Oracle 至 KingbaseES V8 迁移最佳实践

Release V8R6

Nov 21, 2021

北京人大金仓信息技术股份有限公司
Email: support@kingbase.com.cn

目　录

第 1 章

概述

1.1 KingbaseES V8R6 产品介绍

KingbaseES V8R6 数据库（简称 KingbaseES）是北京人大金仓信息技术股份有限公司（简称人大金仓）经过多年努力自主研发的、商用关系型数据库管理系统。

KingbaseES 是国家级、省部级实际项目中应用最广泛的国产数据库产品。它实际应用数量超过 35 万套，覆盖全国二十多个关键领域和行业，及 3650 个县市，并连续五年在国产数据库市场占有率中名列第一。

KingbaseES 支持多种操作系统和硬件平台。它支持中标麒麟、银河麒麟、统信操作系统、其他 Linux 和 Windows 等数十个操作系统产品版本，支持 X86、X86_64 等系列及国产龙芯、飞腾、申威、海光、鲲鹏等 CPU 硬件体系结构，并具备与这些版本服务器和管理工具之间的无缝互操作能力。

针对不同类型的客户需求，KingbaseES 设计并实现了标准版、企业版等多类版本。这些版本全部构建于同一数据库引擎内。在不同平台上，这些版本完全兼容。KingbaseES 数据库应用程序可从笔记本电脑扩展到台式机、大型数据库服务器，以至整个企业网络，而无需重新设计。此外，当用户业务发展需更大的数据处理能力时，KingbaseES 还支持各个版本之间的平滑升级。

1.1.1 企业版

KingbaseES 企业版是人大金仓的核心产品，主要面向企业级的关键业务应用。它具有大型通用、“三高”（高可靠、高性能、高安全）、“两易”（易管理、易使用）和运行稳定等特点。

企业版主要包括如下特性：

- 内置的数据容灾保护
- 高效的查询优化策略
- 多样化数据缓存机制
- 面向大数据的并行处理和集群架构
- 支持国密算法的数据传输和存储保护

- 全方位的访问控制
- 海量数据的管理能力
- 直观易用的系统监控与管理手段
- 与第三方数据的高度兼容
- 对业内主流中间件和其它应用的充分支持

这些特性的强力支撑使得 KingbaseES 面对企业级关键业务应用更加从容和得心应手。

1.1.2 其他

除上述各类版本外，KingbaseES 还为军队和政府等特殊用户提供了定制版本服务。与非定制版本相比，这些版本重点从数据库性能、安全、应用编程接口或客户端工具等方面为客户进行了定制化设计。

1.2 Oracle 兼容特性概览

通常，异构数据库移植的工作量繁重。这些工作量主要来源于：在数据类型、SQL 语言、PL/SQL 语言、甚至客户端应用编程接口等诸多方面对两个数据库所进行的、大量的语法或功能的对齐处理。

同样的，从 Oracle 向 KingbasES 移植的情况也如此。为降低移植工作量，**KingbaseES 在其内部实现了大量的 Oracle 兼容特性**。这些特性从语法或功能上对 Oracle 提供了原生支持。因此，在移植过程中，Oracle 程序只需很少甚至不做任何改动就能在 KingbaseES 环境中运行。

此外，对未提供原生支持的 Oracle 功能，KingbaseES 也给出了相应的移植建议。

1.2.1 数据类型

数据类型是描述数据库系统底层信息资源模式的常用手段。通常，两个数据库系统数据类型的兼容好坏直接影响移植的难易程度。因此，KingbaseES 对 Oracle 的特有数据类型提供了全面的原生支持。这些类型如 Oracle 的 *NUMBER*、*VARCHAR2*、*CHAR(n)* 和 *DATE* 等。不仅如此，KingbaseES 对 Oracle PL/SQL 中使用的复杂数据类型也提供了兼容性支持，这些类型如 Oracle 的 *RECORD* 类型、*%TYPE* 属性、*%ROWTYPE* 属性、关联数组、可变数组和嵌套表等。

关于 interval 数据类型目前 V8R6 和 Oracle 有些差异，主要是精度和语法格式方面。精度方面，V8R6 interval 默认精度为 6 位，Oracle 默认精度为 9 位；语法格式方面，Oracle 支持的语法 V8R6 基本都支持。所以在移植方面主要是精度有些差异，从 Oracle 移植到 V8R6 数据可能会被截断。

1.2.2 SQL 语句

在 KingbasES 中，对大多数常用的 Oracle 特有 SQL 语句均提供了原生支持。这项措施使得 Oracle 应用程序在 KingbaseES 系统中通常只需很少的代码变动就可正常运行。

KingbaseES 主要兼容如下 Oracle 的 SQL 语句：

- *TRUNCATE* 语句
- 层次查询
- *DUAL* 伪表
- *SELECT INTO* 的 *FOR UPDATE* 语句
- *UPDATE*[前缀] 多列更新

- *INSERT INTO TABLE*([前缀] 列)
- *DELETE [FROM]* 语句
- 支持 WITH 子句
- 支持 DBLINK
- 支持 CREATE FORCE VIEW
- 支持 SEQUENCE 访问子句

上述这些 SQL 语句均从语法和语义上实现了 Oracle 兼容。

1.2.3 PL/SQL 语法

KingbaseES 支持如下 Oracle PL/SQL 的常用语法：

- 赋值语句
- *IF-THEN-ELSE* 语句
- *CASE* 语句
- 多种循环语句，如 *LOOP* 语句、*WHILE-LOOP* 语句和 *FOR LOOP* 语句
- *GOTO* 语句
- *%TYPE* 属性和*%ROWTYPE* 属性
- *REF CUSOR* 游标
- *%NOTFOUND*、*%FOUND*、*%ISOPEN* 和 *%ROWCOUNT* 游标属性
- *RETURNING INTO* 语句
- *EXECUTE IMMEDIATE* 语句
- 兼容 Oracle 的动态 SQL
- 支持 BULK COLLECT
- PL/SQL 支持集合类型（关联数组、嵌套表、可变数组）
- PL/SQL 异常处理
- *FORALL* 语句

1.2.4 PL/SQL 对象

KingbaseES 支持如下 Oracle PL/SQL 对象：

- 内置标量函数
- 行级 *BEFORE* 触发器
- 行级 *AFTER* 触发器
- *INSTEAD OF* 触发器
- 匿名块
- 存储过程
- 函数

- 子类型
- 对象类型
- 包

1.2.5 客户端 SQL 交互工具

在实际应用中，通常客户 DDL 脚本和报表是通过 SQL 交互工具移植的。针对这种情况，KingbaseES 提供了如下 SQL 交互工具：

- KSQL：命令行的 SQL 交互工具，类似 Oracle 的 SQL* PLUS。
- 数据库系统管理工具：图形化的 SQL 交互工具，类似 Oracle 的 SQL Developer 图形化工具。

通过上述工具，用户可连接数据库服务器，运行数据库实用程序，发送 SQL 语句，运行 SQL 脚本，或运行 KingbaseES 数据库管理命令实施数据库管理等。

1.3 相关技术资源

本指南重点从语句兼容特性、迁移工具、迁移场景和应用程序移植等几方面描述 Oracle 移植的关键技术和实现方式。在每项技术和实现方式的描述上，本指南只提供有限的内容介绍，并未提供全面的细节说明。所以，用户若需了解某些技术的实现细节还请参照相关的技术资料，这些资料诸如：

- KingbaseES 手册：详尽和全方位地介绍如何高效管理 KingbaseES 数据库系统。其中，这些管理诸如用户管理、存储管理、模式对象管理等。
- KingbaseES 开发指南：提供了 JDBC、ODBC、DCI 和 ESQL 等应用编程接口的详细使用说明。
- KingbaseES 高可用指南：详细描述了系统在高可用方面的支持。
- KingbaseES 快速安装指南：全面介绍了 KingnaseES 各种工具的使用方法，这些工具如 SQL 交互工具 KSQL 和数据迁移工具 EasyTransfer Tool 等。

此外，在本指南示例中，**加粗部分**用来强调当前描述的语言特性，**红色 + 加粗部分**用来标识在当前语言特性上 KingbaseES 和 Oracle 不兼容的内容。

第 2 章

Oracle 语言兼容特性

在 SQL 和 PL/SQL 语言方面，KingbaseES 提供大量的 Oracle 兼容特性。这些特性从数据类型、SQL 语法、标量函数、用户定义包和系统内置包、匿名块、存储过程和触发器等多方面对 Oracle 进行了原生支持。

此外，KingbaseES 对未提供原生支持的 Oracle 功能也提供了相应的解决方案。

2.1 KingbaseES 的 Oracle 兼容配置

KingbaseES 用户可通过设置相关的数据库兼容开关，部分或全部启用 Oracle 兼容特性。在实际应用中，用户可采用以下途径设置 Oracle 兼容开关：

- 在数据库实例 *data* 目录下的 *kingbase.conf* 文件中配置
- 在数据库初始化时设置
- 在用户会话中设置

KingbaseES 提供了多个 Oracle 特性兼容开关。在 Oracle 移植过程中，用户可按需使用这些开关。下表列出 KingbaseES 提供的 Oracle 兼容特性开关。

表 2.1.1: KingbaseES 的 Oracle 兼容特性开关一览表

兼容特性开关	用途说明
ora_input_emptystr_isnull	开关开启时，系统将输入的空串当做 NULL 处理会话级参数，*oracle* 模式下默认是 *on* , *PG* 模式下默认是 *off*）。
ora_forbid_func_polymorphism	开关开启时，函数/存储过程支持多态。（会话级参数，缺省值是 false）
ora_numop_style	开关开启时，integers 操作符当做 numeric 操作符。（会话级参数，缺省值是 false）
ora_open_cursors	设置一个会话中可以同时打开的 DBMS_SQL 游标的最大数目。（限制为：0~65535，默认为 300）
ora_statement_level_rollback	指定 plsql 中是否启用语句级回滚。当指定为 true 时，启动语句级回滚功能；当指定为 false 时，关闭语句级回滚功能。（会话级参数，缺省值是 false）

enable_func_colname	启用或禁用函数兼容 oracle 函数输出格式，设置为 on 启用此功能，函数作为投影列时，输出函数名和其参数列表。设置为 off 时关闭该功能，函数作为投影列时，只输出函数名。（会话级参数，缺省值是 false）
enable_upper_colname	开关开启时，查询结果的列名将转换为大写。（会话级参数，缺省值是 false）
ignore_zero_number	开关开启时，number 类型输出时忽略末尾连续的"0"。（会话级参数，缺省值是 false）
nls_length_semantics	设置字符串类型的长度单位 (char 或 byte)，它和 Oracle 参数 NLS_LENGTH_SEMATICS 的含义一致（会话级参数，缺省值是 char）
trunc_compatible	开关开启时，trunc 函数返回的 timestamp 值与 oracle 一致（会话级参数，缺省值是 false）

2.2 PL/SQL 语言兼容特性

在 PL/SQL 语言方面，KingbaseES 提供了大量的 Oracle 兼容特性。这些特性使得大多数的 Oracle 数据库对象和 SQL 语句移植到 KingbaseES 中无需任何转换。

本节主要介绍 KingbaseES 原生支持的 Oracle 兼容特性，并提供了必要的示例说明。此外，对 KingbaseES 未原生支持的 Oracle 特性，本节还适当地给出了相关的移植建议。

2.2.1 记录类型

为降低 Oracle 移植难度，KingbaseES 在自身的 SQL 数据类型基础上扩展了 *NUMBER* 类型、*VARCHAR2* 类型、*CHAR(n CHAR|BYTE)* 类型和兼容 Oracle 的 *DATE* 类型。不仅如此，在 PL/SQL 中，KingbaseES 还预定义了*%TYPE* 属性、*%ROWTYPE* 属性和 *RECORD* 类型。

2.2.1.1 RECORD 类型

RECORD 是一个复合变量，可以存储不同类型的数据值，类似于 C 语言中的 struct 类型，Table%ROWTYPE 和 Cursor%ROWTYPE 均被认为是 RECORD 类型。PL/SQL 的 *RECORD* 类型只能在 PL/SQL 块和包中进行定义和使用，并且有完整的域定义，但是不能够作为表的列类型。另外 Oracle 的 RECORD 语法 KingbaseES 也已兼容：

RECORD 类型语法如下所示（KingbaseES 已兼容）：

```
TYPE < recordTypeName > IS RECORD (field1 datatype, field2 datatype, …);
```

例 2-1：*RECORD* 类型示例。

```
--KingbaseES 和 Oracle 建表和插入数据的 SQL 语句
CREATE TABLE STUDENT (SID INTEGER, SNAME CHAR(30));
INSERT INTO STUDENT VALUES(1, 'Tom');
INSERT INTO STUDENT VALUES(3, 'John');

--KingbaseES 代码
CREATE OR REPLACE PROCEDURE record (r_sid STUDENT.SID%TYPE) is DECLARE
     TYPE rec_type is RECORD (
                sid STUDENT.SID%TYPE,
                sname STUDENT.SNAME%TYPE);
       rec rec_type;       //定义记录结构
```

(continues on next page)

(continued from previous page)

```
       CURSOR cur IS select SID, SNAME from STUDENT WHERE SID = r_sid;
    BEGIN
       OPEN cur;
       LOOP
          FETCH cur INTO rec.sid, rec.sname;
        EXIT WHEN cur%NOTFOUND;
        RAISE NOTICE 'SID = %', rec.SID;
        RAISE NOTICE 'SNAME = %', rec.SNAME;
    END LOOP;
CLOSE cur;
END;

-- Oracle 代码（KingbaseES 支持）
SQL> CREATE OR REPLACE PROCEDURE record (r_sid STUDENT.SID%TYPE) is
  2     TYPE rec_type is RECORD (
  3             sid STUDENT.SID%TYPE,
  4             sname STUDENT.SNAME%TYPE);
  5     rec rec_type;//定义记录结构
  6     CURSOR cur IS select SID, SNAME from STUDENT WHERE SID = r_sid;
  7  BEGIN
  8     OPEN cur;
  9     LOOP
 10             FETCH cur INTO rec.sid, rec.sname;
 11             EXIT WHEN cur%NOTFOUND;
 12             DBMS_OUTPUT.PUT_LINE('SID = ' || rec.sid);
 13             DBMS_OUTPUT.PUT_LINE('SNAME = ' || rec.sname);
 14     END LOOP;
 15     CLOSE cur;
 16  END;
 17  /
```

2.2.1.2 %TYPE 属性

在 PL/SQL 应用程序中，列属性或变量名加上*%TYPE* 用来标识列类型。此外，*%TYPE* 属性也可用于形式参数的声明。除继承列的数据类型外，*%TYPE* 类型变量不能继承列的其他属性，例如不能继承在列上定义的 *NOT NULL* 属性或者 *DEFAULT* 属性。

KingbaseES *%TYPE* 的语法结构如下所示：

```
name { { table \| view }.column \| variable }%TYPE;
```

该语法与 Oracle 兼容。

例 2-2：*%TYPE* 属性的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
CREATE OR REPLACE PROCEDURE type(r_sid STUDENT.SID%TYPE) IS
        v_sid STUDENT.SID%TYPE;
        v_sname STUDENT.SNAME%TYPE;
        v_sname2 v_sname%TYPE;
begin
 DBMS_OUTPUT.PUT_LINE('r_sid= ' || r_sid);
        select * into v_sid,v_sname from STUDENT WHERE SID = r_sid;
    DBMS_OUTPUT.PUT_LINE('v_sid= ' || v_sid);
    DBMS_OUTPUT.PUT_LINE('v_sname= ' || v_sname);
```

(continues on next page)

(continued from previous page)

```
 DBMS_OUTPUT.PUT_LINE('v_sname2= ' || v_sname || 'abc');
end;
```

2.2.1.3 %ROWTYPE 属性

KingbaseES 的*%ROWTYPE* 属性可用来声明一个记录类型，该类型变量可存放 *SELECT* 检索结果集的一条记录，该属性以表名为前缀，语法如下所示：

```
table_name%ROWTYPE
```

相对的，Oracle 也提供*%ROWTYPE* 属性，它的*%ROWTYPE* 类型变量不仅可存放表数据，而且还可存放通过游标获取的数据，具体语法如下：

```
{table_name|cursor_name|cursor_variable_name}%ROWTYPE
```

如上所述，在该属性上，KingbaseES 和 Oracle 均提供了对表的支持。除此之外，Oracle 还额外提供了对游标的支持。

例 2-3：*%ROWTYPE* 的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
CREATE OR REPLACE PROCEDURE rowtype (r_sid STUDENT.SID%TYPE) is
        v_rec STUDENT%ROWTYPE;
BEGIN
        SELECT SID ,SNAME into v_rec.SID,v_rec.SNAME from STUDENT where SID = r_sid;
        DBMS_OUTPUT.PUT_LINE('SID = ' || v_rec.SID);
        DBMS_OUTPUT.PUT_LINE('SNAME = ' || v_rec.SNAME);
END;
```

2.2.2 集合类型

KingbaseES 支持大多数常用的 Oracle PL/SQL 集合类型 (COLLECTION 类型)。在 KingbaseES 中，它们被称为抽象数据类型。

KingbaseES 的抽象数据类型有三种：关联数组（*Associative Array* ）、嵌套表（*Nested Table* ）和可变数组（*Varray* ）。它们均为基于键-值对的集合类型。在 Oracle 兼容模式中，KingbaseES 用户可使用这些类型获取或修改集合信息。

2.2.2.1 关联数组

关联数组是一种具有唯一键值的集合类型，它具有以下特性：

- 它包含零或多个具有相同数据类型的元素。
- 用户可按键值检索。
- 键值可是整型等数值类型，也可是字符串等非数值类型。
- 键值连续且有序。

KingbaseES 支持关联数组类型，它的语法如下所示：

```
TYPE assoc_type IS TABLE OF  element_type [ NOT NULL ] INDEX BY index_type
index_type ::= INT | VARCHAR | VARCHAR2 | TEXT
```

相对的，Oracle 也支持关联数组类型，它的语法如下所示：

```
TYPE assoc_type IS TABLE OF  element_type [NOT NULL] INDEX BY index_type
index_type ::=INTEGER|BINARY_INTEGER|PLS_INTEGER|VARCHAR2(size)
```

在实际应用中，Oracle 的 *index_type* 通常使用 *PLS_INTEGER* 类型，而 KingbaseES 与之相似的是 *INT* 类型。

例 2-4：关联数组的示例。

```
--KingbaseES 代码
CREATE OR REPLACE PROCEDURE associate_array is DECLARE
TYPE emp_array IS TABLE OF VARCHAR2(10) INDEX BY INT;
    emps emp_array;
    l_low INTEGER;
    v_table VARCHAR2(10);
BEGIN
    emps(1) := 'Tom';
    emps(2) := 'Jone';
    emps(3) := 'Robot';
    emps(4) := 'Mike';
    l_low := emps.first;
    WHILE (l_low is not null) LOOP
        v_table := emps(l_low);
        RAISE NOTICE 'emp1= %', v_table;
        RAISE NOTICE 'emp1= %', emps(l_low);
        l_low := emps.next(l_low);
    END LOOP;
END;

--Oracle 代码
CREATE OR REPLACE PROCEDURE associate_array is TYPE emp_array IS TABLE OF␣
↪VARCHAR2(10) INDEX BY PLS_INTEGER;
    emps emp_array;
    l_low INTEGER;
    v_table VARCHAR2(10);
BEGIN
    emps(1) := 'Tom';
    emps(2) := 'Jone';
    emps(3) := 'Robot';
    emps(4) := 'Mike';
    l_low := emps.first;
    WHILE (l_low is not null) LOOP
             v_table := emps(l_low);
        DBMS_OUTPUT.PUT_LINE('emp1= ' || v_table);
        DBMS_OUTPUT.PUT_LINE('emp1= ' || emps(l_low));
        l_low := emps.next(l_low);
    END LOOP;
END;
```

2.2.2.2 嵌套表

嵌套表是一种基于键-值对的集合类型。与关联数组相比，它的键值只能为整型，即从 1 开始的连续值。

Oracle 和 KingbaseES 嵌套表的语法如下所示：

```
YPE nested_type IS TABLE OF  element_type [ NOT NULL ];
```

2.2.2.3 可变数组

KingbaseES 可变数组的下标类型为 *INT* 且从 1 开始。与嵌套表不同：（1）可变数组在定义时需指定最大元素个数，且实际应用中不能超过该限制。（2）可变数组的下标必须连续。所以，删除元素时，可变数组不能使用 *delete(n)* 和 *delete(m,n)* 方法，但可使用 *delete()* 方法（即删除所有元素），这些方法的相关介绍请参照下一节“集合类型的方法”。

KingbaseES 的可变数组语法和 Oracle 兼容。其中，KingbaseES 的可变数组语法如下所示：

```
TYPE varray_type IS VARRAY(size_limit) OF  element_type [ NOT NULL ];
```

相对的，Oracle 的可变数组语法如下所示：

```
TYPE varray_type IS VARRAY(size_limit ) OF  element_type;
```

2.2.2.4 集合类型的方法

下表列出 KingbaseES 所提供的、兼容 Oracle PL/SQL 的集合方法。

表 2.2.1: KingbaseES 关联数组、可变数组和嵌套表中兼容 Oracle 的集合方法列表

集合方法名	方法用途
COUNT	返回抽象数据类型变量中当前的元素数
FIRST	返回变量的第一个索引值
LAST	返回变量的最后一个索引值
PRIOR(n)	返回指定索引的前一个索引值
NEXT(n)	返回指定索引的后一个索引值
DELETE	删除抽象数据类型变量中的所有元素
DELETE (n)	删除抽象数据类型变量中指定索引的元素 (注：暂时不支持可变数组)
DELETE (n1, n2)	删除索引 index1 和 index2 之间的元素，包括 index1 和 index2(注：不支持可变数组)
LIMIT	返回可变数组中定义的最大元素数限制

表 2.2.2: KingbaseES 可变数组和嵌套表中兼容 Oracle 的集合方法列表

集合方法名	方法用途
EXTEND	在索引最后添加一个值为 NULL 的元素
EXTEND(n)	在索引最后添加 n 个值为 NULL 的元素
EXTEND(n1,n2)	在索引最后复制 n 个索引为 index 的元素值
TRIM	从变量末尾开始删除元素
TRIM(n)	从索引最后删除 n 个元素
EXISTS(n)	判断变量中指定的索引元素是否存在，并返回一个 Boolean 值

2.2.3 子类型

基类型的子集，视定义语句，可能在基类型的基础上新增一些特定的约束。

KingbaseES *SUBTYPE* 的语法结构，如下所示：

```
SUBTYPE sub_type IS base_type [ constraint ] [ NOT NULL ];
constraint ::= ( precision [, scale ] )
            | RANGE low_value..high_value
```

ORACLE *SUBTYPE* 的语法结构与 KingbaseES 不同之处在于 KingbaseES 没有字符类型，因此不支持 *CHARACTER SET* 语法，如下所示：

```
SUBTYPE sub_type IS base_type {[ constraint ] | [CHARACTER SET character_set]}
            [ NOT NULL ];
constraint ::= ( precision [, scale ] )
            | RANGE low_value..high_value
```

在使用上，KingbaseES 的 *SUBTYPE* 与 ORALCE 也存在如下差异：

a. 当函数（存储过程）使用包中定义的 *SUBTYPE* 声明参数时，ORACLE 对相关参数的约束检查仅在 PL/SQL 中调用该函数的情况下生效，如果是在 SQL 中直接调用该函数，则 ORACLE 并不对参数进行约束性检查。KingbaseES 则无论什么情况下调用该函数都会对相关参数进行约束性检查。

例 2-5：子类型的示例。

```
--KingbaseES 代码
DECLARE
  SUBTYPE a IS NUMERIC;
  checking_account a(3, 2);
BEGIN
  checking_account := 2;
  RAISE NOTICE '%', checking_account; -- 2
END;
```

2.2.4 基本过程语句

在 PL/SQL 应用程序中，基本过程语句包括赋值语句和多种 SQL 语句。这些 SQL 语句如 *INSERT*、*UPDATE*、*DELETE*、*SELECT INTO*、*NULL* 和 *EXECUTE IMMEDIATE* 等。

下面，分别介绍在这些语句上 KingbaseES 的 Oracle 兼容情况。

2.2.4.1 赋值语句

在 PL/SQL 中，变量赋值是赋值语句最常见的使用形式。KingbaseES 的赋值语句和 Oracle 兼容，它的语法如下所示：

```
variable := expression;
```

2.2.4.2 块结构

PL/SQL 语言是一种基于块结构的编程语言。它的存储过程、函数或者触发器等均是基于块结构的。

在块结构上，KingbaseES 从语法上兼容 Oracle，具体语法如下所示：

```
[ [ DECLARE ]
     declarations ]
BEGIN
    statements
[ EXCEPTION WHEN exception_condition THEN
        statements [, ...] ]
END
```

例 2-5：块结构的示例。加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
DECLARE
        v_id number(3) := 11;
        v_divided number(3) := 2;
        v_result number(6,2);
BEGIN
        v_result := v_id / v_divided;
       DBMS_OUTPUT.PUT_LINE('v_result= ' || v_result);
       EXCEPTION WHEN others THEN
            DBMS_OUTPUT.PUT_LINE('divided is failed ');
END;
```

2.2.4.3 NULL 语句

NULL 语句是一条" 空" 语句，即它不执行任何操作，只负责把控制传递给下一条语句。例如，在 *IF-THEN* 语句、*LOOP* 语句或存储过程等中的 *NULL* 语句只起到占位符的作用。该语句必须用 *BEGIN* 和 *END* 括起来，否则将报错。有时，*NULL* 语句也可用到异常处理部分。在这种情况下，异常处理将被忽略。

在 *NULL* 语句上，KingbaseES 和 Oracle 从语法上兼容，具体语法如下所示：

```
null_statement ::= NULL;
```

例 2-6：*NULL* 语句的示例。该语句在 KingbaseES 和 Oracle 中均可执行。

```
BEGIN
   NULL;
END;
```

2.2.4.4 FORALL 子句

FORALL 子句利用抽象数据类型，在一次 PL/SQL 操作中执行多次 SQL 语句，而无需循环交互。可以提升 SQL 语句执行的效率，KingbaseES 兼容 Oracle 的 FORALL 子句，具体语法如下：

```
FORALL index IN { [ lower_bound..upper_bound ]
    | [ INDICES OF index_collection [ BETWEEN between_lower AND between_upper ] ]
    | [ VALUES OF index_collection ] }
  [ SAVE EXCEPTIONS ]
  dml_statement;
```

示例：*FORALL* 语句的示例。该语句在 KingbaseES 和 Oracle 中均可执行。

```
DECLARE
  TYPE NumList IS TABLE OF PLS_INTEGER;
  pnums NumList := NumList(11,22,33,44,55);
BEGIN
  FORALL i IN INDICES OF pnums
    INSERT INTO parts1(pnum) VALUES (pnums(i));
END;
```

2.2.4.5 BULK COLLECT 子句

BULK COLLECT 子句利用抽象数据类型，在一次操作中返回整个 SQL 的结果集，而无需依次取数据。可以在 SELECT INTO、FETCH INTO、RETURNING INTO 语句中使用，KingbaseES 兼容 Oracle 的 BULK COLLECT 子句。

例 2-7： *BULK COLLECT* 的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
----Oracle 代码
  CREATE TABLE employees(first_name char(20),last_name char(20),salary int);
  insert into employees values('ftest','ltest',250);
  insert into employees values('ftest1','ltest1',360);
  insert into employees values('ftest2','ltest2',360);
  DECLARE
     TYPE EmployeeSet IS TABLE OF employees%ROWTYPE;
     underpaid EmployeeSet;
     CURSOR c1 IS SELECT first_name, last_name FROM employees;
     TYPE NameSet IS TABLE OF c1%ROWTYPE;
     some_names NameSet;
  BEGIN
     SELECT * BULK COLLECT INTO underpaid FROM employees
        WHERE salary < 5000 ORDER BY salary DESC;
     DBMS_OUTPUT.PUT_LINE(underpaid.COUNT || ' people make less than 5000.');
     FOR i IN underpaid.FIRST .. underpaid.LAST
     LOOP
       DBMS_OUTPUT.PUT_LINE(underpaid(i).last_name || ' makes ' ||
                            underpaid(i).salary);
     END LOOP;
     SELECT first_name, last_name BULK COLLECT INTO some_names FROM employees
        WHERE ROWNUM < 11;
     FOR i IN some_names.FIRST .. some_names.LAST
     LOOP
        DBMS_OUTPUT.PUT_LINE('Employee = ' || some_names(i).first_name
                             || ' ' || some_names(i).last_name);
     END LOOP;
  END;
```

2.2.4.6 RETURNING INTO 子句

在 *INSERT*、*UPDATE* 和 *DELETE* 等 DML 语句后可添加 *RETURNING INTO* 子句，它的作用是：捕获 DML 语句最新插入、修改、或者删除的值。

在 *RETURNING INTO* 子句上，除 returning 外，KingbaseES 均与 Oracle 兼容。例如，当返回多条记录时，二者均抛出查询行数超过请求行数的警告信息。

例 2-8：*RETURNING INTO* 的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
----Oracle 代码
CREATE TABLE STUDENT (SID INTEGER, SNAME CHAR(30));
INSERT INTO STUDENT VALUES(1, 'Tom');
INSERT INTO STUDENT VALUES(3, 'John');
CREATE OR REPLACE PROCEDURE returning(v_sid INT)
IS
    d_id INT;
    i_id INT;
    d_name char(30);
    i_name char(30);
BEGIN
    DELETE FROM student WHERE sid=v_sid RETURNING sid, sname INTO d_id, d_name;

    INSERT INTO student(sid, sname) VALUES(5, 'mike')
    RETURNING sid INTO i_id;

    UPDATE student SET sname = INITCAP(sname)
    WHERE sid=i_id
    RETURNING sname INTO i_name;

    DBMS_OUTPUT.PUT_LINE('d_id= ' || d_id);
    DBMS_OUTPUT.PUT_LINE('d_name= ' || d_name);
    DBMS_OUTPUT.PUT_LINE('i_id= ' || i_id);
    DBMS_OUTPUT.PUT_LINE('i_name= ' || i_name);
END;
```

2.2.4.7 EXECUTE IMMEDIATE 语句

在 PL/SQL 中，*EXECUTE IMMEDIATE* 语句的作用相当于 *PREPARE* 语句和 *EXECUTE* 语句。可使用该语句执行匿名块和动态 SQL 语句。

在该语句上，KingbaseES 兼容 Oracle。

2.2.4.8 PL/SQL 语句属性

用户可利用下面的 PL/SQL 语句属性判断 SQL 语句的执行效果。这些语句属性 KingbaseES 和 Oracle 均支持。

- **SQL%FOUND**

  SQL%FOUND 是布尔属性。当成功执行 *INSERT*、*UPDATE*、*DELETE* 或 *SELECT INTO* 命令后，该属性值为 *true*；否则为 *false*。

  例 2-9：*SQL%FOUND* 的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
BEGIN
     INSERT INTO STUDENT VALUES(3, 'John');
```

(continues on next page)

(continued from previous page)

```
     IF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE('inserted is success');
     END IF;
END;
```

- **SQL%NOTFOUND**

  SQL%NOTFOUND 与 *SQL%FOUND* 的含义相反。如果 *INSERT*、*UPDATE* 或者 *DELETE* 命令对记录操作无效，或者 *SELECT INTO* 命令没有取出任何数据，则该属性值为 *true*；否则为 *false*。

  例 2-10：*SQL%NOTFOUND* 的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
BEGIN
     UPDATE student SET sname = 'Robot' WHERE sid = 9000;
     IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('updated failed');
    END IF;
END;
```

- **SQL%ROWCOUNT**

  SQL%ROWCOUNT 表示 *INSERT*、*UPDATE** 或者 **DELETE* 命令处理的有效记录个数。下面这个示例更新了上一个示例插入的记录，并显示了 *SQL%ROWCOUNT* 的值。

  例 2-10：*SQL%ROWCOUNT* 的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
BEGIN
    UPDATE student SET sname = INITCAP(sname) WHERE sid=1;
    DBMS_OUTPUT.PUT_LINE('# rows updated: ' || SQL%ROWCOUNT);
END;
```

2.2.4.9 SQL 语句

在 PL/SQL 中，用户可通过 SQL 语句修改用户数据和设置执行模式。下面这些 SQL 语句在 KingbaseE 和 Oracle 中定义和使用相同。

表 2.2.3: SQL 语句列表

SQL 语句	说明
SELECT INTO	生成一行结果，并将该结果赋予一个（或多个）记录变量，或者行类型变量。
INSERT	向表中插入记录
UPDATE	修改表记录
DELETE	删除表记录

2.2.5 控制流语句

控制流语句是 PL/SQL 编程中的核心部件。用户使用它们可控制 SQL 或 PL/SQL 语句的执行逻辑。

2.2.5.1 IF 语句

用户可通过 *IF* 语句条件控制 SQL 语句的执行逻辑。

KingbaseES 提供了以下四种方式的 *IF* 语句：

- *IF ... THEN*
- *IF ... THEN ... ELSE*
- *IF ... THEN ... ELSEIF*
- *IF ... THEN ... ELSEIF ... THEN ... ELSE*

在上述语句上，KingbaseES 和 Oracle 兼容。

例 2-11：*IF* 语句的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
CREATE OR REPLACE PROCEDURE if (p_num INT)
IS
    flag  CHAR(1);
BEGIN
    IF p_num > 50 THEN
        flag := 'Y';
    ELSIF p_num = 50 THEN
        flag := 'N';
    ELSE
            flag := 'N';
    END IF;
    DBMS_OUTPUT.PUT_LINE('falg= ' || flag);
END;
```

2.2.5.2 CASE 语句

KingbaseES 提供两种格式的 *CASE* 语句：简单型 *CASE* 语句和搜索型 *CASE* 语句。

- **简单型 CASE 语句**

  KingbaseES 简单型 *CASE* 语句的语法如下所示：

  ```
  CASE search-expression
      WHEN match-expression [,match-expression [ ... ]] THEN
          statement
       [ WHEN match-expression [,match-expression [ ... ]] THEN
          statement
      ... ]
       [ ELSE
           statement ]
  END CASE;
  ```

  相对的，Oracle 简单型 *CASE* 语句的语法如下所示：

```
CASE selector-expression
    WHEN match-expression THEN
        statement
     [ WHEN match-expression THEN
        statement
     ...]
     [ ELSE
        statement ]
END CASE;
```

从上面的语法描述可看出，KingbaseES 在 *WHEN* 后允许多个匹配表达式，而 Oracle 则只允许一个匹配表达式。

例 2-12：PL/SQL 简单型 CASE 语句的示例。

```
--KingbaseES 代码
DECLARE
   v_sid number;
   v_sname varchar(30);
   CURSOR cur FOR select * from student;
   v_rec number;
BEGIN
   OPEN cur;
   LOOP
       FETCH cur INTO v_sid,v_sname;
       EXIT WHEN cur%NOTFOUND;
       CASE v_sid
           WHEN 3, 5 then v_rec :=3*10;
           WHEN 1 THEN v_rec :=10;
        ELSE
           v_rec :=0;
       END CASE;
       RAISE NOTICE 'v_rec=%',v_rec;
    END LOOP;
    CLOSE cur;
END;

--Oracle 代码
DECLARE
    v_sid number;
    v_sname varchar(30);
    CURSOR cur IS select * from student;
    v_rec number;
BEGIN
    OPEN cur;
    LOOP
       FETCH cur INTO v_sid,v_sname;
       EXIT WHEN cur%NOTFOUND;
       CASE v_sid
           WHEN 3 then v_rec :=3*10;
           WHEN 5 then v_rec :=3*10;
           WHEN 1 THEN v_rec :=10;
        ELSE
           v_rec :=0;
        END CASE;
           DBMS_OUTPUT.PUT_LINE('v_rec= ' || v_rec);
    END LOOP;
    CLOSE cur;
```

(continues on next page)

(continued from previous page)

```
END;
```

- **搜索型 CASE 语句**

  搜索型 *CASE* 语句通过一个或多个布尔型表达式控制语句的执行逻辑。

  在搜索型 *CASE* 语句上，KingbaseES 与 Oracle 兼容，它的语法如下所示：

