Базы данных и Информационные системы

13 / 15 Индексы, Последовательности, Триггеры

Кузиков Б.О.
Сумы, СумГУ
2013

Задачи занятия

- После завершения занятия вы должны уметь и знать следующее:
  - Свойства и использование объектов базы данных
  - Создавать, поддерживать и использовать последовательности
  - Создавать и поддерживать индексы
  - Создавать приватные и общедоступные синонимы

Объекты в БД

Объект	Описание
Table	Таблица. Состоит из строк и столбцов
View	Логически представляет подмножество данных из одной или нескольких таблиц
Sequence	Генерирует значения первичных ключей
Index	Увеличивает скорость работы
Synonym	Альтернативное имя объекта

Что есть последовательность?

- Автоматически генерирует уникальные числа
- Является разделяемым объектом
- Обычно используется для формирования значений первичных ключей
- Заменяет дополнительный код
- Увеличивает эффективность доступа к элементам последовательности при кэшировании последовательности в памяти

Выражение CREATE SEQUENCE

- Определяет последовательность, автоматически генерирующую последовательные числа

```
CREATE SEQUENCE sequence
          [INCREMENT BY n]
          [START WITH n]
          [{MAXVALUE n | NOMAXVALUE}]
          [{MINVALUE n | NOMINVALUE}]
          [{CYCLE | NOCYCLE}]
          [{CACHE n | NOCACHE}];
```

Создание последовательности

- Создание последовательности DEPT_DEPTNO, которая будет использоваться для формирования первичных ключей таблицы DEPT.
- Параметр CYCLE не используется.

```
SQL> CREATE SEQUENCE dept_deptno
  2       INCREMENT BY 1
  3       START WITH 91
  4       MAXVALUE 100
  5       NOCACHE
  6       NOCYCLE;
Sequence created.
```

Использование последовательностей

- Добавление нового подразделения “MARKETING” в Сан-Диего.

```
SQL> INSERT INTO dept(deptno, dname, loc)
  2  VALUES (dept_deptno.NEXTVAL,
  3        'MARKETING', 'SAN DIEGO');
1 row created.
```

- Просмотр текущего значения последовательности DEPT_DEPTNO.

```
SQL> SELECT   dept_deptno.CURRVAL
  2  FROM dual;
```

Использование последовательностей

- Кэширование значений последовательности в памяти позволяет увеличить скорость доступа к этим значениям.
- Пропуски в последовательностях возникают при:
  - Откатах транзакций
  - Системных сбоях
  - Использовании последовательности в нескольких таблицах
- Если последовательность была создана с параметром NOCACHE, ее следующее значение можно увидеть, сформировав запрос к таблице USER_SEQUENCES.

Изменение последовательности

- Изменение значения инкремента, максимального и минимального значений, параметров цикличности и кэширования.

```
SQL> ALTER SEQUENCE dept_deptno
  2          INCREMENT BY 1
  3          MAXVALUE 999999
  4          NOCACHE
  5          NOCYCLE;
Sequence altered.
```

Изменение последовательностей

- Вы должны быть владельцем или иметь право ALTER для данной последовательности.
- Изменения вступают в силу только для будущих значений последовательности.
- Для изменения начального числа последовательности ее следует уничтожить и создать заново.

Уничтожение последовательности

- Для удаления последовательности из каталога данных используйте выражение DROP SEQUENCE.
- К удаленной последовательности обращаться нельзя.

```
SQL> DROP SEQUENCE dept_deptno;
Sequence dropped.
```

Какие последовательности есть в БД?

- Вы можете проверить параметры последовательности в таблице USER_SEQUENCES каталога данных.

```
SQL> SELECT   sequence_name, min_value, max_value,
  2         increment_by, last_number
  3  FROM user_sequences;
```

- Столбец LAST_NUMBER показывает следующее число в последовательности.

Зачем нужен индекс?

- Ограничения физического мира
  - скорость random memory access
  - скорость sequencial memory access
  - скорость random disk drive access
  - скорость sequencial disk drive access

Что есть индекс?

- Объект схемы
- Используется Oracle Server для увеличения скорости получения значений строк путем использования указателя
- Снижает количество обращений к жестким дискам за счет быстрого нахождения данных
- Не зависит от таблицы, для которой создан
- Автоматически используется и поддерживается Oracle Server

Как создаются индексы?

