Reporting and the First Law of Holes

Dashboard

Lifetime Sales

$5111,835.01

Average Orders

$68.32

Last 5 Orders

Customer	Items	Grand Total
	4	$85.58
Diane Fortin	2	$113.95
nepton colette	8	$352.24
Marie-Josée Lévesque	1	$229.89
Marie-Josée Lévesque	2	$110.97

Last 5 Search Terms

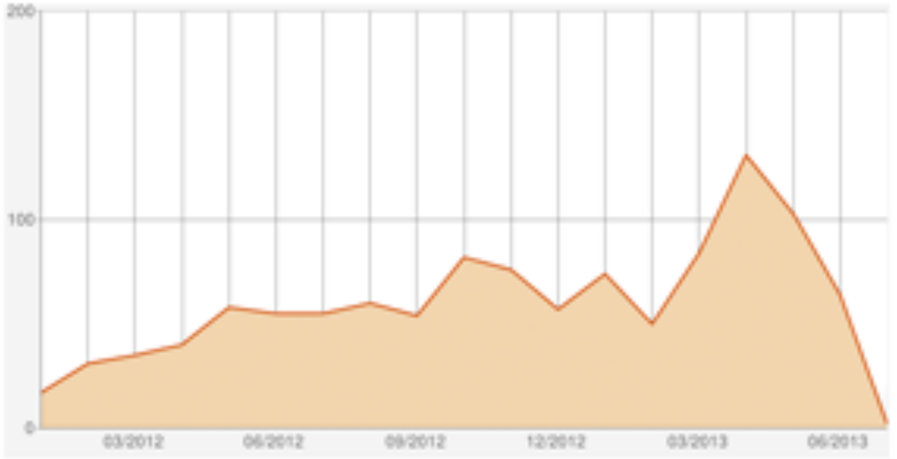
Search Term	Results	Number of Uses
Dressage Mercier	17	310
Chien Hollandais de Chasse ...	2	1186
Animalerie Saint Mathieu de...	2	345
Cherche petit vetements pou...	37	1992
Collier pour chats à gatineau	13	528

Top 5 Search Terms

Search Term	Results	Number of Uses
shampoing ameryst	1	19606
Accessoire pour chien	510	10377
oligo-elements	8	9693
cage pour chien	511	7985
Twit palace special	36	6547

Orders | Amounts

Select Range: 2YTD



Revenue	Tax	Shipping	Quantity
$170,102.40	$11,057.52	$8,478.90	1128

Bestsellers | Most Viewed Products | New Customers | Customers

Product Name	Price	Quantity Ordered
Collation et bâton de dentition petit pour chien	$7.95	107
Collier anti-aboiement électrostatique et ultrason.	$65.94	55
Séchoir toilettage, pour animaux Andis Quiet aire	$43.95	51
Cologne ou parfum pour animal 75g	$7.35	35
Articulations chien et chat, Animoflex 100ml, soins animaux	$32.46	29

There's a better way.

Easier to start with.
Easier to maintain.
Scales to petabytes.

Developers?

Non-developers?

Sebastian von Conrad

@vonconrad

NATIONAL GEOGRAPHIC

Photograph by Natalia Robba, My Shot


envato


themeforest


activeden


videohive


photodune


graphicriver


3docean


codecanyon


audiojungle


envato studio


tuts+

83rd largest site in the world

> $250 million of community earnings

This happened to me.

Everywhere I've worked.

Day after launch.

Mission:

Yesterday's sales?

This is the point where we realise:

We can solve this with our ORM!

(If we have one.)

```
Sale.where(created_at: 📅).count
```

Let's automate!

Admin panel!
Dashboards!

We are masters at putting ourselves into holes we **can't get out** of.

We don't think about reporting up front.

