SQL (and MySQL)

Useful things I have learnt, borrowed and stolen
MySQL quirks

- MySQL truncates data
MySQL quirks

- MySQL truncates data

```sql
CREATE TABLE pets (  
id INT UNSIGNED NOT NULL AUTO_INCREMENT,  
type CHAR(3) NOT NULL,  
PRIMARY KEY id (id)  
);
```
MySQL quirks

- MySQL truncates data

```sql
CREATE TABLE pets (  
id INT UNSIGNED NOT NULL AUTO_INCREMENT,  
type CHAR(3) NOT NULL,
PRIMARY KEY id (id)
);

INSERT INTO pets VALUES (1, 'caterpillar');
```
MySQL quirks

- MySQL truncates data

```sql
CREATE TABLE pets (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    type CHAR(3) NOT NULL,
    PRIMARY KEY id (id)
);

INSERT INTO pets VALUES (1, 'caterpillar');

Query OK, 1 row affected, 1 warning (0.02 sec)
```
MySQL quirks

- MySQL truncates data

```
SELECT * FROM pets;

+----+------+
| id | type |
+----+------+
|  1 | cat  |
+----+------+
1 row in set (0.00 sec)
```
MySQL quirks

- MySQL truncates data
- There is a solution!
MySQL quirks

- MySQL truncates data
- There is a solution!
- Make MySQL use strict
MySQL quirks

- MySQL truncates data
- There is a solution!
- Make MySQL use strict
- In your my.cnf / my.ini

sql-mode=STRICT_ALL_TABLES
MySQL quirks

- MySQL allows zero dates
MySQL quirks

- MySQL allows zero dates

```sql
ALTER TABLE pets ADD COLUMN date_bought DATETIME NOT NULL;
```
MySQL quirks

- MySQL allows zero dates

```
ALTER TABLE pets ADD COLUMN date_bought DATETIME NOT NULL;

SELECT * FROM pets;
```

```
+----+------+---------------------+
<table>
<thead>
<tr>
<th>id</th>
<th>type</th>
<th>date_bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cat</td>
<td>0000-00-00 00:00:00</td>
</tr>
</tbody>
</table>
+----+------+---------------------+

1 row in set (0.00 sec)
MySQL quirks

- MySQL allows zero dates
- There is a solution!
MySQL quirks

- MySQL allows zero dates
- There is a solution!
- In your my.cnf / my.ini

sql-mode=NO_ZERO_DATE
MySQL quirks

- MySQL allows zeroes IN dates
MySQL quirks

- MySQL allows zeroes IN dates
- Even with NO_ZERO_DATE
MySQL quirks

- MySQL allows zeroes IN dates
- Even with NO_ZERO_DATE
- There is a solution!
MySQL quirks

- MySQL allows zeroes IN dates
- Even with NO_ZERO_DATE
- There is a solution!
- In your my.cnf / my.ini:

  sql-mode=NO_ZERO_IN_DATE
MySQL quirks

- But you want to stop truncation, and zero dates, and zeroes in dates?
MySQL quirks

- But you want to stop truncation, and zero dates, and zeroes in dates?
- There is a solution!
MySQL quirks

- But you want to stop truncation, and zero dates, and zeroes in dates?
- There is a solution!

sql-mode=TRADITIONAL
Why use MySQL?

- Free
- Fast
- Full-text searching
- Scalable
- Popular
- Flexible
- Well supported
- It's already there
- Look at the alternatives
The alternatives

- **Oracle**
- **Enterprise Edition** starts at $40,000
- **Express Edition** is free
  - 4GB of user data, use up to 1GB of memory, and use one CPU on the host machine.
- **Worth looking into if:**
  - your apps aren't going to grow
  - your apps are going to grow really big
The alternatives

- SQL Server
  - Enterprise Edition costs around $25,000
  - Standard Edition c.$6000
  - Express Edition is free
    - 4GB of user data, use up to 1GB of memory, and use one CPU on the host machine.
- Worth looking into if:
  - your apps aren't going to grow
  - your apps are going to grow really big
  - you're using a Microsoft platform
The alternatives

- PostgresSQL
- Open source
- Multi-platform
- I've never used it but have heard those that have say things like,

"If Postgres is your answer, you're asking the wrong question"

- Rumour is, it's as quirky as MySQL, in different ways
- But less popular
The alternatives

- IBM DB2
- Anyone?
The alternatives

- SQLite
- Free
- Fast
- Small
- Limited functionality
- Great for
  - Simple, small, datasets
  - High frequency requests
  - Acting as a results cache between an application and a larger database
Some groovy SQL

- Finding all the rows in a table that have non-unique values

```
CREATE TABLE books (  
    id INT AUTO_INCREMENT,  
    title VARCHAR(255),  
    author VARCHAR(255),  
    PRIMARY KEY id (id)  
);

INSERT INTO books VALUES  
(none 'Perl Best Practices','Conway'),  
(none 'Object Oriented Perl','Conway'),  
(none 'Perl Best Practices','Conway');
```
Some groovy SQL

- Use a sub-query:
  ```sql
  SELECT * FROM books WHERE title IN ( SELECT title FROM books GROUP BY title HAVING COUNT(*) > 1 );
  +----+---------------------+--------+
  | id | title | author |
  +----+---------------------+--------+
  | 1 | Perl Best Practices | Conway |
  | 3 | Perl Best Practices | Conway |
  +----+---------------------+--------+
  2 rows in set (0.10 sec)
Some say DBIx::Class
I don't like it
Too much abstraction
Need to learn a new meta-language
Where clauses are just not supported beyond the basics
End up configuring packages when you change your schema
I just don't get what I'm supposed to gain from it
Making life easier: SQL::Abstract

- Great for compiling 90% of the SQL statements
- Great for dealing with inserts and updates

```perl
$sql = SQL::Abstract->new();

my $values = { type => 'dog', date_bought => '2008-09-09 00:00:00' };

my ($insert, @binds) = $sql->insert('pets', $values);
my ($update, @binds) = $sql->update('pets', $values);
```
Making life easier: ORLite

- Object-relation system specifically for SQLite

```perl
package Foo;

use strict;
use ORLite 'data/sqlite.db';

my @dog = Foo::pets->select(
    'where type = ?',
    'dog',
);

- Good for simple datasets
```
Thanks to

- Smylers
  - http://use.perl.org/~Smylers/journal/34246