# VLDB Conference – General Thoughts

Emphasis on traditional areas such as query processing, web data management and search. Not much emphasis on XML processing. Some ideas made the rounds in different contexts (query optimization across batches - An Approach to Optimize Data Processing in Business Processes; data base schema design in the conference’s best paper - Scalable Semantic Web Data Management Using Vertical Partitioning).

Went to two tutorials (Adaptive query processing: Why, How, When, and What Next? and Secure Data Outsourcing) – they were so-so, the latter being the better one.

Interesting papers (research and industrial tracks):

* Snapshot isolation problem detection - Automating the Detection of Snapshot Isolation Anomalies
* Deriving XML schema - Inferring XML Schema Definitions from XML Data
* Query optimization – On the Production of Anorexic Plan Diagrams
* Time-constrained query processing - Supporting Time-Constrained SQL Queries in Oracle
* Bulk deletion - Efficient Bulk Deletes for Multi Dimensionally Clustered Tables in DB2

Talked to many students and professors. Spread the word on job opportunities in SQL Server.

# Keynote Addresses

One keynote per day.

### 09/25/2007 - [Data Access Patterns in The Amazon.com Technology Platform](file:///E%3A%5Cpapers%5Cspecial%5Cp1-vogels.pdf)

Werner Vogels

VP Worldwide Architecture, CTO, Amazon.com

State management is at the dominant factor in scaling

Availability, Scalability, Manageability, Consistency, Efficiency, Performance require customized solution

Published a paper in SOSP 2007 on DYNAMO

* Lots of knobs, consistency window

From retail to technology company

1.1m sellers, 63m customers

Important factors for Customer Experience:

* Selection
* Lower prices

Customer Experience 🡪 Traffic 🡪 Sellers 🡪 Selection

Growth leads to Lower Cost Structure 🡪 Lower Prices

Incremental scalability is key

“An always-on service is said to be scalable if adding resources to facilitate redundancy does not result in loss of performance”

Scalable service is capable of handling heterogeneity

System can be scaled up/down one node at a time

Elastic – capable of growing and shrinking on demand – Amazon is pretty close to achieving this

Challenges

* Millions of commodity systems
* Many geoplexes in concert
* Thousands of services

Scalability, availability, performance and cost-effectiveness are dominated by data management

* No commercial open source or research technology available
* 20 different patterns of services at Amazon requiring different technologies

All Amazon services are applications (running on application servers) accessing the database

Scalability was an issue they encountered early on (partitioning, databases going down)

One year, they tried mainframe – didn’t see any advantage over cluster of servers

Move logic closer to database

Allow access to data through services, not direct access

Two layers have emerged (not a flat space) for supporting web services:

* Foundation (e.g. what constitutes the identity of sellers is not known at this level, only by the web service knows)
* Aggregates, where the data interpretation occurs

Availability:

* Multiple data centers – can lose a data center without losing SLA (availability and performance) for customer

Failures are highly correlated

* Systems are not stable; participants leave, malicious activities occur
* I/O controllers, memory
* In different quantities – nodes, racks, areas, datacenters, geographies, continents
* Systems don’t fail by stopping – must commit suicide

SLA are two-way contracts

* Latency, single service/path through the system, durability and availability, cost
* Difficult to architect for cost efficiency – everyone goes for best architectural solution – no principles for cost efficiency in architectural design

Engineering for performance at 99.9%

* Don’t recommend what the customer has already bought. Lots of computation involved. Best customers end up getting the best quality service

Requirements

* Query model
	+ Request/reply
	+ Asynchonous batch
	+ Stream (in DW, monitoring)
	+ Primary/secondary key
	+ Schema flexibility (items are added everyday)
	+ Query flexibility
* Update patterns
	+ Read-only, publishing, append-only (goes into archive storage), read-write
	+ Read/write ratios
	+ Cache-ability
	+ Main/secondary patterns, Concurrency
* Consistency models
	+ Strong (evil – performance and fault-tolerance would be negatively impacted),
	+ Eventual,
	+ Read-your-writes,
	+ Monotonic reads,
	+ Monotonic writes

No one size fits all

Most access are based on Primary key

* Eventual consistency

ACCESS PERCENTAGES:

If object size > 1MB – non-structured storage. Data not queryable (e.g. no full-text search).

