УДК 681.513.52:004.657

ИССЛЕДОВАНИЯ БЫСТРОДЕЙСТВИЯ ВЫПОЛНЕНИЯ РАСЧЕТОВ, ОСНОВАННЫХ НА ТАБЛИЦАХ И НА ПРЕДСТАВЛЕНИЯХ В СИСТЕМАХ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ (НА ПРИМЕРЕ ORACLE).

***Джалбиев Эмирбек Автандилович**, к.т.н., КГТУ им. Раззакова. Кыргызстан. 720044. г. Бишкек, пр. Мира 66. Тел: 0550-85-95-7. e-mail: edzhalbiev@yandex.ru*
***Луговской Станислав Алексеевич**, магистрант, КГТУ им. Раззакова. Кыргызстан. 720044. г. Бишкек, пр. Мира 66, Тел: 0312-56-13-15, e-mail: StasLugovskoy@yandex.ru*

Аннотация. Исследование быстродействия работы базы данных на Oracle по обработке запросов. Рассмотрен один способов оптимизации работы базы данных при создании запросов. Предложен листинг кода по автоматизации заполнения данными таблицы справочников для проведения экспериментов с базами данных. Представлен сравнительный листинг кода создания запросов посредством таблиц и представлений. Представлены результаты эксперимента по исследованию быстродействия обработки данных в базе данных, посредством создания сложных и простых запросов двумя способами.

Ключевые слова: База данных, оптимизация, запрос, таблица, представление, справочник, листинг, код.

RESEARCHES OF FAST-ACTING OF IMPLEMENTATION OF THE CALCULATIONS BASED ON TABLES AND ON PRESENTATIONS IN CONTROL SYSTEM DATABASES (ON THE EXAMPLE OF ORACLE).

Dzhalbiev Emirbek, *PhD (Engineering). Kyrgyzstan. 720044, c. Bishkek. KSTU named after I. Razzakov. Phone: 0550-85-95-73. e-mail: edzhalbiev@yandex.ru*
Lugovskoy Stanislav, *student of master's degree (Management). Kyrgyzstan. 720044, c. Bishkek. KSTU named after I. Razzakov. Phone: 0312-56-13-15. e-mail: StasLugovskoy@yandex.ru*

Abstract. Research of fast-acting of work of database on Oracle on the inquiry processing. One is considered methods of optimization of work of base of the queries given at creation. Listing of program is offered on automation of filling data of table of reference books for realization of experiments with databases. The comparative listing of program of creation of queries is presented by means of tables and presentations. The results of experiment are presented on research of fast-acting of processing of data in a database, by means of creation of difficult and simple queries two methods.

Keywords: Database, optimization, request, table, representation, reference manual, listing, code.

В данной статье рассмотрим один из вопросов, связанных с оптимизацией выполнения запросов в реляционных системах управления базами данных (СУБД), а более конкретно, исследование скорости выполнения расчетов.

Так как одной из актуальных проблем в настоящее время является сокращение времени обработки данных, в связи с большим объемом данных требующих обработки. Кроме того рассмотрим проблемы оптимизации именно на реляционных системах баз данных по двум основным причинам. [2]

Во-первых, реляционные системы в большей степени нуждаются в оптимизации, а с другой стороны предоставляют большие возможности оптимизации. Так как оптимизационные приемы в реляционных СУБД наиболее развиты. [2]

Во-вторых (и это связано с первой причиной), оптимизации в реляционных СУБД посвящено громадное количество публикаций, что позволяет произвести достаточно полный анализ проблем и решений в этой области. [2]

Оптимизации запросов в реляционных СУБД посвящены несколько интересных обзорных работ, так на русском языке книги Дейта [1], Ульмана [5] и Мейера [4], в которых проблемам оптимизации посвящены отдельные главы. А так же в других работах [6.7].

Существует множество методов и инструментов оптимизации СУБД и работы с ними. Так в данной статье мы хотели бы остановиться, на одном из аспектов оптимизации запросов. Не только более подробно рассмотреть данную схему оптимизации, но и провести натурный эксперимент и просмотреть влияние данной схемы на быстродействие работы по обработке БД.

