The DBO Database System

Florin Rusu, Fei Xu, Luis Perez,
Mingxi Wu, Ravi Jampani,
Chris Jermaine, Alin Dobra
University of Florida, Gainesville

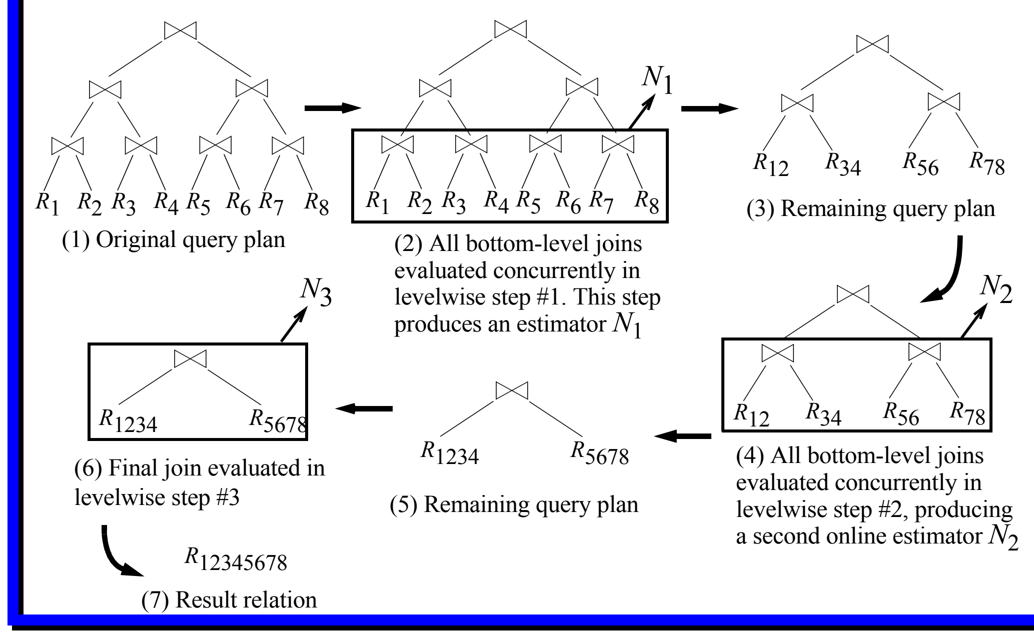
What is DBO?

- DBO (version 0.2) is a prototype database engine for analytic, statistical processing
- Key innovations:
  - Within seconds after query is issued, DBO gives statistically valid guess + bounds
  - Accuracy increases as query is executed; 100% accuracy at query completion
  - Works for arbitrary `SELECT-FROM-WHERE-GROUP BY` aggregate queries
  - For some queries (almost all single-table scans) 99%+ accuracy after only seconds
- Key idea: Happy with the current estimate? Then kill the query!
- DBO extends "classic" online aggregation to full, disk-based query plans; see our SIGMOD '07 paper

How Does DBO Work?

- Data are clustered randomly on the disk, so tuples flow through engine in random order
- During processing, DBO finds "lucky" output tuples whose parts happen to be in memory
- DBO uses those "lucky" tuples that it finds to guess final answer to the query
- Example: `SUM` $_{l\_price}$ (*lineitem* `JOIN` *orders* `ON` *l_okey* = *o_okey* AND *o_shipdate* > '1-1-97')
  - Happen to have ($12.82, 1234) from *lineitem*, (1234, '2-12-98') from *orders* in memory
  - So if probability of finding ($12.82, 1234, 1234, '2-12-98') is *p*, add (12.82 / *p*) to estimate
- By statistically characterizing what "lucky" means, can provide confidence bounds on estimate

Levelwise QP in DBO


R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8
N_1
R_{12} R_{34} R_{56} R_{78}
N_2
N_3
R_{1234} R_{5678}
$R_{12345678}$

(1) Original query plan

(2) All bottom-level joins evaluated concurrently in levelwise step #1. This step produces an estimator N_1

(3) Remaining query plan

(4) All bottom-level joins evaluated concurrently in levelwise step #2, producing a second online estimator N_2

(5) Remaining query plan

(6) Final join evaluated in levelwise step #3

(7) Result relation

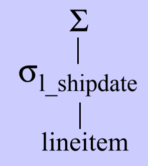
- To search for output tuples, operations communicate their internal state with one another
- Recognizing output tuples generally requires data from all input relations
- Thus, all relational operations at each level of the query plan search for lucky tuples in a coordinated fashion
- Called a ***Levelwise Step***
- Each levelwise step produces an estimate N_i
- Weighted estimate $w_1N_1 + w_2N_2 + \ldots$ is given to user

The Demonstration

- We have prepared five queries over TPC-H benchmark database (scale factor six)
- For comparison, have two identical machines; one running Postgres, one running DBO

```
SELECT  l_returnflag, l_linestatus,
  sum(l_extendedprice*(1-l_discount) * (1+l_tax))
FROM lineitem
WHERE l_shipdate < '1998-09-01'
GROUP BY l_returnflag, l_linestatus
```

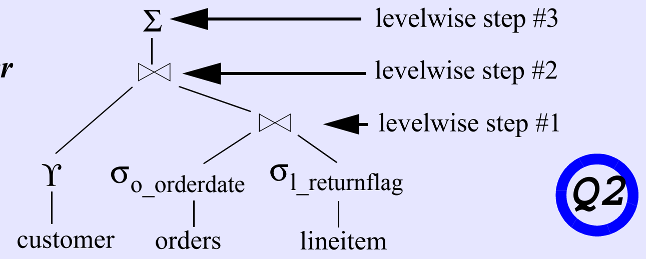
"Find the total amount charged to customers, grouped by order and return status"



