Relations ❤️ SQL ❤️ ORM

Where “❤️” is the “Chris likes it better than” operator

Or,
Relations are better than SQL
and SQL is better than ORMs

Relations

What's a relation?

- A set of tuples & a heading

- A tuple is a set of attribute-values

- A heading is the attributes and their associated types

Supplies

:SID	:PID	:QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Suppliers

:SID	:NAME	:STATUS	:CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Parts

:PID	:NAME	:COLOR	:WEIGHT	:CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Oslo
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris
P6	Cog	Red	19.0	London

Sets!

- No ordering
- Unique zero-element set (empty set or “∅”)
- Super- and sub- sets are also sets
- Attn. math nerds: union and intersection commute, associate, and distribute. Union and intersection are also idempotent: (A ∪ A = A and A ∩ A = A)

Relational algebra

- Operators that, with relations are closed
  - Set operators: (union, difference, intersection)
  - Projection (π) - choose a set of attributes
  - Selection (σ) also “restriction” - choose a set of tuples where some condition holds
  - Rename (ρ) - choose an attribute and rename it
    - important for joining

Joining

- Combine two relations on common attributes
  - (In SQL as “natural join” ⋈)
- Other joins are common, too. θ-join is a join with a condition
  - (e.g. “foo JOIN bar ON foo.id = bar.foo_id”)

Relations are closed

- “Get suppliers who supply at least one red part”
- ```
  matching(suppliers,
           matching(supplies,
                    restrict(parts, color: 'Red')))
  ```
- The output of restrict and matching is the another relation that can be used with subsequent relational operators
  ```

Confusing? SQL: way worse

```
SELECT "t1"."sid" AS “sid",
       "t1"."name" AS “name",
       "t1"."status" AS “status",
       "t1"."city" AS “city"
FROM "suppliers" AS “t1"
WHERE ("t1"."sid" IN
  (SELECT "t2"."sid" AS “sid"
   FROM "supplies" AS “t2"
   WHERE ("t2"."pid" IN
     (SELECT "t3"."pid" AS “pid"
      FROM "parts" AS “t3"
      WHERE ("t3"."color" = 'Red')))))
```

Other relational algebra examples

- “Which suppliers supply no parts?”
- not_matching(suppliers, supplies)

```
+------+-------+----------+---------+
| :sid | :name | :status | :city   |
+------+-------+----------+---------+
| S5   | Adams |       30 | Athens |
+------+-------+----------+---------+
```

SQL

SQL tables aren’t sets

- They’re a bag, duplicates allowed
- Attributes are ordered
- NULL*
  - I can’t emphasize enough how lousy NULL is. In any context!
  - “I call it my billion-dollar mistake” — Tony Hoare

Anatomy of SELECT

project

rename

```
SELECT sid, pid, name, status AS stat
FROM suppliers, supplies
WHERE suppliers.sid = supplies.sid
```

join

restrict

SELECT

- Not orthogonal!
  - SELECT has too many concerns!
- Not composable!
  - Stuff gets syntactically tacked on all over the place
- Basically:

