Making choices:

What kind of relationship are you seeking with your database?

March 27, 2014

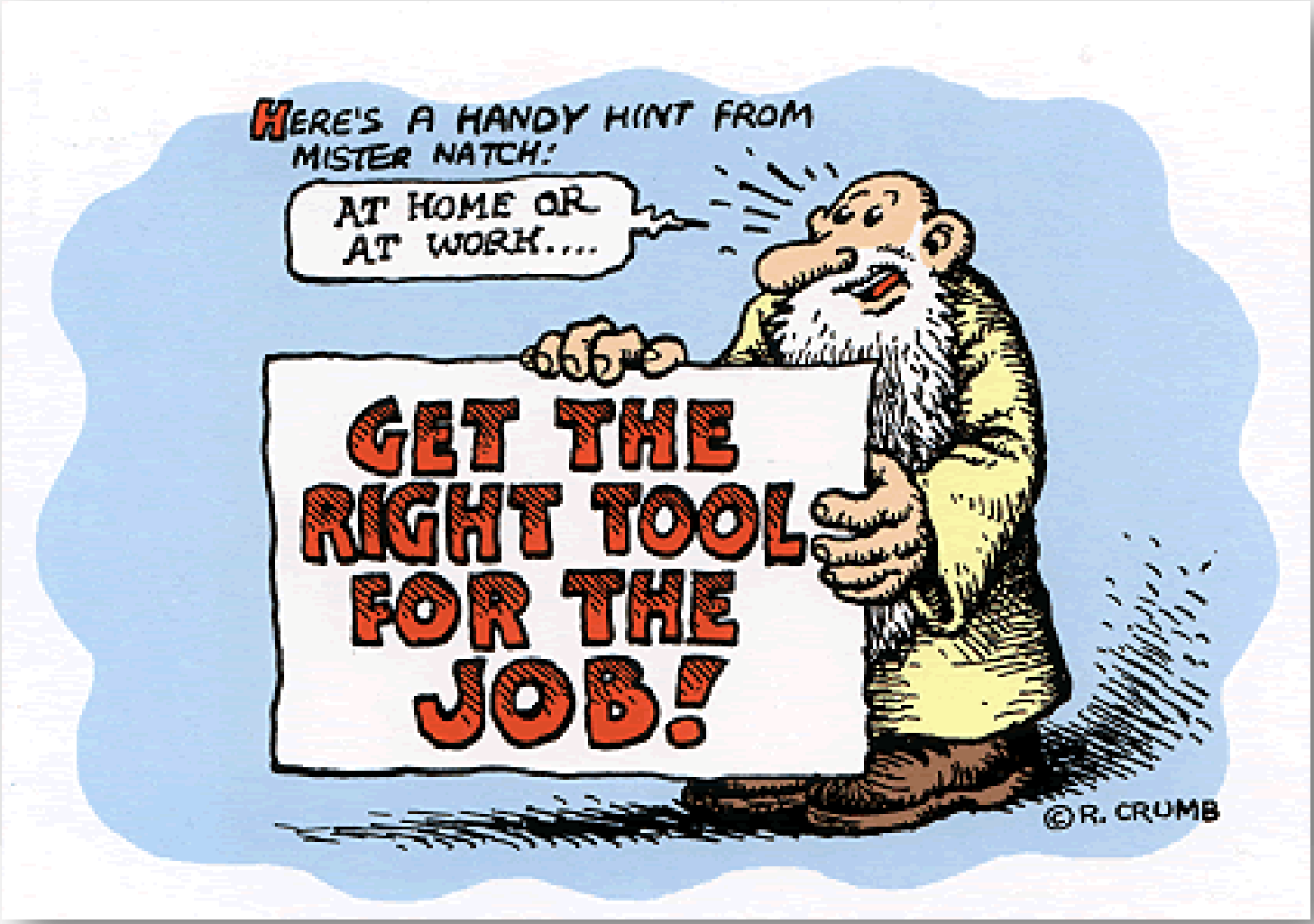
J.R. Arredondo
Director, Data Services Product Marketing
@jrarredondo

rackspace
the open cloud company

What are we going to talk about today?

- Databases are complicated tools
- There are numerous choices
  - How did we get here?
- Understanding some of our choices
  - SQL: Relational
  - MongoDB: Documents
  - Redis: Key-value
  - Hadoop: Large distributed files
- How should I think about managing them?

Common advice these days from smart people



rackspace®
the open cloud company

Let’s take a step back

rackspace®
the open cloud company

Databases are not simple, single purpose tools

The relationship with your database can be complicated

How did we get here?

rackspace®
the open cloud company

App development is changing

	Traditional apps (CRM, HR, Finance apps)	**Modern apps** (mobile, social, media, games)
Infrastructure	**Custom-built for the app**	**Programmable by the app**
Data	Mostly resides on premise	Mostly resides on cloud

Trend

Applications are becoming systems of engagement

	Traditional apps (CRM, HR, Finance apps)	Modern apps (mobile, social, media, games)
Characteristics of the system	**Systems of Record** Highly structured Slow to change Transactional Stable Core to the business Not very social	**Systems of Engagement** Loosely structured Quick to adapt Conversational Dynamic and in flux Edge of the business Fundamentally social
Data	Mostly resides on premise	Mostly resides on cloud