- Автоматически - индекс создается автоматически, если вы указываете ограничения PRIMARY KEY или UNIQUE для таблицы.
- Вручную - пользователь может создавать индексы для увеличения скорости доступа к данным.

Создание индекса

- Синтаксис

```
CREATE INDEX index
ON table (column[, column]...);
```

- Увеличение скорости доступа к столбцу ENAME таблицы EMP

```
SQL> CREATE INDEX emp_ename_idx
  2  ON       emp(ename);
Index created.
```

- Создание индекса для одного или более столбцов

```
SQL> CREATE INDEX emp_ename_idx2
  2  ON       emp(ename,depno);
Index created.
```

Когда создавать индексы?

- Столбец часто используется в предложении WHERE или условиях соединения.
- Столбец содержит широкий спектр значений.
- В столбце большое количество значений null.
- Два или более столбцов часто используются совместно в предложении WHERE или условиях соединения.
- Размеры таблицы велики и большинство запросов предположительно будет возвращать менее 2–4% строк.

Советы по созданию индексов

- Не создавайте индексы, если:
  - Таблица невелика
  - Столбцы редко используются в условиях запросов
  - Большинство запросов предположительно будут возвращать более 2–4% строк
  - Таблица часто изменяется

- Почему не всегда эффективно использовать индексы?

Уничтожение индексов

- Удаление индекса из каталога данных.

```
SQL> DROP INDEX index;
```

- Удаление индекса EMP_ENAME_IDX из каталога данных.

```
SQL> DROP INDEX emp_ename_idx;
Index dropped.
```

- Для уничтожения индекса, вы должны бать его владельцем или иметь право DROP ANY INDEX.

Какие индексы есть в БД?

- Представление USER_INDEXES каталога данных содержит имена индексов и их уникальность.
- Представление USER_IND_COLUMNS содержит имя индекса, имя таблицы и имя столбца.

```
SQL> SELECT   ic.index_name, ic.column_name,
  2         ic.column_position col_pos,ix.uniqueness
  3  FROM user_indexes ix, user_ind_columns ic
  4  WHEREic.index_name = ix.index_name
  5  AND  ic.table_name = 'EMP';
```

Типы индексов

- **B-tree indexes** – создается по умолчанию
- **B-tree cluster indexes** –- ссылка не на строку, а блок, где хранится строка
- **Hash cluster indexes** – в качестве ключа используется не само значение, а его хеш
- **Reverse key indexes** – в ключе порядок битов переставляется на противоположный
- **Bitmap indexes** – битовый вектор присутствия признаков (хорошо работает если диапазон значений мал)
- **Function-based indexes** - содержит предвычисленные значения функций для строк таблицы
- **Domain indexes** - пользовательская реализация индекса

Дополнительная информация

- Обзор типов индексов Oracle, MySQL, PostgreSQL, MS_SQL
  - http://habrahabr.ru/post/102785
- Документация Oracle по индексам:
  - http://docs.oracle.com/cd/E11882_01/server.112/e10713/indexiot.htm#BABHJAJF

Выводы

- Создавайте последовательности чисел, используя объекты последовательностей.
- Просматривайте информацию о последовательностях в таблице USER_SEQUENCES каталога данных.
- Создавайте индексы для увеличения скорости выполнения запросов.
- Просматривайте информацию об индексах в таблице USER_INDEXES каталога данных.