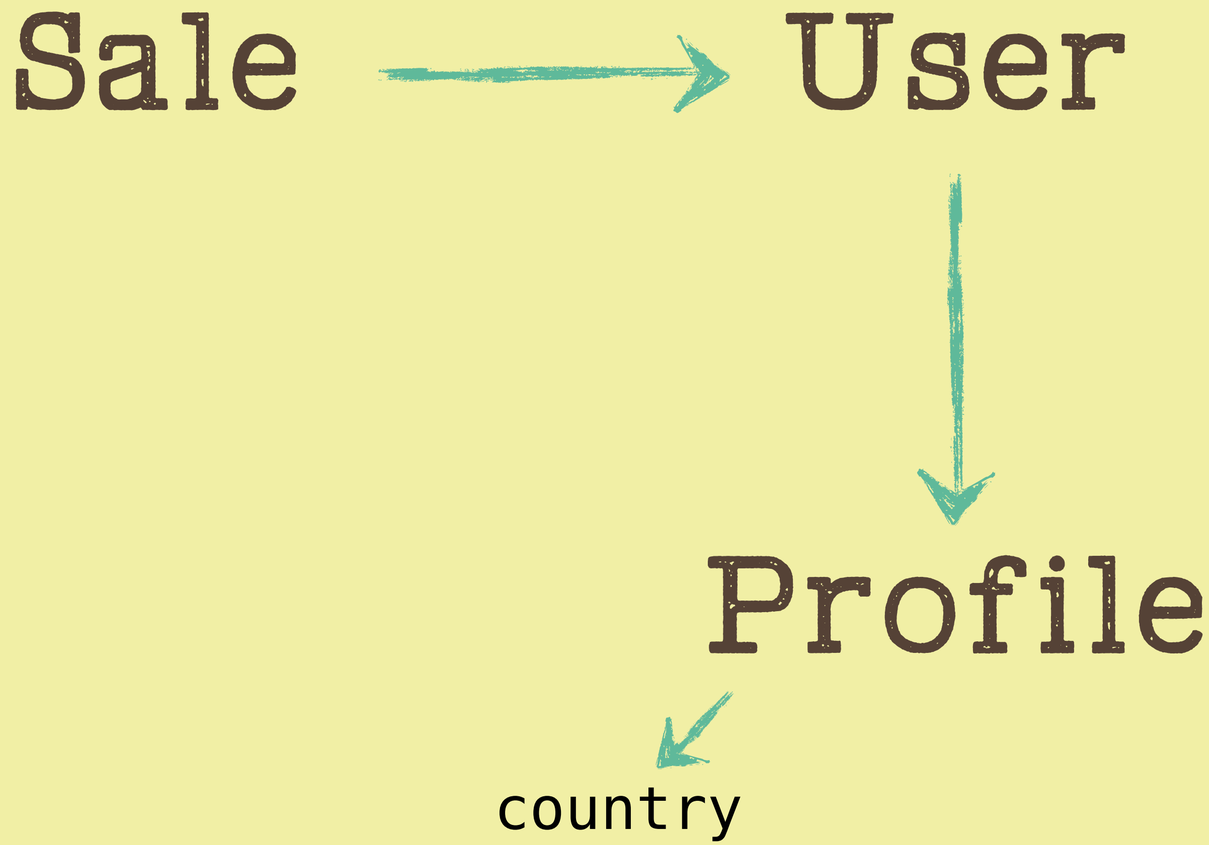
We put reports
inside our apps.

We shouldn't.

We can't anticipate what questions will be asked later.

Mission:

Last month's top countries?


Sale
User
Profile
country

This is the point where we realise:

We can solve this with SQL!

```sql
SELECT p.country, count(s.id) AS num_sales
FROM sales s
INNER JOIN users u ON s.user_id = u.id
INNER JOIN user_profiles p ON p.user_id = u.id
WHERE s.created_at BETWEEN 📅 AND 📅
GROUP BY p.country;
```

Let's add it to the
dashboard.

But there's no time to update existing reports.

The way we build applications does not cater to reporting.

