Semantic Query Optimization in IBM DB2: Initial Results

Jarek Gryz Linqi Liu Xiaoyan Qian

Technical Report CS-1999-01

February 19, 1999

Department of Computer Science

4700 Keele Street North York, Ontario M3J 1P3 Canada

Semantic Query Optimization in IBM DB2: Initial Results

Jarek Gryz Linqi Liu Xiaoyan Qian

Computer Science Department, York University, Toronto, Canada, Feb. 1999

Abstract

Semantic query optimization can play a very important role in database systems. Dramatic improvements in performance may be achieved by using semantic query optimization techniques.

In this report, we present some experiments over large databases, which show the power of semantic query optimization and determine the research direction for the project. Four semantic query optimization techniques are discussed here, including join elimination, predicate introduction, detection of unsatisfiable conditions and predicate elimination. For each technique, we run queries on IBM DB2, and collect their execution times and I/Os using IBM DB2 tools. Afterwards, we optimize the queries using one of the semantic query optimization techniques. The optimization is performed only by hand, because no commercial implementations of semantic query optimization exist today.

1 Testing Beds

1.1 Environment

A Sparc SUN Ultra-1 workstation acts as the server. The operating system is UNIX Solaris 2.5.1.

1.2 Database

The DBMS that the experiments use is IBM DB2 Universal Database for the Solaris Operating Environment Version 5.0. Two databases are created according to TPC-D Benchmark 1.2.1 with the sizes of 12MB and 1GB.

1.3 Tools

To acquire comprehensive information about queries, some explain tools and monitor tools are used, including:

- db2exfmt: collect predicted information about queries,
- Snapshot monitor: collect actual I/O information about queries, and
- Event monitor: collect actual CPU information about queries.

1.4 Criteria

The following criteria are used to compare the results of queries:

For predicted costs, the criteria are Cumulative Total Cost (s) and Cumulative I/O Cost (pages). Both of them come directly from the outputs of db2exfmt.

For execution costs, the criteria are Total Cost (s) and I/O Cost (pages), where

Total Cost (s) = operation close stop time - operation prepare start time

I/O Cost (pages) = buffer pool data physical reads + buffer pool data writes +
buffer pool index physical reads + buffer pool index writes

2 Testing Samples

2.1 Join Elimination (using foreign key constraints)

When there exists a foreign key constraint between two joined tables, the system guarantees that, for each row in the child table with a non-null value in all of its foreign key columns, there is a row in the parent table with a matching value in its primary key columns. So if the purpose of join is only to check if any value in foreign key columns also appears in primary key columns, we are able to eliminate the join. This technique is called Join Elimination (using foreign key constraints).

Example 1: find the return flags and the sums of items whose ship dates are after January 1 1993.

Original query:

```
select l_returnflag, sum(l_quantity) as amount
from tpcd.lineitem, tpcd.orders, tpcd.part, tpcd.supplier
where l_orderkey = o_orderkey
and l_partkey = p_partkey
and l_suppkey = s_suppkey
and l_shipdate >= '1993-01-01'
group by l_returnflag
order by amount desc.
```

The integrity constraints are as follows:

l_orderkey is a foreign key and refers to orders,
l_partkey is a foreign key and refers to part, and
l_suppkey is a foreign key and refers to supplier.

According to the first IC, any value that appears in the l_orderkey field of the table tpcd.lineitem also appears in the o_orderkey field of the table tpcd.orders. Moreover, the fields of the table tpcd.orders neither appear in the output list of the query nor appear in other predicates except join predicates. So the join between the table tpcd.lineitem and the table tpcd.orders can be eliminated, that is, the parent table tpcd.orders can be eliminated.

According to the second IC and the third IC, we can eliminate the join between the table tpcd.lineitem and the table tpcd.part and the join between the table tpcd.lineitem and the table tpcd.supplier as well.

Optimized query:

```
select l_returnflag, sum(l_quantity) as amount
from tpcd.lineitem
where l_shipdate >= '1993-01-01'
group by l_returnflag
order by amount desc.
```

Tables 1 and 2 show the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	13.821	2596	16.504	3514
Optimized	10.689	2505	2.270	2814

Table 1: Join Elimination Example 1 (12MB DB)

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	3229.7	240935	5969.3	240884
Optimized	772.8	208305	806.2	208365

Table 2: Join Elimination Example 1 (1GB DB)

Example 2: find the order status and the total prices of orders whose order dates are after January 1 1993 and order priorities are '2-HIGH'.

Original query:

```
select o_orderstatus, sum(o_totalprice) as price
from tpcd.orders, tpcd.customer
where o_custkey = c_custkey
and o_orderdate >= '1993-01-01'
and o_orderpriority = '2-HIGH'
group by o_orderstatus
order by price desc.
```

The integrity constraint is as follows:

o_custkey is a foreign key and refers to customer.

