Tagging and Folksonomy
Schema Design for Scalability and Performance

Jay Pipes
Community Relations Manager, North America
(jay@mysql.com)
MySQL, Inc.

TELECONFERENCE: please dial a number to hear the audio portion of this presentation.

Toll-free US/Canada: 866-469-3239
0800-295-240 Austria
80-884912 Denmark
0800-90-5571 France
00800-12-6759 Greece
800-780-632 Italy
800-2498 Luxembourg
800-15888 Norway
020-79-7251 Sweden
0800-028-8023 UK
800-90-3575 Hong Kong

Direct US/Canada: 650-429-3300 :
0800-71083 Belgium
0-800-1-12585 Finland
0800-101-6943 Germany
1-800-882019 Ireland
0-800-9214652 Israel
0800-022-6826 Netherlands
900-97-1417 Spain
0800-561-201 Switzerland
1800-093-897 Australia
00531-12-1688 Japan

EVENT NUMBER/ACCESS CODE:

Communicate ... Asterisk, Zimbra, ALCATEL, NOKIA

Connect ... facebook, cyworld, evite, classmates.com, mixi

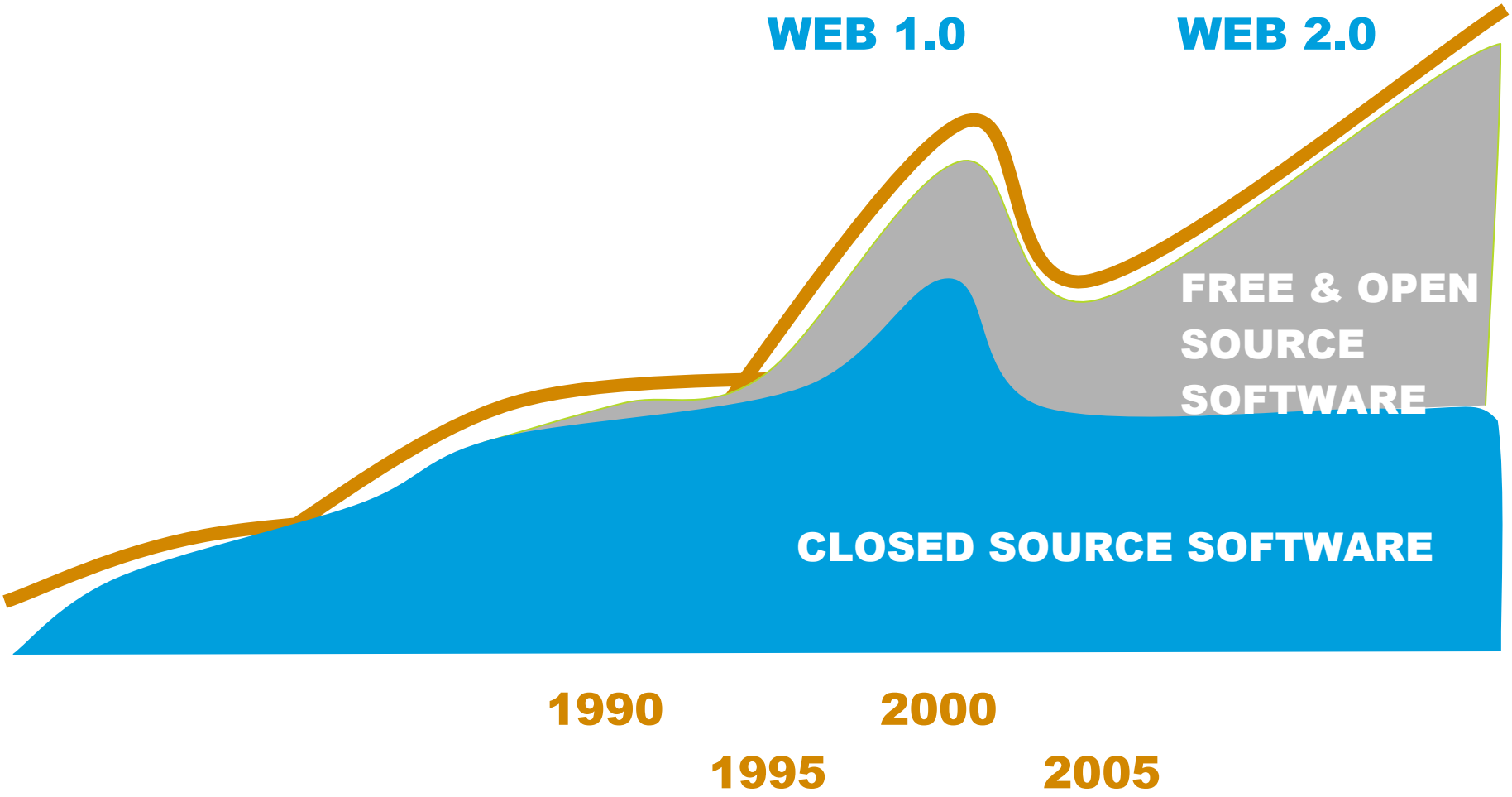
Share ... YouTube, flickr BETA, WordPress, del.icio.us, Wikipedia, FeedBurner, digg

