神通数据库

Hint使用手册

版本 7.0

天津神舟通用数据技术有限公司

2010 年 1 月

版权声明

神通数据库是天津神舟通用数据技术有限公司开发的数据库管理系统软件产品。神通数据库的版权归天津神舟通用数据技术有限公司，任何侵犯版权的行为将追究法律责任。

《神通数据库 hint 使用手册》的版权归天津神舟通用数据技术有限公司所有。

天津神舟通用数据技术有限公司不对因为使用该软件、用户手册或由于该软件、用户手册中的缺陷所造成的任何损失负责。

前言

神通数据库 hint 为用户提供了更好的查询性能。本手册主要介绍了当前 hint 具有的功能以及相关语法的使用。

本手册适用于所有神通数据库的用户。

阅读指南

〖阅读对象〗

本手册是为使用神通数据库管理系统的用户编写的。

〖内容简介〗

本手册介绍了 hint 功能以及 hint 语法使用。

〖手册构成〗

本手册由 2 部分组成：

第 1 章，“功能概述”，介绍了 hint 目前具有的功能。

第 2 章，“hint 语法语义”，具体介绍 hint 使用的语法。

〖相关文档〗

使用本手册时可以参考神通数据库的手册集，手册集包含以下文档：

《神通数据库安装手册》

《神通数据库备份恢复工具使用手册》

《神通数据库 DBA 管理工具使用手册》

《神通数据库系统管理员手册》

《神通数据库嵌入式 SQL 语言手册》

《神通数据库交互式 SQL 查询工具使用手册》

《神通数据库 JDBC 开发指南》

《神通数据库过程语言手册》

《神通数据库 OLEDB/ADO 用户手册》

《神通数据库迁移工具使用手册》

《神通数据库 ODBC 程序员开发指南》

《神通数据库审计管理》

《神通数据库审计工具使用手册》

《神通数据库性能监测工具使用手册》

《神通数据库作业调度工具使用手册》

〖手册约定〗

本手册遵循以下约定：

所有标题均使用黑体字。

如果标题后跟有“【条件】”字样，说明该标题下正文所要求的内容只是在一定条件下必须的。

【注意】的意思是请读者注意那些需要注意的事项。

【警告】的意思是请读者千万注意某些事项，否则将造成严重错误。

【提示】的意思是提供给读者一些实用的操作技巧。

对于手册中出现的正文和程序代码，遵循如下约定：

表 1　手册正文约定

约定	含义	范例
粗体	表示强调	确保控制文件和数据文件不要驻留在同一个磁盘上。
大写等宽字母	表示由系统提供的元素，如参数、权限、数据类型、关键词、命令、函数名，以及由系统提供的列名、数据库对象和结构等。	利用 BACKUP 命令备份数据库。 在 USER_TABLES 数据字典视图中查询 TABLE_NAME 列
小写等宽字母	表示执行程序、文件名、目录名以及需要由使用者提供的元素，包括计算机名、数据库名、数据库对象和结构、列名、程序单元以及参数等。 【注意】：某些元素要求使用大写或者大小写混合形式。此时，应当根据实际的要求输入。	department_id，department_name 以及 location_id 列在 departments 表中。
小写等宽斜体	表示占位符或者变量	

表 2 手册中出现的程序代码的书写约定

约定	含义	范例
[]	表示包含一个或者多个可选项。不需要输入中括号本身。	DECIMAL (digits [, precision])
{}	表示包含两个以上（含两个）的选项，其中有一个是必须的。不需要输入花括号本身。	{ENABLE | DISABLE}
|	分割中括号或者花括号中的两个或者两个以上的选项。不需要输入“|”本身。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
…	表示省略 表示重复	CREATE TABLE … AS subquery; SELECT col1, col2, … , coln FROM employees;
. . .	表示省略了若干行	
斜体	表示占位符或者需要提供特定值的变量	CONNECT SYSTEM/system_password DB_NAME = database_name
大写	表示系统提供的元素，主要是为了与使用者定义的元素相互区分。除了出现在方括号中的元素外，应当按照顺序逐字输入。当然，有些元素在系统中是大小写不敏感的，因此使用者也可以根据系统说明以小写的方式输入。	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
小写	表示由使用者提供的元素。例如：表、列和文件的名字。 【注意】：根据某些具体的要求，有些由使用者提供的元素可能要求使用大写或者大小写混合的形式。此时，应当根据实际的要求输入。	SELECT last_name, employee_id FROM employees; CREATE USER tom IDENTIFIED BY a8M9j7