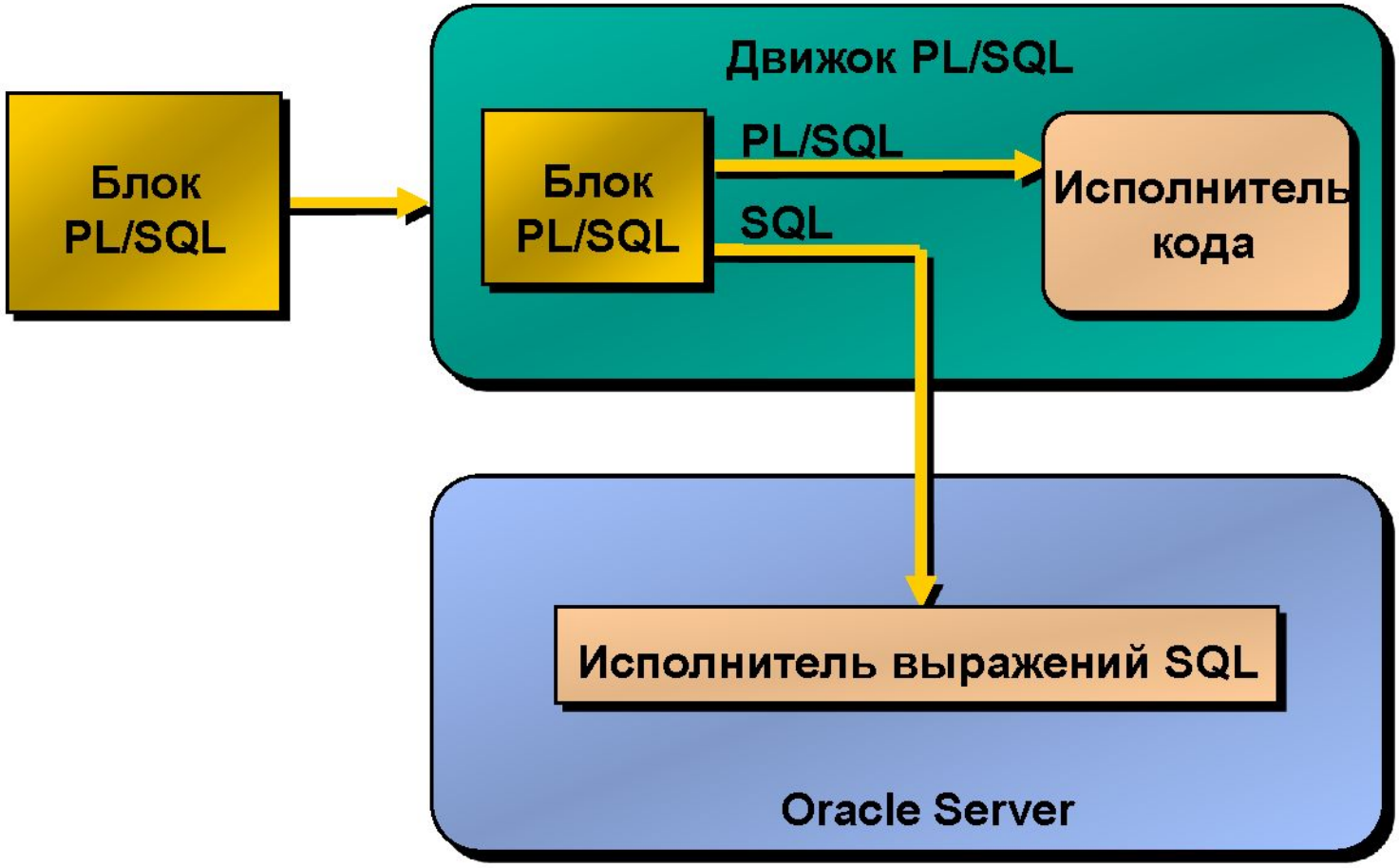
Домашнее чтение

- Триггеры
- PL/SQL
- Оптимизация и план выполнения запроса
- Хинтование
- Синонимы

