

Отчет по лабораторной работе № 4 «Операторы DML»			
дата	Оценка (max 5)	Бонус за сложность	подпись

Цели работы:

Изучение операторов DML

Задачи работы:

-разработка примеров простейших программ с использованием DML для схемы HR

Задание повышенной сложности (бонус за сложность – 10 баллов):

- выполнение задание 5.8.1., 5.8.2, 5.8.3.

Краткий конспект теоретической части (ответы на контрольные вопросы)

SELECT _____

INSERT _____

UPDATE _____

DELETE_____

Язык DML

Язык обработки данных DML позволяет добавлять, изменять, удалять информацию в базе данных. В полном соответствии со своим названием, язык DML предназначен для работы с информационным содержанием базы данных. Операторы языка DML представляют собой SQL-команды, позволяющие изменять и дополнять хранящуюся в базе данных информацию. Наиболее используемыми DML-операторами являются INSERT, UPDATE, DELETE и SELECT

Стандартизация SQL

Процедура разработки международных стандартов в соответствии с регламентом деятельности ISO предусматривает несколько стадий, каждая из которых заключается в подготовке очередной версии документа или комплекта документов, содержащих спецификации проекта стандарта. После подготовки некоторой версии проекта по ней в течение установленного срока проводится голосование (рассмотрение) среди определенного круга экспертов, высказывающих свои замечания. Для принятия решений по этим замечаниям и внесения необходимых поправок в текст проекта созываются редакционные заседания подразделения ISO, ответственного за разработку стандарта.

Серия документов, рождающихся в процессе разработки спецификаций стандарта, включает (в порядке их создания):

Рабочий проект (Working Draft, WD)

Проект комитета (Committee Draft, CD)

Заключительный проект комитета (Final Committee Draft, FCD)

Проект международного стандарта (Draft International Standard, DIS)

Заключительный проект международного стандарта (Final Draft International Standard, FDIS)

Международный стандарт (International Standard, IS).

Часть стандарта	Сроки принятия
Часть 10, SQL/OLB	Середина 2000 (Связывание для объектных языков)
Поправка 1, SQL/OLAP	Конец 2000 (Интерактивная аналитическая обработка)
Часть 9, SQL/MED	2001 (Управление внешними данными)
Часть 7, SQL/Temporal	2003?
SQL:200x	2003?

Часть 10, SQL/OLB – Голосование по **Заключительному проекту комитета** завершилось в начале 1999 г. с рядом существенных замечаний. В январе 2000 года в Санта-Фе (штат Нью-Мексико, США) были успешно разрешены вопросы, касающиеся последних замечаний, и участники рекомендовали перейти к стадии голосования по **Заключительному проекту международного стандарта** (Final Draft International Standard).

В результате опубликована новая часть стандарта SQL:1999, идентифицируемая как ISO/IEC 9075-10:2000.

Новые возможности SQL:1999:

- поддержка позиционируемых (scrollable) и удерживаемых (holdable) курсоров;
- поддержка типов, определяемых пользователем;
- поддержка обновляемых результирующих наборов и пакетных обновлений;
- расширенные средства подключения (connection) и транзакций.

Часть 9, SQL/MED- В конце 1998 года в ISO было инициировано голосование по **проекту комитета**. Голосования по **Заключительному проекту международного стандарта**-середина 2001 года.

SQL/MED опубликован как международный стандарт ISO/IEC 9075 9:2001 в 2001 г.

Новые возможности SQL:1999:

обеспечивает связь между SQL-сервером (т.е. некоторой «локальной» SQL системой управления базами данных) и другим объектом, называемым адаптером внешних данных (foreign-data wrapper). Локальный SQL-сервер и этот адаптер обмениваются информацией, которая позволяет локальному SQL-серверу осуществлять выборку и, возможно, вставку, обновление и удаление данных, которые фактически управляются некоторым внешним сервером (foreign server).

Поправка 1, SQL/OLAP- голосование по предложенному **Заключительному проекту поправки** (Final Proposed Draft Amendment, FPDAM) к SQL/OLAP было инициировано в начале 2000 г. Результат: ISO/IEC 9075-1/AMD1:2000. Этот документ вносит поправки в несколько частей SQL:1999, и поэтому было бы вполне уместно рассматривать его как поправку к части 1, SQL/Framework.

Новые возможности SQL:1999:

- SQL/OLAP содержит ряд новых числовых функций, например:
- LN ;
- EXP;
- POWER (возводит один аргумент в степень, указываемую другим аргументом);
- SQRT (извлекает квадратный корень из аргумента); FLOOR и CEILING (возвращает наибольшее целое, не превышающее значения аргумента, и наименьшее целое, большее или равное значению аргумента);
- функция ранжирования (возвращает ранжирование аргумента на мультимножестве значений);
- функция процентиля (возвращает ранжирование аргумента как значения его процентилей на мультимножестве значений).

Часть 7, SQL/Temporal -Работа над SQL/Temporal приостановилась уже три года назад по трем причинам. Во-первых, не проявляли большого энтузиазма поставщики продуктов SQL – или, в действительности, их заказчики – в вопросе о включении средств поддержки временных данных в механизмы систем управления базами данных. Во-вторых, специалисты, занимающиеся стандартами SQL были вынуждены сконцентрироваться на завершении проекта SQL3. Наконец, имеется фундаментальное расхождение между двумя лагерями относительно того, какие в точности возможности должен предоставлять SQL/Temporal и каким образом эти возможности следует обеспечивать (т.е., и синтаксис и семантика).

Когда разработка SQL3 завершена, участники работы выяснили, что рынок проявляет все более возрастающую информированность относительно преимуществ поддержки временных данных и, по крайней мере, некоторые из поставщиков продуктов SQL стали интересоваться этой проблемой. Поэтому работы над SQL/Temporal недавно начали возрождаться, и можно надеяться увидеть публикацию этой дополнительной части стандарта SQL, вероятно, в течение 2002-2003 гг.



Рис.1.

Итак: **SQL:**

- язык доступа к реляционным базам данных, в том числе и к РБД Oracle;
- может быть использован в любом средстве создания приложений Oracle, где требуется доступ к БД.

PL/SQL:

- процедурный язык для написания подпрограмм и манипулирования данными вне БД;
- может включать ряд команд языка SQL при необходимости обращения к БД;
- не имеет самостоятельных операторов ввода/вывода, но доступен в любом продукте CDE и прекомпиляторах Pro*Oracle для написания подпрограмм. Также доступен на самом сервере Oracle (при наличии процедурной опции). "SQL-PLUS
- продукт Oracle, в котором могут использоваться языки SQL и PL/SQL. Представляет собой среду для задания и выполнения команд языка SQL, а также процедур на языке PL/SQL;
- имеет свой собственный набор команд для настройки программной среды и форматирования выходных результатов.

Таким образом, SQL и PL/SQL - это языки, используемые в продуктах Oracle.

Прежде чем перейти к рассмотрению основных синтаксических конструкций языка (т.е. описание алфавита и правил построения различных конструкций языка из символов алфавита и более простых конструкций языка) введем понятие синтаксической диаграммы.

Синтаксическая диаграмма – это графическое представление правил построения конструкций языка в наглядной форме. При этом принято, символы алфавита изображать блоками в овальных рамках, названия конструкций – в прямоугольных, а правила построения конструкций в виде линий со стрелками (часто овальные и прямоугольные рамки для упрощения опускают). Также выделяют осевую линию синтаксической конструкции, на которой размещаются все обязательные символы алфавита и конструктивные элементы. Все необязательные элементы и символы размещаются ниже или выше осевой линии конструкции. Если конструкция имеет значительную длину, то осевая линия переносится соответственно на следующую строку. Существуют и альтернативные формы отображения синтаксиса языка, например форма Бэкуса-Наура (БНФ) и т.п. [2].

