Fundamental Techniques for Order Optimization

Sahak Maloyan

Outline

- Motivation
- DB2 Optimizer overview
- Fundamental Operations for Order Optimization
- The Architecture for OP in DB2
- Query optimization Example
- Performance Results
- Conclusion

Motivation

- Order Optimization
  - A non-trivial problem
    – Single complex query gives rise to multiple „interesting orders“
  - Interesting order (I)
    – Specification for any ordering of the data
    – Useful for processing a join, an ORDER BY, GROUP BY or DISTINCT
  - Optimizer must detect
    – When indexes provide an interesting order
    - Optimal place to sort
      - Avoid sorting
      - Sort-ahead
    - The satisfaction by combining two or more interesting orders by a single sort

Goal:

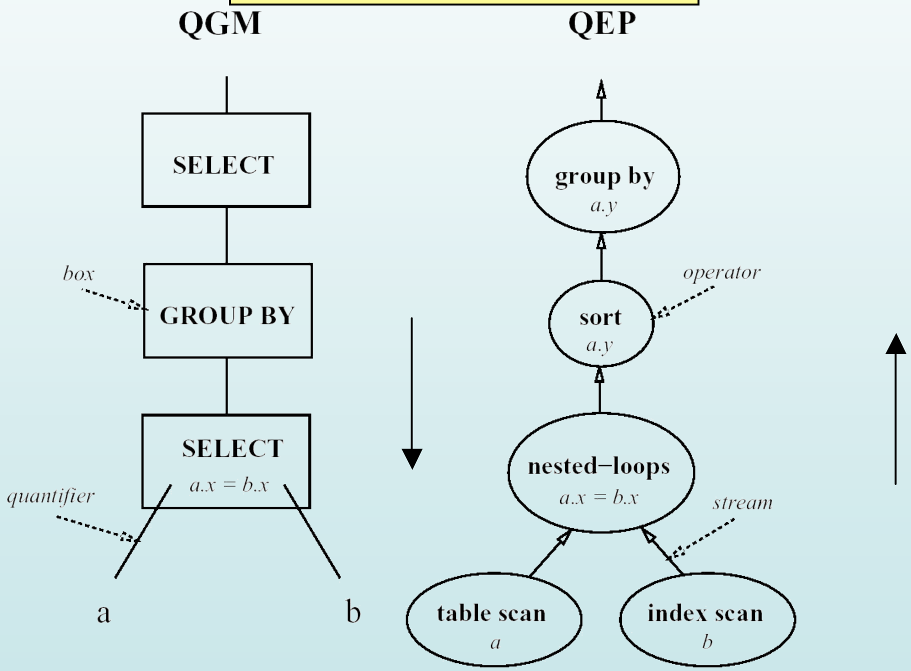
- Describing novel optimization techniques
  - pushing down sorts in joins
  - minimizing the number of sorting columns
  - detecting when sorting can be avoided
  - taking advantage of predicates keys or indexes

DB2 Optimizer overview

- Parse the input query and convert it to an intermediate form called the query graph model (QGM).

- Transform QGM into a semantically equivalent but more „efficient“ QGM using heuristics such as predicate push down, view merging, and subquery-to-join transformation.
- Perform cost-based optimization generate QEP and keep the least costly one.

QUERY
select a.y, sum(b.y)
from a, b
where a.x = b.x
group by a.y



Simpe QGM and QEP Example

• Reduce Order

Reduction is the process of **rewriting** an order specification in a simple canonical form.

Some Exampels:

1. **column=constant:**

    Interesting order I=(x,y) and Input Stream (IS) has the order property OP=(y);

    Naive test => I not satisfied by OP=> add **sort** to the QEP.

    But if x=10 applied to all records of IS

    => I rewritten as I_{new}=(y) => no sort.

2. **column equvalence**

    I=(x,z), OP=(y,z) and we apply further x=y

    => OP_{new}=(x,z);

• Reduce Order (Cont.)