```
CASE WHEN boolean-expression THEN
    statements
[ WHEN boolean-expression THEN
    statements
[ WHEN boolean-expression THEN
    statements ] ...]
[ ELSE
    statements ]
END CASE;
```

例 2-13：PL/SQL 搜索型 CASE 语句的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
DECLARE
   v_sid number;
   v_sname varchar(30);
    CURSOR cur IS select * from student;
    v_rec number;
BEGIN
    OPEN cur;
    LOOP
       FETCH cur INTO v_sid,v_sname;
       EXIT WHEN cur%NOTFOUND;
       CASE
           WHEN v_sid = 3 THEN v_rec :=3*10;
           WHEN v_sid = 5 THEN v_rec :=3*10;
           WHEN v_sid = 1 THEN v_rec :=10;
        ELSE
           v_rec :=0;
        END CASE;
           DBMS_OUTPUT.PUT_LINE('v_rec= ' || v_rec);
    END LOOP;
    CLOSE cur;
END;
```

2.2.5.3 循环语句

在存储过程、函数和匿名块中，可利用循环语句重复执行语句序列。

在循环语句上，KingbaseES 支持三种类型，即简单 *LOOP* 语句、*WHILE-LOOP* 语句和 *FOR-LOOP* 语句。在这三种类型上，KingbaseES 均对 Oracle 提供了兼容性支持。

- **简单 LOOP 语句**

  简单 *LOOP* 语句定义了一个无条件的循环，并只有遇到 *EXIT* 或者 *RETURN* 命令时才终止循环。

  KingbaseES 和 Oracle 的相关语法如下所示：

```
LOOP
     <statements>
     [< EXIT WHEN condition;>]
END LOOP;
```

例 2-14：简单 *LOOP* 语句的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
DECLARE
    a INTEGER := 0;
BEGIN
    a:= 1;
    LOOP
       EXIT WHEN a > 2;
       a:=a+1;
    END LOOP;
END;
```

- **WHILE-LOOP 语句**

  WHILE-LOOP 语句每次循环之前，首先对条件进行求值。如果条件为真，则执行循环体内的语句序列；否则将跳过循环体并把控制传递给下一条语句。

  KingbaseES 和 Oracle 的 *WHILE-LOOP* 语句语法如下所示：

```
WHILE <condition> LOOP
    <statements>
     [< EXIT WHEN condition>]
END LOOP;
```

例 2-15：*WHILE-LOOP* 语句的示例。该代码在 KingbaseES 和 Oracle 中均可执行。

```
DECLARE
   a INTEGER := 0;
BEGIN
   a:= 1;
    WHILE (a <= 10) LOOP
       EXIT WHEN a > 5;
       a:=a+1;
    END LOOP;
END;
```

- **FOR-LOOP 语句**

  FOR-LOOP 语句将遍历给定 *IN* 范围的内容。如果遍历条件满足，则执行循环；否则将结束循环。

  KingbaseES 和 Oracle 的相关语法如下所示：

```
FOR <loop_variable> IN [REVERSE] <range_of_values> LOOP
    <statements>
     [<EXIT WHEN condition>]
END LOOP;
```

  需说明的是，在 *REVERSE* 实现上，Oracle 和 KingbaseES 在递减计数方式上存在差异。前者是从第二个数字递减到第一个数字，而后者从第一个数字递减到第二个数字，例如：

```
--KingbaseE，counter 值为 5,4,3,2,1
FOR counter  IN REVERSE 5..1  LOOP
--Oracle
FOR counter  IN REVERSE1..5  LOOP
```

例 2-16：*FOR-LOOP* 语句的示例。该代码在 KingbaseES 和 Oracle 中均可执行。

```
DECLARE
    a INTEGER := 0;
BEGIN
   a:= 1;
    FOR a in 1..10 LOOP
       EXIT WHEN a > 5;
    END LOOP;
END;
```

2.2.5.4 GOTO 语句和 LABEL 语句

GOTO 语句将跳转到指定的标签执行。

KingbaseES 和 Oracle 的相关语法如下所示：

```
GOTO label;
<<label>>
    statement;
```

例 2-17：*GOTO* 语句和 *LABEL* 语句的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--KingbaseES 代码
DECLARE
    i INTEGER := 0;
BEGIN
    FOR i IN 1..5 LOOP
        IF i = 3 THEN
            GOTO end_loop;
        END IF;
        RAISE NOTICE 'i = %', i;
    END LOOP;
    <<end_loop>>
        RAISE NOTICE 'end loop,  i = %', i;
END;
```

2.2.6 PL/SQL 异常处理

所谓异常处理是指一段用于捕获和处理运行时错误，并被单独封装的 SQL 程序代码。KingbaseES 从语法上支持 Oracle 的大部分系统预定义异常处理功能。此外，KingbaseES 的异常处理方式和 Oracle 也相同，即异常发生后，系统将立即捕获和处理异常。

KingaseES 的异常处理语法如下所示：

```
[ DECLARE
     [<VariableDeclaration>]
     [<CursorDeclaration>]
     [<UserDefinedExceptionDeclaration>]
```

(continues on next page)

(continued from previous page)

```
    ]
BEGIN
    <Statements>
EXCEPTION
    WHEN ExceptionName [ OR ExceptionName... ] THEN
        <SequenceOfStatements>;
    [ WHEN ExceptionName[ OR ExceptionName... ] THEN
        <SequenceOfStatements>;
      ... ]
END;
```

针对各类异常处理，下面分别介绍 KingbaseES 的 Oracle 兼容情况。

2.2.6.1 系统预定义异常

系统预定义异常通常是指一些经常使用的异常。这类异常的定义已存放在数据库系统中，并可直接使用，无须声明。

下表列出了 KingbaseES 兼容 Oracle 的系统预定义异常。

表 2.2.4: KingbaseES 兼容 Oracle 的系统预定义异常列表

KingbaseES 异常名称	Oracle 对应异常名称	异常说明
CASE_NOT_FOUND	CASE_NOT_FOUND	CASE 语句中没有任何 WHEN 子句满足条件，且没有 ELSE 子句。
COLLECTION_IS_NULL	COLLECTION_IS_NULL	调用一个未初始化的嵌套表或者可变数组的方法 (不包含 EXISTS)，或为一个未初始化的嵌套表或者可变数组的元素赋值。
DUPLICATE_CURSOR	CURSOR_ALREADY_OPEN	打开一个已经打开的游标。
UNIQUE_VIOLATION	DUP_VAL_ON_INDEX	给一个有唯一约束条件的数据字段保存相同的值。
INVALID_CURSOR_NAME	INVALID_CURSOR	操作一个不合法的游标。例如关闭一个未打开的游标。
INVALID_TEXT_REPRESENTATION	INVALID_NUMBER	出现运算、转换、截位或长度的约束错误。
NO_DATA_FOUND	NO_DATA_FOUND	未获取到数据。
INTERNAL_ERROR	PROGRAM_ERROR	PL/SQL 内部错误。
SELF_IS_NULL	SELF_IS_NULL	调用一个为空对象的 MEMBER 方法。
OUT_OF_MEMORY	STORAGE_ERROR	内存溢出。
SUBSCRIPT_BEYOND_COUNT	SUBSCRIPT_BEYOND_COUNT	调用嵌套表或者可变数组时，使用的下标索引超出对应元素的总个数。
SUBSCRIPT_OUTSIDE_LIMIT	SUBSCRIPT_OUTSIDE_LIMIT	调用嵌套表或者可变数组时，使用的下标索引不在合法范围内，如 (-1)。

TOO_MANY_ROWS	TOO_MANY_ROWS	返回太多的结果行。
NUMERIC_VALUE_OUT_OF_RANGE	VALUE_ERROR	数值类型超过定义域。
DIVISION_BY_ZERO	ZERO_DIVIDE	除零错误。

2.2.6.2 用户自定义异常

KingbaseES 的用户自定义异常规则与 Oracle 一致，语法如下所示：

```
exception_name EXCEPTION;
PRAGMA EXCEPTION_INIT (exception, error_code) ;
```

2.2.6.3 RAISE_APPLICATION_ERROR 语句

Oracle 提供 *RAISE_APPLICATION_ERROR* 可主动产生异常，并中断 PL/SQL 程序。它的语法如下所示：

```
RAISE_APPLICATION_ERROR(error_number, message);
```

KingbaseES 中也已支持相同功能。

例 2-18：*RAISE_APPLICATION_ERROR* 的示例。

```
CREATE OR REPLACE PROCEDURE raise_error (p_id IN number) IS
    rec student%ROWTYPE;
BEGIN
    select sid, sname  into rec from STUDENT where  sid = p_id;
    if rec.sid < 3 then
        RAISE_APPLICATION_ERROR (-20100, 'this is less than 1');
    elsif rec.sid < 5 then
        RAISE_APPLICATION_ERROR (-20030, 'this is less than 5');
    else
        RAISE_APPLICATION_ERROR (-20040, 'this is more than 5');
    end if;
END;

-- Oracle SQL PLUS 运行结果-
第 1 行出现错误:
ORA-20100: this is less than 1
ORA-06512: 在 "SYSTEM.RAISE_ERROR", line 7
ORA-06512: 在 line 1

-- KingbaseES ksql 运行结果-
ERROR:  ERRCODE-20100: this is less than 1
CONTEXT:  PL/SQL function raise_application_error(integer,text,boolean) line 18 at␣
↪RAISE
SQL statement "CALL RAISE_APPLICATION_ERROR (-20100, 'this is less than 1')"
PL/SQL function raise_error(numeric) line 6 at CALL
```

2.2.7 游标

游标是内置数据类型，它能把 SQL 操作结果集以记录为单位逐条取出。在如下场景中可使用游标：

- 用户定义的游标类型
- 全局变量
- 过程和函数的参数
- 局部变量
- 函数的返回类型

那么，如何使用游标呢？

- 通过 *SELECT* 语句声明一个游标。
- 打开游标，此时可附带可选的输入参数。
- 在循环中，可通过游标获取一条记录，并把该记录的各个字段值分别存放到应用程序变量或 PL/SQL 变量中。
- 关闭游标，并释放游标所用的数据库资源。出于性能考虑，当不再使用游标时，应尽早关闭游标。

此外，对一个 SQL 结果集而言，由于游标方式每次只能获取一条记录，因此，该方式性能逊于一次获取整个结果集的方式。

在 KingbaseES V8R6 中游标的相关功能基本已于 ORACLE 兼容，只是部分功能在语法上与 ORACLE 稍有差距。其中主要包括 refcursor 游标，隐式游标以及游标参数默认值的相关语法。

2.2.7.1 REF 游标

KingbaseES 的 REF 游标（即 REFCURSOR 游标）是一种弱类型的动态游标。相对的，Oracle 也提供了内置弱类型的 SYS_REFCURSOR 游标，该游标是 Oracle 9i 以后系统定义的一个 REF 游标。

所谓弱类型游标是指该类游标可与任何结果集关联，并可在程序的不同地方具有不同的结果集类型。相对的，强类型游标则只能与特定的结果集关联。在处理动态结果集时，由于用户不能预先获知结果集的结构信息，则此时可采用弱类型游标。另外，这类游标只能在运行时获取结果集的结构信息，因此，在编译阶段，无法对它预先进行类型检测。

例 2-19：*REF* 游标的示例。

```
--KingbaseES 代码
create or replace procedure p_test(p_id number) as
declare
   refcur refcursor;
   v_sid student.sid%type;
   v_sname student.sname%type;
begin
    open refcur for select * from student where sid>p_id;
    loop
        fetch refcur into v_sid,v_sname;
        exit when refcur%notfound;
        raise notice 'v_id=%',v_sid;
        raise notice 'v_name=%',v_sname
    end loop;
   close refcur;
end;

--Oracle 代码
```

(continues on next page)

(continued from previous page)

```
CREATE OR REPLACE PROCEDURE p_test( p_id int)
is
    refcur sys_refcursor;
    v_sid student.sid%type;
    v_sname student.sname%type;
begin
    open refcur for select * from student where sid > p_id;
    loop
        fetch refcur into v_sid,v_sname;
        exit when refcur%notfound;
        dbms_output.put_line('v_sid='||v_sid);
        dbms_output.put_line('v_sname='||v_sname);
    end loop;
    close refcur;
end;
```

2.2.7.2 隐式游标

通常，在存储过程中处理 *INSERT*、*UPDATE*、*DELETE* 或 *SELECT INTO* 语句时，将隐式地打开一个游标。用户虽然不能通过 *OPEN*、*FETCH* 或 *CLOSE* 语句控制隐式游标，但他们可通过游标属性获取这些 SQL 语句的执行效果。

其中 ORACLE 的隐式游标调用格式为 SQL%FOUND、SQL%NOTFOUND 和 SQL%ROWCOUNT，其中由于隐式游标总是打开的，所以 SQL%ISOPEN 没有意义。KingbaseES V8R6 在兼容 ORACLE 语法的基础上，也可以通过 FOUND、NOTFOUND 和 ROWCOUNT 直接调用隐式游标的属性功能。需要注意的是，如果此时有变量名为 FOUND、NOTFOUND 或者 ROWCOUNT，那么将会导致隐式游标被变量覆盖而导致隐式游标不可用（一旦有这三个用户自定义变量，也会导致 sql%found 等三个属性变量不可用，所以在需要使用隐式游标时，变量名应避开 FOUND、NOTFOUND 和 ROWCOUNT）。

例 2-20：隐式游标的使用示例。

```
--KingbaseES 代码
create or replace procedure p_test(p_id number) as
declare
   refcur refcursor;
   v_sid student.sid%type;
   v_sname student.sname%type;
begin
    open refcur for select * from student where sid>p_id;
    loop
        fetch refcur into v_sid,v_sname;
        exit when notfound;  --或者 sql%notfound
        raise notice 'v_id=%',v_sid;
        raise notice 'v_name=%',v_sname
    end loop;
   close refcur;
end;

--Oracle 代码
CREATE OR REPLACE PROCEDURE p_test( p_id int)
is
    refcur sys_refcursor;
    v_sid student.sid%type;
    v_sname student.sname%type;
begin
```

(continues on next page)

(continued from previous page)

```
    open refcur for select * from student where sid > p_id;
    loop
        fetch refcur into v_sid,v_sname;
        exit when sql%notfound;
        dbms_output.put_line('v_sid='||v_sid);
        dbms_output.put_line('v_sname='||v_sname);
    end loop;
    close refcur;
end;
```

在特定的 PL/SQL 结构中，隐式游标能被自动声明、打开和关闭，这种游标如 *FOR LOOP* 中使用的游标，或在 *SELECT INTO* 语句中抽取一行记录时所隐含使用的游标。此外，*FOR LOOP* 游标是一种强类型游标。

KingbaseES *FOR LOOP* 游标的语法定义如下。该语法和 Oracle 兼容。

```
FOR recordVar IN cursorVar LOOP

--SequenceOfStatement

END LOOP;
```

游标变量 *cursorVar* 在 *FOR* 语句执行之前不用 *OPEN*，并在 *FOR* 语句执行之后也不用 *CLOSE*，循环结束后系统将自动关闭游标。

例 2-21：*FOR LOOP* 游标的示例。它的加粗部分内容可在 Oracle 环境中运行。

```
--KingbaseES 代码
DECLARE
    rec STUDENT%ROWTYPE;
BEGIN
    FOR rec IN (SELECT sid, sname FROM student) LOOP
        RAISE NOTICE 'rec.sid = %', rec.sid;
        RAISE NOTICE 'rec.sname = %', rec.sname;
    END LOOP;
END;
```

2.2.7.3 参数化游标

所谓参数化游标是指用户可声明一个用于接收参数的静态游标。当打开游标时，系统可通过游标参数传递数值。KingbaseES 和 Oracle 均支持参数化游标，二者在语法上存在差异，它们的语法分别如下所示：

```
--KingbaseES
CURSOR cursorName (parameterName parameterTYpe default Values) FOR /IS
↪<SelectStatement>;
--Oracle
CURSOR cursorName (parameterName parameterTYpe default Values) IS <SelectStatement>;
```

如上面语法所示，二者的差异为：一个可以是 *FOR* 或者 *IS*，另外一个只能是 *IS*。

例 2-22：参数化游标的示例。

```
--KingbaseES 代码
declare
    rec STUDENT%ROWTYPE;
    cursor cur(v_id int :=2) for select * from student where sid >v_id;
begin
```

(continues on next page)

(continued from previous page)

```
    for rec in cur loop
        raise notice 'sid= %',rec.sid;
        raise notice 'sname= %',rec.sname;
    end loop;
end;

--Oracle 代码
declare
    rec STUDENT%ROWTYPE;
    cursor cur(v_id int:=2) is select * from student where sid >v_id;
begin
    for rec in cur loop
        dbms_output.put_line('rec.sid='||rec.sid);
        dbms_output.put_line('rec.sname='||rec.sname);
    end loop;
end;
```

针对有默认值的游标参数，Oracle 的处理方式为在定义时，有默认值的参数排序靠右，当 open 传递参数值时，则从左往右按位置依次赋予。KingbaseES V8R6 在兼容 Oracle 的基础上，也支持指定参数名传参的方式传递参数值。

例 2-23：参数名传参的示例。

```
--KingbaseES 代码
declare
    rec STUDENT%ROWTYPE;
    cursor cur(v_id int :=2, v_name varchar5) for select * from student where sid >v_
↪id, sname = v_name;
begin
    for rec(v_name = 'du') in cur loop
        raise notice 'sid= %',rec.sid;
        raise notice 'sname= %',rec.sname;
    end loop;
end;

   --Oracle 代码
declare
    rec STUDENT%ROWTYPE;
    cursor cur(v_name varchar5 , v_id int:=2) is select * from student where sid >v_
↪id, sname = v_name;
begin
    for rec in cur('du') loop
        dbms_output.put_line('rec.sid='||rec.sid);
        dbms_output.put_line('rec.sname='||rec.sname);
    end loop;
end;
```

2.2.7.4 静态游标

静态游标是相对于前面 *REF CURSOR* 等动态游标而言的，这类游标在编译阶段就和特定的 SQL 语句关联。因此，在编译阶段，对这类游标可进行类型检测。

类似参数化游标，在静态游标上，KingbaseES 和 Oracle 的差异为：一个可以是 *FOR* 或者 *IS*，另外一个只能是 *IS*。

例 2-24：静态游标的示例。

```
--KingbaseES 代码
DECLARE
    rec STUDENT%ROWTYPE;
    CURSOR cur FOR select * from STUDENT;
BEGIN
    OPEN cur;
    LOOP
        FETCH cur INTO rec;
        EXIT WHEN cur%NOTFOUND;
        RAISE NOTICE 'SID = %', rec.SID;
        RAISE NOTICE 'SNAME = %', rec.SNAME;
    END LOOP;
END;

--Oracle 代码
DECLARE
    rec STUDENT%ROWTYPE;
    CURSOR cur IS select * from STUDENT;
BEGIN
    OPEN cur;
    LOOP
        FETCH cur INTO rec;
        EXIT WHEN cur%NOTFOUND;
        dbms_output.put_line('rec.sid='||rec.sid);
        dbms_output.put_line('rec.sname='||rec.sname);
    END LOOP;
END;
```

2.2.7.5 游标属性

每个声明的游标都具有如下属性：*%NOTFOUND*、*%FOUND*、*%ISOPEN* 和 *%ROWCOUNT*。这些属性只能在过程语句，而不能在 SQL 语句中使用。

2.2.8 静态和动态 SQL 语句

在 PL/SQL 中，可使用静态和动态两种 SQL 语句。其中，静态 SQL 语句编译后被存储在数据库中以供将来运行时使用。相对的，动态 SQL 语句是在运行时实时编译的。

KingbaseES 兼容 Oracle 的静态 SQL 语句，这类语句包括 *DDL*、*DML* 和事务控制语句（如 *COMMIT* 和 *ROLL BACK* 语句）。

KingbaseES 还提供了动态 SQL 功能，它的语法如下所示：

```
EXECUTE [IMMEDIATE] sql_expression;
[ INTO { variable [, ...] \| record } ]
[ USING expression [, ...] ]
```

在动态 SQL 语句上，Oracle 只能使用 *EXECUTE IMMEDIATE* 方式。相对的，KingbaseES 不仅兼容 Oracle 此种用法，而且还可不用 *IMMEDIATE*，直接使用 *EXECUTE*。

动态 SQL 语句不是在编译而是在运行时编译，因此其性能稍逊于静态 SQL 语句。此外，动态 SQL 语句实现上也存在安全隐患，例如它为非法用户实施恶意 SQL 注入提供了可能性。

例 2-25：静态语句和动态 SQL 的示例。该代码在 KingbaseES 和 Oracle 均可执行。

```
--静态
CREATE OR REPLACE PROCEDURE static_sql()
AS DECLARE
BEGIN
    INSERT INTO STUDENT VALUES(5, 'Mike');
    INSERT INTO STUDENT VALUES(7, 'Robot');
    UPDATE STUDENT SET sid=99999 where sid=5;
    DELETE STUDENT WHERE sid='7';
END;

--动态
CREATE OR REPLACE PROCEDURE dynamic_sql () AS
DECLARE
BEGIN
    EXECUTE IMMEDIATE 'INSERT INTO STUDENT VALUES(5, ''Mike'')';
    EXECUTE IMMEDIATE 'INSERT INTO STUDENT VALUES(7, ''Robot'')';
    EXECUTE IMMEDIATE 'UPDATE STUDENT SET sid=99999 where sid=5';
    EXECUTE IMMEDIATE 'DELETE STUDENT WHERE sid=''7''';
END;
```

例 2-26：*execute immediate* 的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
declare
    v_sid student.sid%type;
    v_sname student.sname%type;
    v_sql varchar(100) := 'delete from student where sid = :id1 returning sid,sname␣
↪into :1,:2';
begin
    execute immediate v_sql using 5 returning into v_sid,v_sname;
    dbms_output.put_line('v_sid='||v_sid);
    dbms_output.put_line('v_sname='||v_sname);
end;
```

2.2.9 匿名块、存储过程和函数

PL/ SQL 存储过程和函数属于命名块。用户可使用 *CREATE PROCEDURE* 或 *CREATE FUNCTION* 语句创建它们。创建成功后，它们将被存放在数据库中。虽然它们均由可执行语句构成，但二者之间存在差异：

- 存储过程不能有返回值，而函数必须有返回值。
- 在运行时，存储过程可在 PL/SQL 语句、另外一个存储过程、函数、触发器或匿名块中被调用。相对的，函数一般作为 SQL 语句或赋值语句的表达式使用。

与上面命名块相比，匿名块并未永久地存放在数据库中。

KingbaseES 和 Oracle 均支持匿名块、函数和存储过程功能，并在这些功能上二者完全兼容或非常相似。

2.2.9.1 匿名块

匿名块是指未命名的代码块。当它执行结束后，系统通常将其从程序缓存中清除。下一次执行时，则需重新载入。因此，对于需实时修改的简单测试程序来说，匿名块是非常有用的。当一个匿名块需多次重复执行时，用户则可把它转换成存储过程或者函数，以便长期保存在数据库中。

在匿名块方面，KingbaseES 兼容 Oracle，语法如下所示：

```
DECLARE
  object_declaration_1,..., object_declaration_n
  [subprogram_declaration_1,..., subprogram_declaration_n]
BEGIN
  statement_1,...,statement_n
  [EXCEPTION exception_handle_1,..., exception_handle_n]
END;
```

例 2-27：匿名块的示例。

```
declare
  cursor cur for select * from student;
  rec student%ROWTYPE;
begin
  open cur;
  fetch cur into rec;
     dbms_output.put_line(''||rec.sid);
  close cur;
end;
```

2.2.9.2 存储过程

KingbaseES 从存储过程创建、修改、调用和删除等多方面均提供了 Oracle 兼容性支持。

- **存储过程创建**

  KingbaseES 存储过程创建的语法如下所示：

  ```
  CREATE [ OR REPLACE ] PROCEDURE procedurename ([ (parameters) ])
  { IS | AS }
  [DECLARE]
      [<declarations>]
  BEGIN
      statements
  END [ name ];
  ```

  和 KingbaseES 相比，Oracle *IS* 或 *AS* 后面不能加的 *DECLARE*。

  例 2-28：一个简单的存储过程示例。该 Oracle 代码在第二行加“declare”后便可在 KingbaseES 中运行。

  ```
  --Oracle 代码
  create or replace procedure proc_test is
  /*declare*/
      v_id number(3);
      v_divided number(3);
      v_result number(6,2);
  begin
      v_id := 11;
  ```

(continues on next page)

(continued from previous page)

```
    v_divided := 2;
    v_result := v_id / v_divided;
end;
```

- **存储过程修改**

  在存储过程修改上，KingbaseES 和 Oracle 存在差异，下面是它们的语法。

  KingbaseES 的相关语法如下所示：

```
ALTER PROCEDURE ProcedureName [ ( [<ExpressionList>] ) ] { RENAME TO␣
↪NewProcedureName  | COMPILE };
```

  相对的，Oracle 的相关语法如下所示：

```
ALTER PROCEDURE [ schema. ] procedure
COMPILE [ DEBUG ]
[ compiler_parameters_clause ...
[ REUSE SETTINGS ]
;
```

- **存储过程调用**

  KingbaseES 存储过程调用的语法如下所示：

```
[CALL] procName [ (parameters) ];
```

  相对的，Oracle 存储过程调用的语法如下所示：

```
[EXEC] procName [ (parameters) ];
```

  此外，二者在匿名块中调用存储过程的语法如下所示：

```
BEGIN procName [ (parameters) ] END;
```

  例 2-29：从匿名块调用存储过程的示例。该示例代码在 KingbaseES 和 Oracle 上均可运行。

```
BEGIN
proc_test();
END;
```

- **存储过程删除**

  在存储过程删除上，KingbaseES 和 Oracle 兼容，它的语法如下所示：

```
  DROP PROCEDURE p*\ rocedurename;

** 例 2-30：**\ 存储过程删除的示例。该示例代码在 KingbaseES 和 Oracle 上均可运行。
```

```
DROP PROCEDURE proc_test;
```

2.2.9.3 函数

- **函数创建**

  类似存储过程，KingbaseES 函数创建的语法如下所示，该语法兼容 Oracle。

```
CREATE [ OR REPLACE ] FUNCTION name [ (parameters) ]
RETURN data_type
[ AUTHID { DEFINER | CURRENT_USER } ]
{ IS | AS }
[ declarations ]
BEGIN
statements
END [ name ];
```

例 2-31：一个简单的无参数函数创建的示例。该函数能在 KingbaseES 和 Oracle 上运行。

```
CREATE OR REPLACE FUNCTION simple_function
RETURN VARCHAR2
IS
BEGIN
    RETURN 'That''s All Folks!';
END simple_function;
```

- **函数调用**

  在 PL/SQL 中，表达式可出现的任何地方均可进行函数调用。

  KingbaseES 函数调用的语法如下所示：

```
functionName ([parameters])
```

在函数调用上，KingbaseES 和 Oracle 基本兼容。它们函数使用差异如下表所示：

表 2.2.6: KingbaseES 和 Oracle 函数对比表

	KingbaseES	**Oracle**
差异说明	函数定义时，没参数不需要加括号。	
	在调用函数时，KingbaseES 后面必须加括号。	在调用函数时，Oracle 在无参数时，对函数后面的括号可加可不加。
	在传入与传出参数时必须指定类型长度，否则有精度损失，如 number 型，在接收参数时如不指定其精度与标度，其结果所有的 NUMBER 型数据都是以默认的四舍五入的方式整数传入，其标度忽略。	在传入参数时不能定义其类型的长度，其传入与传出参数的长度由系统自动处理。

表 2.2.7: KingbaseES 和 Oracle 函数对比表（示例）

	KingbaseES	Oracle
示例	//传入和传出字符型参数不定义长度 *TEST-# create or replace function p_fun***(name1 varchar) return varchar** *TEST-# is* *TEST-# begin* *TEST-# return name1;* *TEST-# end;* *TEST-# /* *CREATE FUNCTION* *TEST-# call p_fun('abcd');* *TEST-# /* **ERROR: 类型 VARCHAR(1) 的值过长** **CONTEXT: PL/SQL 函数"p_fun" 保存调用参数到本地变量**	//传入和传出字符型参数不定义长度 *SQL> create or replace function p_fun***(name1 varchar) return varchar** *2 is* *3 begin* *4 Return name1;* *5 end;* *6 /* 函数已创建。 *SQL> select p_fun('abcdfd') from dual;* **P_FUN('ABCDFD')** **----------** **abcdfd**
	//传入字符型参数定义长度 *TEST=# create or replace function p_fun(name1* **varchar(20)***) return varchar* *TEST-# is* *TEST-# begin* *TEST-# return name1;* *TEST-# end;* *TEST-# /* *CREATE FUNCTION* *TEST=# call p_fun('abcdf');* *TEST-# /* **P_FUN** **-------** **abcdf** **(1 行)**	//传入字符型参数定义长度 *SQL> create or replace function p_fun(name1* **varchar(20)***) return varchar* *2 as* *3 begin* *4 return name1;* *5 end;* *6 /* 警告: 创建的函数带有编译错误。
	//传入和传出数值型 *TEST=# create or replace function p_fun(***id int,sal number) return number** *TEST-# is* *TEST-# begin* *TEST-# return id*sal;* *TEST-# end;* *TEST-# /* *CREATE FUNCTION* *TEST=# call p_fun(10,11.99);* *TEST-# /* **P_FUN** **-------** **120** **(1 行)**	//传入和传出数值型 *SQL> create or replace function p_fun(***id int,sal number) return number** *2 is* *3 begin* *4 return id*sal;* *5 end;* *6 /* 函数已创建。 *SQL> select p_fun(10,11.999) from dual;* **P_FUN(10,11.999)** **----------** **119.99**

例 2-32：函数调用的示例。它的加粗部分内容可直接迁移到 KingbaseES 环境中。

```
--Oracle 代码
BEGIN
```

(continues on next page)

(continued from previous page)

```
   dbms_output.put_line('res='||simple_function());
END;
```

例 2-33：无参函数调用的示例。下面，加粗部分内容在 Oracle 中可正确执行，而在 KingbaseES 中则报错。

```
BEGIN
    dbms_output.put_line('res='||simple_function);
END;
```

- **函数删除**

  在函数删除上，KingbaseES 和 Oracle 基本兼容。

  KingbaseES 的相关语法如下所示：

```
DROP FUNCTION FunctionName [ ( [<ParameterList>] ) ] [ CASCADE \|RESTRICT ];
```

  相对的，Oracle 的相关语法如下所示：

```
DROP FUNCTION funcationName;
```

  此外，对于函数重载，KingbaseES 与 Oracle 不同，KingbaseES 支持函数重载，而 Oracle 不支持。

  例 2-34：删除一个已创建的、无参函数的示例。

```
--KingbaseES 代码
drop function simple_function();
--Oracle 代码
drop function simple_function;
```

- **存储过程和函数的返回参数处理**

  在存储过程和函数的返回参数上，KingbaseES 和 Oracle 处理不同：KingbaseES 需把返回的 OUT 值赋给一个变量，而 Oracle 可直接对返回 OUT 值进行处理。

  例 2-35：存储过程返回参数处理的示例。

```
--KingbaseES 代码
--创建存储过程
create or replace procedure out_parameter (v_sid in out int,v_sname out␣
↪student.sname%TYPE)
as
begin
        select * into v_sid,v_sname from student where sid = v_sid;
end;

--调用存储过程
declare
   res student%ROWTYPE;
   v_sid int;
    v_sname varchar(30);
begin
   v_sid := 7;
   res = out_parameter(v_sid, v_sname); //把返回值赋给 res 变量
    raise notice 'sid= %',res.sid;
    raise notice 'sname= %',res.sname;
end;
```

(continues on next page)

(continued from previous page)

```
--Oracle 代码
--创建存储过程
create or replace procedure out_parameter(v_sid in out int,v_sname out
↪student.sname%TYPE) as
begin
    select sid, sname into v_sid,v_sname from student where sid = V_sid;
end;
--调用存储过程
declare
   v_sid int;
    v_sname varchar(30);
begin
    v_sid := 7;
    out_parameter(v_sid, v_sname);//没有把返回值赋给一个变量
        dbms_output.put_line('v_sid='||v_sid);
        dbms_output.put_line('v_sname='||v_sname);
end;
```

2.2.10 对象类型

对象类型在使用上与组合类型基本一致。不同于组合类型，对象类型在创建时包括创建类型声明和创建类型体 (CREATE TYPE BODY), 在创建对象类型声明时除了指定与对象类型相同的属性 (列) 外，还可以声明静态函数（存储过程）、成员函数（存储过程）、构造函数三种内部方法。并在相应的类型体内分别为其创建完整的方法定义。

对象类型在创建时总会创建一个默认的构造函数，默认构造函数的参数列表与对象类型的列成员一致，默认构造函数会将作为参数传入的列成员构造成对应的对象返回，当默认构造函数与用户自定义构造函数参数相同时，将优先调用用户自定义的构造函数。

由于 KingbaseES 不提供自定义类型的映射函数，也不提供自定义类型的排序函数。因此不支持以下 Oracle 语法：

```
{ MAP } MEMBER function_spec
{ ORDER } MEMBER function_spec
```

对象类型的创建、修改、调用和删除等多方面均提供了 Oracle 兼容性支持，语法如下所示：

```
CREATE [ OR REPLACE ] TYPE name [ FORCE ] [ AUTHID { CURRENT_USER | DEFINER } ] { AS
↪| IS } OBJECT (
    attribute_name data_type [, ... ]
    [, subprogram_spec [, ... ] ]
    [, constructor_spec [, ... ] ]
)
这里 subprogram_spec 是以下之一：
    { STATIC | MEMBER } FUNCTION function_name
        [ ( [ argname [ argmode ] argtype [ { DEFAULT | := } default_expr ] [, ...] ]
↪) ]
        { RETURNS | RETURN } rettype ]
    { STATIC | MEMBER } PROCEDURE procedure_name
        [ ( [ argname [ argmode ] argtype [ { DEFAULT | := } default_expr ] [, ...] ]
↪) ]

constructor_spec 是:
    CONSTRUCTOR FUNCTION name
```

(continues on next page)

(continued from previous page)

```
        [ ( [ SELF IN OUT name ] [, argname [ argmode ] argtype [ { DEFAULT | := }␣
↪default_expr ] [, ...] ] ) ]
        { RETURNS | RETURN } SELF AS RESULT
```

例 2-36：OBJECT TYPE 的创建示例。

```
--KingbaseES 代码
CREATE OR REPLACE TYPE objtype FORCE AUTHID CURRENT_USER AS OBJECT(
    f1 int,
    f2 varchar2(10),
    MEMBER FUNCTION memfunc(i int) RETURN INT,
    STATIC PROCEDURE staproc,
    CONSTRUCTOR FUNCTION objtype(self in out objtype) RETURN SELF AS RESULT
);

\set SQLTERM /
CREATE OR REPLACE TYPE BODY objtype AS
    MEMBER FUNCTION memfunc(i int) RETURN INT AS
    BEGIN
      RAISE NOTICE 'self.f1 is %', self.f1;
      RAISE NOTICE 'self.f2 is %', f2;
      RETURN i;
    END;

    STATIC PROCEDURE staproc AS
    BEGIN
      RAISE NOTICE 'this is a static procedure in object type';
    END;

    CONSTRUCTOR FUNCTION objtype(self in out objtype) RETURN SELF AS RESULT AS
    BEGIN
      f1 := 1;
      self.f2 := 'a';
      RETURN ;
    END;
END;
/
\set SQLTERM  ;

对象类型成员函数及构造函数的使用：
CREATE TABLE tb1(i objtype);
INSERT INTO tb1 VALUES(objtype()); -- 自定义构造函数
INSERT INTO tb1 VALUES(objtype(2, 'b')); -- 默认构造函数
SELECT t.i.memfunc(1) FROM tb1 t;
对象类型静态存储过程的使用：
CALL objtype.staproc();
```

2.2.11 包

包是由过程化语言书写，并经编译和优化后存储在数据库服务器中的变量、常量、游标、存储过程和函数的集合。

2.2.11.1 用户自定义包

KingbaseES 从包的创建、修改、调用和删除等多方面均提供了 Oracle 兼容性支持。

例 2-36：自定义包的创建示例。

