Как обязать СУБД применять к запросам конкретные приемлемые планы

Владимир Пржиялковский
Преподаватель технологий Oracle
prz@yandex.ru
http://open.oracle.tu2.ru/

Сентябрь 2009 г.

А мои ти куряни – сведоми къмети:
подъ трубами повити,
подъ шеломы възлелеяны,
конець копия въскръмлени;
пути имь ведоми,
яругы имь знаеми,
луци у нихъ напряжени,
тули отворени,
сабли изъстрени;
сами скачють, акы серыи влъци въ поле,
ищучи себе чти, а князю славе.

Слово о полку Игореве

Реферат

Рассматривается система управления планами (SPM), введенная в версии Oracle 11 применительно к повторяющимся запросам приложения. Она позволяет формировать и хранить для запросов допустимые наборы планов (baselines), заставить СУБД работать только по ним и тем самым избежать в отдельных случаях непредусмотренного падения производительности при обработке.

Введение

Как известно, СУБД, получив от приложения запрос, сначала строит программу вычисления ответа («план»), и тут же эту программу отрабатывает. Теоретически план можно построить наилучшим образом: получить множество всех возможных для данного запроса планов и выбрать из них наиболее поизводительный в нужном отношении. На практике же любая промышленная СУБД, включая Oracle, вырабатывает план на скорую руку, достигая при этом приемлемого времени ответа, но зато жертвуя точностью решения задачи. Как следствие применяемые к поступающим запросам планы нередко оказываются не самыми лучшими, и общая производительность СУБД страдает.

Для решения этой проблемы фирма Oracle, равно как и прочие разработчики, дает пользователю средства вмешиваться в выработку плана СУБД. Косвенное вмешательство достигается воздействием на факторы, влияющие на выработку оптимизатором планов: параметры СУБД и сеанса, показатели статистики объектов запроса, употребление подсказок. Прямое вмешательство состоит в диктовке СУБД, какие именно планы следует использовать для конкретных запросов.

Исторически первым инструментом последней категории стали «очертания» (outline) запросов, появившиеся в версии 8.1. «Очертание» запроса – это план, «схваченный» в подходящий момент времени и сохраненный для последующего употребления. Пользователь получает возможность в пределах конкретных сеансов «включать» «очертание», заставляя СУБД пользоваться этим конкретным планом независимо от текущих обстоятельств. Последней версией, где эта техника фиксации плана поддерживается, официально считается Oracle 11.

Вторым по времени инструментом стал «профиль» (profile) запроса, введенный в версии 10. Профили могут возникнуть в результате специальной процедуры анализа СУБД запроса, представляющего интерес, в рамках работы советника SQL Tuning Advisor. Пользователь имеет право в любой момент включить имеющийся профиль, и тогда текущий план запроса подкорректируется в сторону улучшения. Влиянием на применяемый СУБД план будет включение и отключение профиля по мере необходимости.

В версии 11 в Oracle появилось третье по счету средство указания СУБД, каким оприделенным планом ей воспользоваться при обработке того или иного запроса. Оно получило название «управления планами»: SQL plan management (SPM). Его основное назначение – не дать возможность пустить обработку повторяющихся запросов приложения по «плохому» плану. «Плохим» же план может оказаться ненароком при смене обстоятельств очередного поступления запроса – значений переменных привязки, статистики объектов, переменных СУБД или сеанса и даже версии СУБД или ОС.

Система управления планами запросов

SPM позволяет формировать «базу управления запросами» (SQL management base, SMB). Она располагается в AWR (automatic workload repository) и может пополняться и вычищаться как вручную, так и автоматикой AWR. В SMB для каждого представляющего интерес запроса можно накапливать *историю* его *планов* (plan history), а из нее формировать *основную линию* (baseline) планов. Суть «основной линии» в том, администратор формирует ее из планов, которые полагает («назначает») удовлетворительно производительными. Управление планами обработки на деле начнется при переводе оптимизатора CBO в специальный режим «учета SPM» (SPM aware optimizer). Тогда при любой смене обстоятельств запуска запроса (или всего приложения) оптимизатор не применит к нему план, хуже имеющихся в «основной линии».