According to join elimination, the join between the table tpcd.orders and the table tpcd.customer can be eliminated.

Optimized query:

```
select o_orderstatus, sum(o_totalprice) as price
from tpcd.orders
where o_orderdate >= '1993-01-01'
and o_orderpriority ='2-HIGH'
group by o_orderstatus
order by price desc.
```

Tables 3 and 4 show the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	2.507	541	2.120	487
Optimized	2.292	534	0.619	322

Table 3: Join Elimination Example 2 (12MB DB)

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	204.7	44864	194.7	44907
Optimized	177.9	44408	130.6	44446

Table 4: Join Elimination Example 2 (1GB DB)

2.2 Join Elimination (detection of empty join)

Sometimes the results of some joins are empty in a database. If we find them in advance, we may optimize the original query by eliminating the empty joins.

Example 3: find the keys, the names and the account balances of customers who make orders after January 1 1993 and have the same names as those of suppliers.

Original query:

```
select c_custkey, c_name, c_acctbal
from tpcd.supplier, tpcd.customer, tpcd.orders, tpcd.lineitem
where c_custkey = o_custkey
and o_orderdate >= '1993-01-01'
and o_orderkey = l_orderkey
and l_returnflag = 'R'
and s_name = c_name.
```

The integrity constraint is as follows:

No customers have the same names as those of suppliers.

According to the IC, the predicate s_name = c_name is not satisfied by the database. So the result of the join between the table tpcd.supplier and the table tpcd.customer must be empty, that is, the result of the query must be empty. In this case, the query doesn't need to be evaluated. But in current commercial DBMSs, the query is evaluated, and then we can find that the result of the query is empty.

Tables 5 and 6 show the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	9.134	2098	0.320	87
Optimized	0.000	0	0.000	0

Table 5: Join Elimination Example 3 (12MB DB)

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	207.0	59851	39.3	7256
Optimized	0.0	0	0.0	0

Table 6: Join Elimination Example 3 (1GB DB)

2.3 Predicate Introduction (using index introduction)

We sometimes may infer from integrity constraints and some existing predicates in a query, that some new predicates also must be satisfied. If there are some indices on the new predicates, we may introduce them into the query to speed up the evaluation of the query. We call it Predicate Introduction (using index introduction).

Example 4: find the revenues of items whose extended prices are less than or equal to 1K and discounts are less than or equal to 0.05 and receipt dates are before January 1 1997.

Original query:

```
select sum(l_extendedprice*(1-l_discount)) as revenue
from tpcd.lineitem
where l_extendedprice <= 1000.0
and l_discount <= 0.05
and l_receiptdate <= '1997-01-01'.
```

The integrity constraint is as follows:

l_shipdate <= l_receiptdate

In a TPC-D database, there is an index on the fields l_shipdate + l_extendedprice + l_discount + l_suppkey + l_partkey. But we cannot make use of the index to evaluate the original query efficiently because l_shipdate, the first item of the index, is not used by any predicate. Note that l_receiptdate <= '1997-01-01' is a predicate of the original query, so the IC l_shipdate <= l_receiptdate implies l_shipdate <= '1997-01-01'. According to predicate introduction, we can introduce a new predicate l_shipdate <= '1997-01-01' into the original query. Now we can use the above index to speed up the evaluation of the query.

Optimized query:

```
select sum(l_extendedprice*(1-l_discount)) as revenue
from tpcd.lineitem
where l_shipdate <= '1997-01-01'
and l_extendedprice <= 1000.0
and l_discount <= 0.05
and l_receiptdate <= '1997-01-01'.
```

Tables 7 and 8 show the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	6.372	824	6.502	907
Optimized	4.657	596	1.370	668

Table 7: Predicate Introduction Example 4 (12MB DB)

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	832.8	208305	503.0	208365
Optimized	681.2	53862	361.5	52420

Table 8: Predicate Introduction Example 4 (1GB DB)

2.4 Predicate Introduction (using scan reduction)

Similar to Index Introduction, we sometimes may introduce some new predicates into a query. The new predicates reduce search range for some fields thereby improving the performance. The technique is called Predicate Introduction (using scan reduction).

Example 5: find the return flags, the line status and the sums of items whose order dates are after October 1 1995 and ship dates are before October 7 1995.

Original query:

```
select l_returnflag, l_linestatus, sum(l_quantity) as sum_aty
from tpcd.lineitem, tpcd.orders
where l_orderkey = o_orderkey
and o_orderdate >= '1995-10-01'
and l_shipdate <= '1995-10-07'
group by l_returnflag, l_linestatus
order by l_returnflag, l_linestatus.
```

The integrity constraint is as follows:

```
o_orderdate <= l_shipdate
```

The IC o_orderdate <= l_shipdate and the predicate o_orderdate >= '1995-10-01' imply l_shipdate >= '1995-10-01'. If we introduce l_shipdate >= '1995-10-01' as a new predicate, we can reduce so much search range for the field l_shipdate.

