enovafinancial™

How Enova Financial Uses Postgres

Jim Nasby, Lead Database Architect

Overview

- Who are we?
- Some history
- Migration
- Where are we today? (The cheerleading section)
- Cool stuff
- Q&A

enovafinancial™

Who are we?

enovafinancial™

Who are we?

According to our website...

“*At Enova Financial, we provide reliable financial options for everyday, hard-working people.*

As a global financial resource for under-served consumers, we operate a portfolio of businesses that offer a variety of online credit products and services in the United States, United Kingdom, Australia and Canada.”

enovafinancial™

Who are we?

In simpler terms, every year we provide over **$1B** of loans to people who are not served by traditional financial institutions.

Founded in 2004

Launched in August 2005

Migrated from MySQL to Postgres in August 2006 (MySQL free since Aug. 35th!)

Hired first dedicated database person Sept. 2007

Why does Postgres matter to us?

enovafinancial™

Why does Postgres matter to us?

Quite simply, Postgres is our lifeblood

- All of our external systems rely on Postgres
- Most of our internal systems do as well
- During the day, downtime costs over $100k/hour in lost revenue … someone's annual salary!

Why does Postgres matter to us?

Postgres also gives us opportunities not available from other databases:

- Easier feature additions
- Better open-source tool choices
- Unique capabilities

enovafinancial™

The Migration Saga

The Migration Saga

“All databases suck, they just suck in different ways.”

- Me, ca. 1999

The Migration Saga – Why MySQL?

- Default for Rails (at the time)
- “Everyone runs MySQL”
- MySQL did allow us to get everything up and running, creating a fully functional business

The Migration Saga – Why Change?

- Data integrity problems
- Scale
- MySQL made it too easy to do things wrong
  - Wrong table type
  - Silent truncation

“I was approved for a $1200 loan, why did you only give me $200?!!?!”

The Migration Saga – Why Postgres?

- Easy to pronounce name!
- MS-SQL didn't play well with Rails
- DB2 was fairly expensive
- Oracle was unfairly expensive
  - $100k just to get started
  - Would rather hire someone

enovafinancial™

The Migration Saga – Downsides

- Getting started was frustrating due to lack of experience
- Postgres is “unforgiving”
- Finding people to hire is difficult

The Migration Saga – Migration

- Wrote “training wheels” (today you would use mysqlcompat from pgFoundry)
- Minimized and controlled hand-written SQL
  - All SQL for the app was in 2 files
  - All automated reports/queries were in VCS

Migration took 2 months; launched 3 months after project start

The Migration Saga – Lessons

- Methodical
- Automated (or at least repeatable)
- Get all SQL in one or two places
- Test all that SQL

Startup Lessons

- Less raw SQL is good
- ORM and abstraction is good
- Foreign Keys are your friends
- Have your ERD available and teach business people how to write SQL

