云和恩墨
ENMOTECH

ACOUG
All China Oracle User Group
中国 Oracle 用户组

IT大咖说
知识分享平台

数据库性能优化
之
降低I/O — INDEX的使用
(2017-3)

YUNHE ENMO (BEIJING) TECHNOLOGY CO.,LTD

云和恩墨　成就所托

郭成日 |

主要内容

1. 索引的概念及应用
2. 复合索引的应该
3. 索引肯定快吗？
4. 复合索引的策略

影响数据库性能的主要因素

◎ 影响数据库性能的主要因素
- ▷DB Call
- ▷Hard parse + Soft parse
- ▷Wait event
- ▷I/O
- ▷不合理的设计与开发

◎ I/O
- 引起数据库性能问题有很多因素存在。
- 各个项目中存在最多的问题就是 I/O。
- 如果能降低I/O，就能提高 SQL的处理速度。
- 提高核心 SQL的处理速度，会带来整个系统性能的提升。
- Oracle的 I/O是以 BLOCK为单位。
- SQL优化的核心就是用最少的I/O处理想要的数据。
- 与I/O相关的内容就是INDEX。

INDEX 概念及应用 – 1

Index是什么?

- 是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问表中的特定信息。
- Oracle 默认的是 B*Tree 索引。(BITMAP索引一般只用在 DW，局限性)
- 使用索引的话，可以用最少的I/O找到想要的数据。

创建测试表

```
--TEST TABLE CREATE
CREATE TABLE  TEST_SALE
(
  SALE_SEQ NUMBER(18) NOT NULL  -- PK
  ,SALE_YMD VARCHAR2(8)         -- 销售日期
  ,ITEM_ID VARCHAR2(40)         -- 型号
  ,SHOP_ID VARCHAR2(40)         -- 商场 ID
  ,SALE_TP VARCHAR2(40)         -- 销售类型 : B2C, B2B
  ,CALC_TP VARCHAR2(40)         -- 结算类型 : CASH, CARD, WECHAT
  ,SALE_QTY NUMBER(18)          -- 销售数量
  ,SALE_PRC NUMBER(18,2)        -- 销售金额
  ,REGI_DT DATE                 -- 注册日期
)

CREATE UNIQUE INDEX PK_TEST_SALE ON TEST_SALE(SALE_SEQ) NOLOGGING;
ALTER TABLE TEST_SALE ADD CONSTRAINT PK_TEST_SALE PRIMARY KEY(SALE_SEQ) USING INDEX;
```

Tip.1
生成1年的日期数据，格式为 YYYYMMDD

Tip.2
销售类型别生成数据，2个B2C，1个B2B

Tip.3
使用笛卡尔积生成大量数据

生成测试数据 (4,380,000条)

```
INSERT INTO TEST_SALE
       (SALE_SEQ ,SALE_YMD ,ITEM_ID         ,SHOP_ID ,SALE_TP ,CALC_TP         ,SALE_QTY ,SALE_PRC ,REGI_DT)
SELECT  ROWNUM SALE_SEQ ,T_YMD.YMD SALE_YMD ,T_ITEM.ITEM_ID
       ,T_SHOP.SHOP_ID ,T_SALE_TP.SALE_TP, T_CALC_TP.CALC_TP
       ,1 SALE_QTY ,1000 SALE_PRC ,TO_DATE(T_YMD.YMD,'YYYYMMDD') REGI_DT
FROM   (--TIP1.生成1年的日期数据
        SELECT  TO_CHAR((ROWNUM-1) + TO_DATE('20170101','YYYYMMDD'),'YYYYMMDD') YMD
        FROM    DUAL A
        CONNECT BY ROWNUM <= 365
       ) T_YMD
       ,--B2C是 2条, B2B是 1条, 销售类型别生成不同的数据.
        SELECT  'B2C' SALE_TP FROM DUAL CONNECT BY ROWNUM <= 2
        UNION ALL
        SELECT  'B2B' SALE_TP FROM DUAL CONNECT BY ROWNUM <= 1
       ) T_SALE_TP
       ,(SELECT  'CASH' CALC_TP FROM DUAL CONNECT BY ROWNUM <= 2
         UNION ALL
         SELECT  'CARD' CALC_TP FROM DUAL CONNECT BY ROWNUM <= 1
         UNION ALL
         SELECT  'WECHAT' CALC_TP FROM DUAL CONNECT BY ROWNUM <= 1
       ) T_CALC_TP
       ,(SELECT RPAD('ITEM_',8,CHR(ASCII('A')+(ROWNUM+1))) ITEM_ID
         FROM  DUAL A
         CONNECT BY ROWNUM <= 10
       ) T_ITEM
       ,(
         SELECT 'SHOP_'||TO_CHAR(ROWNUM) SHOP_ID
         FROM  DUAL A
         CONNECT BY ROWNUM <= 100
       ) T_SHOP
       ;

COMMIT;

EXEC DBMS_STATS.GATHER_TABLE_STATS('GCR','TEST_SALE'); -- 收集统计信息
```

Tip.1

Tip.2

Tip.3

INDEX 概念及应用- 2

TEST-1 : NO INDEX

Tip.4

```
--TEST-1
ALTER SYSTEM FLUSH BUFFER_CACHE;  --禁止在生产库上执行
ALTER SYSTEM FLUSH SHARED_POOL;   --禁止在生产库上执行
```

Tip.5

```
SELECT  /*+ GATHER_PLAN_STATISTICS */
        SUM(T1.SALE_QTY)
FROM    TEST_SALE T1
WHERE   T1.SALE_YMD BETWEEN '20170301' AND '20170310'
;
```

Tip.6

```
SELECT  SQL_ID, SQL_TEXT
FROM    V$SQL WHERE SQL_TEXT LIKE '%GATHER_PLAN_STATISTICS%'
ORDER BY LAST_ACTIVE_TIME DESC;

SELECT  *
FROM    TABLE(DBMS_XPLAN.DISPLAY_CURSOR('axdbzvp02cjbp',null,'ADVANCED ALLSTATS LAST'));
```

TEST-2 : USE INDEX

```
CREATE INDEX  X_TEST_SALE_1 ON TEST_SALE(SALE_YMD) NOLOGGING;


ALTER SYSTEM FLUSH BUFFER_CACHE;  --禁止在生产库上执行
ALTER SYSTEM FLUSH SHARED_POOL;   --禁止在生产库上执行

SELECT  /*+ GATHER_PLAN_STATISTICS */
        SUM(T1.SALE_QTY)
FROM    TEST_SALE T1
WHERE   T1.SALE_YMD BETWEEN '20170301' AND '20170310'
;

SELECT  SQL_ID, SQL_TEXT
FROM    V$SQL WHERE SQL_TEXT LIKE '%GATHER_PLAN_STATISTICS%'
ORDER BY LAST_ACTIVE_TIME DESC;

SELECT  *
FROM    TABLE(DBMS_XPLAN.DISPLAY_CURSOR('axdbzvp02cjbp',null,'ADVANCED ALLSTATS LAST'));
```

