

17

维护数据完整性

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

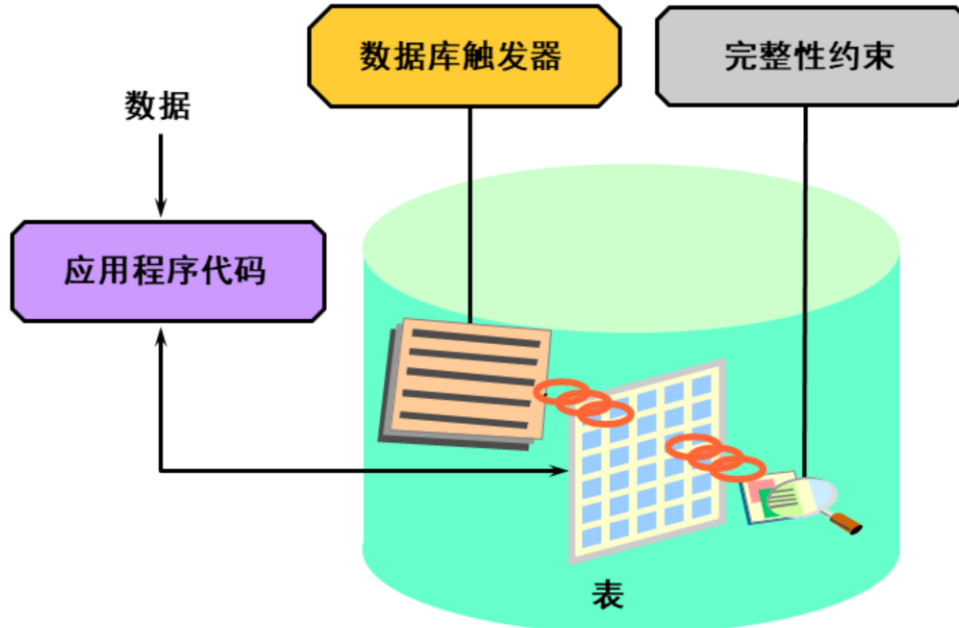
目标

完成这一课的学习后，您应该能达到下列目标：

- 实施数据完整性约束
- 维护完整性约束
- 从数据字典获取约束信息

ORACLE®

数据完整性



ORACLE

13-3

Copyright © Oracle Corporation, 2001. All rights reserved.

数据完整性

数据完整性是指数据库中的数据符合业务规则。维护数据完整性共有三种主要方法：

- 应用程序代码
- 数据库触发器
- 声明完整性约束

具体使用上述哪种方法映射业务规则是设计时应考虑的问题。而数据库管理员主要关心的是实施设计人员选择的方法，并在完整性需求和性能要求之间取得平衡。

应用程序代码既可作为数据库中的存储过程实现，也可作为在客户端上运行的应用程序实现。本课着重讲述完整性约束的使用。

保证数据完整性的方法（续）

数据库触发器：

数据库触发器是 PL/SQL 程序，在表上发生事件（如插入或更新列）时执行。可以启用或禁用触发器，即可以设置触发器在事件发生时执行，或者将触发器设置为不执行（即使已定义）。通常情况下，创建数据库触发器只是为了强制应用不能定义为完整性约束的复杂业务规则。

注：数据库触发器在其它 Oracle 课程中讲述。

完整性约束：

完整性约束是执行业务规则的首选机制，这是因为它可以：

- 改善性能
- 易于声明和修改，不需要进行大量编码
- 集中管理规则
- 使用灵活（启用或禁用）
- 在数据字典中完全文档化

以下部分解释完整性约束的行为，并论述 Oracle 服务器如何执行这些完整性约束。

创建表时定义约束

```
CREATE TABLE hr.employee(  
  id NUMBER(7)  
    CONSTRAINT employee_id_pk PRIMARY KEY  
    DEFERRABLE  
    USING INDEX  
    STORAGE(INITIAL 100K NEXT 100K)  
    TABLESPACE indx,  
  last_name VARCHAR2(25)  
    CONSTRAINT employee_last_name_nn NOT NULL,  
  dept_id NUMBER(7))  
  TABLESPACE users;
```

ORACLE

13-5

Copyright © Oracle Corporation, 2001. All rights reserved.

