11. Подзапросы

11.1 Применение подзапросов

Это занятие посвящено более сложному использованию команды SELECT. Для выборки значений по неизвестному условию можно создать подзапрос в предложении WHERE другой команды SQL.
Предположим, что нужно выяснить, у кого оклад больше, чем у Смирновой Т.Н..
Для этого необходимы два запроса: один для выяснения оклада Смирновой Т.Н и другой для выяснения того, кто получает больше.
Для решения этой задачи можно создать один запрос внутри другого. Значение, возвращаемое внутренним запросом или подзапросом, используется внешним или главным запросом. Использовать подзапрос - это то же самое, что последовательно выполнить два запроса, использовав результат первого в качестве критерия поиска во втором.

```
SELECT     select_list
FROM       table
WHERE      expr operator
           (SELECT  select_list
            FROM    table);
```

- Подзапрос (внутренний запрос) выполняется один раз до главного запроса.
- Результат подзапроса используется главным запросом (внешним запросом).

Подзапрос - это команда SELECT, включенная в предложение другой команды SELECT. С помощью подзапросов можно создавать очень мощные команды из простых. Они очень полезны в случае, если выборка строк из таблицы производится по условию, зависимому от данных в самой таблице.
Подзапрос можно использовать в следующих предложениях языка SQL:

- WHERE
- HAVING
- FROM

Подзапрос часто называют вложенной командой SELECT, подкомандой SELECT или внутренней командой SELECT. Обычно подзапрос выполняется первым, и его результат используется для определения условия выборки в главном или внешнем запросе.

Пример

```
select fio,oklad from inspector t,oklad o where t.kodinsp=o.kodinsp
and o.oklad > (select oklad from inspector a,oklad b
               where a.kodinsp=b.kodinsp and fio='Смирнова Т.Н.')
```

FIO	OKLAD
Чабановская В.С.	3450
Кудрявцев А.А.	2800
Савилов Ю.Н.	3450
Чайка Н.Ф.	3450

Внутренний подзапрос определяет оклад Смирновой Т.Н. Внешний запрос использует результат внутреннего запроса для вывода списка всех служащих, зарабатывающих больше этой суммы.

11.2 Правила использования подзапросов:

- Подзапрос должен быть заключен в скобки.
- Подзапрос должен находиться справа от оператора сравнения.

- Подзапросы не могут содержать предложение ORDER BY. В команде SELECT может быть только одно предложение ORDER BY. Если предложение ORDER BY используется, оно должно быть последним в главной команде SELECT.
- В подзапросах используются операторы сравнения двух типов: однострочные и многострочные.

11.3 Типы подзапросов

- Однострочные подзапросы: запросы, в которых внутренняя команда SELECT возвращает только одну строку
- Многострочные подзапросы: запросы, в которых внутренняя команда SELECT возвращает более одной строки
- Многостолбцовые запросы: запросы в которых внутренняя команда SELECT возвращает более одного столбца

11.4 Однострочные подзапросы

- Возвращают только одну строку
- Используют однострочные операторы сравнения

Оператор	Значение
=	Равно
>	Больше, чем
>=	Больше или равно
<	Меньше, чем
<=	Меньше или равно
<>	Не равно

Внутренняя команда SELECT, возвращающая только одну строку, называется однострочным подзапросом. В подзапросах такого типа используются однострочные операторы. Выше показан список однострочных операторов.

Пример

Вывод списка служащих с такой же должностью, как у служащего под номером 5611.

```
SQL>  SELECT     fio, dol
  2   FROM       inspector
  3   WHERE      dol =
  4              (SELECT     dol
  5              FROM        inspector
  6              WHERE       kodinsp=
                             5611);
```

```
FIO                          DOL
---------------------------- ------------------
Чабановская В.С.     Нач. отдела
Савилов Ю.Н.                 Нач. отдела
Чайка Н.Ф.           Нач. отдела
```

11.5 Выполнение однострочных подзапросов

Пример

```
select fio,oklad from inspector t,oklad o where t.kodinsp=o.kodinsp
and o.oklad > (select oklad from inspector a,oklad b
              where a.kodinsp=b.kodinsp and fio='Смирнова Т.Н.')
and dol =(SELECT dol    FROM  inspector  WHERE kodinsp= 5611);
```

Команду SELECT можно рассматривать как блок запроса. На примере показан запрос для вывода списка служащих, которые работают в той же должности, что служащий номер 5611 и имеют оклад больше, чем у служащего Смирновой Т.Н.

Пример состоит из трех блоков запроса: одного внешнего и двух внутренних. Сначала выполняются внутренние блоки. Затем выполняется внешний блок. Для создания условий поиска он использует результаты внутренних запросов. Оба внутренних запроса возвращают по одному значению, поэтому данная команда SQL называется однострочным подзапросом.

Примечание: внешний и внутренний подзапросы могут выбирать данные из разных таблиц.

11.6 Использование групповых функций в подзапросах

Подзапрос во внешнем запросе может использовать групповую функцию для получения результата в виде одной строки. Подзапрос заключается в круглые скобки и помещается после оператора сравнения. В показан вывод фамилий, должностей и окладов всех служащих, чей оклад равен минимальному. Групповая функция MIN возвращает во внешний запрос только одно значение.

```
select fio, dol,oklad from inspector t,oklad o where
t.kodinsp=o.kodinsp
and o.oklad = (select min(oklad) from  oklad );
```

```
FIO                  DOL          OKLAD
-------------------------------------------------------------------------
Лютикова З.С.              Инспектор    1850
Азарченко Л.М.             Инспектор    1850
```

11.7 Предложение HAVING с подзапросами

- Сервер Oracle сначала выполняет подзапрос.
- Сервер Oracle возвращает результаты в предложение HAVING главного запроса.

Пример

```
select otdel,min(oklad)  from inspector t,oklad o where t.kodinsp=o.kodinsp
group by otdel
having  min(oklad ) > (select min(oklad) from  oklad )
```

```
OTDEL        MIN(OKLAD)
---------- ----------------
56       2750
57       3450
```

Подзапросы можно использовать не только в предложении WHERE, но и в предложении HAVING. Сервер Oracle выполняет подзапрос, и результаты возвращаются в предложение HAVING главного запроса.

Команда SQL в примере выводит список отделов, минимальные оклады в которых превышают минимальный оклад.

Лабораторная работа 11.1 Работа с однострочными подзапросами

Задание:

Напишите запрос, который возвращал бы информацию о имени, фамилии, заработной плате всех сотрудников из таблицы hr.employees, зарплата которых превышает среднюю для отдела продаж (значение Sales в столбце department_name таблицы departments). Решите эту

задачу при помощи подзапроса. Результат выполнения запроса должен выглядеть так, как представлено на рис. Лаб. 6.1-1.