3. **take keys into account**

$I=(x,y)$, $OP=(x,z)$;

x is a key => $I_{new}=(x)$, $OP_{new}=(x)$

- Keys are special cases of functional dependencies(FDs).

FD: A set of columns $A=\{a_1, a_2,...a_n\}$ functionally determines columns $B=\{b_1, b_2,...b_m\}$ if for any two records with the same values for columns in A, the values for columns in B are also the same

•Notation: $A \rightarrow B$

•**Reduction:** Mapping of predicate relationships and keys to FDs

- **Reduce Order (Cont.)**

The Algorithm:

input:
  a set of FDs, applied predicates, and
  order specification $O = (c_1, c_2, ..., c_n)$
output:
  the reduced version of O

1) rewrite O in terms of each column's
   equivalence class head
2) scan O backwards
3) for (each column c_i scanned)
4)   let $B = \{c_1, c_2, ..., c_{i-i}\}$, i.e.,
     the columns of O preceding c_i
5)   if ($B \rightarrow \{c_i\}$) then
6)   remove c_i from O
7)   endif
8) endfor

Example: I=(x); x=10

⇒{} →{x}; "empty headed"

⇒ I_{new}=()

- **Test Order**
  - Testing if OP satisfies *interesting order*
  - If not the sort is added to QEP
    - But with minimal number of sorting columns

    => Minimizing the sort costs

Fundamental Operations for Order Optimization

- **Test Order(cont.)**

  The Algorithm:

input:
  an interesting order I and an order property OP

output:
  true if OP satisfies I, otherwise false

1) reduce I and OP
2) if (I is empty or the columns in I are a prefix of the columns in OP) then
3) return true
4) else
5) return false
6) endif

- **Cover Order**
  - Tries to combine interesting orders in the top down scan of QGM
  - When two interesting orders are combined, a **cover** is created

    Cover: The cover of two interesting orders I_1 and I_2 is a new interesting order C such as that any order property which satisfies C also satisfies **both.**

Examples: 1. I_1 =(x) and I_2 =(x,y) => C=(x,y)

2. I_1 =(y,x) and I_2 =(x,y,z) => no cover

But if x=10 => I_{1new} =(y), I_{2new} =(y,z) => C=(y,z)

- **Cover Order(cont.)**

The Algorithm:

input:

interesting orders I_1 and I_2

output:

the cover of I_1 and I_2; or a return code indicating that a cover is not possible

1) reduce I_1 and I_2
2) *w.l.o.g.*, assume I_1 is the shorter interesting order
3) if (I_1 is a prefix of I_2) then
4) return I_2
5) else
6) return "cannot cover I_1 and I_2"
7) endif

- **Homogenize Order**
  - **Homogenization:** When an interesting order I is pushed down, some columns may have to be substituted with equivalent columns in the **new** context.
  - **Example:**

```
Select *
from a,b
where a.x=b.x
order by a.x,b.y
```

- ORDER BY gives rise to I=(a.x,b.y)
- Order scan tries to push down I to access both tables a and b
- Equivalence class generated by a.x =b.x => I_h =(b.x,b.y).
- But if a.x=b.x is a base table key (key after the join) {a.x} → {b.y} => I_{new}=(a.x) which can be pushed down to the access table a.

- **Note:** Unlike Reduce Order, Homogenize Order can choose any column in the equivalence class for substitution.

- **Homogenize Order(cont.)**

The Algorithm:

input:

an interesting order I and target columns $C = \{c_1, c_2, ..., c_n\}$

output:

I homogenized to C, that is, I_C; or a return code indicating that I_C is not possible

1) reduce I
2) using equivalence classes, try to substitute each column in I with a column in C
3) if (all the columns in I could be substituted) then
4)   return I_C
5) else
6)   return "cannot homogenize I to C"
7) endif

The Architecture for OP in DB2

- The Order Scan of QGM
  - Four stages

1. Determining the input and output requirements for each QGM box
   - output requirements from ORDER BY
   - input requirements currently from GROUP BY
2. Determining the interesting order for each DISTINCT
3. Determining the interesting order for merge-joins and subqueries
4. Traverse the QGM graph in a top-down manner

The Architecture for OP in DB2

- Planning Phase of Optimization
  - Walk the QGM box-by-box and incrementally build a QEP
  - For each box generate the alternative subplan
  - Prune more costly subplan with comparable **properties**
  - Detect whether sort is required or not
  - If an interesting order is pushed down

  ⇒Homogenize Order

The Architecture for OP in DB2

- Planning Phase of Optimization (cont.)

