Oracle 统计信息管理

公益讲座11：00准时开始，请大家先浏览云技术微信公众号技术文章。资料会在各群同步发布，已入群客户请勿重复入群！

20-21

数据库和云讲座群

甲骨文云技术公众号

B站专家系列课程

基于 Oracle 数据库
免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：

400-699-8888

* 活动最终解释权归甲骨文公司所有

驾驭执行计划 - Oracle 统计信息管理

甲骨文技术公益课 - 数据库专场

2023 年 6 月 2 日 11:00

线上直播

Oracle RWP Team
Yuwen zhou

Agenda

1. 为什么要收集统计信息
2. 如何收集统计信息
3. 统计信息包含什么内容
4. 什么时候收集统计信息
5. 其它类型的统计信息
6. 统计信息采集性能
7. 无需收集统计信息的情况
8. 变更管理
9. 案例分析

为什么要收集统计信息

为什么要收集统计信息?

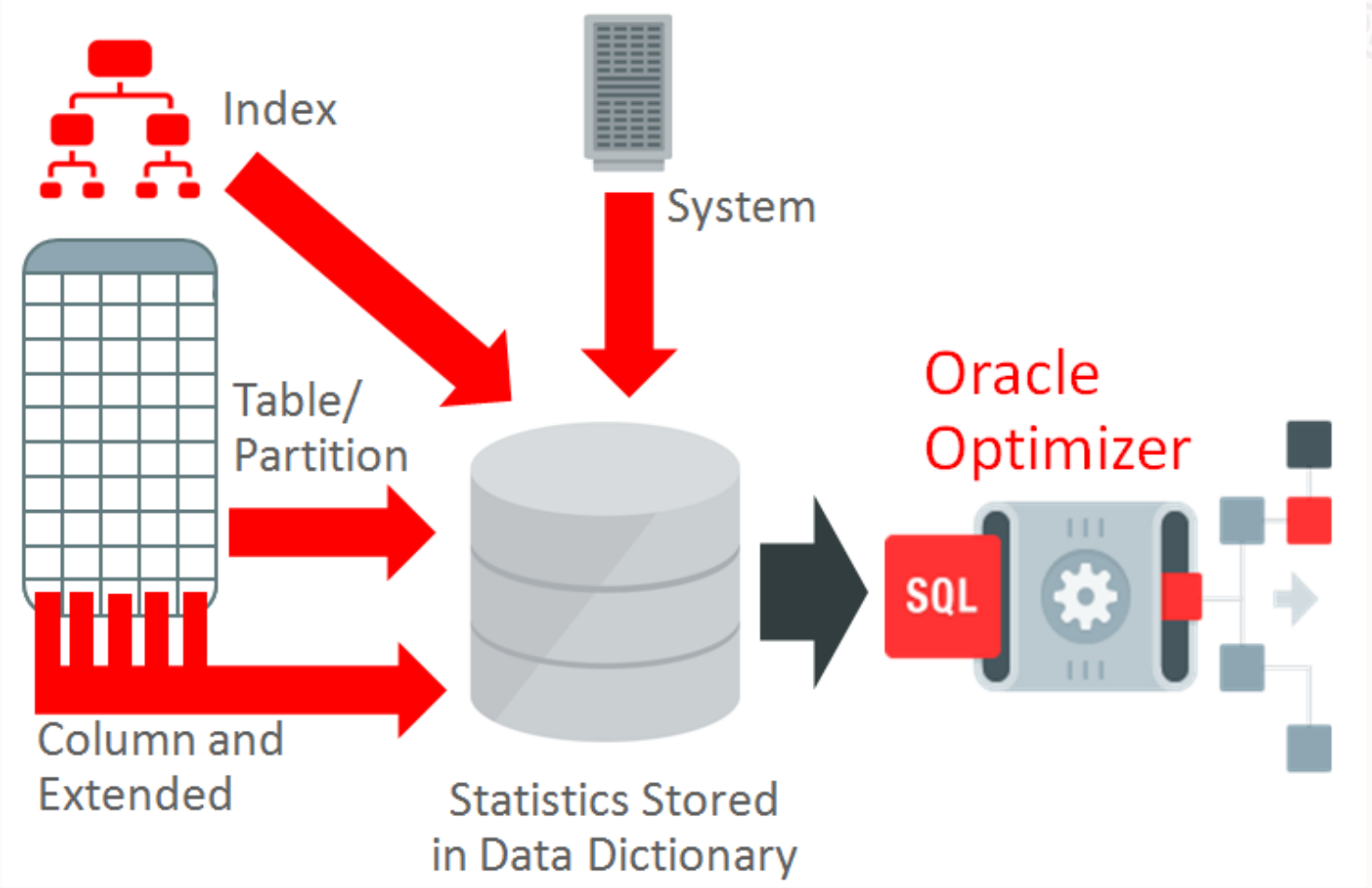
- 统计信息对于优化器生成一个好的执行计划来说是必要的
  - 优化器为了产生最佳的执行计划，依赖于对象统计信息

- 保持最新的统计信息非常重要
  - 当统计信息已经过时，需要刷新

如何收集统计信息

优化器统计信息

对象统计信息，是优化器引擎生成执行计划的必要输入，过时或错误统计信息很可能导致一个错误的执行计划。



自动统计信息采集任务

为所有缺失或过时统计信息的对象自动收集，默认表的数据修改达到10%就是候选对象

- 通过AutoTask在预定义的维护窗口调度

```
BEGIN
  DBMS_AUTO_TASK_ADMIN.ENABLE(
    client_name => 'auto optimizer stats collection',
    operation   => NULL,
    window_name => NULL);
END;
/
```

手工统计信息采集

使用DBMS_STATS包

- Analyze命令已经过时,不建议使用
- GATHER_*_STATS过程包含非常多的参数

  通常只要设置下前面3-4个参数

  SCHEMA NAME 用户名

  TABLE NAME 表名

  DEGREE 并行度

  PARTITION NAME 分区名

调整默认采集参数

有些情况下需要调整默认的统计信息采集参数

- 默认参数没有开启的特性
  - 增量统计信息，适用于分区表
  - 并发统计信息采集，适用于短时间内需要采集大量对象统计信息
- 收集直方图
  - 对于某些列收集直方图

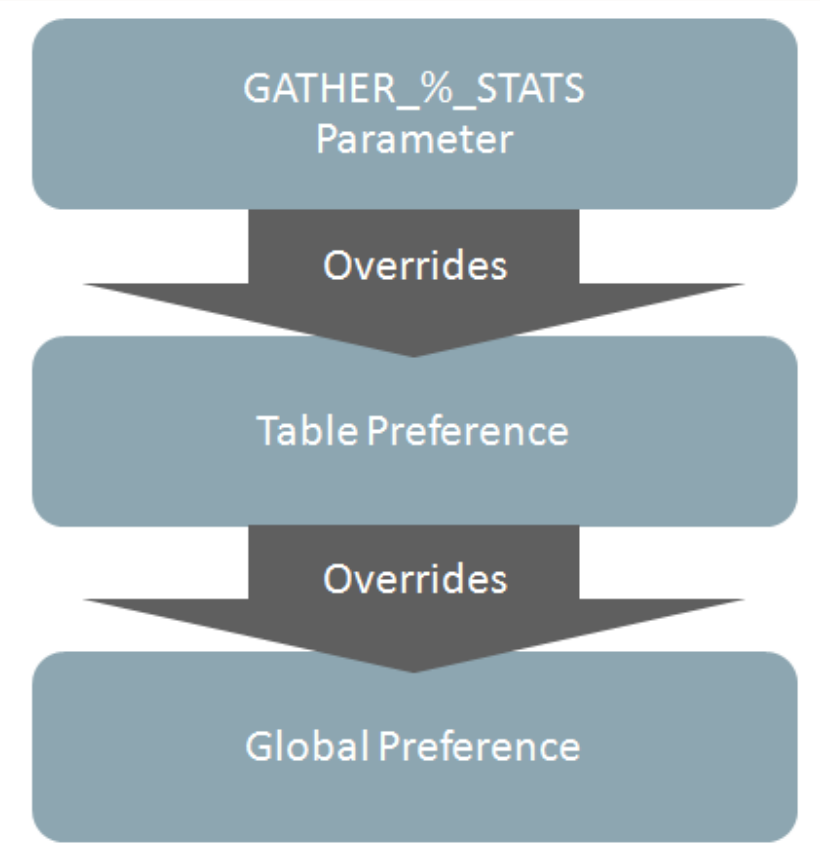
调整默认采集参数

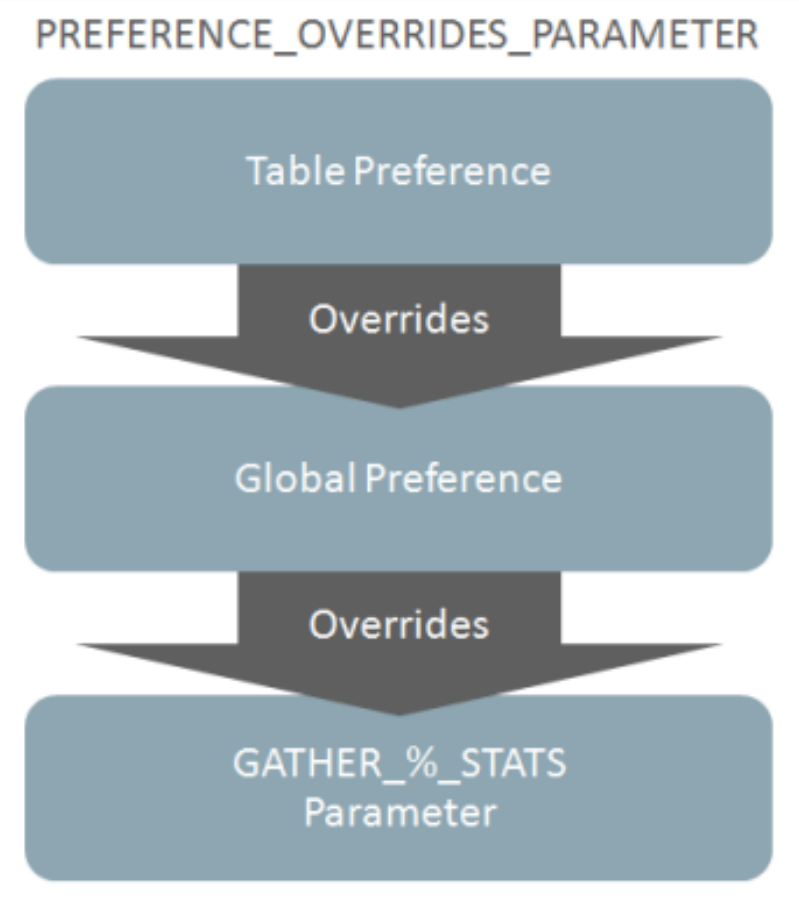
DBMS_STATS.SET_GLOBAL_PREFS/SET_TABLE_PREFS/SET_SCHEMA_PREFS/
SET_DATABASE_PREFS