TEST-1

```
-------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |      1 |        |       |  9954 (100)|          |      1 |00:00:00.63 |   36115 |  36111 |
|   1 |  SORT AGGREGATE    |           |      1 |      1 |    12 |            |          |      1 |00:00:00.63 |   36115 |  36111 |
|*  2 |   TABLE ACCESS FULL| TEST_SALE |      1 |  58885 |   690K|  9954   (1)| 00:02:00 |    120K|00:00:00.13 |   36115 |  36111 |
-------------------------------------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter(("T1"."SALE_YMD"<='20170310' AND "T1"."SALE_YMD">='20170301'))
```

36,115 VS. 1,322 reads
= 00.63sec VS. 00.18 sec

TEST-2

```
-------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |      1 |        |       |   658 (100)|          |      1 |00:00:00.18 |    1322 |   1313 |
|   1 |  SORT AGGREGATE              |               |      1 |      1 |    12 |            |          |      1 |00:00:00.18 |    1322 |   1313 |
|   2 |   TABLE ACCESS BY INDEX ROWID| TEST_SALE     |      1 |  58885 |   690K|   658   (1)| 00:00:08 |    120K|00:00:00.32 |    1322 |   1313 |
|*  3 |    INDEX RANGE SCAN          | X_TEST_SALE_1 |      1 |  58885 |       |   167   (0)| 00:00:03 |    120K|00:00:00.11 |     337 |    337 |
-------------------------------------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   3 - access("T1"."SALE_YMD">='20170301' AND "T1"."SALE_YMD"<='20170310')
```

Tip.4 清除BUFFER与SHARED POOL里的内容，禁止在生产库执行

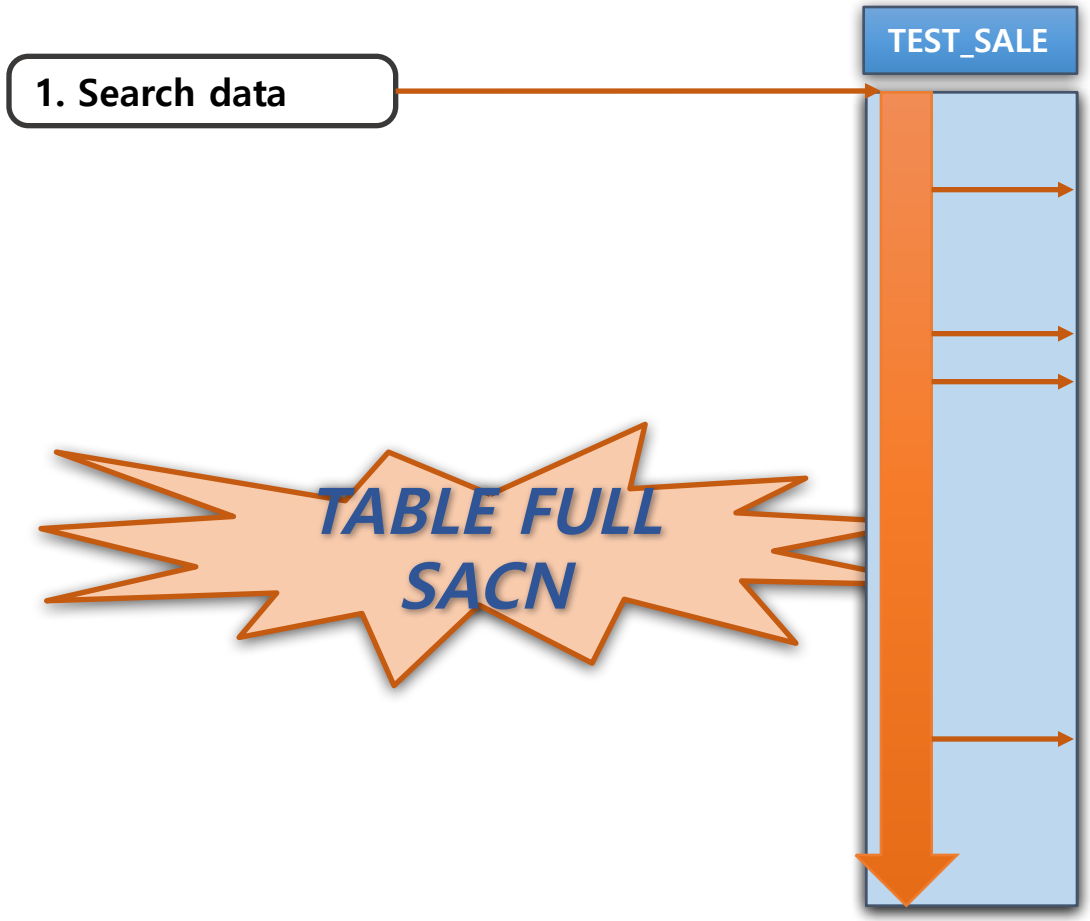
Tip.5 为抓取实际执行计划

Tip.6 查看实际执行计划内容

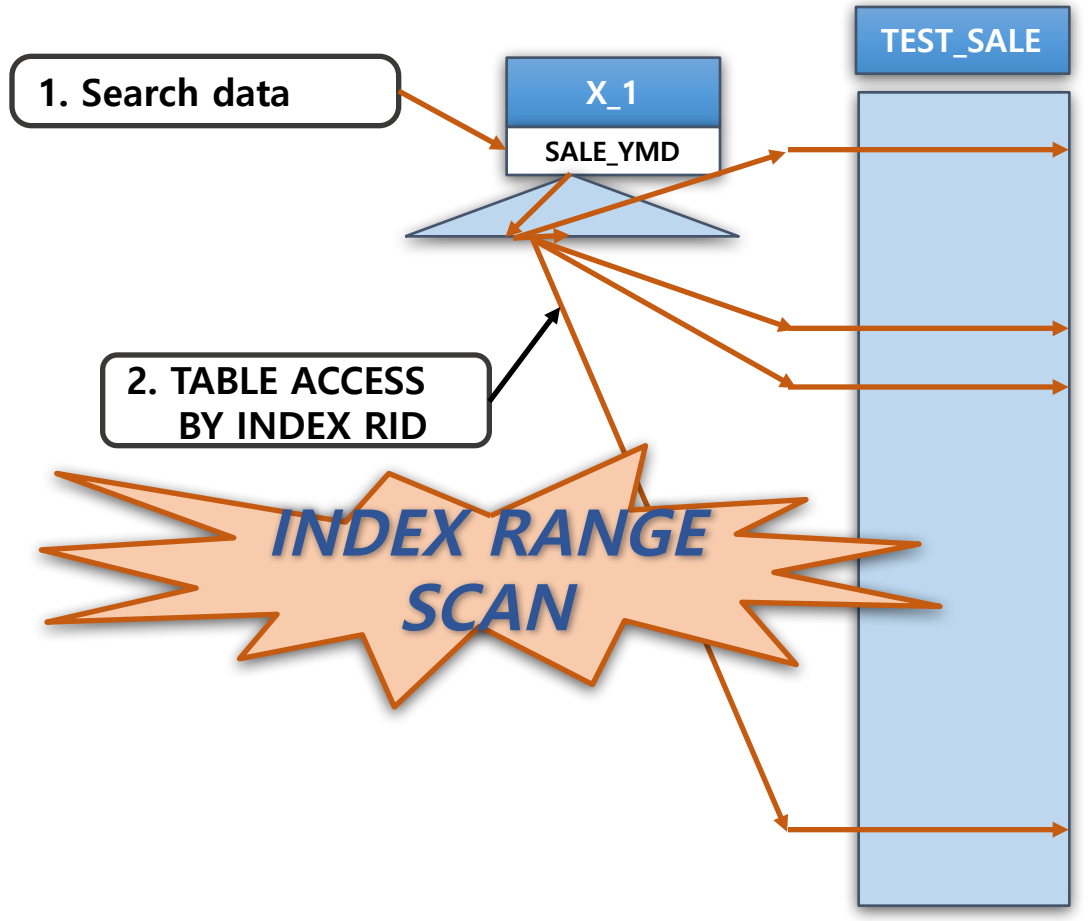
INDEX 概念及应用- 3


```
SELECT   SUM(T1.SALE_QTY)
FROM     TEST_SALE T1
WHERE    T1.SALE_YMD BETWEEN '20170301' AND '20170310'
```

