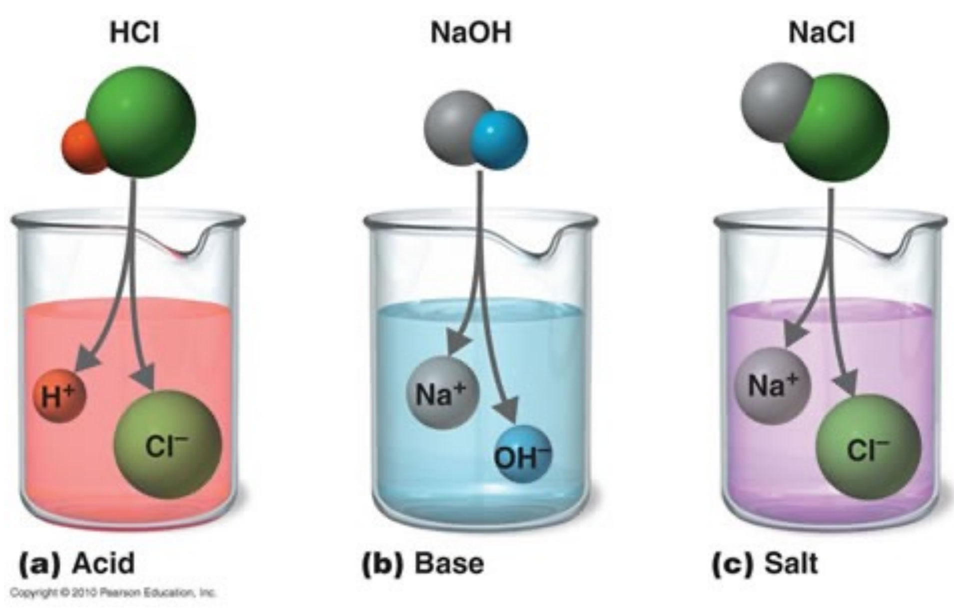
Transaction Properties: ACID vs. BASE



Client/Server with Transaction Processing

- *Transactions are a way to make ACID operations a general commodity*
  [Transaction Processing Concepts and Techniques, Jim Gray and Andreas Reuter, 1993]

- ***Atomicity***
  - a transaction is an indivisible unit of work
  - an all-or-nothing proposition
  - all updates to a database, displays on the clients' screens, message queues
  - e.g., salary increase for all 1 million employees or none

- ***Consistency***
  - a transaction is an indivisible unit of work
  - S -> [T | abort] -> S
  - integrity constraints (e.g., mgr.salaray > salary)

- ***Isolation***
  - a transaction's behavior not affected by other transactions running concurrently
  - e.g., reserve a seat
  - serialization techniques

- ***Durability***
  - persistence
  - a transaction's effects are permanent after it commits.

ACID is like motherhood and apple pie. It's necessary- and you can't have too much of it. OK, enough preaching.

ACID Properties: Failures

- Isolation Failure:

T_1 transfers 10 from A to B. T_2 transfers 10 from B to A. Combined, there are four actions:

  - T_1 subtracts 10 from A.
  - T_1 adds 10 to B.
  - T_2 subtracts 10 from B.
  - T_2 adds 10 to A.

  By interleaving the transactions, the actual order of actions might be:

  - T_1 subtracts 10 from A.
  - T_2 subtracts 10 from B.
  - T_2 adds 10 to A.
  - T_1 adds 10 to B.

  Now, consider what happens if T_1 fails halfway through.

ACID Properties: Failures

- Durability Failure:

  T_1 transfers 10 from A to B.

  - It removes 10 from A. It then adds 10 to B.
  - At this point, a "success" message is sent to the user.
  - However, the changes are still queued in the disk buffer waiting to be committed to the disk.
  - Power fails and the changes are lost.
  - The user assumes (understandably) that the changes have been made.