Немного о PL/SQL

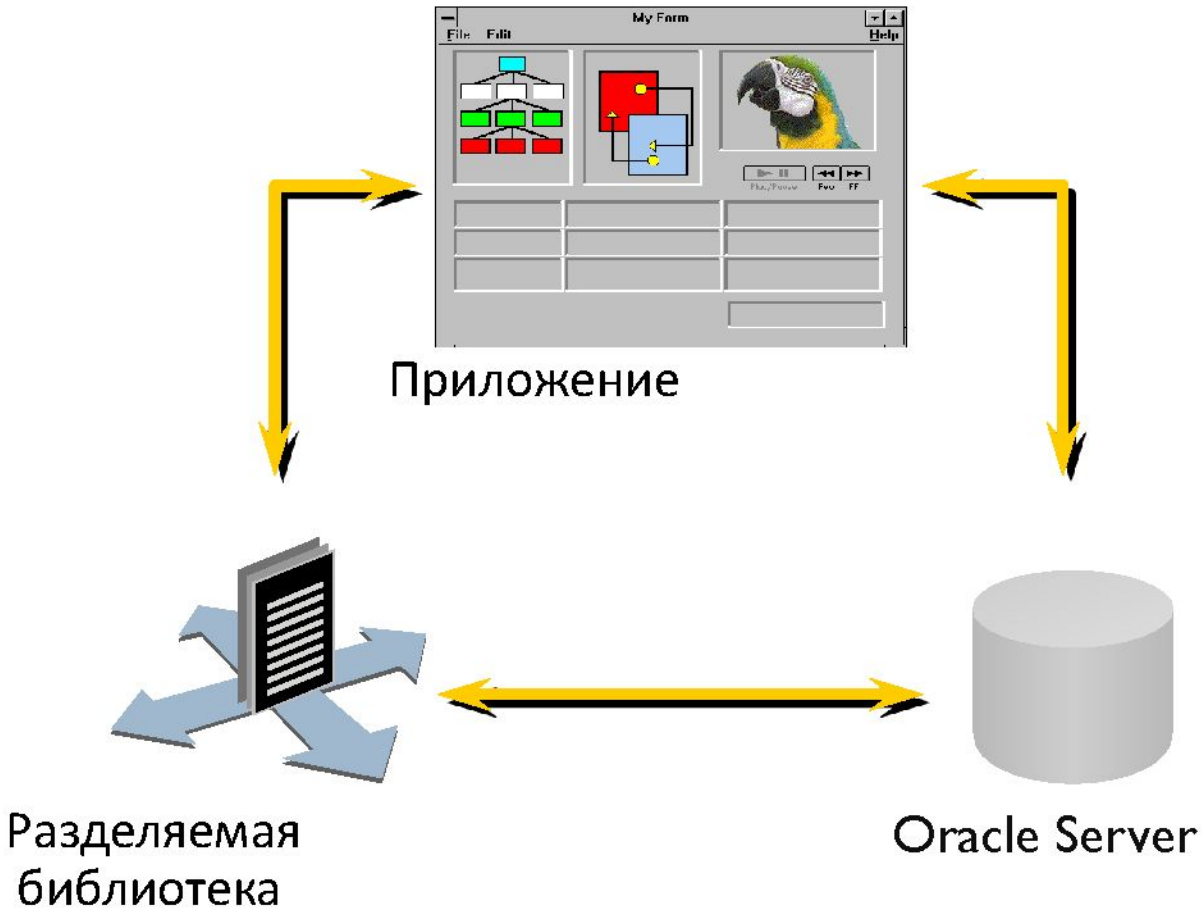
- PL/SQL является расширением SQL, обладающим свойствами языка программирования.
- Манипуляционные выражения и запросы SQL включаются в код.

Среда PL/SQL


Блок PL/SQL
Движок PL/SQL
Блок PL/SQL
PL/SQL
SQL
Исполнитель кода
Исполнитель выражений SQL
Oracle Server

