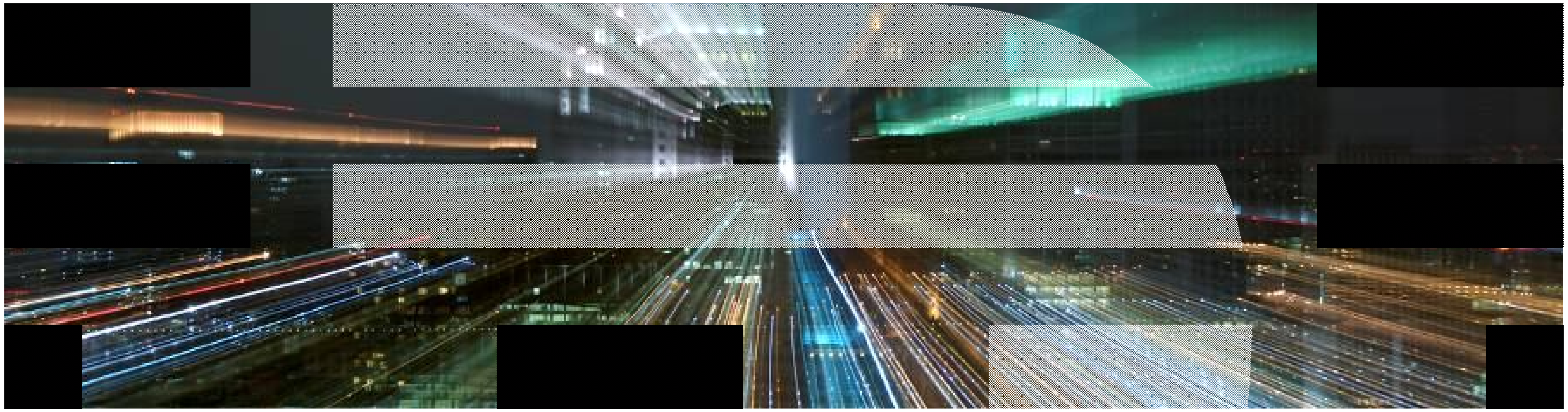
Akmal B. Chaudhri (艾克摩 曹理) -- IBM Senior IT Specialist

9 March 2012

Mirror, mirror on the wall, what's the fairest database technology of all?



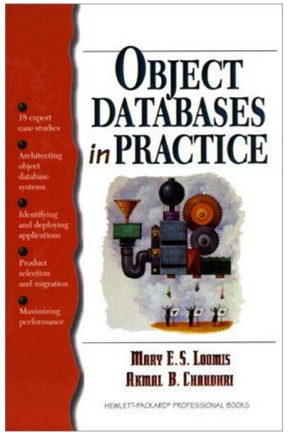
Abstract

“ What’s the best fit of database technology and data architecture for today’s application requirements, such as Big Data and web-scale computing? Akmal B. Chaudhri will present an <u>informative</u> and <u>objective</u> comparison of database technology and data architecture, including NoSQL, relational, NewSQL, graph databases, linked data, native XML databases, column stores and RDF data stores. ”

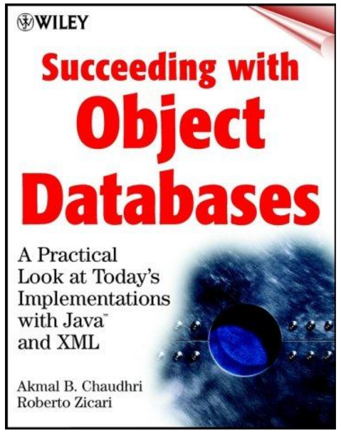
Source: http://www.all-freeware.com/images/small/46590-free_stereogram_screensaver_audio___multimedia_other.jpeg

My background

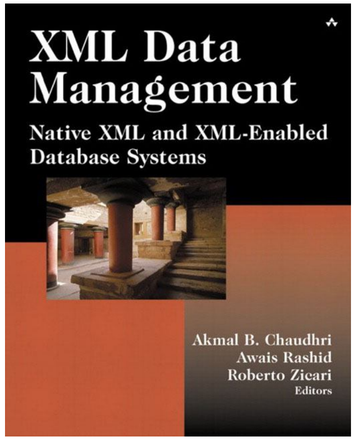
- 20+ years experience in IT
  - –Developer (Reuters)
  - –Academic (City University)
  - –Consultant (Logica)
  - –Technical Architect (CA)
  - –Senior Architect (Informix)
  - –Senior IT Specialist (IBM)
- Broad industry experience
- Worked with various technologies
  - –Programming languages
  - –IDE
  - –Database Systems
- Client-facing roles
  - –Developers
  - –Senior executives
  - –Journalists
- Community outreach
- Publications and presentations



Object Databases in Practice
Mary E.S. Loomis
Akmal B. Chaudhri
Hewlett-Packard Professional Books



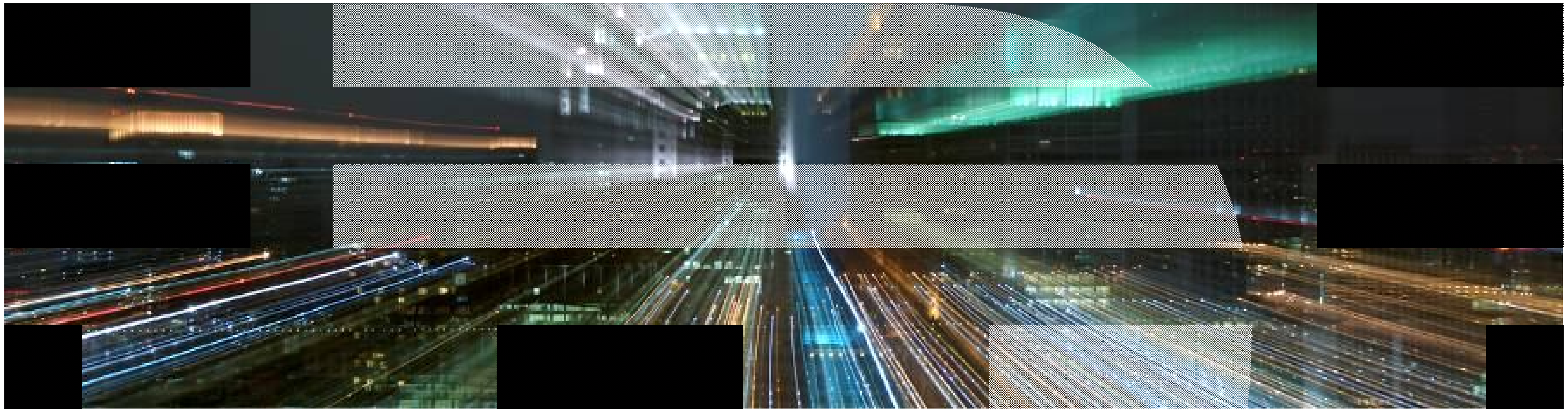
Wiley
Succeeding with Object Databases
A Practical Look at Today's Implementations with Java and XML
Akmal B. Chaudhri
Roberto Zicari



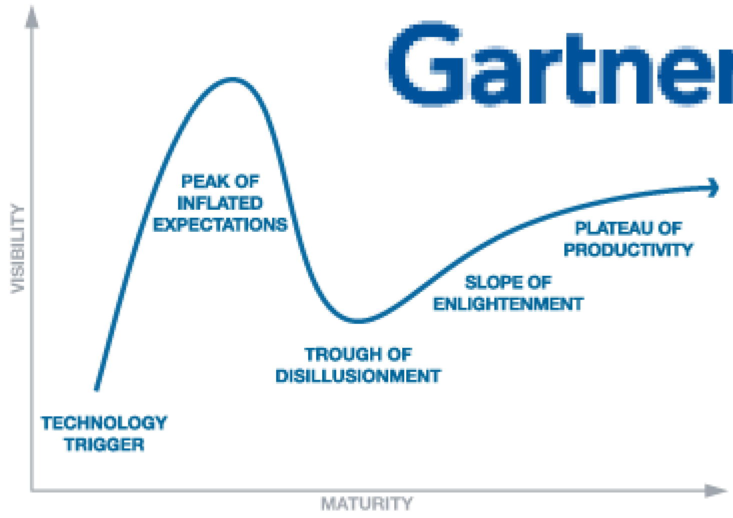
Agenda

- Introduction
- NoSQL
- New SQL
- Column-Oriented
- In-Memory
- Summary

Introduction



Lots of database market analysis

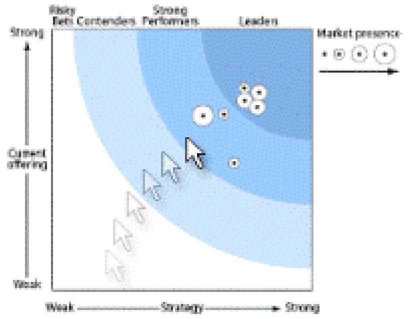


Gartner.

