

Same logical expression, different physical algorithms

```
SELECT *
FROM Order o, Item i
WHERE o.order = i.order

Scan
Item i
Order o
```

Option 1

Which is faster?

for each record i in Item:
for each record o in Order:
if o.order = i.order:
return (r,s)

for each record i in Item: insert into hashtable

Option 2

for each record o in Order: lookup corresponding records in hashtable return matching pairs

Exposing the Algebra: Microsoft SQL Server

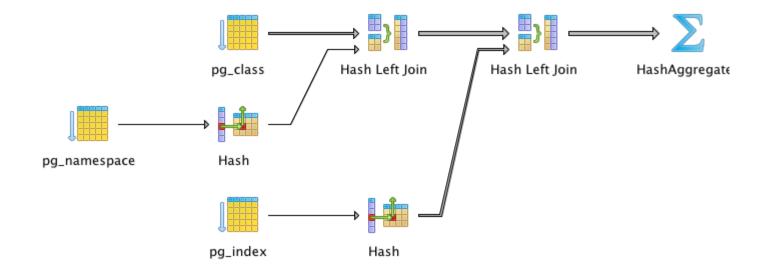
```
select a.term id, b.term id
2 from [billhowe].[reuters] a, [billhowe].[reuters] b
3 where a.doc id = b.doc id
     and a.term id != b.term id
             EXPLAIN
                               Clustered Index Scan (Clustered)
                              [table_reuters_terms.csv].[IX_table...
          (Inner Join)
          Cost: 94 %
                               Clustered Index Scan (Clustered)
                              [table reuters terms.csv].[IX table...
```

Exposing the Algebra: Microsoft SQL Server

```
select a.term id, b.term id
              from [billhowe].[reuters] a, [billhowe].[reuters] b
            3 where a.doc id = b.doc id
               and a.term id != b.term id
                and a.term id = 'parliament'
                       EXPLAIN
                Nested Loops
                                       Clustered Index Scan (Clustered)
 SELECT
                                     [table_reuters_terms.csv].[IX_table...
                 (Inner Join)
Cost: 0 %
                  Cost: 7 %
                                      Clustered Index Seek (Clustered)
                                     [table_reuters_terms.csv].[IX_table...
```

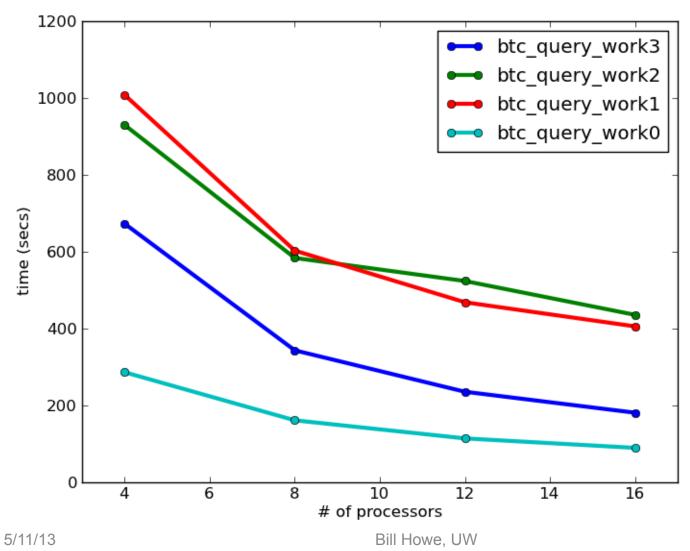
Exposing the Algebra: PostgreSQL

EXPLAIN SELECT



screenshot from pgAdmin3

Algebraic Optimization Matters



BTC 2010 Dataset

3B quads 623 GB processed

5

