Evolving Database Design and Architecture

Patterns and Practices

Pramod Sadalage
ThoughtWorks Inc.
@pramodsadalage

Patterns of Database Changes

Patterns of Database Changes

- Architecture

Patterns of Database Changes

- Architecture
- Structure

Patterns of Database Changes

- Architecture
- Structure
- Data Quality

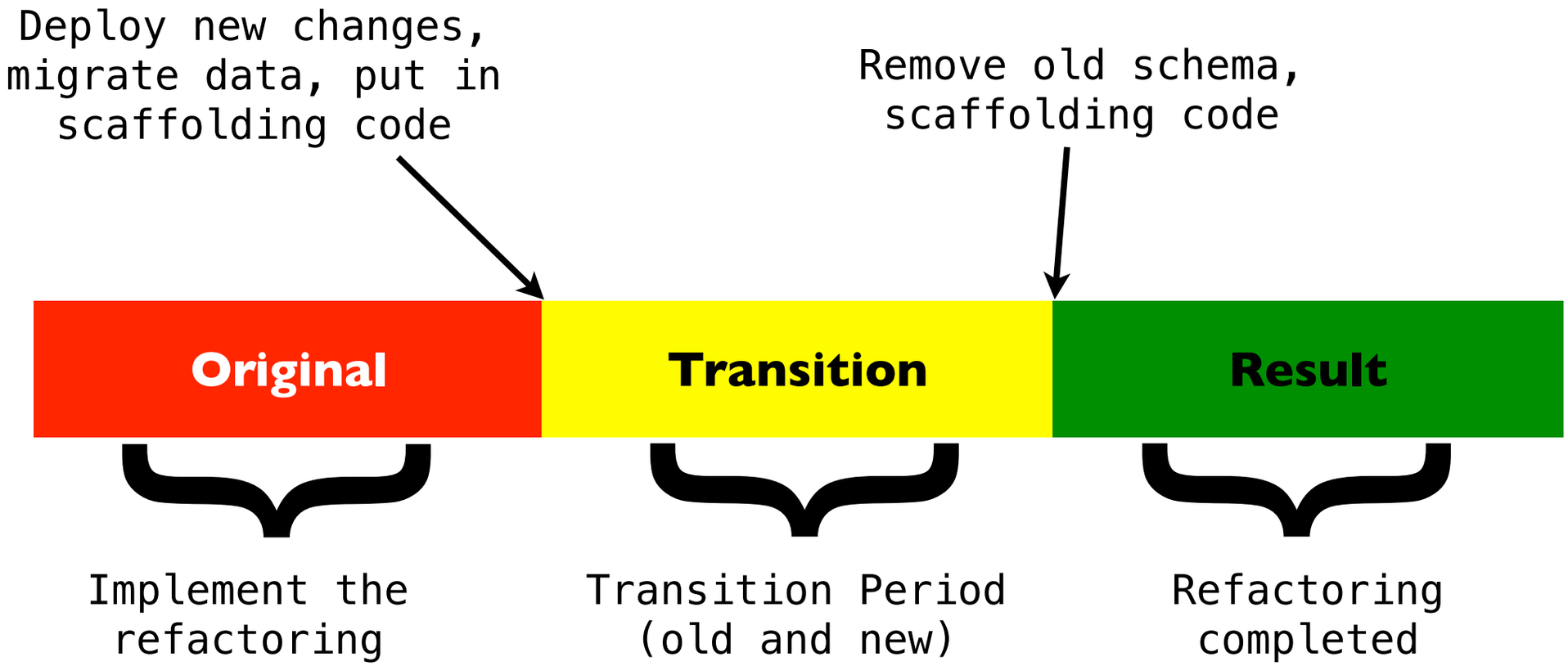
Patterns of Database Changes

- Architecture
- Structure
- Data Quality
- Referential Integrity

Patterns of Database Changes

- Architecture
- Structure
- Data Quality
- Referential Integrity
- Database Code

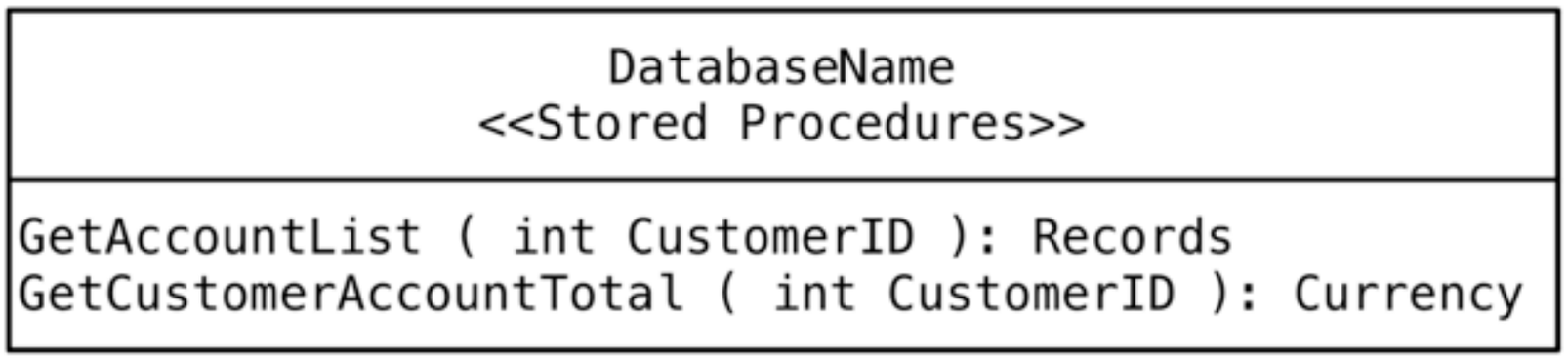
Timeline of Change



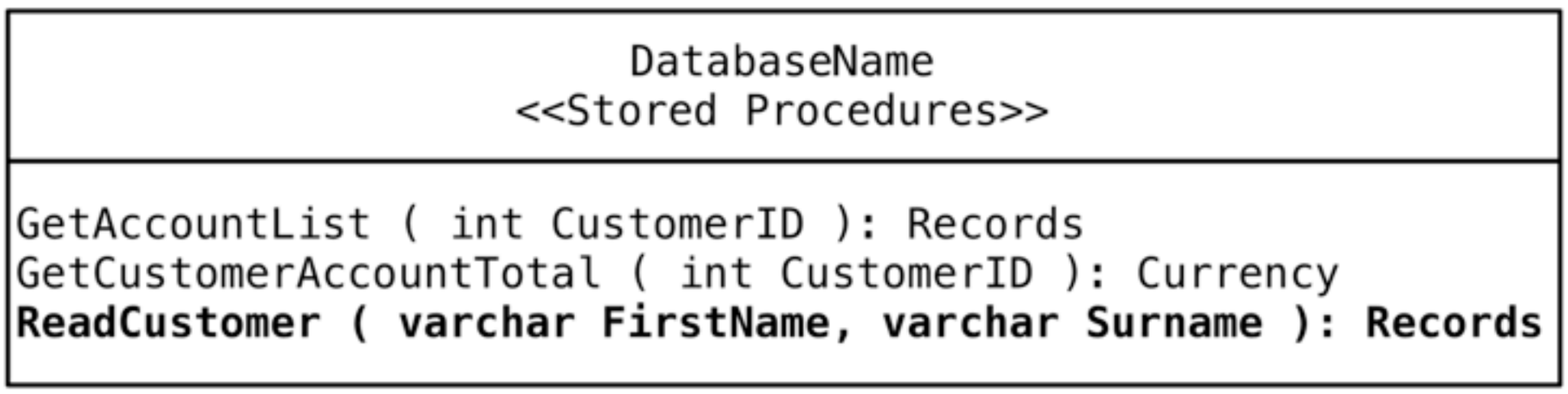
Architecture Patterns

Changes that improve the over all manner in which external programs interact with the database

