УДК 004.45 МРНТИ 50.05.17

ЭФФЕКТИВНОСТЬ РЕАЛИЗАЦИИ ИНДЕКСОВ НА ОСНОВЕ БИТОВЫХ КАРТ

И. Т. Утепбергенов, д.т.н., проф., ***Г. С. Нурбакова,*** к.ф.-м.н., ***Н. К. Смайлов***

Казахская академия транспорта и коммуникаций им. М. Тынышпаева

Приведены результаты работ, которые связаны с моделированием процессов на основе индексов. Рассмотрены примеры блоков и их апробация. Показано, что индекс на основе битовых карт настолько эффективен, что его можно использовать для доступа к большим частям таблицы способом, не считающимся целесообразным при использовании индекса на основе B*-дерева.
Ключевые слова: битовые карты, индексы.

Мақалада индекстер негізінде процесстерді модельдеумен байланысты жұмыстар нәтижесі келтірілген. Блоктар мен оларды сынақтау мысалдары көрсетілген. Биттік карталар негізіндегі индекстердің тиімділігі соншалық, оларды B*-тармақ негізінде индексті қолдану кезінде орынды деп есептелмейтін әдіспен кестелердің үлкен бөліктеріне кіру үшін де қолдануға болатыны көрсетілген.
Түйінді сөздер: Биттік карталар,индекстер.

The article presents results of works which are connected with modeling of processes on the basis of indexes. Examples of blocks and their approbation are shown. It is shown that the index on the basis of bit maps is so effective that it can be used for access to the most parts of the table in the way which is not considered expedient at use of an index on the basis of a B+-tree.
Key words: bit maps, indexes.

Индексы на основе битовых карт - великое благо для некоторых видов приложений, но об их устройстве, использовании и побочных эффектах распространяется достаточно неверная информация [1]. Если провести простое исследование понима-

ния разработчиками индексов на основе битовых карт, скорее всего следующие комментарии окажутся наиболее типичными:

• При наличии индексов на основе битовых карт любое изменение ключевых столбцов в таблице полностью ее блокирует.

• Индексы на основе битовых карт хорошо подходят для столбцов с небольшим количеством различных значений.

• Доступ по индексам на основе битовых карт эффективнее полного просмотра таблицы, даже если по запросу возвращается существенная часть таблицы.

• Третье утверждение на самом деле всего лишь следствие (возможно, не проверенное) второго. Причем все 3 утверждения попадают в смутную зону между ложью и большим заблуждением.

Конечно, в этих утверждениях есть и небольшая доля правды, достаточная, чтобы объяснить их происхождение. Индексы создаются, чтобы сервер Oracle мог максимально эффективно находить запрошенные строки. Индексы на основе битовых карт - не исключение. Однако стратегия, лежащая в основе этих индексов, очень отличается от стратегии, на которой базируются индексы на основе B*-дерева [1,2]. Чтобы продемонстрировать это, можно начать с изучения содержимого нескольких блоков. Рассмотрим SQL-сценарий:

```
create table t1
nologging
as
select rownum id,
mod(rownum,10) btree_col,
mod(rownum,10) bitmap_col,
rpad(«x»,200) padding
from all_objects
where rownum <= 30000;
create index t1_btree on t1(btree_col);
create bitmap index t1_bit on t1(bitmap_col).
```

Обратите внимание, что столбцы btree_col и bitmap_col заданы так, что содержат идентичные данные - числа от 0 до 9, повторяющиеся циклически.

В базе данных версии 9.2 с размером блока 8 Кбайт результирующая таблица займет 882 блока. Индекс на основе B*-дерева будет иметь 57 листовых блоков, а индекс на основе битовых карт - 10 листовых блоков.

Фрагмент листового блока индекса на основе B*-дерева:

```
row#538[2016] flag: ——, lock: 0
col 0; len 2; (2): c1 02
col 1; len 6; (6): 00 40 c5 7d 00 09
row#538[2004] flag: ——, lock: 0
col 0; len 2; (2): c1 02
col 1; len 6; (6): 00 40 c5 7d 00 13
```

Фрагмент листового блока индекса на основе битовых карт:

```
row#2[4495] flag: ——, lock: 0
col 0; len 2; (2): c1 03
col 1; len 6; (6): 00 40 c5 62 00 00
col 2; len 6; (6): 00 40 c7 38 00 1f
col 3; len 3521; (3521):
cb 02 08 20 80 fa 54 01
04 10 fb 53 20 80 00 02
fc 53 04 10 40 00 01 fa
53 02 08 20 fb 53 40 00 . . .
```

Представлены блоки данных, сброшенные в символьном виде. Понятно, что индекс на основе битовых карт несколько плотнее упакован, чем индекс на основе B*-дерева. Чтобы увидеть эту упаковку, можно сбросить блоки данных индекса в символьном виде с помощью команд типа:

```
alter system dump datafile x block y;
```

Результаты представлены выше в приведенном блоке данных. Однако информация, полученная в результате сброса блока в символьном виде, может иногда приводить к неверным выводам, поскольку часть ее - производная от данных, и порядок следования тоже изменен по отношению к реальному для ясности.

