# **Практическая работа №9** Методы контроля качества данных. Индексы и оптимизация запросов

Объём учебного времени – 2ч

Методические рекомендации

1. Цель работы: научиться анализировать план запроса, создавать индексы, применять методы оптимизации запросов
2. Основные теоретические положения:

Методы контроля качества данных

Строковые функции

UPPER – преобразует все символы строки в верхний регистр.

LOWER – преобразует все символы строки в нижний регистр.

INSTR – возвращает n-e вхождение подстроки в строке.

LENGTH – возвращает длину строки.

LTRIM – удаляет все указанные символы с левой стороны строки.

RTRIM – удаляет все указанные символы с правой стороны строки.

TRIM – удаляет все указанные

SUBSTR – извлекает подстроку из строки.

REPLACE – заменяет последовательность символов в строке другим

набором символов.

TRANSLATE – заменяет последовательность символов в строке другим

набором символов (посимвольно).

DUMP – возвращает строку, содержащую код типа данных, длину в байтах

и внутреннее представление поля.

VSIZE – возвращает длину в байтах для внутреннего представления поля

Регулярные выражения

REGEXP\_INSTR – расширение функции INSTR, в строке поиска используются регулярные выражения.

REGEXP\_LIKE – расширение выражения LIKE, в строке поиска используются регулярные выражения.

REGEXP\_REPLACE – расширение функции REPLACE, в строке поиска используются регулярные выражения.

REGEXP\_SUBSTR – расширение функции SUBSTR, в строке поиска используются регулярные выражения

^ — начало строки;

$ — конец строки;

. — любой символ;

\* – любое количество предыдущих символов;

+ – 1 или более предыдущих символов;

? – 0 или 1 предыдущих символов;

( ) – группировка конструкций;

| – оператор «ИЛИ»;

[ ] – любой из перечисленных символов, диапазон. Если первый символ в

этой конструкции – «^», то массив работает наоборот – проверяемый

символ не должен совпадать с тем, что перечислено в скобках;

{ } – повторение символа несколько раз;

\ – обратный слеш. Экранирование служебных символов.

Специальные метасимволы, ими можно заменить некоторые готовые

конструкции:

\b — обозначает не символ, а границу между символами;

\d — цифровой символ;

\D — нецифровой символ;

\s — пробельный символ;

\S — непробельный символ;

\w — буквенный или цифровой символ или знак подчеркивания;

\W — любой символ, кроме буквенного или цифрового символа или знака

подчеркивания.

Индексы и оптимизация запросов

**Индексы Oracle** обеспечивают быстрый доступ к строкам таблиц в базе данных, сохраняя отсортированные значения указанных столбцов и используя эти отсортированные значения для быстрого нахождения ассоциированных строк таблицы

Применение индексов представляет собой компромисс между ускорением получения результатов запросов и замедлением обновлений и вставок данных.

Вообще говоря, если таблицы в основном используются для чтения (выборки) информации, как в хранилищах данных, то лучше иметь много индексов. Если же база данных относится к типу OLTP, с большим количеством вставок, обновлений и удалений, то лучше обойтись меньшим числом индексов.

Просмотр плана запроса

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Клавиша F10 – просмотр плана запроса

Пример создания индекса **create index CI on Sales(Cust\_Id);**

Поиск по индексу все еще делается три раза, по разу для каждого клиента, но каждый поиск по индексу возвращает только те строки, которые ссылаются на C.Cust\_Id и квалифированы предикатом соединения. Таким образом, поиск возвращает только 3 строки по сравнению с 12 строками, возвращаемыми при просмотре таблицы.

Поиск по индексу

Изображение выглядит как текст

Автоматически созданное описание

Пример: Выведите список клиентов (oe.clients) и названий продуктов (oe.product\_information), которые они покупали (связь через oe.orders и oe.order\_items).

С помощью hints добейтесь всех трех способов исполнения соединения – nested loops, sort merge и hash join.

/\*+ USE\_NL(t1 t2) \*/ /\*+ USE\_MERGE(t1 t2) \*/ /\*+ USE\_HASH(t1 t2) \*/

select distinct

cust\_first\_name,

cust\_last\_name,

product\_name

from oe.customers c

left join oe.orders o

on c.customer\_id = o.customer\_id

left join oe.order\_items oi

on oi.order\_id = o.order\_id

left join oe.product\_information pi

on oi.product\_id = pi.product\_id;

select /\*+ USE\_NL(c o) \*/ distinct

cust\_first\_name,

cust\_last\_name,

product\_name

from oe.customers c

left join oe.orders o

on c.customer\_id = o.customer\_id

left join oe.order\_items oi

on oi.order\_id = o.order\_id

left join oe.product\_information pi

on oi.product\_id = pi.product\_id;

select /\*+ USE\_MERGE(c o) \*/ distinct

cust\_first\_name,

cust\_last\_name,

product\_name

from oe.customers c

left join oe.orders o

on c.customer\_id = o.customer\_id

left join oe.order\_items oi

on oi.order\_id = o.order\_id

left join oe.product\_information pi

on oi.product\_id = pi.product\_id;

Партиционирование

**Партиционирование** — **это** разбиение таблиц, содержащих большое количество записей, на логические части по неким выбранным администратором критериям.

