The Star Schema Benchmark and Augmented Fact Table Indexing

Pat & Betty O’Neil, Xuedong Chen and Stephen Revilak

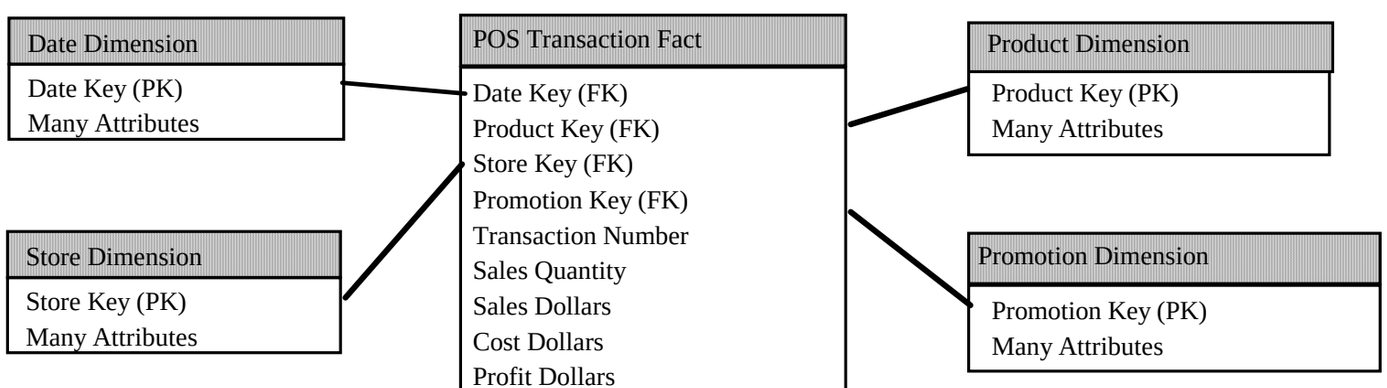
TPC Technology Conference 8/24/09

Outline of Talk

- Star Schema Benchmark
- Clustering crucial to performance on modern disks
- Good DB2 Multi-Dimensional Clustering (MDC)
- Dices fact table on “column axes” only in fact table
- Adjoin copies of Dimension Columns to fact table as axes of MDC, for crucial speed-up of SSB queries
- Can create MDC-like cells on well-indexed DBMS
- All products need mods of query restrictions to work

Star Schema (SS): Data Warehouse

- A Data Warehouse is a Query-mostly Database, typically made up of *multiple* Star Schemas, sometimes called Data Marts
- A Star Schema has a Fact table with simple joins to various ancillary Dimension tables
- Here is the pattern of a Point Of Sale (POS) Star Schema
- ***Snowflake Schema has outlier tables to the dimension tables



Star Schema (SS)

- POS Transaction fact table has 9 columns, ~40 byte rows
- A moderate size POS table on an inexpensive PC might need a few hundred GB of disk, which would amount to 5 billion rows
- Dimension tables are much smaller: few thousand rows in Date & Store Dimensions, 100s in Promotion, millions(?) in Product
- Practitioners keep fact tables lean: Foreign Keys and Measures; but many descriptive columns in Dimension Tables

Star Schema: Data Warehouse

- A Data Warehouse is made up of a collection of Star Schemas often representing a supply chain on conforming dimensions[1]

[1]R. Kimball & M. Ross, The Data Warehouse Toolkit [pgs. 78 & ff.], 2nd Ed., Wiley

	CONFORMING DIMENSIONS								
SUPPLY CHAIN STAR SCHEMAS	Date	Product	Store	Promotion	Warehouse	Vendor	Contract	Shipper	Etc.
Retail Sales	X	X	X	X					
Retail Inventory	X	X	X						
Retail Deliveries	X	X	X						
Warehouse Inventory	X	X			X	X			
Warehouse Deliveries	X	X			X	X			
Purchase Orders	X	X			X	X	X	X	

DB Performance: Star Schema Queries

- Queries on Star Schemas typically retrieve aggregates of Fact table measures (Sales Quantity/Dollars/Cost/Profit)
- Query Where Clauses typically restrict Dimension Columns: Product Category, Store Region, Month, etc.
- Usually some Group By for Aggregation
- Query 2.1 from Star Schema Benchmark (SSB) below

