

Oracle 数据库 10g 第 2 版中的 DSS 性能

Oracle 白皮书

2005 年 7 月

Oracle 数据库 10g 第 2 版中的 DSS 性能

执行概要	5
引言	5
测试环境	5
内存排序	8
基于散列的聚合	10
分区表	11
目标检查点	15
SQL 内建数据挖掘应用功能	16
支持向量机 (SVM) 算法增强特性	19
结论	22

Oracle 数据库 10g 第 2 版中的 DSS 性能

执行概要

现在的数据仓库要求使用高性能数据库管理系统作为主体架构。Oracle 数据库 10g 第 2 版新增了一系列特性和增强功能，实现了性能提升，可满足不断增长的数据~~存储~~仓库需求。升级到 Oracle 数据库 10g 第 2 版后即得获得性能提升，无需更改任何应用程序，也不用增加部署费用。该技术白皮书描述了这些新特性和增强特性，说明了它们带来的性能收益。

引言

每天机构要收集无数业务方面的数据，无数有关客户、~~运营~~运营、产品和员工的全~~部~~数据。如果机构能够有效地利用数据改进日常运作流程、建立与投资入间的稳固关系，以及满足客户需求（也是最重要的），那么它们将有机会获取极大的成功。数据仓库和商务智能已经成为许多企业成功的基础。

当公司竭尽所能收集更多的商务智能来赢得竞争优势时，它们的数据仓库也日渐膨胀。数据仓库体积在急剧增加，同时变得越来越复杂，这不断挑战数据库管理系统的功能，迫使数据库管理系统更为迅速地提供的更为准确的信息，并对不断变化的业务需求快速做出响应。

Oracle 数据库 10g 第 2 版正是在这个背景下应运而生的，它提供高性能系统，满足现在快速增长的数据~~存储~~仓库需求。该版的新特性和增强特性显著提升了数据~~存储~~仓库和商务智能的性能。这些新特性和增强特性在升级到 Oracle 数据库 10g 第 2 版之后，将顺利地~~和~~与现有数据~~存储~~仓库应用程序集成，而不需要数据~~存储~~仓库团队进行任何新的实施或操作。

该技术白皮书描述了这些新特性和增强特性。本白皮书通过具体比较 Oracle 数据库 10g 第 1 版和第 2 版的查询性能来说明性能收益。在该白皮书的后面的部分首先介绍测试环境，接着描述各种测试，这些测试展示了新特性（包括新内存排序算法、散列聚合特性、目标检查点特性、分区表的增强特性）带来的性能提升。本白皮书还说明了两个数据挖掘特性及其实现的性能提升。最后进行总结。

测试环境

本白皮书中的测试系统使用有超线程技术的 2x2.8 GHz Intel Xeon 处理器。数据库在 OCFS 文件系统中，该系统分布于单个磁盘数组中的 10 个

磁盘上。图 1 描述的数据库模式模型，包含来自 Oracle 示例模式模型的销售历史模式模型，和几个其它独立的表格，添加这些表格的目的是进行数据挖掘。

销售表含包含了 7 年（从 1995 年 1 月到 2001 年 12 月）的销售信息，数据总量约为 10GB。SALES 表和 COST 表都经过压缩，而且是按照 time_id 进行范围分区的，分区的总量为 20。下表列出了各个表的行数和行的平均长度。

表	行数	行的平均长度
SALES	300,000,000	31
COSTS	14,600,000	20
CUSTOMERS	50,000	138
PRODUCTS	10,000	241
TIMES	1461	177
PROMOTIONS	501	67
CONUTRIES	19	33
CHANNNELS	5	19
PRODUCTS_AFFINITY_CARD_B	50,000	1,151
PRODUCTS_AFFINITY_CARD_A	2,000,000	1,151
SUPPLEMENTARY_DEMOGRAPHICS_B	32,561	107
SUPPLEMENTARY_DEMOGRAPHICS_T	16,281	107
SUPPLEMENTARY_DEMOGRAPHICS_A	97,684	107