* **Фрагментация по диапазону значений**
* **Фрагментация по списку значений**
* **Фрагментация с использованием хэш функции**

**Создание табличного пространства**

**CREATE TABLESPACE TBLSP1 DATAFILE ‘D:\TBLSP1.DAT'**

**SIZE 10M REUSE AUTOEXTEND ON NEXT 2M MAXSIZE 20M;**

**…**

**Аналогично другие**

Для выполнения потребуется войти под пользователем с правами администратора

Создания и фрагментации таблиц используется дополнительная синтаксическая конструкция в команде **CREATE ТABLE – PATITION BY**

CREATE TABLE Имя таблицы

(столбец1 тип, столбец2 тип, столбец n)

PARTITION BY HASH(имя столбца по которому строится хэшфункция) или

**Описание фрагментов**

(partition имя фрагмента1(партиции) название табличного пространства,

partition имя фрагмента2(партиции) название табличного пространства)

Или

PARTITION BY RANGE(имя столбца для проверки на соответствие диапазону фрагмента)

(PARTITION имя фрагмента1 VALUES LESS THAN

(выражение1) ) имя табличного пространства,

(PARTITION имя фрагмента1 VALUES LESS THAN

(выражение2 )) имя табличного пространства

Выражение 1 , Выражение 2 задают диапазон для данных , для определения фрагмента таблицы(партиции) к которому эти данные относятся.

С помощью оператора **SELECT**есть возможность выбирать как все данные из фрагментированной таблицы, так ииспользовать **SELECT**для выбора данных из заданного фрагмента таблицы.

Пример

**CREATE TABLE pro\_range**

**( summ int,**

**docdate date,**

**docnum number**

**)**

**PARTITION BY RANGE(docdate)**

**(partition pt\_1 values less than (to\_date('01.09.2020','DD.MM.YYYY')) tablespace TBLSP1,**

**partition pt\_2 values less than (to\_date(‘01.10.2020','DD.MM.YYYY')) tablespace TBLSP2,**

**partition pt\_3 values less than (to\_date(‘01.11.2020','DD.MM.YYYY')) tablespace TBLSP3,**

**partition p\_othermax values less than (maxvalue) tablespace TBLSP3**

**);**

**Пример партиционирования по диапазонам (без применения команды TABLESPACE)**

CREATE TABLE summer.ivanov\_orders

(

order\_id NUMBER,

order\_date TIMESTAMP,

order\_mode VARCHAR2(8),

customer\_id NUMBER,

order\_status NUMBER,

order\_total NUMBER(8,2),

sales\_rep\_id NUMBER,

promotion\_id NUMBER

)

PARTITION BY RANGE ( order\_date )

( PARTITION p1 VALUES LESS THAN ( TO\_DATE( '01-01-2005', 'dd-mm-yyyy' ) ),

PARTITION p2 VALUES LESS THAN ( TO\_DATE( '01-01-2007', 'dd-mm-yyyy' ) ),

PARTITION p3 VALUES LESS THAN ( TO\_DATE( '01-01-2008', 'dd-mm-yyyy' ) ),

PARTITION p4 VALUES LESS THAN ( TO\_DATE( '01-01-2009', 'dd-mm-yyyy' ) )

);

insert into summer.ivanov\_orders( order\_id, order\_date, order\_mode, customer\_id, order\_status, order\_total, sales\_rep\_id, promotion\_id )

select order\_id, order\_date, order\_mode, customer\_id, order\_status, order\_total, sales\_rep\_id, promotion\_id from oe.orders;

select \* from summer.ivanov\_orders where order\_date between to\_date( '01-01-2007', 'dd-mm-yyyy' ) and to\_date( '31-12-2007', 'dd-mm-yyyy' ); -- 1 партиция

select \* from summer.ivanov\_orders where order\_date between to\_date( '01-01-2007', 'dd-mm-yyyy' ) and to\_date( '31-12-2008', 'dd-mm-yyyy' ); -- 2 партиции

Фрагментация с использованием списка значений

**CREATE TABLE pro\_list**

**( summ int,**

**docdate date,**

**doctype varchar2(2))**

**PARTITION BY LIST(doctype)**

**(**

**partition pt\_1 values ('PP','PO' ) tablespace TBLSP1,**

**partition pt\_2 values ('RR') tablespace TBLSP2,**

**partition pt\_3 values ('RO','SO') tablespace TBLSP3**

**);**

CREATE TABLE summer.ivanov\_locations

(

location\_id NUMBER,

street\_address VARCHAR2(40),

postal\_code VARCHAR2(12),

city VARCHAR2(30),

state\_province VARCHAR2(25),

country\_id CHAR(2)