创建表时定义约束

可以在创建或改变表时定义约束。请使用 CREATE TABLE 或 ALTER TABLE 语句中的 constraint_clause 子句来定义约束。要定义完整性约束，必须具有所需的权限。要创建引用完整性约束，父表必须位于您自己的方案中，或者您必须对父表中的引用键列拥有 REFERENCES 权限。

外键注意事项（续）

如果从父表中删除行时没有使用 `DELETE CASCADE` 选项，Oracle 服务器必须确保子表中的行不包含相应的外键。同样，仅当子行中不包含旧键值时，才允许更新父键。如果子表的外键上没有索引，则 Oracle 服务器锁定子表并禁止更改以确保引用完整性。如果表上有索引，则可通过锁定索引项并避免子表上有更具限制性的锁来维护引用完整性。如果必须从不同的事务处理同时更新两个表，则在外键列上创建索引。

当在子表中插入数据或更新子表中的外键列时，Oracle 服务器检查父表上用来执行引用关键字的索引。因此，仅当包含索引的表空间联机时，该操作才能成功。注意，包含父表的表空间在子表上执行 DML 操作时不需要联机。

Oracle 在主键上执行更新或删除操作时，不再要求在未建索引的外键上获取共享锁定。它仍然获取表级共享锁定，但在获取后立即释放该锁定。如果更新或删除多个主键，则每行获取和释放一次锁定。

创建表时定义约束（续）

`column_constraint` 语法是表定义的一部分。在创建表时，可以使用以下语法定义约束：

```
column datatype [CONSTRAINT constraint]
    { [NOT] NULL
      | UNIQUE [USING INDEX index_clause]
      | PRIMARY KEY [USING INDEX index_clause]
      | REFERENCES [schema.]table [(column)]
      | [ON DELETE CASCADE]
      | CHECK (condition)
    }
    constraint_state ::=
    [ NOT DEFERRABLE | DEFERRABLE [INITIALLY
    { IMMEDIATE | DEFERRED} ] ]
    ]
    [DISABLE | ENABLE [ VALIDATE | NOVALIDATE ] ]
```

其中：

CONSTRAINT： 使用存储在数据字典中的名称 `constraint` 来标识完整性约束

USING INDEX： 指定将 `index-clause` 中定义的参数用于 Oracle 服务器使用的索引，以执行唯一键约束或主键约束（索引的名称与约束的名称相同。）

DEFERRABLE： 表示可使用 `SET CONSTRAINT` 命令将约束检查延迟到事务处理结束时

NOT DEFERRABLE： 表示在每一 **DML** 语句结束时检查该约束（会话或事务处理不能延迟 **NOT DEFERRABLE** 约束。**NOT DEFERRABLE** 是缺省值。）

INITIALLY IMMEDIATE： 表示在每一事务处理开始时，缺省为在每一 **DML** 语句结束时检查该约束（如果没有指定子句 **INITIALLY**，则缺省情况下为 **INITIALLY IMMEDIATE**。）

INITIALLY DEFERRED： 表示该约束为 **DEFERRABLE**，并指定缺省时只在每一事务处理结束时检查该约束

DISABLE： 禁用完整性约束（如果禁用完整性约束，则 Oracle 服务器不执行该约束。）

创建表时定义约束（续）

表约束：

表约束是表定义的一部分。它可以定义除 NOT NULL 约束以外的任何约束。表约束是使用以下语法定义的：

```
[CONSTRAINT constraint]
{PRIMARY KEY (column [, column ]... )
  [USING INDEX index_clause]
|UNIQUE (column [, column ]... )
  [USING INDEX index_clause]
|FOREIGN KEY (column [, column ]... )
  REFERENCES [schema.]table [(column [, column ]... )]
  [ON DELETE CASCADE]
|CHECK (condition)
}
[constraint_state]
```