Play ... stardoll (your paperdoll heaven), neopets, PokerRoom.com (Meet them and beat them), HABBO

Trade ... trulia (real estate search), mobile.de, PriceGrabber.com, ticketmaster, craigslist

Search & Look Up Google, YAHOO!, Chicago.com (Your Chicago Search Engine), Travelocity, Citysearch, c|net, The Weather Channel

Software Industry Evolution



How MySQL Enables Web 2.0

lower tco

ubiquitous with developers

large objects

tag databases

lamp

full-text search

XML

language support

ease of use

world-wide community

performance

scale out

interoperable

connectors

bundled

open source

replication

reliability

session management

partitioning (5.1)

pluggable storage engines

query cache

Agenda

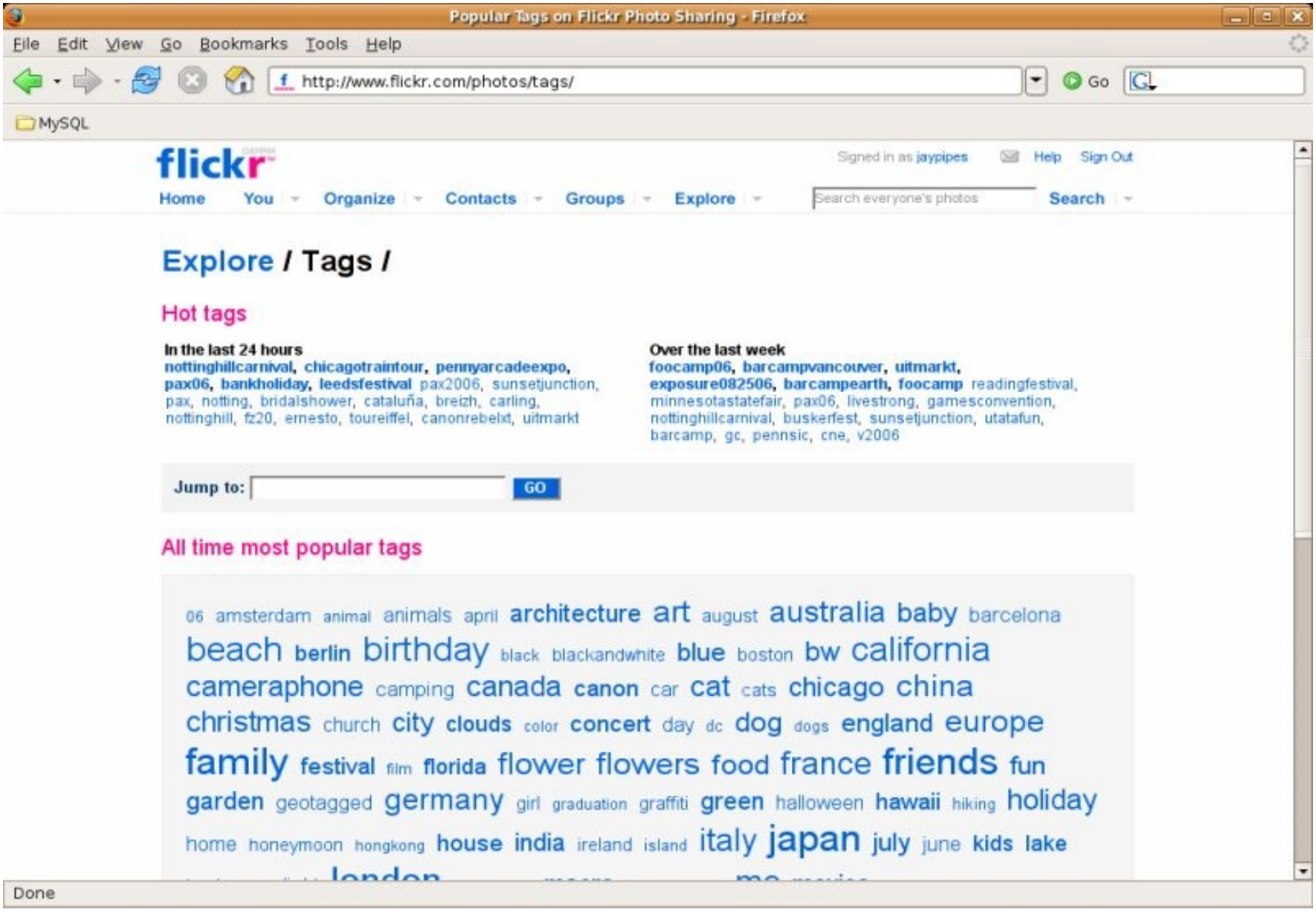
- Web 2.0: more than just rounded corners
- Tagging Concepts in SQL
- Folksonomy Concepts in SQL
- Scaling out sensibly