Add Read Method

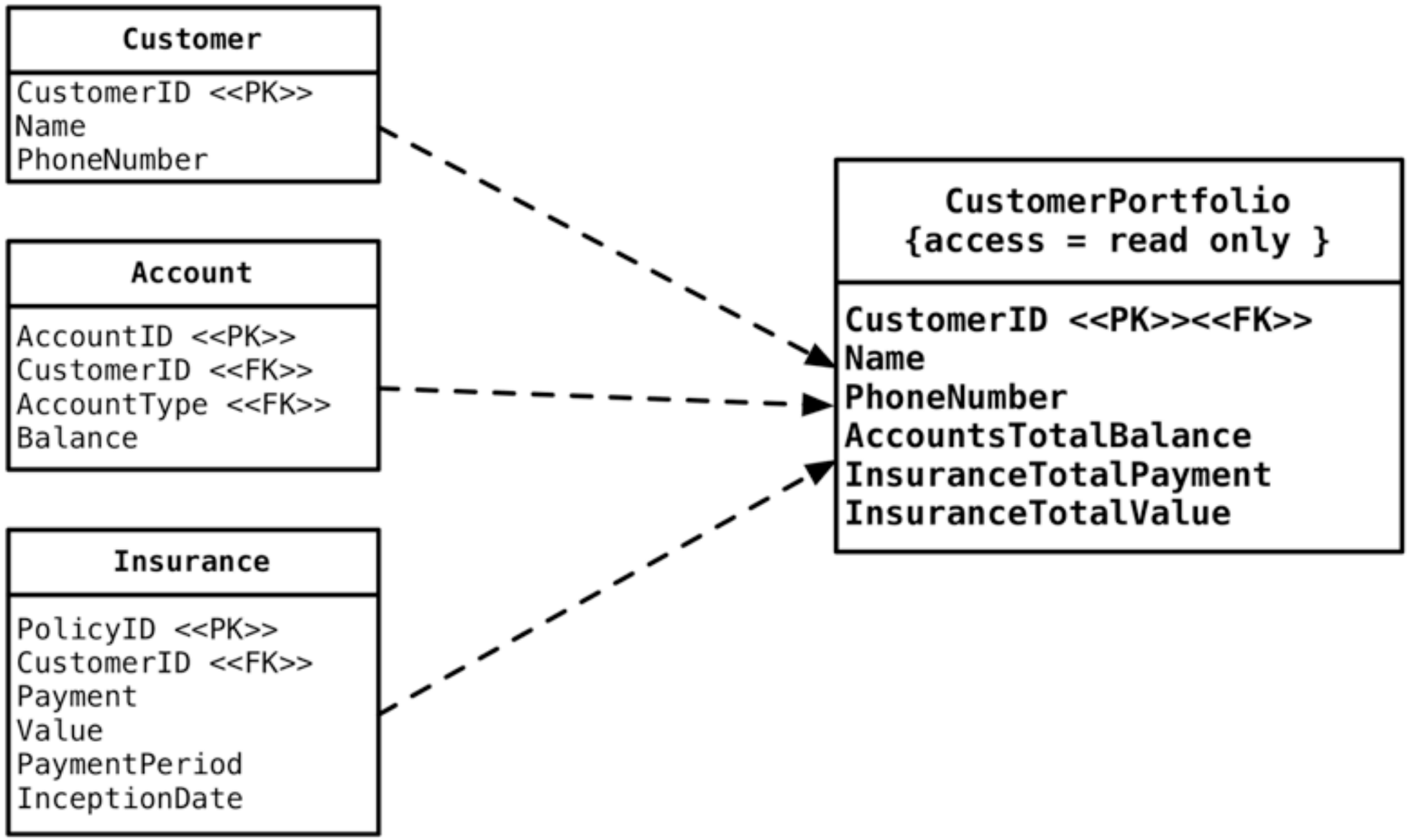


Original Schema

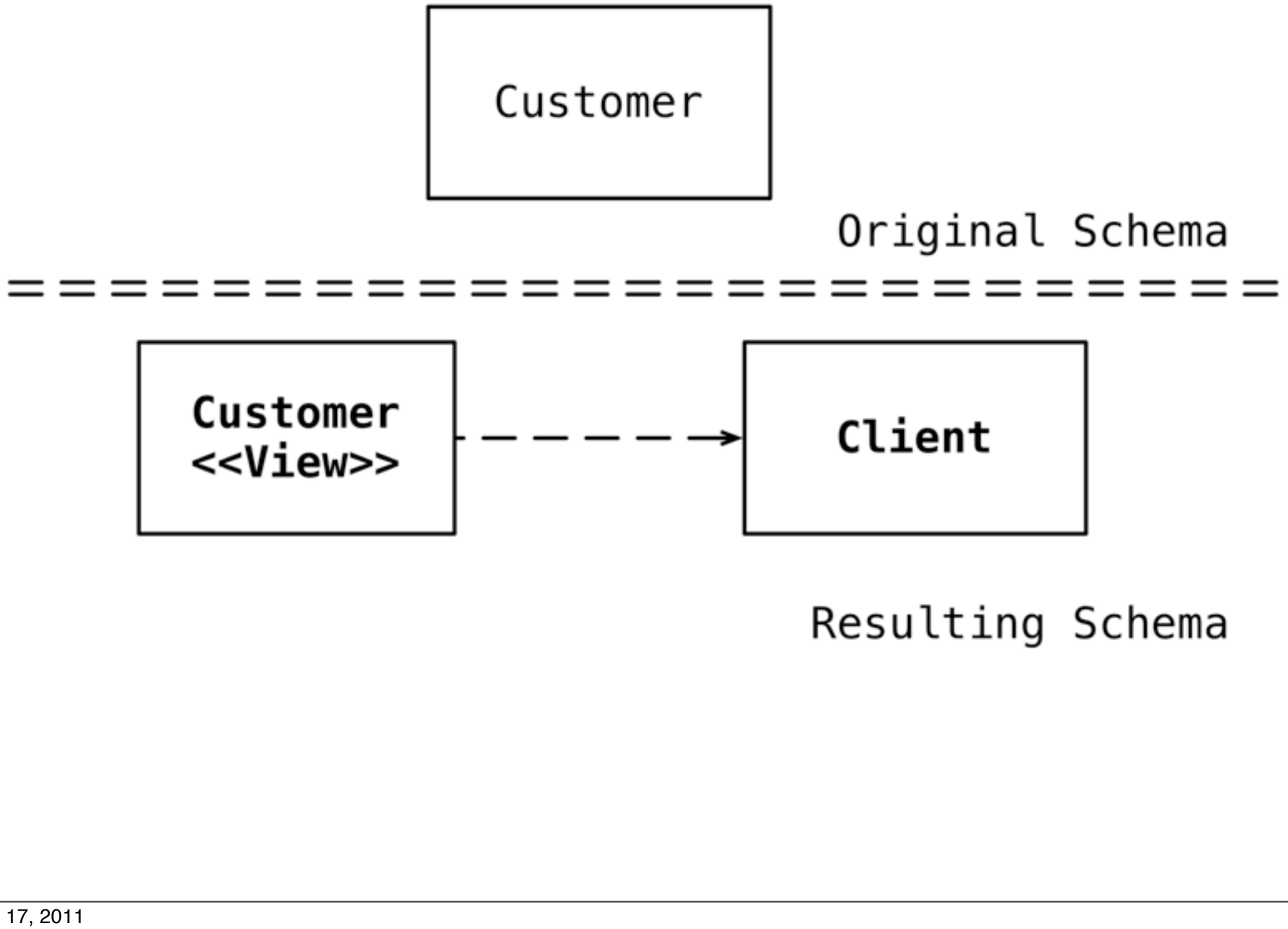


Resulting Schema

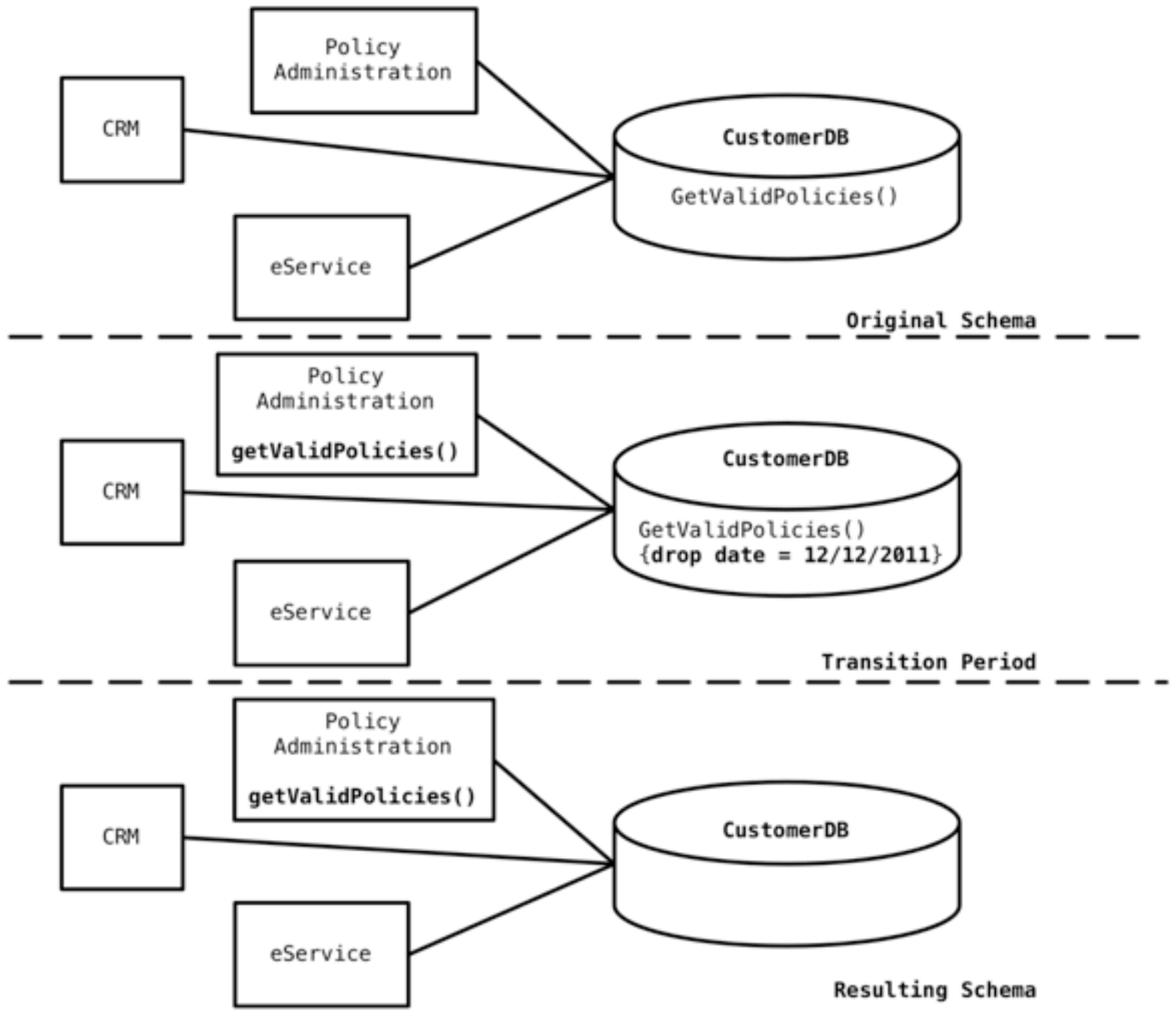
Introduce Read Only Table



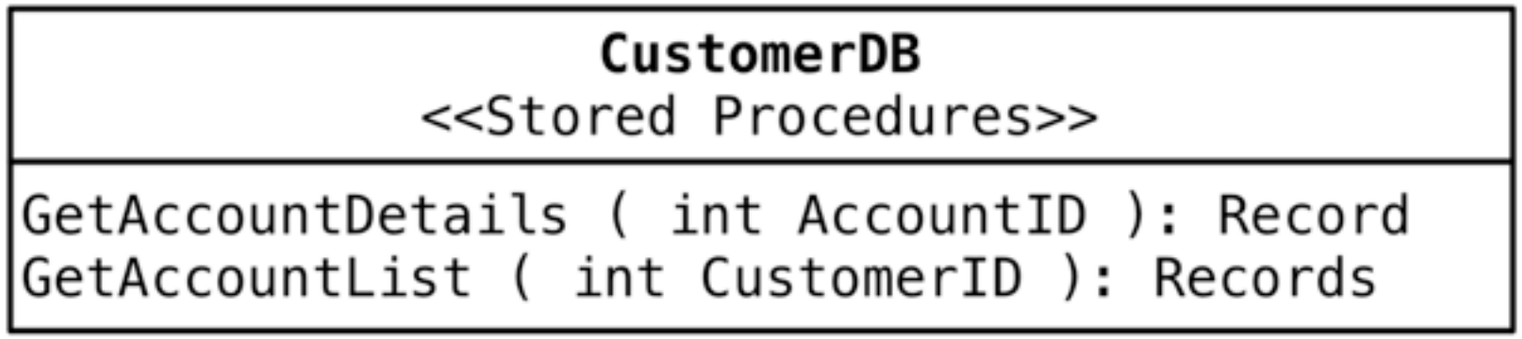
Encapsulate Table with View



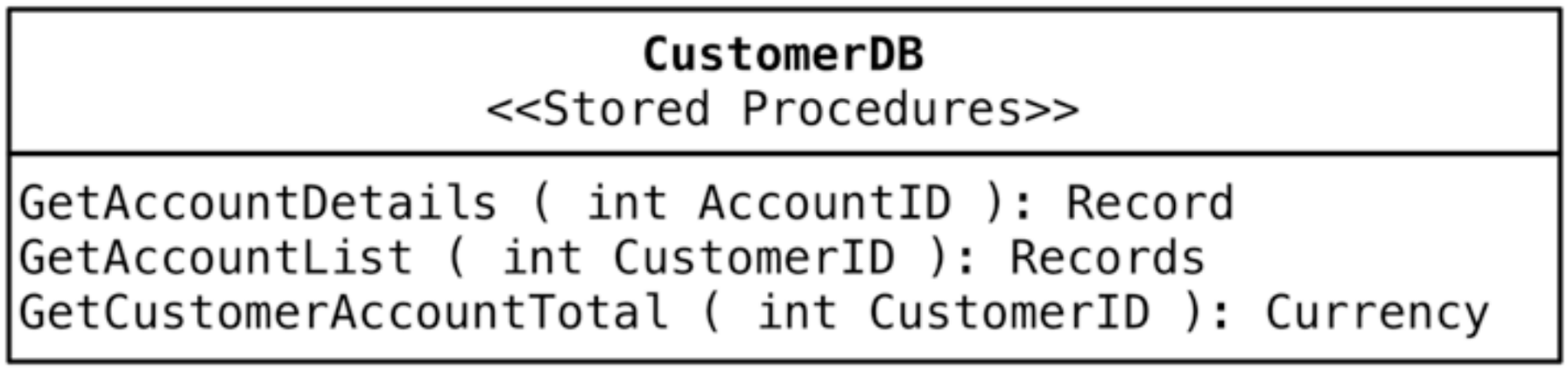
Migrate Method from Database



Introduce Calculation Method

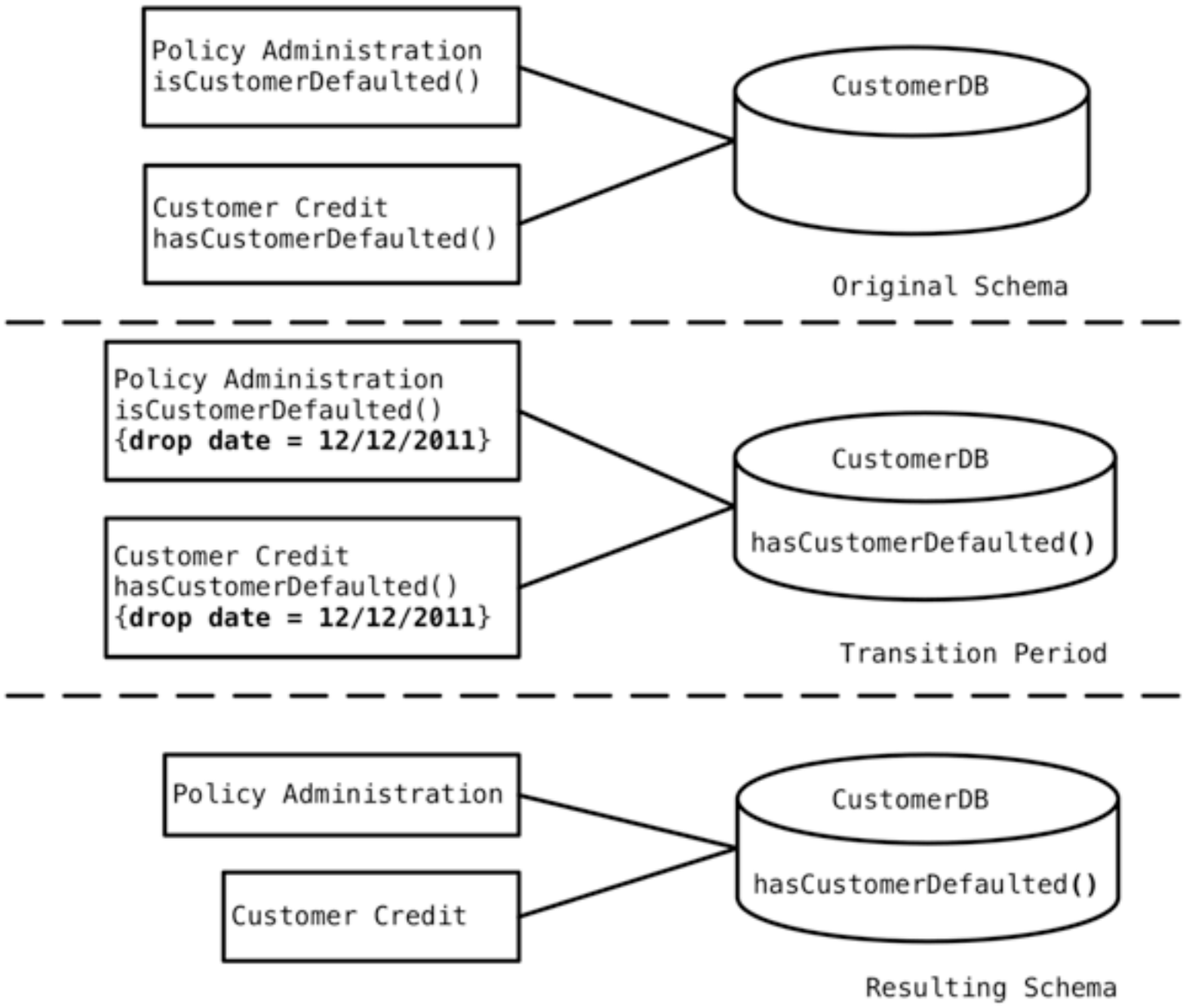


Original Schema



Resulting Schema

Migrate Method to Database