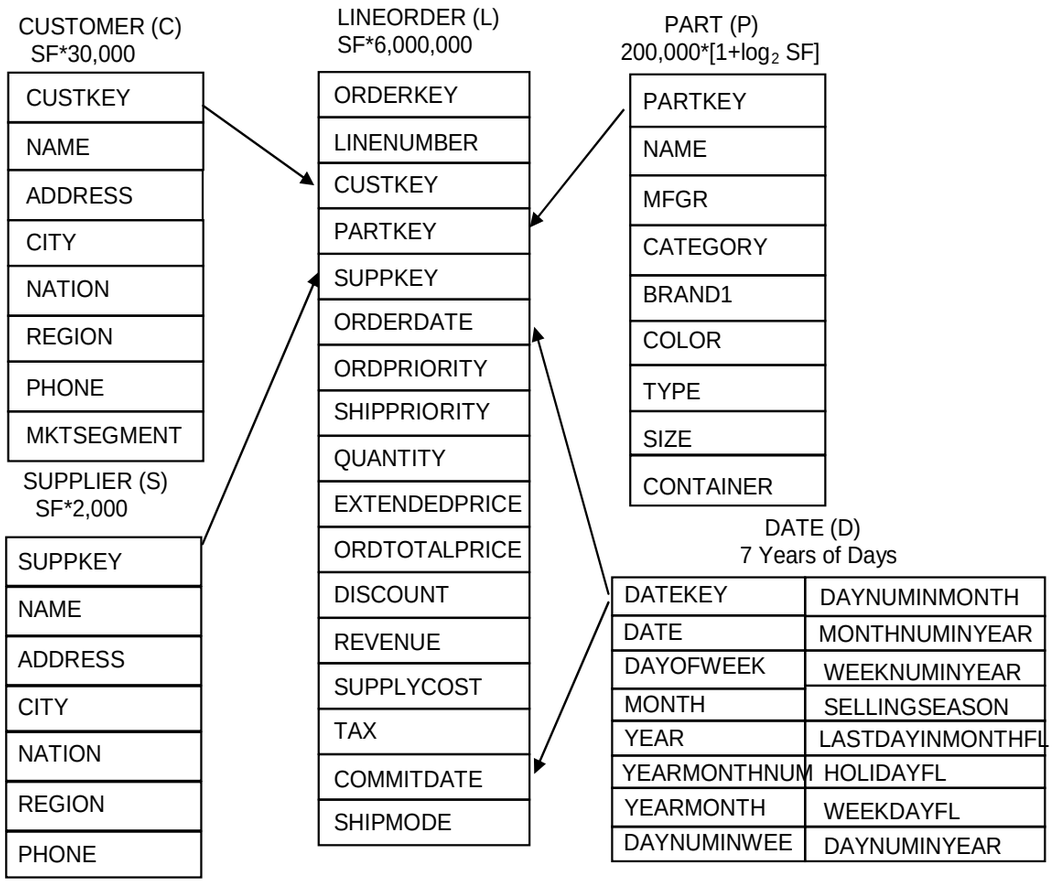
```
select sum(lo_revenue), d_year, p_brand1
    from lineorder, date, part, supplier
    where lo_orderdate = d_datekey and lo_partkey = p_partkey
        and lo_suppkey = s_suppkey and p_category = 'MFGR#12'
        and s_region = 'AMERICA'
    group by d_year, p_brand1 order by d_year, p_brand1;
```

Star Schema Benchmark (SSB)