More than just rounded corners

- What is Web 2.0?
  - Participation
  - Interaction
  - Connection
- A changing of design patterns
  - AJAX and XML-RPC are changing the way data is queried
  - Rich web-based clients
  - Everyone's got an API
  - API leads to increased requests per second. Must deal with growth!

AJAX and XML-RPC change the game

- Traditional (Web 1.0) page request builds entire page
  - Lots of HTML, style **and** data in each page request
  - Lots of data processed or queried in each request
  - Complex page requests and application logic
- AJAX/XML-RPC request returns small data set, or page fragment
  - Smaller amount of data being passed on each request
  - But, often many more requests compared to Web 1.0!
  - Exposed APIs mean data services distribute your data to syndicated sites
  - RSS feeds supply data, not full web page


Popular Tags on Flickr Photo Sharing - Firefox
File Edit View Go Bookmarks Tools Help
http://www.flickr.com/photos/tags/
Go
MySQL
flickr
Signed in as jaypipes
Help
Sign Out
Home
You
Organize
Contacts
Groups
Explore
Search everyone's photos
Search
Explore / Tags /
Hot tags
In the last 24 hours
nottinghillcarnival, chicagotraintour, pennyarcadeexpo, pax06, bankholiday, leedsfestival pax2006, sunsetjunction, pax, notting, bridalshower, cataluña, breizh, carling, nottinghill, fz20, ernesto, toureiffel, canonrebelxt, uitmarkt
Over the last week
foocamp06, barcampvancouver, uitmarkt, exposure082506, barcampearth, foocamp readingfestival, minnesotastatefair, pax06, livestrong, gamesconvention, nottinghillcarnival, buskerfest, sunsetjunction, utatafun, barcamp, gc, pennsic, cne, v2006
Jump to:
GO
All time most popular tags
06 amsterdam animal animals april architecture art august australia baby barcelona beach berlin birthday black blackandwhite blue boston bw california cameraphone camping canada canon car cat cats chicago china christmas church city clouds color concert day dc dog dogs england europe family festival film florida flower flowers food france friends fun garden geotagged germany girl graduation graffiti green halloween hawaii hiking holiday home honeymoon hongkong house india ireland island italy japan july june kids lake
Done

Participation, Interaction and Connection

- Multiple users editing the same content
  - Content explosion
  - Lots of textual data to manage. How do we organize it?
- User-driven data
  - “Tracked” pages/items
  - Allows interlinking of content via the user to user relationship (folksonomy)
- Tags becoming the new categorization/filing of this content
  - Anyone can tag the data
  - Tags connect one thing to another, similar to the way that the folksonomy relationships link users together
- So, the common concept here is “linking”...

What The User Dimension Gives You

- Folksonomy adds the “user dimension”
- Tags often provide the “glue” between user and item dimensions
- Answers questions such as:
  - Who shares my interests
  - What are the other interests of users who share my interests
  - Someone tagged my bookmark (or any other item) with something. What other items did that person tag with the same thing?
  - What is the user's immediate “ecosystem”? What are the fringes of that ecosystem?
    - Ecosystem radius expansion (like zipcode radius expansion...)
- Marks a new aspect of web applications into the world of data warehousing and analysis

Linking Is the “R” in RDBMS

- The schema becomes the absolute driving force behind Web 2.0 applications
- Understanding of many-to-many relationships is critical
- Take advantage of MySQL's architecture so that our schema is as efficient as possible
- Ensure your data store is normalized, standardized, and consistent
- So, let's digg in! (pun intended)

Comparison of Tag Schema Designs

- "MySQLicious"
  - Entirely denormalized
  - One main fact table with one field storing delimited list of tags
  - Unless using FULLTEXT indexing, does not scale well at all
  - Very inflexible
- "Scuttle" solution
  - Two tables. Lookup one-way from item table to tag table
  - Somewhat inflexible
  - Doesn't represent a many-to-many relationship correctly
- "Toxi" solution
  - Almost gets it right, except uses surrogate keys in mapping tables ...
  - Flexible, normalized approach
  - Closest to our recommended architecture ...