	Имя	Фамилия	Оклад
1	Michael	Hartstein	13000
2	Hermann	Baer	10000
3	Shelley	Higgins	12000
4	Steven	King	24000
5	Neena	Kochhar	17000
6	Lex	De Haan	17000
7	Alexander	Hunold	9000
8	Nancy	Greenberg	12000
9	Daniel	Faviet	9000
10	Den	Raphaely	11000

Рис. Лаб. 6.1-1

11.8 Ошибки в подзапросах

Самая распространенная ошибка - это когда однострочный подзапрос возвращает более одной строки.

В команде SQL на рисунке подзапрос содержит предложение GROUP BY (deptno), что предполагает возврат нескольких строк - по одной на каждую найденную группу. В данном случае результатами подзапроса будут 800,1300 и 950.

Внешний запрос берет результаты подзапроса (800,950,1300) и использует их в предложении WHERE. Предложение WHERE содержит оператор равенства “=” - однострочный оператор, ожидающий только одно значение. Оператор равенства не может принять от подзапроса более одного значения и, следовательно, выдаст ошибку. Для исправления ошибки замените оператор " =" оператором IN.

```
Select  oklad from oklad where kodinsp =(select kodinsp from inspector where
fio='Петров');
```

Самая распространенная проблема - это когда внутренний запрос не возвращает ни одной строки.

В примере команда SQL содержит предложение WHERE (fio=’Петров’). Очевидно, цель состояла в поиске служащего по фамилии Петров. Команда кажется правильной, но при выполнении не возвращает ни одной строки.

Проблема в том, что фамилия указана неправильно. Служащего с такой фамилией нет. Поэтому подзапрос ничего не возвращает. Внешний запрос берет результаты подзапроса (неопределенное значение) и использует их в своем предложении WHERE. Внешний запрос не находит ни одного служащего с должностью, равной неопределенному значению, и ничего не возвращает.

11.9 Многострочные подзапросы

- Возвращают более одной строки
- Используют многострочные операторы сравнения

Оператор	**Значение**
IN	Равно любому члену списка
ANY	Сравнение значения с любым значением, возвращаемым подзапросом
ALL	Сравнение значения с каждым значением, возвращаемым подзапросом

Подзапросы, возвращающие более одной строки, называются многострочными.

Вместо однострочного оператора в них используется многострочный. Многострочный оператор ожидает одно или более значений.

Пример

Найдите служащих, оклад которых равен минимальному окладу в отделах.

```
select fio, dol,oklad from inspector t,oklad o where t.kodinsp=o.kodinsp
and o.oklad  in (select min(oklad) from  oklad a,inspector b where
a.kodinsp=b.kodinsp group by b.otdel );
```

Сначала выполняется внутренний запрос. Затем обрабатывается главный блок. Результаты внутреннего запроса используются при этом для завершения условия поиска в главном запросе.

11.10 Использование оператора ANY в многострочных подзапросах

```
SQL> SELECT    fio, dol, oklad
      FROM      inspector I,oklad o
      WHERE     i.kodinsp=o.kodinsp and oklad >
                ANY(SELECT  oklad FROM inspector I,oklad o
                    WHERE i.kodinsp=o.kodinsp  AND dol = 'Инспектор')
      AND      dol < > 'Инспектор' ;
```

FIO	DOL	OKLAD
Чабановская В.С.	Нач. отдела	3450
Савилов Ю.Н.	Нач. отдела	3450
Чайка Н.Ф.	Нач. отдела	3450

Оператор ANY (и его синоним SOME) сравнивает значение с любым значением, возвращаемым подзапросом. Запрос возвращает список служащих, которые не являются инспекторами и оклады которых больше, чем у любого инспектора.

>ANY	больше, чем максимум
<ANY	меньше, чем максимум
=ANY	эквивалент IN

11.11 Использование оператора ALL в многострочных подзапросах

```
 SELECT    fio, dol, oklad
FROM      inspector I,oklad o
WHERE     i.kodinsp=o.kodinsp and oklad < ALL
                  (SELECT avg(oklad) FROM inspector I,oklad o
                   WHERE i.kodinsp=o.kodinsp GROUP BY otdel);
```

FIO	DOL	OKLAD
Лютикова З.С.	Инспектор	1850
Азарченко Л.М.	Инспектор	1850

Оператор ALL сравнивает значение с каждым значением, возвращаемым подзапросом.
Запрос возвращает список служащих, чей оклад меньше среднего оклада любого из отделов.

>ALL	больше, чем максимум
<ALL	меньше, чем минимум

Оператор NOT не может использоваться с операторами ANY и ALL.

Лабораторная работа 11.2 Применение многострочных подзапросов

Задание:

Напишите запрос, который возвращает имена и фамилии сотрудников на основе информации из таблицы hr.employees. Должны вернуться записи только для тех сотрудников, которые выполняют менеджерские функции (для которых их номер employee_id встречается в столбце manager_id той же таблицы). Решите эту задачу при помощи многострочного подзапроса. Результат выполнения запроса должен выглядеть так, как представлено на рис. Лаб. 6.2-1.

	Имя	Фамилия
1	Michael	Hartstein
2	Shelley	Higgins
3	Steven	King
4	Neena	Kochhar
5	Lex	De Haan
6	Alexander	Hunold
7	Nancy	Greenberg
8	Den	Raphaely
9	Matthew	Weiss
10	Adam	Fripp

Рис. Лаб. 6.2-1

ИТОГИ

Подзапросы полезны, когда запрос основан на неизвестных значениях.

```
SELECT     select_list

FROM       table

WHERE      expr operator
           (SELECT    select_list
            FROM      table);
```

Подзапрос - это команда SELECT, включенная в предложение другой команды SQL. Подзапросы полезны в случаях, когда запрос основан на неизвестных критериях поиска.
Характеристики подзапросов:

- Могут передавать одну строку данных в главный запрос, содержащий однострочный оператор (=, <>, >, >=, < или <=)
- Могут передавать более одной строки данных в главный запрос, содержащий многострочный оператор (например, IN)
- Сервер Oracle обрабатывает их первыми, а затем их результаты используются в предложении WHERE или HAVING
- Могут содержать групповые функции

12. Многостолбцовые подзапросы

12.1 Что такое многостолбцовые подзапросы

До сих пор рассматривались только однострочные и многострочные запросы, когда в предложении WHERE или HAVING команды SELECT сравнивалось значение только одного столбца. Если требуется сравнение значений двух или более столбцов, необходимо сложное предложение WHERE с логическими операторами. Многостолбцовые подзапросы позволяют объединять дублируемые условия WHERE в единое предложение WHERE.
Синтаксис:

```
SELECT    column, column, ...
FROM      table
WHERE     (column, column,...) IN
                          (SELECT column, column,...
                           FROM   table
                           WHERE  condition);
```

12.2 Использование многостолбцовых подзапросов.