tdwi

IDC Analyze the Future

FORRESTER®

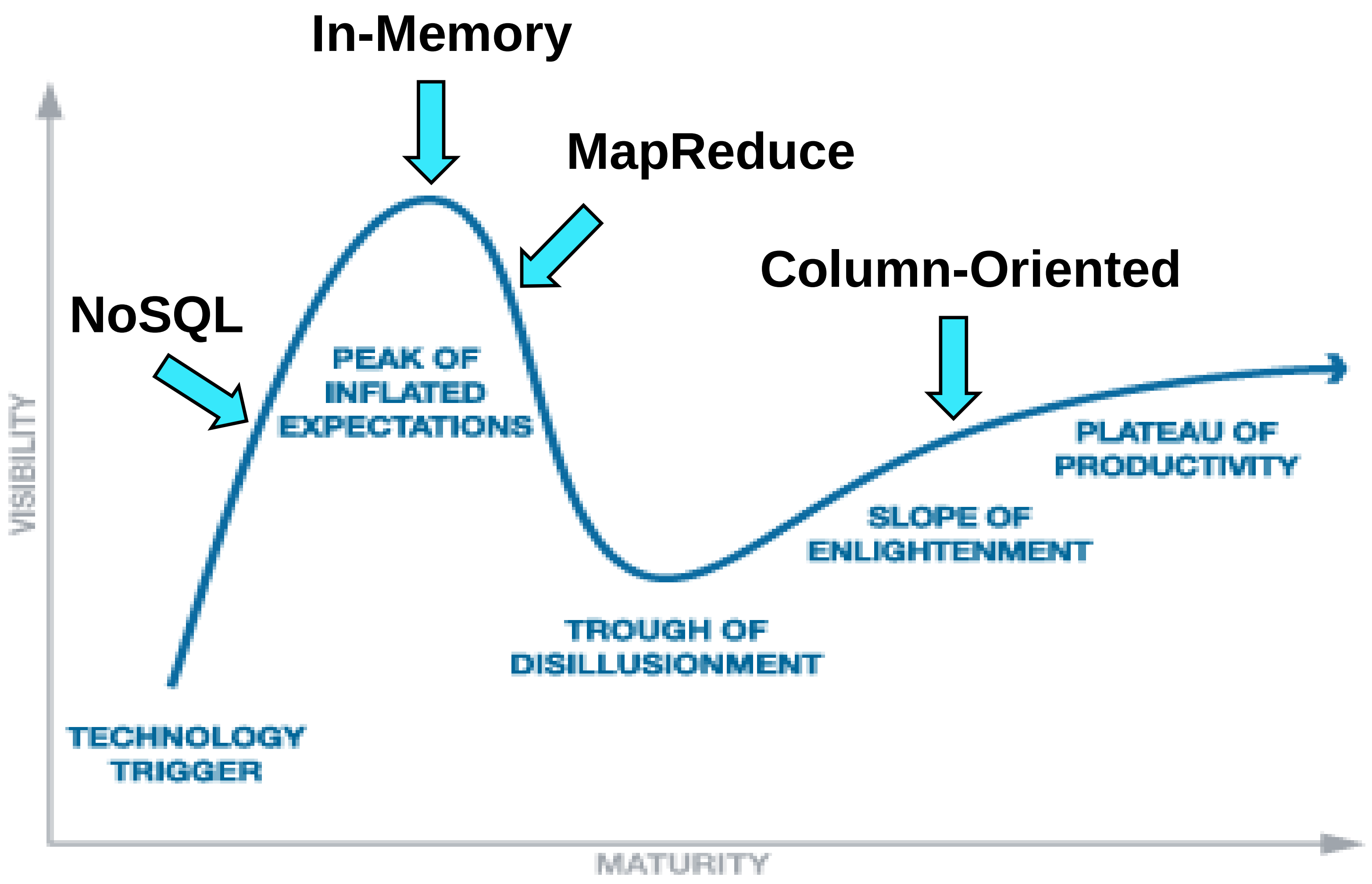


Bloor telling the right story

What analysts are saying

- Gartner 2011 Magic Quadrant for Data Warehouse Data Management
  - Data warehouse infrastructure to manage “extreme data”
  - More emphasis, appreciation and value of column-oriented database systems
  - Increasing adoption of in-memory database systems
- Forrester is including column-oriented, in-memory and Hadoop as part of their evaluation criteria for database and data warehouse technology
- TDWI Newsletter mentions column-oriented and in-memory
- Bloor comments on column-oriented as a major force in data warehousing

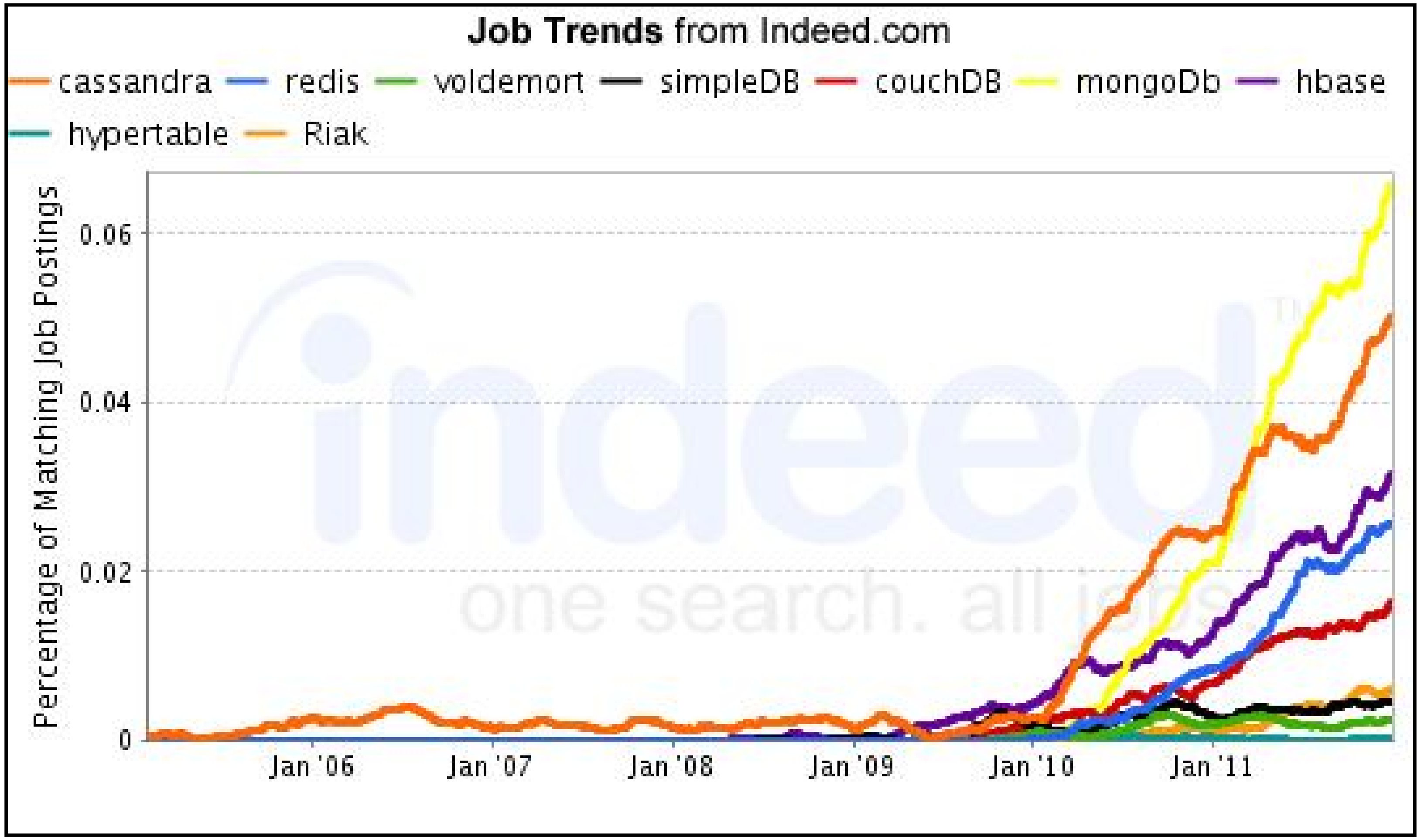
Gartner Hype Curve 2011



NoSQL surveys

- InformationWeek in 2010
  - 755 business technology professionals
  - 5% piloting NoSQL projects
  - 22% interested in NoSQL but need to learn more
  - 44% never heard of NoSQL
- Evans Data in 2011
  - 1200 software developers worldwide
  - Big Data driving adoption of NoSQL
  - Increasing enterprise interest
  - 39% of developers in EMEA plan to use NoSQL

