LibrA Tool
2.7.RC2

Oracle 语法迁移

文档版本	01
发布日期	2017-07-30

华为技术有限公司

商标声明

HUAWEI和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://e.huawei.com

目 录

1 前言

1.1 简介

Oracle语法迁移文档列举了Migration Tool支持的Oracle功能，以及用于实现各功能的Oracle语法及对应的FusionInsight LibrA语法。语法展示了Oracle脚本的内部迁移逻辑。

本文档为数据库迁移团队和客户站点Oracle脚本迁移验证提供参考。

本文档适用于FusionInsight LibrA V100R002C71。

1.2 修订记录

日期	版本	变更说明
2017-07-30	2.7.RC2	初稿

1.3 读者对象

本文档适用于需进行Oracle到FusionInsight LibrA数据迁移的客户和Migration Tool团队。

1.4 前提条件

使用迁移工具的用户需具有Oracle、FusionInsight LibrA、SQL相关知识。

1.5 系统要求

Migration Tool支持如下Oracle和FusionInsight LibrA版本：

Oracle：Oracle Database 11g Release 2

FusionInsight LibrA：V100R006C20,V100R002C70 and V100R002C71

2 支持的 Oracle 功能和语法

以下表格列举了用于迁移的Oracle功能和语法。“版本”列为添加或新增该功能时Migration Tool的版本号。

2.1 MERGE

MERGE是ANSI标准SQL语法。大多数数据库系统，例如Oracle和Teradata，支持该功能。当前FusionInsight LibrA不支持该功能。MERGE语句用于从一个或多个源选择行，并更新或插入到表或视图中。用户可指定新增或插入目标表或视图的条件。

#	版本	Oracle功能	Oracle语法	迁移后的语法
1	V1R3C00SPC010	使用MERGE，并指定UPDATE和INSERT（配置参数mergeImplementation默认设为WITH）	MERGE INTO employees01 e USING (SELECT empid, ename, startdate, address FROM hr_records WHERE empid > 100) h ON (e.id = h.empid) WHEN MATCHED THEN UPDATE SET e.address = h.address , e.ename = h.ename WHEN NOT MATCHED THEN INSERT (empid,ename,startdate,address) VALUES (h.empid,h.ename,h.startdate,h.address);	WITH h AS (SELECT empid, ename, startdate, address FROM hr_records WHERE empid > 100) , UPD_REC AS (UPDATE employees01 e SET e.address = h.address , e.ename = h.ename FROM h WHERE e.id = h.empid returning e.*) INSERT INTO employees01 (empid, ename, startdate, address) SELECT h.empid, h.ename, h.startdate, h.address FROM h WHERE NOT EXISTS (SELECT 1 FROM UPD_REC e WHERE e.id = h.empid);

#	版本	Oracle功能	Oracle语法	迁移后的语法
2	V1R3C00SPC010	使用MERGE，并指定UPDATE和INSERT （假设配置参数mergeImplementation设为SPLIT）	MERGE INTO employees01 e USING (SELECT empid, ename, startdate, address FROM hr_records WHERE empid > 100) h ON (e.id = h.empid) WHEN MATCHED THEN UPDATE SET e.address = h.address , e.ename = h.ename WHEN NOT MATCHED THEN INSERT (empid,ename,startdate,address) VALUES (h.empid,h.ename,h.startdate,h.address);	UPDATE employees01 e SET e.address = h.address , e.ename = h.ename FROM (SELECT empid, ename, startdate, address FROM hr_records WHERE empid > 100) h WHERE e.id = h.empid; INSERT INTO employees01 (empid, ename, startdate, address) SELECT h.empid, h.ename, h.startdate, h.address FROM (SELECT empid, ename, startdate, address FROM hr_records WHERE empid > 100) hLEFT OUTER JOIN employees01 e ON e.id = h.empid WHERE e.id IS NULL;

#	版本	Oracle功能	Oracle语法	迁移后的语法
3	V1R3C00SPC010	使用MERGE，并指定UPDATE、DELETE和INSERT （配置参数mergeImplementation默认设为WITH）	MERGE INTO bonuses D USING (SELECT EMPID, salary, DEPTNO FROM employees WHERE DEPTNO = 30) S ON (D.employee_id = S.EMPID) WHEN MATCHED THEN UPDATE SET D.bonus = D.bonus + S.salary * 100 DELETE WHERE (S.salary > 8000) WHEN NOT MATCHED THEN INSERT (D.employee_id, D.bonus) VALUES (S.EMPID, S.salary * 100) WHERE (S.salary <= 8000);	WITH S AS(SELECT EMPID, salary, DEPTNO FROM employees WHERE DEPTNO = 30) , UPD_REC AS (UPDATE bonuses D SET D.bonus = D.bonus + S.salary * 100 FROM S WHERE D.employee_id = S.EMPID returning D.*) , DEL_REC AS (DELETE FROM bonuses D USING S WHERE (S.salary > 8000) AND D.employee_id = S.EMPID returning D.*) INSERT INTO bonuses (employee_id, bonus) SELECT S.EMPID, S.salary * 100 FROM S WHERE NOT EXISTS (SELECT 1 FROM UPD_REC D WHERE D.employee_id = S.EMPID) AND S.salary <= 8000;

#	版本	Oracle功能	Oracle语法	迁移后的语法
4	V1R3C00SPC010	使用MERGE，并指定UPDATE、DELETE和INSERT （假设配置参数mergeImplementation设为SPLIT）	MERGE INTO bonuses D USING (SELECT EMPID, salary, DEPTNO FROM employees WHERE DEPTNO = 30) S ON (D.employee_id = S.EMPID) WHEN MATCHED THEN UPDATE SET D.bonus = D.bonus + S.salary * 100 DELETE WHERE (S.salary > 8000) WHEN NOT MATCHED THEN INSERT (D.employee_id, D.bonus) VALUES (S.EMPID, S.salary * 100) WHERE (S.salary <= 8000);	UPDATE bonuses D SET D.bonus = D.bonus + S.salary * 100 FROM (SELECT EMPID, salary, DEPTNO FROM employees WHERE DEPTNO = 30) S WHERE D.employee_id = S.EMPID; DELETE FROM bonuses D USING (SELECT EMPID, salary, DEPTNO FROM employees WHERE DEPTNO = 30) S WHERE (S.salary > 8000) AND D.employee_id = S.EMPID; INSERT INTO bonuses (employee_id, bonus) SELECT S.EMPID, S.salary * 100 FROM (SELECT EMPID, salary, DEPTNO FROM employees WHERE DEPTNO = 30) SLEFT OUTER JOIN bonuses D ON D.employee_id = S.EMPID WHERE D.employee_id IS NULL AND S.salary <= 8000;

#	版本	Oracle功能	Oracle语法	迁移后的语法
5	V1R3C00SPC010	使用MERGE，且仅指定UPDATE （配置参数mergeImplementation默认设为WITH）	MERGE INTO student a USING (SELECT id, sname, score FROM student_n) b ON (a.id = b.id) WHEN MATCHED THEN UPDATE SET a.sname = b.sname , a.score = b.score WHERE a.score > 600;	WITH b AS (SELECT id, sname, score FROM student_n) UPDATE student a SET a.sname = b.sname , a.score = b.score FROM b WHERE a.score > 600 AND a.id = b.id;
6	V1R3C00SPC010	使用MERGE，且仅指定UPDATE （假设配置参数mergeImplementation设为SPLIT）	MERGE INTO student a USING (SELECT id, sname, score FROM student_n) b ON (a.id = b.id) WHEN MATCHED THEN UPDATE SET a.sname = b.sname , a.score = b.score WHERE a.score > 600;	UPDATE student a SET a.sname = b.sname , a.score = b.score FROM (SELECT id, sname, score FROM student_n) b WHERE a.score > 600 AND a.id = b.id;
7	V1R3C00SPC010	使用MERGE，并指定UPDATE和DELETE （配置参数mergeImplementation默认设为WITH）	MERGE INTO student a USING (SELECT id, sname, score FROM student_n) b ON (a.id = b.id) WHEN MATCHED THEN UPDATE SET a.sname = b.sname , a.score = b.score DELETE WHERE a.score < 640;	WITH b AS (SELECT id, sname, score FROM student_n) , UPD_REC AS (UPDATE student a SET a.sname = b.sname , a.score = b.score FROM b WHERE a.id = b.id returning a.*) DELETE FROM student a USING b WHERE a.score < 640 AND a.id = b.id;

#	版本	Oracle功能	Oracle语法	迁移后的语法
8	V1R3C00SPC010	使用MERGE，并指定UPDATE和DELETE（假设配置参数mergeImplementation设为SPLIT）	MERGE INTO student a USING (SELECT id, sname, score FROM student_n) b ON (a.id = b.id) WHEN MATCHED THEN UPDATE SET a.sname = b.sname , a.score = b.score DELETE WHERE a.score < 640;	UPDATE student a SET a.sname = b.sname , a.score = b.score FROM (SELECT id, sname, score FROM student_n) b WHERE a.id = b.id; DELETE FROM student a USING (SELECT id, sname, score FROM student_n) b WHERE a.score < 640 AND a.id = b.id;

2.2 INSERT ALL

Oracle的INSERT ALL语句可通过单个INSERT语句向单个或多个表中插入多行。

#	版本	Oracle功能	Oracle语法	迁移后的语法
9	V1R3C00SPC010	使用INSERT ALL	INSERT ALL INTO ap_cust VALUES (customer_id, program_id, delivered_date) INTO ap_orders (ord_dt, Prg_id) VALUES (order_date, program_id) SELECT program_id, delivered_date, customer_id, order_date FROM order WHERE deptno = 10;	WITH Sel AS (SELECT program_id, delivered_date, customer_id, order_date FROM order WHERE deptno = 10) , ins1 AS (INSERT INTO ap_cust (SELECT customer_id, program_id, delivered_date FROM Sel) returning *) INSERT INTO ap_orders (ord_dt, Prg_id) (SELECT order_date, program_id FROM Sel);

#	版本	Oracle功能	Oracle语法	迁移后的语法
10	V1R3C00SPC010	使用INSERT ALL并指定条件	INSERT ALL WHEN deptno <= 10 THEN INTO emp12 VALUES (empno,ename,job,mgr, hiredate,sal) WHEN deptno > 10 and deptno <= 20 THEN INTO emp13 VALUES (empno,ename,job, mgr,hiredate,sal) WHEN deptno > 20 THEN INTO emp14 VALUES (empno,ename,job, mgr,hiredate,sal) SELECT empno,ename,job,mgr, hiredate,sal,Deptno FROM emp WHERE job='MANAGER';	WITH Sel AS (SELECT empno,ename,job,mgr, hiredate,sal,Deptno FROM emp WHERE job='MANAGER') , ins1 AS (INSERT INTO emp12 (SELECT empno,ename,job,mgr,hiredate,sal FROM Sel WHERE deptno <= 10) returning *) , ins2 AS (INSERT INTO emp13 (SELECT empno,ename,job,mgr,hiredate,sal FROM Sel WHERE deptno > 10 AND deptno <= 20) returning *) INSERT INTO emp14 (SELECT empno, ename, job, mgr, hiredate, sal FROM Sel WHERE deptno > 20);

