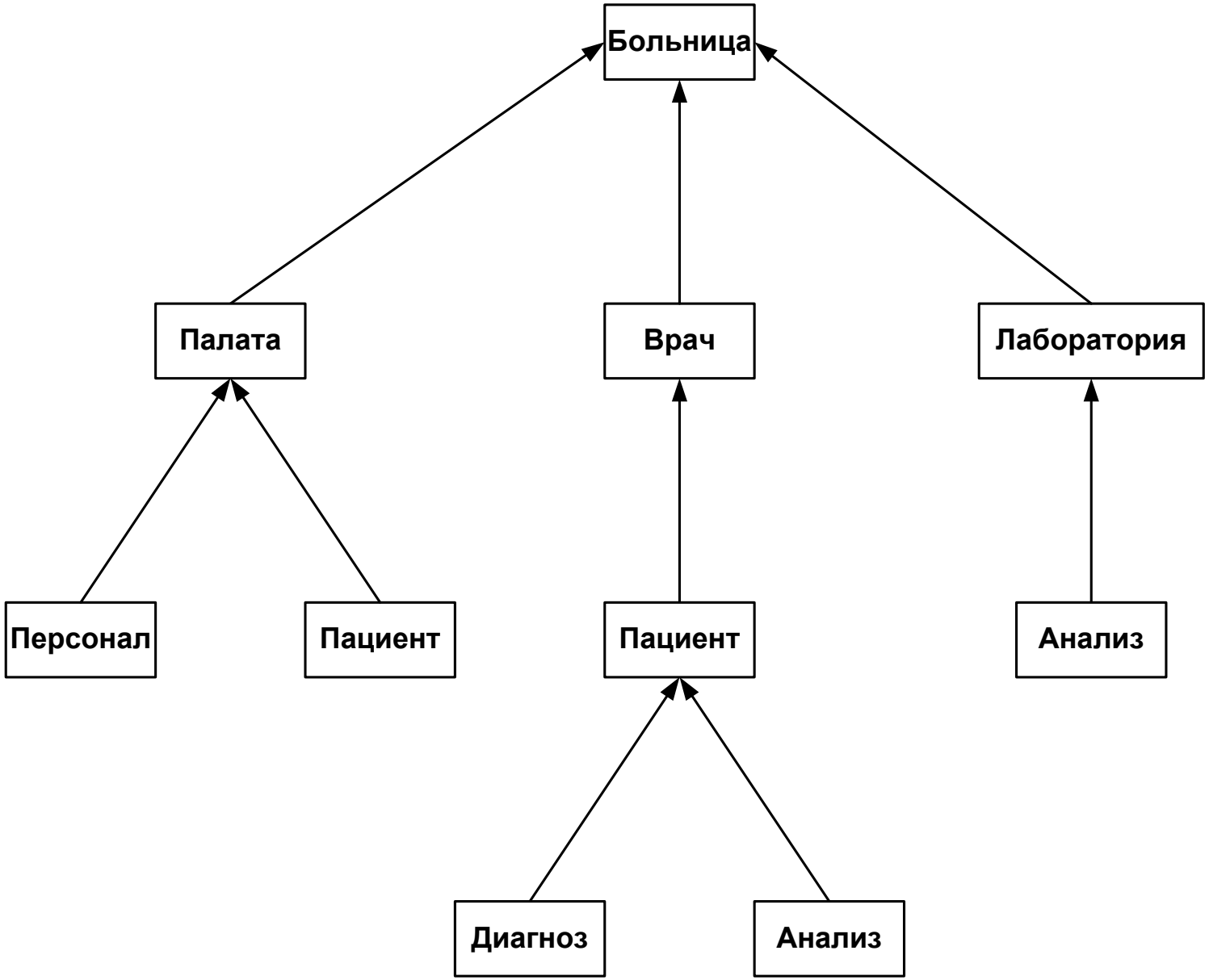
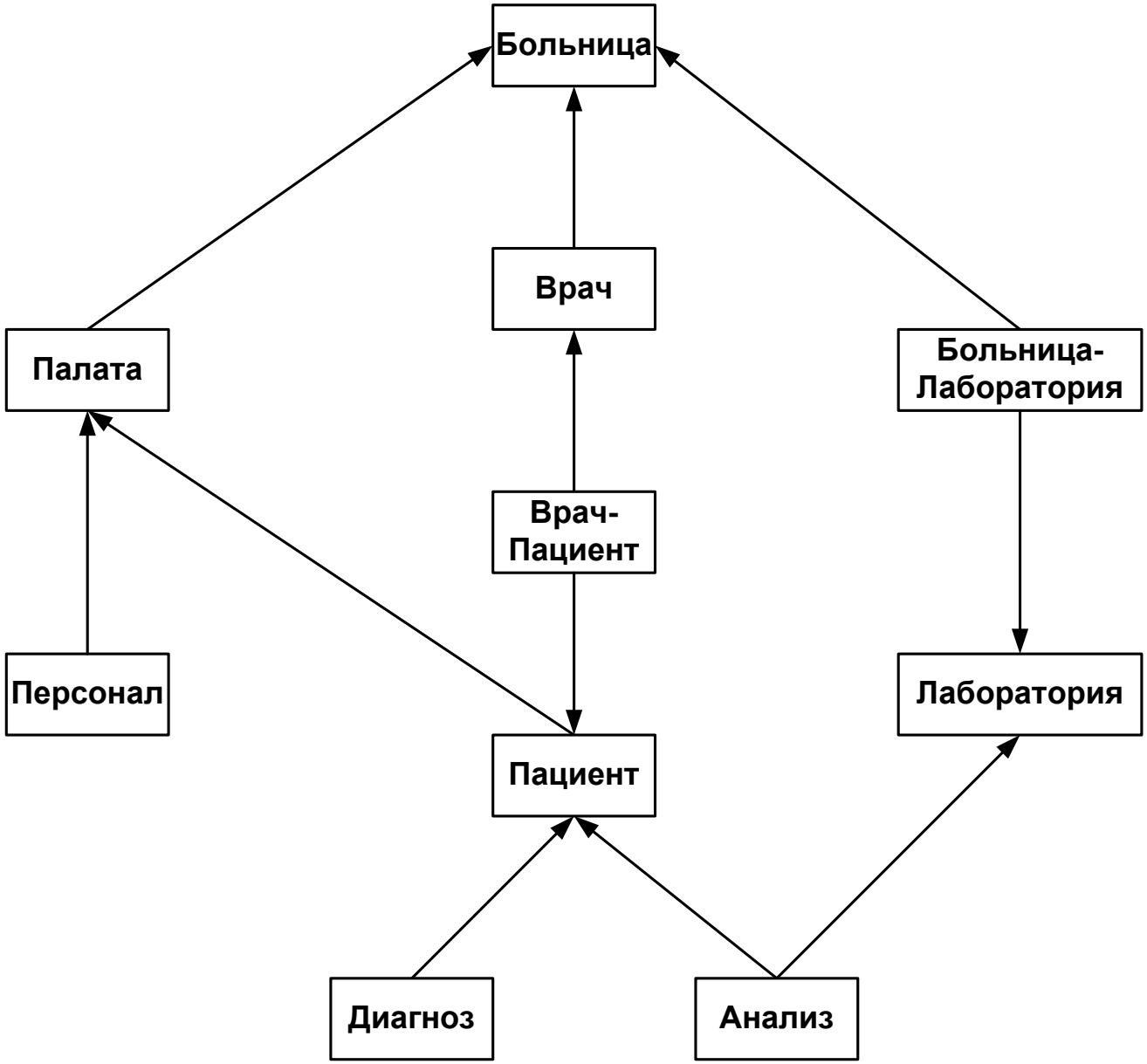
IV. СУБД-ОРИЕНТИРОВАННЫЕ МОДЕЛИ ДАННЫХ

4.1. Обзор СУБД-ориентированных моделей данных

Иерархическая схема медицинской ПрО



Сетевая схема медицинской ПрО



4.2. Реляционная модель данных

4.2.1. Структуры

Основными структурными понятиями реляционной модели данных являются:

- тип данных,
- домен,
- атрибут,
- кортеж,
- отношение.

Понятие **тип данных** в реляционной модели данных полностью аналогично понятию типа данных в языках программирования.

Домен – это множество допустимых значений.

Схема отношения – это именованное множество пар «*имя* ***атрибута***, *имя домена*» (или типа, если понятие домена не поддерживается). Степень или «арность» схемы отношения – мощность этого множества.

Схема БД (в структурном смысле) – это набор именованных схем отношений.

Кортеж, соответствующий данной схеме отношения, – это множество пар «*имя атрибута, значение*», которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. **Значение** является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается).

Термин **отношение** в общепринятом математическом смысле определяется следующим образом. Для заданных множеств S_1, S_2, ..., S_n (не обязательно различных) R является отношением на этих n множествах, если представляет собой множество кортежей степени n, у каждого из которых первый элемент взят из множества S_1, второй – из множества S_2 и т.д. Мы будем называть S_j j-тым **доменом** R. Говорят, что такое отношение R имеет **степень** n. Отношения степени 1 часто называют **унарными**, степени 2 – **бинарными**, степени 3 – **тернарными**, степени 4 – **кватернарными** и степени n – ***n*-арными**.

n-арное «математическое» отношение R обладает следующими свойствами:

- Порядок кортежей не является существенным.
- Все кортежи различны.
- Порядок значений атрибутов в кортежах является существенным – он соответствует порядку доменов, на которых определяется R.

Термин **отношение** в моделировании данных определяется следующим образом.

Пусть задано множество из *n* типов или доменов $T_i (i = 1,..., n)$, причем все они необязательно должны быть различными. Тогда *r* будет отношением, определенным на этих типах, если оно состоит из двух частей: заголовка и тела (заголовок еще иногда называют схемой или интенсионалом отношения, а тело – экстенсионалом отношения), где:

- заголовок – это множество из *n* атрибутов вида $A_i : T_i$;

здесь A_i – имена атрибутов отношения *r*, а T_i – соответствующие имена типов;

- тело – это множество из *m* кортежей *t* ; здесь *t* является множеством компонентов вида $A_i : v_i$, в которых v_i – значение типа T_i , т.е. значение атрибута A_i в кортеже *t*.

Отношения реляционной модели обладают следующими свойствами:

- отсутствие кортежей-дубликатов,
- отсутствие упорядоченности кортежей,
- отсутствие упорядоченности атрибутов,
- атомарность значений атрибутов.

Реляционная схема медицинской ПрО (структуры)

БОЛЬНИЦА (Код больницы (К/Б), Название, Адрес, Число коек (Ч/К))

ПАЛАТА (Код больницы (К/Б), Номер палаты (Н/П), Название, Число коек (Ч/К))

ПЕРСОНАЛ (Код больницы (К/Б), Номер палаты (Н/П), Фамилия, Должность, Смена, Зарплата (З/П))

ВРАЧ (Код врача (К/В), Код больницы (К/Б), Фамилия, Специальность)

ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Адрес, Дата рождения (Д/Р), Пол, Номер медицинского полиса (НМП))

ДИАГНОЗ (Регистрационный номер (Р/Н), Тип диагноза (Т/Д), Осложнения, Предупреждающая информация)

ЛАБОРАТОРИЯ (Код лаборатории (К/Л), Название, Адрес)

АНАЛИЗ (Регистрационный номер (Р/Н), Код Лаборатории (К/Л), Тип анализа (Т/А), Назначенная дата (Н/Д), Назначенное время (Н/В), Номер направления (Н/Н), Состояние)

БОЛЬНИЦА-ЛАБОРАТОРИЯ (Код больницы (К/Б), Код Лаборатории (К/Л))

ВРАЧ-ПАЦИЕНТ (Код врача (К/В), Регистрационный номер (Р/Н))

РАЗМЕЩЕНИЕ (Код больницы (К/Б), Номер палаты (Н/П), Регистрационный номер (Р/Н), Номер койки (Н/К))

ТЕЛЕФОН БОЛЬНИЦЫ (Код больницы (К/Б), Телефон)

ТЕЛЕФОН ЛАБОРАТОРИИ (Код лаборатории (К/Л), Телефон)

Простейшие правила перехода от ER-схемы к реляционной схеме БД

1. Каждое множество сущностей представляется самостоятельным отношением, однозначные атрибуты множества сущностей становятся атрибутами отношения, ключи множества сущностей являются возможными ключами отношения; при необходимости в качестве первичного ключа отношения используется суррогатный атрибут.
2. Бинарные множества связей типа 1:M без атрибутов представляются дублированием первичного ключа 1-отношения в M-отношение.
3. Бинарные множества связей типа M:N без атрибутов представляются самостоятельными отношениями, куда дублируются первичные ключи отношений, построенных для множеств сущностей.
4. Множества связей с атрибутами представляются самостоятельными отношениями, куда дублируются первичные ключи отношений, построенных для множеств сущностей. Однозначные атрибуты множества связей становятся атрибутами этого отношения.

Простейшие правила перехода от ER-схемы к реляционной схеме БД (продолжение)

5. Множества связей степени больше 2-х представляются самостоятельными отношениями, куда дублируются первичные ключи отношений, построенных для множеств сущностей. Однозначные атрибуты множества связей становятся атрибутами этого отношения.
6. Каждый многозначный атрибут множества сущностей представляется отдельным отношением, куда дублируется первичный ключ отношения, построенного для множества сущностей; второй атрибут этого отношения предназначен собственно для значения.
7. Каждый многозначный атрибут множества связей представляется отдельным отношением, куда дублируется первичный ключ отношения, построенного для множества связей; второй атрибут этого отношения предназначен собственно для значения.

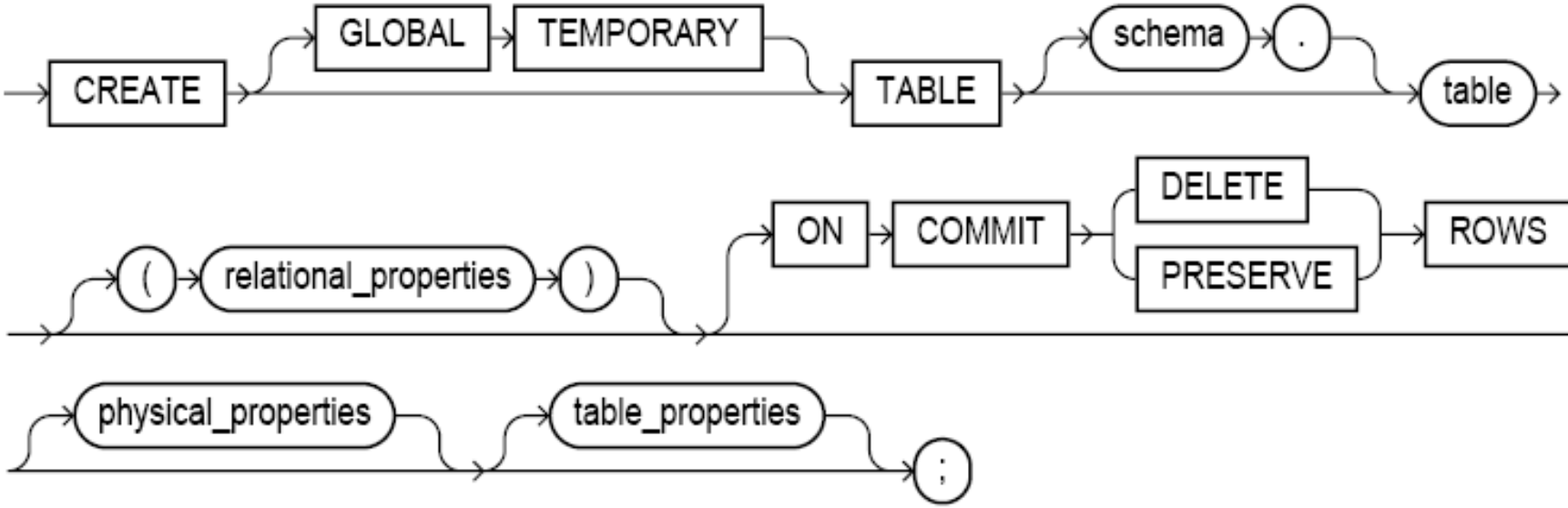
4.2.2. Ограничения целостности

Конструкции языка SQL, связанные с ограничениями целостности:

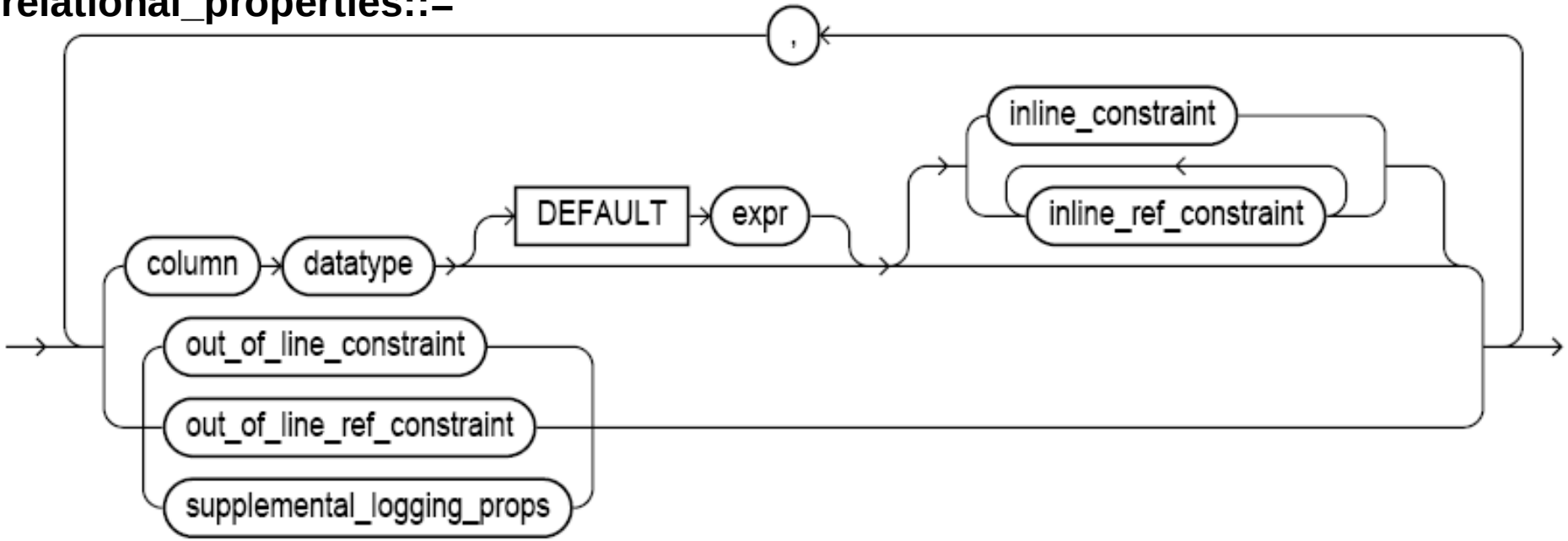
- тип данных атрибута;
- описатели атрибута NULL и NOT NULL, позволяющие определить соответственно, может или нет атрибут иметь неопределенное значение в кортежах отношения;
- описатель атрибута или группы атрибутов PRIMARY KEY, определяющий первичный ключ отношения;
- описатель атрибута или группы атрибутов UNIQUE, определяющий возможный ключ отношения;
- конструкция FOREIGN KEY (REFERENCES), определяющая внешний ключ отношения;
- конструкция CHECK, определяющая условие на значения атрибута(-ов), которому должны удовлетворять все кортежи отношения.