Tagging Concepts in SQL

The Tags Table

- The “Tags” table is the foundational table upon which all tag links are built
- Lean and mean
- Make primary key an INT!

```
CREATE TABLE Tags (
tag_id INT UNSIGNED NOT NULL AUTO_INCREMENT
, tag_text VARCHAR(50) NOT NULL
, PRIMARY KEY pk_Tags (tag_id)
, UNIQUE INDEX uix_TagText (tag_text)
) ENGINE=InnoDB;
```

Example Tag2Post Mapping Table

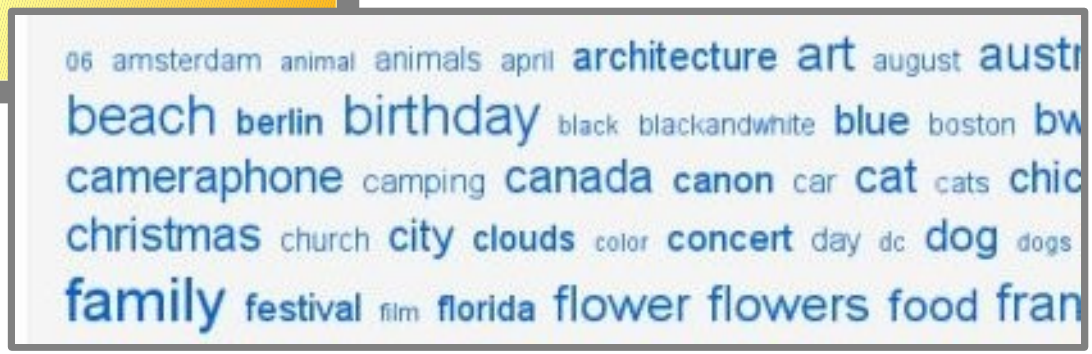
- The mapping table creates the link between a tag and anything else
- In other terms, it maps a many-to-many relationship
- Important to index from both “sides”

```
CREATE TABLE Tag2Post (
tag_id INT UNSIGNED NOT NULL
, post_id INT UNSIGNED NOT NULL
, PRIMARY KEY pk_Tag2Post (tag_id, post_id)
, INDEX (post_id)
) ENGINE=InnoDB;
```

The Tag Cloud

- Tag density typically represented by larger fonts or different colors

```
SELECT tag_text
, COUNT(*) as num_tags
FROM Tag2Post t2p
INNER JOIN Tags t
ON t2p.tag_id = t.tag_id
GROUP BY tag_text;
```



Efficiency Issues with the Tag Cloud

- With InnoDB tables, you don't want to be issuing COUNT() queries, even on an indexed field

```
CREATE TABLE TagStat
tag_id INT UNSIGNED NOT NULL
, num_posts INT UNSIGNED NOT NULL
// ... num_xxx INT UNSIGNED NOT NULL
, PRIMARY KEY (tag_id)
) ENGINE=InnoDB;
```

```
SELECT tag_text, ts.num_posts
FROM Tag2Post t2p
INNER JOIN Tags t
ON t2p.tag_id = t.tag_id
INNER JOIN TagStat ts
ON t.tag_id = ts.tag_id
GROUP BY tag_text;
```

The Typical Related Items Query

- Get all posts tagged with any tag attached to Post #6
- In other words “Get me all posts related to post #6”

```
SELECT p2.post_id
FROM Tag2Post p1
INNER JOIN Tag2Post p2
ON p1.tag_id = p2.tag_id
WHERE p1.post_id = 6
GROUP BY p2.post_id;
```

Problems With Related Items Query

- Joining small to medium sized “tag sets” works great
- But... when you've got a large tag set on either “side” of the join, problems can occur with scalablity
- One way to solve is via derived tables

```
SELECT p2.post_id
FROM (
SELECT tag_id FROM Tag2Post
WHERE post_id = 6 LIMIT 10
) AS p1
INNER JOIN Tag2Post p2
ON p1.tag_id = p2.tag_id
GROUP BY p2.post_id LIMIT 10;
```