  - Properties
    - P_2 prunes P_1 if P_2.cost ≤ P_1.cost and for every property x, P_1.x ≤ P_2.x

    - The Order Property
      - Using the Test Order Algorithm to compare the order properties
      - Propagate straightforward except projections and joins

    - The Predicate property
      - The set of conjunctions which been allied to Stream
      - Determine both: column equivalence
functional dependencies

The Architecture for OP in DB2

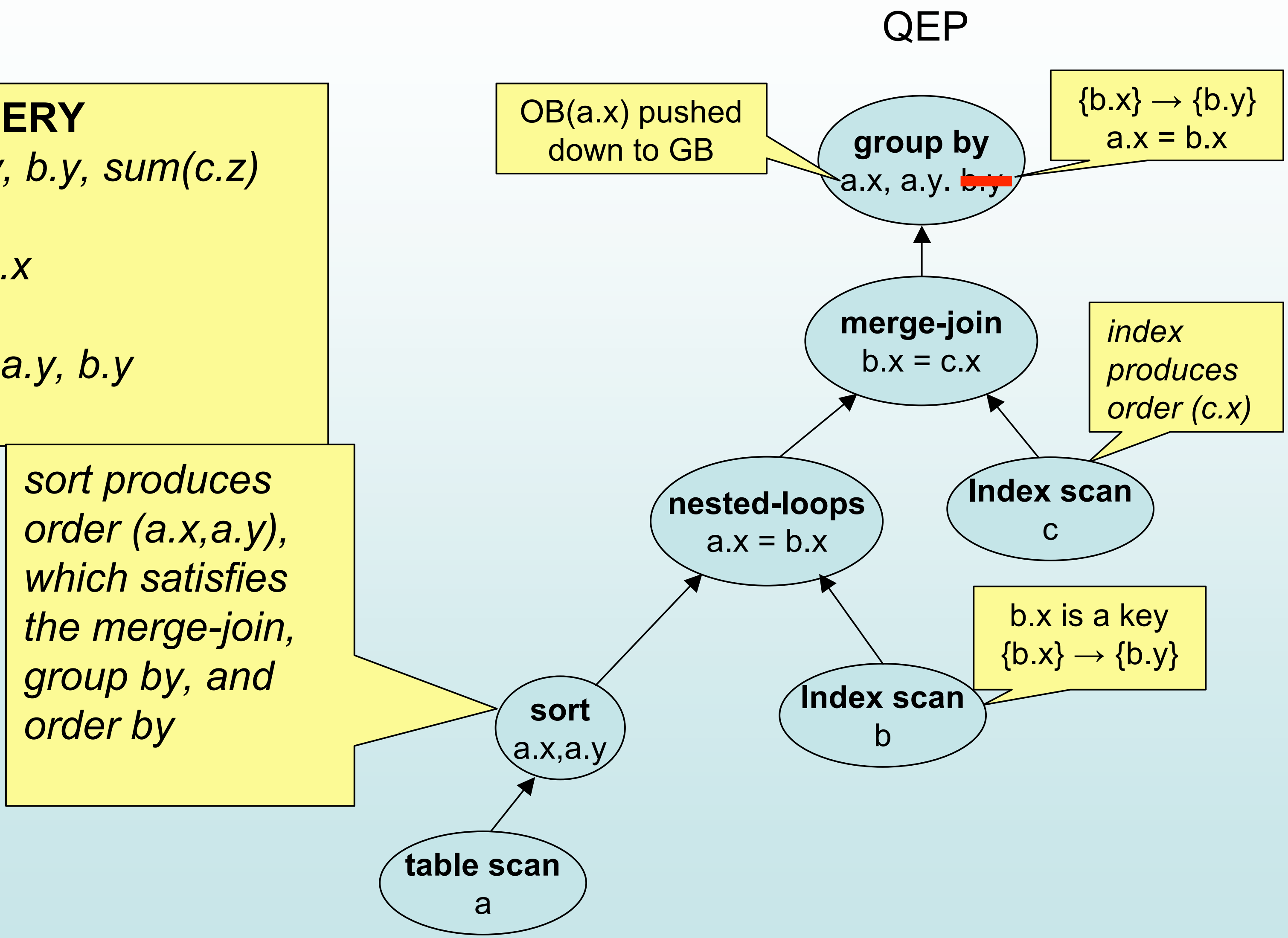
- Planning Phase of Optimization (cont.)
  - Properties (cont.)
    - The Key property
      - Set of unique keys for the stream
      - Represented as a set of columns $K=\{c_1,c_2,\ldots,c_n\}$
      - Keys originate from base-table constrains or can be added via GROUP BY
      - After simplifying each key in the key property, redundant keys removed from key property
    - The FD property
      - Set of FDs which can be empty
      - Originates from a key
      - Example: $K=\{c_1\}$ is a join stream S with columns $\{c_1,c_2,\ldots c_n\}$ Further assume that the key property (KP) of S dos not propagate in the join