Синтаксис команды CREATE TABLE (нотация Oracle) (начало)

relational_table::=

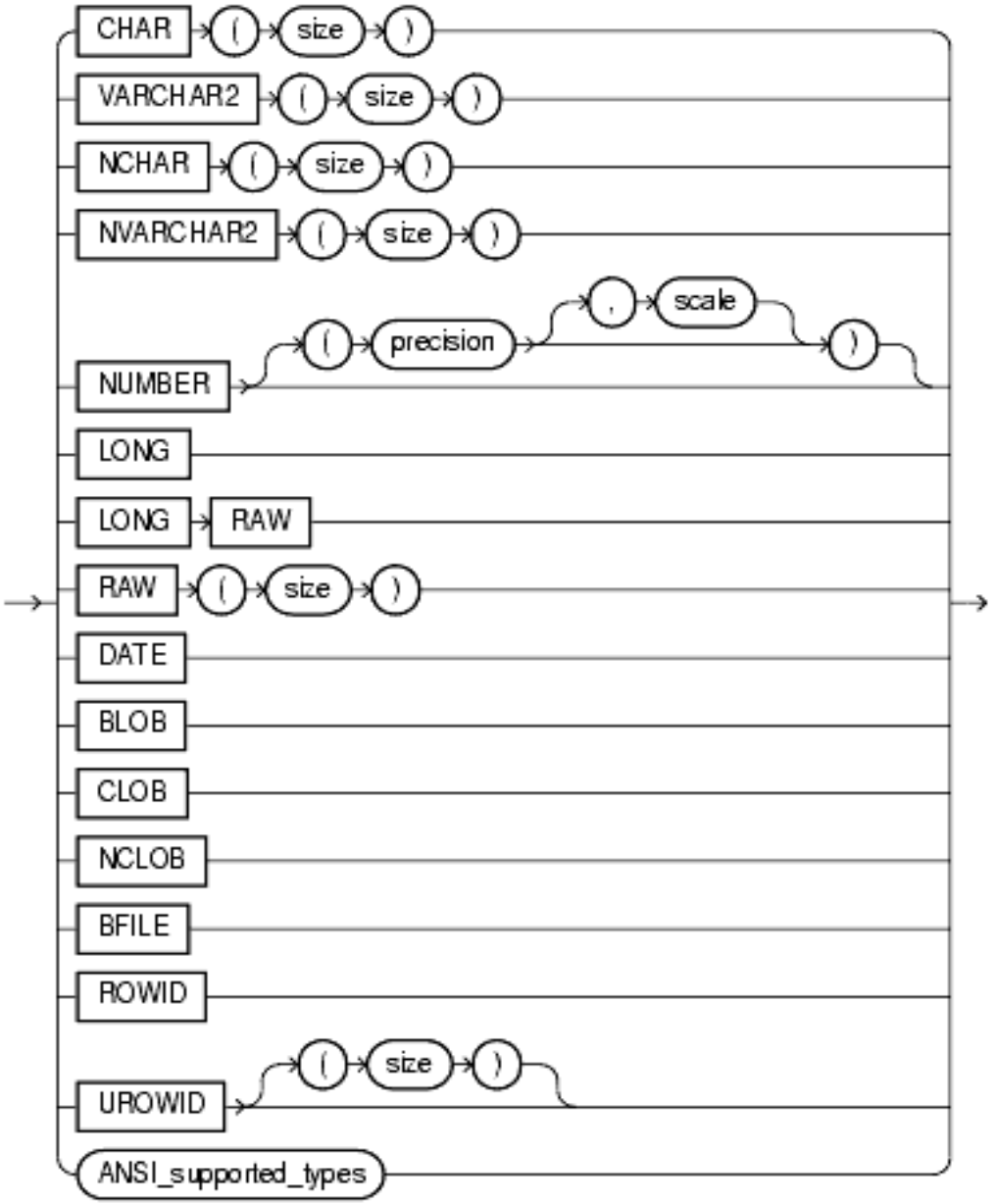


relational_properties::=



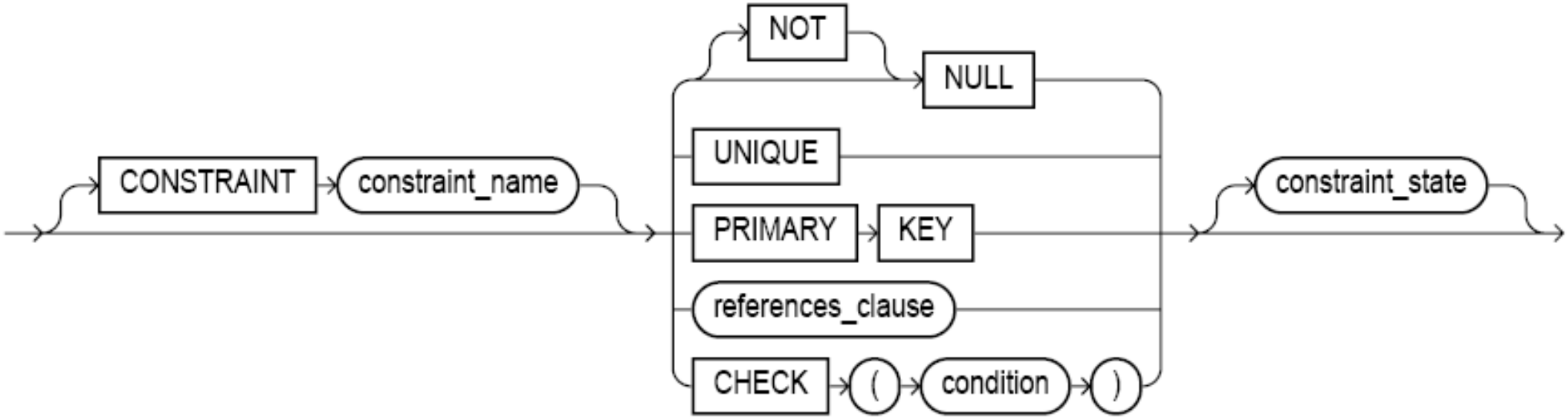
Синтаксис команды CREATE TABLE (нотация Oracle) (продолжение)

datatype::=

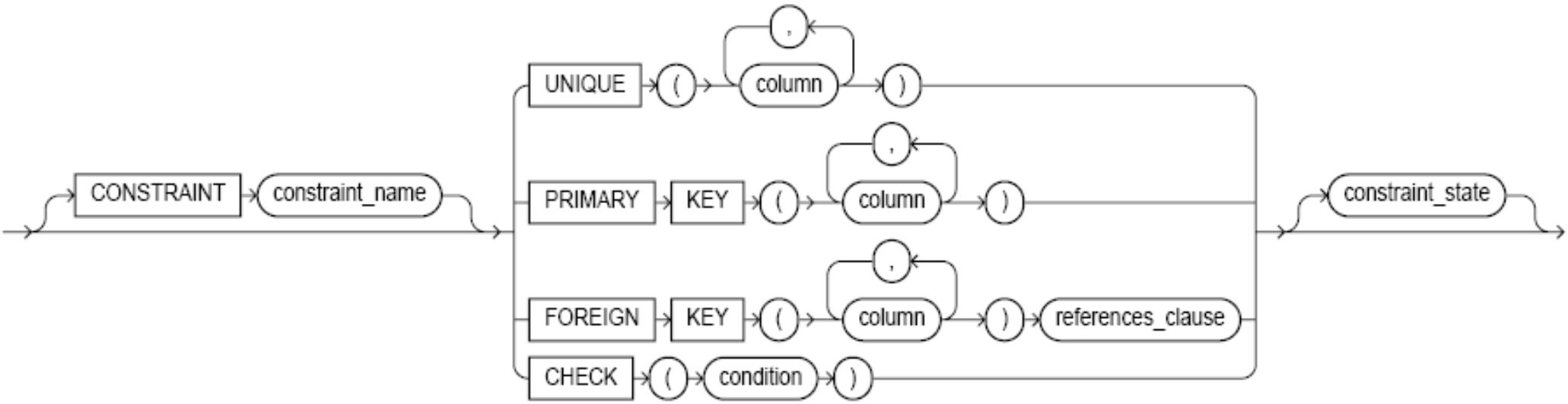


Синтаксис команды CREATE TABLE (нотация Oracle) (продолжение)

inline_constraint::=

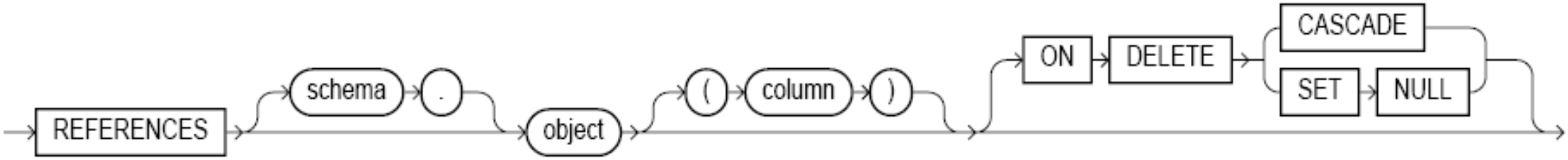


out_of_line_constraint::=



Синтаксис команды CREATE TABLE (нотация Oracle) (продолжение)

references_clause::=

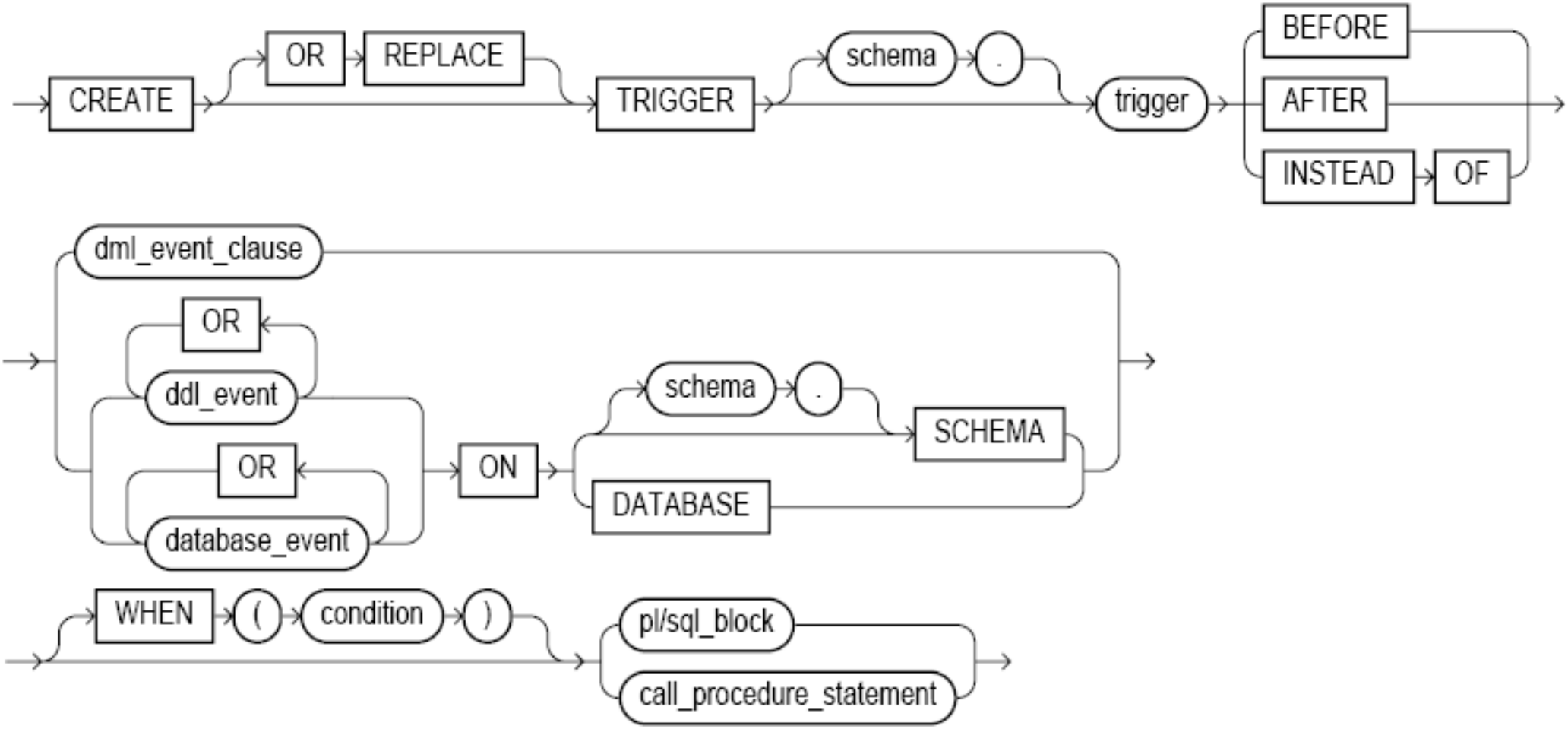


Команды CREATE TABLE для медицинской БД

```
CREATE TABLE БОЛЬНИЦА (
    К/Б                 NUMBER (6)          PRIMARY KEY,
    Название            VARCHAR2 (30)       NOT NULL,
    Адрес               VARCHAR2 (50),
    Ч/К                 NUMBER (4))
CREATE TABLE ПАЛАТА (
    К/Б                 NUMBER (6)          NOT NULL
                                            REFERENCES БОЛЬНИЦА (К/Б) ON DELETE CASCADE,
    Н/П                 NUMBER (6)          NOT NULL,
    Название            VARCHAR2 (30),
    Ч/К                 NUMBER (2),
    PRIMARY KEY (К/Б,Н/П))
CREATE TABLE ВРАЧ (
    К/В                 NUMBER (6)          PRIMARY KEY,
    К/Б                 NUMBER (6)          REFERENCES БОЛЬНИЦА (К/Б) ON DELETE SET NULL,
    Фамилия             VARCHAR2 (30)       NOT NULL,
    Специальность       VARCHAR2 (30))
CREATE TABLE ПАЦИЕНТ (
    Р/Н                 NUMBER (6)          PRIMARY KEY,
    Фамилия             VARCHAR2 (30)       NOT NULL,
    Адрес               VARCHAR2 (50),
    Д/Р                 DATE,
    Пол                 CHAR (1)            CHECK (Пол IN ('М','Ж')),
    НМП                 VARCHAR2 (20)       UNIQUE)
CREATE TABLE ВРАЧ-ПАЦИЕНТ (
    К/В                 NUMBER (6)          NOT NULL
                                            REFERENCES ВРАЧ (К/В) ON DELETE CASCADE,
    Р/Н                 NUMBER (6)          NOT NULL
                                            REFERENCES ПАЦИЕНТ (Р/Н) ON DELETE CASCADE,
    PRIMARY KEY (К/В,Р/Н))
```

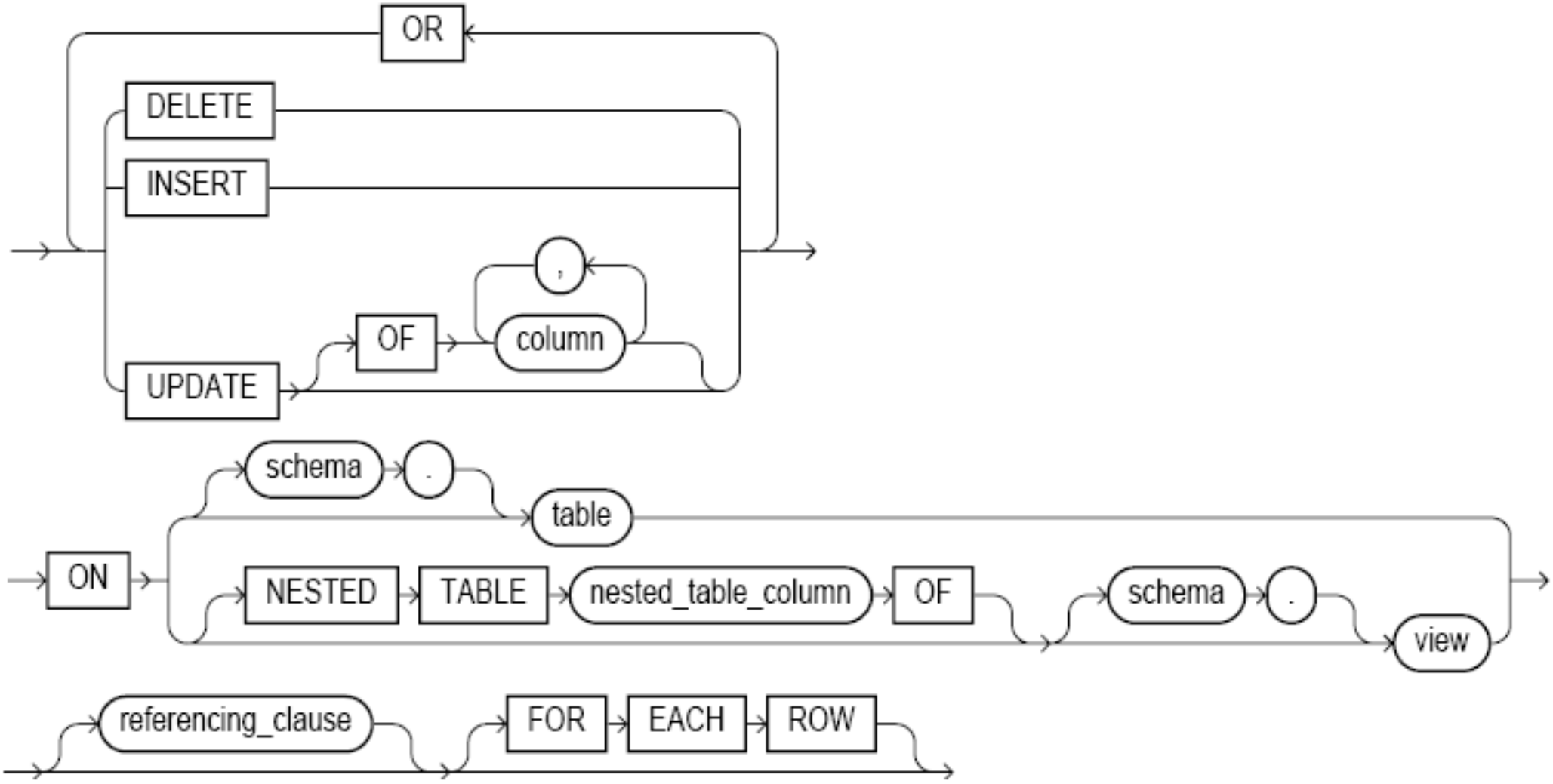
Синтаксис команды CREATE TRIGGER (нотация Oracle) (начало)

create_trigger::=



Синтаксис команды CREATE TRIGGER (нотация Oracle) (продолжение)

dml_event_clause::=



4.2.3. Навигационные операции

Курсоры

CURSOR cursor_name [(parameter[, parameter]...)] [RETURN return_type] IS select_statement;
OPEN cursor_name [(parameter[, parameter]...)];
FETCH cursor_name [BULK COLLECT] INTO var_name[, var_name]…;
CLOSE cursor_name;

Атрибуты для курсоров

Атрибут	*Значение*
%ISOPEN	Имеет значение TRUE, если курсор открыт и FALSE, если нет
%FOUND	Исключительное состояние INVALID_CURSOR, если курсор не открыт с помощью OPEN или закрыт с помощью CLOSE
	NULL – перед первым выполнением FETCH
	TRUE – после успешного выполнения FETCH
	FALSE – если FETCH не сумел выдать строку
%NOTFOUND	Исключительное состояние INVALID_CURSOR, если курсор не открыт с помощью OPEN или закрыт с помощью CLOSE
	NULL – перед первым выполнением FETCH
	FALSE – после успешного выполнения FETCH
	TRUE – если FETCH не сумел выдать строку
%ROWCOUNT	Исключительное состояние INVALID_CURSOR, если курсор не открыт с помощью OPEN или закрыт с помощью CLOSE
	Число – общее число строк, извлеченных после последней операции FETCH для курсора

Синтаксис использования атрибута: «*имя курсора*»«*атрибут*».

