ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ.

Лекция 5

доц. Щербинина О.В.

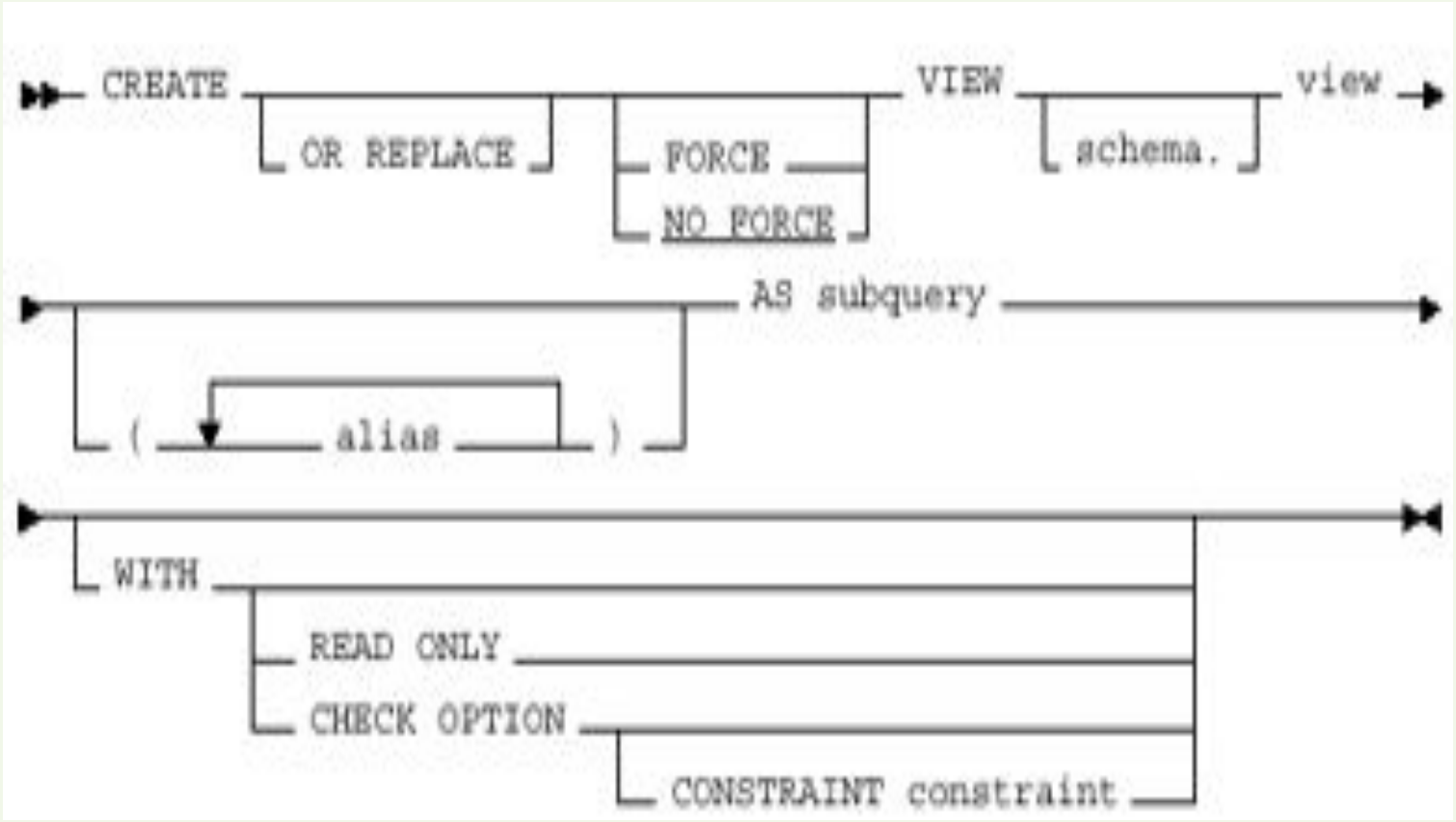
Представления

- Представления добавляют уровень защиты данных (например, можно создать представление для таблицы, где пользователю, выполняющему SELECT над представлением, видны только сведения о зарплате)

- Представления могут скрывать сложность данных, комбинируя нужную информацию из нескольких таблиц

- Представления могут скрывать настоящие имена столбцов, порой трудные для понимания, и показывать более простые имена.