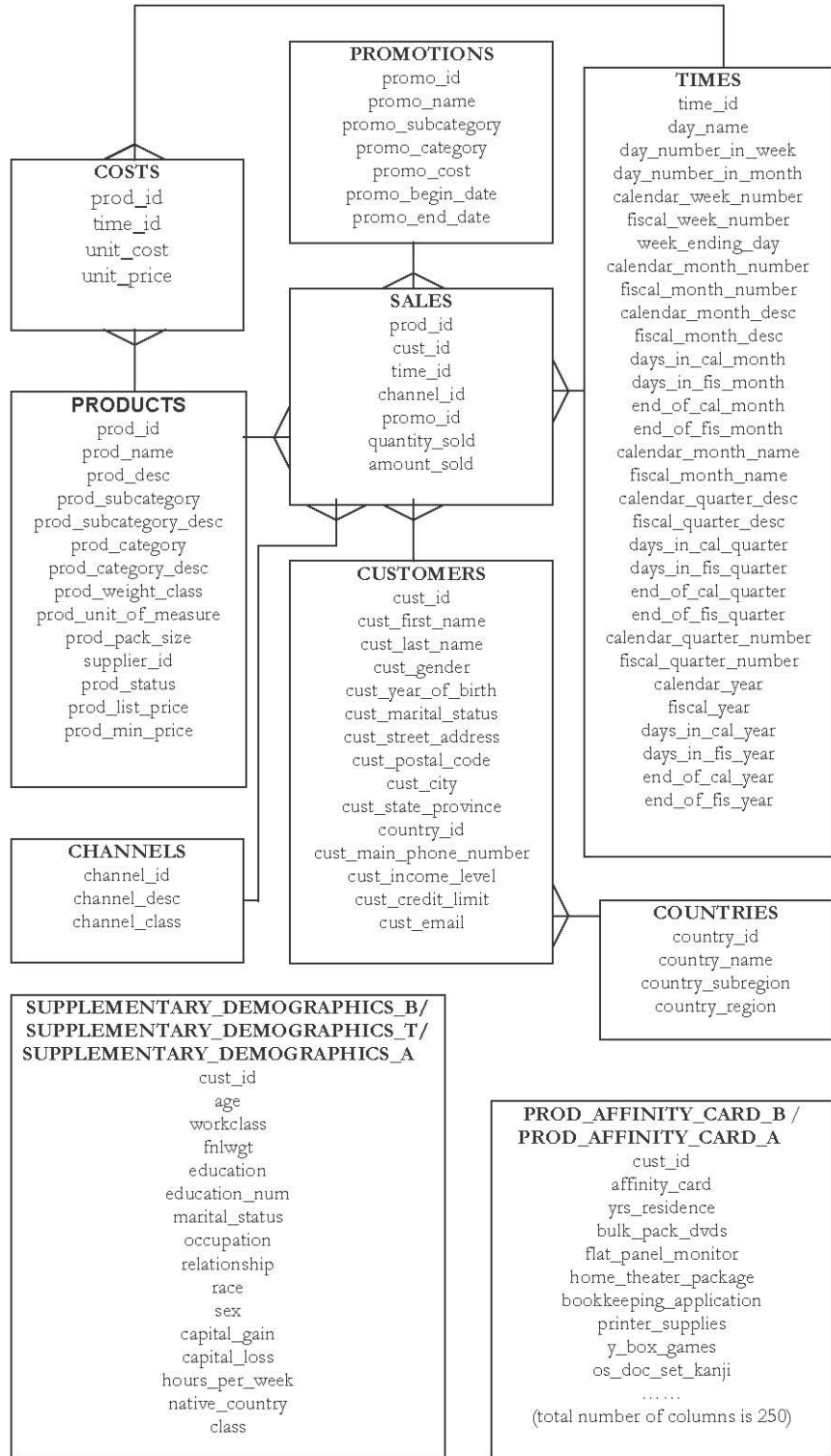


图 1. 数据库模式模型

下面各个部分介绍了 Oracle 数据库 10g 第 2 版中的新特性，测试案例是在以上模式模型上建立起来的。

内存排序

Oracle 数据库 10g 第 2 版引入了新内存排序的算法，与 Oracle 数据库 10g 第 1 版中的现有类算法相比，该算法能够进行更快的排序，具有更为良好的内存系统局部性。

新的内存排序算法对各种查询的性能进行了改进。这种算法可以用于以下情形中：

- 带有“order by”子句的查询
- 使用“分类合并联接”的查询
- 使用“分区外部联接”的查询
- 创建索引

新内存排序算法不支持有垃圾收集、分段流的列，或大于 30,000 字节的列，即使这些列是非关键字列。对于带有关键字的列来说，新的内存排序不支持那些需要进行语义对比(例如逻辑 rowid 和带有时区的时间)的列。

在某些情况下，老的排序算法性能会随着可用内存的增加下降，直至某个极限。新内存排序算法可以解决该问题，因为其具备优秀的内存系统局部性。

第一组测试显示了几个典型查询的性能改进，这些查询受益于新内存排序。测试时您可以将 `workarea_size_policy` 设置为 `AUTO`。第二组测试展示了新内存排序算法的内存系统局部性，方法是将 `workarea_size_policy` 设置为手动，使用不同的 `sort_area_size` 运行，测试一个查询花费的时间。

以下查询用于性能评估。

```
create index sales_cust_id on sales (cust_id) parallel
compute statistics nologging;
```

我们将该查询指定为“索引”查询。

对于选择语句来说，要考虑以下查询：

```
select prod_id, time_id, unit_cost from costs order by
unit_cost;
```

系统实际运行的是以下查询，目的是为了避免在屏幕上输出 14,600,000 行：

```
select count(*) from (select /*+ no_merge */ prod_id, time_id,
unit_cost from costs order by unit_cost);
```

`NO_MERGE` 提示阻止 Oracle 将内嵌视图合并到一个潜在非配置的 SQL 语句中，以执行“order by unit_cost”。

我们把该查询指定为“选择”查询。

以上各个查询在 Oracle 数据库 10g 第 1 版和第 2 版中运行，您可以

比较二者的执行统计。尽管内存排序的底层实施不同，但是这两个版本的执行方法是相同的。

对于索引查询来说，其执行方法为：

Id	Operation	Name	Pstart	Pstop
0	CREATE INDEX STATEMENT			
1	PX COORDINATOR			
2	PX SEND QC (ORDER)	:Q1001		
3	INDEX BUILD NON UNIQUE			
4	SORT CREATE INDEX			
5	PX RECEIVE			
6	PX SEND RANGE	:Q1000		
7	PX BLOCK ITERATOR		1	20
8	TABLE ACCESS FULL	SALES	1	20