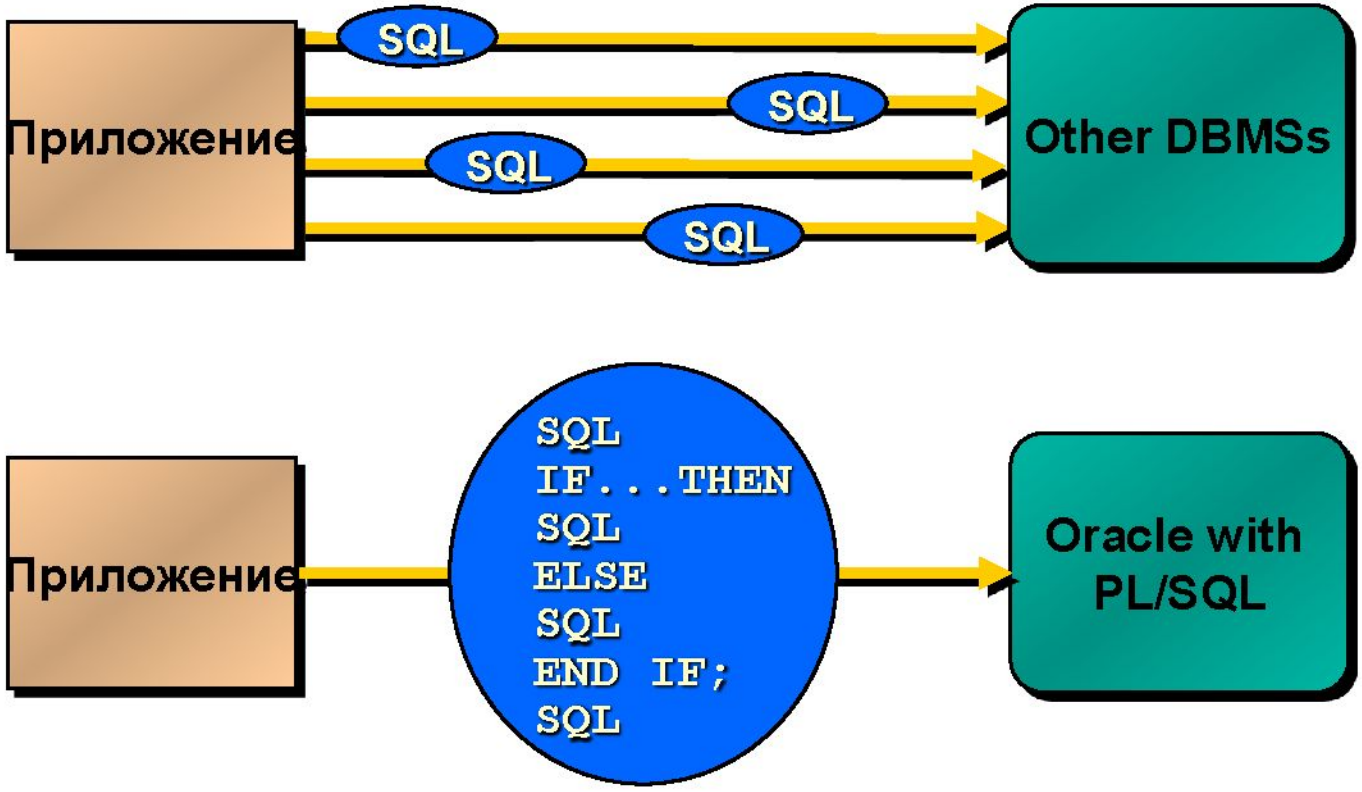
Преимущества PL/SQL

Интеграция



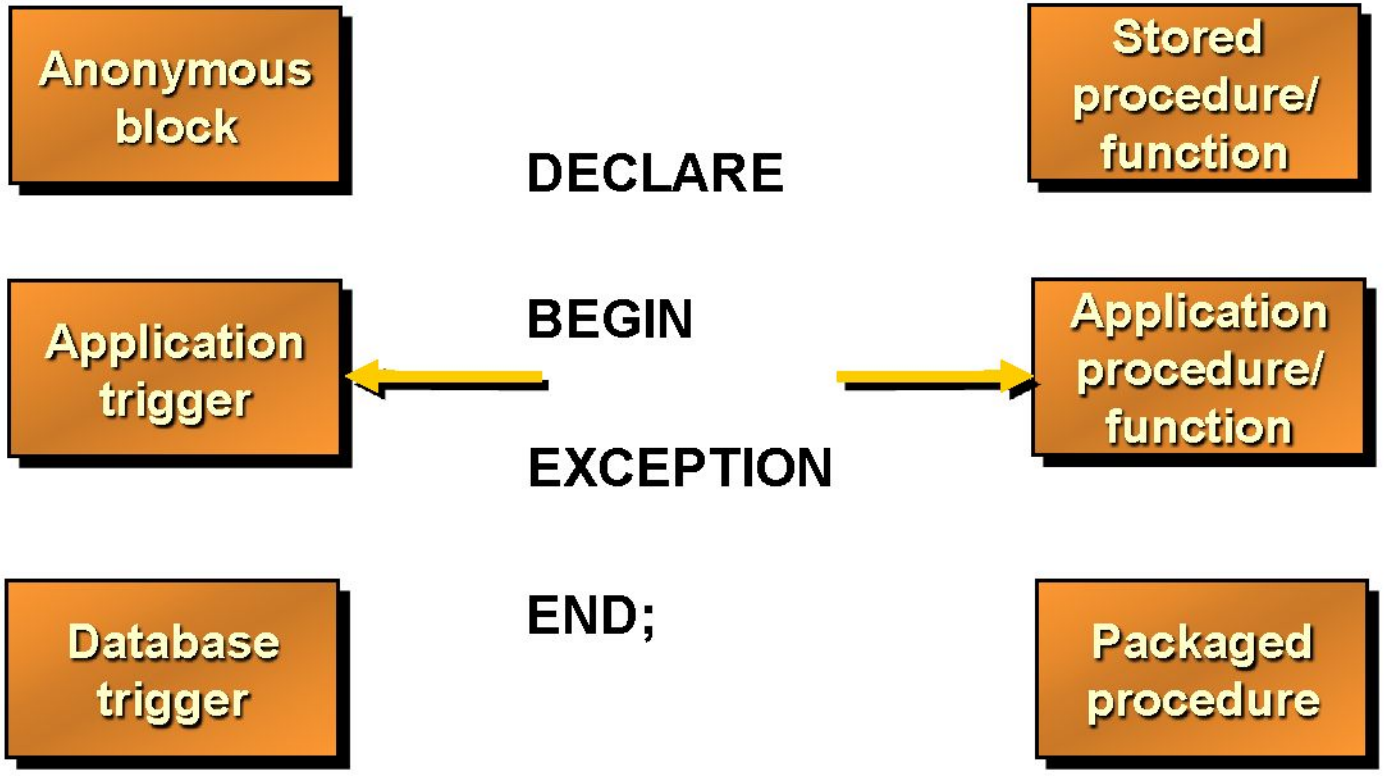
Преимущества PL/SQL

Улучшенное выполнение



Преимущества PL/SQL

Модульный подход к программированию



Преимущества PL/SQL

- Портируемость.
- Возможность использования идентификаторов.
- Управление ходом выполнения программы.
- Отслеживание ошибок.

Применение триггеров

- Расширенный аудит действий
- Предотвращение ошибочных транзакций
- Расширенная проверка целостности (например между распределенными данными)
- Реализация сложной бизнес-логики
- Реализация сложной авторизации
- Прозрачное журналирование событий
- Автоматическое заполнение колонок таблицы
- Отслеживание системных событий

Триггеры

```
1. CREATE OR REPLACE TRIGGER
   Log_salary_increase
2. AFTER UPDATE ON Emp_tab
3. FOR EACH ROW
