DML

Оглавление

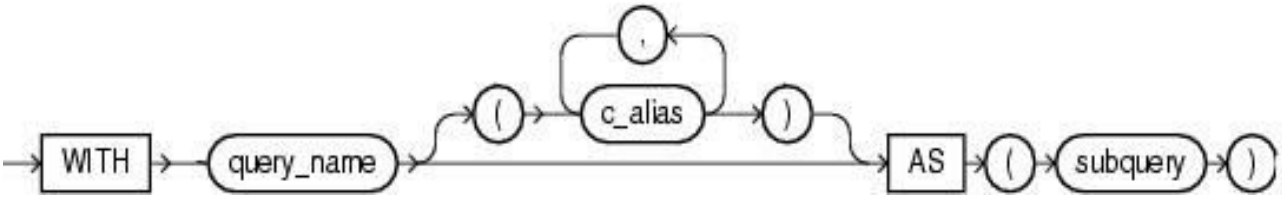
DML

- Таблица DUAL
- Конструкция with
- Иерархические запросы (self joins)
- Псевдостолбцы (Pseudocolumns)
  - ORA_ROWSCN
  - ROWID
  - ROWNUM
- Конструкция SAMPLE
- PIVOT
- UNPIVOT
- INSERT
- UPDATE
- DELETE
- MERGE
- Обновляемая VIEW
- SQL Functions

Таблица DUAL

- ✓ DUAL - одна из таблиц словаря данных
- ✓ Все пользователи базы данных имеют доступ к таблице DUAL
- ✓ Содержит одно поле «Dummy» и одну запись со значением “X” в этом поле
- ✓ Используется для получения результата какого-либо выражения (функции) с помощью оператора SELECT
- ✓ Начиная с 10 версии Oracle при запросе к этой таблице Oracle не выполняет физического или логического чтения. В плане выполнения это отображается как FAST DUAL. Но это только в случае получения значения какого-либо выражения с помощью запроса к этой таблице. Если выбирать значение поля DUMMY этой таблицы, логическое чтение происходит.

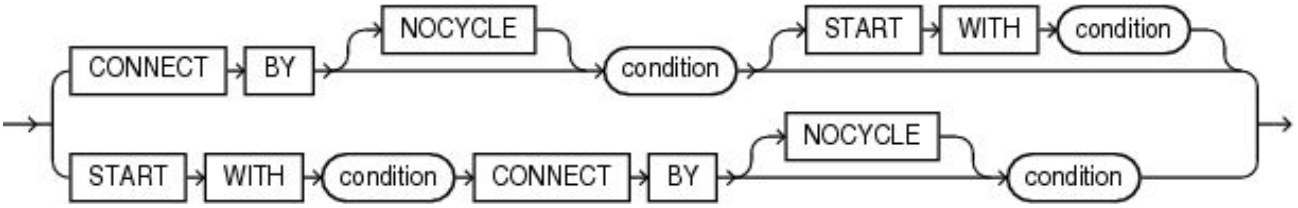
Конструкция with



Позволяет создать именованный подзапрос и использовать его далее в основном запросе несколько раз

- ✓ Oracle при этом будет воспринимать этот подзапрос как вьюху или временную таблицу
- ✓ Имя нашего именованного подзапроса видно в любом месте основного запроса
- ✓ Именованный подзапрос можно сделать рекурсивным, в этом случае он извлекает данные из самого себя – такой подзапрос содержит в себе два блока: якорный блок и рекурсивный блок

Иерархические запросы (self joins)



Выводит записи в иерархическом порядке (соединяет таблицу саму с собой).

