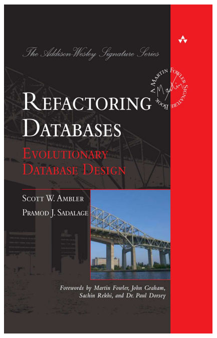

The Addison-Wesley Signature Series
Refactoring Databases
Evolutionary Database Design
Scott W. Ambler
Pramod J. Sadalage
Forewords by Martin Fowler, John Graham, Sachin Rekhi, and Dr. Paul Dorsey

Refactoring Databases: Evolutionary Database Design

Pramod Sadalage (沙朴木)
ThoughtWorks Inc.

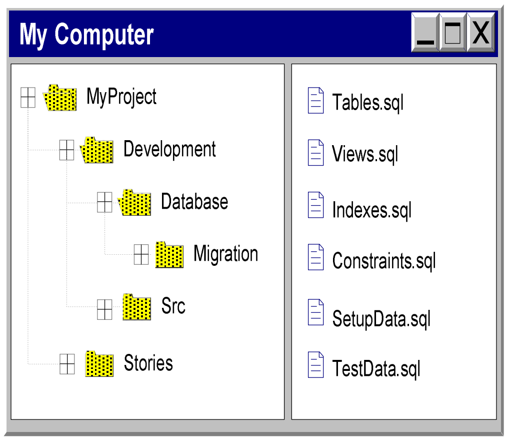
ThoughtWorks

Evolutionary Database Development

- Reality is “Businesses are Changing”
- Change management for database assets
- Learn when you fail, so make it easy to fail
- Functionality added in increments
- Facilitates automated testing
- Knowledge of the functionality
- Acknowledge team interaction
- DBA = Role != Person

Version Control Database Assets

- Greater control over changes
- Couples database and application
- Integrate in version control instead of database



Swap Best Practices

- Educate DBA about coding practices
- Nothing is used only once
- Educate Developers write better SQL
- How to make the DBA redundant
- Automate tasks such as
  - Physical table deployment
  - Usage statistics
  - Schema verification
  - Data migration verification

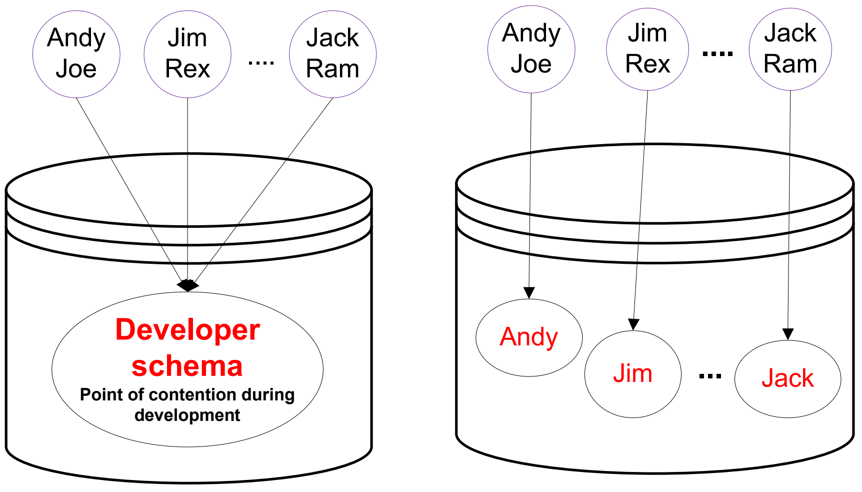
ThoughtWorks

Let the DBA and Developers pair

- Better to have “Generalizing Specialists”
- Helps DBA understand the application
- DBA has a better understanding of other areas of the business data
- Write database tests
- Migration of production data is critical
- Gain knowledge of SQL Tuning etc
- Make the team aware of production data
- Understand performance implications