Задание SQL-команд

При написании SQL-команд важно помнить ряд простых правил, позволяющих конструировать корректные, легко читаемые и редактируемые предложения:

- SQL-команды могут быть на одной или нескольких строках.
- Отдельные компоненты команды (предложения) обычно задаются на разных строках.
- Допускается использование табуляторов.
- Отдельные слова в команде не могут быть разделены на несколько строк с переносом.
- SQL-команды могут задаваться как прописными, так и строчными буквами, если не задано специальных ограничений.
- SQL-команда вводится в ответ на системную подсказку (по умолчанию это: SQL), и строки команды нумеруются.

Каждая из нижеследующих команд верна, но читаемость их различна:

SELECT * FROM emp;	SELECT * FROM EMP;	SELECT * FROM emp;
--------------------	--------------------------	-----------------------

Типы данных обрабатываемых СУБД Oracle представлены в таблице.

Таблица. Типы обрабатываемых данных.

Тип данных	Описание
CHAR(size)	Символьная строка фиксированной длины, имеющая максимальную длину <i>size</i> символов. Длина по умолчанию 1, максимальная -255.
CHARACTER (size)	То же, что и CHAR.
DATE	Правильные даты в интервале от 1 января 4712 года до н.э. до 31 декабря 4712 года.
LONG	Символьные данные переменной длины до 2 Гигабайт.
LONG RAW	Двоичные данные переменной длины вплоть до 2 Гигабайт.
CLOB	Character Long
BLOB	Binary Long
MLSLABEL	Используется в Trusted ORACLE.
NUMBER(p,s)	Число, имеющее <i>p</i> значащих цифр и масштаб <i>s</i> . <i>p</i> может быть от 1 до 38. <i>s</i> может принимать значения от -84 до 127.
RAW(size)	Двоичные данные длиной <i>size</i> байт. Максимальное значение для <i>size</i> - 2000 байт. Параметр <i>me</i> для RAW обязателен.
RAW MLSLABEL	Используется в Trusted ORACLE.
ROWID	Значения псевдостолбца ROWID.
VARCHAR2(size)	Символьная строка переменной длины, имеющая максимальную длину <i>size</i> символов. Длина по умолчанию 1, максимальная - 2000.
VARCHAR(size)	То же что и VARCHAR2.



Задание 5.1.: Выведите структуру таблиц схемы HR в SQL+ с указанием типов данных атрибутов и наложенных ограничений

SQL> desc hr.employees


Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Работа с псевдосто́лбцами

Извлекать данные можно также и из псевдосто́лбцов, которые похожи на столбцы таблиц, но их значения нельзя изменять при помощи операторов DML.

Таблица. Псевдосто́лбцы.

Название столбца	Возвращаемое значение
Sequence.CURRVAL	Текущее значение sequence в данном сеансе (sequence.NEXTVAL должен быть выбран).
Sequence.NEXTVAL	Следующее значение sequence в текущем сеансе.
[table.]LEVEL	1 - для корня дерева, 2 - для узлов второго уровня и так далее. Используется в операторе SELECT в иерархических запросах.
[table.]ROWID	Значение, которое идентифицируют строку в таблице table уникальным образом. Значения псевдосто́лбца ROWID имеют тип данных ROWID, а не NUMBER и не CHAR.
ROWNUM	Порядковый номер строки среди других строк, выбираемых запросом. ORACLE выбирает строки в произвольном порядке и приписывает значения ROWNUM, прежде чем строки будут отсортированы предложением ORDER BY.



Задание 5.2.: Выполните обращение к псевдосто́лбцам таблиц в схеме HR

```

1 CREATE SEQUENCE serial;
2 SELECT serial.CURRVAL
3   FROM dual;
4 SELECT serial.NEXTVAL
5   FROM dual;
6 SELECT level
7   FROM hr.employees
8 CONNECT BY NOCYCLE
9    PRIOR hr.employees.employee_id = hr.employees.manager_id;
10 SELECT hr.employees.rowid
11   FROM hr.employees;
12 SELECT ROWNUM
13   FROM hr.employees;
```

результат

CURRVAL	NEXTVAL
24	25
LEVEL	ROWID
3	AAARz yAAAAAAN8ABi
3	AAARz yAAAAAAN1AAA
3	AAARz yAAAAAAN1AAB
3	AAAA~ ~AAAAAAN1AAC
3	ROWNUM ,AAAAA1AAD
	43
	44
	45
	46
	47
	48

Операции и их приоритеты

Арифметические операции	Символьные операции	Логические операции	Операции сравнения
+ - (один операнд)		NOT	=
* /		AND	!= ^= ~= <>
+ - (два операнда)		OR	> >= < <=
			IN
			NOT IN
			ANY, SOME
			ALL

Приоритеты операций. При вычислении выражения, содержащего несколько операций, ORACLE сначала выполняет операции с более высоким приоритетом. Операции, приведенные на одной и той же строке, имеют одинаковые приоритеты.

Замечание: В выражениях можно использовать круглые скобки, чтобы изменять последовательность выполнения операций, предписываемую приоритетом. Выражения, заключенные в скобки, ORACLE вычисляет в первую очередь. Без скобок операции с одинаковым приоритетом ORACLE выполняет слева направо.



Задание 5.3:

Выбрать сведения о сотрудниках, имеющих оклады, начиная с 2-ого ранга по сумме оклада и кончая 5-ым. (Ранг 1 имеет сотрудник или сотрудники с наибольшим окладом, то есть находящиеся на первом месте по величине оклада) В задании используется таблица HR.EMPLOYEES

Выводимые столбцы:

1. Имя (First_name)
2. Фамилия (Last_name)
3. Название отдела
4. Должность (Job_id)
5. Оклад (Salary)
6. Позиция по окладу (ранг)

Строки в выходной таблице должны быть упорядочены по позиции (рангу) оклада и по названию отдела и должности в возрастающем порядке.

Дополнительные требования к выполнению:

Не использовать аналитические функции. Для ранжирования использовать псевдостолбец ROWNUM.

```
1  SELECT emp.first_name "Имя"
2      ,emp.last_name "Фамилия"
3      ,dept.department_name "Отдел"
4      ,emp.salary "Оклад"
5      ,emp.job_id
6      ,d_s.position "Позиция"
7  FROM hr.employees emp
8      ,hr.departments dept
9      ,(
10     SELECT ROWNUM position
11           ,sal
12     FROM (
13         SELECT DISTINCT salary sal
14         FROM hr.employees
15         ORDER BY sal DESC
16     )
17 ) d_s
18 WHERE d_s.sal = emp.salary
19     AND dept.department_id = emp.department_id
20     AND d_s.position BETWEEN 2 AND 5
21 ORDER BY d_s.position
22           ,dept.department_name
23           ,job id;
```

	Имя	Фамилия	Отдел	Оклад	JOB_ID	Позиция
1	Neena	Yang	Executive	17000	AD_VP	2
2	Lex	Garcia	Executive	17000	AD_VP	2
3	John	Singh	Sales	14000	SA_MAN	3
4	Karen	Partners	Sales	13500	SA_MAN	4
5	Michael	Martinez	Marketing	13000	MK_MAN	5


Результат:

строк 5

Список, зарезервированных слов SQL

Язык SQL включает зарезервированные слова, имеющие определенное значение в операторах SQL. Эти слова нельзя использовать в качестве имен объектов базы данных.