Trend

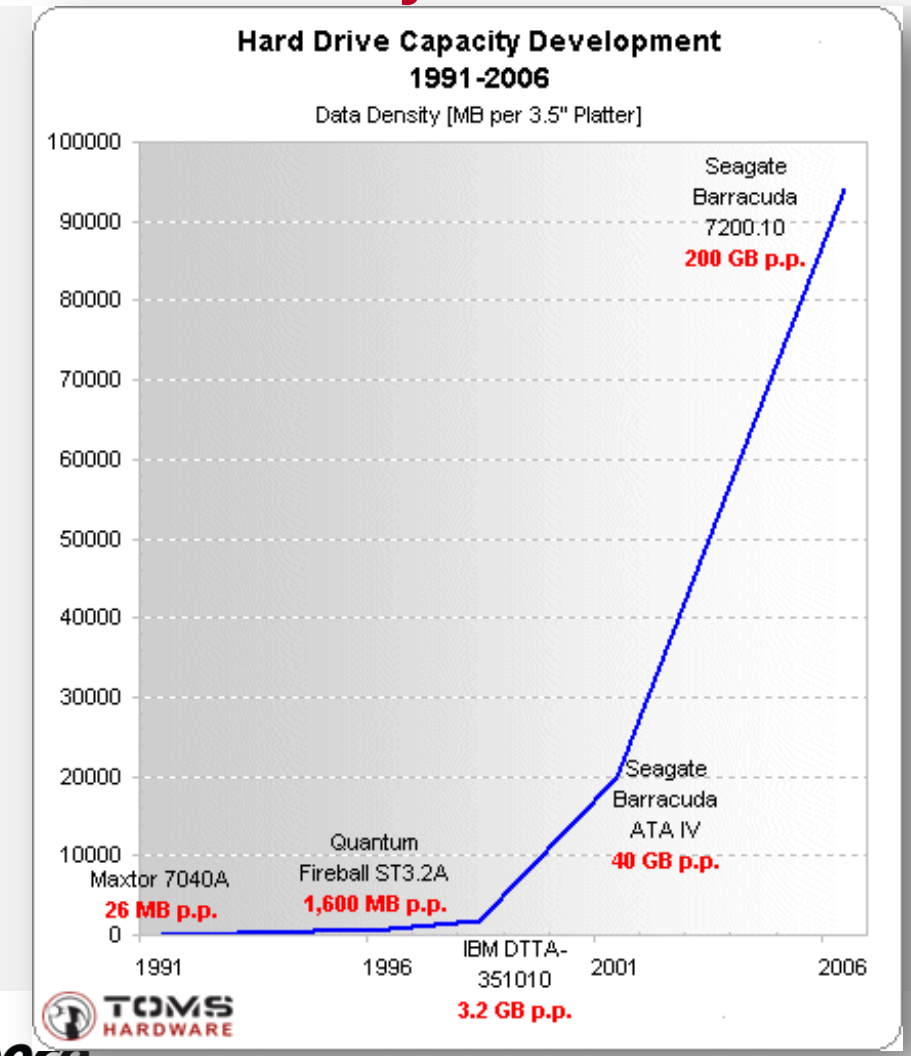
We are building different kinds of applications

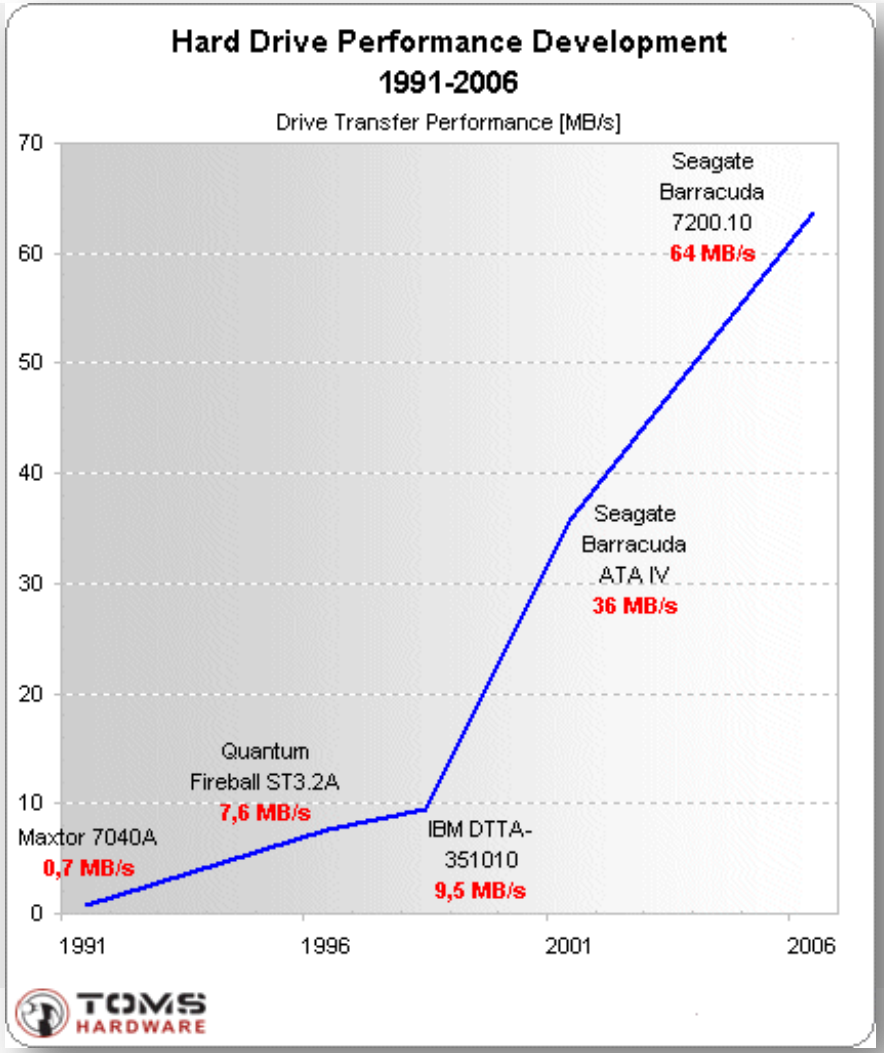
MEDIA | GAMING | M2M | MOBILE | SOCIAL

SOME UNIQUE SCENARIOS

Cloud scale and fast growth
High speed data retrieval needs
Frequently written, rarely read
Binary files
Short term data
Multi-location access
Zero downtime needs
Dynamic or object oriented models
Trying to avoid RAID / storage limits
Large files