Optimized query:

```
select l_returnflag, l_linestatus, sum(l_quantity) as sum_aty
from tpcd.lineitem, tpcd.orders
where l_orderkey = o_orderkey
and o_orderdate >= '1995-10-01'
and l_shipdate <= '1995-10-07'
and l_shipdate >= '1995-10-01'
group by l_returnflag, l_linestatus
order by l_returnflag, l_linestatus.
```

Table 9 shows the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	11.347	2567	2.759	3680
Optimized	1.066	184	0.460	239

Table 9: Predicate Introduction Example 5 (12MB DB)

2.5 Detection of Unsatisfiable Conditions

In the WHERE clause of a query, some conditions may conflict with integrity constraints or be self-contradictory. So the result of query must be empty. If we may detect the conditions early, we will be able to avoid evaluating the empty query. We call it Detection of Unsatisfiable Conditions.

Example 6: find the keys, the order dates and the ship priorities of orders whose total prices are greater than or equal to 65K and order dates are after January 1 1994 and commit dates are before July 1 1993.

Original query:

```
select o_orderkey, o_orderdate, o_shippriority
from tpcd.orders, tpcd.lineitem
where o_orderkey = l_orderkey
and o_totalprice >= 65000.0
and o_orderdate > '1994-01-01'
and l_commitdate < '1993-07-01'
order by o_orderdate.
```

The integrity constraint is as follows:

o_orderdate <= l_commitdate

According to the IC, the predicate o_orderdate > '1994-01-01' and the predicate l_commitdate < '1993-07-01' cannot be satisfied simultaneously. So the result of query will be empty. We need not evaluate the query. But in current commercial DBMSs, the unsatisfiable condition cannot be found in advance and the query will be evaluated.

Table 10 shows the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	11.478	2665	1.980	3492
Optimized	0.000	0	0.000	0

Table 10: Detection of Unsatisfiable Conditions Example 6 (12MB DB)

Example 7: find the revenues of items whose ship dates are between January 1 1994 and January 1 1995 and discount are between 0.05 and 0.07 and quantities are greater than 0 and less than -24.

Original query:

```
select sum(l_extendedprice*l_discount) as revenue
from tpcd.lineitem
where l_shipdate >= '1994-01-01'
and l_shipdate < '1995-01-01'
and l_discount between 0.05 and 0.07
and l_quantity < -24
and l_quantity > 0.
```

In the WHERE clause, there exists an apparent contradiction, l_quantity < -24 and l_quantity > 0. But IBM DB2 cannot find out the contradiction in advance so it will evaluate the query. Using the semantic query optimization - Detection of Unsatisfiable Conditions, it will be easy to solve this problem.

Table 11 shows the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	10.717	2505	1.490	3288
Optimized	0.000	0	0.000	0

Table 11: Detection of Unsatisfiable Conditions Example 7 (12MB DB)

2.6 Predicate Elimination

From integrity constraints, we sometimes may infer that some predicates always hold for a query. So we can eliminate the redundant predicates to improve the performance if no index is on the predicates. This technique is called Predicate Elimination.

Example 8: find the return flags and the sums of items whose order dates are after January 1 1994 and commit dates are after January 1 1992 and receipt dates are after June 1 1992 and taxes are greater than or equal to 0 and total prices of orders are greater than or equal to 0.

Original query:

```
select l_returnflag, sum(l_quantity) as sum_quantity
from tpcd.lineitem, tpcd.orders
where l_orderkey = o_orderkey
and o_orderdate >= '1994-01-01'
and l_commitdate >= '1992-01-01'
and l_receiptdate >= '1992-06-01'
and l_tax >= 0.0
and o_totalprice >= 0.0
group by l_returnflag
order by sum_quantity desc.
```

The integrity constraints are as follows:

```
o_orderdate <= l_commitdate
o_orderdate <= l_shipdate
l_shipdate <= l_receiptdate
l_tax >= 0.0
o_totalprice >= 0.0
```

The IC o_orderdate <= l_commitdate and the predicate o_orderdate >= '1994-01-01' imply l_commitdate >= '1994-01-01'. So the predicate l_commitdate >= '1992-01-01' is redundant and we can eliminate it according to predicate elimination. Please note that there is not any index on the field l_commidate in a TPC-D database so the elimination will speed up the evaluation of the query definitely.