TEST-1 : NO INDEX



TEST-2 : USE INDEX



复合索引的应用- 1

TEST-1 : SINGLE COLUMN INDEX

```
--TEST-1, SINGLE INDEX
   ALTER SYSTEM FLUSH BUFFER_CACHE;  --禁止在生产库上执行
   ALTER SYSTEM FLUSH SHARED_POOL;   --禁止在生产库上执行

   SELECT  /*+ GATHER_PLAN_STATISTICS */
          T1.SALE_YMD
          ,SUM(T1.SALE_QTY) SALE_QTY
   FROM    TEST_SALE T1
   WHERE   T1.SHOP_ID = 'SHOP_39'
   AND     T1.SALE_YMD BETWEEN '20170301' AND '20170331'
   AND     T1.CALC_TP = 'CASH'
   GROUP BY T1.SALE_YMD
   ;


   SELECT  SQL_ID, SQL_TEXT
   FROM    V$SQL WHERE SQL_TEXT LIKE '%GATHER_PLAN_STATISTICS%'
   ORDER BY LAST_ACTIVE_TIME DESC;
```

TEST-1

```
SPLAY_CURSOR('drnqnxghjaasa',null,'ADVANCED ALLSTATS LAST'));
```

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1			1518 (100)		30	00:00:00.57	3969	3994
1	SORT GROUP BY NOSORT		1	11	275	1518 (1)	00:00:19	30	00:00:00.57	3969	3994
* 2	TABLE ACCESS BY INDEX ROWID	TEST_SALE	1	682	17050	1518 (1)	00:00:19	1800	00:00:01.07	3969	3994
* 3	INDEX RANGE SCAN	X_TEST_SALE_1	1	136K		383 (1)	00:00:05	360K	00:00:00.17	1005	1007

```
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter(("T1"."SHOP_ID"='SHOP_39' AND "T1"."CALC_TP"='CASH'))
   3 - access("T1"."SALE_YMD">='20170301' AND "T1"."SALE_YMD"<='20170331')
```

TEST-2 : MULTI COLUMN INDEX

```
CREATE INDEX  X_TEST_SALE_2 ON TEST_SALE(CALC_TP, SHOP_ID, SALE_YMD) NOLOGGING;

ALTER SYSTEM FLUSH BUFFER_CACHE;  --禁止在生产库上执行
ALTER SYSTEM FLUSH SHARED_POOL;   --禁止在生产库上执行

SELECT  /*+ GATHER_PLAN_STATISTICS */
       T1.SALE_YMD
       ,SUM(T1.SALE_QTY) SALE_QTY
FROM    TEST_SALE T1
WHERE   T1.SHOP_ID = 'SHOP_39'
AND     T1.SALE_YMD BETWEEN '20170301' AND '20170331'
AND     T1.CALC_TP = 'CASH'
GROUP BY T1.SALE_YMD
;


SELECT  SQL_ID, SQL_TEXT
FROM    V$SQL WHERE SQL_TEXT LIKE '%GATHER_PLAN_STATISTICS%'
ORDER BY LAST_ACTIVE_TIME DESC;

SELECT  *
```

3,969 VS. 375 reads
= 00.57 sec VS. 00.09 sec

TEST-2

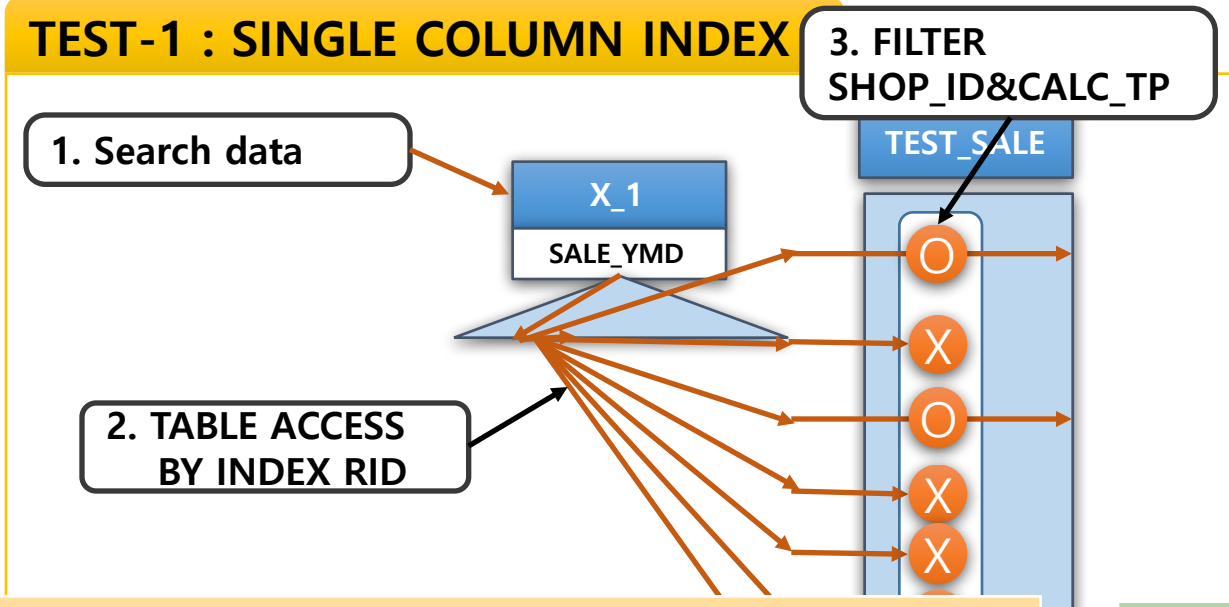
Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	A-Rows	A-Time	Buffers	Reads
0	SELECT STATEMENT		1			129 (100)		30	00:00:00.09	375	1995
1	SORT GROUP BY NOSORT		1	11	275	129 (0)	00:00:02	30	00:00:00.09	375	1995
2	TABLE ACCESS BY INDEX ROWID	TEST_SALE	1	682	17050	129 (0)	00:00:02	1800	00:00:00.29	375	1995
* 3	INDEX RANGE SCAN	X_TEST_SALE_2	1	682		6 (0)	00:00:01	1800	00:00:00.01	12	19

```
3 - access("T1"."CALC_TP"='CASH' AND "T1"."SHOP_ID"='SHOP_39' AND "T1"."SALE_YMD">='20170301' AND "T1"."SALE_YMD"<='20170331')
```