Режим учета SPM (использования основных линий планов) устанавливается значением TRUE параметра СУБД OPTIMIZER_USE_SQL_PLAN_BASELINES. Это значение умолчательное. Изменить его на FALSE или обратно можно как на уровне СУБД, так и отдельных сеансов.

При поступлении в СУБД запроса, для него вырабатывается план. Далее, если OPTIMIZER_USE_SQL_PLAN_BASELINES = FALSE, запрос выполняется по этому плану. Если = TRUE и план присутствует и в истории планов, и в основной линии, он также принимается к исполнению. Если же план отсутствует либо в истории, либо в основной линии, для исполнения запроса будет выбран наиболее «легкий» из имеющихся в основной линии. Но если план отсутствует в истории, он дополнительно будет туда занесен.

Содержимое SMB представлено в таблице DBA_SQL_PLAN_BASELINES. (На деле это, конечно, виртуальная таблица, то есть view с запросом, предъявляющим данные из реальных таблиц SQLOBJ$, SQLOBJ$AUXDATA и SQL$TEXT схемы SYS). Эти данные общесистемные, а потому аналогичных таблиц с префиксами USER и ALL в названиях не существует. Вот некоторые поля этой таблицы:

Поле	***Значение***
SIGNATURE	«Подпись» запроса, вычисляемая по нормализованному тексту запроса (см. функцию DBMS_SQLTUNE.SQLTEXT_TO_SIGNATURE).
SQL_TEXT	Текст запроса.
SQL_HANDLE	Символьное выражение подписи, ключ для удобства поиска планов основной линии.
PLAN_NAME	Символьное выражение для обозначения плана.
ENABLED	Признак нахождения плана в рабочем состоянии. Если установить = 'NO', оптимизатор будет этот план игнорировать.
ACCEPTED	Признак того, что план включен в основную линию как приемлемый.
FIXED	Если в основной линии есть планы, помеченные FIXED = 'YES', считается, что основная линия для запроса не подлежит автоматической перестройке, то есть является фиксированой.
AUTOPURGE	Признак, разрешающий автоматическое удаление плана из SBM автоматикой AWR по прошествии установленного времени.
OPTIMIZER_COST, EXECUTIONS, CPU_TIME и др.	Общие количественные показатели плана.

Помимо этого сведения об основной линии планов для запроса предоставляет в текстовом виде функция DISPLAY_SQL_PLAN_BASELINE из пакета DBMS_XPLAN.

Совершению действий с SPM служит пакет DBMS_SPM. Этот же пакет используется в OEM для графического доступа к своей собственной функциональности.

Далее SPM рассматривается на примере употребления непосредственно программным образом.

Подготовка к примеру

Далее приводится пример, где в обстоятельства выдачи приложением запроса вносятся изменения в виде добавленного индекса. Положим, мы не очень уверены, как это отразится на обработке запроса. Дабы не просчитаться с непредвиденной потерей эффективности, построим для него основную линию, включающую прежний проверенный план. Таким образом после добавления в БД индекса запрос заведомо не ухудшит производительности, но возможно улучшит.

Для примера подобран нереально простой запрос. Это сделано осознано воимя доходчивости изложения техники.

В примере будут переключения в схемы SCOTT и SYS, но предполагается, что работа выполняется в SQL*Plus без выхода из этой программы, что позволит сохранить значения переменных.

Очистим для предотвращения путаницы общую область курсоров в shared pool (технически это необязательно, но упростит здесь обращение к нужным данным в shared pool) , заведем рабочие переменные и сбросим ради простоты показа в файл текст для выдачи плана последнего запроса:

```
CONNECT / AS SYSDBA
ALTER SYSTEM FLUSH SHARED_POOL;
VARIABLE retcode   NUMBER
VARIABLE sqltext   VARCHAR2 ( 1000 )
VARIABLE sqlhandle VARCHAR2 ( 30 )
VARIABLE sqlid     VARCHAR2 ( 13 )
VARIABLE report    CLOB
SELECT * FROM TABLE ( DBMS_XPLAN.DISPLAY_CURSOR ( format => 'basic' ) )
.
SAVE showplan REPLACE
```