enovafinancial™

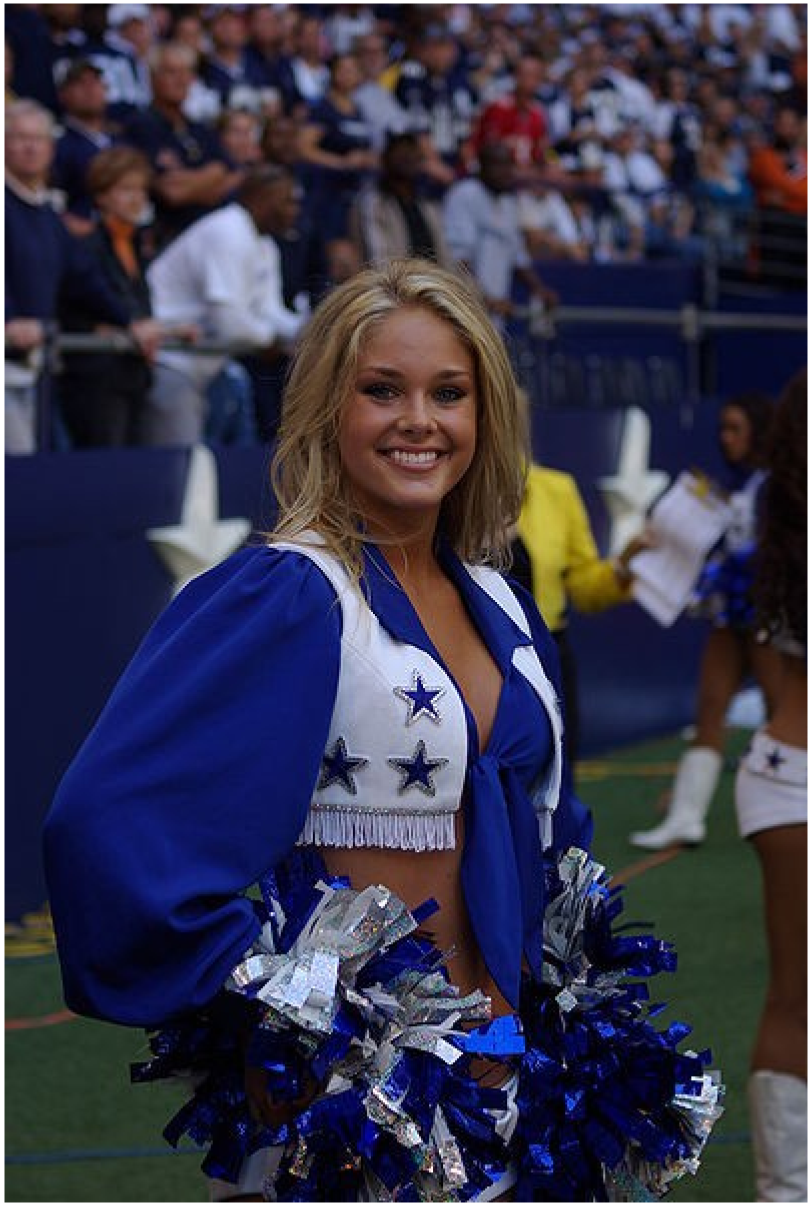
Startup Lessons

- Do the best job you can in the time you have. It's **much** easier to change things early on
- Get as much expertise as you can
- A product that's out produces infinitely more revenue than one that's not
- If you're successful, you **will** have scaling problems, so be ready
- You will eventually get tied to your technology

enovafinancial™

Where are we today?

Where are we today?



Where are we today?

- US OLTP database is 1.3TB
- 640 transactions/second average
- Peak transaction rates of over 4000/sec
- Working set is between 100GB and 200GB
- 2 US londiste slave databases for reporting
- Smaller setups for UK, Australia, Canada and joint venture, as well as some other businesses

Where are we today?

Throwing hardware at a problem **can** work:

Sept 2007 – 300GB, 4x dual-core, 32GB
Oct 2008 – 800GB, 4x 4-core, 96GB
Nov 2009 – 1TB, 4x 6-core, 192GB

Where are we today?

