DB2查询访问计划分析与优化

张广舟

日程

- **优化器（Optimizer）概述**
  - 优化器和优化过程
- **查询访问计划（Query Access Plan）分析**
  - 如何得到查询访问计划
  - 查询访问计划的组成部分
  - 查询访问计划分析与优化方法

日程

- **优化器（Optimizer）概述**
  - 优化器和优化过程
- **查询访问计划（Query Access Plan）分析**
  - 如何得到查询访问计划
  - 查询访问计划的组成部分
  - 查询访问计划分析与优化方法

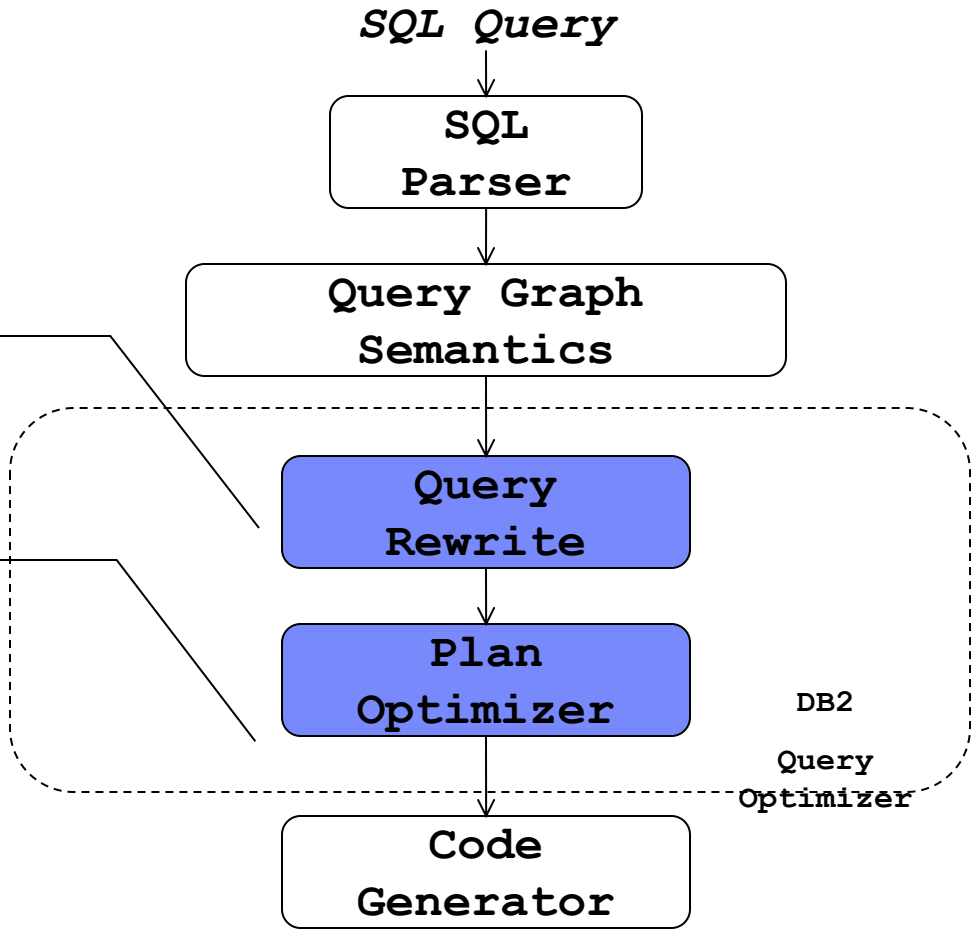
优化器

- SQL是“高级语言”。和一般编程语言类似，对于某一条特定的SQL，在DB2中总是有很多种可能的执行方案。
- 实际应用中SQL通常非常复杂，很难依靠人本身来确定最优的执行方案。
- 优化器就是为我们找出最优执行方案设计的（实际上是一系列的算法）。

优化过程

Determines the optimal query specification

Determines the optimal query execution plan



优化步骤1 – 查询重写

- 用户可能写出各种各样的SQL，不一定是最优形式的。有些SQL还是机器自动生成的
- 有些操作计划优化中不支持
- 查询重写还做一些基本的整理SQL的工作，比如计算常数表达式

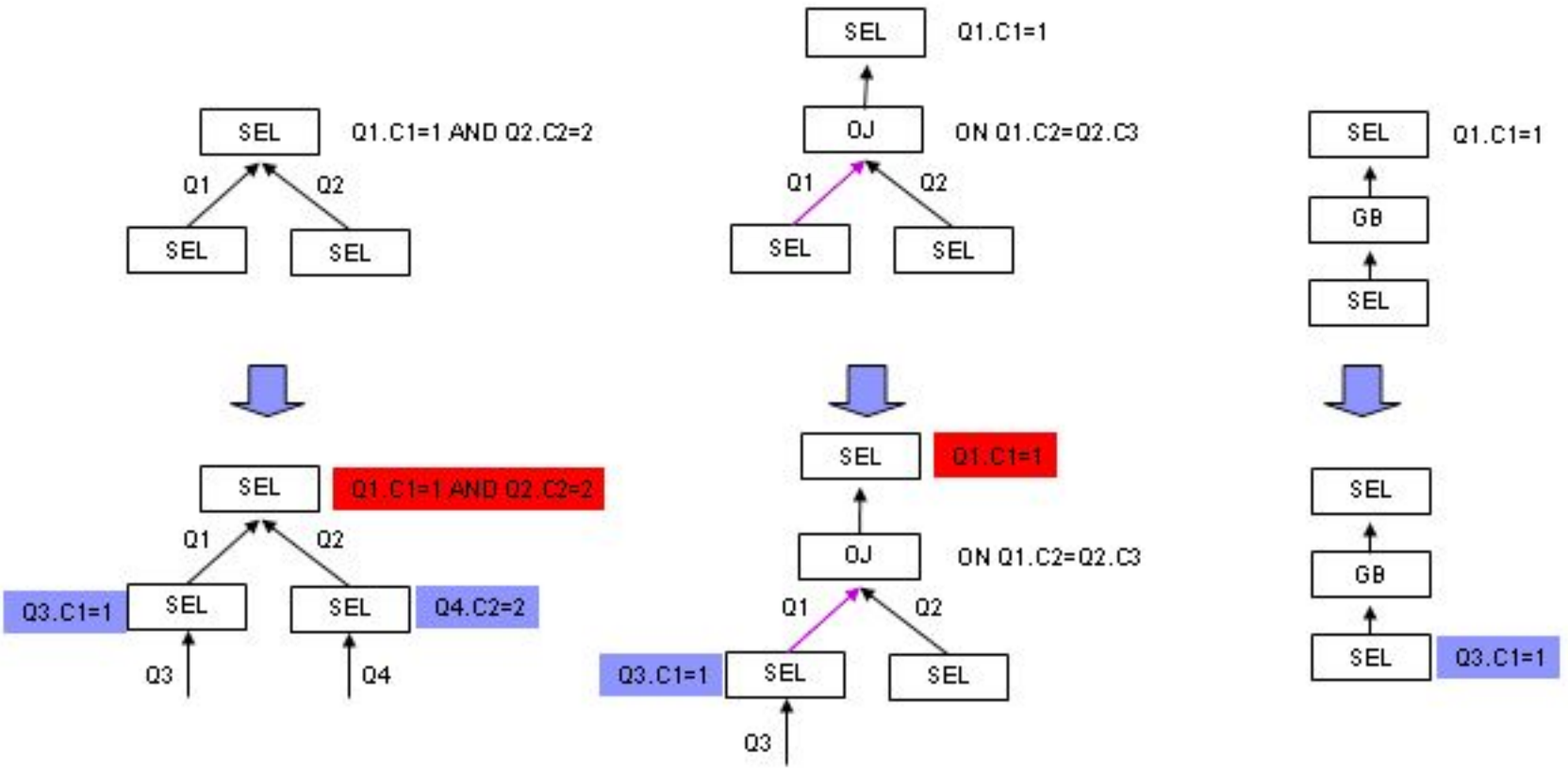
Predicate optimizations (1)