NoSQL job trends



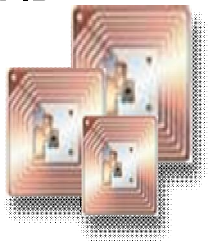
Source: http://regulargeek.com/2012/02/23/nosql-job-trends-february-2012/

NoSQL UK job trends

Extreme data

1.3 Billion RFID tags in 2005

30 Billion RFID tags in 2010



2 Billion Internet users in 2011

By 2013, annual internet traffic will reach **667 Exabytes**

4.6 Billon Mobile Phones World Wide

Google processes **> 24 Petabytes** of data in a single day

twitter processes **7 Terabytes** of data every day

facebook

Facebook processes **10 Terabytes** of data very day

Hadron Collider at CERN generates **40 Terabytes** of data / sec

For every session, NY Stock Exchange captures **1 Terabyte** of trade information

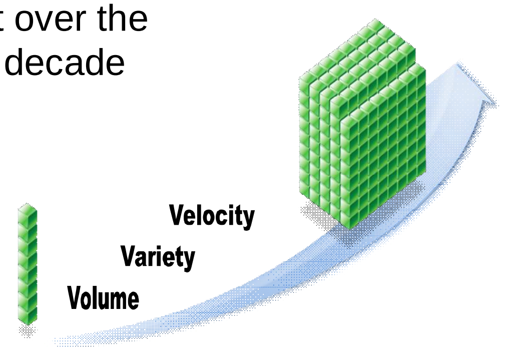
Organizations need deeper insights

44x

as much Data and Content over the coming decade

2020

35 zettabytes



2009

800,000 petabytes

80%

of world's data unstructured

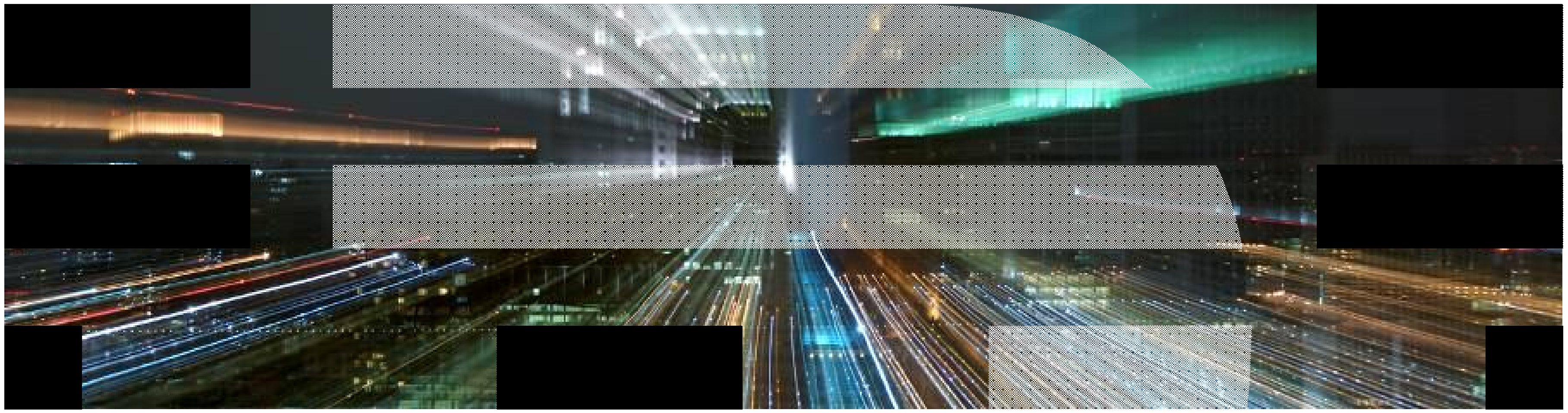
1 in 3 Business leaders frequently make decisions based on information they don't trust, or don't have

1 in 2 Business leaders say they don't have access to the information they need to do their jobs

83% of CIOs cited "Business intelligence and analytics" as part of their plans to enhance competitiveness

60% of CEOs need to do a better job capturing and understanding information rapidly in order to make swift business decisions

NoSQL



What is NoSQL?

- NoSQL, in general, is listed as at the beginning of the Hype Curve by Gartner -- so it is a term to watch for
- NoSQL started as a movement away from relational databases and SQL
- It has grown to include a number of different technologies
- For the most part, they are accessible without the use of SQL
- They are optimized to store different kinds of data -- documents (such as XML), graph, tabular
- There are key-value stores (such as Informix C-ISAM) that allow an application to store its data in a schema-less way
- The data could be stored in a datatype of a programming language or an object, so there is no need for a fixed data model

NoSQL datastores

Focus on	Give up
▪ Commodity servers, networking, disks ▪ Easy elasticity and scalability to multiple racks (10s to 100s of servers) ▪ Fault-tolerance and high availability	▪ Relational data model ▪ SQL APIs ▪ Complex queries (joins, secondary indexes, ACID transactions)

Two Worlds

Transactional

- Custom high-end OLTP
- Scaleout datastores for Cloud/Web 2.0
- Examples
  – MemcacheDB, Cassandra, Dynamo, Voldemort, SimpleDB, Gigaspaces

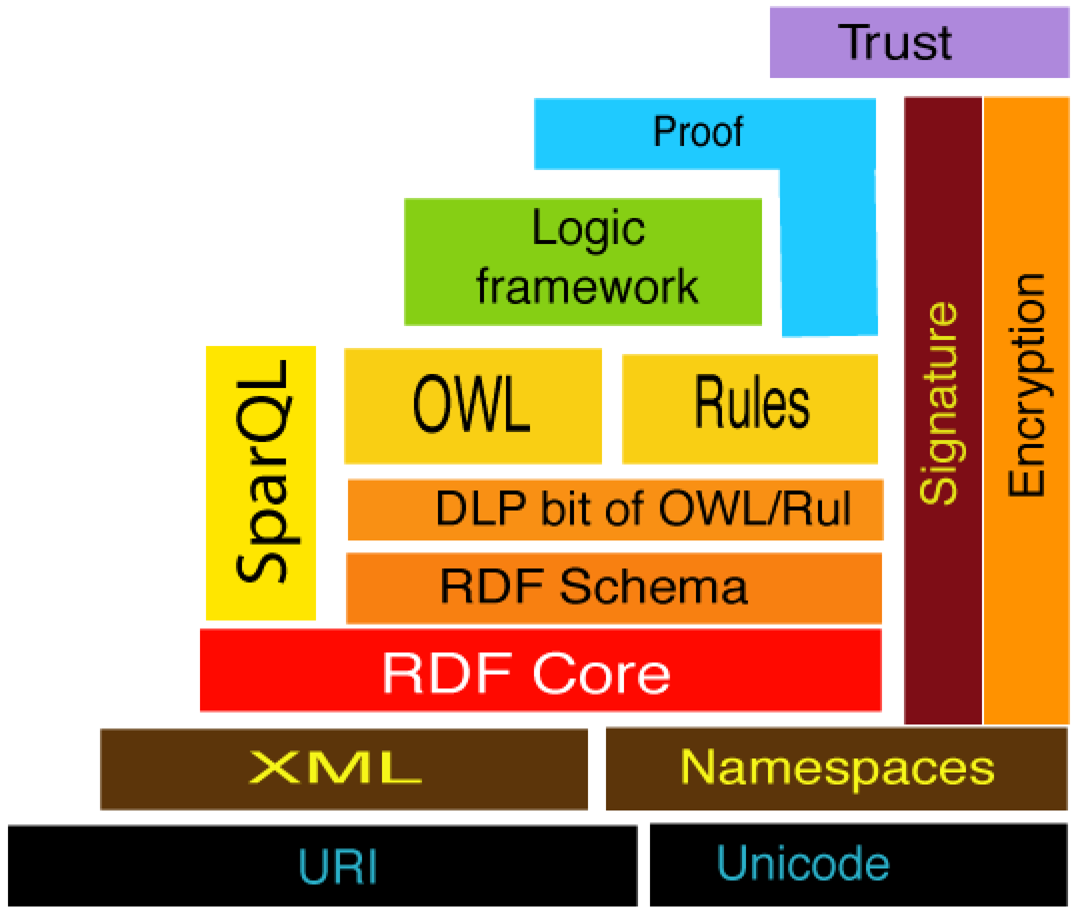
Analytics

- Managing updates
- Support for random access and indexing
- Scaleout content store
- Examples
  – BigTable, HBase, Hypertable

Why NoSQL?