4. WHEN (new.Sal > 1000)
5. BEGIN
6.   INSERT INTO Emp_log (Emp_id, Log_date,
   New_salary, Action) VALUES (:new.Empno,
   SYSDATE, :new.SAL, 'NEW SAL');
7. END;
```

Триггеры

```
1.  CREATE OR REPLACE TRIGGER
    Print_salary_changes
2.  BEFORE DELETE OR INSERT OR UPDATE ON Emp_tab
3.  FOR EACH ROW
4.  DECLARE sal_diff number;
5.  BEGIN
6.  sal_diff := :new.sal - :old.sal;
7.  dbms_output.put('Old salary: ' || :old.sal);
8.  dbms_output.put('New salary: ' || :new.sal);
9.  dbms_output.put_line('Diff:   '|| sal_diff);
10. END;
```

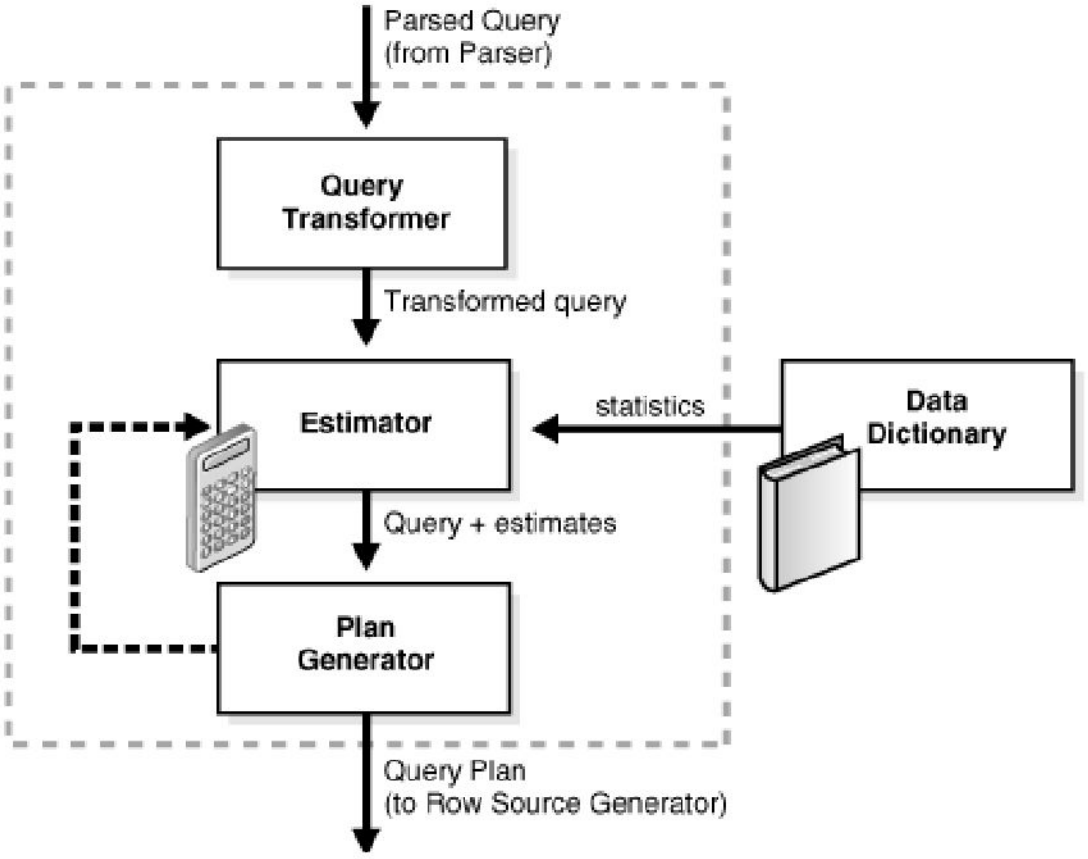
CASCADE UPDATE

```
Create Trigger tu_dept
after update of deptno on dept
FOR EACH ROW
begin
if (:old.deptno != :new.deptno)  then
      begin
      update emp
      set   deptno = :new.deptno
      where       emp.deptno = :old.deptno ;
      end;
end if;
end;
```

