

# Granular Archival and Nearline Storage Using MySQL, S3, and SQS

---

Walt Jones, SEOmoz  
walt@seomoz.org

# Granular archival

---

- Divide your data into logical chunks.
- Actually remove chunks from the db that aren't being used.
- Restore archived chunks on demand.

# Nearline storage

---

## online

- Your data is available in realtime.

## nearline

- Your data is available within seconds.

## offline

- Your data can take some arbitrary length of time to access.

([http://en.wikipedia.org/wiki/Nearline\\_storage](http://en.wikipedia.org/wiki/Nearline_storage))

# S3 and SQS

---

## S3

- Amazon's key/value storage service.
- Access from anywhere.
- < 1 sec access times.
- Only pay for what you use.

## SQS

- Amazon's distributed queue service.
- Access from anywhere.
- Only pay for what you use.

# S3 and SQS

---

- Available anywhere
- No new hardware or self-supported services
- Easy to use
- Cost scales with you

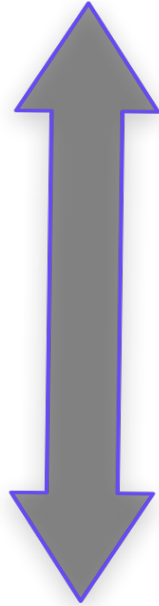
## S3

- S3 rocks. It's hard to recommend anything better.

## SQS

- There are alternatives.
- If you're happy with your queue strategy, keep using it.
- We also use Redis.

**Ephemeral**



**Authoritative**

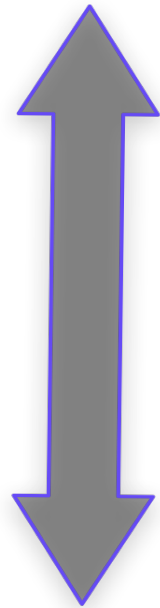
Cache

Query

Archive

Data is always moving.

**Ephemeral**

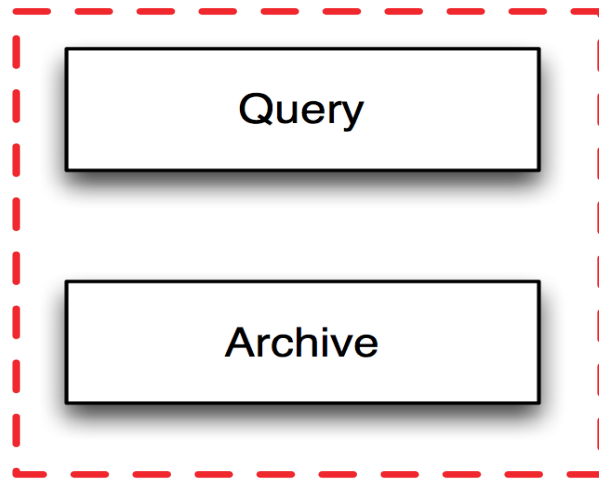


**Authoritative**

Cache

Query

Archive



Data is always moving.

# Goals

---

## Have more control over...

- Where your data is
- The size of your MySQL database

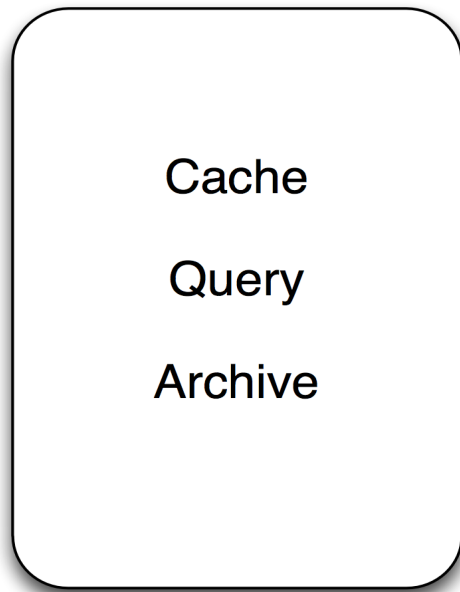
## Use cloud services now

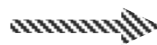
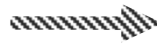
- Regardless of your language environment
- Regardless of your hosting environment