```
CREATE OR REPLACE PACKAGE pac AS
  var INT := 100;
  FUNCTION get_var RETURN INT;
  PROCEDURE set_var(new_var INT);
END;

CREATE OR REPLACE PACKAGE BODY pac AS
  FUNCTION get_var RETURN INT AS
  BEGIN
    RETURN pac.var;
  END;

  PROCEDURE set_var(new_var INT) AS
  BEGIN
    pac.var := new_var;
  END;
END;

select pac.get_var() from dual;
call pac.set_var(200);
select pac.get_var() from dual;
```

2.2.11.2 系统内置包

KingbaseES 数据库提供了一些内置包，开发应用程序时可以直接使用 Kingbase 内置包中的过程与函数，既可以简化应用开发的复杂性，又能提高程序的运行性能。

- DBMS_OUTPUT 包的兼容

  KingbaseES 数据库服务器已经含有 DBMS_OUTPUT 包，但如果想要打印在屏幕上，需要在客户端使用 SET SERVEROUTPUT ON 命令，目前只有 KSQL 工具支持 SET SERVEROUTPUT 命令。

在 Oracle 中通过 DBMS_OUTPUT.put_line 函数在调试时输出一些变量的值。

```
--Oracle 代码：
SET SERVEROUTPUT ON
DECLARE
   i  int = 10;
BEGIN
  DBMS_OUTPUT.PUT_LINE( 'I = '  || i);
END;
/

--KingbaseES 代码：
SET SERVEROUTPUT ON
```

(continues on next page)

(continued from previous page)

```
\set SQLTERM /
DECLARE
   i  int = 10;
BEGIN
  DBMS_OUTPUT.PUT_LINE( 'I = '  || i);
END;
/
\set SQLTERM ;
```

2.2.12 触发器

触发器是一种特殊类型的存储过程，用户不能直接调用。但是，当插入、删除或修改表或视图数据等特定事件发生时，它将被自动触发执行。

KingbaseES 和 Oracle 均支持触发器功能，下表列出了它们在触发器方面的主要差异。除这些差异外，其他方面 KingbaseES 和 Oracle 兼容。

表 2.2.8: KingbaseES 和 Oracle 触发器差异表

对比点	**Oracle**	**KingbaseES**
变量	:NEW :OLD	支持，也支持 NEW OLD
INSTEAD OF 触发器	支持	支持
WHEN 触发器	支持	支持
WHERE 触发器	支持	支持
AS 关键字	不需要 AS 关键字	支持，可加 AS 关键字
编译	支持 COMPILE	不支持
END 语法	END 后可以有触发器名称	支持
INSERTING UPDATING DELETING	支持 INSERTING、UPDATING、DELETING	支持
触发事件	支持 DDL 事件和 DATABASE 事件	支持 DDL 事件和 DATABASE 事件
无法对 DBA 拥有的对象创建触发器	支持	不支持
声明新老值的别名	REFERENCING OLD AS ” OLD” NEW AS ”NEW”	REFERENCING { OLD | NEW } TABLE [AS] transition_relation _name
OLD 和 NEW	在 BEFORE INSERT 的触发器中使用 WHEN (old.pindex is null)	INSERT 的时候没有 OLD DELETE 的时候没有 NEW
触发器名称	创建 trigger 时触发器名称前可以添加模式名 CREATE OR REPLACE TRIGGER ” GJXFJ08” .”XFSXXX_PINDEX”	创建 trigger 时触发器名称前不加模式名，默认是和表一样的模式 CREATE OR REPLACE TRIGGER ” XFSXXX_PINDEX”

示例 1	CREATE OR REPLACE TRIGGER ”GJXFJ08” .”XFSXXX_PINDEX” AFTER INSERT ON ”GJXFJ08”.” XFSXXX” REFERENCING OLD AS old FOR EACH ROW ENABLE begin select xfsxxx_pindex_seq. nextval into :new.pindex from dual; end; /	CREATE OR REPLACE TRIGGER ”XFSXXX_PINDEX” AFTER INSERT ON ”GJXFJ08”.”XFSXXX” REFERENCING NEW TABLE AS new FOR EACH ROW Begin Select xfsxxx_pindex_seq. nextval into :new.pindex from dual; END; /

```
--用例准备：
Create table test1(deptno int, dept varchar2(10));
Create table test2(deptno int, dept varchar2(10));

--ORACLE 代码：
CREATE OR REPLACE TRIGGER TEST_TRG BEFORE INSERT ON TEST1 FOR EACH ROW
BEGIN
  INSERT INTO TEST2 VALUES(:NEW.deptno, :NEW.dept);
END;
/

--KingbaseES 改写方案：
\set SQLTERM /
CREATE OR REPLACE TRIGGER TEST_TRG BEFORE INSERT ON TEST1 FOR EACH ROW AS
BEGIN
  INSERT INTO TEST2 VALUES(NEW.deptno, NEW.dept);
END;
/
\set SQLTERM ;
```

2.2.13 消息输出

在 Oracle 移植中，Oracle 的输出语句“dbms_output.put_line...”KingbaseES 已经支持，另外，也可以用 KingbaseES 提供的“RAISE NOTICE...”替代。例如，Oracle 用户输出如下消息：

```
DBMS_OUTPUT.PUT_LINE('My name is John');
```

则在 KingbaseES 中可替换为：

```
RAISE NOTICE 'My name is John';
```

2.2.14 内置标量函数

在标量函数方面，KingbaseES 对 Oracle 提供了大量兼容性的支持，例如，KingbaseES 兼容 Oracle 的如下函数：

- EMPTY_BLOB 和 EMPTY_CLOB 函数
- NVL 和 NVL2 函数
- TRUNC 函数
- NUMERIC_AND、NUMERIC_OR 函数
- SYSDATE 函数
- TO_NUMBER 函数
- TO_CHAR 函数
- TO_DATE、TO_TIMESTAMP、TO_TIMESTAMPTZ 函数

依据用途不同，标量函数一般可分为：字符和字符串函数、类型转换函数、日期和时间函数、数学函数、序列函数、条件表达式函数和聚集函数等。下面，分别对上述各类函数进行 KingbaseES 和 Oracle 对比说明。

2.2.14.1 字符和字符串函数

字符和字符串函数主要用来处理字符和和字符串。在这方面，KingbaseES 和 Oracle 的对比如下表所示。

表 2.2.10: KingbaseES 和 Oracle 的字符和字符串函数对比表

函数	KingbaseES	Oracle
ASCII	返回参数部分的第一个字符的 ASCII 码，如： SELECT ASCII('abc') AS RESULT1,ASCII('ABC') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 97 65	支持
BIT_LENGTH	计算字符串的二进制长度，如： SELECT BIT_LENGTH('jose') AS RESULT; 输出结果如下所示： RESULT ------- 32	不支持
BTRIM	语法格式：BTRIM(expr1 TEXT[, expr2 TEXT])，表示从 expr1 开头和结尾删除 expr2 里（缺省是空白）的字符，如： SELECT BTRIM('xTomx', 'x') AS RESULT; 输出结果如下所示： RESULT ------- Tom	Oracle 与等效的是 LTRIM 和 RTRIM 函数一起使用，如： SQL>select LTRIM(RTRIM('xTomx', 'x'),'x') from dual;

continues on next page

表 2.2.10 – continued from previous page

CHAR_LENGTH	字符串的字符长度与 CHARACTER_LENGTH 和 LENGTH，语法格式:CHAR_LENGTH(expr TEXT)，如： SELECT CHAR_LENGTH('abc') AS RESULT1, CHAR_LENGTH('ABCDEF') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 3 6	Oracle 与其等效函数是 LENGTH，如： SQL> SELECT LENGTH('abc') FROM DUAL;
CHARACTER_LENGTH	字符串的字符长度与 CHAR_LENGTH 和 LENGTH，语法格式: CHARACTER_LENGTH(expr TEXT)，如： SELECT CH ARACTER_LENGTH('abc') AS RESULT1, CHARA CTER_LENGTH('ABCDEF') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 3 6	Oracle 与其等效函数是 LENGTH，如： SQL> SELECT LENGTH('abc') FROM DUAL;
CHR	返回给出 ASCII 码的字符，如： SELECT CHR(65) AS RESULT1, CHR(97) AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- A a	支持
CONCAT	SQL 标准规定了字符串的连接用操作符“‖”来表示。Oracle10g、DB2 V8R6 和 KingbaseES 可用函数 C ONCAT 或者操作符“‖”来进行字符串的连接。SQL Server2005 用操作符“+”来进行字符串的连接。如： SELECT CONCAT('ab', 'dad') AS RESULT; 输出结果如下所示： RESULT ---------- abdad	支持

continues on next page

表 2.2.10 – continued from previous page

CONVERT	语法格式：CONVERT(expr1 TEXT,[expr2 TEXT,] expr3 TEXT)，把字符串转换为 expr3 中声明的编码。原来的编码是用 expr2 声明的。如果省略了 expr2, 则假设为数据库编码。如： SELECT CONVERT ('text_in_unicode', 'UNICODE', 'LATIN1') AS RESULT; 输出结果是以 ISO8859-1 编码表示的 text_in_unicode 如下所示： RESULT ---------- text_in_unicode	支持
DECODING	把早先用 ENCODE 函数编码的，存放在 expr1 里面的二进制数据按照 expr2 中指定的类型解码，支持的类型有 base64，hex，escape。语法格式：DECODING(expr1 TEXT, expr2 TEXT)。如： SELECT DECODING('MTIzAAE=', 'base64') AS RESULT; 输出结果如下所示： RESULT ---------- \x3132330001 同 MYSQL 中的解码函数 DECODE()。	不支持
ENCODE	把二进制数据编码为只包含 ASCII 形式的数据。支持的类型有 base64，hex，escape。语法格式：ENCODE(expr1 BYTEA, expr2 TEXT)，如： SELECT ENCODE('3132330001', 'base64') AS RESULT; 输出结果如下所示： RESULT --------- MTIzAAE=	不支持
INITCAP	将每个单词的首字母转换为大写字母，语法分析：INITCAP(expr TEXT)，如： SELECT INITCAP('how are you') AS RESULT; 输出结果如下所示： RESULT -------------------- How Are You	支持

continues on next page

表 2.2.10 – continued from previous page

INSTR	语法格式：INSTR(expr1 TEXT, expr2 TEXT, [expr3 INTEGER[, expr4 INTEGER]])，在父字符串 expr1 中的第 expr3 个位置（从 1 开始）以字符为单位开始查找第 expr4 次出现的子字符串 expr2。如果 expr3 为负，则从 expr1 的倒数第 expr3 个位置往前开始查找，如： SELECT INSTR('abababc', 'abc') AS INSTR; 输出结果如下所示： INSTR ------ 5 SELECT INSTR('bcaaaaabbc', 'a', -2) AS INSTR; 输出结果如下所示： INSTR ------ 7	支持
INSTRB	语法格式：INSTRB(expr1 TEXT, expr2 TEXT, [expr3 INTEGER[, expr4 INTEGER]])，在父字符串 expr1 中的第 expr3 个位置（从 1 开始）以字节为单位开始查找第 expr4 次出现的子字符串 expr2。如果 expr3 为负，则从 expr1 的倒数第 expr3 个位置开始查找，如： SELECT INSTRB('abababc', 'abc') AS INSTRB; 输出结果如下所示： INSTRB ------- 5 SELECT INSTRB(' 人大金人大金仓人大金仓', ' 金仓', 3, 2) AS INSTRB; 输出结果如下所示： INSTRB ------- 28	支持

continues on next page

表 2.2.10 – continued from previous page

LCASE	将字符串中的大写字母转换为小写字母，格式：LCASE（expr TEXT)，如： SELECT LCASE('How Are You') AS RESULT1, LCASE('HOW ARE YOU') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- how are you how are you 等同于函数 LOWER(expr TEXT)。	Oracle 与 LOWER 函数等效。
LEFT	取字符串，取父字符串 expr1(第一个参数) 中的最左边的 expr2 (第二个参数) 个字符, 语法格式：LEFT(expr1 TEXT, expr2 INTEGER)，如： SELECT LEFT('abcdefg',2) AS RESULT; 输出结果如下所示： RESULT -------- ab SELECT LEFT(X'10010010',2) AS RESULT; 输出结果如下所示： RESULT -------- 00 标准 SQL2003 中没有 LEFT 函数。类似于 MSSQL 中的 LEFT 函数。	不支持，可以用 substr 替换，如：Sql>select substr（'abcdefg',1,2) from dual;
LENGTH	求字符串的长度, 等效于 CHAR_LENGTH(TEXT) 函数。	Oracle 10g 有此函数
LOWER	将字符串中的大写字母转换为小写字母，与 LCASE 函数等效，如： SELECT LOWER('How Are You') AS RESULT1, LOWER('HOW ARE YOU') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- how are you how are you	支持

continues on next page

表 2.2.10 – continued from previous page

LPAD	语法格式：LPAD(expr1 TEXT, expr2 INTEGER, expr3 TEXT)，用第三个参数给定的字符串从左边填充第一个参数直到第二个参数指定的长度为止，如： SELECT LPAD('abc',1,'#') AS RESULT1, LPAD('abc',2,'#') AS RESULT2, LPAD('abc',3,'#') AS RESULT3, LPAD('abc',4,'#') AS RESULT4, LPAD('abc',6,'#') AS RESULT5; 输出结果如下所示： RESULT1 RESULT2 RESULT3 RESULT4 RESULT5 ---------- a ab abc #abc ###abc	支持
LTRIM	语法格式：LTRIM(expr1 TEXT[, expr2 TEXT])，将字符串（expr1 第一个参数）最左边出现在指定字符集合（expr 2 第二个参数）的删除掉。在第二个参数缺省时，删除字符串左部开头的空格，等价于 Oracle 10g 的 LTRIM 函数。如： SELECT LTRIM('###abc','#') AS RESULT1, LTRIM('###abc','*') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- abc ###abc	支持
MD5	语法格式：MD5(expr)，计算给出 expr 的 MD5 散列，以十六进制串的形式返回结果，如： SELECT MD5('abc'); 输出结果如下所示： MD5 ---------- 900150983cd 24fb0d6963f7d28e17f72	不支持
OCTET_LENGTH	语法格式：OCTET_LENGTH(expr TEXT)，求字符串的字节长度。 注意：CHAR_LENGTH() 函数在处理汉字等多字节字符时，总是把每个汉字等当作一个字符。而 OCTEL _LENGTH 会将汉字等当作两个或多个字节来计算	Oracle 10g 的 LENGTHB 函数与之等价

continues on next page

表 2.2.10 – continued from previous page

OVERLAY	语法格式：OVERLAY(expr1 TEXT placing expr2 TEXT from expr3 integer [for expr4 integer])，替换子字符串。将被替换字符串（expr1）从指定位置（expr3）开始，指定长度（expr4）的子字符串，用字符串（expr2）替换。没有指定长度（expr4）时，替换跟字符串（expr2）相同长度的子字符串，如： SELECT OVERLAY('Txxxxas' PLACING 'hom' FROM 2 FOR 4); 输出结果如下所示： OVERLAY -------- Thomas	不支持，可以使用如下代替： SQL>select replace ('Txxxxas',substr('Txxxxas',2,4),'hom') from dual;
POSITION	语法格式：POSITION(expr1 TEXT IN expr2 TEXT)，求字符串（e xpr1）在字符串（expr2 ）中的起始位置，如果 e xpr1 不是 expr2 的子串，则返回结果值为 0，如： SELECT POSITION('r' in 'are') AS RESULT1, POSITION('x' in 'are') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 -------------------- 2 0	不支持，可以使用 instr 代替，如： Sql>select instr('are' ,' r' ,1,1) from dual;
QUOTE_IDENT	语法格式：QUOTE_IDENT(expr TEXT)， 返回字符串的适用于在 SQL 语句字符串里使用的形式。嵌入的引号被写了双份。如： SELECT QUOTE_IDENT('Foo bar'); 输出结果如下所示： QUOTE_IDENT ------------ "Foo bar" SELECT QUOTE_IDENT('" Foobar'); 输出结果如下所示： QUOTE_IDENT ------------ """"Foo bar"	不支持

continues on next page

表 2.2.10 – continued from previous page

REPEAT	语法格式：REPEAT(expr1 TEXT, expr2 INTEGER)，重复 TEXT（expr1）指定次数（expr2），如： SELECT REPEAT('123',3) AS RESULT; 输出结果如下所示： RESULT ---------- 123123123	不支持，可以使用如下替代： SQL>select lpad(‘123’,9,’123’) from dual;
REPLACE	目前不兼容 Oracle 的 REPLACE 函数。Oracle 表现形式：若 expr2 为 NULL，则返回 expr1；若 expr3 为 NULL，则从 expr1 中去除所有 expr2 子串。 KingbaeES 表现形式，若其中有一个参数为 NULL，返回 NULL。	支持
RIGHT	语法格式：RIGHT(expr1 TEXT, expr2 INTEGER)，取子字符串，取父字符串 expr1(第一个参数) 中的最右边的 expr2(第二个参数) 个字符，如： SELECT RIGHT('abcdefg',2) AS RESULT; 输出结果如下所示： RESULT -------- fg 标准 SQL2003 中没有 RIGHT 函数。类似于 MSSQL 中的 RIGHT 函数。	不支持, 可使用如下代替： Sql>select substr(‘abcdefg’,-2) from dual;
RPAD	语法格式：RPAD(expr1 TEXT, expr2 INTEGER[, expr3 TEXT])，用字符串 expr 3(第三个参数) 将字符串 expr1(第一个参数) 从右边填充到指定的长度 expr2(第二个参数)，在第三个参数缺省时，填充空格，如： SELECT RPAD('abc',1,'#') AS RESU LT1,RPAD('abc',2,'#') AS RESULT2, RPAD('abc',3,'#') AS RESULT3, RPAD('abc',4,'#') AS RESULT4, RPAD('abc',6,'#') AS RESULT5; 输出结果如下所示： RESULT1 RESULT2 RESULT3 RESULT4 RESULT5 -------------------- a ab abc abc# abc###	支持

continues on next page

表 2.2.10 – continued from previous page

RTRIM	语法格式：RTRIM(expr1 TEXT[, expr2 TEXT])，将字符串（expr1 第一个参数）尾部出现在指定字符集合（expr2 第二个参数）的删除掉。在第二个参数缺省时，删除字符串尾部的空格，如： SELECT RTRIM('abc###','#') AS RESULT1, RTRIM('abc###','*')AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- abc abc###	支持
SOUNDEX	KingbaseES 不支持	支持
STRPOS	语法格式：STRPOS(expr1 TEXT, expr2 TEXT)，求字符串（expr2）在字符串（e xpr1）中的位置，如果 e xpr2 不是 expr1 的子串，则返回结果值为 0，等效于 POSITION 函数，如： SELECT STRPOS('are','r') AS RESULT1, STRPOS('are','x') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 2 0	不支持，可使用 instr 代替，如： Sql>select instr(‘are’ ,’ r’ ,1,1) from dual;
SUBSTR	等同于函数 SUBSTRING(expr1, n [,m])。 注意参数支持负数的情况。	支持
SUBSTRING	语法格式：SUBSTRING(expr1 TEXT, [FROM] expr2 INTEGER[,[FOR] expr3 INTEGER])，取子字符串，在父字符串 expr1 (第一个参数) 中的第 expr2(第二个参数) 个位置开始取 expr3(第三个参数) 个字符，如果第三个参数缺省，则从第 expr2(第二个参数) 个位置开始取右面部分的全部，如： SELECT SUBSTRING ('abcdefg',2,3) AS RESULT1, SUBSTRING ('abcdefg',2) AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- bcd bcdefg	Oracle substr 代替

continues on next page

表 2.2.10 – continued from previous page

SUBSTRB	语法格式：SUBSTRB(expr1 TEXT, [FROM] expr2 INTEGER[,[FOR] expr3 INTEGER])，取子字符串，在父字符串 expr1 (第一个参数) 中的第 expr2(第二个参数) 个字节位置开始取 expr3(第三个参数) 个字节，如果第三个参数缺省，则从第 expr2(第二个参数) 个位置开始取右面部分的全部，如果第二个参数为负，则是从父字符串的尾部截取 expr3 个字节，如： SET CLIENT _ENCODING='GBK'; SELECT SUBSTRB('abcdefg',2,3) AS RESULT; 输出结果如下所示： RESULT -------- bcd (1 row) 相当于 Oracle 中的 SUBSTRB	支持
PL/SQLIT_PART	语法格式：SPLIT_PART(expr1 TEXT, expr2 TEXT, expr3 INTEGER)，根据 expr2 分隔 expr1 返回生成的第 expr3 个子串，如： SELECT SPLIT_PART ('abc~@~def~@~ghi' ,'~@~',2); 输出结果如下所示： SPLIT_PART ----------- def	不支持
TEXTCAT	同 concat 函数	同 concat 函数
TO_HEX	语法格式：TO_HEX(expr)，把 expr 转换成其对应的十六进制串形式，如： SELECT TO_HEX(9223372036854775807); 输出结果如下所示： TO_HEX ----------------- 7fffffffffffff	不支持

continues on next page

表 2.2.10 – continued from previous page

TRANSLATE	语法格式：TRANSLATE(expr TEXT, FROM TEXT, TO TEXT)，对于字符串 expr 中的每个字符，如果该字符出现在 FROM 中，将该字符替换为 TO 中相对应位置的字符。如果 FROM 长于 TO，FROM 中多余的字符将被删除。如果 FROM 短于 TO，TO 中多余的字符被忽略掉，如： SELECT TRANSLATE ('abcdefg','df','DF') AS RESULT1, TRANSLATE ('abcdefg','dg','DG') AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- abcDeFg abcDefG	支持
TRIM	语法格式：TRIM([LEADING | TRAILING | BOTH] [expr1] FROM expr2)，从字串 expr2 的开头 /结尾/两边, 删除包含在 expr1 中（缺省是一个空白）的所有字串，如： SELECT TRIM(LEADING 'x' FROM 'xTomx') AS RESULT1, TRIM(BOTH 'x' FROM 'xTomx') AS RESULT2, TRIM(TRAILING 'x' FROM 'xTomx') AS RESULT3; 输出结果如下所示： RESULT1 RESULT2 RESULT3 ---------- Tomx Tom xTom	Oracle RTRIM/LTRIM 达到同样效果
UCASE	等同于函数 UPPER(expr TEXT)。	等同于函数 UPPER(expr TEXT)
UNICODE	相当于 MSSQL 中的 UNICODE(expr)	不支持
UPPER	将字符串中的小写字母转换为大写字母。	支持
TO_MULTI_BYTE (Oracle)	KingbaseES 不支持	TO_MULTI_BYTE(String) 功能: 计算所有单字节字符都替换为等价的多字节字符的 String. 该函数只有当数据库字符集同时包含多字节和单字节的字符的时候有效. 否则, String 不会进行任何处理. TO_MULTI_BYTE 和 TO_SINGLE_BYTE 是相反的两个函数. SQL> select to_multi_byte ('xujianming') from dual; TO_MULTI_BYTE('XUJIA ---------- ｘｕｊｉａｎｍｉｎｇ

continues on next page

表 2.2.10 – continued from previous page

TO_SINGEL_BYTE (Oracle)	KingbaseES 不支持	TO_SINGLE_BYTE(String) 功能: 计算 String 中所有多字节字符都替换为等价的单字节字符. 该函数只有当数据库字符集同时包含多字节和单字节的字符的时候有效. 否则, String 不会进行任何处理. TO_MULTI_BYTE 和 TO_SINGLE_BYTE 是相反的两个函数. SQL> select to_single_byte ('xujianming') from dual; TO_SINGLE_ ---------- xujianming

2.2.14.2 类型转换函数

类型转换函数，通常又称格式化函数，用于把日期/时间、*INTEGER*、*FLOAT*、*NUMERIC* 等各种数据类型转换成格式化的字符串，以及反过来从格式化的字符串转换成指定的数据类型。在这方面，KingbaseES 和 Oracle 的对比如下表所示。

表 2.2.11: KingbaseES 和 Oracle 的类型转换函数对比表

函数	**KingbaseES**	**Oracle**
CAST	把源数据转换为目标类型的数据。	Oracle 10g 中都支持此函数
EXTREAT	其功能和 CAST 函数类似。	

TO_CHAR	1）TO_CHAR 也能将 clob 转换成 varchar2，比如： TEST=# select to_char (clob_import('f:test .txt','utf8')::clob); TO_CHAR --- Hello World+ 你好 + (1 行) TEST=# select to_char((select top 1 * from ctab)::clob); TO_CHAR --- abcd (1 行) 2）在将时间日期类型转换成字符串时，KingbaseES 需要在时间日期类型前指明 date、time、timestamp 。而 Oracle 则不需要。 3）TO_CHAR(DATE TIME) 格式化模板，日期时间转换模板基本上能兼容 Oracle 对应的模板格式。个别的除外：DL 、DS、TS、DY、RR、YEA R。其中 DL、DS、TS 的表现形式受 NLS_TERRITORY 和 NLS_LANGUAGE 影响。 4）TO_CHAR(NUMERIC) 格式化模板，Oracle 中的 $ 可以放在模板中的任意位置，但显示结果都会置于字符串的首位，而 KingbaseES 的数字模板中没有 $ 符号，会将 $ 原样输出。	Oracle 10g 有此函数。
TO_DATE	当 **ora_style_nls_** date_format = on 时，兼容 Oracle 的 TO_DATE 函数。在此种情况下，省略 expr2 参数时 ora_date_style 默认格式为 YYYY-MM-DD HH:MI:SS。	支持
TO_TIMESTAMP	把字符串转换成 TIMESTAMP WITHOUT TIME ZONE 型。	支持
TO_TIMESTAMP_TZ	把字符串转换成 TIEMSTAMP WITH TIME ZONE 型。	支持
TO_NUMBER	把字符串转换成数值型	支持

2.2.14.3 日期和时间函数

日期和时间函数主要用于获取和计算日期和时间的函数集合。在这方面，KingbaseES 和 Oracle 的对比如下表所示。

表 2.2.13: KingbaseES 和 Oracle 的日期和时间函数对比表

函数	KingbaseES	Oracle
AGE	语法格式：AGE([expr1 日期/时间类型,] expr2 日期/时间类型) 功能：参数 expr1 减去参数 expr2，生成一个两个日期/时间的间隔值。 参数说明：expr1 数据类型是日期/时间类型，包括 DATE、TIME、TIMETZ、TIMESTAMP 和 TIMESTAMPTZ。expr1 缺省时是 CURRENT_DATE。expr2 数据类型是日期/时间类型，包括 DATE、TIME、TIMETZ、TIMESTAMP 和 TIMESTAMPTZ。 注意，输入的日期/时间类型的 TEXT 需要用户显式指定类型，例如 CAST('2003-01-01' AS DATE)，CAST('22:10:01' AS TIME)，CAST('1981-10-19' AS TIMESTAMP) 等等。 返回值说明：返回值的数据类型为 INTERVAL 型。	Oracle 中没有 age 函数，类似的函数 months_between() 判断两个日期之间的月份数量，如果需要计算年龄需要 months_between ()/ 12, 例如：SELECT TRUNC(MONTHS_BETWEEN(SYSDATE, to_date(birthday ,'yyyy-mm-dd')/12,0) FROM dual;
CLOCK_TIMESTAMP	语法格式：CLOCK_TIMESTAMP() 功能：返回当前的日期和时间。 参数说明：函数无参数。 返回值说明：返回值的数据类型为 TIMESTAMPTZ 型。	Oracle 中没有此函数，类似函数 current_timestamp() 返回当前会话时区中的当前日期。
CURRENT TIMESTAMP	等同于 CURRENT_TIMESTAMP 不支持 CURRENT TIMESTAMP	Oracle 10g 有此函数。
CURRENT_DATE	当 ora_style_nls_ date_format = on 时，且 ora_date_style 值为 YYYY-MM-DD 与 Oracle 兼容，获取当前（而不是事务 开始时）的系统日期。	支持 Oracle 中获取当前系统时间是 select sysdate from dual; 或者 Select systimestamp from dual;
CURRENT_TIME	获取当前（而不是事务开始时）系统的时间 (机器时间)。	Oracle 中获取当前系统时间是 select sysdate from dual; 或者 Select systimestamp from dual;
CURRENT_TIMESTAMP	与 Oracle 兼容，获取当前（而不是事务开始时）系统的日期和时间 (机器时间)。	Oracle 10g 有此函数。

DATEADD	KingbaseES 不支持	在 Oracle 10g 中有此函数，但形式上和函数语义上有较大差别。 orcale 中一般使用 add_months 例如： to_char(add_mont hs(to_date('199912',' yyyymm'),2),'yyyymm') TO_CHA ------ 200002 或者是直接用, 时间 +N 来得到 例如： to_char((sys date) +1,'yyyy.mm.dd') 系统时间 +1 天
DATEDIFF	KingbaseES 不支持	在 Oracle 10g 中有此函数，但形式上和函数语义上有较大差别。
DATEPART	KingbaseES 不支持	在 Oracle 10g 有此函数。
EXTRACT	与 SQL 标准中对 EXTPACT 的各个域定义一致。	在 Oracle 10g 有此函数。
DATE_TRUNC	DATE_TRUNC(expr1 TEXT, expr2 日期/时间类型)，功能：截取日期成指定的精度 (精度由第一个参数指定)。	Oracle 中与此功能相同的函数为 TRUNC()。
DATE_FORMAT	KingbaseES 不支持	Oracle 没有此函数，类似的函数 TO_DATE，起功能是把字符串转换为数据库中的日期类型。
ISFINITE	功能：测试指定时间是否为有效时间。	Oracle 没有此函数，也没有此功能函数。
LOCALTIME	KingbaseES 支持该函数，但其和 KingbaseES 的 current_date 功能相似。与 Oracle 兼容，获取当前（而不是事务开始时）系统的日期（机器时间）。	Oracle 中获取当前系统时间是 select sysdate from dual;
LOCALTIMESTAMP	KingbaseES 支持该函数，其和 KingbaseES 的 current_timestamp 功能相似。 与 Oracle 兼容，获取当前（而不是事务开始时）系统的日期和时间（机器时间）。	Oracle 中获取当前系统时间是 Select systimestamp from dual;
NOW	返回当前事务开始的日期和时间，等效于 CURRENT_TIMESTAMP。 获取当前（而不是事务开始时）系统的日期和时间 (机器时间)	Oracle 10g CURRENT_TIMESTAMP 函数与之对应。
STATEMENT_TIMESTAMP	返回当前（批处理）语句开始执行的日期和时间。	Oracle 没有此函数，也没有此功能函数。
STR_VALID	相当于 MSSQL 中的 ISDATE(expr), 判断所给定的表达式是否为正确的日期时间	Oracle 没有此函数，也没有此功能函数。
SYSDATE	在 KingbaseES 中，默认情况下 SYSDATE 与 CURRENT_DATE 功能相同。与 Oracle SYSDATE 函数具有相同功能（获取当前时间，而不是事务开始时间）。	Oracle 中获取当前系统时间是 select sysdate from dual;

SYSTIMESTAMP	KingbaseES 支持该函数，该函数功能和 KingbaseES 的 current_timestamp 功能相似。与 Oracle 兼容，获取当前（而不是事务开始时）系统的时间 (机器时间)。	Oracle 中获取当前系统时间是 Select systimestamp from dual;
TIMEOFDAY	返回当前的高精度日期和时间。	Oracle 没有此函数，也没有此功能函数。
TRANSACTION_TIMESTAMP	返回当前事务开始的日期和时间。	Oracle 没有此函数，也没有此功能函数。

2.2.14.4 数学函数

数学函数是用于数学计算的函数集合。通常，它的函数名给出计算方式。在这方面，KingbaseES 和 Oracle 的对比如下表所示。

表 2.2.15: KingbaseES 和 Oracle 的数学函数对比表

函数	**KingbaseES**	**Oracle**
ABS	返回指定值的绝对值，如：SELECT ABS(-17.6) AS NUM1, ABS(17.6) AS NUM2; 输出结果如下所示： NUM1 NUM2 ---------- 17.6 17.6	支持
ACOS	计算反余弦，如： SELECT ACOS(0.1) AS RESULT; 输出结果如下所示： RESULT ---------- 1.470628905633337	支持
ASIN	计算反正弦，如： SELECT ASIN(0.1) AS RESULT; 输出结果如下所示： RESULT ---------- 0.1001674211615598	支持
ATAN	返回一个数字的反正切值，如： SELECT ATAN(0.1) AS RESULT; 输出结果如下所示： RESULT ---------- 0.09966865249116204	支持
ATAN2	计算 expr1 除以 expr2 所得结果的反正切，如： SELECT ATAN2(0.2,0.1) AS RESULT1,ATAN(2) AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 1.10714871779409 1.10714871779409	支持

BITAND	计算 expr1 和 expr2 的位与，如： SELECT BITAND(1, 2); 输出结果如下所示： BITAND ------- 0	支持
CBRT	计算立方根，如： SELECT CBRT(27) AS RESULT; 输出结果如下所示： RESULT --------- 3	不支持
CEIL	计算不小于参数的最小整数。	支持
CEILING	计算不小于参数的最小整数 SELECT CEILING(27.38) AS RESULT1, CEILING(-27.38) AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 28 -27	Oracle 10g 中没有 CEILING 函数，但有 CEIL 函数于其等价。 SQL> select ceil(3.1415927) from dual; CEIL(3.1415927) ---------- 4
COS	计算余弦，如： SELECT COS(0.1) AS RESULT; 输出结果如下所示： RESULT ---------- 0.9950041652780258	支持
COT	计算余切 SELECT COT(0.1) AS RESULT; 输出结果如下所示： RESULT ---------- 9.966644423259238	Oracle 没有余切函数，可以通过数学公式计算于其等效： SQL>select 1/tan(0.1) from dual;
DEGREES	将弧度转换为角度 SELECT DEGREES(0.5) AS RESULT; 输出结果如下所示： RESULT --------- 28.6478897565412	不支持，可以通过计算公式得到： SQL>select 0.5*180/3.14 from dual;
EXP	返回一个数字 e 的 n 次方根, 如： SELECT EXP(1) AS RESULT; 输出结果如下所示： RESULT --------- 2.71828182845905	支持
FLOOR	计算不大于参数的最大整数, 如： SELECT FLOOR(27.36) AS RESULT1, FLOOR(-27.36) AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 27 -28	支持

LOG	LOG(expr) 用来计算以 10 为底的对数。 LOG(x, y) 用来计算以 x 为底，y 的对数。 DLOG10(expr) 用来计算以 10 为底的对数。 SELECT LOG(2.0,64.0) AS RESULT1, LOG(2,64) AS RESULT2; 输出结果如下所示： RESULT1 RESULT2 ---------- 6.0000000000000000 6.0000000000000000	支持 LOG(x, y)，但 不支持 LOG(expr) 和 DLOG10(expr)。
MOD	计算除法 x/y 的余数，如： SELECT MOD(8,3) AS RESULT1, 输出结果如下所示： RESULT1 -------- 2	支持
PI	计算圆周率常量，如： SELECT PI() AS RESULT; 输出结果如下所示： RESULT --------- 3.14159265358979	不支持
POWER	计算 e 的 n 次幂，如： SELECT POWER(3, 2) AS RESULT; 输出结果如下所示： RESULT --------- 9	支持
RADIANS	把角度转换为弧度 SELECT RADIANS(45.0) AS RESULT; 输出结果如下所示： RESULT --------- 0.785398163397448	不支持，可以通过计算公式得到： SQL>select 45*3.14/180 from dual;
RANDOM	计算 0.0 到 1.0 之间的随机数 SELECT RANDOM() AS RESULT; 输出结果如下所示： RESULT ---------- （0 .0 到 1.0 之间的随机数）	Oracle 利用 DBMS_RANDOM 计算随机数，如：产生 N 到 M 之间的随机数 SELECT DBMS_RANDOM .VALUE(N,M) FROM DUAL;
ROUND	ROUND(x, y)，将 x 对小数点后 y 位小数四舍五入计算最终数，y 值可以省略，y 的缺省值为 0，如： SELECT ROUND(32.19) AS RESULT1, ROUND(32.69) AS RESULT2, ROUND(-32.19) AS RESULT3, ROUND(-32.69) AS RESULT4; 输出结果如下所示： RESULT1 RESULT2 RESULT3 RESULT4 ---------- 32 33 -32 -33	支持

SETSEED	为随后的 RANDOM() 调用设置种子 SELECT SETSEED(0.54823) AS RESULT; 输出结果如下所示： RESULT ----------	不支持
SIGN	返回参数的符号（-1, 0, +1），大于 0 返回 1, 小于 0 返回-1, 等于 0 返回 0，如： SELECT SIGN(-8.4) AS RESULT; 输出结果如下所示： RESULT ------- -1	支持
SIN	计算正弦，如： SELECT SIN(0.1) AS RESULT; 输出结果如下所示： RESULT ------------------ 0.0998334166468282	支持
SQRT	DSQRT(expr) 计算平方根，如： SELECT SQRT(CAST(4 AS DOUBLE)) AS RESULT; 输出结果如下所示： RESULT ------- 2	支持
TAN	计算正切，如： SELECT TAN(0.1) AS RESULT; 输出结果如下所示： RESULT ------------------ 0.100334672085451	支持
TRUNC	数值截断，如： SELECT TRUNC(42.8) AS RESULT1, TRUNC(-42.8) AS RESULT2, TRUNC(-42.1) AS RESULT3; 输出结果如下所示： RESULT1 RESULT2 RESULT3 -------------- 42 -42 -42	支持

2.2.14.5 序列函数

序列函数是用来实施各种序列操作的函数集合。在这方面，KingbaseES 和 Oracle 的对比如下表所示。

表 2.2.17: KingbaseES 和 Oracle 的序列函数对比表

函数	KingbaseES	Oracle
CURRVAL	返回序列的当前数值。	支持

NEXTVAL	在默认的情况下，在一个 SQL 语句中多次使用 NEXTVAL 获取的值是递增的。 设置参数 ora_func_style=true 之后，如果在一个 SQL 语句中多次使用 CURRVAL 或者 NEXTVAL 获取相同的序列值。如果使用 NEXTVAL 的 UPDATE 语句影响到多行，同一行里的数据值相同，不同行里的数据递增。	支持
SETVAL	设置序列的当前数值以及 is_called 标志。	不支持
LASTVAL	返回当前会话上次使用的序列的当前值。如果本次会话没有序列通过调用 nextval 得到新值，调用该函数将会报错。	不支持
INCREMENTVAL	返回指定序列的步长。	不支持
SEEDVAL	返回指定序列的开始值。	不支持

Note: 序列函数 NEXTVAL 和 CURRVAL 还支持通过 Database link 进行远程调用。调用格式如下：

