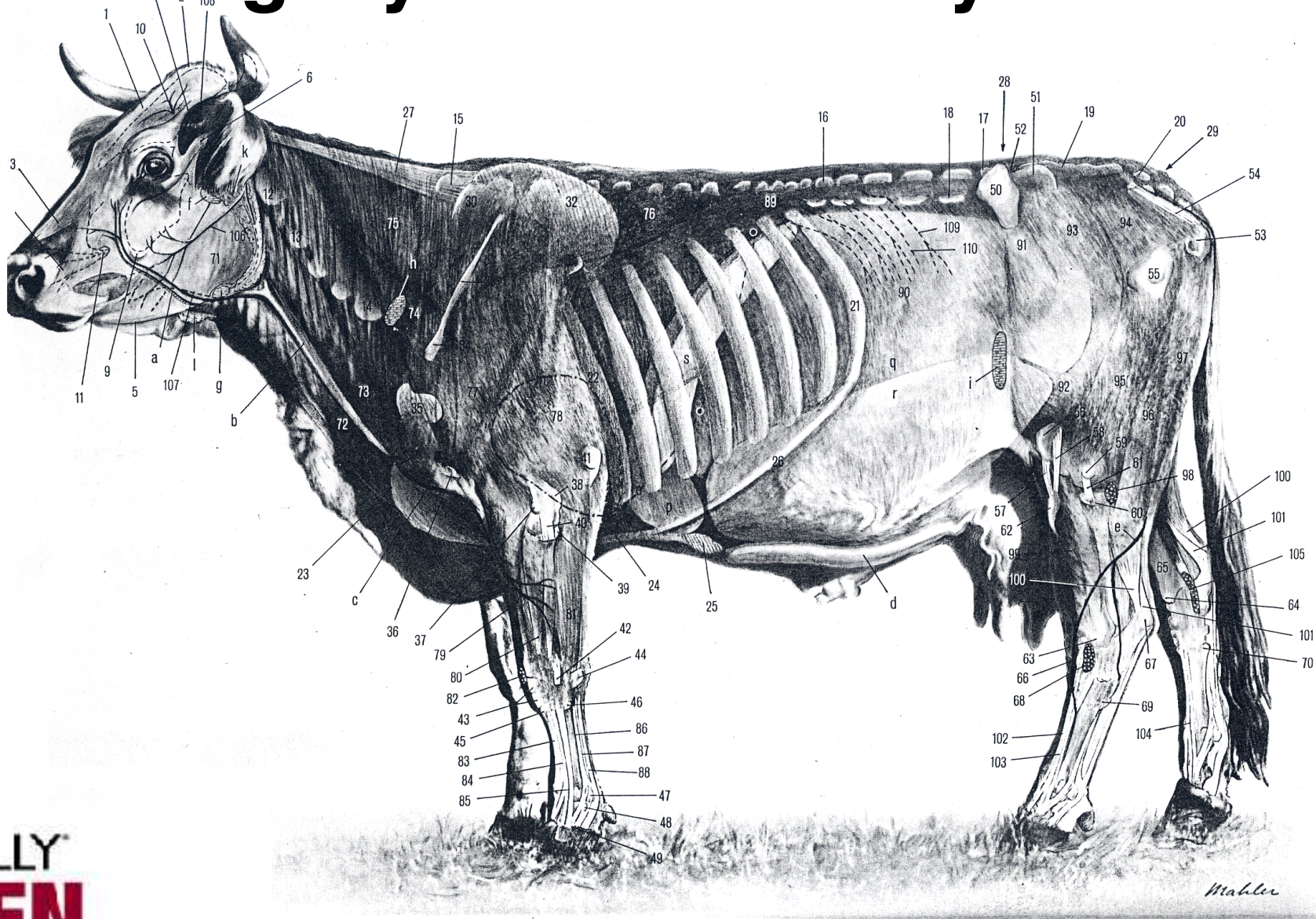


Using MySQL for Binary Data



O'REILLY™
**OPEN
SOURCE**
CONVENTION™

Mike Kruckenberg
Tufts University Sciences Knowledgebase
July 28, 2004
michael.kruckenberg@tufts.edu

What is TUSK?

- Tufts University Sciences Knowledgebase
- Started ~1996
- Web-based course, content & knowledge management
- Digital library (images, pdfs, xml docs)
- Built using Apache, (mod) Perl & MySQL
- Used primarily by Medical, Dental and Veterinary schools at Tufts
- Constant influx of new data
- ~10,000 images in 2000
- Currently ~515,000 images

The Image Serving Problem

- generic image serving
- requiring meta-data
- data integrity
- user interface for uploading images with locking and transactions
- restricting permissions
- scalability & high availability

Why Binary in MySQL?