注

- 采用约束的标准命名约定是一个好习惯。这对 CHECK 约束更是如此，因为可使用不同的名称多次创建同一约束。
- 下列情形需要使用表约束：
 - 当约束命名两列或更多列时
 - 改变表以添加除 NOT NULL 约束外的约束时
- 要在创建表后从类型 NOT NULL 定义约束，只能使用以下语句：

```
ALTER TABLE table MODIFY column CONSTRAINT constraint NOT
NULL;
```

创建表后定义约束：示例

```
SQL> ALTER TABLE hr.employee
2 ADD(CONSTRAINT employee_dept_id_fk FOREIGN KEY(dept_id)
3 REFERENCES hr.department(id)
4 DEFERRABLE INITIALLY DEFERRED);
```

注：本课后面的“启用约束”中讲述了 EXCEPTIONS 子句，该子句可用来找出违反约束（通过 ALTER TABLE 命令添加）的行。

约束的类型

约束	说明
NOT NULL	指示出列不能包含空值
UNIQUE	指示一个列或列的组合是唯一的
PRIMARY KEY	指示一个列或列的组合作为表的主键
FOREIGN KEY	指示一个列或列的组合在引用完整性约束中作为外键
CHECK	指定表中的每一行必须满足的条件

ORACLE

13-9

Copyright © Oracle Corporation, 2001. All rights reserved.

约束的类型

缺省情况下，表中的所有列均可以为空。空意味着没有值。NOT NULL 约束要求表列必须包含值。

UNIQUE 关键字约束要求某列或一组列（关键字）中的值必须是唯一的。表中的任何两行在指定的一列或一组列中不能有重复的值。

数据库中的每个表至多有一个 PRIMARY KEY 约束。PRIMARY KEY 约束确保以下两种情况均为真：

- 表中的任何两行在指定列中没有重复的值。
- 主键列不包含 NULL 值
- 某列或一组列上的 CHECK 完整性约束要求，对于表的每一行，指定的条件必须为真或未知。

虽然 NOT NULL 和 CHECK 约束并不直接要求 DBA 关注，但 PRIMARY KEY、UNIQUE 和 FOREIGN KEY 约束仍须进行管理，以确保高可用性和性能水平可接受。

定义约束

约束类别	语句
非空	Alter table xxx modify xxx not null Alter table xxx add constraint xxx check(xxx is not null)
唯一	Alter table xxx modify xxx unique 或 Alter table xxx add constraint xxx unique(xxx)
主键	Alter table xxx add constraint xxx primary key(xxx)
外键	Alter table xxx add constraint xxx foreign key(xxx) references xxx(xxx)
CHECK	Alter table xxx add constraint xxx check (.....)

ORACLE

外键的删除规则

规则	语法	删除父表记录，如子表有对应记录
限制	默认	拒绝删除
置空	On delete set null	将子表中外键字段置空
级联	On delete cascade	删除子表中对应记录

ORACLE

外键注意事项

目标操作	相应解决方法
删除父表	级联约束
截断父表	禁用或删除外键
删除包含父表的表空间	使用 CASCADE CONSTRAINTS 子句
在子表上执行 DML 操作	确保包含父键的表空间联机

ORACLE®

13-12

Copyright © Oracle Corporation, 2001. All rights reserved.

外键注意事项

维护外键关系中的表时，应该考虑几个因素。

涉及父表的 DDL：

在删除父表之前，必须先删除外键。可以使用以下一条语句同时执行这两个操作：

```
DROP TABLE table CASCADE CONSTRAINTS
```

在未删除或禁用外键之前无法截断父表。

在删除包含父表的表空间之前，必须先删除外键。可使用下列命令完成该操作：

```
DROP TABLESPACE tablespace INCLUDING CONTENTS  
CASCADE CONSTRAINTS
```

删除约束

约束类别	语句
非空	Alter table xxx modify xxx null或 Alter table xxx drop constraint xxx
唯一	Alter table xxx drop constraint xxx
主键	Alter table xxx drop primary key
外键	Alter table xxx drop constraint xxx
CHECK	Alter table xxx drop constraint xxx

ORACLE

修改约束名

Alter table xxx rename constraint xxx to xxx;

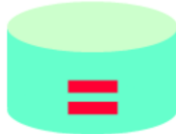
ORACLE

约束的状态

DISABLE
NOVALIDATE



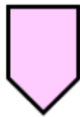
DISABLE
VALIDATE



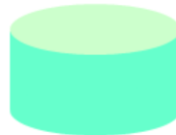
ENABLE
NOVALIDATE



ENABLE
VALIDATE



新数据



现有数据

ORACLE

13-15

Copyright © Oracle Corporation, 2001. All rights reserved.