第1章 hint 概述

尽管 oscar 优化器在生成正确的执行计划方面有着相当可靠的准确性，但面对数据库系统中成千上万的查询语句，优化器也未必就能 100%给出最优的执行计划。因此，对于某一条特定的查询语句，如果用户对查询所涉及的表数据非常了解，那么通过使用 hint，用户就可以强制优化器产生自己指定的执行计划，从而获得更好的查询性能。目前实现的 hint 有以下功能：

1. 当用户输入正确的 hint 时，优化器会完全按照用户的 hint 指示生成相应的执行计划。
2. 当用户输入的 hint 存在语法或语义的错误时，优化器会自动忽略这些错误的 hint，同时不会影响其它正确 hint 的执行。
3. 当用户输入的 hint 所指定的计划是优化器不可能生成的计划时，优化器会自动忽略这些 hint，同时优化器会自己给出一个自认为最优的执行计划。

第2章 hint 类型

Hint 可以分为以下类型：

1. 单表 hint
2. 多表 hint
3. 命令型 hint

2.1 单表 hint

2.1.1 index(a [index_1, … , index_n])

确定在表 a 上利用给出的索引进行扫描。如果只给出一个索引名，表示指定使用该索引进行表扫描；如果给出多个索引名，则表示在多个索引中选择一个进行表扫描；如果不给出索引名，则优化器考虑 a 上的所有可用索引，并从中选择代价最小者。

举例：

```
select /*+ index(t1 idx_t1_tc1, idx_t1_tc2)*/ * from t1 where tc1 > 1 and tc2 > 2;
```

2.1.2 no_index(a [index_1, … , index_n])

确定不利用表 a 上的索引对表 a 进行表扫描。如果只给出一个索引名，表示不使用该索引进行扫描；如果给出多个索引，则表示所列出的多个索引都不考虑；如果不给出索引，则表示表 a 上的所有索引扫描都不考虑。

举例：

```
select /*+ no_index(t1 idx_t1_tc1, idx_t1_tc2)*/ * from t1 where tc1 > 1 and tc2 > 2;
```

2.1.3 full (a)

确定在表 a 上使用 seqscan 计划。

举例：

```
select /*+ full(t1)*/ * from t1 where tc1 > 1;
```

2.1.4 bm_index(a index_1, index_2)

确定在表 a 上使用 bitmapindexjoin，该 bitmapindexjoin 由索引 index_1 和 index_2 组合而成。

举例：

```
select /*+ bm_index(t1 idx_t1_tc1 idx_t1_tc2)*/ * from t1 where tc1 > 1 or tc2 > 2;
```

2.1.5 no_bm_index(a [index_1, index_2])

确定在表 a 上不使用 bitmapindexjoin，该 bitmapindexjoin 由索引 index_1 和 index_2 组合而成。如果不给出索引参数，则优化器不考虑该表上所有的 bitmapindexjoin 计划。

举例：

```
select /*+ no_bm_index(t1 idx_t1_tc1 idx_t1_tc2)*/ * from t1 where tc1 > 1 or tc2 > 2;
```

2.2 多表 hint

2.2.1 use_nl (A , B)

1. 如果 A、B 都是基表，则表示以 A 作为外表，B 作为内表，两者做 nestloop 连接。
2. 如果 A 是表集合，B 是基表，定义 A=（table1, table2, ... ,table n）。此时的语义为：以集合 A 内各表做连接得到的联接表作为外表，B 作为内表，两者做 nestloop 连接。（当 B 是表集合、A 是基表时情况类似）
3. 如果 A、B 均是表集合，则表示以集合 A 内各表做连接得到的联接表作为外表，以集合 B 内各表做连接得到的联接表作为内表，两者做 nestloop 连接。

举例：

```
select /*+ use_nl(c,(a,b))*/* from a,b,c where a.a1=b.b1 and a.a2=c.c2;
```

2.2.2 use_hj (A , B)

1. 如果 A、B 都是基表，则表示以 A 作为外表，B 作为内表，两者做 hash 连接。
2. 如果 A 是表集合，B 是基表，定义 A=（table1, table2, ... ,table n）。此时的语义为：以集合 A 内各表做连接得到的联接表作为外表，B 作为内表，两者做 hash 连接。（当 B 是表集合、A 是基表时情况类似）
3. 如果 A、B 均是表集合，则表示以集合 A 内各表做连接得到的联接表作为外表，以集合 B 内各表做连接得到的联接表作为内表，两者做 hash 连接。