Пример

Вывод фамилии, номера отдела, оклада и должности всех служащих, оклад и должность которых совпадают как с окладом, так и с должностью одного и того же служащего в отделе 52.

```
SQL> SELECT    dol,  otdel,  oklad, fio
FROM      inspector I,oklad o
WHERE     i.kodinsp=o.kodinsp and (oklad , nvl(dol,'*')) IN
                    (SELECT oklad, nvl(dol,'*')
                    FROM    inspector I,oklad o
                    WHERE   i.kodinsp=o.kodinsp  and otdel = 52) ;
```

Подзапрос в примере является многостолбцовым, т.к. возвращает значения нескольких столбцов. Выводятся фамилии, номера отдела, оклады всех служащих, оклады и должности которых равны как как должности, так и окладу одного и того же служащего в отделе 52. Результаты команды SQL будут выглядеть следующим образом:

DOL	OT	OKLAD	FIO
Инспектор	52	1850	Лютикова З.С.
Инспектор	52	1850	Азарченко Л.М.
Нач. отдела	52	3450	Чабановская В.С.
Нач. отдела	56	3450	Савилов Ю.Н.
Нач. отдела	57	3450	Чайка Н.Ф.

Лабораторная работа 12.2 Применение многостолбцовых подзапросов

Задание:

Напишите запрос, возвращающий информацию об именах, фамилиях, должностях (столбец job_id) и заработной плате сотрудников из таблицы hr.employees. При этом должна возвращаться информация только для сотрудников, для которых установлена минимальная заработная плата для их должности. Информацию о минимальной заработной плате можно получить из таблицы hr.jobs (столбец min_salary). Используйте для решения многострочный подзапрос. Результат выполнения запроса должен выглядеть так, как представлено на рис. Лаб. 7.2.-1.

	Имя	Фамилия	Должность	Оклад
1	Karen	Colmenares	PU_CLERK	2500
2	Martha	Sullivan	SH_CLERK	2500
3	Randall	Perkins	SH_CLERK	2500

Рис. Лаб. 7.2-1

12.3 Сравнения столбцов

12.3.1 Парные и непарные сравнения

Сравнения столбцов в многостолбцовом подзапросе могут быть парными и непарными. В примере в предложении WHERE производилось парное сравнение. Каждая строка-кандидат в команде SELECT должна одновременно иметь и такой же оклад, и такую же должность, как у служащего в отделе 52. Если требуется непарное сравнение (прямое или векторное произведение), необходимо использовать предложение WHERE с несколькими условиями.

12.3.2 Подзапрос с непарным сравнением

Пример

Вывод фамилии, номера отдела, оклада и должности всех служащих, окла и должность которых совпадают с должностью и окладом любого служащего в отделе 52.

```
SQL> SELECT    dol,  otdel,  oklad, fio
FROM      inspector I,oklad o
WHERE     i.kodinsp=o.kodinsp and oklad  IN
                  (SELECT oklad
                  FROM    inspector I,oklad o
                  WHERE   i.kodinsp=o.kodinsp  and otdel = 52)
and nvl(dol,'*') IN
                  (SELECT  nvl(dol,'*')
                  FROM    inspector I,oklad o
                  WHERE   i.kodinsp=o.kodinsp  and otdel = 52)
```

В примере производится непарное сравнение столбцов. Фамилия, номер отдела, оклад и должность всех служащих, чей оклад и должность совпадают с окладом и должностью любого служащего в отделе 52. Результаты команды SQL будут выглядеть следующим образом:

DOL	OTDEL	OKLAD	FIO
Инспектор	52	1850	Лютикова З.С.
Инспектор	52	1850	Азарченко Л.М.
Нач. отдела	52	3450	Чабановская В.С.
Нач. отдела	56	3450	Савилов Ю.Н.
Нач. отдела	57	3450	Чайка Н.Ф.

Результаты двух последних запросов идентичны, несмотря на различие условий сравнения. Они получены для конкретных данных в таблице inspector и oklad.

12.3.3 Неопределенные значения в результатах подзапроса

```
SELECT    dol,  otdel,  oklad, fio
FROM      inspector I,oklad o
WHERE     i.kodinsp=o.kodinsp and (oklad , dol) NOT IN
                  (SELECT oklad, dol
                  FROM    inspector I,oklad o
          WHERE   i.kodinsp=o.kodinsp  and otdel = 57) ;
```

Делается попытка вывода фамилий всех служащих, не имеющих должностей и окладов таких, как у служащих в отделе 57. Логически эта команда SQL должна была бы вернуть строки, но одно из значений, возвращаемых внутренним запросом, является неопределенным значением, и, следовательно, запрос в целом не возвращает ни одной строки.

Это объясняется тем, что все условия, где производится сравнение с неопределенным значением, дают неопределенное значение. Поэтому никогда не используйте оператор NOT IN, если существует вероятность того, что подзапрос вернет неопределенное значение.

Оператор NOT IN эквивалентен !=ALL.

Следует отметить, что если используется оператор IN, то проблем с неопределенными значениями, возвращаемыми подзапросом, не возникает. Оператор IN эквивалентен условию =ANY.

12.3.4 Использование подзапроса в предложении FROM

```
SELECT    dol,  i.otdel,  oklad, fio
FROM      inspector I,oklad o ,  (SELECT avg(oklad) salavg, otdel
                  FROM    inspector a,oklad b
          WHERE   a.kodinsp=b.kodinsp  group by otdel) c
WHERE     i.kodinsp=o.kodinsp and i.otdel=c.otdel and o.oklad > c.salavg;
```

DOL	OTDEL	OKLAD	FIO
Нач. отдела	52	3450	Чабановская В.С.
Нач. отдела	56	3450	Савилов Ю.Н.

Подзапрос можно использовать в предложении FROM команды SELECT, что очень похоже на использование представлений. В вышеприведенном примере выводятся фамилии, оклады, номера отделов и средние оклады для всех служащих, оклад которых превышает средний оклад по отделу.

13. Иерархические запросы

13.1 Иерархические запросы

Если в таблице содержатся иерархические данные, то порядок их выбора может быть определен следующими фразами:

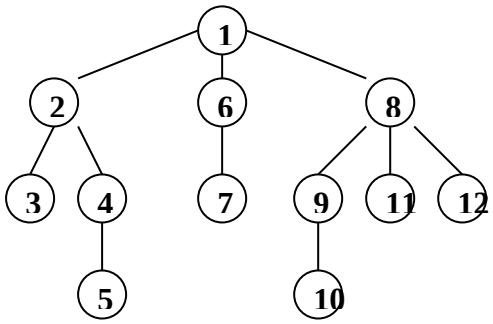
- START WITH - для определения корневой строки (или строк) в иерархии;
- CONNECT BY - для определения отношения между родительскими и дочерними строками в иерархии;
- WHERE - для ограничения строк, возвращаемых запросом.

START WITH	возвращает строки в иерархическом порядке. Определяет, какие строки будут считаться корневыми. Если эта фраза опущена, то Oracle все строки считает корневыми.
CONNECT BY	возвращает строки в иерархическом порядке. Условие должно содержать операцию PRIOR, определяющую отношение между родительскими и дочерними строками. Например, CONNECT BY PRIOR empno = mgr AND sal > comm, будет устанавливать для каждой родительской строки empno набор дочерних строк, для которых значение дочернего mgr эквивалентно значению родительского empno и значение дочернего sal больше значения дочернего comm. Оператор SELECT, выполняющий иерархический запрос, может использовать псевдостолбец LEVEL, который возвращает уровень вложенности: 1 - для корневых строк, 2 - для дочерних 1-го уровня, 3 - для дочерних 2-го уровня и т. д.