Our 3NF database is good for the app, but **bad for reporting**.

JOIN JOIN JOIN

Sub-optimal indexing.

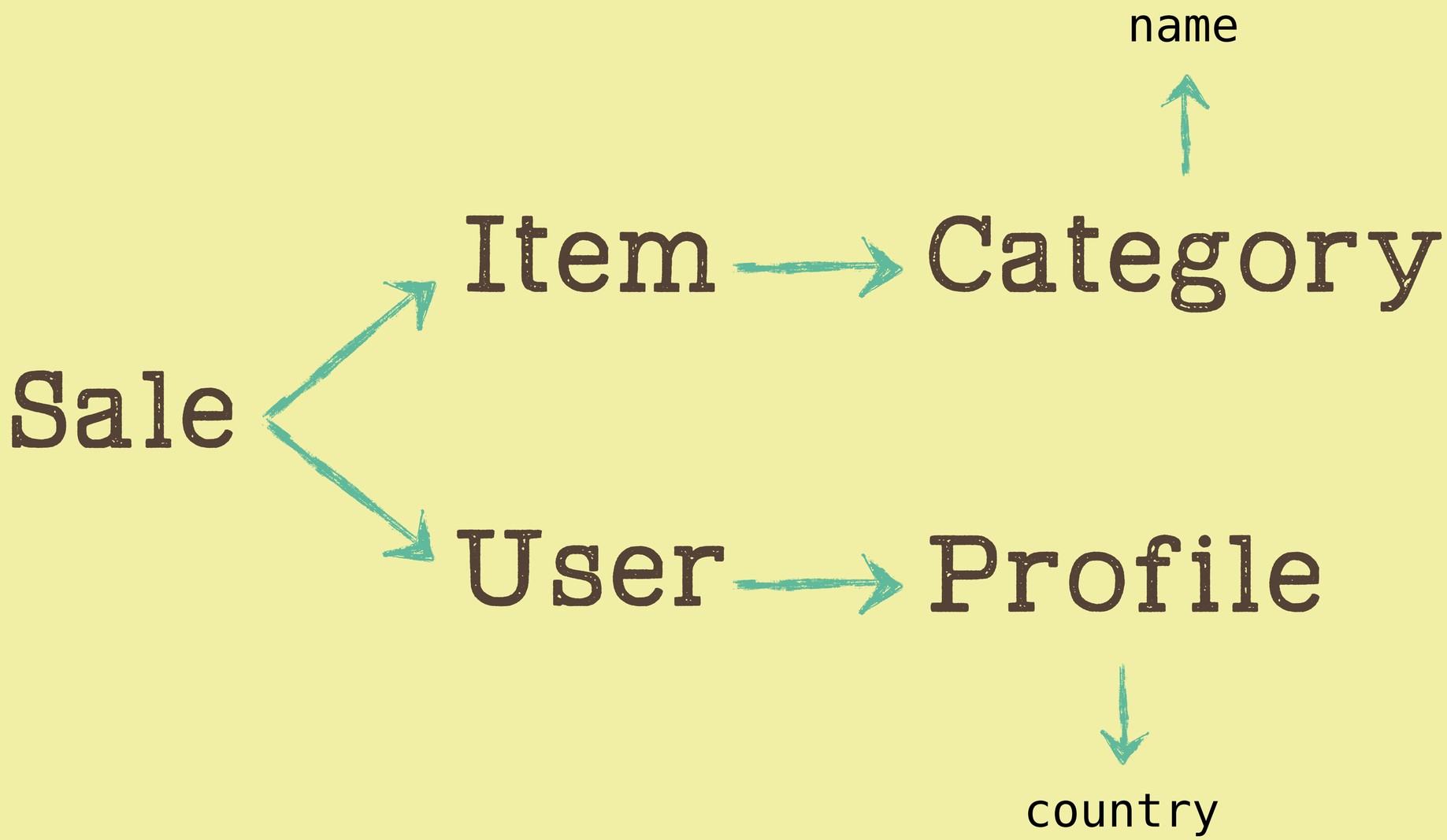
Expensive reporting queries can hurt production systems.