Предполагается, что основная линия запросов изначально пуста. Исходно план запроса не зависит от того, применяет оптимизатор управление планами, или нет:

```
SQL> CONNECT scott/tiger
Connected.
SCOTT> SELECT job FROM emp WHERE ename = 'MILLER';

JOB
---------
CLERK

SQL> @showplan

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------

EXPLAINED SQL STATEMENT:
------------------------
SELECT job FROM emp WHERE ename = 'MILLER'

Plan hash value: 3956160932

---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
```

```
|   1 |  TABLE ACCESS FULL| EMP  |
---------------------------------

13 rows selected.

SQL> ALTER SESSION SET optimizer_use_sql_plan_baselines = FALSE;

Session altered.

SQL> SELECT job FROM emp WHERE ename = 'MILLER';

JOB
---------
CLERK

SQL> @showplan

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------

EXPLAINED SQL STATEMENT:
------------------------
SELECT job FROM emp WHERE ename = 'MILLER'

Plan hash value: 3956160932

---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
|   1 |  TABLE ACCESS FULL| EMP  |
---------------------------------
```

Загрузка плана в базу управления запросами

Вот случай, достойный памяти и занесения в литературу !

Плиний Младший, Панегирик императору Траяну

Сейчас для интересующего нас запроса СУБД завела рабочую память в общей области курсоров в shared pool. Загрузим оттуда план (первый по счету) в основную линию в базе управления запросами SMB, сославшись на идентификатор курсора SQL ID:

```
CONNECT / AS SYSDBA
EXECUTE :sqltext := q'[SELECT job FROM emp WHERE ename = 'MILLER']'
EXECUTE -
 SELECT sql_id INTO :sqlid FROM v$sqlarea WHERE sql_text = :sqltext
EXECUTE :retcode := DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE ( :sqlid )
```

Проверка:

```
SQL> SELECT :retcode "Plans selected:" FROM dual;

Plans selected:
---------------
              1
```

Для простоты последующих обращений к таблице DBA_SQL_PLAN_BASELINES за сведениями о SMB запомним в файле еще один рабочий запрос. Он параметризован ключом прикладного запроса, который узнаем по тексту запроса и поместим в переменную SQLHANDLE:

```
COLUMN sql_text FORMAT A15 WRAP
COLUMN enabled  FORMAT A10
COLUMN accepted FORMAT A10
SELECT
   plan_name, sql_text, enabled, accepted
FROM
   dba_sql_plan_baselines
WHERE sql_handle = &1
.
SAVE baseline REPLACE
SET VERIFY OFF

BEGIN
 SELECT DISTINCT sql_handle INTO :sqlhandle
 FROM   dba_sql_plan_baselines
 WHERE  DBMS_SQLTUNE.SQLTEXT_TO_SIGNATURE ( :sqltext ) = signature;
END;
/
```

Проверка:

```
SQL> PRINT sqlhandle

SQLHANDLE
--------------------------------
SYS_SQL_dd7adbcd38c100c0

SQL> @baseline :sqlhandle

PLAN_NAME                      SQL_TEXT        ENABLED    ACCEPTED
------------------------------ --------------- ---------- ----------
SYS_SQL_PLAN_38c100c0d8a279cc  SELECT job FROM YES        YES
                                emp WHERE enam
                               e = 'MILLER'
```

Несмотря на то, что в нашем случае запрос попал в SMB по ссылке на SQL ID, в самой базе он идентифицируется ключом SQL_HANDLE, который автоматически порождается по подписи запроса, в свою очередь вычисляемой по нормализованому тексту. Это позволяет хранить план в AWR долговременно, независимо от того, представлен ли запрос вообще в курсорной области в данный момент, и под каким именно SQL ID представлен.

Обратите внимание, что использованный способ загрузки плана в основную лонию автоматически выставил признаки ENABLED и ACCEPTED в состояние 'YES', то есть единственный пока план в SMB и в рабочем состоянии, и включен в основную линию.