- **Objectives:**
  - Utilize indexes, provide better filter factor estimation, and benefit other rewrite rules
- **Adding new predicate via transitivity**
  - C1=C2 AND C2=1 → C1=1
- **De Morgan's law**
  - NOT (P1 OR P2) → (NOT P1) AND (NOT P2)
- **Adding new predicate via inequality relationships**
  - C1 > 1 AND C2 > C1 → C2 > 1
  - C1 > 1 AND C2 > C1 AND C2 < 1 → 1=0
- **Rewriting date/time/timestamp predicates**
  - YEAR(date) = 2005 AND MONTH(date) = 10
    → date BETWEEN '1-OCT-2005' AND '31-OCT-2005'
- **Adding new predicate from definition of generated columns**
  - Column G1 defined as C1+C2,
  - Predicates C1=2 AND C2=4 → G1=6

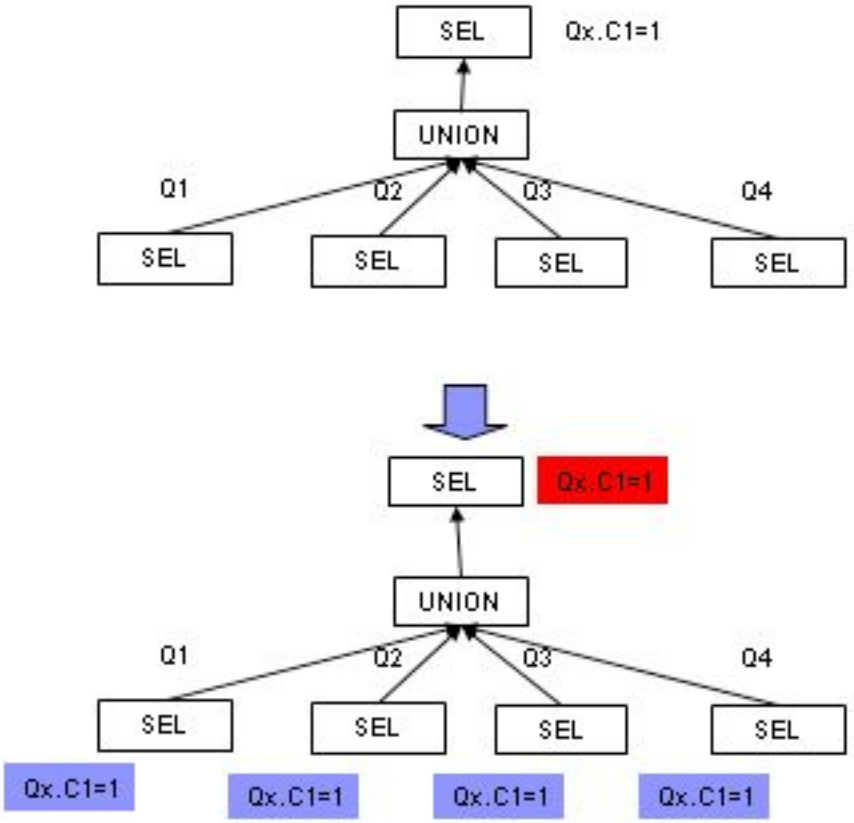
Predicate optimizations (2)

- **Adding redundant predicate**
  - C1 LIKE 'A%' → Adding C1 BETWEEN 'A'+"lowest value" AND 'A'+"highest value" to utilize start-stop key index access
  - SUBSTR(C1,1,2) = 'AB' → C1 BETWEEN 'AB'+"lowest value" AND 'AB'+"highest value"
- **Breaking up string concatenation predicate**
  - C1||C2 = 'A_STRING' → C1='A_' AND C2 = 'STRING' (assuming C1 is CHAR(2))
- **Replicating predicate to the other side of the joins**
  - T1.C1 = T2.C2 and T1.C1 > 0 → T2.C2 > 0