Replication?

Expensive.
Prone to lag.

Mission:

Which categories do well in which countries?


name
Item
Category
Sale
User
Profile
country

```
SELECT p.country, c.name, count(s.id) AS num_sales
FROM sales s
INNER JOIN items i ON s.item_id = i.id
INNER JOIN categories c ON i.category_id = c.id
INNER JOIN users u ON s.user_id = u.id
INNER JOIN user_profiles p ON p.user_id = u.id
WHERE s.created_at BETWEEN 📅 AND 📅
GROUP BY p.country, c.name;
```


FINISH
13

This is the point where we realise:

We never liked writing SQL in the first place.

Right tool for the right job.

Right?

Introducing:

The 18-month data warehouse project!

With an ETL process that reaches into the application database.

And we probably won't change the dashboards that already work.

You have to **understand the object model** in order to understand the data.

Data warehouse developers will need intimate knowledge of the application.

Misunderstandings can lead to wrong information.

Schema changes suck.

Do you CI your reporting queries?

Fine, but do you
CI your ETL?

But we're not done
yet, are we?

Mission:

Items featured (and not) on homepage?

We can do this, because we think about the future.

Right?

item_id	feature_start_at	feature_end_at
34292	2015-02-01	2015-02-03
64233	2015-02-02	2015-02-03
77245	2015-02-05	2015-02-05
212	2015-01-23	2015-02-08
22196	2015-02-06	2015-02-08
...	...	...

No, we haven't.

id	price	category_id	...	featured
209	40.00	112	...	false
210	34.00	2	...	false
211	78.00	61	...	false
212	53.00	54	...	true
213	12.00	14	...	false
...	...	...	...	...

We don't have historical data.

This is the point where we realise:

We can't solve this problem.

ORMs maintain current state, not history.

ETL is not idempotent.

ETL needs to load
everything.

So what can we
do about it?

First Law of Holes:
if you find yourself
in one, stop digging.

– Will Rogers, 1911

So, how do we
stop digging?

This happened to me.

Last product I built.

Step 1: Need?
Step 2: ?
Step 3: ?
Step 4: ?
Step 5: ?

Not a trick question!

Different mental models.

~~What objects are?~~
What do they do?

Series of business processes.

Events!

Signup. Sale.
Subscription.
Cancellation.
Refund. Etc.

These events are key.

Separate concerns.

Single Responsibility Principle.

Applies to applications too.

Decouple reporting
from the application.

Step 1: Need?

Step 2: Know?

Step 3: ?

Step 4: ?

Step 5: ?

When the event happens,
what do you **know** about it?

Who or what is involved in making the event happen?

Buyer
Who are they?
Where are they?
When did they sign up?
Previous purchases?

Buyer
Who are they?
Where are they?
When did they sign up?
Previous purchases?

Seller
Who are they?
Where are they?
When did they sign up?
Previous sales?

Buyer
Who are they?
Where are they?
When did they sign up?
Previous purchases?

Item
Featured on homepage?
Category?
Price?

Seller
Who are they?
Where are they?
When did they sign up?
Previous sales?

Category
Number of items?
Price range?
Facets?

Buyer
Who are they?
Where are they?
When did they sign up?
Previous purchases?

Item
Featured on homepage?
Category?
Price?

Seller
Who are they?
Where are they?
When did they sign up?
Previous sales?

Category
Number of items?
Price range?
Facets?

Buyer
Who are they?
Where are they?
When did they sign up?
Previous purchases?

Item
Featured on homepage?
Category?
Price?

Transaction
Amount?
Payment gateway?
Coupon?
Tax?

Seller
Who are they?
Where are they?
When did they sign up?
Previous sales?

Category
Number of items?
Price range?
Facets?