Курсоры (продолжение)

```
DECLARE
    p_name ПЕРСОНАЛ.Фамилия%TYPE;
    p_sal ПЕРСОНАЛ.З/П%TYPE;
    CURSOR cursor1 IS
        SELECT Фамилия, З/П FROM ПЕРСОНАЛ
        WHERE Должность = ‘СИДЕЛКА’;
BEGIN
…
    OPEN cursor1;
    LOOP
        FETCH cursor1 INTO p_name, p_sal;
        EXIT WHEN cursor1%NOTFOUND;
        -- обработка одной записи
    END LOOP;
    CLOSE cursor1;
…
END;
```

4.2.4. Спецификационные операции

Спецификационные языки реляционной модели:

- алгебраические языки (реляционная алгебра Кодда);

- языки исчисления предикатов
  - реляционное исчисление с переменными-кортежами (ALPHA),
  - реляционное исчисление с переменными на доменах (QBE);

- язык SQL.

4.2.4.1. Реляционная алгебра Кодда

Реляционная алгебра Кодда

Совместимые отношения* – это отношения, у которых совпадают заголовки (количество и имена атрибутов)

Основные операции

1. **ОБЪЕДИНЕНИЕ*** $R \cup S = \{t \mid t \in R \vee t \in S\}$
2. **РАЗНОСТЬ*** $R - S = \{t \mid t \in R \wedge t \notin S\}$
3. **ДЕКАРТОВО ПРОИЗВЕДЕНИЕ** $R \times S = \{r \| s \mid r \in R \wedge s \in S\}$
4. **ПРОЕКЦИЯ**

$$\pi_{i_1,\ldots,i_m}(R) = \{< a_1,\ldots,a_m > \mid \exists < b_1,\ldots,b_k > (< b_1,\ldots,b_k > \in R \wedge \forall j = \overline{1,m}(a_j = b_{i_j}))\}$$

5. **СЕЛЕКЦИЯ** $\sigma_F(R) = \{r \mid r \in R \wedge F\}$

Дополнительные операции

1. **ПЕРЕСЕЧЕНИЕ*** $R \cap S = \{t \mid t \in R \wedge t \in S\} = R - (R - S)$
2. **ЧАСТНОЕ**

$$R \div S = \{t^{(r-s)} \mid \forall u^{(s)}(u \in S \to t \| u \in R)\}$$

$$R \div S = \pi_{1,\ldots,r-s}(R) - \pi_{1,\ldots,r-s}((\pi_{1,\ldots,r-s}(R) \times S) - R)$$

3. **СОЕДИНЕНИЕ**

$$R \overset{\triangleright\triangleleft}{}_{i\theta j} S = \sigma_{i\theta(r+j)}(R \times S)$$

$$R \overset{\triangleright\triangleleft}{}_{X\theta Y} S = \{r \| s \mid r \in R \wedge s \in S \wedge (\pi_X(r)\theta\pi_Y(s))\}$$

Примеры

R

A	B
1	2
1	4
3	4

S

A	B
2	1
5	6
3	4

$R \cup S$

A	B
1	2
1	4
2	1
3	4
5	6

$R - S$

A	B
1	2
1	4

$R \cap S$

A	B
3	4

$\pi_2(R)$

B
2
4

$\sigma_{A=1}(R)$

A	B
1	2
1	4

Примеры

R

A	B	C	D
1	2	5	6
2	3	7	8
3	4	5	6

S

X	Y	Z
5	6	6
7	8	8

$R \times S$

A	B	C	D	X	Y	Z
1	2	5	6	5	6	6
1	2	5	6	7	8	8
2	3	7	8	5	6	6
2	3	7	8	7	8	8
3	4	5	6	5	6	6
3	4	5	6	7	8	8

$R \overset{\triangleright\triangleleft}{}_{\{C,D\}=\{X,Y\}} S$

A	B	C	D	X	Y	Z
1	2	5	6	5	6	6
2	3	7	8	7	8	8
3	4	5	6	5	6	6

Примеры

R

A	B	C	D
1	2	5	6
1	2	7	8
2	3	5	6
2	3	7	8
3	4	5	6

S

C	D
5	6
7	8

$R \div S$

A	B
1	2
2	3

Реляционная алгебра. Примеры запросов

Получить фамилии хирургов

SELECT ВРАЧ WHERE Специальность = ‘ХИРУРГ’ -> TEMP
PROJECT TEMP OVER Фамилия -> RESULT

Получить фамилии врачей, лечащих больных палаты №2 больницы №5

SELECT РАЗМЕЩЕНИЕ WHERE К/Б = 5 AND Н/П = 2 -> T1
PROJECT T1 OVER Р/Н -> T2
JOIN T2 AND ВРАЧ-ПАЦИЕНТ OVER Р/Н -> T3
PROJECT T3 OVER К/В -> T4
JOIN T4 AND ВРАЧ OVER К/В -> T5
PROJECT T5 OVER Фамилия -> RESULT

Получить фамилии врачей, лечащих всех больных палаты №2 больницы №5

SELECT РАЗМЕЩЕНИЕ WHERE К/Б = 5 AND Н/П = 2 -> T1
PROJECT T1 OVER Р/Н -> T2
DIVIDE ВРАЧ-ПАЦИЕНТ BY T2 OVER Р/Н -> T3
JOIN T3 AND ВРАЧ OVER К/В -> T4
PROJECT T4 OVER Фамилия -> RESULT

Реляционная алгебра. Примеры запросов

Добавить новую палату

UNION ПАЛАТА AND {(10, 15, ‘Реанимационная’, 1)} -> ПАЛАТА

Удалить палату

MINUS ПАЛАТА AND {(10, 15, ‘Реанимационная’, 1)} -> ПАЛАТА

Изменить значение атрибута в палате

MINUS ПАЛАТА AND {(10, 15, ‘Реанимационная’, 1)} -> ПАЛАТА

UNION ПАЛАТА AND {(10, 15, ‘Реанимационная’, 2)} -> ПАЛАТА

4.2.4.2. Реляционное исчисление с переменными-кортежами

Реляционное исчисление с переменными-кортежами

Запрос – $\{t \mid \psi(t)\}$

Атомы:

1. $R(s)$
2. $s[i]\Theta u[j]$
3. $s[i]\Theta a$ и $a\Theta s[i]$

Правила образования формул:

1. Атом – формула.
2. Если ψ_1 и ψ_2- формулы, то $\psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \neg\psi_1$ - тоже формулы.
3. Если ψ - формула, то $\exists s(\psi)$ - тоже формула.
4. Если ψ - формула, то $\forall s(\psi)$ - тоже формула.
5. Приоритет операций – $\Theta, \begin{Bmatrix}\exists \\ \forall\end{Bmatrix}, \neg, \wedge, \vee$

   Круглые скобки для изменения приоритета.
6. Ничто иное не является формулой.

Реляционное исчисление с переменными-кортежами

Реализация основных операций реляционной алгебры

Объединение – $R \cup S = \{t \mid R(t) \vee S(t)\}$

Разность – $R - S = \{t \mid R(t) \wedge \neg S(t)\}$

Декартово произведение –

$$R^{(r)} \times S^{(s)} = \{t^{(r+s)} \mid \exists u^{(r)} \exists v^{(s)} (R(u) \wedge S(v) \wedge t[1] = u[1] \wedge \ldots$$

$$\ldots \wedge t[r] = u[r] \wedge t[r+1] = v[1] \wedge \ldots \wedge t[r+s] = v[s])\}$$

Проекция –

$$\pi_{i_1, i_2, \ldots, i_k}(R) = \{t^{(k)} \mid \exists u (R(u) \wedge t[1] = u[i_1] \wedge \ldots \wedge t[k] = u[i_k])\}$$

Селекция – $\sigma_F(R) = \{t \mid R(t) \wedge F'\}$,

где *F'*- это *F*, в которой *i* заменено на *t[i]*

Реляционное исчисление с переменными-кортежами. Язык ALPHA. Примеры запросов

Получить полные сведения о больницах

$$\{t \mid БОЛЬНИЦА(t)\}$$

GET W (БОЛЬНИЦА)

GET W (БОЛЬНИЦА.К/Б, БОЛЬНИЦА.Название, БОЛЬНИЦА.Адрес, БОЛЬНИЦА.Ч/К)

Получить фамилии хирургов

$$\{t \mid ВРАЧ(t) \wedge t[\text{Специальность}] = '\text{ХИРУРГ}'\}$$

GET W (ВРАЧ.Фамилия):
ВРАЧ.Специальность = ‘ХИРУРГ’

Язык ALPHA. Примеры запросов (продолжение)

Получить названия и число коек палат больницы с кодом 5, упорядочить их по возрастанию числа коек

```
GET W (ПАЛАТА.Название, ПАЛАТА.Ч/К):
    ПАЛАТА.К/Б = 5 UP ПАЛАТА.Ч/К
```

Получить Р/Н и фамилию самого молодого пациента

```
GET W (1) (ПАЦИЕНТ.Р/Н, ПАЦИЕНТ.Фамилия):
    DOWN ПАЦИЕНТ.Д/Р
```

Получить коды врачей, лечащих пациента с Р/Н 111111

```
GET W (ВРАЧ-ПАЦИЕНТ.К/В):
    ВРАЧ-ПАЦИЕНТ.Р/Н = 111111
```

```
RANGE ВРАЧ-ПАЦИЕНТ X
GET W (X.К/В):
     X.Р/Н = 111111
```

Язык ALPHA. Примеры запросов (продолжение)

Получить фамилии врачей, лечащих пациента с Р/Н 111111

RANGE ВРАЧ-ПАЦИЕНТ X

GET W (ВРАЧ.Фамилия):

∃X (X.К/В = ВРАЧ.К/В ∧ X.Р/Н = 111111)

Получить фамилии врачей, лечащих женщин (вар.1)

RANGE ПАЦИЕНТ X

GET W1 (ВРАЧ-ПАЦИЕНТ.К/В):

∃X (X.Р/Н = ВРАЧ-ПАЦИЕНТ.Р/Н ∧ X.Пол = ‘ЖЕНСКИЙ’)

RANGE W1 WX

GET W2 (ВРАЧ.Фамилия):

∃WX (WX.К/В = ВРАЧ.К/В)

Язык ALPHA. Примеры запросов (продолжение)

Получить фамилии врачей, лечащих женщин (вар.2)

RANGE ПАЦИЕНТ X

RANGE ВРАЧ-ПАЦИЕНТ Y

GET W (ВРАЧ.Фамилия):

$\exists Y$ (Y.К/В = ВРАЧ.К/В $\wedge$ $\exists X$ (X.Р/Н = Y.Р/Н $\wedge$ X.Пол = ‘ЖЕНСКИЙ’))

или

$\exists X$ $\exists Y$ (Y.К/В = ВРАЧ.К/В $\wedge$ X.Р/Н = Y.Р/Н $\wedge$ X.Пол = ‘ЖЕНСКИЙ’)

Получить фамилии врачей, которые лечат тех же пациентов, что и врач с кодом 999

RANGE ВРАЧ-ПАЦИЕНТ X

RANGE ВРАЧ-ПАЦИЕНТ Y

GET W (ВРАЧ.Фамилия):

$\exists X$ (X.К/В = ВРАЧ.К/В $\wedge$ $\exists Y$ (Y.Р/Н = X.Р/Н $\wedge$ Y.К/В = 999))

Язык ALPHA. Примеры запросов (продолжение)

Получить список фамилий больных с их диагнозами

GET W (ПАЦИЕНТ.Фамилия, ДИАГНОЗ.Т/Д):

ДИАГНОЗ.Р/Н = ПАЦИЕНТ.Р/Н

Получить фамилии врачей, которые не лечат пациента с номером 111111

RANGE ВРАЧ-ПАЦИЕНТ X

GET W (ВРАЧ.Фамилия):

$\forall X$ **(X.К/В** $\neq$ **ВРАЧ.К/В** $\vee$ **X.Р/Н** $\neq$ **111111)**

или

$\neg \exists X$ **(X.К/В = ВРАЧ.К/В** $\wedge$ **X.Р/Н = 111111)**

Язык ALPHA. Примеры запросов (продолжение)

Получить фамилии пациентов, которые лечатся у всех врачей

RANGE ВРАЧ-ПАЦИЕНТ X

RANGE ВРАЧ Y

GET W (ПАЦИЕНТ.Фамилия):

$$\forall Y\ \exists X\ (X.Р/Н = ПАЦИЕНТ.Р/Н \wedge X.К/В = Y.К/В)$$

Получить Р/Н пациентов, которые лечатся у всех врачей

RANGE ВРАЧ-ПАЦИЕНТ X

RANGE ВРАЧ Y

GET W (ПАЦИЕНТ.Р/Н):

$$\forall Y\ \exists X\ (X.Р/Н = ПАЦИЕНТ.Р/Н \wedge X.К/В = Y.К/В)$$

неверно:

GET W (X.Р/Н): $\forall Y\ \exists X\ (X.К/В = Y.К/В)$

GET W (X.Р/Н): $\forall Y\ (X.К/В = Y.К/В)$

Язык ALPHA. Примеры запросов (продолжение)

Получить фамилии пациентов, которые лечатся у всех практикующих врачей

RANGE ВРАЧ-ПАЦИЕНТ X

RANGE ВРАЧ-ПАЦИЕНТ Y

GET W (ПАЦИЕНТ.Фамилия):

$$\forall Y\ \exists X\ (X.Р/Н = ПАЦИЕНТ.Р/Н \wedge X.К/В = Y.К/В)$$

Получить коды врачей, лечащих по крайней мере тех же пациентов, что и врач с кодом 999

RANGE ПАЦИЕНТ X

RANGE ВРАЧ-ПАЦИЕНТ Y

RANGE ВРАЧ-ПАЦИЕНТ Z

GET W (ВРАЧ.К/В):

$$\forall X\ (\exists Y\ (Y.К/В = 999 \wedge Y.Р/Н = X.Р/Н) \rightarrow$$

$$\exists Z\ (Z.К/В = ВРАЧ.К/В \wedge Z.Р/Н = X.Р/Н))$$

Язык ALPHA. Примеры запросов (продолжение). Операции модификации БД

Изменить значение атрибута «Число коек» у больницы с кодом 2, сделать его равным 100

```
HOLD W (БОЛЬНИЦА.К/Б, БОЛЬНИЦА.Ч/К):
    БОЛЬНИЦА.К/Б = 2
W.Ч/К = 100
UPDATE W
```

Добавить в БД новую больницу

```
W.К/Б = 10
W.Название = ‘Госпиталь Святой Елены’
W.Адрес = ‘Торонто’
W.Ч/К = 200
PUT W (БОЛЬНИЦА)
```

Язык ALPHA. Примеры запросов (продолжение). Операции модификации БД

Удалить больницу с кодом 5

```
HOLD W (БОЛЬНИЦА):
        БОЛЬНИЦА.К/Б = 5
DELETE W
```

Изменить код больницы с 2 на 20

```
HOLD W (БОЛЬНИЦА):
        БОЛЬНИЦА.К/Б = 2
DELETE W
W.К/Б = 20
PUT W (БОЛЬНИЦА)
```

Удалить все анализы

```
HOLD W (АНАЛИЗ)
DELETE W
```

4.2.4.3. Реляционное исчисление с переменными на доменах

Реляционное исчисление с переменными на доменах

Запрос – $\{x_1 x_2 ... x_k \mid \psi(x_1, x_2, ..., x_k)\}$

Атомы:

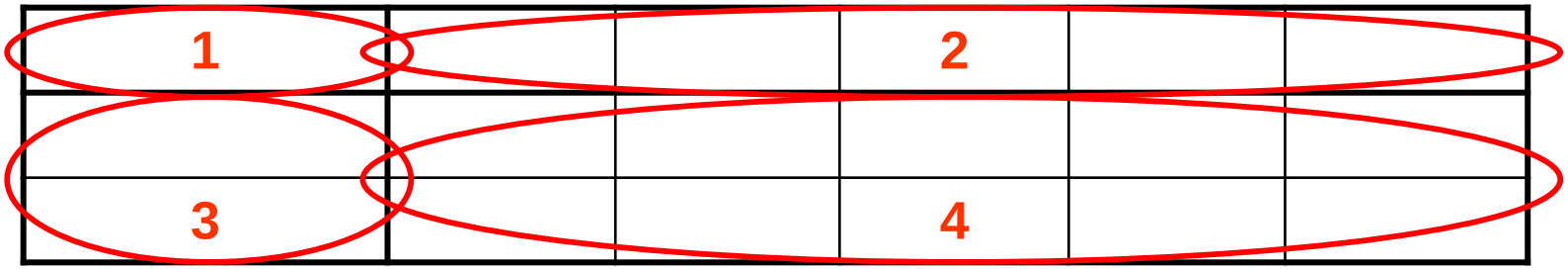
1. $R(x_1 x_2 ... x_k)$
2. $x \Theta y$

Правила образования формул:

1. Атом – формула.
2. Если ψ_1 и ψ_2 - формулы, то $\psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \neg\psi_1$ - тоже формулы.
3. Если ψ - формула, то $\exists x(\psi)$ - тоже формула.
4. Если ψ - формула, то $\forall x(\psi)$ - тоже формула.
5. Приоритет операций – $\Theta, \left\{\begin{matrix}\exists \\ \forall\end{matrix}\right\}, \neg, \wedge, \vee$

   Круглые скобки для изменения приоритета.
6. Ничто иное не является формулой.