ThoughtWorks

Give everyone a sandbox



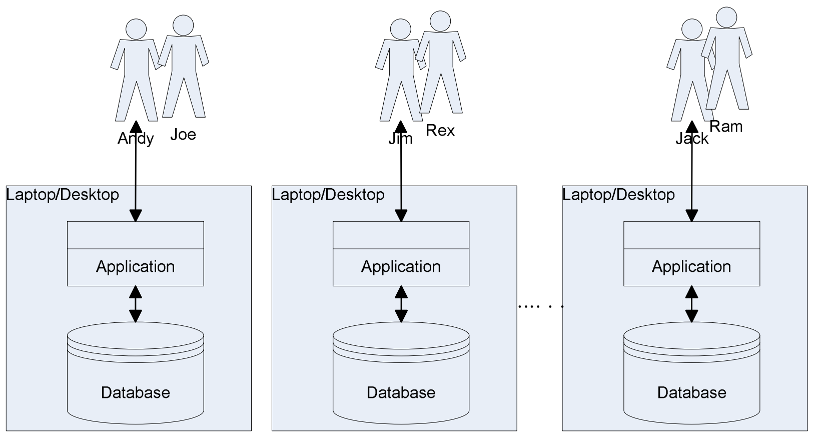
ThoughtWorks®

Why?

- One Application instance = One Database instance
- Developers work independent of other developers
- Liberty of experimenting with the database
- Use ANT/Make/Rake to automate tasks
- Database could just be a Schema
- Reduces contention
- Integrate in version control not in database

ThoughtWorks®

Ultimate Nirvana



ThoughtWorks

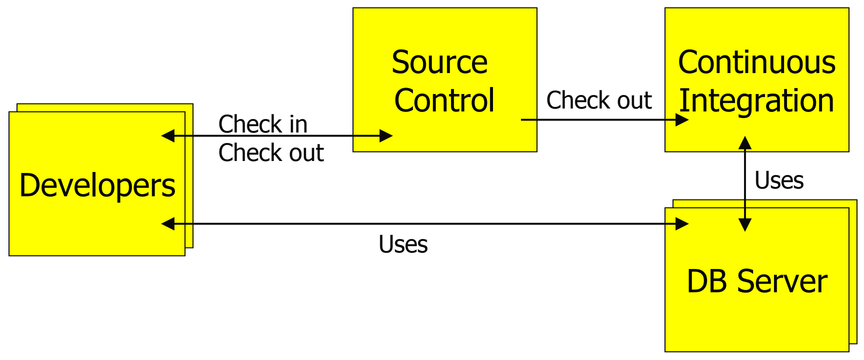
Demo: Sandbox

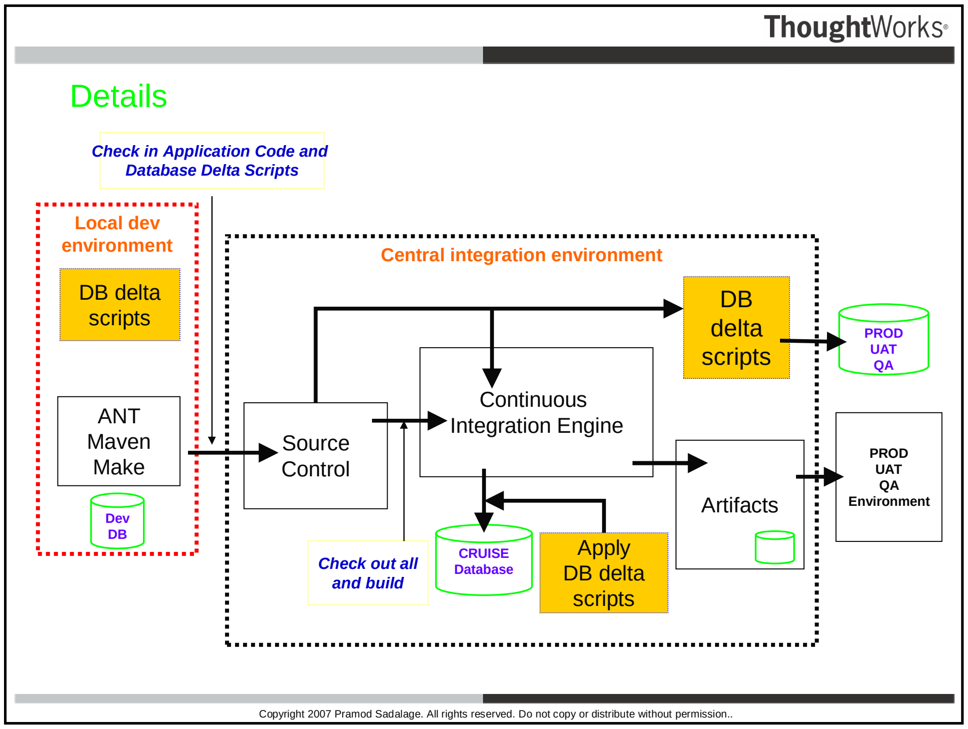
- Shows how new person joining project can startup effortlessly
- Shows reduction in waste (Lean)
- Shows how to share DBA tasks
- Shows how the DBA will get to do more
- Shows productivity gains by automation
- Shows ANT targets to Create, Clean, Initialize(populate) and Drop databases
- Shows how changes are local