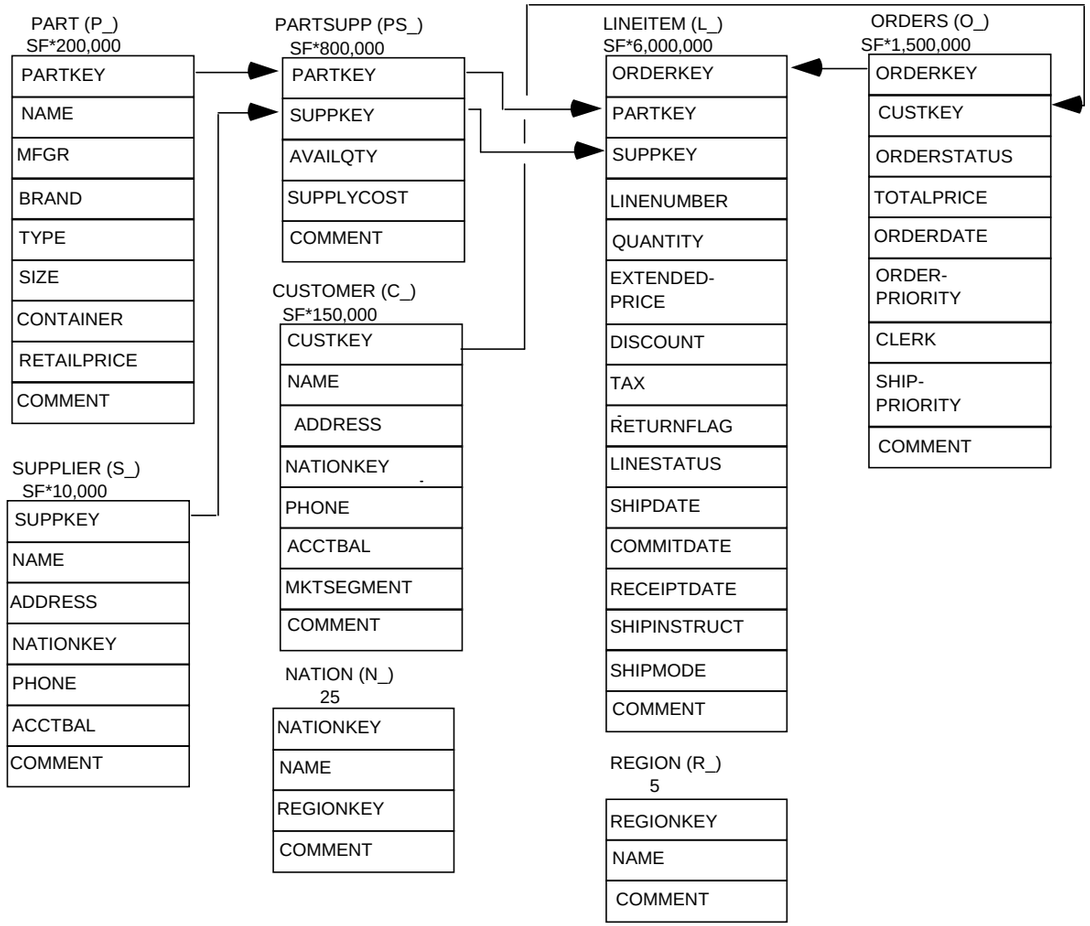
- Stonebraker's Vertica commissioned us to develop a Star Schema Benchmark (SSB)[1] to measure query performance
- The SSB design is based on TPC-H benchmark[2]
- TPC-H has joins. Quote: "[two dozen CIOs have] never seen a data warehouse that didn't use a Star/Snowflake schema"[3]
- Future TPC-DS (Decision Support) benchmark is Snowflake
- Kimball[4] argues cogently a Star Schema is better than Snowflake: Complex Snowflake makes users, optimizers, browsing, bitmap indexing struggle; saves little space in small dimension tables

[1]P. O'Neil, E. O'Neil, X. Chen. http://www.cs.umb.edu/~poneil/StarSchemaB.PDF
[2]TPC-H, an ad-hoc decision support benchmark. http://www.tpc.org/tpch/
[3]M. Stonebraker et al., One Size Fits All? — Part 2: Benchmarking Results. http://www-db.cs.wisc.edu/cidr/cidr2007/index.html, press 'Electronic Proceedings' to download
[4]R. Kimball & M. Ross, The Data Warehouse Toolkit [pgs. 55-57], 2nd Ed., Wiley

Star Schema Benchmark (SSB)
Created to Measure Star Schema Query Performance



SSB is Derived From TPC-H Benchmark



Why Changes to TPC-H Make Sense

- TPC-H has PARTSUPP table listing Suppliers of ordered Parts to provide information on SUPPLYCOST and AVAILQTY
- This is silly: there are seven years of orders (broken into ORDERS and LINEITEM table) and SUPPLYCOST not stable for that period
- PARTSUPP used in OLTP, not a Query table: as I fill orders I would want to know AVAILQTY, but meaningless over 7 years
- AVAILQTY and SUPPLYCOST are never Refreshed in TPC-H either; We say PARTSUPP has the wrong temporal "grain"[1]
- One suspects the real reason for PARTSUPP is to break up what might be a Star Schema, so Query Plans are not too simple
- Combining LINEITEM and ORDER in TPC-H to get LINEORDER in SSB, with one row for each one in LINEITEM, is common practice[2]

[1,2]R. Kimball & M. Ross, The Data Warehouse Toolkit [pg. 18, pg. 121], 2nd Ed., Wiley

Why Changes to TPC-H Make Sense

- To take the place of PARTSUPP, we have a column SUPPLYCOST in LINEORDER of SSB to give cost of each item when ordered
- See: http://www.cs.umb.edu/~poneil/StarSchemaB.pdf

[1]R. Kimball & M. Ross, The Data Warehouse Toolkit [pg. 129], 2nd Ed., Wiley

Star Schema Benchmark Queries

- SSB has 13 queries in 4 ‘Flights’ varying numbers of dimension columns restricted & selectivity of restrictions
- Successive Queries in a flight (Q1.1, Q1.2, Q1.3 in flight Q1) drill down in column hierarchies to reduce Where clause selectivity
- Dimension Hierarchies
Date	Part	Customer	Supplier
_Year 7	Mfgr 5	Region 5	Region 5
Month 84	Category 25	Nation 25	Nation 25
Week 364	Brand1 1000	City 250	City 250

