FinTime - a financial time series benchmark

Kaippallimalil J. Jacob
Morgan Stanley Dean Witter & Co., New York
kjacob@acm.org

Dennis Shasha
Professor, Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
http://cs.nyu.edu/cs/faculty/shasha/index.html
shasha@cs.nyu.edu

About FinTime

FinTime (http://cs.nyu.edu/cs/faculty/shasha/fintime.html) is a set of data and queries that reflects the needs of financial analysts who are studying patterns in stock market data, but it should appeal to the designers of any system that has pretensions of handling ordered data well.

Unlike some other benchmarks, this one makes no requirement that all queries be expressed in a given language (e.g. SQL 2000). If you claim you have a query language, that's good enough. It's up to your customers to decide on syntactic and semantic elegance.

FinTime has evolved from a tutorial on time series databases given by Shasha during VLDB 98 (see his web page) and reflects Jacob and Shasha's best understanding of typical data analysis queries issued by users. Since many vendors have products that handle ordered data, such a benchmark can help would-be customers to evaluate them. Already several large and small vendors have expressed interest.

Benchmark Description and Generation

The models suggested in FinTime reflect two frequently occurring cases in the financial industry, namely, a historical market data system (decision support) and real-time price tick database (on-line transaction processing). FinTime also suggests and defines metrics that capture three useful dimensions of any time-series system, namely, performance in a single-user mode, performance in a multi-user mode and the price to performance ratio.

Models for a Time-series Benchmark

Before deciding on a model, we have to examine the different parameters that determine a model for time-series system. The most important parameters that influence a time-series database system are:

1. Periodicity of data (Regular/irregular)
2. Density of data (Dense/Sparse)

3. Schedule of updates (periodic, continuous)
4. Types of queries (Simple/Complex)
5. Time interval between queries (Ad hoc/Batch)
6. Number of concurrent users (Few/Many)

Combinations of these factors will give rise to 64 possible models but for simplicity we can focus on the following commonly occurring cases in the financial industry.

Model 1: Historical market Information

Attribute	**Specification**
Periodicity of Data	Periodic
Density of Data	Dense
Schedule of updates	Periodic updates (e.g. at the end of a business day)
Complexity of queries	Complex (e.g. Decision support queries)
Nature of query arrival	Batch
Concurrency of users	Low (e.g. Few concurrent users)

Model 2: Tick databases for financial instruments

Attribute	**Specification**
Periodicity of Data	Non-periodic
Density of Data	Sparse to moderately dense
Schedule of updates	Continuous
Complexity of queries	Simple
Nature of query arrival	Ad hoc
Concurrency of users	High (e.g. Many concurrent users)

Let us now discuss the characteristics of the two models in some detail.

<u>Model 1: Historical Market Information</u>

Historical Market data systems are closely related to decision support systems of the traditional relational database world. The various elements of this model are:

- *Data Model:* In the case of historical time-series databases for market information, the model consists of a few relational tables that typically contain infrequently changing (static) information and a number of time-series tables. We omit the detailed data type specifications in this brief summary so we can concentrate on the queries.

- *Data population, volume of data and update mechanism*. Market information is typically provided as a set of input files by a market data vendor at the *end of each trading day*. While the data for the *Base Information* table, *Split Adjustment* Table and *Dividend* Table is irregular (i.e. an external event triggers an entry into these tables), the *Market Information* Table has an entry for every trading day. For the purposes of this benchmark the implementation can select a scale factor combination of the number of securities under consideration and the number of events for those securities. We suggest 3 scale factors, namely, 50,000 securities, 100,000 securities, and 1,000,000 securities, all for 4,000 days. These roughly correspond to all equity securities in the US, all equity securities in the G7 countries and all equity securities in the world. We provide code for generating data.

- *Query characteristics*. The benchmark queries exercise the following functionality:

    1. Join of relational data and time-series information
    2. Access of long depth time-series information for a few keys (deep history query)
    3. Access of a short depth time-series for a large number of keys (cross-sectional queries)
    4. Sorting
    5. Grouping and Aggregation

Following are the benchmark queries to be run against the data model defined above. Many of them include a notion of ``specified set of securities'' or ``specified period''. This notion is defined in the glossary with respect to a simple random model.