При построении запроса, содержащего информацию об иерархии данных, Oracle выполняет следующие действия:

1. Выбирает корневую строку (или строки) иерархии, удовлетворяющие условию START WITH.

2. Выбирает для каждой корневой строки дочерние строки, которые удовлетворяют условию CONNECT BY.
3. Последовательно выбирает наборы дочерних строк: для каждой из дочерних строк, полученных на первом шаге, выбираются свои дочерние строки и т. д.
4. Если в запросе содержится фраза WHERE, то всё строки, которые не удовлетворяют указанному условию, удаляются из иерархии.
5. Возвращает результирующий набор, содержащий строки в порядке, приведенном на следующей схеме:



Отметим, что если оператор SELECT выполняет иерархический запрос, то он не может ни выполнять объединение, ни выбирать данные из вида, выполняющего объединение.
*Пример.*Иерархический запрос, в котором дочернее значение столбца mgr должно соответствовать его родительскому значению столбца kodinsp.

```
SELECT substr(LPAD(' ',2*(LEVEL-1))|| fio,1,40), kodinsp, mgr, dol FROM
inspector
 START WITH dol like 'Нач%' CONNECT BY PRIOR kodinsp = mgr
```

Результатом выполнения этого SQL-оператора в SQL*Plus будет вывод следующих строк:

```
SUBSTR(LPAD(',2*(LEVEL-1))||FIO,1,40)   KODI MGR  DOL
---------------------------------------- ---- ---- ------------------------------
Чабановская В.С.                         5218          Нач.отдела
  Лютикова З.С.                          5205      5218 Инспектор
  Азарченко Л.М.                         5214      5218 Инспектор
Савилов Ю.Н.                             5611          Нач.отдела
  Кудрявцев А.А.                         5605      5611 Инспектор
  Смирнова Т.Н.                          5612      5611 Инспектор
Чайка Н.Ф.                               5712          Нач.отдела
                                         5700      5712 Инспектор
```

Если в качестве условия для данного примера задать
PRIOR kodinsp = mgr AND level < 2,
ограничив уровень иерархии равным 1, то результатом будет вывод следующих строк:

```
SUBSTR(LPAD(',2*(LEVEL-1))||FIO,1,40)   KODI MGR  DOL
---------------------------------------- ------- -------- ----------------
Чабановская В.С.                         5218     Нач.отдела
Савилов Ю.Н.                             5611     Нач.отдела
```

Лабораторная работа 13.1

Задание 1:

Напишите запрос, который бы возвращал из таблицы hr.employees информацию:

- об имени сотрудника;
- о фамилии сотрудника;
- о уровне подчиненности (самый высокий уровень – главный начальник, который никому не подчиняется — уровень 0);
- о пути подотчетности в формате /руководитель 1/руководитель 2/рядовой сотрудник.

Отчет должен быть отсортирован по уровню подчиненности.

Результат выполнения отчета должен выглядеть так, как представлено на рис. Лаб. 8.1-1

	Имя	Фамилия	Уровень подчиненности	Подотчетность
1	Steven	King	0	/Steven King
2	Neena	Kochhar	1	/Steven King/Neena Kochhar
3	Lex	De Haan	1	/Steven King/Lex De Haan
4	Den	Raphaely	1	/Steven King/Den Raphaely
5	Matthew	Weiss	1	/Steven King/Matthew Weiss
6	Adam	Fripp	1	/Steven King/Adam Fripp
7	Payam	Kaufling	1	/Steven King/Payam Kaufling
8	Shanta	Vollman	1	/Steven King/Shanta Vollman
9	Kevin	Mourgos	1	/Steven King/Kevin Mourgos
10	John	Russell	1	/Steven King/John Russell

Рис. Лаб. 8.1-1

13.2 Использование опций UNION, INTERSECT, MINUS в запросе

UNION, INTERSECT, MINUS	Выполняют объединение результатов двух запросов. При этом число столбцов и их тип в каждом запросе должны совпадать. Если запросов больше двух, то они объединяются в пары слева направо. Можно использовать скобки для изменения порядка объединения запросов.

13.2.1 UNION-

Оператор UNION позволяет просто объединить результаты двух запросов в единое целое. При этом по умолчанию оператор UNION отбрасывает повторяющиеся записи (как обычный SELECT с ключевым словом DISTINCT). Чтобы возвращались все записи, включая повторяющиеся, нужно использовать UNION ALL.

Создадим вертикальный список на основе псевдо-таблицы Dual, состоящей из 1 записи:

```
select '1' zn,'Да' otv from dual union select '0' zn,'Нет' otv from dual;
```

```
Z OTV
- ---
0 Нет
1 Да
```

13.2.2 INTERSECT

Этот оператор также выводит результаты двух запросов. Но, в олтичии от UNION, выводятся только те записи, которые возвращают оба запроса.

Определим предприятия, имеющие сальдо по пени,штрафам и по налогу:

```
select distinct n_pred from n_ t where kod=3
intersect
select distinct n_pred from n_ t where kod=1
intersect
select distinct n_pred from n_ t where kod=2;
```

N_PRED

20503

13.2.3 MINUS

Этот оператор возвращает все записи из первого запроса, которые не встречаются в результатах второго запроса.

Определим номера предприятий, имеющих ТОЛЬКО сальдо по пени:

```
select distinct n_pred from n_ t where kod=3
minus
select distinct n_pred from n_ t where kod in (1,2);
```

N_PRED

20501

14. Начало работы с PL/SQL. Блоки PL/SQL. Переменные и типы данных

14.1 Структура блока PL/SQL

PL/SQL - это язык с блочной структурой. Это означает, что программы могут делиться на логические блоки. Блок PL/SQL может содержать до трех секций: декларативная (необязательная), исполняемая (обязательная) и секция обработки исключений (необязательная). Обязательны только слова BEGIN и END. Можно объявлять локальные переменные для блока, который их использует. Обработка ошибок ("исключений") может производиться конкретно внутри блока, к которому они относятся. Можно также хранить и изменять значения внутри блока PL/SQL путем объявления переменных и прочих идентификаторов и ссылок на них.

Следующая таблица описывает три секции блока.

Секция	**Описание**
Декларативная (необязательная)	Содержит все переменные, константы, курсоры и исключения заданные пользователем, ссылка на которые производится в исполняемой секции и секции обработки
Исполняемая (обязательная)	Содержит команды SQL для манипулирования данными в БД и команды PL/SQL для манипулирования данными в
Обработка исключений (необязательная)	Задает действия, которые должны выполняться при возникновении ошибок и аномальных условий в исполняемой секции.