The Typical Related Tags Query

- Get all tags related to a particular tag via an item
- The “reverse” of the related items query; we want a set of related tags, not related posts

```
SELECT t2p2.tag_id, t2.tag_text
FROM (SELECT post_id FROM Tags t1
INNER JOIN Tag2Post
ON t1.tag_id = Tag2Post.tag_id
WHERE t1.tag_text = 'beach' LIMIT 10
) AS t2p1
INNER JOIN Tag2Post t2p2
ON t2p1.post_id = t2p2.post_id
INNER JOIN Tags t2
ON t2p2.tag_id = t2.tag_id
GROUP BY t2p2.tag_id LIMIT 10;
```

Dealing With More Than One Tag

- What if we want only items related to each of a set of tags?
- Here is the typical way of dealing with this problem

```
SELECT t2p.post_id
FROM Tags t1
INNER JOIN Tag2Post t2p
ON t1.tag_id = t2p.tag_id
WHERE t1.tag_text IN ('beach','cloud')
GROUP BY t2p.post_id
HAVING COUNT(DISTINCT t2p.tag_id) = 2;
```

Dealing With More Than One Tag (cont'd)

- The GROUP BY and the HAVING COUNT(DISTINCT ...) can be eliminated through joins
- Thus, you eliminate the Using temporary, using filesort in the query execution

```
SELECT t2p2.post_id
FROM Tags t1 CROSS JOIN Tags t2
INNER JOIN Tag2Post t2p1
ON t1.tag_id = t2p1.tag_id
INNER JOIN Tag2Post t2p2
ON t2p1.post_id = t2p2.post_id
AND t2p2.tag_id = t2.tag_id
WHERE t1.tag_text = 'beach'
AND t2.tag_text = 'cloud';
```

Excluding Tags From A Resultset

- Here, we want have a search query like “Give me all posts tagged with “beach” and “cloud” but not tagged with “flower”. Typical solution you will see:

```
SELECT t2p1.post_id
FROM Tag2Post t2p1
INNER JOIN Tags t1 ON t2p1.tag_id = t1.tag_id
WHERE t1.tag_text IN ('beach','cloud')
AND t2p1.post_id NOT IN
(SELECT post_id FROM Tag2Post t2p2
INNER JOIN Tags t2
ON t2p2.tag_id = t2.tag_id
WHERE t2.tag_text = 'flower')
GROUP BY t2p1.post_id
HAVING COUNT(DISTINCT t2p1.tag_id) = 2;
```

Excluding Tags From A Resultset (cont'd)