ThoughtWorks

Typical Integration Environment

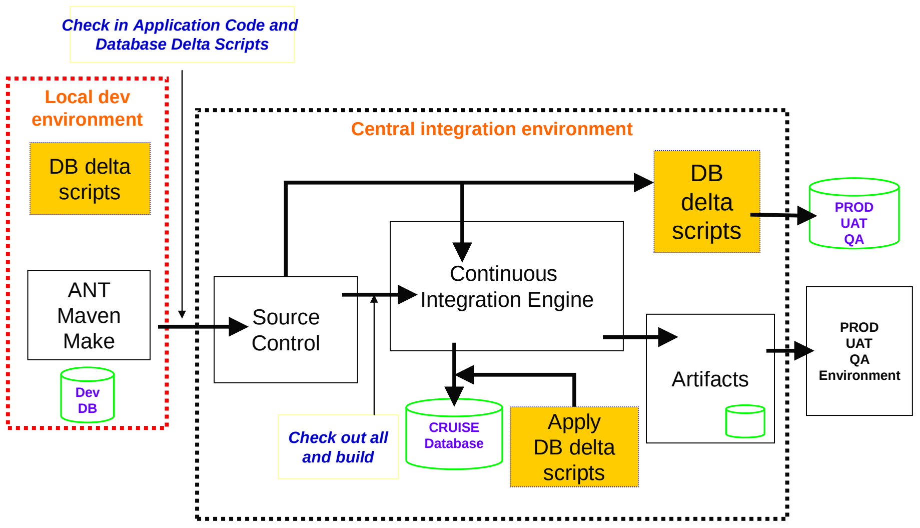
Integrate more than once a day
http://martinfowler.com/articles/continuousIntegration.html

