

ОСНОВНЫЕ ОБЪЕКТЫ БАЗЫ ORACLE

Графеева Н.Г.

2020

Представления(view)

Представление - это именованное правило выборки данных
(SQL запрос)

Base Table

employees						
employee_id	last_name	job_id	manager_id	hire_date	salary	department_id
203	marvis	hr_rep	101	07-Jun-94	6500	40
204	baer	pr_rep	101	07-Jun-94	10000	70
205	higgins	ac_rep	101	07-Jun-94	12000	110
206	gietz	ac_account	205	07-Jun-94	8300	110

View

staff				
employee_id	last_name	job_id	manager_id	department_id
203	marvis	hr_rep	101	40
204	baer	pr_rep	101	70
205	higgins	ac_rep	101	110
206	gietz	ac_account	205	110

Назначение представлений

- Скрытие некоторых данных от пользователей
- Сохранение и дальнейшее использование полезных запросов
- Скрытие реальной структуры данных от пользователей

Создание и удаление представлений (SQL DDL)

Синтаксис:

```
CREATE OR REPLACE VIEW view-name [(field-list)]  
    AS {SELECT-statement| UNION-statement} [ WITH CHECK OPTION ]
```

```
DROP VIEW view-name
```

Примеры

```
CREATE VIEW STAFF
```

```
AS SELECT employee_id, last_name, job_id, manager_id, department_id  
FROM EMPLOYEES
```

```
=====
```

```
CREATE VIEW JOB_IDS
```

```
AS SELECT DISTINCT job_id FROM EMPLOYEES
```

```
=====
```

```
CREATE VIEW STAFF_MANAGER_101
```

```
AS SELECT employee_id, last_name, job_id, manager_id, department_id  
FROM EMPLOYEES WHERE manager_id = 101
```

```
=====
```

Ограничения на создание редактируемых представлений

З а п р е щ е н ы:

- О п ц и я DISTINCT
- Агрегатные функции: AVG, COUNT, MAX, MIN, SUM
- Операции: UNION, UNION ALL, INTERSECT, MINUS
- К о н с т р у к т ы: GROUP BY или HAVING
- П с е в д о с т о л б е ц ROWNUM
- Использование нескольких таблиц после FROM

Пример (использование **check option**)

```
CREATE VIEW Sales_staff  
AS SELECT Empno,  
           Ename,  
           Deptno  
FROM Emp_tab  
WHERE Deptno = 10  
WITH CHECK OPTION
```

Пример (использование **check option**)

```
INSERT INTO Sales_staff  
VALUES (7584, 'OSTER', 10) -- ok
```

```
INSERT INTO Sales_staff  
VALUES(7591, 'WILLIAMS', 30)--not ok
```


Системные представления

- USER_VIEWS
- ALL_VIEWS
- DBA_VIEWS (доступно администратору)

Задание 1

Создайте следующие представления:

- *номера и имена сотрудников, которых принимали на работу зимой;*
- *имена сотрудников, которые являются непосредственными начальниками не менее, чем трех подчиненных .*

Взгляните на содержимое соответствующих системных представлений в своей схеме (в APEX, SQL Commands).

Секвенции

Секвенция - это объект базы данных, который генерирует целые числа в соответствии с правилами, установленными во время его создания. Для последовательности можно указывать как положительные, так и отрицательные целые числа. Последовательности применяют для самых разных целей, но в основном для автоматической генерации первичных ключей. Тем не менее к первичному ключу таблицы последовательность никак не привязана. При определении секвенции указывается следующая информация:

- *имя последовательности*
- *стартовое значение (опционально)*
- *интервал между числами*
- *максимальное значение (опционально)*
- *минимальное значение (опционально)*
- *размер кэша для очередного набора сгенерированных чисел (опционально)*

Создание и удаление секвенций (SQL DDL)

```
CREATE SEQUENCE seq-name  
  [START WITH start-value]  
  INCREMENT BY step-value  
  [MAXVALUE max-value]  
  [MINVALUE min-value]  
  [CYCLE]  
  [CACHE cache-size]
```

```
ALTER SEQUENCE seq-name  
  [START WITH start-value]  
  INCREMENT BY step-value  
  [MAXVALUE max-value]  
  [MINVALUE min-value]  
  [CYCLE]  
  [CACHE cache-size]
```

```
DROP SEQUENCE seq-name
```