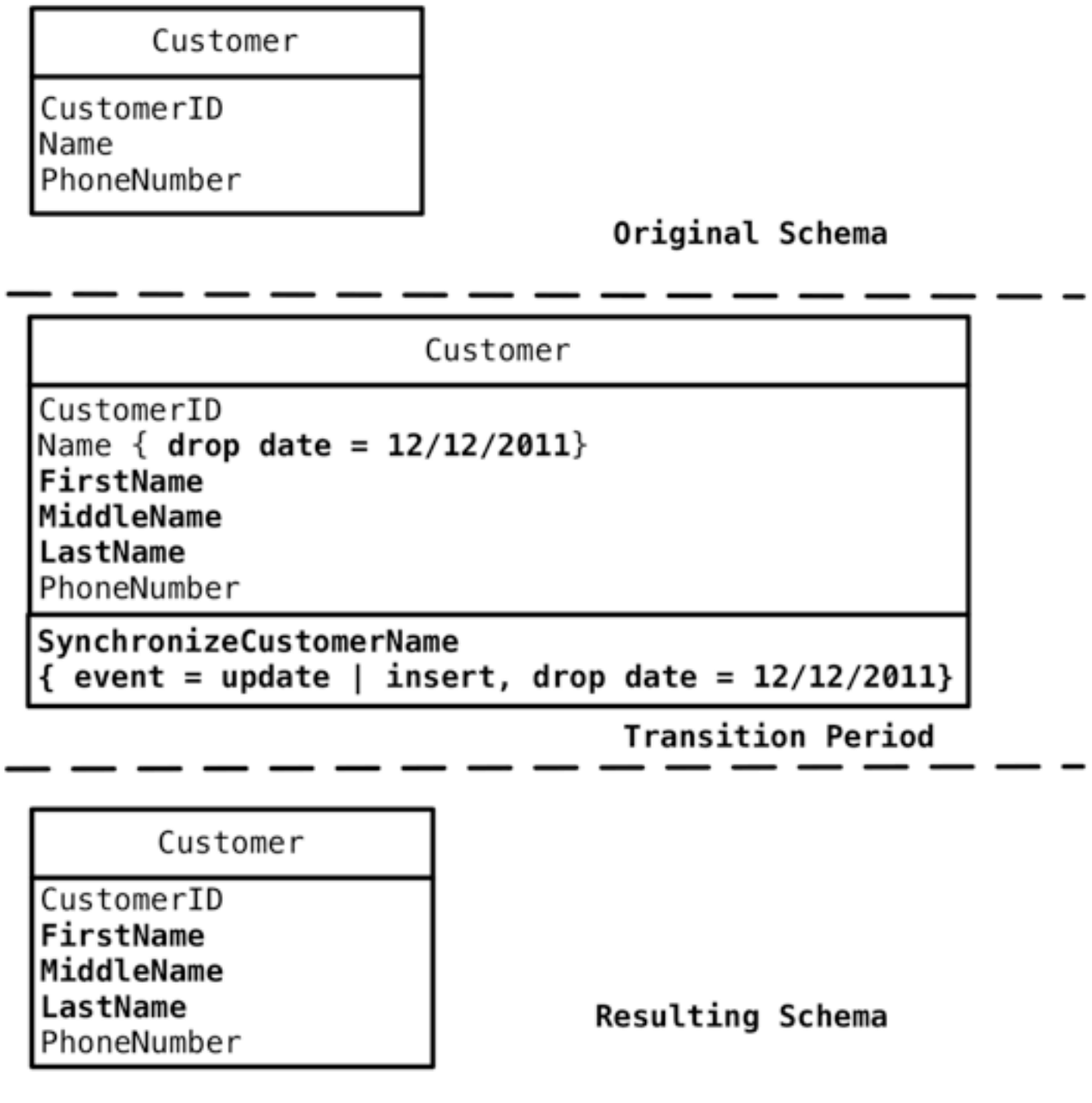
more at

http://databaserefactoring.com

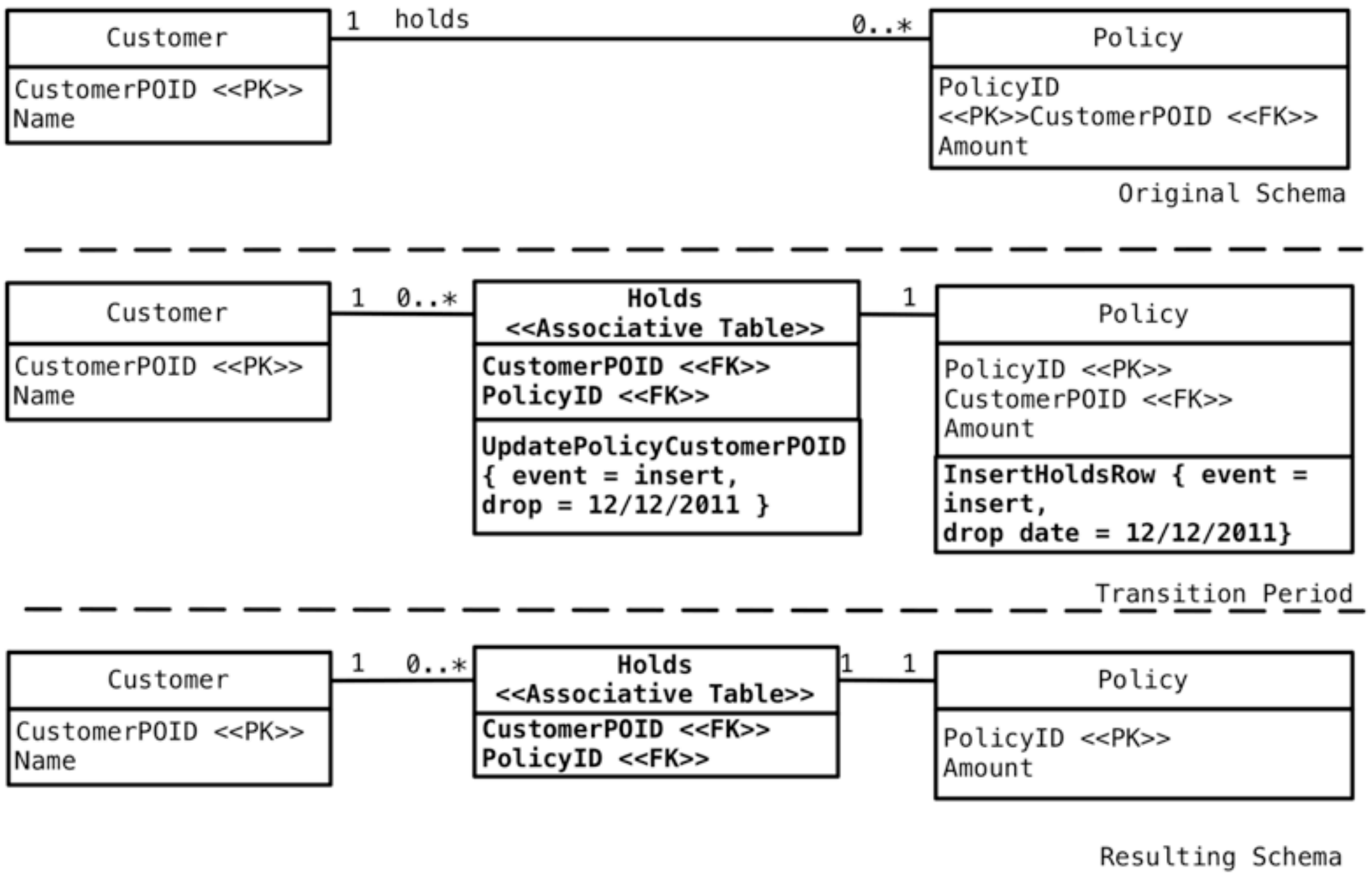
Structural Change Patterns

Change the structure of the database schema, for better database design

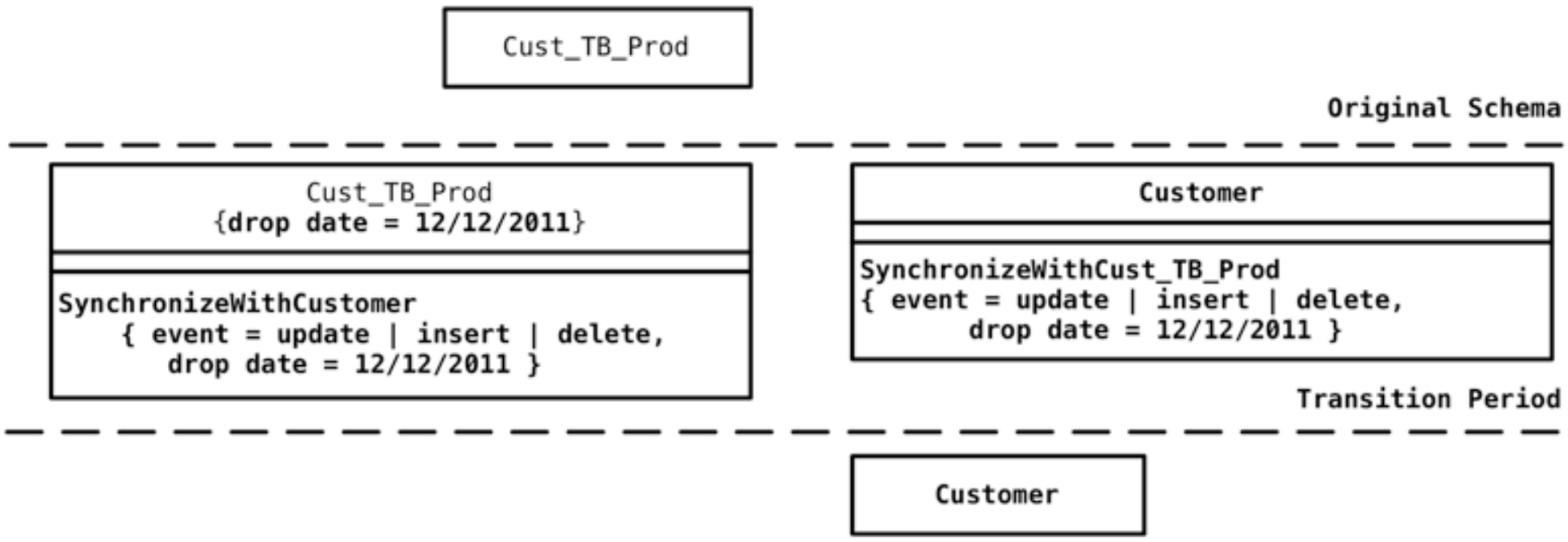
Split Column



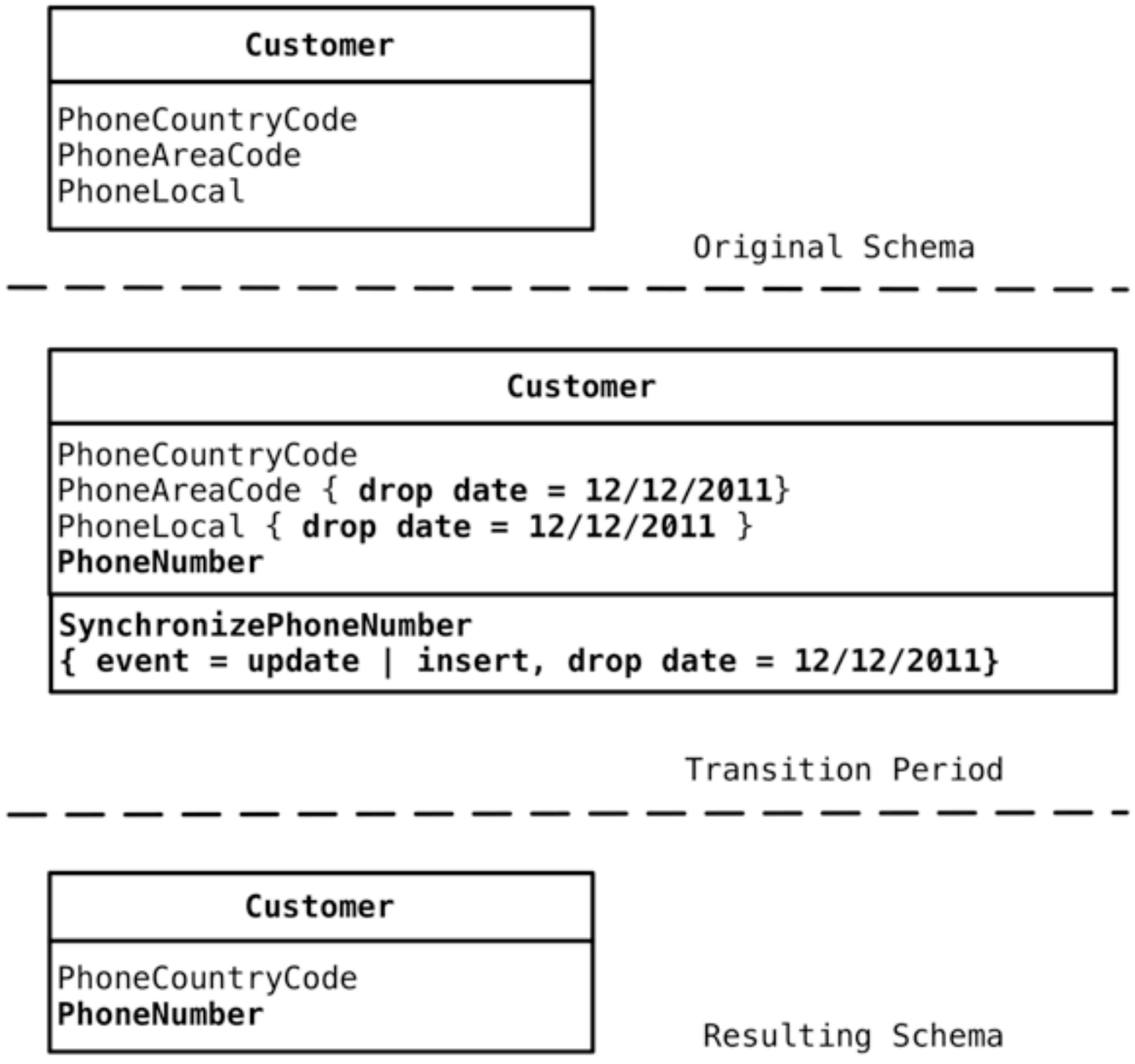
Replace One-To-Many with Associative Table



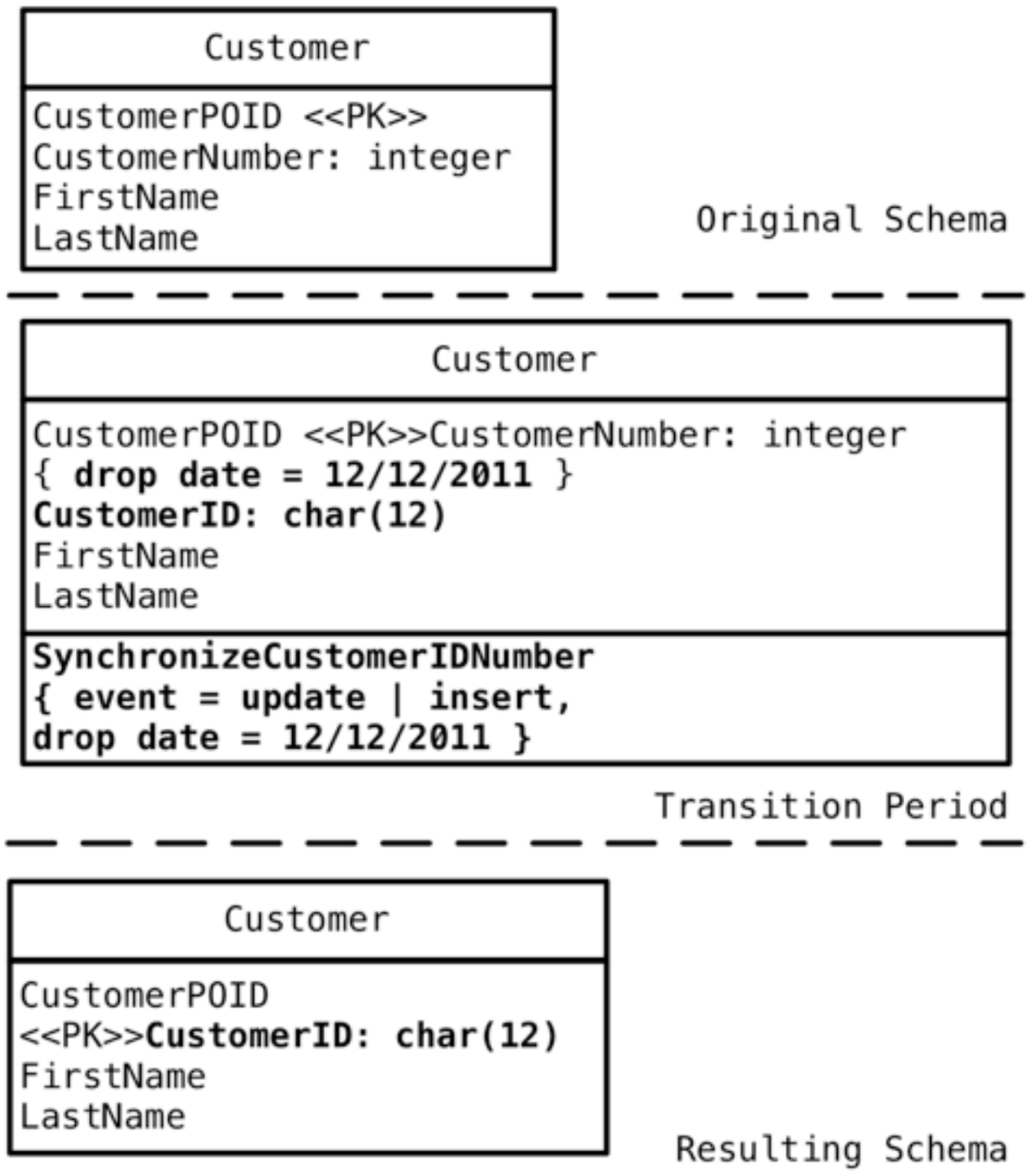
Rename Table



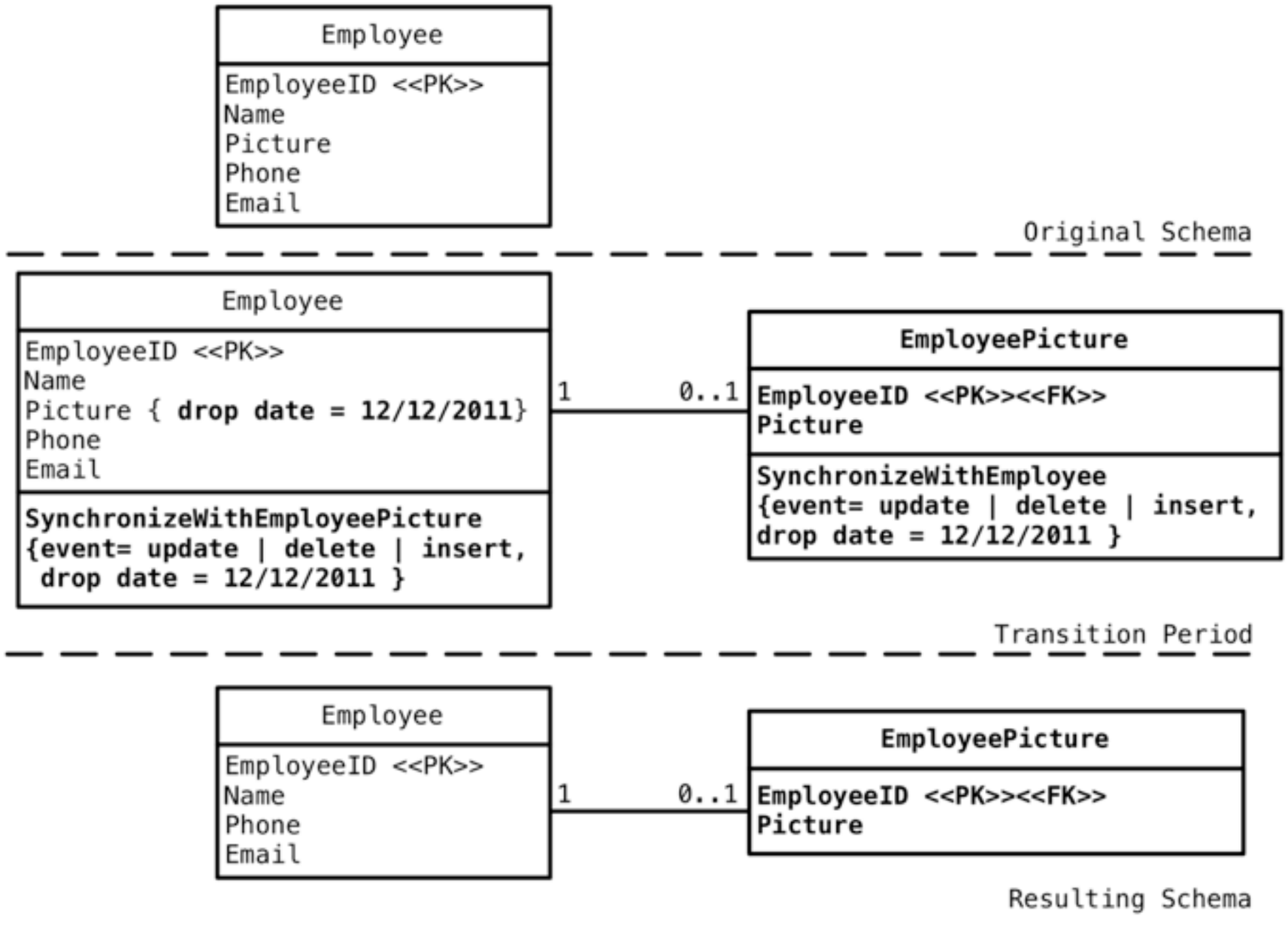
Merge Columns



Replace Column



Split Table


Employee
EmployeeID <<PK>>
Name
Picture
Phone
Email
Original Schema
Employee
EmployeeID <<PK>>
Name
Picture { drop date = 12/12/2011}
Phone
Email
SynchronizeWithEmployeePicture
{event= update | delete | insert,
drop date = 12/12/2011 }
1
0..1
EmployeePicture
EmployeeID <<PK>><<FK>>
Picture
SynchronizeWithEmployee
{event= update | delete | insert,
drop date = 12/12/2011 }
Transition Period
Employee
EmployeeID <<PK>>
Name
Phone
Email
1
0..1
EmployeePicture
EmployeeID <<PK>><<FK>>
Picture
Resulting Schema