```
sequencename.nextval@dblink_name
sequencename.currval@dblink_name
```

2.2.14.6 条件表达式函数

条件表达式函数是用来实现条件表达式的函数集合。在这方面，KingbaseES 和 Oracle 的对比如下表所示。

表 2.2.19: KingbaseES 和 Oracle 的条件表达式函数对比表

函数	**KingbaseES**	**Oracle**
CASE	类似于编程语言中的 if-then-else 结构，CASE 是 SQL 中的条件表达式。CASE 可以用于任何可以出现表达式的地方。	类似于编程语言中的 if-then-else 结构，CASE 是 SQL 中的条件表达式。CASE 可以用于任何可以出现表达式的地方。
DECODE	KingbaseES 对参数个数无限制。 KingbaseES 在得到返回值之前对各个参数类型都将进行类型转化检查。	Oracle 中允许的 DECODE 函数的参数个数（包括 Expression, Search, Result, Default）不能超过 255 个。Oracle 中 DECODE 函数以第一个 Search 值的数据类型为标准，只把遇到匹配的 Search 值之前（包括该 Search 值）的 Expression 值和 Search 值，进行类型转化检查，并且只把返回值转化为第一个 Result 值的数据类型。
COALESCE	返回参数中第一个不是 NULL 的值。如果所有参数的值都是 NULL，则返回 NULL。	COALESCE 函数是 NVL 函数的扩展，但是 COALESCE 函数可以使用多个值，联合救市该函数要执行的操作；返回参数中第一个不是 NULL 的值。如果所有参数的值都是 NULL，则返回 NULL。
ISNULL	ISNULL(expr1 任意类型, expr2 任意类型)，当 expr1 为 NULL 时，用 expr2 代替本函数式的值；否则本函数的值保持 expr1 的原值。	Oracle 中的 NVL 函数功能与 KingbaseES 中的 ISNULL 功能相当。
ISNUMERIC	KingbaseES 不支持该表达式函数	在 Oracle 中没有现成的判断是否为数字函数，可以用如下三种方法来实现： 1. 利用 to_number。 2. 利用 regexp_like。 3. 利用 TRANSLATE。

IF	KingbaseES 不支持该表达式函数	Oracle 的 sql 里面是没有直接的 if else 语句的，可以用 decode 函数或者 case when 简单的代替。比如 selcect decode(表 1 的列 1，等于 a，结果 a，等于 b，结果 b，其他结果 c) from 表 1。
IFNULL	等同于函数 ISNULL(expr1 任意类型, expr2 任意类型)。	Oracle 中的 NVL 函数功能与 KingbaseES 中的 ISNULL 功能相当。
NULLIF	NULLIF(Value1, Value2)：如果 Value1 的值和 Value2 的值相等，则返回 NULL。否则返回 Value1。	Oracle 中 nullif 函数功能与 KingbaseES 相当，如果 Value1 的值和 Value2 的值相等，则返回 NULL。否则返回 Value1。
NVL	相当于 KingbaseES 中的 ISNULL 函数、Oracle 中的 NVL 函数。	Oracle 中此函数。
NVL2	KingbaseES 中的 IF 函数、Oracle 中的 NVL2 函数。	Oracle 中有此函数。
GREATEST	返回数值型列表中的最大值	Oracle 中有此函数，功能是取得值最大值。影响版本: Oracle 8i, Oracle 9i, Oracle 10g, Oracle 11g
LEAST	返回数值型列表中的最小值	Oracle 中有此函数，功能是取得值最小值。影响版本: Oracle 8i, Oracle 9i, Oracle 10g, Oracle 11g

2.2.14.7 聚集函数

聚集函数通过对一组输入值计算获得一个结果值，其函数名标明计算方式。聚集函数通常在查询语句中与子句 *GROUP BY* 或 *HAVING* 联合使用。在这方面，KingbaseES 和 Oracle 的对比如下表所示。

表 2.2.21: KingbaseES 和 Oracle 的聚集函数对比表

函数	**KingbaseES**	**Oracle**
AVG	计算所有输入值的均值（算术平均值），略去空值（字段值为 NULL 的）。	Oracle 中的 AVG 函数是对一个数字列或计算列求取的平均值。
COUNT	计数功能	返回记录的统计数量
MAX	求最大值，不考虑空值（字段值为 NULL 的）。	返回一个数字列或者计算列中的最大值
MIN	求最小值，不考虑空值（字段值为 NULL 的）。	返回一个数字列或者计算列中的最小值
STDDEV	输入值的标准采样偏差。	Oracle 中 StdDev 函数返回 expr 的样本标准偏差。它可用作聚集和分析函数。
SUM	求和，略去空值 (字段值为 NULL 的)。	返回一个数字列或者计算列中的汇总和
VARIANCE	输出值的采样方差（标准采样偏差的平方）。	Oracle 中 VARIANCE 函数是用于计算 x 的方差。方差是一个统计函数，其定义为一组样本数据的偏离程度，等于标准差的平方。

2.2.15 其他

2.2.15.1 pipeline 和 pipe row

KingbaseES 已支持兼容 ORACLE 的 pipelined 表函数，另外也可以将 pipelined 表函数通过 KingbaseES 的 return next 和函数返回值 setof 返回集合的方式改写。

```
--ORACLE 示例：
CREATE TYPE mytype AS OBJECT (
field1 NUMBER,
field2 VARCHAR2 (50)
);
CREATE TYPE mytypelist AS TABLE OF mytype;
CREATE OR REPLACE FUNCTION pipelineme
RETURN mytypelist PIPELINED IS
v_mytype mytype;
BEGIN
FOR v_count IN 1 .. 20
LOOP
v_mytype := mytype (v_count, 'Row ' || v_count);
PIPE ROW (v_mytype); END LOOP;
RETURN;
END pipelineme;

--KingbaseES 的改写方案：
CREATE TYPE mytype AS (
field1 NUMBER,
field2 VARCHAR2 (50)
);

CREATE OR REPLACE INTERNAL FUNCTION testf()
  RETURNS SETOF mytype AS
$BODY$
DECLARE
    v_mytype mytype;
BEGIN
FOR v_count IN 1 .. 20
LOOP
    v_mytype := (v_count, 'Row ' || v_count);
    return next v_mytype;
END LOOP;
RETURN;
END;
$BODY$
  LANGUAGE plsql;
```

2.3 SQL 语句兼容特性

对于大多数常用的 Oracle SQL 语句，KingbaseES 均提供了原生支持。该措施使得 Oracle 应用程序移植到 KingbaseES 系统时，通常只需很少的代码变动。

下面给出 KingbaseES 中原生支持的 Oracle SQL 语句。此外，若未做特殊说明，本节示例的代码在 KingbaseES 和 Oracle 上均可运行。

2.3.1 Truncate 语句

TRUNCATE 功能是删除整表数据，但它不能删除表结构。和 *TRUNCATE* 相比，*Drop table* 不仅能删除表结构，还能删除整表数据。*Delete* 虽然不能删除表结构，但它不仅能删除整表数据，也能有选择地删除表的部分记录。

值得注意的是，与 *Delete* 相比，*TRUNCATE* 数据不能回滚。KingbaseES、Oracle、Sybase 和 Microsoft SQL Server 都采用了这种处理方式。

TRUNCATE 的语法如下所示：

```
TRUNCATE [TABLE] [SchemaName.]TableName [ , ... ] [ CASCADE | RESTRICT ]
```

例 2-37：*TRUNCATE* 的示例。

```
CREATE TABLE trunct1(c1 int);
INSERT INTO truncT1 VALUES (1), (2), (3);
SELECT * FROM truncT1;
C1
-----------
1
2
3
3 record(s) selected.
TRUNCATE truncT1;
SELECT * FROM truncT1;
C1
-----------
0 record(s) selected.
```

2.3.2 层次查询

层次查询是一种特定类型的查询，用于在基于父子关系的数据中以层次顺序返回结果集中的记录。通常，层次是用一个反转顺序的树结构表示。树由相互连接的节点组成。每个节点可能会连接 0 个或多个子节点。在层次查询中，结果集的记录为一或多棵树中的节点。

KingbaseES 和 Oracle 均支持层次查询，且二者兼容。

下表列出了层次查询中的所有操作符：

表 2.3.1: 层次查询的操作符列表

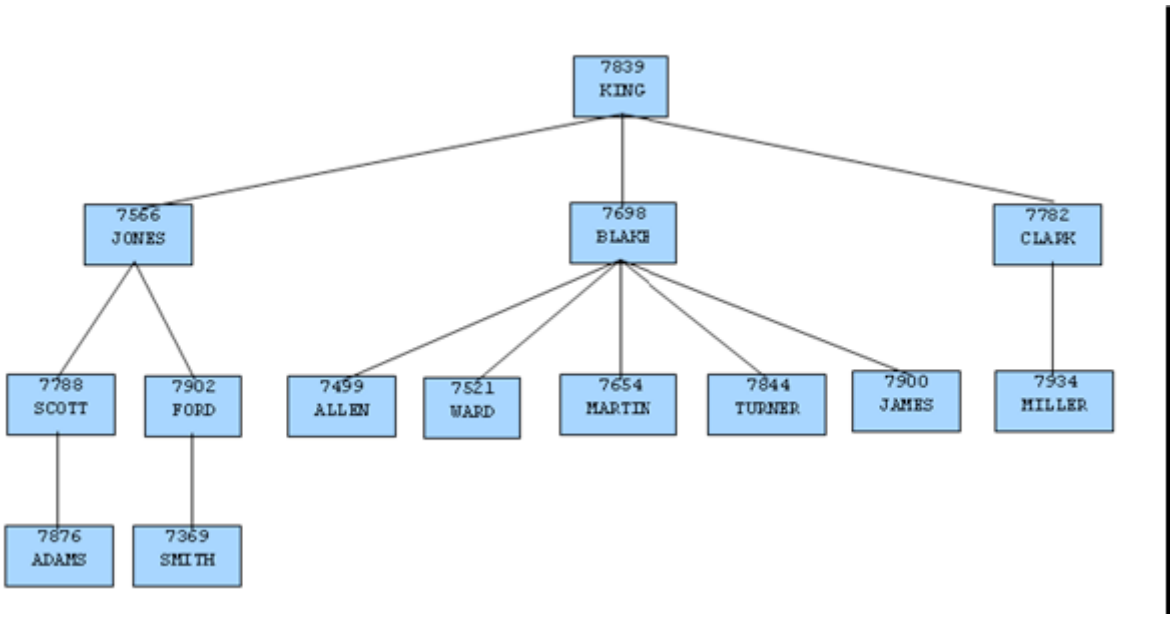
操作符	功能
PRIOR	在当前元组的父元组上求值，如果当前元组是根（Root）元组，则求值为 NULL。
CONNECT_BY_ROOT	在当前层次查询的根（Root）元组上求值。

伪列与普通列十分相似，但其值并不是和表数据存储在一起的。层次查询具有三个伪列，具体如下表所示。

表 2.3.2: 层次查询的伪列列表

伪列	描述
LEVEL 伪列	描述当前元组的所在的层
CONNECT_BY_ISLEAF 伪列	描述当前节点是否为叶节点。若是为 1，否则为 0。
CONNECT_BY_ISCYCLE 伪列	如果一个元组的 CONNECT_BY_ISCYCLE 值是 1，则代表这个元组有子元组，并且这个子元组又是它的祖先元组，即数据库中的数据成环；否则为 0。

例 2-38：层次查询应用的示例。在该例中，表 *emp* 表数据的树状结构图如下所示：



执行以下附带层次查询的 *SELECT* 命令：

```
SELECT ename, empno, mgr
FROM emp
START WITH mgr IS NULL
CONNECT BY PRIOR empno = mgr;
```

运行结果如下所示：

```
  ENAME    | EMPNO | MGR
-----------+-------+------
 KING      |  7839 |
 JONES     |  7566 | 7839
 SCOTT     |  7788 | 7566
 ADAMS     |  7876 | 7788
 FORD      |  7902 | 7566
 SMITH     |  7369 | 7902
 ANDERSON  |  8142 | 7902
 BLAKE     |  7698 | 7839
 ALLEN     |  7499 | 7698
 WARD      |  7521 | 7698
```

(continues on next page)

(continued from previous page)

```
 MARTIN   |  7654 | 7698
 TURNER   |  7844 | 7698
 JAMES    |  7900 | 7698
 CLARK    |  7782 | 7839
 MILLER   |  7934 | 7782
(15 行)
```

例 2-41：层次查询伪列的示例。通过对伪列 *LEVEL* 值执行 *LPAD* 操作，雇员名称被缩进，这样能够进一步强调每条记录在层次中的深度。

```
SELECT LEVEL, LPAD (' ', 2 * (LEVEL - 1)) || ename "employee", empno, mgr
FROM emp START WITH mgr IS NULL
CONNECT BY PRIOR empno = mgr;
```

这个查询的检索结果如下所示：

```
 LEVEL |    employee    | EMPNO | MGR
-------+----------------+-------+------
     1 | KING           |  7839 |
     2 |   JONES        |  7566 | 7839
     3 |     SCOTT      |  7788 | 7566
     4 |       ADAMS    |  7876 | 7788
     3 |     FORD       |  7902 | 7566
     4 |       SMITH    |  7369 | 7902
     4 |       ANDERSON |  8142 | 7902
     2 |   BLAKE        |  7698 | 7839
     3 |     ALLEN      |  7499 | 7698
     3 |     WARD       |  7521 | 7698
     3 |     MARTIN     |  7654 | 7698
     3 |     TURNER     |  7844 | 7698
     3 |     JAMES      |  7900 | 7698
     2 |   CLARK        |  7782 | 7839
     3 |     MILLER     |  7934 | 7782
(15 行)
```

例 2-39：层次查询对同层节点排序的示例。层次查询语句如下所示：

```
SELECT LEVEL, LPAD (' ', 2 * (LEVEL - 1)) || ename "employee", empno, mgr
FROM emp START WITH mgr IS NULL
CONNECT BY PRIOR empno = mgr
ORDER SIBLINGS BY ename ASC;
```

这个查询的检索结果如下所示：

```
 LEVEL |    employee    | EMPNO | MGR
-------+----------------+-------+------
     1 | KING           |  7839 |
     2 |   BLAKE        |  7698 | 7839
     3 |     ALLEN      |  7499 | 7698
     3 |     JAMES      |  7900 | 7698
     3 |     MARTIN     |  7654 | 7698
     3 |     TURNER     |  7844 | 7698
     3 |     WARD       |  7521 | 7698
     2 |   CLARK        |  7782 | 7839
     3 |     MILLER     |  7934 | 7782
     2 |   JONES        |  7566 | 7839
```

(continues on next page)

(continued from previous page)

```
     3 |      FORD       |  7902 | 7566
     4 |        ANDERSON |  8142 | 7902
     4 |        SMITH    |  7369 | 7902
     3 |      SCOTT      |  7788 | 7566
     4 |        ADAMS    |  7876 | 7788
(15 行)
```

例 2-40：层次查询的 *WHERE* 过滤的示例，查询语句如下所示：

```
SELECT LEVEL, LPAD (' ', 2 * (LEVEL - 1)) || ename "employee", empno, mgr
FROM emp WHERE mgr IN (7839, 7782, 7902, 7788)
START WITH ename IN ('BLAKE','CLARK','JONES')
CONNECT BY PRIOR empno = mgr
ORDER SIBLINGS BY ename ASC;
```

这个查询的检索结果如下，在下面的结果中不满足 *WHERE* 子句的记录不会在输出中出现。

```
 LEVEL |   employee    | EMPNO | MGR
-------+---------------+-------+------
     1 | BLAKE         |  7698 | 7839
     1 | CLARK         |  7782 | 7839
     2 |   MILLER      |  7934 | 7782
     1 | JONES         |  7566 | 7839
     3 |     ANDERSON  |  8142 | 7902
     3 |     SMITH     |  7369 | 7902
     3 |     ADAMS     |  7876 | 7788
```

(7 行)

2.3.3 ROWNUM 伪列

*ROWNUM** 伪列返回一个数值指示该条记录从表读取或连接产生时的顺序。第一条被选择的记录的 **ROWNUM* 值是 1，第二条的值是 2，以此类推。可使用 *ROWNUM* 限制查询返回的记录数量。

Kingbase 和 Oracle 在 *ROWNUM* 伪列上完全兼容。

例 2-41：*ROWNUM* 伪列在 *select* 中的应用。

```
SELECT * FROM STUDENT WHERE ROWNUM < 10 ;
```

例 2-42：*ROWNUM* 伪列在 *delete* 和 *update* 中的应用。

```
UPDATE STUDENT
SET SID = 5
WHERE SID = 6
AND ROWNUM <=3;

DELETE FROM STUDENT
WHERE SID = 5
AND ROWNUM <= 1;
```

2.3.4 ROWID 伪列

KingbaseES 虽然从语法上没有支持 Oracle 的 *ROWID* 伪列，但 KingbaseES 的 *OID* 伪列提供了和 *ROWID* 伪列相同的功能。KingbaseES 中提供了一个会话级开关参数，即 *default_with_oids*，它为 *on* 表示新建表具有 *OID* 伪列，否则新建表无 *OID* 伪列。此外，在 *CREATE TABLE* 创建表时，用户也可通过 *WITHOUT OIDS* 或 *WITH OIDS* 选项设定新建表是否包含 *OID* 伪列。

虽然 *ROWID* 与 *ROWNUM* 均属于伪列，但二者的用途是不同的：前者用来唯一标识数据库对象的每条记录，而后者用来标识检索结果集的每条记录。

2.3.5 外连接操作符 (’+’)

Oracle 和 KingbaseES 均支持三类外部联接的 ANSI SQL 语法，即右外连接，左外连接和全外连接。此外，KingbaseES 还兼容 Oracle 的外连接语法 (+)。通常，(+) 可用于一些简单的外连接，但不建议使用 (+) 做多表的复杂外连接操作。

例 2-43：*WHERE* 子句的连接条件中使用外连接操作符 (+) 的示例。

```
--创建表
CREATE TABLE emp (
  empno NUMBER(4),
  ename VARCHAR2(10),
  job VARCHAR2(9),
  mgr NUMBER(4),
  hiredate DATE,
  sal NUMBER(7,2),
  comm NUMBER(7,2),
  deptno NUMBER(2)
);
CREATE TABLE dept (
      Deptno    int NOT NULL,
      Dname     text,
      Loc       text,
      Mgr_no    int,
      Dept_type int);
--插入数据
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'17-DEC-80',800,NULL,20);
INSERT INTO emp(empno,ename,job,mgr,hiredate,sal,comm,deptno)
VALUES (7499,'ALLEN','SALESMAN',7698,'20-FEB-81',1600,300,30);
INSERT INTO emp(empno,ename,job,mgr,hiredate,sal,deptno)
VALUES (7369,'SMITH','CLERK',7902,'17-DEC-80',800,20);

INSERT INTO dept VALUES (50,'FINANCE','CHICAGO');
INSERT INTO emp (empno,ename,deptno) VALUES (9001,'JONES',50);
INSERT INTO emp (empno,ename,deptno) VALUES (9002,'ALICE',50);

--外连接
SELECT emp.ename, emp.sal, dept.deptno, dept.dname, dept.loc FROM dept, emp WHERE
emp.deptno(+) = dept.deptno;
TEST-# emp.deptno(+) = dept.deptno;
 ENAME | SAL | DEPTNO |  DNAME  |   LOC
-------+-----+--------+---------+---------
 JONES |     |     50 | FINANCE | CHICAGO
 ALICE |     |     50 | FINANCE | CHICAGO
(2 行)
```

2.3.6 DUAL 伪表

Oracle 提供“*dummy*”表，该表称为 *DUAL* 伪表。通常用它来检索系统信息。KingbaseES 兼容 Oracle 的 *DUAL* 伪表。该表可被所有的用户访问，但只有数据库管理员才能删除、增加、修改 *DUAL* 表的内容。

例 2-44：伪表 *DUAL* 的示例。

```
SELECT SYSDATE AS CURRENT_DATE_TIME FROM DUAL;
```

2.3.7 SELECT INTO 的 FOR UPDATE 子句

FOR UPDATE 子句的作用是锁住特定 *SELECT* 检索结果记录，用来避免这些记录在当前事务结束前被其它事务修改或者删除。也就是说，其它企图对这些记录进行 *UPDATE* 、*DELETE* 或 *SELECT FOR UPDATE* 操作的事务将被阻塞，直到当前事务结束。此外，为了避免该操作等待其它事务提交，该操作还提供了 *NOWAIT* 选项。

Oracle 和 KingbaseES 均支持 *SELECT INTO* 的 *FOR UPDATE* 子句，但二者的差异是：KingbaseES 的 *FOR UPDATE* 是表级粒度，而 Oracle 是字段级粒度。

例 2-45：*SELECT INTO* 的 *FOR UPDATE* 子句的示例。

```
--KingbaseES 代码
DECLARE
  v_sid student.sid%TYPE;
  v_sname student.sname%TYPE;
  new_sname student.sname%TYPE;
BEGIN
  v_sid := 3;
  new_sname := 'NEW_NAME';
  SELECT sname INTO v_sname FROM student
  WHERE sid = v_sid FOR UPDATE OF student; //表级粒度
  UPDATE student SET sname = new_sname
  WHERE sid = v_sid;
END;

--Oracle 代码
DECLARE
  v_sid student.sid%TYPE;
  v_sname student.sname%TYPE;
  new_sname student.sname%TYPE;
BEGIN
  v_sid := 3;
  new_sname := 'NEW_NAME';
  SELECT sname INTO v_sname FROM student
  WHERE sid = v_sid FOR UPDATE OF sname;//字段级粒度
  UPDATE student SET sname = new_sname
  WHERE sid = v_sid;
END;
```

2.3.8 UPDATE[前缀] 多列更新

为兼容 Oracle，KingbaseES 提供 *UPDATE* 的多列更新功能，相关语法如下所示：

```
UPDATE [SchemaName.]{ TableName | ViewName }
  SET { [Catalog.SchemaName.TableName. | SchemaName.TableName. | TableName.
↪]ColumnName  =  { <Expression> | DEFAULT }} [, ...n]  [ WHERE <ConditionExpression>]
[ RETURNING { * | output_expression [ AS output_name ] [, ...n] } [ INTO placeholder␣
↪[, ...n] ]
```

例 2-46：*UPDATE* 多列更新的示例。该例在 KingbaseES 和 Oracle 中均可执行。

```
CREATE TABLE a (id INTEGER, f1 VARCHAR(30), f2 VARCHAR(30),f3 VARCHAR(30));
INSERT INTO a VALUES(1, 'Tom1', 'Tom2', 'Tom3');
INSERT INTO a VALUES(2, 'John1', 'John2','John3');

CREATE TABLE b (id INTEGER, f1 VARCHAR(30), f2 VARCHAR(30),f3 VARCHAR(30));
INSERT INTO b VALUES(1, 'Tom1b', 'Tom2b', 'Tom3b');
INSERT INTO b VALUES(2, 'John1b', 'John2b','John3b');
```

多列更新：

```
Update a Set (a.f1,a.f2,a.f3)=(select b.f1,b.f2,b.f3 from b where a.id=b.id) where a.
↪id=2;
--显示更新结果
TEST=# select * from a;
 ID |   F1   |   F2   |   F3
----+--------+--------+--------
  1 | Tom1   | Tom2   | Tom3
  2 | John1b | John2b | John3b
(2 行)
```

```
该多列更新语句的等价语句：
Update a SET
    a.f1=(select b.f1 from b where a.id=b.id),
    a.f2=(select b.f2 from b where a.id=b.id),
    a.f3=(select b.f3 from b where a.id=b.id)
where a.id=2;
```

2.3.9 INSERT INTO TABLE([前缀] 列)

为兼容 Oracle，KingbaseES 提供 INSERT [前缀] 列功能，相关语法如下所示：

```
INSERT INTO table_name [ AS alias ] [ ( column_name [, ...] ) ]
    { DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) [, ...] | query }
    [ ON CONFLICT [ conflict_target ] conflict_action ]
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]
```

例 2-47：*INSERT* [前缀] 列的示例。

```
--KingbaseES 代码
CREATE SCHEMA TESTINST;
CREATE TABLE TESTINST.INS1(ID INT, NAME VARCHAR);
INSERT INTO TESTINST.INS1(TESTINST.INS1.ID) VALUES(1);
INSERT INTO TESTINST.INS1(TESTINST.INS1.ID,TESTINST.INS1.NAME) VALUES(1,'TOM');
```

(continues on next page)

(continued from previous page)

```
SELECT * FROM TESTINST.INS1;
 ID | NAME
----+------
  1 |
  1 | TOM
(2 rows)

--Oracle 代码
CREATE USER C##TESTINST IDENTIFIED BY 123;
GRANT DBA TO C##TESTINST;
CREATE TABLE C##TESTINST.INS1 (ID INT, NAME VARCHAR(10));
INSERT INTO C##TESTINST.INS1(C##TESTINST.INS1.ID) VALUES(1);
INSERT INTO C##TESTINST.INS1(C##TESTINST.INS1.ID,C##TESTINST.INS1.NAME) VALUES(1,'TOM
↪');
SELECT * FROM C##TESTINST.INS1;
SELECT * FROM C##TESTINST.INS1;
    ID  NAME
---------- ----------
     1
     1 TOM
```

2.3.10 *DELETE [FROM]* 语句

为兼容 Oracle，KingbaseES 提供 DELETE [FROM] 功能，相关语法如下所示：

```
IDELETE [FROM] [ ONLY ] table_name [ * ] [ [ AS ] alias ]
    [ USING using_list ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [ [ AS ] output_name ] [, ...] ]
```

例 2-48：*DELETE [FROM]* 功能，该例在 *KingbaseES* 和 *Oracle* 中均可执行。

```
CREATE TABLE TESTFR(A INT, B INT);
INSERT INTO TESTFR VALUES(1, 1);
INSERT INTO TESTFR VALUES(2, 2);
INSERT INTO TESTFR VALUES(3, 3);
DELETE FROM TESTFR WHERE A = 3;
DELETE TESTFR WHERE A = 2;
SELECT * FROM TESTFR;
 A | B
---+---
 1 | 1
(1 row)
```

2.3.11 MERGE INTO 语句

MERGE INTO 语句根据与源表连接的结果，对目标表进行插入和更新操作，通常用来实现表同步。KingbaseES 不支持 MERGE INTO 语句，但是可以使用 INSERT ON CONFLICT 进行移植。

MERGE INTO 的语法如下所示：

```
MERGE INTO [SchemaName.] { TableName | ViewName }
    USING { TableName | ViewName | <SelectStatement> }
    ON <ConditionExpression>
    <MergeUpdate>
    <MergeInsert>
```

例 2-49：*MERGE INTO* 语句的示例。

```
--建表语句和数据插入语句
CREATE TABLE TB1(A INT,B CHAR(10));
CREATE TABLE TB2(A INT,B CHAR(10));
INSERT INTO TB1 VALUES(1, 'A');
INSERT INTO TB1 VALUES(3, 'D');
INSERT INTO TB2 VALUES(1, 'B');
INSERT INTO TB2 VALUES(2, 'C');
--B2 中的数据同步到 TB1 中
MERGE INTO TB1 USING TB2
ON TB1.A=TB2.A
WHEN MATCHED THEN UPDATE SET TB1.B=TB2.B
WHEN NOT MATCHED THEN INSERT VALUES(TB2.A,TB2.B);
--MERGE INTO 后 TB1 数据
TEST=# SELECT * FROM TB1 ORDER BY A;
 A |     B
---+------------
 1 | B
 2 | C
 3 | D
(3 行)
```

2.3.12 WITH 子句

KingbaseES 的 WITH 子句和 Oracle 兼容。WITH 后面的临时结果集被主语句或位置上位于其后的公用表表达式以及位置上位于其后的下层次公用表表达式使用。

WITH 子句语法如下：

```
with_clause ::=
WITH [ RECURSIVE ] with_query [, ...]
with_query ::=
with_query_name [ ( column_name [, ...] ) ] AS ( select )
```

例 2-50：*WITH* 语句的示例。

```
--建表语句和数据插入语句
CREATE TEMPORARY TABLE y (a INTEGER);
INSERT INTO y VALUES(1);
--WITH 语句
WITH t AS
(
```

(continues on next page)

(continued from previous page)

```
    SELECT a FROM y
)
INSERT INTO y
SELECT a+20 FROM t RETURNING *;
```

2.3.13 FORCE VIEW 语句

KingbaseES 的 FORCE VIEW 语句和 Oracle 兼容。强制创建视图，不管创建视图依赖的对象是否存在。

FORCE VIEW 语法:

```
CREATE [ OR REPLACE ] FORCE VIEW [SchemaName.]ViewName [ (ColumnName [, ...n] ) ]
   AS <SelectStatement> [ WITH [ CASCADED ] CHECK OPTION ];
```

例 2-51：*FORCE VIEW* 语句的示例。

```
--假设 t 表不存在，强制建立视图 view_t
CREATE OR REPLACE FORCE VIEW VIEW_T AS SELECT * FROM T;
```

2.3.14 支持 SEQUENCE 语句

1、定义一个新的序列发生器，语法和 oracle 有一点小出入

```
KingabSE 语法：
CREATE [ TEMPORARY | TEMP ] SEQUENCE [ IF NOT EXISTS ] name
[ INCREMENT [ BY ] increment ]
[ MINVALUE minvalue | NO MINVALUE ]
[ MAXVALUE maxvalue | NO MAXVALUE ]
[ START [ WITH ] start ]
[ CACHE cache ] [ [ NO ] CYCLE ]
[ OWNED BY { table_name.column_name | NONE } ]

Oracle 语法：
CREATE SEQUENCE sequence_name
[START WITH num]
[INCREMENT BY increment]
[MAXVALUE num|NOMAXVALUE]
[MINVALUE num|NOMINVALUE]
[CYCLE|NOCYCLE]
[CACHE num|NOCACHE]
[ORDER|NOORDER]
[KEEP|NOKEEP]
[SESSION|GLOBAL]
```

2、修改一个序列发生器，语法和 oracle 有一点小出入

```
KingabSE 语法：
ALTER SEQUENCE [ IF EXISTS ] name
[ INCREMENT [ BY ] increment ]
[ MINVALUE minvalue | NO MINVALUE ]
[ MAXVALUE maxvalue | NO MAXVALUE ]
[ START [ WITH ] start ]
[ RESTART [ [ WITH ] restart ] ]
```

(continues on next page)

(continued from previous page)

```
[ CACHE cache ] [ [ NO ] CYCLE ]
[ OWNED BY { table_name.column_name | NONE } ]
ALTER SEQUENCE [ IF EXISTS ] name OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
ALTER SEQUENCE [ IF EXISTS ] name RENAME TO new_name
ALTER SEQUENCE [ IF EXISTS ] name SET SCHEMA new_schema

Oracle 语法：
ALTER SEQUENCE sequence_name
[START WITH num]
[INCREMENT BY increment]
[MAXVALUE num|NOMAXVALUE]
[MINVALUE num|NOMINVALUE]
[CYCLE|NOCYCLE]
[CACHE num|NOCACHE]
[ORDER|NOORDER]
[KEEP|NOKEEP]
[SESSION|GLOBAL]
```

3、删除一个序列发生器，语法和 oracle 有一点小出入

```
KingabSE 语法：
DROP SEQUENCE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
Oracle 语法：
DROP SEQUENCE sequence_name
```

4、一个简单 sequence 使用：

例 2-51：简单 sequence 的示例。

```
 --创建一个 sequence
 create sequence seq;
 CREATE SEQUENCE
 select SEQ.nextval;
  NEXTVAL
 ---------
    1
 (1 row)
 select SEQ.nextval;
  NEXTVAL
 ---------
     2
(1 row)
select SEQ.currval;
 CURRVAL
 ---------
     2
 (1 row)
 select SEQ.nextval@dblk;
  NEXTVAL
 ---------
    3
 (1 row)
 select SEQ.currval@dblk;
  CURRVAL
 ---------
    3
 (1 row)
```

(continues on next page)

(continued from previous page)

```
drop sequence seq;
DROP SEQUENCE
```

2.3.15 INSERT ALL|FIRST 语句

KingbaseES 的 insert alllfirst 语句基础功能和 Oracle 兼容，部分语法不兼容, 比如 insert all 分区表、物化视图等。insert alllfirst 实现一条向多个目标表插入数据，方便快捷。

INSERT ALL|FIRST 语法:

```
 --KingbaseES 语法
 INSERT ALL into_clause [into_clause]..  subquery;
 INSERT {ALL|FIRST} condition_clause [condition_clause].. condition_else_clause␣
↪subquery;

 into_clause ::=
     INTO [schema.]{table_name|view_name} [t_alias][(column_name [,column_name]..)]␣
↪[values_clause]

 condition_clause ::=
     WHEN condition_expr THEN into_clause [into_clause]..

 condition_else_clause ::=
     [ELSE into_clause [into_clause]..]

 values_clause ::=
     VALUES ({expre|default}[,{expre|default}]..)