设置完成后，下次采集生效

- APPROXIMATE_NDV_ALGORITHM
- AUTO_STAT_EXTENSIONS
- AUTO_TASK_STATUS
- AUTO_TASK_MAX_RUN_TIME
- AUTO_TASK_INTERVAL
- AUTOSTATS_TARGET
- CASCADE
- CONCURRENT
- DEGREE
- ESTIMATE_PERCENT
- GLOBAL_TEMP_TABLE_STATS
- GRANULARITY
- INCREMENTAL
- INCREMENTAL_LEVEL
- INCREMENTAL_STALENESS
- METHOD_OPT
- NO_INVALIDATE
- OPTIONS
- PREFERENCE_OVERRIDES_PARAMETER
- PUBLISH
- STALE_PERCENT
- STAT_CATEGORY
- TABLE_CACHED_BLOCKS
- WAIT_TIME_TO_UPDATE_STATS

参数的优先级别





12.2版本新增加preference_overrides_parameter参数,默认为FALSE，如果设置为TRUE，优先级别的顺序发生了变化。

统计信息相关的其它操作

下面是常用的统计信息相关操作

Feature	Database	Dictionary	Schema	Table*	Index*
Gather/Delete	✓	✓	✓	✓	✓
Lock/Unlock			✓	✓	
Copy				✓	
Restore	✓	✓	✓	✓	
Export/Import	✓	✓	✓	✓	✓
Get/Set				✓	✓

Table和index可以对单个分区进行统计信息操作

如何收集统计信息

常用采集命令

- 使用 dbms_stats
  - gather_table_stats收集单表统计信息
  - gather_schema_stats收集用户统计信息

- 例子：exec dbms_stat.gather_table_stats(user, 'TABLE_NAME', degree=>16);

- 参数
  - schema
  - table
  - degree
  - estimate_percent
  - cascade
  - method_opt
  - ……

采样比例

统计采集时候都会纠结：*“我应该使用多大的采样比例呢”*
采样比例通过参数ESTIMATE_PRECENT来控制
模式设置是AUTO_SAMPLE_SIZE

- 使用hash方式进行采样
- 相当于10%的采样开销
- 采样精度和100%采样几乎一样

如何采集统计信息

固定采样比例 vs AUTO_SAMPLE_SIZE

Run Num	AUTO_SAMPLE_SIZE	10% SAMPLE	100% SAMPLE
1	00:02:21.86	00:02:31.56	00:08:24.10
2	00:02:38.11	00:02:49.49	00:07:38.25
3	00:02:39.31	00:02:38.55	00:07:37.83
Column Name	**NDV with AUTO_SAMPLE_SIZE**	**NDV with 10% SAMPLE**	**NDV with 100% SAMPLE**
C1	59852	31464	60351
C2	1270912	608544	1289760
C3	768384	359424	777942

建议收集统计信息使用默认AUTO_SAMPLE_SIZE，对于oracle12c 如果要生成HYBIRD或TOP Frequency，必须使用AUTO_SAMPLE_SIZE进行采样

统计信息采集那些信息

统计信息包含

- 表/索引/分区都有各自的统计信息
- 表或分区统计信息

table_name	num_rows	blocks	empty_blocks	avg_row_len
lineorder	53,986,608	697,567	0	87
date_dim	2,554	38	0	87

- 表或分区列的统计信息

table_name	column_name	num_distinct	low_value	high_value	density	num_nulls
date_dim	d_datekey	2,554	31-DEC-01	29-DEC-08	1/2554	0
date_dim	d_dayofweek	7	1	7	1/7	0
date_dim	d_month	12	1	12	1/12	0

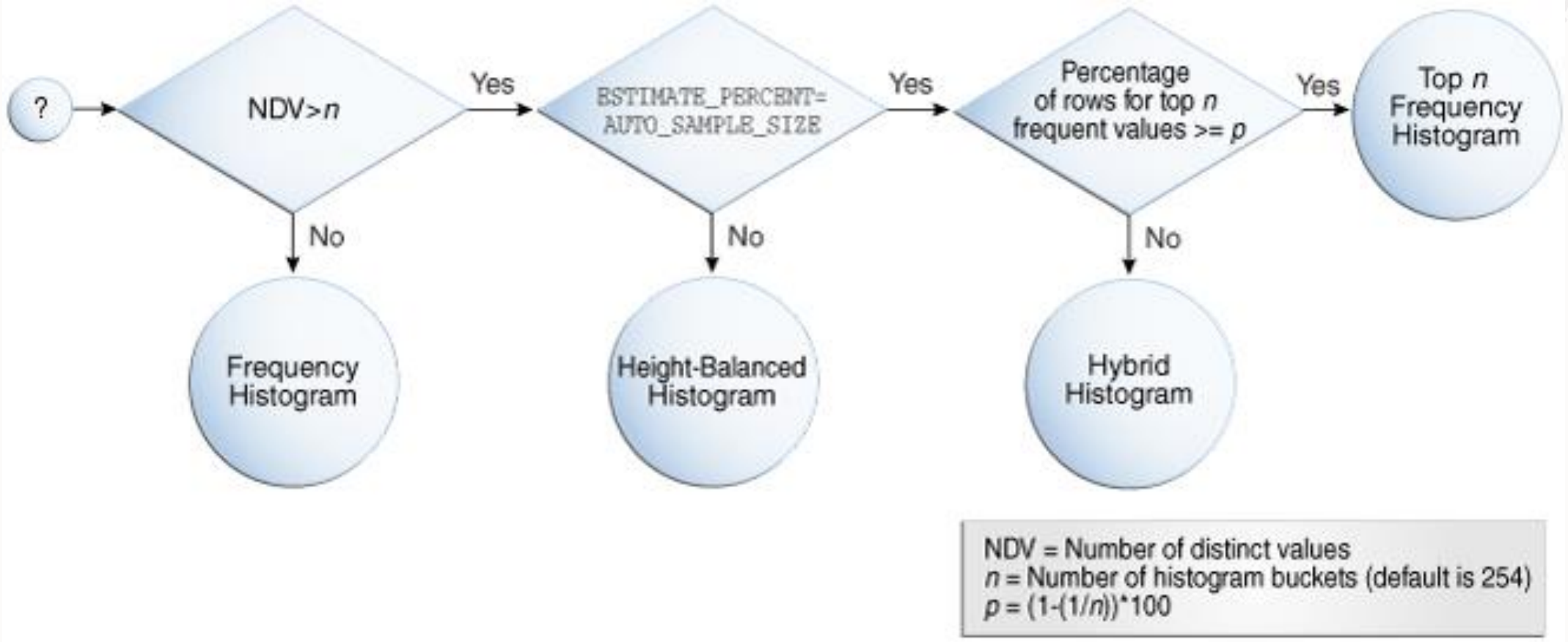
- 表或分区列的统计信息

index_name	blevel	Leaf_blocks	Distinct_keys	Clustering_factor	Num_rows	Avg_leaf_blocks_per_key
INDEX1_Pk	1	2	1000	979	1000	1
INX_COL2	1	2	431	478	497	1
IDX_COL3	1	3	6	175	1000	1

直方图

- 用来更进一步描述某列的数据分布情况
- 基于SQL使用情况来创建
  - sys.col_usage$保存了谓词条件列和join条件列的使用情况
  - 当采集统计信息时候，这些列就是创建直方图的候选列
- 直方图类型
  - Frequency ---- 列的NDV<=254
  - Top-frequency----列的NDV>254 但是高频的254个列值占总行数比例大于100-100/n
  - Hybrid Histogram ---列的NDV>254 但是高频的254个列值占总行数比例小于100-100/n
  - Height-balanced---如果使用了AUTO_SAMPLE_SIZE将不再创建这个类型的直方图
    - 被Hybrid直方图替换
    - 列的NDV > 254
    - 如果使用estimate_percent依然会创建
    - 升级后保留

