Eventual Heat Death

The Ultimate Consistency

Thomas Munro — PGCon 2016

An anonymous company's non-relational datastore

Site S100

Site S200

Site S300

Asset A001

Asset A002

Asset A003

Asset A004

Employee E1

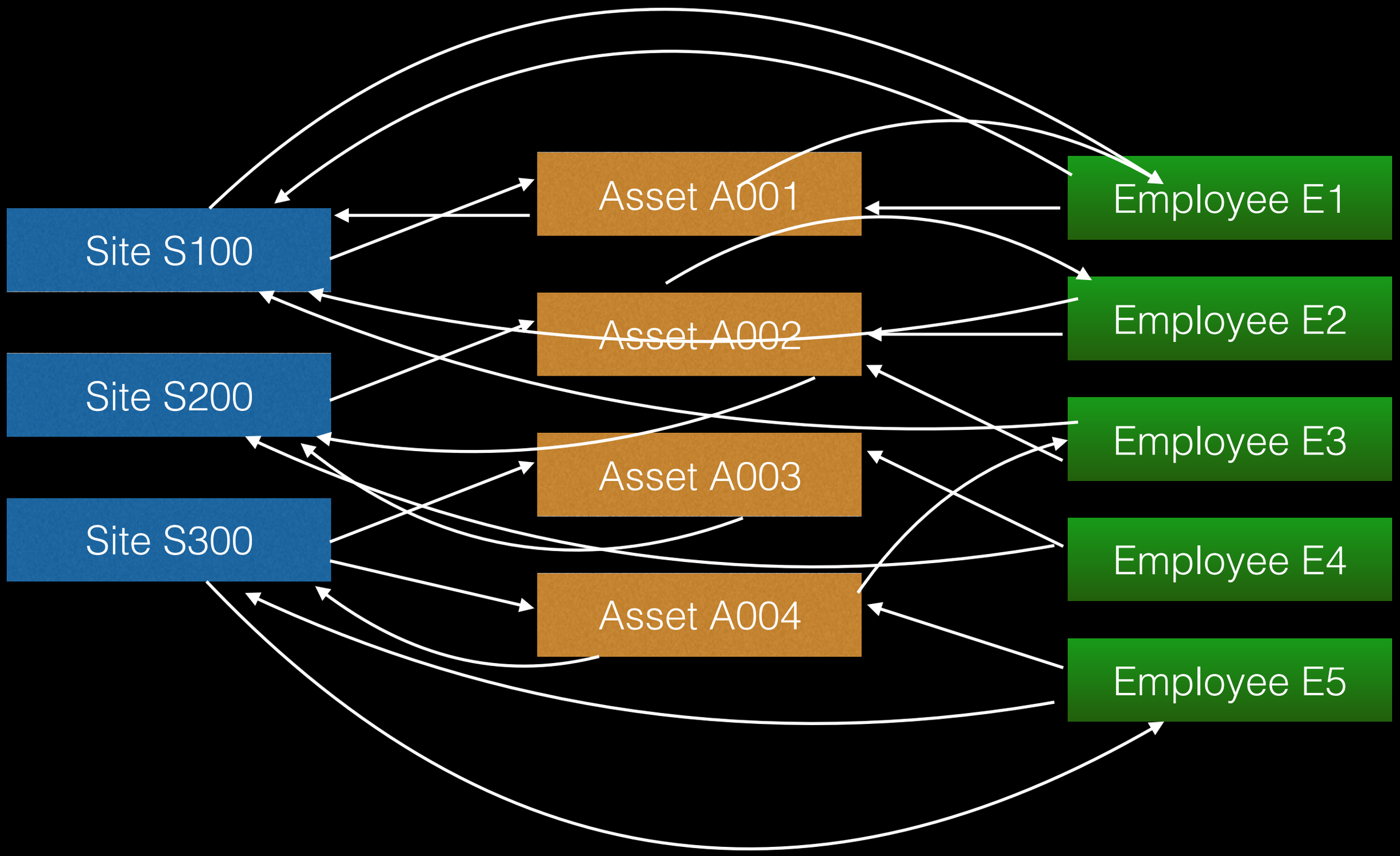
Employee E2

Employee E3

Employee E4

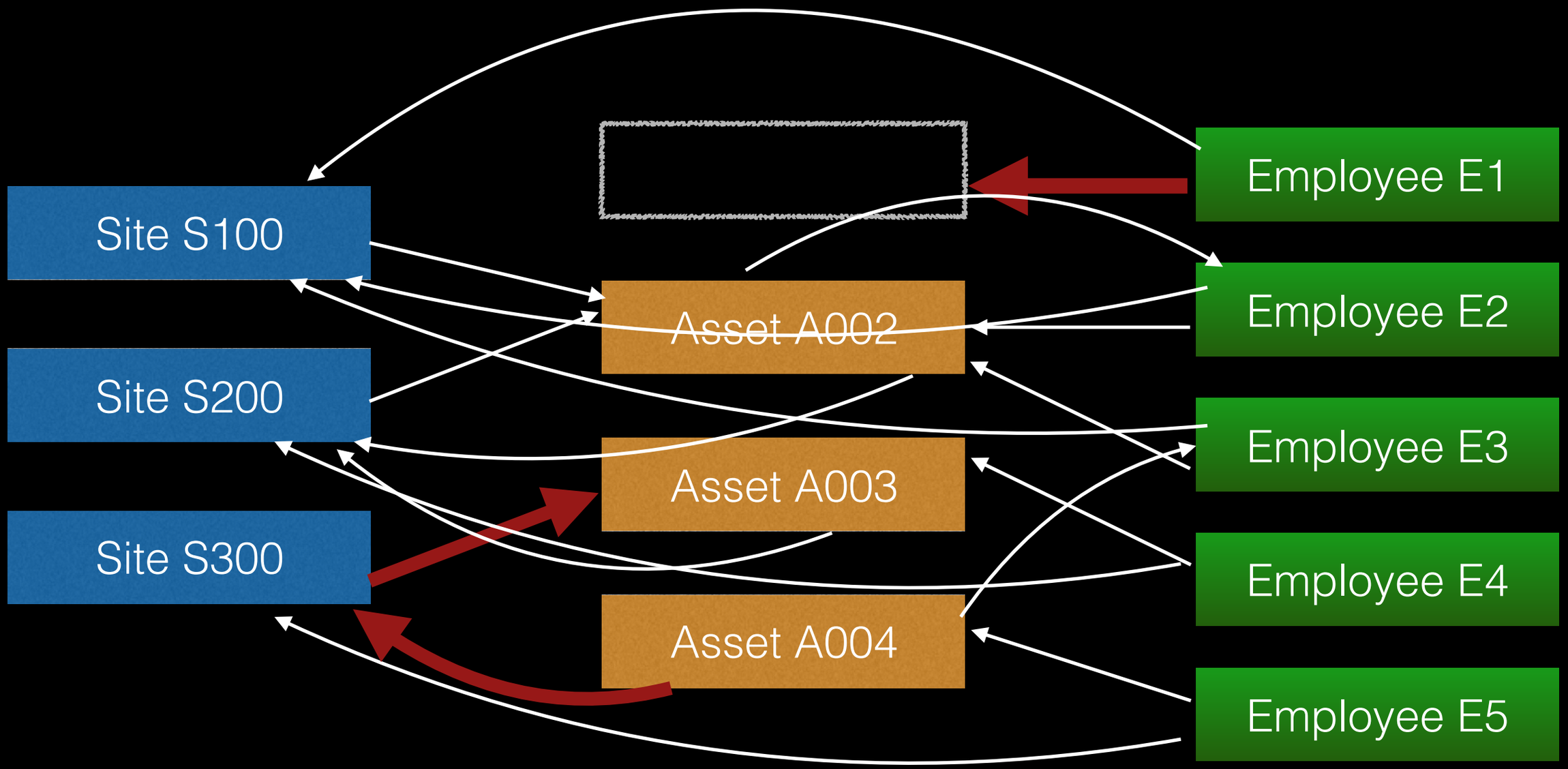
Employee E5