Условие **start with** определяет корневую запись(записи) иерархии

Условие **connect by** определяет отношения между родительской и дочерней записями в иерархии. При этом родительская запись задается ключевым словом **PRIOR**

NOCYCLE - возвращает результат запроса даже если есть замкнутые циклы

Oracle выполняет такие запросы в следующей последовательности:

1. Сначала выбираются корневые записи, удовлетворяющие условию **start with**
2. Далее для каждой корневой записи извлекаются ее дочерние записи согласно условию **connect by**
3. После, для дочерних записей выбираются их дочерние записи и так далее
4. Если есть условия во where секции, записи, не удовлетворяющие условию, удаляются из выборки. Каждая запись проверяется индивидуально. Т.е. допустима

ситуация, когда запись удалена из выборки, а ее дочерние записи в выборке остались

5. Возвращается набор записей в определенном порядке: дочерние записи следуют сразу за родительскими

Псевдоколонки, относящиеся к иерархическим запросам:

- **level** - возвращает 1 для корневых записей, 2 для дочерних от корневых, 3 для дочерних следующего уровня и т.д. (т.е. определяет уровень записи в иерархии)
- **connect_by_iscycle** - возвращает 1, если текущая запись имеет дочернюю, которая также является для нее родительской (т.е. запись в «иерархическом цикле»). Иначе возвращает 0.
- **connect_by_isleaf** - возвращает 1, если запись является листом иерархического дерева (т.е. если запись не имеет дочерних). Иначе возвращает 0.

Функции, относящиеся к иерархическим запросам:

- **sys_connect_by_path** возвращает путь к текущей записи от корневой
- **connect_by_root** возвращает корневую запись для текущей

Restrictions:

✓ В этих запросах нельзя использовать group by и order by. Вместо order by используется структура **order siblings by**

Псевдостолбцы (Pseudocolumns)

ORA_ROWSCN

Возвращает scn (system change number) последнего изменения записи. Это может быть изменение, относящееся к блоку или относящееся к записи, в зависимости от параметров создания таблицы. Ora_rowscn не обязательно возвращает конкретный номер scn последнего изменения записи. Возвращаемый номер может быть больше, чем scn последнего изменения записи (но никогда меньше).

Функции, работающие с ora_rowscn:

- **scn_to_timestamp** - возвращает timestamp по переданному номеру scn
- **timestamp_to_scn** – возвращает номер scn по переданному timestamp

Restrictions:

- не поддерживается при запросах external tables
- не поддерживается при запросах к view

ROWID

Возвращает адрес строки, состоит из:

- object_id (dba_objects.object_id) таблицы
- Блок данных в файле
- Позиция строки в блоке данных (первая строка – это 0)

- Номер файла данных по отношению к Tablespace (нумеруются с 1)

Rowid уникально идентифицирует строку в базе данных, однако строки в разных таблицах хранящиеся вместе в одном кластере могут иметь одинаковый rowid.

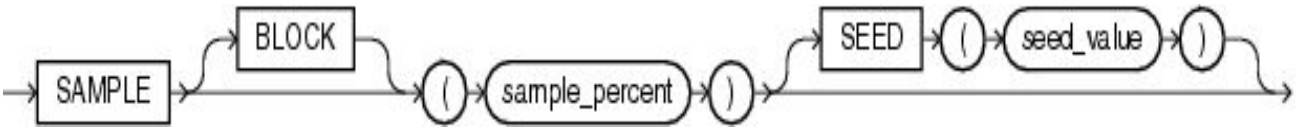
Доступ по rowid – самый быстрый способ извлечь строку

С помощью значений rowid и пакета dbms_rowid можно анализировать, как хранятся строки.

ROWNUM

Возвращает номер строки в результате запроса в том порядке, в котором строки возвращает Oracle. Нумерация начинается с 1.

