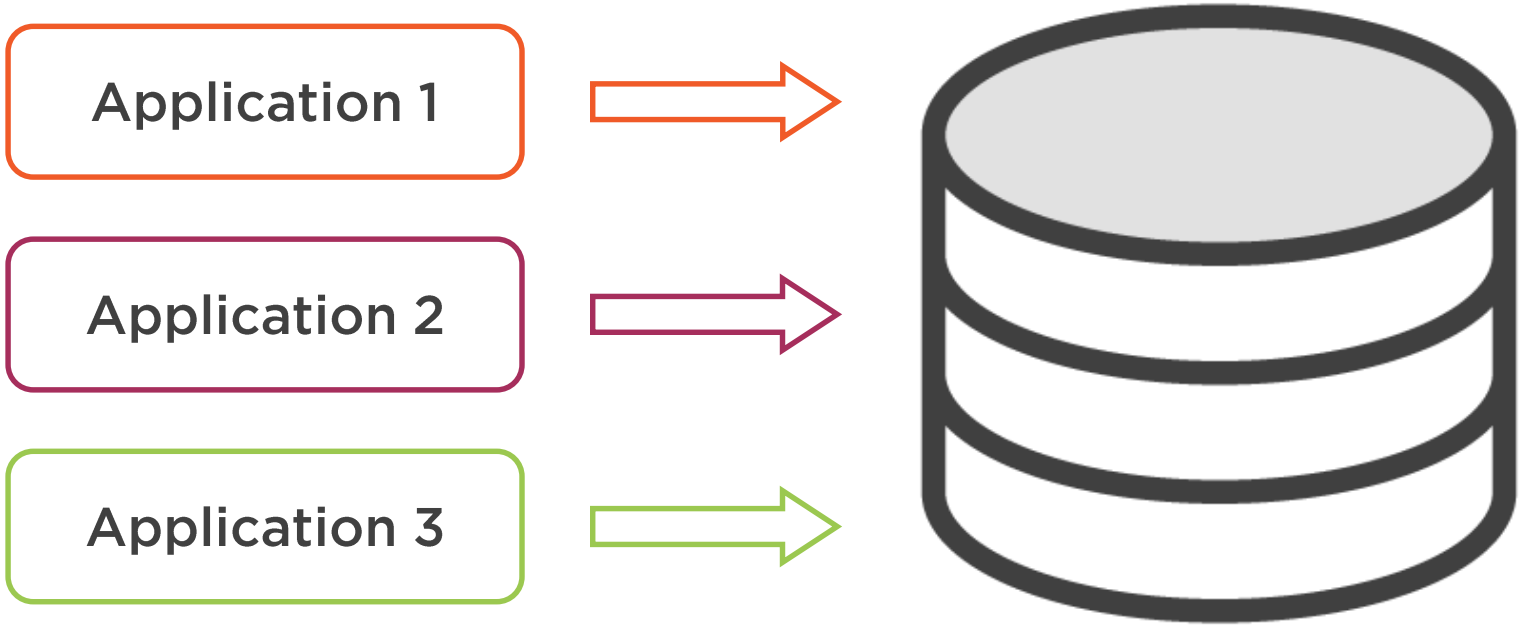
Refactoring Integration Databases Using Evolutionary Design