In the 15 year period before 2006, storage density increased 10,000x, but performance only increased about 100x





rackspace
the open cloud company

Source: “15 Years of Hard Drive History: Capacities outran performance” (November 27, 2006)
http://www.tomshardware.com/reviews/15-years-of-hard-drive-history,1368-6.html

As a result, a revolution ensued in the world of Data Services

Polyglot persistence is here to stay: there are about 150+ choices **just** in the “NoSQL” subset

Two key issues

How do you ensure best fit for your app?

What is the long term view of your relationship with your database?

Get to know your choices well

Crash course!

rackspace®
the open cloud company

Understand the personality of your database

Let’s use these examples

Relational	Documents	Key-value	Distributed large sets
Data Integrity SQL	Flexible Schema Scale	Fast Retrieval Data structures	Distributed Processing Big Data
(SQL)	(MongoDB)	(Redis)	(Hadoop)

rackspace
the open cloud company

Relational databases (SQL)

They literally saved the world from running on paper

Strengths

- **Data integrity** through data types and semantic rules
  - AGE >= 0
  - Person must have a NAME
- **Querying**
- **Aggregation**
- **SQL**

```
SELECT SUM(VALUE)
FROM CAR
GROUP BY MODEL;
```

"Weaknesses"

- **Complex development** as developer needs to map relational model with object oriented code
- **Complexity grows** exponentially as relational model grows
- **Difficult to scale**
- **Expensive** (hardware, software)

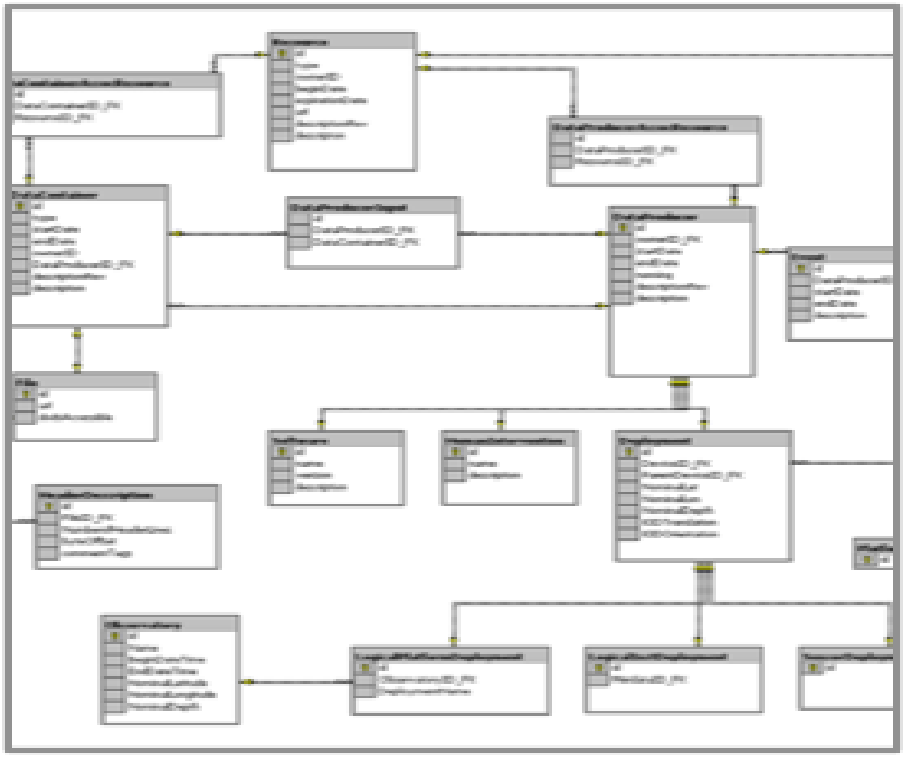
If your operation depends on the integrity of your business rules, the relational model rules.

Scaling is a little difficult and performance is key.

The complexities of relational databases led to NoSQL

- Allow new data without a defined schema
- Designed for scale
- Faster, agile development
- Databases in the cloud!

Documents Databases



VS.

```
{
    _id : ObjectId("4c4ba5e5e8aabf3"),
    car_make: "Volkswagen",
    model : "Rabbit",
    tires : [
{type : “driver front”,
 brand: “Michelin”},
{type : “driver rear”,
 brand: “Michelin”},
{type : “passenger front”,
 brand: “Michelin”},
{type : “passenger rear”,
 brand: “Michelin”}, ]

  }
```

MongoDB has emerged as a leader in Document databases

mongoDB

- Leading NoSQL database
- Open Source
- Agility and flexibility (no set schema)
- Better fit to modern development methodologies
- New types of records (fields) are added easily
- Imagine it like a folder you add pages to

MongoDB

- Document databases and collections
- Indexes
- Rich query language

- Replication (transparent to the app)
  - Writes to primary ensure consistency
  - Configurable reads to secondaries to help performance
  - Eventual consistency on secondary reads
  - Election on failures of primary nodes
  - Configurable write concerns for flexible write guarantees depending on app needs