- Many applications (especially in Web 2.0) need fewer features, high scale
- Sharding databases poses high management overhead
- High-scale relational systems too expensive for simple applications
- Need for a flexible data model
- Need for a low-latency, low-overhead API to access data
- Need to scale-out on cheap commodity nodes with locally attached SATA disks
- Increasing use of distributed analytics

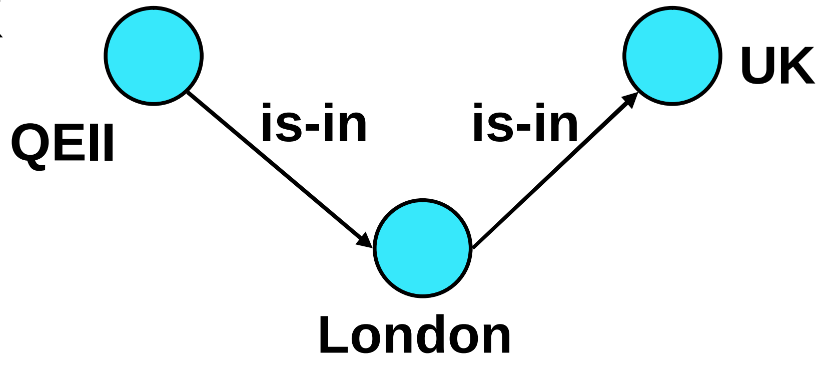
Semantic web



Source: http://www.w3.org/2005/Talks/1107-iswc-tbl/

What is Resource Definition Framework (RDF) or linked data? ...

- RDF is a family of w3 specifications
  - A mechanism for modelling information (often web-resources)
- RDF model
  - Information is described in the form of **Subject–Predicate–Object** expressions (Triples)
    e.g. QEII is-in London, London is-in UK

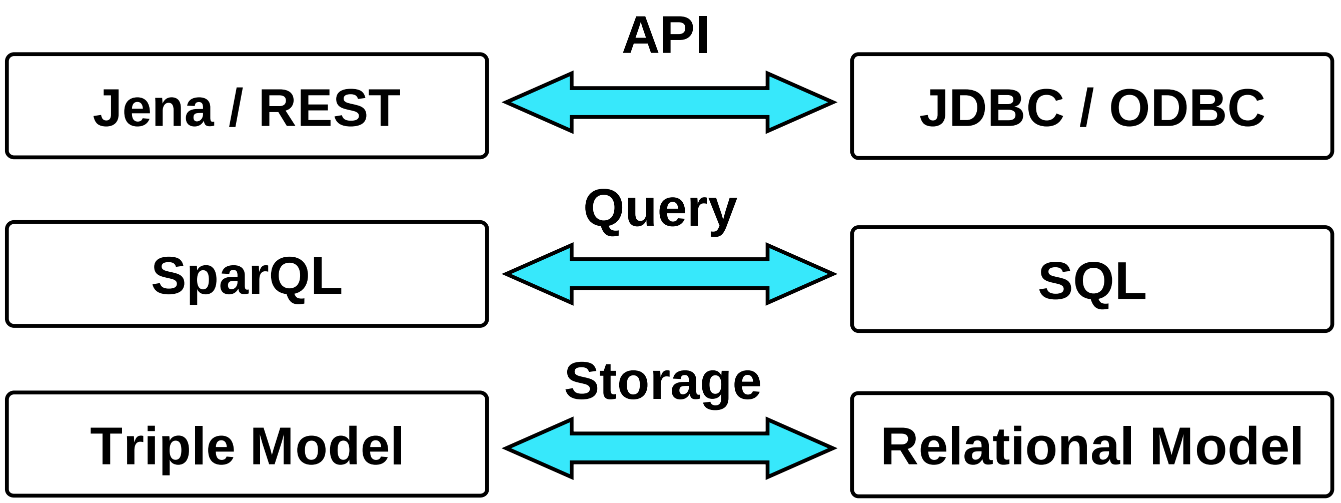


- Querying RDF
  - SparQL, which is SQL-like

```
e.g.  SELECT ?title
      WHERE { <http://example.org/book/book1>
                <http://purl.org/dc/elements/1.1/title> ?title . }
```

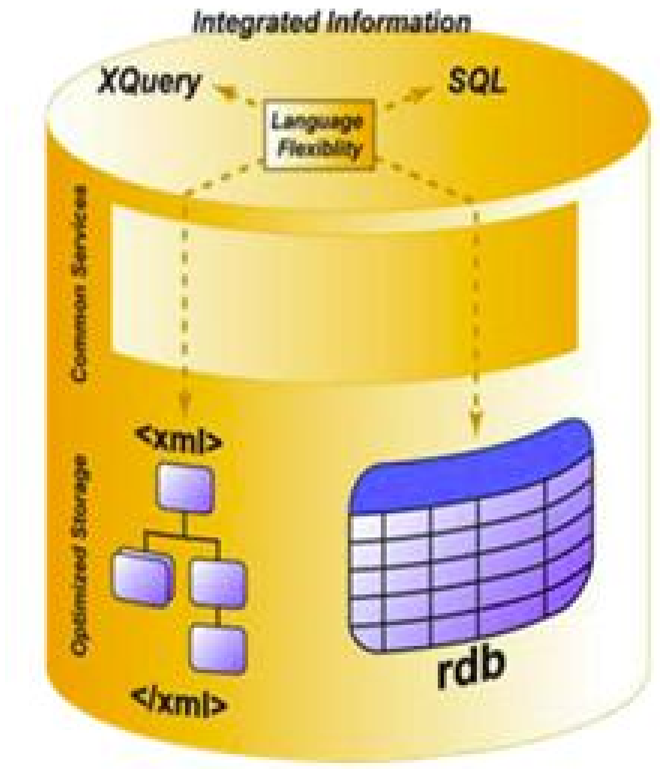
What is Resource Definition Framework (RDF) or linked data?