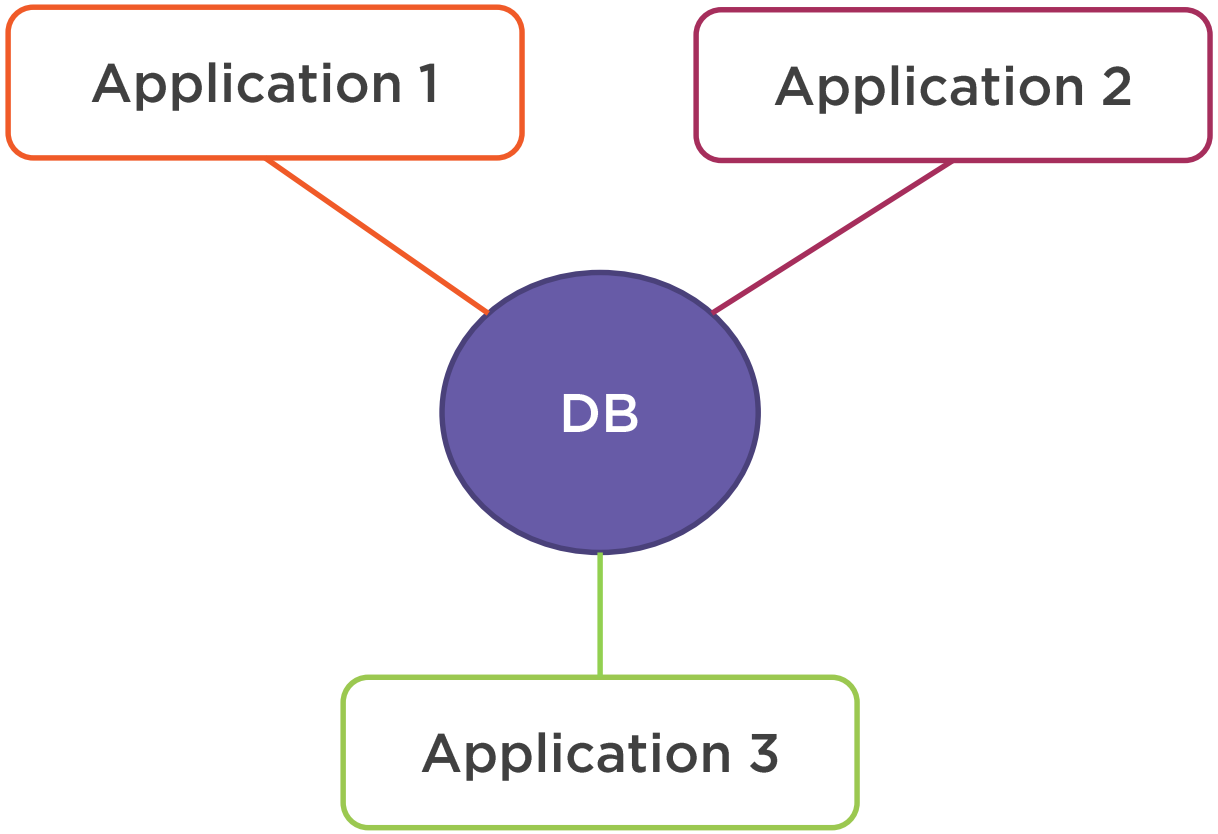
Vladimir Khorikov
PROGRAMMER

@vkhorikov www.enterprisecraftsmanship.com

Integration Databases



Integration Databases

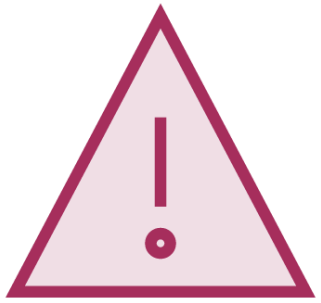


- Point of coupling
- Cannot be refactored the regular way

Integration Databases

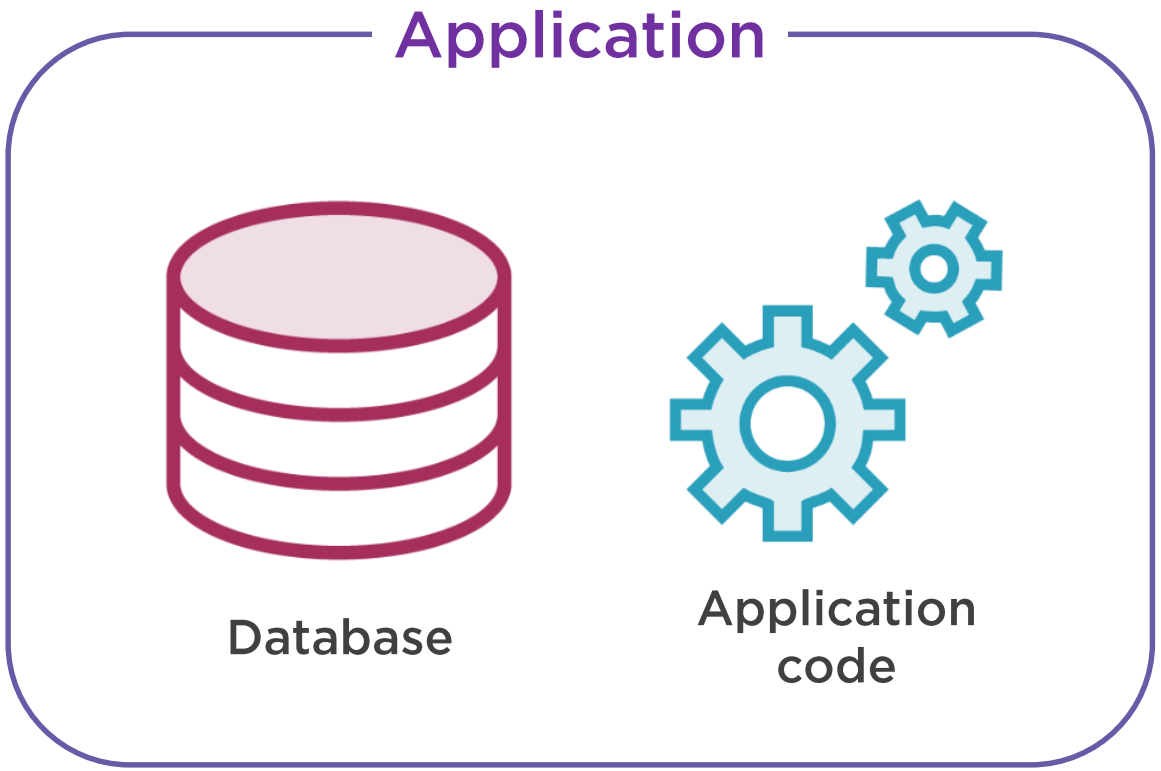
Accumulates technical debt

Integration Databases



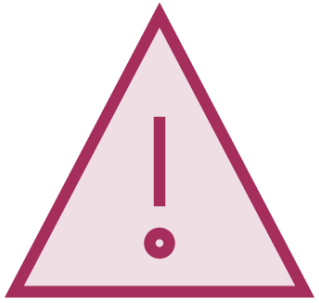
Try to avoid integration databases

Application Database



- Intrinsic part of the application
- Refactor along with the application code
- Don’t use the DB for communication purposes