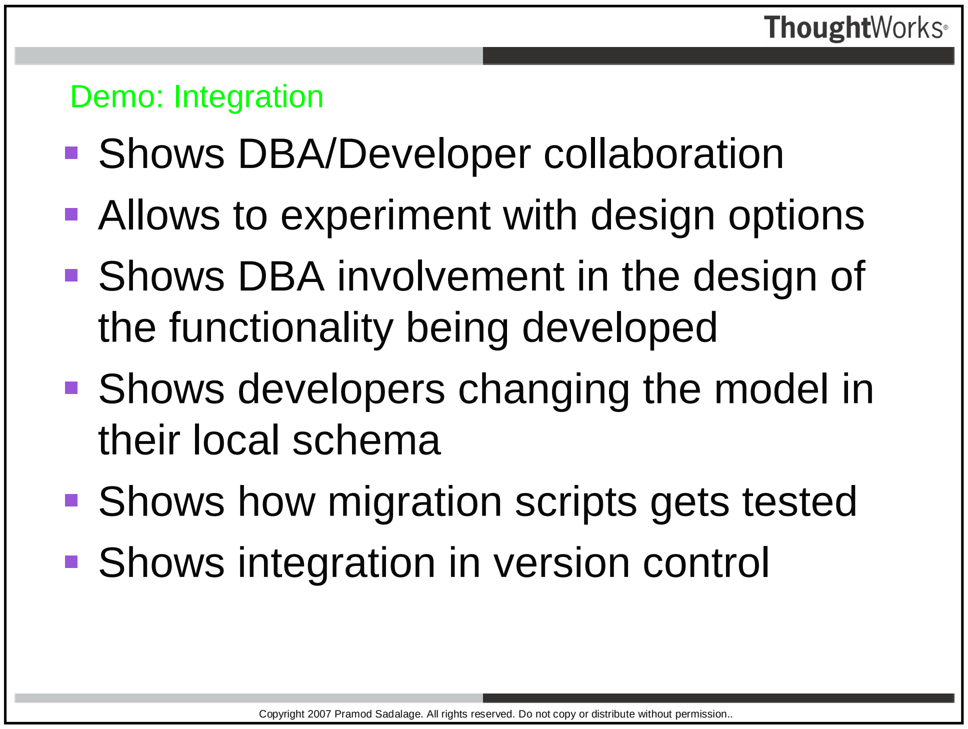




ThoughtWorks

Details

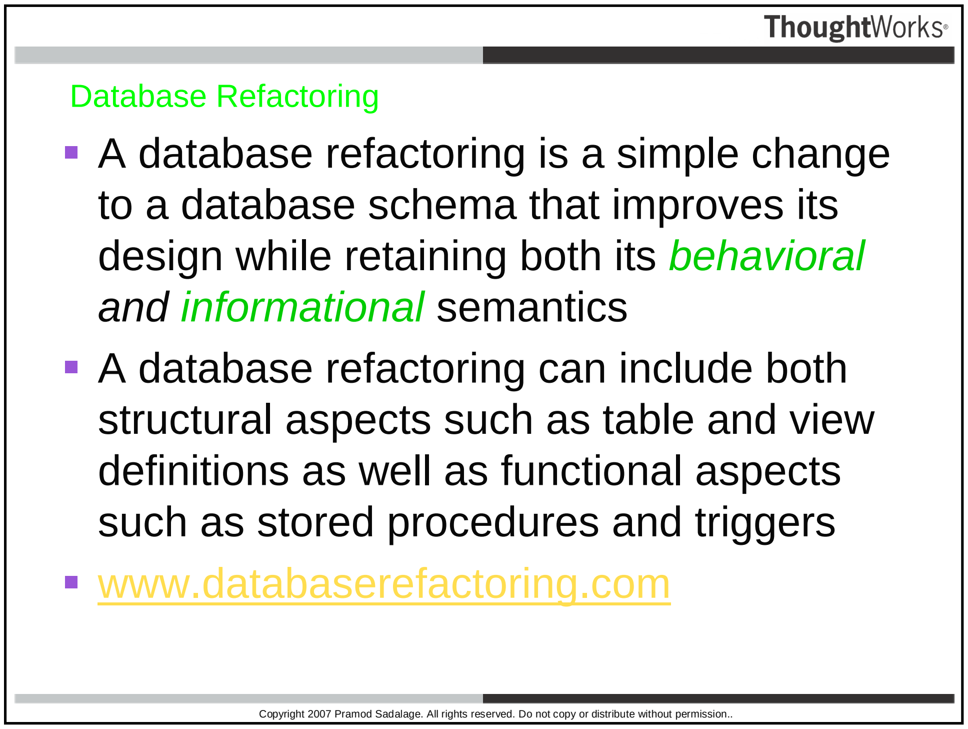




ThoughtWorks

Demo: Integration

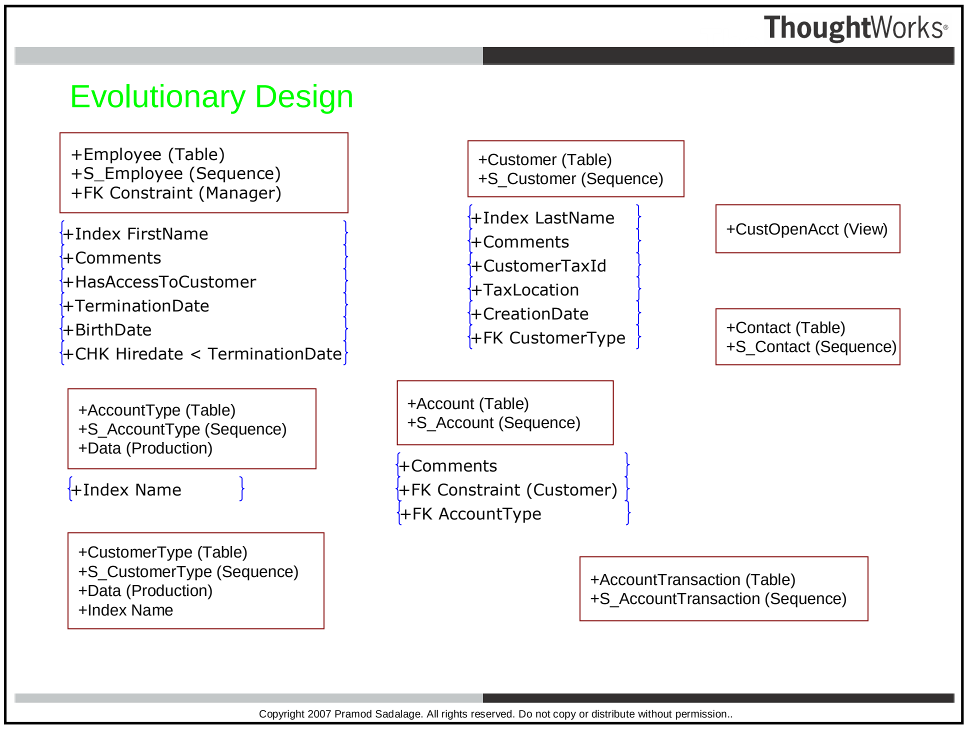
- Shows DBA/Developer collaboration
- Allows to experiment with design options
- Shows DBA involvement in the design of the functionality being developed
- Shows developers changing the model in their local schema
- Shows how migration scripts gets tested
- Shows integration in version control



