The NoSQL RDBMS

One of first uses of the phrase *NoSQL* is due to Carlo Strozzi, circa 1998.

**NoSQL:**
- A fast, portable, open-source RDBMS
- A derivative of the RDB database system (Walter Hobbs, RAND)
- Not a full-function DBMS, per se, but a shell-level tool
- User interface – Unix shell
- Based on the “operator/stream paradigm”
NoSQL Today

More recently:
- The term has taken on different meanings
- One common interpretation is “not only SQL”

Most modern NoSQL systems diverge from the relational model or standard RDBMS functionality:

<table>
<thead>
<tr>
<th>The data model:</th>
<th>relations</th>
<th>documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>tuples</td>
<td>vs.</td>
<td>graphs</td>
</tr>
<tr>
<td>attributes</td>
<td></td>
<td>key/values</td>
</tr>
<tr>
<td>domains</td>
<td></td>
<td></td>
</tr>
<tr>
<td>normalization</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The query model:</th>
<th>relational algebra</th>
<th>graph traversal</th>
</tr>
</thead>
<tbody>
<tr>
<td>tuple calculus</td>
<td>vs.</td>
<td>text search</td>
</tr>
<tr>
<td></td>
<td></td>
<td>map/reduce</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The implementation:</th>
<th>rigid schemas</th>
<th>flexible schemas</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACID compliance</td>
<td>vs.</td>
<td>(schema-less)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BASE</td>
</tr>
</tbody>
</table>

In that sense, NoSQL today is more commonly meant to be something like “non-relational”
NoSQL Today
(a partial, unrefined list)

Hbase  Cassandra  Hypertable  Accumulo  Amazon SimpleDB  ScidB  Stratosphere  flare
Clouddata  Bigtable  QD Technology  SmartFocus  KDI  Alterian  Cloudera  C-Store
Vertica  Qbase–MetaCarta  OpenNeptune  HPCC  Mongo DB  CouchDB  Clusterpoint ServerTerrastore
Jackrabbit  OrientDB  Perservere  CoudKit  Djondb  SchemaFreeDB  SDB  JasDB
RaptorDB  ThruDB  RavenDB  DynamoDB  Azure Table Storage  Couchbase Server  Riak
LevelDB  Chordless  GenieDB  Scalaris  Tokyo  Kyoto Cabinet  Tyrant  Scalien
Berkeley DB  Voldemort  Dynomite  KAI  MemcacheDB  Faircom C-Tree  HamsterDB  STSdb
Tarantool/Box  Maxtable  Pincaster  RaptorDB  TIBCO Active Spaces  allegro-C  nessDBHyperDex
Mnesia  LightCloud  Hibari  BangDB  OpenLDAP/MDB/Lightning  Scality  Redis
KaTree  TomP2P  Kumofs  TreapDB  NMDB  luxio  actord  Keyspace
schema-free  RAMCloud  SubRecord  Mo8onDb  DovetailDb JDBM  Neo4  InfiniteGraph
Sones  InfoGrid  HyperGraphDB  DEX  GraphBase  Trinity  AllegroGraph  BrightstarDB
Bigdata  Meronomy  OpenLink  Virtuoso  VertexDB  FlockDB  Execom  IOG  Java Univ Netwrk/Graph Framework
OpenRDF/Sesame  Filament  OWLim  NetworkX  iGraph  Jena  SPARQL  OrientDb
ArangoDB  AlchemyDB  Soft NoSQL Systems  Db4o  Versant  Objectivity Starcounter
ZODB  Magma  NEO  PicoList  siaqodb  Sterling  Morantex  EyeDB
HSS Database  FramerD  Ninja Database Pro  StupidDB  KikuDB  Perl solution  Durus
GigaSpaces  Infinispan  Queplix  Hazelcast  GridGain  Galaxy  SpaceBase  JoafipCoherence
eXtremeScale  MarkLogic Server  EMC Documentum  xDB  eXist  Sedna  BaseX  Qizx
Berkeley DB XML  Xindice  Tamino  Globals  Intersystems Cache  GT.M  EGTM
U2  OpenInsight  Reality  OpenQm  ESENT  jBASE  MultiValue  Lotus/Domino
eXtremeDB  RDM Embedded  ISIS Family  Prevayler  Yserial  Vmware vFabric GemFire  Btrieve
KirbyBase  Tokutek  Recutils  FileDB  Armadillo  illuminate Correlation Database  FluidDB
Jeeet DB  Twisted Storage  Rindo  Sherpa  tin  Dryad  SkyNet  Disco
MUMPS  Adabas  XAP In-Memory Grid  eXtreme Scale  MckoiDDB  Mckoi SQL Database
Primary NoSQL Categories

- General Categories of NoSQL Systems:
  - Key/value store
  - (wide) Column store
  - Graph store
  - Document store

- Compared to the relational model:
  - Query models are not as developed.
  - Distinction between abstraction & implementation is not as clear.
Key/Value Store


The basic data model:
- Database is a collection of key/value pairs
- The key for each pair is unique

Primary operations:
- `insert(key, value)`
- `delete(key)`
- `update(key, value)`
- `lookup(key)`

Additional operations:
- variations on the above, e.g., reverse lookup
- iterators

No requirement for normalization (and consequently dependency preservation or lossless join)
Wide Column Store


The basic data model:

- Database is a collection of key/value pairs
- Key consists of 3 parts – a row key, a column key, and a time-stamp (i.e., the version)
- Flexible schema - the set of columns is not fixed, and may differ from row-to-row

One last column detail:

- Column key consists of two parts – a column family, and a qualifier

Warning #1!
Wide Column Store

Column families

Row key

Personal data

Professional data

Column qualifiers
## Wide Column Store

### One “table”

<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Date of Birth</th>
<th>Job Category</th>
<th>Salary</th>
<th>Date of Hire</th>
<th>Employer</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Middle Name</th>
<th>Last Name</th>
<th>Job Category</th>
<th>Employer</th>
<th>Hourly Rate</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Job Category</th>
<th>Salary</th>
<th>Employer</th>
<th>Group</th>
<th>Seniority</th>
<th>Bldg #</th>
<th>Office #</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Last Name</th>
<th>Job Category</th>
<th>Salary</th>
<th>Date of Hire</th>
<th>Employer</th>
<th>Insurance ID</th>
<th>Emergency Contact</th>
</tr>
</thead>
</table>
Wide Column Store

One “row” in a wide-column NoSQL database table

= Many rows in several relations/tables in a relational database
Graph Store


The basic data model:

Directed graphs

Nodes & edges, with properties, i.e., “labels”
Document Store


The basic data model:

☐ The general notion of a document – words, phrases, sentences, paragraphs, sections, subsections, footnotes, etc.

☐ Flexible schema – subcomponent structure may be nested, and vary from document-to-document.

☐ Metadata – title, author, date, embedded tags, etc.

☐ Key/identifier.

One implementation detail:

☐ Formats vary greatly – PDF, XML, JSON, BSON, plain text, various binary, scanned image.
ACID vs. BASE

Database systems traditionally support ACID requirements:

- Atomicity, Consistency, Isolation, Durability

In a distributed web applications the focus shifts to:

- Consistency, Availability, Partition tolerance

CAP theorem - At most two of the above can be enforced at any given time.

Reducing consistency, at least temporarily, maintains the other two.

Thus, distributed NoSQL systems are typically said to support some form of BASE:

- Basic Availability
- Soft state
- Eventual consistency*