Integration Databases

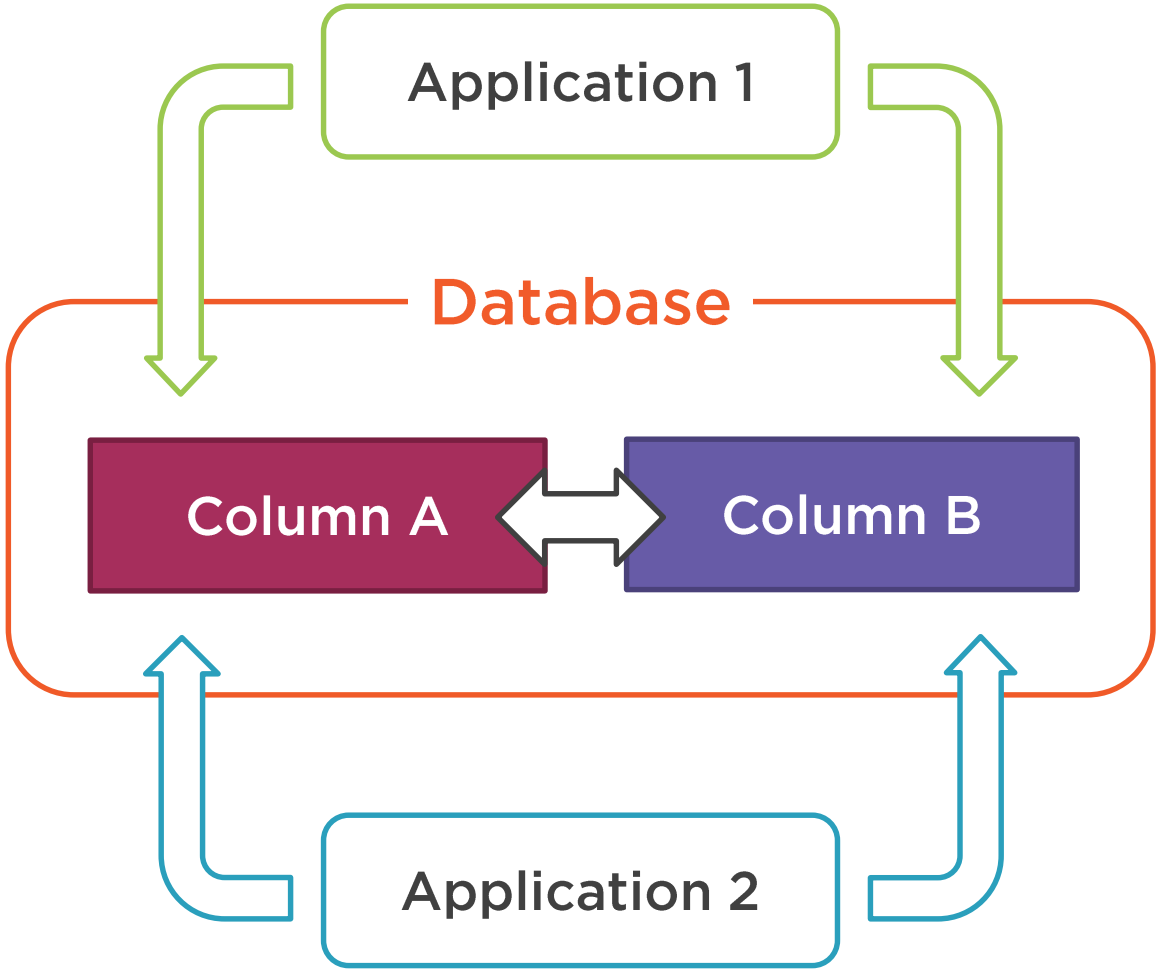


Not always possible to avoid integration databases

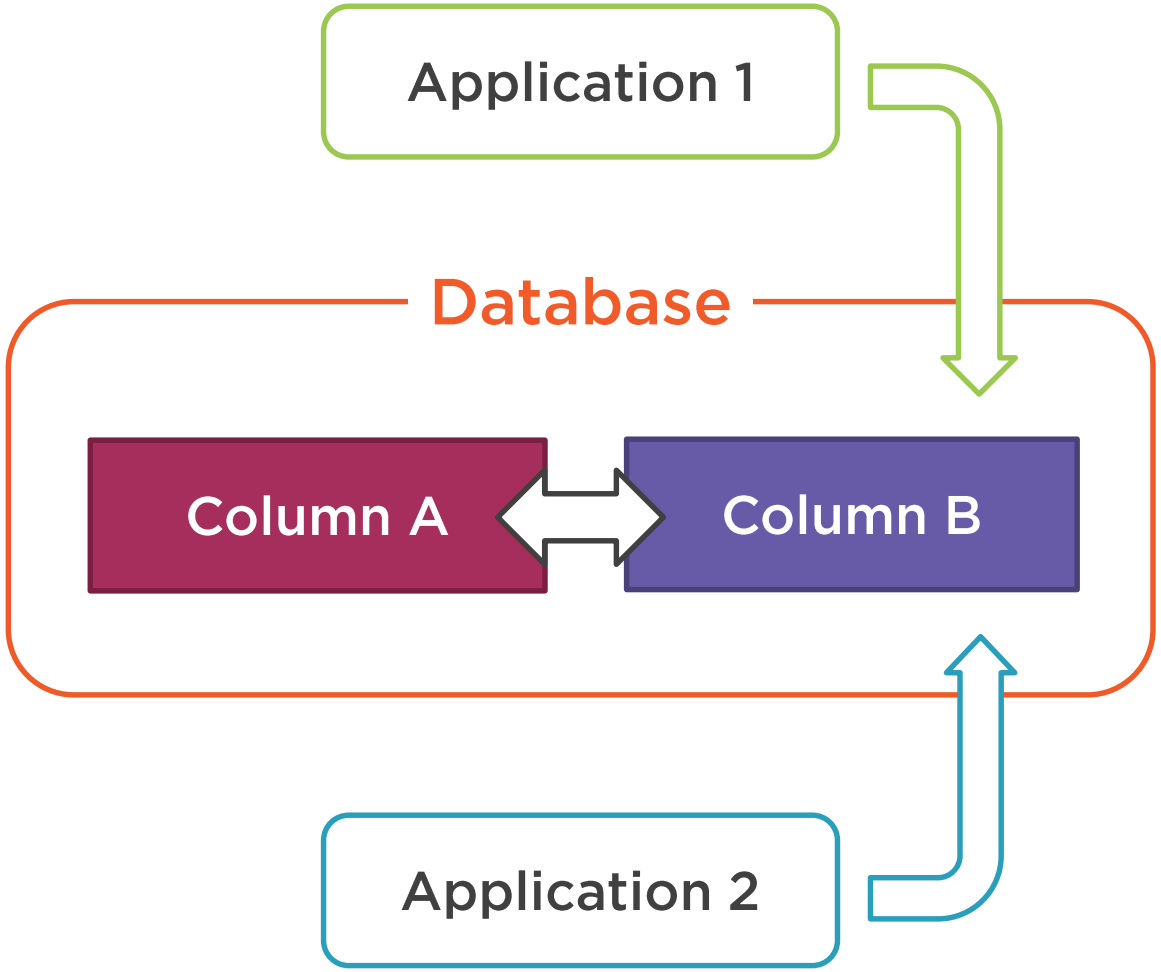