Язык QBE



Области таблицы-шаблона запроса:

1. **Действия с отношениями.**
2. **Действия с атрибутами.**
3. **Действия с кортежами.**
4. **Действия с компонентами кортежей.**

Основная грамматическая конструкция

[<действие>.][<операция сравнения>]
[{<элемент-константа>|<элемент-пример>}]

<действие>::= {I|U|D|P}

Язык QBE. Диалог с пользователем

P. ТАВ				

БОЛЬНИЦА P.				

БОЛЬНИЦА	К/Б	Название	Адрес	Ч/К

Реляционное исчисление с переменными на доменах. Язык QBE. Примеры запросов

Получить полные сведения о больницах

$$\{x_1 x_2 x_3 x_4 \mid БОЛЬНИЦА(x_1 x_2 x_3 x_4)\}$$

БОЛЬНИЦА	К/Б	Название	Адрес	Ч/К
	P.<u>X1</u>	P.<u>X2</u>	P.<u>X3</u>	P.<u>X4</u>

БОЛЬНИЦА	К/Б	Название	Адрес	Ч/К
P.				

Получить фамилии хирургов

$$\{f \mid \exists x \exists y(ВРАЧ(xyf'ХИРУРГ'))\}$$

ВРАЧ	К/В	К/Б	Фамилия	Специальность
			P.<u>F</u>	ХИРУРГ

Язык QBE. Примеры запросов (продолжение)

Выдать названия палат больницы с кодом 5, имеющих более 10 коек

ПАЛАТА	К/Б	Н/П	Название	Ч/К
	5		P.X	>10

Выдать фамилии пациентов – женщин до 30 лет или мужчин после 50 лет

ПАЦИЕНТ	Р/Н	Фамилия	Адрес	Д/Р	Пол	НМП
		P.X		>1976	Ж	
		P.Y		<1956	М	

Выдать фамилии пациентов от 30 до 50 лет

ПАЦИЕНТ	Р/Н	Фамилия	Адрес	Д/Р	Пол	НМП
		P.X		<1976		
		X		>1956		

Язык QBE. Примеры запросов (продолжение)

Выдать фамилии врачей, лечащих пациента с Р/Н 111111

ВРАЧ	К/В	К/Б	Фамилия	Специальность
	Y		Р.X	

ВРАЧ-ПАЦИЕНТ	К/В	Р/Н
	Y	111111

Выдать фамилии врачей, не лечащих пациента с Р/Н 111111

ВРАЧ	К/В	К/Б	Фамилия	Специальность
	Y		Р.X	

ВРАЧ-ПАЦИЕНТ	К/В	Р/Н
]	Y	111111

Выдать фамилии врачей, лечащих кого-нибудь, кроме пациента с Р/Н 111111

ВРАЧ	К/В	К/Б	Фамилия	Специальность
	Y		Р.X	

ВРАЧ-ПАЦИЕНТ	К/В	Р/Н
	Y	<>111111

Язык QBE. Примеры запросов (продолжение)

Выдать коды врачей, лечащих по крайней мере одного больного, что и врач с кодом 999

ВРАЧ-ПАЦИЕНТ	К/В	Р/Н
	P.X	Y
	999	Y

Выдать Р/Н пациентов, которые ходили ко всем врачам

ВРАЧ	К/В	К/Б	Фамилия	Специальность
	ALL.Y			

ВРАЧ-ПАЦИЕНТ	К/В	Р/Н
	ALL.Y	P.X

Язык QBE. Примеры запросов (продолжение)

Выдать коды врачей, лечащих по крайней мере тех же больных, что и врач с кодом 999

ВРАЧ-ПАЦИЕНТ	К/В	Р/Н
	P.X	ALL.Y *
	999	ALL.Y

Язык QBE. Примеры запросов (продолжение). Операции модификации БД

Добавить нового служащего

ПЕРСОНАЛ	К/Б	Н/П	Фамилия	Должность	Смена	З/П
I.	5	2	Смит	Няня	Н	1000

Сделать всему персоналу палаты номер 2 больницы с кодом 5 дневную смену

ПЕРСОНАЛ	К/Б	Н/П	Фамилия	Должность	Смена	З/П
	5	2			U.Д	

Удалить служащих по фамилии Смит

ПЕРСОНАЛ	К/Б	Н/П	Фамилия	Должность	Смена	З/П
D.			Смит			

Удалить всех служащих

ПЕРСОНАЛ	К/Б	Н/П	Фамилия	Должность	Смена	З/П
D.						

Язык QBE. Примеры на определение схемы отношения

I. БОЛЬНИЦА I.	К/Б	Название	Адрес	Ч/К

I. БОЛЬНИЦА I.	К/Б	Название	Адрес	Ч/К
P. XX				

БОЛЬНИЦА	К/Б	Название	Адрес	Ч/К
TYPE LENGTH KEY DOMAIN NULL				

БОЛЬНИЦА		К/Б	Название	Адрес	Ч/К
TYPE	I.	INTEGER	CHAR	CHAR	INTEGER
LENGTH	I.	4	40	50	5
KEY	I.	K	NK	NK	NK
DOMAIN	I.	IDENT	NAME	ADDRESS	-
NULL	I.	NOT	NOT	YES	YES

Язык QBE. Примеры на определение схемы отношения (продолжение)

БОЛЬНИЦА	К/Б	Название	Адрес	Ч/К
P. XX P.				

БОЛЬНИЦА	К/Б	Название	Адрес	Ч/К	I. УРОВЕНЬ
TYPE	INTEGER	CHAR	CHAR	INTEGER	I. CHAR
LENGTH	4	40	50	5	I. 20
KEY	K	NK	NK	NK	I. NK
DOMAIN	IDENT	NAME	ADDRESS	-	I. ATO_TYPE
NULL	NOT	NOT	YES	YES	I. YES

4.2.4.4. Язык SQL

Стандарт SQL. Конструкции запросов

<cursor specification> ::= <query expression> [<order by clause>]
<query expression> ::= <query term> | <query expression> {UNION
[ALL] | INTERSECT | MINUS} <query term>
<query term> ::= <query specification> | (<query expression>)
<query specification> ::= (SELECT [ALL | DISTINCT] <select list>
<table expression>)

<select statement> ::= SELECT [ALL | DISTINCT] <select list> INTO
<select target list> <table expression>

<subquery> ::= (SELECT [ALL | DISTINCT] <result specification>
<table expression>)

<table expression> ::= <from clause> [<where clause>]
[<group by clause>] [<having clause>]

Стандарт SQL. Конструкции табличного выражения

<from clause> ::= FROM <table reference> [{,<table reference>}...]
<table reference> ::= <table name> [<correlation name>]

<where clause> ::= WHERE <search condition>
<search condition> ::= <boolean term> | <search condition> OR
<boolean term>
<boolean term> ::= <boolean factor> | <boolean term> AND
<boolean factor>
<boolean factor> ::= [NOT] <boolean primary>
<boolean primary> ::= <predicate> | (<search condition>)

<comparison predicate> ::= <value expression> <comp op>
{<value expression> | <subquery>}
<comp op> ::= = | <> | < | > | <= | >=

<between predicate> ::= <value expression> [NOT] BETWEEN
<value expression> AND <value expression>

Стандарт SQL. Конструкции табличного выражения (продолжение)

<in predicate> ::= <value expression> [NOT] IN
{<subquery> | (<in value list>)}
<in value list> ::= <value specification> {,<value specification>}...

<like predicate> ::= <column specification> [NOT] LIKE
<pattern> [ESCAPE <escape character>]
<pattern> ::= <value specification>
<escape character> ::= <value specification>

<null predicate> ::= <column specification> IS [NOT] NULL

<quantified predicate> ::= <value expression> <comp op> <quantifier>
<subquery>
<quantifier> ::= <all> | <some>
<all> ::= ALL
<some> ::= SOME | ANY

<exists predicate> ::= EXISTS <subquery>

Стандарт SQL. Конструкции табличного выражения (продолжение)

<group by clause> ::= GROUP BY <column specification>
[{,<column specification>}...]

<having clause> ::= HAVING <search condition>

Агрегатные функции

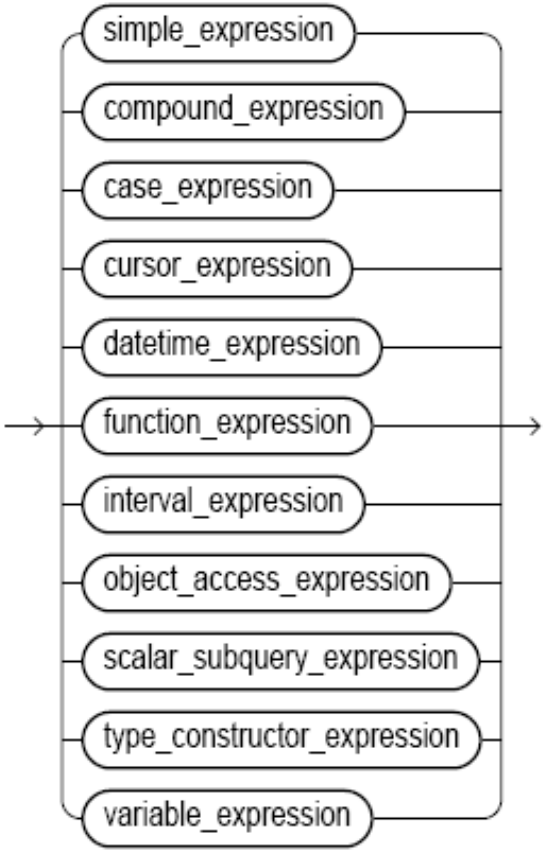
<set function specification> ::= COUNT(*) | <distinct set function> |
<all set function>
<distinct set function> ::= { AVG | MAX | MIN | SUM | COUNT }
(DISTINCT <column specification>)
<all set function> ::= { AVG | MAX | MIN | SUM } ([ALL]
<value expression>)

SQL (диалект Oracle)
Выражения (expressions) (начало)

Выражения используются:

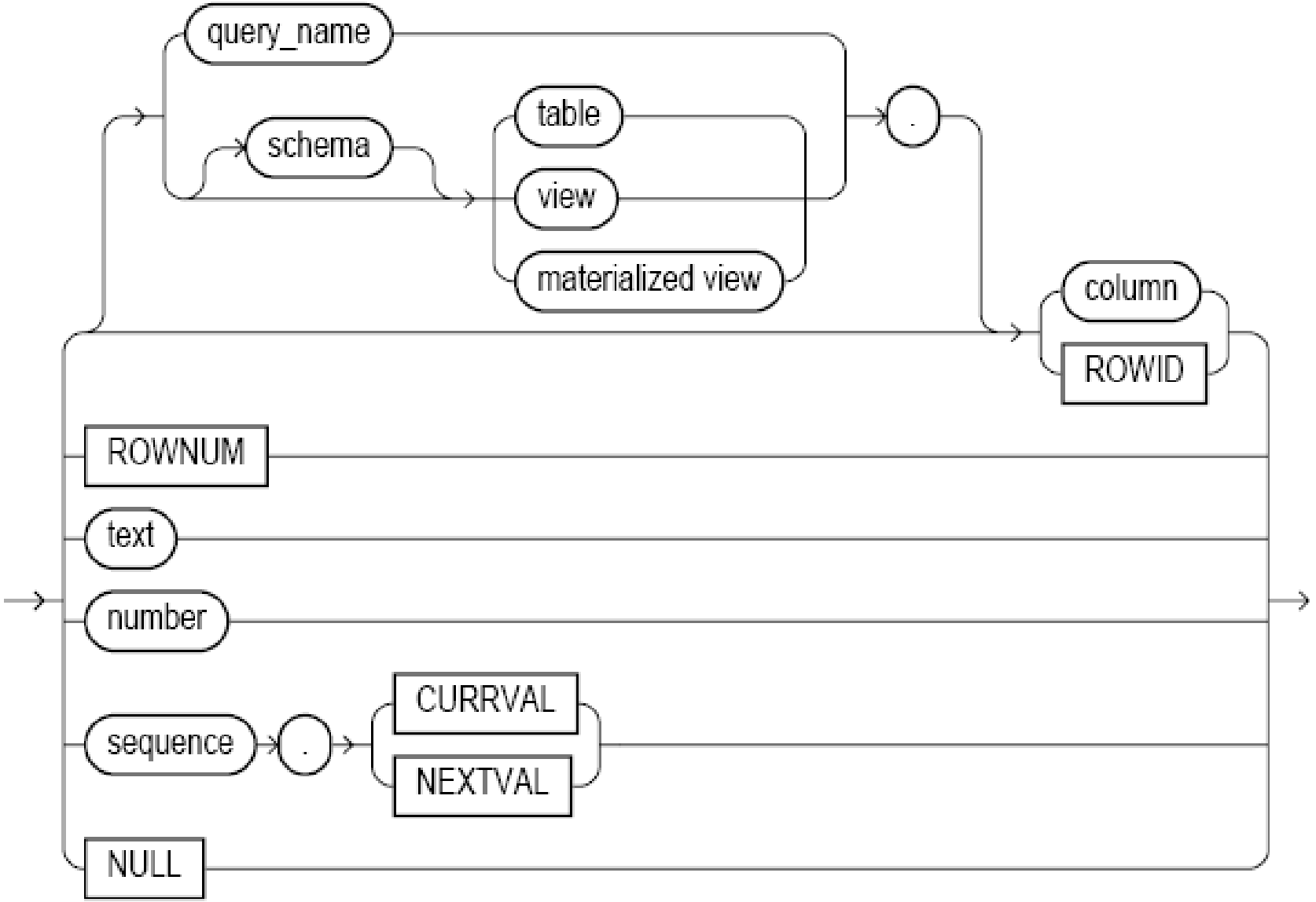
- **в целевых списках команд SELECT**
- **в условиях конструкций WHERE и HAVING**
- **в конструкциях CONNECT BY, START WITH и ORDER BY**
- **в конструкциях VALUES команд INSERT**
- **в конструкциях SET команд UPDATE**

expr::=



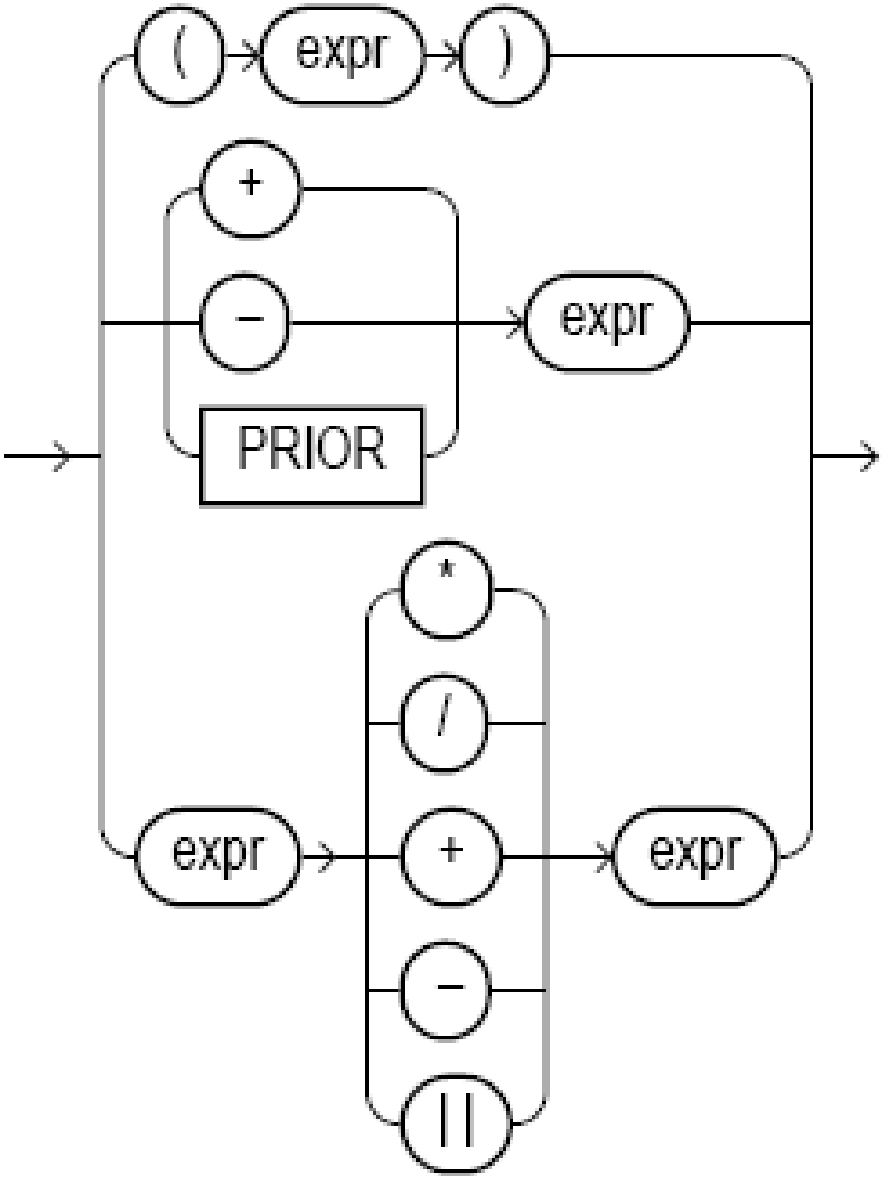
Выражения (expressions) (продолжение)

simple_expression::=



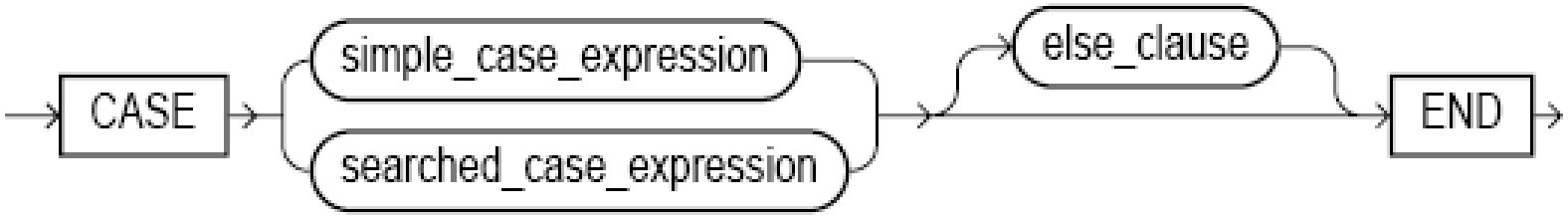
Выражения (expressions) (продолжение)

compound_expression::=

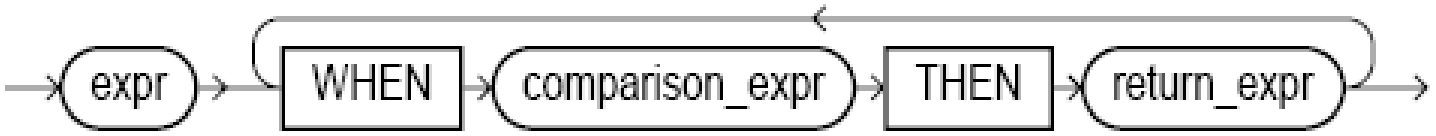


Выражения (expressions) (продолжение)