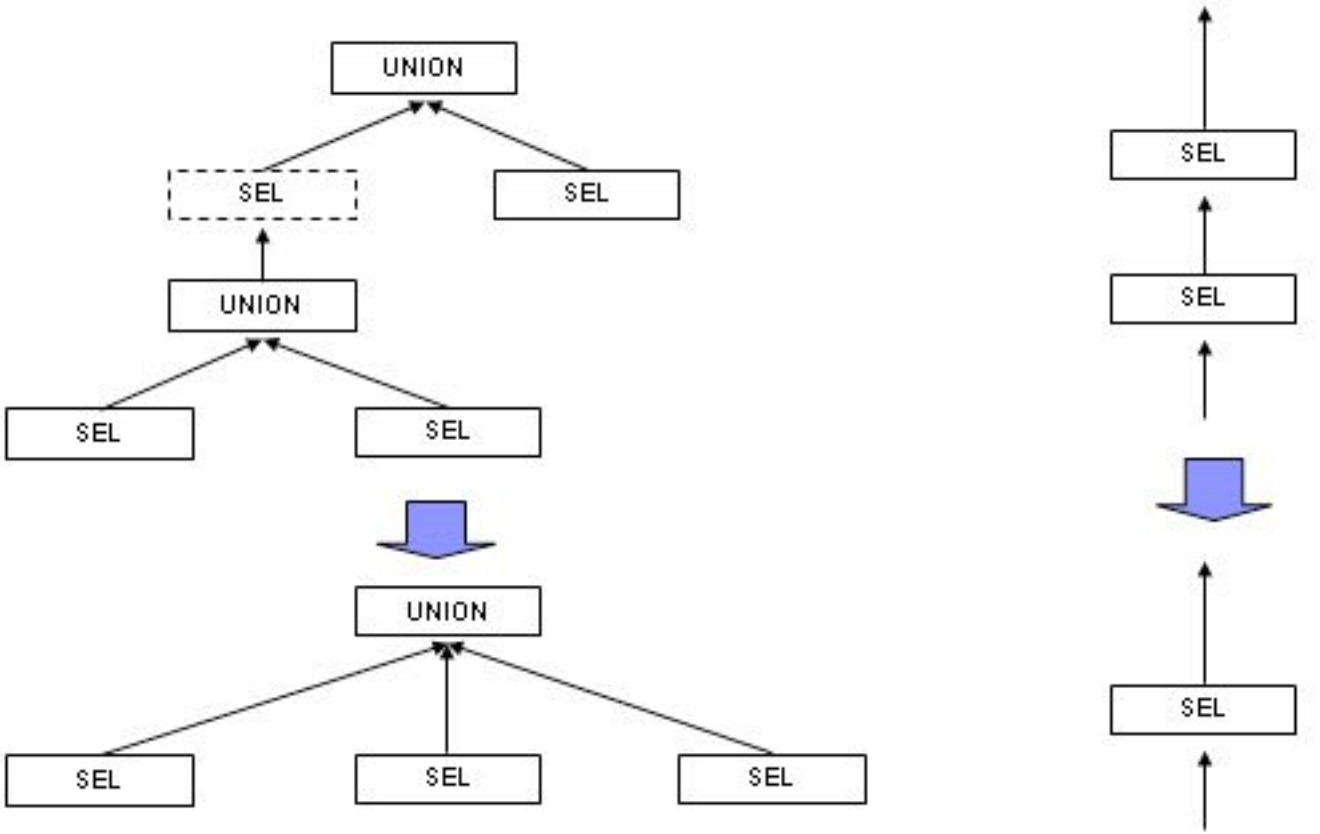
Predicate push down (1)



Predicate push down (2)



Merging operations



IN-list to join

```
select e.*
from   employee e
where e.empno in (123, 234, 345)
```

```
select e.*
from   employee e,
       (values(123), (234), (345)) as q(no)
where e.empno=q.no
```

- Only when there is an index on the "in" column
- Join is generally better than index-ORing

Subquery to join

```
SELECT *
FROM   T1
WHERE T1.C1 IN
        (SELECT T2.C1
         FROM   T2)
```

```
SELECT DISTINCT T1.ROWID, T1.*
FROM   T1,T2
WHERE T1.C1 = T2.C1
```

```
SELECT T1.*
FROM   T1
WHERE EXISTS
        (SELECT 1
         FROM   T2
         WHERE T2.C1=T1.C1)
```

If T1's columns cannot be sorted

```
SELECT T1.C1, T1.LOB
FROM   T1,
        (SELECT DISTINCT T2.C1
         FROM T2) AS Q.C1
WHERE T1.C1 = Q.C1
```