Refactoring Integration Databases

Must preserve backward compatibility

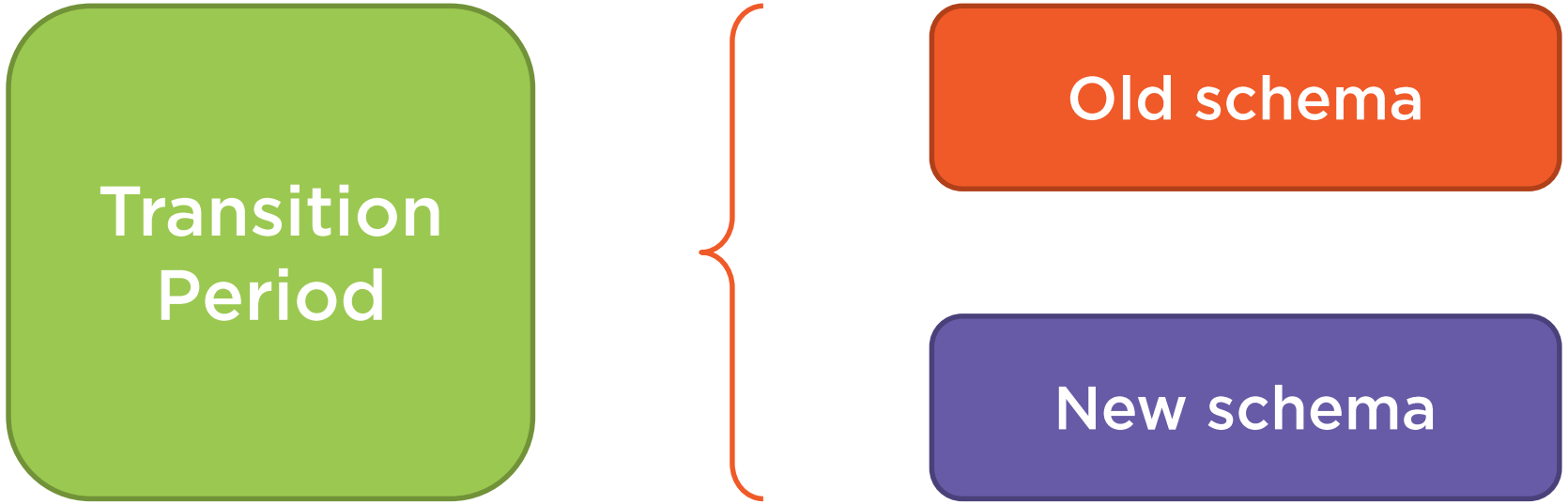
Refactoring Integration Databases



Refactoring Integration Databases