)

PARTITION BY LIST( country\_id )

( PARTITION p1 VALUES ('US', 'SG' ),

PARTITION p2 VALUES ('CA', 'CH' ),

PARTITION p3 VALUES ('IT', 'MX' ),

PARTITION p4 VALUES ('CN', 'DE'),

PARTITION p5 VALUES ('JP', 'IN'),

PARTITION p6 VALUES ('AU', 'UK'),

PARTITION p7 VALUES ('BR', 'NL')

);

Фрагментация с использованием хэш-функции

**CREATE TABLE pro\_hash**

**( summ int,**

**docdate date,**

**docnum number**

**)**

**PARTITION BY HASH(docnum)**

**(partition pt\_l tablespace TBLSP1,**

**partition pt\_2 tablespace TBLSP2,**

**partition pt\_3 tablespace TBLSP3**

**);**

1. Практические указания:
   1. **Методы контроля качества данных**
      1. Работа с неструктурированными данными
         * Создайте таблицу, в которой в одно поле сохраните **Фамилия, Год РОЖД, категория**, таких записей у вас должно быть несколько
         * Произведите поиск по данным всех фамилий
         * Произведите поиск по данным всех годов рождений
         * Произведите поиск по данным всех категорий
         * Разбейте неструктурированные данные (воспользуйтесь методом работы с регулярными выражениями в сочетании с командой CONNECT by и преобразуйте каждую строку с разделителями в строку таблицы Users)
      2. Добавьте в таблицу книги поле Shifr, заполните его значениями, при этом используйте check проверку с применением регулярного выражения (первые две буквы латинские, потом четыре цифры)
   2. **План выполнения запроса**
      1. Напишите инструкцию для анализа статистики таблицы Bilet
         * ANALYZE TABLE Имя\_таблицы COMPUTE STATISTICS
      2. Просмотрите данные анализа
         * SELECT table\_name, last\_analyzed FROM dba\_tables WHERE table\_name = ‘Имя\_таблицы’;
      3. Постройте план запроса для нахождения данных о выдачи книг указанному пользователю
         * EXPLAIN PLAN FOR SQL-запрос
      4. Выведите информацию о полученном плане
         * SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY)
      5. Просмотрите план запроса с помощью клавиши F10
   3. **Создание индексов**
      1. Создайте индекс в таблице **AVTOR** – включающий поля **Fam** и **Name**.
      2. Добавьте поле – **kod\_bilet** – таблицу **Bilet**. Создайте уникальный индекс **kod\_bilet** в таблице **Bilet**.
         * CREATE INDEX <index\_name> ON <table\_name> ( <field\_name> );
      3. Создайте индекс в таблице **USERS**, включающий поле категория пользователя (по убыванию значения).
      4. Посмотрите план запроса с применением индекса.
      5. План запроса с индексом и без
         * Создайте копию таблицы users. Запросите одного пользователя по его id. Постройте план запроса, определите способ доступа.
         * Создайте индекс на поле client\_id:
         * Выбирайте некоторое количество (id <5,10,15 и т.д.). Стройте план запроса и оцените границу (процент выбираемых строк по сравнению с общим количеством) когда изменится способ доступа к данным (с доступа по индексу на full table scan).
      6. Выберите всех пользователей, которые в текущем году не взяли ни одной книги.
      7. Выведите список пользователей и название книг, которые они брали. С помощью hints добейтесь всех трех способов исполнения соединения.
   4. **Секционирование** 
      1. Создание таблицы с разбиением по значению поля даты
         * Создайте секционированную таблицу **Sec\_Book**, содержащую данные о шифрах книги и даты ее выдачи, создайте три секции, в зависимости от даты выдачи (распределите по годам или по месяцам, в зависимости от данных вашей таблицы)
         * Напишите запрос на вставку записей в созданную таблицу из таблицы Bilet
         * Напишите запрос на выборку данных из каждой созданной секции, например, select \* from Sec\_Book partition(part1);
      2. Создайте таблицу **Book\_razd**, которая будет содержать список книг по разделам
      3. Создайте таблицу **Users**, которая будет содержать список пользователей в зависимости по годам рождения (диапазонам годов).
2. Содержание отчёта по практической работе
   1. Скрины выполнения работы
   2. Запросы и комментарии к ним
3. Контрольные вопросы
   1. Методы контроля качества данных
   2. Преимущества и недостатки при использовании индекса.
   3. Оптимизация запросов
   4. Фрагментация таблиц
4. Список ссылок:
   1. <https://tproger.ru/translations/types-of-nosql-db/>
   2. <https://oracle-patches.com/oracle/prof/3013-%D0%B8%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D1%8B-oracle>
   3. <https://tproger.ru/articles/regexp-for-beginners/>
   4. <https://www.sql.ru/blogs/oracleandsql/1372>
   5. Документация по партиционированию: <http://gg.gg/kxx2j>