В данной статье все примеры реализованы в среде СУБД Oracle, для проведения экспериментов сгенерирована база данных и применена соответствующая вычислительная и коммуникационная техника. Основной литературой по данной тематике являлась книга Кайт Т. Oracle для профессионалов [3].

Рассмотрим некоторые аспекты практического программирования. Так в практике нет формализованных критериев «простого» и «сложного» кода - это субъективные предпочтения. По нашему мнению особенно тяжело понимать вложенный код (когда делается запрос из запроса) и работа со сводными таблицами (функции pivot/unpivot). Для разрешения данной проблемы рекомендуется провести оптимизацию управления базой данных, то есть использовать представления, так как каждый раз переписывать по 80 строчек

кода (в среднем) - трудоемко. Так же помимо упрощения кода программы, представления дают возможность разграничения прав доступа пользователей. Рассмотрим это на практическом примере.

Пример: Имеется 2 таблицы базы данных. Стоит задача получить доступ на определённый набор данных 1-ой таблицы во 2-ой таблице. Так реализацию данного алгоритма можно провести несколькими способами. Более эффективно, безопаснее и проще это реализовать через представление и дать в нем доступ, чем давать доступ на каждый объект посредством запроса, что более трудоемко. Хотя минусом работы с представлениями является то, что:

1. Не всегда возможно изменять данные напрямую внутри представления
2. Они всегда выполняются медленнее, чем запрос к 1 таблице

Так для подтверждения выше приведенных размышлений проведем эксперимент. Суть эксперимента - создать несколько таблиц в базе данных. заполнить их большим количеством данных. После чего создать представление данных и сравнить время выполнения запросов из представления или же напрямую из таблиц.

В качестве примера я решил использовать «журнал продаж», это экспериментальная база данных (данные имеют только теоретических характер).

Описание базы данных «журнала продаж» представлена на Диаграмме (Рис 1).

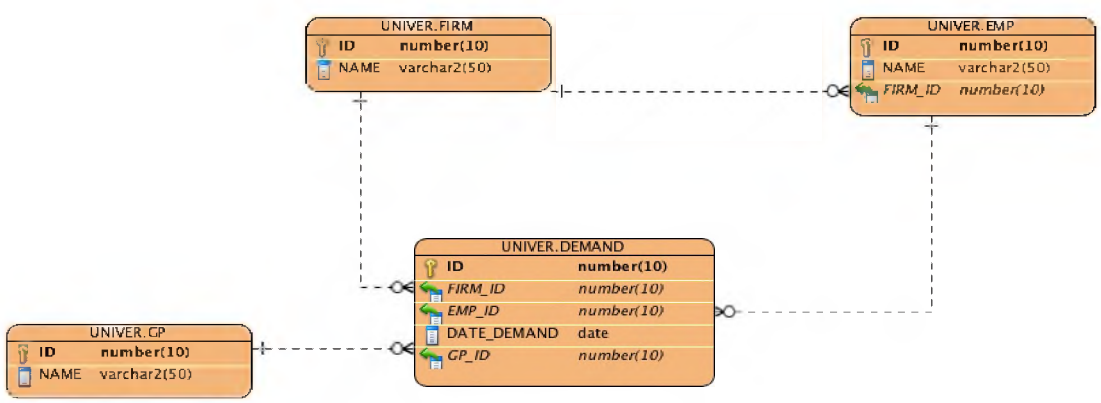

Рис 1 Диаграмма. Тестовая диаграмма данных

Общее представление о входящих в базу данных «Журнал продаж» справочников (таблиц) представлена в Таблице 1, приведенная ниже по тексту. И так база данных состоит из пяти таблиц, каждая имеет уникальный ключ записи и другие поля, содержащие данные.

Ниже приведены описания каждой из таблиц, не нужно забывать что, данные таблицы сформированы только для проведения эксперимента.

Таблица 1.

Общее описание «Журнала продаж»

Наименование	Описание
UNIVER.FIRM	Справочник фирм
UNIVER.EMP	Справочник сотрудников
UNIVER.DEMAND	Журнал продаж
UNIVER.GP	Справочник товаров

Описание полей таблиц входящих в базу данных «Журнала продаж.