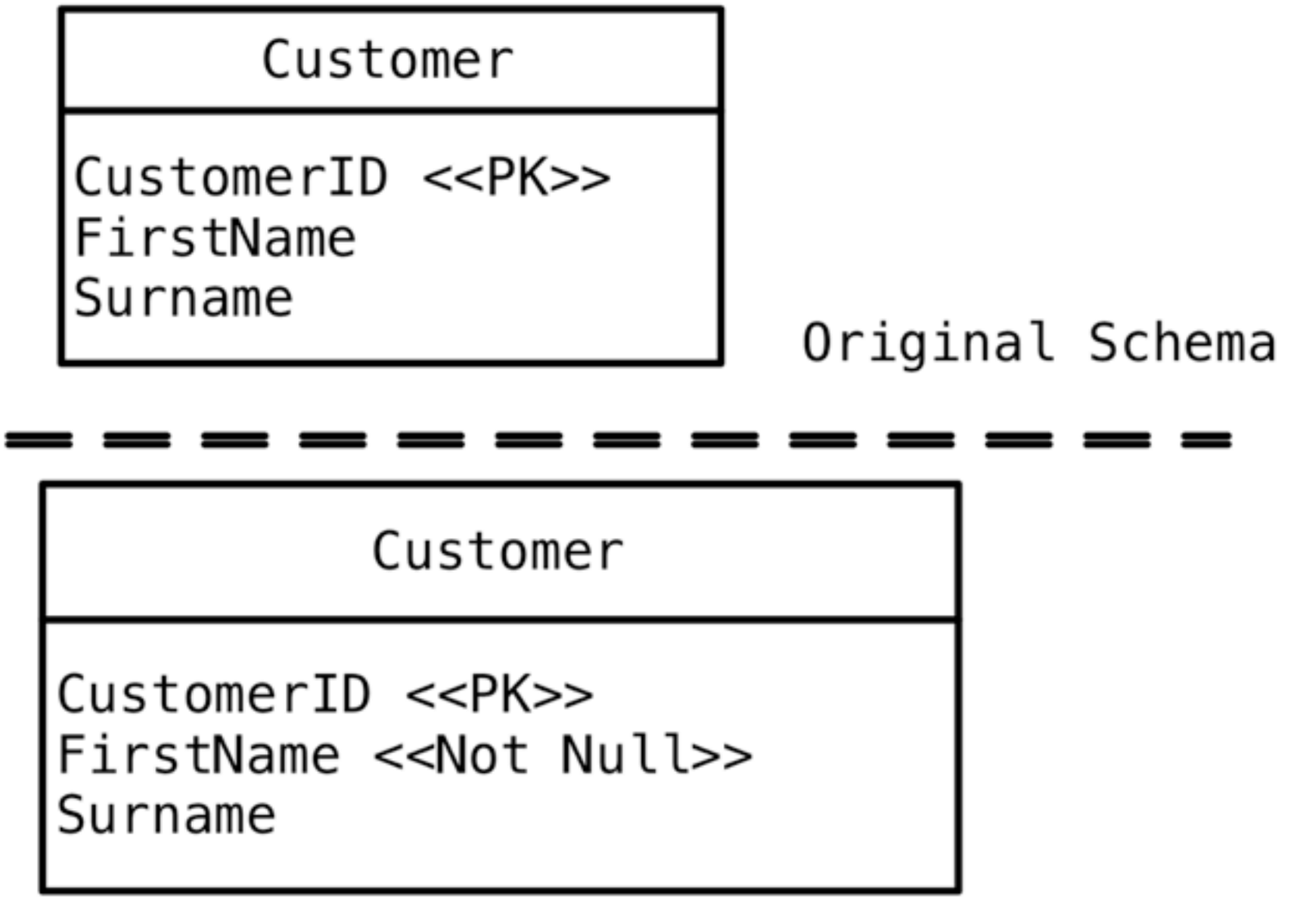
more at

http://databaserefactoring.com

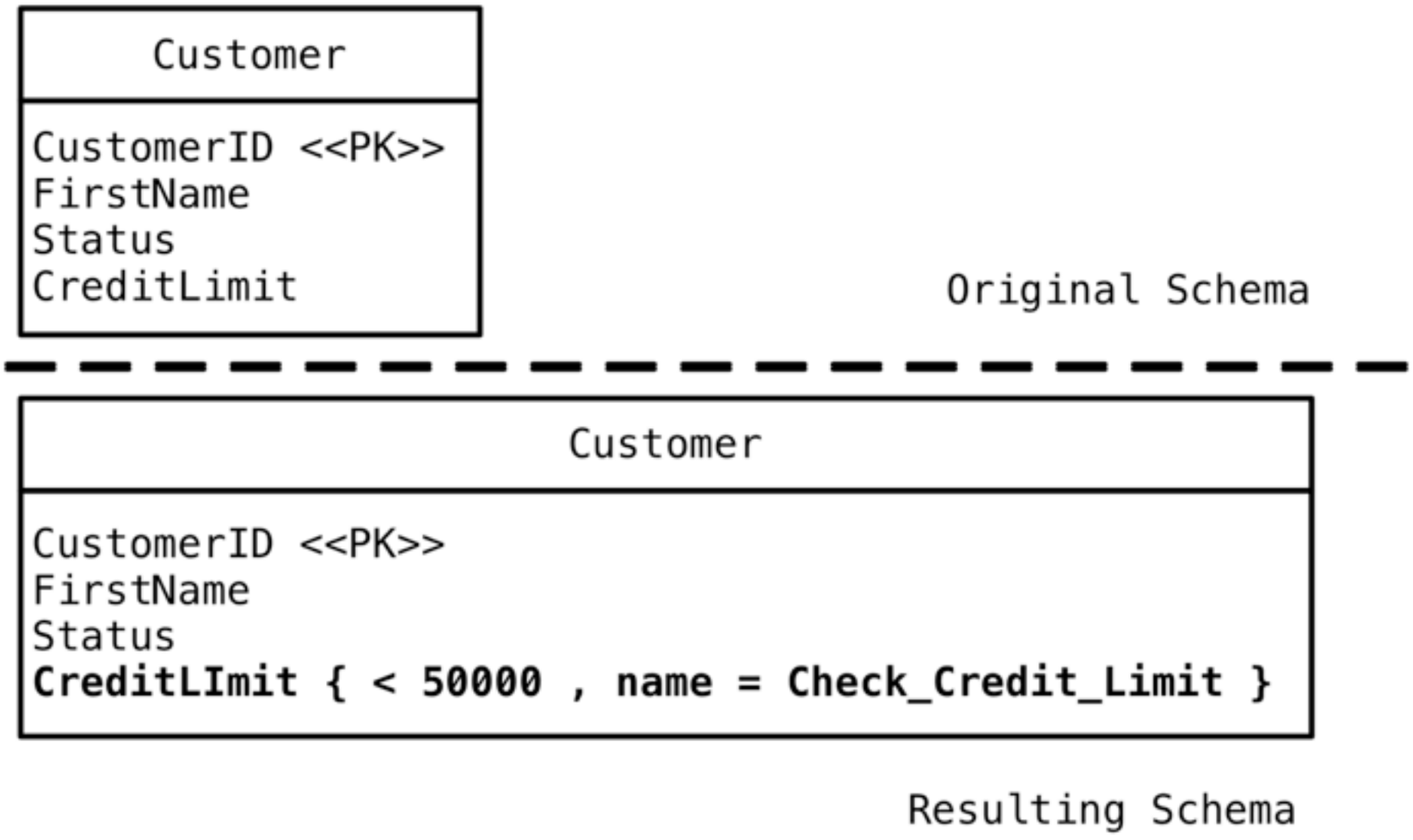
Data Quality Patterns

Changes that improve the quality of the information with in a database or ensure the consistency and usage of data