约束的状态

可以启用 (ENABLE) 或禁用 (DISABLE) 完整性约束。如果启用某个约束，则在数据库中输入或更新数据时，就会对数据进行检查。禁止输入不符合约束规则的数据。如果禁用某个约束，则可以在数据库中输入不符合约束规则的数据。完整性约束可处于以下状态之一：

- DISABLE NOVALIDATE
- DISABLE VALIDATE
- ENABLE NOVALIDATE
- ENABLE VALIDATE

DISABLE NOVALIDATE: 不检查处于 DISABLE NOVALIDATE 状态的约束。表中的数据（包括输入或更新的新数据）可以不符合约束所定义的规则。

DISABLE VALIDATE: 当约束处于此状态时，不允许对受约束的列进行任何修改。另外，约束上的索引将被删除并且禁用约束。**注：**如果约束可延迟，则不删除索引。

约束的状态（续）

ENABLE NOVALIDATE: 如果约束处于此状态，则不能输入违反约束的新数据。但是，表可能包含无效的数据，即数据违反约束。启用处于 NOVALIDATE 状态的约束对正在上传有效 OLTP 数据的数据仓库配置是非常有用的。

ENABLE VALIDATE: 如果约束处于此状态，则不能将违反约束的行插入到表中。但是，禁用该约束时，可以插入此类行。此类行称为该约束的例外。如果约束处于 ENABLE NOVALIDATE 状态，则在禁用约束时输入的数据所引起的违反情况仍然存在。要将约束置于已验证状态，必须更新或删除违反约束的行。

当某一约束由禁用状态更改为 ENABLE VALIDATE 时，将锁定表并对表中的所有数据进行一致性检查。这可能会引起 DML 操作（如等待数据加载），因此，建议先从禁用状态转为 ENABLE NOVALIDATE，然后再转为 ENABLE VALIDATE。

这些状态之间的转换须符合以下规则：

- 除非指定 NOVALIDATE，否则 ENABLE 表示 VALIDATE。
- 除非指定 VALIDATE，否则 DISABLE 表示 NOVALIDATE。
- VALIDATE 和 NOVALIDATE 没有缺省的 ENABLE 和 DISABLE 状态。
- 当唯一键或主键从 DISABLE 状态转为 ENABLE 状态且没有现有索引时，将自动创建唯一索引。（如果索引可延迟，则将存在异常。）与此类似，当唯一键或主键从 ENABLE 转为 DISABLE 且是使用唯一索引启用时，则删除该唯一索引。
- 当任何约束从 NOVALIDATE 状态转为 VALIDATE 状态时，必须检查所有的数据。但是，从 VALIDATE 转为 NOVALIDATE 时，将忽略数据已经过检查这一事实。
- 将单个约束从 ENABLE NOVALIDATE 状态转为 ENABLE VALIDATE 状态时，并不禁止使用读取、写入或其它 DDL 语句。

约束检查



ORACLE

13-17

Copyright © Oracle Corporation, 2001. All rights reserved.