14.2 Структура блока PL/SQL

Выполнение команд и блоков PL/SQL из SQL*Plus

- После команды SQL или управляющей команды PL/SQL ставится точка с запятой (;).
- Для выполнения анонимного блока PL/SQL в буфер SQL помещается косая черта (/).
- Если блок выполнен удачно без необработанных ошибок или ошибок компиляции, на экране должно появиться следующее сообщение:

```
PL/SQL procedure successfully completed
```

- Для закрытия буфера SQL требуется точка (.).
- Блок PL/SQL рассматривается как одно непрерывное предложение в буфере.
- Символы точки с запятой (;) внутри блока не закрывают буфер и не вызывают его выполнение.

Примечание: ошибка в PL/SQL называется исключением (exception).

После ключевых слов DECLARE, BEGIN и EXCEPTION, обозначающих начало секций, символы точки с запятой не ставятся. Но для завершения команды END и всех остальных команд PL/SQL точка с запятой обязательна. На одной строке можно разместить несколько команд, но делать это не рекомендуется, чтобы не усложнять чтение и редактирование.

14.3 Типы блоков

Каждая единица PL/SQL содержит один или несколько блоков. Эти блоки могут быть полностью автономными или вложенными. Основные единицы, составляющие программу PL/SQL (т.е. процедуры и функции, называемые также подпрограммами и анонимными блоками) - это логические блоки, которые могут включать любое количество вложенных подблоков. Следовательно, блок может представлять небольшую часть другого блока, который, в свою очередь, может быть частью целой программы. Из двух типов имеющихся конструкций PL/SQL (анонимных блоков и подпрограмм) в этом курсе изучаются только анонимные блоки.

14.4 Анонимные блоки

Анонимные блоки не имеют имен. Они описываются в той точке приложения, где будут выполняться, и передаются на исполнение ядру PL/SQL во время выполнения приложения. Анонимный блок можно включить в программу предкомпилятора, а также в SQL*Plus или Server Manager. Из таких блоков состоят триггеры в компонентах Developer/2000 (Forms Designer).

14.5 Подпрограммы

Это именованные блоки PL/SQL, которые могут принимать параметры и которые можно вызывать. Объявлять их можно как процедуры или функции. Обычно процедура используется для выполнения действия, а функция - для вычисления и возврата значений. Хранить подпрограммы можно как на сервере, так и в приложении. С помощью компонентов Developer/2000 (Forms, Reports и Graphics) можно объявить процедуры и функции как часть приложения (формы или отчета) и вызывать их по необходимости из других процедур, функций и триггеров (см. следующую страницу) в том же приложении.

Примечание: функция аналогична процедуре с той разницей, что функция должна возвращать значение. Процедуры и функции изучаются в следующем курсе по PL/SQL.

14.6 Программные конструкции

Следующая таблица содержит сводную информацию о разнообразных программных конструкциях, использующих простой блок PL/SQL. Доступность этих конструкций зависит от среды выполнения.

Программная конструкция	**Описание**	**Среда выполнения**
Анонимный блок	Неименованный блок PL/SQL, вставленный в приложение или создаваемый	Все среды PL/SQL
Хранимая процедура или функция	Именованный блок PL/SQL, который хранится в сервере Oracle, может принимать параметры и многократно вызываться по имени. Объединяются в	Сервер Oracle

Процедура или функция приложения	Именованный блок PL/SQL, который хранится в приложении Developer/2000 или в разделяемой библиотеке, может принимать параметры и многократно	Компоненты Developer/2000 (например, Forms)
Пакет	Именованный модуль PL/SQL, объединяющий воедино взаимосвязанные процедуры, функции и идентификаторы. Имеет компоненты (спецификацию и тело	Сервер Oracle и компоненты Developer/2000 (например, Forms)
Триггер базы данных	Блок PL/SQL, связанный с таблицей базы данных и выполняемый автоматически, когда над этой таблицей выполняются	Сервер Oracle
Триггер приложения	Блок PL/SQL, связанный с событием в приложении и выполняемый автоматически.	Компоненты Developer/2000 (например, Forms)

14.7 Использование переменных

С помощью PL/SQL можно объявить переменные, а затем использовать их в командах SQL и процедурных командах в любом месте, где допустимо использование выражений:

- Временное хранение данных

  Данные могут временно храниться в одной или нескольких переменных для использования при проверке входных данных в процессе обработки данных.
- Манипулирование хранимыми значениями

  Переменные могут использоваться для вычислений и прочих манипуляций с данными без обращения к базе данных.
- Возможность повторного использования

  Объявленные переменные могут многократно использоваться в приложении путем ссылки на них в других командах, включая декларативные команды.
- Простота сопровождения

Если используются атрибуты %TYPE и %ROWTYPE (%ROWTYPE обсуждается в одном из следующих уроков), это значит, что вы объявляете переменные на основе определений столбцов базы данных. Переменные PL/SQL или переменные курсора, уже объявленные в пределах текущей области видимости, также могут использовать %TYPE и %ROWTYPE как указатели типов данных. Если определение, на основе которого объявлена переменная, меняется, определение переменной меняется соответственно во время выполнения. Это обеспечивает независимость данных, уменьшает стоимость сопровождения и позволяет программам адаптироваться к изменениям в базе данных, отражающим новые потребности бизнес правил.

14.8 Обработка переменных в PL/SQL

1. Объявление и инициализация переменных в декларативной секции.
2. Присвоение новых значений переменным в исполняемой секции.
3. Передача значений в блоки PL/SQL с помощью параметров.
4. Просмотр результатов с помощью выходных переменных.

1. Переменные объявляются и инициализируются в декларативной секции. Переменные можно объявлять в декларативной части любого блока PL/SQL, подпрограммы или пакета. При объявлении переменной выделяется пространство для хранения значения, задается тип данных переменной и имя места ее хранения, на которое можно ссылаться. При объявлении переменной можно присвоить ей начальное значение и установить ограничение NOT NULL.
2. Новые значения присваиваются переменным в исполняемой секции.

- Существующее значение переменной замещается новым.
- Ссылки вперед недопустимы. Необходимо объявить переменную прежде, чем ссылаться на нее в других командах, включая декларативные.

3. Значения передаются в подпрограммы PL/SQL с помощью параметров. Имеется три типа параметров: IN (по умолчанию), OUT и IN OUT. Параметры IN используются для передачи параметров в вызываемую подпрограмму. Параметры OUT используются для возврата значений в вызывающую программу. Параметр IN OUT используется для передачи начальных значений в вызываемую подпрограмму и возврата обновленных значений в вызывающую программу. Параметры IN и OUT подпрограмм обсуждаются в курсе "Программные единицы PL/SQL".
4. Результаты исполнения блока PL/SQL можно просмотреть через выходные переменные.
Для ввода и вывода в командах манипулирования данными SQL можно использовать ссылочные переменные.