Команда CREATE VIEW



Команда CREATE VIEW

OR REPLACE - пересоздает представление, если оно уже существует
FORCE - создает представление независимо от того, существуют ли базовые таблицы этого представления, и от того, имеет ли владелец схемы, содержащей представление, привилегии по этим таблицам. По умолчанию применяется параметр NOFORCE;
NOFORCE - создает представление только в том случае, если существуют базовые таблицы этого представления, а владелец схемы, содержащей представление, имеет привилегии по этим таблицам;
Schema - схема, в которой создается представление
View - имя создаваемого представления;
Alias - специфицирует имена для выражений, выбираемых запросом представления;
AS subquery - идентифицирует столбцы и строки таблиц, на которых базируется представление;

Команда CREATE VIEW

WITH CHECK OPTION – при указании данного параметра пользователь не может вводить, удалять и обновлять информацию таблицы, из которой он не имеет возможности считать информацию через простое представление (создаваемое из данных одной таблицы). Обновляемое представление, использующее несколько связанных таблиц, нельзя создавать с данным параметром;

Constraint - имя, которое присваивается ограничению CHECK OPTION. Если этот идентификатор опущен, то ORACLE автоматически назначает этому ограничению имя следующего вида:

SYS_Cn , где n - целое, которое делает имя ограничения уникальным внутри базы данных.

Обновляемое представление:

- должно включать первичный ключ таблицы
- не должно содержать полей, полученных в результате применения функций агрегирования
- не должно содержать DISTINCT, GROUP BY, HAVING в своем определении
- может быть определено на другом представлении, но это представление должно быть обновляемым
- не может содержать константы, строки или выражения (например, comm*100) в списке выбираемых выходных полей

Команда CREATE VIEW

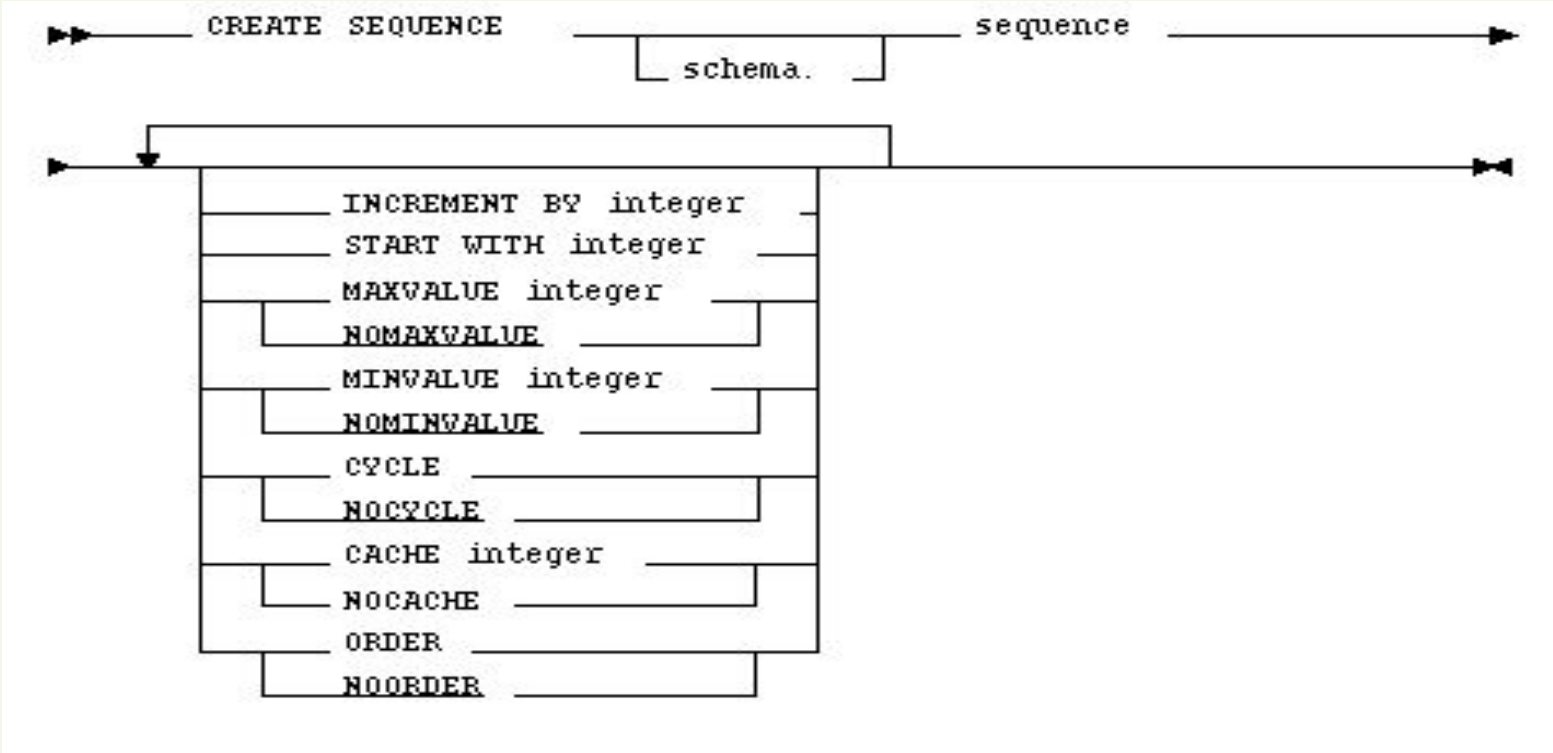
```
CREATE VIEW vstudent  AS
            SELECT * FROM student WHERE ball > 80 ;

CREATE VIEW vstudent(kod_s,kol)  AS
        SELECT kod_s,COUNT(*) FROM student
                        GROUP BY kod_s;

CREATE VIEW vstudent  AS
            SELECT nazv_s, fam, ball  FROM spec, student
                    WHERE spec.kod_s =student.kod_s;
CREATE VIEW vstudent AS
        SELECT nazv_s, fam   FROM spec, student
            WHERE spec.kod_s =student.kod_s  AND
            student.ball =(SELECT AVG(ball) FROM student);
```