Конструкция SAMPLE



Позволяет извлечь данные из случайной выборки(части) таблицы, а не из всей таблицы (т.е. позволяет получить «примерный» результат)

BLOCK - random block sampling instead of random row sampling – извлекает данные из произвольного количества блоков, а не строк

sample_percent – процент блоков/строк, из которых будут извлекаться данные

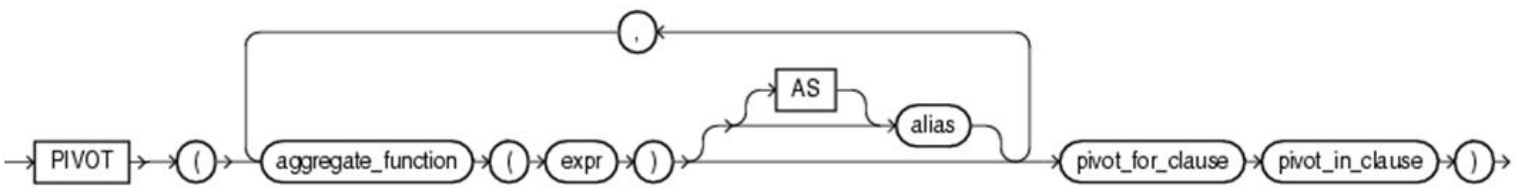
SEED (seed_value) – целое число (integer) - позволяет закрепить используемую выборку (sample) для всех выполнений запроса

Restrictions:

- Нельзя использовать в подзапросе
- Работает не на всех представлениях (только на key preserving view)

PIVOT

Переводит строки в столбцы и агрегирует данные в процессе такого преобразования.

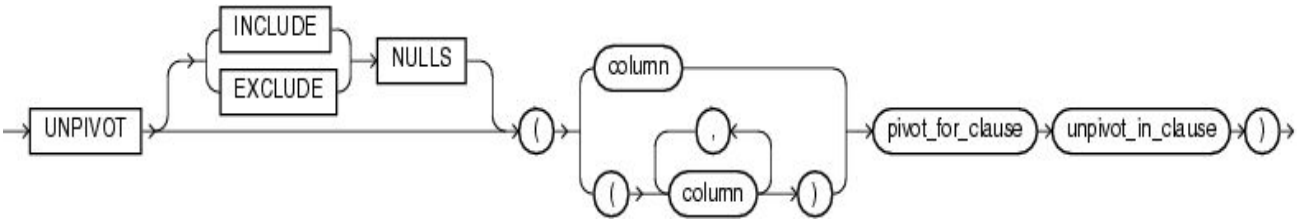


Выполняет неявный group by по полям, не упомянутым в pivot_for_clause, со значениями в pivot_in_clause. Результат PIVOT операции обычно

возвращает набор данных, в котором меньше строк, но больше столбцов, чем в исходном наборе данных.

UNPIVOT

Преобразует столбцы в строки



INSERT

Вставляет записи в таблицу

Restrictions:

- Необходима привилегия INSERT на таблицу (или INSERT ANY TABLE)
- Если таблица находится на удаленной базе данных, для вставки в нее необходима также привилегия select на эту таблицу

Примеры синтаксиса:

1. Простейшая форма оператора

```
INSERT INTO hr.departments
   VALUES (280, 'Recreation', default, 1700);
```

2. Полная форма оператора

```
INSERT INTO hr.employees (employee_id, last_name, email,
      hire_date, job_id, salary, commission_pct)
   VALUES (207, 'Gregory', 'pgregory@example.com',
      sysdate, 'PU_CLERK', 1.2E3, NULL);
```

3. Insert select

```
INSERT INTO hr.bonuses
   SELECT employee_id, salary*1.1
   FROM hr.employees
   WHERE commission_pct > 0.25;
```

4. Конструкция returning

```
INSERT INTO employees
      (employee_id, last_name, email, hire_date, job_id, salary)
   VALUES
   (employees_seq.nextval, 'Doe', 'john.doe@example.com',
       SYSDATE, 'SH_CLERK', 2400)
   RETURNING employee_id, job_id INTO :bnd1, :bnd2;
```

Оператор INSERT также позволяет вставлять записи в несколько таблиц сразу. Особенности синтаксиса такой формы INSERT описаны здесь.

Пример синтаксиса такого оператора:

```
INSERT ALL
   WHEN order_total <= 100000 THEN
      INTO small_orders
   WHEN order_total > 100000 AND order_total <= 200000 THEN
      INTO medium_orders
   WHEN order_total > 200000 THEN
      INTO large_orders
   SELECT order_id, order_total, sales_rep_id, customer_id
      FROM orders;
```

UPDATE

Меняет значения в полях записей таблицы.