#	版本	Oracle功能	Oracle语法	迁移后的语法
11	V1R3C00SPC010	使用INSERT ALL和ELSE子句	INSERT ALL WHEN deptno <= 10 THEN INTO emp12 WHEN deptno > 10 AND deptno <= 20 THEN INTO emp13 ELSE INTO emp14 (empno, ename, job) VALUES (empno, ename, job) SELECT empno, ename, job, mgr, hiredate, sal, deptno FROM emp WHERE job='MANAGER';	WITH Sel AS (SELECT empno, ename, job, mgr, hiredate, sal, deptno FROM emp WHERE job='MANAGER') , ins1 AS (INSERT INTO emp12 (SELECT empno, ename, job, mgr, hiredate, sal, deptno FROM Sel WHERE deptno <= 10) returning *) , ins2 AS (INSERT INTO emp13 (SELECT empno, ename, job, mgr, hiredate, sal, deptno FROM Sel WHERE deptno > 10 AND deptno <= 20) returning *) INSERT INTO emp14 (empno, ename, job) (SELECT empno, ename, job FROM Sel) MINUS SELECT empno, ename, job FROM Sel WHERE deptno <= 10 OR (deptno > 10 AND deptno <= 20);

2.3 INSERT FIRST

Oracle的INSERT FIRST语句用于在first条件为真时执行INSERT语句。其他语句会被忽略。

#	版本	Oracle功能	Oracle语法	迁移后的语法
12	V1R3C00SPC010	使用INSERT FIRST	INSERT FIRST WHEN deptno <= 10 THEN INTO emp12 WHEN comm > 500 THEN INTO emp13 SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno FROM emp WHERE deptno IS NOT NULL;	WITH Sel AS (SELECT ROW_NUMBER( ) OVER( ) AS Ins_First_RN , empno, ename, job, mgr, hiredate, sal, comm, deptno FROM emp WHERE deptno IS NOT NULL) , ins1 AS (INSERT INTO emp12 (SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno FROM Sel WHERE deptno <= 10) returning 1) INSERT INTO emp13 (SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno FROM (SELECT * FROM Sel WHERE comm > 500) s1 LEFT JOIN (SELECT Ins_First_RN FROM Sel WHERE deptno <= 10) s2 ON s1.Ins_First_RN = s2.Ins_First_RN WHERE s2.Ins_First_RN IS NULL);

#	版本	Oracle功能	Oracle语法	迁移后的语法
13	V1R3C00SPC010	使用INSERT FIRST和ELSE子句	INSERT FIRST WHEN ottl < 100000 THEN INTO small_orders VALUES (oid, ottl, sid, cid,cl,cem) WHEN ottl > 100000 AND ottl < 200000 THEN INTO medium_orders (moid, mottl, msid, mcid,mcl,mcem) VALUES (oid, ottl, sid, cid,cl,cem) ELSE INTO special_orders SELECT o.order_id oid, o.orders_total ottl, o.customer_id cid , o.sales_rep_id sid, c.credit_limit cl, c.cust_email cem FROM orders1 o, customers1 c WHERE o.customer_id = c.customer_id;	WITH Sel AS (SELECT ROW_NUMBER( ) OVER( ) AS Ins_First_RN , o.order_id oid, o.orders_total ottl, o.customer_id cid, o.sales_rep_id sid, c.credit_limit cl, c.cust_email cem FROM orders1 o, customers1 c WHERE o.customer_id = c.customer_id) , ins1 AS (INSERT INTO small_orders (SELECT oid, ottl, sid, cid, cl, cem FROM Sel WHERE ottl < 100000) returning 1) , ins2 AS (INSERT INTO medium_orders (moid, mottl, msid, mcid, mcl, mcem) (SELECT oid, ottl, sid, cid, cl, cem FROM (SELECT * FROM Sel WHERE ottl > 100000 AND ottl < 200000) s1 LEFT JOIN (SELECT Ins_First_RN FROM Sel WHERE ottl < 100000) s2 ON s1.Ins_First_RN = s2.Ins_First_RN WHERE s2.Ins_First_RN IS NULL) returning 1) INSERT INTO special_orders

#	版本	Oracle功能	Oracle语法	迁移后的语法
				(SELECT oid, ottl, cid, sid, cl, cem FROM Sel s1 LEFT JOIN (SELECT Ins_First_RN FROM Sel WHERE ottl < 100000 UNION ALL SELECT Ins_First_RN FROM Sel WHERE ottl > 100000 AND ottl < 200000) s2 ON s1.Ins_First_RN = s2.Ins_First_RN WHERE s2.Ins_First_RN IS NULL);

2.4 OUTER JOIN

OUTER JOIN会返回所有满足关联条件的行。此外，如果无法为一个表中的某些行在另一个表中找到任何满足关联条件的行，则该语句会返回这些行。在Oracle中：

- 通过在WHERE条件中对表B的所有字段使用外连接操作符“+”，表A和B的左外连接返回表A中的所有行和所有满足关联条件的行。
- 通过在WHERE条件中对表A的所有字段使用外连接操作符“+”，表A和B的右外连接返回表B中的所有行和所有满足关联条件的行。

FusionInsight LibrA不支持“+”操作符。该操作符的功能通过LEFT OUTER JOIN和RIGHT OUTER JOIN关键词实现。

#	版本	Oracle功能	Oracle语法	迁移后的语法
14	V1R3C00SPC010	使用OUTER JOIN :: 1st_table = 2nd_table (+)	SELECT empno, ename, job, dname, loc FROM emp, dept WHERE emp.deptno = dept.deptno (+) AND salary > 50000;	SELECT empno, ename, job, dname, loc FROM emp LEFT OUTER JOIN dept ON emp.deptno = dept.deptno WHERE salary > 50000;

#	版本	Oracle功能	Oracle语法	迁移后的语法
15	V1 R3 C00 SPC 010	使用OUTER JOIN :: 2nd_table = 1st_table (+)	SELECT empno, ename, job, dname, loc FROM emp, dept WHERE dept.deptno = emp.deptno (+) AND salary > 50000;	SELECT empno, ename, job, dname, loc FROM dept LEFT OUTER JOIN emp ON dept.deptno = emp.deptno WHERE salary > 50000;
16	V1 R3 C00 SPC 010	使用OUTER JOIN :: 1st_table (+) = 2nd_table	SELECT empno, ename, job, dname, loc FROM emp, dept WHERE emp.deptno (+) = dept.deptno AND salary > 50000;	SELECT empno, ename, job, dname, loc FROM emp RIGHT OUTER JOIN dept ON emp.deptno = dept.deptno WHERE salary > 50000;
17	V1 R3 C00 SPC 010	使用OUTER JOIN :: 2nd_table (+) = 1st_table	SELECT empno, ename, job, dname, loc FROM emp, dept WHERE dept.deptno (+) = emp.deptno AND salary > 50000;	SELECT empno, ename, job, dname, loc FROM dept RIGHT OUTER JOIN emp ON dept.deptno = emp.deptno WHERE salary > 50000;
18	V1 R3 C00 SPC 010	使用OUTER JOIN，并指定表别名	SELECT s.supplier_id, s.supplier_name, o.order_date FROM suppliers s, orders o WHERE s.supplier_id = o.supplier_id (+) AND s.suppller_id > 1000;	SELECT s.supplier_id, s.supplier_name, o.order_date FROM suppliers s LEFT OUTER JOIN orders o ON s.supplier_id = o.supplier_id WHERE s.suppller_id > 1000;
19	V1 R3 C00 SPC 010	使用OUTER JOIN并指定多个关联和表达式	SELECT b.zoneno, b.phybrno, t1.zoneno, t1.brno FROM schema1.tab1 B, schema2.tab2 t1 WHERE SUBSTR(b.zoneno, 1, 4) = SUBSTR(t1.zoneno, 2, 4) (+) AND SUBSTR(b.phybrno, 1, 4) = SUBSTR(t1.brno, 2, 4)(+);	SELECT b.zoneno, b.phybrno, t1.zoneno, t1.brno FROM schema1.tab1 B LEFT OUTER JOIN schema2.tab2 t1 ON SUBSTR(b.zoneno, 1, 4) = SUBSTR(t1.zoneno, 2, 4) AND SUBSTR(b.phybrno, 1, 4) = SUBSTR(t1.brno, 2, 4);

#	版本	Oracle功能	Oracle语法	迁移后的语法
20	V1 R3 C00 SPC 010	使用OUTER JOIN，并指定多个表和子查询	SELECT stru_id, organ1_no, SUBSTR(p_i_date, 1, 8), v_task_date1, TA000 FROM ccm_TA280362_h d , (SELECT * FROM tab1 where table_code = 'DA200251007' ORDER BY 1) d1 , (SELECT D.dict_cd, D.dict_name FROM tab2 D WHERE work_dt = SUBSTR(p_i_date, 1, 6)) d2 , (SELECT * FROM tab3 where table_code = 'DA200251011') d3 , (SELECT tab4.dict_cd, tab5.c2 FROM tab4, tab5 WHERE tab5.c1 (+) = tab4.c1 AND table_code = 'DA200251012') d4 WHERE d.TA005 = d1.dict_cd(+) AND d.TA007 = d2.dict_cd(+) AND d.TA012 = d3.dict_cd(+) AND d.TA013 = d4.dict_cd(+) AND d.TA000 = v_task_date1;	SELECT stru_id, organ1_no, SUBSTR(p_i_date, 1, 8), v_task_date1, TA000 FROM ccm_TA280362_h d LEFT OUTER JOIN (SELECT * FROM tab1 where table_code = 'DA200251007' ORDER BY 1) d1 ON d.TA005 = d1.dict_cd LEFT OUTER JOIN (SELECT D.dict_cd, D.dict_name FROM tab2 D WHERE work_dt = SUBSTR(p_i_date, 1, 6)) d2 ON d.TA007 = d2.dict_cd LEFT OUTER JOIN (SELECT * FROM tab3 where table_code = 'DA200251011') d3 ON d.TA012 = d3.dict_cd LEFT OUTER JOIN (SELECT tab4.dict_cd, tab5.c2 FROM tab5 RIGHT OUTER JOIN tab4 ON tab5.c1 = tab4.c1 WHERE table_code = 'DA200251012') d4 ON d.TA013 = d4.dict_cd WHERE d.TA000 = v_task_date1;

#	版本	Oracle功能	Oracle语法	迁移后的语法
21	V1 R3 C00 SPC 010	使用OUTER JOIN并指定静态值	SELECT d.department_name, e.employee_name FROM departments d, employees e WHERE d.department_id = e.department_id (+) AND e.salary (+) >= 2000 AND d.department_id >= 30 ORDER BY d.department_name, e.employee_name;	SELECT d.department_name, e.employee_name FROM departments d LEFT OUTER JOIN employees e ON d.department_id = e.department_id AND e.salary >= 2000 WHERE d.department_id >= 30 ORDER BY d.department_name, e.employee_name;

2.5 模式对象

2.5.1 表

FusionInsight LibrA仅支持RANGE分区，不支持其他分区类型（如LIST和HASH）或子分区。

FusionInsight LibrA中的所有表均不支持存储参数PCTINCREASE。此外，分区表不支持存储参数（如pctfree、minextents、maxextents）。