Обратившись к вышеприведенному блоку, можно увидеть, что запись индекса на основе B*-дерева состоит из набора флагов, байта блокировки и (в данном случае) 2-х столбцов данных, которые представляют собой проиндексированное значение и

идентификатор строки. Причем для каждой строки в таблице имеется запись этого вида в индексе. (Если бы индекс был уникальным, содержимое каждой записи было бы таким же, но расположение немного отличалось.)

В индексе на основе битовых карт каждая запись состоит из набора флагов, байта блокировки (в данном случае) из 4-х столбцов данных. Эти 4 столбца на самом деле - проиндексированное значение, пара идентификаторов строк и поток битов. Пара идентификаторов строк задает непрерывную часть таблицы, а поток битов определяет, какие строки в этом диапазоне идентификаторов строк содержат соответствующее значение.

Нужно обратить внимание, что на размер потока битов - длина столбца - в представленном примере составляет 3521 байт, или около 27000 битов. Около 12 % - накладные расходы на контрольные суммы и т.п., поэтому эта запись может покрыть порядка 24000 строк таблицы. Но на всю запись имеется только 1 байт блокировки, которая будет влиять на 24000 строк таблицы.

Вот откуда происходит сомнительное утверждение о блокировании всей таблицы: если кажется, что изменение столбца ключа индекса на основе битовых карт вызывает блокирование всей таблицы, значит, вы экспериментировали со слишком маленькими таблицами.

Одна блокировка битовой карты может затрагивать тысячи строк, что, несомненно, плохо, но вся таблица не блокируется.

Конечно, при использовании битовых карт возникают некоторые проблемы, выходящие за рамки конфликтов при изменении.

Вставки и удаления из таблицы вызывают изменения всех ее индексов. С учетом большого количества строк, покрываемых одной записью индекса на основе битовых карт, при любом количестве одновременных вставок или удалении велика вероятность обращения к одним и тем же секциям индекса и, следовательно, массовых конфликтов доступа [2,3]. Более того, даже последовательное выполнение операторов ЯМД, затрагивающих индексы на основе битовых карт, может куда существеннее сказаться на производительности, чем можно было предположить.

Ранее подчеркивалось, что простое изменение одной строки обычно приводит к копированию всей соответствующей секции битовой карты. В первом блоке данных можно увидеть, насколько большой может быть секция битовой карты. В этом примере она была размером 3500 байтов (в Oracle 9 максимальный размер составляет около половины блока). Можно обнаружить, что небольшое изменение данных очень существенно влияет на размер любого изменяемого вследствие этого индекса на основе битовых карт [2].

В общем случае стоит исходить из предположения, что даже последовательные пакетные изменения будут выполняться эффективнее, если удалить индексы на основе битовых карт перед их выполнением, а затем пересоздать. Часто утверждается, что «индексы на основе битовых карт подходят для столбцов с небольшим количеством значений». Несколько точнее будет формулировка «с небольшим количеством различных значений». В любом случае речь идет о столбцах, содержащих сравнительно мало различных значений.

Это утверждение действительно достаточно верное, если его соответствующим образом уточнить и разъяснить. К сожалению, многие в результате думают, что индекс на основе битовых карт чудесным образом настолько эффективен, что его можно использовать для доступа к большим частям таблицы способом, не считающимся целесообразным при использовании индекса на основе B*-дерева.

Классическим примером применимости индекса на основе битовых карт является экстремальный случай столбца, представляющего пол. В этом столбце может быть всего 2 значения (или 3, если включить требуемое стандартом ISO значение «n/a» - неизвестен).

Рассмотрим пример, основанный на данных о населении Казахстана. Пусть используются блоки размером 8 Кбайт и строки (весьма типичным) размером 200 байтов, что дает 40 строк в блоке. Вставим в таблицу несколько миллионов строк, обеспечив равномерно случайное распределение по странам. Таким образом, в среднем в каждом блоке будет по 10 строк для каждой страны.

Если использовать индекс на основе битовых карт для доступа ко всем строкам по Казахстану, придется (10 раз) последовательно прочитать каждый блок таблицы. Вне всякого сомнения, эффективнее будет выполнить полный просмотр таблицы, а не использовать такой индекс.