复合索引的应用- 2

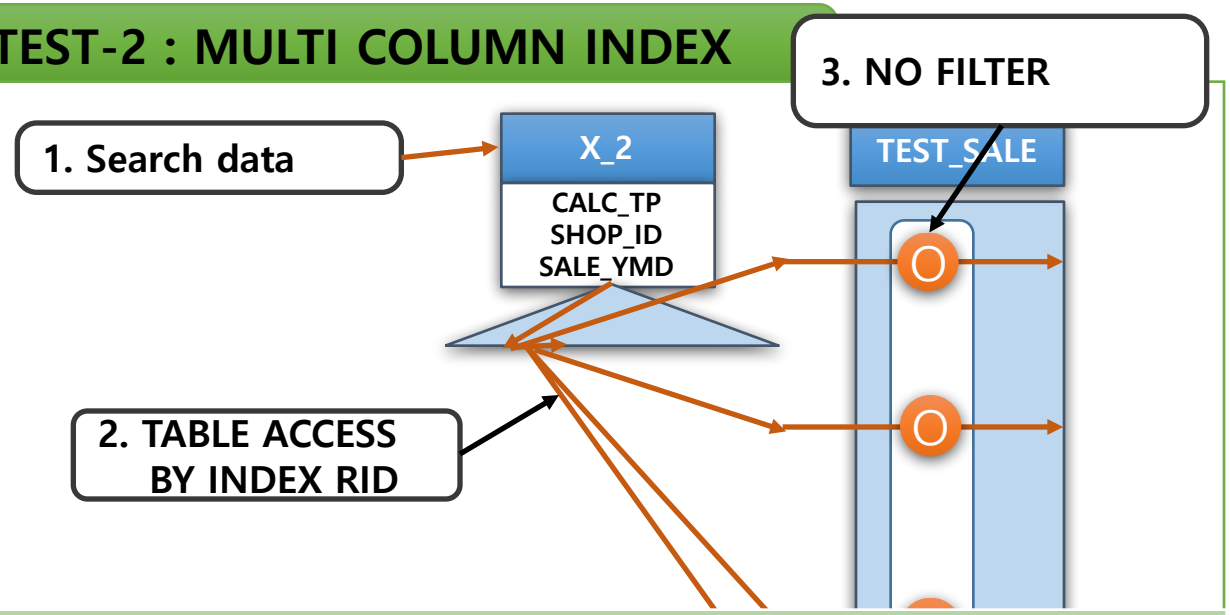
```
SELECT  T1.SALE_YMD
        ,SUM(T1.SALE_QTY) SALE_QTY
FROM    TEST_SALE T1
WHERE   T1.SHOP_ID = 'SHOP_39'
AND     T1.SALE_YMD BETWEEN '20170301' AND '20170331'
AND     T1.CALC_TP = 'CASH'
GROUP BY T1.SALE_YMD;
```

TEST-1 : SINGLE COLUMN INDEX




```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |     30 |00:00:00.57 |    3969 |   3994 |
|   1 |  SORT GROUP BY NOSORT        |               |     30 |00:00:00.57 |    3969 |   3994 |
|*  2 |   TABLE ACCESS BY INDEX ROWID| TEST_SALE     |   1800 |00:00:01.07 |    3969 |   3994 |
|*  3 |    INDEX RANGE SCAN          | X_TEST_SALE_1 |   360K |00:00:00.17 |    1005 |   1007 |
-------------------------------------------------------------------------------------------------------
   2 - filter(( "T1"."SHOP_ID"='SHOP_39' AND "T1"."CALC_TP"='CASH' )) => 非效率
   3 - access("T1"."SALE_YMD">='20170301' AND "T1"."SALE_YMD"<='20170331')
```

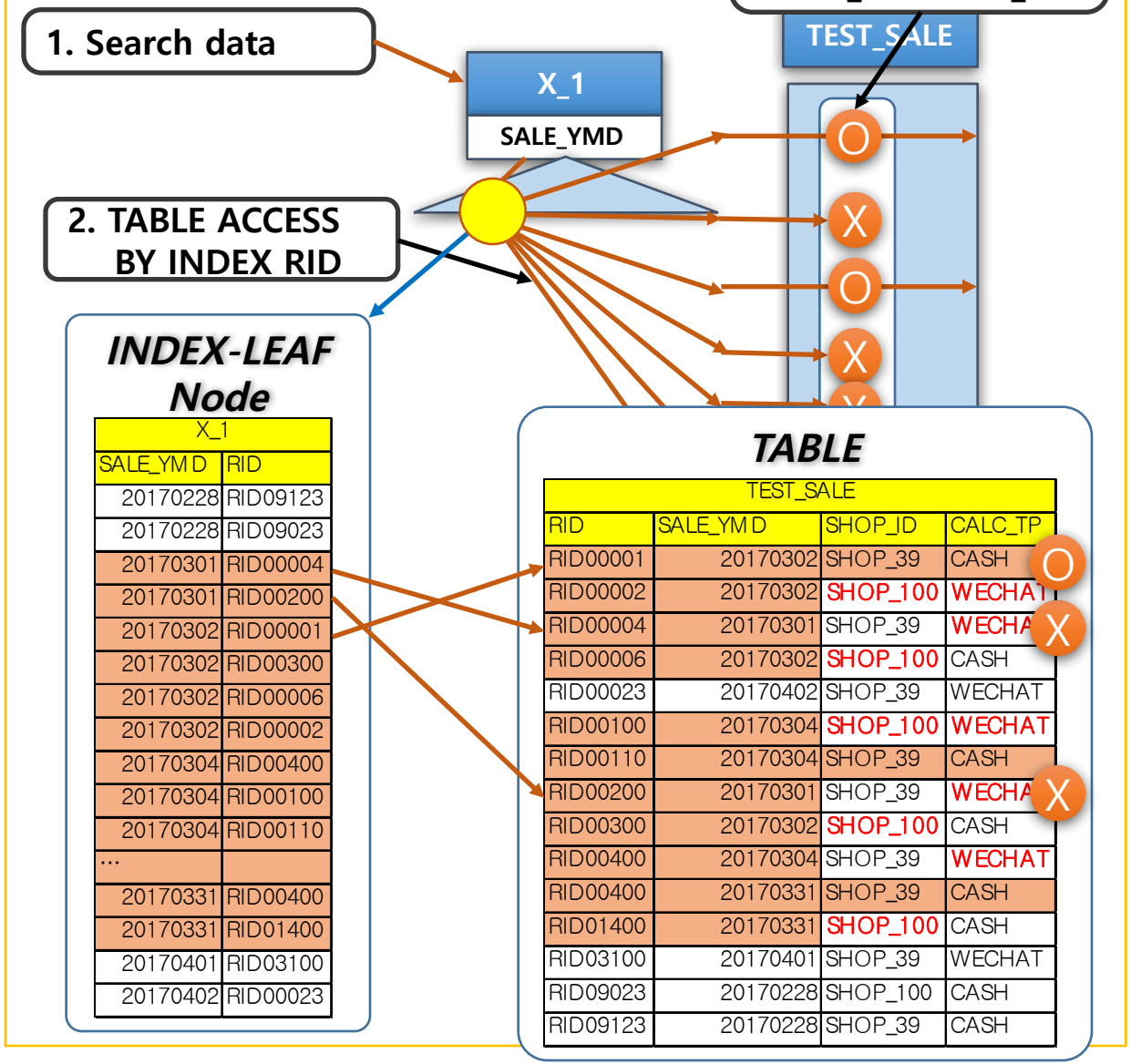
TEST-2 : MULTI COLUMN INDEX




```
-------------------------------------------------------------------------------------
| Id  | Operation                    | Name          | Buffers | Reads  |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |               |     375 |   1995 |
|   1 |  SORT GROUP BY NOSORT        |               |     375 |   1995 |
|   2 |   TABLE ACCESS BY INDEX ROWID| TEST_SALE     |     375 |   1995 |
|*  3 |    INDEX RANGE SCAN          | X_TEST_SALE_2 |      12 |     19 |
-------------------------------------------------------------------------------------
3 - access("T1"."CALC_TP"='CASH' AND "T1"."SHOP_ID"='SHOP_39' AND "T1"."SALE_YMD">='20170301' AND "T1"."SALE_YMD"<='20170331')
```