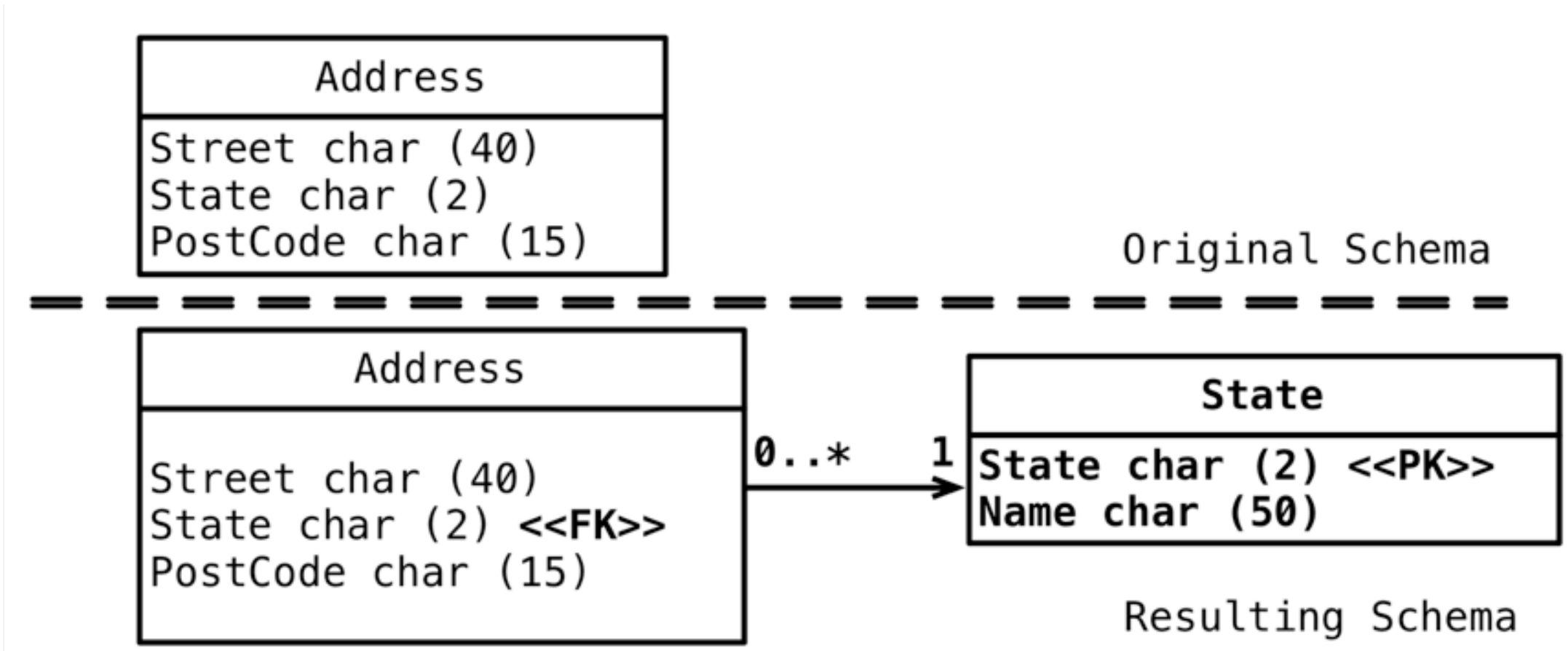
Make Column Non-Nullable



Introduce Column Constraint



Add Lookup Table



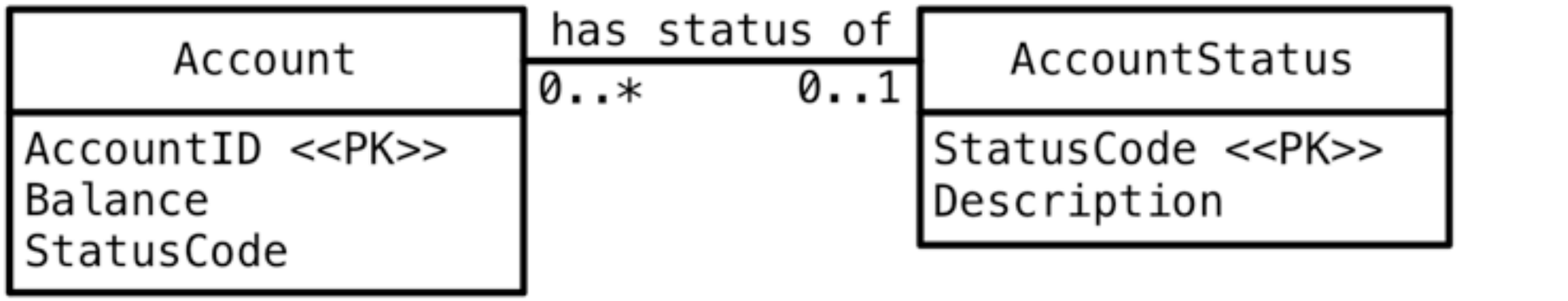
more at

http://databaserefactoring.com

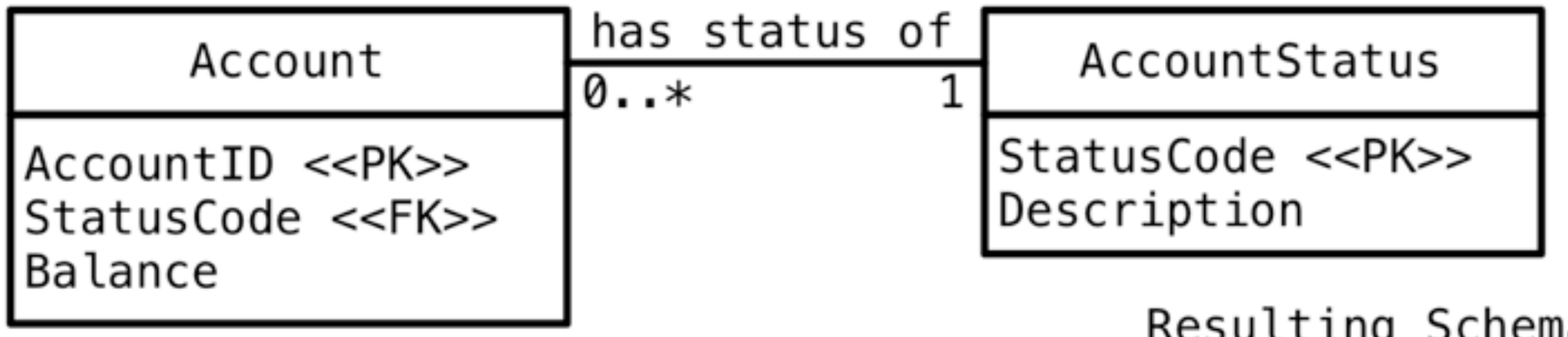
Referential Integrity Patterns

Changes ensure Referential Data is maintained making sure Data Quality is improved

Add Foreign Key Constraint

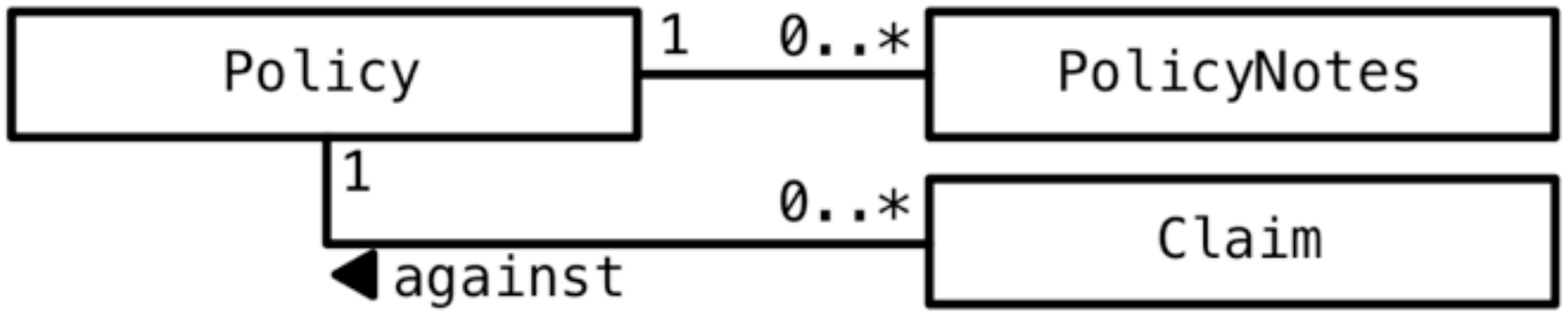


Original Schema

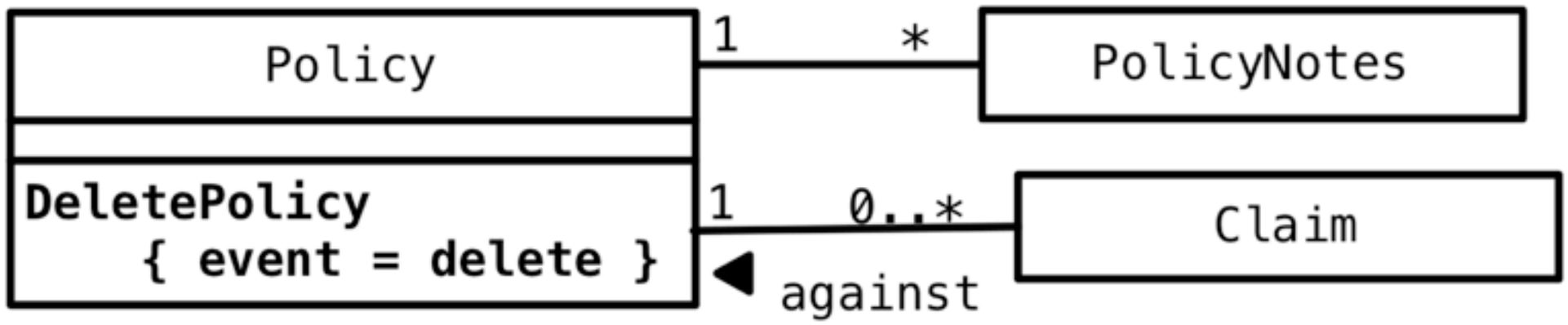


Resulting Schema

Introduce Cascading Delete



Original Schema



Resulting Schema

more at

http://databaserefactoring.com

Database Code

Like code refactoring (Fowler 1999) refactor database code to improve the design of database code i.e Stored Procs and Triggers

Practices

Without good development **Practices**, using and implementing the **Patterns** is going to be difficult.

Configuration Management

Configuration Management

- Allow for common code ownership

Configuration Management

- Allow for common code ownership
- All database artifacts belong in source control repository

Configuration Management

- Allow for common code ownership
- All database artifacts belong in source control repository
- Include setup/config data

Configuration Management

- Allow for common code ownership
- All database artifacts belong in source control repository
- Include setup/config data
- Publish database artifacts with Continuous Integration

Give everyone a database Sandbox

Give everyone a database Sandbox

- Reduce waste and waiting time

Give everyone a database Sandbox

- Reduce waste and waiting time
- Automate data tasks

Give everyone a database Sandbox

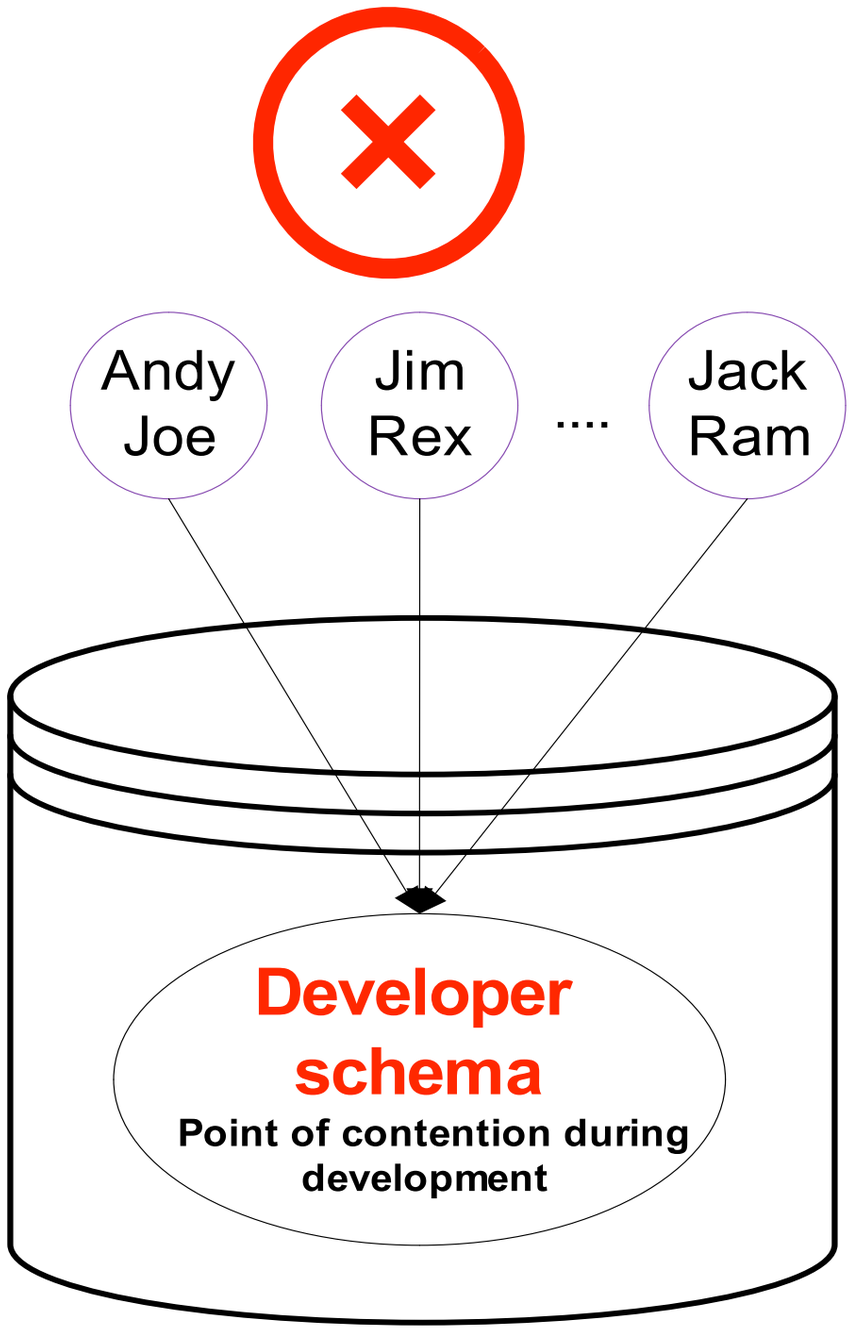
- Reduce waste and waiting time
- Automate data tasks
- Allows local changes and experimenting

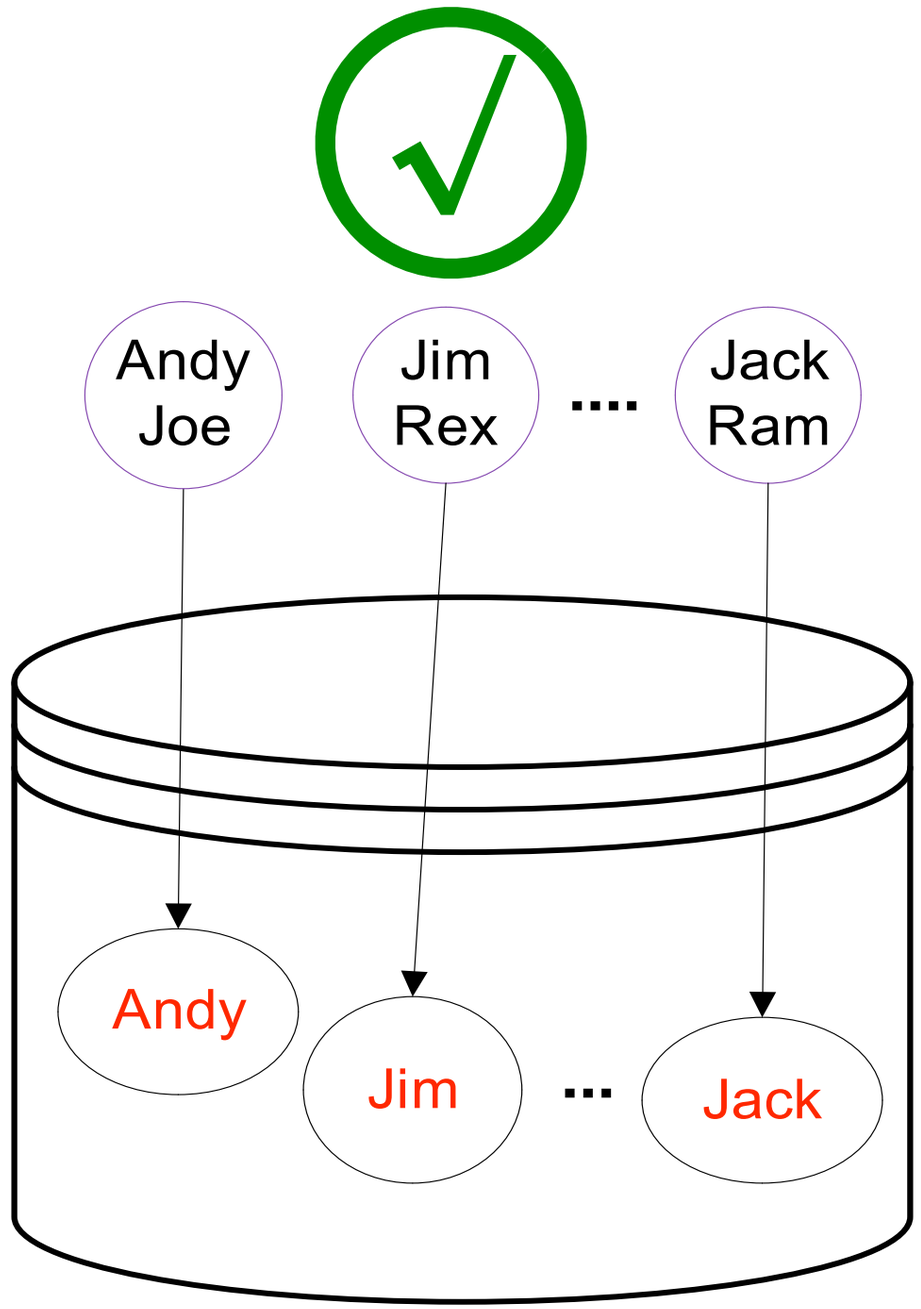
Give everyone a database Sandbox

- Reduce waste and waiting time
- Automate data tasks
- Allows local changes and experimenting
- Improves productivity

