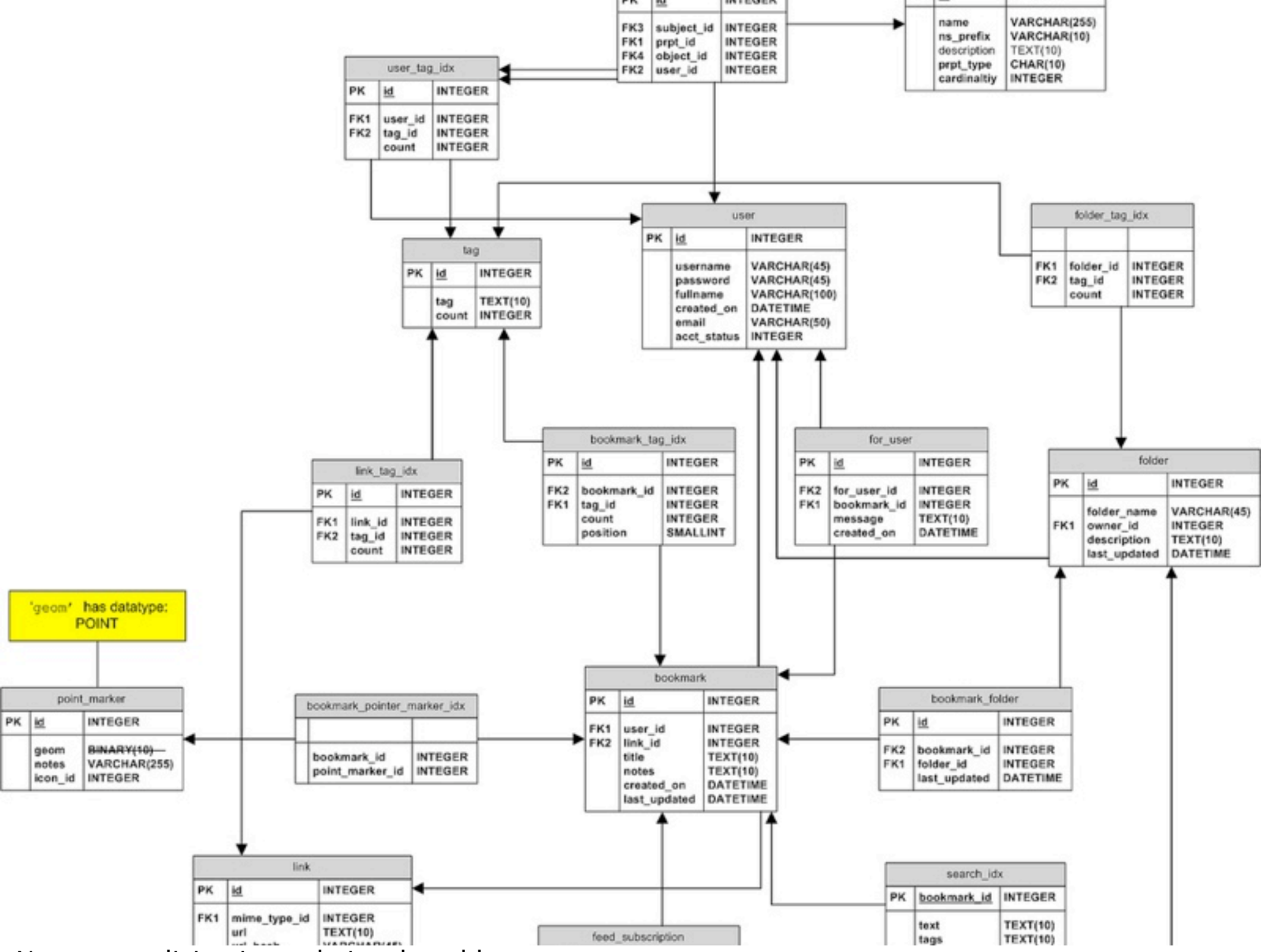
CASSANDRA

Your New Best Girl

Toby DiPasquale | Philly ETE 2010

In the beginning, we had paper filing systems
Library == database, card catalog == index

Then came mainframes and hierarchical (ISAM) storage
ISAM == Indexed Sequential Access Method



Now we are living in a relational world
Default tool for most engineers for most jobs