ACCESS*	DEFAULT*	INTEGER	OPTION*	START*
ADD*	DELETE*	INTERSECT*	OR*	SUCCESSFUL
ALL*	DESC*	INTO*	ORDER*	SYNONYM
ALTER*	DISTINCT*	IS*	PCTFREE*	SYSDATE
AND*	DROP*	LEVEL*	PRIOR*	TABLE*
ANY*	ELSE*	LIKE*	PRIVILEGES	THEN*
AS*	EXCLUSIVE	LOCK	PUBLIC*	TO*
ASC*	EXISTS*	LONG	RAW	TRIGGER
AUDIT	FILE	MAXEXTENTS	RENAME*	UID
BETWEEN*	FLOAT	MINUS*	RESOURCE*	UNION*
BY*	FOR*	MODE	REVOKE	UNIQUE*
CHAR*	FROM*	MODIFY	ROW	UPDATE*
CHECK*	GRANT*	NOAUDIT	ROWID	USER
0CLUSTER*	GROUP*	NOCOMPRESS*	ROWLABEL	VALIDATE
COLUMN	HAVING*	NOT*	ROWNUM*	VALUES*
COMMENT	IDENTIFIED*	NOWAIT	ROWS	VARCHAR*
COMPRESS*	IMMEDIATE	NULL*	SELECT*	VARCHAR2*
CONNECT*	IN*	NUMBER*	SESSION	VIEW*
CREATE*	INCREMENT	OF*	SET*	WHENEVER
CURRENT*	INDEX*	OFFLINE	SHARE	WHERE*
DATE*	INITIAL	ON*	SIZE*	WITH*
DECIMAL*	INSERT*	ONLINE	SMALLINT	

 <p>Задание 5.4.: Попробуйте создать публичный синоним (или иной объект БД) с именем соответствующим зарезервированному слову</p>	<p>Результат и его анализ</p>
<p>Error starting at line : 1 in command - CREATE SEQUENCE level Error report - ORA-02277: неверное имя последовательности https://docs.oracle.com/error-help/db/ora-02277/02277.00000 - "invalid sequence name" *Cause: The specified sequence name is not a valid identifier name. *Action: Specify a valid identifier name for the sequence name.</p>	

Комментарии

Комментарии, заданные ограничителями '/' и '*/', могут стоять в любом месте оператора SQL:

```
ALTER USER petrov /* Это комментарий */ IDENTIFIED BY petr;
```

Можно использовать стандартные комментарии ANSI. Все символы после двух дефисов до конца строки игнорируются.

ALTER USER petrov /* Это комментарий продолжен до конца строки IDENTIFIED BY petr;

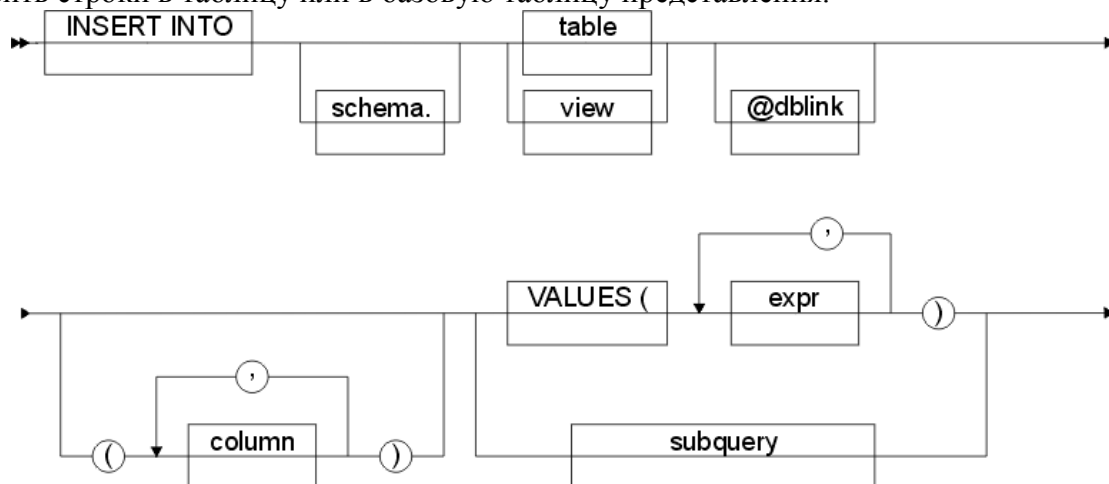
Оператор INSERT - Занесение/изменение данных в таблицах/

Команда INSERT используется для ввода новых строк в таблицу. Синтаксис команды INSERT:

INSERT INTO имя таблицы (столбец, столбец, ...) VALUES (значение, значение, ...);

Допускается пропуск списка столбцов после имени таблицы, если список значений соответствует количеству и порядку столбцов в описании таблицы. Рекомендуется всегда указывать список столбцов. Если список не задан, то при изменении структуры таблицы необходимо будет исправлять и программное обеспечение. Символьные значения и даты должны быть взяты в одинарные кавычки.

Вставить строки в таблицу или в базовую таблицу представления.



Условные обозначения на синтаксических диаграммах:

INSERT INTO	Наименование команды
HEMA	Используемая схема
TABLE (VIEW)	Таблица или представление
COLUMNS	Наименование столбца
VALUES	Значение ячейки
alias	Псевдоним (базы данных, таблицы, столбца)
Column_element	Определение столбца таблицы
Conditions	Условное выражение, которое после вычисления может принимать значение TRUE, FALSE или UNKNOWN
Constraint	Имя, идентифицирующее правило целостности и хранящееся в словаре данных
@dblink	Канал связи с БД. (Полный путь к базе данных)
expr	Любое выражение или описание
subquery	Запрос в каком либо другом операторе

Ввод строк из других таблиц

Эта форма команды INSERT позволяет заносить в таблицу набор строк, найденных по запросу команды SELECT. Список столбцов в команде SELECT должен соответствовать списку столбцов предложения INTO.

ПРИМЕР: Копирования сведений о служащих 10-го отдела в таблицу D10 HISTORY.

```
insert into D10_HISTORY (EMPNO, ENAME, SAL, JOB, HIREDATE)
select EMPNO, ENAME, SAL, JOB, HIREDATE
from EMP where DEPTNO = 10;
```

Обратите внимание на то, что ключевое слово 'VALUES' здесь не задано.



Задание 5.5.:

Приведите пример sql-скрипта вставки данных в таблицы вашего модуля, созданные в ЛР №4

```
INSERT INTO "HR"."EMPLOYEES" (
    employee_id
    ,first_name
    ,last_name
    ,email
    ,phone_number
    ,hire_date
    ,job_id
    ,salary
) VALUES ( 1
    , 'Valentin'
    , 'Kruglov'
    , 'kruglov@mail.ru'
    , '88005553535'
    , TO_DATE('2025-04-15', 'YYYY-MM-DD HH24:MI:SS')
    , 1
    , 24000 );
```