На самом деле, даже если расширить данные так, чтобы они включали информацию по 40 странам, все равно вполне вероятно получить по одной строке в каждом блоке таблицы. Вероятно, когда данные разрастутся до глобального масштаба (скажем, охватят 640 стран, чтобы строка для данной страны встречалась в среднем лишь в каждом 16-м блоке), может, проще обращаться к ним по индексу на основе битовых карт, а не путем полного просмотра таблицы. Но столбец, имеющий 640 различных значений, вряд ли на первый взгляд попадает под определение «с небольшим количеством различных значений».

Конечно, описательные выражения типа «небольшой», «маленький», «близкий к нулю» требуют определенного уточнения. Например, близко ли значение 10000 к нулю? Если сравнивать с десятью миллиардами, то да!

Не нужно использовать неопределенные выражения вроде «небольшое количество». В большинстве случаев при выборе индексов на основе битовых карт необходимо учитывать только 2 фактора. Во-первых, количество различных блоков в таблице, в которых может находиться типичное значение индекса - это основной фактор выбора отдельного индекса. Изменение структуры индекса с B*-дерева на набор битовых карт не сделает этот индекс в это отношение лучше чудесным образом. Во-вторых, используемый оптимизатором Oracle механизм комбинирования нескольких битовых индексов делает их действительно полезными.

Рассмотрим следующий пример, основанный на данных по примерно 17-миллионному населению Казахстана [4]:

9 млн. имеют карие глаза
8 млн. - женщины
13 млн. - темноволосые
0,5 млн. живут в центре Алматы

0,8 млн. имеют возраст 25 лет
750 тыс.работают в Алматы

Каждому критерию отдельно соответствует очень много людей, но сколько кареглазых, темноволосых женщин в возрасте 25 лет живет в центре Алматы и работают в Алматы? Приблизительно 2 десятка. Моделируем население Казахстана следующим блоком:

```
create table junk asselect rownum id
from all_objects
where rownum <= 8000;
create table t1 nologging pctfree 0
as select /*+ ordered use_nl(v2) */
«x» facts,
mod(rownum,2) sex,
mod(rownum,3) eyes,
mod(rownum,7) hair,
mod(rownum,31) town,
mod(rownum,47) age,
mod(rownum,79) work,
from
junk v1, junk v2;
create bitmap index i1 on t1(sex) nologging pctfree 0;
create bitmap index i2 on t1(eyes) nologging pctfree 0;
create bitmap index i3 on t1(hair) nologging pctfree 0;
create bitmap index i4 on t1(town) nologging pctfree 0;
create bitmap index i5 on t1(age) nologging pctfree 0;
create bitmap index i6 on t1(work) nologging pctfree 0;
analyze table t1 estimate statistics;
```

Отдельный индекс (будь то на основе B*-дерева или битовых карт) по любому из этих столбцов будет абсолютно бесполезен для выполнения такого запроса к таким данным в СУБД Oracle.

Многостолбцовый индекс на основе B*-дерева по соответствующим 6 столбцам может существенно помочь, пока нас не заинтересуют мужчины ростом 180 см с бородой вместо темноволосых и кареглазых женщин. Для апробации данного блока моделирования понадобится около 2,0 Гбайт места на диске и

пару часов работы процессора с тактовой частотой порядка 500 МГц.

Для модели, которая эмулирует население порядка 36 млн., время построения и размеры объектов для компьютера с тактовой частотой процессора 600 МГц, ОС Win2000 и сервером Oracle версии 9.2.0.1 представлены в следующей таблице.

Всего лишь один многостолбцовый индекс по тем же 6 столбцам (даже с максимальным сжатием) займет минимум 430 Мбайт, а для его построения потребуется установить параметру sort_area_size значение около 900 Мбайт, в связи с этим данный процесс труднореализуем.

Объект	Размер, Мбайт	Время построения, мин:сек
T1	845	16:12
I1 (sex)	11	1:39
I2 (eyes)	16	1:43
I2 (hair)	37	2:17
I4 (town)	40	2:25
I5 (age)	42	2:28
I6 (work)	45	2:42

Рассмотрим запрос:

```
select count(facts) from t1
here eyes = 1 and sex = 1 and hair = 1
and town = 15 and age = 25 and work = 40;
```

На сокращенном наборе данных при вводе подсказки, требующей выполнения полного просмотра таблицы, запрос выполняется 1 мин. 20 с (вернув ответ 8). Конечно, при реальном наборе фактических данных таблица была бы существенно больше, как и время выполнения.

При использовании полного шестистолбцового индекса объемом 430 Мбайт этот запрос выполнился бы, вероятно, за время, необходимое для выполнения около 10 физических чтений (один блок таблицы для каждой строки и пару дополнительных блоков индекса) - буквально за доли секунды.

