讲师: collen7788@126.com

其他数据库对象

本章目标

1. 创建简单和复杂视图
2. 从视图中获取数据
3. 创建、维护和使用序列
4. 创建和维护索引
5. 创建私有的和公有的同义词

常见的数据库对象

对象	描述
表	基本的数据存储集合，由行和列组成。
视图	从表中抽出的逻辑上相关的数据集合。
序列	提供有规律的数值。
索引	提高查询的效率
同义词	给对象起别名

视图

- 表EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALA
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	25
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	105
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	110
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	86
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	70
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	44
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	130
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	60
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	120
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	83

20 rows selected.

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600

视图

- 视图是一种虚表.
- 视图建立在已有表的基础上, 视图赖以建立的这些表称为基表。
- 向视图提供数据内容的语句为 SELECT 语句, 可以将视图理解为存储起来的 SELECT 语句.
- 视图向用户提供基表数据的另一种表现形式

视图的优点

限制数据访问

简化复杂查询

提供数据的相互独立

同样的数据，可以有不同的显示方式

但视图不能提高性能

创建视图

- 使用下面的语法格式创建视图

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
 AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

  - FORCE：子查询不一定存在
  - NOFORCE：子查询存在（默认）
  - WITH READ ONLY:只能做查询操作
- 子查询可以是复杂的 **SELECT** 语句

创建视图

- 创建视图举例

```
CREATE VIEW  empvu80
 AS SELECT   employee_id, last_name, salary
    FROM     employees
    WHERE    department_id = 80;
View created.
```

- 描述视图结构

```
DESCRIBE empvu80
```

创建视图

- 创建视图时在子查询中给列定义别名

```
CREATE VIEW  salvu50
 AS SELECT  employee_id ID_NUMBER, last_name NAME,
            salary*12 ANN_SALARY
    FROM    employees
    WHERE   department_id = 50;
View created.
```

- 在选择视图中的列时应使用别名

查询视图

```
SELECT *
FROM  salvu50;
```

ID_NUMBER	NAME	ANN_SALARY
124	Mourgos	69600
141	Rajs	42000
142	Davies	37200
143	Matos	31200
144	Vargas	30000

简单视图和复杂视图

特性	简单视图	复杂视图
表的数量	一个	一个或多个
函数	没有	有
分组	没有	有
DML 操作	可以	有时可以

注意：不建议通过视图对表进行修改

修改视图

- 使用**CREATE OR REPLACE VIEW**子句修改视图

```
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT  employee_id, first_name || ' ' || last_name,
           salary, department_id
   FROM    employees
   WHERE   department_id = 80;
View created.
```

- **CREATE VIEW**子句中各列的别名应和子查询中各列相对应

创建复杂视图

- 复杂视图举例：查询各个部门的最低工资，最高工资，平均工资

```
CREATE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT    d.department_name, MIN(e.salary),
             MAX(e.salary),AVG(e.salary)
   FROM      employees e, departments d
   WHERE     e.department_id = d.department_id
   GROUP BY  d.department_name;
View created.
```

视图中使用DML的规定

- **可以在简单视图中执行 DML 操作**
- **当视图定义中包含以下元素之一时不能使用delete:**
  - 组函数
  - GROUP BY 子句
  - DISTINCT 关键字
  - ROWNUM 伪列

视图中使用DML的规定

- **当视图定义中包含以下元素之一时不能使用update**
  - 组函数
  - GROUP BY子句
  - DISTINCT 关键字
  - ROWNUM 伪列
  - 列的定义为表达式

视图中使用DML的规定

- **当视图定义中包含以下元素之一时不能使用insert**
  - 组函数
  - GROUP BY 子句
  - DISTINCT 关键字
  - ROWNUM 伪列
  - 列的定义为表达式
  - 表中非空的列在视图定义中未包括

屏蔽 DML 操作

- 可以使用 WITH READ ONLY 选项屏蔽对视图的 **DML** 操作
- 任何 **DML** 操作都会返回一个**Oracle server** 错误

屏蔽 DML 操作

```
CREATE OR REPLACE VIEW empvu10
    (employee_number, employee_name, job_title)
AS SELECT   employee_id, last_name, job_id
   FROM     employees
   WHERE    department_id = 10
   WITH READ ONLY;
View created.
```

删除视图

- 删除视图只是删除视图的定义，并不会删除基表的数据

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
View dropped.
```

什么是序列?

- **序列: 可供多个用户用来产生唯一数值的数据库对象**
  - 自动提供唯一的数值
  - 共享对象
  - 主要用于提供主键值
  - 将序列值装入内存可以提高访问效率

CREATE SEQUENCE 语句

- 定义序列:

```
CREATE SEQUENCE sequence
       [INCREMENT BY n]
       [START WITH n]
       [{MAXVALUE n | NOMAXVALUE}]
       [{MINVALUE n | NOMINVALUE}]
       [{CYCLE | NOCYCLE}]
       [{CACHE n | NOCACHE}];
```

创建序列

- 创建序列 DEPT_DEPTID_SEQ为表 DEPARTMENTS 提供主键
- 不使用 CYCLE 选项