Open community means more options for feature development

- Hot Standby
- Multi-master londiste
- Various small tools

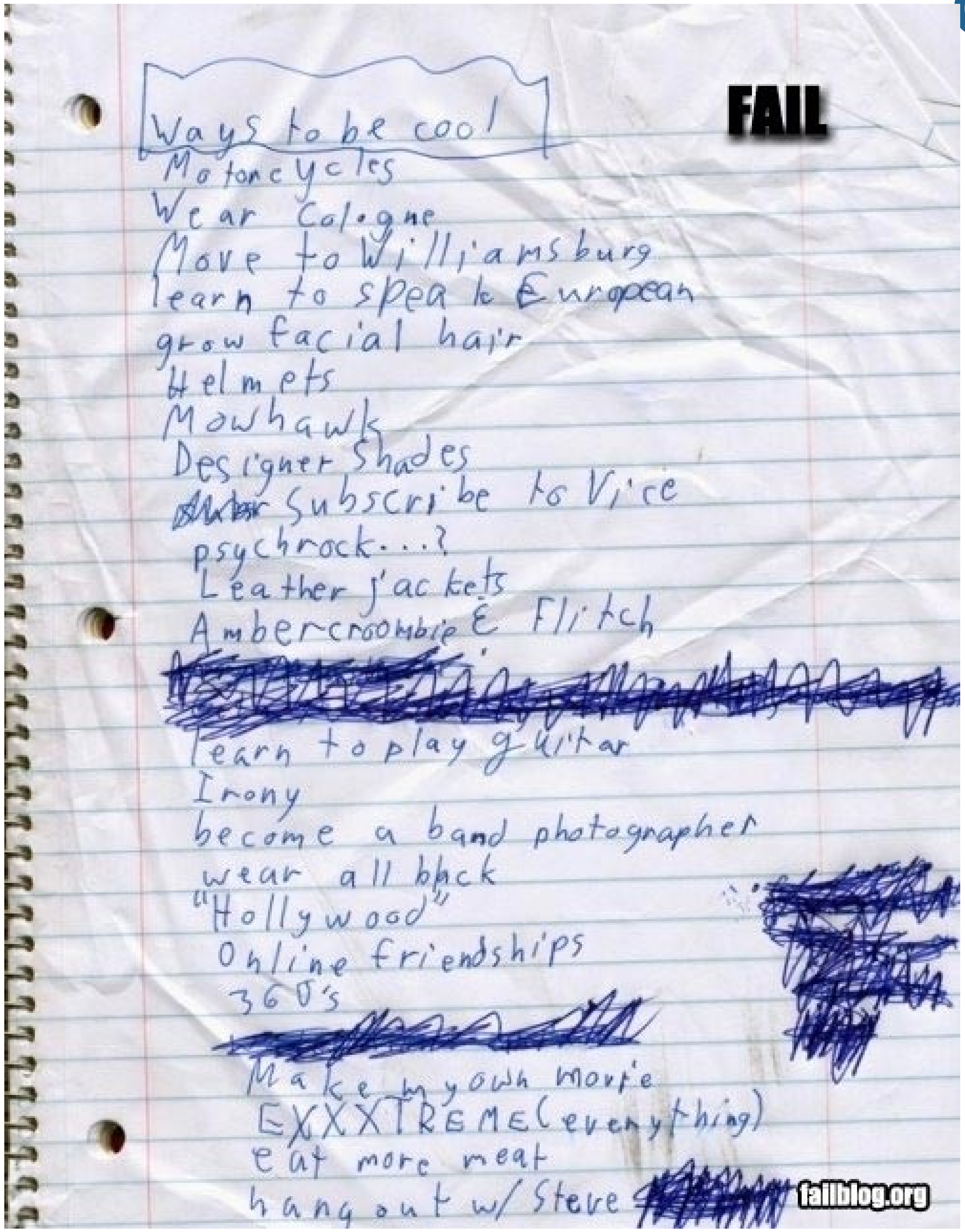
Open community also means more leverage for community efforts