- RDF vs. Relational



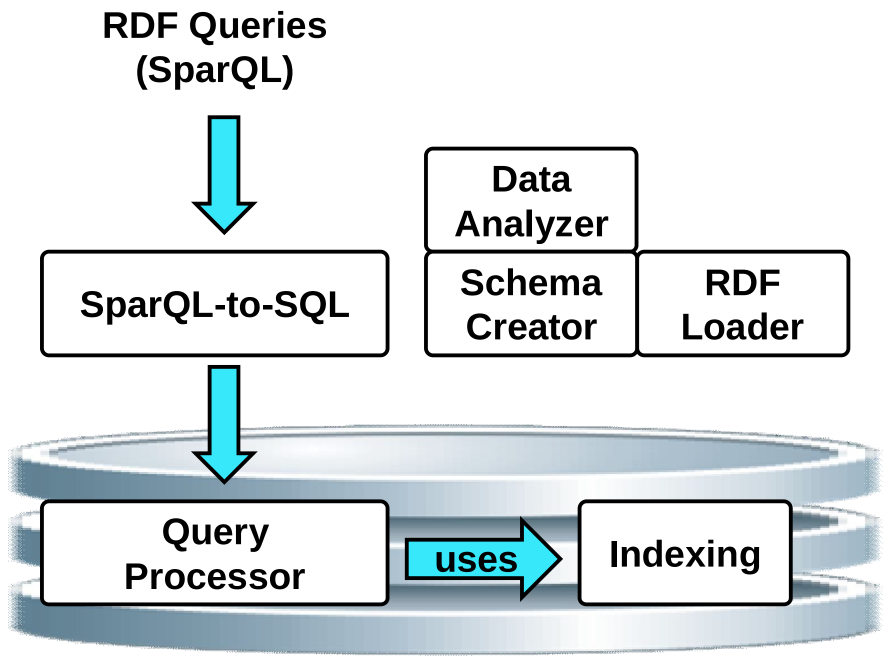
RDBMS and XML

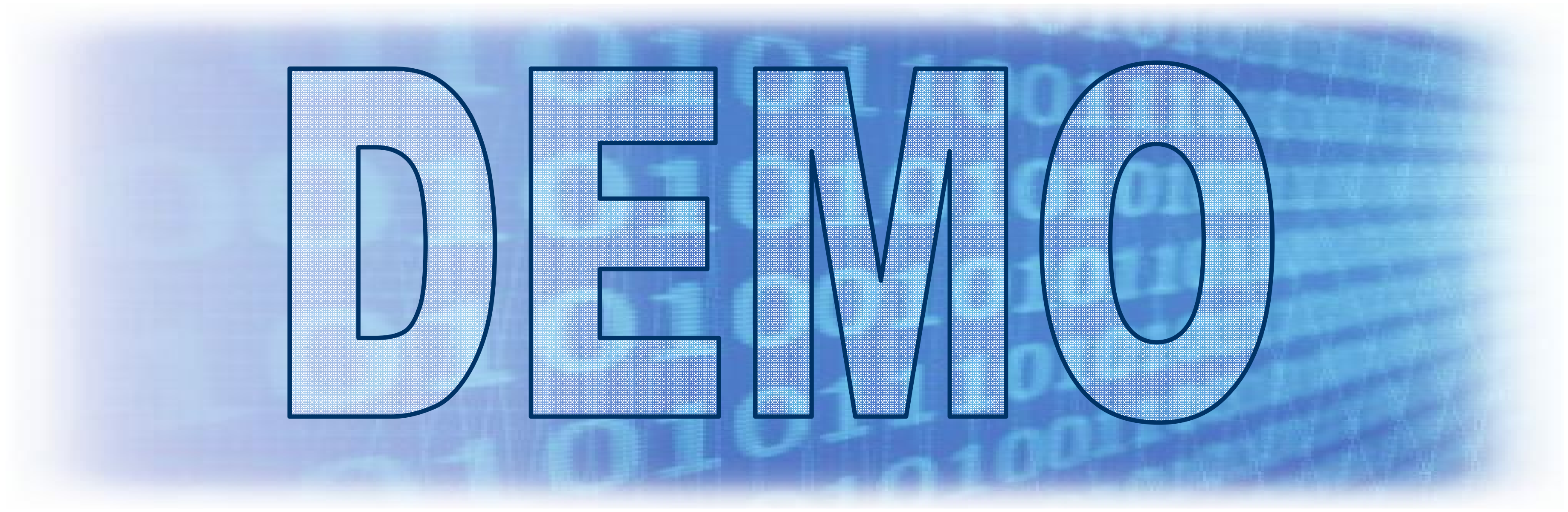
- Native XML storage, indexing and processing
- Tight integration of XML and relational data management
- XQuery, XPath, SQL/XML, XML Schema, XSLT, Namespaces, etc.
- Provide end-to-end XML
- Applications include creating business reports, SOA, web services, forms, etc.



RDBMS and RDF

- New mechanism to store Graph (RDF) data in relational systems
  - SparQL-to-SQL Translator
- Evaluating various API options





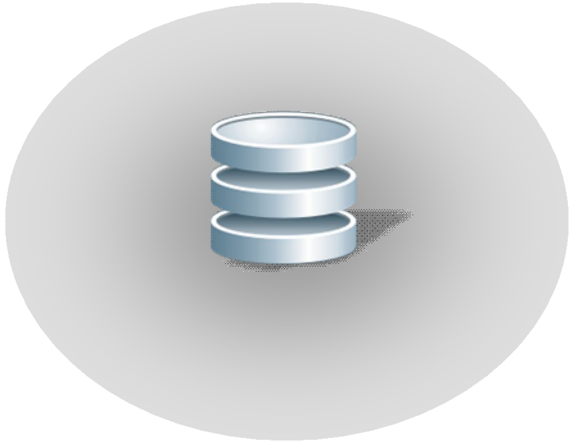
RDBMS and XML

NoSQL landscape

Document Stores
12 +
CouchDB, MongoDB

Graph Stores
11+
Jena, Sesame

XML Stores
9 +
Sedna, BaseX



Tabular Stores
6 +
HBase, Cassandra

Key Value Stores
23+
MemcacheDB, Redis

Object Stores
12 +
db4o

Others

Currently, there are more than 100+ NoSQL systems

Scalable datastores by Rick Cattell

- Key-value stores
  - Voldemort, Riak, Redis, Membase, Membrain, Dynamo
- Document stores
  - CouchDB, MongoDB, Terrastore, SimpleDB
- Extensible record stores
  - BigTable, HBase, HyperTable, Cassandra
- Scalable RDBMSs
  - MySQL Cluster, VoltDB, Clustrix, ScaleDB
- http://cattell.net/datastores/

What about performance?

- Yahoo Cloud Serving Benchmark (YCSB)
  - –http://research.yahoo.com/node/3202
- RDF Store Benchmarking
  - –http://www.w3.org/wiki/RdfStoreBenchmarking

Yahoo Cloud Serving Benchmark

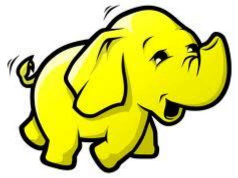
- Tested systems
  - Cassandra, HBase, Yahoo!'s PNUTS, sharded MySQL
- Tier 1 (performance)
  - Latency by increasing the server load
- Tier 2 (scalability)
  - Scalability by increasing the number of servers

PNUTS

Assertions for NoSQL

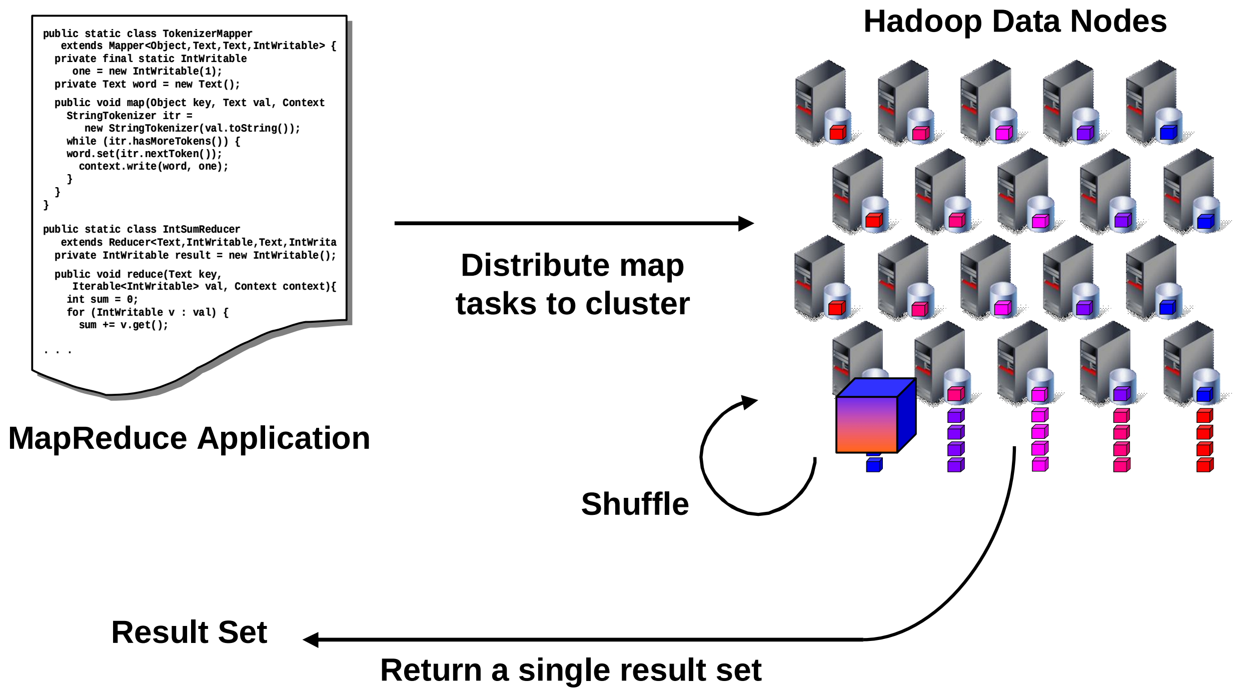
- There are some data management challenges that traditional systems do not handle well
  - For these challenges, emerging technologies like those under the NoSQL umbrella and those in the Big Data camp should be considered
  - These emerging technologies have been developed to handle specific challenges really well, and they are not positioned to address the broader general set of challenges that traditional systems excel at
- IT environments are usually centered around traditional database technology, but augmented with these emerging technologies where it makes sense
  - Integration of these emerging technologies with traditional systems is going to be a key consideration for IT departments
  - These emerging technologies need to be “enterprise ready”

What is Hadoop?