Таблица 2.

Описание таблицы UNIVER.FIRM

Имя поля	Тип	Ограничения	Описание
ID	number(10)	PK	Уникальный ключ записи в таблице
NAME	varchar2(50)		Название фирмы

Таблица 3.

Описание таблицы UNIVER.EMP

Имя поля	Тип	Ограничения	Описание
ID	number(10)	PK	Уникальный ключ записи в таблице
NAME	varchar2(50)		Имя сотрудника
FIRM_ID	number(10)	FK (UNIVER.FIRM.ID)	Ссылка на фирму, где работает данный сотрудник

Таблица 4.

Описание таблицы UNIVER.DEMAND

Имя поля	Тип	Ограничения	Описание
ID	number(10)	PK	Уникальный ключ записи в таблице
FIRM_ID	number(10)	FK (UNIVER.FIRM.ID)	Ссылка на фирму, которой продали товар
EMP_ID	number(10)	FK (UNIVER.EMP.ID)	Ссылка на сотрудника, который продал товар
DATE_DEMAND	date(10)		Дата продажи
GP_ID	number(10)	FK (UNIVER.GP.ID)	Ссылка на товар, который продали

Таблица 5.

Описание таблицы UNIVER.GP

Имя поля	Тип	Ограничения	Описание
ID	number(10)	PK	Уникальный ключ записи в таблице
NAME	varchar2(50)		Название товара

После разработки таблиц баз данных, напишем код генерирующий таблицы, но без ограничений (первичных ключей и вторичных ключей) и индексации данных. Данный код приведен на рис. 2.

```
CREATE TABLE demand
  (id                NUMBER,
  firm_id            NUMBER,
  emp_id             NUMBER,
  date_demand        DATE,
  gp_id              NUMBER)
 SEGMENT CREATION IMMEDIATE
 PCTFREE    10
 INITRANS   1
 MAXTRANS   255
 TABLESPACE users
 STORAGE   (
  INITIAL    65536
  NEXT       1048576
```

```
CREATE TABLE emp
  (id                NUMBER,
  name               VARCHAR2(50 CHAR),
  firm_id            NUMBER)
 PCTFREE    10
 INITRANS   1
 MAXTRANS   255
 TABLESPACE users
 NOCACHE
 MONITORING
 NOPARALLEL
 LOGGING
/
```

```
    MINEXTENTS  1
    MAXEXTENTS  2147483645
   )
   NOCACHE
   MONITORING
   NOPARALLEL
   LOGGING
  /

  CREATE TABLE firm
    (id                   NUMBER,
    name                  VARCHAR2(50 BYTE))
   PCTFREE    10
   INITRANS   1
   MAXTRANS   255
   TABLESPACE  users
   NOCACHE
   MONITORING
   NOPARALLEL
   LOGGING
  /

  CREATE TABLE gp
  (id                   NUMBER,
    name                VARCHAR2(50 BYTE))
   PCTFREE    10
   INITRANS   1
   MAXTRANS   255
   TABLESPACE  users
   NOCACHE
   MONITORING
   NOPARALLEL
  LOGGING
  /
```

Рис. 2. Листинг кода, генерирующий таблицы базы данных «Журнал продаж»