- Londiste builds on some of Slony

enovafinancial™

Cool Stuff

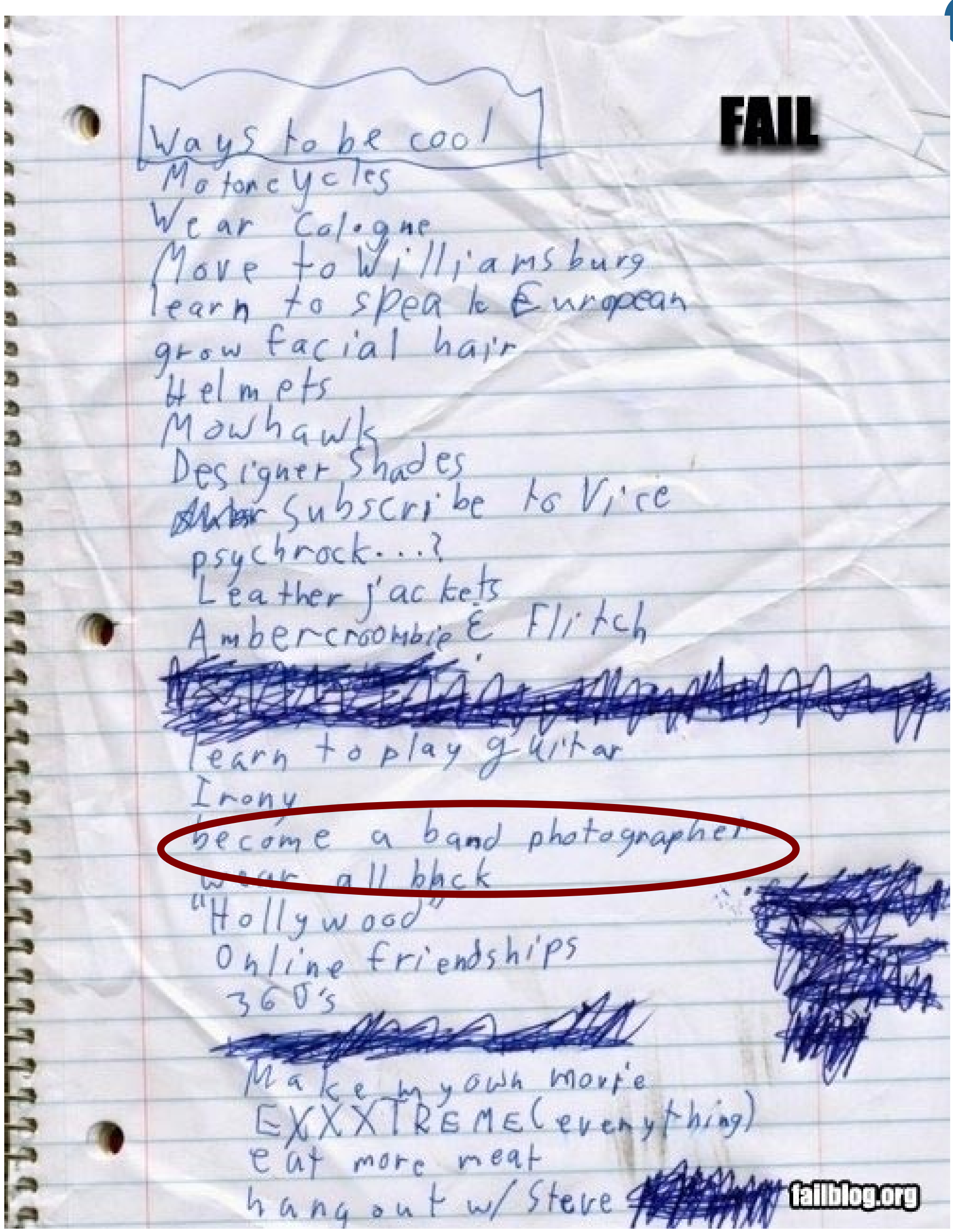
Cool Stuff



Ways to be cool
Motorcycles
Wear Cologne
Move to Williamsburg
learn to speak European
grow facial hair
Helmets
Mowhawk
Designer Shades
[illegible] Subscribe to Vice
psychrock...?
Leather jackets
Ambercrombie & Flitch
[illegible]
learn to play guitar
Irony
become a band photographer
wear all black
"Hollywood"
Online friendships
360's
[illegible]
Make my own movie
EXXXTREME(everything)
eat more meat
hang out w/ Steve [illegible]

enovafinancial™

Cool Stuff


Ways to be cool

Motorcycles

Wear Cologne

Move to Williamsburg

learn to speak European

grow facial hair

Helmets

Mowhawk

Designer Shades

[illegible] Subscribe to Vice

psychrock...?