复合索引的应用- 3

TEST-1 : SINGLE COLUMN INDEX



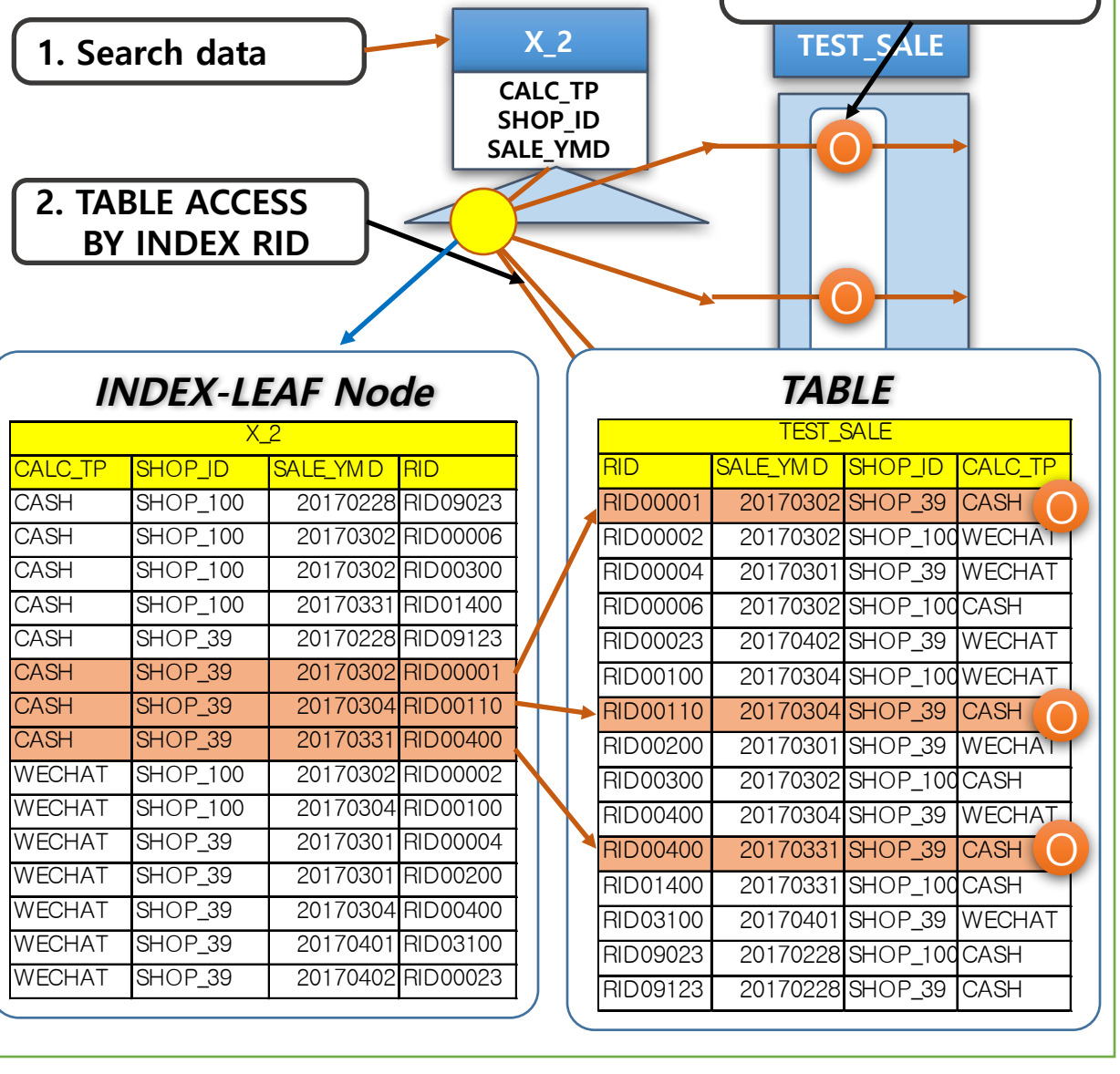
X_1

SALE_YMD	RID
20170228	RID09123
20170228	RID09023
20170301	RID00004
20170301	RID00200
20170302	RID00001
20170302	RID00300
20170302	RID00006
20170302	RID00002
20170304	RID00400
20170304	RID00100
20170304	RID00110
…	
20170331	RID00400
20170331	RID01400
20170401	RID03100
20170402	RID00023

TEST_SALE

RID	SALE_YMD	SHOP_ID	CALC_TP
RID00001	20170302	SHOP_39	CASH
RID00002	20170302	SHOP_100	WECHAT
RID00004	20170301	SHOP_39	WECHA
RID00006	20170302	SHOP_100	CASH
RID00023	20170402	SHOP_39	WECHAT
RID00100	20170304	SHOP_100	WECHAT
RID00110	20170304	SHOP_39	CASH
RID00200	20170301	SHOP_39	WECHA
RID00300	20170302	SHOP_100	CASH
RID00400	20170304	SHOP_39	WECHAT
RID00400	20170331	SHOP_39	CASH
RID01400	20170331	SHOP_100	CASH
RID03100	20170401	SHOP_39	WECHAT
RID09023	20170228	SHOP_100	CASH
RID09123	20170228	SHOP_39	CASH

TEST-2 : MULTI COLUMN INDEX



X_2

CALC_TP	SHOP_ID	SALE_YMD	RID
CASH	SHOP_100	20170228	RID09023
CASH	SHOP_100	20170302	RID00006
CASH	SHOP_100	20170302	RID00300
CASH	SHOP_100	20170331	RID01400
CASH	SHOP_39	20170228	RID09123
CASH	SHOP_39	20170302	RID00001
CASH	SHOP_39	20170304	RID00110
CASH	SHOP_39	20170331	RID00400
WECHAT	SHOP_100	20170302	RID00002
WECHAT	SHOP_100	20170304	RID00100
WECHAT	SHOP_39	20170301	RID00004
WECHAT	SHOP_39	20170301	RID00200
WECHAT	SHOP_39	20170304	RID00400
WECHAT	SHOP_39	20170401	RID03100
WECHAT	SHOP_39	20170402	RID00023