举例：

```
select /*+use_nl((b,d),(a,c))*/* from a,b,c,d where a.a1=b.b1 and a.a2=c.c2 and b.b2=d.d2;
```

2.2.3 use_mj (A , B)

1. 如果 A、B 都是基表，则表示以 A 作为外表，B 作为内表，两者做 merge 连接。
2. 如果 A 是表集合，B 是基表，定义 A=（table1, table2, ... ,table n）。此时的语义为：以集合 A 内各表做连接得到的联接表作为外表，B 作为内表，两者做 merge 连接。（当 B 是表集合、A 是基表时情况类似）
3. 如果 A、B 均是表集合，则表示以集合 A 内各表做连接得到的联接表作为外表，以集合 B 内各表做连接得到的联接表作为内表，两者做 merge 连接。

举例：

对于基表 a、b、c、d、e、f，用下列一套 hint 组合可准确指定这六张表的连接方式和顺序。

```
/*+use_nl((a,b,c,d),(e,f)) use_hj(a,(b,c,d)) use_mj(b,(c,d)) use_nl(c,d) use_nl(e,f)*/
```

上述 hint 语句表示的各表连接顺序和方式如下图 2-1 所示：

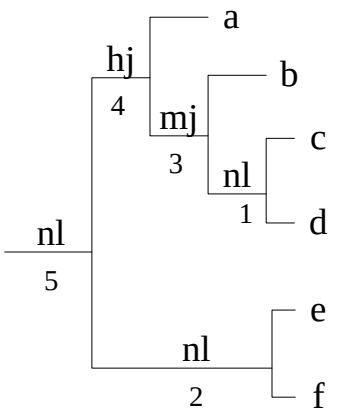

图 2-1 hint 语义图

2.2.4 use_join (A, B)

表示在做表连接时，A 做外表，B 做内表，但不指定具体连接方法。该 hint 的参数情况与 use_nl 类似。

如果只是想要指定多表连接的先后顺序，而不指定两表间的连接方法，使用该 hint。

举例：

```
/*+ use_join (d,(a, b,c))    use_join ((a, b),c)    use_join (a, b)*/
```

上述 hint 表示 a 做外表 b 做内表进行连接，然后的到的联结表（a,b）做外表，c 做内表进行连接，最后得到的联结表（a,b,c）做内表 d 做外表进行连接；而所有连接的连接方法（nl、mj 或者 hj）均由优化器自己选择。

2.2.5 use_semi_nl (A, B)

用在父查询中，表明 A 做外表，B 做内表，两者做 nestloop 连接，连接方式为 semi-join。

举例：

```
select /*+ use_semi_nl(a,b)*/* from a where a.a1 in (select b1 from b where b.b1<10);
```

2.2.6 use_semi_hj (A, B)

用在父查询中，表明 A 做外表，B 做内表，两者做 hash join 连接，连接方式为 semi-join。

举例：

```
select /*+ use_semi_hj(a,(b,c))*/* from a where a.a1 in (select b1 from b,c where b.b1=c.c1 and c.c2<10);
```

2.2.7 use_semi_mj (A, B)

用在父查询中，表明 A 做外表，B 做内表，两者做 merge join 连接，连接方式为 semi-join。

举例：

```
select /*+ use_semi_mj((a,b),(c,d))*/* from a,b where a.a1=b.b1 and b.b2 in (select c2 from c,d where c.c2=d.d2 and d.d1<10);
```

2.2.8 use_nl_with_material (A, B)

表明 A 做外表，B 做内表，两者做 nestloop 连接，并且采用 material 计划作为内表扫描计划，内外表做连接的 jointype 类型由优化器自己决定。

举例：

```
select /*+use_nl_with_material(a,(b,c))*/ * from a,b,c where a.a1=b.b1 and b.b1=c.c1;
```

2.2.9 use_semi_nl_with_material(A, B)

表明 A 做外表，B 做内表，两者做 nestloop 连接，并且采用 material 计划作为内表扫描计划，内外表做连接的 jointype 类型为 semi 类型。

举例：

```
select /*+use_semi_nl_with_material((a,b),c)*/ * from a,b,c where a.a1=b.b1 and b.b1 in (select c1 from c where c.c2>6);
```