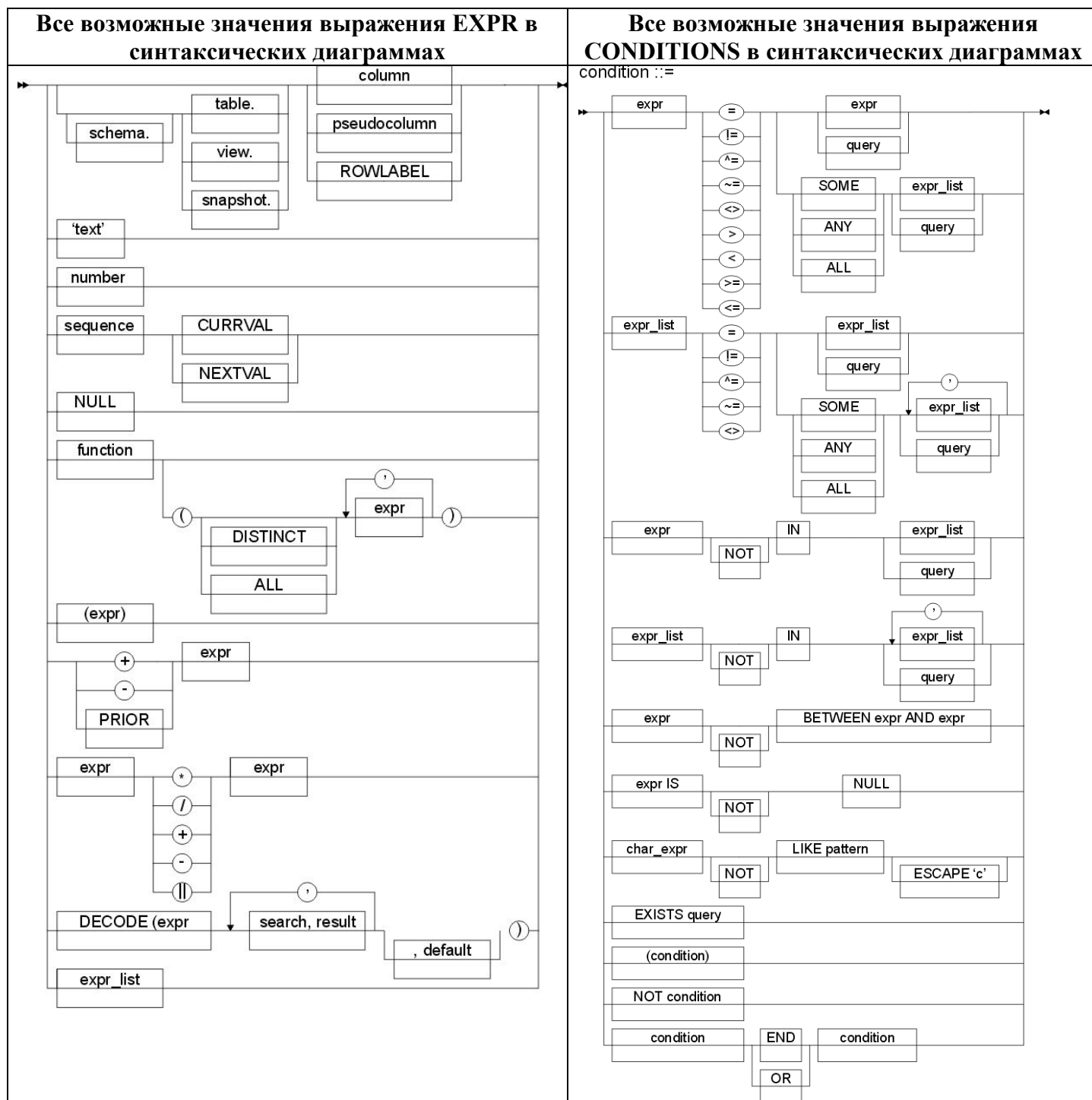


Задание:

Выполните проверку правильности вставки данных

```
✓ SELECT *
  FROM hr.employees
 WHERE employee_id = 1;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	Valentin	Kruglov	kruglov@gmail.com	88005553535	15.04.25	AD_VP	24000



Условные обозначения:

(+) - Указывает, что предшествующий столбец является столбцов внешнего соединения.

* - Указатель на выбор всех столбцов из таблицы или представления.

PRIOR - Используется в иерархическом запросе для определения зависимости между родительскими и дочерними строками.

ALL - Обеспечивает выбор всех строк в запросе (в том числе и повторяющихся).

DISTINCT - Удаляет повторяющиеся строки из результата запроса.

Условные обозначения:

NOT	Отрицание логического выражения
AND	Объединение нескольких логических выражений: TRUE когда все условия истинны.
OR	Объединение нескольких логических выражений, TRUE когда хотя бы одно из выражений истинно.
END	Символ конца списка элементов.
ALL	TRUE, если операция сравнения выполняется для всех элементов списка выражений или множества значений, возвращаемых запросом.
NOT BETWEEN x AND y	[Не]Больше или равно X и меньше или равно Y
IN	Равно любому элементу множества или подзапроса.
NOT IN	Не равно любому элементу множества или подзапроса.
IS [NOT] NULL	Проверка на непустое значение
ANY, SOME	Результат TRUE, если указанная операция сравнения выполняется хотя бы для одного элемента списка выражений или множества значений, возвращаемых запросом.
[NOT] EXISTS	TRUE, если подзапрос [не] возвращает хотя бы одну строку.
[NOT] LIKE p	TRUE, если не совпадает с заданным шаблоном: % - любая последовательность символов, - любой символ.

Пример 1:

```
INSERT INTO doctors (dc_name) VALUES('ИВАНОВ')
```

Пример 2:

```
INSERT INTO doctors (dc_name,  
                    dc_shtat_nnn)  
VALUES('ИВАНОВ',  
      'психотерапевт')
```

Пример 3:

```
INSERT INTO doctors (dc_name,  
                    dc_shtat_nnn)  
VALUES('ИВАНОВ',  
      SELECT shtat_nnn FROM shtat  
      WHERE UPPER(shtat_name) = 'ПСИХОТЕРАПЕВТ'))
```

Пример 4: В качестве примера рассмотрим вставку данных в таблицу "Праздничные дни" (PRAZDNIKI)

```
insert into prazdniki values ('понедельник');  
insert into prazdniki values ('вторник');  
insert into prazdniki values ('среда');  
insert into prazdniki values ('четверг');  
insert into prazdniki values ('пятница');  
insert into prazdniki values ('суббота');  
insert into prazdniki values ('воскресенье');
```

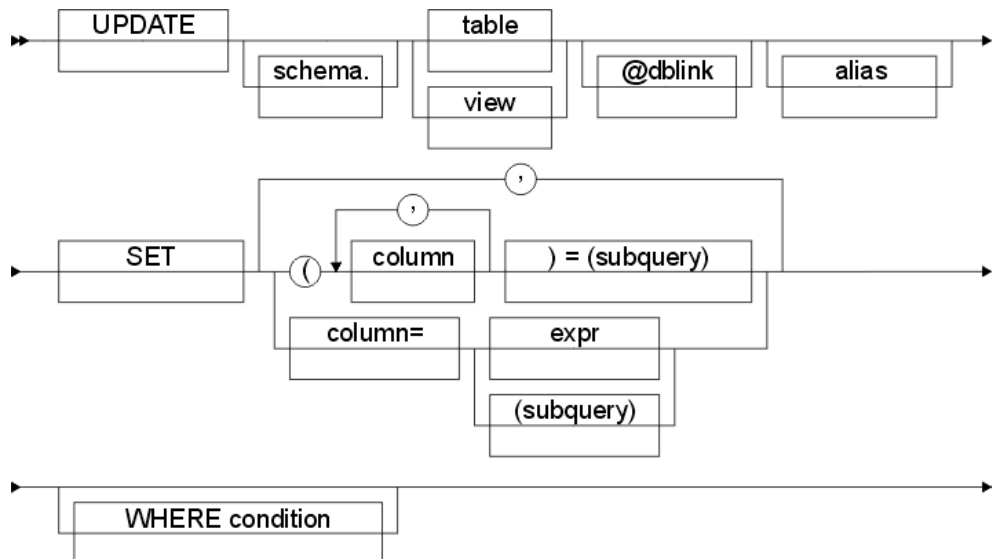
Изменение строк (UPDATE)

Команда UPDATE позволяет исправлять содержимое строк в таблице. Синтаксис:

UPDATE таблица [заменитель] SET столбец [, столбец] = (выражение, подзапрос)
[WHERE условие];

UPDATE

Изменяет существующие значения в таблице или в представлении (View).



Пример:

UPDATE doctors set dc_shtat_nnn = 11 WHERE dc_name = 'ИВАНОВ';

Результат 'ИВАНОВ' переводится в отдел с номером 11, но таких ивановых может быть несколько, следовательно такое обновление должно осуществляться не по имени пользователя, а по его NNN.



Задание 5.6.:

Приведите пример sql-скрипта для изменения значений ряда атрибутов в таблицах вашего модуля, созданных в ЛР №4