#	版本	Oracle功能	Oracle语法	迁移后的语法
22	V1R3C00SPC010	在CREATE TABLE中使用PCTINCREASE	CREATE TABLE tab1 (col1 < datatype > , col2 < datatype > , ... , colN < datatype >) TABLESPACE testts PCTFREE 10 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 5 M NEXT 5 M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);	CREATE TABLE tab1 (col1 < datatype > , col2 < datatype > , ... , colN < datatype >) TABLESPACE testts PCTFREE 10 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 5 M NEXT 5 M MINEXTENTS 1 MAXEXTENTS UNLIMITED);

#	版本	Oracle功能	Oracle语法	迁移后的语法
23	V1R3C00SPC010	使用范围分区表，并指定存储参数	CREATE TABLE CCM_TA550002_H (STRU_ID VARCHAR2 (10) , ORGAN1_NO VARCHAR2 (10) , ORGAN2_NO VARCHAR2 (10)) partition by range (ORGAN2_NO) (partition CCM_TA550002_01 VALUES LESS than ('00100') tablespace users pctfree 10 initrans 1 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited pctincrease 0) , partition CCM_TA550002_02 VALUES LESS than ('00200') tablespace users pctfree 10 initrans 1 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited pctincrease 0));	CREATE TABLE CCM_TA550002_H (STRU_ID VARCHAR2 (10) , ORGAN1_NO VARCHAR2 (10) , ORGAN2_NO VARCHAR2 (10)) partition BY range (ORGAN2_NO) (partition CCM_TA550002_01 VALUES LESS than ('00100') tablespace users , partition CCM_TA550002_02 VALUES LESS than ('00200') tablespace users);

#	版本	Oracle功能	Oracle语法	迁移后的语法
24	V1R3C00SPC010	范围分区中的子分区	CREATE TABLE sample_regional_sales (deptno NUMBER , item_no varchar2 (20) , txn_date DATE , txn_amount NUMBER , state varchar2 (2)) PARTITION BY RANGE (txn_date) SUBPARTITION BY LIST (state) (PARTITION q1_1999 VALUES LESS THAN (TO_DATE(‘1-APR-1999’ ,'DD-MON-YYYY')) (SUBPARTITION q1_1999_northwest VALUES ('OR', 'WA') , SUBPARTITION q1_1999_southwest VALUES ('AZ', 'UT', 'NM') , SUBPARTITION q1_1999_northeast VALUES ('NY', 'VM', 'NJ') , SUBPARTITION q1_1999_southeast VALUES ('FL', 'GA') , SUBPARTITION q1_others VALUES (DEFAULT)) , PARTITION q2_1999 VALUES LESS THAN	CREATE TABLE sample_regional_sales (deptno NUMBER , item_no varchar2 (20) , txn_date DATE , txn_amount NUMBER , state varchar2 (2)) PARTITION BY RANGE (txn_date) (PARTITION q1_1999 VALUES LESS THAN (TO_DATE('1-APR-1999' ,'DD-MON-YYYY')) , PARTITION q2_1999 VALUES LESS THAN (TO_DATE('1-JUL-1999' ,'DD-MON-YYYY')));

#	版本	Oracle功能	Oracle语法	迁移后的语法
			(TO_DATE('1-JUL-1999' ,'DD-MON-YYYY')) (SUBPARTITION q2_1999_northwest VALUES ('OR', 'WA') , SUBPARTITION q2_1999_southwest VALUES ('AZ', 'UT', 'NM') , SUBPARTITION q2_1999_northeast VALUES ('NY', 'VM', 'NJ') , SUBPARTITION q2_1999_southeast VALUES ('FL', 'GA') , SUBPARTITION q2_1999_northcentral VALUES ('SD', 'WI') , SUBPARTITION q2_1999_southcentral VALUES ('OK', 'TX')));	

#	版本	Oracle功能	Oracle语法	迁移后的语法
25	V1R3C00SPC010	范围分区中的子分区模板	CREATE TABLE composite_rng_rng (cust_id NUMBER (10) , cust_name VARCHAR2 (25) , cust_state VARCHAR2 (2) , time_id DATE) PARTITION BY RANGE (time_id) SUBPARTITION BY RANGE (cust_id) SUBPARTITION TEMPLATE (SUBPARTITION original VALUES LESS THAN (1001) , SUBPARTITION acquired VALUES LESS THAN (8001) , SUBPARTITION recent VALUES LESS THAN (MAXVALUE)) (PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000' ,'DD/MM/YYYY')) , PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005' ,'DD/MM/YYYY')) , PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010' ,'DD/MM/YYYY'))	CREATE TABLE composite_rng_rng (cust_id NUMBER (10) , cust_name VARCHAR2 (25) , cust_state VARCHAR2 (2) , time_id DATE) PARTITION BY RANGE (time_id) (PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000' ,'DD/MM/YYYY')) , PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005' ,'DD/MM/YYYY')) , PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010' ,'DD/MM/YYYY')) , PARTITION future VALUES LESS THAN (MAXVALUE));

#	版本	Oracle功能	Oracle语法	迁移后的语法
			, PARTITION future VALUES LESS THAN (MAXVALUE));	
26	V1R3C00SPC010	使用列表分区表	CREATE TABLE q1_sales_by_region (deptno NUMBER , deptname varchar2 (20) , quarterly_sales NUMBER (10,2) , state varchar2 (2)) PARTITION BY LIST (state) (PARTITION q1_northwest VALUES ('OR', 'WA') , PARTITION q1_southwest VALUES ('AZ', 'UT', 'NM') , PARTITION q1_northeast VALUES ('NY', 'VM', 'NJ') , PARTITION q1_southcentral VALUES ('OK', 'TX'));	CREATE TABLE q1_sales_by_region (deptno NUMBER , deptname varchar2 (20) , quarterly_sales NUMBER (10,2) , state varchar2 (2));

#	版本	Oracle功能	Oracle语法	迁移后的语法
27	V1R3C00SPC010	使用列表分区表，并指定存储参数	CREATE TABLE store_master (Store_id NUMBER , Store_address VARCHAR2 (40) , City VARCHAR2 (30) , State VARCHAR2 (2) , zip VARCHAR2 (10) , manager_id NUMBER) TABLESPACE users STORAGE (INITIAL 100 k NEXT 100 k PCTINCREASE 0) PARTITION BY LIST (city) (PARTITION south_florida VALUES ('MIA', 'ORL') TABLESPACE users STORAGE (INITIAL 100 k NEXT 100 k PCTINCREASE 0) , PARTITION north_florida VALUES ('JAC', 'TAM', 'PEN') TABLESPACE users STORAGE (INITIAL 100 k NEXT 100 k PCTINCREASE 0) , PARTITION south_georga VALUES ('BRU', 'WAY', 'VAL') TABLESPACE users STORAGE (INITIAL 100 k NEXT 100 k PCTINCREASE 0) , PARTITION north_georgia	CREATE TABLE store_master (Store_id NUMBER , Store_address VARCHAR2 (40) , City VARCHAR2 (30) , State VARCHAR2 (2) , zip VARCHAR2 (10) , manager_id NUMBER) TABLESPACE users STORAGE (INITIAL 100 k NEXT 100 k);

#	版本	Oracle功能	Oracle语法	迁移后的语法
			VALUES ('ATL', 'SAV', NULL));	

#	版本	Oracle功能	Oracle语法	迁移后的语法
28	V1R3C00SPC010	列表分区中的子分区	CREATE TABLE t_accounts (id NUMBER , account_number NUMBER , customer_id NUMBER , balance NUMBER , branch_id NUMBER , region VARCHAR(2) , status VARCHAR2 (1)) PARTITION BY LIST (region) SUBPARTITION BY RANGE (balance) (PARTITION p_northwest VALUES ('OR', 'WA') (SUBPARTITION p_nw_low VALUES LESS THAN (1000) , SUBPARTITION p_nw_average VALUES LESS THAN (10000) , SUBPARTITION p_nw_high VALUES LESS THAN (100000) , SUBPARTITION p_nw_extord VALUES LESS THAN (MAXVALUE)) , PARTITION p_southwest VALUES ('AZ', 'UT', 'NM') (SUBPARTITION p_sw_low	CREATE TABLE t_accounts (id NUMBER , account_number NUMBER , customer_id NUMBER , balance NUMBER , branch_id NUMBER , region VARCHAR(2) , status VARCHAR2 (1));

#	版本	Oracle功能	Oracle语法	迁移后的语法
			VALUES LESS THAN (1000) , SUBPARTITION p_sw_average VALUES LESS THAN (10000) , SUBPARTITION p_sw_high VALUES LESS THAN (100000) , SUBPARTITION p_sw_extord VALUES LESS THAN (MAXVALUE))) ENABLE ROW MOVEMENT;	

#	版本	Oracle功能	Oracle语法	迁移后的语法
29	V1R3C00SPC010	列表分区中的子分区模板	CREATE TABLE q1_sales_by_region (deptno NUMBER , deptname varchar2 (20) , quarterly_sales NUMBER (10,2) , state varchar2 (2)) PARTITION BY LIST (state) SUBPARTITION BY RANGE (quarterly_sales) SUBPARTITION TEMPLATE (SUBPARTITION original VALUES LESS THAN (1001) , SUBPARTITION acquired VALUES LESS THAN (8001) , SUBPARTITION recent VALUES LESS THAN (MAXVALUE)) (PARTITION q1_northwest VALUES ('OR', 'WA') , PARTITION q1_southwest VALUES ('AZ', 'UT', 'NM') , PARTITION q1_northeast VALUES ('NY', 'VM', 'NJ') , PARTITION q1_southcentral VALUES ('OK', 'TX'));	CREATE TABLE q1_sales_by_region (deptno NUMBER , deptname varchar2 (20) , quarterly_sales NUMBER (10,2) , state varchar2 (2));

#	版本	Oracle功能	Oracle语法	迁移后的语法
30	V1R3C00SPC010	从另一张表中创建列表分区表	CREATE TABLE tab1_list PARTITION BY LIST (col1) (partition part1 VALUES (1) , partition part2 VALUES (2, 3, 4) , partition part3 VALUES (DEFAULT)) AS SELECT * FROM tab1;	CREATE TABLE tab1_list AS (SELECT * FROM tab1);

#	版本	Oracle功能	Oracle语法	迁移后的语法
31	V1R3C00SPC010	使用ALTER TABLE，并添加主键	CREATE TABLE tab3 (col1 < datatype > , col2 < datatype > ... , colN < datatype >) tablespace tablespace1 pctfree 10 initrans 1 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited pctincrease 0); ALTER TABLE tab3 ADD CONSTRAINT PK_tab3 PRIMARY KEY (col1, col2) pctfree 10 initrans 2 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited pctincrease 0);	CREATE TABLE tab3 (col1 < datatype > , col2 < datatype > ... , colN < datatype > , CONSTRAINT PK_tab3 PRIMARY KEY (col1, col2)) tablespace tablespace1 pctfree 10 initrans 1 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited);