      => $\{c_1\} \rightarrow \{c_2,\ldots,c_n\}$ added to the FD property of the join

An Example

QUERY

select a.x, a.y, b.y, sum(c.z)
from a, b, c
where a.x = b.x
and b.x = c.x
group by a.x, a.y, b.y
order by a.x



An Example (cont.)

- the optimizer determined that pushing down the sort before first join in the most efficient QEP
- This is true because the size of table a is smaller that the result of the join.
- If there existed an ordered index on a.x, ay the sort could be eliminated

Advanced Issues

- The order based GROUP BY and DISTINCT operators do not recognize an exact interesting order
- Example: GROUP BY x,y,sum(distinct s) can be satisfied by (x,y,z) or (y,x,z).
- Moreover x,y,z can be ascending or descending order

  => In real Implementation used only one **general Interesting order:** Includes information which columns can permuted and which optimization can be ascending or descending order

- The Optimizer uses this information and **detects** any order that satisfies the Order based GROUP BY

Performance Results

- The biggest improvements
  - decision-support of environments with lots of indexes
- Trademark of the Transaction Processing Council (TPC-D) Benchmark Results
  - TPC-D Database: 1 GB
  - IBM RS/600 Model 59H(66Mhz) server
    - Memory: 512Mb; AIX 4.1
    - Data Striped over 15 discs and 4 I/O controllers
    - Big Block prefetching and I/O Parallelism