```
UPDATE hr.employees
SET
  last_name = 'Kruglovvs'
WHERE employee_id = 1;
```



Задание :

Выполните проверку правильности изменения данных

```
SELECT *
FROM hr.employees
WHERE employee_id = 1;
```

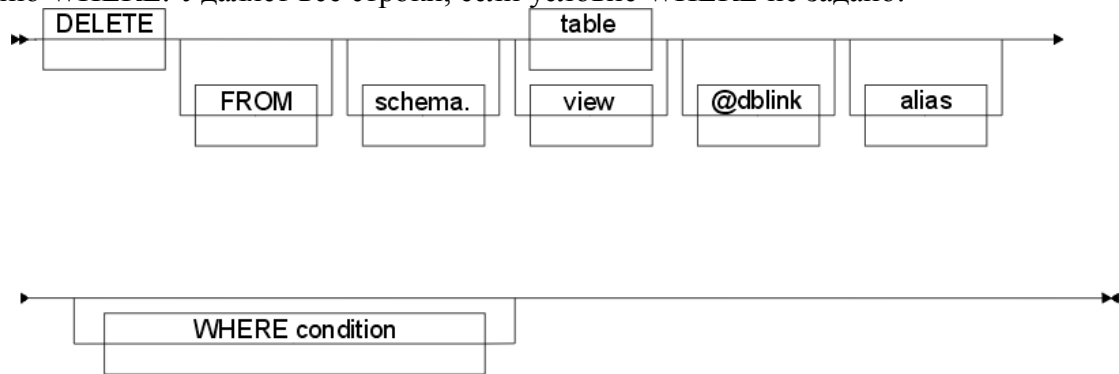
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	1	Valentin	Kruglovvs

Удаление строк из таблицы (DELETE)

Команда DELETE позволяет удалять одну или более строк из таблицы. Синтаксис: DELETE FROM таблица [WHERE условие];
При отсутствии предложения WHERE будут удалены все строки таблицы.

DELETE

Удаляет строки из таблицы или из базовой таблицы представления, удовлетворяющие условию WHERE. Удаляет все строки, если условие WHERE не задано.



Пример:

Удаляем все записи из таблицы Праздничных дней. delete from prazdniki;



Задание 5.7.:

Приведите пример sql-скрипта для удаления строк в таблицах вашего модуля, созданных в ЛР №4 по условию

```
DELETE FROM hr.employees  
WHERE employee_id = 1;
```



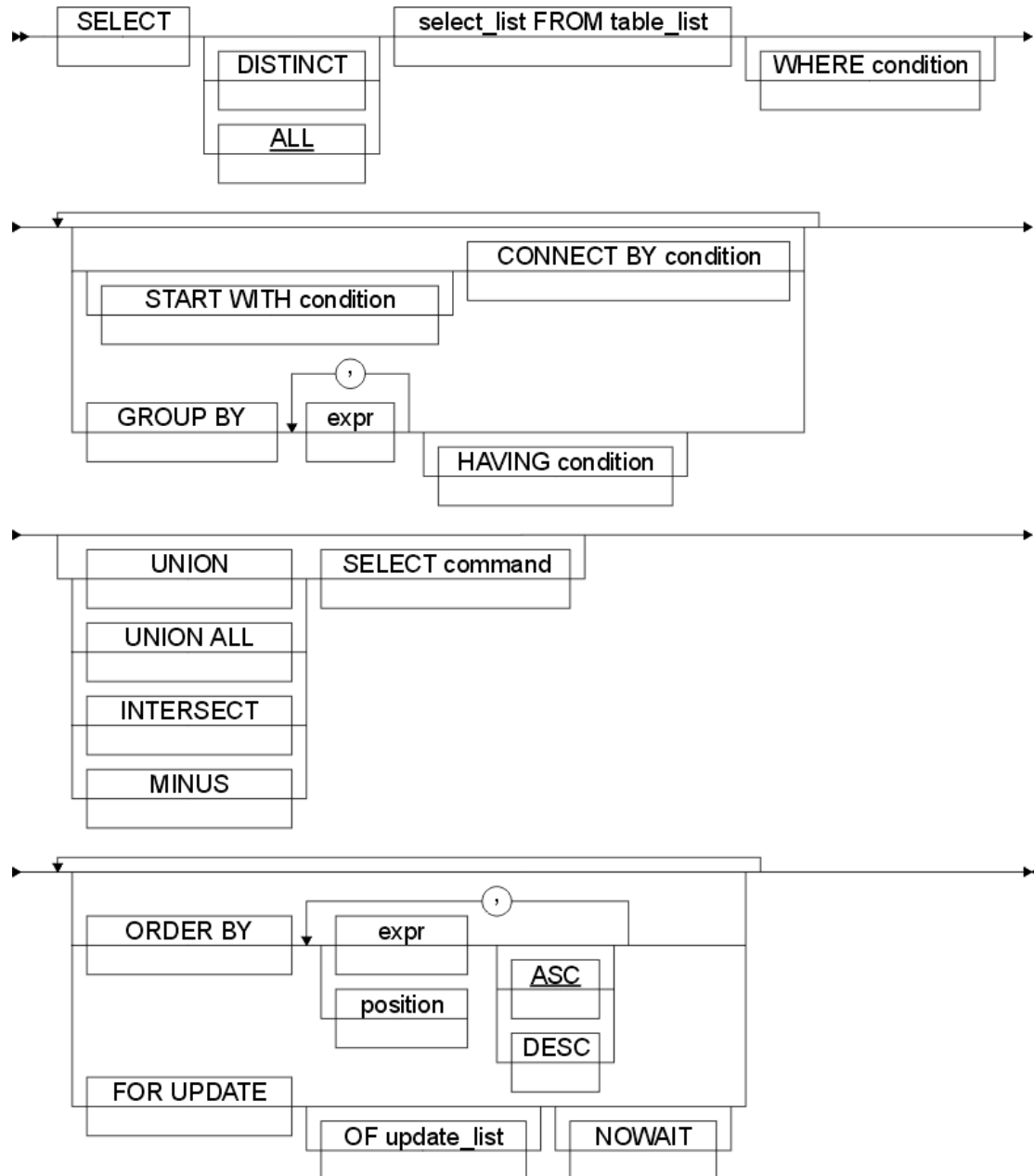
Задание:

Выполните проверку правильности изменения данных