--Oralce 语法
 INSERT
  {
    [
      ALL insert_into_clause
      [values_clause]
      [error_logging_clause]
      [insert_into_clause [values_clause][error_logging_clause]]..
    ]
    | condititional_insert_clause
  }
  subquery

 insert_into_clause ::=
     INTO
     dml_table_expression_clause
     [t_alias]
     [(column_name[,coloumn_name]..)]

 value_valuse ::=
     values ({expr|default}[,{expr|default}]..)

 error_logging_clause ::=
     LOG ERRORS [INTO [schema.] table_name] [( simple_expression )]
     [REJECT LIMIT {integer|UNLIMITED}]

 condititional_insert_clause ::=
```

(continues on next page)

(continued from previous page)

```
    [{ALL|FIRST}]
    WHEN condition THEN
      insert_into_clause
      [values_clause]
      [error_logging_clause]
      [insert_into_clause [values_clause] [error_logging_clause]]..
    [
      WHEN condition THEN insert_into_clause
      [values_clause]
      [error_logging_clause]
      [insert_into_clause [values_clause] [error_logging_clause]]..
    ]..
    [
      ELSE insert_into_caluse
      [values_clause]
      [error_logging_clause]
      [insert_into_caluse [values_clause] [error_logging_clause]]..
    ]

 dml_table_expression_clause ::=
    {
      [schema.] {[table_name [{partition_extension_clause | @dblink}]] | [{view |␣
↪materialized view} [@dblink]]}
      | (subquery [subquery_restriction_clause])
      | table_collection_expression
    }

 partition_extension_clause ::=
    {
      PARTITION {(partition) | FOR(partition_key_value,[partition_key_value]..)}
      | SUBPARTITION {(subpartition)
      | FOR(subpartition_key_value,[subpartition_key_value]..)}
    }
```

例 **2-52**：*INSERT ALL | FIRST* 语句的示例。

```
--insert all
insert all
  when c0 < 2 then into temp_t1
  when c0 > 1 and  c0 < 3 then into temp_t2
  else into temp_t3
select * from temp_t0;

--insert first
insert first
   when c0 < 2 then into temp_t1
   when c0 < 3 then into temp_t2
   else into temp_t4
select * from temp_t0 where c0 < 23;
```

2.4 模式兼容特性

本节内容旨在为移植过程的相关模式修改操作提供参考指南。

2.4.1 扩展数据类型

为兼容 Oracle 的数据类型，KingbaseES 扩展了 Oracle 的 *NUMBER*、*VARCHAR2*、*CHAR(n)* 和 *DATE* 类型。该措施使得移植 Oracle 的 *Create Table* 等 DDL 语句时，无需任何修改就能直接在 KingbaseES 环境中运行。

下面各表对比了 KingbaseES 和 Oracle 在各种数据类型上的异同点。

表 2.4.1: Kingbase 与 Oracle 数值数据类型对比表

数据类型名	**KingbaseES**	**Oracle**
NUMERIC(p,s)	标度 s 缺省值相同，都是 0。 p、s 都有默认值。 差异： 1）标度值域：Kingbase 是 [0,38]。 2）精度表示方式不同：KingbaseES 不允许用 * 代替。 3）精度缺省值不同：KingbaseES 缺省为 10。 4）值域不同：Kingbase[-999..9..9~9 99..9]。Numeric 族数据类型没有明确说明值域，但太大时会导致内存不够。Kingbase 的 Number 存储的都是精确数值。 5）精度和标度大小关系：KingbaseES 标度必须小于等于精度。	1）标度值域：Oracle 是 [-84,127]。即 Oracle 标度可为负数，表示精确到小数点左侧多少位。如：1234.56 存到 Number(6,-2) 中，表示精确到百位，最后结果为 1200。 2）精度表示方式不同：Oracle 精度可用*代替，表示精度为 38。 3）精度缺省值不同：Oracle 根据给定的值存储，如 123.456，则 p=3。 4） 值域不同：Oracle[-9 .99..9*10^{125} ~ -10^{-130}]，0，[10:sup:‘ -130‘~9.99..9*10:sup:*125*，保留 38 位有效数字。Oracle 的 Number 可以存储近似数值。 5）精度和标度大小关系：Oracle 无要求，精度小于标度时，表示小数点后有效数字最多可以是标度长度，如 0. 000123 存在 Number(4,5) 中，结果为：0.00012。
DECIMAL(p,s), DEC(p,s)	同 NUMERIC(p,s)	同 NUMERIC(p,s)
NUMBER(p,s)	同 NUMERIC(p,s)	同 NUMERIC(p,s)。Oracle 的 NUMBER 是一个超类型，它涵盖 integers、floats、decimals 等附带精度和标度的所有数据类型，因此在 Oracle 移植中应按需把其转换成 K ingbaseES 的 integers、floats、numeric、decimals 等数据类型。
BIGINT	1. 说明：有符号整数，占用 8 个字节。 2. 值域：-2^63 到 2^63-1，即：-9223372036854775808 到 9223372036854775807。 3. 赋值：对 BIGINT 类型赋值或从其他类型转换到 BIGINT 类型时，如果输入值是带有小数点的数值，则小数点后面的部分截断。例如：给 BIGINT 类型变量或列赋值 12.55，则实际上所赋的值为 12 。这与 SQLServer 一致。	1. 对 BIGINT 类型赋值或从其他类型转换到 BIGINT 类型时，如果输入值是带有小数点的数值，则进行四舍五入。

INTEGER, INT	1. 别名：INT 2. 说明：有符号整数，占用 4 个字节。 3. 值域：-2^31 到 2^31-1，即：-2147483648 到 +2147483647 赋值：与 BIGINT 相同。	内部都按 NUMBER 处理
SMALLINT/INT2	1. 说明：有符号整数，占用 2 个字节。 2. 值域：-32768 到 +32767 赋值：与 BIGINT 相同	无此数据类型定义为 smallint, int 或 integer 时，实际的数据类型为 number(22，0)。
TINYINT	1. 说明：有符号整数，占用 1 个字节。 2. 值域：-128 到 +127 赋值：与 BIGINT 相同	不支持
FLOAT[(n)]	两者均支持 FLOAT[n] 的格式。 KingbaseES 的 n 表示科学计数法尾数的二进制位数，范围是 [1 ,53]，n 在 [1,24] 时，相当于 REAL，精度为 7，值域是 [-3.40E+38,-1.4e- 45]、0、[1.4e-45,3.40 E+38]；n 在 [25,53] 时，相当于 DOU BLE，精度为 15，值域是 [-1.79E+308,-4.94E-324]、0、[4.94E-324,1. 79E+308]；n 默认为 53。	Oracle 的 n 表示二进制精度，范围是 [1,126]，要转为 10 进制精度需要乘以因子 0.30103，所以对应的 10 进制精度范围是 [1,38]，n 默认是 126。
REAL	1. 说明：单精度浮点数字数据，规格定义方法采用 S QL92 标准。占用 4 个字节，精度为 7 位十进制数字（相当于 FLOAT(24)）。 2. 值域：-3.40E+38 至-1.4e-4 5、0 以及 1.4e-45 至 3.40 E+38	不支持
DOUBLE PRECISION,DOUBLE	1. 别名:DOUBLE 2. 说明: 与 FLOAT 缺省定义相同。精度为 15 位十进制数，相当于 FLOAT(53) 值域：与 FLOAT 缺省定义相同。	不支持

表 2.4.3: Kingbase 与 Oracle 字符串类型对比表

数据类型名	**KingbaseES**	**Oracle**
CHARACTER(n), CHAR(n), NCHAR(n)	char 表示一个字符 Byte 表示一个字节。 默认为 1 值域：10485760 char | byte。	值域： CHAR=2000 byte， NCHAR=2000 char| byte。
CHARACTER VARYING(n), NVARCHAR(n), NVARCHAR2(n), VARCHAR(n), VARCHAR2(n)	值域：10485760 char | byte。 默认长度：可以不指定，默认为 1。	值域： VARCHAR2=4000 byte， NVARCHAR2=4000 char|byte。 默认长度： VARCHAR2 必须指定长度
CLOB	值域：2G-1 byte	有 CLOB 和 NCLOB，值域 4G-1 byte
TEXT	值域：变长，无长度限制，但最大长度最好不要超过 64K。太长时会导致内存不够。	LONG。最大长度为 2G-1 byte。
XML	值域：变长，无长度限制，服务器内部按照 TEXT 存储	Oracle 12c 没有 XML 类型

表 2.4.4: Kingbase 与 Oracle 二进制串数据类型对比表

数据类型名	KingbaseES	Oracle
BLOB	值域：2G-1byte	值域：4G-1byte
BYTEA	长度不限。太长时会导致内存不够。	Oracle 有 RAW 和 LONGRAW。值域是：RAW=2000 byte；LONG RAW=2G-1 byte
BFILE	支持	Oracle 支持外部的二进制文件，大小受操作系统限制，最大支持 4G

表 2.4.5: Kingbase 与 Oracle 日期时间数据类型对比表

数据类型名	KingbaseES	Oracle
DATE	KingbaseES 的 Date 只有在打开兼容 Oracle 开关时才是“年月日时分秒”，否则只包括“年月日”，打开兼容 Oracle 开关后 Date 变为 Timestamp(0) without time zone。 值域：现在可用的合理范围。 DATE 默认形式为（年，月，日）。打开兼容 Oracle 开关时 DATE 默认形式为（年，月，日，时，分，秒）占 19 位。	Oracle 的 Date 包括“年月日时分秒”，对应 Kingbase 的 Timestamp(0) without time zone。 值域：Oracle 的 date 为 [-4712.1.1 ~ 4712.12.31] 缺省值为 0 格式：yyyy-mm-dd hh:mm:ss 取值范围：-4712-01-01 00:00:00 -- 9999-12-31 23:59:59
TIME[(p)] WITH TIME ZONE	TIMETZ 相当于 TIME WITH TIME ZONE，如果需要含时区时间中指定 n 则不能使用别名方式定义，而只能使用 TIME(n) WITH TIME ZONE 的形式指定 n 值。如果指定 WITH TIME ZONE，则时间中会保留相应的时区。值域：00:00:00.000000 [-14:59] -- 23:59:59.999999 [-14:59]	不支持
TIME [WITHOUT TIME ZONE]	TIME（n）相当于 TIME（n）WITHOUT TIME ZONE，缺省 n=6，最大为 6。缺省时区设置为 WITHOUT TIME ZONE，占用 8 个字节。输出的毫秒部分长度与 n 值一致。值域：00:00:00.000000 [-14:59] -- 23:59:59.999999 [-14:59]	不支持
TIMESTAMP[(p)][WITH TIME ZONE]	TIMESTAMPTZ（n）相当于 TIMESTAMP（n）WITH TIME ZONE 精度：0 <= n <= 6，缺省精度为 6 值域：以下的时间边界值均以零时区为标准，一旦时间超出该边界值，则会提示时间越界的错误或输出特殊时间字面值。TIMESTAMP WITH TIME ZONE 类型的范围转换为 GMT 时间后与 TI MESTAMP 类型一致（除边界值外）。时区范围在 -12:59 到 +14:00 之间。	取值范围：0~9，省值为 6。 格式：yyyy-mm-dd hh:mm:ss.ff tzh:tzm 取值范围：-4712-01-01 00:00:00:000000000 13:00— 9999/12/31 23:59:59:99999999 -12:00 9i 和 10g 中有此数据类型，8i 中无。 时差 (TZH) 的范围是“-12 ~ 13”

TIMESTAMP[(p)] [WITHOUT TIME ZONE]	TIMESTAMP（n）相当于 TIMESTAMP（n）WITHOUT TIME ZONE 精度：0 <= n <= 6，缺省精度为 6 值域：以下的时间边界值均以零时区为标准，一旦时间超出该边界值，则会提示时间越界的错误或输出特殊时间字面值。取值范围从 BC 4714-11-24 00:00:00.000000 到 AD 294277-01-09 04:00:54.775806	取值范围：0~9，省值为 6。 格式： yyyy-mm-dd hh:mm:ss.ff 取值范围： -4712-01-01 00:00:00:000000000—9999-12-31 23:59:59:999999999

表 2.4.7: Kngbase 与 Oracle 布尔类型对比表

数据类型名	**KingbaseES**	**Oracle**
4. BOOLEAN	别名：BOOL 说明：逻辑布尔量。占用一位。说明：根据 SQL2003 标准规定，在对两个布尔类型值进行比较运算时，TRUE 值大于 FALSE 值。说明：对于任意能够直接地通过赋值类型转换转换为布尔类型的数值类型输入，非 0 数值将会转换为真值，否则转换为假值。直接地通过赋值类型转换定义为转换过程中不借助其他数据类型转换规则或其他中间数据类型的类型进行转换操作。 值域：TRUE|FALSE，1|0，结果显示为 t|f。	不支持

表 2.4.9: Kingbase 与 Oracle 时间间隔类型对比表

数据类型名	**KingbaseES**	**Oracle**

5. INTERVAL	主要分为两类：YEAR-MONTH 型和 DAY-TIME 型 值域：year_precision、month_precision、day_precision、hour_precision、minute_precision、second_precision 的取值范围都是从 1 到 9，缺省为 2。fractional_seconds_precision 的取值范围是从 0 到 6，缺省为 6。 INTERVAL YEAR/MONTH/DAY/ HOUR/ MINUTE/ SECOND 形式的 INTERVAL 类型支持两种输入格式：包含数值的字符串格式和包括数值和单位的表达式，INTERVAL YEAR TO MONTH / DAY TO SECOND 形式的 INTERVAL 类型仅支持字符串格式的输入，格式为：INTERVAL YEAR TO MONTH 形式：'yy-mm'INTERVAL DAY TO SECOND 形式：'dd hh:mm:ss'	INTERVAL DAY [(day_precision)] TO SECOND [(fractional_seconds_precision)] INTERVAL YEAR [(year_precision)] TO MONTH 说明： day_precision , year_precision: 取范围都是从 1 到 9，缺省为 2 fractional _seconds_precision: 取值范围是从 0 到 6，缺省为 6。

表 2.4.11: Kingbase 与 Oracle 位串数据类型对比表

数据类型名	**KingbaseES**	**Oracle**
BIT[(n)]	1. 说明：定长位串，即：0、1 串，字串全部由 0/1 组成。BIT 类型的数据必须准确匹配长度 n。如果显式地把一个位串值转换成 BIT(n)，那么右边将被截断或者补零到刚好 n 位，而不是报告错误。n 缺省为 1。 2. 值域：定长，不足补 0；最长为 64000 位（BIT）。 3. 此外，服务器在解析使用十六进制格式的字符串时，总是以 8 个 bit 作为对齐边界的	Oracle 并不支持此数据类型，Oracle 要存放此类数据，可用二进制类型或者字符型来表示。
BIT VARYING[(n)]	1. 说明：除长度可变外，其他规格与 BIT[(n)] 基本相同，缺省长度为 1。如果指定长度为 n 而输入长度不足 n 时，则保持输入的数据。 2. 值域：如果指定 n，则 n 最大为 64000 位 (BIT)。	Oracle 并不支持此数据类型，Oracle 要存放此类数据，可用二进制类型或者字符型来表示。

2.4.2 模式变更

为有效地降低数据库系统的管理复杂度，KingbaseES 提供灵活多样的模式变更方式，这些方式诸如：

- **模式对象名标识符的最大长度支持**

  在 KingbaseES 中，模式对象名（如索引名和约束名）标识符的最大长度可达到 63 字节。该长度有效地降低因源系统标识符长度太长而修改模式对象名的情况发生。

- **CREATE OR REPLACE 方式**

  与 Oracle 相同，在创建模式对象时，KingbaseES 也提供了 *CREATE OR REPLACE* 方式。该方式极大地简化了每次重新创建模式对象的流程。

2.4.3 序列

在序列上，KingbaseES 和 Oracle 具有相同的定义和语法。

表 2.4.13: 序列

差异	**Oracle**	**KingbaseES**
序列最大值	最大值 =999,999,999,999----999 (27 位)	最大值 =**2^63-1** 922,337,203,685,.....,7 (19 位)

例 2-52：序列的示例。该示例代码在 KingbaseES 和 Oracle 上均可运行。

```
--创建一个叫 test_serial 的递增序列，从 101 开始：
CREATE SEQUENCE test_serial START WITH 101;
--查看某一序列发生器的详细信息：
SELECT * FROM  test_serial;

--在一个 INSERT 中使用此序列：
CREATE TABLE test_serial_table(col_serial INT);
INSERT INTO test_serial_table(col_serial) VALUES(NEXTVAL('test_serial'));

select * from test_serial_table;
TEST=# select * from test_serial_table;
 COL_SERIAL
------------
         101
(1 行)
```

2.4.4 索引

KingbaseES 支持 Oracle 的大多数索引特性。这些特性如 *UNIQUE* 索引、函数索引、全局和分区索引、B-树索引、升序/降序索引方式、及 *NULL* 排最前或最后的方式。但是，KingbaseES 不支持 Oracle 的 **bitmap** 和 **Domain** 索引。

下面分别给出 KingbaseES 和 Oracle 在索引实现方面的主要相同点和异同点。

2.4.4.1 聚簇索引

KingbaseES 可根据已有普通索引建立聚簇索引，具体语法格式如下所示：

```
CLUSTER IndexName ON TableName
```

Oracle 也支持类似的功能，语法格式如下所示：

```
CREATE CLUSTER …
```

2.4.4.2 函数索引

函数索引是根据在表的一或多个列上的函数结果值所建立的索引，主要用来提高基于函数调用结果的数据访问效率。函数索引的对象既可是表达式，也可是函数。

KingbaseES 和 Oracle 均支持函数索引，且二者兼容。

例 2-53：在 *emp* 表的 *ename* 列上建立 *length* 函数索引。该示例代码在 KingbaseES 和 Oracle 上均可运行。

```
CREATE INDEX idx4 ON scott.emp (length(ename));
```

2.4.4.3 重命名索引

KingbaseES 和 Oracle 均支持重命名索引功能。

例 2-55：重命名索引的示例。该示例代码在 KingbaseES 和 Oracle 上均可运行。

```
ALTER INDEX name RENAME TO new_name;
```

2.4.4.4 嵌套表索引

KingbaseES 和 Oracle 均支持嵌套表索引。KingbaseES 嵌套表的索引类型为 *INT*。如果要与 Oracle 兼容，则必须把 *INT* 映射成 Oracle 的 *PLS_INTEGER*。

2.4.5 约束

KingbaseES 和 Oracle 支持的约束类型相同。这些类型如主键约束、外键约束、非空约束、缺省约束、唯一约束和 *Check* 约束。

此外，用户执行某些特定操作时，可能需临时禁掉约束，而完成操作后将重新启用约束。针对这种情形，Oracle 提供了约束禁止/启用功能，语句如下：

```
MODIFY CONSTRAINT ConstraintName { ENABLE \| DISABLE };
```

KingbaseES 暂时不支持该功能。

例 2-57：约束禁用和启用的示例。该示例代码在 Oracle 和 KingbaseES 中均可执行。

```
CREATE TABLE c (id INT);
alter table c add constraint c_pk primary key(id);
ALTER TABLE c MODIFY CONSTRAINT c_pk DISABLE;
INSERT INTO c values(1);
INSERT INTO c values(2);
ALTER TABLE c MODIFY CONSTRAINT c_pk ENABLE;
```

2.4.6 同义词

同义词的用途是什么呢？在实际应用中，用户通常会为远程对象创建一个简单易记的同义词标识，然后使用该标识，从本地数据库中透明地访问远程数据库对象。这种方式既可保证访问安全，又可简化冗长的远程对象标识。

Oracle 支持同义词功能，KingbaseES 暂时不支持同义词。

依据所属特性，Oracle 的同义词可分为：

- 公有同义词：当创建同义词使用 *PUBLIC* 关键字时，则创建的同义词是公有同义词，其他用户都可访问。
- 私有同义词：如果不指定 *PUBLIC* 关键字，则同义词会创建到*"$user"* 模式下，此时的同义词是私有同义词，其他用户应该使用“*模式名.同义词名*”来引用这个同义词。

例 2-58：同义词的示例。该示例代码在 Oracle 中可执行。

```
CREATE TABLE emp(empno INT, ename VARCHAR(20));
INSERT INTO emp VALUES(1, 'Tom');
CREATE OR REPLACE SYNONYM my_syn FOR emp;
SELECT * FROM my_syn;
TEST=# SELECT * FROM my_syn;
 EMPNO | ENAME
-------+-------
     1 | Tom
(1 行)
```

2.4.7 视图

在视图上，KingbaseES 和 Oracle 是非常相似的。Oracle 的大多数 *CREATE VIEW* 语句都无需改动就可在 KingbaseES 上运行，并支持对视图的更新 update 操作。

例 2-59：视图的示例。该示例代码在 Oracle 和 KingbaseES 中均可执行。

```
CREATE table DEPARTMENTS (
"DEPT_CODE" CHAR(3) NOT NULL,
"DEPT_NAME" VARCHAR2(30),
"TOTAL_PROJECTS" NUMBER,
"TOTAL_EMPLOYEES" NUMBER);

CREATE table EMPLOYEES (
"EMP_ID" NUMBER(5) NOT NULL,
"FIRST_NAME" VARCHAR2(20),
"LAST_NAME" VARCHAR2(20),
"CURRENT_PROJECTS" NUMBER(3),
"EMP_MGR_ID" NUMBER(5),
"DEPT_CODE" CHAR(3) NOT NULL,
"ACCT_ID" NUMBER(3) NOT NULL,
"OFFICE_ID" NUMBER(5),
"BAND" CHAR(1),
"CREATE_DATE" TIMESTAMP(3) DEFAULT SYSDATE);

CREATE OR REPLACE VIEW organization_structure
("LEVEL", "FULL_NAME", "DEPARTMENT", "ASSIGNMENTS") AS
SELECT
LEVEL,
SUBSTR((LPAD(' ', 4 * LEVEL - 1) || INITCAP(e.last_name) || ', '
```

(continues on next page)

(continued from previous page)

```
|| INITCAP(e.first_name)), 1, 40),
NVL(d.dept_name, 'Uknown') ,COALESCE (d.DEPT_CODE, '001') as assignments
FROM
employees e,
departments d
WHERE
e.dept_code=d.dept_code(+)
START WITH emp_id = 1 CONNECT BY NOCYCLE PRIOR emp_id = emp_mgr_id;
```

这个例子用到了如下 SQL 功能：

- *CREATE OR REPLACE VIEW*
- 层次查询：*START WITH ... CONNECT BY*
- *LEVEL* 伪列
- *COALESCE*、*INITCAP*、*NVL* 标量函数
- 外连接 (+)

2.4.8 物化视图

在物化视图上，KingbaseES 和 Oracle 是基本相似的。

```
KingbaseES 语法：
CREATE MATERIALIZED VIEW view_name
    [ (column_name [, ...] ) ]
[ TABLESPACE tablespace_name ]
    AS query
    [ WITH [ NO ] DATA ]
```

相比 oracle 的多种刷新方式，目前 KingbaseES 支持全部刷新：refresh materialized view，支持并发刷新。

例 2-60：物化视图的使用用例。

```
SAMPLES=# create table T1(id int);
CREATE TABLE
SAMPLES=# insert into T1 values(1),(2),(3);
INSERT 0 3
SAMPLES=# create materialized view A as select * from t1 ;                    ---创建物化视
图
CREATE MATERIALIZED VIEW
SAMPLES=# select * from A;
 ID
----
  1
  2
  3
(3 行)

SAMPLES=# insert into T1 values(4),(5),(6);
INSERT 0 3
SAMPLES=# select * from A;                                                    --插入数据后，物化视图
没有刷新
 ID
----
```

(continues on next page)

(continued from previous page)

```
  1
  2
  3
(3 行)

SAMPLES=# refresh materialized view A ;                                    --刷新物化视图
REFRESH MATERIALIZED VIEW
SAMPLES=# select * from A;
 ID
----
  1
  2
  3
  4
  5
  6
(6 行)
```

2.4.9 全局临时表

KingbaseES 支持本地临时表和全局临时表，而 Oracle 只支持全局临时表。全局临时表又分为事务级和会话级，对于 on commit delete rows 类型的事务临时表，事务提交后会话内的临时表数据即被销毁，对于 on commit preserve rows 类型的会话临时表，会话结束后数据销毁。

KingbaseES 临时表的创建语法，与 oracle 相同

语法：

```
CREATE GLOBAL [{ TEMPORARY | TEMP } ]
     TABLE TableName
 (column [, ...n ] )
[ON COMMIT [preserve | delete] ROWS ]
```

2.4.10 水平分区

Oracle 支持水平分区功能。KingbaseES 可以使用继承模拟水平分区，也可以使用兼容 Oracle 的语法创建水平分区。

查看 oracle 上有哪些分区表：

```
select * from user_tables where partitioned='YES';
```

迁移 Orcale 的分区表时，需要手动在 KingbaseES 库中创建分区表后，利用 EasyTransfer Tool 工具进行插入迁移。

例 2-61：列表分区的示例。该示例代码在 Oracle 上可运行，也可以直接在 KingbaseES 上运行。

```
CREATE TABLE employee (id INT, dept CHAR(10))
    PARTITION BY LIST (id)
    (
        PARTITION p11 VALUES (10),
        PARTITION p12 VALUES (20, 30),
        PARTITION p13 VALUES (40, null),
        PARTITION p14 VALUES (DEFAULT)
```

(continues on next page)

(continued from previous page)

```
    ) ;

--插入到 p11 分区中：
INSERT INTO employee VALUES(10, 'A');

--插入到 p12 分区中：
INSERT INTO employee VALUES(20, 'B');
INSERT INTO employee VALUES(30, 'B');

--插入到 p13 分区中：
INSERT INTO employee VALUES(40, 'C');
INSERT INTO employee VALUES(null, 'C');

--插入到 p14 分区中：
INSERT INTO employee VALUES(50, 'N');
INSERT INTO employee VALUES(1,'N');

--查看分区表，返回各个分区中的数据：
SELECT * FROM employee;
SQL> SELECT * FROM employee;

        ID DEPT
---------- --------------------
        10 A
        20 B
        30 B
        40 C
           C
        50 N
         1 N

已选择 7 行。
```

例 2-62：哈希分区的示例。该示例代码在 Oracle 可运行，也可以直接在 KingbaseES 上运行。

```
CREATE TABLE employees (id INT, store_id INT)
    PARTITION BY HASH (store_id)
    (
        PARTITION P21,
        PARTITION P22,
        PARTITION P23,
        PARTITION P24
    );

--插入数据：
INSERT INTO employees VALUES(1, 10);
INSERT INTO employees VALUES(2, 15);
INSERT INTO employees VALUES(3, 5);
INSERT INTO employees VALUES(4, 20);

--查看分区表，返回各个分区中的数据：
SELECT * FROM employees;
SQL> SELECT * FROM employees;

        ID   STORE_ID
---------- ----------
         1         10
```

(continues on next page)

(continued from previous page)

```
         3          5
         4         20
         2         15
```

例 2-63：范围分区的示例。该示例代码在 Oracle 可运行，也可以直接在 KingbaseES 上运行。

```
CREATE TABLE sales_demo (year INT, month INT, day INT, amount_sold INT)
     PARTITION BY RANGE (year,month)
    (
        PARTITION before2001 VALUES LESS THAN (2001,1),
        PARTITION q1_2001 VALUES LESS THAN (2001,4),
        PARTITION q2_2001 VALUES LESS THAN (2001,7),
        PARTITION q3_2001 VALUES LESS THAN (2001,10),
        PARTITION q4_2001 VALUES LESS THAN (2002,1) ,
        PARTITION future VALUES LESS THAN (MAXVALUE,0)
    );

--插入到 before2001 分区中：
INSERT INTO sales_demo VALUES(2000,12,12, 1000);

--插入到 q1_2001 分区中：
INSERT INTO sales_demo VALUES(2001,3,17, 2000);

--插入到 q4_2001 分区中：
INSERT INTO sales_demo VALUES(2001,11,1, 5000);

--插入到 future 分区中：
INSERT INTO sales_demo VALUES(2002,1,1, 4000);

--查看分区表，返回各个分区中的数据：
SELECT * FROM sales_demo;
SQL> SELECT * FROM sales_demo;

      YEAR      MONTH        DAY AMOUNT_SOLD
---------- ---------- ---------- -----------
      2000         12         12        1000
      2001          3         17        2000
      2001         11          1        5000
      2002          1          1        4000
```

2.4.11 数据库链接

所谓数据库链接（Database Link）是数据库管理系统提供的、用来访问外部数据库对象的机制。KingbaseES 和 Oracle 均支持该功能，但它们的使用语法不同。

例 2-64：Oracle 的 DATABASE LINK 和 Kingbae 的 dblink 语法不同。下面是一个 oracle 的 DATABASE LINK 连接和执行的示例。

```
--创建连接
create database link linkName
  connect to linkUser identified by linkPwd
  using '(DESCRIPTION =
                (ADDRESS_LIST =
                  (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.8.121)(PORT = 1521))
                )
```

(continues on next page)

(continued from previous page)

```
                (CONNECT_DATA =
                  (SERVICE_NAME = linkServiceName)
                )
              )';
--using 后面可以在配置文件 tnsname.ora 中定义
zytydic =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.8.121)(PORT = 1521))
)
(CONNECT_DATA =
(SID = orcl)
)
)
...

--使用 DBLink 进行查询
select * from tableName@linkName;
```

例 2-65：Kingbase 打开一个到远程数据库的持久连接，在一个远程数据库中执行一个查询，关闭一个到远程数据库的持久连接。更多使用方法，请参见用户手册 dblink。

```
--打开一个到远程数据库的持久连接
SELECT dblink_connect('myconn', 'dbname=TEST port=54321 user=myuser' );
 dblink_connect
----------------
 OK
(1 row)
--在一个远程数据库中执行一个查询
SELECT * FROM dblink('myconn','SELECT * FROM foo') AS t(a int, b text, c text[]);
 a  | b |       c
----+---+---------------
  0 | a | {a0,b0,c0}
  1 | b | {a1,b1,c1}
  2 | c | {a2,b2,c2}
  3 | d | {a3,b3,c3}
  4 | e | {a4,b4,c4}
  5 | f | {a5,b5,c5}
  6 | g | {a6,b6,c6}
  7 | h | {a7,b7,c7}
  8 | i | {a8,b8,c8}
  9 | j | {a9,b9,c9}
 10 | k | {a10,b10,c10}
(11 rows)
--关闭一个到远程数据库的持久连接
SELECT dblink_disconnect('myconn');
 dblink_disconnect
-------------------
 OK
(1 row)
```

2.4.12 系统视图

为进一步降低移植难度，KingbaseES 在数据字典上也对 Oracle 进行了兼容性支持。具体实现方式是在 KingbaseES 内部提供了大量兼容 Oracle 的系统视图。这些视图部分或全部地兼容 Oracle 的 *USER*_、* *ALL*_* 和 *DBA*_* 类系统视图。

下面各表分别给出了 KingbaseES 对 Oracle 各类系统视图的兼容情况。

表 2.4.14: 完全兼容 Oracle 的 KingbaseES 系统视图

兼容系统视图	描述
ALL_CONSTRAINTS	描述当前用户可以访问的表上的约束定义
ALL_IND_COLUMNS	描述当前用户所能获取到的表上建有索引的列信息
ALL_JOBS	记录数据库中当前用户所能查看的所有作业的信息。该视图是兼容 Oracle 的。除了最后一项 DBNAME，Oracle 中不存在此项。
ALL_SYNONYMS	视图 ALL_SYNONYMS 提供了可以被当前用户引用的所有同义词的信息
ALL_TAB_COLS	描述了当前用户可以访问的表，视图的列。
ALL_TAB_COLUMNS	描述当前所在数据库中，所有的表、视图的非隐藏列信息。该视图的所有列来自于 DBA_TAB_COLS 的同名列，但是该视图显示的是非隐藏列信息。
ALL_VIEWS	描述当前用户所能查看的所有的视图信息
DBA_IND_COLUMNS	视图 DBA_IND_COLUMNS 提供了在数据库中所有表上索引中包含的所有列的信息
DBA_SYS_PRIVS	描述用户或角色被授予的系统权限
DBA_TAB_COLS	描述当前所在数据库中，所有的表、视图的列信息。
DBA_TAB_CLOUMNS	描述当前所在数据库中，所有的表、视图的非隐藏列信息。该视图的所有列来自于 DBA_TAB_COLS 的同名列，但是该视图显示的是非隐藏列信息。
DBA_USERS	数据库中所有用户的信息
DBA_VIEWS	数据库中所有视图的信息
USER_CONSTRAINT	描述当前用户可以访问的表上的约束定义
USER_IND_COLUMNS	当前用户的所有建有索引的列的信息
USER_SYS_PRIVS	描述当前用户被授予的系统权限
USER_TAB_COLS	描述了当前用户所拥有的表，视图的列
USER_TAB_COLUMNS	描述了当前用户所拥有的的表，视图的非隐藏列。
USER_VIEWS	当前用户的所有视图的信息

表 2.4.16: 部分兼容 Oracle 的 KingbaseES 系统视图

兼容系统视图	描述
ALL_COL_PRIVS	当前用户下可以查看的所有列级权限，既包括 USER_COL_PRIVS 中的内容，同时也包含 GRANTEE 是当前角色的列级权限。
ALL_INDEXES	视图 ALL_INDEXES 提供了在当前用户可访问的表上定义的索引的信息
ALL_OBJECTS	Oracle 视图 ALL_OBJECTS 提供下列数据库对象的信息-包括表，索引，序列，同义词，视图，触发器，函数，存储过程，包定义，和包主体。但 KingbaseES 的是这个视图中只显示了 PL/SQL 的触发器，函数，存储过程，包定义和包主体。
ALL_TABLES	视图 ALL_TABLES 提供所有用户定义表的信息
DBA_COL_PRIVS	所有列级权限
DBA_OBJECTS	数据库中可以查看的所有对象
DBA_TABLES	数据库中所有表的信息

USER_COL_PRIVS	当前用户下的列级权限视图
USER_OBJECTS	当前用户的所有对象的信息
USER_TABLES	当前用户的所有表的信息

2.4.13 大对象

Oracle 支持四种大对象类型，即 *BLOB*、*CLOB*、*NCLOB* 和 *BFILE*。它们都是变长类型。每个大对象的最大数据容量为 4GB。其中，*BLOB* 和 *BFILE* 存储的是可变长的二进制数据，而 *CLOB* 和 *NCLOB* 存储的是可变长的字符型数据。与 *CLOB* 不同，*NCLOB* 数据所用字符集是 Unicode。此外，与其他三种类型不同，*BFILE* 数据并不存储在数据文件中，而是独立于数据文件存在的。BFILE 字段只存储了文件指针信息。

相对的，目前 KingbaseES 只支持 *BLOBCLOB* 和 *BFILE* 类型，不支持 *NCLOB* 类型。此外，它的每个大对象的最大数据容量为 1GB。

2.4.14 表空间模式

Oracle 表空间模式支持两类模式 *ONLINE/OFFLINE*, *READ WRITE/ONLY*。其中 *ONLINE* 模式表示表空间在线，具体能否写访问依赖于是 *READ WRITE* 模式还是 *READ ONLY*，前者提供读写服务，后者提供只读服务。如果是 *OFFLINE* 模式的话表示表空间离线，那么将不能提供任何的服务。

目前 KingbaseES 也支持了上述的四种模式的设置，其设置方法兼容了 Oracle 的语法。

如创建过程中设置离线模式：

语法：

```
CREATE TABLESPACE tblspace_tsp1 LOCATION '/data/tsp1' OFFLINE;
```

又如变更表空间的模式：

语法：

```
ALTER TABLESPACE tblspace_tsp1 ONLINE;

ALTER TABLESPACE tblspace_tsp1 READ ONLY;
```

这些都是和 Oracle 的语法相同的，同时各个模式下对表空间对象的访问策略也基本兼容，不同点也是存在的，比如 Oracle 的只读模式对于意向写操作是允许的，而 KingbaseES 则是禁止。

而在表空间模式的变更过程中也会有一些和 Oracle 异同点，总结如下：

- KingbaseES 支持从在线 READ WRITE 状态切换到 READ ONLY 状态，切换过程会等待已发生的写事务提交或者回滚后进行。且在切换后会立即禁止所有后续发生的写事务发生。这一点基本和 Oracle 行为相同。反之从在线 READ ONLY 切换到 READ WRITE 的过程中，写事务需要等待切换动作完成才可以进行。这一系列过程对于读事务是透明的，不会产生任何影响。
- Oracle 表空间的离线分为三种 NORMAL、TEMPORARY、IMMEDIATE，而 KingbaseES 仅支持类似 NORMAL 的模式，在切换为 OFFLINE 的过程会触发脏页的回刷，等待脏页回刷完成后方可完成。此外从 ONLINE 切换为 OFFLINE 的模式时，会等待所有已发生的读、写事务完成，而对于后续发生的任何事务都会禁止。反之从 OFFLINE 切换到 ONLINE 状态时，读写事务要等待状态切换完成后才可以进行。
- KingbaseES 支持事务内进行表空间模式的切换，但并不支持子事务或者 savepoint 中设置表空间模式。
- 后台的 AUTOVACCUM 进程总是能够对只读或离线的表空间进行表空间所有对象的数据回收和整理。

- 为 DBA 用户提供了一个 GUC 参数 skip_tablespace_check，该参数开启后 DBA 用户可以访问只读表空间对象、甚至是离线表空间对象。注意：现阶段 skip_tablespace_check 参数默认开启，如需要使用表空间只读、离线功能，请关闭该参数。

2.4.15 隐含列特性

隐含列功能为兼容 Oracle 隐含列而开发的特性，同时也可以解决 V8R3 向 V8R6 进行迁移过程中导致的带有 Oid 的列丢失的问题。具体特性如下：

1、隐含列为表中各个列的一个特殊的属性，可以在建表过程中指定，也可以通过表的 ALTER 语句变更某一列的隐含属性，具体语法可以参考创建表语法和 ALTER TABLE 语法。

例 2-53：*CREATE* 和 *ALTER* 隐含列语句的示例