下表比较了索引查询性能：

	花费的时间（秒）	用户 CPU(%)
Oracle 数据库 10g 第 1 版	1432	88.54
Oracle 数据库 10g 第 2 版	706	55.98

因为 Oracle 数据库 10g 第 2 版有新的排序算法，所以该版的索引查询比第 1 版要快一倍。而且，索引查询在 CPU 的占用率上第 2 版为 55.98%，第 1 版为 88.54%，显然 Oracle 数据库 10g 第 2 版占用率要少得多，所以第 2 版可以同时进行更多的查询。

对于选择查询来说，该执行计划为：

Id	Operation	Name	Pstart	Pstop
0	SELECT STATEMENT			
1	SORT AGGREGATE			
2	PX COORDINATOR			
3	PX SEND QC (ORDER)	:Q1001		
4	SORT AGGREGATE			
5	VIEW			
6	SORT ORDER BY			
7	PX RECEIVE			
8	PX SEND RANGE	:Q1000	1	
20				
9	PX BLOCK ITERATOR			
10	TABLE ACCESS FULL	COSTS	1	20

下表比较了选择查询性能。

	花费的时间（秒）	用户 CPU(%)
Oracle 数据库 10g 第 1 版	19	83.21
Oracle 数据库 10g 第 2 版	10	73.33

带有排序要求的 **SELECT** 查询和 Oracle 数据库 10g 第 1 版中的同等查询相比，速度上要快 90%，而且占用 CPU 的时间也短得多。

和 Oracle 数据库 10g 第 1 版中的排序算法不同，第 2 版中的新排序算法有较好的内存系统局部性。而且产生更少的高速缓存，TLB 丢失也少得多。当该类可用的内存增加时，使用新排序算法的查询的性能也得以改进。

我们将索引查询作为示例改变 `sort_area_size`，比较 Oracle 数据库 10g 第 1 版和第 2 版的查询性能。下表使用不同的 `sort_area_size` 保存了索引查询的花费时间（单位为秒）。

<code>SORT_AREA_SIZE</code>	1MB	10MB	100MB
Oracle 数据库 10g 第 1 版	1318	1389	1653
Oracle 数据库 10g 第 2 版	911	804	715

在以上所有示例中，索引创建决不完全在内存中进行。然而，因为新内存排序算法的内存系统局部性较好，所以和 Oracle 数据库 10g 第 1 版相比，当我们将 Oracle 数据库 10g 第 2 版的 `sort_area_size` 从 1MB 增加到 100MB 时，其索引查询的速度增长从 44% 提升到 131%。

总之，Oracle 数据库 10g 第 2 版中的新内存排序算法显著地减少了花费的时间，从而提高了查询性能，同时减少 CPU 占用率，可运行更多的并行查询。因为新内存排序算法提供更好的内存系统局部性，所以当可用的内存增加时，使用新内存排序算法的查询的性能也得以改进。

基于散列的聚合

在数据仓库和 OLAP 环境中，数据聚合是很普遍的操作。数据要么聚合得很快，要么经过预先计算，并且以物化视图方式存储。在两种情况下，数据聚合都将具有相同 `group-by` 列价值的数据集集合在一起，然后聚合这些数据。

Oracle 数据库 10g 第 1 版中的数据聚合使用基于类的方法。第 2 版引入了基于散列的模式。该模式是这样运作的：基于 `group-by` 列的值对数据建立散列表，并且通过探查该散列表寻找具有相同 `group-by` 列价值的记录。这种基于散列的模式显著地改进了数据聚合性能。

以下查询说明了基于散列的方法的性能改进情况，该查询计算客户的收入贡献情况，首先列举贡献最大的客户。

```
select c.cust_last_name last_name,
       s.revenue revenue
from customers c,
     (select cust_id,
            sum(amount_sold) revenue
      from sales
      group by cust_id ) s
where s.cust_id = c.cust_id
order by s.revenue desc;
```

该查询是 CPU 密集型的。为确保比较的公平性，关闭 Oracle 数据库 10g 第 2 版中的新内存排序特性。下表比较了 Oracle 数据库 10g 第 1 版和第 2 版的查询性能。

	花费的时间（秒）
Oracle 数据库 10g 第 1 版	395
Oracle 数据库 10g 第 2 版	171

和 Oracle 数据库 10g 第 1 版的查询相比，使用了基于散列的数据聚合模式的第 2 版的查询要快 130%。

因为数据聚合在数据仓库和商务智能任务中很普遍，所以 Oracle 数据库 10g 第 2 版中基于散列的聚合带来的主要性能改进取得效果显著。

分区表

Oracle 数据库 10g 第 2 版分区改变体现在两个方面，一是为提高伸缩性和灵活性，每个表的分区数量已经从 64K 增加到 100 万。二是运用更有效率的修剪技术，确保我们只访问那些满足任何查询谓词需要的分区。我们从两个不同的方面说明 Oracle 数据库 10g 第 2 版中的修剪技术——分离或者谓词分区修剪，使用索引访问方法进行分区修剪。