```
SQL> select * from hr.employees where employee_id = 1;  
  
no rows selected
```

Оператор SELECT

Выбирает данные из одной или нескольких таблиц или представлений. Может использоваться как оператор или как подзапрос в другом операторе.



Пример 1: Лучшим примером, иллюстрирующим работу оператора SELECT, является юмористический пример "Как программист SQL охотится на слонов". Дано: Слон живет в Африке. Задача: Что надо сделать, чтобы найти слона? Метод решения: Программист SQL делает SELECT.

SELECT "СЛОН" FROM AFRICA;

Итог: Все африканские слоны найдены ☺.

Проиллюстрируем использование оператора SELECT на нескольких примерах.

Арифметические выражения

Выражение - это комбинация одного или более числовых значений, операторов и функций, вырабатывающая числовое значение. Арифметические выражения могут содержать имена полей, числовые константы и арифметические операторы:

Оператор конкатенации

Оператор конкатенации (||) позволяет соединять столбцы с другими столбцами, арифметическими выражениями или константами для формирования символьных строк. Столбцы по обе стороны оператора сливаются воедино.

Обработка пустых значений

Если в поле нет значения, то оно считается пустым (NULL). Пустое значение - это значение, которое недоступно, не присвоено, неизвестно или не определено. Пустое значение - это не нулевое значение. Ноль - это число. Пустые значения занимают один байт памяти. Пустые значения поддерживаются языком SQL. Если хотя бы одно поле в выражении имеет пустое значение, то и результат не определен.

Сортировка результатов

При выдаче таблицы результатов порядок строк в ней не определен. Предложение ORDER BY используется для сортировки строк по значениям полей в столбцах. Предложение ORDER BY всегда стоит в конце команды SELECT.

Чтобы изменить порядок сортировки, т.е. отсортировать строки по убыванию значений полей столбца (Descending order), задайте квалификатор DESC в предложении ORDER BY.

Допускается сортировка по нескольким столбцам, вплоть до всех столбцов таблицы результатов. Необходимо задавать столбцы сортировки в предложении ORDER BY через запятую. При этом первичный ключ сортировки стоит в предложении ORDER BY первым, вторичный - вторым и т.д.

Правила соединения таблиц

Чтобы соединить три таблицы, необходимо задать два условия соединения. Чтобы соединить четыре таблицы, нужны как минимум три соединяющих условия.

Общее простое правило: количество соединяемых таблиц минус один = минимальное число условий соединения.

Это правило может не работать, если в вашей таблице есть конкатенированный (состоящий из нескольких полей) первичный ключ (PRIMARY KEY), однозначно идентифицирующий каждую запись. Когда условие соединения задано некорректно или вообще отсутствует, результатом соединения оказывается произведение двух таблиц.

Использование сокращенных имен таблиц

Довольно утомительно несколько раз набирать имена таблиц в одной команде - имена могут быть длинными и повторяться много раз в предложениях SELECT, WHERE, ORDER BY. Чтобы избежать этого, можно использовать временные синонимы (или заменители) имен таблиц, задаваемые в предложении FROM и действующие только в данной команде. Заменители имен таблиц (называемые также табличными метками) следует также задавать в предложении SELECT. Это повышает эффективность выполнения запроса, упрощая его компиляцию

ПРИМЕР: select E.ENAME, D.DEPTNO, D.DNAME from EMP E, DEPT.D
where E.DEPTNO = D.DEPTNO
order by D.DEPTNO

Внешние соединения

Оператор внешнего соединения позволяет выбрать те строки из одной таблицы, которым нет пар в другой. Он может быть задан только с одной стороны соединяющего условия - с той, где отсутствует информация. Для получения не имеющих пары строк в обеих таблицах сразу, необходимо воспользоваться оператором UNION (см. ниже).

Другие ограничения на внешние соединения:

- Вы не можете сделать внешнее соединение одной таблицы с двумя другими в одной команде SELECT (т.е. поставить (+) напротив одной и той же таблицы в двух операторах сравнения).
- Условие внешнего соединения не может включать оператор IN или быть связано с другим условием оператором OR.

ПРИМЕР: select E.ENAME, D.DEPTNO, D.DNAME
from EMP E, DEPT D
where E.DEPTNO(+) = D.DEPTNO
and D.DEPTNO IN (30,40);

Рекурсивные соединения

С помощью табличных меток можно подсоединить таблицу к самой себе, как будто это две различные таблицы. Это дает возможность соединить строки таблицы со строками этой же таблицы.

ПРИМЕР: select E.ENAME "EMP NAME", E.SAL "EMP SAL",
M.ENAME "MGR NAME", M.SAL "MNR SAL"
from EMP E, EMP M
where E.MGR = M.EMPNO
and E.SAL < M.SAL;

Вложенные подзапросы

Подзапросы могут быть вложенными (т.е. в одних подзапросах могут задаваться другие). Допускается до 255 уровней вложенности подзапросов.

Основные правила при задании подзапросов

Внутренний запрос берется в круглые скобки и должен стоять в правой части оператора сравнения внешнего запроса.

- Подзапрос не может содержать предложения ORDER BY.
- Предложение ORDER BY ставится последним в основном запросе.
- Имена столбцов в предложении SELECT внутреннего запроса должны стоять в той же последовательности, что и имена столбцов в левой части оператора сравнения внешнего запроса. Типы столбцов также должны попарно соответствовать.
- Подзапросы всегда выполняются от внутренних к внешним, пока не коррелируют друг с другом (обсуждается позже). Выполнение сравнения во внешнем запросе базируется на результатах выполнения внутреннего запроса.
- При задании критерия поиска могут использоваться логические операторы, SQL-операторы, а также операторы ANY и ALL.
- Подзапросы могут:
 - возвращать одну или более строк;

- возвращать один или более столбцов;
- использовать группы или групповые функции;
- задаваться в сложных критериях поиска внешних запросов с использованием предикатов AND и OR;
- соединять таблицы;
- обращаться к таблице, отличной от той, к которой обращается внешний запрос;
- стоять в командах SELECT, UPDATE, DELETE, INSERT, CREATE TABLE.
- коррелировать с внешним запросом;
- использовать операторы над множествами.

Коррелирующие подзапросы

Коррелирующие подзапросы - это вложенные подзапросы, выполняющиеся для каждой "строки-кандидата" из главного запроса, и для выполнения которых требуется информация из строк главного запроса. Такая корреляционная зависимость подразумевает иной алгоритм выполнения запроса, чем в запросах с обычными вложенными подзапросами. Коррелирующий подзапрос задается постановкой во внутренний подзапрос имени столбца из внешнего запроса. В обычном вложенном подзапросе внутренний запрос выполняется первым и один раз, возвращая значения, используемые в главном запросе. Коррелирующий подзапрос выполняется по разу для каждой строки (кандидата), выбранной внешним запросом.

ПРИМЕР: Найти поиска всех сотрудников, получающих больше, чем средняя зарплата в их отделе:

```
select EMPNO, ENAME, SAL, DEPTNO
from EMP E
where SAL > (SELECT AVG(SAL) FROM EMP
              where DEPTNO = E.DEPTNO)
order by DEPTNO;
```



Задание 5.8.:

Выведите сведения о сотрудниках, которые подчиняются такому же менеджеру, что и сотрудники с номерами (EMPLOYEE_ID) 141 или 174, и работают в одном отделе, что и сотрудники с номерами 141 или 174, не включая самих сотрудников в список.