Oracle12c中新的直方图类型



使用数据字典查看统计信息

可以使用下面的数据字典查看对象统计信息

Object	Table/Index-Level Statistics	Partition-Level Statistics	Subpartition-Level Statistics
Tables	user_tab_statistics	user_tab_statistics	user_tab_statistics
Columns	user_tab_col_statistics user_tab_histograms	user_part_col_statistics user_part_histograms	user_subpart_col_statistics user_subpart_histograms
Indexes	user_ind_statistics	user_ind_statistics	user_ind_statistics

```
SELECT column_name, num_distinct, histogram
FROM   user_tab_col_statistics
WHERE  table_name = 'SALES';

COLUMN_NAME                    NUM_DISTINCT HISTOGRAM
------------------------------ ------------ ---------------
CUST_ID                                 630 HEIGHT BALANCED
TIME_ID                                 620 HEIGHT BALANCED
CHANNEL_ID                                5 FREQUENCY
PROMO_ID                                116 FREQUENCY
QUANTITY_SOLD                            44 FREQUENCY
AMOUNT_SOLD                             583 HEIGHT BALANCED
```

METHOD_OPT

默认是FOR ALL COLUMNS SIZE AUTO

- 列上面是否创建直方图由列的使用情况决定(COL_USAGE$)

全局禁用直方图

- dbms_stats.set_global_prefs(method_opt => 'FOR ALL COLUMNS SIZE 1’);

给指定的某些列创建直方图

- method_opt => 'FOR ALL COLUMNS SIZE 1, FOR COLUMNS SIZE 254 CUST_ID PROD_ID');

给指定的某些列禁用直方图

- dbms_stats.set_table_prefs('SH', 'SALES','METHOD_OPT', 'FOR ALL COLUMNS SIZE AUTO, FOR COLUMNS SIZE 1 PROD_ID');

更新直方图统计信息

- METHOD_OPT=>’FOR ALL COLUMNS REPEAT’
- 备注: 不会修改桶的数量

METHOD_OPT

使用非默认的 METHOD_OPT...

- 不要使用 ‘FOR ALL INDEX COLUMNS SIZE 254’
  - 收集统计信息时间变长
  - 统计信息占用空间更多
  - 不收集非索引列直方图
- 不要使用 ‘FOR ALL COLUMNS SIZE 254’
  - 收集统计信息时间变长
  - 统计信息占用空间更多

什么时候收集统计信息

过时统计信息导致谓词越界问题

保持最新统计信息

查询: **SELECT Sum(total_sales)**
FROM Sales
WHERE sales_date='02-DEC-2015';

列 sales_date的统计信息认为最大值是 **'31-JAN-2015'**

优化器检查谓词条件是否在列的min和max值之间

谓词越界意味着条件min和max值之间

优化器按照越界的距离预估返回行数

- 基于谓词和min和max值之间差值决定，差值越大，选择率就越低

什么时候收集统计信息

自动统计信息采集

- Oracle 自动为所有没有或统计信息已经过时的对象收集统计信息
- 统计信息作业在预先定义好的维护窗口运行
- dbms_auto_task_admin.enable(client_name => 'auto optimizer stats collection',operation => NULL,window_name => NULL)

```
SQL> SELECT task_name, status
  2  FROM dba_autotask_task
  3  WHERE client_name = 'auto optimizer stats collection';

TASK_NAME         STATUS
----------------- -------
gather_stats_prog ENABLED
SQL> SELECT w.window_name, w.repeat_interval, w.duration, w.enabled
  2  FROM dba_autotask_window_clients c, dba_scheduler_windows w
  3  WHERE c.window_name = w.window_name
  4  AND c.optimizer_stats = 'ENABLED';

WINDOW_NAME      REPEAT_INTERVAL                                    DURATION      ENABLED
---------------- -------------------------------------------------- ------------- -------
MONDAY_WINDOW    freq=daily;byday=MON;byhour=22;byminute=0; bysecond=0 +000 04:00:00 TRUE
TUESDAY_WINDOW   freq=daily;byday=TUE;byhour=22;byminute=0; bysecond=0 +000 04:00:00 TRUE
WEDNESDAY_WINDOW freq=daily;byday=WED;byhour=22;byminute=0; bysecond=0 +000 04:00:00 TRUE
THURSDAY_WINDOW  freq=daily;byday=THU;byhour=22;byminute=0; bysecond=0 +000 04:00:00 TRUE
FRIDAY_WINDOW    freq=daily;byday=FRI;byhour=22;byminute=0; bysecond=0 +000 04:00:00 TRUE
SATURDAY_WINDOW  freq=daily;byday=SAT;byhour=6;byminute=0; bysecond=0  +000 20:00:00 TRUE
SUNDAY_WINDOW    freq=daily;byday=SUN;byhour=6;byminute=0; bysecond=0  +000 20:00:00 TRUE
```

什么时候收集统计信息

如果自动采集作业时间不合适

自动统计任务窗口之外进行大量数据装载
- 在ETL或ELT作业完成后收集统计信息

假设数据加载到分区表
- 使用 dbms.stats.copy_table_stats()

其它类型统计信息

其它类型统计信息

数据字典统计信息
- 数据字典表的统计信息
- DBMS_STATS.GATHER_DICTIONARY_STATS采集

Fixed object统计信息
- X$相关表的统计信息
- DBMS_STATS.GATHER_FIXED_OBJECTS_STATS采集

系统统计信息
- 最好保存现状…
  - 使用默认设置（不要采集）

其它类型的统计信息

当仅有表和列统计无法产生合理估算时

两种类型的扩展统计信息

- Column groups统计
  - 当SQL语句中单个表的多个列被where条件同时引用时候，Column groups统计非常有用
- Expression 统计
  - Expression statistics 当表列在where条件中以复杂表达式出现时候，表达式统计信息非常有用

可以手动或自动创建

当表统计信息收集时候可以自动维护

扩展统计信息 – Column Group

列之间的相关性会导致错误的估计Cardinality

```
SELECT  COUNT(*)
FROM    calendar
WHERE   month_name = 'JANUARY'
AND     day_name   = 'MONDAY';
```

Cardinality = #Rows * 1/(NDV month_name) * 1/(NDV day_name)

= #Rows * 1/12 * 1/7　估算正确!

```
SELECT  COUNT(*)
FROM    calendar
WHERE   month_name = 'NOVEMBER'
AND     star_sign   = 'SCORPIO';
```

Cardinality = #Rows * 1/(NDV month_name) * 1/(NDV star_sign

= #Rows * 1/12 * 1/12　估算错误!

天蝎座从 10月23 – 11月 21，大部分时间都是在11月份

扩展统计信息 – Column Group

使用 column group 统计提升Cardinality估算的准确度

```
dbms_stats.create_extend_stats(NULL, 'CALENDAR', '(MONTH_NAME,STAR_SIGN)');
dbms.gather_tabe_stats(NULL, 'CALENDAR');

SELECT
  TABLE_NAME,
  COLUMN_NAME,
  NUM_DISTINCT as NDV
FROM
  USER_TAB_COL_STATISTICS
ORDER BY 1,2;
```

```
TABLE_NAME  COLUMN_NAME                      NDV
----------  ------------------------------ --------
CALENDAR    DATE_ID                             365
CALENDAR    MONTH_NAME                           12
CALENDAR    SYS_STUWHPY_ZSVI_W3#C$I3EUUYB4       24
CALENDAR    STAR_SIGN                            12
```

```
SELECT  COUNT(*)
FROM    calendar
WHERE   month_name = 'NOVEMBER'
AND     star_sign   = 'SCORPIO';
```

Cardinality = #Rows * 1/(24) Correct!

扩展统计信息 – 表达式

比如下面的查询

```
SELECT *
FROM   customers
WHERE UPPER(CUST_LAST_NAME) = 'SMITH';
```

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |      1 |        |      1 |
|   1 |  SORT AGGREGATE              |           |      1 |      1 |      1 |
|*  2 |   TABLE ACCESS STORAGE FULL| CUSTOMERS |      1 |    555 |     79 |
-------------------------------------------------------------------------------
```

当在列上面使用函数时候，优化默认
无法知道函数对列的影响
优化器统一认为将返回1%的数据

```
SELECT COUNT(*) FROM customers;
```

COUNT(*)

55500

Cardinality 估算是1%的总行数

扩展统计信息 – 表达式解决方法

```
SQL> BEGIN
  2  dbms_stats.gather_table_stats(null,'customers',method_opt =>'for all columns size skewonly for columns (UPPER(CUST_LAST_NAME))');
  3  END;
  4  /

PL/SQL procedure successfully completed.

SQL>
SQL> SELECT column_name, num_distinct, histogram
  2  FROM   user_tab_col_statistics
  3  WHERE  table_name = 'CUSTOMERS';

COLUMN_NAME                    NUM_DISTINCT HISTOGRAM
------------------------------ ------------ ---------------
SYS_STUSKCCJE8MV8IIBWT5PA5A41V          908 HEIGHT BALANCED
CUST_ID                               55500 HEIGHT BALANCED
CUST_FIRST_NAME                        1300 HEIGHT BALANCED
CUST_LAST_NAME                          908 HEIGHT BALANCED
CUST_GENDER                               2 FREQUENCY
CUST_YEAR_OF_BIRTH                       75 FREQUENCY
CUST_MARITAL_STATUS                      11 FREQUENCY
CUST_STREET_ADDRESS                   49900 HEIGHT BALANCED
CUST_POSTAL_CODE                        623 HEIGHT BALANCED
```

系统自动产生
新的列名

扩展统计信息

自动创建column group

1. 开始 捕获column group的使用（12c之后自动创建column group ）

```
SQL> conn / as sysdba
Connected.
SQL> begin
  2   dbms_stats.seed_col_usage(null,null,300);
  3  end;
  4  /

PL/SQL procedure successfully completed.
```

监控接下来的5分钟内, SQL语句where条件中使用的列和group by引用的列都会被监控

扩展统计信息

自动创建column group

2. 运行相关的SQL，查询1