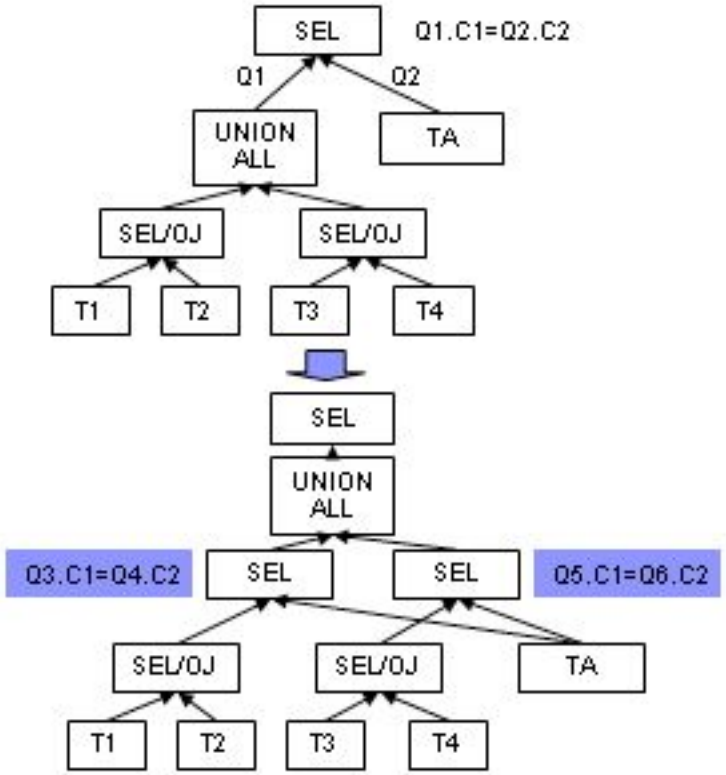
NOT EXISTS to anti-join

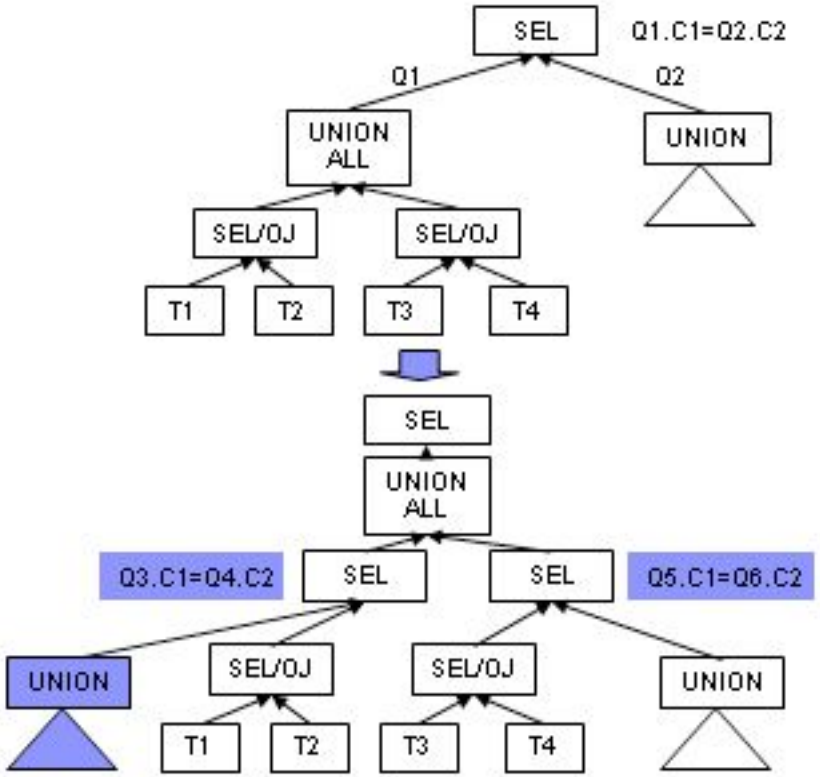
```
SELECT *
FROM   T1
WHERE NOT EXISTS
        (SELECT 1
         FROM   T2
         WHERE T2.C1=T1.C1)
```

```
SELECT T1.*
FROM   T1 LEFT OUTER JOIN T2
ON     T1.C1 = T2.C1
WHERE T2.C1 IS NULL
```

T1.C1	T2.C1	T1 ANTIJOIN T2 ON T1.C1=T2.C1
1	1	2
2	3	2
2	null	null
null		

Join predicate push down through UNION





优化步骤2 – 计划优化

- 使用动态规划算法，不断选取代价较优子计划，自底向上生成更大的计划，最终生成满足查询的完整计划
- 生成不同访问方法、连接顺序和连接方法的备选子计划，同时依据**Cardinality, CPU, I/O, Communication**代价和内存使用评估代价

优化器考虑的信息

- **Table Statistics**
  - number of rows (CARD), pages (NPAGES, FPAGES)
- **Column Statistics**
  - Number of distinct values (COLCARD), Average Length, High and Low Values (HIGH2KEY, LOW2KEY)
  - Non-uniform distribution statistics
    - N most frequent values (default 10)
    - M quantiles (default 20)
- **Index Statistics**
  - NLEAF, NLEVEL, FirstNKeyCard, CLUSTERRATIO, CLUSTERFACTOR / PAGE FETCH PAIRS

优化器考虑的信息

- **Key configuration parameters (not an exhaustive list):**
  - CPU Speed strongly influences cost
  - Communications Bandwidth influences the costs as the number of nodes increases
  - I/O Overhead and Transfer Rate
    - I/O is typically one of the largest parts of all query costs and often is the key to a good or bad plan
  - Buffer Pool
    - Optimizer considers the SUM of all Buffer Pools, not individual Buffer Pools
  - Optimization Level
  - Sort Heap
  - Statement Heap
  - Average Applications
  - Locks Available (Lock List Size * Maximum Lock List %)

优化器考虑的信息

- **Table Design**
- **DPF**
  - Forces the optimizer to consider how tables need to be partitioned for performing joins, grouping, distincting, ordering
- **Range Partitioning**
  - Allows the optimizer to potentially eliminate large ranges of a table from considering during base table accesses
- **MDC (Multi-Dimensional Clustering)**
  - Also allows the optimizer to potentially eliminate large ranges of a table, and prefetch blocks of data instead of individual rows

没有被优化器考虑的方面

- **The Cost Optimizer does not care so much about:**
  - Sort Heap Threshold
  - Containers and their layout
  - HA (High Availability)
  - Logging
  - etc …