为了显示分离或者谓词的分区修剪改进，我们使用以下查询：

```
select    p.promo_name promo_name,
         (s.profit - p.promo_cost)      profit
from      promotions p,
         ( select promo_id,
```

```

sum(sales.QUANTITY_SOLD * (costs.UNIT_PRICE -
costs.UNIT_COST)) profit
      from sales, costs
      where
      (
(sales.time_id BETWEEN
TO_DATE('01-JAN-1998','DD-MON-YYYY',
'NLS_DATE_LANGUAGE = American') and
TO_DATE('01-JAN-1999','DD-MON-YYYY',
'NLS_DATE_LANGUAGE = American'))
      OR
(sales.time_id BETWEEN
TO_DATE('01-JAN-2001','DD-MON-YYYY',
'NLS_DATE_LANGUAGE = American') and TO_DATE('01-JAN
2002','DD-MON-YYYY', 'NLS_DATE_LANGUAGE = American'))
)
      AND sales.time_id = costs.time_id
      AND sales.prod_id = costs.prod_id
group by promo_id
)s
where
      s.promo_id = p.promo_id
order by profit desc;

```

该查询联接销售表和成本表。销售表分区的根据是列 **TIME_ID** 上的范围。该查询中的一个条件就是 **TIME_ID** 上有两个谓词，这两个谓词通过 **OR** 联接。

在 **Oracle 数据库 10g 第 1 版**中，该查询使用 **OR** 扩展转换成两个子查询，后接一个串连。各个子查询使用 **OR** 谓词的一个分支联接 **SALES** 表和 **COST** 表。因此，除串连操作外。**COST** 表可以访问两次。这显示在以下查询执行计划中。处理分离 **OR** 谓词的部分计划得到突出显示。

Id	Operation	Name	Pstart	Pstop
0	Select STATEMENT			
1	PX COORDINATOR			
2	PX SEND QC (ORDER)	:Q1006		
3	SORT ORDER BY			
4	PX RECEIVE			
5	PX SEND RANGEW	:Q1005		
6	HASH JOIN			
7	PX RECEIVE			
8	PX SEND HASH	:Q1003		

```

9          VIEW
10         SORT GROUP BY
11         PX RECEIVE
12         PX SEND HASH           :Q1002
13         SORT GROUP BY
14         CONCATENATION
15         HASH JOIN
16         PX BLOCK ITERAOR           1    20
17         TABLE ACCESS PULL  COSTS  1    20
18         PX RECEIVE
19         PX SEND BPDADCAST LOCAL :Q1000
20         BUFFER SORT
21         PX BLOCK ITERATCP         17    20
22         TABLE ACCESS FULL  SALES  17    20
23         HASH JOIN
24         PX BLOCK ITERATOR           1    20
25         TABLE ACCESS FULL  COSTS  1    20
26         PX RECEIVE
27         PX SEND BROADCAST LOCAL :Q1001
28         BUFFER SORT
29         PX BLOCK ITERATOR           5     9
30         TABLES ACCESS FULL  SALES  5     9
31     PX RECEIVE
32     PX SEND HASH           :Q1004
33     PX BLOCK ITERATOR
34     TABLE ACCESS FULL           PROMOTION

```

在 Oracle 数据库 10g 第 2 版中, OR 谓词直接用来修剪 SALES 表中的分区, SALES 表和 COST 表之间进行了单一联接, 这些都显示在以下查询执行计划中。本例关闭了 Oracle 数据库 10g 第 2 版中的散列聚合特性, 关闭以确保比较的公平性。处理分离 OR 谓词的部分计划再次得到突出显示。

Id Operation	姓名	Pstart	Pstop
0 Select STATEMENT			
1 PX COORDINATOR			
2 PX SEND QC (ORDER)	:TQ1005		
3 SORT ORDER BY			
4 PX RECEIVE			
5 PX SEND RANGE	:TQ1004		
6 HASH JOIN			
7 PX RECEIVE			
8 PX SEND HASH	:TQ1002		
9 VIEW			
10 SORT GROUP BY			
11 PX RECEIVE			
12 PX SEND HASH	:TQ1001		
13 SORT GROUP BY			
14 HASH JOIN			
15 PX BLOCK ITERAOR		1	20
16 TABLE ACCESS PULL COSTS		1	20

```

17          PX RECEIVE
18          PX SEND BPDADCAST LOCAL :TQ10000
19          PX BLOCK ITERAOR                      KEY
KEY
20          TABLE ACCESS FULL          SALES      KEY
KEY
21          PX RECEIVE
22          PX SEND HASH                      :TQ10003
23          PX BLOCK ITERATOR
24          TABLE ACCESS PULL          PROMOTIONS

```

因为 Oracle 数据库 10g 第 1 版修剪效率低下，所以访问的数据更多。另外，散列联接方式溢出到磁盘，从而进一步增加了物理读取请求的次数，增加物理写入请求的次数。然而，因为物理 IO 请求数量的减少以及修剪的改善，在 Oracle 数据库 10g 第 2 版中，该查询性能得到极大地提升。该查询速度（花费的时间）提高了 40%。用户 CPU 时间（就是所有处理器上的 CPU 用于运行查询的时间量）提高了 30%。

	花费的时间（秒）	用户 CPU	物理读取 IO 请求	物理写入 IO请求
Oracle 数据库 10g 第 1 版	417	1334	87668	20772
Oracle 数据库 10g 第 2 版	294	1021	17402	72