```
SQL> select * from customers where cust_city='Los Angeles'
  2  and cust_state_province='CA' and country_id=52790;

932 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     1 |   189 |   444   (1)| 00:00:06 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |     1 |   189 |   444   (1)| 00:00:06 |
-------------------------------------------------------------------------------
```

实际返回932行，优化器低估了1000倍，因为不知道3个列之间有相关性

扩展统计信息

自动创建column group

2. 运行相关的SQL，查询2

```
SQL> select country_id,cust_state_province,count(cust_city)
  2  from customers
  3  group by country_id,cust_state_province;

145 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1577413243

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |  1949 | 31184 |   446   (1)| 00:00:06 |
|   1 |  HASH GROUP BY     |           |  1949 | 31184 |   446   (1)| 00:00:06 |
|   2 |   TABLE ACCESS FULL| CUSTOMERS | 55500 |   867K|   444   (1)| 00:00:06 |
```

实际返回145行，优化器高估了一个数据量级，因为不知道列country 和 state有相关性

扩展统计信息

自动创建column group

3. 检查列的使用情况

```
SQL> SELECT dbms_stats.report_col_usage(user, 'customers')
       FROM dual;

  COLUMN USAGE REPORT FOR SH.CUSTOMERS
  1. COUNTRY_ID                    : EQ
  2. CUST_CITY                     : EQ
  3. CUST_STATE_PROVINCE           : EQ
  4. (CUST_CITY, CUST_STATE_PROVINCE,  COUNTRY_ID)    : FILTER
  5. (CUST_STATE_PROVINCE, COUNTRY_ID)  : GROUP_BY
```

EQ 表示查询1对3个列都使用了等值查询

FILTER 表示查询1中，3个列用作谓词过滤条件而不是join列

GROUP_BY 表示查询2使用了group by表达式

扩展统计信息

自动创建column group

4. 按照列使用情况创建扩展统计信息

```
SQL>  SELECT dbms_stats.create_extended_stats(user, 'customers')  FROM dual;

DBMS_STATS.CREATE_EXTENDED_STATS(USER,'CUSTOMERS')
------------------------------------------------------------------------------
###############################################################################

EXTENSIONS FOR SH.CUSTOMERS
...........................

1. (CUST_CITY, CUST_STATE_PROVINCE,
    COUNTRY_ID)                          : SYS_STUMZ$C3AIHLPBROI#SKA58H_N created
2. (CUST_STATE_PROVINCE, COUNTRY_ID)     : SYS_STU#S#WF25Z#QAHIHE#MOFFMM_ created
###############################################################################
```

Column group 统计信息将自动创建，并且每次统计信息更新时候会自动维护

12c开始SQL Plan Directives 会触发自动创建column group统计信息

扩展统计信息

检查扩展统计信息后的结果

5. 对上面2个SQL执行explain plan for 解析下执行计划，然后收集表统计信息，
   再检查估算和实际的差异

```
SQL> select * from customers where cust_city='Los Angeles'
  2  and cust_state_province='CA' and country_id=52790;

932 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

-------------------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |   874 |   177K|   444   (1)| 00:00:06 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |   874 |   177K|   444   (1)| 00:00:06 |
```

```
SQL> select country_id,cust_state_province,count(cust_city)
  2  from customers group by country_id,cust_state_province;

145 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1577413243

-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |   145 |  2320 |   446   (1)| 00:00:06 |
|   1 |  HASH GROUP BY     |           |   145 |  2320 |   446   (1)| 00:00:06 |
|   2 |   TABLE ACCESS FULL| CUSTOMERS | 55500 |   867K|   444   (1)| 00:00:06 |
```

可以看到，估算已经非常准确了，后面的系统会自动维护column group的统计信息。

收集统计信息性能

统计信息采集的性能

如何加速统计信息采集

并行或并发选项加速统计信息采集

- 使用DEGREE并行执行，同一个表的分区之间还是串行执行
- 使用CONCURRENT并发，分区之间可以并行收集
- DEGREE和CONCURRENT同时使用

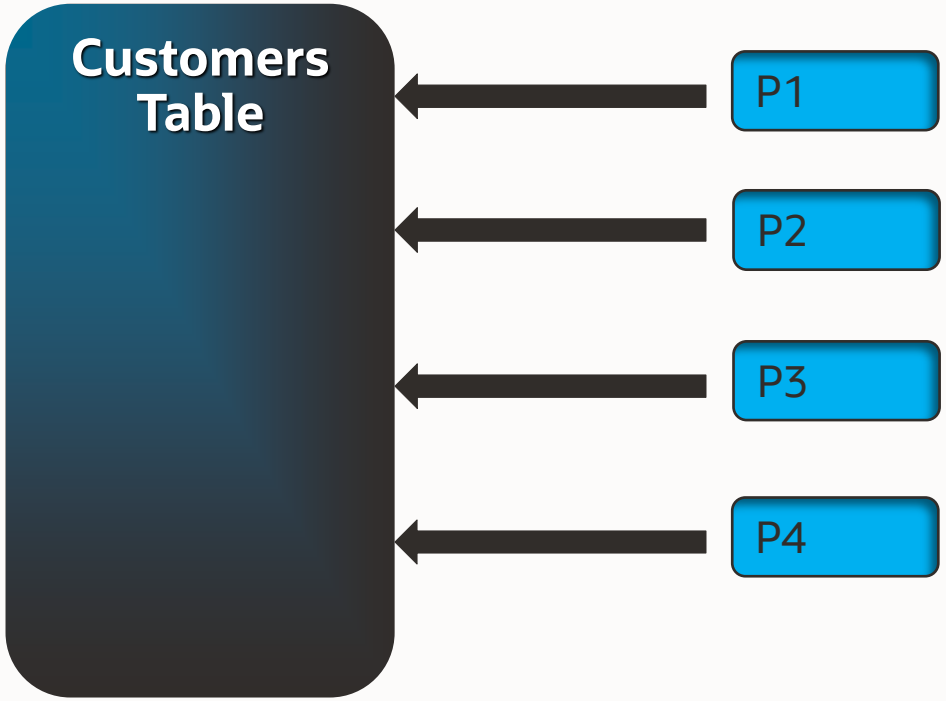
对于分区表使用增量统计信息采集

统计信息采集的性能

单个对象的并行统计信息收集

通过 GATHER_*_STATS 参数 DEGREE控制

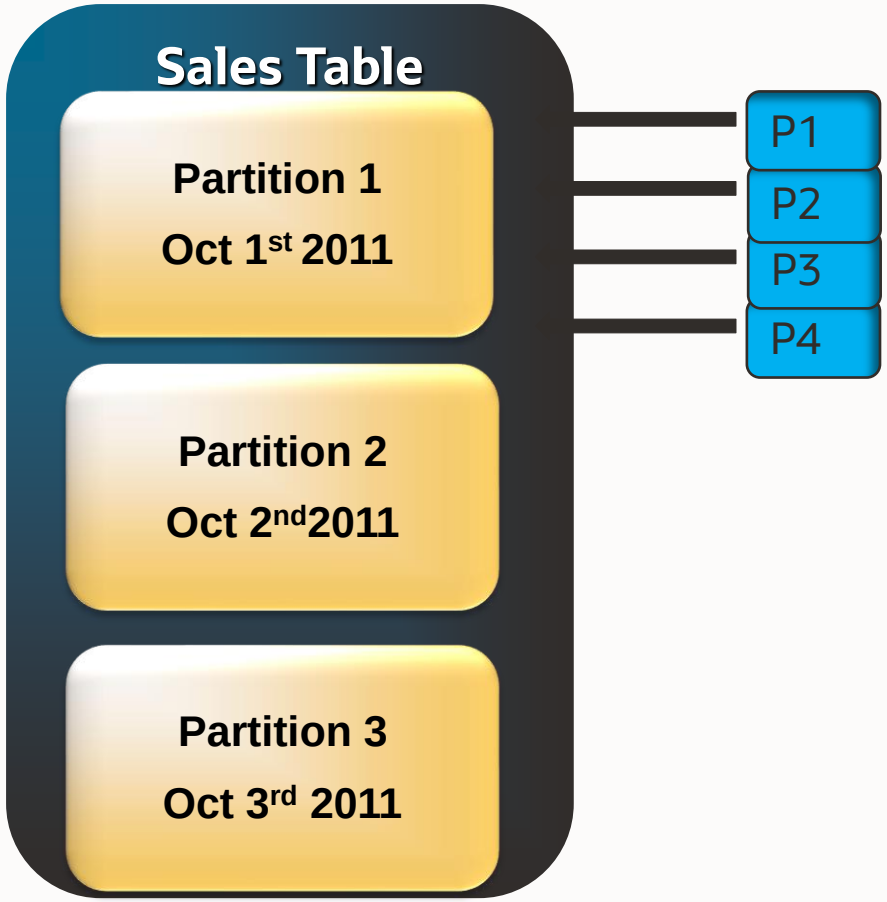
假设Customers表的degree定义为4，
那么收集统计信息时候有4个进程运行。



统计信息采集的性能

单个对象并行统计信息收集的限制

Exec DBMS_STATS.GATHER_TABLE_STATS(null, 'SALES');



单个分区可以并行收集，收集完一个分区，再收集下一个，多个分区不能同时收集，对于分区表首次采集时间可能比较长。

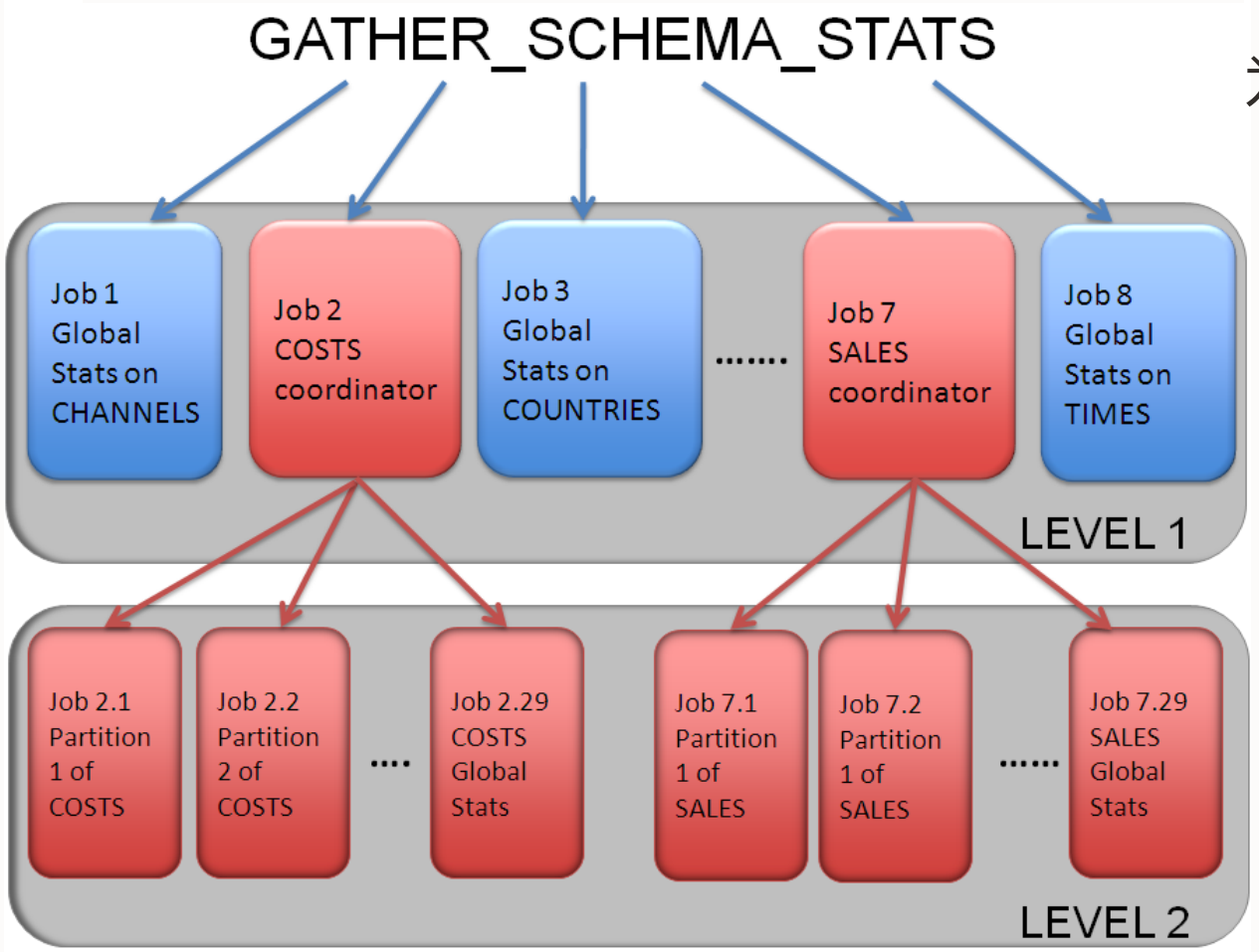
统计信息采集的性能

对象之间并行统计信息收集

- 同一时间收集多个对象统计信息
- 通过DBMS_STATS的 CONCURRENT属性控制
- 使用Scheduler 和高级队列调度
- 并发进程数由参数*job_queue_processes* 决定
- 每个正在运行的任务仍然可以是并行的

统计信息采集的性能