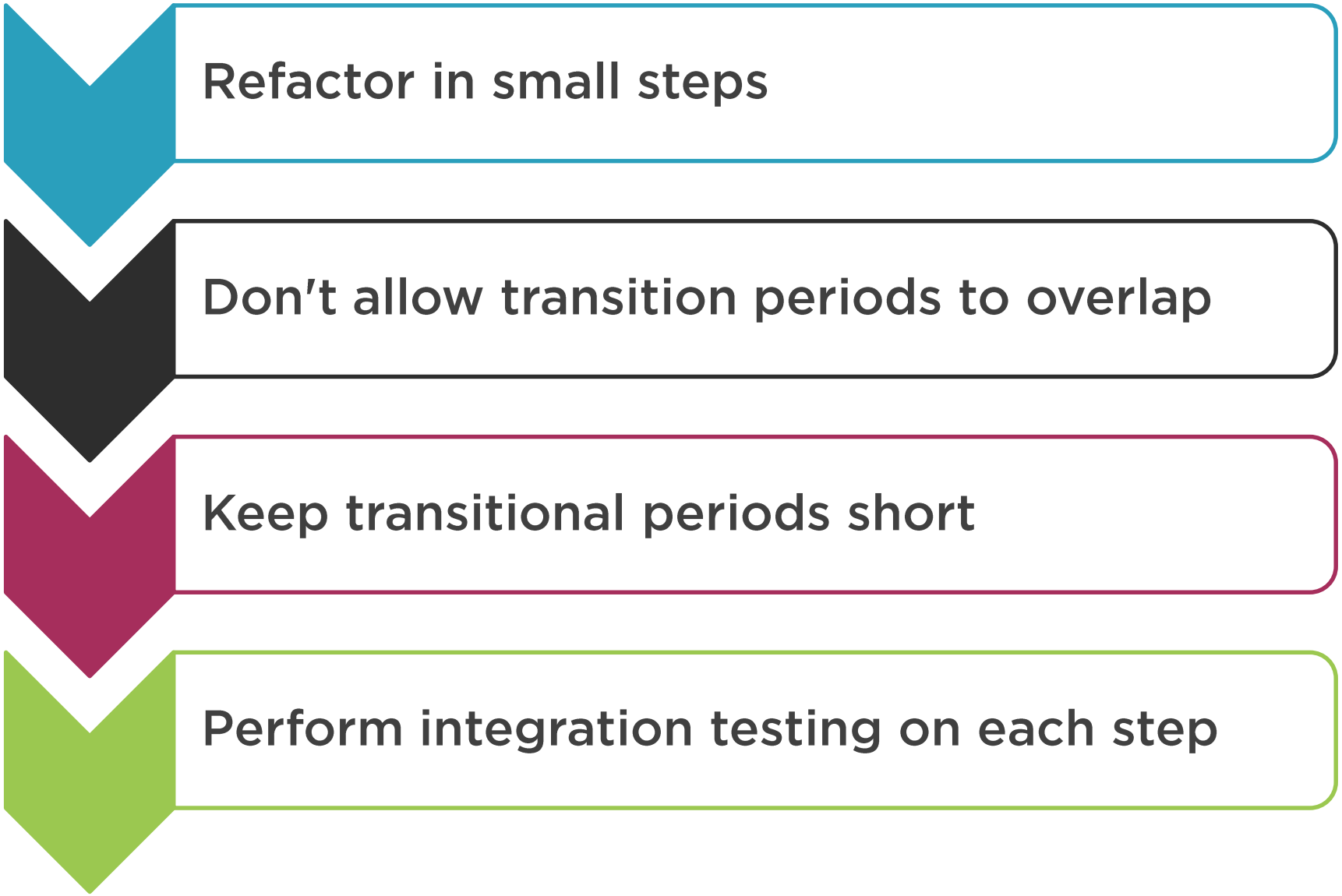


Refactoring Integration Databases

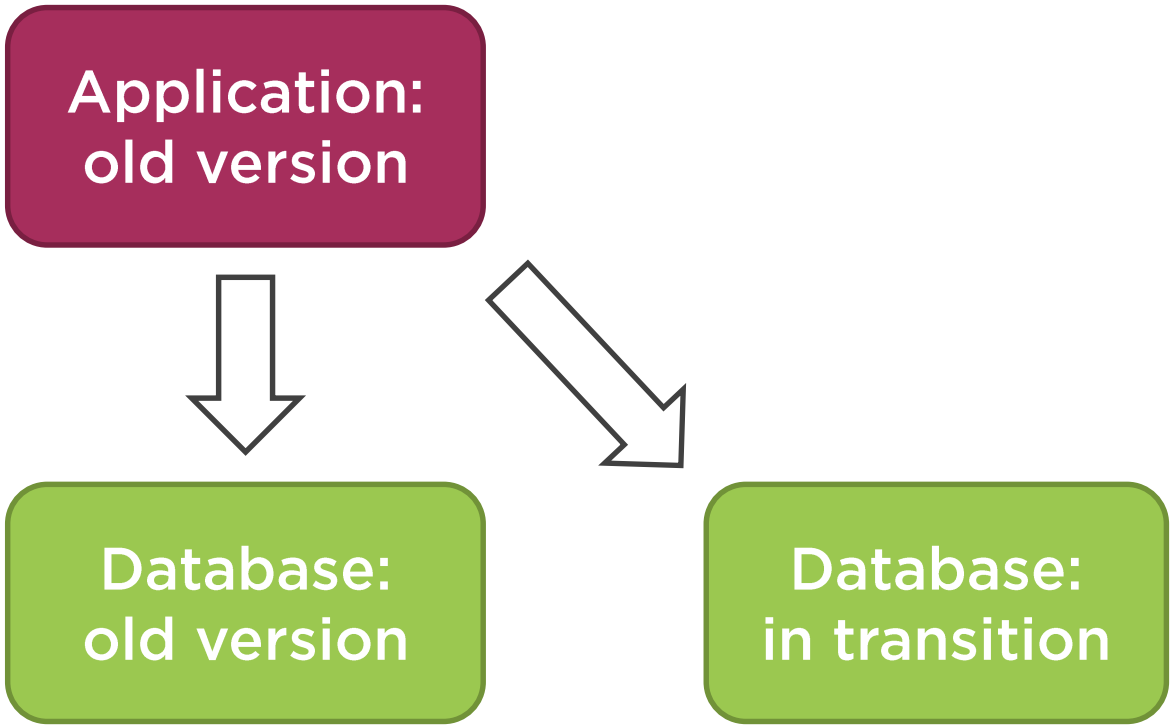


Application databases don't