2.2.10 use_norm_nl_with_material(A, B)

表明 A 做外表，B 做内表，两者做 nestloop 连接，并且采用 material 计划作为内表扫描计划，内外表做连接的 jointype 类型不是 semi 类型。

举例：

```
select /*+use_norm_nl_with_material(a,(b,c))*/ * from a,b,c where a.a1=b.b1 and b.b1=c.c1;
```

2.2.11 use_nl_with_index(A, b index_name)

【注意】

参数 b 只能是单表，不能是表集合，表名和索引名之间用空格分隔，该 hint 必须指定且只能指定一个索引名。注意该 hint 与单表的 index(a, [index_1, … , index_n])的不同。

表明 A 做外表，b 做内表，两者做 nestloop 连接，并且内表条件下降，扫描采用指定索引，内外表做连接的 jointype 类型由优化器自己决定。

举例：

```
select /*+use_nl_with_index((a,b),c c2_index)*/ * from a,b,c where a.a1=b.b1 and b.b2=c.c2 and c.c2<10;
```

对于该 hint 的实现，将 largs 用来存储 A，用 rargs 来存储 b 和索引。在 process_single_table 中传入 rargs 就可以保证原来 hint 的单表处理函数 process_singletable_hints(Query *root, RelOptInfo *rel)的函数接口不变了。

2.2.12 use_semi_nl_with_index(A, b index_name)

表明 A 做外表，b 做内表，两者做 nestloop 连接，并且内表条件下降，扫描采用指定索引，内外表做连接的 jointype 类型为 semi 类型。

举例：

```
select /*+use_semi_nl_with_index((a,b),c c1_index)*/ * from a,b,c where a.a1=b.b1 and b.b1 in (select c1 from c where c.c2>6);
```

2.2.13 use_norm_nl_with_index(A, b index_name)

表明 A 做外表，b 做内表，两者做 nestloop 连接，并且内表条件下降，扫描采用指定索引，内外表做连接的 jointype 类型不是 semi 类型。

举例：

```
select /*+use_norm_nl_with_index((a,b),c c2_index)*/ * from a,b,c where a.a1=b.b1 and b.b2=c.c2 and c.c2<10;
```

2.3 命令型 hint

2.3.1 no_pullup_subquery

用在子查询中，表示不提升子查询。

举例：

```
select * from a,b,c where a.a1=b.b1 and b.b1 in (select /*+ no_pullup_subquery */  c1 from c where c.c2>6);
```

2.3.2 no_use_mergeview

用在视图中，确定该视图不被合并到父查询中。

举例：

```
select * from t1, (select /*+ no_use_mergeview*/ tc1 from t2) T where t1.tc1 =T.tc1;
```

2.3.3 use_hashdistinct

确定查询使用 hashagg 算子实现 distinct 操作。

举例：

```
select /*+ use_hashdistinct*/ distinct tc1 from t1;
```

2.3.4 use_sortdistinct

确定查询使用 sortagg 算子实现 distinct 操作。

举例：

```
select /*+ use_sortdistinct*/ distinct tc1 from t1;
```

2.3.5 use_hashsp

用在子查询中，确定子查询使用 hashed subplan。

举例：

```
select * from t1 where t1.tc1 in (select /*+ use_hashsp*/ tc1 from t2);
```

2.3.6 no_use_hashsp

用在子查询中，确定子查询不使用 hashed subplan。

举例：

```
select * from t1 where t1.tc1 in (select /*+ no_use_hashsp*/ tc1 from t2);
```

2.3.7 no_use_residual

用在子查询中，确定父查询与该子查询不通过 Semi Residual Join 连接。

举例：

```
select /*+ no_use_residual*/* from t1 where t1.tc1 in (select    tc1 from t2) or t1.tc1 in (select tc1 from t2)
```

2.3.8 ordered_predicates

用在查询中，确定谓词条件按照用户书写的顺序排序。

2.3.9 use_sqltrans

确定查询使用 SQL_TRANS 机制。

2.3.10 no_use_sqltrans

确定查询不使用 SQL_TRANS 机制。

2.3.11 result_cache

对查询结果进行缓存。

第3章 hint 使用

在输入 hint 时，表和表之间用逗号隔开，索引和索引之间也用逗号隔开，但表和索引之间则用空格隔开，多个 hint 之间也用空格隔开。