约束检查

可以将约束有效性检查延迟到事务处理完成时进行。

不可延迟的约束或立即约束：

不可延迟的约束（也称立即约束）在每个 DML 语句结束时执行。违反约束将导致语句回退。如果约束导致删除级联等操作，则将该操作作为引起该操作的语句的一部分。不能将定义为不可延迟的约束修改为在事务处理结束时执行。

可延迟的约束：

可延迟的约束是仅在提交事务处理时才检查的约束。如果提交时检测到违反约束的行，则回退整个事务处理。当同时输入外键关系中的父行和子行时（如在订单输入系统中同时输入订单和订单所包含的项目），这些约束非常有用。

可以将定义为可延迟的约束指定为下列模式之一：

- “最初为立即”表示，除非另外明确设定，否则在缺省情况下作为立即约束使用。
- “最初为延迟”表示，缺省情况下只在事务处理结束时执行约束。

约束的执行特性(延迟/不延迟)

约束的固有特性
Deferrable,immediate/deferred

Set constraint改变约束的特性
Set constraints all immediate/deferred

Alter session 改变约束的特性
Alter session set constraints=
Immediate/deferred/default

ORACLE

13-18

Copyright © Oracle Corporation, 2001. All rights reserved.

约束的执行特性

定义约束时可指定**deferrable**，表示支持延迟检查

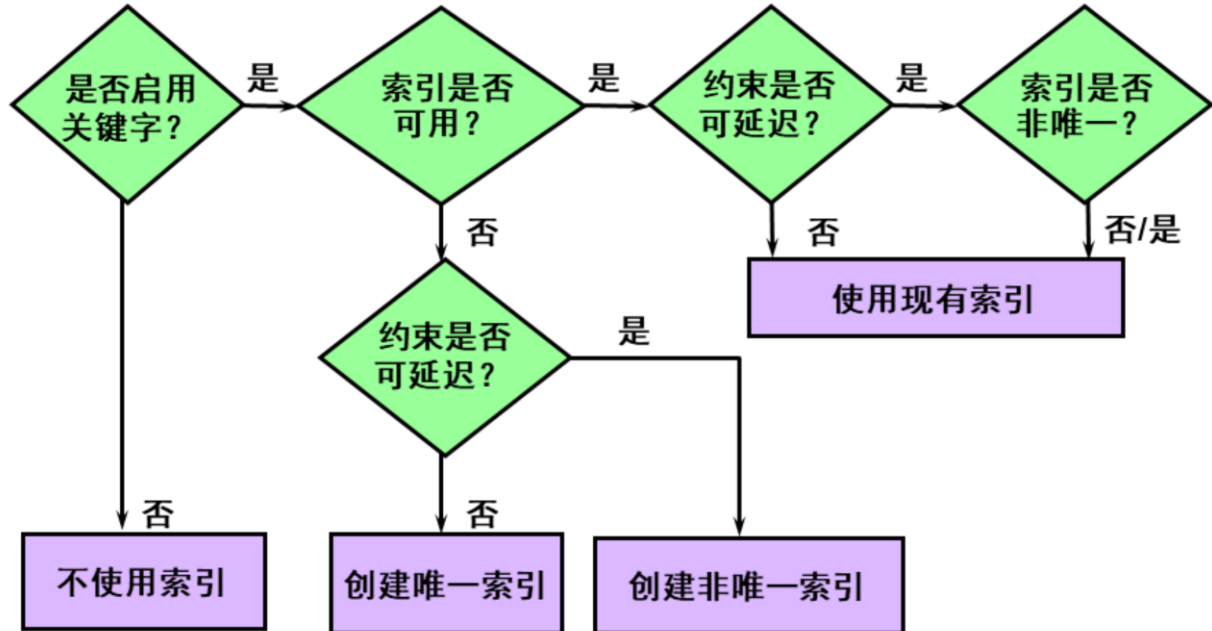
指定**not deferrable**或不指定，表示不支持延迟检查，也就是立即检查。

支持延迟检查的约束并不能立即实现延迟检查，还需要下列条件之一，才可实现延迟检查：

- 1) 当前状态是**deferred**，且会话指定**constraint**为**default**
- 2) 当前会话指定**constraint**是**deferred**
- 3) 当前会话指定**constraint**是**default**，且**set constraint**指定是

deferred

执行主键和唯一键约束



ORACLE

13-19

Copyright © Oracle Corporation, 2001. All rights reserved.