- One of the NoSQL frameworks
- Apache Hadoop = free, open source framework for data-intensive applications
  - Inspired by Google technologies (MapReduce, GFS)
  - Well-suited to batch-oriented, read-intensive applications
  - Originally built to address scalability problems of Nutch, an open source Web search technology
- Enables applications to work with thousands of nodes and petabytes of data in a highly parallel, cost effective manner
  - CPU + commodity disks = Hadoop "node"
  - Boxes can be combined into clusters
  - New nodes can be added as needed without changes to data formats, how data are loaded, how jobs are written

Hadoop explained



```
public static class TokenizerMapper
   extends Mapper<Object,Text,Text,IntWritable> {
  private final static IntWritable
     one = new IntWritable(1);
  private Text word = new Text();

  public void map(Object key, Text val, Context
    StringTokenizer itr =
       new StringTokenizer(val.toString());
    while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}

public static class IntSumReducer
   extends Reducer<Text,IntWritable,Text,IntWrita
  private IntWritable result = new IntWritable();

  public void reduce(Text key,
     Iterable<IntWritable> val, Context context){
    int sum = 0;
    for (IntWritable v : val) {
      sum += v.get();

. . .
```

1. **Map Phase**
   (break job into small parts)
2. **Shuffle**
   (transfer interim output for final processing)
3. **Reduce Phase**
   (boil all output down to a single result set)

Why is Hadoop important?

Traditional Approach

Structured & Repeatable Analysis

Business Users

Determine what question to ask

IT

Structures the data to answer that question

- Monthly sales reports
- Profitability analysis
- Customer surveys

Big Data Approach

Iterative & Exploratory Analysis

IT

Delivers a platform to enable creative discovery

Business Users

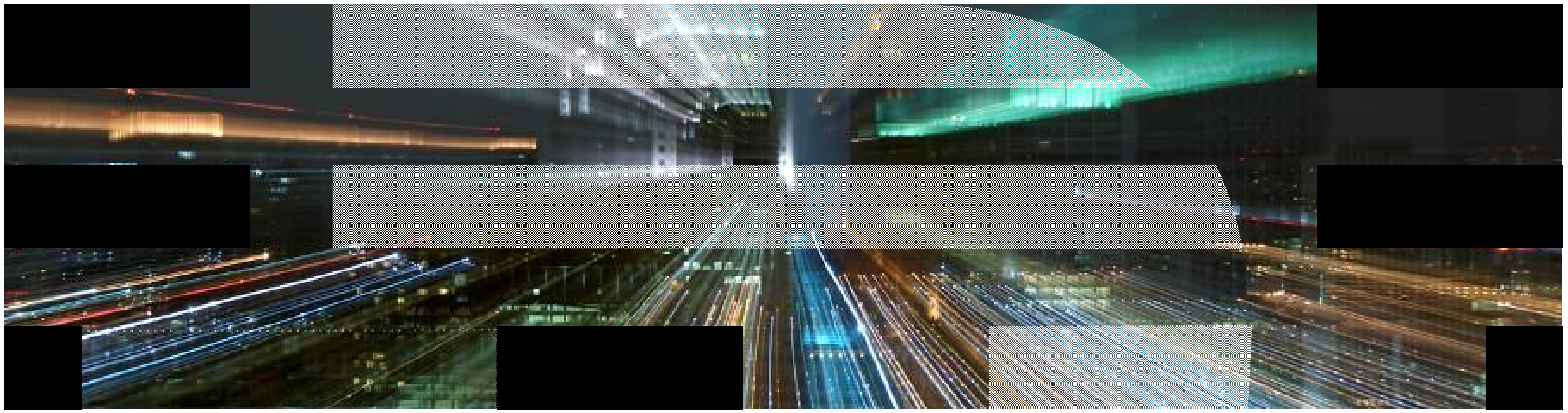
Explores what questions could be asked

- Brand sentiment
- Product strategy
- Maximum asset utilization

NoSQL Summary

Relational Database System	NoSQL Database System
Relational database systems store data in tables (relations) of information with keys to show relationships	Non-Relational Databases where data are contained in regular UNIX ASCII files
Data are accessed via SQL queries	To extract information, a file of data is fed to one or more "operators" via the UNIX Input/Output redirection mechanism (SQL enabled on some)
Data are structured with Data Definition Languages (DDLs), schemas with defined relationships based on keys between rows or tables	Columns of the row do not have a fixed schema and allow for rows to be completely unrelated
Mature database vendors with decades of experience	Typically, open source-based projects store “Big Data” in distributed Maps as Key/Value or Key/Object-based pairs
Row or column oriented storage	Graph, Document (XML) Store, Key-Value, Multivalue, Tabular, Object
Use SQL	Generally use other access methods

New SQL



What is New SQL?

- Old SQL / Old OLTP ("gold-standard in enterprise computing")
  - Collection of OLTP systems
  - Connected to Extract-Transform-Load (ETL)
  - Connected to one or more data warehouses
- New SQL / New OLTP ("web changes everything")
  - Need for far more OLTP throughput
  - Need for real-time analytics

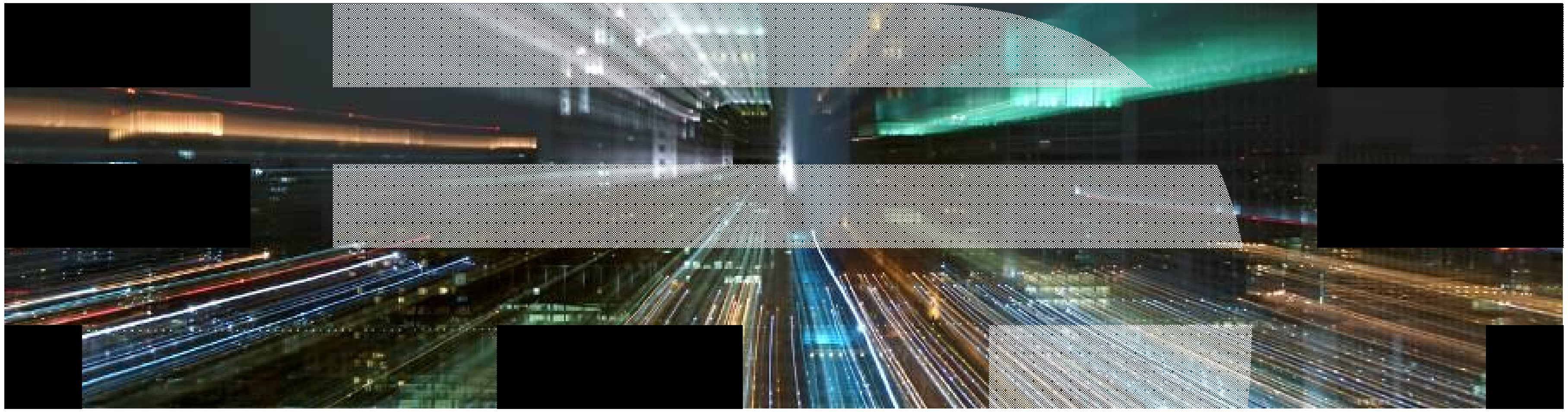
Source: http://cacm.acm.org/blogs/blog-cacm/109710-new-sql-an-alternative-to-nosql-and-old-sql-for-new-oltp-apps/

New SQL / New OLTP deployment options

- Use Old SQL
  - –New requirements may exceed capabilities of Old OLTP systems
  - –Data warehouses stale -- cannot provide real-time analytics
- Use NoSQL
  - –Lack of full ACID support
  - –Lack of SQL makes queries hard work
- Use New SQL
  - –Full ACID support
  - –Preserve SQL

Source: http://cacm.acm.org/blogs/blog-cacm/109710-new-sql-an-alternative-to-nosql-and-old-sql-for-new-oltp-apps/

Column-Oriented



What is a Column-Oriented (Columnar) database?