```
1 SELECT employee_id
2     ,last_name
3     ,manager_id
4     ,department_id
5 FROM hr.employees
6 WHERE manager_id IN (
7     SELECT manager_id
8     FROM hr.employees
9     WHERE employee_id IN ( 174
10                          ,141 )
11 )
12 AND department_id IN (
13     SELECT department_id
14     FROM hr.employees
15     WHERE employee_id IN ( 174
16                          ,141 )
17 )
18 AND employee_id NOT IN ( 174
19                          ,141 );
```

```
SELECT employee_id
     ,last_name
     ,manager_id
     ,department_id
FROM hr.employees
WHERE manager_id = ANY (
    SELECT manager_id
    FROM hr.employees
    WHERE employee_id = ANY ( 174
                            ,141 )
)
AND department_id = ANY (
    SELECT department_id
    FROM hr.employees
    WHERE employee_id = ANY ( 174
                            ,141 )
)
AND employee_id <> ALL ( 174
                       ,141 );
```

EMPLOYEE_ID	LAST_NAME	MANAGER_ID	DEPARTMENT_ID
142	Davies	124	50
143	Matos	124	50
144	Vargas	124	50
196	Walsh	124	50
197	Feeney	124	50
198	OConnell	124	50
199	Grant	124	50
175	Hutton	149	80
176	Taylor	149	80
177	Livingston	149	80
179	Johnson	149	80

11 rows selected.

Операции над множествами

Операторы **UNION**, **INTERSECT** и **MINUS** удобны при составлении запросов к разным таблицам. Они комбинируют результаты по нескольким запросам в одну результирующую таблицу. Таким образом, запрос может состоять из нескольких SQL-команд, связанных одним или более операторами над множествами. Операторы над множествами часто называются вертикальным соединением, поскольку соединение в данном случае происходит не по строкам, а по столбцам.

Операция	Выполняемые функции
UNION	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные хотя бы одним из запросов.
UNION ALL	Комбинирует два запроса; возвращает все строки, извлеченные хотя бы одним из запросов, включая повторяющиеся.
INTERSECT	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные каждым из запросов.
MINUS	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные первым запросом, но не извлеченные вторым.

Операция	Выполняемые функции
(+)	Указывает, что предшествующий столбец является столбцом внешнего соединения.
*	Используется вместо имен столбцов при выборке всех столбцов из таблицы или представления.
PRIOR	Используется в иерархическом древовидном запросе для определения зависимости между родительскими и дочерними строками. Смотрите оператор SELECT.
ALL	Оставляет повторяющиеся строки в результате запроса (установлен по умолчанию ALL, но не DISTINCT).
DISTINCT	Удаляет повторяющиеся строки из результата запроса.

Правила, которые необходимо соблюдать при использовании операторов над множествами.

- Команды SELECT должны выбирать одинаковое количество столбцов.
- Соответствующие столбцы должны иметь одинаковый тип.
- Одинаковые строки автоматически исключаются (за исключением UNION ALL). Предикат DISTINCT не допускается.
- В результирующей таблице столбцы именуются по первой SQL-команде.

UNION

Служит для получения всех неодинаковых строк, выбираемых по нескольким командам (объединение результирующих таблиц):

ПРИМЕР: select JOB from EMP where DEPTNO=10
 UNION
 select JOB from EMP where DEPTNO=30;

INTERSECT

Служит для получения всех неодинаковых строк, которые выбраны каждой из команд (пересечение результирующих таблиц)

ПРИМЕР: select JOB from EMP where DEPTNO=10
 INTERSECT
 select JOB from EMP where DEPTNO=30;

MINUS

Служит для того, чтобы выбрать все строки, найденные по одному запросу, но не найденные по другому (разность результирующих таблиц).

Операторы SOME|ANY и ALL

Операторы ANY и ALL могут применяться в подзапросах, возвращающих более одной строки. Они задаются в предложениях WHERE или HAVING вместе с логическими операторами (=, <>, <, >, >=, <=). ANY (или его синоним SOME) сравнивает значение левой части оператора сравнения с каждым значением, возвращаемым подзапросом. Результат сравнения положителен, если хотя бы одно значение из найденных по подзапросу удовлетворяет оператору сравнения.

ПРИМЕР: Все сотрудники, зарабатывающие больше минимальной зарплаты в отделе 30;
 select ENAME, SAL, JOB, DEPTNO
 from EMP
 where SAL > SOME (select DISTINCT SAL
 from EMP
 where DEPTNO=30)
 order by SAL DESC;

ПРИМЕР: Все служащие, которые зарабатывают больше, чем любой из сотрудников 30-го отдела;
 select ENAME, SAL, JOB, DEPTNO
 from EMP
 where SAL > ALL (select DISTINCT SAL
 from EMP where DEPTNO = 30);



Задание 5.8.1:

Выполните задание 5.8. с использованием операций над множествами

Функции группировки

Статистическая информация (такая, как среднее или суммарное значение) может быть получена для группы строк таблицы с помощью так называемых групповых функций. Групповые функции оперируют с набором строк. Они формируют результаты, основанные на обработке группы строк и для этой группы, в отличие от простых функций, работающих с отдельными строками. По умолчанию все строки таблицы результатов трактуются как одна группа. Чтобы разбить строки на группы, используется предложение **GROUP BY**, специфицируемое в команде SELECT.

Ниже приведен список групповых функций.

Функция	Возвращаемое значение
AVG ([DISTINCT/ALL]n)	Среднее значение группы полей в столбце, не включая пустых значений
COUNT ([DISTINC/ALL] выражение)	Количество строк в группе, в которых имеет непустое значение
MAX ([DISTINCT/ALL] выражение)	Максимальное значение выражения в группе
MIN ([DISTINCT/ALL] выражение)	Минимальное значение выражения в группе
STDDEV ([DISTINCT/ALL] n)	Функция, определяющая математическое ожидание в группе
SUM ([DISTINCT/ALL] n)	Сумма значений, игнорируя пустые значения
VARIANCE ([DISTINCT/ALL] n)	Дисперсия в группе.

В приведенной таблице под числом понимается имя поля, константа или арифметическое выражение числового типа. Выражение может иметь любой тип. Чаще всего в качестве чисел и выражений используются имена столбцов таблиц. Перечисленные функции оперируют с группами строк выходных таблиц (например, со всей таблицей, как с одной группой) и в связи с этим называются групповыми или агрегированными функциями. Квалификатор DISTINCT дает команду групповой функции оперировать только с неодинаковыми строками в группах, т.е. дублированные строки при работе функции отбрасываются. Квалификатор ALL оставляет дублированные строки для обработки групповыми функциями. По умолчанию все функции используют квалификатор ALL. Тип данных выражения может быть CHAR, NUMBER или DATE (для функций, использующих в качестве аргумента выражение). Все групповые функции, кроме COUNT(*), игнорируют пустые значения. Для обработки пустых значений используйте функцию NVL.

ПРИМЕР: Вычислить среднее значение зарплаты для каждой должности
select JOB, AVG(SAL) from EMP group by JOB

	Задание 5.9: Составьте список всех сотрудников, зарабатывающих больше среднего оклада по отделу, в котором они работают.																																	
<pre>SELECT last_name, salary, department_id FROM employees outer WHERE salary > (SELECT AVG(salary) FROM employees WHERE department_id = outer.department_id);</pre>	<table><tr><th>LAST_NAME</th><th>SALARY</th><th>DEPARTMENT_ID</th></tr><tr><td>Fox</td><td>9600</td><td>80</td></tr><tr><td>Abel</td><td>11000</td><td>80</td></tr><tr><td>Sarchand</td><td>4200</td><td>50</td></tr><tr><td>Bull</td><td>4100</td><td>50</td></tr><tr><td>Chung</td><td>3800</td><td>50</td></tr><tr><td>Dilly</td><td>3600</td><td>50</td></tr><tr><td>Bell</td><td>4000</td><td>50</td></tr><tr><td>Everett</td><td>3900</td><td>50</td></tr><tr><td>Martinez</td><td>13000</td><td>20</td></tr><tr><td>Higgins</td><td>12000</td><td>110</td></tr></table>	LAST_NAME	SALARY	DEPARTMENT_ID	Fox	9600	80	Abel	11000	80	Sarchand	4200	50	Bull	4100	50	Chung	3800	50	Dilly	3600	50	Bell	4000	50	Everett	3900	50	Martinez	13000	20	Higgins	12000	110
LAST_NAME	SALARY	DEPARTMENT_ID																																
Fox	9600	80																																
Abel	11000	80																																
Sarchand	4200	50																																
Bull	4100	50																																
Chung	3800	50																																
Dilly	3600	50																																
Bell	4000	50																																
Everett	3900	50																																
Martinez	13000	20																																
Higgins	12000	110																																
Результат:	строк																																	
38 rows selected.																																		

38 rows selected.

	Задание 5.10: Выведите Название стран и среднюю зарплату (с округлением до целого значения) сотрудников работающих в этих странах, отсортировав результат в порядке убывания среднего оклада.										
<pre>select country_name, round(avg(salary)) as n from countries c,locations l,departments d,employees e where c.country_id = l.country_id and l.location_id = d.location_id and e.department_id = d.department_id group by country_name order by n desc</pre>	<table> <thead> <tr> <th>COUNTRY_NAME</th><th>N</th></tr> </thead> <tbody> <tr><td>Germany</td><td>10000</td></tr> <tr><td>Canada</td><td>9500</td></tr> <tr><td>United Kingdom of Great Britain and Northern Ireland</td><td>8886</td></tr> <tr><td>United States of America</td><td>5065</td></tr> </tbody> </table>	COUNTRY_NAME	N	Germany	10000	Canada	9500	United Kingdom of Great Britain and Northern Ireland	8886	United States of America	5065
COUNTRY_NAME	N										
Germany	10000										
Canada	9500										
United Kingdom of Great Britain and Northern Ireland	8886										
United States of America	5065										
Результат:											