优化器不是完美的

- 它基于一种计算模型，设计上是为了能在大多数（而不是所有）情况下得到相对较优且可接受的结果
- 它依赖于我们提供给它的各种信息才能正常工作

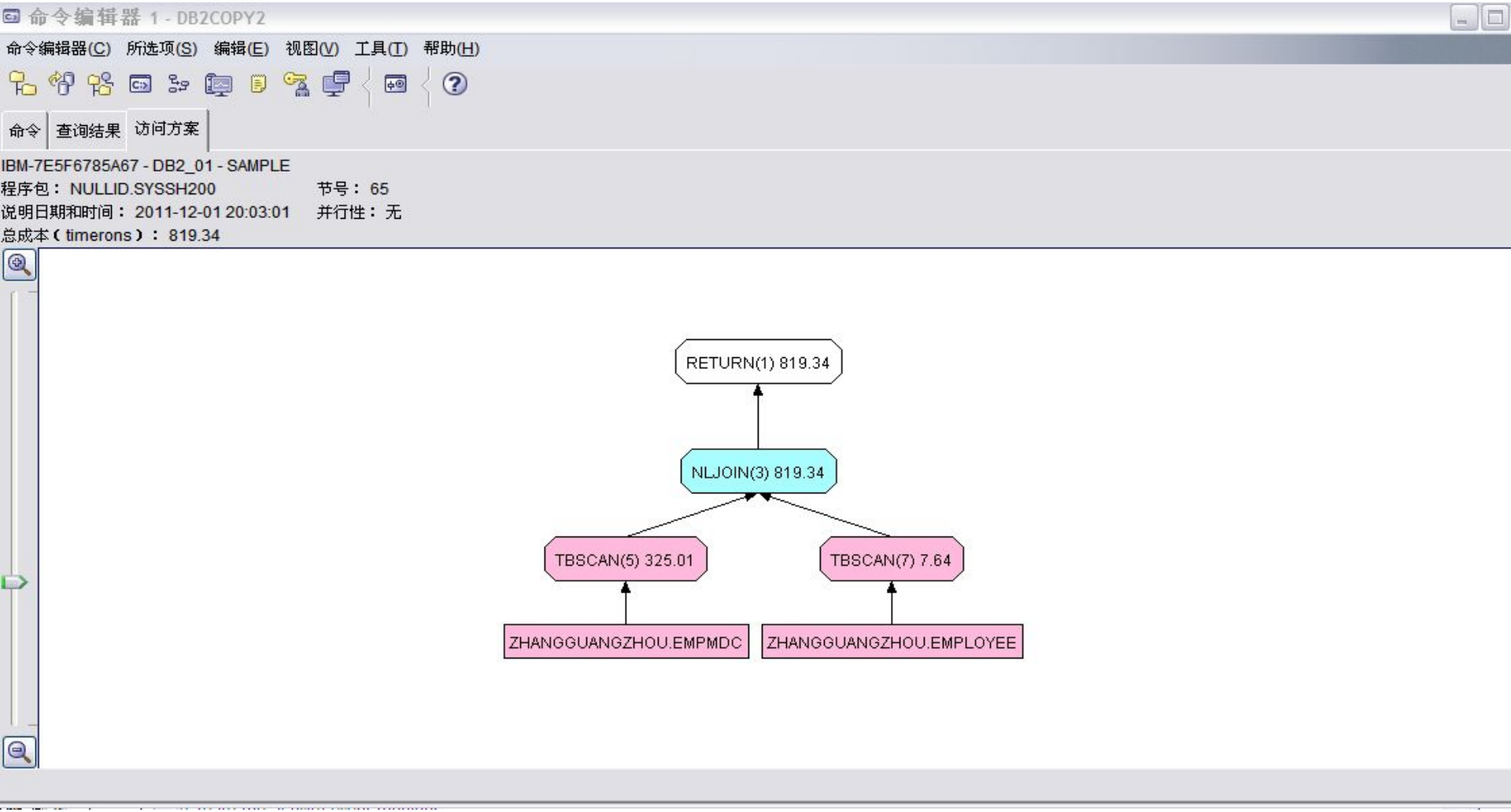
日程

- **优化器（Optimizer）概述**
  - 优化器和优化过程
- **查询访问计划（Query Access Plan）分析**
  - 如何得到查询访问计划
  - 查询访问计划的组成部分
  - 查询访问计划分析与优化方法

获得查询访问计划

- **Visual Explain – 图形工具**
- **Db2exfmt – 文本工具**

Visual Explain 的输出



文本形式的查询计划

- **生成查询计划**
  - db2 "explain plan for <sql text>”
- **用db2exfmt输出查询计划到文本文件，最简单的输出命令：**
  - db2exfmt –d <dbname> -1 –o <output file name>

日程

- **优化器（Optimizer）概述**
  - 优化器和优化过程
  - 影响优化器的因素
- **查询访问计划（Query Access Plan）简介**
  - 如何得到查询访问计划
  - 查询访问计划的组成部分
  - 查询访问计划分析与优化方法

- 基本信息
  - 包含了查询访问计划需要的环境信息如数据库信息、程序包信息、解析时间及解析请求用户等信息

- DB2_VERSION: 09.07.5
- SOURCE_NAME: SQLC2H22
- SOURCE_SCHEMA: NULLID
- SOURCE_VERSION:
- EXPLAIN_TIME: 2012-09-02-03.44.50.117007
- EXPLAIN_REQUESTER: IIDEV5
- Database Context:
- ----------
- Parallelism: None
- CPU Speed: 4.000000e-05
- Comm Speed: 0
- Buffer Pool size: 34265
- Sort Heap size: 446
- Database Heap size: 2345
- Lock List size: 9898
- Maximum Lock List: 60
- Average Applications: 1
- Locks Available: 190041
- Package Context:
- ----------
- SQL Type: Dynamic
- Optimization Level: 5
- Blocking: Block All Cursors
- Isolation Level: Cursor Stability

Original/Optimized Statement

Original Statement:

```
------------------
select ...
from tpcd.lineitem
```

Optimized Statement:

```
-------------------
SELECT ...
FROM
    (SELECT ...
     FROM
     (SELECT ...
      FROM TPCD.L_SUMMARY AS Q1
     ) AS Q2
    ) AS Q3
```

- 查询计划树及其扩展信息
  - 计划树以树型结构形象地提供了查询执行计划的整体情况
  - 计划树后的扩展信息包含了计划树中各个操作符的详细信息。

```
Access Plan:
-----------
        Total Cost:             3.77705e+07
        Query Degree:           1

              Rows
             RETURN
             (   1)
              Cost
               I/O
               |
           6.55036e+07
             NLJOIN
             (   4)
           3.77705e+07
           4.76894e+06
           /----+----\
         19          3.44756e+06
       TBSCAN          TBSCAN
       (   5)          (   8)
       16.6098       1.98792e+06
          1            250997
          |              |
         19          6.55036e+07
   TABLE:MYSCHEMA  DP-TABLE: MYSCHEMA
         DWT2            DWT1

Plan Details:
-----------
    1) RETURN: (Return Result)
        Cumulative Total Cost:      3.77705e+07
        Cumulative CPU Cost:        8.75522e+11
        Cumulative I/O Cost:        4.76894e+06
        Cumulative Re-Total Cost:   3.77705e+07
```

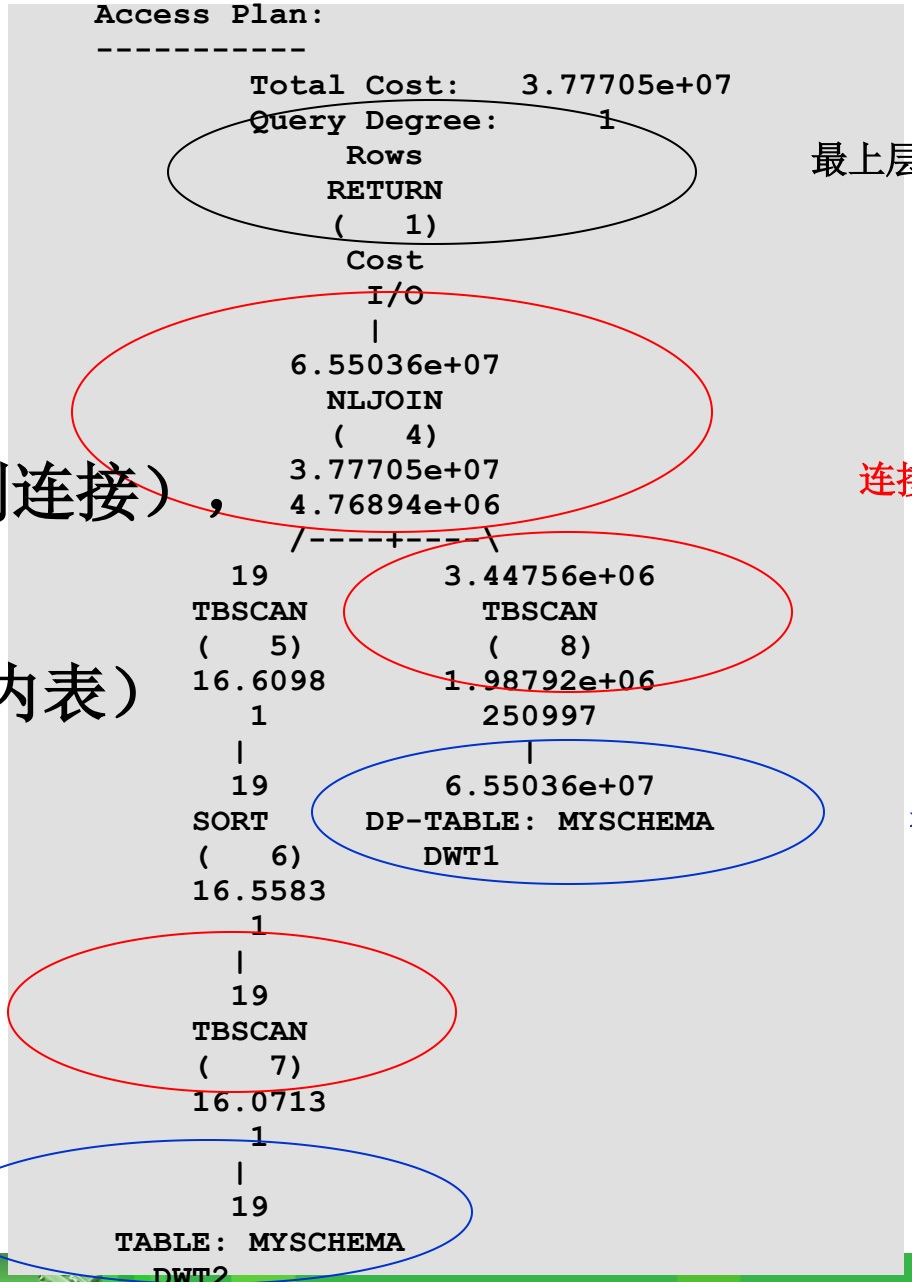
```
select  a.*, b.*
from
        MYSCHEMA.DWT1 a,
        MYSCHEMA.DWT2 b
where
    a.bc3_cd = b.bc3_cd
    and
    a.region = 'Asian'
```