ThoughtWorks

Database Refactoring

- A database refactoring is a simple change to a database schema that improves its design while retaining both its *behavioral and informational* semantics
- A database refactoring can include both structural aspects such as table and view definitions as well as functional aspects such as stored procedures and triggers
- www.databaserefactoring.com

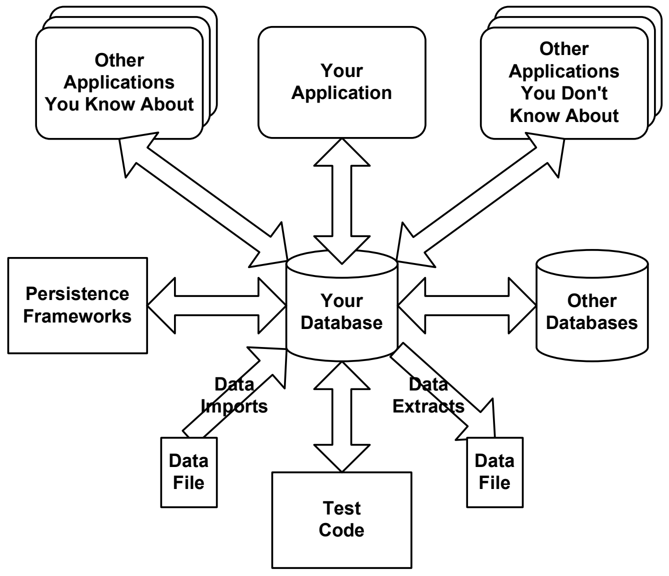


ThoughtWorks

Evolutionary Design

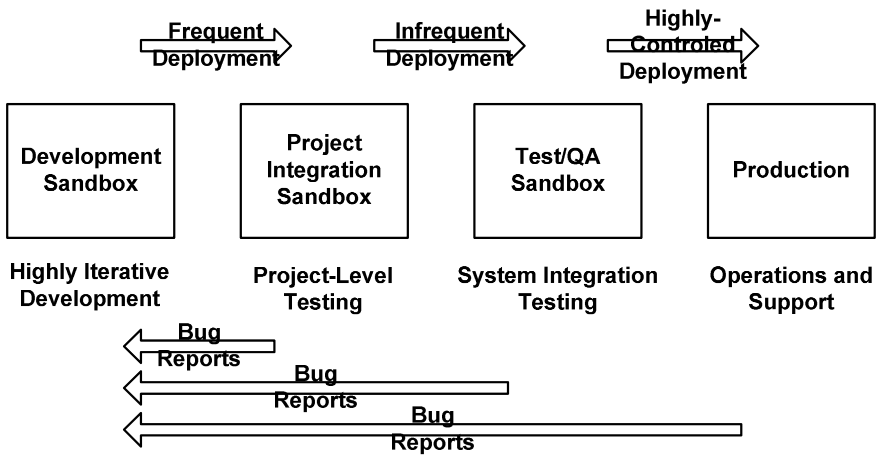
ThoughtWorks®

Database refactoring is Hard.