Leather jackets

Ambercroombie & Flitch

[illegible]

learn to play guitar

Irony

become a band photographer

wear all black

"Hollywood"

Online friendships

360's

[illegible]

Make my own movie

EXXXTREME(everything)

eat more meat

hang out w/ Steve [illegible]

Cool Stuff - Inheritance

Customers have different accounts for sending and receiving money:

- Bank
- Debit / Credit card
- Paycard

enovafinancial™

Cool Stuff - Inheritance

Some attributes are common across all these different types of accounts

- account_id
- customer_id
- account_status_id

Cool Stuff - Inheritance

Some attributes are unique to specific types of accounts

- routing_number / account_number
- card_token

Cool Stuff - Inheritance

How can you reconcile the different fields?

- Master table for common stuff referenced by other tables for detailed stuff

```
SELECT …
    FROM customer_account c
        JOIN bank_account b
   USING( account_id )
```

Cool Stuff - Inheritance

How can you reconcile the different fields?

- Lots of null fields
  - customer_id NOT NULL
  - routing_number NULL
  - account_number NULL
  - card_token NULL

enovafinancial™

Cool Stuff - Inheritance

Inheritance gives the best of both worlds!

```
CREATE TABLE bank_account(...)
   INHERITS( customer_account )
CREATE TABLE debit_card(...)
   INHERITS( customer_account )
```

Cool Stuff - Inheritance

- customer_account has common fields
- bank_account and debit_card have common **and** specific fields
- Data is only stored once (in the child table)
- Which table you select from depends on what you're trying to do... is it something generic or is it specific?
  - List of customer accounts
  - Send money to customer

Cool Stuff - Constraints

My primary job responsibility is data quality. But how do I ensure that?

- Code can have bugs
  - QA isn't perfect
  - Two checks are better than one
- Application developers don't think the same way that database developers do
- Database developers often have a better “big picture” view

Cool Stuff - Constraints

What do these constraints look like?

Some are simple

- CHECK Amount >= 0
- TRIGGER due_date > today

Some are more complex

- Valid status transitions
- Cross-table conditions

Cool Stuff - Constraints

Too much of a good thing is a bad thing!

- Does this constraint make sense to be in the database? Does it have external dependencies?
- How hard will it be to write in the database?
- How general is the constraint?
- How critical is the condition we're checking? (Risk)

Cool Stuff - plPerl

What happens when pl/pgsql doesn't cut it?

- Heavy string manipulation
- Access to the outside world
- $_GLOBAL
- Building queries around NEW and OLD

Switch to pl/Perl or another procedure language

Cool Stuff - PgQ

PgQ is the queuing framework that Londiste is built on

- Items are pulled out in batches, in order of insertion
- Items can be marked for retry
- Interface is simple

enovafinancial™

Cool Stuff - PgQ

We're using PgQ to drive our “object monitor”, which is used to update our MS-SQL data warehouse

- All inserts and updates on specific objects are logged to PgQ
- There is a set returning function that will return all the rows in a table that have been inserted or updated

Cool Stuff – Patch Framework

We typically release every 2 weeks; far too often to maintain a master schema document.

cnudump.pl: Dumps complete schema from a production database, as well as data from “seed” tables

tools.schema_patches: Table that tracks what patches have been applied to a database

Cool Stuff – Patch Framework

Patches can have an arbitrary number of dependencies

```
-- patchdeps: some_older_patch
BEGIN;
SELECT tools.patch('new_patch');
…
COMMIT;
```

enovafinancial™

Cool Stuff – Unit Tests

YES YOU WANT UNIT TESTS!!!

But... maybe not to start. Typically, 50-70% of the time spent creating new functionality is devoted to unit test creation.

Our unit tests don't use scaffolds; instead, they use test data from previous tests.

Cool Stuff – Metacode

Code that writes code:

SELECT code.lookup_table_static(...);

Creates table, 2 indexes, 3 functions.

Simple tag replace system.

We're hiring!

We're looking for both DBAs and database developers and have opportunities in both Chicago and Austin, TX

http://enovafinancial.com/tech

recruiting@enovafinancial.com

Q & A

Questions?

jnasby@enovafinancial.com
jim@nasby.net