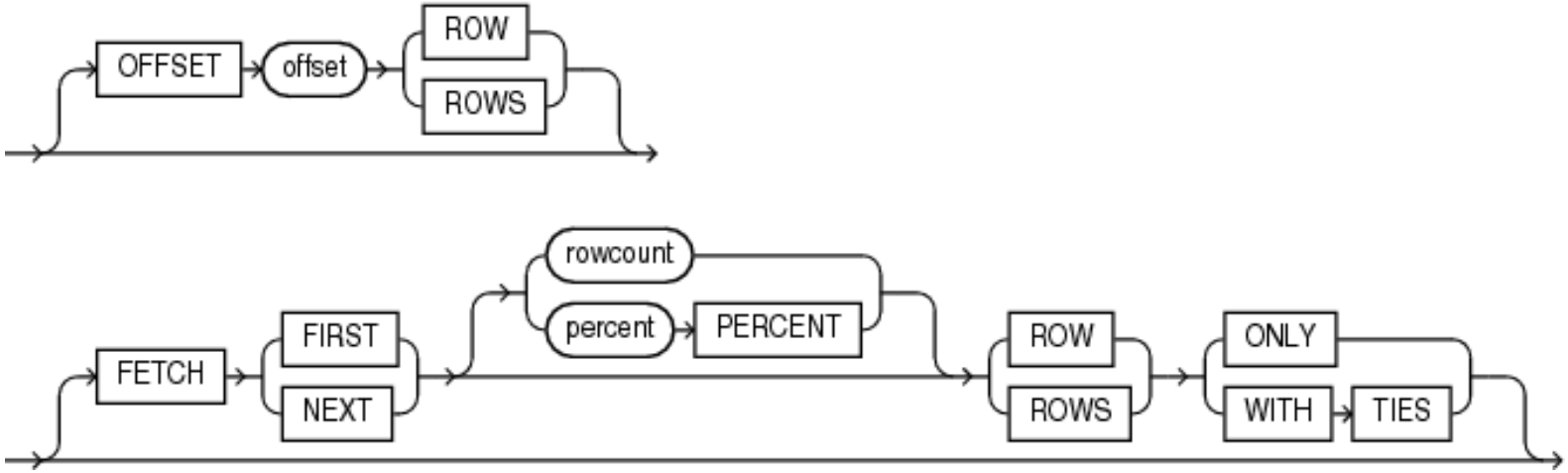
```
SQL> select empno, ename, deptno from emp
  2  order by sal, comm
  3  fetch first 5 rows only;

     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7369 SMITH              20
      7900 JAMES              30
      7876 ADAMS              20
      7521 WARD               30
      7654 MARTIN             30

SQL>
```

```
SQL> select empno, ename, deptno from emp
  2  order by sal, comm;

     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7369 SMITH              20
      7900 JAMES              30
      7876 ADAMS              20
      7521 WARD               30
      7654 MARTIN             30
      7934 MILLER             10
      7844 TURNER             30
      7499 ALLEN              30
      7782 CLARK              10
      7698 BLAKE              30
      7566 JONES              20
      7788 SCOTT              20
      7902 FORD               20
      7839 KING               10

14行が選択されました。
```

Query Row Limits and Row Offsets

機能概要

- ANSI SQLの**Fetch First構文**が可能
  - 問合せの出力結果を制限

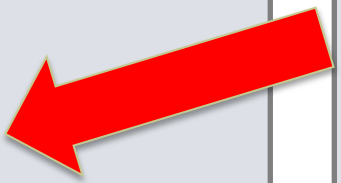


Query Row Limits and Row Offsets

機能概要

- ANSI SQLの**Fetch First構文**が可能
  - 問合せの出力結果を制限

```
SQL> select empno, ename, deptno from emp
  2  order by sal, comm
  3  offset 5 rows fetch first 5 rows
 only;

     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7934 MILLER             10
      7844 TURNER             30
      7499 ALLEN              30
      7782 CLARK              10
      7698 BLAKE              30
```



```
SQL> select empno, ename, deptno from emp
  2  order by sal, comm;

     EMPNO ENAME          DEPTNO
---------- ---------- ----------
      7369 SMITH              20
      7900 JAMES              30
      7876 ADAMS              20
      7521 WARD               30
      7654 MARTIN             30
      7934 MILLER             10
      7844 TURNER             30
      7499 ALLEN              30
      7782 CLARK              10
      7698 BLAKE              30
      7566 JONES              20
      7788 SCOTT              20
      7902 FORD               20
      7839 KING               10

14行が選択されました。
```

Extended Data types

“拡張データ型”

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

DEMONSTRATION

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®