Создаем процедуру, которая будет автоматически заполнять данными наши таблицы. На рис. 3 приведен код процедуры.

```
PROCEDURE univer (i_type            IN NUMBER,
                  i_count_rec_spr   IN NUMBER,
                  i_count_rec_demand  IN NUMBER)
/*XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
** Процедура заполнения данных для тестового расчета КГТУ
**
** Author: stasl
** Parameters:
** i_type - тип расчета (0 - зачистить данные,1 - зачистить и заполнить)
** i_count_rec - количество строк в базовых справочниках
** i_count_rec_demand - количество строк в журнале продаж
**
**
** Modification History
** Modified By        Date              Remarks
** stas             17.12.2016          создана
**
**XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*/
IS
   temp    NUMBER;
   temp1   NUMBER;
   v_type  NUMBER;
BEGIN
   -- очистка данных
   DELETE FROM demand;
   DELETE FROM gp;
   DELETE FROM emp;
   DELETE FROM firm;
   COMMIT;
   -- заполние данными ( если нужно)
   IF i_type = 1
   THEN
      BEGIN
        FOR temp IN 1 .. i_count_rec_spr
```

```
        LOOP
          INSERT INTO firm
          VALUES (temp, 'Firm' || temp);
          INSERT INTO gp
          VALUES (temp, 'GP' || temp);
        END LOOP;
        FOR temp IN 1 .. i_count_rec_spr * 5
        LOOP
          INSERT INTO emp
            VALUES (
                  temp,
                  'EMP' || temp,
                  ROUND (DBMS_RANDOM.VALUE (0, i_count_rec_spr)));
        END LOOP;

        FOR temp IN 1 .. i_count_rec_demand
        LOOP
          INSERT INTO demand
          VALUES (temp,
               ROUND (DBMS_RANDOM.VALUE (0, i_count_rec_spr)),
               ROUND (DBMS_RANDOM.VALUE (0, i_count_rec_spr * 5)),
               SYSDATE - ROUND (DBMS_RANDOM.VALUE (0, 50)),
               ROUND (DBMS_RANDOM.VALUE (0, i_count_rec_spr)));
        END LOOP;
        COMMIT;
      END;
   END IF;
EXCEPTION
   WHEN OTHERS
   THEN
      raise_application_error (-20001, CHR (13) || SQLERRM);
END;
```

Рис. 3 Листинг кода для автоматического заполнения данными таблиц базы данных «Журнал продаж»

Как и было описано в самом коде процедуры, на вход подается 3 параметра:

1. Тип расчета (на случай, если нужно сделать только очистку без заполнения);
2. Количество строк в каждом из базовых справочников (GP,FIRM,EMP);
3. Количество строк в журнале продаж.

После завершения подготовки по созданию базы данных, напишем представление, которое даст информацию в удобном виде. Код представления приведена на рис. 4.

```
CREATE OR REPLACE VIEW view_univer (
id,
  date_demand,
  firm_from,
  firm_ti,
  gp )
AS
SELECT demand.id,
     demand.date_demand,
     firm_from.name firm_from,
     firm_to.name firm_ti,
     gp.name gp
  FROM demand
     INNER JOIN emp ON emp.id = demand.emp_id
     INNER JOIN gp ON gp.id = demand.gp_id
     INNER JOIN firm firm_from ON firm_from.id = demand.firm_id
     INNER JOIN firm firm_to ON firm_to.id = emp.firm_id)
/
```

Рис. 4. Листинг кода Представления

Сравним трудоемкость написания кода. Так трудоемкость написания кода для простого запроса, в представления наименее трудоемка по сравнению с таблицей. А при написании сложного запроса, они равнозначны. Код процедур приведен в таблице 6.

Таблица 6

Источник	Таблицы	Представление
Код запроса простой	SELECT demand.id, demand.date_demand, firm_from.name firm_from, firm_to.name firm_ti, gp.name gp FROM demand INNER JOIN emp ON emp.id = demand.emp_id INNER JOIN gp ON gp.id = demand.gp_id INNER JOIN firm firm_from ON firm_from.id = demand.firm_id INNER JOIN firm firm_to ON firm_to.id = emp.firm_id	SELECT a.id, a.date_demand, a.firm_from, a.firm_ti, a.gp FROM view_univer a
Код запроса сложный (с группировкой)	select firm.name,count(gp_id) from demand inner join firm on firm.id = demand.firm_id group by firm.name order by firm.name	select firm_from,count(gp) from view_univer group by firm_from order by firm_froma

После проведения подготовительных работ, проведен сам эксперимент. Все результаты экспериментов приведены в таблице 7 и графическое представление на Диаграмме Результаты экспериментов рис. 5.

Талица 7.