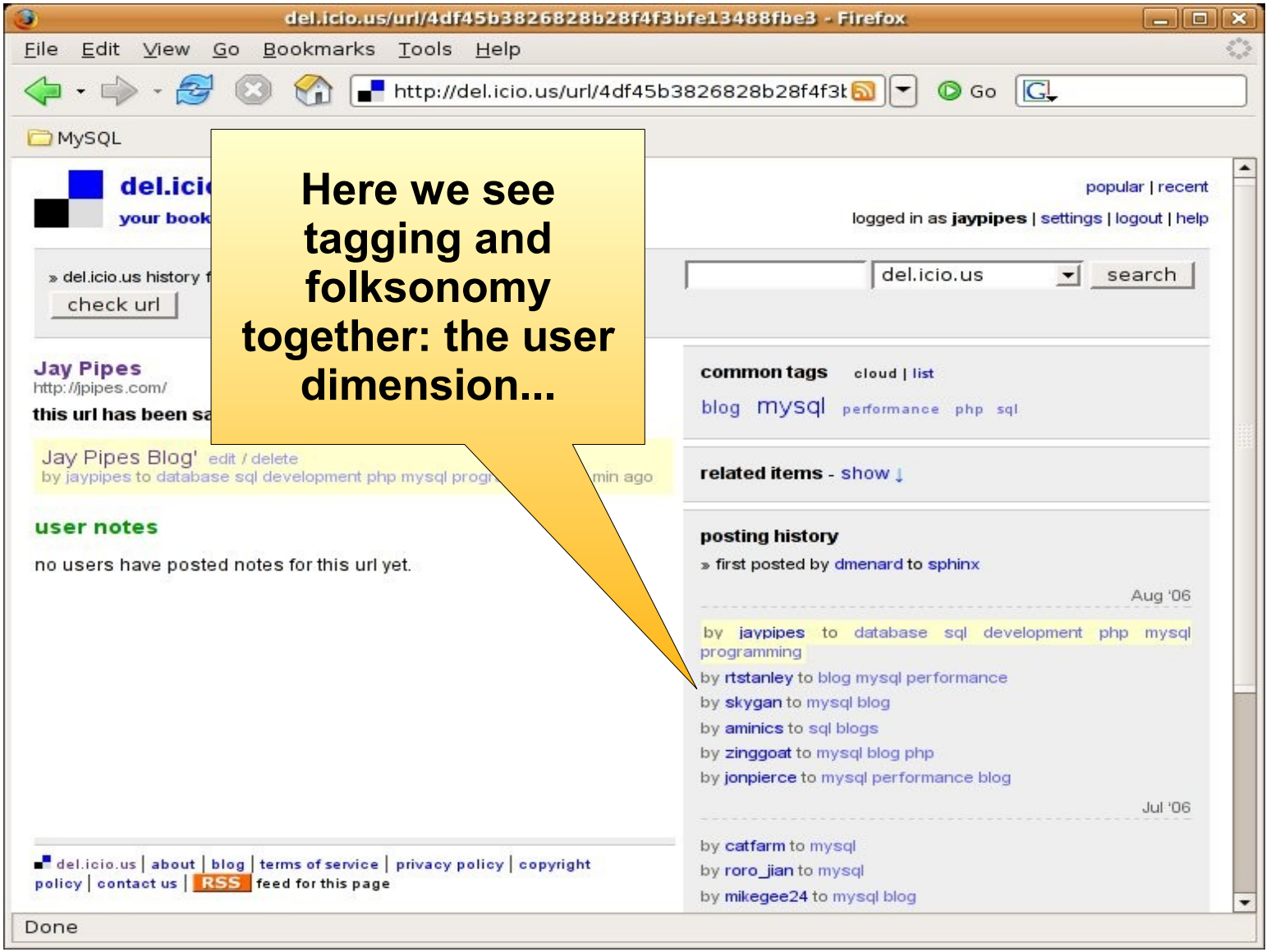
- More efficient to use an outer join to filter out the "minus" operator, plus get rid of the GROUP BY...

```
SELECT t2p2.post_id
FROM Tags t1 CROSS JOIN Tags t2 CROSS JOIN Tags t3
INNER JOIN Tag2Post t2p1
ON t1.tag_id = t2p1.tag_id
INNER JOIN Tag2Post t2p2
ON t2p1.post_id = t2p2.post_id
AND t2p2.tag_id = t2.tag_id
LEFT JOIN Tag2Post t2p3
ON t2p2.post_id = t2p3.post_id
AND t2p3.tag_id = t3.tag_id
WHERE t1.tag_text = 'beach'
AND t2.tag_text = 'cloud'
AND t3.tag_text = 'flower'
AND t2p3.post_id IS NULL;
```

Summary of SQL Tips for Tagging

- Index fields in mapping tables properly. Ensure that each GROUP BY can access an index from the left side of the index
- Use summary or statistic tables to eliminate the use of COUNT(*) expressions in tag clouding
- Get rid of GROUP BY and HAVING COUNT(...) by using standard join techniques
- Get rid of NOT IN expressions via a standard outer join
- Use derived tables with an internal LIMIT expression to prevent wild relation queries from breaking scalability

Folksonomy Concepts in SQL


del.icio.us/url/4df45b3826828b28f4f3bfe13488fbe3 - Firefox
File Edit View Go Bookmarks Tools Help
http://del.icio.us/url/4df45b3826828b28f4f3b
Go
MySQL
popular | recent
logged in as jaypipes | settings | logout | help
del.icio.us
search
check url
Jay Pipes
http://jpipes.com/
Jay Pipes Blog' edit / delete
by jaypipes to database sql development php mysql progr
min ago
user notes
no users have posted notes for this url yet.
common tags cloud | list
blog mysql performance php sql
related items - show
posting history
» first posted by dmenard to sphinx
Aug '06
by jaypipes to database sql development php mysql programming
by rtstanley to blog mysql performance
by skygan to mysql blog
by aminics to sql blogs
by zinggoat to mysql blog php
by jonpierce to mysql performance blog
Jul '06
by catfarm to mysql
by roro_jian to mysql
by mikegee24 to mysql blog
del.icio.us | about | blog | terms of service | privacy policy | copyright policy | contact us | RSS feed for this page
Done
Here we see tagging and folksonomy together: the user dimension...

Folksonomy Adds The User Dimension

- Adding the user dimension to our schema
- The tag is the relationship glue between the user and item dimensions

```
CREATE TABLE UserTagPost (
user_id INT UNSIGNED NOT NULL
, tag_id INT UNSIGNED NOT NULL
, post_id INT UNSIGNED NOT NULL
, PRIMARY KEY (user_id, tag_id, post_id)
, INDEX (tag_id)
, INDEX (post_id)
) ENGINE=InnoDB;
```

Who Shares My Interest Directly?