have a transition period

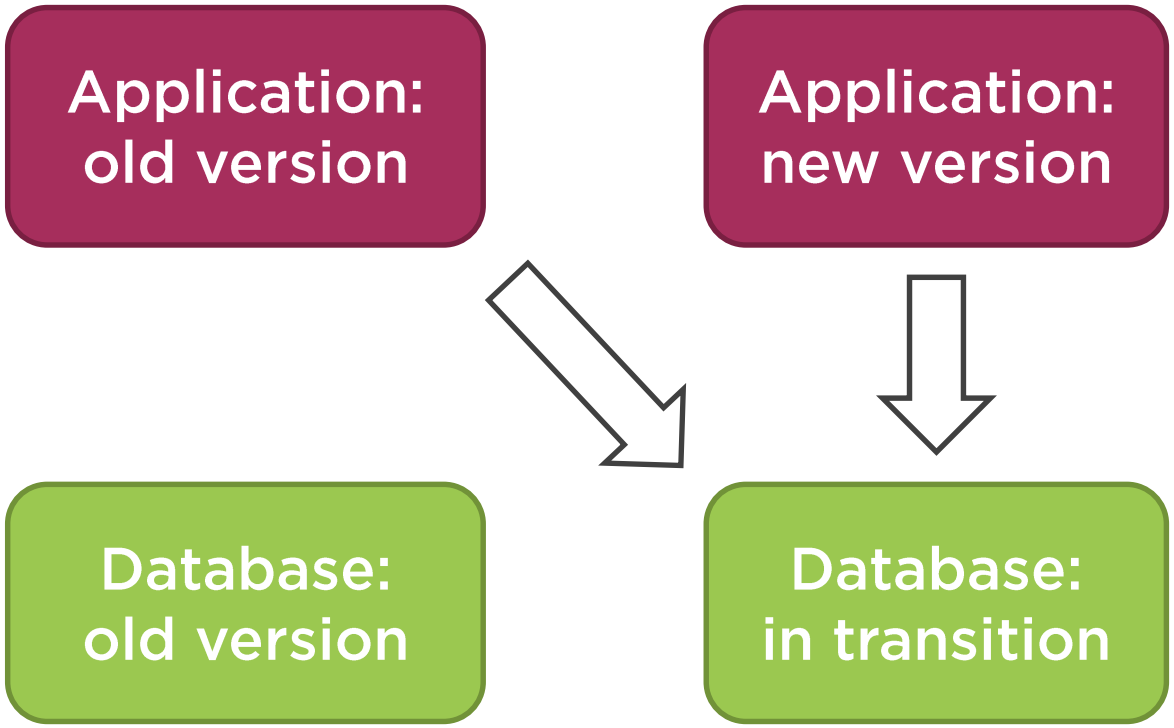
Refactoring Integration Databases



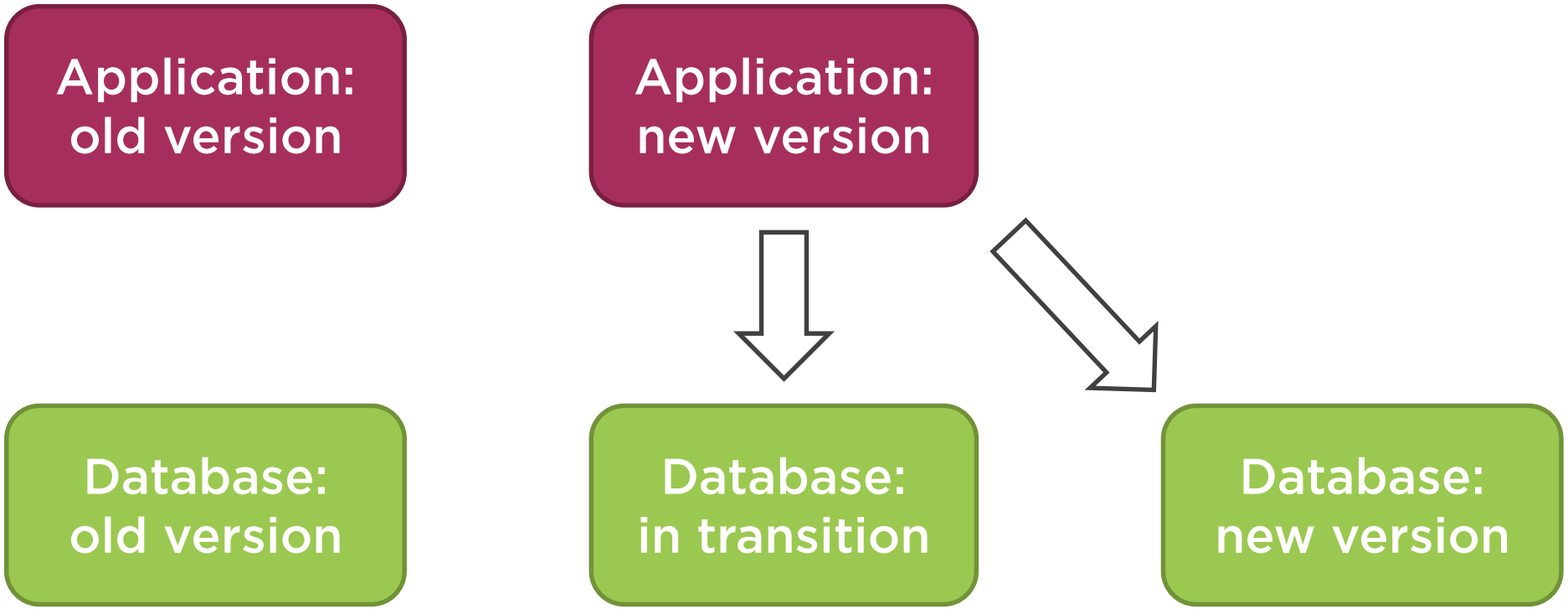
Integration Testing