SSB Filter Factors

Query	**FF on lineorder**	**FFs of indexable predicates on dimension columns**				**Combined FF Effect on lineorder**
	FF on Quantity(50) discnt (11)	FF on date wk/mo roll-up	FF on part Brand1 roll-up	FF on supplier city roll-up	FF on customer city roll-up	
Q1.1	.48*3/11	1/7 yr				.019
Q1.2	.2*3/11	1/84 mo				.00065
Q1.3	.1*3/11	1/364 wk				7.5E-5
Q2.1			1/25 category	1/5 region		.0080
Q2.2			1/125brnd1.betw	1/5		.0016
Q2.3			1/1000 brnd1.eq	1/5		.00020
Q3.1		6/7 yr.betw		1/5 region	1/5 region	.034
Q3.2		6/7		1/25 nation	1/25 nation	.0014
Q3.3		6/7		1/125 city	1/125 city	5.5E-5
Q3.4		1/84		1/125	1/125	7.62E-7
Q4.1			2/5 mfgr.betw	1/5	1/5	.016
Q4.2		2/7	2/5	1/5	1/5	.0046
Q4.3		2/7	1/25	1/25	1/5	9.1E-5

Clustering Crucial to Performance

- Clustering is ***crucial*** to performance on modern disks since query retrievals with filter factors ≥ 0.0005 use sequential disk search
- We demonstrate in paper changes in last 20 years as follows
- We compare a query from Set Query benchmark run in 1989 on MVS DB2 with same query (larger table) run on 2008 DB2 UDB

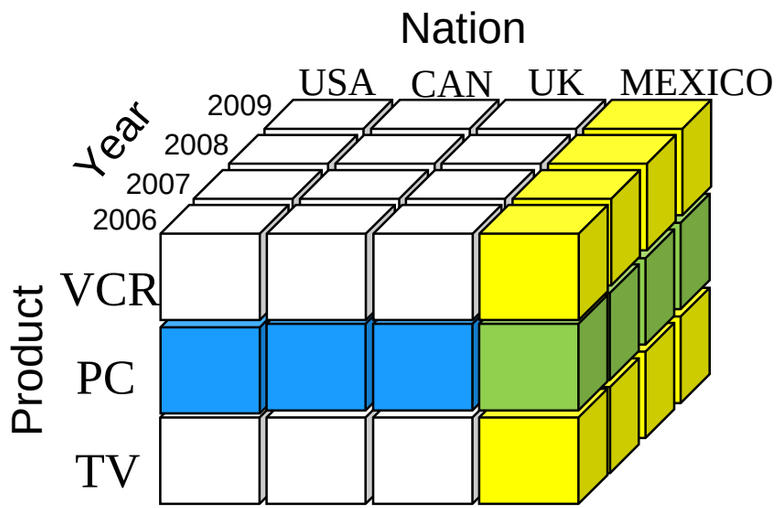
Clustering: Set Query Comparison

- Both indexed retrieval of individual rows and sequential scan have sped up since DB2 MVS, but sequential scan much more!
- Sequential scan speed has increased from MVS to UDB by a factor of 88; random row retrieval speed by a factor of 4
- FFs need to be much smaller to merit use of indexed access, but clustering reduces sequential scan range and is always valuable
- Bottom line: today, need a megabyte of sequential access to justify one random access

DB2's Multi-Dim Clustering (MDC)

- Example cube on dimensions: Year, Nation, Product Category
- Each row is placed in Cell on disk (intersection of dim values)
- Queries with multiple range predicates on axis columns will retrieve only data from Cells in range intersection
- Blocks that make up cells must be one MB or more so sequential scan access time on the cell no worse than inter-cell access time

[1]S. Lightstone, T. Teory and T. Nadeau. *Physical Database Design*. Morgan Kaufmann



DB Performance: Multi-Dim Clustering

- DB2's MDC treats table columns as orthogonal axes of a Cube; these columns are called "Dimensions"
- But MDC "Dimensions" are columns in a table turned into a Cube, not columns in Dimension tables of a Star Schema
- We can't embed all Dimension columns of a Star Schema in the Fact table: takes up too much space for huge number of rows
- How does DB2 MDC handle Star Schemas? Not entirely clear from DB2 Documentation

Adjoined Dimension Columns (ADC)

- To use columns of Star Schema Dimension tables in MDC as Cube Axes we adjoin copies of Dimension columns to Fact table
- We call this “ADC”: Adjoined Dimension Columns for Cube Axes
- Only 4-5 columns adjoined, of relatively low cardinality, since the number of Cells is the product of table Axis column values
- Why is low cardinality important? Each contiguous Block of a Cell must sit on at least a megabyte of disk, so sequential access time in Cell no more than seek time between Cells.