- Find out the users who have linked to the same item I have
- Direct link; we don't go through the tag glue

```
SELECT user_id
FROM UserTagPost
WHERE post_id = @my_post_id;
```

Who Shares My Interests Indirectly?

- Find out the users who have similar tag sets...
- But, how much matching do we want to do? In other words, what *radius* do we want to match on...
- The first step is to find *my* tags that are within the search radius... this yields my “top” or most popular tags

```
SELECT tag_id
FROM UserTagPost
WHERE user_id = @my_user_id
GROUP BY tag_id
HAVING COUNT(tag_id) >= @radius;
```

Who Shares My Interests (cont'd)

- Now that we have our “top” tag set, we want to find users who match all of our top tags:

```
SELECT others.user_id
FROM UserTagPost others
INNER JOIN (SELECT tag_id
FROM UserTagPost
WHERE user_id = @my_user_id
GROUP BY tag_id
HAVING COUNT(tag_id) >= @radius) AS my_tags
ON others.tag_id = my_tags.tag_id
GROUP BY others.user_id;
```

Ranking Ecosystem Matches

- What about finding our “closest” ecosystem matches
- We can “rank” other users based on whether they have tagged items a number of times similar to ourselves

```
SELECT others.user_id
, (COUNT(*) - my_tags.num_tags) AS rank
FROM UserTagPost others
INNER JOIN (
SELECT tag_id, COUNT(*) AS num_tags
FROM UserTagPost
WHERE user_id = @my_user_id
GROUP BY tag_id
HAVING COUNT(tag_id) >= @radius
) AS my_tags
ON others.tag_id = my_tags.tag_id
GROUP BY others.user_id
ORDER BY rank DESC;
```

Ranking Ecosystem Matches Efficiently

- But... we've still got our COUNT(*) problem
- How about another summary table ...

```
CREATE TABLE UserTagStat (
user_id INT UNSIGNED NOT NULL
, tag_id INT UNSIGNED NOT NUL
, num_posts INT UNSIGNED NOT NULL
, PRIMARY KEY (user_id, tag_id)
, INDEX (tag_id)
) ENGINE=InnoDB;
```