- In a Column-Oriented database, the data are stored by COLUMN instead of ROW; each COLUMN is an INDEX
- For compression in columnar databases, they can use the common compression for repeating, but differential compression techniques that take advantage of the fact that the data are always in the same order
- Operations that impact one or two columns -- such as MIN, MAX, SUM, COUNT and AVG -- can be performed very rapidly
- Not designed for real-time or dynamic warehouses because the column storage structures are not good for insert/update/delete operations; they are optimal for bulk load and query
- Columnar requires specialized skills, and a specialized knowledge of the data and how it will be requested

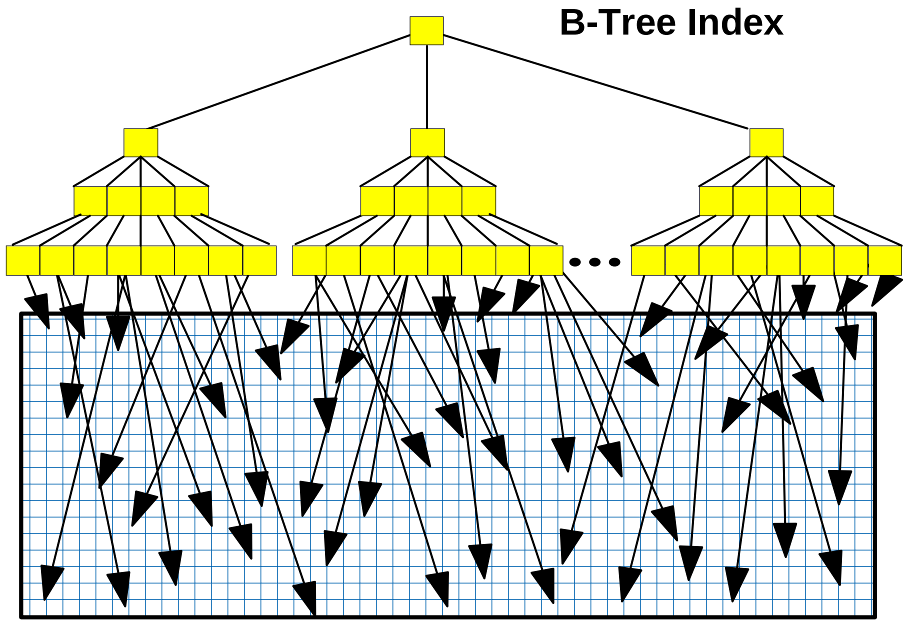
Why Column-Oriented?

- Columnar databases can be very good for analytic workloads
- Retrieving a small number of columns from a wide table
  - Many database tables can have upwards of 100 columns; most business questions only request a handful of columns
  - For very large tables and petabyte-sized systems, columnar databases are a good option
- Very good compression
- Significantly lower need for indexes, helps keep database smaller and up and running quickly

Example

Employee Table

Name	salary	pers_id	skill
Jones	**25000**	**8HJ9**	**student**
Jackson	**45000**	**XVJ4**	**expert**
Smith	**20000**	**2AB6**	**expert**
Snow	**99000**	**8HJ8**	**student**
Bruce	**35000**	**LK1A**	**student**



Example

Row Storage
Access one file

Name	salary	pers_id	skill
Jones	**25000**	**8HJ9**	**student**
Jackson	**45000**	**XVJ4**	**expert**
Smith	**20000**	**2AB6**	**expert**
Snow	**99000**	**8HJ8**	**student**
Bruce	**35000**	**LK1A**	**student**

Column Storage
Access multiple files

Name
Jones
Jackson
Smith
Snow
Bruce

salary
25000
45000
20000
99000
35000

pers_id
8HJ9
XVJ4
2AB6
8HJ8
LK1A

skill
student
expert
expert
student
student

Use cases

- Retail
  - Market basket analysis -- MIN / MAX / COUNT / AVG across all stores
  - Correlate transaction ID and Stock Keeping Unit (SKU)
- Telcos
  - Looking at the specific state of a cell tower or tower activity
- Insurance
  - Total number of policies in an geographic area
  - Value of policies in an area

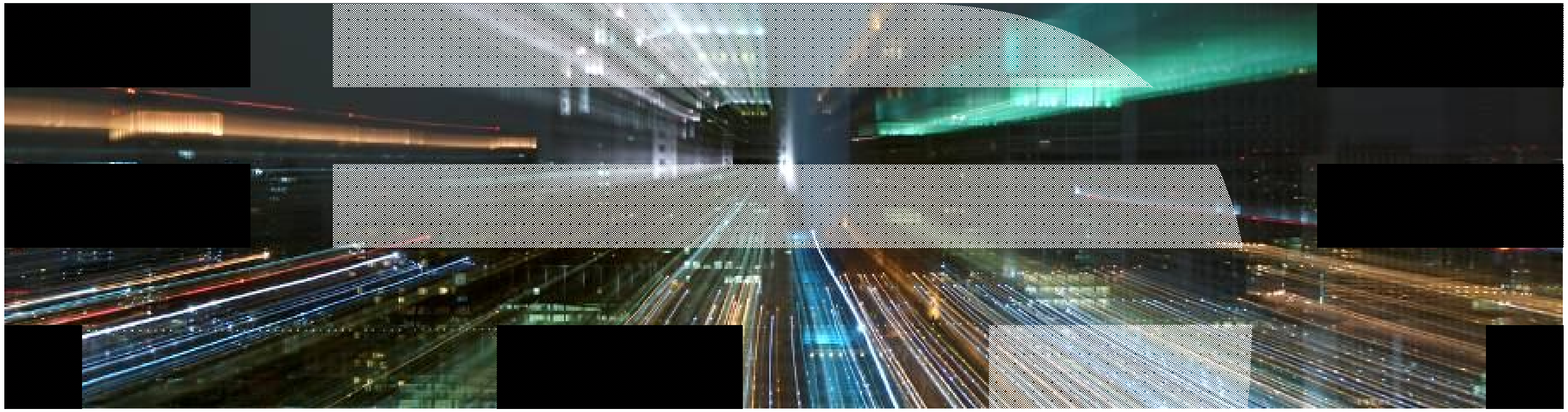
Assertions for Column-Oriented

- Columnar is poor at full table loads
- Good if you are doing one to two column aggregates
- Columnar cannot do real time Business Intelligence because they do not handle updates which is something that is increasingly required by businesses
- Good for certain workloads; choose those workloads with care

Column-Oriented Summary

Row-Oriented Database System	Column-Oriented Database System
All data stored in a table with columns together/ contiguously on pages that can contain many rows	All data are stored by column instead of row and that each column is like an index
Database size based on data stored and index, etc.	Database size is based on the data stored
Optimal for read/write/update/delete (OLTP and OLAP)	Optimal for read workloads (OLAP)
Compression across values in a table, page or rows	Various compression techniques
Row operations and joins between tables can be performed very rapidly	Columnar operations -- MIN, MAX, SUM, COUNT and AVG -- can be performed very rapidly

In-Memory



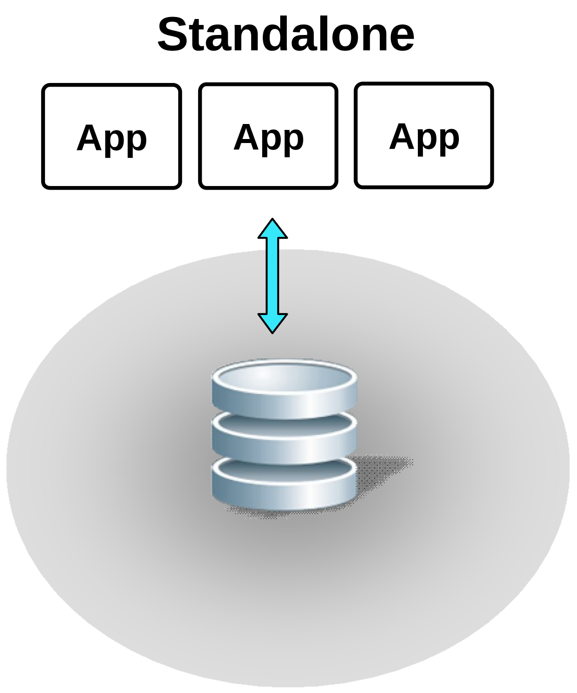
What is an In-Memory database?