  (DB2 MDC documents don’t describe this, but that’s the idea)
- NOTE: We keep ALL the data in this scheme: no summarizing!

ADC Example SSB at SF100

- To adjoin columns: d_year, c_region, s_nation and p_category to the SSB fact table LINEORDER, we can simply modify the table
- Note cardinality is 7X5X25X25 = 21875; then ad_lineorder will have 76 GB, and 76 GB/21875 = 3.5 MB per cell: Good size
- Then queries with restrictions on C.c_region, for example, must translate to restriction on ad_lineorder.c_region
- Or can create an MQT (materialized query table, in general terms a Materialized view) from the lineorder table to ad_lineorder
- That means inserts to lineorder table will automatically insert in the MQT, but we use up twice as much space & extra load time
- In either case if restrict to d_yearmonth = 199401 or c_nation = ‘Canada’ we’ll also need d_year = 1994 and c_region = ‘America’
- There’s no dimensional hierarchy knowledge in DB2 or other DBs

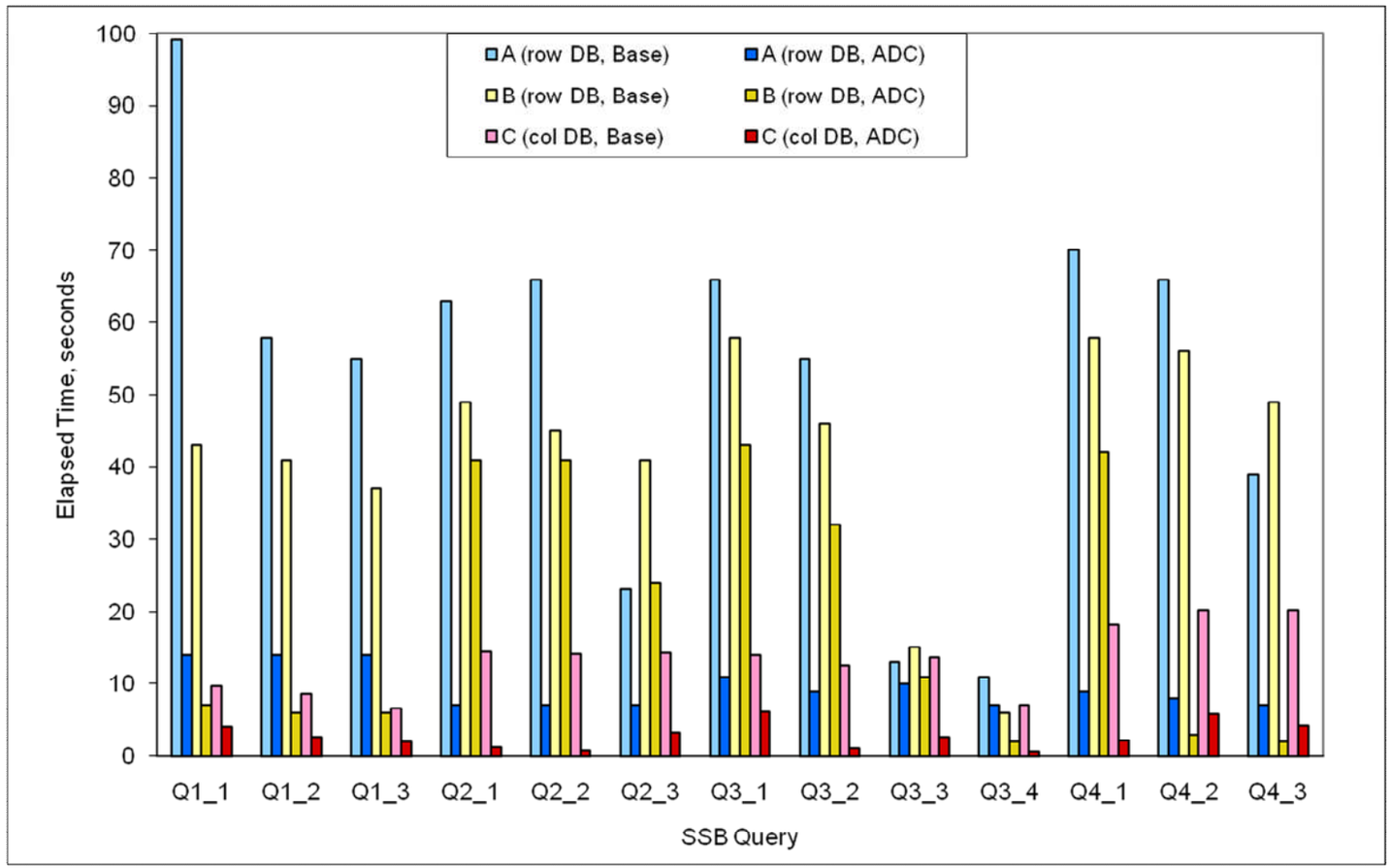
ADC on other DBs

- Oracle has a partitioning feature that supports cubing into cells