```

-- Для схемы SCOTT определить среднюю зарплату
SELECT j.job_title
      ,round(avg(e.salary)) AS avg_salary
FROM hr.employees e
LEFT JOIN hr.jobs j
ON j.job_id = e.job_id
GROUP BY j.job_title
HAVING avg_salary > 3000;

```

```

SELECT EXTRACT(YEAR FROM e.hire_date) AS year
      ,COUNT(*)
FROM hr.employees e
GROUP BY year;

```

HAVING

Предложение **HAVING** задается, если необходимо выбрать не все формируемые предложением GROUP BY группы, а лишь часть их по определенному критерию (на групповые характеристики).

ПРИМЕР: Определить среднюю зарплату в отделах, имеющих более трех служащих:
 select DEPTNO, AVG(SAL) from EMP group by DEPTNO having COUNT (*) > 3;



Задание 5.8.2:

ЗАДАНИЕ: Для схемы SCOTT определить среднюю зарплату по должностям для каждого из отделов, величина которой превышает 3000;

ЗАДАНИЕ: Для схемы SCOTT определить в каком году в компанию было зачислено наибольшее количество человек? Выдайте этот год и количество зачисленных служащих.

Оператор EXISTS

Оператор EXISTS часто задается с коррелирующими подзапросами. Он проверяет, найдена ли по подзапросу хотя бы одна строка. Если да, то условие выполнено и оператор возвращает значение TRUE; если нет - то FALSE (оператор NOT EXISTS возвращает TRUE, если, наоборот, ни одной строки по подзапросу не найдено).

ПРИМЕР: Найти всех служащих, у которых имеется хотя бы один подчиненный
 select EMPNO, ENAME, JOB, DEPTNO
 from EMP E
 where EXISTS (select EMPNO from EMP
 where EMP.MGR=E.EMPNO)

```

SELECT first_name
      ,last_name
      ,job_title
FROM hr.employees e
LEFT JOIN hr.jobs j
ON e.job_id = j.job_id
WHERE NOT EXISTS (
  SELECT ROWNUM
  FROM hr.employees slave
  WHERE slave.manager_id = e.employee_id
);

```



Задание 5.8.3:

ЗАДАНИЕ: Найти всех служащих, к которым не существует подчиненных. Проверка ошибки след. Запроса:
 select ENAME, JOB FROM EMP
 where EMPNO NOT IN (select MGR from EMP)

Правила задания комментариев в тексте SQL программ

Введение комментариев в исходные тексты программ:

- значительно упрощает разработку и отладку программ, особенно при ведении совместных разработок.
- использование комментариев при разметке сложной программы на языке PL/SQL, позволяет отметить отдельные разделы, а затем вернуться к каждому разделу и заполнять его операторами.
- комментарий в заголовке модуля, содержащий основную информацию о модуле, поможет получить общую информацию и значительно сокращает время на анализ сценария;

После компиляции сценария, сохраненного с такими многострочными комментариями, компилятор PL/SQL выводит их на экран в виде блока описания программ DOC:

Это удобно в случае компиляции полного набора модулей (при этом все результаты выводятся в файл). Рекомендуется ограничивать количество строк в комментарии заголовка модуля, так как он может быть выведен на экран в процессе компиляции и сбить с толку оператора. Если первый ограничитель многострочного комментария не размещен в строке один, то первая строка комментария не выводится на экран.



Задание 5.11: Вывести из таблицы HR.EMPLOYEES информацию о сотрудниках отделов с номерами 10,30,50,90. Вывод должен быть оформлен в таблицу, содержащую столбцы:

1. Сквозной порядковый номер сотрудника.
2. Порядковый номер сотрудника внутри отдела.
3. Номер отдела для данного сотрудника (Department_id).
4. Должность сотрудника (Job_id).
5. Фамилия сотрудника (Last_name).
6. Оклад (Salary).
7. Ранг зарплаты сотрудника в отделе (1-самый высокооплачиваемый).

Строки в выводимой таблице должны удовлетворять следующим условиям:

1. Строки, представляющие сотрудников одного отдела должны располагаться друг за другом.
2. Строки, представляющие сотрудников одного отдела должны располагаться в порядке убывания окладов.

Дополнительные требования к выполнению:

ито ранг в отделе зависит от нумерации в отделе.

```
SELECT ROW_NUMBER()
  OVER(
    ORDER BY department_id
  ) "Сквозной №"
, ROW_NUMBER()
  OVER(PARTITION BY department_id
    ORDER BY salary DESC
  ) "№ внутри отдела"
, department_id "# отд."
, job_id "Должность"
, last_name "Фамилия"
, salary "Зарплата"
, RANK()
  OVER(PARTITION BY department_id
    ORDER BY salary DESC
  ) "Ранг в отделе"
FROM hr.employees
WHERE department_id IN ( 10
                        ,30
                        ,50
                        ,90 )
ORDER BY department_id DESC
       ,salary;
```

43	36	50 ST_CLERK	Patel	2500	36
44	37	50 SH_CLERK	Perkins	2500	36
45	38	50 ST_CLERK	Vargas	2500	36
46	39	50 ST_CLERK	Marlow	2500	36
47	40	50 SH_CLERK	Sullivan	2500	36
48	41	50 ST_CLERK	Landry	2400	41
49	42	50 ST_CLERK	Gee	2400	41
50	43	50 ST_CLERK	Philtanker	2200	43
51	44	50 ST_CLERK	Markle	2200	43
52	45	50 ST_CLERK	Olson	2100	45
53	1	90 AD_PRES	King	24000	1
54	2	90 AD_VP	Yang	17000	2
55	3	90 AD_VP	Garcia	17000	2

55 rows selected.

строк

Контрольные вопросы

1. Синтаксическая диаграмма SELECT, пример.
2. Синтаксическая диаграмма INSERT, пример.
3. Синтаксическая диаграмма UPDATE, пример.
4. Синтаксическая диаграмма DELETE, пример.

СПИСОК ЛИТЕРАТУРЫ

1. Власов А.И., Лыткин С.Л., Яковлев В.Л. Краткое практическое руководство по языку PL/SQL - М.: Машиностроение. 2000. 64 с.
2. Сервер Oracle. Справочное руководство по языку SQL / Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1994.
3. Сервер ORACLE. Основные концепции/ Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1996.
4. Проектирование и эксплуатация конструкторско-технологических баз данных на основе СУБД Oracle/ Конспект лекций - М.: Москва, МГТУ им. Н.Э. Баумана, 2001, 120 с.