Примеры

```
CREATE SEQUENCE sequence_1 INCREMENT BY 10
```

```
=====
```

```
CREATE SEQUENCE sequence_2
```

```
    START WITH 20
```

```
    INCREMENT BY -1
```

```
    MAXVALUE 20
```

```
    MINVALUE 0
```

```
    CYCLE
```

```
    CACHE 2
```

```
=====
```

```
DROP SEQUENCE sequence_1
```

Использование секвенций

- Seq-name.NEXTVAL - генерирует и выдает очередное значение секвенции
- Seq-name.CURRVAL - выдает текущее значение секвенции

Примеры использования

```
SELECT departments_seq.NEXTVAL FROM DUAL
```

```
=====
```

```
INSERT INTO DEPARTMENTS(DEPARTMENT_ID, DEPARTMENT_NAME, LOCATION_ID )  
VALUES  
(  
  DEPARTMENTS_SEQ.NEXTVAL,  
  'DEPARTMENT' || TO_CHAR(DEPARTMENTS_SEQ.CURRVAL),  
  1700  
)
```

Особенности использования

- Необходимо обращать особое внимание на параметр CASH!
- Значения из кэша не обязаны выдаваться подряд.
- При порождении значений с помощью секвенций могут образовываться “дыры” в последовательности значений.

Системные представления для секвенций

- USER_SEQUENCES
- ALL_SEQUENCES
- DBA_SEQUENCES (доступно только для администраторов)

Задание 2

- Создайте подходящую секвенцию для таблицы DEPT1 с параметром CASH = 20.
- Сгенерируйте 10 значений с использованием этой секвенции (заполните таблицу DEPT1).
- Убедитесь, что она заполняется так, как вы ожидали.
- Найдите эту секвенцию в соответствующем системном представлении (в APEX, SQL Commands).

Процедуры, функции, пакеты

- Хранимые процедуры (stored procedure) и функции — это подпрограммы, которые выполняют некоторые действия с информацией в базе данных и при этом сами хранятся в базе данных. В Oracle хранимые процедуры и функции можно писать на языках PL/SQL (процедурное расширение SQL) и Java. Хранимые процедуры и функции никогда не передаются на клиентские компьютеры. Они всегда находятся в базе данных и выполняются СУБД на том компьютере, где располагается сервер базы данных.
- Процедуры и функции могут быть с параметрами и без параметров. Способы передачи параметров:
- **IN** — параметр используется как параметр, передающий начальное значение от фактического параметра формальному при старте процедуры. Этот способ передачи параметра используется по умолчанию, т.е. когда способ передачи в явном виде не задан.
- **OUT** — параметр передает значение в конце работы процедуры/функции от формального параметра фактическому
- **IN OUT** — при старте процедуры/функции передает начальное значение от фактического параметра формальному, а в конце работы процедуры/функции от формального параметра фактическому

Определение функций

CREATE OR REPLACE FUNCTION name[(parameters...)]

RETURN type

IS

 [variables....]

BEGIN

 ...

 RETURN ...

 ...

[EXCEPTION

 WHEN ... THEN

.....]

END [name]

Пример

Определение функции:

```
CREATE OR REPLACE FUNCTION summ (a IN NUMBER, b IN NUMBER)
RETURN NUMBER
IS
    var_result NUMBER;
BEGIN
    var_result := a + b;
    RETURN var_result;
END summ;
```

Вызов функции:

```
SELECT summ(2,3) FROM DUAL
```

Задание 3

- Создайте и вызовите хранимую функцию, вычисляющую факториал натурального числа.
- Создайте и продемонстрируйте вызов функции, определяющий по номеру сотрудника количество дней, которое он проработал (HIREDATE – дата приема в таблице EMP).