ThoughtWorks®

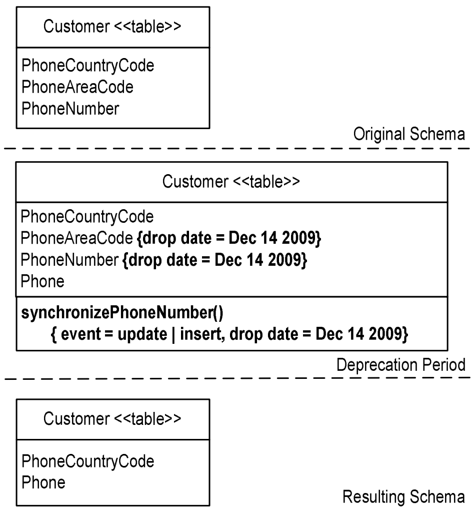
Continuous Refactoring Controlled release



ThoughtWorks®

Demo: Merge Columns

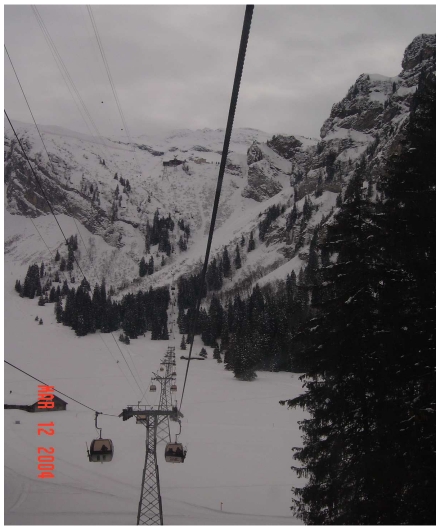
- Isolated change
- Use of dbdeploy
- Developer involvement
- Migration and new development
- How application build, links to database version

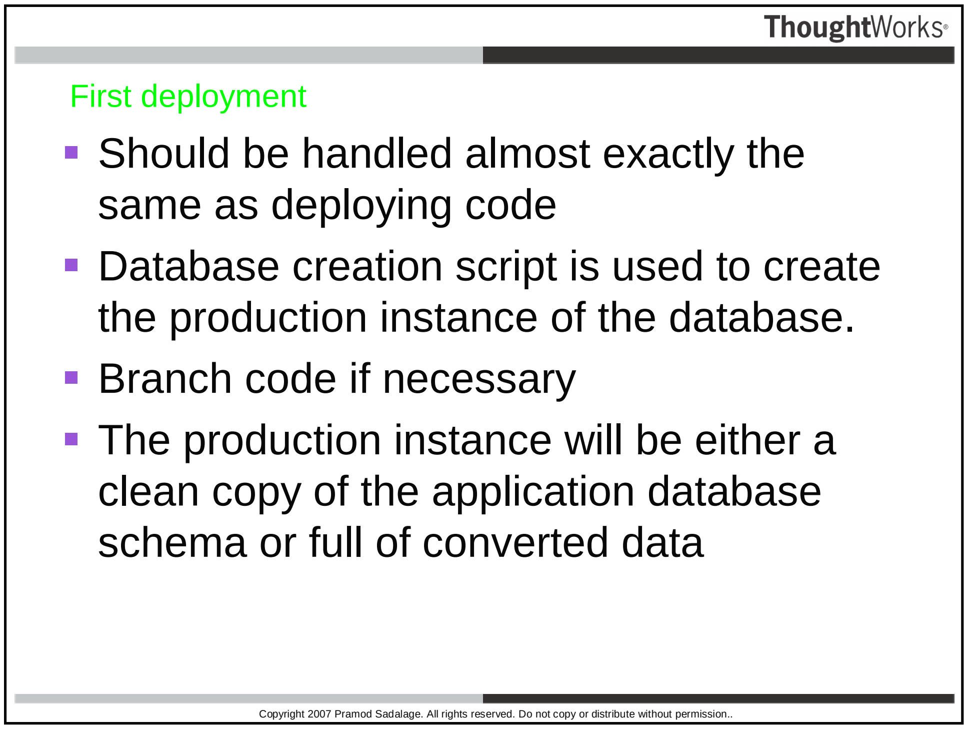


ThoughtWorks®

Demo: Assert Database Behavior

- Like any framework test your database
- These tests are your database specification
- No inadvertent change can be made to the database.