Things point to other things, and vice versa

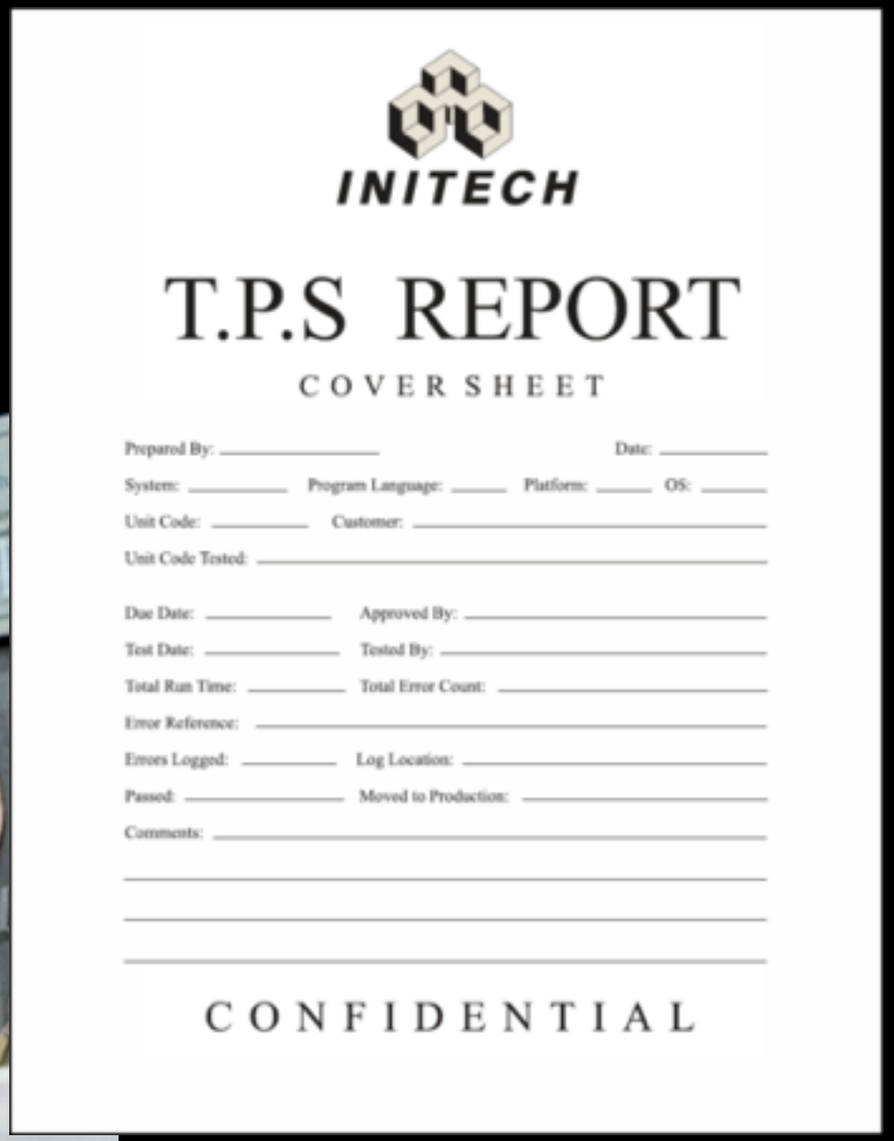


Time for some concurrent updates!

Things point to other things, and … wait, what?