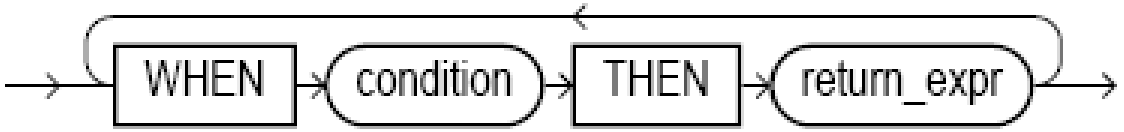
case_expression::=



simple_case_expression::=



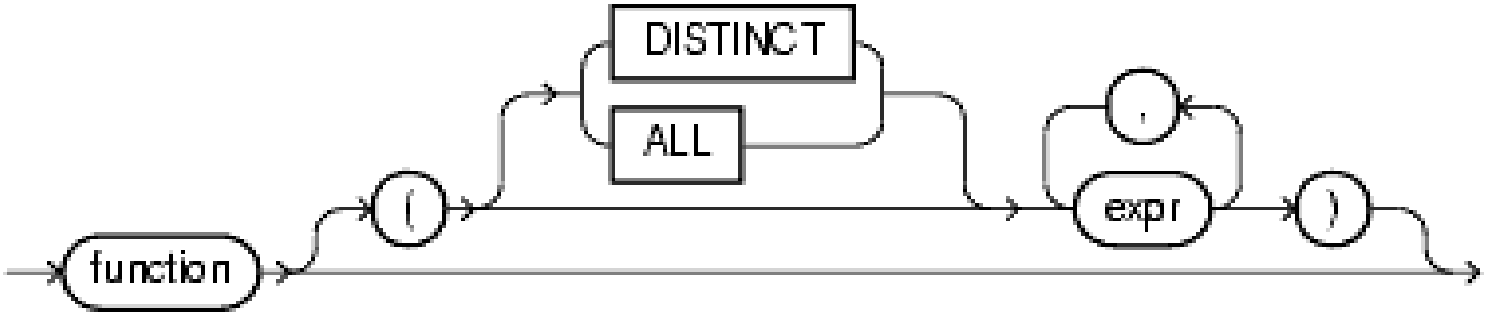
searched_case_expression::=



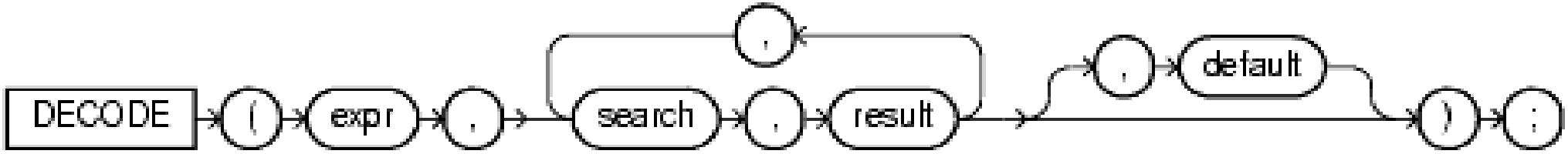
else_clause::=

Выражения (expressions) (продолжение)

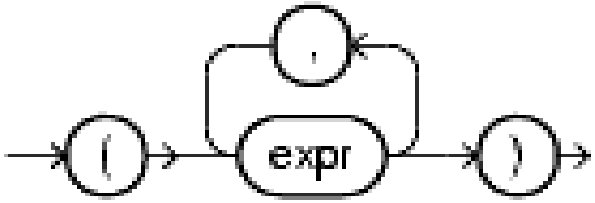
function_expression::=



DECODE_expression::=



expression_list::=

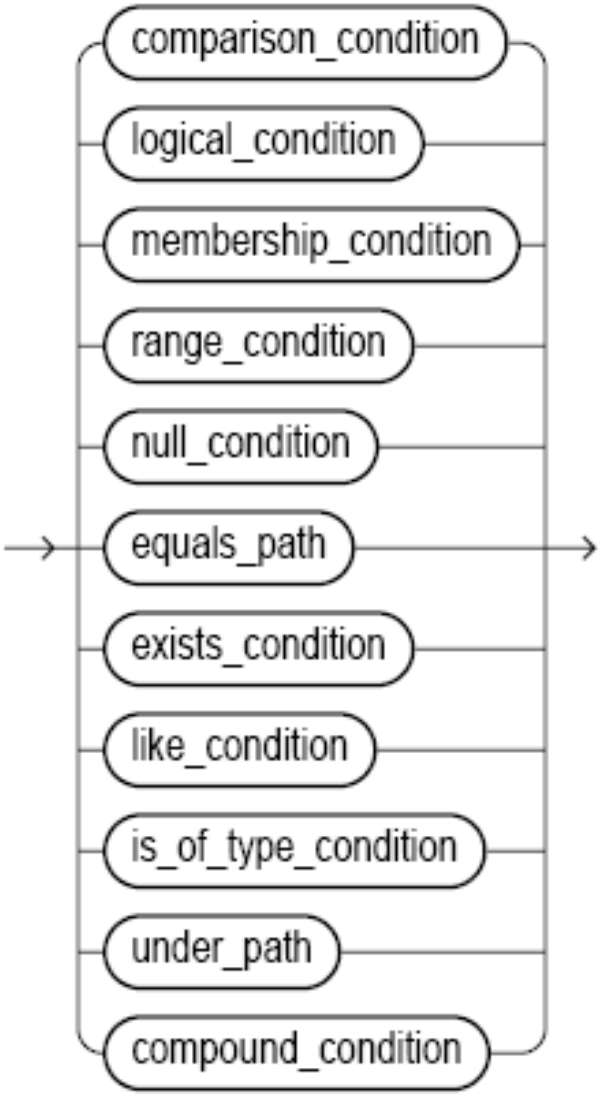


Условия (conditions) (начало)

Условия используются в командах SELECT, UPDATE и DELETE в конструкциях :

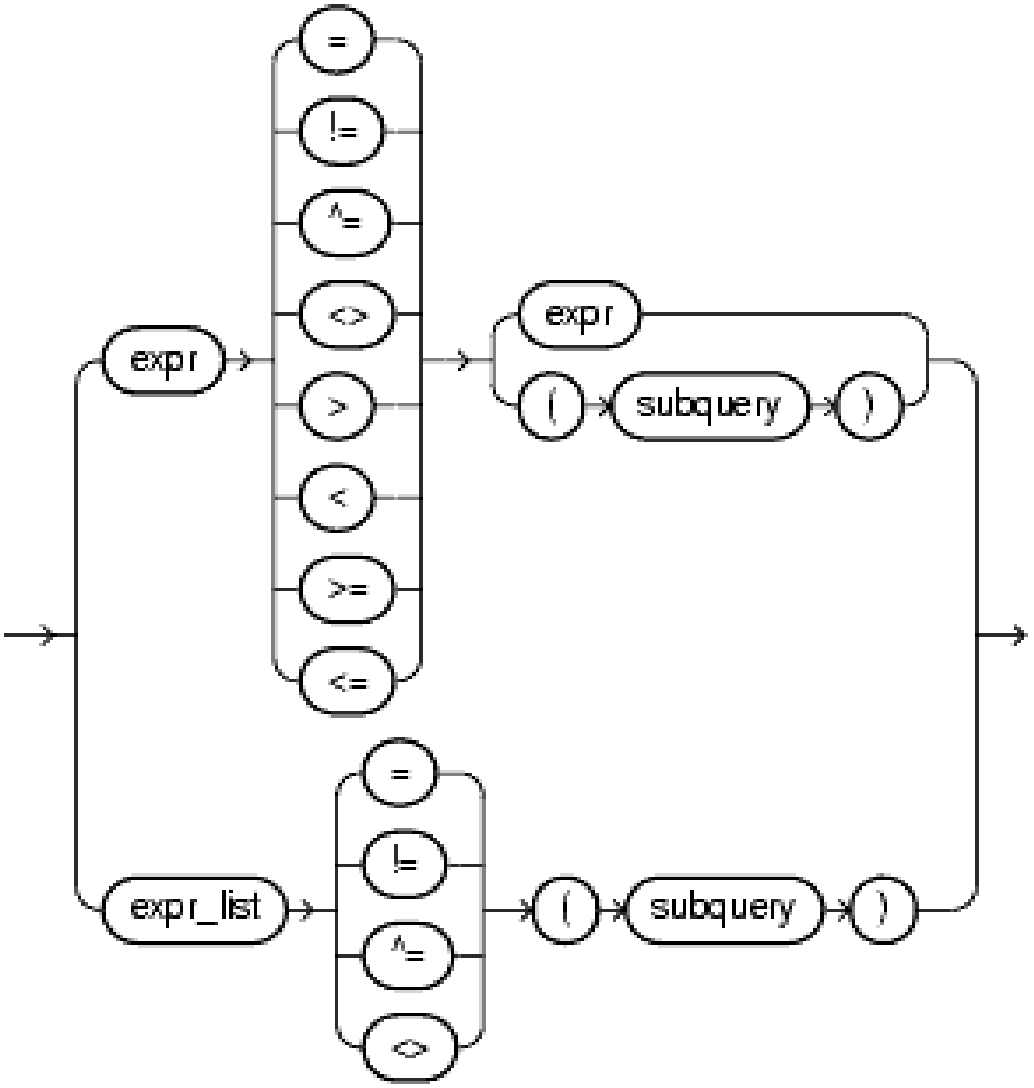
- **WHERE и HAVING**
- **CONNECT BY и START WITH**

condition::=



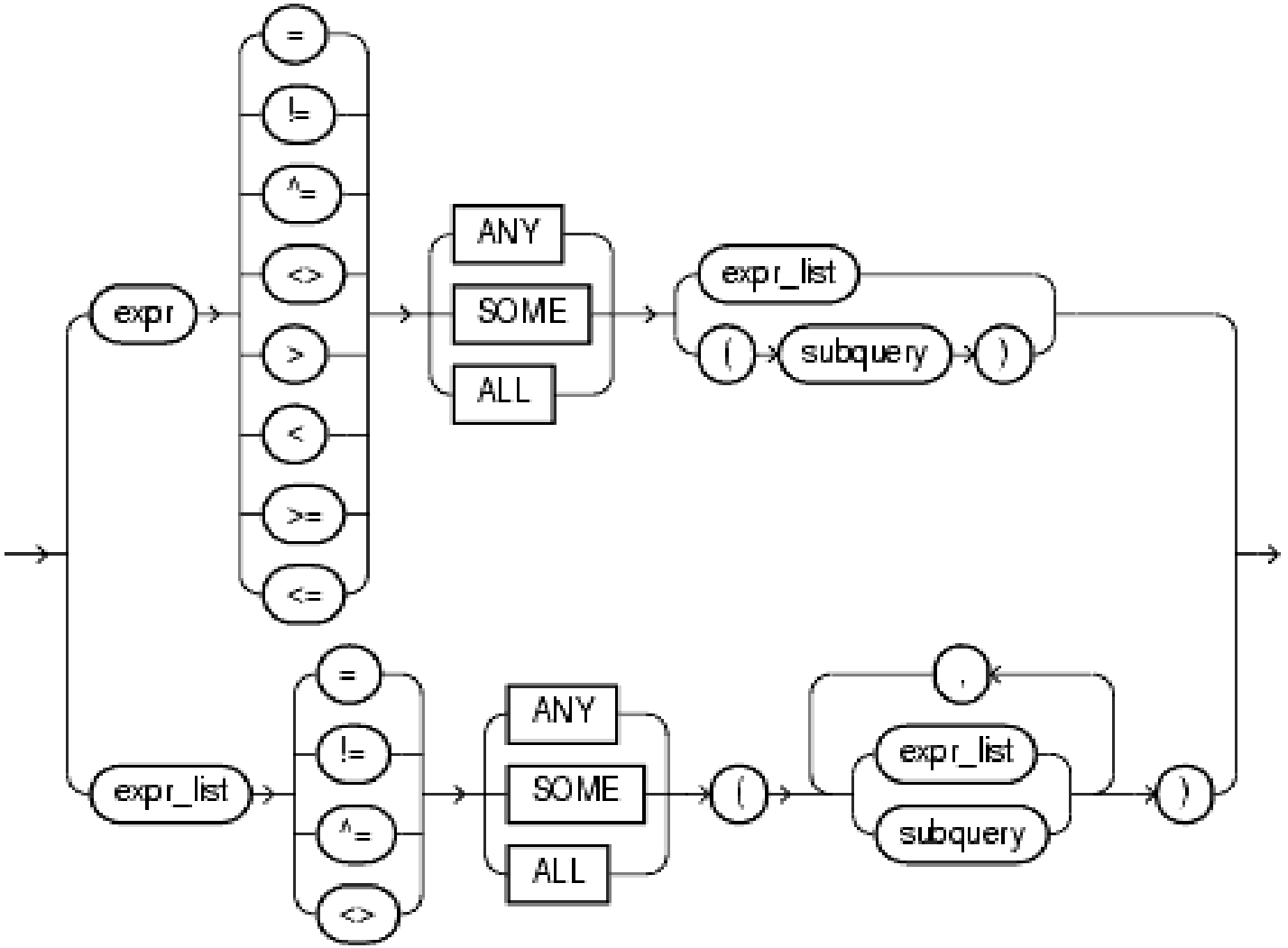
Условия (conditions) (продолжение)

simple_comparison_condition::=



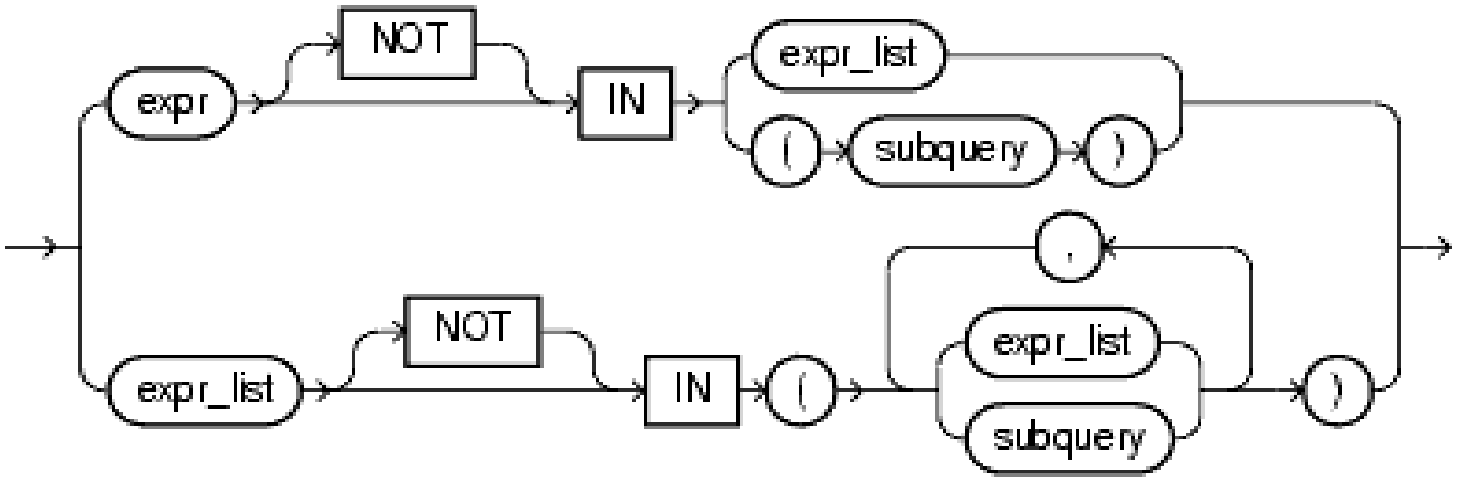
Условия (conditions) (продолжение)

group_comparison_condition::=



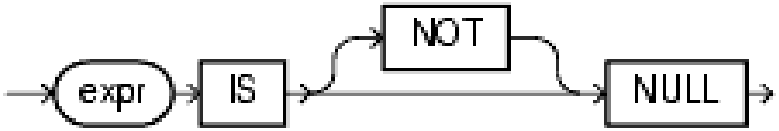
Условия (conditions) (продолжение)

membership_condition::=



range_condition::=

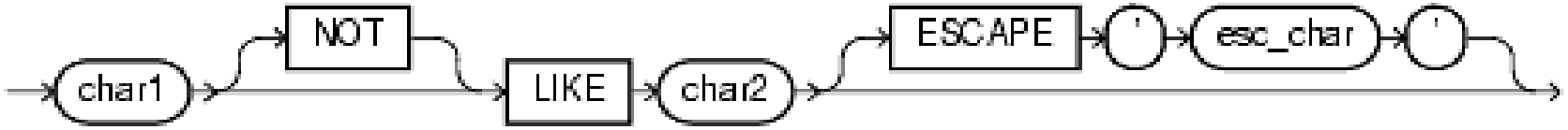
null_condition::=



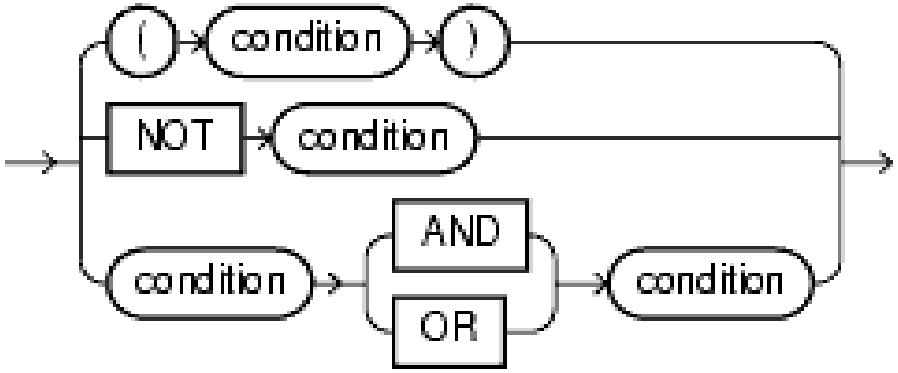
Условия (conditions) (продолжение)