```
Exec DBMS_STATS.GATHER_SCHEMA_STATS('SH');
```



为每个表的每个分区单独创建调度作业任务

Level 1 除了包含每个非分区表的采集作业，还包含分区表协调作业

Level 2 中包含分区表每一个分区的统计信息采集作业

统计信息采集的性能

分区表的增量统计信息采集

默认情况下，对于分区表的一个分区加载数据，在收集全局统计信息时候需要扫描所有的分区

- 消耗大量的CPU和IO资源

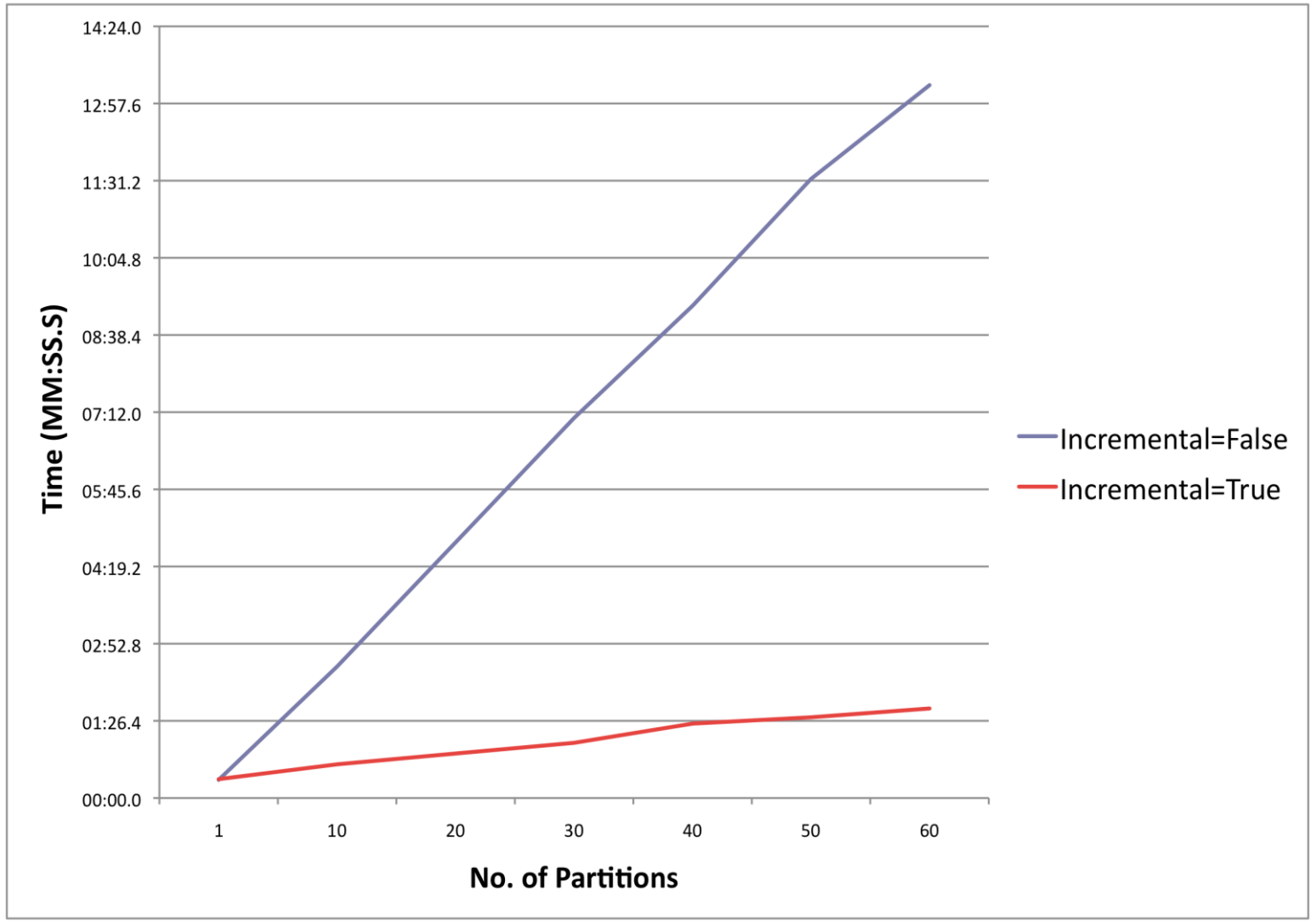
增量统计信息适用于分区表

- 表的全局统计信息通过每个分区合并得到
- 可以大大减少统计信息收集时间
- 通过表优先级参数INCREMENTAL控制
- EXEC dbms_stats.set_table_prefs(null,'SALES','INCREMENTAL','TRUE')

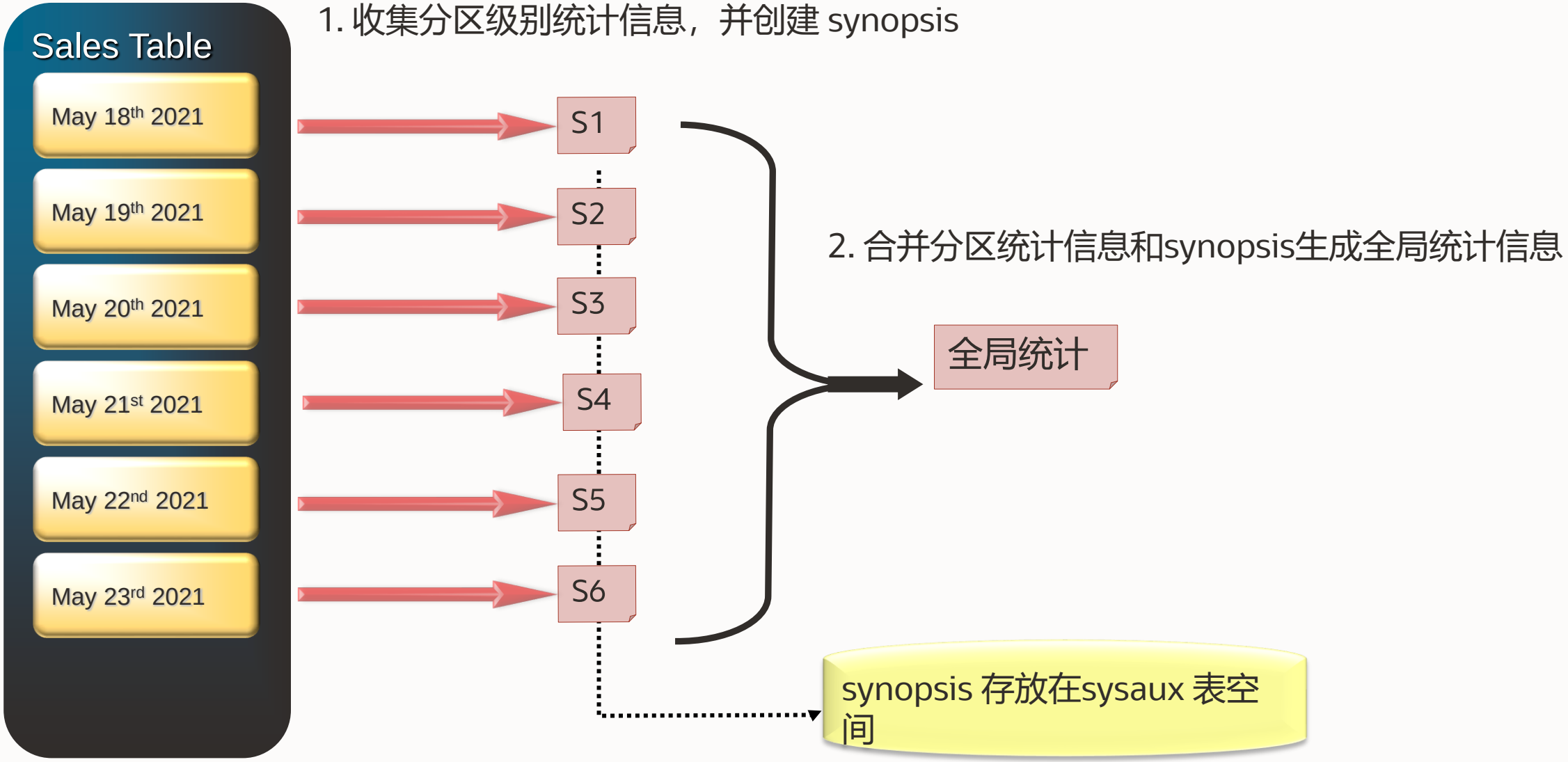
增量统计信息时间对比

- 按天分区，一共60个分区
- 每天1,000,000 行数据

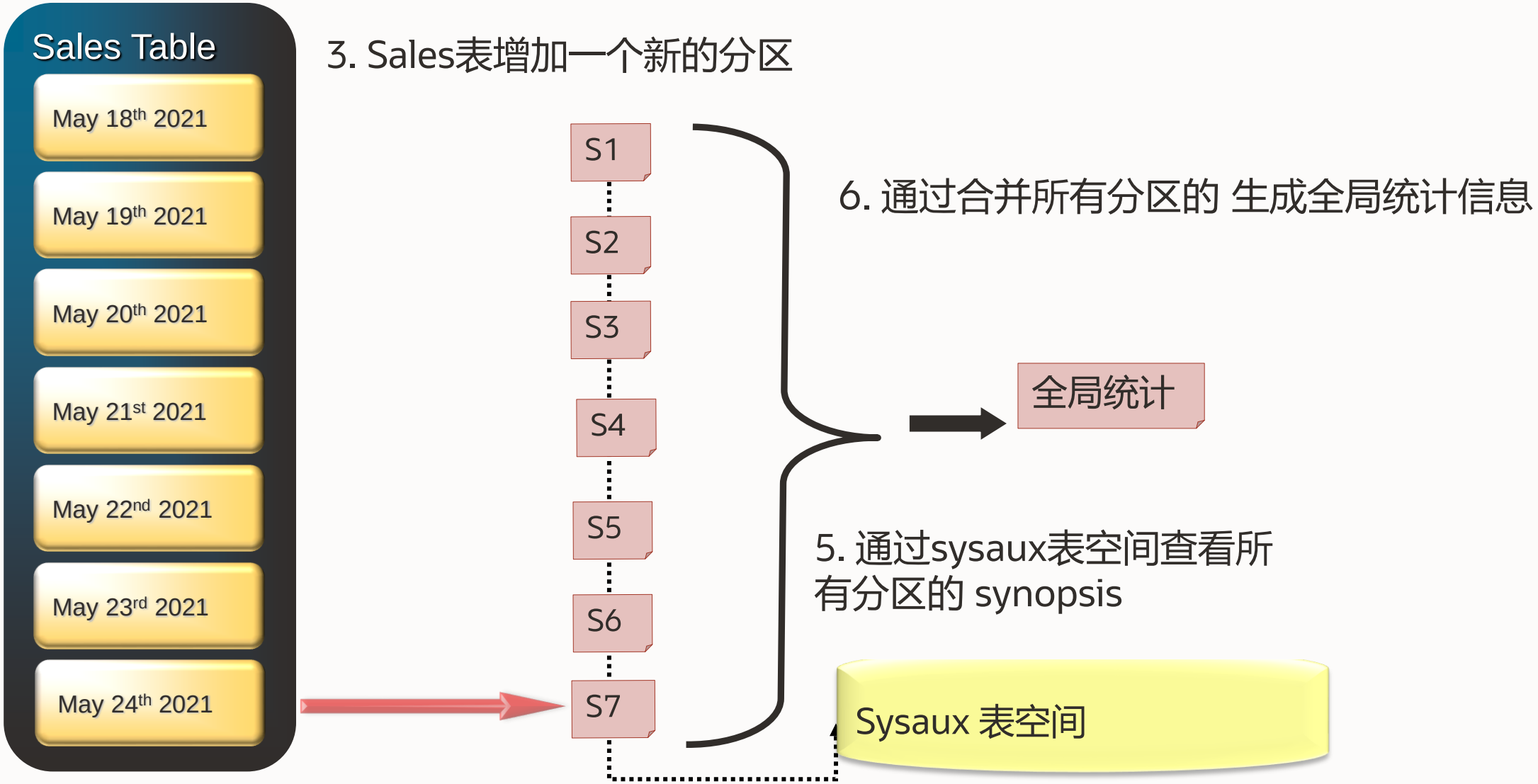
对比打开增量统计信息后，统计收集时间和没有打开的时间。



12c以后增量统计信息采集



12c以后增量统计信息采集



12c以后增量统计信息采集– 分区交换

从12c以后的版本，对于要执行分区交换的非分区表，
可以在这个表上面创建synopsis增量信息。

dbms_stats.set_table_prefs ('SH','EXTAB','**INCREMENTAL**','**TRUE**');
dbms_stats.set_table_prefs ('SH','EXTAB','**INCREMENTAL_LEVEL**','**TABLE**');

统计信息采集的性能

分区数据变更带来问题

Oracle 数据库 11g 中使用增量统计，分区或子分区上的单行DML 操作将使它成为统计更新目标，即使它没有标记为 stale。换句话说，假设更新包含 100 万行的分区中的一行，下一次统计信息作业会对整个分区收集统计信息。

Oracle Database 12c 默认和11g一样，但可以通过将 DBMS_STATS 首选项参数 INCREMENTAL_STALENESS 从其默认值 (NULL) 更改为USE_STALE_PERCENT（默认10%），可以控制什么时候分区统计信息是过时的。