Заметки о триггерах

- Используйте триггеры, чтобы выполнить дополнительный код при наступлении определенных событий
- Не нужно дублировать триггерами существующий функционал Oracle Database. (например проверка ограничений)
- Если триггер длиннее 60 строк – лучше реализовать хранимую процедуру, которая будет вызвана из триггера
- Триггер выполняется для события независимо от пользователя, который вызвал событие
- Не создавайте рекурсивные триггеры (например AFTER UPDATE триггер, который вставляет в туже таблицу.)

План выполнения запроса




```
set autotrace [on | off |traceonly] [explain] [statistic]
```

План выполнения запроса

```
SQL> set autotrace traceonly statistics
SQL> SELECT * from emp where empno > 7500;
```

```
15 rows selected.

Statistics
----------------------------------------------------------
   0  recursive calls
   0  db block gets
   3  consistent gets
   0  physical reads
   0  redo size
1437  bytes sent via SQL*Net to client
 419  bytes received via SQL*Net from client
   2  SQL*Net roundtrips to/from client
   0  sorts (memory)
   0  sorts (disk)
  15  rows processed
```

План выполнения запроса

```
SQL> set autotrace on explain
SQL> select ename from emp where empno < 7500;
```

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |    13 |   507 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |    13 |   507 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
1 - filter("EMPNO">7500)
```

Оптимизатор

- **Full table scans** – из БД прочитаны все строки, для каждой проверены условия
- **Rowid scans** – БД знает, где находятся строки-кандидаты и выбирает только часть строк по номерам, после чего проверяет их.
- **Index scans** - поиск нужных строк ведется по индексу. В некотрых случаев данные из таблицы даже не читаются.
- **Cluster scans** – читается не вся таблица, а только ее часть (блок). Номер блока выясняется по номеру строки.
- **Hash scans** – читается не вся таблица, а только ее часть. Номер части узнается из хеш-функции ключа.

Подсказки

```
SELECT /*+ full(t) */ t.name FROM tbl1 t
WHERE t.DATE = SYSDATE
SELECT /*+ index(t ind_date) */ t.name FROM
tbl1 t WHERE t.DATE = SYSDATE
```

- http://iusoltsev.wordpress.com/profile/individual-sql-and-cbo/cbo-hints/

Синонимы

- Вы можете упростить доступ к объектам, создав синоним (дополнительное имя объекта).
  - Доступ к таблицам других пользователей.
  - Сокращение длинных имен.

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

Создание и уничтожение синонимов

- Создание сокращенного имени для представления DEPT_SUM_VU.

```
SQL> CREATE SYNONYM d_sum
  2  FOR      dept_sum_vu;
Synonym Created.
```

- Уничтожение синонима

```
SQL> DROP SYNONYM d_sum;
Synonym dropped.
```