#	版本	Oracle功能	Oracle语法	迁移后的语法
32	V1R3C00SPC010	使用ALTER TABLE，并添加主键、使用索引表空间	CREATE TABLE tab4 (col1 < datatype > , col2 < datatype > ... , colN < datatype >) tablespace tablespace1 pctfree 10 initrans 1 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited pctincrease 0); ALTER TABLE tab4 ADD CONSTRAINT PK_tab4 PRIMARY KEY (col1, col2) USING index tablespace tablespace2 pctfree 10 initrans 2 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited pctincrease 0);	CREATE TABLE tab4 (col1 < datatype > , col2 < datatype > ... , colN < datatype > , CONSTRAINT PK_tab4 PRIMARY KEY (col1, col2) USING index tablespace tablespace2) tablespace tablespace1 pctfree 10 initrans 1 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited);

#	版本	Oracle功能	Oracle语法	迁移后的语法
33	V1R3C00SPC010	创建无日志表	CREATE TABLE tab5 (col1 < datatype > , col2 < datatype > , ... , colN < datatype >) NOLOGGING TABLESPACE testts PCTFREE 10 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 5 M NEXT 5 M MINEXTENTS 1 MAXEXTENTS UNLIMITED PCTINCREASE 0);	CREATE UNLOGGED TABLE tab5 (col1 < datatype > , col2 < datatype > , ... , colN < datatype >) TABLESPACE testts PCTFREE 10 INITRANS 1 MAXTRANS 255 STORAGE (INITIAL 5 M NEXT 5 M MINEXTENTS 1 MAXEXTENTS UNLIMITED);
34	V1R3C00SPC010	使用另一张表创建无日志表	CREATE TABLE tab8 NOLOGGING AS SELECT * FROM tab5 WHERE 1=2;	CREATE UNLOGGED TABLE tab8 AS (SELECT * FROM tab5 WHERE 1=2);

2.5.2 索引

#	版本	Oracle功能	Oracle语法	迁移后的语法
35	V1R3C00SPC010	指定索引名和模式名	CREATE INDEX scott.ix_tab1_col1 ON scott.tab1 (col1) tablespace users pctfree 10 initrans 2 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited);	CREATE INDEX ix_tab1_col1 ON scott.tab1 (col1) tablespace users pctfree 10 initrans 2 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited);

#	版本	Oracle功能	Oracle语法	迁移后的语法
36	V1R3C00SPC010	使用“局部分区索引::范围分区表”，并指定全局分区索引	CREATE TABLE sales (prod_id NUMBER(6) , quantity_sold NUMBER(3) , amount_sold NUMBER(10,2) , time_id DATE) PARTITION BY RANGE (time_id) (PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006' ,'dd-MON-yyyy')) TABLESPACE users , PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006' ,'dd-MON-yyyy')) TABLESPACE users); CREATE INDEX amount_sold_ix ON sales(amount_sold, quantity_sold) GLOBAL PARTITION BY RANGE(amount_sold) (PARTITION p_100 VALUES LESS THAN (100) , PARTITION p_1000 VALUES LESS THAN (1000) , PARTITION p_greater_than_1000000 VALUES	CREATE TABLE sales (prod_id NUMBER(6) , quantity_sold NUMBER(3) , amount_sold NUMBER(10,2) , time_id DATE) PARTITION BY RANGE (time_id) (PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy')) TABLESPACE users , PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy')) TABLESPACE users); CREATE INDEX amount_sold_ix ON sales (amount_sold, quantity_sold) LOCAL;

#	版本	Oracle功能	Oracle语法	迁移后的语法
			LESS THAN (maxvalue));	

#	版本	Oracle功能	Oracle语法	迁移后的语法
37	V1R3C00SPC010	使用列表分区表，并指定全局分区索引	CREATE TABLE sales_by_region (deptno NUMBER , deptname varchar2 (20) , quarterly_sales NUMBER (10,2) , quantity_sold NUMBER(3) , state varchar2 (2)) PARTITION BY LIST (state) (PARTITION q1_northwest VALUES ('OR', 'WA') , PARTITION q1_southwest VALUES ('AZ', 'UT', 'NM') , PARTITION q1_northeast VALUES ('NY', 'VM', 'NJ') , PARTITION q1_southcentral VALUES ('OK', 'TX')); CREATE INDEX sale_by_reg_ix ON sales_by_region (quarterly_sales,quantity_sold) GLOBAL PARTITION BY RANGE(quarterly_sales) (PARTITION p_100 VALUES LESS THAN (100)	CREATE TABLE sales_by_region (deptno NUMBER , deptname varchar2 (20) , quarterly_sales NUMBER (10,2) , quantity_sold NUMBER(3) , state varchar2 (2)); CREATE INDEX sale_by_reg_ix ON sales_by_region (quarterly_sales,quantity_sold);

#	版本	Oracle功能	Oracle语法	迁移后的语法
			, PARTITION p_1000 VALUES LESS THAN (1000) , PARTITION p_greater_than_1000000 VALUES LESS THAN (maxvalue));	

#	版本	Oracle功能	Oracle语法	迁移后的语法
38	V1R3C00SPC010	使用“局部分区索引::范围分区表”，并指定局部分区索引	CREATE TABLE sales2 (prod_id NUMBER(6) , quantity_sold NUMBER(3) , amount_sold NUMBER(10,2) , time_id DATE) PARTITION BY RANGE (time_id) (PARTITION sales2_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006' ,'dd-MON-yyyy')) TABLESPACE users , PARTITION sales2_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006' ,'dd-MON-yyyy')) TABLESPACE users); CREATE INDEX amount_sold_ix ON sales2(amount_sold) LOCAL (PARTITION sales_q1_ix , PARTITION sales_q2_ix);	CREATE TABLE sales2 (prod_id NUMBER(6) , quantity_sold NUMBER(3) , amount_sold NUMBER(10,2) , time_id DATE) PARTITION BY RANGE (time_id) (PARTITION sales2_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy')) TABLESPACE users , PARTITION sales2_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy')) TABLESPACE users); CREATE INDEX amount_sold_ix ON sales2(amount_sold) LOCAL (PARTITION sales_q1_ix , PARTITION sales_q2_ix);

#	版本	Oracle功能	Oracle语法	迁移后的语法
39	V1R3C00SPC010	使用“局部分区索引::含有分区索引的范围分区表”，不指定分区名	CREATE TABLE sales3 (prod_id NUMBER(6) , quantity_sold NUMBER(3) , amount_sold NUMBER(10,2) , time_id DATE) PARTITION BY RANGE (time_id) (PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006' ,'dd-MON-yyyy')) TABLESPACE users , PARTITION sales_q2_2006 VALUES LESS THAN(TO_DATE('01-JUL-2006' ,'dd-MON-yyyy')) TABLESPACE users); CREATE INDEX amount_sold_ix3 ON sales3(amount_sold) <GLOBAL/LOCAL/>;	CREATE TABLE sales3 (prod_id NUMBER(6) , quantity_sold NUMBER(3) , amount_sold NUMBER(10,2) , time_id DATE) PARTITION BY RANGE (time_id) (PARTITION sales_q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006','dd-MON-yyyy')) TABLESPACE users , PARTITION sales_q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006','dd-MON-yyyy')) TABLESPACE users); CREATE INDEX amount_sold_ix3 ON sales3(amount_sold) LOCAL;

2.5.3 临时表

#	版本	Oracle功能	Oracle语法	迁移后的语法
40	V1R3C00SPC010	使用临时表和ON COMMIT DELETE ROWS	CREATE GLOBAL TEMPORARY TABLE schema1.temp_tbl1 (col1 VARCHAR2 (400) , col2 DATE NOT NULL) ON COMMIT DELETE ROWS;	CREATE LOCAL TEMPORARY TABLE schema1_temp_tbl1 (col1 VARCHAR2 (400) , col2 DATE NOT NULL) ON COMMIT PRESERVE ROWS;
41	V1R3C00SPC010	使用临时表和ON COMMIT PRESERVE ROWS	CREATE GLOBALTEMPORARY TABLE schema2.temp_tbl2 (col1 VARCHAR2 (400) , col2 DATE NOT NULL) ON COMMIT PRESERVE ROWS; comment ON TABLE schema2.temp_tbl2 IS 'ÁÙÊ±×Öµä±í1';	CREATE LOCAL TEMPORARY TABLE schema2_temp_tbl2 (col1 VARCHAR2 (400) , col2 DATE NOT NULL) ON COMMIT PRESERVE ROWS; comment ON TABLE schema2_temp_tbl2 IS 'ÁÙÊ±×Öµä±í1';

2.5.4 FusionInsight LibrA 关键词

#	版本	Oracle功能	Oracle语法	迁移后的语法
42	V1R3C00SPC010	CREATE TABLE中的FusionInsight LibrA关键词	CREATE TABLE NAME (NAME VARCHAR2(50) NOT NULL , VALUE VARCHAR2(255) , DESCRIPTION VARCHAR2(4000) , LIMIT NUMBER(9)) tablespace users pctfree 10 initrans 1 maxtrans 255 storage (initial 256K next 256K minextents 1 maxextents unlimited); SELECT NAME, VALUE, DESCRIPTION, LIMIT FROM NAME;	CREATE TABLE "NAME" ("NAME" VARCHAR2 (50) NOT NULL , VALUE VARCHAR2 (255) , DESCRIPTION VARCHAR2 (4000) , "LIMIT" NUMBER (9)) tablespace users pctfree 10 initrans 1 maxtrans 255 storage (initial 256 K NEXT 256 K minextents 1 maxextents unlimited); SELECT "NAME", VALUE, DESCRIPTION, "LIMIT" FROM "NAME";

2.5.5 序列

#	版本	Oracle功能	Oracle语法	迁移后的语法
43	V1R3C00SPC010	序列	CREATE SEQUENCE GROUP_DEF_SEQUENE minvalue 1 maxvalue 10000000000000000000 start with 1152 increment by 1 cache 50 order;	CREATE SEQUENCE GROUP_DEF_SEQUENE minvalue 1 maxvalue 9223372036854775807 start WITH 1152 increment BY 1 cache 50;
44	V1R3C00SPC010	使用序列，并指定NOCACHE	CREATE SEQUENCE customers_seq START WITH 1000 INCREMENT BY 1 NOCACHE NOCYCLE;	CREATE SEQUENCE customers_seq START WITH 1000 INCREMENT BY 1 NOCYCLE;

2.5.6 视图

#	版本	Oracle功能	Oracle语法	迁移后的语法
45	V1R3C00SPC010	为视图指定模式名和FusionInsight LibrA关键词	CREATE OR REPLACE VIEW schema1.v_view_name AS SELECT dict_code code, dict_name name FROM tab1 WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT#PRTYCODE' AND WORK_WT = (SELECT MAX(WORK_DT) FROM tab2 WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT#PRTYCODE') AND WORK_WT = (SELECT MAX(WORK_DT) FROM schema2.tab3 WHERE BEAN_CODE ='LOA_PERSONAL_ACCOUNT#PRTYCODE'); /	CREATE OR REPLACE VIEW schema1.v_view_name AS (SELECT dict_code code, dict_name "name" FROM schema1.tab1 WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT#PRTYCODE' AND WORK_WT = (SELECT MAX(WORK_DT) FROM schema1.tab2 WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT#PRTYCODE') AND WORK_WT = (SELECT MAX(WORK_DT) FROM schema2.tab3 WHERE BEAN_CODE = 'LOA_PERSONAL_ACCOUNT#PRTYCODE'));

2.5.7 PURGE

在Oracle中，DROP TABLE语句将一张表放入回收站。PURGE语句用于将一张表或一个索引移动至回收站，并释放所有与该对象相关的空间，或清空整个回收站，或从回收站中删除一个已删除表空间的部分内容。