# A Minimal Example

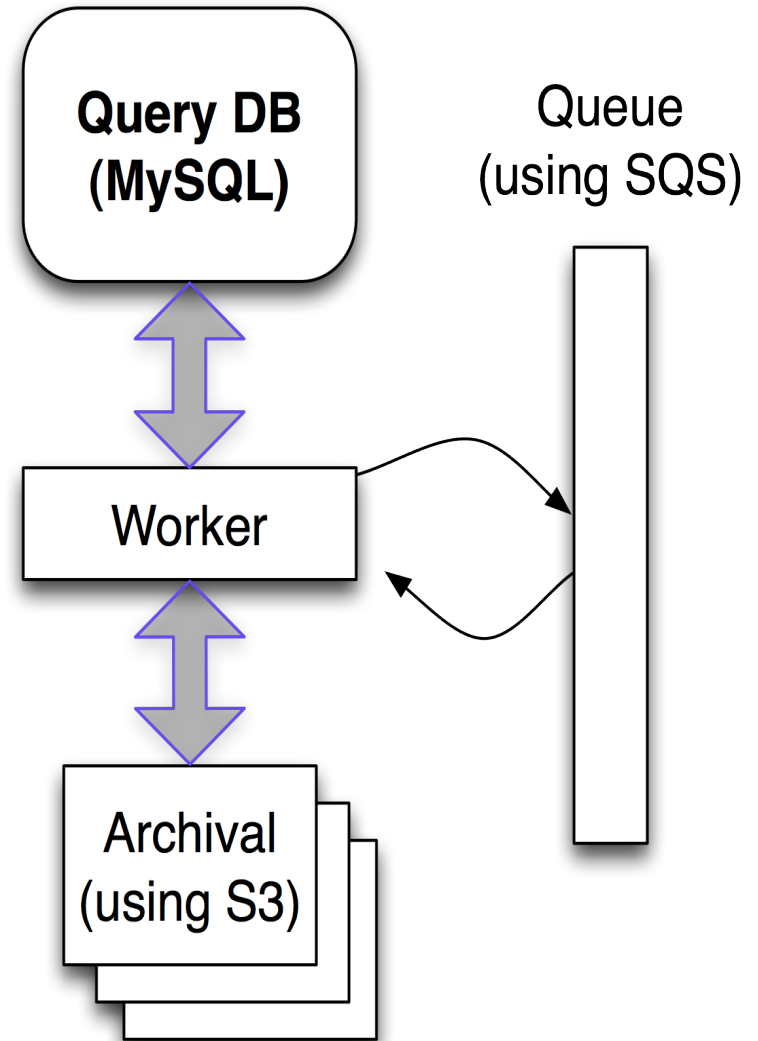
MySQL only

MySQL



-  mysql dump
-  Slave Replication

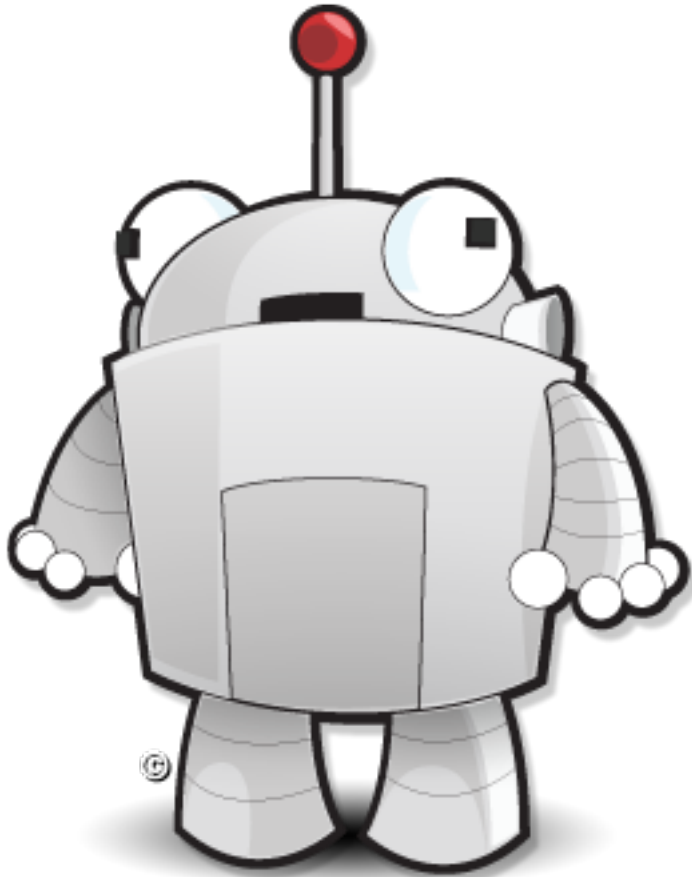
*Separate archival layer*



# Project-oriented data

---

*Typical user*

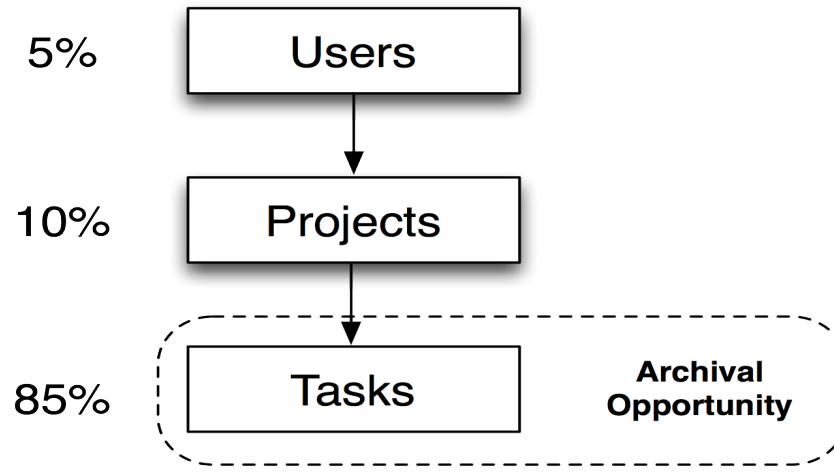


**50 projects**

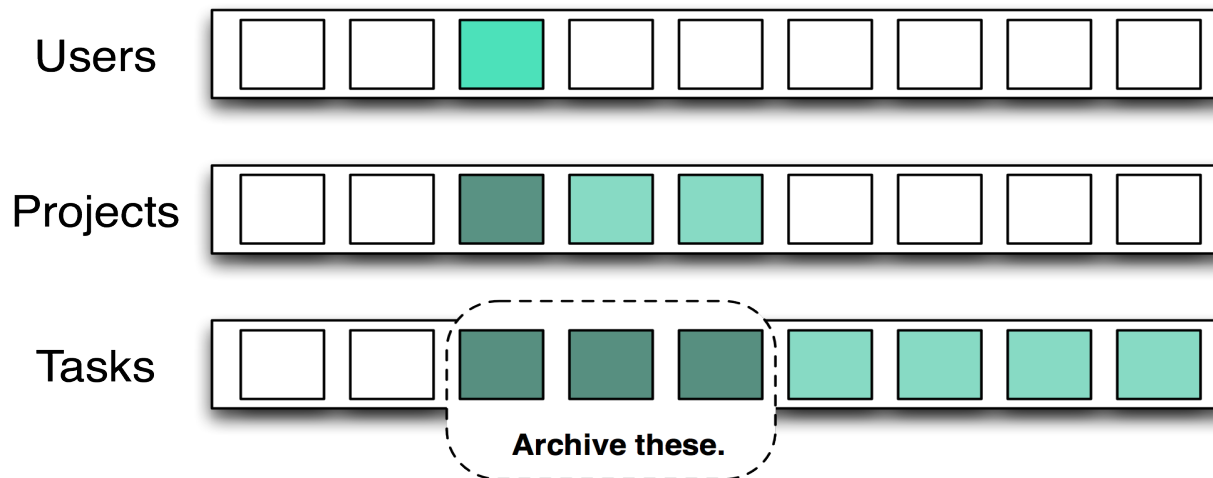
- 5 active
- 45 archived
- Up to 90% reduction in database size.
- Faster, smaller indexes
- Faster, smaller backups
- Cost savings
- (But sometimes you need to get those projects back.)

# Archival opportunity

---



or



# How do we do this?

---

## Schema

- To support safely removable and restorable chunks of data.

