

ЯЗЫК **SQL**



Последовательности

Представления

Индексы

Последовательность



Последовательность – это объект базы данных, предназначенный для генерации последовательных целых чисел.

SEQUENCE

Создание



```
CREATE SEQUENCE ИМЯ  
  [INCREMENT BY n]  
  [START WITH n]  
  [CACHE n | NOCACHE];
```

Пример



```
CREATE SEQUENCE seq1;
```

```
CREATE SEQUENCE seq2
```

```
INCREMENT BY -3
```

```
START WITH 25
```

```
CACHE 5;
```

Последовательность



Иногда бывают

- Циклические (CYCLED)
- С мин и макс значениями (MINVALUE, MAXVALUE)

Данные хранятся:

- USER_SEQUENCES (ORACLE)
- INFORMATION_SCHEMA.SEQUENCES (H2)

Использование



- CURVAL
- NEXTVAL

Можно:

- в списке SELECT команды SELECT;
- в предложении VALUES команды INSERT;
- в предложении SET команды UPDATE.

Использование



CURVAL, NEXTVAL

НЕЛЬЗЯ:

- в подзапросе;
- в запросе к представлению;
- SELECT с ключевым словом DISTINCT;
- SELECT с предложением GROUP BY, HAVING или ORDER BY;
- WHERE команды SELECT;
- в (DEFAULT) значении столбца в предложении CREATE TABLE или ALTER TABLE;

● в условии строгости CHECK

Пример



```
SELECT seq2.CURRVAL  
FROM dual;
```

Ошибка или предыдущее значение (СУБД !?!?)

```
SELECT seq2.NEXTVAL  
FROM dual;
```

25

В следующий раз вернет 22.

Пример



В таблице s_emp все значения столбца id обновить, используя последовательность seq1.

```
UPDATE s_emp  
SET id=seq1.nextval;
```

Каждый раз или один раз ???

```
SELECT seq1.CURRVAL, seq1.NEXTVAL, seq1.NEXTVAL,  
       seq1.NEXTVAL*3, seq1.CURRVAL*3, id  
FROM s_region;
```

<u>N1</u>	<u>N2</u>	<u>N3</u>	<u>N4</u>	<u>N5</u>	<u>ID</u>
34	35	36	111	111	1
37	38	39	120	120	2
40	41	42	129	129	3
43	44	45	138	138	4
46	47	48	147	147	5

Изменение



ALTER SEQUENCE имя
[RESTART WITH long]
[INCREMENT BY n]

Пример: Созданную ранее последовательность **seq2** нужно изменить таким образом, чтобы следующий элемент был **100** а интервал **-4**.

```
ALTER SEQUENCE seq2  
RESTART WITH 100  
INCREMENT BY -4;
```

Проверим:

```
SELECT seq2.NEXTVAL  
FROM dual;
```

100

96

Удаление



DROP SEQUENCE имя;

Пример:

```
DROP SEQUENCE seq2 ;
```

Индексы



- Индекс – объект базы данных, с помощью которого можно ускорить выполнение некоторых запросов.

INDEX

Создание индекса



- индекс в Вашей схеме и индексируется таблица Вашей схемы привилегия CREATE INDEX.
- индекс в Вашей схеме, но индексируется таблица схемы другого пользователя, CREATE INDEX и INDEX по индексируемой таблице.
- индексы в своей схеме, индексируя при этом таблицы любых пользователей CREATE ANY INDEX.
- индекс в другой схеме CREATE ANY INDEX.

Создание индекса



**CREATE INDEX индекс
ON таблица (столбец, ...);**

```
CREATE INDEX s_emp_dept_id_idx  
ON s_emp (dept_id);
```


Когда и зачем?



- Столбец часто используется в предложении WHERE или условии соединения.
- Столбец имеет широкий диапазон значений.
- Столбец содержит большое количество неопределенных значений.
- Таблица большого размера.

Когда не стоит?



- Таблица небольшого размера.
- Столбцы не очень часто используются как условие в запросе.
- Таблица часто обновляется.

Удаление индекса



DROP INDEX имя_индекса;

```
DROP INDEX s_emp_dept_id_idx;
```

Представления



Представление (view – представление, обзор, вид) – логическая таблица, созданная на основе реальных таблиц или других представлений, называемых *базовыми таблицами* представления.

VIEW

Создание



**CREATE [FORCE | NOFORCE] VIEW
представление [(псевдоним, ...)] AS
подзапрос;**

Пример




```
CREATE VIEW dept_45  
AS (SELECT *  
FROM s_emp  
WHERE dept_id=45) ;
```

Пример



```
CREATE VIEW dept_sal  
  (name, minsal, maxsal, avgсал)  
AS (SELECT d.name, MIN(e.salary),  
  MAX(e.salary), AVG(e.salary)  
FROM s_emp e, s_dept d  
WHERE e.dept_id=d.id  
GROUP BY d.name);  
  
SELECT * FROM dept_sal;
```

Создать представление, содержащее номер, фамилию, имя, зарплату и номер начальника для каждого подчиненного сотрудника №1.



```
CREATE VIEW man_1
AS (SELECT id, last_name,
first_name, salary, manager_id
FROM s_emp
WHERE manager_id=1);
```


Изменение данных



Можно *удалять* и *обновлять* строки, если нет:

- условий соединения;
- групповых функций и предложения GROUP BY;
- ключевого слова DISTINCT.


Можно *добавлять* если нет NOT NULL*

Всем сотрудникам, имеющим зарплату **1400**
через представление **man_1** увеличить ее до
1800.



```
UPDATE man_1  
SET salary=1800  
WHERE salary=1400;
```

Служащих с зарплатой **1450** посредством представления **man_1** перевести к начальнику №**2**.



```
UPDATE man_1
SET manager_id=2
WHERE salary=1450;
```

Удаление представления



`DROP VIEW имя;`

Удалить представление `man_1`:

```
DROP VIEW man_1;
```