Restrictions:

- Необходима привилегия UPDATE на таблицу (или UPDATE ANY TABLE)
- Если таблица находится на удаленной базе данных, для изменения ее записей необходима также привилегия select на эту таблицу

Примеры синтаксиса:

1. Простейшая форма оператора

```
UPDATE hr.employees SET
    job_id = 'SA_MAN', salary = salary + 1000, department_id = 120
    WHERE first_name||' '||last_name = 'Douglas Grant';
```

2. Update с подзапросами

```
UPDATE hr.employees a
    SET department_id =
        (SELECT department_id
            FROM hr.departments
            WHERE location_id = '2100'),
        (salary, commission_pct) =
        (SELECT 1.1*AVG(salary), 1.5*AVG(commission_pct)
          FROM hr.employees b
          WHERE a.department_id = b.department_id)
    WHERE department_id IN
        (SELECT department_id
          FROM hr.departments
          WHERE location_id = 2900
              OR location_id = 2700);
```

3. Конструкция RETURNING (обновление одной записи)

```
UPDATE hr.employees
  SET job_id ='SA_MAN', salary = salary + 1000, department_id = 140
  WHERE last_name = 'Jones'
  RETURNING salary*0.25, last_name, department_id
    INTO :bnd1, :bnd2, :bnd3;
```

4. Конструкция RETURNING (обновление нескольких записей)

```
UPDATE hr.employees
   SET salary = salary * 1.1
   WHERE department_id = 100
   RETURNING SUM(salary) INTO :bnd1;
```

DELETE

Удаляет записи из таблицы

Restrictions:

- Необходима привилегия DELETE на таблицу (или DELETE ANY TABLE)
- Если таблица находится на удаленной базе данных, для удаления ее записей необходима также привилегия select на эту таблицу

Примеры синтаксиса:

1. Простейшая форма оператора

```
DELETE FROM hr.employees
   WHERE job_id = 'SA_REP'
   AND commission_pct < .2;
```

2. Конструкция RETURNING

```
DELETE FROM hr.employees
   WHERE job_id = 'SA_REP'
   AND hire_date + TO_YMINTERVAL('01-00') < SYSDATE
   RETURNING salary INTO :bnd1;
```

MERGE

Позволяет сделать выборку данных из одного или нескольких источников чтобы изменить или вставить данные в таблицу/представление

Restrictions:

- Требуются привилегии INSERT и UPDATE на целевые таблицы и SELECT на таблицу-источник.
- Если Merge включает в себя условие delete, необходима также привилегия DELETE на целевую таблицу.
- Привилегии INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE и SELECT ANY TABLE тоже подойдут.

Пример синтаксиса:

```
MERGE INTO hr.bonuses D
   USING (SELECT employee_id, salary, department_id FROM hr.employees
   WHERE department_id = 80) S
   ON (D.employee_id = S.employee_id)
   WHEN MATCHED THEN UPDATE SET D.bonus = D.bonus + S.salary*.01
     DELETE WHERE (S.salary > 8000)
   WHEN NOT MATCHED THEN INSERT (D.employee_id, D.bonus)
     VALUES (S.employee_id, S.salary*.01)
     WHERE (S.salary <= 8000);
```

Обновляемая VIEW

Чтобы view была обновляемая (т.е., чтобы к этой view можно было применять операторы insert, update или delete), в ней **не** должны использоваться следующие возможности и конструкции:

1. Set операторы
2. Distinct оператор
3. Агрегатные или аналитические функции
4. GROUP BY, ORDER BY, MODEL, CONNECT BY, START WITH
5. Коллекция в select-списке
6. Подзапрос в select-списке
7. Подзапрос с WITH READ ONLY
8. Joins (с некоторыми исключениями)

Если joins все таки есть, необходимо наличие одной базовой таблицы (key preserved view).
Key-preserved table – такая таблица в представлении (view), ключ которой также является ключом результата join-а (при этом ключи не обязательно извлекать из таблицы в select-списке)