#	版本	Oracle功能	Oracle语法	迁移后的语法
46	V1R3C00SPC010	使用DROP，并指定PURGE	EXECUTE IMMEDIATE 'drop table TEMP_PCM_TA500273_H PURGE'; DROP TABLE test.emp PURGE;	EXECUTE IMMEDIATE ' drop table TEMP_PCM_TA500273_H PURGE'; DROP TABLE test.emp;
47	V1R3C00SPC010	使用PURGE语句	PURGE RECYCLEBIN; PURGE INDEX ix_emp_ename; PURGE DBA_RECYCLEBIN; PURGE TABLESPACE testts USER test; EXECUTE IMMEDIATE 'PURGE RECYCLEBIN';	/* PURGE RECYCLEBIN; */ /* PURGE INDEX ix_emp_ename; */ /* PURGE DBA_RECYCLEBIN; */ /* PURGE TABLESPACE testts USER test; */ /* EXECUTE IMMEDIATE 'PURGE RECYCLEBIN'; */

2.6 SELECT

2.6.1 子句顺序

#	版本	Oracle功能	Oracle语法	迁移后的语法
48	V1R3C00SPC010	SELECT子句顺序：HAVING在前，GROUP BY在后	SELECT t1.col1, t2.col2 , Aggregate_fn1, Aggregate_fn2 FROM tab1 t1, tab2 t2 WHERE … HAVING Aggregate_fnX <operator> <value> GROUP BY t1.col1, t2.col2 ORDER BY …	SELECT t1.col1, t2.col2 , Aggregate_fn1, Aggregate_fn2 FROM tab1 t1, tab2 t2 WHERE … GROUP BY t1.col1, t2.col2 HAVING Aggregate_fnX <operator> <value> ORDER BY …

2.6.2 扩展 Group By 子句

指定GROUP BY子句可让数据库将所选行基于expr(s)的值分组。如果该子句包含CUBE，ROLLUP，或GROUPING SETS扩展项，则数据库除正则分组外还会进行超聚合分组。这些功能在FusionInsight LibrA中不可用，可通过UNION ALL操作符实现。

#	版本	Oracle功能	Oracle语法	迁移后的语法
49	V1R3C00SPC010	使用ROLLUP	SELECT d.dname, e.job, MAX(e.sal) FROM emp e RIGHT OUTER JOIN dept d ON e.deptno=d.deptno WHERE e.job IS NOT NULL GROUP BY ROLLUP (d.dname, e.job);	SELECT dname, job, ColumnAlias1 FROM (SELECT MAX(e.sal) AS ColumnAlias1, d.dname, e.job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL GROUP BY d.dname ,e.job UNION ALL SELECT MAX(e.sal) AS ColumnAlias1, d.dname, NULL AS job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL GROUP BY d.dname UNION ALL SELECT MAX(e.sal) AS ColumnAlias1, NULL AS dname, NULL AS job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL);

#	版本	Oracle功能	Oracle语法	迁移后的语法
50	V1R3C00SPC010	使用ROLLUP，并指定ORDER BY子句和RANK	SELECT syear AS sale_year, prd_type_id AS product_type_id, SUM(amount) sale_amount, RANK() OVER (ORDER BY SUM(amount) DESC) AS rnk FROM all_sales GROUP BY ROLLUP (syear, prd_type_id) ORDER BY syear, prd_type_id;	SELECT sale_year, product_type_id, sale_amount, RANK () OVER(ORDER BY sale_amount DESC) AS rnk FROM (SELECT syear AS sale_year, prd_type_id AS product_type_id, SUM (amount) sale_amount, prd_type_id, syear FROM all_sales GROUP BY syear ,prd_type_id UNION ALL SELECT syear AS sale_year, NULL AS product_type_id, SUM (amount) sale_amount, NULL AS prd_type_id, syear FROM all_sales GROUP BY syear UNION ALL SELECT NULL AS sale_year, NULL AS product_type_id, SUM (amount) sale_amount, NULL AS prd_type_id, NULL AS syear FROM all_sales) ORDER BY sale_year, product_type_id;

#	版本	Oracle功能	Oracle语法	迁移后的语法
51	V1R3C00SPC010	使用ROLLUP，并指定HAVING子句	SELECT deptno, job, AVG(sal) AS avg_sal FROM emp GROUP BY ROLLUP(deptno, job) HAVING AVG(sal) > 3000 ORDER BY deptno, job;	SELECT deptno, job, avg_sal FROM (SELECT AVG (sal) AS avg_sal, deptno, job FROM emp GROUP BY deptno ,job HAVING AVG(sal) > 3000 UNION ALL SELECT AVG(sal) AS avg_sal, deptno, NULL AS job FROM emp GROUP BY deptno HAVING AVG(sal) > 3000 UNION ALL SELECT AVG(sal) AS avg_sal, NULL AS deptno, NULL AS job FROM emp HAVING AVG(sal) > 3000) ORDER BY deptno, job;

#	版本	Oracle功能	Oracle语法	迁移后的语法
52	V1R3C00SPC010	使用ROLLUP，并指定CASE	SELECT empid, designation , CASE WHEN DEPTNO=10 THEN 'Department 10' WHEN DEPTNO=20 THEN 'Department 20' WHEN DEPTNO=30 THEN 'Department 30' Else 'No Such department' END AS endcase , SUM(SALARY) FROM EMPLOYEES GROUP BY ROLLUP(empid, designation,deptno);	SELECT empid, designation, endcase, ColumnAlias1 FROM (SELECT CASE WHEN DEPTNO = 10 THEN 'Department 10' WHEN DEPTNO = 20 THEN 'Department 20' WHEN DEPTNO = 30 THEN 'Department 30' ELSE 'No Such department' END AS endcase , SUM (SALARY) AS ColumnAlias1 , deptno, designation, empid FROM EMPLOYEES GROUP BY empid , designation , deptno UNION ALL SELECT CASE WHEN NULL = 10 THEN 'Department 10' WHEN NULL = 20 THEN 'Department 20' WHEN NULL = 30 THEN 'Department 30' ELSE 'No Such department' END AS endcase , SUM (SALARY) AS ColumnAlias1 , NULL AS deptno, designation, empid FROM EMPLOYEES GROUP BY empid ,designation UNION ALL SELECT CASE WHEN NULL = 10 THEN 'Department 10' WHEN NULL = 20 THEN 'Department 20' WHEN NULL = 30 THEN 'Department 30' ELSE 'No Such department' END AS endcase

#	版本	Oracle功能	Oracle语法	迁移后的语法
				, SUM (SALARY) AS ColumnAlias1 , NULL AS deptno, NULL AS designation, empid FROM EMPLOYEES GROUP BY empid UNION ALL SELECT CASE WHEN NULL = 10 THEN 'Department 10' WHEN NULL = 20 THEN 'Department 20' WHEN NULL = 30 THEN 'Department 30' ELSE 'No Such department' END AS endcase , SUM (SALARY) AS ColumnAlias1 , NULL AS deptno, NULL AS designation, NULL AS empid FROM EMPLOYEES);

#	版本	Oracle功能	Oracle语法	迁移后的语法
53	V1R3C00SPC010	使用CUBE	SELECT d.dname, e.job, MAX(e.sal) FROM emp e RIGHT OUTER JOIN dept d ON e.deptno=d.deptno WHERE e.job IS NOT NULL GROUP BY CUBE (d.dname, e.job);	SELECT dname, job, ColumnAlias1 FROM (SELECT MAX(e.sal) AS ColumnAlias1, d.dname, e.job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL GROUP BY d.dname, e.job UNION ALL SELECT MAX(e.sal) AS ColumnAlias1, d.dname, NULL AS job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL GROUP BY d.dname UNION ALL SELECT MAX(e.sal) AS ColumnAlias1, NULL AS dname, e.job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL GROUP BY e.job UNION ALL SELECT MAX(e.sal) AS ColumnAlias1, NULL AS dname, NULL AS job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL);

#	版本	Oracle功能	Oracle语法	迁移后的语法
54	V1R3C00SPC010	使用CUBE，并指定ORDER BY和ROW_NUMBER	SELECT syear AS sale_year, prd_type_id AS product_type_id, SUM(amount) sale_amount,ROW_NUMBER() OVER (ORDER BY SUM(amount) DESC) AS rnk FROM all_sales GROUP BY CUBE (syear, prd_type_id) ORDER BY syear, prd_type_id;	SELECT sale_year, product_type_id, sale_amount , ROW_NUMBER( ) OVER(ORDER BY sale_amount DESC) AS rnk FROM (SELECT syear AS sale_year, prd_type_id AS product_type_id , SUM (amount) sale_amount, prd_type_id, syear FROM all_sales GROUP BY syear ,prd_type_id UNION ALL SELECT syear AS sale_year, NULL AS product_type_id , SUM (amount) sale_amount , NULL AS prd_type_id, syear FROM all_sales GROUP BY syear UNION ALL SELECT NULL AS sale_year, prd_type_id AS product_type_id , SUM (amount) sale_amount, prd_type_id, NULL AS syear FROM all_sales GROUP BY prd_type_id UNION ALL SELECT NULL AS sale_year, NULL AS product_type_id , SUM (amount) sale_amount , NULL AS prd_type_id, NULL AS syear FROM all_sales) ORDER BY sale_year, product_type_id;

#	版本	Oracle功能	Oracle语法	迁移后的语法
55	V1R3C00SPC010	使用CUBE，并指定HAVING子句	SELECT deptno, job, AVG(sal) AS avg_sal FROM emp GROUP BY CUBE(deptno, job) HAVING AVG(sal) > 3000 ORDER BY deptno, job;	SELECT deptno, job, avg_sal FROM (SELECT AVG (sal) AS avg_sal, deptno, job FROM emp GROUP BY deptno, job HAVING AVG (sal) > 3000 UNION ALL SELECT AVG (sal) AS avg_sal, deptno, NULL AS job FROM emp GROUP BY deptno HAVING AVG (sal) > 3000 UNION ALL SELECT AVG (sal) AS avg_sal, NULL AS deptno, job FROM emp GROUP BY job HAVING AVG (sal) > 3000 UNION ALL SELECT AVG (sal) AS avg_sal, NULL AS deptno, NULL AS job FROM emp HAVING AVG (sal) > 3000) ORDER BY deptno, job;

#	版本	Oracle功能	Oracle语法	迁移后的语法
56	V1R3C00SPC010	使用GROUPING SETS	SELECT d.dname, e.job, MAX(e.sal) FROM emp e RIGHT OUTER JOIN dept d ON e.deptno=d.deptno WHERE e.job IS NOT NULL GROUP BY GROUPING SETS(d.dname, e.job);	SELECT dname, job, ColumnAlias1 FROM (SELECT MAX(e.sal) AS ColumnAlias1 , d.dname, NULL AS job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL GROUP BY d.dname UNION ALL SELECT MAX(e.sal) AS ColumnAlias1 , NULL AS dname, e.job FROM emp e RIGHT OUTER JOIN dept d ON e.deptno = d.deptno WHERE e.job IS NOT NULL GROUP BY e.job);
57	V1R3C00SPC010	使用GROUPING SETS，并指定ORDER BY子句和RANK	SELECT syear AS sale_year, prd_type_id AS product_type_id, SUM(amount) sale_amount, RANK() OVER (ORDER BY SUM(amount) DESC) AS rnk FROM all_sales GROUP BY GROUPING SETS(syear, prd_type_id) ORDER BY syear, prd_type_id;	CAST((CAST('20150801' AS DATE) - 1) AS DATE)