    => 100% utilization of CPU

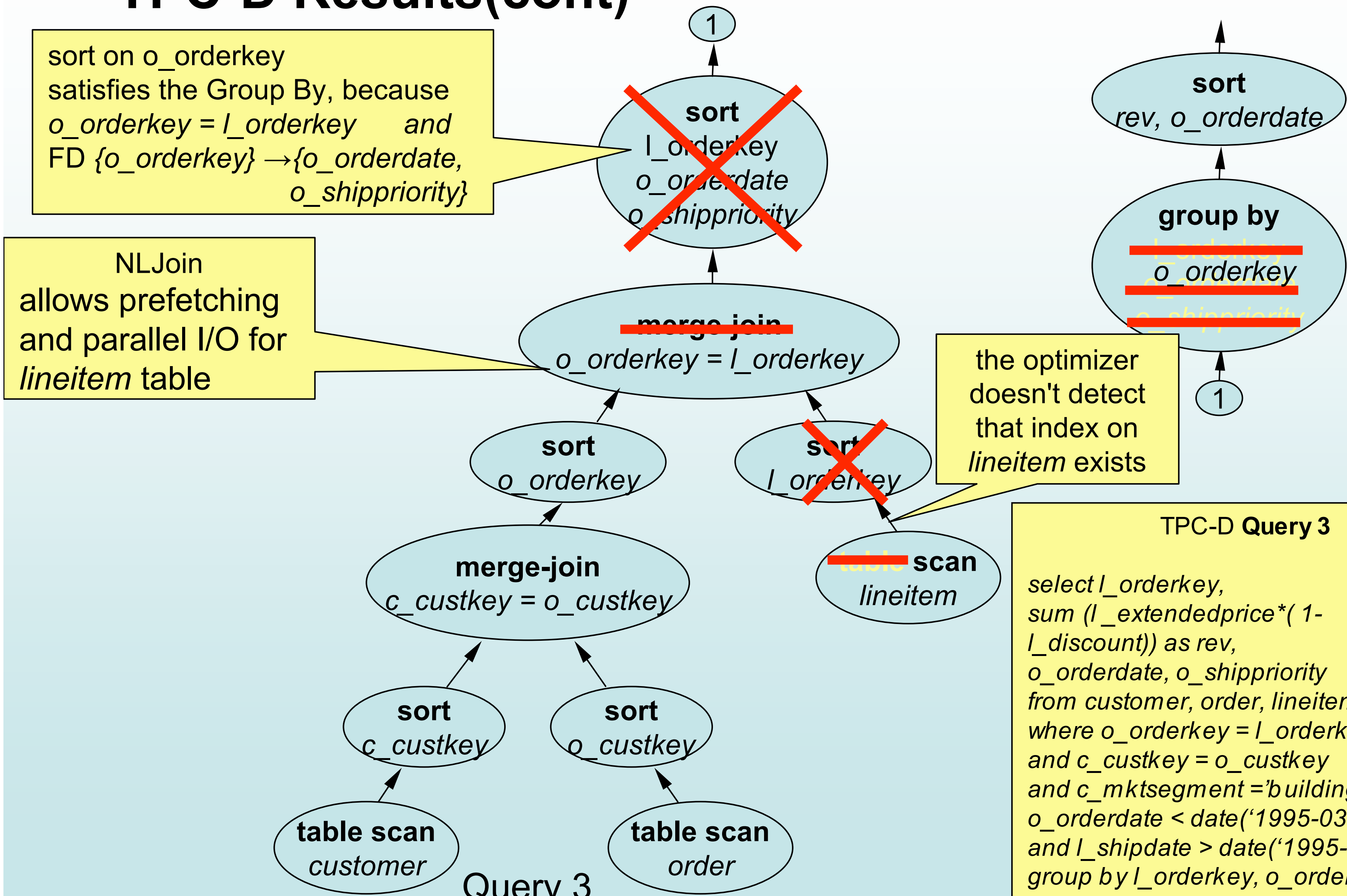
Performance Results

- **TPC-D Benchmark Results(cont.)**

TPC-D **Query 3**

select *l_orderkey,*
sum *(l _extendedprice*(1- l_discount))* ***as*** *rev,*
o_orderdate, o_shippriority
from *customer, order, lineitem*
where *o_orderkey* **=** *l_orderkey*
and c_custkey **=** *o_custkey*
and c_mktsegment **=***'building'*
and o_orderdate < date('1995-03-15')
and l_shipdate > date('1995-03-15')
group by *l_orderkey, o_orderdate, o_shippriority*
order by *rev desc, o_orderdate*