1	Get the closing price of a set of 10 stocks for a 10-year period and group into weekly, monthly and yearly aggregates. For each aggregate period determine the low, high and average closing price value. The output should be sorted by id and trade date.
2	Adjust all prices and volumes (prices are multiplied by the split factor and volumes are divided by the split factor) for a set of 1000 stocks to reflect the split events during a specified 300 day period, assuming that events occur before the first trade of the split date. These are called split-adjusted prices and volumes.
3	For each stock in a specified list of 1000 stocks, find the differences between the daily high and daily low on the day of each split event during a specified period.
4	Calculate the value of the S&P500 and Russell 2000 index for a specified day using unadjusted prices and the index composition of the 2 indexes (see appendix for spec) on the specified day

5	Find the 21-day and 5-day moving average price for a specified list of 1000 stocks during a 6-month period. (Use split adjusted prices)
6	(Based on the previous query) Find the points (specific days) when the 5-month moving average intersects the 21-day moving average for these stocks. The output is to be sorted by id and date.
7	Determine the value of $100,000 now if 1 year ago it was invested equally in 10 specified stocks (i.e. allocation for each stock is $10,000). The trading strategy is: When the 20-day moving average crosses over the 5-month moving average the complete allocation for that stock is invested and when the 20-day moving average crosses below the 5-month moving average the entire position is sold. The trades happen on the closing price of the trading day.
8	Find the pair-wise coefficients of correlation in a set of 10 securities for a 2 year period. Sort the securities by the coefficient of correlation, indicating the pair of securities corresponding to that row. [Note: coefficient of correlation defined in appendix]
9	Determine the yearly dividends and annual yield (dividends/average closing price) for the past 3 years for all the stocks in the Russell 2000 index that did not split during that period. Use unadjusted prices since there were no splits to adjust for.

- *Operational parameters*

  Since the benchmark model assumes that the level of concurrency is small, the number of users should be 5. The benchmark methodology should be that each of the five users executes all the queries listed above in a randomly generated permutation. The five users will be simultaneously active at any point in time. Where the output order is not specified, it can be displayed/stored in any order.

Model 2: Tick databases for financial instruments

This benchmark models a very different, but commonly occurring, scenario from the one above. In this case, the benchmark attempts to model a case where the database is expected to keep up with a very high rate of updates while responding to several users issuing fairly simple queries. This case is quite similar to OLTP systems in the relational world.

The case being modeled is the tick database for a financial system. Ticks are price quotation or trade (transaction) prices and associated attributes for individual securities that occur either on the floor of a stock exchange or in an electronic trading system, such as the NASDAQ market system. Ticks include 2 basic kinds of data (1) Trades are transactions between buyers and a sellers at a fixed price and quantity (2) Quotes are price quotations offered by buyers and sellers. Quotes can have the ask quote, the bid quote or both along with their associate attributes such as quantity offered.

- *Data population, Update frequency and volume.* Tick databases are generally populated by adapters that receive data from real-time feeds. The frequency of updates is constantly increasing but on the average we can assume that each security ticks (each corresponding to a record in the database) about 100 times during an 8-hour trading day. Additionally, we can assume that the set of securities being tracked is traded around the world and hence there is no quiescent period. For the purposes of this benchmark we will assume that the system monitors ticks on 1,000, 10,000 or 100,000 securities, where each represents a different scale factor.

  A very important consideration in tick databases is the ability to quickly apply "cancel/correct". Occasionally an incorrect quote or trade record is published. The vendor will then send a correction record with the identifier of the security and its sequence number. The record will either be corrected according to the newly published data or simply deleted.

- *Query characteristics*. The kinds of queries issued against an online tick databases are usually simple and pre-determined. Listed below are the set of queries to be used in the benchmark.

- *Updates Between Queries.* After each query, 200 rows from the Trades/Quotes table are chosen randomly and uniformly, and each price field is changed to a value chosen randomly and uniformly between 0 and twice its current price.

1	Get all ticks for a specified set of 100 securities for a specified three hour time period on a specified trade date.
2.	Determine the volume weighted price of a security considering only the ticks in a specified three hour interval
3.	Determine the top 10 percentage losers for the specified date on the specified exchanges sorted by percentage loss. The loss is calculated as a percentage of the last trade price of the previous day.
4.	Determine the top 10 most active stocks for a specified date sorted by cumulative trade volume by considering all trades
5.	Find the most active stocks in the "COMPUTER" industry (use SIC code)
6.	Find the 10 stocks with the highest percentage spreads. Spread is the difference between the last ask-price and the last bid-price. Percentage spread is calculated as a percentage of the mid-point price (average of ask and bid price).

- *Operational parameters.* Since usually tick systems have a relatively large number of concurrent users, we will assume that there are 50 concurrent users for the benchmark. These users will execute all of the queries listed above in a random permutation. Each user will submit each query only once and run through the complete list of queries. Where not specified the output order is not significant.