```
-- CREATE TABLE
 CREATE TABLE IMPLICT_T1(
   OID INT INVISIBLE,
   B INT,
   NAME VARCHAR(100),
   PRIMARY KEY(OID, B));

 ALTER TABLE IMPLICT_T1 MODIFY OID VISIBLE;

 ALTER TABLE IMPLICT_T1 MODIFY NAME INVISIBLE;
```

2、如果一个列被标记为隐含列之后，所有非指定该列的 DML 行为，例如 SELECT *、COPY、bulkload 等等，都会忽略该列。而所有指定该列的语句，都会命中该列。

例 2-54：隐含列相关的 *DML* 语句的示例

```
-- INSERT 语句需要指定隐含列
INSERT INTO IMPLICT_T1(OID, B, NAME) VALUES(1, 2, 'IMPLICT');
-- 查询语句不指定隐含列，将忽略该列
SELECT * FROM IMPLICT_T1;
postgres=# select * from implict_t1;
b |  name
---+---------
2 | implict
(1 row)
```

3、参考 Oracle 的特性，所有列都赋予了一个 COLUMN ID 的属性，COLUMN ID 从 1 开始，如果某一列被设置为隐含列，那么它的 COLUMN ID 则设置为 NULL。同时比该列原有的 COLUMN ID 大的列的 ID 都会减少，同时如果该列被恢复为正常列，那么该列的 COLUMN ID 会被重新设置为有效 ID 的最大值加 1。可以通过 USER_TAB_COLS 等 Oracle 兼容视图查看 COLUMN ID。注：Oracle 的所有非指定列的语句，其输出顺序实际上是按照 COLUMN ID 进行排序，而 KingbaseES 由于自身存储架构设计在这一点上没有做到彻底的兼容。其输出顺序仍然按照建表时候指定列的顺序输出。

例 2-55：*COLUMN ID* 相关的语句的示例

```
postgres=# select table_name, column_name, column_id from user_tab_cols where table_
↪name = 'IMPLICT_T1';
 table_name | column_name | column_id
------------+-------------+-----------
 IMPLICT_T1 | OID         |
 IMPLICT_T1 | B           |         1
 IMPLICT_T1 | NAME        |         2
(3 rows)
```

(continues on next page)

(continued from previous page)

```
postgres=# alter table implict_t1 modify b invisible;
ALTER TABLE
postgres=# select table_name, column_name, column_id from user_tab_cols where table_
↪name = 'IMPLICT_T1';
 table_name | column_name | column_id
------------+-------------+-----------
 IMPLICT_T1 | OID         |
 IMPLICT_T1 | B           |
 IMPLICT_T1 | NAME        |         1
(3 rows)
```

4、迁移考虑，如果从 V8R3 向 V8R6 迁移，那么 R3 的表中都会存在 OID 的伪列，而迁移过程又会将该列丢弃。因此可以考虑开启 default_with_oids 这个参数。该参数打开后，会为每个表建立一个 OID 的隐含列作为首列，并且类型为 SERIAL，是一个单表的 SEQUENCE，该办法可以解决迁移问题。

2.5 KingbaseES 客户端 SQL 交互工具

在客户端 SQL 交互工具方面，KingbaseES 提供了以下工具：

- KSQL：命令行的 SQL 交互工具，类似与 Oracle 的 SQL* PLUS 命令行工具。
- 数据库服务器系统管理工具：图形化的 SQL 交互工具，类似与 Oracle 的 SQL Developer 图形化工具。

通过上述工具，用户可连接数据库服务器，运行数据库实用程序，发送 SQL 语句，运行 SQL 脚本，或实施数据库管理等。

第 3 章

Oracle 数据库移植实战

由于 KingbaseES 内部提供了大量的 Oracle 兼容特性，因此，在实际应用中，一般只需很少甚至不做任何修改，用户便可把 Oracle 数据库移植到 KingbaseES 环境中运行。不仅如此，用户还可利用 EasyTransfer Tool 等多种工具简化移植过程。

本节重点描述了在实际应用中移植一个 Oracle 数据库系统的完整过程，以及其中的主要移植内容、常用移植方法和关键移植步骤。

3.1 主要移植内容

在实际应用中，一个 Oracle 数据库系统的移植主要包括如下内容。这些内容的迁移是存在先后顺序的。若违反该顺序，则可能导致迁移受阻。

3.1.1 数据库、用户和模式移植

数据库和模式是各种 SQL 和 PL/SQL 数据库对象的存放容器，而用户是这些对象的管理者和使用者。因此，在迁移数据库对象之前，一般应先迁移数据库、用户和模式。

那么，如何移植这些内容呢？应在目的数据库 KingbaseES 上创建与源数据库 Oracle 同名的数据库、用户和模式, 并授予新建用户具有使用该数据库和新建模式的所有或适当的权限。

另外，所创建数据库的字符集应与 Oracle 数据库字符集一致。如果 KingbaseES 已有同名数据库，则登录该数据库后，则只需创建同名用户和属主为该用户的同名模式。

3.1.2 数据库对象移植

迁移数据库、用户和模式以后，则可通过迁移用户开始迁移数据库的 SQL 和 PL/SQL 对象。这些对象是数据库系统的核心资源，依据迁移特征，通常可分为：

- **简单迁移对象**：从语法角度考虑，这类对象通常符合 SQL 标准。换句话说，各个数据库厂商提供创建和修改这些对象的 DDL 语句是相互兼容的。因此，一般可利用迁移工具对它们进行自动迁移。这类对象主要包括用户表、各种约束、索引、GRANT 权限和视图等。其中，各种约束包括外键、缺省、检查和唯一性约束等。
- **复杂迁移对象**：对这类对象，一般各个数据库厂商都采用各自特有的实现方式，没有遵循统一的标准，因此在异构数据库之间迁移这类对象很难采用自动化方式，只能采用手动或半自动化方式实现。这类对象主要包括序列、同义词、数据库链接、触发器、函数、存储过程和包等。

那么，什么是数据库对象移植呢？数据库对象移植是指对 Oracle 数据库系统的简单对象和复杂对象的移植。其中，简单对象移植可通过 EasyTransfer Tool 工具进行自动迁移，而复杂对象只能采用人工方式迁移。

3.1.3 Oracle 数据迁移

使用 EasyTransfer Tool 将 Oraacle 数据迁移到 KingbaseES 中。

3.1.4 应用程序移植

在完成数据库对象迁移以后，才可开始迁移应用程序，主要原因是：在用程序中，可能会访问和操作前面迁移的数据库对象。

应用程序移植是指对 Oracle API 方式或嵌入式 SQL 方式的应用程序的移植。它主要包括接口驱动程序和连接方法的移植，以及 Oracle 扩展或私有的、且 KingbaseES 未兼容的 API 移植。通常，该项任务的工作量较少。

在实际应用中，通常应用程序移植与移植系统测试与调试交叉进行。

3.2 常用移植方法

目前，Oracle 数据库移植主要包括两种方法：自动迁移和手动迁移。在实际应用中，二者应取长补短、相互补充、共同完成整个迁移任务。

3.2.1 自动迁移

KingbaseES EasyTransfer Tool 是自动化迁移工具。利用该工具可实现 Oracle 数据库的户表、各种约束、Grant 权限、索引和视图等数据库对象的自动迁移。但是，对于 PL/SQL 存储过程等较复杂的数据库对象，不能使用该工具，而只能采用手动迁移方式。

详细信息参照：第三章迁移工具 EasyTransfer Tool

3.2.2 手动迁移

针对自动迁移无法完成的内容，如自定义函数和存储过程等，则可通过手动方式迁移。此外，在迁移中，还可借助 PL/SQL Developer、Oracle SQL Developer 等辅助工具加速迁移。

手动迁移的基本步骤如下：

- 以 SQL 脚本方式导出迁移的 Oracle 数据库对象并保存。

  （可以使用第三方数据库连接工具导出：例如 Navicat Premium）
- 使用 KingbaseES 查询分析器打开保存的 SQL 脚本文件。
- 对不能正常执行执行的 SQL 脚本，若 KingbaseES 存在兼容的语法或功能，则应修改为对应的兼容方式；否则，应及时与 KingbaseES 支持工程师咨询迁移方法。
- 在 KingbaseES 查询分析器中调试修改的 SQL 脚本。在实际应用中，修改和调试 SQL 脚本工作一般交叉进行。
- 涉及到 copy 语句时还需要注意 client_encoding

在使用 Copy 时（server_encoding 为 GBK，client_encoding 为 UTF-8）需要在导出和导入数据之前 *set client_encoding=GBK;* 否则可能出现串行或者乱码

3.3 关键移植步骤

作为一个典型的项目过程，Oracle 数据库移植应具有健全的项目团队和全面细致的的项目执行过程。通常，移植一个 Oracle 数据库主要包括以下步骤：

- 确定移植目标
- 评估移植任务
- 组建移植团队
- 准备迁移环境
- **迁移 Oracle 数据库**
- 移植系统测试与调试

这些步骤指之间的关系是：前四个步骤是迁移前的准备工作，这些准备工作是确保后续 Oracle 移植顺利进行的前提条件，而最后一步是保证最终移植系统正确性和可用性的关键步骤。

下面，分别对上述各个步骤进行详细说明。

3.3.1 确定移植目标

开始迁移前，应根据用户的实际需求，确定移植目标。这些目标诸如：

- 迁移 Oracle 数据库的规模。
- 迁移 Oracle 数据库对象的种类和特征，如简单和复杂迁移对象所占比例等。
- 迁移的难易程度，如是否迁移大对象，是否迁移大量约束等。
- 迁移的工期要求。
- 对目标系统的技术指标要求，诸如平台、版本、应用编程接口、工具、可用性、安全性和性能指标要求等。

明确移植目标以后，则可开始移植任务评估。

3.3.2 评估移植任务

当计划把一个 Oracle 数据库系统移植到 KingbaseES 环境时，如果一头扎进去，不做评估或评估不充分的话，那么这个移植工作恐怕会有太多的潜在风险等着移植工程师，他们也无法回答领导和用户对移植时间的追问，一直被动的颠簸在风浪的漩涡里，而不知道岸边究竟在哪儿！因此，移植前对移植的可行性、工作量、难易程度和工作进度等进行充分评估是非常必要的。

通常，移植评估主要包括以下内容：

- 移植技术指标，如移植业务压力和性能指标等。
- 移植数据规模，如移植各类数据库对象的数量，PL/SQL 程序的规模等。
- 移植中 KingbaseES 不支持功能的种类和数量。
- 移植的约束种类和数量。
- 移植过程中可能遇到的其他问题。

在 Oracle 移植中常用的评估模板如下表所示：

表 3.3.1: 移植评估的数据库/应用概况模板

项目	描述	备注
Oracle 数据库版本	8.1.7.4	
操作系统版本	Winodws 2000/2003 Server	
服务器型号	联想/SUN	
CPU 配置		
内存（RAM）		
磁盘（Disk Profile）		
服务器个数（# of Servers）	1 或 2	
用户数/天（# Users/Day）	几十/天	
事务量/天（# Transactions / Day）		
当前数据库大小	几个 GB	
数据库增长速率（#GB/month）		
目标用户（Schema）		
应用方式（OLTP/OLAP）	OLTP	
应用服务器（中间件）	无	
客户端应用类型（C/S，B/S）	C/S	
客户端应用编程语言	Delphi7	
客户端应用连接接口	ODAC/ADO	
是否深入的 SQL 应用	无	
监控工具	无	
备份方式	Exp/imp	
其它工具（备份软件等）	无	
高可用要求	较高	
高可用配置方案	VCS 或单机	

表 3.3.2: 移植评估的移植报告总结模板

项目	描述
移植分析日期	**20111009** 下午
移植分析人员	朱河龙
KingbaseES 版本	
Oracle 版本	8.1.7.4
Oracle Schema	
Oracle DB Size (GB)	几个 GB
Oracle Schema Size (MB)	几个 GB

表 3.3.3: 移植评估的对象统计模板

类型	小计	备注
Function	7	较少用
Index	有	
LOB	有	最大到几十 MB，主要是照片、word、视频（较少）
Materialized View	有 >10	
Pro*Cedure	25	
Sequence	有 >10	
Table	1660	约束较多
Table Partition	无	
Trigger	<30	
JOB	无	
Package	无	
Package Body	无	
Type	无	
View	>200	
Synonym	>300	
对象共计		

表 3.3.4: 移植评估的约束统计模板

类型	小计	备注
CHECK OR NOT NULL		
FOREIGN KEY		
PRIMARY KEY		
UNIQUE KEY		
OTHER		
约束共计		

表 3.3.5: 移植评估的其它方面模板

特性	小计	备注
数据压缩	无	
索引组织表	无	
维度（Dimensions）	无	
物化视图	无	
存储概要	无	
高级队列	无	
空间数据管理	无	
全文搜索	有	
数据库链接	无	
数据复制	无	
RAC	有	
逻辑 standby	无	
物理 Standby	无	
自动存储管理 ASM	无	
自动工作负载信息库 AWR	无	
共计		

3.3.3 组建移植团队

任何一个高效、成功的项目都应具备一个健全和良好的团队，Oracle 数据库移植也不例外。如果没有这样团队互相配合和支持，那么 Oracle 数据库移植将可能存在巨大的风险。所以，组建一个高效的移植团队是非常必要的。

那么，移植团队的组成人员应具备哪些条件呢？他们应至少具备以下的知识与技能：

- 熟悉 Oracle 和 KingbaseES 的 SQL 语言和 PL/SQL 语言特性，以及相关的 KingbaseES Oracle 兼容特性。
- 熟悉 Oracle 和 KingbaseES 的各种应用编程接口，以及相关的 KingbaseES Oracle 兼容特性。
- 熟悉 Oracle 和 KingbaseES 的相关客户端工具，以及这些工具间的相同点和异同点。

由这些优秀人员组建的团队是高效移植 Oracle 数据库的可靠保障。

3.3.4 准备迁移环境

在上述步骤完成以后，移植工程师应开始准备迁移环境了，这些准备工作诸如：

3.3.4.1 部署源和目的数据库服务器

部署源和目的数据库服务器应遵循以下原则：

- 源和目的数据库服务器的 CPU、内存、网络环境等硬件应尽量采用较高的配置。
- 如果移植的 Oracle 数据库系统规模较大，如超过 1GB，则建议把 Oracle 和 KingbaseES 分别部署在不同的物理机器上。
- 为确保迁移效率，应尽量把 KingbaseES 和 Oracle 服务器部署到同一局域网内。

3.3.4.2 获取并安装必要的软件

迁移前应获取并安装如下软件：Oracle 数据库系统、KingbaseES 数据库系统、PL/SQL Developer、JDBC 和 ODBC 驱动程序、C 语言开发工具、OCI 软件、DCI 软件、TPC-C 测试工具、LoadRunner 等。

如果迁移数据规模较大，建议对安装的 KingbaseES 数据库服务器进行适当的优化，如增大 *shared_buffer* 大小、预先创建较大的日志文件，预先申请足够的表空间数据库文件等。

完成上述准备工作后，移植工程师便可开始 Oracle 数据库移植工作了。

3.3.5 数据库、用户和模式迁移

数据库、用户和模式迁移主要包括以下内容：

- 获取源 Oracle 数据库的 IP 地址、实例名、网络服务端口号、用户名/密码等信息。
- 在目的 KingbaseES 数据库上，使用 ISQL 或 JSQL 工具上执行如下操作：
- 创建与源 Oracle 用户同名的用户，例如创建与 Oracle 同名的 scott 用户。
- 创建与源 Oracle 同名的数据库，例如创建与 Oracle 同名的 ORCL 数据库，它的属主为 scott。
- 创建与源 Oracle 同名的模式，例如创建与 Oracle 同名的 scott 模式，它的属主为 scott。在 KingbaseES 的 Oracle 兼容模式下，如果通过查询分析器或 ISQL 工具创建同名用户，则省略此步。但是，如果通过企业管理器创建同名用户，则此步则不能省略。

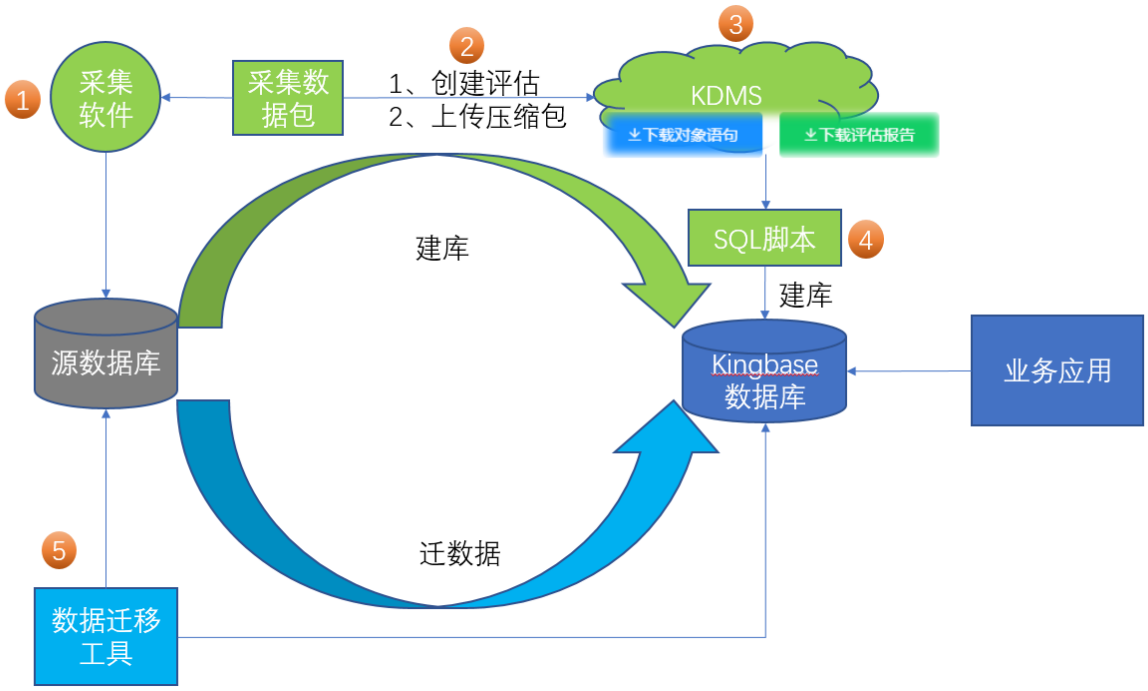
3.3.6 数据库对象迁移

迁移数据库、用户和模式以后，则可开始数据库对象迁移。

这部分内容可以使用人大金仓的在线 KDMS 系统进行，也可以使用 EasyTransfer Tool 完成。

- 使用 KDMS 系统进行迁移

  使用 KDMS 时，首先使用 Oracle 的工具导致数据对象定义，包括存储过程等定义，之后将包上传到 KDMS 产品，将自动转换为可以在 KingbaseES 上执行的脚本，之后再 ES 中执行脚本就可以完成本个步骤。



- 使用 EasyTransfer Tool 进行迁移

  包括以下内容：

  - 简单对象迁移：使用迁移工具 EasyTransfer Tool 自动迁移 Oracle 的各种简单对象，这些对象如表、数据、约束、索引和视图等。
  - 复杂对象迁移：迁移完成简单对象以后，采用手动方式迁移 Oracle 的各种复杂对象，这些对象如同义词、物化视图、数据库链接、触发器、函数、存储过程和包等。它的的主要步骤如下：
    - 以 SQL 脚本方式导出源 Oracle 数据库的复杂对象并保存。导出对象时，可使用 Oracle 的 imp/exp、SQL Plus、SQL Developer 等工具。
    - 打开导出 SQL 脚本文件。
    - 对导出文件中的每个对象依次在 KingbaseES 查询分析器上重新创建，并建议如下：
      * 应先创建同义词的移植，这样做的主要原因是：视图等其他对象可能要依赖同义词。此外，KingbaseES 不支持同义词，对于表的同义词，可以使用 view 进行移植。
      * KingbaseES V6 及其以前版本不支持物化视图，此时可把物化视图迁移为普通视图，但应去掉 Oracle 有关刷新物化视图的 SQL 语句（注：现有 KingbaseES V7 版本已经支持物化视图，无需修改）。
      * 虽然 KingbaseES 与 Oracle 数据库链接的语法不相同。
      * 触发器、函数、存储过程和包的脚本可能需适当修改后方能使用。但移植这些对象一般只需修改较少内容。
      * 迁移开始时，可使用 UE 或 KingbaseES 查询分析器进行迁移。一旦发现迁移规律以后，则可采用批量替换方式迁移。
      * 对极个别无法移植、或不易绕过、或 KingbaseES 不兼容的 Oracle 特性，请及时咨询 KingbaseES 支持工程师。
      * 修改完成后，千万别忘记备份！

具体请参考第二章，下面给出两个从 Oracle 迁移到 KingbaseES 环境的示例。

例 5-1：Oracle 简单函数迁移的示例。

Oracle 的函数	KingbaseES 的函数
CREATE OR REPLACE FUNCTION get_name_version *(v_name***varchar***, v_version varchar)* *RETURN varchar IS* *res***varchar(100)***;* *BEGIN* *IF v_version IS NULL THEN* *RETURN v_name;* *END IF;* *res := v_name || '/' || v_version;* *RETURN res;* *END;*	*CREATE OR REPLACE FUNCTION get_name_version* *(v_name***TEXT***, v_version***TEXT**) *RETURNS varchar IS* **DECLARE** *res***TEXT***;* *BEGIN* *IF v_version IS NULL THEN* *RETURN v_name;* *END IF;* *res := v_name || '/' || v_version;* *RETURN res;* *END;*

例 5-2：Oracle 匿名块迁移的示例。

Oracle 的匿名块	KingbaseES 的匿名块
DECLARE **TYPE r_cursor IS REF CURSOR;** **c_emp r_cursor;** **TYPE rec_emp is RECORD** **(ename VARCHAR2(20),** **sal NUMBER(6));** **er rec_emp;** *BEGIN* *OPEN c_emp FOR SELECT ename,sal FROM emps;* *LOOP* *FETCH c_emp INTO er;* *EXIT WHEN c_emp%NOTFOUND;* **DBMS_OUTPUT.PUT_LINE(er.ename || ' - ' || er.sal);** *END LOOP;* *CLOSE c_emp;* *END;*	*DECLARE* **c_emp REFCURSOR;**--直接定义为 *refcursor* 类型 **er RECORD;**--直接定义为 *record* 类型 *BEGIN* *OPEN c_emp FOR SELECT ename,sal FROM emp;* *LOOP* *FETCH c_emp INTO er;* *EXIT WHEN c_emp%NOTFOUND;* --使用 *raise* 语句替代 *dbms_output.put_line* **RAISE NOTICE '% - %', er.ename, er.sal;** *END LOOP;* *CLOSE c_emp;* *END;*

3.3.7 数据迁移

KingbaseES 数据迁移工具 EasyTransfer Tool 使用插件方式动态加载待迁移的数据库访问接口，方便用户定制和使用。

KingbaseES 数据迁移工具 EasyTransfer Tool 支持同、异构数据源之间的数据迁移

同构数据源间数据迁移：支持 Kingbase V7 到 Kingbase V8R6 的数据迁移。

异构数据源之间的数据迁移：支持 Oracle9i、10g、11g、12c 到 Kingbaselversionl 的数据前迁移。

KingbaseES 数据迁移工具 EasyTransfer Tool 支持结构迁移、支持全量数据迁移、支持列名映射，支持数据迁移过滤，在配置数据任务时，可以对迁移的表配置 where 条件、通过匹配的 where 条件过滤需要迁移的数据。

3.3.7.1 迁移前准备

在使用 EasyTransfer Tool 迁移 Oracle 数据库之前，应先做如下准备工作：

3.3.7.1.1 获取 Oracle 数据库的相关信息

迁移前，应获取源数据库 Oracle 服务名及迁移的数据规模信息。其中，前者用于 PL/SQL Developer 工具的登录操作，后者用于估算数据迁移时间和设计迁移方案。

1) Oracle 数据库基本信息

   获取源 Oracle 数据库的：

   a. IP 地址；

   b. 实例名；

   c. 网络服务端口号；

   d. 用户名/密码。

   在目标 KingbaseES V8R6 上：

   a. 创建与源 Oracle 用户（如 scott）同名的用户（scott）；

   b. 创建与源 Oracle（如 ORCL）同名的数据库（ORCL），属主为 scott；

   c. 创建与源 Oracle（与用户名相同 t）同名的模式 scott，属主为 scott。Oracle 兼容模式下，使用查询分析器或 isql 工具创建用户 scott，则省略此步。使用企业管理器创建用户 scott，此步则不能省略。

2) 大小写是否敏感

   oracle 默认大小写不敏感，而 Kingbaselversionl 默认大写，可以在初始化数据库的时候进行修改。

```
./initdb -D /home/kingbase/Kingbase/ES/|version| /data -U SYSTEM -A trust--
↪locale=C--case-insensitive
```

3) 查询 Oracle 数据库编码方式

```
 select userenv('language') from dual;
        USERENV('LANGUAGE')
    SIMPLIFIED CHINESE_CHINA.ZHS16GBK
【Kingbase 初始化设置编码方式】
 --encoding=GBK（支持 GBK UNICODE ASCII）
```

4) 查看表数据量大小

   查看当前用户在 Oracle 中的表大小，按从大到小排序（单位 GB）

```
select segment_name,bytes/1024/1024/1024 from user_segments where segment_
↪type='TABLE' order by bytes desc ;
XFJXX 16.046875
XFRXX 7.779296875
PCK       7.4375
BLFSXX    5.0625
XFSXXX    2.3125
DFGZXX    1.3359375
FJB       0.53125
TSJXX 0.078125
```

5) 在 oracle 上查看有哪些表为分区表 (分区表不能使用 EasyTransfer Tool 进行迁移)

```
select * from user_tables where partitioned='YES';
```

迁移参照：2.4.11 水平分区

6) 检查数据库日期格式

时间的默认格式为：ISO, MDY

在配置文件中添加：datestyle ='ISO,YMD' 修改为年月日的格式（99 会改为 1999）

在信访局迁移数据时遇到：服务器报错，迁移工具中断，迁移停滞

oracle 数据库中有日期"0099-09-30 00:00:00"，迁移工具输出为"99-09-30 00:00:00"，KingbaseES 中将 99 识别为月份报错：ERROR: date/time field value out of range

```
--即使没有报错也会出现错误
set ora_date_sytle = true;
CREATE TABLE T_DATE(COL DATE);
INSERT INTO T_DATE VALUES('11-10-10 10:10:10');
SELECT * FROM T_DATE;
     COL
---------------------
 2010-11-10 10:10:10
(1 row)
```

3.3.7.1.1.1 配置 KingbaseES 的 Oracle 兼容开关

根据实际情况，应对目的数据库 KingbaseES 进行适当的 Oracle 兼容配置。通常，应配置以下会话级兼容参数：

1）*char_default_type*：设定 *char* 类型字段默认单位是 *byte* 还是 *char*。此外，标识符最大长度以此值为单位。如果它为 *char*，则标识符最大长度为 63 个 *char*，否则为 63*byte*。

在 KingbaseES 系统参数 *char_default_type* 缺省值是"CHAR"，需要与待迁移的 Oracle 相同。

Oracle 字符类型的 bytel char 属性的默认值是由 Oracle 提供的数据库参数 *NLS_LENGTH_SEMANTICS* 决定的，可通过下方语句进行查询：