Oracle 数据库 10g 第 2 版更加有效的使用查询谓词，查询谓词指定非邻近的分区范围来修剪不需要的分区。我们阐明这一点的的方法是：考虑在两个非邻近的时间段中给客户单计算整个 AMOUNT_SOLD 的以下查询。为了进一步阐明，我们已经在分区表销售的列 CUST_ID 上创建了一个全球索引 SALES_GIDX。该查询谓词使用 OR 谓词指定两个分离分区的范围。在 Oracle 数据库 10g 第 2 版中，Oracle 将仅仅访问属于 OR 谓词指定的分区的行。为此，Oracle 维护该谓词指定的有效分区清单。在 Oracle 数据库 10g 第 1 版中，维护的却是单个范围，维护的目的是为了潜在有效地分区，因此带来了效率低下的修剪活动。

```

select cust_id, sum(amount_sold)
from sales
where
(
(time_id BETWEEN
TO_DATE('01-JAN-1997','DD-MON-YYYY',
'NLS_DATE_LANGUAGE = American') and
TO_DATE('01-JAN-1998','DD-MON-YYYY',
'NLS_DATE_LANGUAGE = American'))
OR
(time_id BETWEEN TO_DATE('01-JAN-2001','DD-MON-YYYY',
'NLS_DATE_LANGUAGE = American') and
TO_DATE('01-JAN-2002','DD-MON-YYYY',

```

```

'NLS_DATE_LANGUAGE = American'))
)
AND
    cust_id in (50, 60, 70, 80, 1000, 1100, 1590, 4500, 80000,
250000, 350000, 400100, 430000)

group by cust_id;

```

下表概括了 Oracle 数据库 10g 第 1 版和第 2 版中的查询性能。

	花费的时间（秒）	物理读取 IO 请求
Oracle 数据库 10g 第 1 版	31	75968
Oracle 数据库 10g 第 2 版	10	25436

因为修剪活动更加高效，查询的物理读取请求数量在 Oracle 数据库 10g 第 2 版中大大减少了，所以速度提高了 210%。

正如该部分中的两个示例所表明的那样，在 Oracle 数据库 10g 第 2 版中运用更加有效的修剪技术极大地提高了与分区表有关的查询性能。随着每个表的分区数量限制的增多，这些分区修剪增强特性进一步提高了可伸缩性和灵活性，使具有大量分区的大型数据集能够得以有效地处理。

目标检查点

要最充分地利用硬件资源，Oracle 并行查询可以使用直接路径读取。直接路径读取使查询从属绕过读取请求的数据高速缓存，提交操作系统大小允许的读取请求。然而，在目标可以通过直接路径读取访问之前，该目标的脏缓冲区必须通过目标检查点请求写回磁盘中去。Oracle 数据库 10g 第 1 版通过发布该目标所属的表空间的表空间检查点，写出整个表空间的所有脏缓冲区，来处理目标检查点请求。因为大量的目标对象驻留在相同的表空间中，该实施可能引起大量不要的磁盘写入。对目标对象的检查点请求可能引起驻留在相同表空间中的目标对象和其它对象的脏缓冲区被写回到磁盘。为了消除这种低效率状况，在 Oracle 数据库 10g 第 2 版中，对目标对象的检查点请求将仅仅写出该对象的脏缓冲区，而不导致对其它对象的脏缓冲区进行任何额外的写入。

为了说明新的 Oracle 数据库 10g 第 2 版实施的性能价值，我们假设因 PRODUCTS_AFFINITY_CARD_A 表的需要提交以下更新语句。

```

update products_affinity_card_a set home_theater_package=0
where yrs_residence <2;

```

该更新语句修改 808,760 行。在缓冲区缓存执行该更新语句之后，该缓存立即将来自表 PRODUCTS_AFFINITY_CARD_A 中的约 84,000 更改页包含其中。

就在更新发生之后，系统运行以下并行选择语句，计算产品的最大平均成本。

```
select max(avg_costs)
from (select avg(unit_cost) avg_costs
      from costs
      group by prod_id);
```

Oracle 数据库 10g 第 1 版在执行选择语句之前需要对表 PRODUCTS_AFFINITY_CARD_A 的脏缓冲区进行抽查检，而第 2 版能够直接执行选择语句，无需将更改页写入磁盘。

要确保比较的公平性，您要禁止 Oracle 数据库 10g 第 2 版中的内存排序特性和散列聚合特性，使 Oracle 数据库 10g 第 1 版和第 2 版根据以下执行计划执行选择查询：

Id	Operation	姓名	Pstart	
	Pstop			
0	Select STATEMENT			
1	SORT AGGREGATE			
2	PX COORDINATOR			
3	PX SEND QC (RANDOM)	:TQ10001		
4	SORT AGGREGATE			
5	VIEW			
6	SORT GROUP BY			
7	PX RECEIVE			
8	PX SEND HASH	:TQ10000		
9	SORT GROUP BY			
10	PX BLOCK ITERATOR		1	20
11	TABLE ACCESS FULL	COSTS	1	20

在 Oracle 数据库 10g 第 1 版中，更新后执行并行选择查询需要 27 秒；而在 Oracle 数据库 10g 第 2 版中完成此过程则只需要 13 秒。这是因为更新语句更新表，该表存在于和该选择语句访问的对象一样的表空间中。在 Oracle 数据库 10g 第 1 版中，执行该选择查询之前额外时间用于写出 PRODUCTS_AFFINITY_CARD_A 表中的脏缓冲区。

因为在相同表空间中对其它对象的脏缓冲区进行抽查检查不会使对对象的直接路径访问慢下来，所以 Oracle 数据库 10g 第 2 版中的新实施显著地提高并行查询性能。