- Shards for horizontal scaling
  - Shard Key used to partition data based on ranges or hashes
  - Partition strategy depends on how evenly you want data distributed, and the nature of your queries (single vs. ranges)

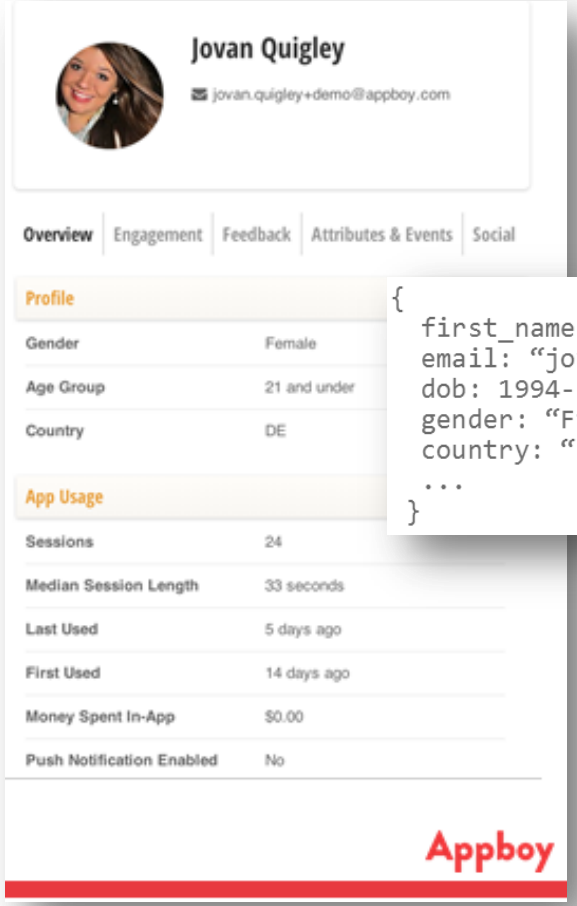
```
db.friends.insert (
  {
  name: “J.R.”,
  email: “email@rackspace.com”,
  twitter_handle: “jrarredondo”,
  teams: [ “Mariners”, “Rangers” ],
  group: 1
  }
)

db.friends.ensureIndex( { group: 1} )

var myCursor = db.friends.find( { group: { $gt: 0 } } )
```

Flexibility of data model (and its problems) with document databases

Appboy: App marketing automation platform for mobile apps

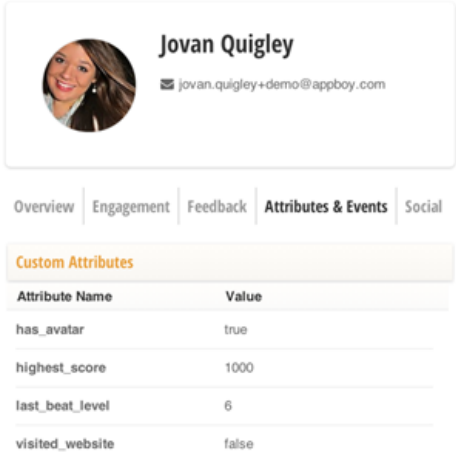



```
{
  first_name: "Jovan",
  email: "jovan+demo@appboy.com",
  dob: 1994-10-24,
  gender: "F",
  country: "DE",
  ...
}
```

Extensible User Profiles

Custom attributes can go alongside other fields!

```
{
  first_name: "Jovan",
  email: "jovan+demo@appboy.com",
  dob: 1994-10-24,
  gender: "F",
  custom: {
    has_avatar: true,
    highest_score: 1000,
    visited_website: false,
    ...
  },
  ...
}
```




```
db.users.find(…).update({$set: {
```

```
{ visited_website: true }
{ visited_website: "yes" }
```

Appboy

rackspace
the open cloud company

Courtesy of Jon Hyman, CIO and Co-Founder of Appboy

Sometimes…
you combine databases

MySQL and MongoDB together

What is Untappd?

A social discovery and sharing network for beer drinkers

- Heavily used during weekends and at night
- Complex SQL queries
- "What are my friends drinking?"
- "Where can I find this beer?"

It's not one or the other

- What works best for the workflow?
  - MySQL worked best for reference data for us
  - Not everything moved to MongoDB

What stayed in MySQL?

Check-ins
Users
Relationships Data
Primary Datastore

What moved to MongoDB?

Activity Feed (Friend's Graph)
Recommendation Data
Location-based Check-ins

Courtesy of Greg Avola, CTO and Co-Founder, Untappd

Key-value stores: Redis

- Think about it as a single huge hash table
- Simple concepts
  - GET / SET / DELETE <data> based on some <key>
- High performance, in memory
- Persistence
  - Point-in-time Snapshots
  - Append only / Journal
- Partitioning
  - Redis Cluster (future)
  - Proxy-based solutions such as Twemproxy