Определение процедур

```
CREATE OR REPLACE PROCEDURE name [(parameters...)]  
IS  
[variables...]  
BEGIN  
    ....  
[EXCEPTION  
    WHEN ... THEN  
    .....]  
END [name]
```

Пример

Определение процедуры:

```
CREATE OR REPLACE PROCEDURE avgnumbers
( a          IN      NUMBER,
  b          IN      NUMBER,
  ar         OUT     NUMBER,
  geom       OUT     NUMBER
)
IS
BEGIN
    ar := (a + b) / 2;
    geom := sqrt(a * b);
END avgnumbers;
```


Пример

Вызов процедуры:

```
DECLARE
    A      NUMBER;
    B      NUMBER;
    AR     NUMBER;
    GEOM   NUMBER;
BEGIN
    A := 2;
    B := 3;
    AVGNUMBERS ( A, B, AR, GEOM );

    dbms_output.put_line('AR=' || AR);
    dbms_output.put_line('GEOM =' || GEOM);
END;
```

Задание 4

Создайте хранимую процедуру, выдающую возможную статистику по сотрудникам и департаментам (сколько всего сотрудников, сколько департаментов, сколько различных должностей, какая суммарная зарплата). Для вывода используйте пакет `dbms_output`.

Как сохранять в базе результаты работ процедур и функций?

Создать вспомогательную таблицу, например:

`debug_log(id, LogTime, Message, inSource)`

`id` - идентификатор записи,

`LogTime` – дата и время появления записи,

`Message` – сообщение,

`inSource` – имя процедуры или функции (от которой пришло сообщение)

Такую таблицу **`debug_log`** можно использовать как журнал для фиксации результатов работы процедур и функций.

Пример

Вызов процедуры:

```
DECLARE
  A      NUMBER;
  B      NUMBER;
  AR     NUMBER;
  GEOM   NUMBER;
BEGIN

  A := 2;

  B := 3;

  AVGNUMBERS ( A, B, AR, GEOM );

  INSERT INTO debug_log(id, LogTime, Message, inSource)
  VALUES(debug_log_seq.nextval, sysdate, 'AR =' || AR || ' GEOM=' || GEOM, 'AVGNUMBERS ');

END;
```

Задание 5

- Создайте таблицу **debug_log** и подходящую для нее секвенцию.
- Создайте и вызовите процедуру, определяющую, даты приема на работу сотрудника, который работает дольше всех и сотрудника, который работает меньше всех. Результаты работы процедуры зафиксируйте в `debug_log`.
- Просмотрите содержимое **debug_log** после вызова процедуры.

Системные представления для процедур и функций

- USER_PROCEEDURES
- ALL_PROCEEDURES
- DBA_PROCEEDURES (только для администраторов)

Обработка динамических ошибок

....

EXCEPTION

[WHEN NO_DATA_FOUND THEN....]

[WHEN ZERO_DIVIDE THEN...]

[WHEN TOO_MANY_ROWS THEN ...]

.....

[WHEN OTHERS THEN...]

END

Пример (процедура для фиксации динамических ошибок)

```
create or replace procedure LogInfo
    ( inInfoMessage in varchar2, inSource in varchar2 )
is
    PRAGMA AUTONOMOUS_TRANSACTION;
begin
    insert into debug_log(id, LogTime, Message, inSource)
    values (seq_debug_log.nextval, sysdate, inInfoMessage, inSource);
    commit;
exception
    when others then
        return;
end LogInfo;
```


Пример (вызовы процедуры LogInfo)

```
create or replace procedure Calculate(...)
is
    .....
begin
    .....
    LogInfo('A=' || A, 'Calculate');
    .....
exception
    when others then
        LogInfo(substr(sqlerrm, 1, 100), 'Calculate');

end;
```

Задание 6

- Создайте процедуру для фиксации динамических ошибок.
- Создайте функцию или процедуру, которая может привести к появлению динамической ошибки.
- Спровоцируйте появление и фиксацию 3-х динамических ошибок в журнале **debug_log**.

Зачетное задание 3(10 баллов)

- Оформите все задания в виде одного ***переиспользуемого скрипта***.

Результат предъявите или отправьте по адресу N.Grafeeva@spbu.ru. Тема письма – DB_Application_2019_job3.

Примечание: задание должно быть сдано (отправлено) в течение 2 недель. За более позднее отправленное будут сниматься штрафные баллы (по баллу за каждые 2 недели).