Metrics

Here are metrics to allow customers to compare one set of results against another. Each metric should be stated with respect to a scale factor and results for Historical Market Information should be reported separately from results for Tick Databases. For Historical Market Information, the scale factor is the number of securities, assuming 4,000 days of history. For Tick Databases, the scale factor is also the number of securities, assuming 100 ticks per security per day and 90 days of history. Since the queries entail random selections from larger sets as explained in the glossary (e.g. ``specified set of 100 securities''), these tests should be run 10 times and vendors should report averages and standard deviations.

1. **Response Time Metric**: A single user metric

   This can be defined as the geometric mean of the execution time of each of the queries executed one after the other in a single-user mode

2. **Throughput Metric**: A multi-user metric

   The Throughput Metric is defined as the average time taken by a user to complete his workload in a multi-user system.

3. **Cost Metric**. A cost measure

The cost metric is used to bring the capabilities of the hardware into the equation. It is based on the assumption that additional hardware capabilities would be reflected in higher hardware costs.

Constraints

1. Full ACID property for transactions. This constraint is applicable to both models, where updates are applied concurrently with queries.

2. For either benchmark, each user must execute the complete query set once. They do not have the option of selecting a subset of queries or re-executing a query.

Conclusions

Benchmarks help compare different products that solve the same problem. It gives the potential user of these systems a meaningful way of assessing the capabilities of differing products. Since it is quite difficult to capture all the information about a system in a few benchmark measures, it is important that the implementation of this benchmark record the exact conditions under which the tests were carried out and the results that were used to arrive at the metrics. Eventually, these conditions will fall under a certification procedure.

Bibliography

1. Transaction Processing Council (1998), "TPC-D Benchmark Specification" Version 2.0.

2. M. Stonebraker (1998), "Performance and Database Machines", Readings in Database Systems, *Morgan-Kaufmann, San Mateo*

3. D. Biiton, D. De Witt and C. Turbyfill (1983), "Benchmarking Database Systems: a systematic approach", *Proceedings of the Ninth Very Large Database Conference, Florence, Italy*

4. M. J. Carey, D. J. deWitt and J. F. Naughton (1993), "The OO7 Benchmark", *Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington D.C.*

5. A. B. Chaudhri (1996), "An Annotated Bibliography of Benchmarks for Object Databases",

Glossary

Defined below are some concepts and measures commonly used in the financial industry and referred to in the benchmark specification

1. Split and Reverse Splits: A stock split is a process in which one share of a security is converted into a specified number of shares. A reverse split is the opposite process where a number of shares of a security is converted into 1 share. The ratio between the number of old securities and the new ones is called the split factor. E.g. If a security splits "2 for 1" the split factor is 0.5. If a security reverse splits "1 for 3" the split factor is 3/1 or 3.

2. Adjusted prices and volume. Since splits and reverse splits change the value of a security at the point the event occurs. E.g. Assume that an order to purchase 100 shares of a security at $50 per share was placed before a split. After a "2 for 1" split is effected, the order will be for 200 shares at $25. Notice that the total value of the order has not changes. However the price was multiplied by the split factor and the volume was divided it.

3. Volume-weighted price: This measure is a weighted average of the trade prices where the size of each trade is used as the weight.

4. Standard Industry Code: Most market data vendors provide a categorization of securities by the industry in which they operate. For example, IBM is categorized to be in the "COMPUTERS" business. While typically they are codes, in this benchmark we have expanded them in meaningful words.

5. Dividends & Yield: Dividends are pay-outs of earnings to shareholders. They are specified as a dollar amount per share and the date of disbursement is termed as the Ex-Dividend date. Dividends may be paid out per quarter or annually or according to any other schedule that the company chooses. Annual Yield for a security, as defined for this benchmark, is the total dividend paid out during a calendar year as a percentage of the average share price (split adjusted). It is important to note that dividend payments should also be split adjusted (dividend rate * split factor).

6. Spread: is defined to be the difference between the price at which a broker will sell a security and the price at which the broker will buy it. This is the source of profits for a broker. Typically, large well recognized securities have small spread while securities of small companies have larger spreads.

7. The notions of ``specified period of x days'' or ``set of y securities'' are used in the query descriptions. To compute a specified period of x days from a set of days D, choose x days without replacement randomly and uniformly from D. If they must be consecutive days, then choose a single day randomly and uniformly from the set of days that are at least x less than the most recent day in the database. The same holds for hours in the tick database. To compute a set of x securities from some set S, choose the x randomly and uniformly but without replacement from S.

8. S&P/Russell indices. The composition of these indexes can be generated by randomly selecting 500 and 2000 distinct securities at random uniformly and without replacement from the universe of securities.