Optimized query 1:

```
select l_returnflag, sum(l_quantity) as sum_quantity
from tpcd.lineitem, tpcd.orders
where l_orderkey = o_orderkey
and o_orderdate >= '1994-01-01'
and l_commitdate >= '1992-01-01'
and l_receiptdate >= '1992-06-01'
and l_tax >= 0.0
and o_totalprice >= 0.0
group by l_returnflag
order by sum_quantity desc.
```

Similarly, the IC o_orderdate <= l_shipdate and the IC l_shipdate <= l_receiptdate imply o_orderdate <= l_receiptdate. So we know l_receiptdate >= '1994-01-01', that is, the predicate l_receiptdate >= '1992-06-01' is redundant. Eliminate the predicate.

Optimized query 2:

```
select l_returnflag, sum(l_quantity) as sum_quantity
from tpcd.lineitem, tpcd.orders
where l_orderkey = o_orderkey
and o_orderdate >= '1994-01-01'
and l_commitdate >= '1992-01-01'
and l_receiptdate >= '1992-06-01'
and l_tax >= 0.0
and o_totalprice >= 0.0
group by l_returnflag
order by sum_quantity desc.
```

According to the IC l_tax >= 0.0, we know that l_tax >= 0.0 holds for the database. Moreover, there is not any index on the field l_tax in the database. So the predicate l_tax >= 0.0 should be eliminated.

Optimized query 3:

```
select l_returnflag, sum(l_quantity) as sum_quantity
from tpcd.lineitem, tpcd.orders
where l_orderkey = o_orderkey
and o_orderdate >= '1994-01-01'
and l_commitdate >= '1992-01-01'
and l_receiptdate >= '1992-06-01'
and l_tax >= 0.0
and o_totalprice >= 0.0
group by l_returnflag
order by sum_quantity desc.
```

Similarly, the predicate o_totalprice >= 0.0 is eliminated.

Optimized query 4:

```
select l_returnflag, sum(l_quantity) as sum_quantity
from tpcd.lineitem, tpcd.orders
```

```
where l_orderkey = o_orderkey
and o_orderdate >= '1994-01-01'
and l_commitdate >= '1992-01-01'
and l_receiptdate >= '1992-06-01'
and l_tax >= 0.0
and o_totalprice >= 0.0
group by l_returnflag
order by sum_quantity desc.
```

Table 12 shows the execution times and I/Os of the original query and the optimized query.

Queries	Predicted Cost		Execution Cost	
	Total Cost (s)	I/O Cost (pages)	Total Cost (s)	I/O Cost (pages)
Original	15.052	3218	8.342	3399
Optimized 1	14.907	3218	8.131	3360
Optimized 2	14.770	3218	7.592	3310
Optimized 3	14.686	3218	7.101	3302
Optimized 4	13.144	2785	7.022	2831

Table 12: Predicate Elimination Example 8 (12MB DB)

3 Lessons Learned

Based on the above experiment results, we have ranked several types of semantic query optimization techniques with respect to the estimated difficulty of their implementation and with respect to the expected improvement in query execution time.

Join Elimination

- very good optimization
- easy to implement (for primary-foreign key joins)
- relatively common

Predicate Introduction

- good optimization
- medium difficulty to implement
- more work needed to specify precisely when it should be applied

Detection of Unsatisfiable Conditions

- very good optimization
- moderate difficulty to implement
- not likely to be common

Predicate Elimination

- ♦ medium optimization
- ♦ easy to implement
- ♦ common

Table 13 summarizes these estimations.

Techniques	Implementation Difficulty	When Applicable	How often applicable
Join Elimination	L (H)	VH	M
Predicate Introduction	M	H	L
Detection of Unsatisfiable Conditions	M	VH	L
Predicate Elimination	L	M	H

Table 13: Ranking of Semantic Query Optimization Techniques

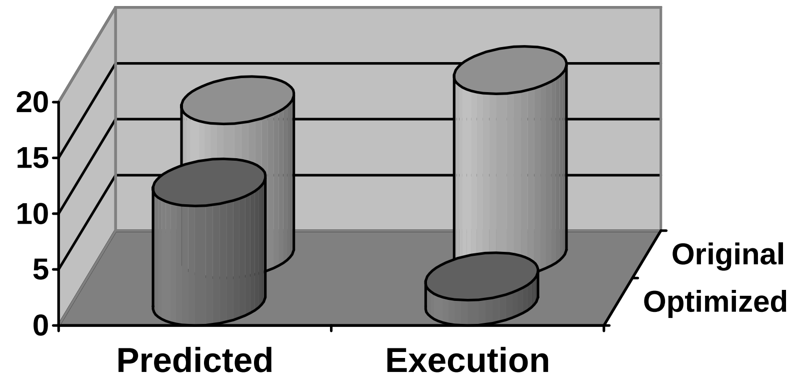

Chart 1: Join Elimination Example 1 (12MB DB) – Total Cost (s)