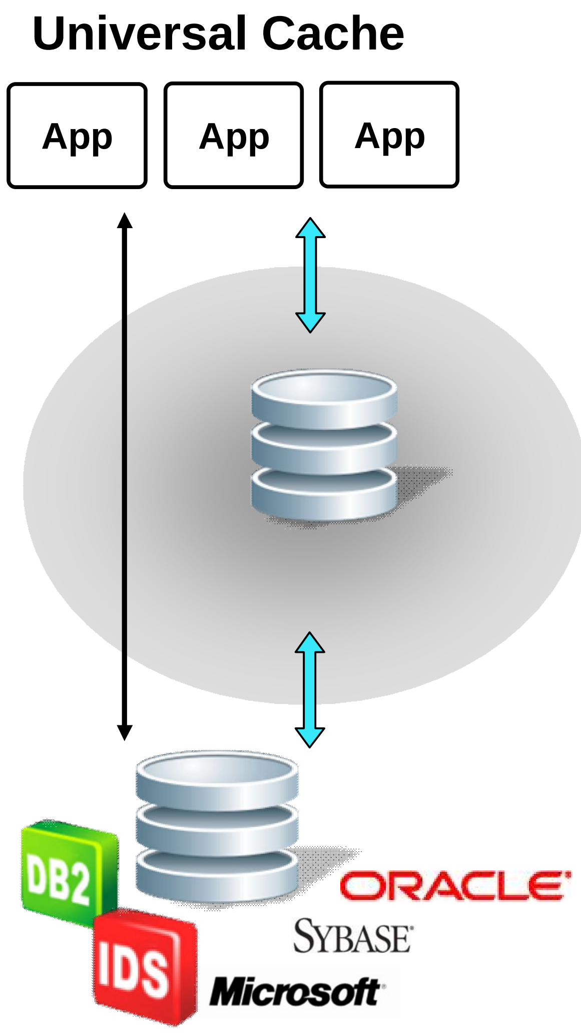
- In-memory databases use RAM to house a database
- RAM is lightening quick and no need to access storage -- save on both CPU and I/O
- Most in-memory databases will be Gigabytes not Terabytes in size
- In-memory databases outperform traditional databases with large bufferpools (or caches)
  - Even if the entire database is cached, there is additional overhead in a traditional database for the database manager to lookup the location of the data page
  - In-memory databases, the data are always in-memory, so a lookup step is never required
  - In-memory write operations are faster because they don’t need to store it to disk before making it available in main memory

Why In-Memory?

- Business pain points
  - Deliver near real time response to a massive and growing number of queries
  - Capture and store growing number of events
  - Process large data sets efficiently
  - Quick response to user driven interactive business transactions, such as credit card checks, fraud detection, cross sell, up-sell to web customers

Example


Standalone
App
App
App


Universal Cache
App
App
App
DB2
IDS
ORACLE
SYBASE
Microsoft

Use cases

Communications

Financial Services

Web 2.0

Online Charging

- Authenticate and authorize
- Initiate service
- Manage credit balance
- Manage volume discounts

Brokerage Application

- Receive market feed
- Evaluate equity positions
- Check for fraud

Online Retail Web Site

- Authenticate user
- Manage personal wishlists
- Generate page contents with cross-sell data

- **100,000s to 1,000,000s of concurrent requests**
- **10s of microseconds for database calls**

- **Evaluate 30,000+ rules on 500 trades per second for 15 million trades per day**

- **Facebook: 10,000,000 concurrent sessions = two billion page views a day**
- **Wikipedia: 3000 page views a second and 25,000 SQL requests per second**

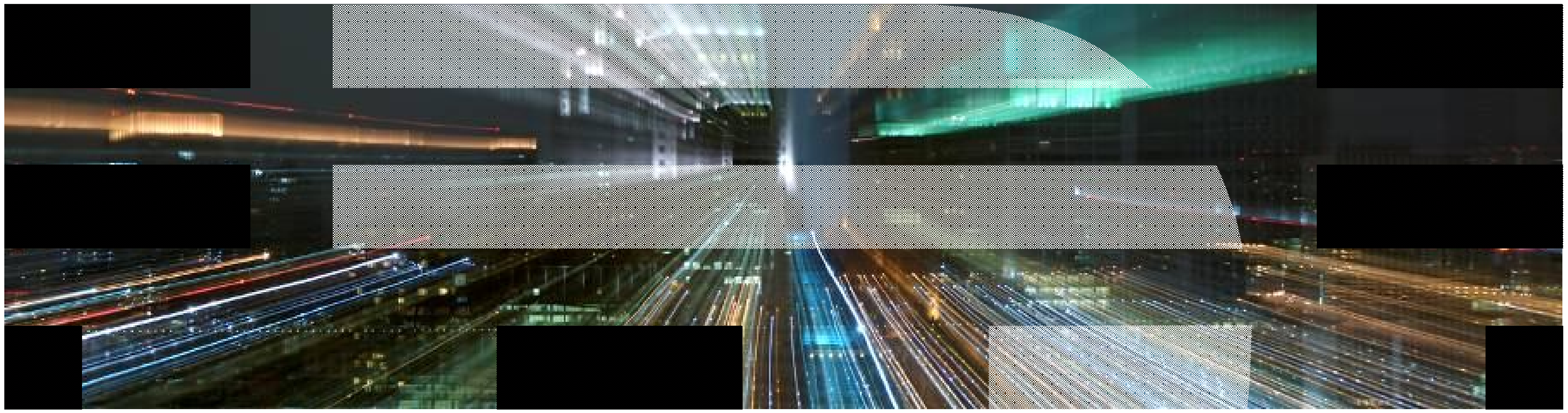
Assertions for In-Memory

- In-Memory outperforms traditional
  - High volume OLTP
  - Sub-millisecond latency with extreme throughput requirements
  - Data or logical subset of the data fits into RAM (a must)
  - Data can be co-located with the application
  - Small transactions, small to medium row sizes (no large BLOBs)
  - Application does a lot of point lookups (exact key match)
- In-Memory on par with traditional
  - Table scans and non-indexed queries
  - GROUP BY and ORDER BY for big (thousands of rows) result sets
  - Complex or less optimal SQL

In-Memory Summary

On-Disk Database System	In-Memory Database System
All data stored on disk, disk I/O needed to move data into main-memory when needed	All data stored in main-memory, no need to perform disk I/O to query or update data
Data are always persisted to disk	**Data are persistent or volatile depending on the in-memory database product**
Traditional data structures like B-Trees designed to store tables and indexes efficiently on disk	Specialized data structures and index structures assume data are always in main-memory
Support very broad set of workloads, such as OLTP, data warehousing, mixed workloads, etc.	**Optimized for specialized workloads**
Virtually unlimited database size (order of Terabytes, Petabytes)	Database size limited by the amount of main-memory (Gigabytes)

Summary



The Great Debate

“ About every ten years or so, there is a “great debate” between, on the one had, those who see the problem of data modelling through a more or less relational lens, and on the other, a noisier set of “refuseniks” who have a hot new thing to promote. The debate usually goes like this:

Refuseniks: Hah! You relational people with your flat tables and silly query languages! You are so unhip! You simply cannot deal with the problem of [INSERT NEW THING HERE]. With an [INSERT NEW THING HERE]-DBMS we will finish you, and grind your bones into dust!

R-people: You make some good points. But unfortunately a) there is an enormous amount of money invested in building scalable, efficient and reliable database management products and no one is going to drop all of that on the floor and b) you are confusing DBMS engineering decisions with theoretical questions. We plan to incorporate the best of these ideas into our products. ” -- Paul Brown

History repeats itself

	[INSERT NEW THING HERE]	RDBMS
20 years ago	**Object-Oriented Database Management Systems**	**Added support for objects, extensibility, so on**
10 years ago	**Native XML Database Management Systems**	**Added support for XML, XQuery, so on**
Today	**NoSQL**	**Adding support for RESTful interfaces, RDF, so on**

Some lessons from history

- Relational technology is good for general purpose database needs
- New technology is often used around the edges to complement relational
- Need good development tools
- Need good development skills
- Need a corporate champion
- Choose the right product for the job

What is different today?

- Technology from major vendors
  - Amazon (Dynamo, SimpleDB)
  - Facebook (Cassandra)
  - Google (BigTable)
- Strong and vibrant open-source community
- Major relational vendors using open source technology
  - Product offerings
  - Consulting services

How to contact me

- LinkedIn
  - http://uk.linkedin.com/in/akmalchaudhri
- IBM
  - *firstname.lastname*@uk.ibm.com

Thank You