SQL 内建数据挖掘应用功能

数据挖掘流程包括第一步：使用[培训训练](#)数据集创建模型。然后将该模型应用于新数据集以生产[预示预测](#)。因为现有模型可以应用到许多不同的新

型数据集中，所以客户经常借助应用操作来挖掘自己的数据库。因此，应用操作的性能和可用性是很重要的。

在数据挖掘术语中，APPLY 也称作 SCORING，因此 APPLY 操作的输入数据集（表）可以称作 SCORING 数据（表）。在 Oracle 数据库 10g 第 1 版中，使用 PL/SQL API 的 APPLY 是一个“批量” SCORING 过程，该过程将表当作输入，并提供一个输出表。使用 JAVA API 的 APPLY 提供记录应用操作。使用 Oracle 数据库 10g 第 1 版的缺点包括：

- 模式模型表和得分表之间的复杂联接限制了性能。多个具体的中间结果对无缝的 SQL 查询处理进行抑制。许多映射和转化可能非常耗时，支配了执行应用的时间。
- 因为该模式模型不在相同应用操作的多个调用中共享，所以每次 APPLY 调用时该模式模型需要被加载到内存中。
- 而且，在使用 JAVA 和 PL/SQL API 之前，初学者需要对它们很熟悉。

Oracle 数据库 10g 第 2 版通过提供 SQL 内建模式模型得分功能，使 APPLY 功能性符合标准。使用 Oracle 数据库 10g 第 2 版的好处包括：

- 因为 SQL 内建功能提供的执行在改变，所以 APPLY 性能得以改进。在某些情况下，改进是很显著的。
- 现有的查询执行功能，例如公用指针缓存模型元数据，允许跨越不同 APPLY 调用共享模型。
- 为 APPLY 提供 SQL 内建功能极大地提高了可用性。初学者能够快速学习如何将现有模型应用到新的数据集。因为通过这些新应用功能可以很容易地增强现有 SQL 语句，所以在现有应用程序下部署模型是很简单的。SQL 内建功能也实现对与数据挖掘预示有关的结果进行流水线操作，还能将少数结果快速返回给终端用户。

Oracle 数据库 10g 第 2 版中的 SQL 内建得分功能可以应用不同数据挖掘算法建立的模型，适应的贝叶斯网络 (ABN) 算法用于为该白皮书构建模型。ABN 是一种分类算法。通过来自培训训练数据集的 ABN 算法建立的分类模型可用于给应用数据集中的各条记录进行分类，分类的方法是显示记录所属的类别。例如，我们可能想预知某个客户是否有基于客户购买历史的认同卡，比如该客户已经购买了多少散装 DVD，该客户拥有多少平板显示器，以及该客户已经购买了多少记帐应用程序，等等。这种情况下的每位客户分为以下两类 — 有认同卡的客户和没有认同卡的客户。我们将认同卡称作目标，将客户购买历史称作预报器。这种分类任务是以培训训练数据集开始的，我们已经从培训训练数据集得知某客户是否有认同卡。我们将培训训练数据中的预报器和目标之将的关系存储在一个模型中，该模型可以应用到新数据集以预测这些新客户是否有认同卡。

对于 Oracle 数据库 10g 第 1 版和第 2 版，我们都使用相同的培训训练数据集以构建 ABN 模型，然后将该模型应用到相同的得分数据集。我们比较了这两种版本的 APPLY 性能。

我们的数据集 PRODUCTS_AFFINITY_CARD_B 列数和行数分别为 250 和 50,000。该表包括一个 ID 列 (CUST_ID)、一个目标

(AFFINITY_CARD), 以及 248 个预报器, 这些预报器有 YRS_RESIDENCE、BULK_PACK_DVDS、FLAT_PANEL_MONITOR, 等等。在 Oracle 数据库 10g 第 1 版和第 2 版中构建 ABN 模型 'ABN_Clas' 需要约 40 分钟。因为构建了 'ABN_Clas' 模型, 该数据集可对应用数据集进行预测。应用表 PRODUCTS_AFFINITY_CARD_A 的行数为 200 万。为了将模型 'ABN_Clas' 应用到应用表中, 以及将结果存储在结果表 APPLY_RESULT 中, 您可以在 Oracle 数据库 10g 第 1 版和第 2 版中使用以下 PL/SQL 程序:

```
--The APPLY table needs to be binned in the same way as the
training table.
--abn_num is the bin table that we generated when binningthe
training table.
BEGIN
DBMS_DATA_MINING_TRANSFORM.XFORM_BIN_NUM
  ( bin_table_name => 'abn_num', data_table_name =>
    'PRODUCTS_AFFINITY_CARD_A', xform_view_name =>
    'abn_apply_prepared');
END;

/

--materialize the prepared apply table for performance.

create table abn_apply_prepared_table as select *
from abn_apply_prepared;alter table abn_apply_prepared_table
parallel;

--APPLY THE MODELBEGIN
DBMS_DATA_MINING.APPLY( model_name => 'ABN_Clas',
  data_table_name => 'abn_apply_prepared_table',
  case_id_column_name => 'CUST_ID', result_table_name =>
  'APPLY_RESULT');
END;

/
```