Restrinctions:

1. Любой insert, update или delete изменяет только одну базовую таблицу из view
2. Можно обновлять только поля key-preserved table. Если key-preserved table участвует в представлении (view) несколько раз и объявлено условие WITH CHECK OPTION, такую view обновлять нельзя
3. Удалять записи разрешено только из той view, в которой есть только одна key-preserved table. Если объявлено условие WITH CHECK OPTION и key-preserved table участвует в запросе несколько раз, удалять записи из такой view запрещено
4. В выражении INSERT могут участвовать только поля key-preserved table. Если объявлено условия WITH CHECK OPTION, вставлять в такую view запрещено.

SQL Functions

Datetime functions

Функция	Описание
Sysdate	Возвращает текущую дату, установленную в операционной системе, в которой запущена БД
Extract(year/month/day/hour/minute/second.. from datetime)	Извлекает из даты месяц, год, день, час и т.п.
Add_months(date, integer)	Добавляет к дате date количество месяцев в integer
Months_between(date1, date2)	Возвращает количество месяцев между датами date1 и date2 (считает за месяц 31 день)

Character functions

Функция	Описание
Lower(char)	Возвращает строку в нижнем регистре
Upper(char)	Возвращает строку в верхнем регистре
Length(char)	Возвращает длину строки
Substr(char, position, length)	Возвращает подстроку из строки char начиная с позиции position длинны length. если последний параметр не указан, то возвращается подстрока с position до конца строки. Position может быть отрицательным. 0 заменяется на 1.
Instr(string, substring, positions, occurrence)	Ищет подстроку substring в строке string и возвращает номер

	позиции найденной подстроки (или 0, если не найдено вхождение). Если указан параметр position – ищет, начиная с этой позиции. Если необходимо найти не первое вхождение, указываем параметр occurrence.
Replace(char, search_string, relpacement_string)	В строке char заменяет все вхождения search_string на replacement_string (если replacement_string не указано, просто удаляет вхождения search_string)
Ltrim(char, set)	Удаляет в строке char слева все пробелы (или символы, указанные в параметре set)
Rtrim(char, set)	Удаляет в строке char справа все пробелы (или символы, указанные в параметре set)
Trim(LEADING|TRAILING|BOTH trim_char FROM char)	Удаляет слева|справа|с обоих сторон все пробелы (или символы trim_char) в строке char
Lpad(expr1, n, expr2)	Длину строки expr1 доводит до n-символов, заполняя слева пробелами или символами expr2 (если указываем этот параметр)
Rpad(expr1, n, expr2)	Длину строки expr1 доводит до n-символов, заполняя справа пробелами или символами expr2 (если указываем этот параметр)

Conversion functions

Функция	Описание
To_char(arg, frmt)	Преобразует аргумент (nchar, nvarchar2, clob, nclob, date, number) к типу varchar2. Формат преобразования задается параметром frmt
To_date(char, frmt)	Преобразует строку к дате, формат даты в строке задается параметром frmt
To_number(expr, frmt)	Преобразует выражение expr (binary_double или строковый тип) к типу number. Формат числа в expr задается параметром frmt (при необходимости)

Numeric functions

Функция	Описание
Abs(n)	Возвращает абсолютное значение n
Ceil(n)	Округляет в сторону большего целого значения

Floor(n)	Округляет в сторону меньшего целого значения
Power(n1, n2)	Возводит n1 в степень n2
Round(n1)	Округляет до ближайшего целого значения

Other functions

Функция	Описание
User	Возвращает имя пользователя, под которым создали соединение с базой данных
trunc(date)/trunc(number)	Обрезает переданное значение
Greatest(...)	Возвращает максимальное значение из всех переданных аргументов. Определяет тип возвращаемого значения по первому аргументу
Least(...)	Возвращает минимальное значение из всех переданных аргументов. Определяет тип возвращаемого значения по первому аргументу