常见的计划节点类型	所执行的操作
DELETE	删除
EISCAN	扩展索引扫描
FETCH	使用指定的记录标识符从表中获取列。
FILTER	通过应用一个或多个谓词过滤数据。
GENROW	生成行
GRPBY	按指定列或函数的公共值组织行，并对集合函数求值。
HSJOIN	显示一个散列连接，其中一个或多个表在连接列上是混编的。
INSERT	插入
IXAND	动态位索引与操作
IXSCAN	使用可选的启动/停止条件扫描表索引，产生有序的行流。
MSJOIN	显示合并连接，其中外部和内部表必须按连接谓词的顺序排列。
NLJOIN	显示嵌套循环连接，为外部表中的各行访问内部表一次。
RETURN	返回结果集
RIDSCN	行标识RowID扫描
RPD	远程下推，常用于非关系型包装器对象。在联邦系统中使用。
SHIP	从远程数据库源中检索数据。在联邦系统中使用。
SORT	排序
TBSCAN	通过直接从数据页中读取所有数据而检索行。
TEMP	将数据存储在临时表中以便读回（很可能要读回多次）。
TQ	Table Queue
UNIQUE	消除特定列值重复的行。
UNION	串联来自多个表的行流。
UPDATE	更新
XISCAN	索引扫描。用于XML数据。
XSCAN	XML文件扫描。

日程

- **优化器（Optimizer）概述**
  - 优化器和优化过程
- **查询访问计划（Query Access Plan）分析**
  - 如何得到查询访问计划
  - 查询访问计划的组成部分
  - 查询访问计划分析与优化方法

如何分析查询计划？



最上层节点

从下往上看（表到扫描到连接），

连接和扫描节点

从左往右看（外表到内表）

表

查询计划和SQL语句的关系

```
select  a.*, b.*
from
        MYSCHEMA.DWT1 a,
        MYSCHEMA.DWT2 b
where
    a.bc3_cd = b.bc3_cd
    and
    a.region = 'Asian'
```

连接

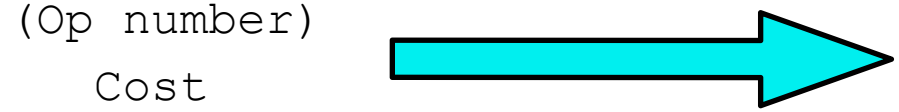
扫描

```
Access Plan:
-----------
        Total Cost:     3.77705e+07
        Query Degree:        1
            Rows
           RETURN
           (   1)
            Cost
             I/O
             |
          6.55036e+07
           NLJOIN
           (   4)
          3.77705e+07
          4.76894e+06
          /----+----\
        19          3.44756e+06
      TBSCAN          TBSCAN
      (   5)          (   8)
      16.6098       1.98792e+06
         1             250997
         |               |
        19          6.55036e+07
      SORT        DP-TABLE: MYSCHEMA
      (   6)          DWT1
      16.5583
         1
         |
        19
      TBSCAN
      (   7)
      16.0713
         1
         |
        19
    TABLE: MYSCHEMA
       DWT2
```

最上层节点

表

理解一个计划节点



```
    Rows
Operator Name
(Op number)
    Cost
    I/O
     |
 Cardinality
```

```
….
  |
478775
TBSCAN
(   7)
22453.8
 12728
  |
496100
……
```

分析查询计划第一式-检查表扫描（TABSCAN）

- 检查计划中的扫描节点，看是否有表扫描（TABSCAN）
- DB2有两种扫描方法，索引扫描（IXSCAN）和表扫描（TABSCAN），一般索引扫描比表扫描快
- 非常多的性能问题是由于表扫描引起的！

检查表扫描

```
select  a.*, b.*
from
        MYSCHEMA.DWT1 a,
        MYSCHEMA.DWT2 b
where
    a.bc3_cd = b.bc3_cd
    and
    a.region = 'Asian'
```

```
Access Plan:
-----------
        Total Cost:    3.77705e+07
        Query Degree:      1
             Rows
            RETURN
            (   1)
             Cost
              I/O
              |
           6.55036e+07
            NLJOIN
            (   4)
           3.77705e+07
           4.76894e+06
           /----+----\
         19           3.44756e+06
       TBSCAN           TBSCAN
       (   5)           (   8)
       16.6098        1.98792e+06
          1             250997
          |               |
         19           6.55036e+07
       SORT       DP-TABLE: MYSCHEMA
       (   6)        DWT1
       16.5583
          1
          |
         19
       TBSCAN
       (   7)
       16.0713
          1
          |
         19
   TABLE: MYSCHEMA
      DWT2
```