## Serialization

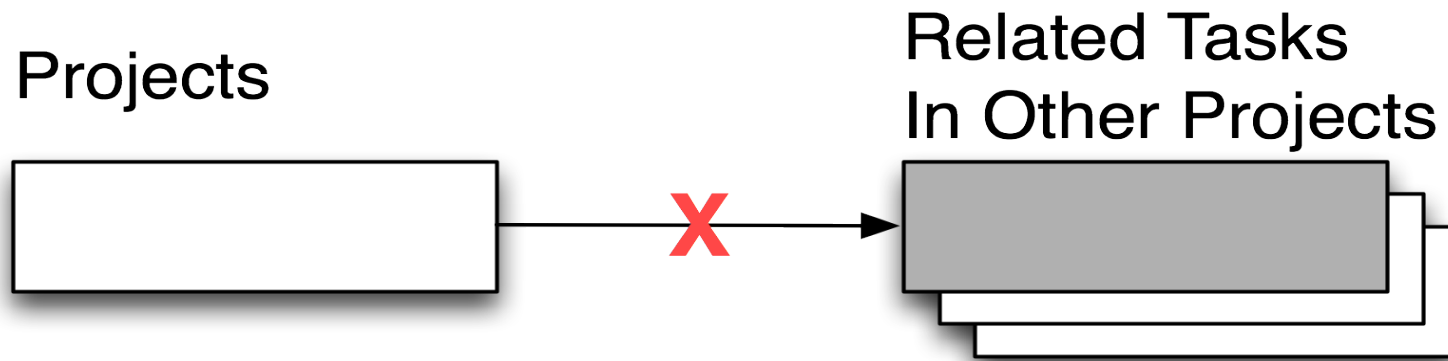
- Requires a data format and storage location.
- Requires serialize/deserialize support.

## State

- Track and update the status of each chunk.

# Removable and restorable

Archived rows can't be joined.

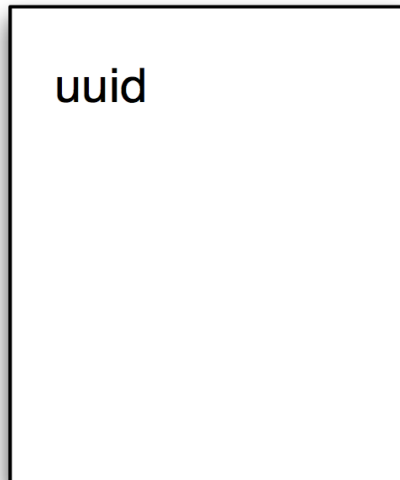


- Use unique IDs and multiple queries.
- Denormalize.

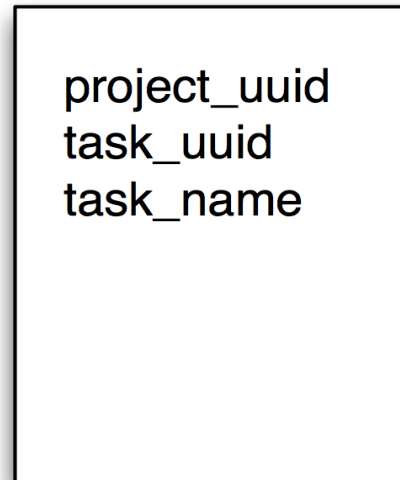
# Denormalize

---

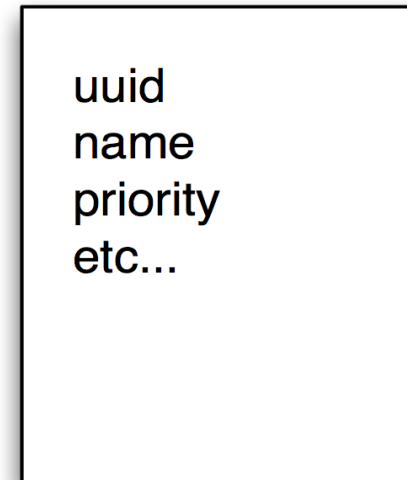
projects



project\_tasks



tasks



# Unique IDs

---

## MD5

- A convenient length
- Collisions are theoretically possible.
- May be acceptable depending on collision resolution.
- Truncated SHA1 may be better.
- Works well when natural keys are desired.

## Timestamp UUIDs

- Generate quickly
- Can be guaranteed unique
- Portable
- Ruby SimpleUUID works well.

# Killed by wolves

---

"A higher probability exists that every member of your programming team will be attacked and killed by wolves in unrelated incidents on the same night."

(referring to SHA1 collisions)

<http://progit.org/book/ch6-1.html>

# Storing IDs in MySQL

---

## 16 byte fixed binary

- Small, fast indexes
- Hard to read
- Lots of translating to/from the hex form.