执行主键和唯一键约束

主键和唯一键通过索引执行。可控制用来执行这些约束的索引的位置和类型。

Oracle 服务器按下列步骤实现唯一键和主键约束：

- 如果约束被禁用，则不需要索引。
- 如果启用约束且约束中的列构成索引的主要部分，则无论是否将索引本身创建为唯一还是非唯一索引，都可以使用该索引执行约束。
- 如果启用约束且没有任何索引将约束列用作索引的主要部分，则按照下列规则创建一个名称与约束相同的索引：
 - 如果关键字为可延迟，则在关键字列上创建一个非唯一索引。
 - 如果关键字为不可延迟，则将创建一个唯一索引。
- 如果可以使用某个索引，并且约束是不可延迟的，则使用现有索引。如果约束是可延迟的，并且索引是非唯一的，则使用现有索引。

约束定义原则

- 主约束和唯一性约束：
 - 将索引放在单独的表空间中。
 - 如果经常使用批量加载，请使用非唯一索引。
- 自引用外键：
 - 在初始加载后定义或启用外键。
 - 延迟约束检查。

ORACLE

13-20

Copyright © Oracle Corporation, 2001. All rights reserved.

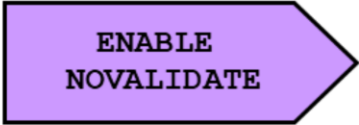
约束定义原则

定义约束时遵循下列原则十分有用：

- 将用于执行主键约束和唯一性约束的索引与表放在不同的表空间中。这可通过指定 `USING INDEX` 子句或通过创建表、创建索引并改变表以添加或启用约束来实现。
- 如果经常向表中批量加载数据，则最好先禁用约束，执行完加载后再启用约束。如果唯一索引用于执行主键约束或唯一性约束，则在禁用约束时必须删除该索引。在这种情况下，可以使用非唯一索引执行主键约束或唯一性约束来改善性能：创建可延迟的键，或者在定义或启用键之前创建索引。
- 如果表中包含自引用外键，请使用下列方法之一加载数据：
 - 在初始加载后定义或启用外键。
 - 将约束定义为可延迟的约束。

在频繁加载数据的情况下，第二种方法非常有用。

启用约束



ENABLE
NOVALIDATE

- 没有表锁定
- 主键和唯一键必须使用非唯一索引

```
ALTER TABLE hr.departments  
ENABLE NOVALIDATE CONSTRAINT dept_pk;
```

ORACLE

13-21

Copyright © Oracle Corporation, 2001. All rights reserved.

启用约束

可以采用下列两种方法之一来启用当前禁用的约束：ENABLE NOVALIDATE 或 ENABLE VALIDATE

启用 NOVALIDATE:

对于当前已有索引的 PRIMARY KEY 和 UNIQUE 约束，启用 NOVALIDATE 约束比启用 VALIDATE 约束要快得多，这是因为，如果约束是可延迟的，则不检查现有数据是否违反约束。如果使用该选项启用约束，则不要求锁定表。这种方法适合表上有许多 DML 活动的情况，如在 OLTP 环境中。

下列命令可用于启用 ENABLE NOVALIDATE 约束：

```
ALTER TABLE [ schema. ] table  
ENABLE NOVALIDATE {CONSTRAINT constraint  
                    | PRIMARY KEY  
                    | UNIQUE ( column [, column ] ... ) }  
[ USING INDEX index_clause ]
```

启用约束（续）

限制：

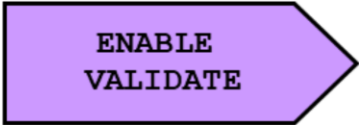
USING INDEX 子句仅适用于创建为可延迟的主键约束或唯一性约束，并且下列条件之一为真的情况：

- 约束被创建为禁用。
- 约束被禁用且索引已删除。

但是，如果需要创建索引，使用这种启用约束的方法并不能比 ENABLE VALIDATE 带来更多的好处，因为 Oracle 服务器在建立索引时锁定表。

注：有关禁用约束的讨论，请参见 *SQL 和 PL/SQL 简介* 课程。

启用约束



ENABLE
VALIDATE

- 锁定表
- 可以使用唯一或非唯一的索引
- 需要有效的表数据

```
ALTER TABLE hr.employees  
ENABLE VALIDATE CONSTRAINT emp_dept_fk;
```

ORACLE

13-23

Copyright © Oracle Corporation, 2001. All rights reserved.