Two-way **P**ointer **S**ynchronization Report



INITECH

T.P.S REPORT

COVER SHEET

Prepared By: _____ Date: _____

System: _____ Program Language: _____ Platform: _____ OS: _____

Unit Code: _____ Customer: _____

Unit Code Tested: _____

Due Date: _____ Approved By: _____

Test Date: _____ Tested By: _____

Total Run Time: _____ Total Error Count: _____

Error Reference: _____

Errors Logged: _____ Log Location: _____

Passed: _____ Moved to Production: _____

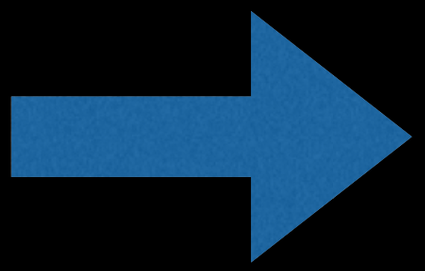
Comments: _____

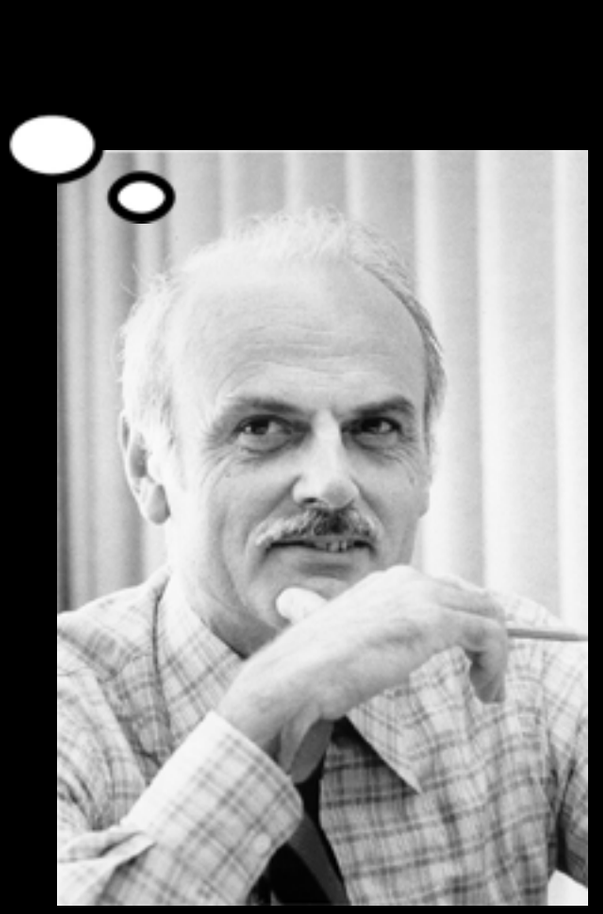
CONFIDENTIAL

“Eventual Constraints Team”

1970

Relational
data banks





We don't have problems like these, because we have SQL constraints

- `UNIQUE (foo, bar)`
- `FOREIGN KEY site_id REFERENCES site(id)`
- `CHECK (foo < 42)`

But wait, there's more! Standard SQL says you can do this:

- "There shouldn't be more than 15 students in any class"
- "No two classes containing the same student or teacher may be scheduled at the same time"

- CHECK (
  (SELECT COUNT(*)
  FROM enrolment e
  WHERE e.class_id = class_id) <= 15)

- CREATE ASSERTION the_world_is_sane
  CHECK ((SELECT …) = 42);

`ERROR: cannot use subquery in check constraint`

- No existing RDBMS supports general CHECK*
- It's quite hard to implement without concurrency
- It's really hard to implement with concurrency
- SERIALIZABLE could help with that, but it's broken, feared and/or runs like molasses
- Application code can deal with such high level stuff anyway, right?

*A couple of RDBMSs accept the syntax but fail to enforce the constraint when referenced data changes

Crazy idea:

- **Think really hard** and write an analyser that can efficiently determine which constraints need to be checked when rows in a given table are modified, and how

- Note: The same type of machinery will probably be needed for incremental materialized view maintenance

- **Require SERIALIZABLE isolation** for DML involving tables referenced by general checks

Example

- BEGIN;
  INSERT INTO enrolment VALUES (‘SQL101’, 1234);
  ERROR: insufficient transaction isolation for constraint “max_class_size_check"
  HINT: Run the statement again in transaction isolation level SERIALIZABLE
  ROLLBACK;

- BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
  INSERT INTO enrolment VALUES (‘SQL101’, 1234);
  ERROR: new row for relation “enrolment" violates check constraint “max_class_size_check"
  ROLLBACK;

A killer app for SSI?

- Users often ask how to impose such constraints, and know how to express them as queries
- Implementing equivalent concurrency-safe logic, especially in READ COMMITTED, is hard and error-prone (if you think SERIALIZABLE is only for experts, wait till you try our other levels!)
- We may have the only RDBMS that actually *could* implement general SQL CHECK without becoming unusable, thanks to our amazing SSI system

<EOF>