What’s wrong with RDMBS? Served us well for a long time
Too often are default tool without thinking about the constraints

Consistency is king

Favors ACID consistency above all other considerations
Doesn't allow for CAP tradeoffs (by design)

Difficult to scale

3NF makes it difficult to partition data across physical instances
JOINs have to be denormalized in order to shard, losing consistency

Complex Implementations

General case architecture makes it slow, many features not needed for a given app
Operations overhead is very high, most complex software in most apps

Consistency

Availability

Partition-Tolerance

you can pick two
framework for understanding tradeoffs in distributed system design

RDBMS* = CA

* Using asynchronous replication

RDBMS forces you to give up partition-tolerance
This is a bad choice for large distributed systems

What if I need something else?


Cassandra

Dynamo + BigTable = Cassandra

Takes AP/architecture from Dynamo and column-orientation/access patterns from BigTable

Originally created by Facebook, dumped source, abandoned it
Now top-level Apache project

Pros

Column-oriented

Data is stored by ColumnFamily
Makes it fast for querying, better able to be compressed

No SPoF

All nodes are equal, all service queries
Node will proxy to other node if asking for data it doesn't have

Auto Scaling

(fo realz)

Just add more nodes with same config
New nodes request data from old ones and split ring amongst themselves

Read Repair

Will choose replica with latest timestamp in case of conflict (only when R > 1)
Requires that **client clocks be synchronized** to work right

$$R + W > N$$

Tunable consistency
if R + W > N, strong(ish) consistency, else eventual consistency

Location-aware

Will take into account rack and datacenter location if given
Used to balance replicas for robustness

Gossip

Uses gossip protocols to bring nodes back up to speed with the current state
Detects failures of its peers constantly

Anti-entropy

Disks fail, uses Merkle trees to determine bad instances
Helps Cassandra choose most correct replica in cases of conflict

Fast Reads

lg(n) access time as dataset grows
Bloom filters help reduce disk access

Super Fast Writes

LSM (append-only) structure of SSTable makes writes really, really cheap

Cons

No JOINs

Need to denormalize data to get use out of Cassandra

No (automatic) Indices

No secondary indices maintained by Cassandra
Need to create and maintain your own

Thrift

Thrift doesn’t currently allow streaming
Somewhat hard to work with

Datamodel

Takes some getting used to
Resembles a 4- or 5-dimensional map

Keyspace ~= Schema

Namespace, doesn't carry any schema metadata
One keyspace per app, usually

ColumnFamily ~= Table

Think of it like: SortedMap<String, Row>

Row ~= Row

Again, best thought of as: SortedMap<Name, Value>
Rows can have many columns (or supercolumns)

Column ~= Column

Contains name, value and last-update timestamp

SuperColumn ~= Column of Columns

SuperColumn is a SortedMap<Name, Value>, where value is a Column
Set ColumnFamily to use Columns or SuperColumns in config

Applications

- Logging
- Output of batch jobs
- Social network data
- Search engine

```
Digg: {                          // Keyspace
   Friend_Diggs: {               // ColumnFamily
      12345: {                   // Row
         user_id: {              // SuperColumn
            friend_id1: true,
            friend_id2: true,
         }
      }
   }
}
```

```
Twitter: {
  Users: {
    1: {
      screenname: "codeslinger",
      website: "http://cbcg.net",
    },
    2: {
      screenname: "evan",
      website: "http://cloudbur.st",
    }
  }
}
```


Twitter: {
Statuses: {
1: {
text: "I like posting",
user_id: 1,
},
2: {
text: "I wish you wouldn't",
user_id: 2,
in_reply: 1,
}
}
}

```
Twitter: {
  UserRelationships: {
    1: {
      user_timeline: {
        2: true,
        1: true,
      },
      home_timeline: {
        3: true,
        2: true,
      }
    },
    ...
  }
}
```

Thanks!

Moar Info!

- http://incubator.apache.org/cassandra/
- http://wiki.apache.org/cassandra
- http://www.mail-archive.com/cassandra-user@incubator.apache.org/
- #cassandra on irc.freenode.net

Photo Credits

http://www.flickr.com/photos/shindotv/3835362925/
http://www.flickr.com/photos/richardpluck/235207109/
http://www.flickr.com/photos/14804582@N08/2111269218/
http://www.flickr.com/photos/gregpc/3350620031/