Использование основной линии планов запроса оптимизатором CBO

Изменим обстоятельства запуска запроса, индексировав таблицу. План должен поменяться. Однако при включенном (по умолчанию) управлении планами мы этого не увидим. Примечательно, что убедиться в этом удастся только со второй попытки:

```
SQL> CONNECT scott/tiger
Connected.
SQL> create index emp_ename on emp ( ename );

Index created.

SQL> SELECT job FROM emp WHERE ename = 'MILLER';

JOB
---------
CLERK
```

```
SQL> @showplan

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------

SQL_ID  a7zgruuhu1nkf, child number 3

An uncaught error happened in prepare_sql_statement : ORA-01403: no data
found

NOTE: cannot fetch plan for SQL_ID: a7zgruuhu1nkf, CHILD_NUMBER: 3
      Please verify value of SQL_ID and CHILD_NUMBER;
      It could also be that the plan is no longer in cursor cache (check v
$sql_plan)


8 rows selected.

SQL> SELECT job FROM emp WHERE ename = 'MILLER';

JOB
---------
CLERK

SQL> @showplan

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------

EXPLAINED SQL STATEMENT:
------------------------
SELECT job FROM emp WHERE ename = 'MILLER'

Plan hash value: 3956160932

----------------------------------
| Id  | Operation         | Name |
----------------------------------
|   0 | SELECT STATEMENT  |      |
|   1 |  TABLE ACCESS FULL| EMP  |
----------------------------------
```

Первый раз оптимизатор построил новый план, с учетом индекса, но в SMB его не обнаружилось. Тогда оптимизатор занес план в историю и выполнил запрос по единственному в основной линии плану – старому. Со второго раза рабочая область в shared pool оказалась заведена, но запрос по-прежнему был отработан по единственному в основной линии старому плану. Если же управление планами отключить, СУБД отработает по более выгодному в этой версии оптимизатора новому плану:

```
SQL> ALTER SESSION SET optimizer_use_sql_plan_baselines = FALSE;

Session altered.

SQL> SELECT job FROM emp WHERE ename = 'MILLER';

JOB
---------
CLERK

SQL> @showplan

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
```

```
EXPLAINED SQL STATEMENT:
------------------------
SELECT job FROM emp WHERE ename = 'MILLER'

Plan hash value: 106684950

-----------------------------------------------
| Id  | Operation                   | Name      |
-----------------------------------------------
|   0 | SELECT STATEMENT            |           |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP       |
|   2 |   INDEX RANGE SCAN          | EMP_ENAME |
-----------------------------------------------
```

Пополнение основной линии планов путем оценки планов

По результату выполненых действий в основной линии планов для нашего запроса оказалось два плана: с учетом индекса (признак ACCEPTED = 'NO') и без учета (признак ACCEPTED = 'YES'):

```
SQL> CONNECT / AS SYSDBA
SQL> @baseline :sqlhandle

PLAN_NAME                      SQL_TEXT        ENABLED    ACCEPTED
------------------------------ --------------- ---------- ----------
SYS_SQL_PLAN_38c100c08916fd8c  SELECT job FROM YES        NO
                               emp WHERE enam
                               e = 'MILLER'

SYS_SQL_PLAN_38c100c0d8a279cc  SELECT job FROM YES        YES
                               emp WHERE enam
                               e = 'MILLER'
```

Можно выдать оптимизатору задание проверить с планы признаками ACCEPTED = 'NO' (то есть учтеные в SMB, но не причисленые к приемлемым) на эффективность и включить их в основную линию (пометить как «приемлемые»), если они окажутся не хуже ранее там имевшихся:

```
BEGIN
 SELECT DISTINCT sql_handle INTO :sqlhandle
 FROM   dba_sql_plan_baselines
 WHERE  DBMS_SQLTUNE.SQLTEXT_TO_SIGNATURE ( :sqltext ) = signature;
END;
/
EXECUTE :report :=                                                -
    DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE ( sql_handle => :sqlhandle )
```

Первую проверку изменений в SMB выполним по признаку ACCEPTED в справочной таблице:

```
SQL> @baseline :sqlhandle

PLAN_NAME                      SQL_TEXT        ENABLED    ACCEPTED
------------------------------ --------------- ---------- ----------
SYS_SQL_PLAN_38c100c08916fd8c  SELECT job FROM YES        YES
                               emp WHERE enam
                               e = 'MILLER'

SYS_SQL_PLAN_38c100c0d8a279cc  SELECT job FROM YES        YES
                               emp WHERE enam
                               e = 'MILLER'
```