## char(32)

- Still a decent index
- Easy to work with

## Custom index types in Rails/ActiveRecord/MySQL

- I wrote about this on our dev blog.
- <http://devblog.seomoz.org/2010/10/non-integer-primary-keys-in-rails/>
- Requires patching ActiveRecord
- Requires mysql gem (but can probably be updated for mysql2 gem)

# Serialization

---

## We use JSON

- Readable
- Portable
- Compact

## Compress with zlib

- Portable
- Fast enough
- (Otherwise LZO or other very fast compression.)
- Zlib Gzip Reader/Writer when you need a real IO object

## Store with aws-s3 gem

- <https://github.com/marcel/aws-s3>

# S3 gems

---

## RightAws::S3

- Catches and retries at the HTTP layer
  - Short circuits our error handling
  - S3 timestamps not updated, causes S3 time skewed errors

## aws-s3

- Used by many
- Supports stream mode for input and output
- Has been solid in production
- <https://github.com/marcel/aws-s3>

# Serialize to S3

---

```
def put(bucket_name, key, data, opts={})
  options = {:content_type => 'binary/octet-stream'}.merge(opts)
  data = StringIO.new(Zlib::Deflate::deflate(data)) if data.class == String
  AWS::S3::S3Object.store(key, data, bucket_name, options)
end
```

Code at: <https://gist.github.com/916085>

- *Accepts String or IO objects*
- Use Zlib::GzipReader to handle very large data as IO

# Deserialize from S3

---

```
def get(bucket_name, key, io=nil, &block)
  if io.respond_to?(:write)
    AWS::S3::S3Object.stream(key, bucket_name) do |chunk|
      io.write chunk
    end
  elsif block
    AWS::S3::S3Object.stream(key, bucket_name, {}, &block)
  else
    Zlib::Inflate::inflate(AWS::S3::S3Object.value(key, bucket_name))
  end
end
```

Code at: <https://gist.github.com/916085>

- Accepts String or IO objects
- Use Zlib::GzipWriter to handle very large data as IO.

# Keeping state

---

- Many queues including SQS don't guarantee in-order delivery.
- In the simplest case, use a token to ensure actions are applied correctly.

# Working with SQS

---

- It's pretty nondeterministic.
- Available messages aren't always delivered when requested.
- Keep requesting, you'll eventually see all messages.
- Out-of-order delivery.
- Removal is a two-step process.
- Uses visibility timeout.
- RightAws::SqsGen2 gem works.([https://github.com/rightscale/right\\_aws](https://github.com/rightscale/right_aws))

# Working with SQS

---

```
def send(hash)
  @queue.send_message(JSON[hash])
end
```

```
def pop
  if json = @queue.pop
    JSON[json]
  end
end
```

Code at: <https://gist.github.com/916085>

- RightAws gem handles receive and remove to support pop.

# Managing state

---

*Use unique tokens to ensure only the most recent action is executed.*

*Queue messages may arrive out of order.*

projects



```
{  
  project: <uuid>,  
  action: <archive/restore>,  
  token: <unique_token>  
}
```

# Resources

---

This presentation

- nnn

Ruby examples for S3 and SQS

- <https://gist.github.com/916085>

Non-integer keys in ActiveRecord

- <http://devblog.seomoz.org/2010/10/non-integer-primary-keys-in-rails/>

aws-s3 gem for S3

- <https://github.com/marcel/aws-s3>

RightAws gem for SQS

- [https://github.com/rightscale/right\\_aws](https://github.com/rightscale/right_aws)

SimpleUUID gem

- [https://github.com/ryanking/simple\\_uuid](https://github.com/ryanking/simple_uuid)



**Aaron Wheeler**

Customer Service



**Adam Feldstein**

VP Product



**Anh-Kiet Ngo**

Code Chugger



**Ben Hendrickson**

Senior Software Engineer



**Casey Henry**

Marketing Ninja



**Chas Williams**

Developer



**Christine V.**

Director of Operations



**Crissy Hall**

Customer Service



**Cyrus Shepard**

Customer Service



**David Joslin**

Systems Engineer



**Derric Wise**

Graphic Designer



**Gillian Muessig**

President & Co-Founder



**Jamie Steven**

VP Marketing



**Jeff Pollard**

Lead Front-End Engineer



**Jen Sable Lopez**

Community Director



**Joanna Lord**

Director of Customer Acquisition & Engagement



**Kate Matsudaira**

VP Engineering



**Lauren Hall-Stigerts**

Marketing Communications Manager



**Lindsay P. Wassell**

Q & A Manager



**Matt Heilman**

Art Director



**Megan Singley**

Help Teamster



**Miranda Rensch**

Program Manager



**Phil Smith**

Developer



**Rand Fishkin**

The Wizard of Moz



**Sarah Bird**

Chief Operations Officer



**Shelly Matsudaira**

Release Manager



**Walt Jones**

Software Engineer

We're hiring! (And we pay relocation.)

[kate@seomoz.org](mailto:kate@seomoz.org)

# questions?

---

Walt Jones  
walt@seomoz.org