exists_condition::=

like_condition::=

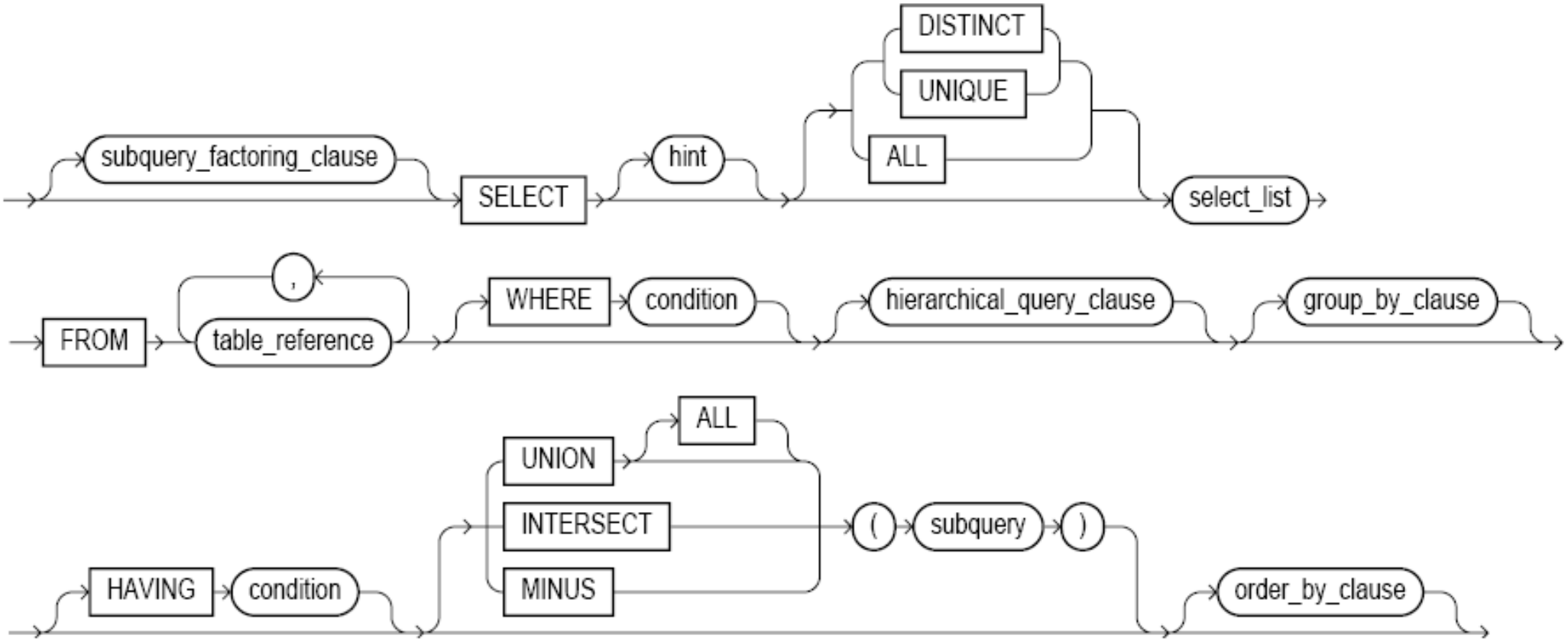


compound_condition::=



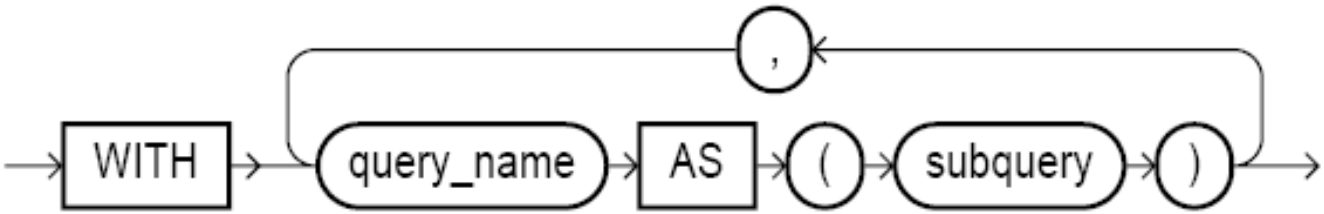
Команда SELECT (начало)

subquery::=

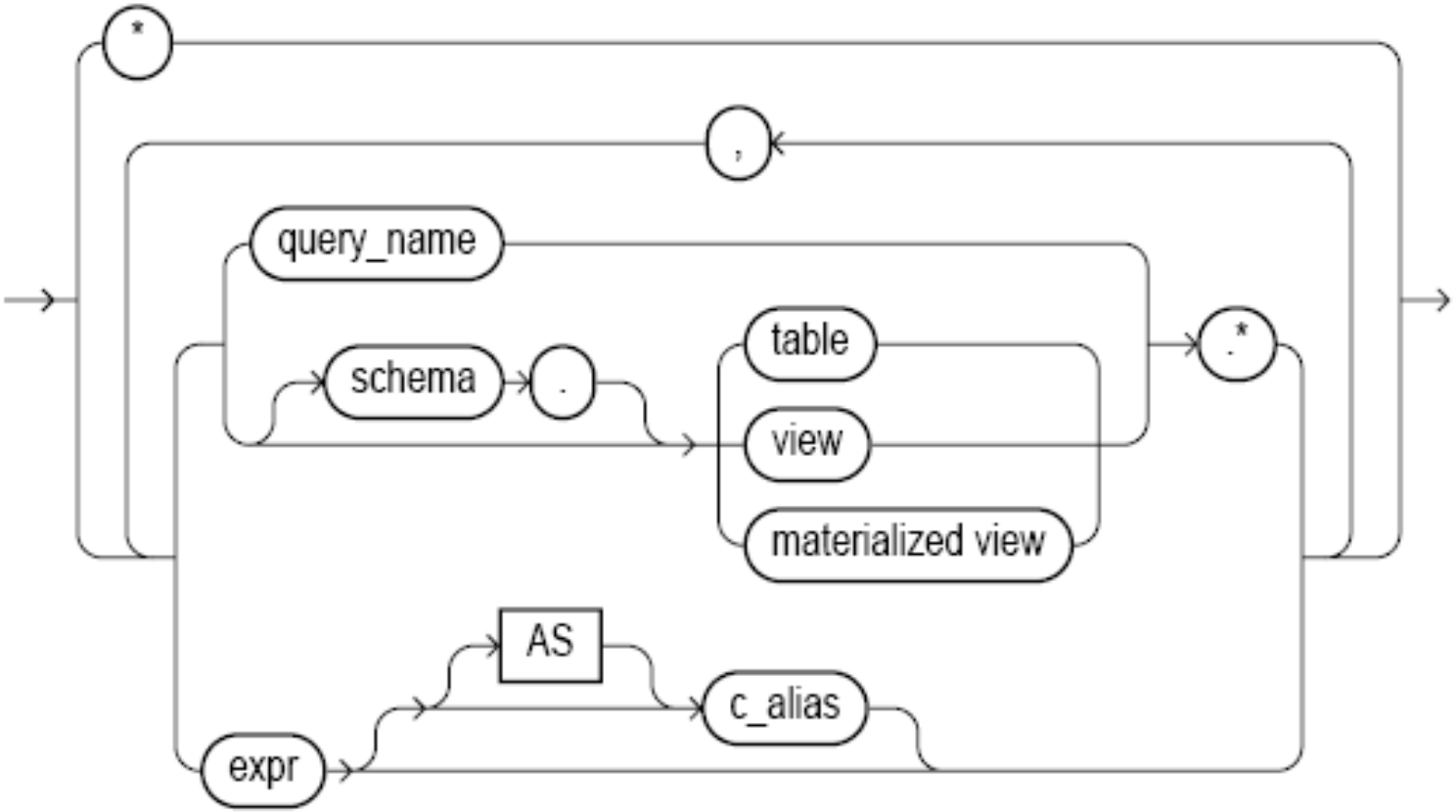


Команда SELECT (продолжение)

subquery_factoring_clause::=

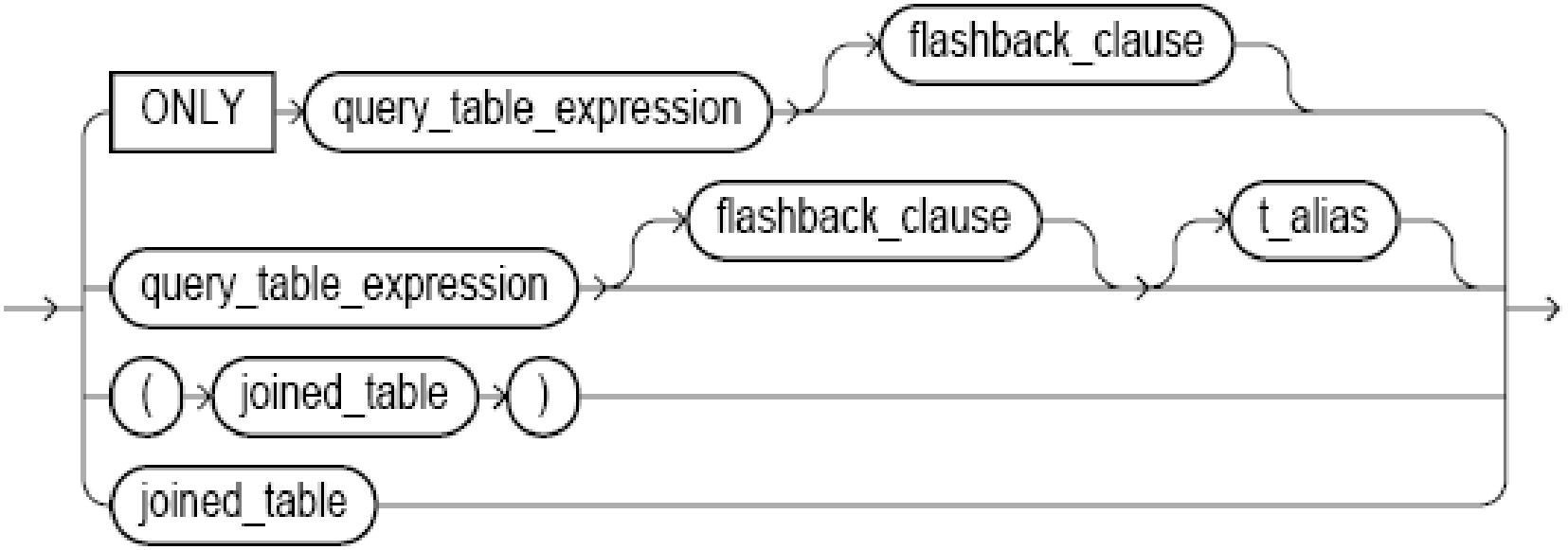


select_list::=



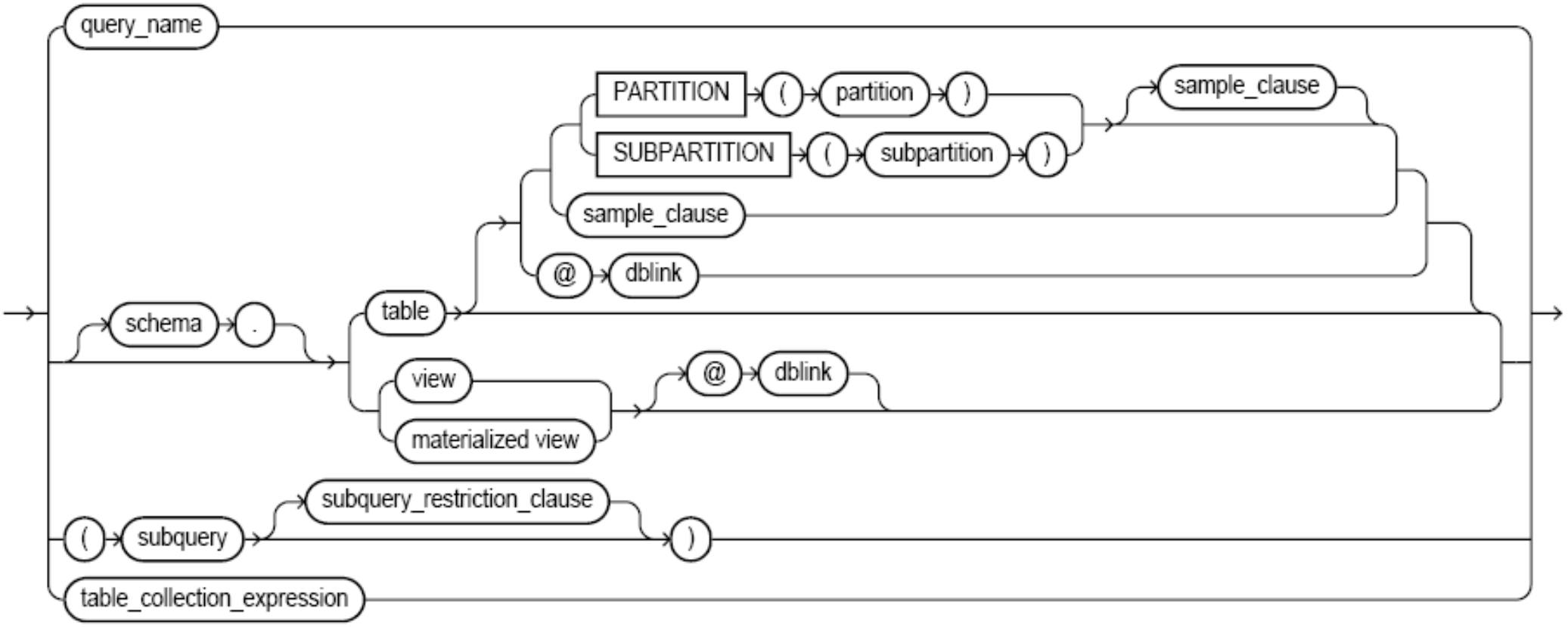
Команда SELECT (продолжение)

table_reference::=



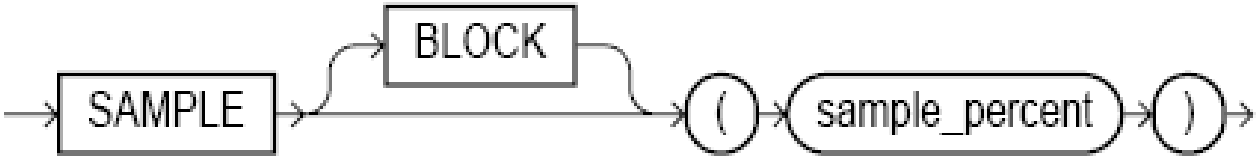
Команда SELECT (продолжение)

query_table_expression::=

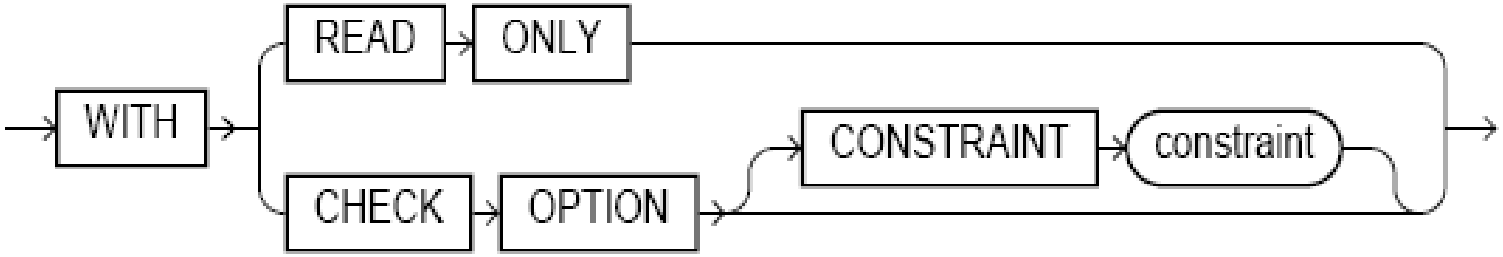


Команда SELECT (продолжение)

sample_clause::=

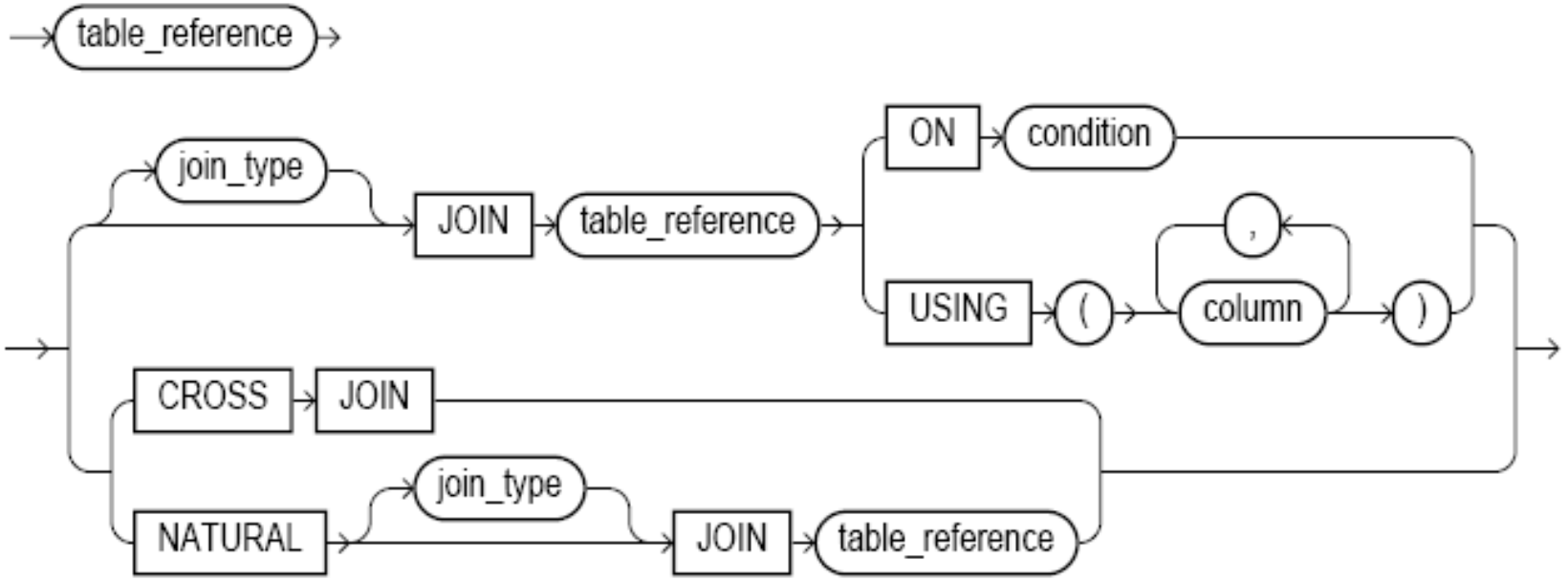


subquery_restriction_clause::=

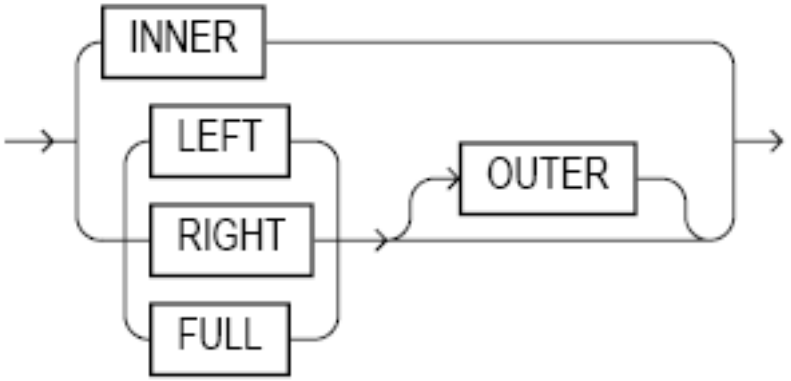


Команда SELECT (продолжение)