Последовательности

Последовательность (sequence) – объект базы данных, который генерирует целые числа в соответствии с правилами, установленными во время его создания. Создается командой **CREATE SEQUENCE.**



Ключевые слова и параметры

- **schema** — схема, в которой создается последовательность.
- **sequence** — имя создаваемой последовательности
- **start with** — позволяет создателю последовательности указать первое генерируемое ею значение
- **increment by n** — определяет приращение последовательности при каждой ссылке на виртуальный столбец NEXVAL
- **minvalue** — определяет минимальное значение, создаваемое последовательностью.
- **nominvalue** — указывает, что минимальное значение равно 1, если последовательность возрастает, или -10^{26}, если последовательность убывает
- **maxvalue** — определяет максимальное значение, создаваемое последовательностью
- **nomaxvalue** — указывает, что максимальное значение равно 10^{27}, если последовательность возрастает, или -1, если последовательность убывает

Ключевые слова и параметры

- **cycle** — позволяет последовательности повторно использовать созданные значения при достижении MAXVALUE или MINVALUE. Т.е. последовательность будет продолжать генерировать значения после достижения своего максимума или минимума
- **nocycle** — указывает, что последовательность не может продолжать генерировать значения после достижения своего максимума или минимума
- **cache n** — указывает, сколько значений последовательности ORACLE распределяет заранее и поддерживает в памяти для быстрого доступа
- **order** — гарантирует, что номера последовательности генерируются в порядке запросов
- **noorder** — не гарантирует, что номера последовательности генерируются в порядке запросов

Создание последовательности

```
CREATE SEQUENCE sequence_1 INCREMENT BY 10;

CREATE SEQUENCE sequence_2
START WITH 20
INCREMENT BY –1
MAXVALUE 20
MINVALUE 0
CYCLE
ORDER
CACHE 2;
```

Обращение к последовательности

CURRVAL -возвращает текущее значение последовательности
NEXTVAL -выполняет приращение последовательности и возвращает ее следующее значение

- Текущее и следующее значения последовательности пользователи базы данных получают, выполняя команду SELECT
- Первое обращение к NEXTVAL возвращает начальное значение последовательности
- Последующие обращения к NEXTVAL изменяют значение последовательности на приращение, которое было определено, и возвращают новое значение
- Любое обращение к CURRVAL всегда возвращает текущее значение последовательности

Обращение к последовательности

<имя последовательности>.**CURRVAL**
<имя последовательности>. **NEXTVAL**

Обращение к текущему или следующему значению последовательности, принадлежащей схеме другого пользователя:

- пользователь должен иметь объектную привилегию SELECT по этой последовательности
- либо системную привилегию SELECT ANY SEQUENCE
- пользователь должен дополнительно квалифицировать эту последовательность именем содержащей ее схемы

<имя схемы>.<имя последовательности>.**CURRVAL**
<имя схемы>. <имя последовательности>. **NEXTVAL**

Использование последовательности

Значения CURRVAL и NEXTVAL используются:

- в списке SELECT предложения SELECT
- в фразе VALUES предложения INSERT
- в фразе SET предложения UPDATE

Нельзя использовать значения CURRVAL и NEXTVAL:

- в подзапросе
- в предложении SELECT с оператором DISTINCT
- в предложении SELECT с фразой GROUP BY или ORDER BY
- в предложении SELECT, объединенном с другим предложением SELECT оператором множеств UNION
- в фразе WHERE предложения SELECT
- в умалчиваемом (DEFAULT) значении столбца в предложении CREATE TABLE или ALTER TABLE
- в условии ограничения CHECK

Пример действия циклической последовательности

```
SQL> SELECT sequence_2.NEXTVAL FROM dual;
NEXTVAL
-------------
         20
SQL> /
NEXTVAL
-------------
         19
…..

SQL> /
NEXTVAL
-------------
          1
SQL> /
NEXTVAL
-------------
          0
SQL> /
NEXTVAL
-------------
         20
```