- In DB products with good indexing (Sybase IQ, Vertica) we can sort rows by the concatenated key of the ADC columns: (2,7,4,3)
- Later IQ inserts don’t sit contiguously in cells; still accessible by index in queries: DB2 clustered table has the same problem.
- Vertica DBMS places new insert into memory, which is later merged out to disk version, so new inserts go into proper cells (Until then, quickly retrieved from memory)
- Vertica also recognizes query restrictions on dimension columns that are ADCs on fact table, a valuable ability for other products

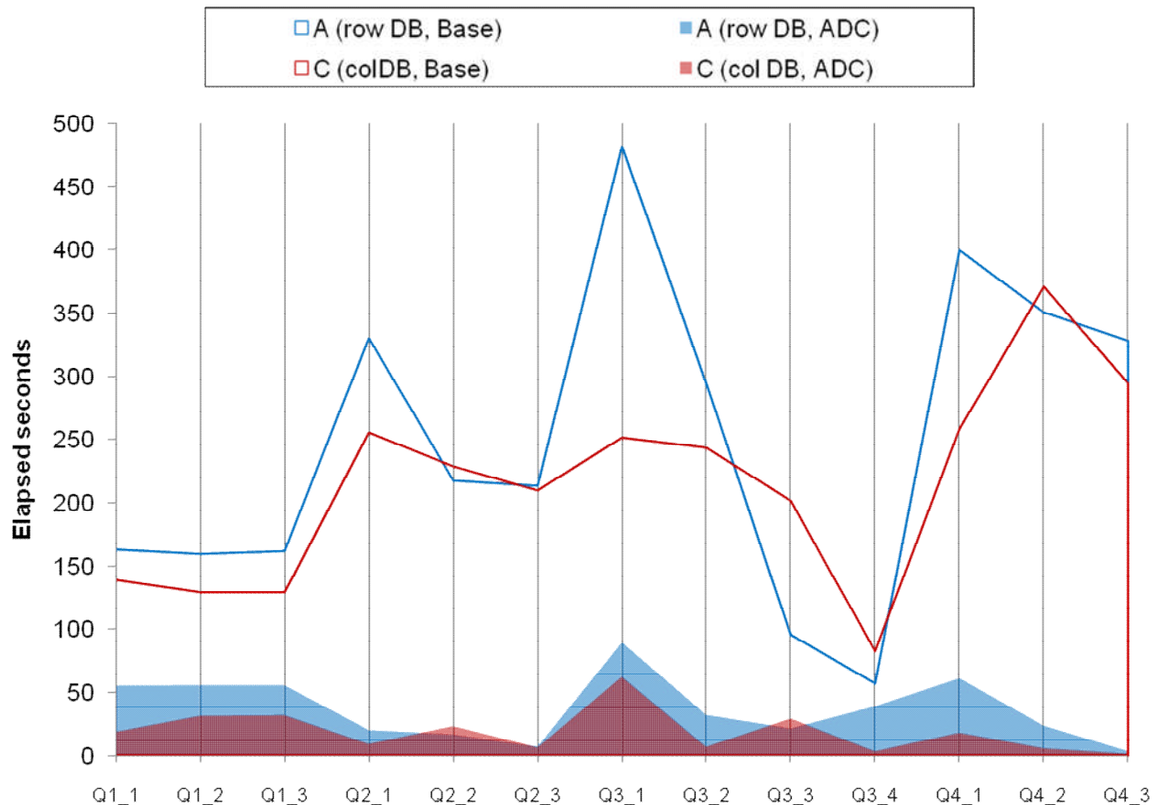
SSB Test Specifications

- We measured 3 DB products (none were Vertica) named A, B, and C, with SSB tables at SF10, where LINEORDER has ~7.5 GB; also measured products A and C at SF100 with 76 GB LINEORDER
- Adjoined dimension columns were, e.g., d.year, s.region, c.region, and p.category (brand-hierarchy), cardinalities 7, 5, 5, and 25
- In LINEORDER table these adjoined columns were named d_year, c_region, s_nation and p_category, and used in the SSB Queries

