**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_extendedprice) JOIN orders ON l_orderkey = o_orderkey AND l_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**

- 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 $N_1$ weighted estimate $w_1 N_1 + w_2 N_2 + \ldots$ is given to user

**The DBO Database System**

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

**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 1_returnflag, 1_linenumber, sum(l_extendedprice * (1 - l_discount)) FROM orders, lineitem WHERE l_shipdate < '1989-09-01' GROUP BY 1_returnflag, 1_linenumber**

```
sum(l_extendedprice * (1 - l_discount))
```

**SELECT 1_linenumber, sum(l_extendedprice * (1 - l_discount)) FROM orders, lineitem WHERE l_shipdate < '1991-09-01' GROUP BY 1_linenumber**

```
sum(l_extendedprice * (1 - l_discount))
```

**SELECT 1_linenumber, sum(l_extendedprice * (1 - l_discount)) FROM orders, lineitem WHERE l_shipdate < '1992-09-01' GROUP BY 1_linenumber**

```
sum(l_extendedprice * (1 - l_discount))
```

**SELECT 1_linenumber, sum(l_extendedprice * (1 - l_discount)) FROM orders, lineitem WHERE l_shipdate < '1993-09-01' GROUP BY 1_linenumber**

```
sum(l_extendedprice * (1 - l_discount))
```

**SELECT 1_linenumber, sum(l_extendedprice * (1 - l_discount)) FROM orders, lineitem WHERE l_shipdate < '1994-09-01' GROUP BY 1_linenumber**

```
sum(l_extendedprice * (1 - l_discount))
```

**SELECT 1_linenumber, sum(l_extendedprice * (1 - l_discount)) FROM orders, lineitem WHERE l_shipdate < '1995-09-01' GROUP BY 1_linenumber**

```
sum(l_extendedprice * (1 - l_discount))
```

**SELECT 1_linenumber, sum(l_extendedprice * (1 - l_discount)) FROM orders, lineitem WHERE l_shipdate < '1996-09-01' GROUP BY 1_linenumber**

```
sum(l_extendedprice * (1 - l_discount))
```

**DBO does 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!