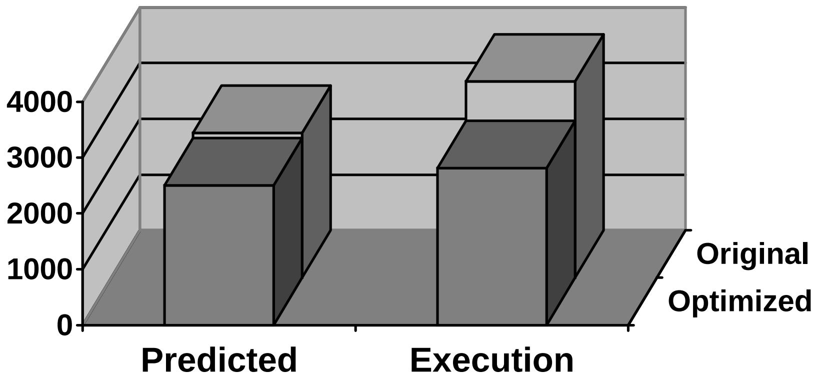

Chart 2: Join Elimination Example 1 (12MB DB) – I/O Cost (Pages)

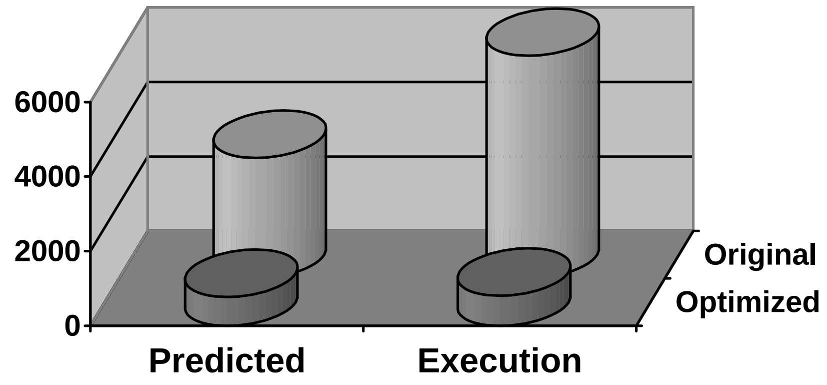

Chart 3: Join Elimination Example 1 (1GB DB) – Total Cost (s)

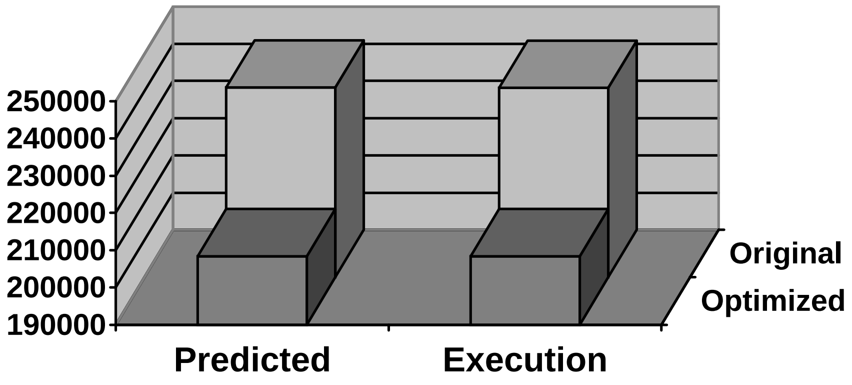

Chart 4: Join Elimination Example 1 (1GB DB) – I/O Cost (Pages)

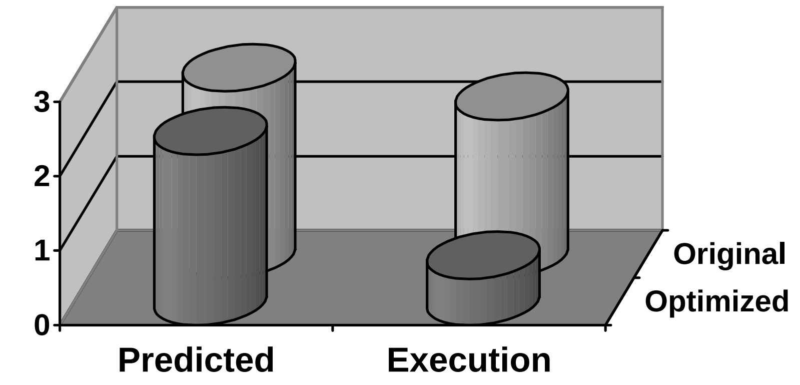

Chart 5: Join Elimination Example 2 (12MB DB) – Total Cost (s)

