Query Optimization

Exercise Session 2

Andrey Gubichev

October 20, 2014

Homework

- Find all professors whose lectures attended at least two students
- No Group By in TinyDB

```
select p.name
 from Professors p, Lectures v,
        Attends h1, Attends h2
 where p.persnr=v.gelesenvon
   and v.vorlnr=h1.vorlnr
   and v.vorlnr=h2.vorlnr
   and h1.matrnr<>h2.matrnr;
```

Info for Homework

C++11:

- Bjarne Stroustrup. *A Tour of C++*: Short and comprehensive reference, available in the library
- `http://en.cppreference.com`: various helpful data structures and alogrithms from Standard Template Library
- `http://isocpp.org/faq`: FAQ covering lots of topics from basics and how to get started over OOP to advanced stuff and a preview of C++14
- Please refrain from using any libraries other than the STL (and googletest for unit testing)
- tutorial on Make: `http://www.cs.umd.edu/class/fall2002/cmsc214/Tutorial/makefile.html`

Logical optimization: preliminary

Cardinality and Selectivity

Logical optimization: preliminary

Cardinality and Selectivity
Selectivity of a predicate, selectivity of a join

Logical optimization: preliminary

Cardinality and Selectivity
Selectivity of a predicate, selectivity of a join

- example of a predicate with (very) high selectivity

Logical optimization: preliminary

Cardinality and Selectivity
Selectivity of a predicate, selectivity of a join

- example of a predicate with (very) high selectivity
- (now: with joins)

Logical optimization: preliminary

Cardinality and Selectivity

Selectivity of a predicate, selectivity of a join

- example of a predicate with (very) high selectivity
- (now: with joins)
- example of a predicate with (very) low selectivity

Logical optimization: preliminary

Cardinality and Selectivity
Selectivity of a predicate, selectivity of a join

- example of a predicate with (very) high selectivity
- (now: with joins)
- example of a predicate with (very) low selectivity
- (now: with joins)

Logical optimization: preliminary

Cardinality and Selectivity
Selectivity of a predicate, selectivity of a join

- example of a predicate with (very) high selectivity
- (now: with joins)
- example of a predicate with (very) low selectivity
- (now: with joins)
- independent and correlated conditions

Logical optimization

- $|\text{Students}| = 1000$
- $|\text{Lectures}| = 100$
- $|\text{Attends}| = 5000$
- $f_{s,l} = 0.001$
- $f_{a,l} = 0.01$

Find the students that attend the course 'Ethics'

- SQL query
- canonical transformation, compute cardinalities
- push down selections, compute cardinalities

Logical optimization

```
select distinct s.name
  from Vorlesungen v, Hoeren h, Studenten s
  where v.titel='Ethik'
    and v.vorlnr=h.vorlnr
    and v.matrnr=s.matrnr
```

Cost Estimation

The goal of optimization is to minimize the cost function

Reminder: C_{out}

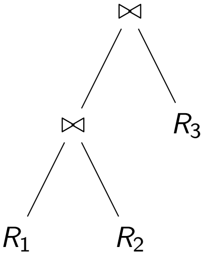
$$C_{\text{out}}(T) = \begin{cases} 0 & \text{if } T \text{ is a leaf } R_i \\ |T| + C_{\text{out}}(T_1) + C_{\text{out}}(T_2) & \text{if } T = T_1 \bowtie T_2 \end{cases}$$

Cost Estimation

The goal of optimization is to minimize the cost function

Reminder: C_{out}

$$C_{\text{out}}(T) = \begin{cases} 0 & \text{if } T \text{ is a leaf } R_i \\ |T| + C_{\text{out}}(T_1) + C_{\text{out}}(T_2) & \text{if } T = T_1 \bowtie T_2 \end{cases}$$

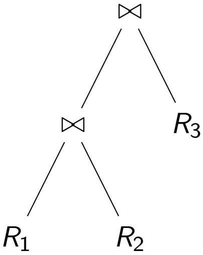


Cost Estimation

The goal of optimization is to minimize the cost function

Reminder: C_{out}

$$C_{\text{out}}(T) = \begin{cases} 0 & \text{if } T \text{ is a leaf } R_i \\ |T| + C_{\text{out}}(T_1) + C_{\text{out}}(T_2) & \text{if } T = T_1 \bowtie T_2 \end{cases}$$



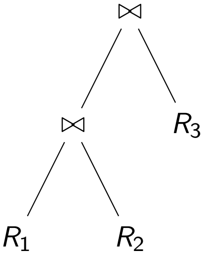
- $|R_1| = 100$
- $|R_2| = 200$
- $|R_3| = 100$
- $f_{1,2} = 0.1$
- $f_{2,3} = 0.0001$

Cost Estimation

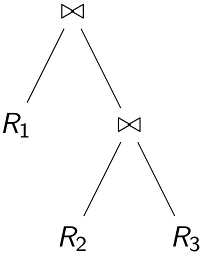
The goal of optimization is to minimize the cost function

Reminder: C_{out}

$$C_{\text{out}}(T) = \begin{cases} 0 & \text{if } T \text{ is a leaf } R_i \\ |T| + C_{\text{out}}(T_1) + C_{\text{out}}(T_2) & \text{if } T = T_1 \bowtie T_2 \end{cases}$$



- $|R_1| = 100$
- $|R_2| = 200$
- $|R_3| = 100$
- $f_{1,2} = 0.1$
- $f_{2,3} = 0.0001$