Key	Value
<key>	<value>
<key>	<value>
<key>	<value>
<key>	<value>

Key-value stores: Redis

- Volatile keys: automatic expiration of keys
  - SET *<key>* *<value>* EX *<seconds>*
  - SETEX *<key>* *<seconds>* *<value>*
- Data structures
  - LISTS, SETS / SORTED SETS, HASHES
- Publish / Subscribe
  - SUBSCRIBE *<channel>*
  - PUBLISH *<channel>* *<message>*
- Transactions (*)
  - MULTI
    - Commands to be executed as a single, atomic isolated operation
  - EXEC / DISCARD
  - (*) Warning: VERY different behaviors than in SQL
- Eviction policies
  - Useful to implement Least Recently Used caches

```ruby
# usage:
# ruby pub.rb channel username

require 'rubygems'
require 'redis'
require 'json'

$redis = Redis.new

data = {"user" => ARGV[1]}

loop do
  msg = STDIN.gets
  $redis.publish ARGV[0], data.merge('msg' => msg.strip).to_json
end
```

```ruby
require 'rubygems'
require 'redis'
require 'json'

$redis = Redis.new(:timeout => 0)

$redis.subscribe('rubyonrails', 'ruby-lang') do |on|
  on.message do |channel, msg|
    data = JSON.parse(msg)
    puts "##{channel} - [#{data['user']}]: #{data['msg']}"
  end
end
```

http://robots.thoughtbot.com/redis-pub-sub-how-does-it-work

Redis scenarios

Cache

Making another application better

- MySQL
- MongoDB
- Magento

Data Structures

(Example: Leaderboards!)

LISTS
SETS
SORTED SETS
HASHES

“Big Data”: generating insights with Hadoop

Volume

Velocity

Variety

Complexity

V^3C

Mining social data for sentiment
Analyzing web clickstreams
Analyzing log data for security breaches
Telemetry from sensors and machines
eCommerce predictive analytics