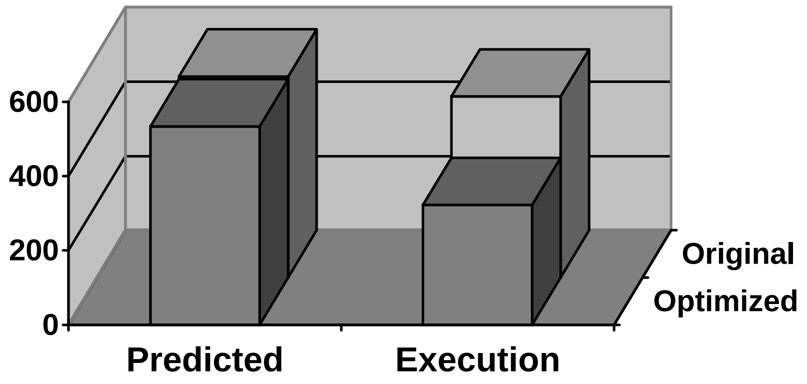

Chart 6: Join Elimination Example 2 (12MB DB) – I/O Cost (Pages)

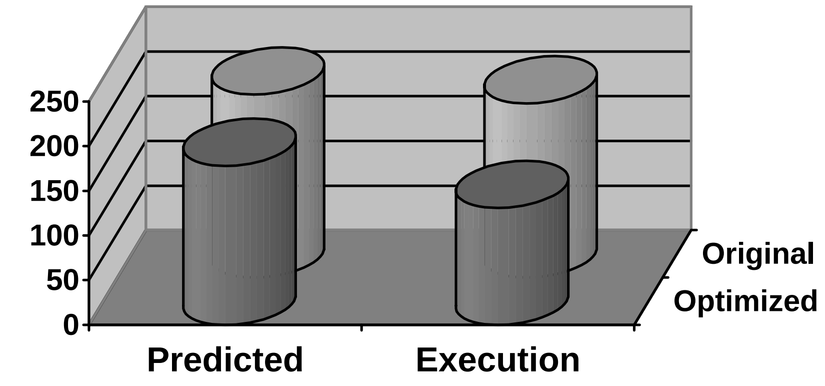

Chart 7: Join Elimination Example 2 (1GB DB) – Total Cost (s)

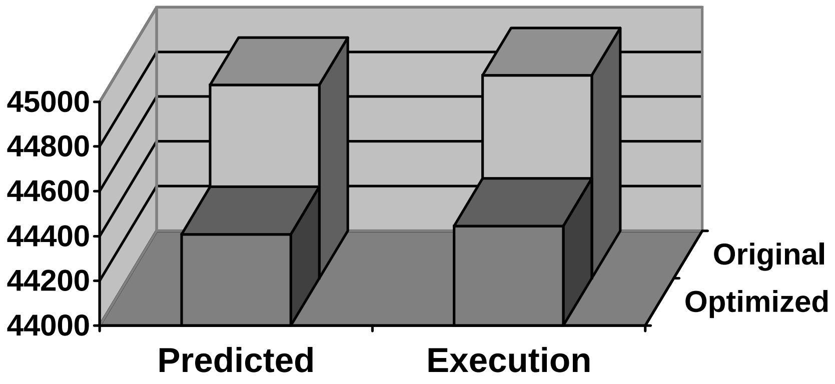

Chart 8: Join Elimination Example 2 (1GB DB) – I/O Cost (Pages)

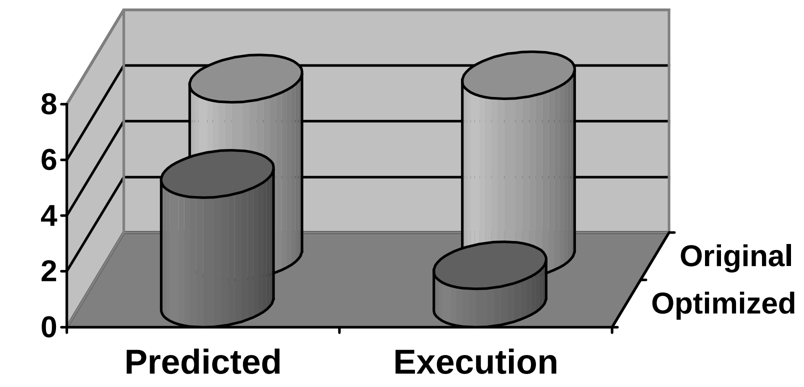

Chart 9: Predicate Introduction Example 4 (12MB DB) – Total Cost (s)

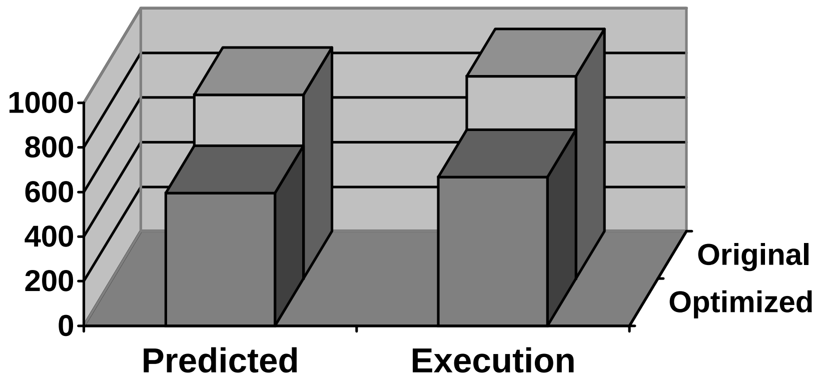

Chart 10: Predicate Introduction Example 4 (12MB DB) – I/O Cost (Pages)