尽管以上代码可以在 Oracle 数据库 10g 第 1 版和第 2 版中使用, 但是对 DBMS_DATA_MINING.APPLY 的根本实施是不同的。在 Oracle 数据库 10g 第 2 版中, 当我们调用 DBMS_DATA_MINING.APPLY 时, SQL 内建功能 PREDICTION_SET 就被调用了。另外, 我们可以在 SQL 语句中调用 Oracle 数据库 10g 第 2 版提供的这些 SQL 内建功能, 使得应用功能应用变得很容易。例如, 对于存储在 abn_apply_prepared_table 中的各个客户来说, 以下 SQL 语句使用预构建的分类模型 ABN_Clas 预测某个客户是否有认同卡。

```
select cust_id, prediction(ABN_Clas using *) as my_pred
from abn_apply_prepared_table;
```

SQL 内建功能 PREDICTION 比 PREDICTION_SET. SQL 简单。假如有预报器集, 该语句返回分类模型的最佳预测, 而 PREDICTION_SET 函数返回对象的可变数组, 这些对象包含所有类别和各种类别的可能性。因为有 200 万客户存储在 abn_apply_prepared_table 中, 以上 SQL 查询返回 200 万行。为了不在测量输出的 200 万行上花费时间, 要按照以下方法修改上述查询来帮助性能测量:

```
select count(distinct prediction(ABN_Clas using *)) from
abn_apply_prepared_table;
```

Oracle 数据库 10g 第 2 版客户应用模型时, 可能从三种方法中选择一种方法。正如以上示例所表明, 第一种方法是使用 PL/SQL 程序。这种方法意味着使用 Oracle 数据库 10g 第 2 版 (PL/SQL)。第二种方法就是使用 SQL 查询, 如同上述所显示的。这种方法意味着使用 Oracle 数据库 10g 第 2 版 (SQL)。最后, 您也可以使用在该白皮书中没有进行标准检查的 JAVA API, 就像 PL/SQL API, SQL 内建应用顶部的 Oracle 数据库 10g 第 2 版层中的 JAVA API 运行。

下表使用所描述的 PL/SQL 和 SQL 方法, 概述 Oracle 数据库 10g 第 1 版和第 2 版中应用的性能结果。

	花费的时间 (秒)	临时空间
Oracle 数据库 10g 第 1 版	1H59M36.56S	10G
Oracle 数据库 10g 第 2 版(PL/SQL)	59.6S	0
Oracle 数据库 10g 第 2 版(SQL)	10.38S	0

SQL 内建得分功能在花费的时间和临时空间占用方面都得到显著地改进。临时空间占用的改进表明了节约了很多内存。直接在 SQL 语句中使用 SQL 内建得分功能会产生更快的应用操作, 会产生一个初学者更容易掌握的用户界面。Oracle 数据库 10g 第 2 版中改进的性能和增强的应用可用性使大型数据集的处理变得既容易又高效。

支持向量机 (SVM) 算法增强特性

Oracle 数据库 10g 第 1 版中引入的支持向量机 (SVM) 是一种艺术级的数据挖掘算法, 该算法支持分类和回归任务。分类任务根据预报器集对目标类进行预测。例如, 我们可能希望根据预报器 (例如客户年龄、职业, 等等) 预测某个人的收入水平是否很高。在这种情况下, 某个人就属于两种目标类中的一种: 高收入群体, 或低收入群体。回归任务预测用数字表示的/持续的目标, 而不预测目标类 (离散的或绝对的值)。例如, 我们可能希望根据教育、职业等预测实际收入。

在 SVM 术语中, 人们将和分析实体 (例如客户) 相关的逻辑行视作模式。

每个模式是预报器值的向量和目标值。SVM 寻找预测性的模式（支持向量）。例如，为了预测收入的高低，SVM 分类器通过决定边界将这两个目标类分开。在多维空间中，该决定边界代表的是超平面。在非多维数据情形下，非线形核心功能（例如 高斯核心）可以用来将输入空间转变为多维特型空间。多维特型空间然后精确地给 SVM 模型提供灵活性，可以任意安装复杂非线形决定界面。对多维数据来说，从非线形核心转化到多维特性空间通常是不必要的，线形核心是可以被使用的。

正如其它任何分类或衰退算法一样，使用 SVM 的数据挖掘过程包括 *创建* 模型以发现预报器和目标之间的关系。在模型创建之后，*测试* 模型评估准确性。测试过程中，将该模型应用到独立的数据集，目标值可以在该数据集获得。具有令人满意准确性的模型可以用于对数据集进行预测，目标值在该数据集是不可知的。这就是所谓的 *应用*。

在速度和准确性方面，Oracle 数据库 10g 第 1 版中的 SVM 执行可以兼容其它一般可用的 SVM 工具（例如 SVMLight、LIBSVM、SVMTorch，以及 SVMFu）。但是在标准 SVM 方法中有某些限制。SVM 模型的 **培训训练** 时间和数据记载的数量之间是成二次方甚至三次方比例的。另外，具有非线性核心的模型（某种情况下具有线性核心的模型），可能变得很大，导致粗糙的模型构造和应用性能。

Oracle 数据库 10g 第 2 版将主动学习和分层取样结合起来，解决标准 SVM 方法中固有的可测量性问题。该方法有效地选择包含信息最多的示例减少支持矢量的数量。因此，用于构建模型和应用现有模型的时间大大地减少了。主动学习方法提供的解决方案是相似的，但是在大多数情况下，这些相似解决方案的精确性和标准 SVM 模型的准确性是可以相比的（不是很明显得差）。