Вторую проверку выполним по содержимому переменной REPORT, составленному функцией EVOLVE_SQL_PLAN_BASELINE:

```
SQL> SET LONG 10000
SQL> SELECT :report "Baseline evolution results:" FROM dual;

Baseline evolution results:
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
                        Evolve SQL Plan Baseline Report
--------------------------------------------------------------------------------

Inputs:
-------
  SQL_HANDLE = SYS_SQL_dd7adbcd38c100c0
  PLAN_NAME  =
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = YES

Plan: SYS_SQL_PLAN_38c100c08916fd8c

-----------------------------------
  Plan was verified: Time used .06 seconds.
  Passed performance criterion: Compound improvement ratio >= 1.5.
  Plan was changed to an accepted plan.

                     Baseline Plan      Test Plan     Improv. Ratio
                     -------------      ---------     -------------
  Execution Status:        COMPLETE       COMPLETE
  Rows Processed:                 1              1
  Elapsed Time(ms):               0              0
  CPU Time(ms):                   0              0
  Buffer Gets:                    3              2             1.5
  Disk Reads:                     0              0
  Direct Writes:                  0              0
  Fetches:                        0              0
  Executions:                     1              1

--------------------------------------------------------------------------------
                                  Report Summary
--------------------------------------------------------------------------------
Number of SQL plan baselines verified: 1.
Number of SQL plan baselines evolved: 1.
```

Вердикт о влючении плана в основную линию производится на основании сравнения взвешенных суммарных оценок по перечисленным в отчете показателям («Compound improvement ratio»; точное правило не оглашается). В нашем случае он оказался благоприятным для плана-кандидата (коэффициент улучшения эффективности >= 1.5).

Теперь можно проверить прежним манером, как появление второго плана в основной линии отразилось на отработке запроса, выдав последовательность:

```
CONNECT scott/tiger
SELECT job FROM emp WHERE ename = 'MILLER';
@showplan
ALTER SESSION SET optimizer_use_sql_plan_baselines = FALSE;
SELECT job FROM emp WHERE ename = 'MILLER';
@showplan
```

На этот раз планы окажутся одинаковыми, «новыми» (INDEX RANGE SCAN → TABLE ACCESS BY INDEX ROWID).

Прочие способы управления основной линией планов

Пополнение и ручная чистка основной линии планов

Основную линию планов какого-нибудь запроса можно пополнять («развивать», evolve): вручную либо автоматически.

Ручное пополнение основной линии в результате запуска задания на проверку приемлемости плана выполняется функцией EVOLVE_SQL_PLAN_BASELINE и демонстрировалось выше.

Процедура LOAD_PLANS_FROM_SQLSET позволяет загружать основную линию планы из настроечного набора (SQL tuning set). Настроечный набор может быть получен любым доступным путем, например перенесен из другой БД, возможно даже из версии 10.

Процедуры PACK_STGTAB_BASELINE и UNPACK_STGTAB_BASELINE разрешают сохранить планы основных линий в специально созданной таблице и загружать их из такой таблицы.

Включение в сеансе параметра СУБД OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES в состояние TRUE вызовет *автоматическое* пополнение SMB планами запросов, поступающих из приложения. Для запросов приложения, основные линии планов которых желательно исключить из процедуры автоматического пополнения (то есть «зафиксировать»), можно использовать значение атрибута FIXED = 'YES' планов, составляющих соответствующую линию. Наличие планов с атрибутом FIXED = 'YES' препятствует только автоматическому пополнению и не сказывается на возможности добавлять планы вручную, по SQL ID и по настроечному набору.

Автоматическое пополнение основных линий также может осуществляться в результате «одобрения» (принятия) администратором профиля, рекомендованного для запроса советником SQL Tuning Advisor. По умолчанию этот советник запускается автоматическим заданием в «окошко поддержки» СУБД ежесуточно.