CAP Theorem (Brewer's Theorem)

- it is impossible for a ***distributed*** computer system to simultaneously provide all three of the following guarantees:
  - *Consistency*: all nodes see the same data at the same time
  - *Availability*: Node failures do not prevent other survivors from continuing to operate (a guarantee that every request receives a response about whether it succeeded or failed)
  - *Partition tolerance*: the system continues to operate despite arbitrary partitioning due to network failures (e.g., message loss)
- A distributed system can satisfy any two of these guarantees at the same time but not all three.

CAP Theorem (Brewer's Theorem): So, what can be done?

- In a distributed system, a network (of networks) in inevitable (by definition).
- Failures can, and will, occur to a networked system -> partitioned tolerance should be accommodated.
- Then, the only option left is choosing between Consistency and Availability- i.e., CA doesn't make any sense (expect when we have, e.g., a single-site databases; 2-phase commit, cache validation protocols))
- Not necessarily in a mutually exclusive manner, but possibly by partial accommodation of both -> trade-off analyses important

  [In terms of architecture, what ontological concept is this about?]

- AP: A partitioned node returns

  a correct value, if in a consistent state;

  a timeout error or an error, otherwise

  e.g., DynamoDB, CouchDB, and Cassandra
- CP: A partitioned note returns the most recent version of the data, which could be stale.

  e.g., MongoDB, Redis, AppFabric Caching, and MemcacheDB

BASE

(Basically Available, Soft-State, Eventually Consistent)

- **SQL databases:**
  - Structured query language for
  - Traditional relational databases (unique keys, single valued, no update/insertion/deletion anomalies)
  - Well structured data
  - ACID properties should hold
- **NoSQL (Not only SQL) databases:**
  - triggered by the storage needs of Web 2.0 companies such as Facebook, Google and Amazon.com
  - Not necessarily well structured – e.g., pictures, documents, web page description, video clips, etc.
  - Lately of increasing importance due to big data
  - ACID properties may not hold -> no properties at all then???
  - focuses on availability of data even in the presence of multiple failures
  - spread data across many storage systems with a high degree of replication.

BASE

(Basically Available, Soft-State, Eventually Consistent)

- **Rationale:**
  - It's ok to use stale data (Accounting systems do this all the time. It's called "closing out the books.") ; it's ok to give approximate answers
  - Use resource versioning -> say what the data really is about – no more, no less.

    The value of x is 5, at time T and date D

  - So, shift the PH from 0-6 (acidic) to 8-14 (basic) – pure water's PH is 7 and neutral