Пример действия циклической последовательности

```
SQL> SELECT sequence_2.CURRVAL FROM dual;
CURRVAL
--------------
          20

SQL> SELECT sequence_2. NEXTVAL FROM dual;
NEXTVAL
--------------
          19

SQL> SELECT sequence_2.CURRVAL FROM dual;
CURRVAL
--------------
          19
```

Использование последовательности

Ссылка на последовательности при изменении данных:

```
INSERT INTO student
VALUES (sequence_3.nextval,1,'Андреева','Ирина','Александровна', 200, 90);

UPDATE student SET kod_stud = sequence_3.currval
                WHERE fam ='Андреева';
```

Изменение описание последовательности:

```
ALTER SEQUENCE  sequence_2 INCREMENT BY –4;
```

Удаление последовательности:

```
DROP SEQUENCE sequence_2
```

Управление транзакциями

Транзакции создаются с помощью:

- набора команд, определяющих начало, контрольные точки и окончание транзакции
- специального механизма блокирования, предотвращающего изменение информации строк базы данных несколькими пользователями одновременно

- **SET TRANSACTION** – начинает транзакцию и устанавливает ее базовые характеристики.
- **COMMIT** – заканчивает текущую транзакцию сохранением изменений в базе данных и начинает новую транзакцию
- **ROLLBACK** – заканчивает текущую транзакцию отменой изменений в базе данных и начинает новую транзакцию
- **SAVEPOINT** – устанавливает контрольные точки (точки прерывания) для транзакции, разрешая неполный откат.

Начало транзакции

- после регистрации пользователя в Oracle с помощью SQL*Plus и исполнения им первой команды
- после выдачи команды ROLLBACK или COMMIT, заканчивающей транзакцию
- после выхода пользователя из системы
- в результате аварии системы
- после выдачи команды описания данными, например ALTER DATABASE

Установка контрольных точек

В программном блоке сохраняются только те изменения, которые были внесены до описания точки сохранения:

```
UPDATE my_table.products SET price=50 WHERE product=10010;

SAVEPOINT точка_сохранения;

UPDATE my_table.products SET price=200;

ROLLBACK TO SAVEPOINT точка_сохранения;

COMMIT;
```

Команда DESCRIBE

Возвращает описание таблицы, включая описание всех ее столбцов, тип данных для каждого столбца и указание на возможность хранения в столбце NULL-значений.

Синтаксис команды: **DESCRIBE tablename**

```
SQL> describe student;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 KOD_STUD                                  NOT NULL NUMBER(38)
 KOD_S                                              NUMBER(38)
 FAM                                                CHAR(30)
 IM                                                 CHAR(15)
 OT                                                 CHAR(15)
 STIP                                               NUMBER(3)
 BALL                                               NUMBER(3)

SQL>
```

Использование таблицы DUAL

DUAL - общедоступная таблица словаря данных

Содержит:

- один столбец DUMMY
- одну строку со значением X

Используется:

- для возврата результатов SQL-функций
- для извлечения данных виртуальных столбцов

Примеры:

SELECT SQRT(120) **FROM** dual;

SELECT SYSDATE **FROM** dual;

Сценарии

Сценарий – последовательность исполняемых SQL операторов

Сохранение сценария:

- открыть текстовый редактор, ввести часто исполняемые операторы и сохранить сценарий как обычный текстовый файл
- использование команды **SAVE**, которая помещает содержимое буфера SQl*Plus в текстовый файл с расширением.sql

Загрузка и исполнение сценария:

- команда GET
- команда / или @

Пример:

SAVE my_s.sql

@my_s.sql

Команды, улучшающие внешний вид выходных данных

- COLUMN{col}FORMAT{fmt}HEADING{string}
- BREAK
- TTITLE
- BTITLE

Пример 1:

```
COLUMN price FORMAT $999,999.99
SELECT price FROM my_table;
price
--------------
$60,000.00
$22,000.00
$110,085.00
```

Команды, улучшающие внешний вид выходных данных

Пример 2:

```
BREAK ON kod_s
SELECT kod_s,fam FROM student ORDER BY kod_s;
kod_s        fam
-------------------------
1            Кошкин
             Мышкин
             Воробьев
2            Ромашкина
             Кузнецова
```

Команды, улучшающие внешний вид выходных данных

Пример 3:

TTITLE CENTER ‘Ведомость’
BTITLE CENTER ‘Конец страницы’

```
                Ведомость
kod_s           fam                 stip
-------------------------------------------------------
1               Кошкин              500
                Мышкин              500
                Воробьев            800
2               Ромашкина               500
                Кузнецова           500
                                ----------
sum                                 2800
                Конец страницы
```