Параметр	Таблица	Представление
	млс	млс
Сложный запрос(первый запуск)	1,265	2,063
Сложный запрос(повторный запуск)	1,672	3,437
Простой запрос (первый запуск)	0,094	0,078
Простой запрос (повторный запуск)	0,218	0,157

Результаты экспериментов

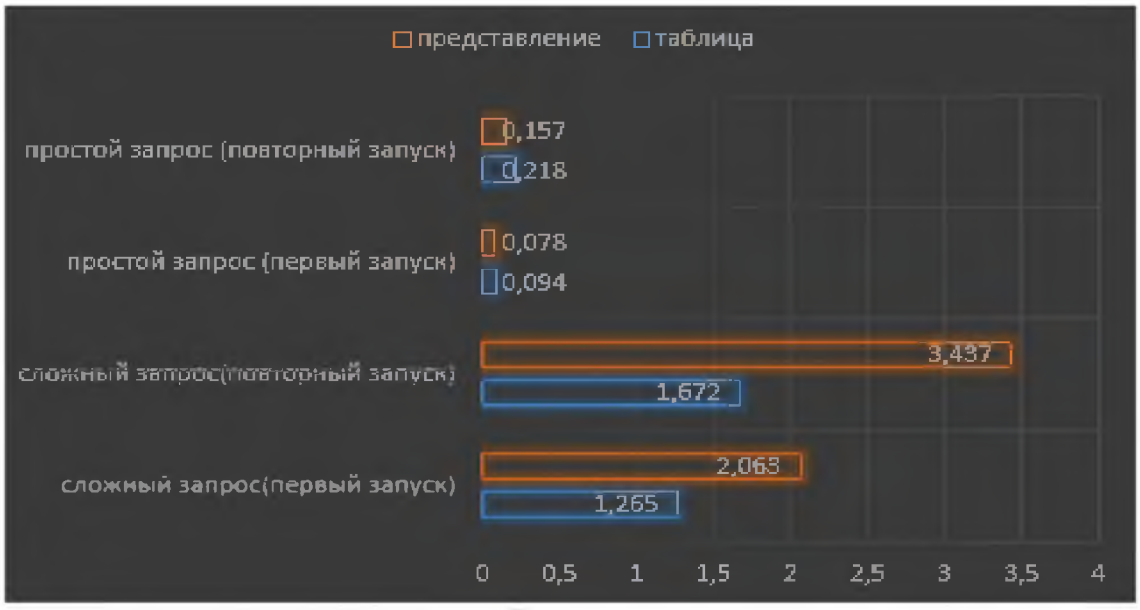

Рис. 5. Диаграмма «Результаты экспериментов».

Так как видно из таблицы и диаграммы:

- при простом запросе разница в быстродействие на стороне применения представления. 0,157 млс и 0,078 млс против 0,218 и 0,094. Но код написания проще в представлении;
- при сложном запросе, запрос из таблицы показывает лучшее быстродействие чем посредством представления.

В соответствии с чем, по результатам эксперимента можно сделать следующие **выводы**:

1. Использование представлений в простых запросах эффективнее, нежели работа напрямую с таблицами за счет оптимизации БД Oracle. Результаты экспериментов прямо соотносятся с трудоемкостью написания кода;
2. В случае сложных запросов или запросов с агрегированием, представления не дают такой эффективности;
3. Для оптимизации работы Базы данных требуется комбинировать представления и таблицы между собой, то есть реализовывать сценарии по обработке данных, приводящих к наиболее оптимальному соотношению.

Список литературы:

1. Дейт К. Введение в системы баз данных.- М.: Наука.1980.- 463 с.
2. Кузнецов С. Методы оптимизации выполнения запросов в реляционных СУБД citforum.ru/database/articles/art_26.shtm
3. Кайт Т. Oracle для профессионалов. Архитектура, методики программирования и особенности версий 9i, 10g и 11g. 2-е издание – М.: Вильямс – 2011 – 848 с
4. Мейер Д. Теория реляционных баз данных.- М.: Мир.1987.- 608 с.
5. Ульман Д. Основы систем баз данных.- М.: Финансы и статистика.- 1983.- 335 с.
6. Date C.J. An Introduction to Database Systems. V.1. 4th ed.- Reading, Mass.: Addison-Wesley.- 1984.- 639 c.
7. Jarke M., Koch J. Query Optimization in Database Systems // ACM Comput. Surv.- 1984.16, N 2.- C. 111-152