Fundamentals of Hadoop v1

Data Services

Flume
Log data aggregation and movement

Sqoop
Bulk data transfer from and to relational DB

HBase
Distributed, scalable, non relational database

Pig
Data flow scripting language

Hive
DW analysis layer through HiveQL (SQL-like) queries

HCatalog
Metadata and table management system

Core Services

MapReduce
Data processing framework

HDFS
Distributed File System

Operational Services

Zookeeper
Configuration, sync and naming registry

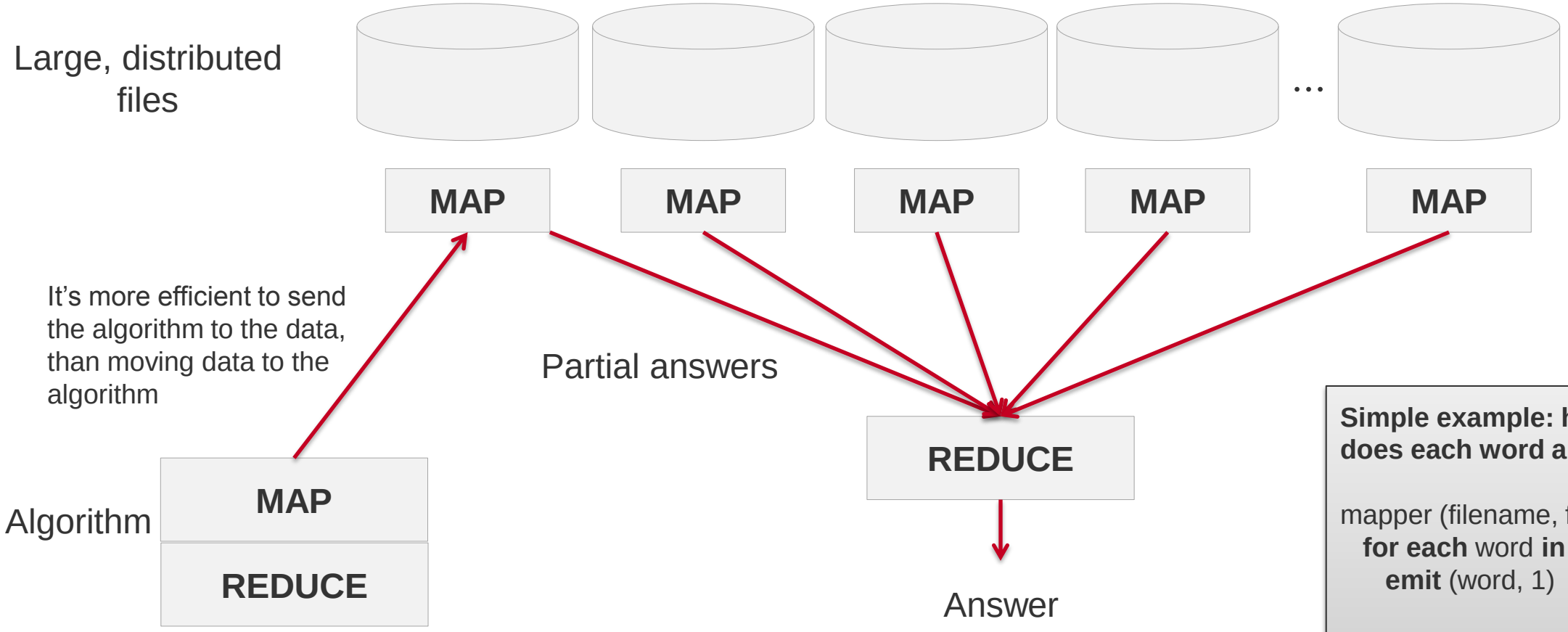
Knox
Auth and access

Oozie
Workflow and job scheduling

Falcon
Data pipeline framework

Ambari
Installation, monitoring, administration

MapReduce



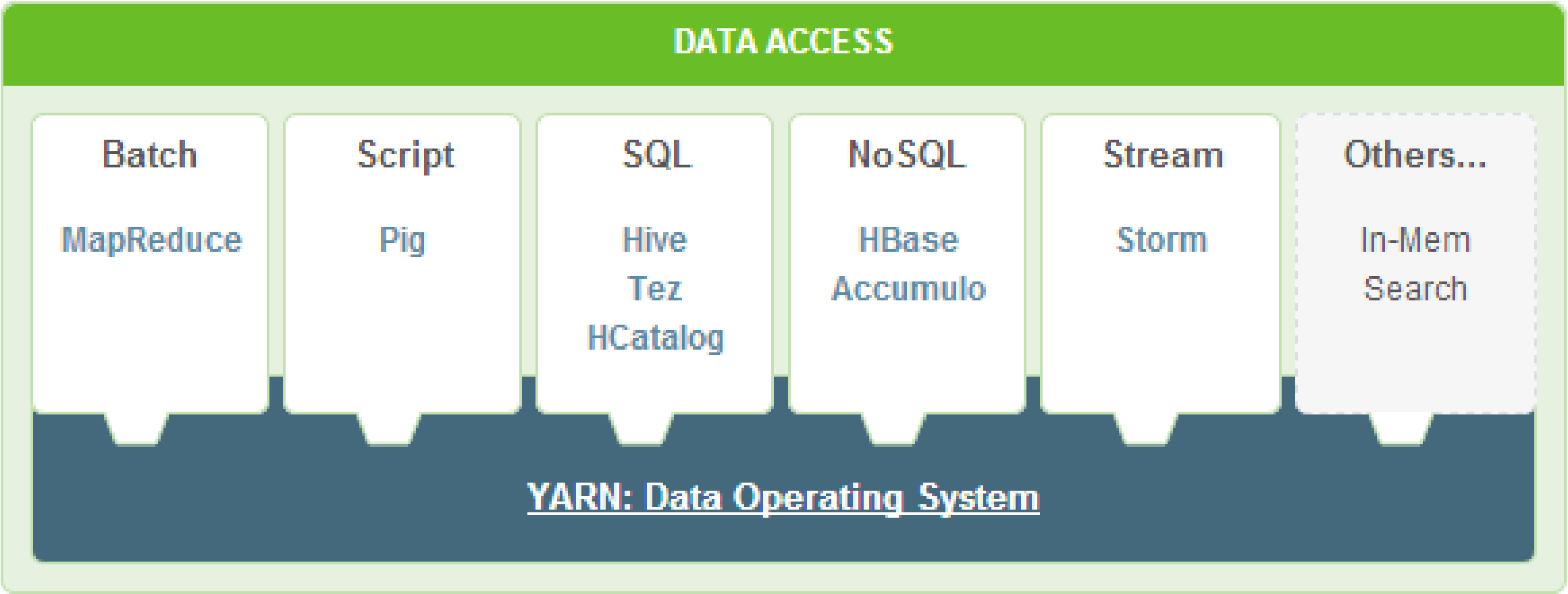
Simple example: how many times does each word appear in all files?

```
mapper (filename, file-contents):
   for each word in file-contents:
      emit (word, 1)

reducer (word, values):
   sum = 0
   for each value in values:
      sum = sum + value
   emit (word, sum)
```

rackspace
the open cloud company

Beyond MapReduce / batch with Hadoop 2.0



Source: Hortonworks

Other ideas

rackspace®
the open cloud company

Really understand the personality of your database

First impressions can be deceiving

“Redis is ‘just a cache’”

- SET
- GET

Redis is a server for <u>data structures</u>

- Strings
- Hashes
- Lists
- Sets / Sorted Sets
- Publish / Subscribe

Huge difference!

rackspace®
the open cloud company

Focus on the tradeoffs

Data integrity
Business rules
Consistency
Transaction isolation
Atomicity

and

Rigidity

Flexibility of schema
Dynamic data models
Horizontal scale
Easier to get started

and

Inconsistency of data

Simple things work some times: just map your data

(remember that **it always "depends"** and use it as the foundation for your data access layer)

Relational	Documents	Key-value	Distributed large sets
Customer contact **Reference data** Order Details (Ship To, Bill To SKU, Quantity, Price) Billing transactions Inventory Prices Member Info (user, pwd)	Customer relationships Notes / Social Partitions (shards) Promotional materials **Dynamic schemas** Statements **Product Catalog**, Images Product Configuration Personalized catalog Member Comments Product Reviews Product Q&As	Session info Cart Recent orders **Home page info** Latest comments Recommendations **Product "stars"** Upsell/Cross sell	**Customer attributes** (non personally identifiable information, geo) **Sales history** Churn info Price history Social info Comments "NPS" **Recommendations** All kinds of analysis
(SQL)	**(MongoDB)**	**(Redis)**	**(Hadoop)**

It's good to understand the fundamental "theory"

What does your problem really need?

ACID

- **Atomicity**: A transactions either happens completely, or not at all
  - No partial transactions
- **Consistency**: Transactions end in a "valid" state
  - No violation of rules
- **Isolation**: Transaction appears as if it is the only thing happening to the database
  - Relaxed most times
  - Deals with phantom, dirty reads or non repeatable reads
- **Durability**: Committed transactions are permanent
  - Even after failure

BASE

- **Basically available**:
  - Supporting partial failures without complete system failure
  - Design as if users would end up in different partitions
- **Soft state**:
  - Things can be in flux for a little bit of time
- **Eventual consistency**:
  - Things right themselves