Give everyone a database Sandbox

- Reduce waste and waiting time
- Automate data tasks
- Allows local changes and experimenting
- Improves productivity
- Spinning up new environments is easy


Andy
Joe
Jim
Rex
....
Jack
Ram
Developer
schema
Point of contention during
development


Andy
Joe
Jim
Rex
....
Jack
Ram
Andy
Jim
...
Jack

Behavior of Database

Behavior of Database

- Like objects, database has behavior

Behavior of Database

- Like objects, database has behavior
- Develop database objects using BDD style tests

Behavior of Database

- Like objects, database has behavior
- Develop database objects using BDD style tests
- Allows easy changes to the database

Behavior of Database

- Like objects, database has behavior
- Develop database objects using BDD style tests
- Allows easy changes to the database
- Protects against changes that affect dependent functionality

Requirements:
Store a “Vehicle” with

- Unique VIN Number
- Model Year 2005 and above
- Model Year Not Null
- Model Name Not Null
- Make Not Null
- Miles not above 10000

Requirements:
Store a “Vehicle” with

- Unique VIN Number
- Model Year 2005 and above
- Model Year Not Null
- Model Name Not Null
- Make Not Null
- Miles not above 10000

Requirements:
Store a “Vehicle” with

- Unique VIN Number
- Model Year 2005 and above
- Model Year Not Null
- Model Name Not Null
- Make Not Null
- Miles not above 10000

Behavior Tests:

- shouldNotSaveDuplicateVIN()
- shouldSaveModelYear2010()
- shouldNotSaveModelYear2004()
- shouldNotSaveNullModelName()
- shouldNotSaveNullMake()
- shouldSaveMiles5000()
- shouldNotSaveMiles12000()

```
CREATE TABLE vehicle(
   id NUMBER(18) NOT NULL,
   vin VARCHAR2(32) NOT NULL,
   name VARCHAR2(32) NOT NULL,
   make VARCHAR2(32) NOT NULL,
   year NUMBER(4) NOT NULL,
   miles NUMBER(10) NULL,
CONSTRAINT chk_vehicle_year_gt_2005
                  CHECK (year> 2004));
CONSTRAINT chk_vehicle_miles_lt_10001
                  CHECK (miles< 10001));
CREATE UNIQUE INDEX uidx_vehicle_vin
                  ON  vehicle(vin);

ALTER TABLE VEHICLE ADD CONSTRAINT
            pk_vehicle PRIMARY KEY (id);
```

Tracking Changes

Tracking Changes

- Write each change as a delta (migration) script

Tracking Changes

- Write each change as a delta (migration) script
- Migration scripts are development time activity not deployment time project

Tracking Changes

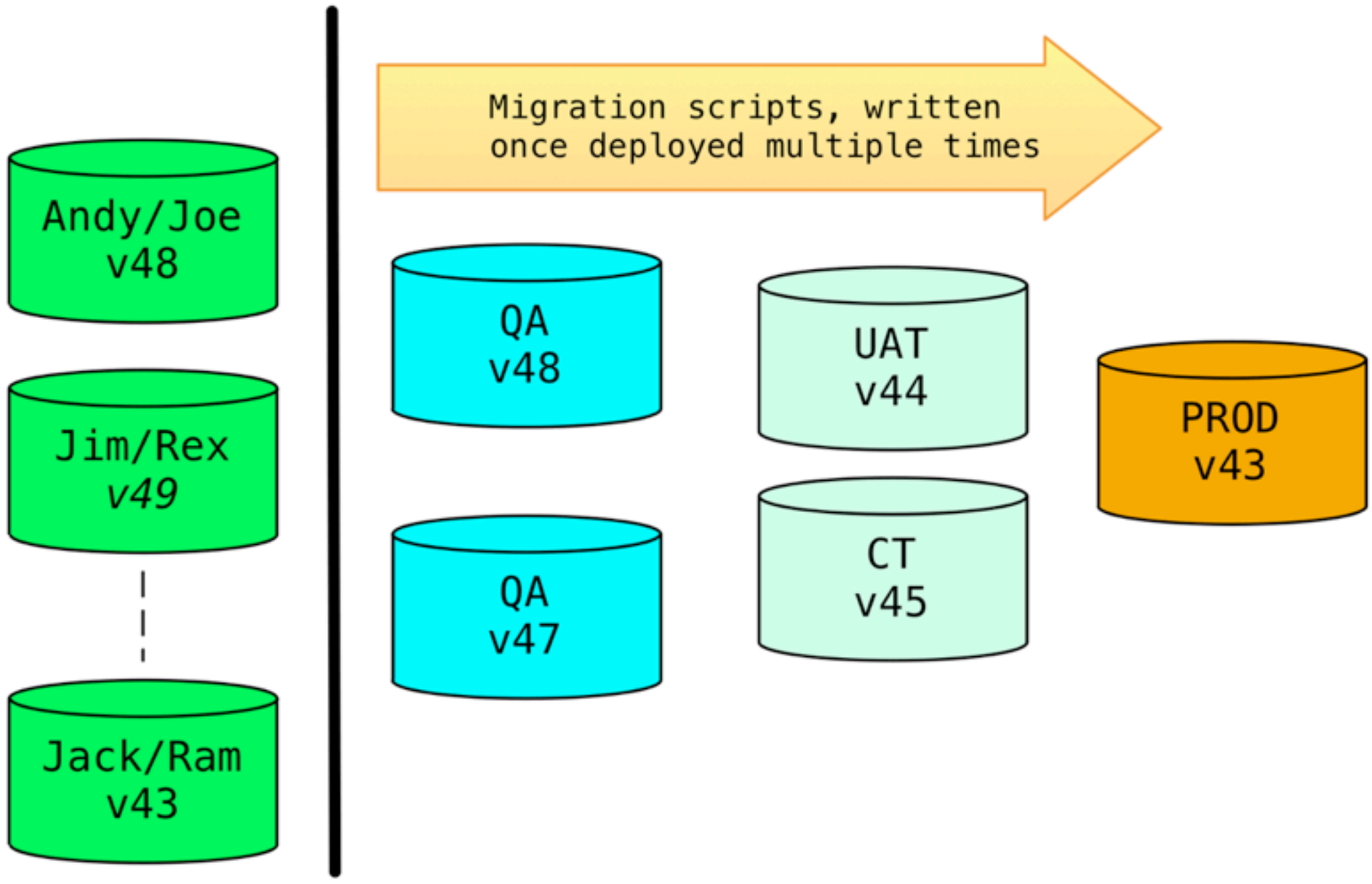
- Write each change as a delta (migration) script
- Migration scripts are development time activity not deployment time project
- Package delta scripts for automated deployment

Tracking Changes

- Write each change as a delta (migration) script
- Migration scripts are development time activity not deployment time project
- Package delta scripts for automated deployment
- Same scripts for: developers, QA, UAT and Production

```
ALTER TABLE customer ADD customeridentifier VARCHAR2(12);

UPDATE customer SET customeridentifier = customernumber;

--If No Transistion Period

ALTER TABLE customer DROP COLUMN customerNumber;

--//@UNDO

ALTER TABLE customer ADD customernumber NUMBER(10);

UPDATE customer SET customernumber = customeridentifier;

ALTER TABLE customer DROP COLUMN customeridentifier;
```

Deployed Database Version



Continuous Integration

Continuous Integration

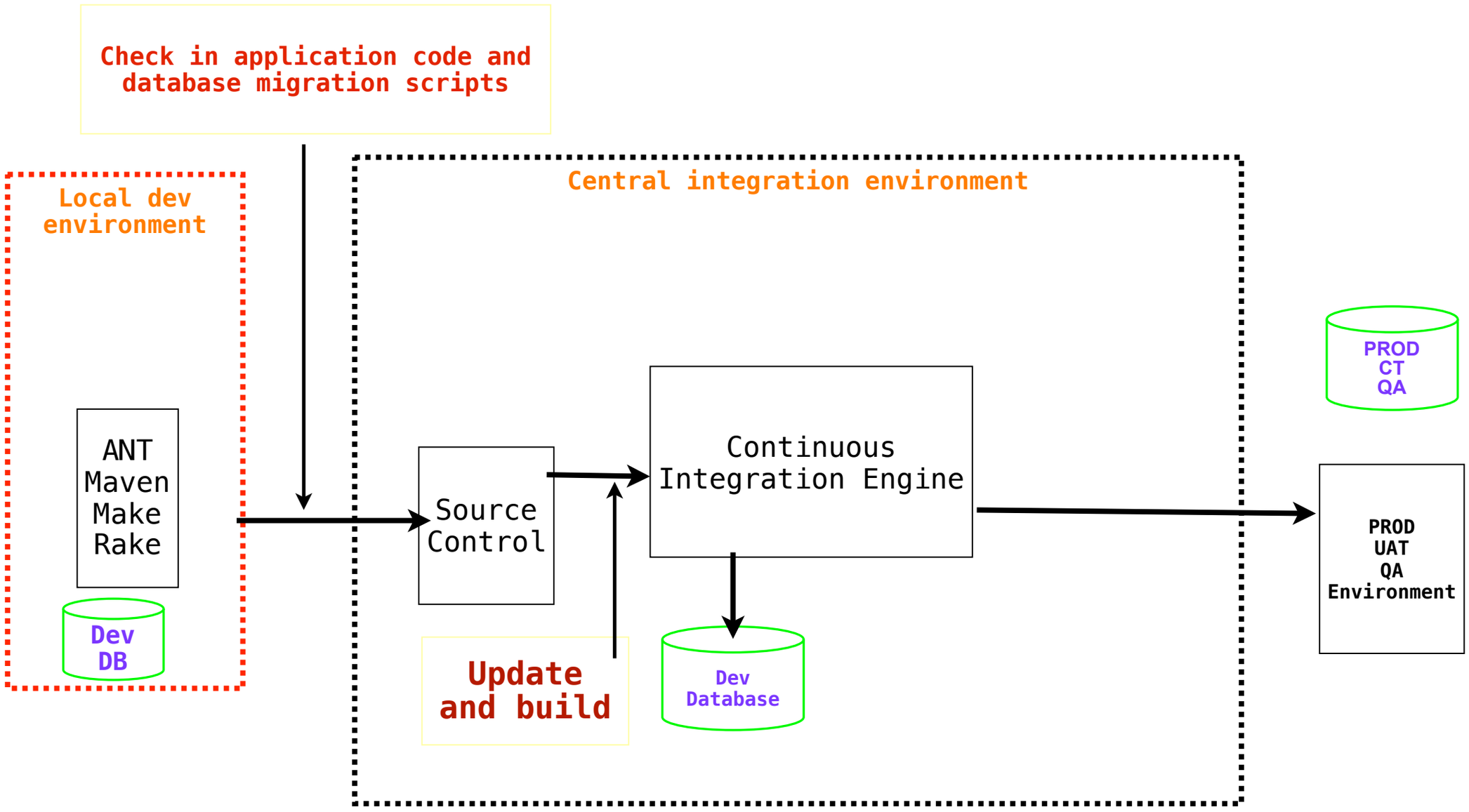
- Test application code and database at one place