Q1

```
SELECT n_name, sum(l_extendedprice * (1-l_discount))
FROM customer, orders, lineitem, nation
WHERE c_custkey = o_custkey
 AND l_orderkey = o_orderkey AND l_returnflag = 'R'
 AND (o_orderdate < '1994-01-01')
 AND (o_orderdate > '1993-09-30')
 AND c_nationkey = n_nationkey
GROUP BY n_name
```

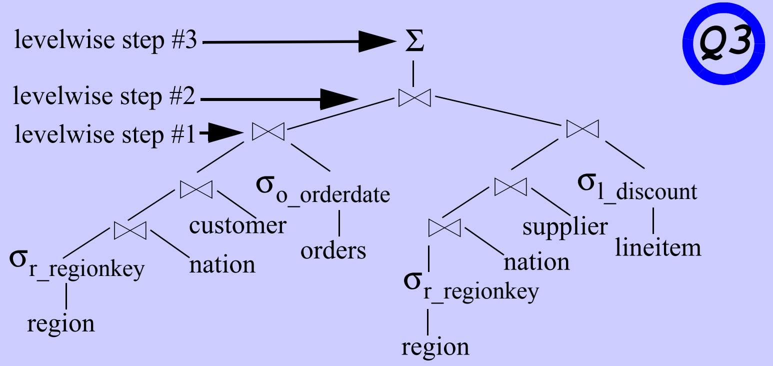
"Find the revenue lost per country due to parts that have been returned"



Q2

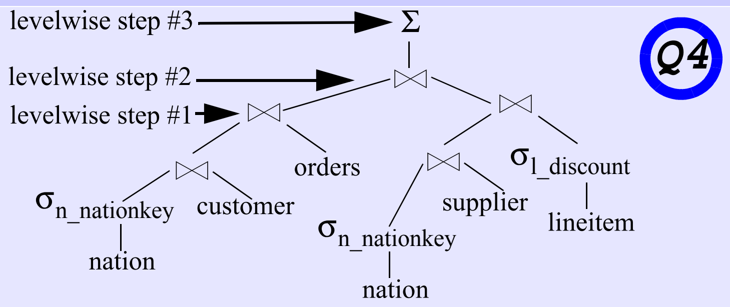
```
SELECT n_name, sum(l_extendedprice * (1 - l_discount))
FROM customer, orders, lineitem, supplier, nation, region
WHERE c_custkey=o_custkey AND l_orderkey = o_orderkey
 AND l_suppkey = s_suppkey
 AND c_nationkey = s_nationkey
 AND s_nationkey = n_nationkey
 AND n_regionkey = r_regionkey
 AND l_discount > 0.08
 AND r_name = 'ASIA'
 AND (o_orderdate > '1993-12-31')
 AND (o_orderdate < '1995-01-01')
GROUP BY n_name
```

"Find the revenue from heavily discounted parts in 1994 that were sold to a customer in the Asian region, on a per-country basis"



Q3

```
SELECT n1.nationname, n2.nationname, extract(year from
  l_shipdate) as l_year, SUM(l_extendedprice * (1 - l_discount))
FROM supplier, lineitem, orders, customer, nation n1, nation n2
WHERE s_suppkey = l_suppkey AND o_orderkey = l_orderkey
 AND c_custkey = o_custkey AND s_nationkey = n1.n_nationkey
 AND c_nationkey = n2.n_nationkey
 AND ((n1.n_name = 'GERMANY' AND n2.n_name = 'FRANCE')
   OR (n1.n_name = 'FRANCE' AND n2.n_name = 'GERMANY'))
 AND (l_shipdate < '1997-01-01')
 AND (l_shipdate > '1995-01-01') AND l_discount > 0.08
GROUP BY n1.nationname, n2.nationname, l_year
```

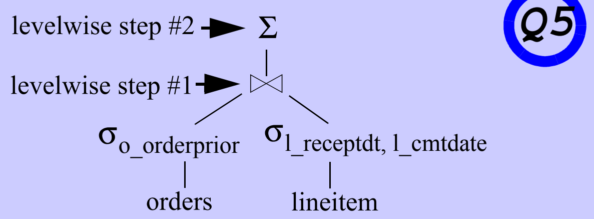


Q4

"Find the total revenue for heavily discounted items shipped between Germany and France in 1995 and 1996"

```
SELECT l_shipmode, extract(year from l_shipdate) as l_year, sum(1)
FROM orders, lineitem
WHERE o_orderkey = l_orderkey AND o_orderprior > '1-URGENT'
 AND (l_receiptdate > '1993-12-31')
 AND (l_receiptdate < '1995-01-01')
 AND (l_commitdate < l_receiptdate)
 AND (l_shipdate < l_commitdate)
GROUP BY l_shipmode, l_year
```

"Find the number of shipments (grouped by shipment mode and ship year) recevied in 1994 that did not have high priority"



Q5

- DBO has a simple, GUI front end that shows estimation progress over time (like classic online agg. interface)
- Complete re-write of DBO has recently begun... our ultimate goal is to scale to multiple terabytes, be as fast (or faster) than any existing system, and give statistically valid guess the whole way!