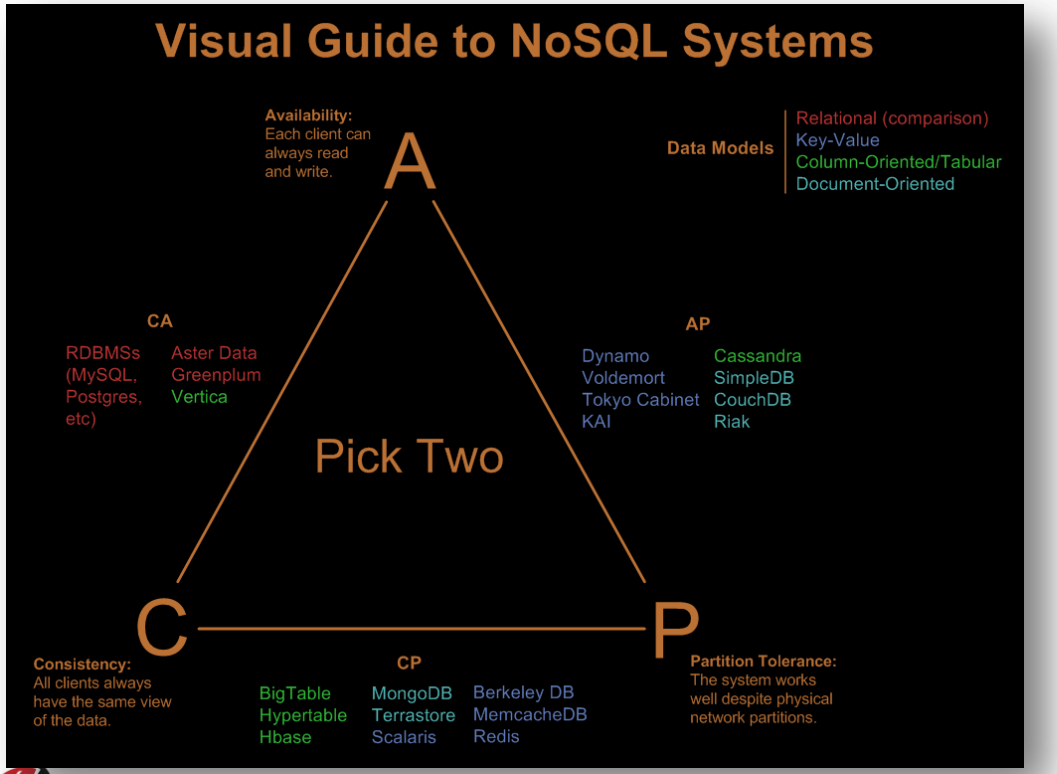
New ways of thinking:
Do customers really need to know the level of inventory of a product to place an order? Maybe all they want is to know that it is not zero

http://queue.acm.org/detail.cfm?id=1394128

Know your CAP, really

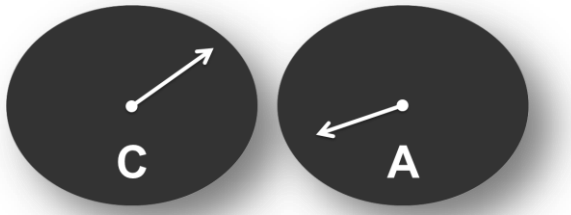
Consistency, Availability and Partition Tolerance

You can only have 2 out of 3 in CAP!



Wait! It's not that simple

- Partitions are not generally common
- Choosing Consistency or Availability is not final
- "It depends"
  - Maybe on user
  - Maybe on system
  - Maybe on type of data



- Just think:
  - How am I going to detect a problem in the network? (P)
  - How am I going to limit operations once I detect that?
  - How am I going to compensate to recover?

Hurst 2010 (http://blog.nahurst.com/visual-guide-to-nosql-systems)

Eric Brewer 2012 (http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed)

The “ilities” and their cousins

These are some of the challenges indirectly related to data that we must deal with

- Stability
- Fit for core scenarios
- Configurability to different scenarios
- Integration with development languages
- Integration with other databases
- SQL compatibility
- End user vs. Developer skillset
- Conceptual changes
- Platform availability
- Data type and semantic needs
- Security

- Performance
- Scalability
- Consistency
- Resiliency
- Data model
- Flexibility
- Cost
- Training
- Tools availability
- Development experience