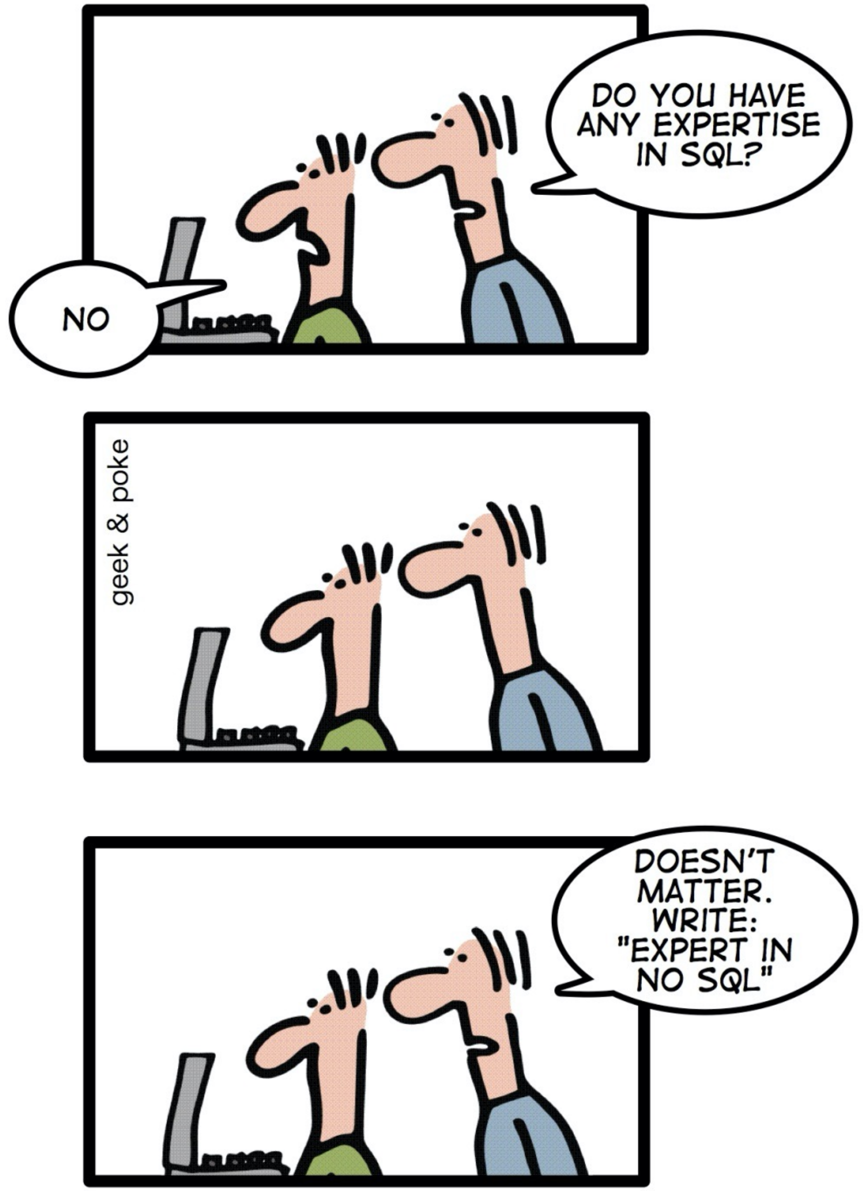
- **Can some compromise be made between C and A?:**
  - instead of completely giving up on C, for A
  - <u>Instead of completely giving up on A, instead of C</u>

BASE

(Basically Available, Soft-State, Eventually Consistent)

- Basic Availability: fulfill request, even in partial consistency.
- Soft State: abandon the consistency requirements of the ACID model pretty much completely
- Eventual Consistency: at some point in the future, data will converge to a consistent state; delayed consistency, as opposed to immediate consistency of the ACID properties.
  - purely a liveness guarantee (reads eventually return the requested value); but
  - does not make safety guarantees, i.e.,
  - an eventually consistent system can return any value before it converges