#	版本	Oracle功能	Oracle语法	迁移后的语法
58	V1R3C00SPC010	使用GROUPING SETS，并指定HAVING子句	SELECT deptno, job, AVG(sal) AS avg_sal FROM emp GROUP BY GROUPING SETS(deptno, job) HAVING AVG(sal) > 3000 ORDER BY deptno, job;	SELECT deptno, job, avg_sal FROM (SELECT AVG(sal) AS avg_sal , deptno, NULL AS job FROM emp GROUP BY deptno HAVING AVG (sal) > 3000 UNION ALL SELECT AVG (sal) AS avg_sal , NULL AS deptno, job FROM emp GROUP BY job HAVING AVG (sal) > 3000) ORDER BY deptno, job;

#	版本	Oracle功能	Oracle语法	迁移后的语法
59	V1R3C00SPC010	使用GROUPING，并指定ROLLUP	SELECT STime, Region, SUM(Profit), GROUPING (STime) AS T, GROUPING (Region) AS R FROM Sales WHERE Profit > 90000 GROUP BY ROLLUP (STime, Region) ORDER BY STime, Region;	SELECT STime, Region, ColumnAlias1, T, R FROM (SELECT SUM (Profit) AS ColumnAlias1 , 0 AS T, 0 AS R, Region, STime FROM Sales WHERE Profit > 90000 GROUP BY STime, Region UNION ALL SELECT SUM (Profit) AS ColumnAlias1 , 0 AS T, 1 AS R, NULL AS Region, STime FROM Sales WHERE Profit > 90000 GROUP BY STime UNION ALL SELECT SUM (Profit) AS ColumnAlias1 , 1 AS T, 1 AS R, NULL AS Region, NULL AS STime FROM Sales WHERE Profit > 90000) ORDER BY STime, Region;

#	版本	Oracle功能	Oracle语法	迁移后的语法
60	V1R3C00SPC010	使用GROUPING，并指定CUBE和ROW NUMBER	SELECT STime, Region, SUM(profit) AS tot_profit, ROW_NUMBER() OVER (ORDER BY SUM(profit) DESC) AS RN, GROUPING (STime) as T, GROUPING (Region) as R FROM Sales GROUP BY CUBE (STime, Region) ORDER BY STime, Region;	SELECT STime, Region, tot_profit , ROW_NUMBER( ) OVER(ORDER BY tot_profit DESC) AS RN , T, R FROM (SELECT SUM (profit) AS tot_profit , 0 AS T, 0 AS R, Region, STime FROM Sales GROUP BY STime ,Region UNION ALL SELECT SUM (profit) AS tot_profit , 0 AS T, 1 AS R, NULL AS Region, STime FROM Sales GROUP BY STime UNION ALL SELECT SUM (profit) AS tot_profit , 1 AS T, 0 AS R, Region, NULL AS STime FROM Sales GROUP BY Region UNION ALL SELECT SUM (profit) AS tot_profit , 1 AS T, 1 AS R, NULL AS Region, NULL AS STime FROM Sales) ORDER BY STime, Region;

#	版本	Oracle功能	Oracle语法	迁移后的语法
61	V1R3C00SPC010	使用GROUPING，并指定ROLLUP和DECODE	SELECT DECODE(GROUPING(dname), 1, 'All Departments', DNAME) AS department, DECODE(GROUPING(job), 1, 'All Jobs', job) AS job, COUNT(*) total_Emp, AVG(sal) * 12 average_sal FROM emp e, dept d WHERE d.deptno = e.deptno GROUP BY ROLLUP (DNAME, job) ORDER BY department, job, total_Emp, average_sal;	SELECT DECODE (func_alias1, 1, 'All Departments', DNAME) AS department , DECODE (func_alias2, 1, 'All Jobs', job) AS job , total_Emp, average_sal FROM (SELECT COUNT(*) total_Emp, AVG (sal) * 12 average_sal , 0 AS func_alias1, 0 AS func_alias2 , DNAME, job FROM emp e, dept d WHERE d.deptno = e.deptno GROUP BY DNAME, job UNION ALL SELECT COUNT(*) total_Emp, AVG (sal) * 12 average_sal , 0 AS func_alias1, 1 AS func_alias2 , DNAME, NULL AS job FROM emp e, dept d WHERE d.deptno = e.deptno GROUP BY DNAME UNION ALL SELECT COUNT(*) total_Emp, AVG (sal) * 12 average_sal , 1 AS func_alias1, 1 AS func_alias2 , NULL AS DNAME, NULL AS job FROM emp e, dept d WHERE d.deptno = e.deptno) ORDER BY department, job, total_Emp, average_sal;

#	版本	Oracle功能	Oracle语法	迁移后的语法
62	V1R3C00SPC010	使用GROUPING，并在DECODE中指定GROUPING SETS、CASE、GROUPING	SELECT empid, designation , CASE WHEN DEPTNO = 10 THEN 'Department 10' WHEN DEPTNO = 20 THEN 'Department 20' WHEN DEPTNO = 30 THEN 'Department 30' ELSE 'No Such department' END AS dept_det , DECODE (GROUPING (DEPTNO), 1, 'All Departments', 'Not All Departments') AS DEPT_grp , DECODE (GROUPING (EMPID), 1, 'All Employees', 'Not All Employees') AS EMP_grp , SUM (SALARY) AS sum_sal , GROUPING (empid) AS grping_empid , GROUPING (designation) AS grping_desig , GROUPING (DEPTNO) AS grping_dept FROM EMPLOYEES1 GROUP BY GROUPING SETS (empid, designation, DEPTNO)	SELECT empid, designation , dept_det , DECODE (func_alias2, 1, 'All Departments', 'Not All Departments') AS DEPT_grp , DECODE (func_alias1, 1, 'All Employees', 'Not All Employees') AS EMP_grp , sum_sal , grping_empid , grping_desig , grping_dept FROM (SELECT CASE WHEN NULL = 10 THEN 'Department 10' WHEN NULL = 20 THEN 'Department 20' WHEN NULL = 30 THEN 'Department 30' ELSE 'No Such department' END AS dept_det , SUM (SALARY) AS sum_sal , 0 AS grping_empid, 1 AS grping_desig, 1 AS grping_dept , 0 AS func_alias1, 1 AS func_alias2 , NULL AS DEPTNO, NULL AS designation, empid FROM EMPLOYEES1 GROUP BY empid UNION ALL SELECT CASE WHEN NULL = 10 THEN 'Department 10' WHEN NULL = 20 THEN 'Department 20' WHEN NULL = 30 THEN 'Department 30' ELSE 'No Such department' END AS dept_det

#	版本	Oracle功能	Oracle语法	迁移后的语法
			ORDER BY empid, designation, DEPTNO, sum_sal;	, SUM (SALARY) AS sum_sal , 1 AS grping_empid, 0 AS grping_desig, 1 AS grping_dept , 1 AS func_alias1, 1 AS func_alias2 , NULL AS DEPTNO, designation, NULL AS empid FROM EMPLOYEES1 GROUP BY designation UNION ALL SELECT CASE WHEN DEPTNO = 10 THEN 'Department 10' WHEN DEPTNO = 20 THEN 'Department 20' WHEN DEPTNO = 30 THEN 'Department 30' ELSE 'No Such department' END AS dept_det , SUM (SALARY) AS sum_sal , 1 AS grping_empid, 1 AS grping_desig, 0 AS grping_dept , 1 AS func_alias1, 0 AS func_alias2 , DEPTNO, NULL AS designation, NULL AS empid FROM EMPLOYEES1 GROUP BY DEPTNO) ORDER BY empid, designation, DEPTNO, sum_sal;

#	版本	Oracle功能	Oracle语法	迁移后的语法
63	V1R3C00SPC010	使用GROUPING，并在CASE中指定GROUPING SETS、DECODE、GROUPING	SELECT empid, designation , CASE WHEN GROUPING (EMPID) = 1 THEN 'All Employees' ELSE 'Not All Employees' END AS EMP_details , DECODE (DEPTNO, 10, 'Department 10' , 20, 'Department 20' , 30, 'Department 30' , 'No Such department') AS DEPT_details , SUM (SALARY) AS sum_sal , GROUPING (empid) AS grping_empid , GROUPING (designation) AS grping_desig , GROUPING (DEPTNO) AS grping_case FROM EMPLOYEES1 GROUP BY GROUPING SETS (empid, designation, DEPTNO) ORDER BY empid, designation, DEPTNO, sum_sal;	SELECT empid, designation , CASE WHEN func_alias1 = 1 THEN 'All Employees' ELSE 'Not All Employees' END AS EMP_details , DEPT_details , sum_sal , grping_empid, grping_desig, grping_case FROM (SELECT DECODE (NULL, 10, 'Department 10' , 20, 'Department 20' , 30, 'Department 30' , 'No Such department') AS DEPT_details , SUM (SALARY) AS sum_sal , 0 AS grping_empid, 1 AS grping_desig, 1 AS grping_case , 0 AS func_alias1 , NULL AS DEPTNO, NULL AS designation, empid FROM EMPLOYEES1 GROUP BY empid UNION ALL SELECT DECODE (NULL, 10, 'Department 10' , 20, 'Department 20' , 30, 'Department 30' , 'No Such department') AS DEPT_details , SUM (SALARY) AS sum_sal , 1 AS grping_empid, 0 AS grping_desig, 1 AS grping_case , 1 AS func_alias1

#	版本	Oracle功能	Oracle语法	迁移后的语法
				, NULL AS DEPTNO, designation, NULL AS empid FROM EMPLOYEES1 GROUP BY designation UNION ALL SELECT DECODE (DEPTNO, 10, 'Department 10' , 20, 'Department 20' , 30, 'Department 30' , 'No Such department') AS DEPT_details , SUM (SALARY) AS sum_sal , 1 AS grping_empid, 1 AS grping_desig, 0 AS grping_case , 1 AS func_alias1 , DEPTNO, NULL AS designation, NULL AS empid FROM EMPLOYEES1 GROUP BY DEPTNO) ORDER BY empid, designation, DEPTNO, sum_sal;

#	版本	Oracle功能	Oracle语法	迁移后的语法
64	V1R3C00SPC010	在HAVING子句中使用GROUPING	SELECT deptno, job, SUM(sal) AS sales_value, GROUPING(deptno) AS f1g, GROUPING(job) AS f2g FROM emp GROUP BY CUBE (deptno, job) HAVING GROUPING(deptno) = 1 OR GROUPING(job) = 1;	SELECT deptno, job, sales_value , f1g, f2g FROM (SELECT SUM (sal) AS sales_value , 0 AS f1g, 0 AS f2g , deptno, job FROM emp GROUP BY deptno, job UNION ALL SELECT SUM (sal) AS sales_value , 0 AS f1g, 1 AS f2g , deptno, NULL AS job FROM emp GROUP BY deptno UNION ALL SELECT SUM (sal) AS sales_value , 1 AS f1g, 0 AS f2g , NULL AS deptno, job FROM emp GROUP BY job UNION ALL SELECT SUM (sal) AS sales_value , 1 AS f1g, 1 AS f2g , NULL AS deptno, NULL AS job FROM emp) WHERE f1g = 1 OR f2g = 1;