Rackspace's vision is Data as a Service

From databases to data as a service

rackspace®
the open cloud company

Two key issues

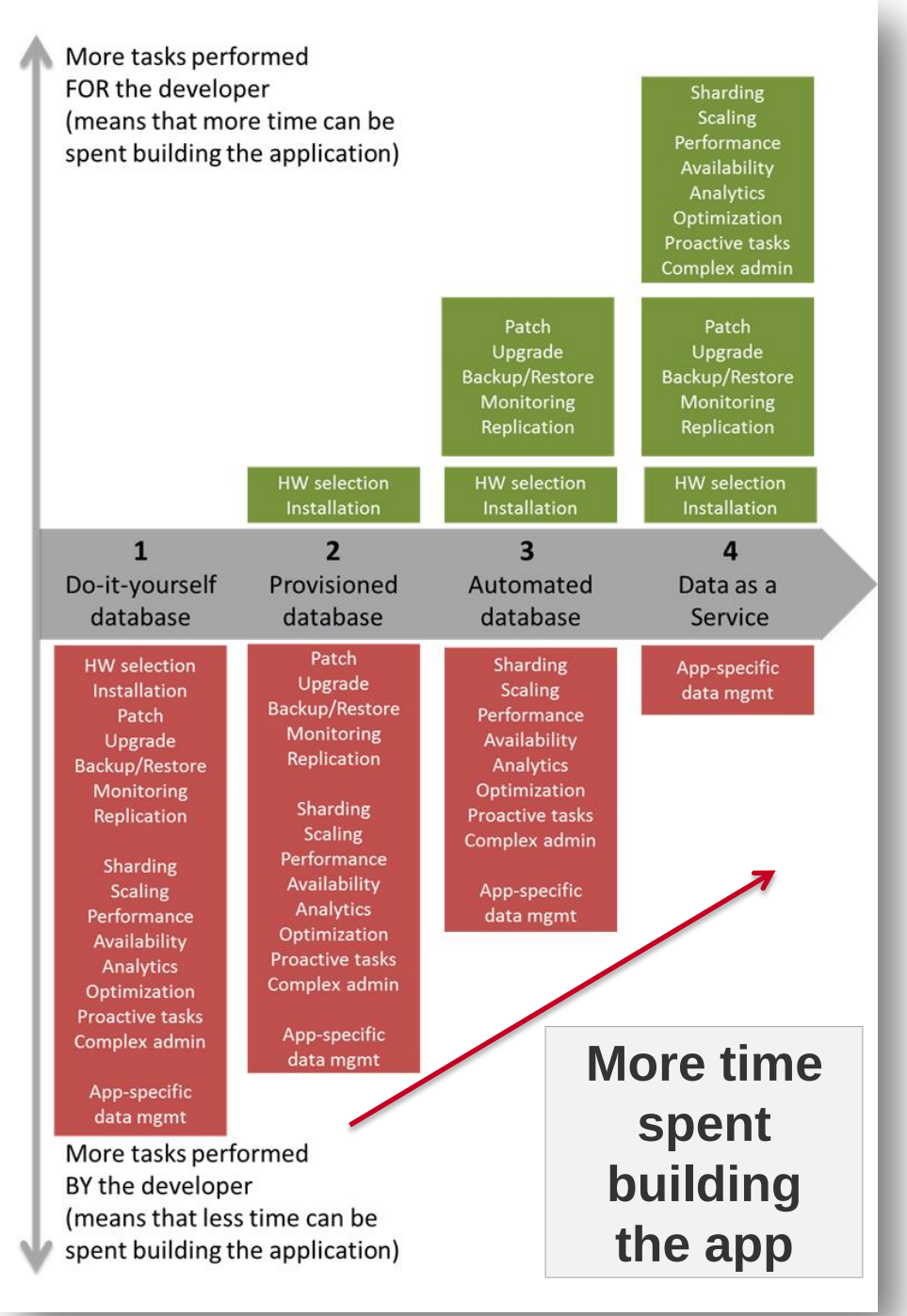
How do you ensure best fit for your app?

What is the long term view of your relationship with your database?

Data-as-a-Service: more time building, less time managing databases

Four levels of DaaS transparency

- For some businesses, database or infrastructure management **IS core of the business**
- For most software-based businesses, database or infrastructure management represents time and resources **not spent building the application**
- **You must answer for yourself**: are you in the business of managing infrastructure, or in the business of [**your market here**]?



rackspace
the open cloud company

Source: "Choosing The Right Cloud Provider" (December 5, 2013)
http://www.rackspace.com/blog/choosing-the-right-cloud-provider-for-your-mongodb-database/

From Database-as-a-Service to Data-as-a-Service

Focus on **building your app**, not managing databases

Highest value activity for your application

Build your application
(i.e. game, startup, mobile app, site)

YOU WANT TO BE FOCUSED HERE

This is the only job that YOU MUST DO without anybody's help because this is your intellectual property

Manage software infrastructure
(i.e. databases)

Manage hardware infrastructure

YOU DON'T WANT TO HAVE TO MANAGE DATABASES OR SERVERS

It only takes away from time building your application

The next vision for databases: Data-as-a-Service

Applications just access the **data as a service**, while the database is transparent

Highest value activity for your application

Build your application
and
manage your data

YOU WANT TO BE FOCUSED HERE

This is the only job that YOU MUST DO without anybody's help because this is your intellectual property

hostname, port number

Data

as a Service

The app just interacts with THE DATA

The application does not see the infrastructure

Towards transparent databases

Data has mass and gravity: you need choices for your hybrid app

(Or: “Divorces are expensive”)

Public Cloud	Managed Cloud
Your Private Cloud on prem	Private Cloud

Data Services at Rackspace are about specialized platforms and services for your application

2 offerings in partnership with Hortonworks for Hadoop-based applications

RACKSPACE® CLOUD BIG DATA PLATFORM POWERED BY OPENSTACK®

RACKSPACE® MANAGED BIG DATA PLATFORM

2 acquisitions for MongoDB and Redis apps

ObjectRocket

REDIS TO GO

Strong portfolio of traditional offerings

ORACLE

MySQL

Microsoft SQL Server

RACKSPACE® CLOUD DATABASES

Maybe two slides would have been sufficient

(but at least you can steal these slides and present them as yours!)

rackspace®
the open cloud company

From “The Lord of the Rings”

“One does not simply walk into Mordor. Its black gates are guarded by more than just Orcs. There is evil there that does not sleep. The great Eye is ever watchful. It is a barren wasteland, riddled with fire, ash, and dust. The very air you breathe is a poisonous fume.”

--Boromir, at the Council of Elrond

If you can only remember ONE THING:
Don’t let a database just happen to you

“One does not simply pick a database. Each was made for a specific set of patterns. Applying one for the wrong pattern will make you lose sleep. Your customers are ever watchful. They want performance, scale and more features. **More importantly,** time spent managing a database is like a poisonous fume, taking time away from what only you can do, which is building an app that delights your customers.”

-- J.R. Arredondo
Rackspace

THANK YOU

Let us know how we can help you
@jrarredondo

RACKSPACE® HOSTING | 5000 WALZEM ROAD | SAN ANTONIO, TX 78218
US SALES: 1-800-961-2888 | **US SUPPORT:** 1-800-961-4454 | WWW.**RACKSPACE**.COM