表扫描

表扫描

索引的优势

- 显著提高数据定位的速度
- 减少被扫描的行数
- ORDER BY 和GROUP BY 字句
- 带INCLUDE的索引
  - CREATE UNIQUE INDEX EMP_IX ON EMPLOYEE(EMPNO) INCLUDE(FIRSTNAME, JOB)
- 注意，UPDATE, INSERT, DELETE和 LOAD 的操作，会导致额外的CPU 和I/O 开销

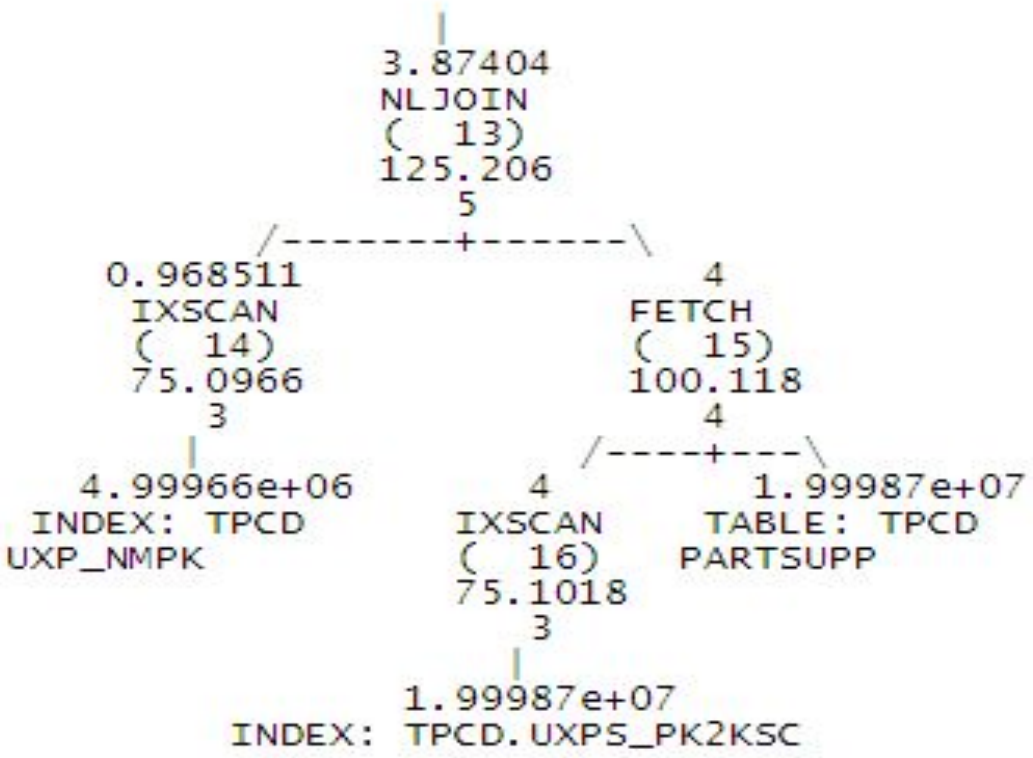
建索引的最佳实践

- **为每个主键和外键建立索引**
- **为WHERE里面频繁出现的字段建立索引**
  - 合理的确定索引字段的顺序
    - 例如，where c1 > 3 and c2 = 5
- **为WHERE里面用于等式谓词（包括连接和选择谓词）**
- **考虑建立index-only索引**
  - Select deptname from department where did = 3 order by deptname
- **避免不必要的索引**

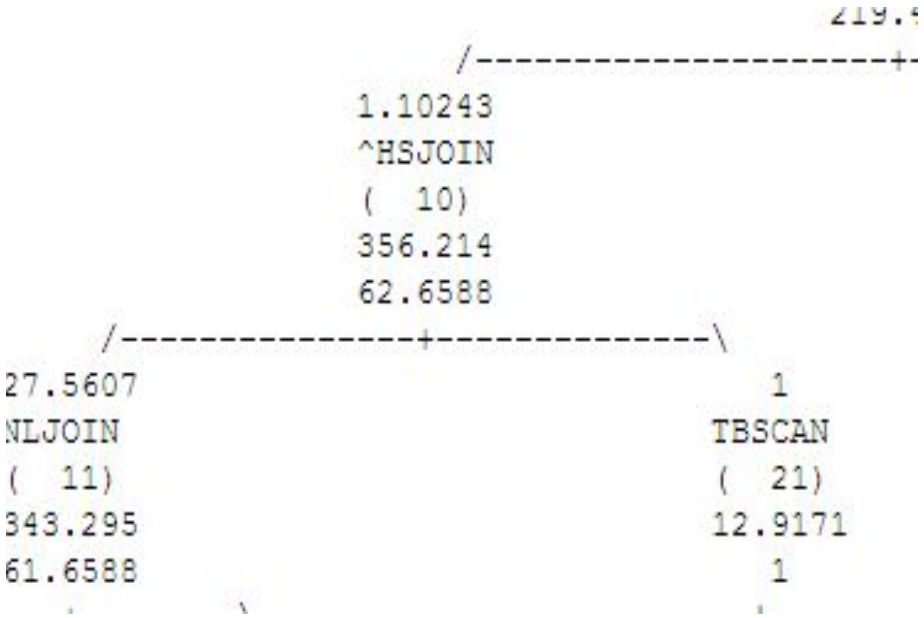
分析查询计划第二式-连接方法

- 检查计划中的连接方法
- DB2有三种连接方法: 嵌套（NLJN）、哈希（HSJN）和归并（MSJN）
- 我们要使用合理的连接谓词，使三种方法的计划都成为可能