2.6.3 括号中的表名

#	版本	Oracle功能	Oracle语法	迁移后的语法
65	V1R3C00SPC010	括号中的表名	SELECT * from (emp) e where e.deptno = 1;	SELECT * FROM emp e WHERE e.deptno = 1;
66	V1R3C00SPC010	括号中的表名和别名	SELECT * from (emp e) where e.deptno = 1;	SELECT * FROM emp e WHERE e.deptno = 1;

2.7 虚拟字段

虚拟字段与表的字段类似，但不存储在表中。用户可在虚拟字段中进行SELECT操作，但无法插入、更新、或删除其中的值。

ROWID：ROWID虚拟字段返回特定行的具体地址。

ROWNUM：对于查询返回的每行数据，ROWNUM虚拟字段会返回一个表示顺序的数字。Oracle依据该数字表示的顺序从表或关联行中选择行。选中的第一行的ROWNUM为1，第二行的ROWNUM为2，依次类推。

2.7.1 ROWID

#	版本	Oracle功能	Oracle语法	迁移后的语法
67	V1R3C00SPC010	使用ROWID，不指定别名	SELECT empid, ename, ROWID FROM employees;	SELECT empid, ename, CAST((xc_node_id || '#' || tableoid || '#' || ctid) AS TEXT) AS ROWID FROM employees;

#	版本	Oracle功能	Oracle语法	迁移后的语法
68	V1R3C00SPC010	使用ROWID，指定别名	SELECT empno, ename, e.ROWID AS Row_ID, d.deptno FROM emp e, dept d WHERE d.deptno = e.deptno;	SELECT empno, ename, CAST((e.xc_node_id || '#' || e.tableoid || '#' || e.ctid) AS TEXT) AS Row_ID , d.deptno FROM emp e, dept d WHERE d.deptno = e.deptno;
69	V1R3C00SPC010	包含内部查询表ROWID的外部查询	SELECT col1, col2, ..., GROUPID FROM (SELECT T.*, ROWIDTOCHAR(ROWID) AS GROUPID FROM (SELECT * FROM tab1 WHERE col1 = '120106198706265017' ....) T);	SELECT col1, col2, ..., GROUPID FROM (SELECT T.* FROM (SELECT CAST((xc_node_id || '#' || tableoid || '#' || ctid) AS TEXT) AS GROUPID, * FROM tab1 WHERE col1 = '120106198706265017' ....) T);

2.7.2 ROWNUM

#	版本	Oracle功能	Oracle语法	迁移后的语法
70	V1R3C00SPC010	在WHERE子句中使用ROWNUM	SELECT e.empid, e.ename FROM employees e WHERE ROWNUM < 6;	SELECT e.empid, e.ename FROM employees e LIMIT 6 - 1;

#	版本	Oracle功能	Oracle语法	迁移后的语法
71	V1R3C00SPC010	在WHERE子句中使用ROWNUM，设置额外条件，并指定ORDER BY子句	SELECT M.hiredate ,M.ename FROM (SELECT ename, hiredate FROM emp ORDER BY hiredate) M WHERE M.hiredate > SYSDATE - 10 AND ROWNUM < 12 ORDER BY M.hiredate ;	SELECT M.hiredate, M.ename FROM (SELECT * FROM (SELECT ename, hiredate FROM emp ORDER BY hiredate) M WHERE M.hiredate > SYSDATE - 10 LIMIT 12 - 1) M ORDER BY M.hiredate;
72	V1R3C00SPC010	使用ROWNUM、>=运算符和GROUP BY子句	SELECT e.deptno FROM emp **e** WHERE ROWNUM >= 10 GROUP BY e.deptno;	SELECT e.deptno FROM (SELECT * FROM emp e LIMIT 0) e GROUP BY e.deptno;
73	V1R3C00SPC010	在SELECT列表中指定ROWNUM	SELECT ROWNUM, e.ename,e.empid FROM employees e WHERE e.deptno = 10 AND ROWNUM < 3 ORDER BY e.ename;	SELECT ROW_NUMBER( ) OVER( ) AS ROWNUM , e.ename, e.empid FROM (SELECT * FROM employees e WHERE e.deptno = 10 LIMIT 3 - 1) e ORDER BY e.ename;

#	版本	Oracle功能	Oracle语法	迁移后的语法
74	V1R3C00SPC010	使用ROWNUM，指定ROLLUP、CUBE、GROUPING SETS	SELECT Time, Region, SUM(Profit) AS Profit , GROUPING (Time) as T, GROUPING (Region) as R FROM Sales WHERE ROWNUM <= 6 GROUP BY ROLLUP (Time, Region) ORDER BY Time, Region;	SELECT TIME, Region, Profit , T, R FROM (SELECT SUM (Profit) AS Profit, 0 AS T, 0 AS R , Region, TIME FROM (SELECT * FROM Sales LIMIT 6) GROUP BY TIME ,Region UNION ALL SELECT SUM (Profit) AS Profit, 0 AS T, 1 AS R , NULL AS Region, TIME FROM (SELECT * FROM Sales LIMIT 6) GROUP BY TIME UNION ALL SELECT SUM (Profit) AS Profit, 1 AS T, 1 AS R , NULL AS Region, NULL AS TIME FROM (SELECT * FROM Sales LIMIT 6)) ORDER BY TIME, Region;

2.8 系统函数

2.8.1 LISTAGG

#	版本	Oracle功能	Oracle语法	迁移后的语法
75	V1R3C00SPC010	使用LISTAGG	SELECT deptno, ename , LISTAGG(ename, ': ') OVER(PARTITION BY deptno,ename ORDER BY ename) as rn FROM emp ORDER BY deptno,ename;	SELECT deptno, ename , STRING_AGG (ename, ': ') OVER (PARTITION BY deptno, enameORDER BY ename) AS rn FROM tab2 ORDER BY deptno, ename;
76	V1R3C00SPC010	使用LISTAGG，不指定分隔符	SELECT LISTAGG(ename) within group (order by deptno) as departmentno , LISTAGG (hiredate) as dates FROM emp;	SELECT STRING_AGG (ename, '' ORDER BY deptno) AS departmentno , STRING_AGG (hiredate, '') AS dates FROM tab1;
77	V1R3C00SPC010	使用LISTAGG，单独指定PARTITION	SELECT deptno, hiredate, ename , LISTAGG(last_name, ': ') WITHIN GROUP (ORDER BY hiredate, ename) OVER (PARTITION BY deptno) as "Emp_list" FROM emp WHERE hiredate < '01-SEP-2003' ORDER BY deptno , hiredate , ename ;	SELECT deptno, hiredate, ename , STRING_AGG (last_name, ': ') OVER (PARTITION BY deptnoORDER BY hiredate, ename) AS "Emp_list" FROM emp WHERE hiredate < '01-SEP-2003' ORDER BY deptno, hiredate, ename;

2.8.2 STRAGG

#	版本	Oracle功能	Oracle语法	迁移后的语法
78	V1R3C00SPC010	使用STRAGG，并指定PARTITION BY	SELECT deptno, ename , STRAGG(ename) OVER (PARTITION BY deptno ORDER BY ename RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as ename_str FROM emp;	SELECT deptno, ename , STRING_AGG (ename, ',') OVER (PARTITION BY deptno ORDER BY ename RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS ename_str FROM emp;
79	V1R3C00SPC010	使用STRAGG，并指定PARTITION BY和ROWS BETWEEN	SELECT deptno, CASE WHEN row_number() over (partition by deptno)=1 THEN STRAGG(ename) OVER (partition by deptno order by ename rows between unbounded preceding and unbounded following) ELSE NULL END AS enames FROM emp;	SELECT deptno, CASE WHEN row_number() over (partition BY deptno ORDER BY ename) = 1 THEN STRING_AGG (ename, ',') OVER (partition BY deptno ORDER BY ename ROWS BETWEEN unbounded preceding AND unbounded following) ELSE NULL END AS enames FROM emp;
80	V1R3C00SPC010	使用STRAGG，并指定DISTINCT	SELECT deptno, STRAGG(DISTINCT ename) ename FROM emp GROUP BY deptno ;	SELECT deptno, STRING_AGG (DISTINCT ename, ',') ename FROM emp GROUP BY deptno;

2.8.3 WM_CONCAT

#	版本	Oracle功能	Oracle语法	迁移后的语法
81	V1R3C00SPC010	使用WM_CONCAT，并指定PARTITION BY	SELECT deptno, WM_CONCAT(ename) over (partition by deptno order by ename) as output , COUNT(ename) over (partition by deptno) as tot_count FROM emp ;	SELECT deptno, STRING_AGG (ename, ',') over (partition BY deptno ORDER BY ename) as output , COUNT(ename) over(partition BY deptno) AS tot_count FROM emp;
82	V1R3C00SPC010	使用WM_CONCAT，并指定DISTINCT	SELECT deptno, WM_CONCAT(distinct ename) FROM emp GROUP BY deptno;	SELECT deptno, STRING_AGG (DISTINCT ename, ',') FROM emp GROUP BY deptno;

2.8.4 SYSTIMESTAMP

#	版本	Oracle功能	Oracle语法	迁移后的语法
83	V1R3C00SPC010	使用**SYSTIMESTAMP**	SELECT SYSTIMESTAMP FROM tab1;	SELECT CURRENT_TIMESTAMP FROM tab1;

2.8.5 NVL2

#	版本	Oracle功能	Oracle语法	迁移后的语法
84	V1R3C00SPC010	NVL2	NVL2(Expr1, Expr2, Expr3)	DECODE(Expr1, NULL, Expr3, Expr2)

2.8.6 日期截断

#	版本	Oracle功能	Oracle语法	迁移后的语法
85	V1R3C00SPC010	使用TRUNC截断到年（YY或YEAR）	TRUNC(<date expression>, <'YY'> / <'YEAR'>)	DATE_TRUNC('YEAR', <date expression>)
86	V1R3C00SPC010	使用TRUNC截断到月（MM、MON或MONTH）	TRUNC(<date expression>, <'MM'> / <'MON'> / <'MONTH'>)	DATE_TRUNC('MONTH', <date expression>)
87	V1R3C00SPC010	使用TRUNC截断到天（DD或DDD）	TRUNC(<date expression>, <'DD'> / <'DDD'>)	DATE_TRUNC('DAY', <date expression>)

2.8.7 QUOTE

#	版本	Oracle功能	Oracle语法	迁移后的语法
88	V1R3C00SPC010	在SQL中使用引号操作符（q）	SELECT q'[It's a string quote operator.]' FROM dual;	SELECT qIt's a string quote operator.q FROM dual;

89	V1R3C00SPC010	在PL/SQL中使用引号操作符（q）	BEGIN
			v_quoteoper := q'[It's a string
			quote operator.]';
			DBMS_OUTPUT.PUT_LINE(v_quoteoper);
			END;
				BEGIN
				v_quoteoper := qIt's a string quote operator.q;
				DBMS_OUTPUT.PUT_LINE (v_quoteoper);
				END;