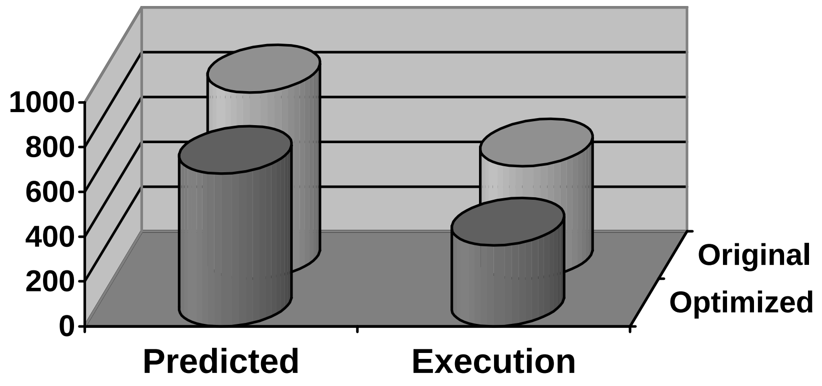

Chart 11: Predicate Introduction Example 4 (1GB DB) – Total Cost (s)

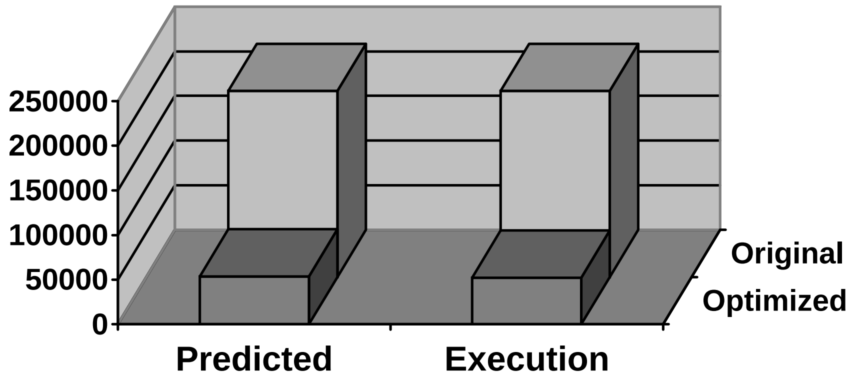

Chart 12: Predicate Introduction Example 4 (1GB DB) – I/O Cost (Pages)

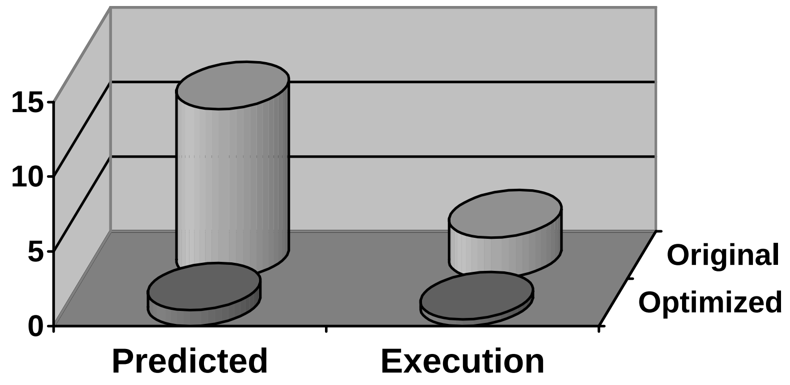

Chart 13: Predicate Introduction Example 5 (12MB DB) – Total Cost (s)

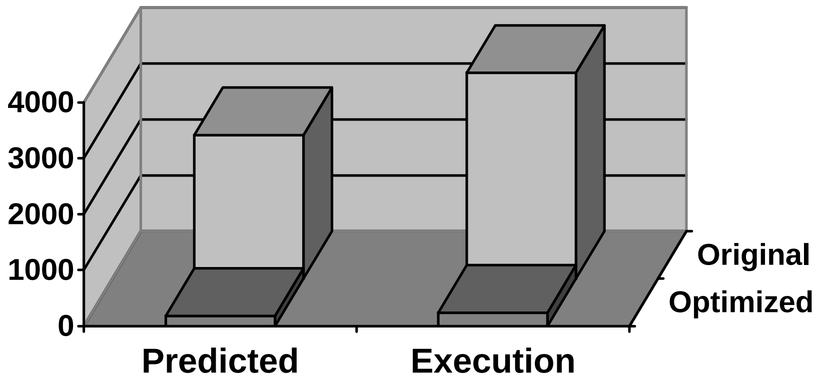

Chart 14: Predicate Introduction Example 5 (12MB DB) – I/O Cost (Pages)