TEST_SALE

RID	SALE_YMD	SHOP_ID	CALC_TP
RID00001	20170302	SHOP_39	CASH
RID00002	20170302	SHOP_100	WECHAT
RID00004	20170301	SHOP_39	WECHAT
RID00006	20170302	SHOP_100	CASH
RID00023	20170402	SHOP_39	WECHAT
RID00100	20170304	SHOP_100	WECHAT
RID00110	20170304	SHOP_39	CASH
RID00200	20170301	SHOP_39	WECHAT
RID00300	20170302	SHOP_100	CASH
RID00400	20170304	SHOP_39	WECHAT
RID00400	20170331	SHOP_39	CASH
RID01400	20170331	SHOP_100	CASH
RID03100	20170401	SHOP_39	WECHAT
RID09023	20170228	SHOP_100	CASH
RID09123	20170228	SHOP_39	CASH

索引肯定快么？ - 1

TEST-1 : NO INDEX

```
ALTER SYSTEM FLUSH BUFFER_CACHE;  --禁止在生产库上执行
ALTER SYSTEM FLUSH SHARED_POOL;   --禁止在生产库上执行

SELECT  /*+ GATHER_PLAN_STATISTICS */
       T1.SALE_TP
      ,SUM(T1.SALE_QTY) SALE_QTY
FROM    TEST_SALE T1
WHERE   T1.CALC_TP = 'CASH'
GROUP BY T1.SALE_TP;



SELECT  SQL_ID, SQL_TEXT
FROM    V$SQL WHERE SQL_TEXT LIKE '%GATHER_PLAN_STATISTICS%'
ORDER BY LAST_ACTIVE_TIME DESC;

SELECT  *
FROM    TABLE(DBMS_XPLAN.DISPLAY_CURSOR('cszrp84t99t7w',null,'ADVANCED ALLSTATS LAST'));
```

TEST-1

```
-------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |      1 |        |       | 10023 (100)|          |      2 |00:00:00.96 |   36115 |  36111 |
|   1 |  HASH GROUP BY     |           |      1 |      2 |    24 | 10023   (2)| 00:02:01 |      2 |00:00:00.96 |   36115 |  36111 |
|*  2 |   TABLE ACCESS FULL| TEST_SALE |      1 |   2190K|    25M|  9964   (1)| 00:02:00 |   2190K|00:00:00.53 |   36115 |  36111 |
-------------------------------------------------------------------------------------------------------------------------------------------
```

MULTI BLOCK IO ????

TEST-2 : MULTI COLUMN INDEX

```
--TEST - 2 MAKE INDEX
CREATE INDEX  X_TEST_3 ON TEST_SALE(CALC_TP) NOLOGGING;

ALTER SYSTEM FLUSH BUFFER_CACHE;  --禁止在生产库上执行
ALTER SYSTEM FLUSH SHARED_POOL;   --禁止在生产库上执行


--USE HINT
SELECT  /*+ GATHER_PLAN_STATISTICS INDEX(T1 X_TEST_3) */
       T1.SALE_TP
      ,SUM(T1.SALE_QTY) SALE_QTY
FROM    TEST_SALE T1
WHERE   T1.CALC_TP = 'CASH'
GROUP BY T1.SALE_TP;

SELECT  SQL_ID, SQL_TEXT
FROM    V$SQL WHERE SQL_TEXT LIKE '%GATHER_PLAN_STATISTICS%'
ORDER BY LAST_ACTIVE_TIME DESC;

SELECT  *
FROM    TABLE(DBMS_XPLAN.DISPLAY_CURSOR('f5z6wq75ffdn5',null,'ADVANCED ALLSTATS LAST'));
```

36,115 VS. 40,921 reads
= 00.96 sec VS. 4.08 sec

TEST-2

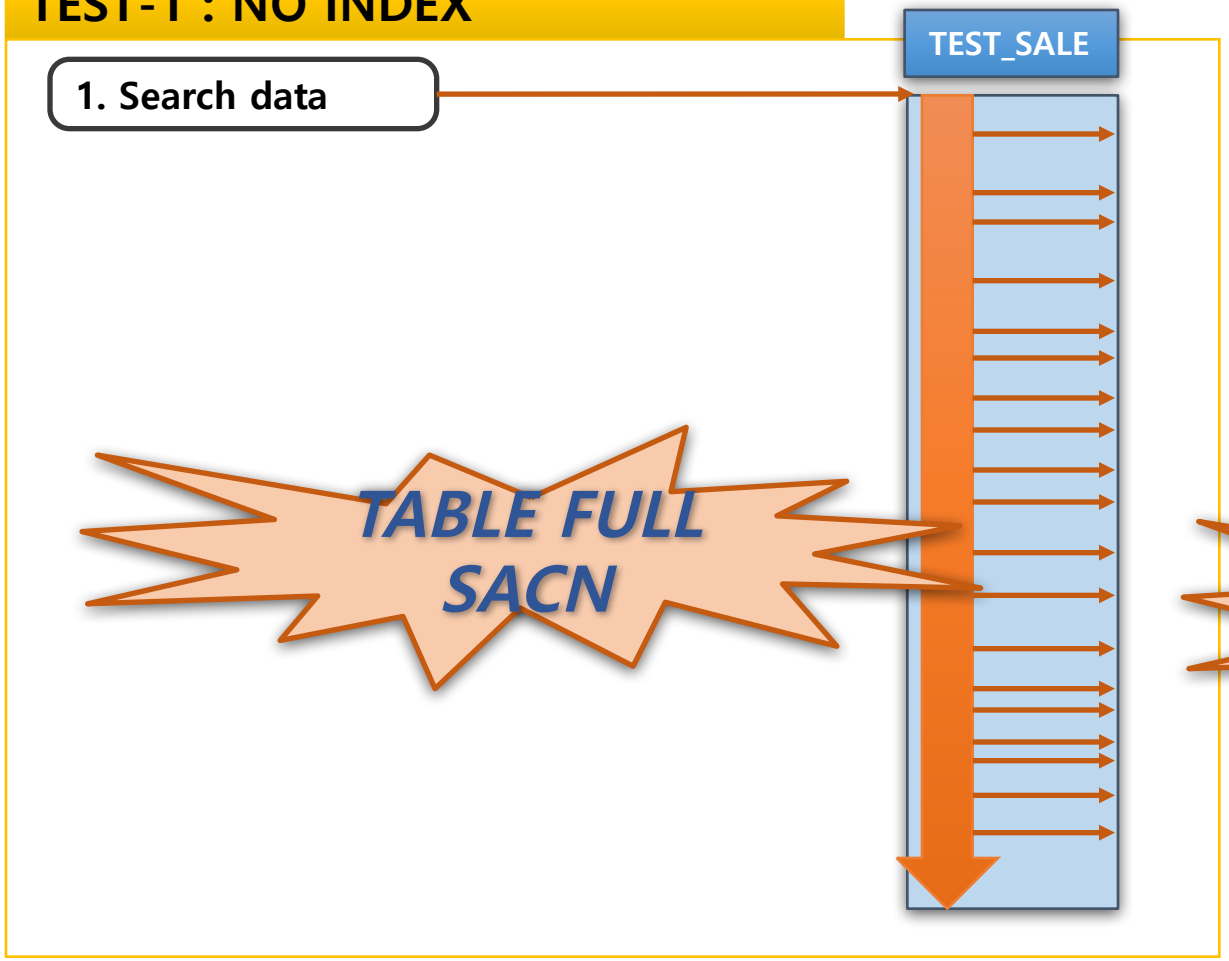
```
-------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Starts | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | A-Rows |   A-Time   | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |      1 |        |       | 41015 (100)|          |      2 |00:00:04.08 |   40921 |  40921 |
|   1 |  HASH GROUP BY               |           |      1 |      2 |    24 | 41015   (1)| 00:08:13 |      2 |00:00:04.08 |   40921 |  40921 |
|   2 |   TABLE ACCESS BY INDEX ROWID| TEST_SALE |      1 |   2190K|    25M| 40957   (1)| 00:08:12 |   2190K|00:00:03.72 |   40921 |  40921 |
|*  3 |    INDEX RANGE SCAN          | X_TEST_3  |      1 |   2190K|       |  4893   (1)| 00:00:59 |   2190K|00:00:01.06 |    4881 |   4881 |
-------------------------------------------------------------------------------------------------------------------------------------------

   3 - access("T1"."CALC_TP"='CASH')
```