```
def do_everything(attr1, …, attr99)
  # 800 lines of “ಠ_ಠ”
end
```

So…SQL

ORMs

What are the data?

- E.g. A blog application
- Entities: Post, Comment, User
- Users write Posts and Comments
- Posts have Comments

Expression

- Post: for each do:
  - display Post
  - display Comments
  - link to User appropriately

Questions

- Is a Post or a User or a Comment really the entity we seek to describe?
- Isn’t a blog:
  - a series of posts?
  - a list of Comments?
  - a group of Users?

Data → Relationships

- “All the Posts in the last year”
- “All the Posts by Chris”
- Data are interesting in their relationship to other data

Summary

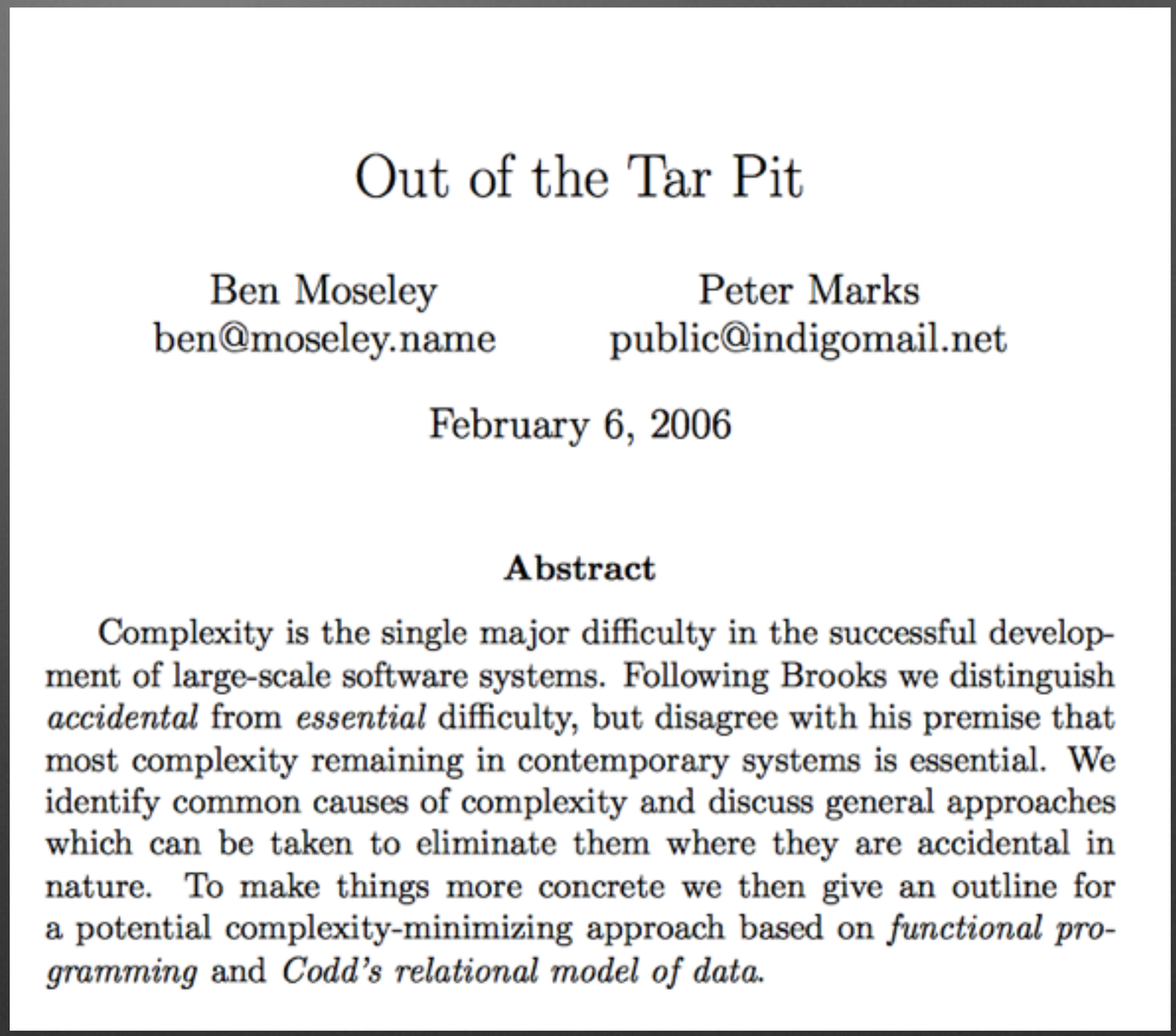
- I <u>really</u> like relational theory
- SQL diverges from relational theory in many critical ways but can be used mostly relationally
- The ORM approach does not seem to even address many of these issues

In Ruby…

- Alf is an implementation of a relational EDSL that compiles to SQL (it's a Relational-Relational Mapper)
- `gem install alf`
- http://www.try-alf.org

More info?

- Introduces FRP - Functional Relational Programming (aka “the other FRP”)



Out of the Tar Pit

Ben Moseley
ben@moseley.name

Peter Marks
public@indigomail.net

February 6, 2006

Abstract

Complexity is the single major difficulty in the successful development of large-scale software systems. Following Brooks we distinguish *accidental* from *essential* difficulty, but disagree with his premise that most complexity remaining in contemporary systems is essential. We identify common causes of complexity and discuss general approaches which can be taken to eliminate them where they are accidental in nature. To make things more concrete we then give an outline for a potential complexity-minimizing approach based on *functional programming* and *Codd's relational model of data*.

More info?

- Basically anything by C.J. Date

- This is a good intro to SQL and the relational algebra