```
select value from nls_database_parameters where parameter = 'NLS_LENGTH_
↪SEMANTICS';
 VALUE
BYTE
```

如果未修改可能会出现：迁移 char 类型时，由于数据库存储的类型不同，导致迁移的数据存在多余空格的情况。

2）*search_path*：模式搜索路径。例如 *search_path* 为 *$USER,SCOTT,PUBLIC* 时，系统将首先搜索与登录用户同名的模式对象，然后搜索 *SCOTT* 模式对象，最后搜索 *PUBLIC* 模式对象。

3）*default_with_oids*：*OID* 伪列开关。KingbaseES 的 *OID* 伪列可兼容 Oracle 的 *ROWID* 伪列。因此，如果 Oracle 移植对象有 *ROWID* 伪列，则建议用 *OID* 伪列替代。

3.3.7.1.1.2 移植数据库、用户和模式

在目的数据库 KingbaseES 上创建与源数据库 Oracle 同名的用户、数据库和模式，并且授予新建用户具有使用该数据库和新建模式的所有或适当的权限。另外，所创建数据库的字符集应与 Oracle 数据库字符集一致。如果 KingbaseES 已有同名数据库，则登录该数据库后，只需创建同名用户和属主为该用户的同名模式。

3.3.7.1.1.3 配置 JDBC 数据源

配置 KingbaseES 和 Oracle 的 JDBC 数据源，并设置相关的连接信息。

3.3.7.1.1.4 配置目的库 KingbaseES 性能参数

为了提高迁移速度，应对目的库 KingbaseES 进行性能优化配置。

例如：

1) 根据迁移数据规模的大小，迁移前可预先创建适当大小的的数据和日志文件。

   开始迁移之前根据待迁移数据库的大小，保证 KingbaseES 数据目录所在位置有足够的空间。

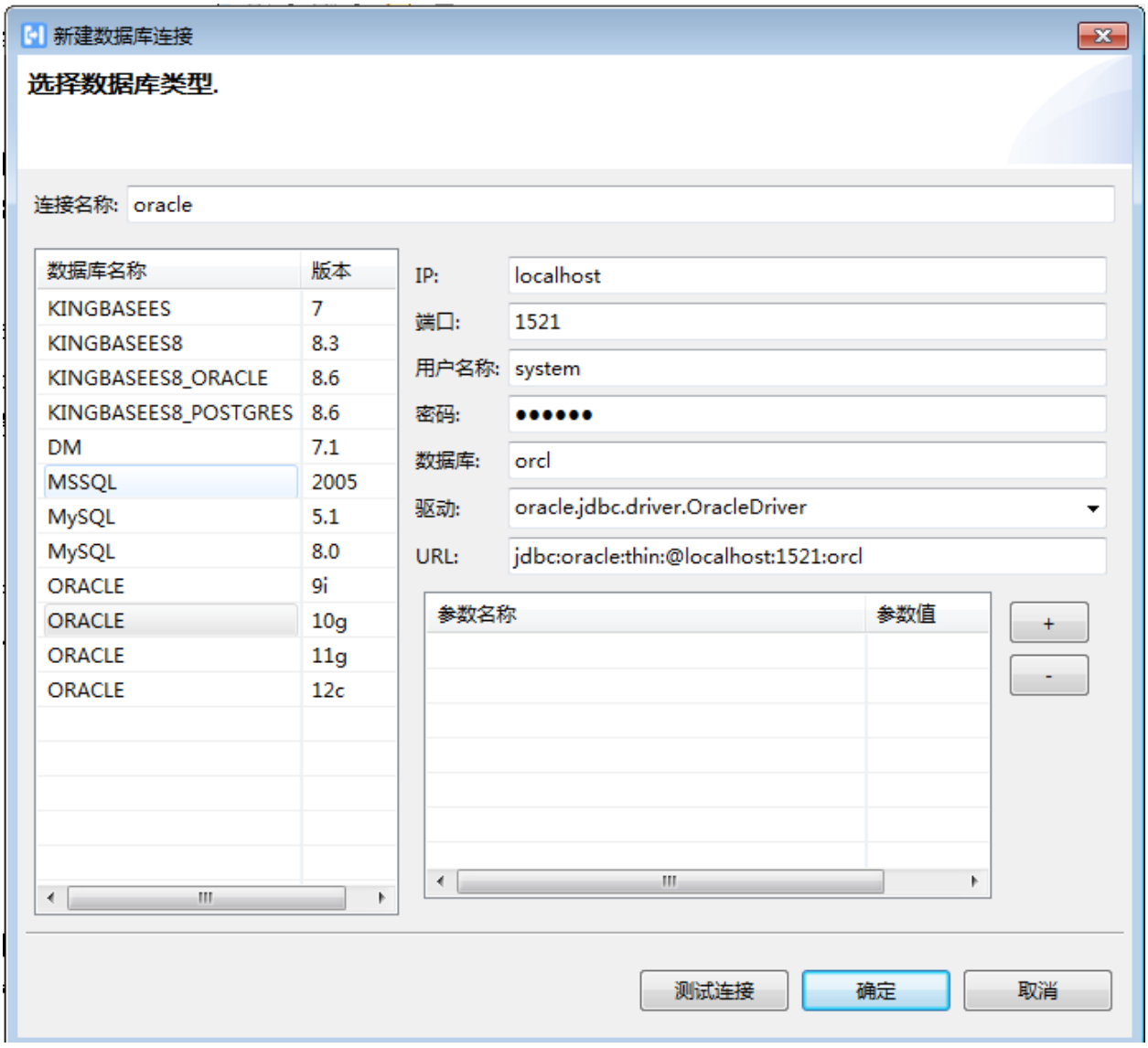
2）根据 KingbaseES 服务器硬件配置的实际情况调整 *shared_buffers* 大小，默认是 128M，建议调整为内存的 1/4 大小。

3.3.7.2 数据迁移

在完成上述准备工作以后，用户可进行数据迁移，具体步骤如下：

- **创建数据库连接**

  分别创建源库、目标库数据库连接。创建数据库连接界面如下，填写数据源信息，保存。

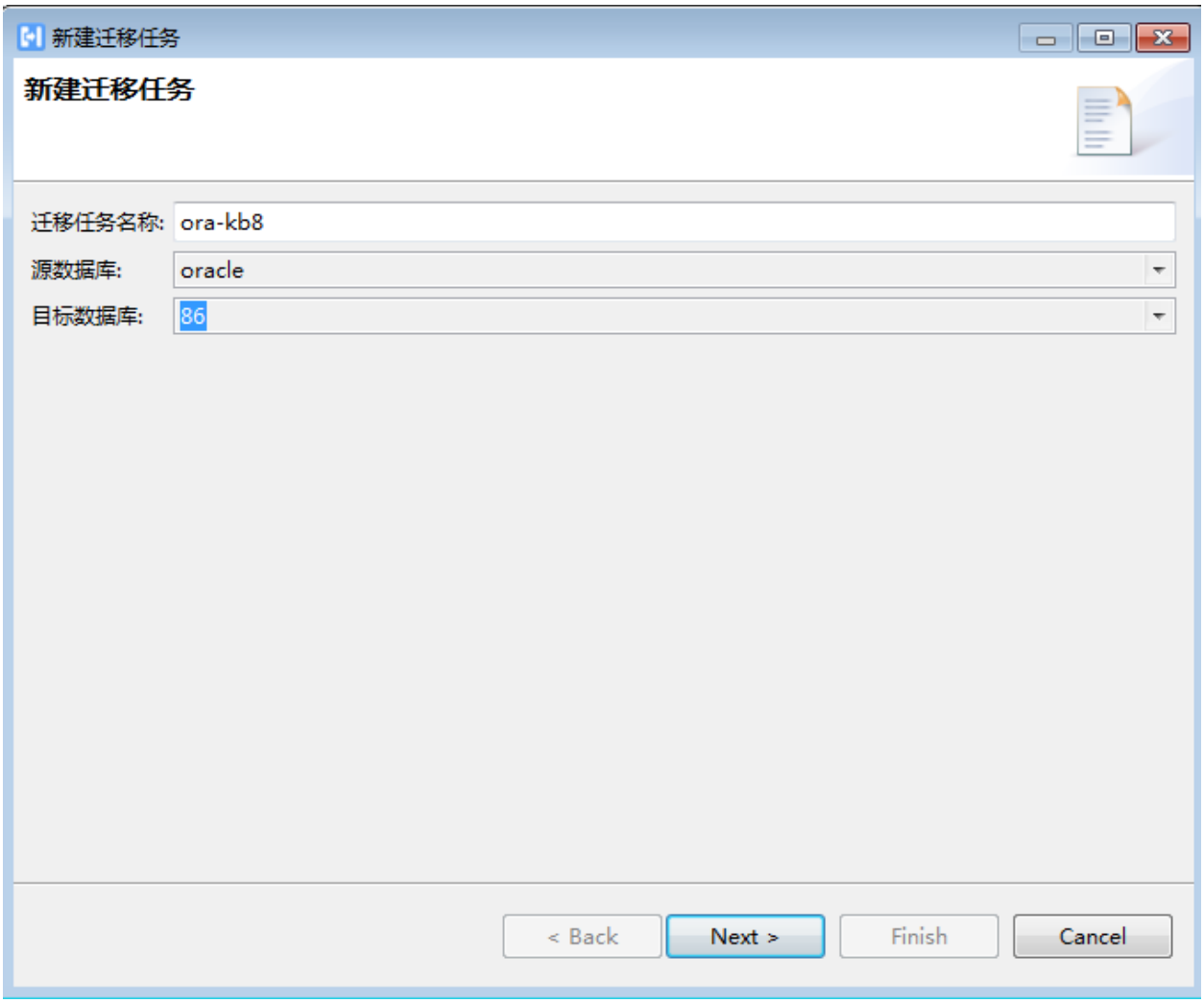


- 新建迁移任务

  迁移工具采用向导页的方式指导用户新建迁移任务，简单易用，用户依次配置”选择数据源“-选择模式”-“选择对象”-“迁移配置”-“数据类型映射”，即可快速配置一个迁移任务。

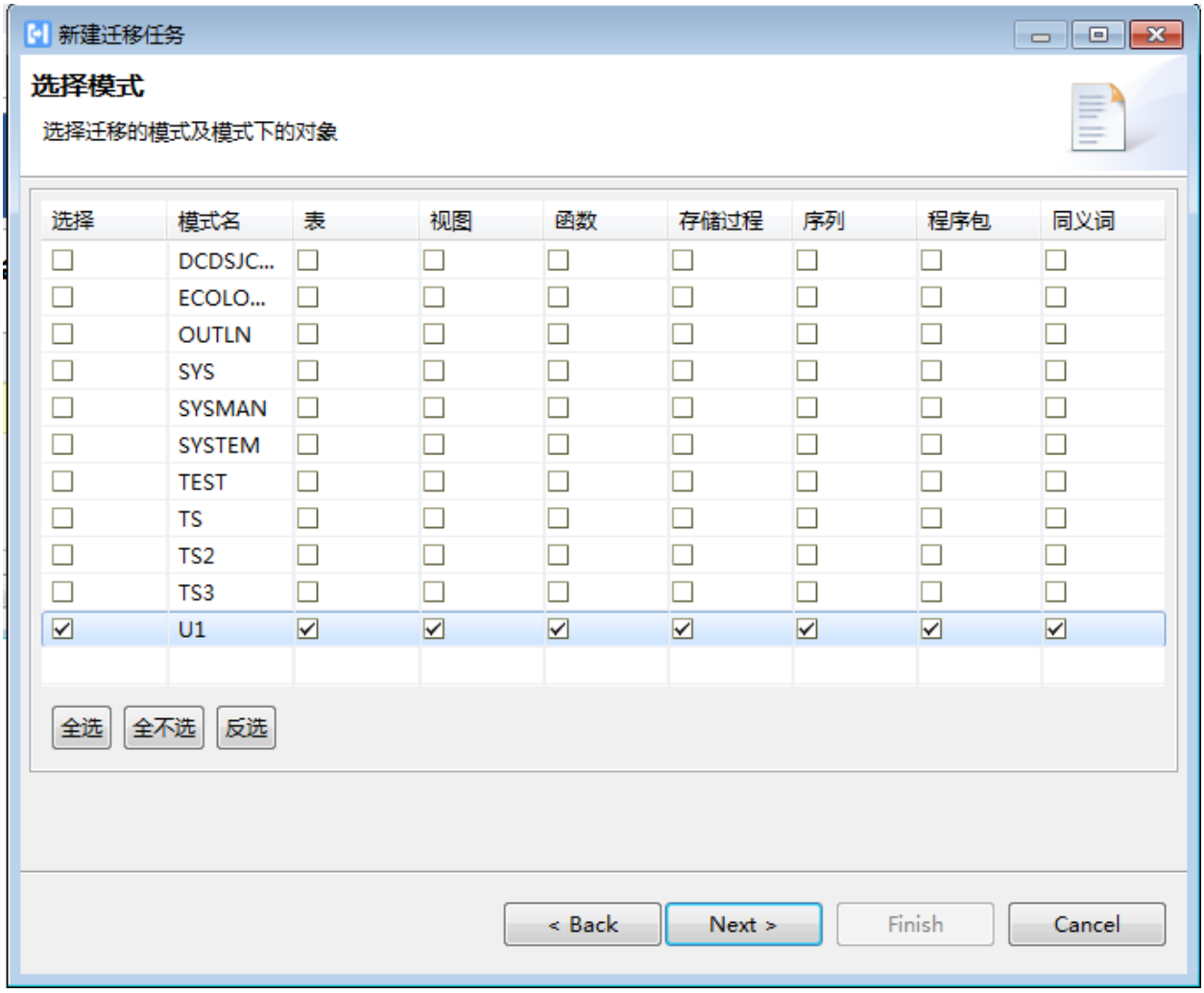
  1) 选择数据源

     选择源库连接，目标库连接。



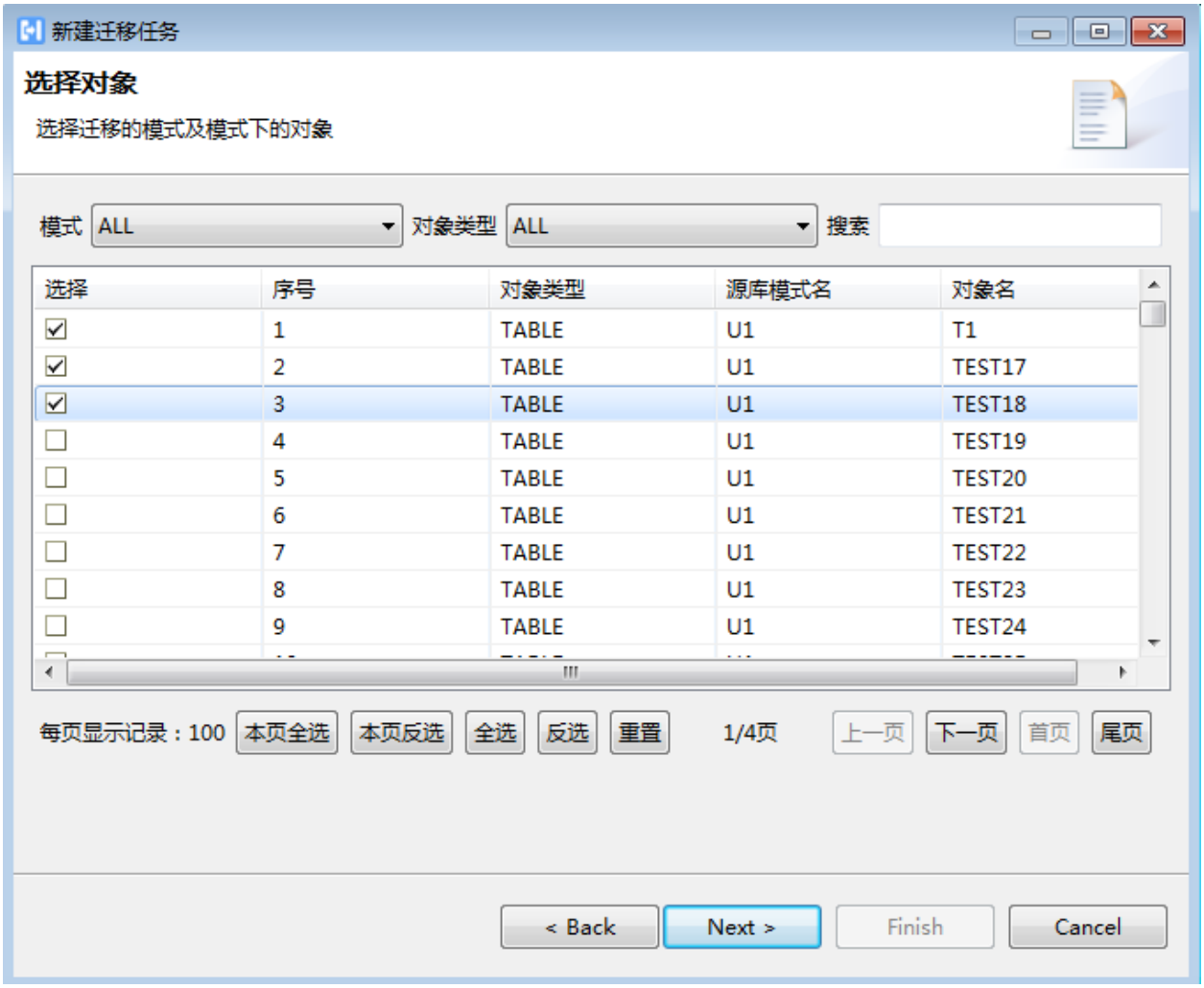
2) 选择模式

勾选要迁移的模式，以及要迁移的数据库对象类型。



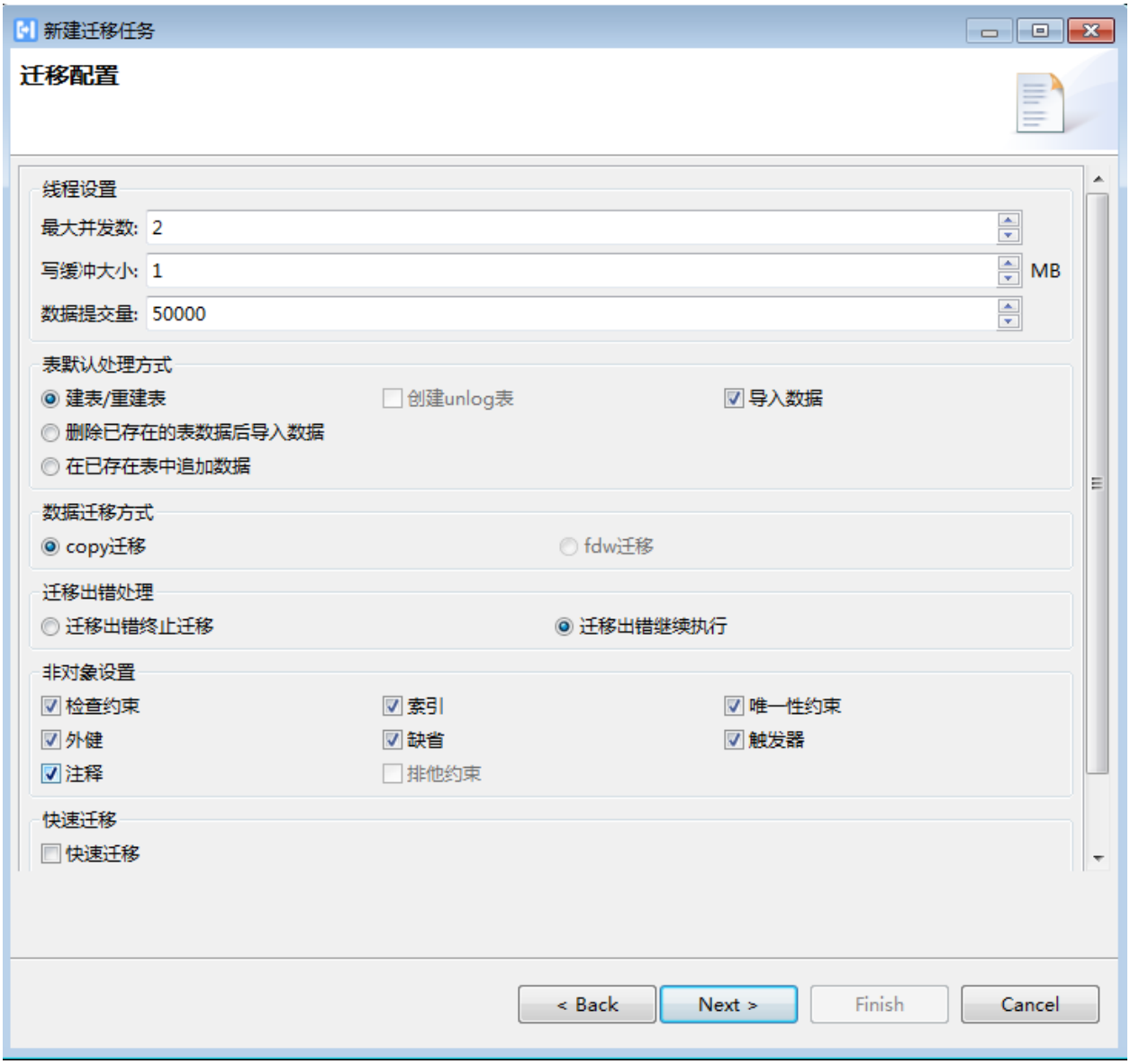
3) 选择对象

勾选要迁移的数据库对象。



4) 迁移配置

进行迁移配置，该页面可配置并发数，表默认处理方式，迁移出错处理方式，非对象设置，快速迁移设置。



a. **”最大并发数”**：执行迁移任务线程数

b. **”写缓冲大小”**：迁移数据时，每个线程一次写入目标数据库数据量大小，单位 MB。内存使用量 = 最大并发数 * 写缓冲大小

c. **”数据提交量”**：指进行迁移数据步骤时，向目的端数据库一次提交的记录个数，默认是 50000。

d. **建表/重建表并导入数据**：将原来的表删除，并创建新表，其数据类型是根据类型映射表转换后的数据类型，数据是源表中的数据时，表示迁移正确。

e. **删除已存在的表数据后导入数据**：保留目的库中的同名表，将该表中数据删除，迁移源表中的数据到该表。以此种方式迁移后的表，其数据类型是原来目的表的数据类型，表中数据是源表中的数据。

f. **追加数据**：保留目的库中的同名表，及表中的数据，迁移源表中的数据到该表，如果源表中记录和目的表中的原有记录有约束冲突，则不迁移这些记录。以此种方

式迁移后的表中包含了源表和目的表中所有数据（不包含有约束冲突的数据）。

g. **“迁移出错时终止迁移”**：当一次迁移任务中的任何一个表记录出错时, 迁移工具就会退出。要求迁移的数据准确率很高时，可以使用该选项；

h. **“迁移出错时继续执行”**：某个表出现迁移错误时不会影响其它迁移任务的执行。

i. **“copy 迁移 “**：

在目标数据库中会有多个连接同时对该表执行 copy 数据的行为。

j. **” fdw 迁移 “**：在目标数据库中会有多个连接同时对该表执行 insert into 数据的行，暂不支持。

k. **非对象设置:** 支持配置迁移非表对象，表示迁移任务是否迁移源数据库的此类对象，非表对象包括检查约束，索引，唯一性约束，外键，排他约束，缺省，触发器，注释。

l. **“快速迁移”**：对于数据库对象很多的场景（如 1000 张表），可勾选“快速迁移”，提高迁移速度。

5) 数据类型映射

可添加、删除、编辑数据类型映射关系。

- **执行迁移任务**

  打开迁移任务，可以看到任务概述，预览总体迁移信息，点击“执行”按钮，即可执行迁移任务。

- **查看迁移报告及问题处理**

  执行迁移任务完成后，会生成迁移报告。迁移报告有两种：文本报告和柱状图报告，用户可直观看到迁移结果

  控制台视图可以显示用户的操作日志和迁移日志，其中迁移日志可以持久化本地，用户可以在资源管理器中找到本次迁移的日志文件。文件中包含创建表的 sql、迁移过程、创建主键 sql 和所有非表对象创建的 sql 语句；还包括迁移异常信息。

  在迁移过程中一旦某个对象创建失败，DTS 会将该对象的创建 sql 保留到本次迁移任务文件夹下的 ErrorScripts 目录中，用户可以手动修改后通过 ksql 或者对象管理工具手动执行。

3.3.7.3 迁移常见问题及应对措施

EasyTransfer Tool 数据迁移中的常见问题及应对措施如下：

- **复杂表和约束迁移失败**

  当迁移 Oracle 的复杂表和约束时，如果总是遇到错误，则可能是因为 KingbaseES 自带的 Oracle JDBC 驱动程序版本与 Oracle JDBC 版本不兼容。此时，用户可用 Oracle 数据库安装目录 JDBC/lib 文件夹下的 JDBC 驱动程序，替换 *$KINGBASE_HOME/plugins/database/Oracle* 目录下的 JDBC 驱动程序，然后重新迁移。

  在 KingbaseES 之间可以迁移的数据库对象包括：

  表、视图、存储过程、函数、程序包、触发器、索引、主外键、非空约束、检查约束、唯一约束、缺省值。注意: 若目的库中存在与要迁移的存储过程、函数、触发器或程序包相同的名称, 不支持迁移这样的存储过程、函数、触发器或程序包.。

  其他数据库向 KingbaseES 可以迁移的数据库对象包括：

  表、索引、主外键、非空约束、检查约束、唯一约束、缺省值。需要注意的是：不支持函数索引和表达式索引的迁移；不支持聚集索引的迁移；不支持含有函数或者表达式的默认值的迁移。

- **FAQ**：

  Q：为什么在迁移时使用【追加数据】的迁移方式会导致迁移失败？

  A：EasyTransfer Tool 为了提高迁移速度，在【配置】对话框中，将【一次批量提交记录个数】设置为 50。在追加数据时，如果在一次提交的 50 条记录中，如果有某些记录的主键值与已存记录的主键值相同，会导致 50 条记录全部迁移失败。因此在使用【追加数据】的迁移方式时，建议将【一次批量提交记录个数】设置为 1，这样可以使主键值重复的记录迁移失败，主键值不重复的记录迁移成功。

  Q：将其它数据库的表导入到 KingbaseES 数据库时,KingbaseES 数据库并没有与要导入的表重名的表, 为什么迁移日志会报告表重名错误, 从而导致迁移失败?

  A：在 KingbaseES 中同一个模式下表、视图、序列都使用同一个命名空间，即表、视图和序列之间不可以重名，否则后创建的表会将覆盖已经存在的表。

- **不支持 Oracle 特性问题**

  对于 KingbaseES 不支持的 Oracle 特性，请及时告知 KingbaseES 支持工程师进行处理

3.3.8 应用程序迁移

在应用编程接口方面，KingbaseES 与 Oracle 兼容程度较高，所以，一般情况下，应用程序迁移比较容易。应用程序迁移通常应和移植系统测试同时进行。这样可及时修改测试过程中发现的问题。

3.3.8.1 概述

如何在一个应用程序中访问和操纵数据库呢？通常，可采用以下方式：

该方式通过数据库厂商提供的各种标准应用编程接口在应用程序中与数据库进行交互。常用的应用编程接口如 JDBC 和 ODBC 等。目前，大多数数据库厂商均提供很多标准的数据库 API 及其驱动程序。

在实际应用中，应首先加载驱动程序。加载成功后，利用 API 函数与数据库交互并完成对数据库数据的操作。

详细参考 客户端编程接口。

1) Windows 数据源配置

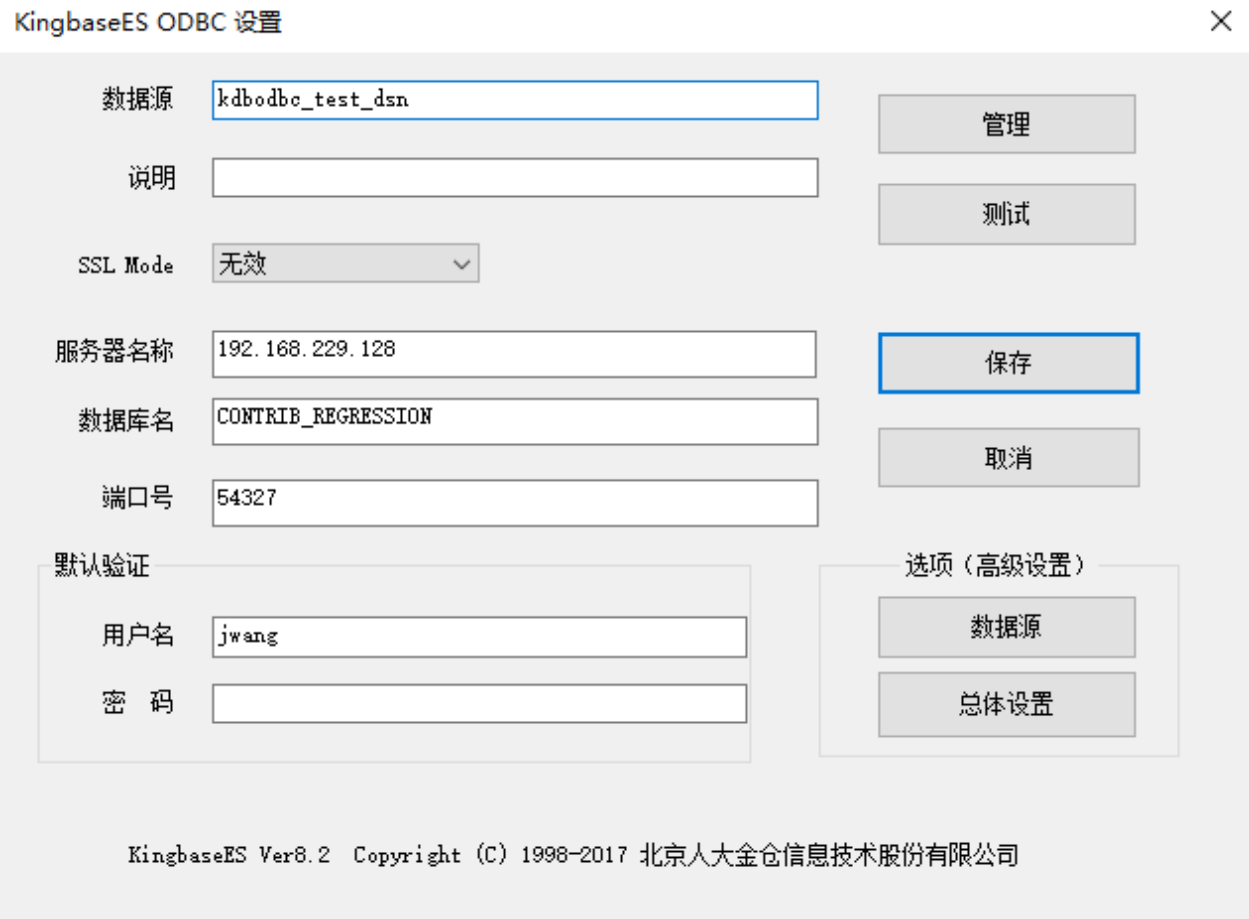


图 3.3.1: Windows 平台 ODBC 数据源配置 1

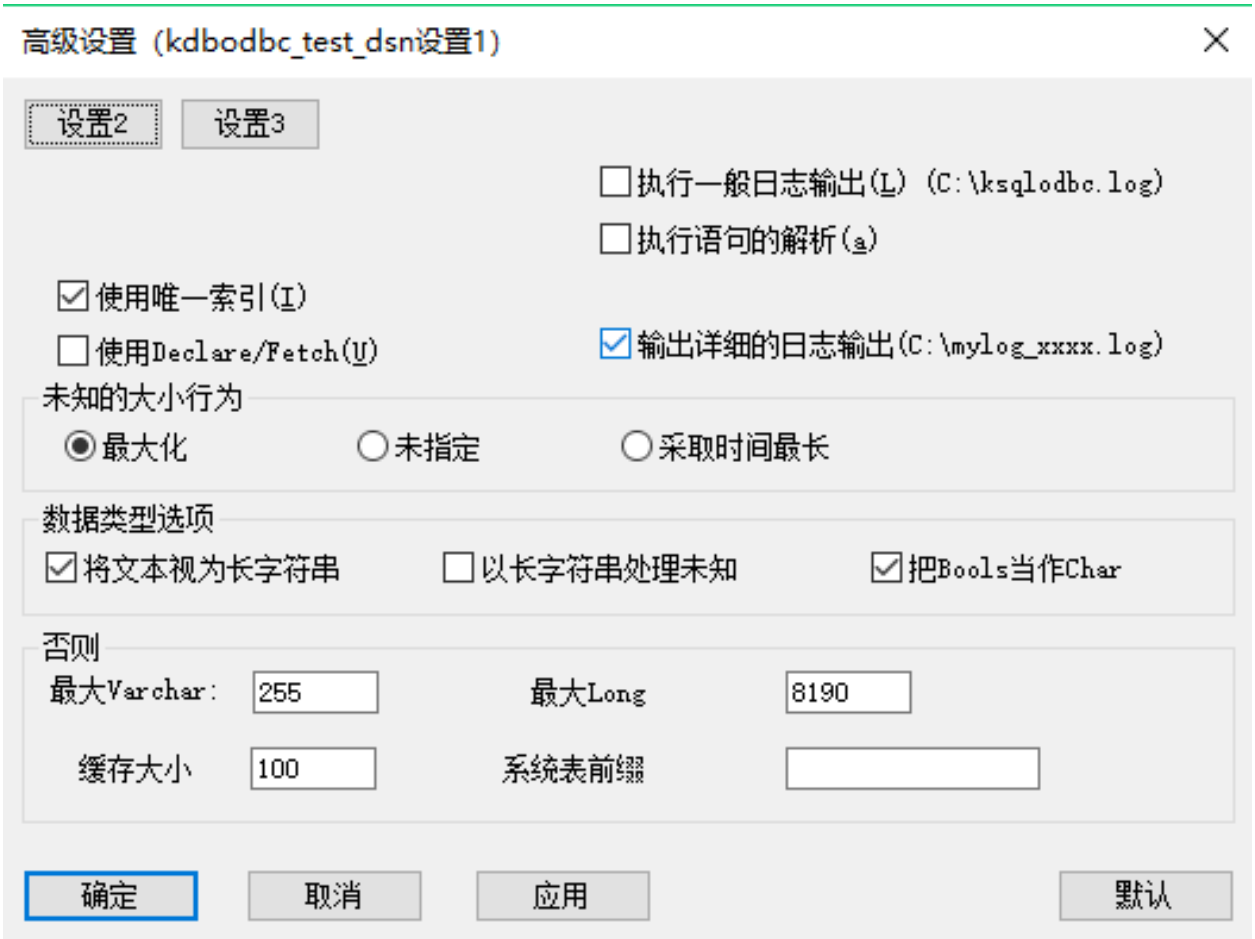


图 3.3.2: Windows 平台 ODBC 数据源配置 2

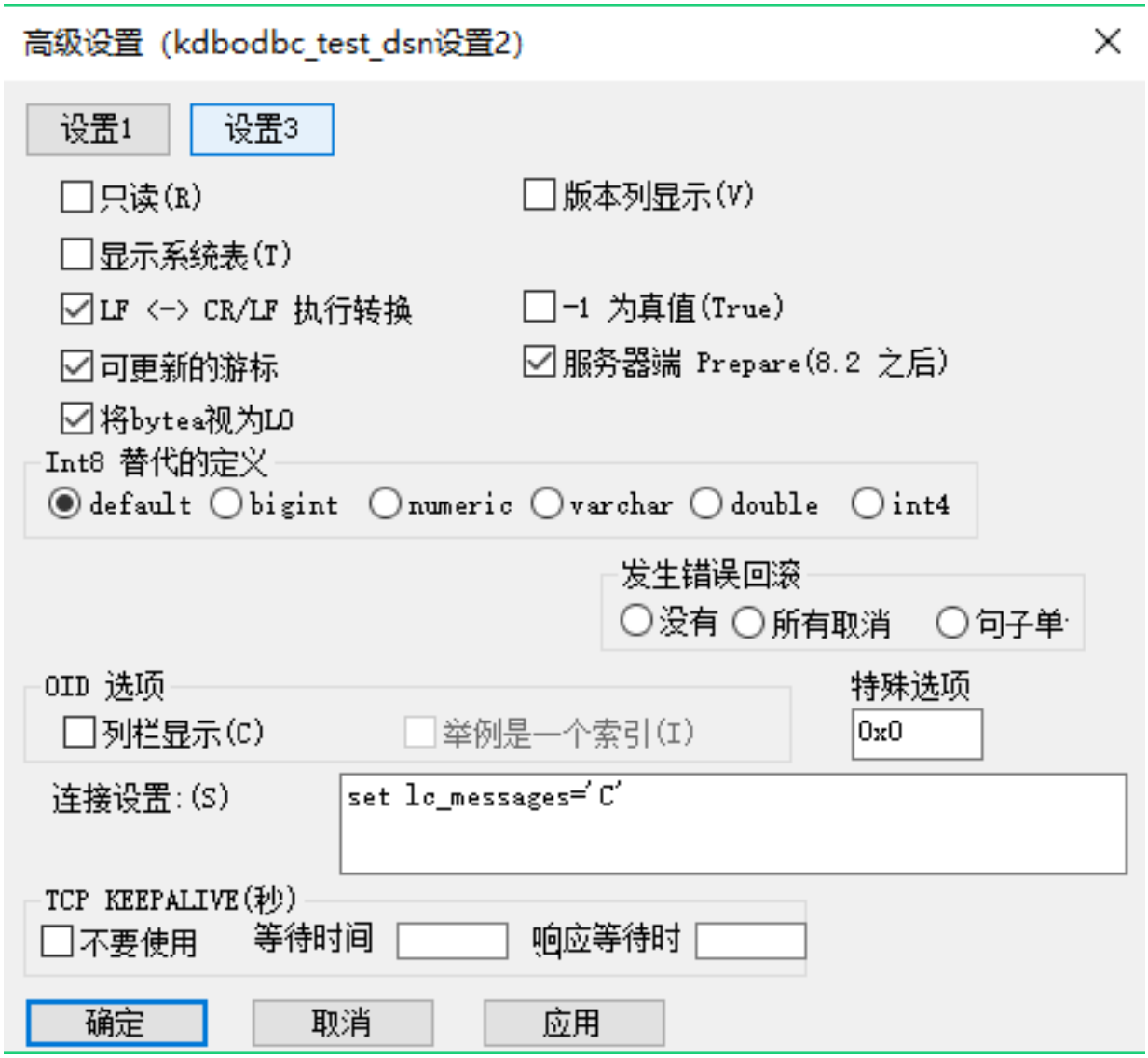


图 3.3.3: Windows 平台 ODBC 数据源配置 3

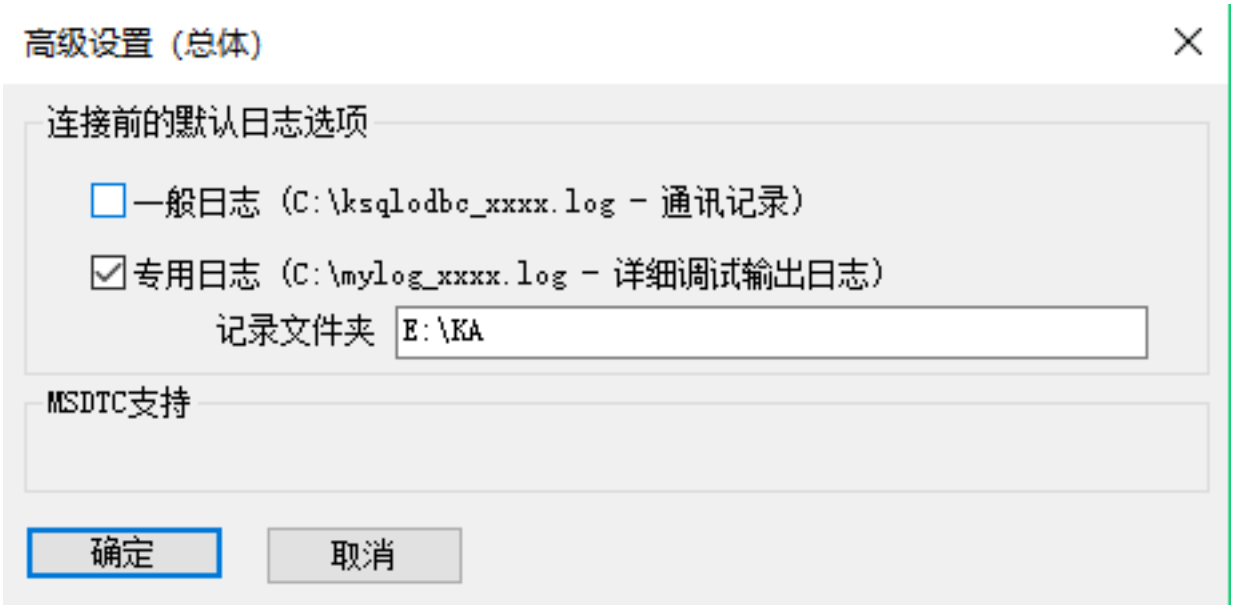


图 3.3.4: Windows 平台 ODBC 数据源配置 4

具体配置参数解释请参考 ODBC 指南。

2) Linux 数据源配置

首先检查 ODBC Driver 是否已经安装。在系统中找到 odbcinst.ini 文件，和/usr/bin/odbcinst 对应的 odbcinst.ini 在 /etc 目录下，和/usr/local/bin/odbcinst 对应的 odbcinst.ini 在 /usr/local/etc 目录下。在 odbcinst.ini 文件中查找 [KingbaseES 8 ODBC Driver] 这一项。

如果没有，则参考增加如下内容：

```
[KingbaseES 8 ODBC Driver]

Description = KingbaseES 8 ODBC Driver for Linux

Driver = /opt/Kingbase/Odbc/lib/kdbodbcw.so

增加 odbc.ini 文件，内容如下：

[kingbase]

Description = KingbaseES

Driver = KingbaseES 8 ODBC Driver

Servername = 127.0.0.1

Port = 54321

Username = SYSTEM

Password = MANAGER

Database = TEST
```

具体配置参数解释请参考 ODBC 指南

表 3.3.6: JDBC 的连接串比较说明

说明	**Oracle**	KingbaseES V8R6
JDBC 基本使用	--声明连接 Connection con; --加载驱动程序 Class.forName (" oracle.jdbc.driver.OracleDriver"); --连接串 String url = "jdbc:oracle:thin:@192.168.0.1:1521:databasename"; --获得连接 con= DriverManager.getConnection(url,user,pwd);	--声明连接 Connection con; --加载驱动程序 Class.forName("com.kingbase8.Driver"); --连接串 String url = "jdbc:kingbase8://192.168.0.1:54321/databasename"; --获得连接 con= DriverManager.getConnection(url,user,pwd);

其它问题请参考 JDBC 指南

表 3.3.8: Hibernate 的比较说明

说明	**Oracle**	KingbaseES V8R6
Hibernate 基本使用	定义 hibernate 配置文件，根据用户选择更改以下配置文件。 在 hibernate.properties 中增加如下声明： hibernate.dialectorg .hibernate.dialect .OracleDialect 在 hibernate.cfg.xml 中增加如下声明： <property name=" dialect" >org .hibernate.dialectOracleDialect </property> 在 persistence.xml 中增加如下声明： <property name =" hibernate.dialect" value ="org.hibernate. dialect.OracleDialect" />	定义 hibernate 配置文件，根据用户选择更改以下配置文件。 在 hibernate.properties 中增加如下声明： hibernate.dialectorg .hibernate.dialect .Kingbase8Dialect 在 hibernate.cfg.xml 中增加如下声明： <property name =" dialect" >org. hibernate.dialect. Kingbase8Dialect </property> 在 persistence.xml 中增加如下声明： <property name =" hibernate.dialect" value =" org.hibernate. dialect. Kingbase8Dialect"/> 支持 Hibernate3.2.7 Hibernate4.0.1 Hibernate4.3.2

目前 Mybatis3.2.8，Mybatis3.3.0 和 Mybatis3.4.5 均已通过 KingbaseES V8R6 2 版本适配的验证测试

MyBatis 的 jar 包可以从官方网站下载，Mybatis 所以使用的 JDBC 包 kingbase8-8.2.0.jar 位于 $KINGBASE_HOME/jdbc 目录下。使用时将 Mybatis 包和 JDBC 包导入到项目的 Libraries 中并定义相关配置项即可。

定义 Mybatis 配置文件，跟据用户选择，更改一下配置文件。

在 config.xml 中配置 JDBC 的驱动信息参数，数据库服务器信息参数和登陆用户信息参数。当然，这些参数也可以根据用户应用需求，单独生成 property 文件，针对不用的应用场景，导入不同的属性文件。这里以 property 为例说明下 config.xml 的配置使用。

在 property 中增加如下声明：