启用约束

启用 VALIDATE 约束后将检查现有数据中是否违反约束。这是启用约束时的缺省操作。若在禁用约束时执行，则会产生下列影响：

- 锁定表，以防在验证完现有数据前对表进行更改。
- 如果索引列上不存在索引，Oracle 服务器就会创建一个索引。当启用不可延迟的主键约束或唯一性约束时，Oracle 服务器将创建一个唯一索引。对于可延迟的主键约束或唯一性约束，将建立一个非唯一索引。

如果在执行约束时执行此命令，则不要求在验证过程中锁定任何表。执行的约束将保证在验证期间不会出现违反约束的情况。这有如下好处：

- 所有约束并发启用。
- 每一约束在内部保持并行。
- 允许表上存在并发操作。

启用约束（续）

使用以下命令启用约束 ENABLE VALIDATE:

```
ALTER TABLE [ schema. ] table
ENABLE [ VALIDATE ] {CONSTRAINT constraint
                    | PRIMARY KEY
                    | UNIQUE ( column [, column ] ... ) }
[ USING INDEX index_clause ]
[ EXCEPTIONS INTO [ schema. ] table ]
```

注:

- VALIDATE 选项为缺省设置，不需要在启用被禁用约束时指定。
- 如果表中的数据违反约束，则语句回退，约束仍被禁用。
- 下面部分将讨论 EXCEPTIONS 子句的使用。

使用 EXCEPTIONS 表

- 通过运行 `utlexcpt1.sql` 脚本来创建 **EXCEPTIONS** 表。
- 执行带有 **EXCEPTIONS** 选项的 **ALTER TABLE** 语句。
- 使用 **EXCEPTIONS** 上的子查询定位包含无效数据的行。
- 纠正错误。
- 重新执行 **ALTER TABLE** 以启用约束。

ORACLE

13-25

Copyright © Oracle Corporation, 2001. All rights reserved.

如何识别行违反

EXCEPTIONS 子句标识出任何违反已启用约束的行。使用下列步骤检测违反约束的行为，纠正它们并重新启用约束：

1. 如果尚未创建 **EXCEPTIONS**，则运行 `utlexcpt1.sql` 脚本：

```
SQL> @@/rdbms/admin/utlexcpt1
```

```
Statement processed.
```

```
SQL> DESCRIBE exceptions
```

Name	Null?	Type
-----	-----	-----
ROW_ID	ROWID	
OWNER		VARCHAR2 (30)
TABLE_NAME		VARCHAR2 (30)
CONSTRAINT		VARCHAR2 (30)

注：`utlexcpt1.sql` 脚本的确切名称和位置视操作系统而定。有关详细信息，请参阅专用于该操作系统的 Oracle 文档。

如何识别行违反（续）

2. 执行使用 EXCEPTIONS 子句的 ALTER TABLE 命令:

```
SQL> ALTER TABLE hr.employee
  2  ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
  3  EXCEPTIONS INTO system.exceptions;
ALTER TABLE hr.employee
*
ORA-02298: cannot enable (HR.EMPLOYEE_DEPT_ID_FK) - parent keys
not found
```

如果未使用所有者姓名限定 EXCEPTIONS 表，则它必须属于被修改的表的所有者。

将行插入到 EXCEPTIONS 表中。如果重新运行该命令，将截断 EXCEPTIONS 表以删除全部现有的行。

3. 使用 EXCEPTIONS 表上的子查询标识无效数据:

```
SQL> SELECT rowid, id, last_name, dept_id
  2  FROM hr.employee
  3  WHERE ROWID in (SELECT row_id
  4  FROM exceptions)
  5  FOR UPDATE;

ROWID                                ID LAST_NAME DEPT_ID
-----
AAAAeyAADA1AAA 1003 Pirie 50
1 row selected.
```

4. 更正数据中的错误:

```
SQL> UPDATE hr.employee
  2  SET dept_id=10
  3  WHERE rowid='AAAAeyAADA1AAA';
1 row processed.
SQL> COMMIT;
Statement processed.
```