При наличии заданных ранее битовых индексов запрос выполнялся в течение 5 с. Большая часть времени ушла на сканирование диапазонов индексов, требующих физического считывания блоков индекса в память. Фактический план выполнения представлен в следующем блоке:

```
SORT (AGGREGATE)
TABLE ACCESS (BY INDEX ROWID) OF T1
BITMAP CONVERSION (TO ROWIDS)
BITMAP AND BITMAP INDEX (SINGLE VALUE) OF I6
BITMAP INDEX (SINGLE VALUE) OF I5
BITMAP INDEX (SINGLE VALUE) OF I4
```

В этом плане выполнения запроса следует отметить два интересных момента. Во-первых, сервер Oracle проигнорировал 3 «худших» (т. е. наименее избирательных) индекса. Во-вторых, хотя время выполнения - заметное, но размеры индексов настолько малы, что имеет смысл подумать об их размещении в достаточно большом KEEP-пуле, buffer_pool_keep (в Oracle 9 его размер задается параметром db_keep_cache_size), чтобы избежать физических чтений. Этот вариант вряд ли подходит для нескольких многостолбцовых индексов на основе B*-дерева, поддерживающих такие же запросы.

Бывали случаи, когда сервер Oracle использовал более 5 проигнорированных индексов (это предел для способа доступа and_equal при использовании одностолбцовых индексов на основе B*-дерева). В плане может использоваться практически любое количество индексов на основе битовых карт.

Три оставшихся индекса были проигнорированы не из-за какого-то искусственного ограничения. Стоимостный оптимизатор сравнил стоимость чтения каждого дополнительного индекса с достигаемой дополнительной точностью, и не выбрал их. Это означает, что индексы на основе битовых карт по классическому столбцу (пол) обычно игнорируются, несмотря на противоположные утверждения. (Удалите конструкцию work = 40 из запроса и убедитесь, что индекс по столбцу sex в этом случае используется.)

Конечно, такие индексы на основе битовых карт можно построить очень быстро, и места они займут немного, так что их

можно создавать просто на всякий случай.

Размер индексов и возможность максимальной буферизации необходимо, конечно, учитывать при определении стоимости, вот почему часто возникает вопрос, насколько большим будет индекс на основе битовых карт?

В представленном выше примере можно попытаться воссоздать наихудший случай, максимально затруднив серверу Oracle получение преимуществ от сжатия.

В худшем случае размер битовой карты в битах будет составлять:

- количество различных возможных значений для столбца *
- количество строк, которое может поместиться в блок, по мнению сервера Oracle*, количество блоков до отметки максимального уровня.

Чтобы получить размер в байтах, нужно добавить около 10 % на информацию контрольных сумм и накладные расходы и поделить на 8.

Сервер Oracle позволяет предпринять ряд действий для сокращения размера индекса, наиболее действенным из которых является команда, сообщающая серверу, сколько в точности строк в блоке помещается в худшем случае для данной таблицы:

```
Alter table XXX
minimize records_per_block;
```

Однако помимо информирования сервера Oracle с помощью этой команды, очень существенное влияние на размер индекса имеет кластеризация данных.

Нами созданы максимально разбросанные данные. Например, столбец town последовательно получает значения от 0 до 30. Если реструктурировать (по сути, отсортировать) данные так, что информация для всех городов с кодом 0 хранится вместе, а затем предлагается вся информация для городов с кодом 1, размер индекса можно сократить с 40 Мбайт практически до 7 Мбайт.

Таким образом, об индексах на основе битовых карт есть несколько существенно ошибочных представлений. Некоторые из них могут приводить к отказу от использования этих индексов, когда они могут существенно помочь. Другие приводят к

созданию абсолютно бесполезных битовых индексов.

К счастью, сделать серьезные ошибки при работе с индексами на основе битовых карт достаточно сложно. Но хорошее понимание того, как работает с ними сервер Oracle поможет получить от них максимальную пользу.

Надо запомнить следующие ключевые факты:

- Если индекс на основе B*-дерева не является эффективным механизмом доступа к данным, маловероятно, что он станет намного эффективнее, если создавать индекс на основе битовых карт.
- Индексы на основе битовых карт обычно создаются быстрее и могут занимать удивительно мало места.
- Размер индекса на основе битовых карт существенно зависит от распределения данных.

Литература

1. *Луни К., Брила Б.* Администрирование Oracle 10g. Лори. 2009 г.
2. *Урман С., Хардман Р.* Программирование PL/SQL. Лори. 2007 г.
3. *Милсап К., Хольт Д.* Оптимизация производительности Оракл // Символ-Плюс, 2006.
4. http://www.languages-study.com/demography/kazakhstan.html