DB Products A, B & C Times in Seconds: SF10

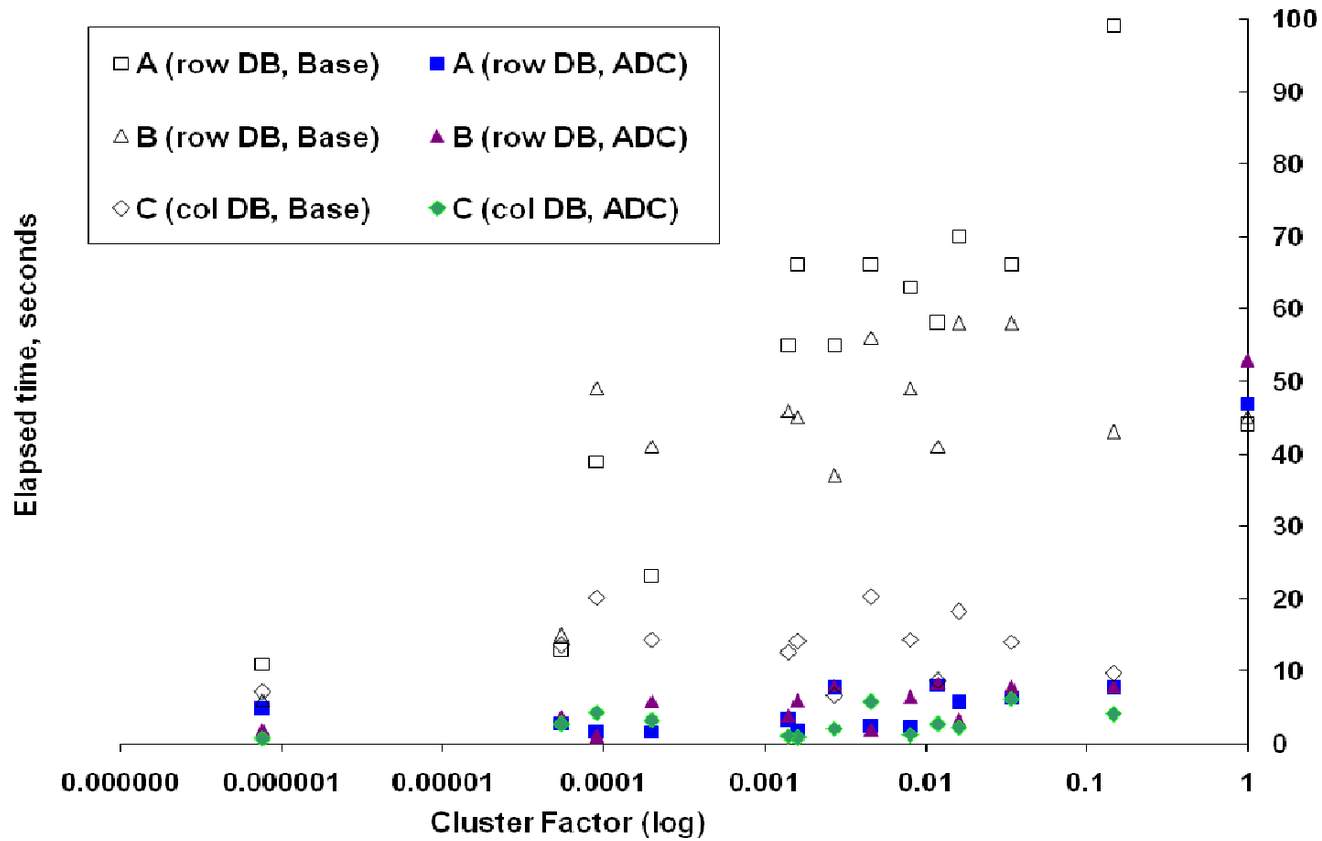


DB Products A & C Times in Seconds: SF100



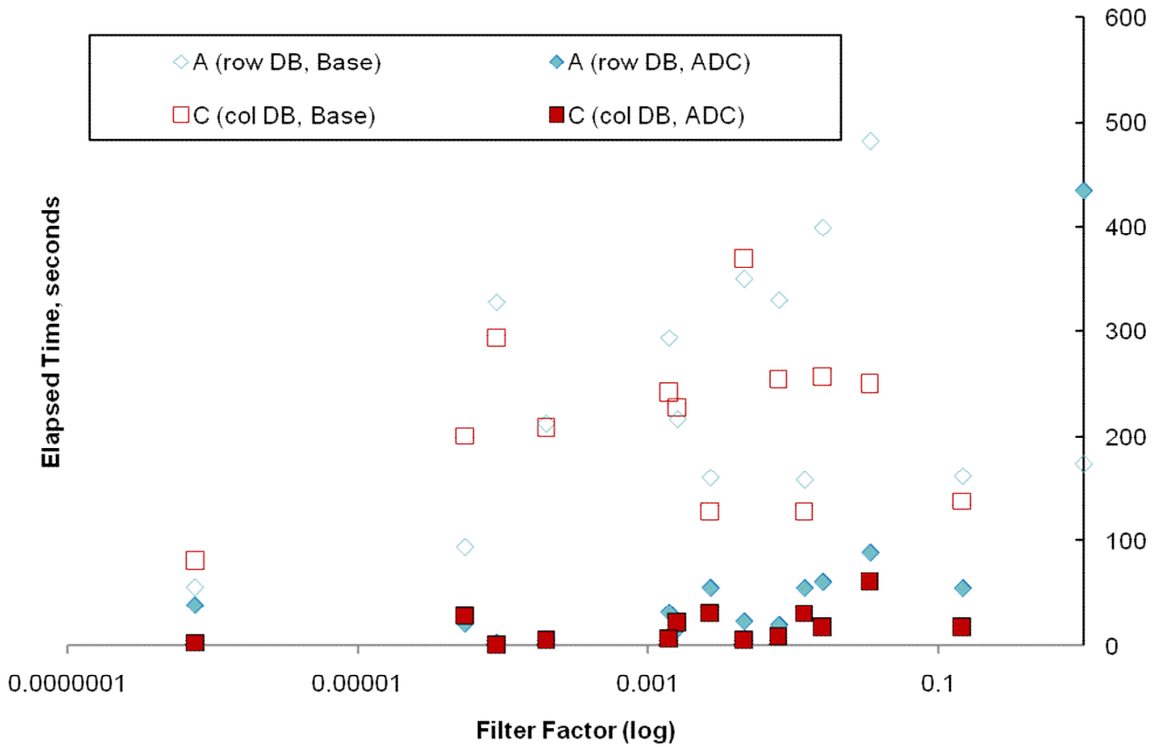
Analysis of SSB and Conclusions SF10

Query Elapsed Times vs. Filter Factor, FF (on log scale)



Analysis of SSB and Conclusions SF100

Query Elapsed Times vs. Filter Factor, FF (on log scale)



Analysis of SSB and Conclusions

- On prior two plots, two regions of Filter Factor: FF
  - At low end of the FF Axis (FF < 1/10000), secondary indexes are effective at retrieving few rows qualified: ADC has little advantage
  - For FF in the intermediate range where most queries lie, ADC reduces elapsed times very effectively vs. BASE case
- Elapsed time is reduced from approximately that required for a sequential scan down to a few seconds
- Believe this demonstrates validity of ADC approach

Analysis: ADC Weaknesses/Solutions

- In an MQT/Materialized View, queries referencing dimension cols adjoined to a fact table will automatically translate to the ADCs
- But cols in query restrictions that are in the dimensional hierarchy of ADC will NOT restrict the corresponding ADC in the fact table (restriction on Nation won't limit cell search on LINEORDER)
- Human mod of query needed since no DB understands hierarchy, though many OLAP products do: a solved problem, need for ADC
- Also, shouldn't need to create a MQT or MV (waste of space) to identify dimension columns with same columns adjoined to fact table; can support this kind of aliasing as if ADC were MV