Buyer
Who are they?
Where are they?
When did they sign up?
Previous purchases?

Affilitates
Were they referred?
Who is the affiliate?

Item
Featured on homepage?
Category?
Price?

Transaction
Amount?
Payment gateway?
Coupon?
Tax?

Seller
Who are they?
Where are they?
When did they sign up?
Previous sales?

Collect everything.

All attributes from domain objects == valid strategy.

Step 1: Need?
Step 2: Know?
Step 3: Denormalise
Step 4: ?
Step 5: ?

Denormalise!

Seriously, let's add some redundant data.

Design data structure for how we want it read.

1NF is better for reporting than 3NF.

Make it immutable.

Data is never wrong.

At worst, it's only missing some events.

State as of event is maintained forever.

Then, we serialise.
(I like JSON.)

job_id
job_purchased_at
service_id
service_name
service_price_in_cents
service_turnaround_in_days
service_revision_requests
service_currently_featured
service_created_at
service_approved_at
service_approved_by_id
service_approved_by_envato_id
service_approved_by_username
service_approved_by_full_name
service_approved_by_email_address
service_category_id
service_category_name
service_category_top_level_name
service_category_min_price_in_cents
service_category_max_price_in_cents
service_category_parent_id
service_category_parent_name
duration_between_approval_and_purchase_in_seconds
service_enquiry_count
service_enquiry_message_count
custom_job
buyer_login_method

buyer_id
buyer_envato_id
buyer_username
buyer_full_name
buyer_email_address
buyer_ip_address
buyer_city
buyer_country
buyer_account_created_at
provider_id
provider_envato_id
provider_username
provider_full_name
provider_email_address
provider_city
provider_country
provider_account_created_at
provider_became_provider_at
payment_uuid
payment_gateway_name
payment_gateway_reference
payment_discount_amount_in_cents
payment_charged_amount_in_cents
coupon_code
coupon_expiry
coupon_name
coupon_discount_percentage
coupon_discount_dollar_value_in_cents

Your application is not responsible for reporting.

Application generates
data, not reports.

Step 1: Need?
Step 2: Know?
Step 3: Denormalise
Step 4: Store
Step 5: ?

Application must not be allowed to read the data.

Write to log table.

Difficulty: Easy.

Might not work forever,
but good starting point.

Transactions as a bonus!

Write to log file
(with Splunk or Hadoop).

Difficulty: Moderate.

Send to SaaS in the cloud
(e.g. Tableau, GoodData).

Difficulty: Moderate

Data warehouse.

Difficulty: “Expert.”

Still an ETL, but it won't reach deep into the database.

No need to understand
the object model.

Immutability
<insert superlatives>.

ETL process
becomes idempotent.

Don’t need to keep the data in the database.

Example: Write to log table, periodically archive.

Step 1: Need?
Step 2: Know?
Step 3: Denormalise
Step 4: Store
Step 5: GOTO 1

Rinse, repeat.

Keep logging events that are important to the business.

Make report design a
first class citizen.

Build reports with tools designed to build reports.

Reports can be built
by non-developers.

Developers become responsible for supplying data, not generating reports.

Ad-hoc reporting
becomes a breeze.

Example:

Export to .csv, import in Excel.

You can capture new data for future events, but never backfill.

(Unless your job is on the line.)

Combine event logs for realtime data.

* Caveats:

You might still not be able to answer all questions.

~~Let's build another dashboard!~~
Let's log the event!

(Event Sourcing.)

How big of a hole
are you in?

Only at the beginning?

Start with this!

Already in a hole?

How can you transition to something better?

Start with the
biggest pain point.

Step 1: Need?
Step 2: Know?
Step 3: Denormalise
Step 4: Store
Step 5: GOTO 1

Iterate from there.

Make sure to delete your dashboards.

Separate reporting from your application.

Climb out of the hole.
Don't make it deeper.

Thank you!

Follow me on Twitter:
@vonconrad