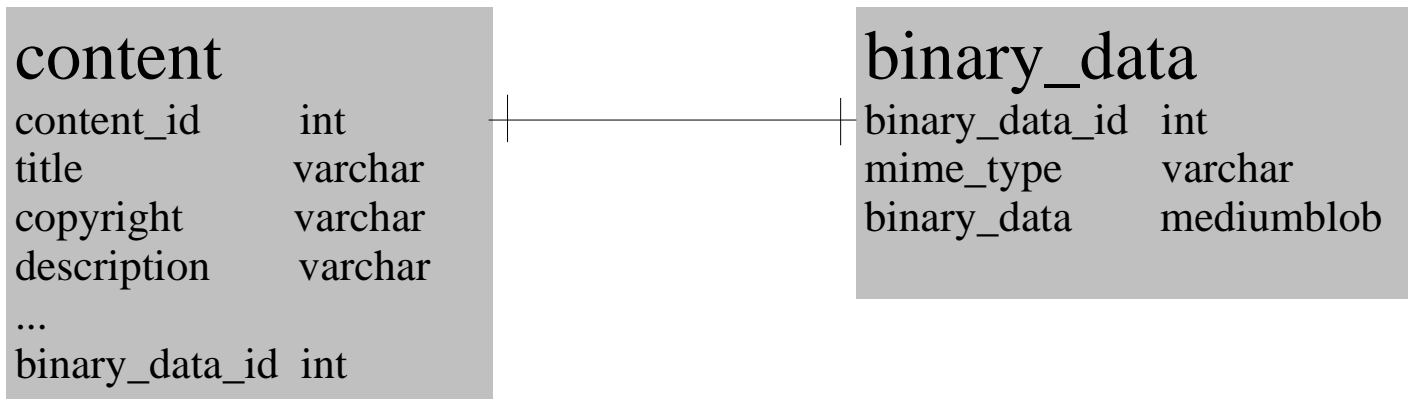
- utilize database built-in features
 - referential integrity
 - locking mechanisms
 - transactions
 - access control
 - network access
 - replication/clustering
- tunable
- centralize backup/recovery
- extra layer of security

Why not Binary in MySQL?

- more work to set up
- requires extra step to get data in and out (load data infile, select into outfile)
- slower than the filesystem?
- need appropriate amount of memory
- throughput of large data ties thing up
- no filesystem quotas
- battle against “conventional wisdom”

Data Model

- know your data
- keep binary in table with fields needed only along with the binary
 - query cache more effective
 - blob not loaded when not needed
 - keeps frequent changing meta-data table smaller
 - performance gain for fixed-width columns on meta data



Apache Handler

```
package Apache::TUSKData;
use strict;
use Apache::Constants qw(:common);
use TUSK::Core::DB;

sub handler {
    my $r = shift;
    my $id = $r->path_info;
    ## a little check to make sure we've got an id
    $id =~ s/\\//g;
    return NOT_FOUND unless ($id && $id =~ /^\\d+$/);

    ## grab the data from the database
    my $dbh = TUSK::Core::DB::getReadHandle;
    my $query = "select mime_type,binary_data
                from binary_data where binary_data_id = ?";
    my $sth = $dbh->prepare($query) || die "can't prepare";
    $sth->execute($id) || die "can't execute";

    # the id is a unique primary key, only one row to get
    my $row_ref = $sth->fetchrow_arrayref;

    # Set up the header and send it
    $r->content_type($row_ref->[0]);
    $r->send_http_header;

    # And return the data
    $r->print($row_ref->[1]);
    return OK;
}
1;
```

Apache Config

```
<Location /binary/>  
  SetHandler perl-script  
  PerlFixupHandler Apache::SizeLimit  
  Options +ExecCGI  
  PerlHandler Apache::TuskData  
</Location>
```

Performance Comparison

- simulate user experience, metrics using http requests
- Apache 1.3.31/Perl 5.8.0/MySQL 4.0.20 – 32bit
- Sun E250 dual UltraSparc – 2G mem

		avg seconds to get image		
		<u>UFS</u>	<u>MyISAM</u>	<u>InnoDB</u>
image size	1-4K	0.04	0.11	0.12
	22-24K	0.06	0.13	0.15
	100K	0.12	0.21	0.21
	3-4Mb	2.50	3.20	3.24

Things to consider . . .

- database tuning (query cache etc)
- breaking data into smaller bits
- prepared statements/stored procedures

Lessons Learned

- backup time is no longer quick
- set `max_allowed_packets` (server and client)
- not all files can be served from database
- build application to work along with database scaling methods

Thank You

- Using MySQL for Binary Data (this presentation)
 - http://mike.kruckenberg.com/presentations/using_mysql_binary.pdf
- MySQL Mailling List
 - <http://lists.mysql.com/mysql>
- Using Oracle for Binary Data
 - http://otn.oracle.com/products/intermedia/htdocs/why_images_in_database.html
- Using PHP to break and serve images in chunks
 - http://techrepublic.com.com/5100-6315_11-5053795.html
- VB and MySQL Blobs
 - <http://www.devaricles.com/c/a/MySQL/MySQL-and-BLOBs>

Comments, Questions?