```
dbms_stats.set_table_prefs(
ownname=>null,
tabname=>'SALES’,
pname =>'INCREMENTAL_STALENESS’,
pvalue=>'USE_STALE_PERCENT, USE_LOCKED_STATS');
```

什么时候不建议收集统计信息

不需要收集统计信息的情况

数据量变化非常频繁的表

- 随时间表的数据量变化非常快的表，当表中具有代表性数据量时收集统计信息
  锁定统计信息以确保统计信息收集作业不会覆盖代表性统计信息
- 比如排队信息表，开始表为空，然后数据入表, 处理数据，处理完成表为空

11g版本中的全局临时表

- 用来临时存放中间结果
- 一些会话有很多记录，另外一些可能只有少数记录

12c中全局临时表默认包含session级别统计信息

- DBMS_STATS 设置 GLOBAL_TEMP_TABLE_STATS 为 SHARED 或者SESSION

中间表

- 一次写入，一次读取，然后被Truncate或deleted
- 常见于ETL过程中的中间表

不需要收集统计信息的情况

中间表

常见于ETL过程中的中间表
一次写入，一次读取，然后被Truncate或deleted

这样的表无需收集统计信息，否则会增加ETL的时间
建议使用高级别的动态采样

- 通过 对中间表使用dynamic_sampling hint或修改session参数

```
SELECT /*+ dynamic_sampling(cst 2) */ *
FROM   customers_staging_tab cst
WHERE  cust_address_change ='Y';
```

变更管理

Pending 统计信息

当使用DBMS_STATS.GATHER_*_STATS 过程设置或收集统计信息时候, 需要先校验后才能正式发布统计信息，过程如下：

- DBMS_STATS.SET_TABLE_PREFS(... ‘PUBLISH’, ‘FALSE’)
  设置表统计信息的PUBLISH属性为FALSE;
- DBMS_STATS.GATHER_TABLE_STATS(...); 收集统计信息
- 使用USER_TAB_PENDING_STATS检查pending信息
- DBMS_STATS.DIFF_TABLE_STATS_IN_PENDING 比较PENDING统计信息
- alter session set optimizer_use_pending_statistics = TRUE 验证新的统计信息;
- dbms_stats.publish_pending_stats发布新的统计信息;

还原/比较统计信息

数据库会维护统计信息的历史，默认保留31天

```
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.17.0.0.0

SQL> SELECT dbms_stats.get_stats_history_retention() AS retention FROM dual;

 RETENTION
----------
        31
```

查看差异

- DBMS_STATS.DIFF_TABLE_STATS_IN_HISTORY
- DBMS_STATS.DIFF_TABLE_STATS_IN_PENDING
- DBMS_STATS.DIFF_TABLE_STATS_IN_STATTAB

还原到之前的统计信息

- **DBMS_STATS.RESTORE_TABLE_STATS**
  (ownname=>'SCOTT',tabname=>'SALES',
  as_of_timestamp => systimestamp – 1)

```
TABLE         : T
OWNER         : SYS
SOURCE A      : Current Statistics in dictionary
SOURCE B      : Pending Statistics
PCTTHRESHOLD  : 10
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

TABLE / (SUB)PARTITION STATISTICS DIFFERENCE:
.........................................