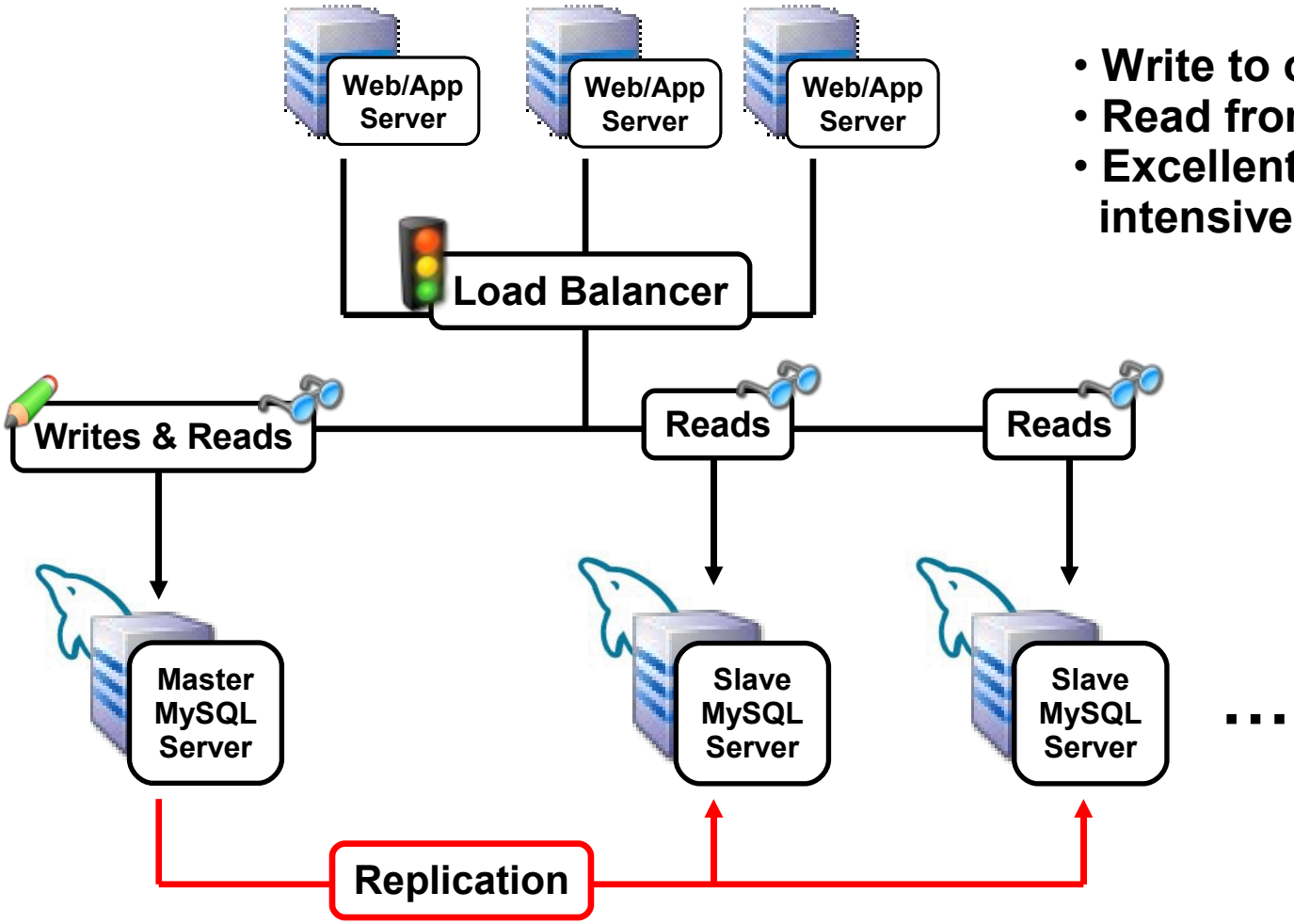
Ranking Ecosystem Matches Efficiently 2

- Hey, we've eliminated the aggregation!

```
SELECT others.user_id
, (others.num_posts - my_tags.num_posts) AS rank
FROM UserTagStat others
INNER JOIN (
SELECT tag_id, num_posts
FROM UserTagStat
WHERE user_id = @my_user_id
AND  num_posts >= @radius
) AS my_tags
ON others.tag_id = my_tags.tag_id
ORDER BY rank DESC;
```

Scaling Out Sensibly

MySQL Replication (Scale Out)

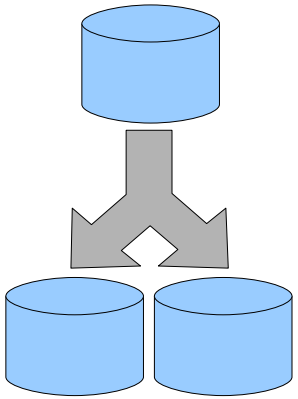


- Write to one master
- Read from many slaves
- Excellent for read intensive apps

Scale Out Using Replication

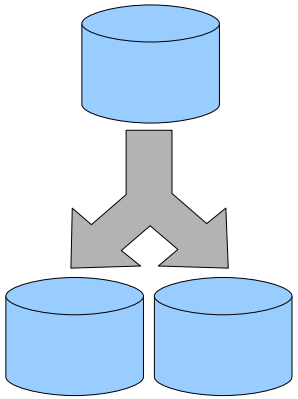
- Master DB stores all writes
- Master has InnoDB tables
- Slaves handle aggregate reads, non-realtime reads
- Web servers can be load balanced (directed) to one or more slaves
- Just plug in another slave to increase read performance (that's scaling out...)
- Slave can provide hot standby as well as backup server

Scale Out Strategies



- Slave storage engine can differ from Master!
  - InnoDB on Master (great update/insert/delete performance)
  - MyISAM on Slave (fantastic read performance and well as excellent concurrent insert performance, plus can use FULLTEXT indexing)
- Push aggregated summary data in batches onto slaves for excellent read performance of semi-static data
  - Example: “this week's popular tags”
    - Generate the data via cron job on each slave. No need to burden the master server
    - Truncate every week

Scale Out Strategies (cont'd)



- Offload FULLTEXT indexing onto a FT indexer such as Apache Lucene, Mnogosearch, Sphinx FT Engine, etc.
- Use Partitioning feature of 5.1 to segment tag data across multiple partitions, allowing you to spread disk load sensibly based on your tag text density
- Use the MySQL Query Cache effectively
  - Use SQL_NO_CACHE when selecting from frequently updated tables (ex: TagStat)
  - Very effective for high-read environments: can yield 200-250% performance improvement

Questions?

MySQL Forge
http://forge.mysql.com/

MySQL Forge Tag Schema Wiki pages
http://forge.mysql.com/wiki/TagSchema

PlanetMySQL
http://www.planetmysql.org

Jay Pipes
(jay@mysql.com)
MySQL AB