

The SlickChair Conference Management System

Olivier Blanvillain

June 19, 2014

Table of contents

Introduction

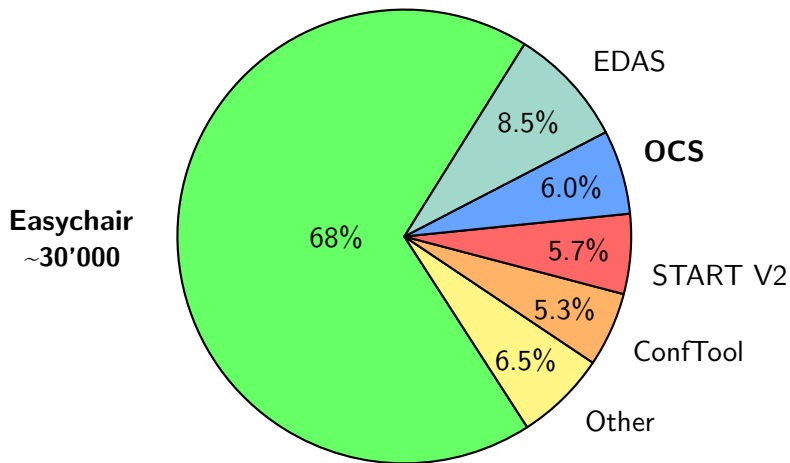
Overview of SlickChair

Versioned data and immutable database

Automatic Paper-Reviewer assignment

Conclusion and future work

Market share per number of hosted conferences, 2013



SlickChair Technologie Stack



Demo

It's demo time!

Workflow

```
case class Configuration(  
  name: String,  
  authorCanMakeNewSubmissions: Boolean = false,  
  authorCanEditSubmissions: Boolean = false,  
  pcmemberCanBid: Boolean = false,  
  ...  
)
```

```
case class Phase(  
  configuration: Configuration,  
  emails: Database => List[Email],  
  warning: Database => Option[String]  
)
```

```
val workflow: List[Phase] = List(  
  ...  
)
```

Reminder for late reviewers phase example

```
Phase(  
  
  Configuration("Review Reminder",  
    pcmemberCanReview=true,  
    pcmemberCanComment=true,  
    chairCanDecideOnAcceptance=true),  
  
  { db => List(Email(  
    lateReviewerEmails(db),  
    "Reminder: review deadline",  
    "Dear Program Committee Member, ...")) },  
  
  { _ => None }  
  
)
```

Slick Query example

```
def lateReviewerEmails(db: Database): List[String] = {  
  
    val assignmentPairs = db.assignments  
        .filter(_ .value)  
        .map(a => (a.personId, a.paperId))  
  
    val reviewsPairs = db.reviews  
        .map(r => (r.personId, r.paperId))  
  
    (assignmentPairs diff reviewsPairs)  
        .join(db.persons).on(_._1 is _ .id)  
        .map(_._2.email).list(db.s)  
  
}
```


Versioned data and immutable database

Requirement: log all actions and events

Writing logs in text files?

- Ok for debugging

- Bad to present to users

- Hard to query

Idea: save all changes made on the database!

Versioned data and immutable database: Implementation

All records must have an *updatedAt* field

Append only: updated records are added with new timestamps

No additional tables for the history

Multiple rows with the same natural key

Database objects carry a *TemporalFilter* used on all tables

Database objects are values

Queries are pure functions

Groups of *inserts/updates* are atomic

Versioned data and immutable database: DSL

```
case class Database(f: TemporalFilter, s: Session) {  
  val persons = table[Person]  
  val papers = table[Paper]  
  val paperAuthors = table[PaperAuthor]  
  ...  
  
  def asOf(date: DateTime): Database  
  def since(date: DateTime): Database  
  def until(date: DateTime): Database  
}  
  
case class Connection(s: Session) {  
  def currentDatabase(): Database  
  def insert(x: List[Model[_]]): (Database, Database)  
}
```

Automatic Paper-Reviewer assignment

Inputs:

Set of *papers*

Set of *reviewers*

Matrix of *preferences*

List of *conflicts*

$n \in \mathbb{N}$

Looking for the *best* assignment without conflicts such each papers is reviewed exactly n times.

It's NP-Complete :(

No *practical* approximation algorithm

From here it's either brute-force or heuristics

Implementation with OscalaR

```
val m = makeMatrix(nReviewers, nPapers, 0 to 1)

m.columns foreach { c => add(sum(c) == nReviewPerReviewer) }
m.rows foreach { r => add(sum(r) == nReviewPerPaper) }
conflicts foreach { c => add(m(c._1, c._2) == 0) }

maximize(weightedSum(preferences, m)) search {
  binaryMaxDegree(m.flatten.toSeq)
}
onSolution { return m } start()
```

Conclusion and future work

My personal takeaways:

- I'm never writing another line of SQL

- Who needs tests when you have type-checking?

- Agile development worked very well in this project

Future work:

- PDF metadata extraction, co-authorship, plagiarism

- Turn the versioned data DSL into a proper library

- Scala.js

Thank you for your attention!

Questions?

Bonus slide: Slick example compiled

```
SELECT x2.x3 FROM
  (SELECT x4.x5 AS x6 FROM
    (SELECT x7."PERSONID" AS x5, x7."PAPERID" AS x8
      FROM "ASSIGNMENT" x7,
        (SELECT x9."ID" AS x10, max(x9."UPDATEDAT") AS x11
          FROM "ASSIGNMENT" x9 GROUP BY x9."ID") x12
      WHERE ((x7."ID" = x12.x10)
        AND (x7."UPDATEDAT" = x12.x11)) AND x7."VALUE"
      EXCEPT SELECT x13."PERSONID" AS x5, x13."PAPERID" AS x8
      FROM "REVIEW" x13,
        (SELECT x14."ID" AS x15, max(x14."UPDATEDAT") AS x16
          FROM "REVIEW" x14 GROUP BY x14."ID") x17
      WHERE (x13."ID" = x17.x15)
        AND (x13."UPDATEDAT" = x17.x16)) x4) x18
INNER JOIN
  (SELECT x19."ID" AS x20, x19."EMAIL" AS x3
    FROM "PERSON" x19,
      (SELECT x21."ID" AS x22, max(x21."UPDATEDAT") AS x23
        FROM "PERSON" x21 GROUP BY x21."ID") x24
    WHERE (x19."ID" = x24.x22)
      AND (x19."UPDATEDAT" = x24.x23)) x2 ON x18.x6 = x2.x20
```