Continuous Integration

- Test application code and database at one place
- Generate code and database artifacts

Continuous Integration

- Test application code and database at one place
- Generate code and database artifacts
- Integrate application and database changes in an independent environment


Check in application code and database migration scripts
Local dev environment
ANT
Maven
Make
Rake
Dev DB
Central integration environment
Source Control
Continuous Integration Engine
Update and build
Dev Database
PROD
CT
QA
PROD
UAT
QA
Environment

Check in application code and database migration scripts

Local dev environment

Migration scripts

ANT
Maven
Make
Rake

Dev DB

Central integration environment

Source Control

Continuous Integration Engine

Update and build

PROD
CT
QA

PROD
UAT
QA
Environment

Check in application code and database migration scripts

Local dev environment

Migration scripts

ANT
Maven
Make
Rake

Dev DB

Central integration environment

Migration scripts

Source Control

Continuous Integration Engine

Update and build

Integration Database

PROD
CT
QA

PROD
UAT
QA
Environment

Check in application code and database migration scripts

Local dev environment

Migration scripts

ANT
Maven
Make
Rake

Dev DB

Central integration environment

Migration scripts

Source Control

Continuous Integration Engine

Update and build

Integration Database

PROD
CT
QA

PROD
UAT
QA
Environment

Check in application code and database migration scripts

Local dev environment

Migration scripts

ANT
Maven
Make
Rake

Dev DB

Central integration environment

Migration scripts

Source Control

Continuous Integration Engine

Update and build

Integration Database

Apply migration scripts

PROD
CT
QA

PROD
UAT
QA
Environment

Check in application code and database migration scripts

Local dev environment

Migration scripts

ANT
Maven
Make
Rake

Dev DB

Central integration environment

Migration scripts

Source Control

Continuous Integration Engine

Update and build

Integration Database

Apply migration scripts

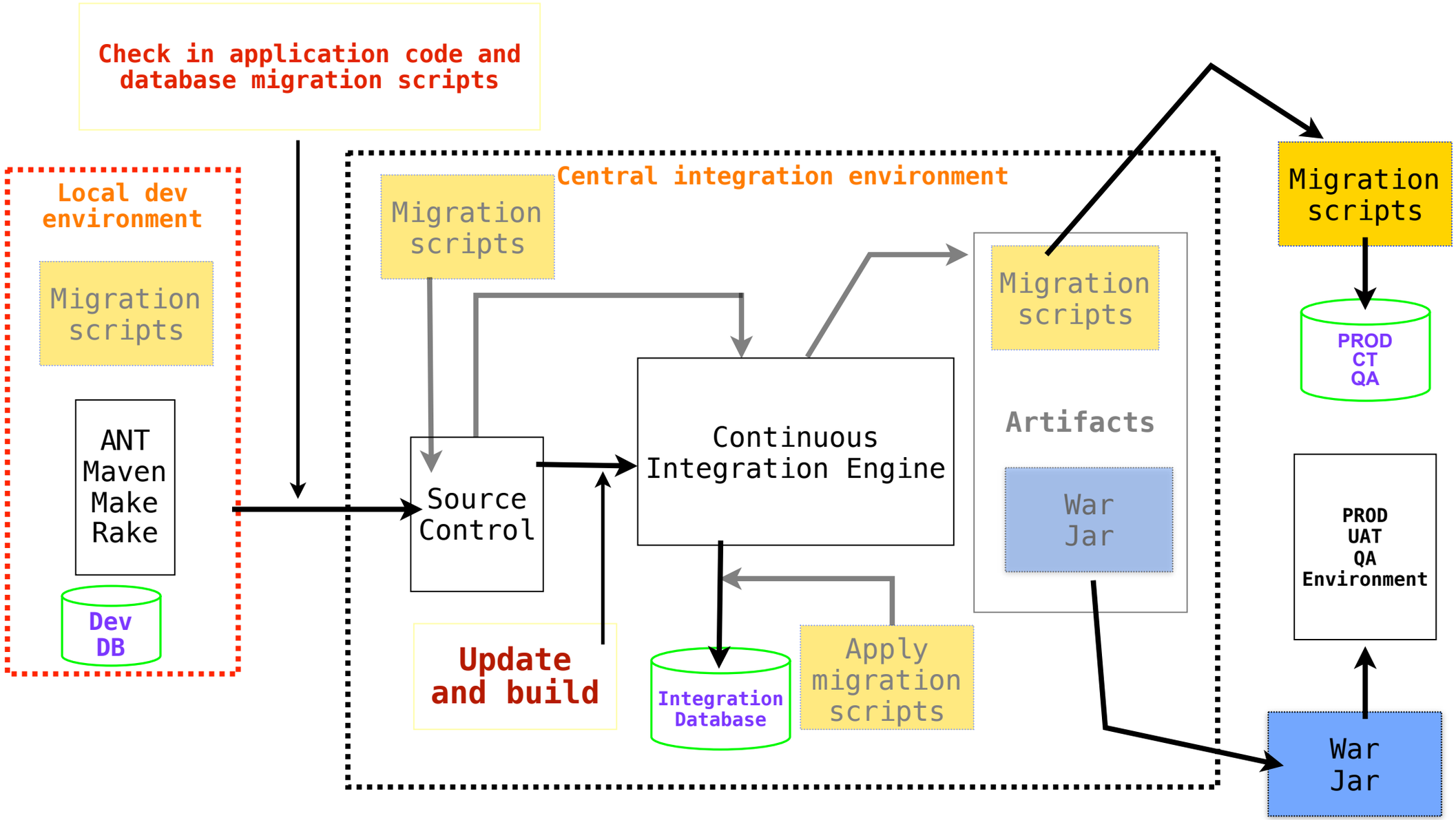
Migration scripts

Artifacts

War
Jar

PROD
CT
QA

PROD
UAT
QA
Environment


Check in application code and database migration scripts
Local dev environment
Migration scripts
ANT
Maven
Make
Rake
Dev DB
Central integration environment
Migration scripts
Source Control
Continuous Integration Engine
Update and build
Integration Database
Apply migration scripts
Migration scripts
Artifacts
War
Jar
Migration scripts
PROD
CT
QA
PROD
UAT
QA
Environment
War
Jar

Deployment

Deployment

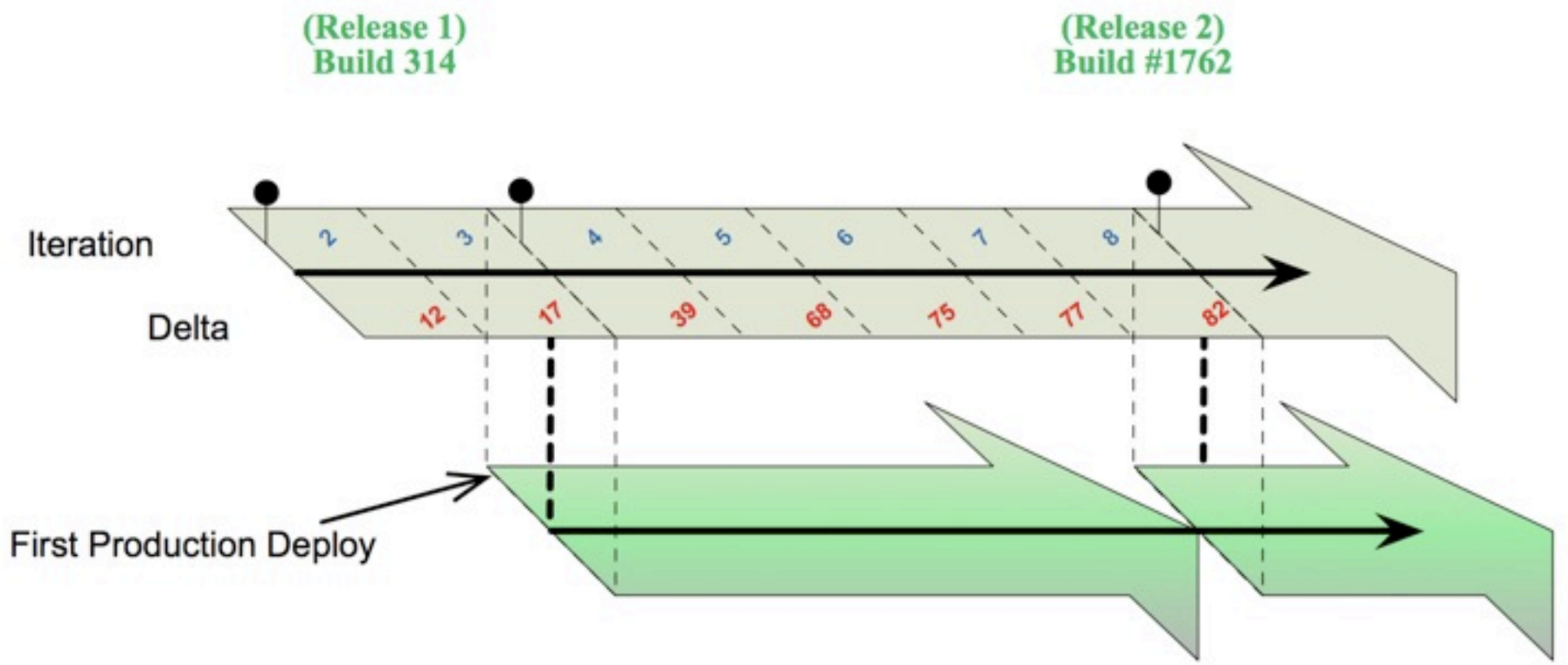
- Database migration/upgrade should be a development time task not deployment time task

Deployment

- Database migration/upgrade should be a development time task not deployment time task
- Package all the migration scripts, during Continuous Integration cycle

Deployment

- Database migration/upgrade should be a development time task not deployment time task
- Package all the migration scripts, during Continuous Integration cycle
- Apply these migration scripts


(Release 1)
Build 314
(Release 2)
Build #1762
Iteration
Delta
2
3
4
5
6
7
8
12
17
39
68
75
77
82
First Production Deploy

Pair with the Data Team

Pair with the Data Team

- Break down the silos

Pair with the Data Team

- Break down the silos
- Allows continuous reviews

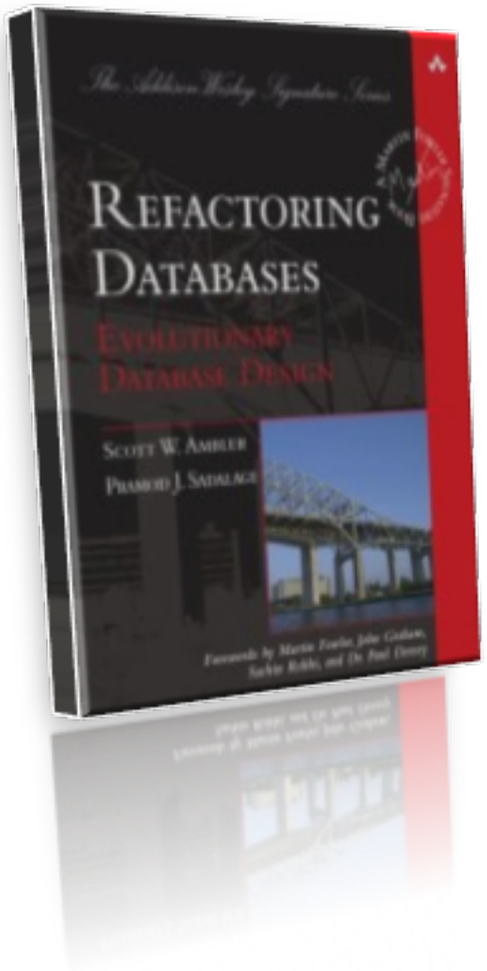
Pair with the Data Team

- Break down the silos
- Allows continuous reviews
- Understand performance implications early

Pair with the Data Team

- Break down the silos
- Allows continuous reviews
- Understand performance implications early
- Put database code and application code in same repository

Resources



bit.ly/qconbddd
bit.ly/evolvedb

Thanks

@pramodsadalage

www.sadalage.com

www.databaserefactoring.com