为了说明新方法所做的改进，我们首先在 Oracle 数据库 10g 第 1 版和第 2 版中使用 **培训训练** 数据集构建 SVM 模型，比较构建时间、临时空间使用情况、模型大小，以及支持矢量的数量。然后，使用测试数据集对这两个模型进行测试，比较模型的精确性。最后，我们将这两个模型应用到应用数据集，比较花费的时间和临时空间使用情况。

我们的分类任务是根据以下预报器确定某人收入的高低：AGE、WORKCLASS、FNLWGT、EDUCATION、EDUCATION_NUM、MARITAL_STATUS、OCCUPATION、RELATIONSHIP、RACE、SEX、CAPITAL_GAIN、CAPITAL_LOSS、HOURS_PER_WEEK 和 NATIVE_COUNTRY。

培训训练 数据集 SUPPLEMENTARY_DEMOGRAPHICS_B 的行数为 32,561。我们在 Oracle 数据库 10g 第 1 版和第 2 版中构建一个具有高斯内核的 SVM 模型。下表显示了这两个版本中构建的模型的性能。

	花费的时间（秒）	临时空间	模型大小 (MB)	#支持矢量
Oracle 数据库 10g 第 1 版 11.756	10M47.40S	3	6	

在 Oracle 数据库 10g 第 2 版中构建 SVM 模型所使用的时间要比在 Oracle 数据库 10g 第 1 版中构少。另外，第 2 版需要的临时表空间也少得多，这表明新方法的临时需求减少了。第 2 版中花费的时间减少了，临时空间占用减少了，都是因为模型大小缩小的缘故。

为了证明模型大小的确缩小了，我们使用 16281-行测试数据 SUPPLEMENTARY_DEMOGRAPHICS_T 对两种模型进行测试，比较模型的精确性。

	计算混合矩阵得到的精确性	ROC曲线下面积
Oracle 数据库 10g 第 1 版	.8469	.8935
Oracle 数据库 10g 第 2 版	.8227	.8673

有不同的测量方法可以测量模型精确性。计算 Oracle 数据库 10g 第 1 版混合矩阵得到的精确性为 0.8469。这就意味着，Oracle 数据库 10g 第 1 版中构建的 SVM 模型正确预测目标类的次数为 $0.8469 * 16281$ 。这里的 16281 指测试数据集中的总行数。不正确预测目标属性类的次数为 $(1-0.8469) * 16281$ 。

评估分类模型的另一种实用度量衡量标准是接受者工作特征 (ROC)。ROC 曲线下面积 (AUC) 对二进制分类模型的不同功能进行测量。AUC 越大，正面（高收入）比负面（低收入）高收入分配的可能性越高。对具有不均衡的目标分配的数据集（也就是说，目标类支配另一个）来说，AUC 测量特别实用。

我们可以从上述表格中看到，因为模型相似，所以模型不够精确得情况很少。

最后，为比较 APPLY 性能，我们使用有 97,684 条记录的数据集 SUPPLEMENTARY_DEMOGRAPHICS_A。下表概述了 Oracle 数据库 10g 第 1 版和第 2 版中的 APPLY 性能。

	花费的时间	临时占用空间 (MB)
Oracle 数据库 10g 第 1 版	6M 6.29S	7
Oracle 数据库 10g 第 2 版	36.51 S	0

因为模型规模缩小了，Oracle 数据库 10g 第 2 版的 APPLY 性能速度是第 1 版的 10 倍。正如临时空间占用情况显示的那样，它不仅运行速度更快了，而且需要更少的内存。该白皮书的前面解释过，Oracle 数据库 10g 第 2 版中新引入的应用 SQL 内建功能对该性能改进也有一定作用。

第 2 版中的 SVM 方法显著地提高了 SVM 算法构建和应用性能，以及

模型的精确性。该方法还允许在大型数据集上使用 SVM 模型，在第 1 版中是不能处理这些大型数据集的。

结论

结果不言而喻。因为 Oracle 数据库 10g 第 2 版中的增强性能和新特性的设计能满足甚至超过数据仓库应用程序的需求，所以该版本能提供更快、更高的性能。您只需要升级到 Oracle 数据库 10g 第 2 版，就可以获得明显的性能改进，无需额外的部署费用。Oracle 的最新版本使数据仓库团队能够有效容易地为更多的商务智能补充支持日益增加的数据量。



Oracle 数据库 10g 第 2 版中的 DSS 性能
2005 年 7 月
作者: Linan Jiang
副编: Xumin Nie

Oracle Corporation
全球总部
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

全球咨询热线:
电话: +1.650.506.7000
传真: +1.650.506.7200
oracle.com

版权所有 © 2005, Oracle。保留所有权利。
本文档只用于提供信息, 其中的内容如有更改, 恕不通知。
本文档不保证没有错误, 也不受其他任何口头表达或法律
暗示的担保或条件的约束, 包括对特定用途的适销性或适
用性的暗示担保和条件。我们特别声明: 拒绝承担与本
文档有关的任何责任, 本文档不直接或间接形成任何合
约职责。未经预先书面许可, 不允许以任何形式或任何
方式(电子方式或机械方式)、出于任何目的, 复制或传播
本文档。
Oracle、JD Edwards、和 PeopleSoft 是 Oracle 公司
或其关联公司的注册商标。其他名称可能是其各自所有
者的商标。