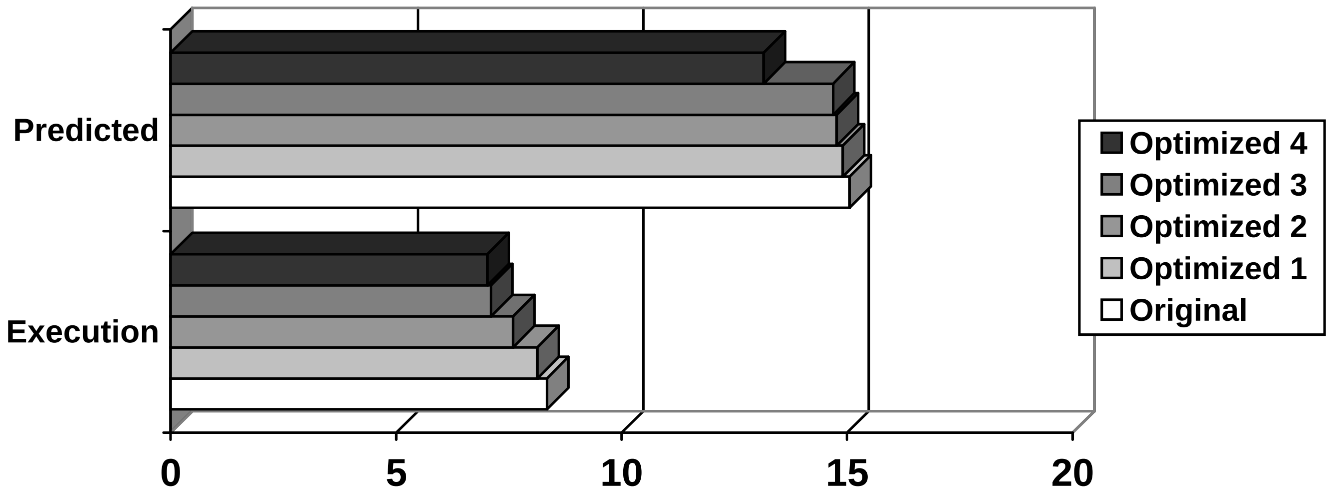

Chart 15: Predicate Elimination Example 8 (12MB DB) – Total Cost (s)

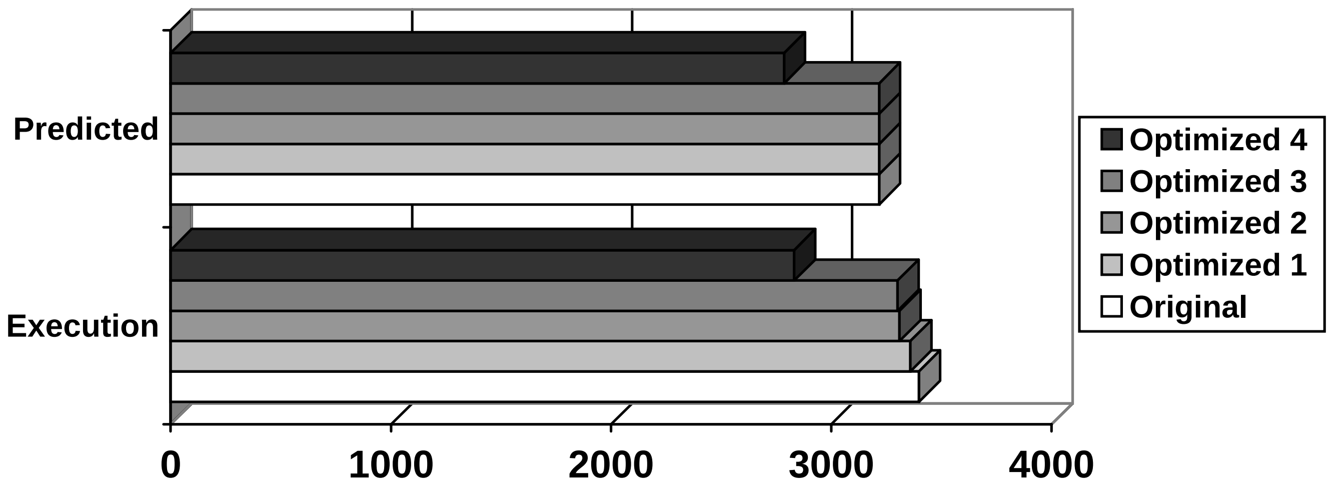

Chart 16: Predicate Elimination Example 8 (12MB DB) – I/O Cost (Pages)