```
CREATE SEQUENCE dept_deptid_seq
                INCREMENT BY 10
                START WITH 120
                MAXVALUE 9999
                NOCACHE
                NOCYCLE;
Sequence created.
```

查询序列

- 查询数据字典视图 **USER_SEQUENCES** 获取序列定义信息

```
SELECT   sequence_name, min_value, max_value,
         increment_by, last_number
FROM     user_sequences;
```

- 如果指定**NOCACHE** 选项，则列 **LAST_NUMBER** 显示序列中下一个有效的值

NEXTVAL 和 CURRVAL 伪列

- **NEXTVAL** 返回序列中下一个有效的值，任何用户都可以引用
- **CURRVAL** 中存放序列的当前值
- NEXTVAL 应在 CURRVAL 之前指定，二者应同时有效

序列应用举例

```
INSERT INTO departments(department_id,
            department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
            'Support', 2500);
1 row created.
```

- 序列 **DEPT_DEPTID_SEQ** 的当前值

```
SELECT   dept_deptid_seq.CURRVAL
FROM     dual;
```

使用序列

- 将序列值装入内存可提高访问效率
- 序列在下列情况下出现裂缝：
  - 回滚
  - 系统异常
  - 多个表同时使用同一序列
- 如果不将序列的值装入内存(NOCACHE)，可使用表 USER_SEQUENCES 查看序列当前的有效值

修改序列

- 修改序列的增量，最大值，最小值，循环选项，或是否装入内存

```
ALTER SEQUENCE dept_deptid_seq
               INCREMENT BY 20
               MAXVALUE 999999
               NOCACHE
               NOCYCLE;
Sequence altered.
```

修改序列的注意事项

- 必须是序列的拥有者或对序列有 **ALTER** 权限
- 只有将来的序列值会被改变
- 改变序列的初始值只能通过删除序列之后重建序列的方法实现

删除序列

- 使用DROP SEQUENCE 语句删除序列
- 删除之后，序列不能再次被引用

```
DROP SEQUENCE dept_deptid_seq;
Sequence dropped.
```

索引

- **索引：**
  - 一种独立于表的模式对象，可以存储在与表不同的磁盘或表空间中
  - 索引被删除或损坏，不会对表产生影响，其影响的只是查询的速度
  - 索引一旦建立，Oracle 管理系统会对其进行自动维护，而且由 Oracle 管理系统决定何时使用索引. 用户不用在查询语句中指定使用哪个索引
  - 在删除一个表时，所有基于该表的索引会自动被删除
  - 通过指针加速 Oracle 服务器的查询速度
  - 通过快速定位数据的方法，减少磁盘 I/O

创建索引

- 自动创建：在定义 PRIMARY KEY 或 UNIQUE 约束后系统自动在相应的列上创建唯一性索引
- 手动创建：用户可以在其它列上创建非唯一的索引，以加速查询

创建索引

- 在一个或多个列上创建索引

```
CREATE INDEX index
ON table (column[, column]...);
```

- 在表 **EMPLOYEES**的列 **LAST_NAME** 上创建索引

```
CREATE INDEX  emp_last_name_idx
ON            employees(last_name);
Index created.
```

什么时候创建索引

- **以下情况可以创建索引：**
  - 列中数据值分布范围很广
  - 列经常在 WHERE 子句或连接条件中出现
  - 表经常被访问而且数据量很大，访问的数据大概占数据总量的2%到4%

什么时候不要创建索引

- **下列情况不要创建索引：**
  - 表很小
  - 列不经常作为连接条件或出现在WHERE子句中
  - 查询的数据大于2%到4%
  - 表经常更新

查询索引

- 可以使用数据字典视图 **USER_INDEXES** 和 **USER_IND_COLUMNS** 查看索引的信息

```
SELECT   ic.index_name, ic.column_name,
         ic.column_position col_pos,ix.uniqueness
FROM     user_indexes ix, user_ind_columns ic
WHERE    ic.index_name = ix.index_name
AND      ic.table_name = 'EMPLOYEES';
```

删除索引

- 使用DROP INDEX 命令删除索引

```
DROP INDEX index;
```

- 删除索引UPPER LAST NAME IDX

```
DROP INDEX upper_last_name_idx;
Index dropped.
```

- 只有索引的拥有者或拥有DROP ANY INDEX权限的用户才可以删除索引

同义词

- **使用同义词访问相同的对象：**
  - 方便访问其它用户的对象
  - 缩短对象名字的长度

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

创建和删除同义词

- 为视图DEPT_SUM_VU 创建同义词

```
CREATE SYNONYM  d_sum
FOR  dept_sum_vu;
Synonym Created.
```

- 删除同义词

```
DROP SYNONYM d_sum;
Synonym dropped.
```

Thank you