Otherwise –

* 65% primary key access (
	+ 60% Eventual consistency [
		- Always writable 15%,
		- no 45%
			* {Latency < 50ms – 99.9%}],
	+ 5% Strong consistency),
* 35% multi-attribute access (
	+ Eventual consistency – 30%
		- [Relationship driven 20%
			* {RDBMS features – 5%, no 15%},
		- no 10%],
	+ Strong consistency 5%)

Secondary key and iterators (e.g. meta-data access) are on slow paths – mostly in management path

Don’t need schema, relational features, query engine, transaction isolation, consistency 🡪 why use RDBMS?

Real-time metrics warehouse – GBs

Aggregate Data warehouse

### 09/26/2007 - Technology for Developing Regions

Eric Brewer, professor, UC Berkeley

River blindness – eradicate blackfly in W. Africa – network of sensors – protects 30m people from infection – freed up 100,000 sq miles of land for agriculture

Grameen Bank – 46.5% of borrowers crossed the poverty line – small loans, $50 typical, maximum $375. Bookkeeping for millions of loans.

Grameen Telecom - $200 loans for cell phones to villages in Bangladesh – covers 50,000 of 68,000 villages – 60M users, makes it the largest telecom organization in the world!

Telemedicine – Aravind Hostital (eye) in Tamil Nadu – 2500 patients/month –– 10% get significant eyesight improvement (i.e. curable blindness to eyesight) – set up WiFi connections over distances of 50km to connect clinics – patients go to one clinic, over web cam doctors in a different location examine them. In a separate conversation , Eric said there is reverse technology transfer of telemedicine to West Virginia.

TIER – Technology and Infrastructure for Developing Regions

* Small deployments at a time

Focus on availability, eventual consistency (not ACID)

Many social impacts. Phone minutes as ecash (currency) – move away from cash – send phone minutes using SMS – transfer over country boundaries where you could not send money

IEEE Computer, June 2005. “The Case for Technology for Developing Regions” – Brewer et al.

### 09/27/2007 - Self-Tuning Database Systems: A Decade of Progress

Surajit and Vivek talked about the database engine tuning advisor, which won the 10-year best paper award. Good insight into the state of the art in 1997, the technical challenges and some of the solutions, and the current state of the industry on automatic tuning. Two video clips, very entertaining as well as informative!

# Most Controversial - The End of an Architectural Era (It's Time for a Complete Rewrite)

Michael Stonebraker talked about how the current database systems are missing the mark on the data storage and query requirements of the present time. For many applications such as search, efforts outside the database systems have been more successful, such as Google. For structured data, he focused on the TPC-C benchmark and argued that it becomes a main memory database with the current hardware availability. Disk I/O costs – for which many technologies such as locking were invented – are no longer relevant.

* Programming paradigm – using stored procedures. Break up each transaction into two phases: read during the first phase and possibly abort; read and write during the second phase but cannot abort.
* No need for locking – Databases can be single-threaded, transaction completes in 1 ms or less, so the throughput is high. This also achieves “strong” consistency. In a private conversation, he said he did not know much about Amazon and did not know why they would go after “eventual” consistency – he is a firm believer in “strong” consistency.
* No need for redo log – keep enough replicas, restore state from
* No need for undo log – run transactions in parallel on the replicas, either all of them will fail (e.g. constraint violation) or a quorum will succeed (e.g. hardware failure on some machines). If fewer than quorum succeed, then those can restore their state from the majority - shouldn’t be a common occurrence.

Using this approach and his H-store architecture, he was able to achieve a throughput of 82x over commercial databases. They are investigating the number of x’s due to locking, log maintenance, etc.