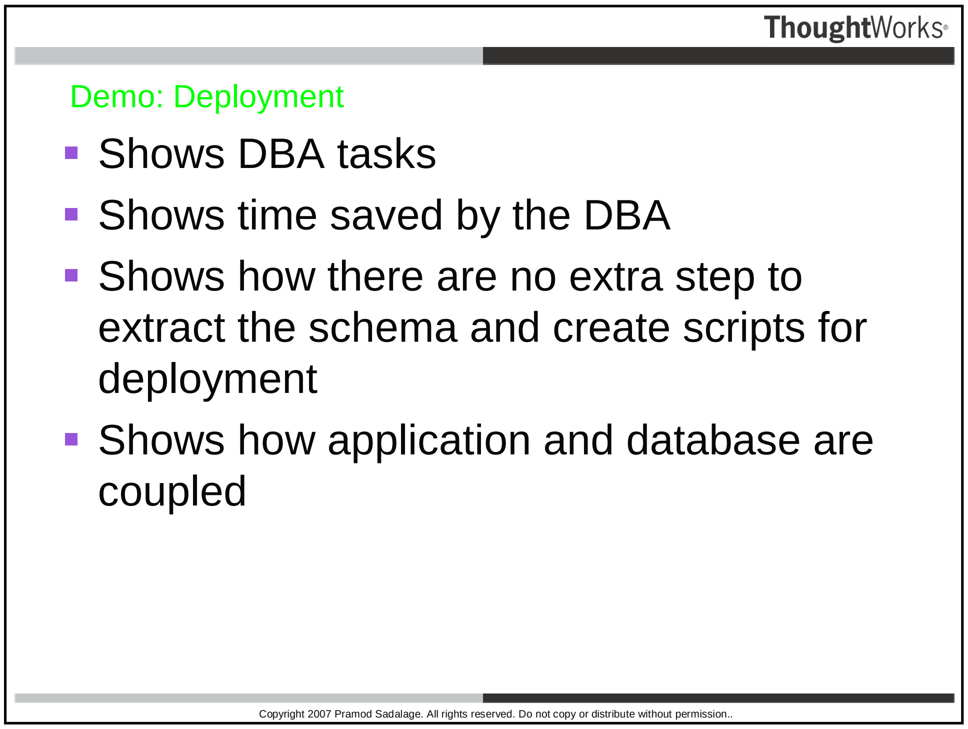




ThoughtWorks®

First deployment

- Should be handled almost exactly the same as deploying code
- Database creation script is used to create the production instance of the database.
- Branch code if necessary
- The production instance will be either a clean copy of the application database schema or full of converted data



ThoughtWorks®

Demo: Deployment

- Shows DBA tasks
- Shows time saved by the DBA
- Shows how there are no extra step to extract the schema and create scripts for deployment
- Shows how application and database are coupled

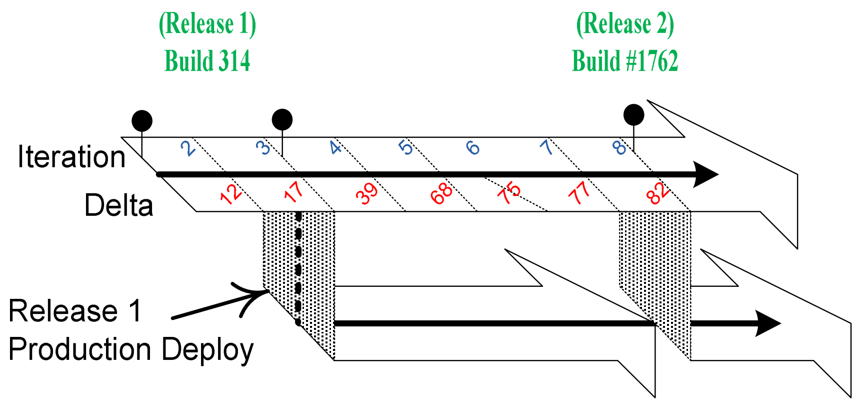
ThoughtWorks®

Manage Additional Releases

- All of the change scripts should be applied following the last production release. In chronological order. (Change Logs are stored by Build/Release/Day/Sequential)
- Since data migration scripts are part of change logs, data is also migrated
- Allows to test the migration on a test database for performance impacts

ThoughtWorks®

Additional Releases



Fast Forward

- Production database instances can be migrated to a future state of deployment.
- New application releases can be acceptance tested against the most recent data currently in production.
- Migrations can be tested continuously against copy of production data.

Boat Metaphor (Deploy Frequently)

ThoughtWorks®

More Info

Keep in touch

www.sadalage.com

Additional Resources

groups.yahoo.com/group/agileDatabases

databaserefactoring.com

martinfowler.com

dbdeploy.com