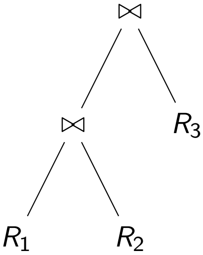


Cost Estimation

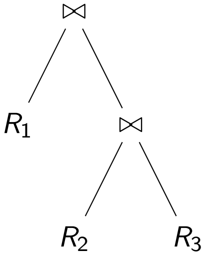
The goal of optimization is to minimize the cost function

Reminder: C_{out}

$$C_{\text{out}}(T) = \begin{cases} 0 & \text{if } T \text{ is a leaf } R_i \\ |T| + C_{\text{out}}(T_1) + C_{\text{out}}(T_2) & \text{if } T = T_1 \bowtie T_2 \end{cases}$$



- $|R_1| = 100$
- $|R_2| = 200$
- $|R_3| = 100$
- $f_{1,2} = 0.1$
- $f_{2,3} = 0.0001$



That's why we need join ordering!

Physical Optimization

the step after logical optimization

- choosing indexes or table scan
    - index vs table scan: 10% selectivity threshold
    - clustered index
    - non-clustered index
- choosing types of joins
    - nested loop join
    - block nested loop join
    - (index nested loop join)
    - merge join
    - hash join

Physical Optimization

- Courses(ID,Title,Room,Time)
- Exercises(ID,CID,TID,Room)
- Tutors(ID,Name)

```
select C.Name, T.Name, E.Room
from Courses C, Tutors T, Exercises E
where C.ID = E.CID and T.ID = E.TID
      and C.Room like '02.09.%'
      and E.Room not like '02.09.%';
```

Physical Optimization

- Courses(ID,Title,Room,Time)
- Exercises(ID,CID,TID,Room)
- Tutors(ID,Name)

```
select C.Name, T.Name, E.Room
from Courses C, Tutors T, Exercises E
where C.ID = E.CID and T.ID = E.TID
      and C.Room like ’02.09.%’
      and E.Room not like ’02.09.%’;
```

- non-clustered index on Courses.Room
- a) clustered indexes on Exercises.TID, Tutors.ID

Physical Optimization

- Courses(ID,Title,Room,Time)
- Exercises(ID,CID,TID,Room)
- Tutors(ID,Name)

```
select C.Name, T.Name, E.Room
from Courses C, Tutors T, Exercises E
where C.ID = E.CID and T.ID = E.TID
      and C.Room like ’02.09.%’
      and E.Room not like ’02.09.%’;
```

- non-clustered index on Courses.Room
- a) clustered indexes on Exercises.TID, Tutors.ID
- b) only clustered index on Tutors.ID

Query Graphs

```
select v.titel
  from Lectures v, Professors p
  where v.gelesenvon = p.persnr
    and p.name = 'Kant'
    and v.sws = 2;
```

Query Graphs

```
select r.a, s.c
  from R r, S s, T t, U u
  where r.a = s.a
    and r.b = t.b
    and r.b = u.b;
```

Query Graphs

```
select r.a, s.c
  from R r, S s
  where r.a + s.a = 7;
```

Query Graphs

```
select r.a, s.c
  from R r, S s, T t, U u
  where (r.a + s.b) = (t.b + u.a);
```

Roadmap

Good optimizer deals with the following issues:

- Cost Model
  - Cost Function Done
  - Selectivity estimation, statistics Homework
- Logical Optimization
  - Search Space Next time
  - Algorithms for Optimal Plan finding Rest of the course
- Physical Optimization
  - Enhancing the logical plan with physical operators Seen

PostgreSQL query plans

- Understanding Explain: `http://www.dalibo.org/_media/understanding_explain.pdf`

Homework: Task 1 (5 points)

Consider the TPC-H benchmark (http://www.tpc.org/tpch/) and the query:

```
select *
 from lineitem l, orders o, customers c
 where l.l_orderkey=o.o_orderkey
           and o.o_custkey=c.c_custkey
           and c.c_name='Customer#000014993'.
```

Do canonical translation and logical optimization.

Homework: Task 2 (10 points)

Given $|R1|$, $|R2|$, and sizes of domains $|R1.x|$ and $|R2.y|$ and the information if $R1.x$ and/or $R2.y$ are keys of $R1$ and $R2$

- How can we estimate the selectivity of $\sigma_{R1.x=c}$, where c is a constant?
- How can we estimate the selectivity of $\bowtie_{R1.x=R2.y}$?

Assume uniform distribution of values in all domains.
NB: we can not assume that we know the size of $\bowtie_{R1.x=R2.y}$ (the other way round, we estimate the join size using the selectivity estimation. But how to estimate the selectivity?)

Homework: Task 3 (10 points)

- Given are two relations R and S, with sizes 1,000 and 100,000 pages respectively.
- Each page has 50 tuples.
- The relations are stored on a disk, the average access time for the disk is 10 ms and the transfer speed is 10,000 pages/sec.
- **Question 1:** How long does it take to perform the Nested Loops Join of R and S?
- **Question 2:** How long does it take to perform the Block Nested Loops Join with a block size of 100 pages?
- Assume that CPU costs are negligible and ignore I/O costs for the join output.

Info

- Slides and exercises:
  http://www3.in.tum.de/teaching/ws1415/queryopt/
- Send any comments, questions, solutions for the exercises etc. to Andrey.Gubichev@in.tum.de
- Exercises due: 9 AM, November 3, 2014