#	版本	Oracle功能	Oracle语法	迁移后的语法
90	V1R3C00SPC010	引号操作符（q）	SELECT q'[']' FROM dual ; SELECT q'(')' FROM dual ; SELECT q'{'}' FROM dual ; SELECT q'$'$' FROM dual ; SELECT q'^'^' FROM dual ; SELECT q'@'@' FROM dual ; SELECT q'!'!' FROM dual ; SELECT q'*'*' FROM dual ; SELECT q'[']' || '[' || 'HAI' || ']' || q'[']' FROM dual;	SELECT q'q FROM dual; SELECT q'q FROM dual; SELECT q'q FROM dual; SELECT q'q FROM dual; SELECT q'q FROM dual; SELECT q'q FROM dual; SELECT q'q FROM dual; SELECT q'q FROM dual; SELECT q'q || '[' || 'HAI' || ']' || q'q FROM dual;

2.8.8 类型转换

#	版本	Oracle功能	Oracle语法	迁移后的语法
91	V1R3C00SPC010	静态字符串类型转换	SELECT col1 AS column1 , '31203039' AS RESID , 'test123' AS TABLENAME FROM tab1 WHERE col1 = '120222198209204032';	SELECT col1 AS column1 , CAST('31203039' AS text) AS RESID , CAST('test123' AS text) AS TABLENAME FROM tab1 WHERE col1 = '120222198209204032';

2.8.9 DBMS_LOB.SUBSTR

#	版本	Oracle功能	Oracle语法	迁移后的语法
92	V1R3C00SPC010	SQL中的DBMS_LOB.SUBSTR	SELECT expr1, …, DBMS_LOB.SUBSTR(str, len, pos) FROM tab1 WHERE …;	SELECT expr1, …, SUBSTR(str, len, pos) FROM tab1 WHERE …;

93	V1R3C00SPC010	PL/SQL中的DBMS_LOB.SUBSTR	BEGIN
			…
			res := DBMS_LOB.SUBSTR(str, len, pos);
			…
			END;
			/
				BEGIN
				…
				res := SUBSTR(str, len, pos);
				…
				END;
				/

2.8.10 DBMS_LOB.INSTR

#	版本	Oracle功能	Oracle语法	迁移后的语法
94	V1R3C00SPC010	SQL中的DBMS_LOB.INSTR	SELECT expr1, …, DBMS_LOB.INSTR(str, septr, 1, 5) FROM tab1 WHERE …;	SELECT expr1, …, INSTR(str, septr, 1, 5) FROM tab1 WHERE …;

95	V1R3C00SPC010	PL/SQL中的DBMS_LOB.INSTR	BEGIN
			…
			pos := DBMS_**LOB.INSTR**(str,septr,1, i);
			...
			END;
			/
				BEGIN
				…
				pos := **INSTR**(str,septr,1, i);
				...
				END;
				/

2.9 PL/SQL

#	版本	Oracle功能	Oracle语法	迁移后的语法

96	V1R3 C00SPC010	PL/SQL赋值运算符（:=），冒号和等号中间含空额	BEGIN
			...
			v_rowcount **:** = SQL %ROWCOUNT;
			..
			END;
				BEGIN
				...
				v_rowcount **:=** SQL%ROWCOUNT;
				..
				END;

97	V1R3 C00SPC010	PL/SQL赋值运算符用于设置默认值	CREATE OR REPLACE FUNCTION Get_emp_names (Dept_num IN NUMBER **:** = 20) **…**	CREATE OR REPLACE FUNCTION Get_emp_name Get_emp_names (Dept_num IN NUMBER **:=** 20) **…**
98	V1R3 C00SPC010	使用END，指定过程名	CREATE OR REPLACE PROCEDURE sp_ins_emp **…** **…** ... END sp_ins_emp;	CREATE OR REPLACE PROCEDURE sp_ins_emp **…** **…** ... END;
99	V1R3 C00SPC010	使用END，指定函数名	CREATE FUNCTION fn_get_bal **…** **…** ... END get_bal; /	CREATE FUNCTION fn_get_bal **…** **…** ... END; /

#	版本	Oracle功能	Oracle语法	迁移后的语法

100	V1R3C00SPC010	处理EXCEPTION（配置文件中exceptionHandler应设为true。默认值为false。）
		该功能仅在V1R2C60中可用。
			CREATE OR REPLACE PROCEDURE sp_mig_test1 (…)
			IS
			….
			BEGIN
			…
			BEGIN
			…
			EXCEPTION
			WHEN others/<exception code> THEN
			...
			END;
			…
			…
			EXCEPTION
			WHEN others/<exception code> THEN
			...
			END sp_mig_test1;
			/
				CREATE OR REPLACE PROCEDURE sp_mig_test1 (…)
				IS
				….
				BEGIN
				…
				BEGIN
				…
				/*
				EXCEPTION
				WHEN others/<exception> THEN
				…
				*/
				END;
				…
				…
				/*
				EXCEPTION
				WHEN others/<exception code> THEN
				…
				*/
				END;
				/

#	版本	Oracle功能	Oracle语法	迁移后的语法

101	V1R3C00SPC010	子事务处理（配置文件中TxHandler应设为true。该方法用于临时避免运行错误。）	CREATE OR REPLACE PROCEDURE sp_mig_test1 (…)
			IS
			….
			BEGIN
			…
			BEGIN
			…
			EXCEPTION
			WHEN others/<exception code> THEN
			ROLLBACK;
			...
			END;
			...
			IF conditon THEN
			...
			COMMIT;
			ELSE
			...
			ROLLBACK;
			END IF;
			…
			…
			EXCEPTION
			WHEN others/<exception code> THEN
			ROLLBACK;
			...
			END sp_mig_test1;
			/
				CREATE OR REPLACE PROCEDURE sp_mig_test1 (…)
				IS
				….
				BEGIN
				…
				BEGIN
				…
				EXCEPTION
				WHEN others/<exception> THEN
				/* ROLLBACK; */
				NULL;
				…
				END;
				...
				IF conditon THEN
				...
				/* COMMIT; */
				NULL;
				ELSE
				...
				/* ROLLBACK; */
				NULL;
				END IF;
				…
				…
				EXCEPTION
				WHEN others/<exception code> THEN
				/* ROLLBACK; */
				NULL;
				…
				END;
				/

2.10 SQL Formatter

#	版本	Oracle功能	Oracle语法	迁移后的语法
102	V1R3C00SPC010	格式化：使用DELETE，指定多行	DELETE FROM dwsumdata_mc.C00_DATA_DT ALL;	DELETE FROM dwsumdata_mc.C00_DATA_DT ;
103	V1R3C00SPC010	格式化：DROP TABLE	drop table if exists TMP_CUST_CINO_INFO cascade ;	DROP TABLE IF EXISTS TMP_CUST_CINO_INFO CASCADE ;
104	V1R3C00SPC010	格式化：CASE statement	SELECT P1.col1 AS col1_alias ,CASE WHEN P4.col2 IN ('123','456') AND P1.col2 IS NOT NULL THEN 'Consume' WHEN P4.col2 IN ('ABC','XYZ') AND P1.col2 IS NOT NULL THEN 'Business' ELSE '' END AS col2_alias FROM tab1 P4 INNER JOIN tab2 P1 ON P1.col3 = P4.col3 where P4.STATUS NOT IN ('6','8','9') ;	SELECT P1.col1 AS col1_alias ,CASE WHEN P4.col2 IN('123' ,'456') AND P1.col2 IS NOT NULL THEN 'Consume' WHEN P4.col2 IN('ABC' ,'XYZ') AND P1.col2 IS NOT NULL THEN 'Business' ELSE '' END AS col2_alias FROM tab1 P4 INNER JOIN tab2 P1 ON P1.col3 = P4.col3 WHERE P4.STATUS NOT IN('6' ,'8' ,'9') ;

#	版本	Oracle功能	Oracle语法	迁移后的语法
105	V1R3C00SPC010	格式化：Aggregate函数/CASE/JOIN	SELECT P1.col1 AS col1_alias ,SUM(CASE WHEN (P4.col2 > CAST('20170414' AS DATE FORMAT 'YYYYMMDD') OR P4.col3 > CAST('20170414' AS DATE FORMAT 'YYYYMMDD')) THEN ABS(P4.col4 + P4.col5) ELSE ABS(P4.col6 + P4.col7) END) AS Consm_Loan_Bal1 FROM tab1 P4 INNER JOIN tab2 P1 ON P1.col8 = P4.col8 where P4.STATUS NOT IN ('6','8','9') GROUP BY 1 ;	SELECT P1.col1 AS col1_alias ,SUM (CASE WHEN(P4.col2 > CAST('20170414' AS DATE) OR P4.col3 > CAST('20170414' AS DATE)) THEN ABS(P4.col4 + P4.col5) ELSE ABS(P4.col6 + P4.col7) END) AS Consm_Loan_Bal1 FROM tab1 P4 INNER JOIN tab2 P1 ON P1.col8 = P4.col8 WHERE P4.STATUS NOT IN('6' ,'8' ,'9') GROUP BY 1 ;
106	V1R3C00SPC010	格式化：CAST/FORMAT	select '${ODSVIEWDB}' (FORMAT '9(17)') (CHAR(17)) ;	SELECT CAST(LPAD('${ODSVIEWDB}' , 17 ,'0') AS CHAR(17)) ;

#	版本	Oracle功能	Oracle语法	迁移后的语法
107	V1R3C00SPC010	格式化：UNION ALL	select loc, deptno, count(*) as dept_emp from emp where loc = 'INDIA' union all select loc, deptno, count(*) as dept_emp from emp where loc = 'USA';	SELECT loc ,deptno ,COUNT(*) AS dept_emp FROM emp WHERE loc = 'INDIA' UNION ALL SELECT loc ,deptno ,COUNT(*) AS dept_emp FROM emp WHERE loc = 'USA' ;
108	V1R3C00SPC010	格式化：MOD	Select (col1 MOD (9*(10**8)))/(10**6) from tab1;	SELECT (col1 % (9 * (10 ^ 8))) / (10 ^ 6) FROM tab1 ;

#	版本	Oracle功能	Oracle语法	迁移后的语法
109	V1R3C00SPC010	格式化：WITH语句	WITH top_percent_ties AS (SELECT c1 AS x,c2,rank () OVER( ) AS TOP_RNK FROM t1 WHERE c1 > 10) SELECT x,c2 FROM top_percent_ties WHERE TOP_RNK <= (SELECT COUNT(*) * 25 / 100 FROM top_percent_ties) ORDER BY TOP_RNK ;	WITH top_percent_ties AS (SELECT c1 AS x ,c2 ,rank ( ) OVER( ) AS TOP_RNK FROM t1 WHERE c1 > 10) SELECT x ,c2 FROM top_percent_ties WHERE TOP_RNK <= (SELECT COUNT(*) * 25 / 100 FROM top_percent_ties) ORDER BY TOP_RNK ;

#	版本	Oracle功能	Oracle语法	迁移后的语法
110	V1R3C00SPC010	格式化：EXISTS	SELECT * FROM tabl T1 WHERE EXISTS (SELECT Party_id FROM tab2 WHERE Party_id = T1.Party_Id);	SELECT * FROM tabl T1 WHERE EXISTS (SELECT Party_id FROM tab2 WHERE Party_id = T1.Party_Id) ;