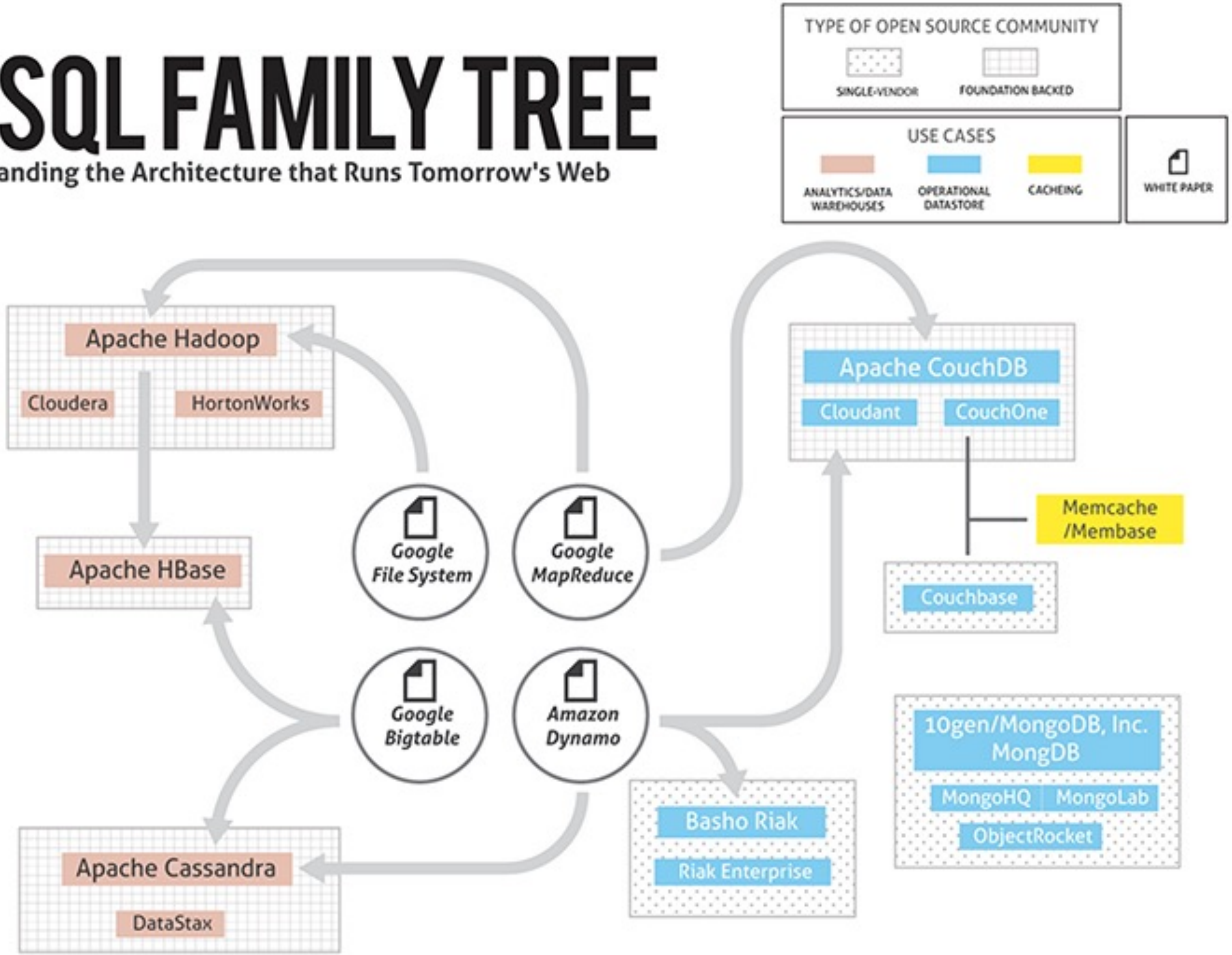
HOW TO WRITE A CV



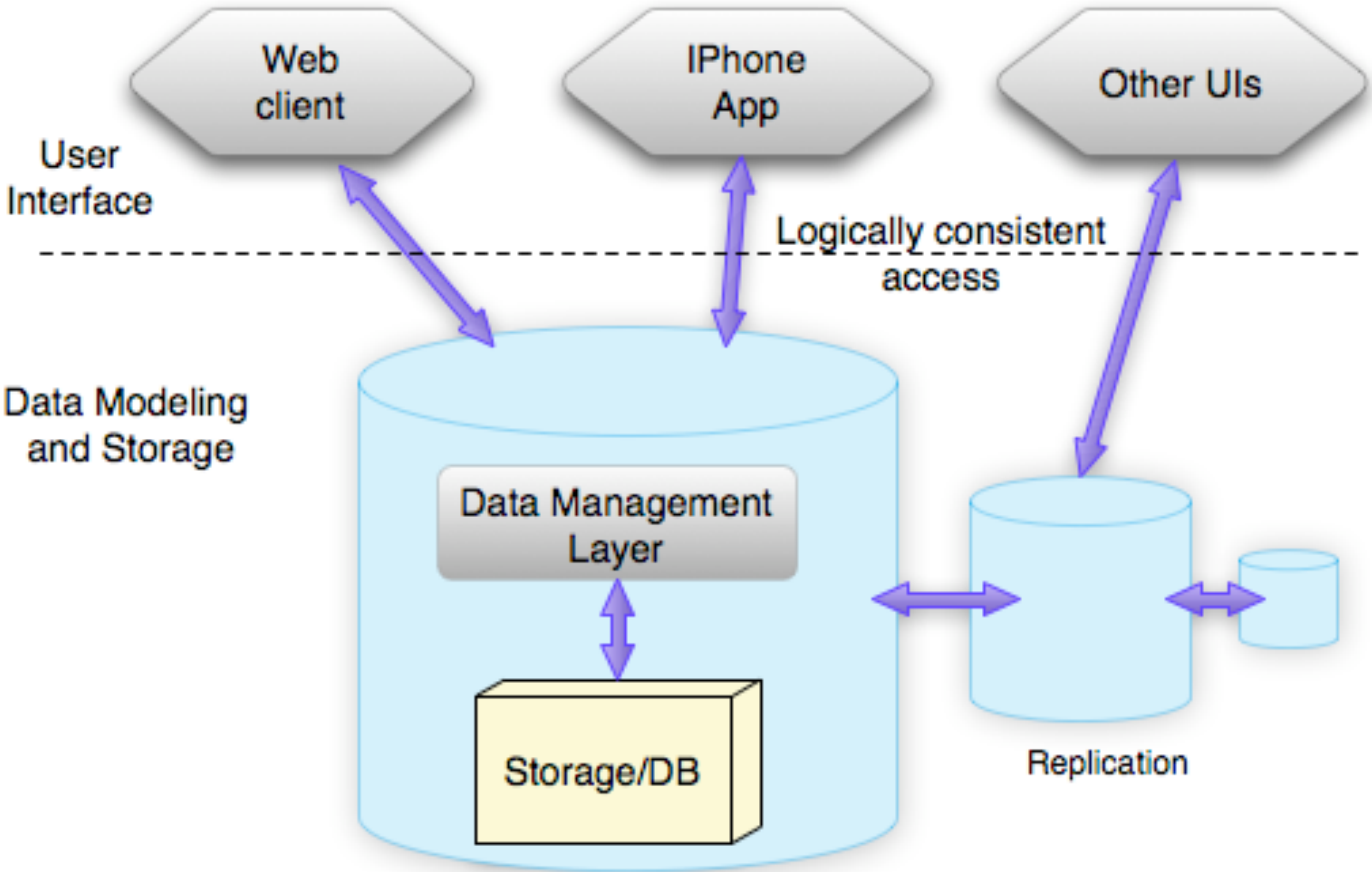
Leverage the NoSQL boom

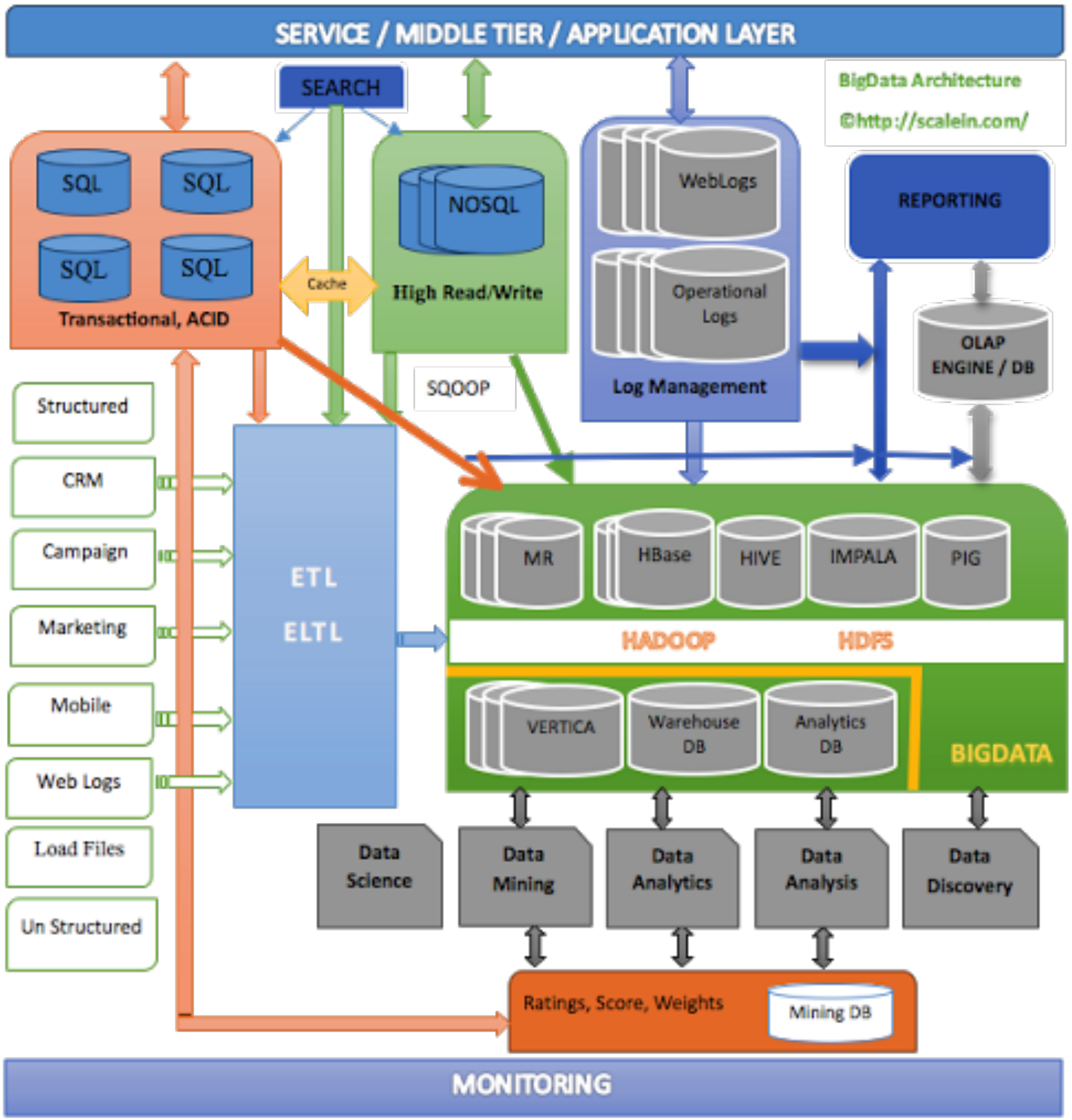
NoSQL FAMILY TREE

Understanding the Architecture that Runs Tomorrow's Web



•


Web client
IPhone App
Other UIs
User Interface
Logically consistent access
Data Modeling and Storage
Data Management Layer
Storage/DB
Replication


SERVICE / MIDDLE TIER / APPLICATION LAYER
SEARCH
BigData Architecture
©http://scalein.com/
SQL
SQL
SQL
SQL
Transactional, ACID
Cache
NOSQL
High Read/Write
WebLogs
Operational Logs
Log Management
REPORTING
OLAP ENGINE / DB
SQOOP
Structured
CRM
Campaign
Marketing
Mobile
Web Logs
Load Files
Un Structured
ETL
ELTL
MR
HBase
HIVE
IMPALA
PIG
HADOOP
HDFS
VERTICA
Warehouse DB
Analytics DB
BIGDATA
Data Science
Data Mining
Data Analytics
Data Analysis
Data Discovery
Ratings, Score, Weights
Mining DB
MONITORING

s

a