



链滴

# HINT 优化指示器

作者: [xixiaoming](#)

原文链接: <https://ld246.com/article/1506477020759>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

# 1. 和优化器相关的hint

## 1、/+ ALL\_ROWS /

表明对语句块选择基于开销的优化方法,并获得 **最佳吞吐量**,使资源消耗最小化.

```
SELECT /+ ALL_ROWS / EMP_NO,EMP_NAM,DAT_IN FROM BSEMPMS WHERE EMP_NO='SCOTT';
```

## 2、/\*+ FIRST\_ROWS(n) /

表明对语句块选择基于开销的优化方法,并获得 **最佳响应时间**,使资源消耗最小化.

```
SELECT /+FIRST_ROWS(20) */ EMP_NO,EMP_NAM,DAT_IN FROM BSEMPMS WHERE EMP_NO=SCOTT;
```

## 3、/+ RULE/

表明对语句块选择 **基于规则**的优化方法.

```
SELECT /*+ RULE */ EMP_NO,EMP_NAM,DAT_IN FROM BSEMPMS WHERE EMP_NO='SCOTT';
```

# 2. 和访问路径相关的hint

## 1、/+ FULL(TABLE)/

表明对表选择 **全局扫描**的方法.

```
SELECT /*+FULL(A)*/ EMP_NO,EMP_NAM FROM BSEMPMS A WHERE EMP_NO='SCOTT';
```

## 2、/\*+ INDEX(TABLE INDEX\_NAME) \*/

表明对表选择 **索引扫描**的方法.

```
SELECT /*+INDEX(BSEMPMS SEX_INDEX) */ * FROM BSEMPMS WHERE SEX='M';
```

## 5、/+ INDEX\_ASC(TABLE INDEX\_NAME)/

表明对表选择 **索引升序**的扫描方法.

```
SELECT /*+INDEX_ASC(BSEMPMS PK_BSEMPMS) */ * FROM BSEMPMS WHERE DPT_NO='SCOTT';
```

## 6、/+ INDEX\_COMBINE/

为指定表选择 **位图访问路径**,如果INDEX\_COMBINE中没有提供作为参数的索引,将选择出位图索引的尔组合方式.

```
SELECT /*+INDEX_COMBINE(BSEMPMS SAL_BMI HIREDATE_BMI) */ * FROM BSEMPMS WHERE SAL<5000000 AND HIREDATE
```

## 7、/\*+ INDEX\_JOIN(TABLE INDEX\_NAME1 INDEX\_NAME2) \*/

当谓词中引用的列都有索引的时候,可以通过指定采用 **索引关联**的方式,来访问数据

```
select /*+ index_join(t t_ind t_bm) */ id from t where id=100 and object_name='EMPLOYEES'
```

## 8、/+ INDEX\_DESC(TABLE INDEX\_NAME)/

表明对表选择 **索引降序**的扫描方法.

```
SELECT /*+INDEX_DESC(BSEMPMS PK_BSEMPMS) */ * FROM BSEMPMS WHERE DPT_NO='SOTT';
```

9、 /\*+ INDEX\_FFS(TABLE INDEX\_NAME) \*/

对指定的表执行 **快速全索引扫描**,而不是全表扫描的办法.

```
SELECT /* + INDEX_FFS(BSEMPMS IN_EMPNAM)*/ * FROM BSEMPMS WHERE DPT_NO='TEC35';
```

10、 /\*+ INDEX\_SS(T T\_IND) \*/

从9i开始, oracle引入了这种索引访问方式。当在一个联合索引中, 某些谓词条件并不在联合索引的一列时, 可以通过Index Skip Scan来访问索引获得数据。当联合索引第一列的唯一值个数很少时, 用这种方式比全表扫描效率高。

### 3. 和表的关联相关的hint

1、 /\*+ leading(table\_1,table\_2) \*/

在多表关联查询中, 指定哪个表作为驱动表, 即告诉优化器首先要访问哪个表上的数据。

```
select /*+ leading(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

2、 /\*+ order \*/

让Oracle根据from后面表的顺序来选择驱动表, oracle建议使用leading, 他更为灵活

```
select /*+ order */ t.* from t,t1 where t.id=t1.id;
```

3、 /\*+ use\_nl(table\_1,table\_2) \*/

在多表关联查询中, 指定使用nest loops方式进行多表关联。

```
select /*+ use_nl(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

4、 /\*+ use\_hash(table\_1,table\_2) \*/

在多表关联查询中, 指定使用hash join方式进行多表关联。

```
select /*+ use_hash(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

在多表关联查询中, 指定使用hash join方式进行多表关联, 并指定表t为驱动表。

```
select /*+ use_hash(t,t1) leading(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

5、 /\*+ use\_merge(table\_1,table\_2) \*/

在多表关联查询中, 指定使用merge join方式进行多表关联。

```
select /*+ use_merge(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

6、 /\*+ no\_use\_nl(table\_1,table\_2) \*/

在多表关联查询中, 指定不使用nest loops方式进行多表关联。

```
select /*+ no_use_nl(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

7、 /\*+ no\_use\_hash(table\_1,table\_2) \*/

在多表关联查询中，指定不使用hash join方式进行多表关联。

```
select /*+ no_use_hash(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

8、 /\*+ no\_use\_merge(table\_1,table\_2) \*/

在多表关联查询中，指定不使用merge join方式进行多表关联。

```
select /*+ no_use_merge(t,t1) */ t.* from t,t1 where t.id=t1.id;
```

## 4. 其他常用的hint

1、 /\*+ parallel(table\_name n) \*/

在sql中指定执行的并行度，这个值将会覆盖自身的并行度

```
select /*+ parallel(t 4) */ count(*) from t;
```

2、 /\*+ no\_parallel(table\_name) \*/

在sql中指定执行的不使用并行

```
select /*+ no_parallel(t) */ count(*) from t;
```

3、 /\*+ append \*/

以直接加载的方式将数据加载入库

```
insert into t /*+ append */ select * from t;
```

4、 /\*+ dynamic\_sampling(table\_name n) \*/

设置sql执行时动态采用的级别，这个级别为0~10

```
select /*+ dynamic_sampling(t 4) */ * from t where id > 1234
```

5、 /\*+ cache(table\_name) \*/

进行全表扫描时将table置于LRU列表的最活跃端，类似于table的cache属性

```
select /*+ full(employees) cache(employees) */ last_name from employees
```