OBJECTNAME                 TYP SRC ROWS       BLOCKS     ROWLEN     SAMPSIZE
.....................................................................................

T                          T   A   10110      59         37         4908
                               B   12371      76         37         5426
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

COLUMN STATISTICS DIFFERENCE:
.............................

COLUMN_NAME     SRC NDV     DENSITY    HIST NULLS   LEN  MIN   MAX   SAMPSIZ
.....................................................................................

ID              A   10082   .000099186 NO   0       4    C103  C302  4894
                B   12371   .000080834 NO   0       5    C102  C3026 5443
VAL2            A   9       .000049524 YES  0       3    C10C  C114  4901
                B   9       .000039886 YES  0       3    C10C  C114  5498
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

INDEX / (SUB)PARTITION STATISTICS DIFFERENCE:
.............................................

OBJECTNAME      TYP SRC ROWS    LEAFBLK DISTKEY LF/KY DB/KY CLF     LVL SAMPSIZ
.....................................................................................


                                INDEX: T_PK
                                ...........

T_PK            I   A   10000   20      10000   1     1     9819    1   10000
                    B   12500   28      12500   1     1     12152   NUL 12500
#####################################################################################
```

Oracle12c和19c其它统计信息增强

在线统计信息采集

Oracle12c开始，对于大批量数据操作，可以在线收集统计信息。

- CREATE TABLE ... AS SELECT (CTAS)
- INSERT /*+ APPEND */ INTO ... SELECT
- 空的分区表数据加载

```
SQL> CREATE TABLE tab1 AS
  2  SELECT level AS id, 'This is level '||level  AS  descipt
  3  FROM DUAL CONNECT BY level <= 1000;

Table created.

SQL> SELECT table_name, num_rows FROM   user_tables
  2  WHERE  table_name = 'TAB1';

TABLE_NAME                     NUM_ROWS
------------------------------ ----------
TAB1                             1000
```

限制

- Segment要求初始为空
- Index和直方图统计信息需要单独收集
- 对于分区表无法同时采集到全局和分区统计信息

19c实时统计信息

Oracle19c开始，对于执行常规的DML操作，数据库也会自动ONLINE收集统计信息。

```
PDBORCL> truncate table real_stats;
    Truncated (0.009s).
PDBORCL> exec dbms_stats.delete_table_stats('SCOTT','REAL_STATS');
    Statement completed (0.016s).
PDBORCL> insert /*+ append */ into real_stats select object_id,object_name from user_objects;
    195 rows inserted (0.024s).
PDBORCL> commit;
    Committed (0.005s).
PDBORCL> select table_name,num_rows,blocks,notes from user_tab_statistics where table_name='REAL_STATS';
    TABLE_NAME NUM_ROWS BLOCKS NOTES
    ---------- -------- ------ -----
    REAL_STATS      195      4

    1 rows returned (0.005s).

PDBORCL> select column_name,sample_size,notes from user_tab_col_statistics where table_name='REAL_STATS';

    COLUMN_NAME SAMPLE_SIZE     NOTES
    ----------- ----------- -------------
    OBJECT_ID           195 STATS_ON_LOAD
    OBJECT_NAME         195 STATS_ON_LOAD

    2 rows returned (0.131s).

PDBORCL> insert into real_stats select 1000+rownum,'real stats'||rownum from dual connect by rownum<=1000;
    1000 rows inserted (0.006s).
PDBORCL> commit;
    Committed (0.002s).
PDBORCL> select table_name,num_rows,blocks,notes from user_tab_statistics where table_name='REAL_STATS';
    TABLE_NAME NUM_ROWS BLOCKS          NOTES
    ---------- -------- ------ -------------------------
    REAL_STATS      195      4
    REAL_STATS     1195      8 STATS_ON_CONVENTIONAL_DML
```

空表首次加载了195行数据，统计信息自动更新

常规路径加载1000行数据

常规路径加载数据，触发了统计信息自动更新

19c高频率统计信息采集

Oracle 19c中引入的称为 高频自动优化器统计信息收集的新功能，作为标准的自动统计信息收集作业的补充。默认情况下，高频统计信息收集每 15 分钟进行一次，主要针对

- 没有统计信息的对象
- 统计信息过时的对象
- 前面计划中Cardinality估算严重错误的对象

通过下面命令开启，默认关闭，ADB默认开启

EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_STATUS','ON');

定义任务最长运行时间，默认3600秒

EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_MAX_RUN_TIME','900')

定义任务运行时间间隔，默认900秒

EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_TASK_INTERVAL','900')

23c基于机器学习的实时统计信息

Oracle 23c 融合了...

21c + **23c**

Oracle 21c 创新版的所有特性

300 多项新功能和增强功能

重点关注领域：JSON、图、微服务、开发人员生产力

在23c中，数据库引擎利用机器学习来提高实时统计的准确性。这种变化使我们能够更好地预测数据将如何随时间变化，从而提供更好的信息来规划查询执行。这减少了对昂贵的统计信息收集工作的需求。

利用统计信息解决问题案例

案例1---创建一个index后SQL性能变差了

```
SQL> select count(*) from part_tab where n1=:b1 and daima=:b2 order by n2;

  COUNT(*)
----------
      1350

-------------------------------------------------------------------------------------------------
| Id  | Operation                                    | Name          | Rows  | Bytes | Cost (%CPU)
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                             |               |     1 |    14 |  1369   (0)
|   1 |  SORT AGGREGATE                              |               |     1 |    14 |
|   2 |   TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED| PART_TAB      |  1357 | 18998 |  1369   (0)
|*  3 |    INDEX RANGE SCAN                          | IDX1_PART_TAB |  1357 |       |    12   (0)
-------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   3 - access("N1"=TO_NUMBER(:B1) AND "DAIMA"=:B2)
       filter("DAIMA"=:B2)
Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1363  consistent gets
```

```
SQL> create index idx1_part_tab on part_tab(n1,col1,daima);
Index created.
SQL> exec dbms_stats.gather_table_stats(USER,'PART_TAB',CASCADE=>TRUE);
PL/SQL procedure successfully completed.
SQL> select column_name ,num_distinct from user_tab_col_statistics where table_name='PART_TAB';

COLUMN_NAME                    NUM_DISTINCT
------------------------------ ------------
TIMESTAMP                              3250
ID                                    14321
COL1                                   3250
N1                                      190
N2                                      509
PAD                                  254272
FDMA                                      1
```

锁定统计信息后，添加了index

```
SQL> exec dbms_stats.lock_table_stats('SCOTT','PART_TAB');
SQL>  create index idx2_part_tab on part_tab(n2,col1,id,daima);
Index created.
SQL> select count(*) from part_tab where n1=:b1 and daima=:b2 order by n2;

  COUNT(*)
----------
      1350

-------------------------------------------------------------------------------------------------
| Id  | Operation                                    | Name          | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                             |               |     1 |    14 |   826   (0)|
|   1 |  SORT AGGREGATE                              |               |     1 |    14 |            |
|*  2 |   TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED| PART_TAB      |  1357 | 18998 |   826   (0)|
|*  3 |    INDEX SKIP SCAN                           | IDX2_PART_TAB |  257K|       |    26   (0)|
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter("N1"=TO_NUMBER(:B1))
   3 - access("DAIMA"=:B2)
       filter("DAIMA"=:B2)


Statistics
----------------------------------------------------------
          3  recursive calls
          0  db block gets
     256719  consistent gets
```

```
Table Stats::
  Table: PART_TAB  Alias: PART_TAB  (Using composite stats)

  #Rows: 205884  SSZ: 0  LGR: 0  #Blks:  8928  AvgRowLen:  163.00  NEB: 0  ChainCnt:  0.00  ScanRate:  0.00  SPC: 0  RFL: 0  RNF: 0  CBK: 0  CHR: 0  KQDFLG: 1
  #IMCUs: 0  IMCRowCnt: 0  IMCJournalRowCnt: 0  #IMCBlocks: 0  IMCQuotient: 0.000000
Index Stats::
  Index: IDX1_PART_TAB  Col#: 4 3 7
  LVLS: 2  #LB: 934  #DK: 75620  LB/K: 1.00  DB/K: 2.00  CLUF: 205884.00  NRW: 205884.00 SSZ: 0.00 LGR: 0.00 CBK: 0.00 GQL: 0.00 CHR: 0.00 KQDFLG: 1 BSZ: 8192
  KKEISFLG: 1
  Index: IDX2_PART_TAB  Col#: 5 3 2 7    (NOT ANALYZED)
  LVLS: 1  #LB: 25  #DK: 100  LB/K: 1.00  DB/K: 1.00  CLUF: 800.00  NRW: 2500.00 SSZ: 0.00 LGR: 0.00 CBK: 0.00 GQL: 0.00 CHR: 0.00 KQDFLG: 1 BSZ: 8192
  KKEISFLG: 0
try to generate single-table filter predicates from ORs for query block SEL$1 (#0)
finally: "PART_TAB"."N1"=:B1 AND "PART_TAB"."DAIMA"=:B2
```

使用了新的index，因为cost只有826，小于第一计划的cost

10053事件显示，系统默认给了index的统计信息赋值

Skip scan效率低下，逻辑读多了200倍

强制收集统计信息，执行计划回到了第一个

```
SQL> select index_name,num_rows,leaf_blocks,clustering_factor
  2  from user_ind_statistics where index_name='IDX2_PART_TAB';

INDEX_NAME        NUM_ROWS LEAF_BLOCKS CLUSTERING_FACTOR
--------------- ---------- ----------- -----------------
IDX2_PART_TAB

SQL> exec dbms_stats.gather_table_stats(USER,'PART_TAB',FORCE=>TRUE);

PL/SQL procedure successfully completed.