Путь попадания плана в основную линию обозначен в таблице DBA_SQL_PLAN_BASELINES в поле ORIGIN:

```
SQL> SELECT
  2     sql_handle, plan_name, origin
  3  FROM
  4     dba_sql_plan_baselines
SQL> ;

SQL_HANDLE                PLAN_NAME                      ORIGIN
------------------------- ------------------------------ ------------
SYS_SQL_dd7adbcd38c100c0 SYS_SQL_PLAN_38c100c08916fd8c AUTO-CAPTURE
SYS_SQL_dd7adbcd38c100c0 SYS_SQL_PLAN_38c100c0d8a279cc MANUAL-LOAD
```

Ручное удаление плана из основной линии выполняется функцией DROP_SQL_PLAN_BASELINE.

Изменение свойств планов в SMB

Процедура ALTER_SQL_PLAN_BASELINE позволяет устанавливать атрибутам ENABLED, FIXED и AUTOPURGE плана требуемые значения явочным порядком. Пример:

```
EXECUTE :retcode := DBMS_SPM.ALTER_SQL_PLAN_BASELINE                -
    ( :sqlhandle, 'SYS_SQL_PLAN_38c100c08916fd8c', 'enabled', 'no' )

SELECT :retcode "Plans disabled:" FROM dual;

@baseline :sqlhandle
```

Теперь система управления планами снова откажется применять «новый» план к запросу, несмотря на его присутствие в основной линии планов – благодаря нерабочему состоянию.

Регулирование накопления и хранения планов в основных линиях

Технически основные линии планов в составе SMB хранятся в AWR в табличном пространстве SYSAUX, и регламент их хранения определяется внутренними проверками и автоматикой AWR. Этот регламент задается следующими характеристиками:

- максимальным процентом данных SMB в пространстве SYSAUX (от 1 до 50%, исходно 10%);
- максимальным периодом отсутствия интереса к плану, после чего он будет автоматически удален (от 5 до 523 недель, исходно 53).

Узнать текущие характеристики регламента накопления и хранения SMB можно запросом:

```
SELECT
   parameter_name, parameter_value
FROM
   dba_sql_management_config
;
```

Изменить эти характеристики можно процедурой CONFIGURE, например:

```
EXECUTE DBMS_SPM.CONFIGURE (                  -
   parameter_name  => 'space_budget_percent' -
,  parameter_value => 20                      )
```

Уроки системы управления планами в Oracle

Система управления планами в Oracle способна не только сохранить производительность при смене обстоятельств запуска запросов, но и преподать пользователю уроки.

Рассмотрим план для запроса SELECT DISTINCT … (далее все аналогично для запросов с UNION и GROUP BY). Как известно, в версиях до 9 включительно этот запрос обрабатывался с применением внутренней сортировки SORT UNIQUE. С версии 10 оптимизатор предлагает для такого запроса план с HASH UNIQUE, внутренней процедурой расстановки строк, с «хешированием». Большинство пользователей, обративших на это внимание, посчитали его целесообразным новшеством, улучшающим производительность отработки. Однако попытка применить для таких запросов управление планами (хотя бы ради сохранения производительности) заставляет в этом усомниться.

Действительно, попробуем в стиле вышеизложеного построить основную линию планов для другого отправного случая и другого запроса. Выдадим:

```
ALTER SESSION SET optimizer_features_enable = '9.2.0';
ALTER SESSION SET optimizer_mode = ALL_ROWS;
SELECT DISTINCT job FROM emp;
```

Включим план в основную линию, как выше. Это будет план с SORT UNIQUE. Поменяем обстоятельства выдачи запроса, например:

```
ALTER SESSION SET optimizer_features_enable = '11.1.0.6.1';
SELECT DISTINCT job FROM emp;
```

Получится план с HASH UNIQUE. Однако попытка дополнить им основную линию планов запроса функцией EVOLVE_SQL_PLAN_BASELINE обречена. SPM не считает новый план, который дают версии 10+ для этого запроса, лучше прежнего !

Более пристальное изучение обнаруживает, что план с HASH UNIQUE имеет бо́льшую стоимость обработки (cost), нежели «старый» с SORT UNIQUE (10 единиц против 5), хотя с ростом размера таблицы этот проигрыш и сокращается.

Я не нашел объяснения этому явлению, однако если здесь нет подводных камней, система управления планами способна и в этом случае предотвратить неожиданый нежелательный рост трат на обработку.