joined_table::=

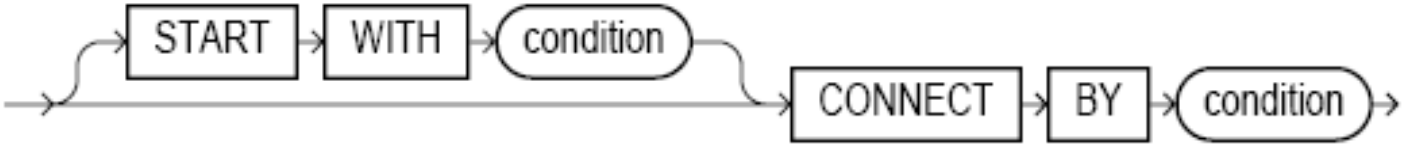


join_type::=

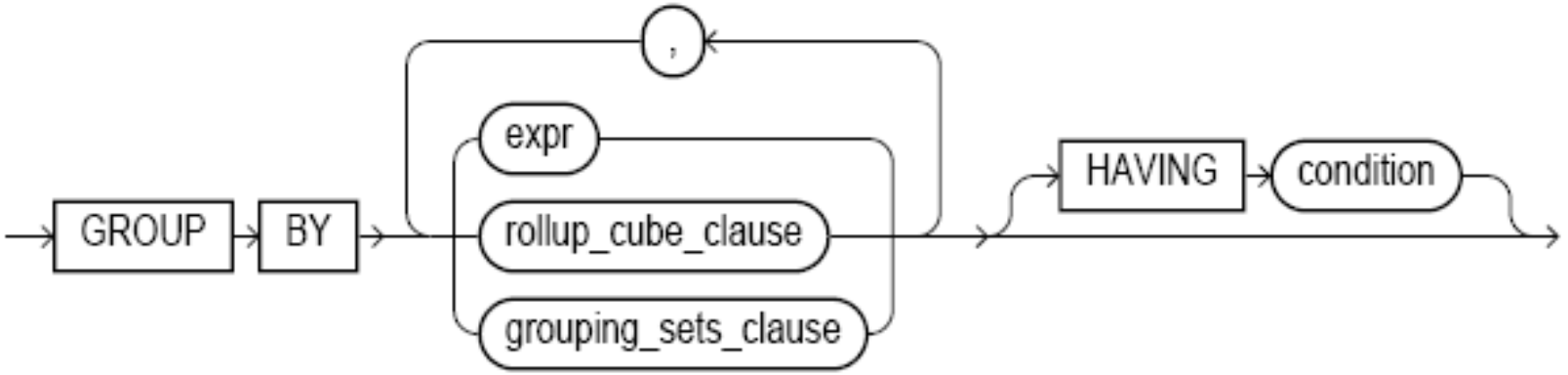


Команда SELECT (продолжение)

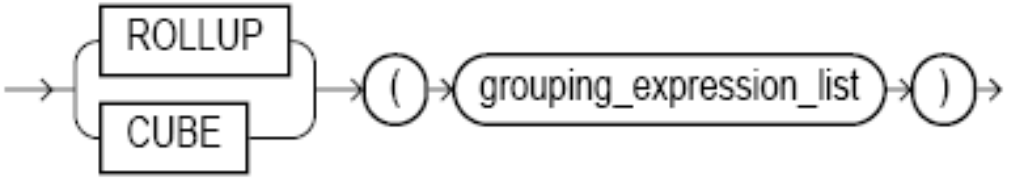
hierarchical_query_clause::=



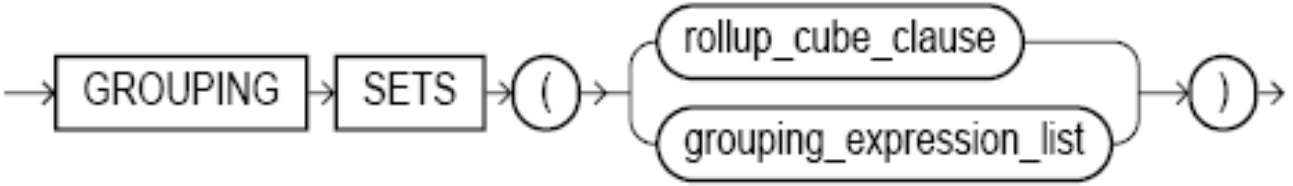
group_by_clause::=



rollup_cube_clause::=

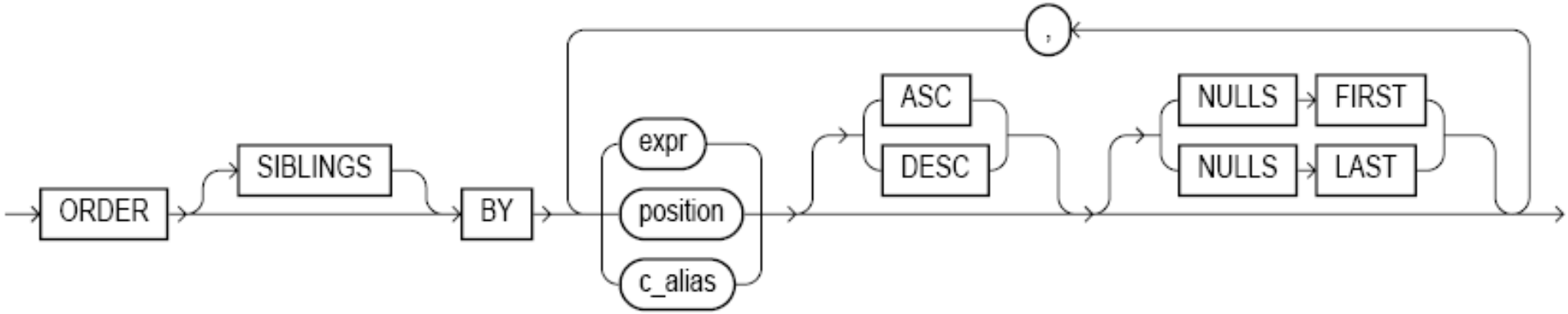


grouping_sets_clause::=

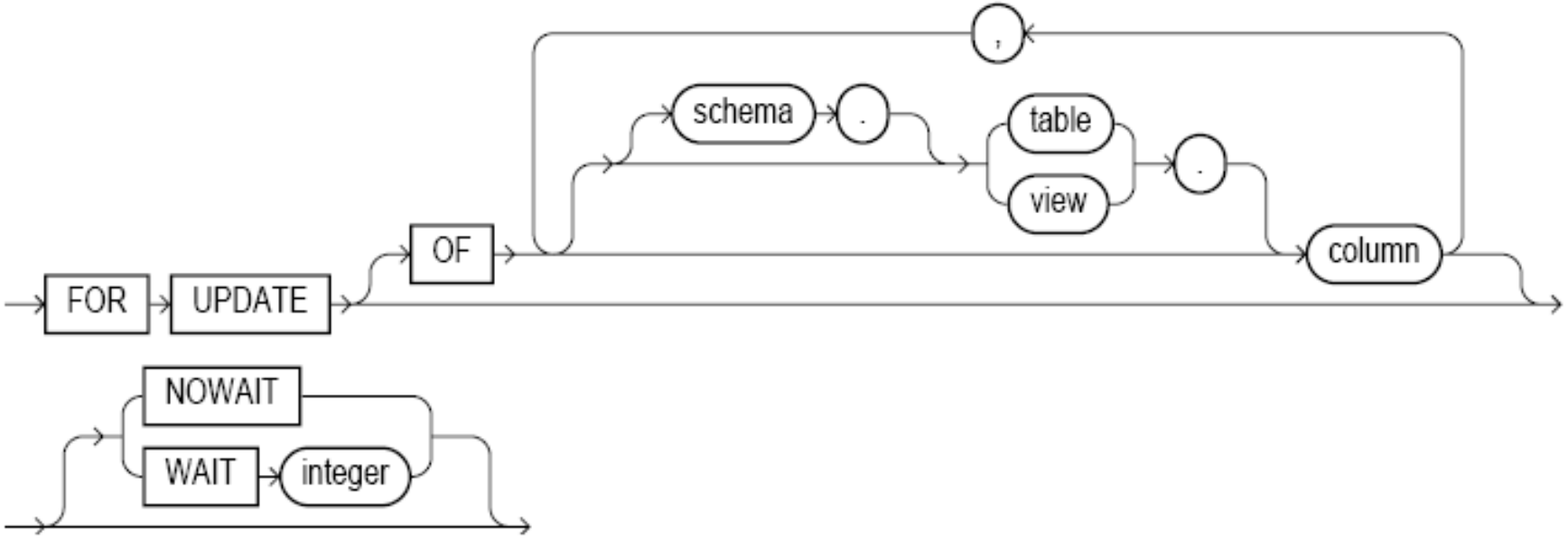


Команда SELECT (продолжение)

order_by_clause::=



for_update_clause::=



SQL. Примеры запросов.
Демонстрационная БД Oracle

```
CREATE TABLE dept
    (deptno NUMBER(2) CONSTRAINT pk_dept PRIMARY KEY,
    dname VARCHAR2(14),
    loc VARCHAR2(13) );

CREATE TABLE emp
    (empno NUMBER(4) CONSTRAINT pk_emp PRIMARY KEY,
    ename VARCHAR2(10),
    job VARCHAR2(9),
    mgr NUMBER(4) CONSTRAINT fk_empno REFERENCES
                                        emp (empno),
    hiredate DATE,
    sal NUMBER(7,2),
    comm NUMBER(7,2),
    deptno NUMBER(2) CONSTRAINT fk_deptno REFERENCES
                                        dept (deptno));
```

SQL. Примеры запросов.
Демонстрационная БД Oracle

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SQL. Примеры запросов

```
SELECT * FROM emp WHERE deptno = 30;

SELECT ename, job, sal, deptno FROM emp
      WHERE NOT (job = 'SALESMAN' AND deptno = 30);

SELECT        a.deptno "Department",
              a.num_emp * 100 / b.total_count "%Employees",
              a.sal_sum * 100 /b.total_sal "%Salary"
      FROM (SELECT deptno, COUNT(*) num_emp, SUM(sal) sal_sum
              FROM scott.emp GROUP BY deptno) a,
           (SELECT COUNT(*) total_count, SUM(sal) total_sal
              FROM scott.emp) b ;

SELECT COUNT(*) * 10 FROM emp SAMPLE BLOCK (10);
```

SQL. Примеры запросов

```
SELECT deptno, MIN(sal), MAX (sal) FROM emp GROUP BY deptno;
DEPTNO MIN(SAL) MAX(SAL)
10          1300        5000
20          800         3000
30          950         2850

SELECT deptno, MIN(sal), MAX (sal) FROM emp
                    WHERE job = 'CLERK' GROUP BY deptno;
DEPTNO MIN(SAL) MAX(SAL)
10          1300        1300
20          800         1100
30          950         950

SELECT deptno, MIN(sal), MAX (sal) FROM emp
                    WHERE job = 'CLERK' GROUP BY deptno HAVING MIN(sal) < 1000;
DEPTNO MIN(SAL) MAX(SAL)
20          800         1100
30          950         950
```

SQL. Примеры запросов

```
SELECT LPAD(' ', 2 * (LEVEL - 1)) || ename org_chart, empno, mgr, job
FROM emp START WITH job = 'PRESIDENT'
CONNECT BY PRIOR empno = mgr;
```

ORG_CHART	EMPNO	MGR	JOB
KING	7839		PRESIDENT
JONES	7566	7839	MANAGER
SCOTT	7788	7566	ANALYST
ADAMS	7876	7788	CLERK
FORD	7902	7566	ANALYST
SMITH	7369	7902	CLERK
BLAKE	7698	7839	MANAGER
ALLEN	7499	7698	SALESMAN
WARD	7521	7698	SALESMAN
MARTIN	7654	7698	SALESMAN
TURNER	7844	7698	SALESMAN
JAMES	7900	7698	CLERK
CLARK	7782	7839	MANAGER
MILLER	7934	7782	CLERK

SQL. Примеры запросов

```
SELECT DECODE(GROUPING(dname), 1, 'All Departments', dname) AS dname,
       DECODE(GROUPING(job), 1, 'All Jobs', job) AS job,
       COUNT(*) "Total Empl", AVG(sal) * 12 "Average Sal"
FROM emp JOIN dept USING (deptno) GROUP BY CUBE (dname, job);
```

DNAME	JOB	Total Empl	Average Sal
All Departments	All Jobs	14	24878,57143
All Departments	CLERK	4	12450
All Departments	ANALYST	2	36000
All Departments	MANAGER	3	33100
All Departments	SALESMAN	4	16800
All Departments	PRESIDENT	1	60000
SALES	All Jobs	6	18800
SALES	CLERK	1	11400
SALES	MANAGER	1	34200
SALES	SALESMAN	4	16800
RESEARCH	All Jobs	5	26100
RESEARCH	CLERK	2	11400
RESEARCH	ANALYST	2	36000
RESEARCH	MANAGER	1	35700
ACCOUNTING	All Jobs	3	35000
ACCOUNTING	CLERK	1	15600
ACCOUNTING	MANAGER	1	29400
ACCOUNTING	PRESIDENT	1	60000

SQL. Примеры запросов

```
SELECT ename, deptno
      FROM emp WHERE deptno =
        (SELECT deptno FROM emp WHERE ename = 'TURNER');

SELECT ename, job, deptno, dname
        FROM emp NATURAL JOIN dept
        WHERE job = 'CLERK';
```

ENAME	JOB	DEPTNO	DNAME
SMITH	CLERK	20	RESEARCH
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING

SQL. Примеры запросов

```
SELECT e1.ename||' works for '||e2.ename "Employees and their Managers"
        FROM emp e1 JOIN emp e2 ON e1.mgr = e2.empno;
```

```
Employees and their Managers
SMITH works for FORD
ALLEN works for BLAKE
WARD works for BLAKE
JONES works for KING
MARTIN works for BLAKE
BLAKE works for KING
CLARK works for KING
SCOTT works for JONES
TURNER works for BLAKE
ADAMS works for SCOTT
JAMES works for BLAKE
FORD works for JONES
MILLER works for CLARK
```

SQL. Примеры запросов

```
SELECT ename, job, dept.deptno, dname
      FROM emp RIGHT OUTER JOIN dept ON
      emp.deptno = dept.deptno AND job = 'CLERK';
```

ENAME	JOB	DEPTNO	DNAME
SMITH	CLERK	20	RESEARCH
ADAMS	CLERK	20	RESEARCH
JAMES	CLERK	30	SALES
MILLER	CLERK	10	ACCOUNTING
		40	OPERATIONS

```
SELECT SYSDATE FROM DUAL;

SELECT zseq.nextval FROM dual;
```

SQL. Примеры запросов

Получить фамилии хирургов

```
SELECT Фамилия FROM ВРАЧ
    WHERE Специальность = 'ХИРУРГ'
```

Получить фамилии врачей, лечащих больных палаты №2 больницы №5

```
SELECT В.Фамилия
    FROM (ВРАЧ В JOIN ВРАЧ-ПАЦИЕНТ ВП USING (К/В))
    JOIN РАЗМЕЩЕНИЕ Р USING (Р/Н)
    WHERE Р. К/Б = 5 AND Р. Н/П = 2
```

```
SELECT Фамилия FROM ВРАЧ WHERE К/В IN
    (SELECT К/В FROM ВРАЧ-ПАЦИЕНТ WHERE Р/Н IN
            (SELECT Р/Н FROM РАЗМЕЩЕНИЕ
                    WHERE К/Б = 5 AND Н/П =2))
```

SQL. Примеры запросов

Получить фамилии врачей, лечащих всех больных палаты №2 больницы №5

```
SELECT Фамилия FROM ВРАЧ WHERE К/В IN
   (SELECT К/В FROM ВРАЧ-ПАЦИЕНТ WHERE Р/Н IN
         (SELECT Р/Н FROM РАЗМЕЩЕНИЕ
         WHERE К/Б = 5 AND Н/П =2)
         GROUP BY К/В
         HAVING COUNT(*) = (SELECT COUNT(*) FROM
         РАЗМЕЩЕНИЕ WHERE К/Б = 5 AND Н/П =2))
```

Получить полный список специальностей врачей

```
SELECT DISTINCT Специальность FROM ВРАЧ
```

Получить названия больниц, которые имеют более 5 педиатров

```
SELECT Название FROM БОЛЬНИЦА WHERE К/Б IN
   (SELECT К/Б FROM ВРАЧ
   WHERE Специальность = ‘ПЕДИАТР’
   GROUP BY К/Б
   HAVING COUNT(*) > 5)
```

SQL. Примеры запросов

Получить фамилии врачей, не лечащих пациента с Р/Н = 111111

```
SELECT Фамилия FROM ВРАЧ В WHERE NOT EXISTS
   (SELECT К/В FROM ВРАЧ-ПАЦИЕНТ WHERE
   К/В = В.К/В AND Р/Н = 111111)
```

Получить фамилии врачей, лечащих всех пациентов

```
SELECT Фамилия FROM ВРАЧ WHERE К/В IN
   (SELECT К/В FROM ВРАЧ-ПАЦИЕНТ
   GROUP BY К/В
   HAVING COUNT(*) =
            (SELECT COUNT(*) FROM ПАЦИЕНТ))
```

Получить фамилии врачей, лечащих по крайней мере одного пациента врача с кодом 999

```
SELECT Фамилия FROM ВРАЧ WHERE К/В IN
   (SELECT К/В FROM ВРАЧ-ПАЦИЕНТ  WHERE Р/Н IN
            (SELECT Р/Н FROM ВРАЧ-ПАЦИЕНТ
                     WHERE К/В =999))
```

SQL. Примеры запросов

Получить фамилии врачей, лечащих всех пациентов врача с кодом 999 и может быть еще кого-то

```
SELECT Фамилия FROM ВРАЧ WHERE К/В IN
        (SELECT К/В
        FROM ВРАЧ-ПАЦИЕНТ ВП
        LEFT OUTER JOIN ВРАЧ-ПАЦИЕНТ ВП999
        ON ВП.Р/Н = ВП999.Р/Н AND ВП999.К/В = 999
          GROUP BY ВП.К/В
          HAVING COUNT (DISTINCT ВП999.Р/Н) =
            (SELECT COUNT(*) FROM ВРАЧ-ПАЦИЕНТ  WHERE К/В = 999))
```