```
jdbc.driverClassName=com.kingbase8.Driver
jdbc.url=jdbc:kingbase8://192.168.229.128:54322/TEST
jdbc.username= 登录名
jdbc.password= 登录密码
在 config.xml 中增加如下声明：
```

(continues on next page)

(continued from previous page)

```
<environments default="development">
   <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
         <property name="driver" value="${jdbc.driverClassName}"/>
         <property name="url" value="${jdbc.url}"/>
         <property name="username" value="${jdbc.username}" />
         <property name="password" value="${jdbc.password}" />
      </dataSource>
   </environment>
</environments>
```

具体使用样例请参考 MyBatis 指南

或者 MyBatis 官网 http://www.mybatis.org/mybatis-3/zh/index.html

3.3.8.2 移植 Oracle OCI 应用程序

Kingbase V8R6 支持 OCI 的所有常用接口。

具体参见手册。

3.3.8.3 修改应用框架

如果应用使用的框架支持多种数据库和各自方言，修改配置使得框架使用 postgresql 库及其方言。

分享【教育部语言文字系统-应用迁移】:

未修改之前框架适配使用的是 oracle 的方言和表结构，其中用到了 BLOB/CLOB。

经调研框架也支持其他库，例如 mysql 和 postgresql，除 oracle 外，类型均为 bytea/binary，目前解决方法是配置框架使用 postgresql 库及其方言。

具体修改：

1. BaseInterceptor.java 中 Kingbase 类型库使用 postgresql 方言
2. Spring-Context-Activiti.xml 中指定 database type 为 postgres，database schema update 设为 false
3. 数据库实例需要配置为大小写不敏感，工作流使用的库需要建在默认 schema 下

3.3.9 测试与调试移植系统

任何一个成熟的应用系统如果代码、尤其是关键代码变动后，则应进行全面细致的测试。类似的，更换新的后台数据库系统以后，也应对移植后的数据库系统进行全面的功能和性能测试。

3.3.9.1 功能测试和排错

功能测试是指对移植数据库系统的每一个模块和功能进行全面的系统回归测试，用以确保新系统各个功能的正确性。

因此，完成数据库对象和应用程序迁移后，应对移植系统进行全面的功能测试，并对测出问题及时分析、排查和修改。对那些很难定位的问题，请及时联系 KingbaseES 支持工程师。

3.3.9.2 性能测试和调优

移植系统性能测试和调优是在完成移植系统功能测试后和系统上线前，在实际或模拟生产数据上，对移植系统进行的性能测试和调优。

移植系统性能测试和调优的主要步骤如下：

- **构造测试数据**：若条件允许的话，建议构造与实际生产数据规模相同的数据，并模拟构造未来一年、两年、五年或更长生命周期的数据进行测试。
- **部署测试软硬件环境**：根据测试数据规模的大小，配置适当的测试软硬件环境。
- **性能测试**：既可采用手动方式，也可利用 TPCC 测试工具、LoadRunner 等工具对移植系统进行自动测试。
- **性能调优**：对未达到性能指标的功能模块及其 SQL 语句进行优化并给出相关建议。

通常，性能测试效果与测试数据规模、软硬件配置等因素密切相关。因此，建议性能测试时，测试数据规模、软硬件配置应尽量与将来的实际生产环境一致。必要时，在未来一年、两年、五年等不同模拟数据规模场景下，应分别测试移植系统的性能指标，用以保证移植系统未来仍能具有良好的性能表现。

第 4 章

KingbaseES V8R6 与 Oracle 12c 对比

4.1 数据类型

4.1.1 KingbaseES 数据类型到 Oracle 数据类型转换

表 4.1.1: KingbaseES 数据类型到 Oracle 数据类型转换

序号	KingbaseES 数据类型	Oracle 数据类型	备注说明（KingbaseES）
数值型			
1	tinyint	NUMBER	单字节整数 -128 to +127
2	smallint	NUMBER	小范围整数 -32768 to +32767
3	integer	NUMBER	整数的典型选择 -2147483648 to +2147483647
4	bigint	NUMBER	大范围整数 -9223372036854775808 to +9223 372036854775807
5	decimal	NUMBER	用户指定精度，精确最高小数点前 131072 位，以及小数点后 16383 位
6	numeric, number	NUMBER	用户指定精度，精确最高小数点前 131072 位，以及小数点后 16383 位
7	real	FLOAT	可变精度，不精确 6 位十进制精度
8	float	FLOAT	
9	double precision, double	FLOAT	8 字节可变精度，不精确 15 位十进制精度

continues on next page

表 4.1.1 – continued from previous page

10	smallserial	NUMBER	自动增加的小整数 1 到 32767
11	serial	NUMBER	自动增加的整数 1 到 2147483647
12	bigserial	NUMBER	单字节整数 -128 to +127
字符型			
1	character (n[char | byte]), char(n[char | byte])	CHAR/CLOB	定长，最大到 8000，空格填充。当长度不大于 1000 时转换为 CHAR, 当长度超过 1000 则转换为 CLOB
2	character varying(n[char | byte]), varchar(n[char | byte]), varchar2(n[char | byte])	VARCHAR2/CLOB	有限制的变长, 最大到 8000。当长度不大于 2000 时转换为 CHAR, 当长度超过 2000 则转换为 CLOB
3	text	CLOB	无限变长
大对象类型			
1	clob	CLOB	字符大对象
2	bytea	BLOB	变长二进制串
3	blob	BLOB	二进制大对象
日期时间			
1	timestamp [(p)] [without time zone]	TIMESTAMP	日期和时间（无时区）
2	timestamp [(p)] with time zone	TIMESTAMP() WITH TIME ZONE	包括日期和时间，有时区
3	date	DATE	日期
4	time [(p)] [without time zone]	DATE	一天中的时间
5	time [(p)] with time zone	DATE	仅仅是一天中的时间,带有时区
6	interval year	NUMBER	时间间隔
7	interval month	NUMBER	时间间隔
8	interval day	NUMBER	时间间隔
9	interval hour	NUMBER	时间间隔
10	interval minute	NUMBER	时间间隔
11	interval second	FLOAT	时间间隔
12	interval year to month	INTERVAL YEAR() TO MONTH	时间间隔
13	interval day to second	INTERVAL DAY() TO SECOND	时间间隔
布尔类型			
1	boolean	NUMBER	状态为真或假
位串类型			
1	bit(n) bit varying(n)	BLOB	一串 1 和 0 的串
XML 类型			
1	xml	XMLTYPE	用来存储 XML 数据

4.1.2 Oracle 数据类型到 KingbaseES 数据类型转换

表 4.1.2: Oracle 数据类型到 KingbaseES 数据类型转换

序号	Oracle 数据类型	KingbaseES 数据类型	备注说明（Oracle）
数值型			
1	NUMBER(p,s)	numeric (precision, scale)	1.0 x |image16| 到 1.0 x |image17|
2	FLOAT	double	Number 的子类型
3	BINARY_FLOAT	double	4 字节，单精度浮点数
4	BINARY_DOUBLE	double	8 字节，双精度浮点数
字符型			
1	CHAR	char	定长字符串
2	VARCHAR2	varchar	变长字符串
3	NCHAR	char	Unicode 编码字符串
4	NVARCHAR2	varchar	Unicode 编码字符串
日期时间			
1	DATE	timestamp [(p)] [without time zone]	日期类型
2	TIMESTAMP	timestamp [(p)] [without time zone]	时间类型
3	TIMESTAMP() WITH TIME ZONE	timestamp [(p)] with time zone	带时区时间类型
4	TIMESTAMP() WITH LOCAL TIME ZONE	timestamp [(p)] with time zone	带本地时区时间类型
5	INTERVAL DAY() TO SECOND	interval year to month	时间间隔
6	INTERVAL YEAR() TO MONTH	interval day to second	时间间隔
大对象类型			
1	BLOB	blob	二进制大对象
2	CLOB	clob	字符大对象
3	NCLOB	clob	存储 Unicode 数据
其他类型			
1	LONG	text	存储可变长字符串
2	RAW	bytea	存储字符型数据
3	LONG RAW	bytea	存储图像文档声音等二进制数据
4	ROWID	varchar	长度 64, 可以存储 A-Z, a-z, 0-9, + /
5	UROWID	varchar	存储 Index -organized 表和外部表的地址

4.2 常用函数

4.2.1 有区别的函数

表 4.2.1: 有区别的函数

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
数值函数			
1	REMAINDER (n2,n1)	/	取第一个参数除以第二个参数的余数，参数类型是 number 或者可以转换为 number 的
2	NANVL(n2,n1)	/	当 n2 为 NaN 时，函数返回 n1 当 n2 不为 NaN 时，函数返回 n2，参数类型是浮点数
3	/	cbrt(dp)	立方根函数
4	/	degrees(dp)	把弧度转换为角度
5	/	radians(dp)	把角度转换为弧度
6	/	div(y numeric, x numeric)	y/x 的整数商
7	/	pi()	” π” 常数值 3.14159265358979
8	/	scale(numeric)	返回参数的精度，小数点后位数
字符函数			
1	CHR(int,[USING NCHAR_CS])	chr(int)	返回 int 类型参数指定的值的字符，KingbaseES 不允许为 0，O racle 可以输入 0.
2	NCHR (number)	/ or chr(int)	返回 number 类型值的国际字符，Kingb aseES 不支持超出字符范围的数值
3	REGEXP_REPLACE (source_char , pattern, replace_string, position, occurrence, match_param)	regexp_replace (string text, pattern text, replacement text [, flags text])	替换匹配。Oracle 与 KingbaseES 使用方式部分不同，Oracle :position: 从源第几个字符开始 occurrence：第几次替换。KingbaseES 没有这个参数
4	REGEXP_COUNT (source_char ,pattern [,position [,match_param]])	regexp_count (source_char ,pattern [,position [,match_param]])	返回匹配字符串数量。Oracle 与 KingbaseES 的某些数据类型不同导致结果不同，例如 time 类型。Oracle 与 KingbaseES 的正则中 match_param 的意义有部分不同导致结果不同，可参考 正则表达式元语法。

continues on next page

表 4.2.1 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
5	REGEXP_INSTR (source_char ,pattern [,position [,occurrence [,return _option [,match_param [,subexpr]]]]])	regexp_instr (source_char ,pattern [,position [,occurrence [,return _option [,match_param [,subexpr]]]]])	返回字符串的起始位置，或者结束位置。Oracle 与 KingbaseES 的某些数据类型不同导致结果不同，例如 time 类型。Oracle 与 KingbaseES 的正则中 match_param 的意义有部分不同导致结果不同，可参考 正则表达式元语法。
6	/	strpos(string, substring) position (substring in string)	返回指定字符串位置
7	/	convert(expr1 text, [, expr2 text], expr3 text)	把字符串转换为 expr3 中声明的编码
8	NLS_INITCAP(char,['NLS_SORT = sort'])	INITCAP(string)	将每一个词的第一个字母转换为大写形式并把剩下的字母转换为小写形式。
9	NLS_LOWER(char ,'nlsparam')	lower(string)	将字符串转换为小写形式
10	CONCAT (char1,char2)	concat(str "any" [, str "any" [, ...]])	串接所有参数的文本表示。NULL 参数被忽略。Oracle：支持 2 个参数 KingbaseES：支持多个参数
11	/	regexp _matches(string text, pattern text [, flags text])	返回对 string 匹配一个 POSIX 正则表达式得到的所有子串
日期类函数			
1	CURRENT _TIMESTAMP(precision)	CURRENT _TIMESTAMP	返回当前日期时间区别是精度不同
2	/	age(timestamp, timestamp) age(timestamp)	减去参数，生成一个使用年、月（而不是只用日）的"符号化" 的结果
3	/	clock_timestamp ()	返回当前日期和时间
4	/	date_part(text, timestamp) date_part(text, interval)	返回日期时间的子域
5	TRUNC(date, fmt)	date_trunc (text, timestamp) date_trunc (text, interval)	返回截断到参数指定的日期时间
6	TO_DSINTERVAL ('sql_format')	make_interval (years int DEFAULT 0, months int DEFAULT 0, weeks int DEFAULT 0, days int DEFAULT 0, hours int DEFAULT 0, mins int DEFAULT 0, secs double precision DEFAULT 0.0)	构造一个 interval，输入参数一个是字符串，一个是数字
7	/	now()	返回当前日期和时间
比较函数			

continues on next page

表 4.2.1 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
1	GREATEST (expr ,expr,expr,...) LEAST (expr ,expr,expr,...)	GREATEST(value [, ...]) LEAST(value [, ...])	选取最大或者最小的数值 oracle：只要有一个参数是 null 则结果返回 null KingbaseES：全部参数是 null 才返回 null，NULL values in the list are ignored
序列函数			
1	/	currval (regclass)	返回最近一次用 nextval 获取的指定序列的值
2	/	nextval (regclass)	递增序列并返回新值
3	/	setval (regclass, bigint) setval (regclass, bigint, boolean)	设置序列的当前值
4	/	lastval()	返回最近一次用 nextval 获取的任何序列的值
大对象函数			
1	/	clob_import(string，string)	将指定的文件以 clob 大对象的形式导入数据库。
2	/	clob_export (clob，string，string)	将 clob 大对象的内容导出到磁盘文件。
3	/	blob_import(string，string)	将指定的文件以 blob 大对象的形式导入数据库。
4	/	blob_export (blob，string，string)	将 blob 大对象的内容导出到磁盘文件。
Null 相关函数			
1	NANVL (n2 , n1)	CASE WHEN n2 is nan THEN n1 ELSE n2 END	浮点数 NAN 值判断，如果 n2 是 NAN 则返回 n1，否则返回 n2
2	/	NULLIF(value1, value2)	当 value1 和 valu e2 相等时，NULLI F 返回一个空值。否则它返回 value1
Environment 和 Identifier 函数			
1	/	version()	返回当前版本号
分析和聚集函数			
1	COLLECT ([DISTINCT UNIQUE] column[ORDER BY expr])	array_agg (expression)	形式上类似：KingbaseES 返回参数类型的数组 Oracle 返回参数类型的嵌套表
2	CORR_KCORR_S (expr1 , expr2, [COEFFICIENT ONE_SIDED_SIG ONE_SIDED_SIG _POS ONE_SIDED_SIG _NEG TWO_SIDED_SIG])	/	相关率
3	LISTAGG (measure_expr, ‘delimiter’) WITHIN GROUP (order_by_clause)	string_agg (expression, delimiter)	输入值连接成一个串，用定界符分隔
4	MEDIAN (expr)	/	返回中间值

continues on next page

表 4.2.1 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
5	STATS _BINOMIAL_TEST (expr1 , expr2 , p)	/	统计二项测试是一个精确概率测试用于二分变量, 那里只有两个可能值存在。它测试一个样品之间的差异比例和给定的比例
6	STATS_CROSSTAB (expr1 , expr2)	/	用于分析两个变量
7	STATS_F_TEST （ expr1 , expr2）	/	测试两个方差是否有明显的不同
8	STATS_KS_TEST （ expr1 , expr2)	/	是柯尔莫哥洛夫斯米尔诺夫函数比较两个样品测试他们是否来自相同的总体或有相同分布的总体
9	STATS_MODE(expr)	/	统计模式需要一组值作为它的参数并且返回发生以最大的频率
10	STATS_MW_TEST(expr1 , expr2)	/	一个曼惠特尼测试比较两个独立样本来测试该无效假设, 这两个种群具有相同的分布函数与替代并且假设这两个分布函数是不同的
11	STATS _ONE_WAY_ANOVA(expr1 , expr2)	/	单向方差分析函数 (统计一维方差分析) 测试差异在意味着 (为团体或变量), 通过比较两种统计学意义不同的估计方差
12	STATS_T_TEST_ONE （ expr1,)	/	A one-sample t-test
13	STATS _T_TEST_PAIRED(expr1, expr2)	/	A two-sample, paired t-test (also known as acrossed t-test)
14	STATS _T_TEST_INDEP(expr1, expr2)	/	A t-test of two independent groups with the same variance(pooled variances)
JSON 函数			
1	/	to_json (anyelement) to_jsonb (anyelement)	返回为 json 或者 jsonb
2	/	array_to_json (anyarray [, pretty_bool])	把数组作为一个 JSON 数组返回
3	/	row_to_json (record [, pretty_bool])	把行作为一个 JSON 对象返回
4	/	json_build_array (VARIADIC ” any”) jsonb_build _array(VARIADIC ”any”)	从一个可变参数列表构造一个可能包含异质类型的 JSON 数组。
5	/	json_build _object(VARIADIC ” any”) jsonb_build _object(VARIADIC ”any”)	从一个可变参数列表构造一个 JSON 对象

continues on next page

表 4.2.1 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
6	/	json _object(text[]) jsonb _object(text[])	从一个文本数组构造一个 JSON 对象
7	/	json_arr ay_length(json) jsonb_array _length(jsonb)	返回最外层 JSON 数组中的元素数量
8	/	json_each(json) jsonb _each(jsonb) json_each_text (json) jsonb_each_text (jsonb)	扩展最外层的 JSON 对象成为一组键/值对
9	/	json_extract _path(from_json json, VARIADIC path_elems text[]) jsonb_extract _path(from_json jsonb, VARIADIC path_elems text[])	返回由 path_elems 指向的 JSON 值（等效于 #> 操作符）
10	/	json _extract_path _text(from_json json, VARIADIC path_elems text[]) jsonb _extract_path _text(from_json jsonb, VARIADIC path_elems text[])	text 以文本返回由 path_elems 指向的 JSON 值（等效于 #» 操作符）
11	/	json_ob ject_keys(json) jsonb_obj ect_keys(jsonb)	返回最外层 JSON 对象中的键集合
12	/	json_popul ate_record(base anyelement, from_json json) jsonb_popul ate_record(base anyelement, from_json jsonb)	扩展 from_json 中的对象成一个行，它的列匹配由 base 定义的记录类型（见下文的注释）
13	/	json_populate _record-set(base anyelement, from_json json) jsonb_populate _record-set(base anyelement, from_json jsonb)	扩展 from_json 中最外的对象数组为一个集合，该集合的列匹配由 base 定义的记录类型
14	/	json_array _elements(json) jsonb_array _ele-ments(jsonb)	把一个 JSON 数组扩展成一个 JSON 值的集合
15	/	json_array_elem ents_text(json) jsonb _ar-ray_elements _text(jsonb)	把一个 JSON 数组扩展成一个 text 值集合
16	/	json _typeof(json) jsonb _typeof(jsonb)	把最外层的 JSON 值的类型作为一个文本字符串返回
17	/	json _to_record(json) jsonb_to _record(jsonb)	从一个 JSON 对象（见下文的注解）构建一个任意的记录
18	/	json_to _recordset(json) jsonb_to _recordset (jsonb)	从一个 JSON 对象数组（见下文的注解）构建一个任意的记录集合

continues on next page

表 4.2.1 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
19	/	json_strip _nulls(from_json json) jsonb_strip _nulls(from_json jsonb)	返回具有空值对象域的 from_json 。其它空值不变
20	/	jsonb_set (target jsonb, path text[], new_value jsonb[, create_missing boolean])	如果 create _missing 是真的（缺省是 true ）并且通过 path 指定部分不存在，那么返回 target，它具有 path 指定部分，new_value 替换部分，或者 new_value 添加部分
21	/	jsonb_insert (target jsonb, path text[], new_value jsonb, [insert_after boolean])	返回被插入了 new_value 的 target
22	/	jsonb_pretty (from_json jsonb)	作为缩进 JSON 文本返回 from_json

4.2.2 无区别的函数

表 4.2.2: 无区别的函数

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
数值函数			
1	ABS(n)	abs(x)	绝对值
2	ACOS(n)	acos(x) acosd(x)	反余弦
3	ASIN(n)	asin(x) asind(x)	反正弦
4	ATAN(n)	atan(x) atand(x)	反正切
5	ATAN2(n1,n2)	atan2(y, x)	*y/x* 反正切
6	TAN(n)	tan(x) tand(x)	正切
7	EXP	exp(dp or numeric)	指数
8	FLOOR	floor(dp or numeric)	不大于参数的最大整数 mod(y, x)
9	MOD	mod(y, x)	y/x 的余数
10	SIGN	sign(dp or numeric)	参数的符号（-1, 0, +1）
11	WIDTH_BUCKET	width_bucket	返回一个桶
12	BITAND (expr1,expr2)	bitand(expr1, expr2)	按位与，参数类型都是 number 或者可以转换为 number 的
13	LN(n)	ln(dp or numeric)	自然对数
14	LOG(n2,n1)	log(dp or numeric) log(b numeric, x numeric)	KingbaseES：单个参数是以 10 为底的对数两个参数的形式 KingbaseES 和 Oracle 一致
15	POWER(n2,n1)	power(a dp,bdp) power(a numeric, b numeric)	返回 a 的 b 次幂
16	ROUND (n[, integer])	round(dp or numeric) round(v numeric, s int)	圆整为最接近的整数圆整为 s 位小数数字
17	SQRT (n)	sqrt(dp or numeric)	平方根
18	TRUNC (n1[, n2])	trunc(dp or numeric) trunc(v numeric, s int)	截断（向零靠近）截断为 s 位小数位置的数字

continues on next page

表 4.2.2 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
19	CEIL (n)	ceil(dp or numeric) ceiling(dp or numeric)	不小于参数的最小整数
20	BIN_TO_NUM(expr ,...)	bin_to_num(expr ,...)	将二进制数转换为十进制。将任何数据类型或可以隐式转换为数据类型的非数据类型作为参数，每个 expr 必须为 0 或 1。
字符函数			
1	SUBSTR	substring (string [from int] [for int])	提取子串
2	RPAD	rpad(string text, length int [, fill text])	用字符串 expr3(第三个参数) 将字符串 expr1(第一个参数) 从右边填充到指定的长度 expr2(第二个参数)，在第三个参数缺省时，填充空格。
3	REPLACE (char , search_string, replacement _string)	replace(string text, from text, to text)	将 string 中出现的所有子串 from 替换为子串 to
4	REGEXP_SUBSTR(source_char , pattern, position, occurrence, match_param ,subexpr)	regexp_substr (string text, pattern text [, position int [, occurrence int [, flags text [, num int]]]])	在 string 搜索一个 POSIX 正则表达式字符串，返回的搜索到的子字符串
5	TRANSLATE (expr,from_string,to_string)	translate (string text, from text, to text)	string 中任何匹配 from 集合中一个字符的字符会被替换成 to 集合中的相应字符
6	INSTRB(string, substring, position, occurrence)	instrb(expr1 text, expr2 text, [expr3 int [,expr4 int]])	在父字符串 expr1 中的第 expr3 个位置（从 1 开始）以字符为单位开始查找第 expr4 次出现的子字符串的位置，0 表示不包含子字符串 expr2 。如果 expr3 为负，则从 expr1 的倒数第 expr3 个位置开始反向查找，位置依然是从字符串开头算起。
7	LENGTH	length(string) length(string bytea, encoding name)	返回字符串长度
8	reverse(str)	reverse(str)	返回反转的字符串。
9	UPPER	upper(string)	将字符串转换成大写形式
10	LPAD(expr1 [,n,expr2])	lpad(string text, length int [, fill text])	将 string 通过前置字符 fill（默认是一个空格）填充到长度 length。如果 string 已经长于 length ，则它被（从右边）截断。

continues on next page

表 4.2.2 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
11	LTRIM(char,set)	ltrim(string text [, characters text])	从 string 的开头删除最长的只包含 characters （默认是一个空格）的字串。在兼容 ORACLE 的时候（ora_func_style 为 true），characters 只能是一个字符。ora_func_style 为 false 时可以为多个字符。
12	RTRIM(char,set)	rtrim(string text [, characters text])	从 string 的结尾删除最长的只包含 characters（默认是一个空格）的字串。在兼容 ORACLE 的时候（ora_func_style 为 true），characters 只能是一个字符。ora_func_style 为 false 时可以为多个字符。
13	TRIM ([LEADIN G | TRAILING | BOTH [trim_character] from] [trim_character from] trim_source)	trim([leading | trailing | both] [characters from] string)	从 string 的开头 /结尾/两端删除最长的只包含 characters（默认是一个空格）的串
14	ASCII(char)	ascii(string)	返回参数第一个字符的 ASCII 代码
15	LENGTHB(char)	lengthb(string)	返回二进制串中的字节数
16	SUBSTRB(char,position[, sub string_length])	substrb(expr1 text, [from] expr2 int [, [for] expr3 int])	取子字符串，在父字符串 expr1(第一个参数) 中的第 expr2(第二个参数) 个字节位置开始取 expr3(第三个参数) 个字节，如果第三个参数缺省，则从第 expr2 (第二个参数) 个位置开始取右面部分的全部，如果第二个参数为负，则是从父字符串的尾部截取 expr3 个字节。
日期时间			
1	EXTRACT (field from expr)	extract (fieldfrom timestamp) extract(field from interval)	从当前日期时间抽取时间字段
2	TO_CHAR (datetime,fmt)	to_char (timestamp, text)	日期时间转换为字符
3	TO_TIMESTAMP (char, fmt)	to_timestamp (text, text)	返回日期时间
4	ADD_MONTHS (date , integer)	ADD_MONTHS (expr1 日期/时间类型, expr2 INT)	返回 expr1 加上 expr2 个月的日期时间值
5	LAST_DAY (date)	LAST_DAY(expr1 日期/时间类型)	返回日期中的月份的最后一天
6	MONTHS_BETWEEN (date1 , date2)	MONTHS_BETWEEN (date1 , date2)	返回日期的月份差
7	NEXT_DAY (date , char)	NEXT_DAY(expr1 日期/时间类型, expr2 TEXT)	返回 expr 1 日期后的第一个由参数 expr2 命名的日期，参数 expr2 的值必须在周一到周日范围中
8	SYSDATE	SYSDATE	返回当前系统日期
9	SYSTIMESTAMP	SYSTIMESTAMP	返回当前系统时间

continues on next page

表 4.2.2 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
10	TO_DATE(char, fmt)	to_date(text, text)	返回日期值输入参数一个是字符串，一个是数字
11	TO_TIMESTAMP (char, fmt)	make_time(hour int, min int, sec double precision) make_timestamp (year int, month int, day int, hour int, min int, sec double precision) make_timestamptz (year int, month int, day int, hour int, min int, sec double precision, [timezone text]) to_timestamp (text, text)	返回日期时间值输入参数一个是字符串，一个是数字
12	OVERLAPS	(start1, end1) OVERLAPS (start2, end2)	在两个时间域（用它们的端点定义）重叠的时候得到真，当它们不重叠时得到假
13	CURRENT_DATE	current_date	当前日期
大对象函数			
1	EMPTY_BLOB() EMPTY_CLOB()	empty_blob() empty_clob()	返回空大对象
Decoding 函数			
1	DECODE (expr,search, result,default)	decode(expr ,search , result, default)	将 expr 和 search 值一个一个比较。如果相等则返回 result 如果不等返回 default 值。
NULL 相关函数			
1	COALESCE (expr,...)	COALESCE(value [, ...])	COALE SCE 函数返回它的第一个非空参数的值。当且仅当所有参数都为空时才会返回空。
2	LNNVL(condition)	LNNVL(condition)	用于 where 条件中，condition 为真时返回假，假时返回真
3	NULLIF (expr1 , expr2)	NULLIF(value1, value2)	KingbaseES :value1 可以为 null oracle：value1 不能为 null
4	NVL(expr1, expr2)	NVL(expr1, expr2)	当 expr1 为 NULL 时，用 expr2 代替本表达式的值
5	NVL2 (expr1 , expr2 , expr3)	NVL2 (expr1 , expr2 , expr3)	当 expr1 为 NULL 时，返回值为 expr3；否则为 expr2
Environment 和 Identifier 函数			
1	sys_guid()	sys_guid()	返回一个全局唯一的标识符
2	uid	uid	返回目前执行环境下的用户 ID
3	user	User /current_user	返回当前用户

continues on next page

表 4.2.2 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
分析和聚集函数			
1	COUNT (* DISTINCT ALL expr)	count	非空的输入行的数目
2	COVAR_POP (expr1 , expr2) OVER (a nalytic_clause)	covar_pop(Y, X)	总体协方差
3	COVAR_SAMP (expr1 , expr2) OVER (analytic_clause)	covar_samp(Y, X)	样本协方差
4	CUME_DIST (expr ,) WITHIN GROUP (ORDER BY expr)	cume_dist(args) WITHIN GROUP (ORDER BY sorted_args)	当前行的相对排名
5	DENSE_RANK (expr,) WITHIN GROUP(ORDER BY expr)	d ense_rank(args) WITHIN GROUP (ORDER BY sorted_args)	不带间隙的当前行排名
6	MAX (DISTINCT ALL expr)	max(expression)	所有输入值中 expression 的最大值
7	MIN (DISTINCT ALL expr)	min(expression)	所有输入值中 expression 的最小值
8	PERCENT_RANK (expr ,) WITHIN GROUP(ORDER BY expr)	percent_rank (args) WITHIN GROUP (ORDER BY sorted_args)	假想行的相对排名，范围从 0 到 1
9	PERCENTILE_CONT (expr) WITHIN GROUP (ORDER BY expr)	percentile_cont (fractions) WITHIN GROUP (ORDER BY sort_expression)	连续百分率
10	PERCENTILE_DISC (expr) WITHIN GROUP (ORDER BY expr)	percentile_disc (fraction) WITHIN GROUP (ORDER BY sort_expression)	离散百分率
11	RANK (expr,) WITHIN GROUP (ORDER BY expr)	rank(args) WITHIN GROUP (ORDER BY sorted_args)	假想行的排名，为重复的行留下间隔
12	REGR_SLOPE	regr_slope(Y, X)	由（X, Y）对决定的最小二乘拟合的线性方程的斜率
13	REGR_INTERCEPT	regr_intercept (Y, X)	由（X, Y）对决定的最小二乘拟合的线性方程的 y 截距
14	REGR_COUNT	regr_count(Y, X)	两个表达式都不为空的输入行的数目
15	STDDEV (DISTINC TALL expr)	stddev (expression)	输入值的样本标准偏差
16	STDDEV_POP (expr)	stddev_pop (expression)	输入值的总体标准偏差
17	STDDEV_SAMP (expr)	stddev_samp (expression)	输入值的样本标准偏差
18	SUM (DISTINCT ALL expr)	sum(expression)	所有输入的 expression 的和
19	VAR_POP (expr)	var_pop (expression)	输入值的总体方差
20	FIRST	first_value (value any)	返回在窗口帧中第一行上计算的 value
21	FIRST_VALUE *	first_value (value any)	返回在窗口帧中第一行上计算的 value

continues on next page

表 4.2.2 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
22	LAG	lag(value anyelement [, offset integer [, default anyelement]])	返回 value，它在分区内当前行的之前 offset 个位置的行上计算；如果没有这样的行，返回 default 替代。(作为 value 必须是相同类型)。offse t 和 default 都是根据当前行计算的结果。如果忽略它们，则 offset 默认是 1，default 默认是空值
23	LAST	last_value (value any)	返回在窗口帧中最后一行上计算的 value
24	LAST_VALUE *	last_value (value any)	返回在窗口帧中最后一行上计算的 value
25	LEAD	lead(value anyelement [, offset integer [, default anyelement]])	返回 value，它在分区内当前行的之后 offset 个位置的行上计算；如果没有这样的行，返回 default 替代。(作为 value 必须是相同类型)。offset 和 default 都是根据当前行计算的结果。如果忽略它们，则 offset 默认是 1，default 默认是空值
26	NTH_VALUE *	nth_value(value any, nth integer)	返回在窗口帧中第 nth 行（行从 1 计数）上计算的 value；没有这样的行则返回空值
27	NTILE	ntile (num_buckets integer)	从 1 到参数值的整数范围，尽可能等分分区
28	ROW_NUMBER	row_number()	当前行在其分区中的行号，从 1 计
29	CORR (expr1 , expr2)[OVER (analytic_clause)]	corr(Y, X)[OVER (analytic_clause)]	相关系数
30	AVG (DISTINCT ALL expr) OVER (analytic_clause)	avg(expression)OVER (analytic_clause)	所有输入值的平均值（算术平均）
XML 函数			
1	XMLCOMM ENT(value_expr)	xmlcomment(text)	创建了一个 XML 值
2	XMLCONCAT (XM LType_instance,)	xmlconcat(xml[, ...])	由单个 XML 值组成的列表串接成一个单独的值，这个值包含一个 XML 内容片断
3	XMLELEMENT	xmlelement(name name [, xmla ttributes(value [AS attname] [, ...])] [, content, ...])	表达式 xmlelement 使用给定名称、属性和内容产生一个 XML 元素。
4	XMLFOR EST(value_expr)	xmlforest (content [AS name] [, ...])	表达式 xmlforest 使用给定名称和内容产生一个元素的 XML 森林（序列）。
5	XMLPI	xmlpi(name target [, content])	表达式 xmlpi 创建一个 XML 处理指令。如果存在内容，内容不能包含字符序列
6	XMLROOT	xmlroot(xml, version text | no value [, standalone yes | no | no value])	表达式 xmlroot 修改一个 XML 值的根结点的属性

continues on next page

表 4.2.2 – continued from previous page

序号	Oracle 函数名	KingbaseES 函数名	功能简要说明
7	XMLAGG	xmlagg(xml)	函数 xmlagg 是一个聚集函数。它将聚集函数调用的输入值串接起来
8	APPENDCHILDXML	appe ndchildxml(xml, XPath_string text, value_expr text[, nam espace_string])	将用户提供的值作为 XPath 表达式指示的节点的子节点附加到目标 XML 上
9	UPDATEXML	updatexml(xml, XPath_string text, value_expr text[, nam espace_string])	将 XML 实例、XPath 及新的 xml 更新值对作为参数，并返回 XML 具有更新值的实例
层次查询函数			
1	SYS_CONNECT _BY_PATH(column ,char)	SYS_CONNECT _BY_PATH (Expression, char)	返回从根元组到当前元组的路径，这个路径是由 c har 字符分隔的在各个元组上计算的 Expression 的值连接构成的。

第 A 章

版权声明

人大金仓版权所有，并保留对本手册及本声明的一切权利。未得到人大金仓的书面许可，任何人不得以任何方式或形式对本手册内的任何部分进行复制、摘录、备份、修改、传播、翻译成其他语言、将其全部或部分用于商业用途。

免责声明

本手册内容依据现有信息制作，由于产品版本升级或其他原因，其内容有可能变更。人大金仓保留在没有任何通知或者提示的情况下对手册内容进行修改的权利。本手册仅作为使用指导，人大金仓在编写本手册时已尽力保证其内容准确可靠，但并不确保手册内容完全没有错误或遗漏，本手册中的所有信息也不构成任何明示或暗示的担保。

技术支持

- 人大金仓官方网站：http://www.kingbase.com.cn/ 您可以在官网中获得人大金仓所有产品的资讯信息，销售联系方式。
- 金仓数据库子网站：http://kes.kingbase.com.cn/ 您可以在产品子网站中获得最新的产品技术资料、产品故障原因及问题分析、产品的应用解决方案、软件升级资料等等。
- 全国服务热线：400-601-1188
- 人大金仓技术支持与反馈信箱：support@kingbase.com.cn