14.8 Типы переменных

1. Переменные PL/SQL
- Скалярные
- Составные
- Ссылочные
- LOB (большие объекты)
2. Прочие переменные (не PL/SQL)
- Связанные переменные и хост - переменные

Все переменные PL/SQL имеют тип данных, определяющий формат хранения, ограничения и диапазон допустимых значений. PL/SQL поддерживает четыре категории типов данных, которые можно использовать при объявлении переменных, констант и указателей.
1. Скалярные типы данных содержат одно значение. Основные типы - это те, которые соответствуют типам столбцов в таблицах сервера Oracle: PL/SQL поддерживает также логические (булевы) переменные.
2. Составные типы (например, RECORD) позволяют определять группы полей и манипулировать ими в блоках PL/SQL. В этом курсе они лишь кратко упоминаются.
3. Ссылочные типы данных содержат значения, называемые указателями (pointers) и обозначающие другие элементы программы. В этом курсе они не изучаются.
4. Типы данных LOB содержат так называемые указатели места (locators), указывающие местоположение больших объектов (например, графических образов), которые хранятся в подпрограммной части программы. В этом курсе они только кратко упоминаются.
Переменные, не являющиеся переменными PL/SQL, включают переменные хост-языка, объявленные в программах предкомпилятора, экранные поля в приложениях Forms и хост-переменные SQL*Plus.
Переменные подстановки в SQL*Plus позволяют хранить части синтаксиса команд и редактировать команды перед выполнением. Это истинные хост-переменные в том смысле, что могут использоваться для передачи числовых или символьных значений в блок PL/SQL и из него во время выполнения. Затем можно ссылаться на них в блоке PL/SQL как на хост-переменные с двоеточием перед именем переменной.

14.9 Типы переменных: примеры

Существуют следующие основные типы данных в переменных.
1. TRUE (“истинно”) - это логическое значение.
2. ‘25-OCT-99’ представляет данные типа DATE.
3. Фотографии представляет данные типа BLOB.
4. Текст речи представляет данные типа LONG RAW.
5. 256120.08 представляет числовые данные типа NUMBER с точностью до двух знаков.
6. Фильм представляет данные типа BFILE.
7. Название города представляет данные типа VARCHAR2.

14.10 Объявление переменных PL/SQL и констант

Синтаксис

```
Identifier  [CONSTANT] datatype [NOT NULL]

      [ := | DEFAULT expr];
```

Примеры

```
Declare
v_hiredate          DATE;
v_deptno            NUMBER(2) NOT NULL := 10;
v_location          VARCHAR2(13) := 'Atlanta';
c_comm.             CONSTANT NUMBER := 1400;
```

Прежде, чем ссылаться на идентификаторы PL/SQL в блоке PL/SQL, необходимо объявить их в декларативной секции. По желанию можно присвоить переменной начальное значение. Для объявления переменной это необязательно. Если при объявлении переменной вы ссылаетесь на другие переменные, они уже должны быть объявлены отдельно в предыдущей команде.

Синтаксис:

identifier	имя переменной или константы
CONSTANT	запрещает изменять значение переменной. Такие переменные, называемые константами, необходимо инициализировать (присваивать им значение).
Datatype	скалярный, составной, ссылочный тип данных или тип данных LOB (в этом курсе обсуждаются только скалярные и составные типы данных).
NOT NULL	означает, что переменная должна иметь определенное значение: переменная, объявленная как NOT NULL, должна быть инициализирована.
Expr	любое выражение PL/SQL. Это может быть литерал, другая переменная или выражение, включающее операторы и функции.

14.11 Указания

- Следуйте правилам присвоения имен.
- Инициализируйте переменные, описанные как NOT NULL.
- Инициализируйте идентификаторы с помощью оператора присваивания или зарезервированного слова DEFAULT.
- Объявляйте не более одного идентификатора на строку кода.

Заданное выражение может быть литералом, другой переменной или выражением, включающим операторы и функции.

1. Имена идентификаторам присваиваются по тем же правилам, что имена объектов SQL.
2. Можно следовать правилам присвоения имен - например, использовать v_name в качестве имени переменной и c_name в качестве имени константы.
3. Переменная инициализируется с помощью оператора присваивания (:=) или зарезервированного слова DEFAULT. Если вы не задаете начальное значение сами, то по умолчанию новая переменная принимает неопределенное значение (NULL), которое сохраняется до тех пор, пока вы не присвоите другое значение позже.
4. Если при объявлении переменной используется ограничение NOT NULL, то инициализировать ее обязательно.
5. Если объявлять по одному идентификатору на строку кода, это облегчает чтение и сопровождение.

6. Если объявляется константа, то указателю типа должно предшествовать слово CONSTANT. В следующем примере объявляется константа типа NUMBER подтипа REAL, и константе присваивается значение 50000. Инициализация константы при объявлении обязательна: в противном случае выдается ошибка компиляции.

```
c_sal   CONSTANT REAL := 50000.00;
```

14.12 Правила присвоения имен

1. Две переменные могут иметь одинаковые имена, если находятся в разных блоках.
2. Имена (идентификаторы) переменных не должны совпадать с именами столбцов таблицы, используемыми в этом блоке.

Два объекта могут иметь одинаковые имена, если эти объекты объявлены в различных блоках. Если такое дублирование имен имеет место, то может использоваться только объект, объявленный в текущем блоке.

Имя переменной не должно совпадать с именами столбцов таблицы, используемых в блоке. Если переменные PL/SQL используются в командах SQL и их имя совпадает с именем столбца, то сервер Oracle предполагает, что это - ссылка на столбец. Хотя код, приведенный в качестве примера на слайде, работает, совпадение имени таблицы с именем переменной затрудняет его чтение и сопровождение.

Используйте правила присвоения имен, как в следующем примере. Если использовать префикс “v_” для имени переменной и “g_” для имени глобальной переменной, это предотвращает конфликты с именами объектов базы данных.

```
DECLARE

v_hiredate          Date
g_deptno            NUMBER(2) NOT NULL := 10;
BEGIN
```

Примечание: длина идентификатора не должна превышать 30 символов, первый символ должен быть алфавитным: в качестве остальных символов могут использоваться буквы, цифры и специальные символы.

14.13 Присвоение значений переменным

Синтаксис

```
identifier := expr;
```

Примеры

Установка определенной даты найма для новых служащих.

```
v_hiredate := ’31-DEC-98’;
```

Установка имени служащего "Maduro."

```
v_ename := ‘Maduro’;
```

Для присваивания и отмены значения переменной используется команда присваивания PL/SQL. Переменная, которой присваивается значение, должна быть явно задана слева от оператора присваивания (:=).

Синтаксис:

Identifier	имя скалярной переменной.
Expr	может быть переменной, литералом или вызовом функции, но не столбцом базы данных.

Еще один способ присвоения значений переменным - это выборка значений базы данных в эти переменные. В следующем примере Oracle вычисляет 10-процентную премию, когда вы выбираете оклад служащего.

```
SQL>  SELECT    oklad * 0.10
  2   INTO      bonus
  3   FROM      oklad
  4   WHERE     kodinsp = 5611;
```

Переменную bonus можно затем использовать в других вычислениях или вставить ее значение в таблицу базы данных.

Примечание: при работе с явными курсорами для записи значения из базы данных в переменную в блоке PL/SQL используется команда SELECT или FETCH.