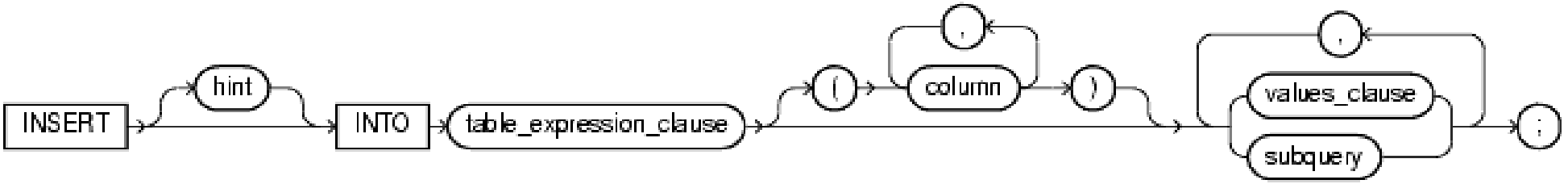
Получить фамилии врачей, лечащих всех пациентов врача с кодом 999 и обязательно еще кого-то

```
HAVING COUNT (DISTINCT ВП999.Р/Н) =
            (SELECT COUNT(*) FROM ВРАЧ-ПАЦИЕНТ WHERE К/В = 999)
          AND COUNT (*) >
            (SELECT COUNT(*) FROM ВРАЧ-ПАЦИЕНТ WHERE К/В = 999)
```

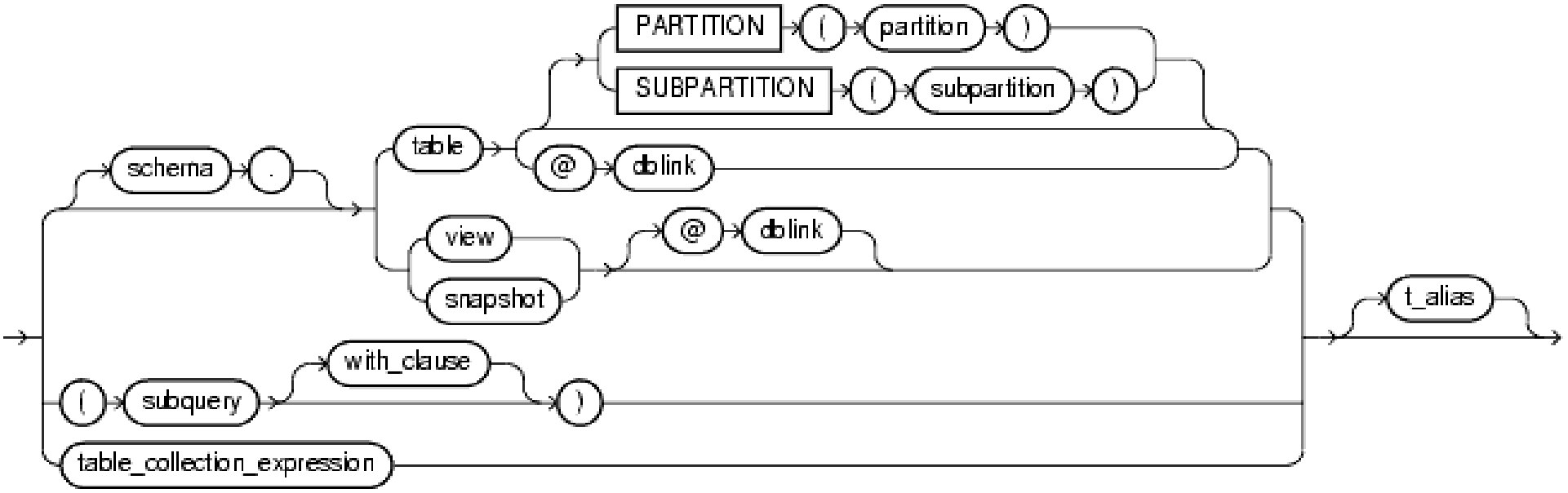
Получить фамилии врачей, лечащих не всех пациентов врача с кодом 999

```
HAVING COUNT (DISTINCT ВП999.Р/Н) <>
            (SELECT COUNT(*) FROM ВРАЧ-ПАЦИЕНТ WHERE К/В = 999)
```

Команда INSERT (начало)

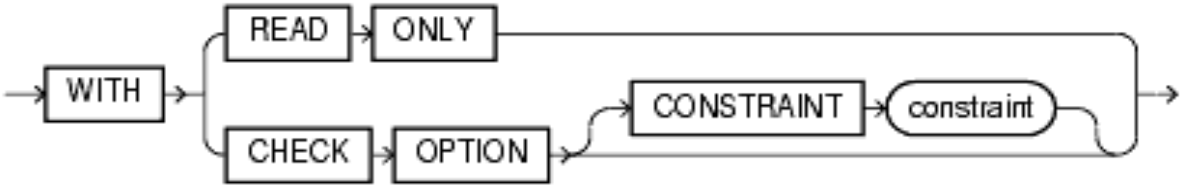


table_expression_clause::=

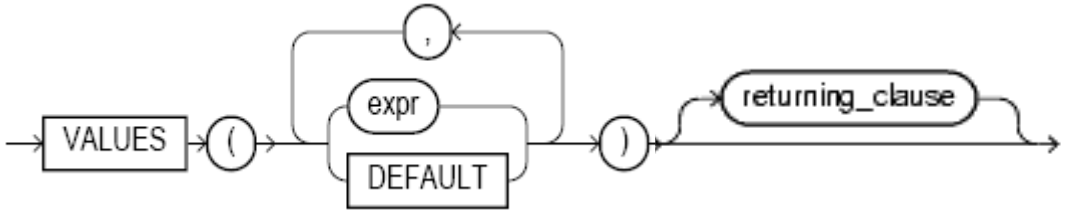


Команда INSERT (продолжение)

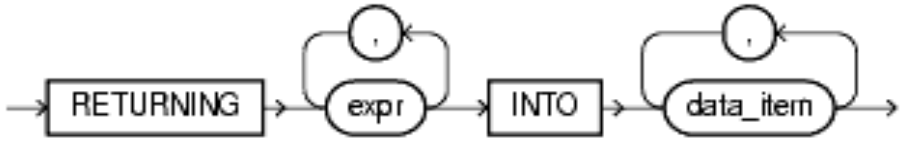
with_clause::=



values_clause::=



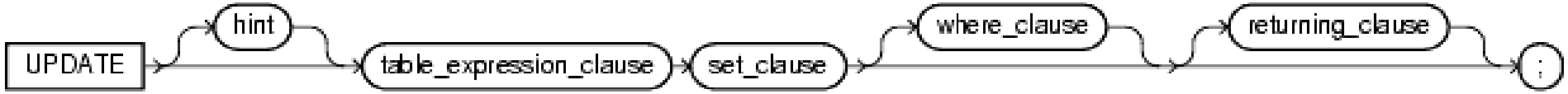
returning_clause::=



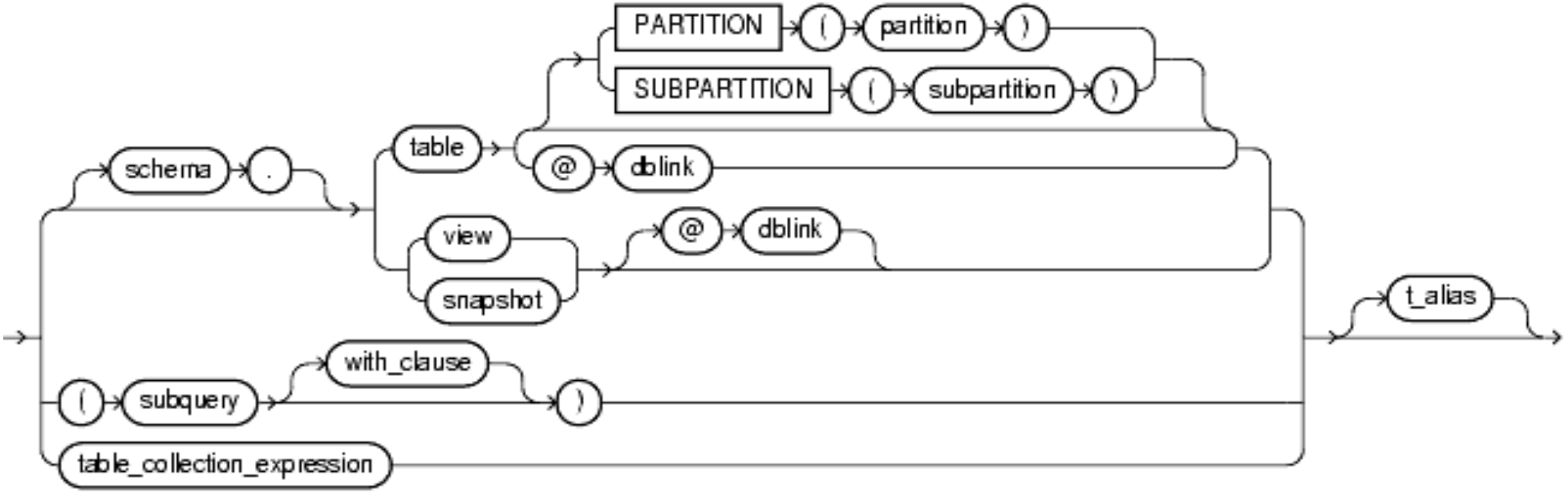
SQL. Примеры запросов. INSERT

```
INSERT INTO dept
    VALUES (50, 'PRODUCTION', 'SAN FRANCISCO');
INSERT INTO emp (empno, ename, job, sal, comm, deptno)  VALUES (7890,
    'JINKS', 'CLERK', 1.2E3, NULL, 40);
INSERT INTO (SELECT empno, ename, job, sal, comm, deptno FROM emp)
    VALUES (7890, 'JINKS', 'CLERK', 1.2E3, NULL, 40);
INSERT INTO bonus
    SELECT ename, job, sal, comm  FROM emp  WHERE comm > 0.25 *
    sal  OR job IN ('PRESIDENT', 'MANAGER');
INSERT INTO emp  VALUES  (empseq.nextval, 'LEWIS', 'CLERK', 7903,
    SYSDATE, 1200, NULL, 20);
INSERT INTO emp VALUES (empseq.nextval, 'LEWIS', 'CLERK', 7903,
    SYSDATE, 1200, NULL, 20)  RETURNING sal*12, job INTO :bnd1,
    :bnd2;
INSERT INTO (SELECT empno, ename, deptno FROM emp
    WHERE deptno < 10)  VALUES (8903, 'Taylor', 20);
INSERT INTO (SELECT empno, ename, deptno FROM emp
    WHERE deptno < 10 WITH CHECK OPTION) VALUES (8903, 'Taylor', 20);
```

Команда UPDATE (начало)

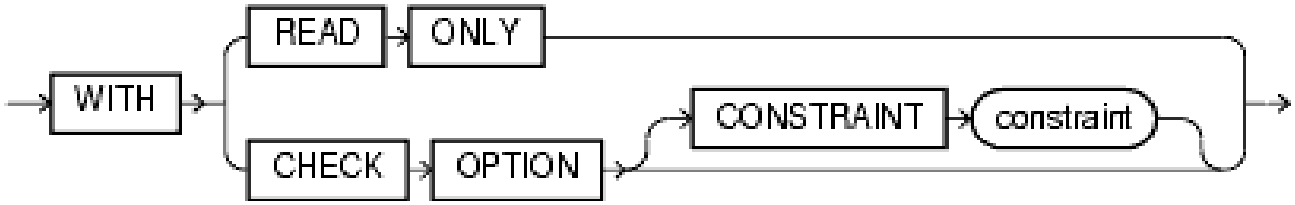


table_expression_clause::=

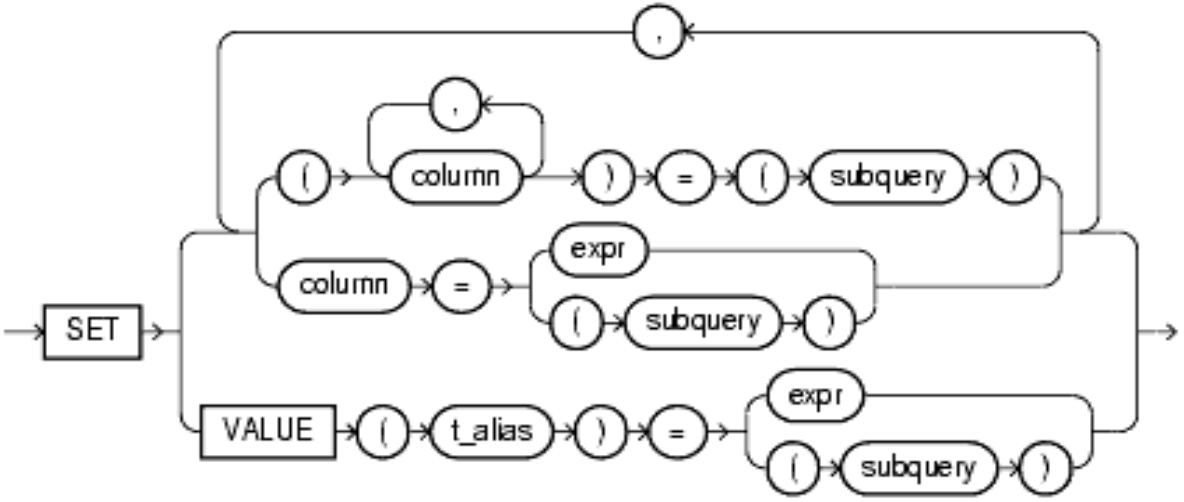


Команда UPDATE (продолжение)

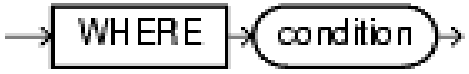
with_clause::=



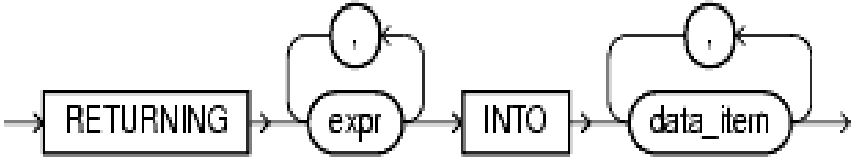
set_clause::=



where_clause::=



returning_clause::=

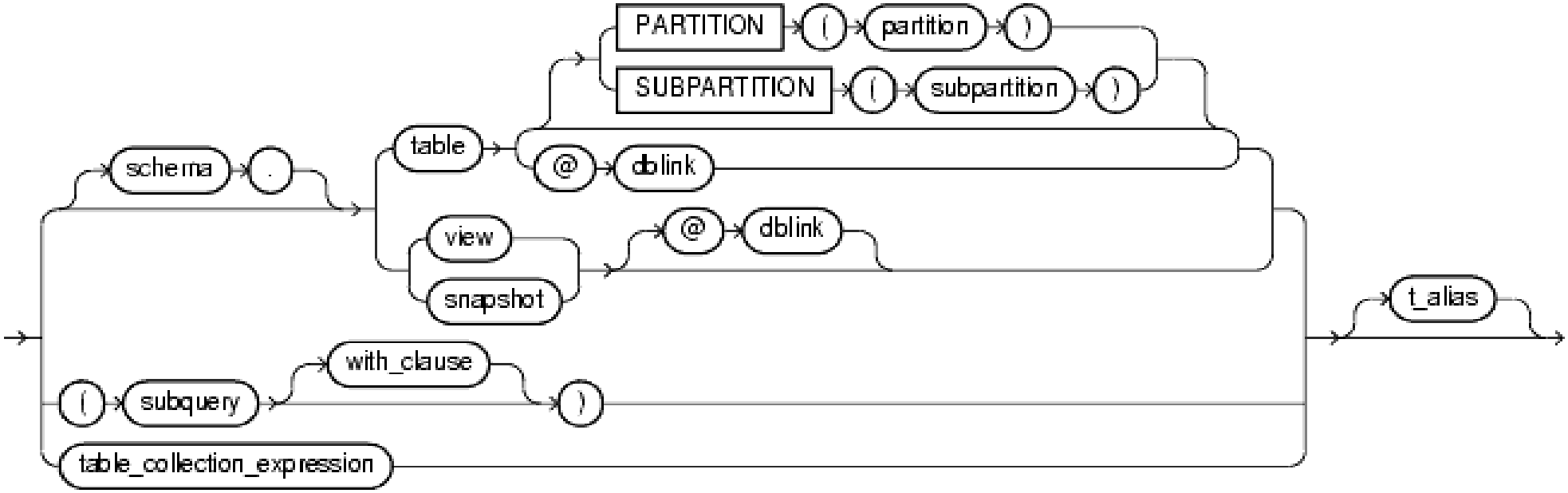


SQL. Примеры запросов. UPDATE

```
UPDATE emp SET comm = NULL WHERE job = 'TRAINEE';
UPDATE emp SET job = 'MANAGER', sal = sal + 1000, deptno =
    20 WHERE ename = 'JONES';
UPDATE emp a
    SET deptno = (SELECT deptno FROM dept WHERE
                  loc = 'BOSTON'),
    (sal, comm) = (SELECT 1.1*AVG(sal), 1.5*AVG(comm)
                   FROM emp b WHERE a.deptno = b.deptno)
    WHERE deptno IN (SELECT deptno FROM dept
      WHERE loc = 'DALLAS' OR loc = 'DETROIT');
UPDATE emp  SET job ='MANAGER', sal = sal + 1000, deptno =
    20  WHERE ename = 'JONES‘
    RETURNING sal*0.25, ename, deptno INTO :bnd1, :bnd2,
    :bnd3;
```

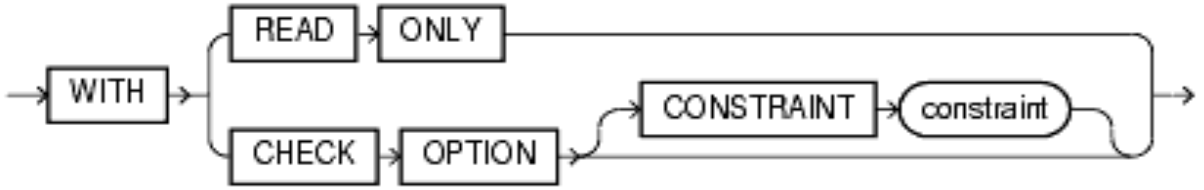
Команда DELETE (начало)

table_expression_clause::=

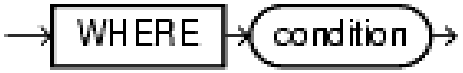


Команда DELETE (продолжение)

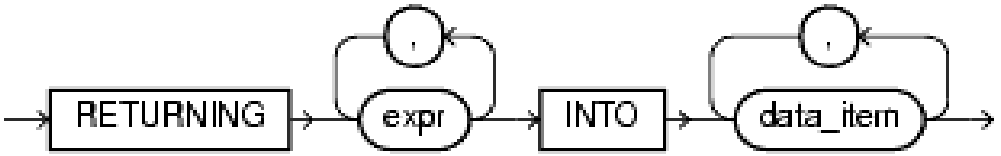
with_clause::=



where_clause::=



returning_clause::=



SQL. Примеры запросов. DELETE

```
DELETE FROM temp_assign;
DELETE FROM emp
      WHERE JOB = 'SALESMAN'    AND COMM < 100;
DELETE FROM (select * from emp)
      WHERE JOB = 'SALESMAN'    AND COMM < 100;
DELETE FROM emp
      WHERE ename = 'JONES' RETURNING sal INTO :bnd1;
```