RADNOM IO

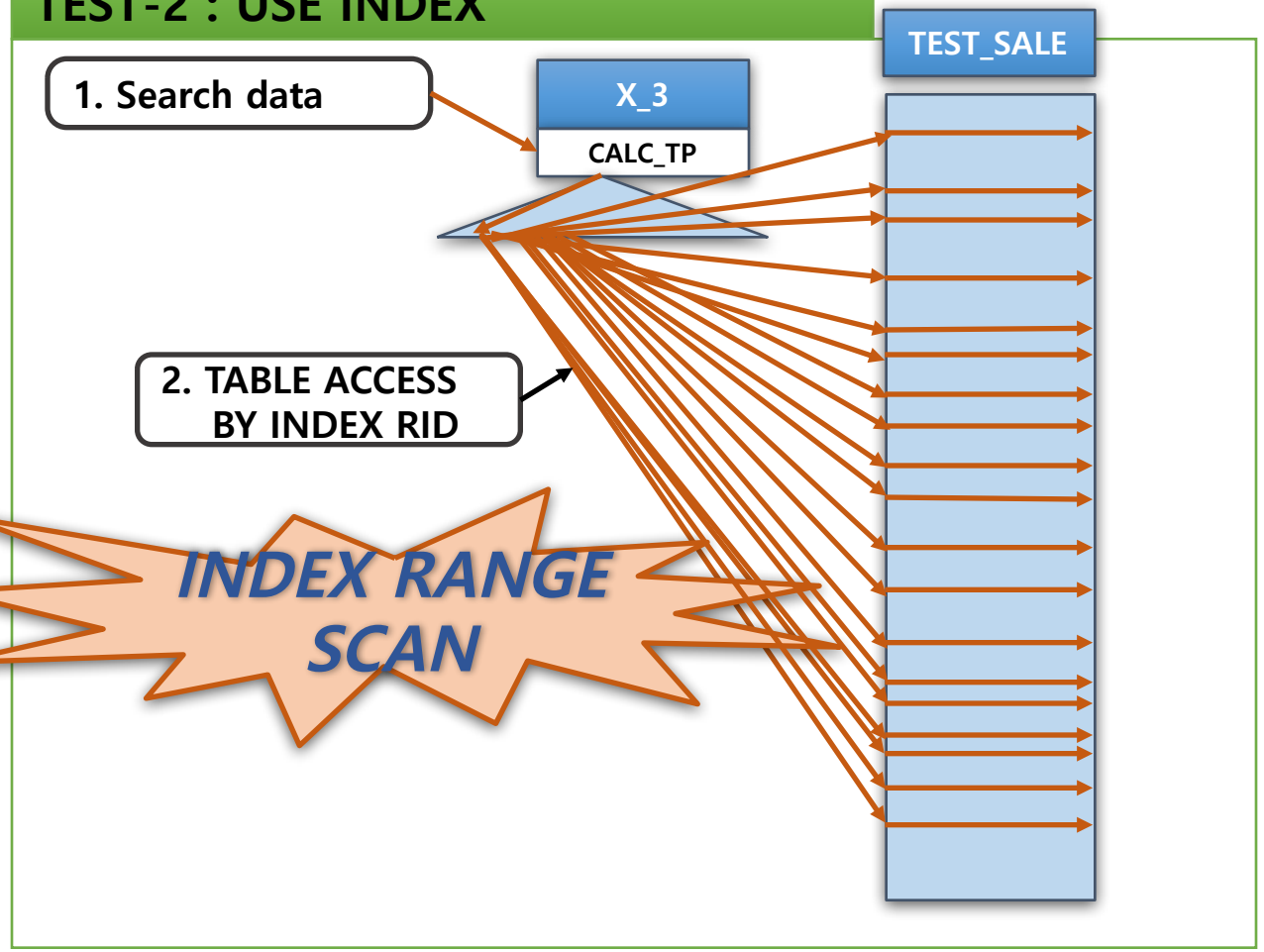
索引肯定快么？ – 2


```
SELECT   T1.SALE_TP
        ,SUM(T1.SALE_QTY) SALE_QTY
FROM    TEST_SALE T1
WHERE   T1.CALC_TP = 'CASH'
GROUP BY T1.SALE_TP;
```

TEST-1 : NO INDEX



TEST-2 : USE INDEX



复合索引的策略 – 1

SQL-1 : 10 exec per a day

```
SELECT   T1.SALE_YMD
        ,COUNT(*) SALE_CNT
FROM     TEST_SALE T1
WHERE   T1.SALE_YMD BETWEEN '20170301' AND '20170310'
GROUP BY T1.SALE_YMD;
```

SQL-2 : 1000 exec per a day

```
SELECT   T1.SALE_YMD
        ,COUNT(*) SALE_CNT
FROM     TEST_SALE T1
WHERE    T1.SHOP_ID = 'SHOP_23'
AND      T1.SALE_YMD BETWEEN '20170301' AND '20170310'
GROUP BY T1.SALE_YMD;
```

SQL-3 : 100 exec per a day

```
SELECT   T1.CALC_TP
        ,SUM(T1.SALE_QTY) SALE_QTY
FROM     TEST_SALE T1
WHERE    T1.SHOP_ID = 'SHOP_23'
AND      T1.SALE_YMD BETWEEN '20170301' AND '20170310'
AND      T1.SALE_TP = 'B2B'
GROUP BY T1.CALC_TP;
```

SQL-4 1000 exec per a day

```
SELECT   T1.ITEM_ID
        ,SUM(T1.SALE_QTY) SALE_QTY
FROM     TEST_SALE T1
WHERE    T1.SHOP_ID = 'SHOP_23'
AND      T1.SALE_YMD = '20170310'
GROUP BY T1.ITEM_ID;
```

收集表使用的所有 SQL，制作如下表格进行分析

TABLE	Program Name	Program Type	SQL Type	WHERE	重要度	Decide Index
TEST_SALE	SQL-1	XML	SELECT	SALE_YMD(BW)	低	SALE_YMD
TEST_SALE	SQL-2	Procedure	SELECT	SHOP_ID(=), SALE_YMD(BW)	高	SHOP_ID, SALE_YMD
TEST_SALE	SQL-3	Procedure	SELECT	SHOP_ID(=), SALE_TP(=), SALE_YMD(BW)	中	SHOP_ID, SALE_TP, SALE_YMD
TEST_SALE	SQL-4	XML	SELECT	SHOP_ID(=), SALE_YMD(=)	高	SHOP_ID, SALE_YMD/SALE_YMD, SHOP_ID