- Only DB2 MDC & Vertica place new inserts in existing MDC cells; Clustered Indexes in DB2 have same problem, but rows placed at end of table aren't very inefficient if reorganized occasionaltaly

Summary

- Star Schema Benchmark
- Clustering crucial to performance on modern disks
- Good DB2 Multi-Dimensional Clustering (MDC)
- Dices fact table on “column axes” only in fact table
- Adjoin copies of Dimension Columns to fact table as axes of MDC, for crucial speed-up of SSB queries
- Can create MDC-like cells on well-indexed DBMS
- All products need mods of query restrictions to work

Bibliography

- Jim Gray, ed., The Benchmark Handbook, in ACM SIGMOD Anthology, http://www.sigmod.org/dblp/db/books/collections/gray93.html (Set Query bench.)
- R. Kimball & M. Ross, The Data Warehouse Toolkit, 2nd Ed., Wiley, 2002
- S. Lightstone, T. Teory and T. Nadeau. *Physical Database Design*, Morgan Kaufmann, 2007
- P. O'Neil, E. O'Neil, X. Chen. The Star Schema Benchmark. http://www.cs.umb.edu/~poneil/StarSchemaB.pdf
- M. Stonebraker et al, *C-Store, A Column-Oriented DBMS*, VLDB 2005. http://db.csail.mit.edu/projects/cstore/vldb.pdf
- M. Stonebraker et al., One Size Fits All? — Part 2: Benchmarking Results. CIDR 2007 http://www-cs.wisc.edu/cidr/cidr2007/index.html, press 'Electronic Proceedings' to download
- TPC-H, an ad-hoc decision support benchmark. http://www.tpc.org/tpch/
- TPC-DS, http://www.tpc.org/tpcds/The_Making_of_TPCDS.pdf; http://www.tpc.org/tpcds/TPCDS_Workload_Analysis.pdf