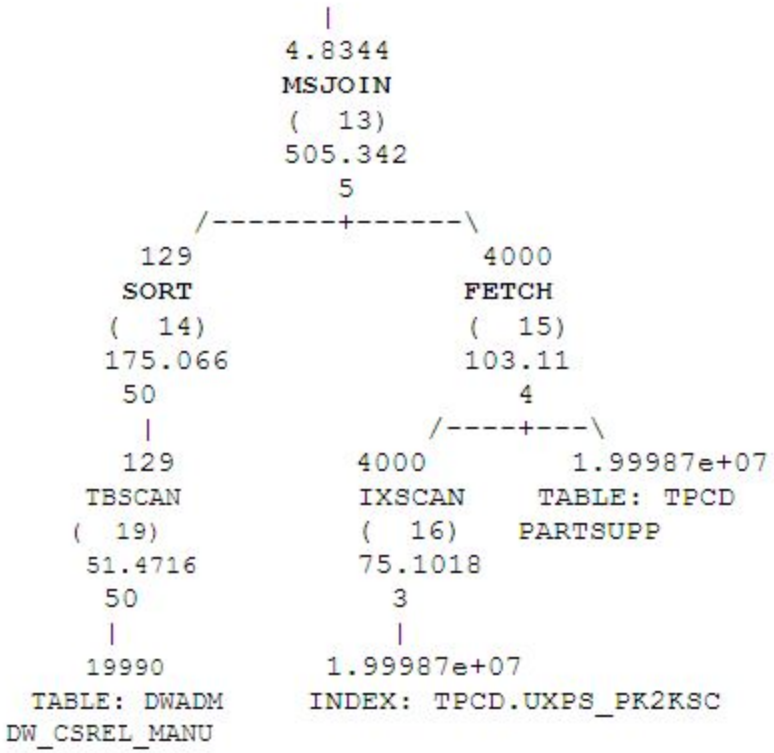
NLJOIN - 适用于内表有index 的情况，不会访问全部的内表



HSJOIN- 需要等式等长连接谓词，适用于内表无索引，且可以放入全部内存的情况



MSJOIN –需要等式连接谓词，适用内外表已排序，且重复记录不多、重叠少的情况



分析查询计划第三式-检查表和计划节点记录数

- 检查计划中的表和扫描、连接等节点的记录数，看是否有不合理（比如<1，1000）的情况
- 表的记录从统计信息得出，其他节点的记录由DB2根据统计信息计算出来
- 如果记录数不对，需要收集更详细的统计信息：

  runstats on table schema.employee on columns(workdept) with distribution and detailed indexes all

分析查询计划第三式-检查表和计划节点记录数

表节点

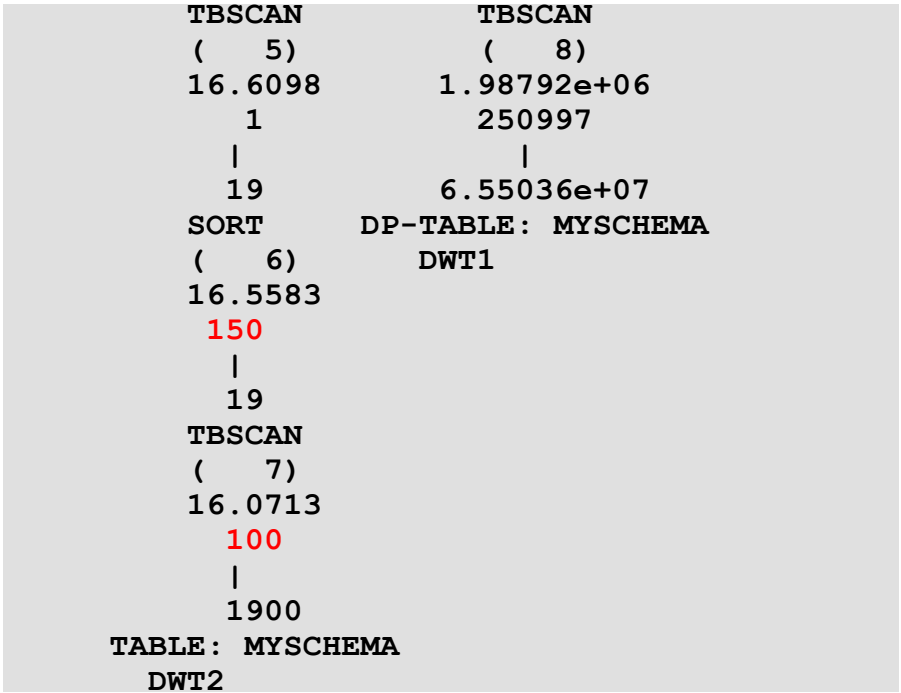
```
  |
1000
TABLE: MYUSER.TABLE1
```

其他节点

```
....
  |
478775
TBSCAN
(   7)
22453.8
 12728
  |
```

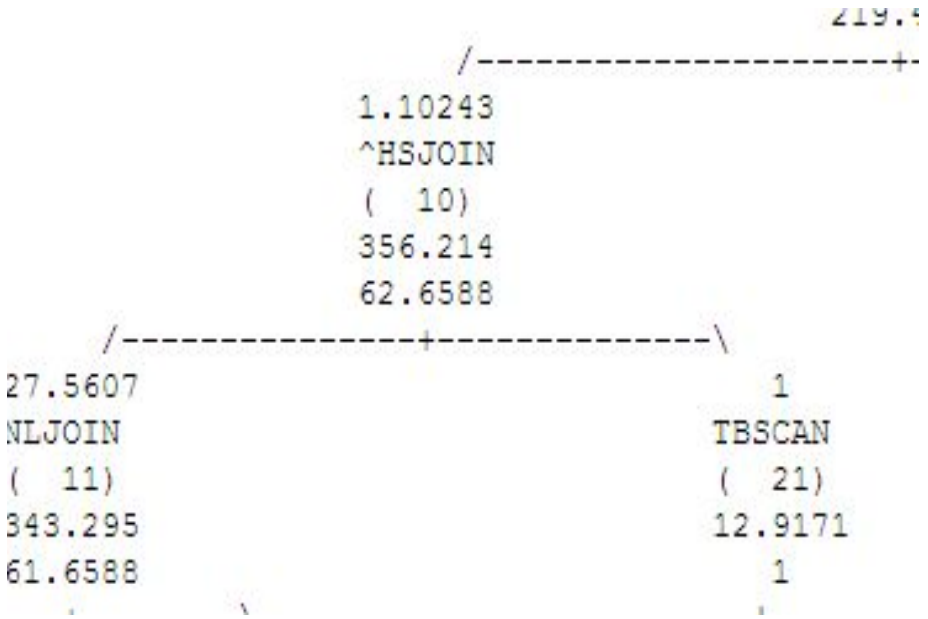
分析查询计划第四式-查看IO溢出

- 比较计划中的扫描、连接等节点的IO数，看是否有IO溢出的情况



分析查询计划第五式-查看代价最大的计划节点

- 每个节点的执行代价等于本节点的代价数减去相邻下面节点的代价和



分析查询计划第六式-分析SQL语句

- **计划中有Optimized Statement，可从中找出SQL语句不符合编码规范的地方**

分析查询计划第七式-其他优化方法

- **物化视图MQT**
- **Optimizer Profile**
  - XML based plan hint
- **Selectivity子句**
  - Where Employee.Employee_Name = ‘John’ selectivity 0.0001

总结：SQL优化关键点

- 发现表扫描，创建索引
- 检查连接方法
- 记录数分析，runstats
- 代价分析
- IO分析
- SQL编写
- 其他方法

谢　　谢