复合索引的策略 – 2

TABLE	Program Name	Program Type	SQL Type	WHERE	重要度	Decide Index
TEST_SALE	SQL-1	XML	SELECT	SALE_YMD(BW)	低	SALE_YMD
TEST_SALE	SQL-2	Procedure	SELECT	SHOP_ID(=), SALE_YMD(BW)	高	SHOP_ID, SALE_YMD
TEST_SALE	SQL-3	Procedure	SELECT	SHOP_ID(=), SALE_TP(=), SALE_YMD(BW)	中	SHOP_ID, SALE_TP, SALE_YMD
TEST_SALE	SQL-4	XML	SELECT	SHOP_ID(=), SALE_YMD(=)	高	SHOP_ID, SALE_YMD/SALE_YMD, SHOP_ID

INDEX GROUP - 1

X_1	X_2	X_3	X_4
SALE_YMD	SHOP_ID SALE_YMD	SHOP_ID SALE_TP SALE_YMD	SALE_YMD SHOP_ID

对各个 SQL 生成最理性的 INDEX 。

INDEX GROUP - 2

X_1	X_2	X_3	X_4
SALE_YMD	SHOP_ID SALE_YMD	SHOP_ID SALE_TP SALE_YMD	SALE_YMD SHOP_ID

SQL-4可以被 X_2代替使用，这时X_4去掉。
或者，反过来X_4 代替 X_2使用也可以。
但是，SQL-2 为点与线段的条件组合，如使用 X_4 效率不高。

INDEX GROUP - 3

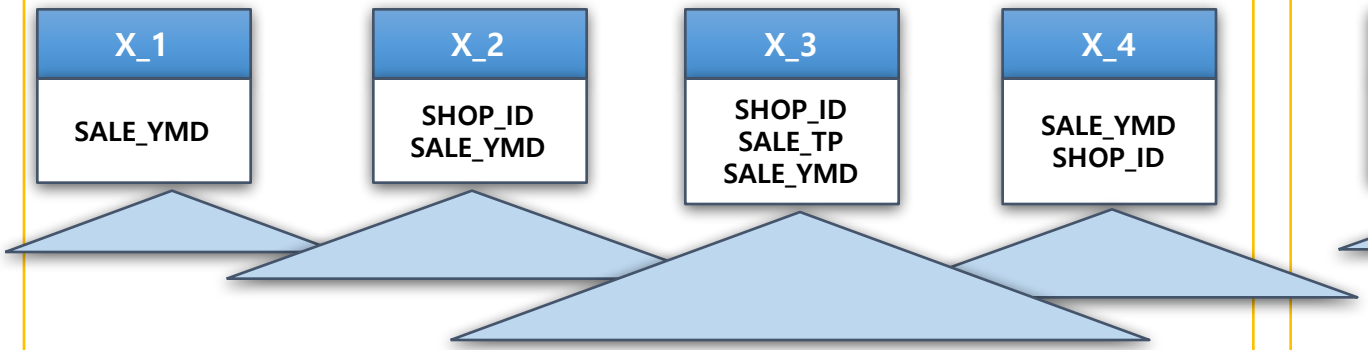
X_1	X_2	X_3	X_4
SALE_YMD SHOP_ID	SHOP_ID SALE_YMD	SHOP_ID SALE_TP SALE_YMD	SALE_YMD SHOP_ID

索引多的话会对DML操作有负担，所以最终合并为2个索引。
但是，其中 SQL-2 的 SQL写法变换为右边的写法。

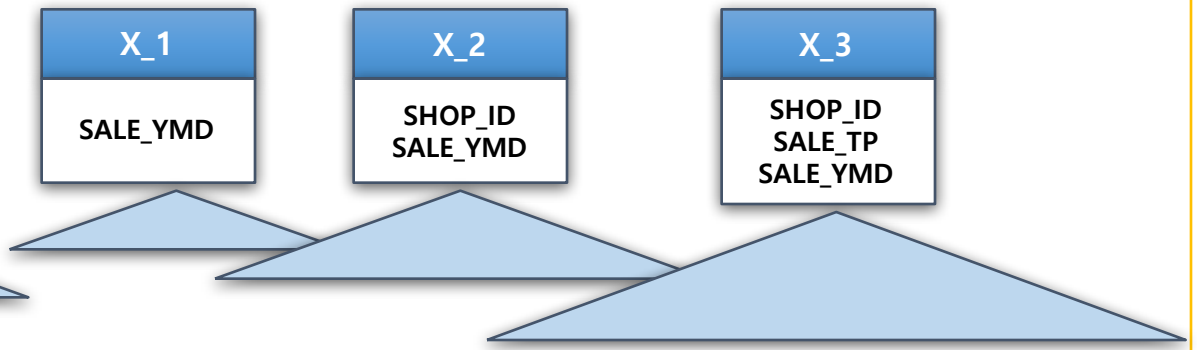
```
SELECT   T1.SALE_YMD
        ,COUNT(*) SALE_CNT
FROM     TEST_SALE T1
WHERE    T1.SHOP_ID = 'SHOP_23'
AND      T1.SALE_YMD BETWEEN '20170301' AND '20170310'
AND      T1.SALE_TP IN ('B2B','B2C')
GROUP BY T1.SALE_YMD;
```

复合索引的策略 – 3

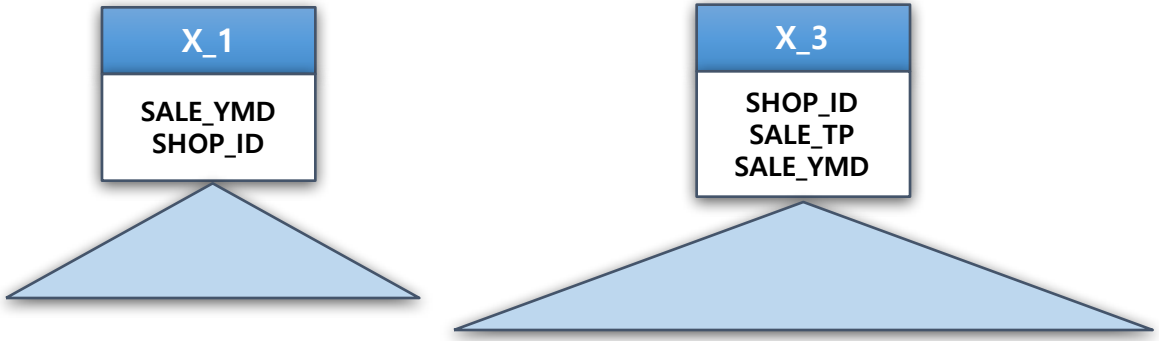
INDEX GROUP - 1



INDEX GROUP - 2



INDEX GROUP - 3



索引策略不合理的话，对数据增删改有很大的负担。

云和恩墨
ENMOTECH

数据驱动 成就客户 未来！

谢谢！