14.14 Инициализация переменных и ключевые слова

Операторы и ключевые слова:

1. Оператор присваивания ":="
2. DEFAULT
3. NOT NULL

Переменные инициализируются каждый раз при входе в блок или подпрограмму. По умолчанию переменные принимают неопределенное значение (NULL). Если вы не задаете начальное значение явно, то значение переменной считается неопределенным.

Для переменных, не имеющих стандартного значения, используйте оператор присваивания (:=).

```
v_hiredate := TO_DATE('15-SEP-99', 'DD-MON-YY');
```

Т.к. формат даты по умолчанию, установленный в сервере Oracle, в разных базах данных может быть разным, вы можете пожелать присвоить стандартные значения дат, как в предыдущем примере.

1. DEFAULT: для инициализации переменных можно использовать слово DEFAULT вместо оператора присваивания. Используйте DEFAULT для переменных, имеющих типичные значения.

```
g_mgr NUMBER(4) DEFAULT 7839;
```

2. NOT NULL: если переменная должна содержать значение, задайте ограничение NOT NULL. Присвоить неопределенное значение переменной, имеющей ограничение NOT NULL, невозможно. За предложением NOT NULL обязательно должно следовать предложение инициализации переменной; иначе возникает ошибка компиляции.

```
v_location VARCHAR2(13) NOT NULL := 'CHICAGO';
```

Примечание: литеральные строки должны быть заключены в апострофы – например, 'Hello, world'. Если апостроф содержится в самой строке символов, он должен указываться дважды - например, 'Account wasn''t found'.

14.15 Скалярные типы данных

Элемент данных скалярного типа:

- Содержит только одно значение
- Не имеет внутренних компонентов

Скалярный тип данных содержит одно значение и не имеет внутренних компонентов. Скалярные типы данных делятся на четыре категории: числовые, символьные данные, даты и логические данные. Символьный и числовой типы данных имеют подтипы, которые налагают определенные ограничения на базовый тип. Например, INTEGER (“целое число") и POSITIVE ("положительное число") являются подтипами базового типа NUMBER.

14.16 Основные скалярные типы данных

- VARCHAR2(maximum_length)
- NUMBER [(precision, scale)]
- DATE
- CHAR [(maximum_length)]
- LONG
- LONG RAW
- BOOLEAN
- BINARY_INTEGER
- PLS_INTEGER

Тип данных	**Описание**
VARCHAR2 (maximum_length)	Основной тип для символьных данных переменной длины до 32767 байтов. Для переменных VARCHAR2 и констант размера по умолчанию не существует.
NUMBER [(precision, scale)]	Основной тип данных для чисел с фиксированной и плавающей запятой.
DATE	Основной тип для дат и времени. Значения DATE включают время в секундах с полуночи. Диапазон дат: 4712 г. до н. э. - 9999 н. э.
CHAR [(maximum_length)]	Основной тип для символьных данных постоянной длины до 32767 байтов. Если макс, длина (maximum_length) не задана, то по умолчанию она равна 1.
LONG	Основной тип для символьных данных переменной длины до 32760 байтов. Макс, ширина столбца LONG - 2147483647 байтов.
LONG RAW	Основной тип для двоичных данных и строк байтов длиной до 32760 байтов. PL/SQL данные LONG RAW не интерпретирует.
BOOLEAN	Основной тип для хранения значений, используемых в логических вычислениях: TRUE (истинно), FALSE (ложно) или NULL (не определено).
BINARY_INTEGER	Основной тип для целых чисел от -2147483647 до 2147483647.
PLS_INTEGER	Основной тип для целых чисел со знаком от -2147483647 до 2147483647. Значения PLS INTEGER требуют меньше памяти и быстрее значений NUMBER и BINARYJNTEGER

Примечание: Тип данных LONG аналогичен типу VARCHAR2 с той разницей, что максимальная, длина значения LONG равна 32760 байтам. Следовательно, выборка значений длиной более 32760 байтов из столбца базы данных LONG в переменную LONG PL/SQL невозможна.

14.17 Объявление скалярных переменных: примеры

```
v_job          VARCHAR2(9);
```

v_count	BINARY INTEGER := 0;
v_total sal	NUMBER(9,2) := 0;
v_orderdate	DATE := SYSDATE + 7;
c_tax_rate	CONSTANT NUMBER(3,2) := 8.25;
v_valid	BOOLEAN NOT NULL := TRUE;

Описание примеров объявления переменных :
- Объявление переменной для хранения должности служащего.
- Объявление переменной для хранения счетчика циклов и присвоение ей начального значения 0.
- Объявление переменной, в которой будет накапливаться общая заработная плата отдела, и присвоение ей начального значения 0.
- Объявление переменной для хранения даты отгрузки заказа и присвоение ей начального значения - неделя с сегодняшней даты.
- Объявление идентификатора с постоянным значением для хранения ставки налога, которая для данного блока PL/SQL не изменится.

- Объявление переменной-индикатора. Индикатор будет показывать, действителен ли элемент данных. Начальное значение переменной - TRUE ("истинно").

Лабораторная работа 14.1 Работа с переменными

Задание:

Используя возможности PL/SQL, модифицируйте команду на вставку записи для нового сотрудника, созданный на предыдущих лабораторных (см. ниже) таким образом, чтобы:

- для добавляемых значений использовались переменные;
- переменным присваивались значения при помощи диалоговых окон;
- информация о должности сотрудника запрашивалась только один раз;
- производилась фиксация транзакции.

Просмотреть информацию о структуре таблицы hr.employees можно при помощи команды DESCRIBE или графическими средствами Oracle SQL Developer.

Исходная команда:

```
insert into hr.employees values(employees_seq.nextval,
'&Имя_сотрудника', '&Фамилия_сотрудника',
'&Адрес_электронной_почты', '&Номер_телефона' ,sysdate,
'&Должность_сотрудника', (SELECT MIN(SALARY) from hr.employees
where job_id='&Должность_сотрудника'), null, null, null)
```

14.18 Атрибут %TYPE

1.Используется для объявления переменной:
- На основе столбца базы данных
- На основе уже объявленной переменной

2.Перед %TYPE указываются:
- Имена таблицы и столбца базы данных
- Имя уже объявленной переменной

Объявляя переменные PL/SQL для хранения значений столбцов, необходимо удостовериться в том, что переменные имеют правильный тип данных и точность. В противном случае при выполнении возникнет ошибка PL/SQL. Вместо того, чтобы жестко кодировать тип данных и точность переменной, можно использовать атрибут %TYPE, который позволяет объявить переменную на основе уже объявленной переменной или столбца базы данных. Атрибут %TYPE обычно используется в случае, если значение, которое запоминается в переменной, выбирается из таблицы БД или предполагается запись значения переменной в БД. Чтобы использовать этот атрибут вместо указания типа данных при объявлении переменной,

укажите перед атрибутом имя таблицы базы данных и имя столбца. Если вы ссылаетесь на уже объявленную переменную, укажите перед атрибутом имя этой переменной.
PL/SQL определяет тип данных и размер переменной во время компиляции блока, чтобы переменная всегда была совместима со столбцом, данными которого она заполняется. Это определенное преимущество для написания и сопровождения кода, т.к. не приходится думать об изменениях типов данных в столбцах, сделанных на уровне БД. Можно также объявить переменную на основе уже объявленной переменной. Для этого следует указать имя переменной в качестве префикса перед атрибутом.