- **TPC-D Results(cont)**



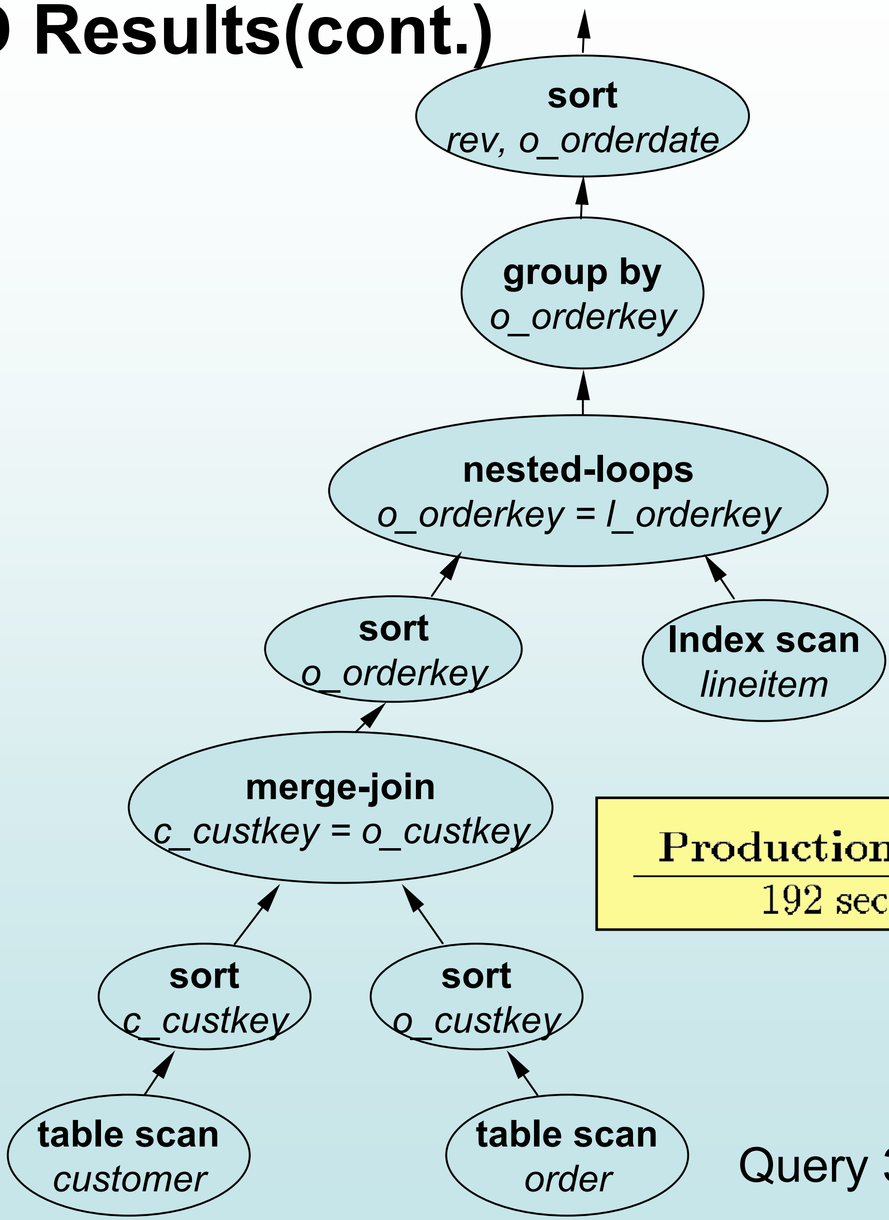
TPC-D **Query 3**

select l_orderkey,
sum (l _extendedprice(1-l_discount)) as rev,*
o_orderdate, o_shippriority
from customer, order, lineitem
where o_orderkey = l_orderkey
and c_custkey = o_custkey
and c_mktsegment ='building' and
o_orderdate < date('1995-03-15')
and l_shipdate > date('1995-03-15')
group by l_orderkey, o_orderdate, o_shippriority
order by rev desc, o_orderdate

Query 3
Order Optimization **Disabled**

Performance Results

- **TPC-D Results(cont.)**



Production DB2	Disabled DB2	Ratio
192 sec.	393 sec.	2.04

Query 3 in Production version of DB2

Conclusion

- We have described the general techniques which can be used by any query optimizer
- This can mean the difference between execution plan that finishes in few minutes versus one that takes hours to run
- Further improvements
  - Next presentation: Avoiding Sorting and **Grouping** in Processing Queries

Any Questions ?