如何识别行违反（续）

5. 截断 EXCEPTIONS 表并重新启用约束：

```
SQL> TRUNCATE TABLE exceptions;
```

```
Statement processed.
```

```
SQL> ALTER TABLE hr.employee
```

```
2  ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
```

```
3  EXCEPTIONS INTO system.exceptions;
```

```
Statement processed.
```

获取约束信息

通过查询以下视图获取有关约束的信息：

- **DBA_CONSTRAINTS**
- **DBA_CONS_COLUMNS**

ORACLE

13-28

Copyright © Oracle Corporation, 2001. All rights reserved.

获取约束信息

使用下列查询获得 HR 的 EMPLOYEE 表上所有约束的名称、类型和状态：

```
SQL> SELECT constraint_name, constraint_type, deferrable,  
2 deferred, validated  
3 FROM dba_constraints  
4 WHERE owner='HR'  
5 AND table_name='EMPLOYEES';
```

CONSTRAINT_NAME	C	DEFERRABLE	DEFERRED	VALIDATED
EMPLOYEE_DEPT..	R	DEFERRABLE	DEFERRED	VALIDATED
EMPLOYEE_ID_PK	P	DEFERRABLE	IMMEDIATE	VALIDATED
SYS_C00565	C	NOT DEFERRABLE	IMMEDIATE	VALIDATED

3 rows selected.

获取约束信息（续）

下表列出 DBA_CONSTRAINTS 视图中根据名称不易确定用途的列：

约束中的列：

名称	说明
要获得 HR 的 EMPLOYEE 表内约束中的列，请使用下列查询：	
<pre>SQL> SELECT c.constraint_name, c.constraint_type, 2 cc.column_name 3 FROM dba_constraints c, dba_cons_columns cc WHERE c.constraint_name = 'EMPLOYEE_ID_PK' AND c.table_name = 'EMPLOYEE'</pre>	如果为主键约束，则约束类型为 P；如果为唯一性约束，则为 U；如果为外键约束，则为 R；如果为检查约束，则为 C。NOT NULL 约束存储为检查约束。
<pre>SQL> SELECT c.constraint_name, c.constraint_type, 4 c.condition 5 AND c.table_name = 'EMPLOYEE'</pre>	显示为检查约束指定的条件
<pre>SQL> SELECT c.constraint_name, c.constraint_type, 6 c.owner 7 AND c.constraint_name = cc.constraint_name 8 ORDER BY cc.position,</pre>	为外键定义引用约束的所有者和名称
<pre>CONSTRAINT_NAME C COLUMN_NAME BAD-----</pre>	指示约束名是否由系统生成（有效值为 USERNAME 和 GENERATED NAME。）
<pre>EMPLOYEE_DEPT... R DEPT</pre>	指示将重写约束以避免千年虫等问题
<pre>EMPLOYEE_ID PK P ID LAST_CHANGE- SYS_C00565 C LAST_NAME</pre>	如果设置此标志，则此标志用于优化程序
	显示上次启用或禁用约束的日期

3 rows selected.

获取约束信息（续）

查找主键-外键关系:

要查找 HR 的 EMPLOYEE 表上的外键和父约束，请使用下列查询：

```
SQL> SELECT c.constraint_name AS "Foreign Key",
 2  p.constraint_name AS "Referenced Key",
 3  p.constraint_type,
 4  p.owner,
 5  p.table_name
 6  FROM dba_constraints c, dba_constraints p
 7  WHERE c.owner='HR'
 8  AND c.table_name='EMPLOYEE'
 9  AND c.constraint_type='R'
10  AND c.r_owner=p.owner
11  AND c.r_constraint_name = p.constraint_name;
```

```
Foreign Key Referenced Key C OWNER TABLE_NAME
```

```
-----
```

```
EMPLOYEES_DEPT.. DEPT_PK P HR DEPARTMENT
```

```
1 row selected.
```

小结

在这一课中，您应该能够掌握：

- 实现数据完整性
- 使用相应的策略创建和维护约束
- 从数据字典获取信息

ORACLE®