14.19 Объявление переменных с атрибутом %TYPE: примеры

```
...
v_ename             emp.ename%TYPE;
v_balance           NUMBER(7,2);
v_min_balance       v_balance%TYPE := 10;
...
```

Объявление переменных для хранения имени служащего:

```
...
v_ename             emp.ename%TYPE;
...
```

Объявление переменных для хранения остатка на банковском счете, а также минимального остатка, начальное значение которого -10:

```
...
v_balance           NUMBER(7,2);
v_min_balance       v_balance%TYPE := 10;

...
```

Ограничение NOT NULL, применяемое к столбцам, не распространяется на переменные, объявленные с помощью атрибута %TYPE. Следовательно, объявляя переменную с помощью атрибута %TYPE и используя при этом столбец, имеющий ограничение NOT NULL, можно присвоить переменной неопределенное значение (NULL).

Лабораторная работа 14.2 Объявление переменных при помощи %TYPE

Ситуация:

Типы данных в таблице hr.employees могут изменяться разработчиками. Вы хотите, чтобы типы переменных в ваших программных блоках PL/SQL изменялись автоматически при внесении изменений в типы данных для столбцов таблицы.

Задание:

1. Измените программный код PL/SQL, созданный на предыдущей лабораторной работе, таким образом, чтобы типы данных для переменных назначались автоматически на основе соответствующих типов данных из столбцов таблицы hr.employees.
2. Присвоение значений переменным должно происходить внутри блока BEGIN...END.

14.20 Объявление логических (булевых) переменных

1. Булевой переменной может быть присвоено только значение TRUE (ИСТИННО), FALSE (ЛОЖНО) или NULL (НЕ ОПРЕДЕЛЕНО).
2. Переменные связываются логическими операторами AND, OR и NOT.
3. Переменные всегда возвращают значение TRUE, FALSE или NULL.

4. Возвращать булево значение могут арифметические, символьные выражения и выражения даты.

PL/SQL позволяет сравнивать переменные как в командах SQL, так и в процедурных командах. Эти сравнения, называемые логическими или булевыми выражениями, состоят из простых или сложных выражений, связанных логическими операторами. В команде SQL с помощью логических выражений можно задать строки таблицы, на которые распространяется команда. В процедурной команде логические выражения являются основой для управления по условию.

NULL означает отсутствующее, неприменимое или неизвестное значение.

Примеры

```
v_sal1 := 50000;

v_sal2 := 60000;
```

Следующее выражение дает результат TRUE ("истинно").

```
v_sal1 < v_sal2
```

Объявление и инициализация логической переменной.

```
v_comm_sal BOOLEAN := (v_sal1 < v_sal2)
```

14.21 Составные типы данных

Типы:

- Таблицы PL/SQL (TABLE)
- Записи PL/SQL (RECORD)

К составным типам данных, называемым также коллекциями (collection), относятся TABLE , RECORD , Nested TABLE и VARRAY. Тип данных RECORD ("запись") позволяет рассматривать взаимосвязанные, но различные элементы данных как единую логическую единицу. Тип данных TABLE позволяет ссылаться на коллекции данных и манипулировать ими как целым объектом. Типы данных RECORD и TABLE подробно обсуждаются в следующих уроках. Типы данных Nested TABLE и VARRAY в этом курсе не обсуждаются.

14.22 Переменные типа LOB

Типы данных LOB ("большой объект") сервера Oracle8-9 позволяют хранить блоки неструктурированных данных размером до 4 гигабайтов (текст, графические образы, видеоклипы, формы звуковых воли и т.д.). Типы данных LOB обеспечивают эффективный, произвольный и выборочный доступ к данным и могут быть атрибутами типа объекта. Типы данных LOB обеспечивают также произвольный доступ к данным.

1. Тип данных CLOB ("большой символьный объект") используется для хранения больших блоков однобайтных символьных данных в базе данных.
2. Тип данных BLOB ("большой двоичный объект") используется для хранения в базе данных больших двоичных объектов, как в строке кода основной программы (in line), так и в подпрограммах (out of line).
3. Тип данных BFILE ("двоичный файл") используется для хранения больших двоичных объектов в файлах операционной системы вне базы данных.
4. Тип данных NCLOB ("большой символьный объект с национальными символами") используется для хранения в базе данных больших блоков однобайтных данных или многобайтных данных NCHAR фиксированной ширины, как в строке кода основной программы (*in line*), так и в подпрограммах *(out of line).*

14.23 Связанные переменные

Связанная переменная (*bind variable*) - это переменная, которая объявляется в хост- среде, а затем используется для динамической передачи параметров во время выполнения в одну или несколько программ PL/SQL или из них (в данном курсе термины "связанная переменная" и

"хост-переменная" взаимозаменяемы). Эта переменная может представлять собой число или символ и использоваться программами PL/SQL, как любая другая переменная. Вы можете ссылаться на переменные, объявленные в вызывающей среде (хост-среде), за исключением случая, когда команда является частью процедуры, функции или пакета. Это включает переменные хост-языка, объявленные в программах предкомпилятора, экранные поля в приложениях Developer/2000 Forms и связанные переменные SQL*Plus.

14.24 Создание связанных переменных

Для объявления связанной переменной в среде SQL*Plus используется команда VARIABLE. Например, переменная типа NUMBER объявляется так:

```
VARIABLE return_code NUMBER
```

Ссылаться на связанную переменную могут как SQL, так и SQL*Plus, а SQL*Plus может также вывести значение переменной.

14.25 Вывод значений связанных переменных

В среде SQL*Plus для вывода текущих значений связанных переменных используется команда PRINT. Например:

```
SQL>  VARIABLE          n
...
SQL>  PRINT n
```

Ссылки на переменные, не являющиеся переменными PL/SQL

Сохранение годовой заработной платы в хост-переменной SQL*Plus.

```
:g_monthly_sal := v_sal / 12;
```

1. **Ссылка на переменную, не являющуюся переменной PL/SQL, как на хост - переменную.**
2. **Ссылке предшествует двоеточие (:).**

14.26 Присваивание значений переменным

Различать хост-переменные и переменные, объявленные в PL/SQL, можно по ссылке на хост-переменную. Перед ссылкой на хост -переменную ставится двоеточие (:).

Примеры

```
:host_var1 := v_sal;
:global_var1 := 'YES';
```

Итоги

- **Идентификаторы PL/SQL:**
  - **Определяются в декларативной секции**
  - **Могут быть скалярными, составными, ссылочными или типа данных LOB**
  - **Могут быть объявлены на основе другой переменной или другого объекта базы данных**

- **Могут инициализироваться**