SQL> select index_name,num_rows,leaf_blocks,clustering_factor
  2  from user_ind_statistics where index_name='IDX2_PART_TAB';

INDEX_NAME        NUM_ROWS LEAF_BLOCKS CLUSTERING_FACTOR
--------------- ---------- ----------- -----------------
IDX2_PART_TAB       205884        1075            203495
```

新创建index没有统计信息

强制采集统计信息，新创建的index有了统计信息，cost评估正常了，变成了204K，所以还是选择最开始的执行计划

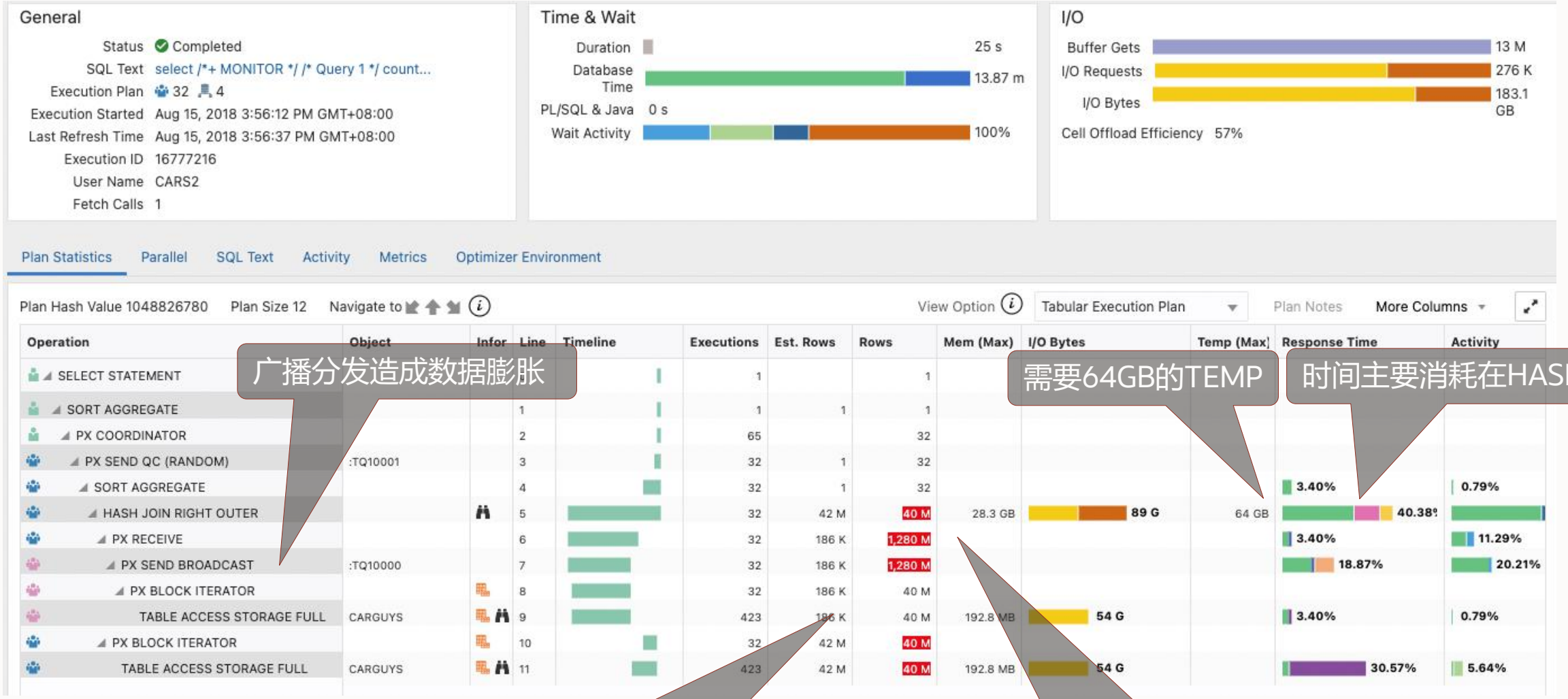
案例2

```
select /*+ MONITOR parallel(32)  */
       count(p1.text) "Ferraris",
       count(p2.text) "Ferrari 458s",
       sum(decode(p2.text, null, 1, 0) ) "Other Ferraris"
from    (select owner_id,
                'Ferraris' as text
             from cars2.carguys
             where  make = 'Ferrari') p1
left outer join
       (select owner_id,
                'Ferrari 458s' as text
        from    cars2.carguys
        where   country = 'Italy'
        and     make = 'Ferrari'
        and     model = '458 Italia') p2
on      p1.owner_id = p2.owner_id
```

在一体机上面并行查询需要25秒完成

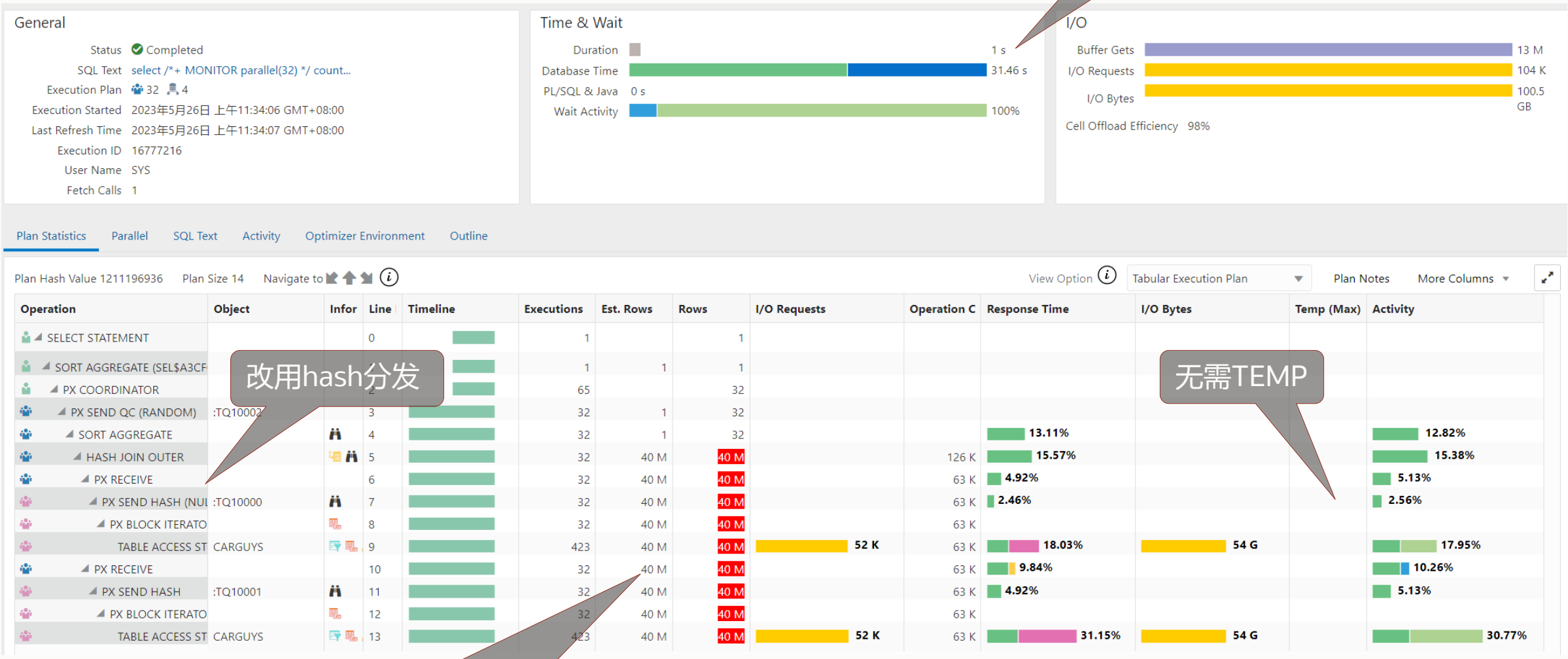
Carguys表有10亿条记录，表大小为55G，表统计信息是最新的。

基于SQL Monitor报告分析问题



解决办法：创建扩展统计信息，并收集直方图信息

一秒完成查询



改用hash分发

无需TEMP

估计和实际都是4千万，准确

基于 Oracle 数据库
免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：

400-699-8888

* 活动最终解释权归甲骨文公司所有

Our mission is to help people see data in new ways, discover insights,
unlock endless possibilities.

数据库和云讲座群

20-21

甲骨文云技术公众号

技术专家1V1深入交流