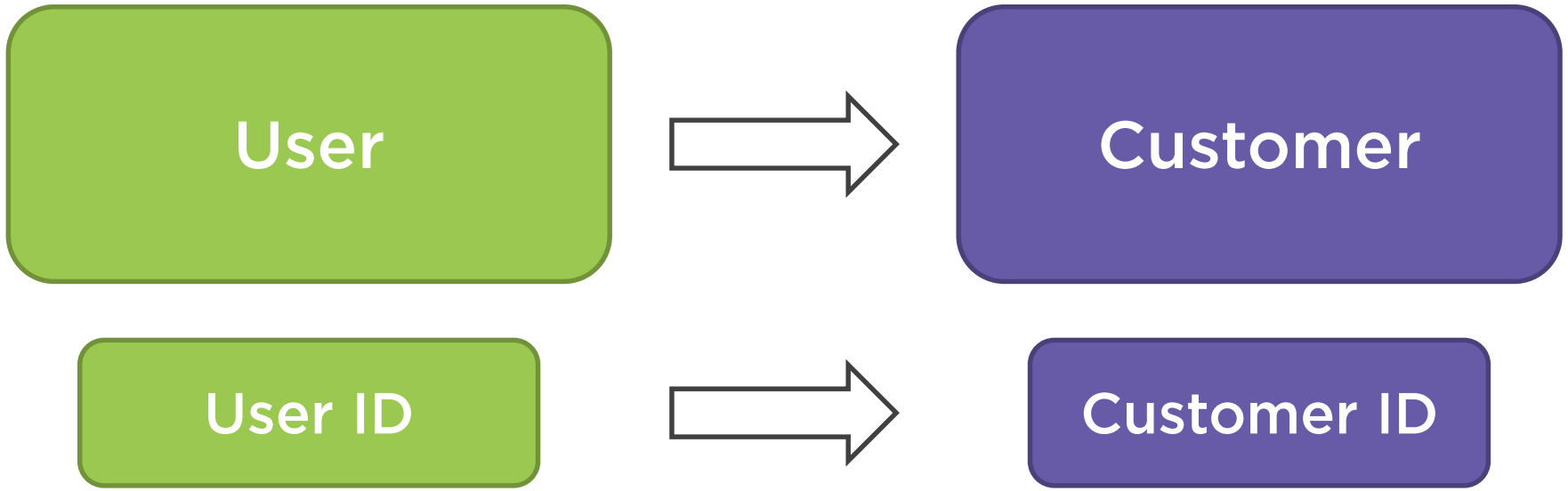
Integration Testing



Integration Testing

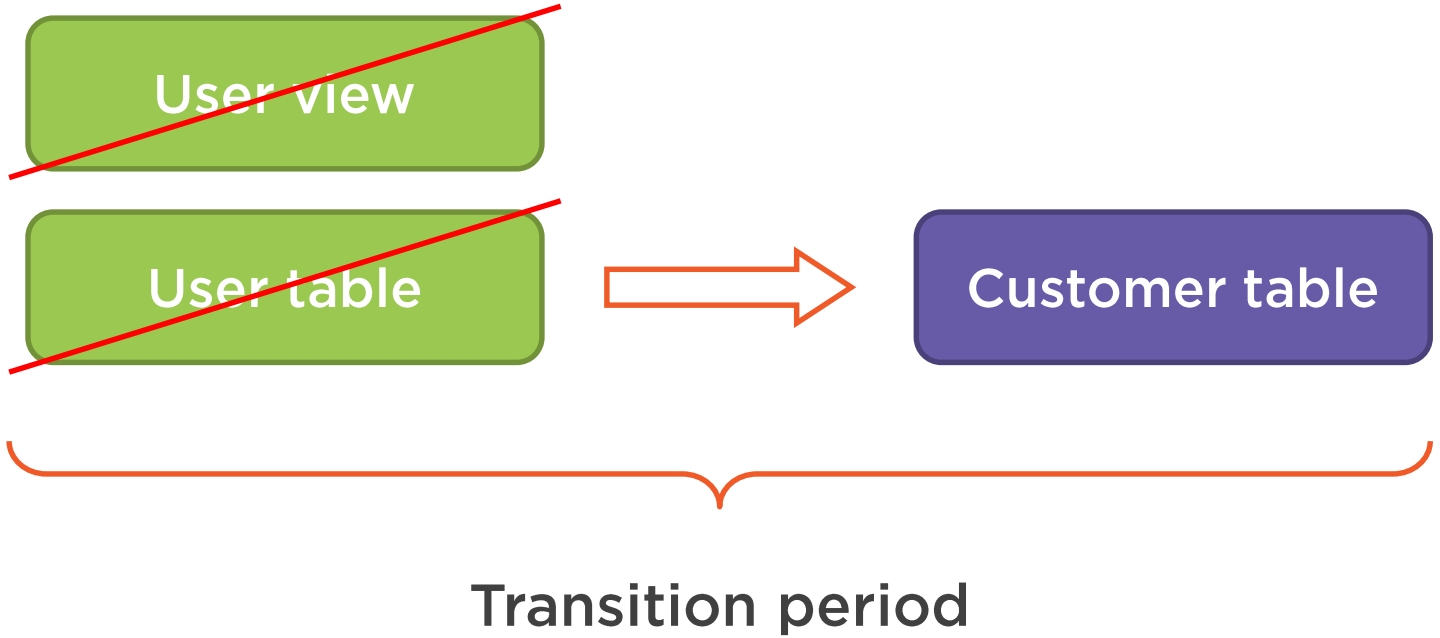


Renaming the User Table



Will use a database view for backward compatibility

Recap: Renaming the User Table



Recap: Renaming the User Table

02_RenameUser_begin.sql

03_RenameUser_end.sql

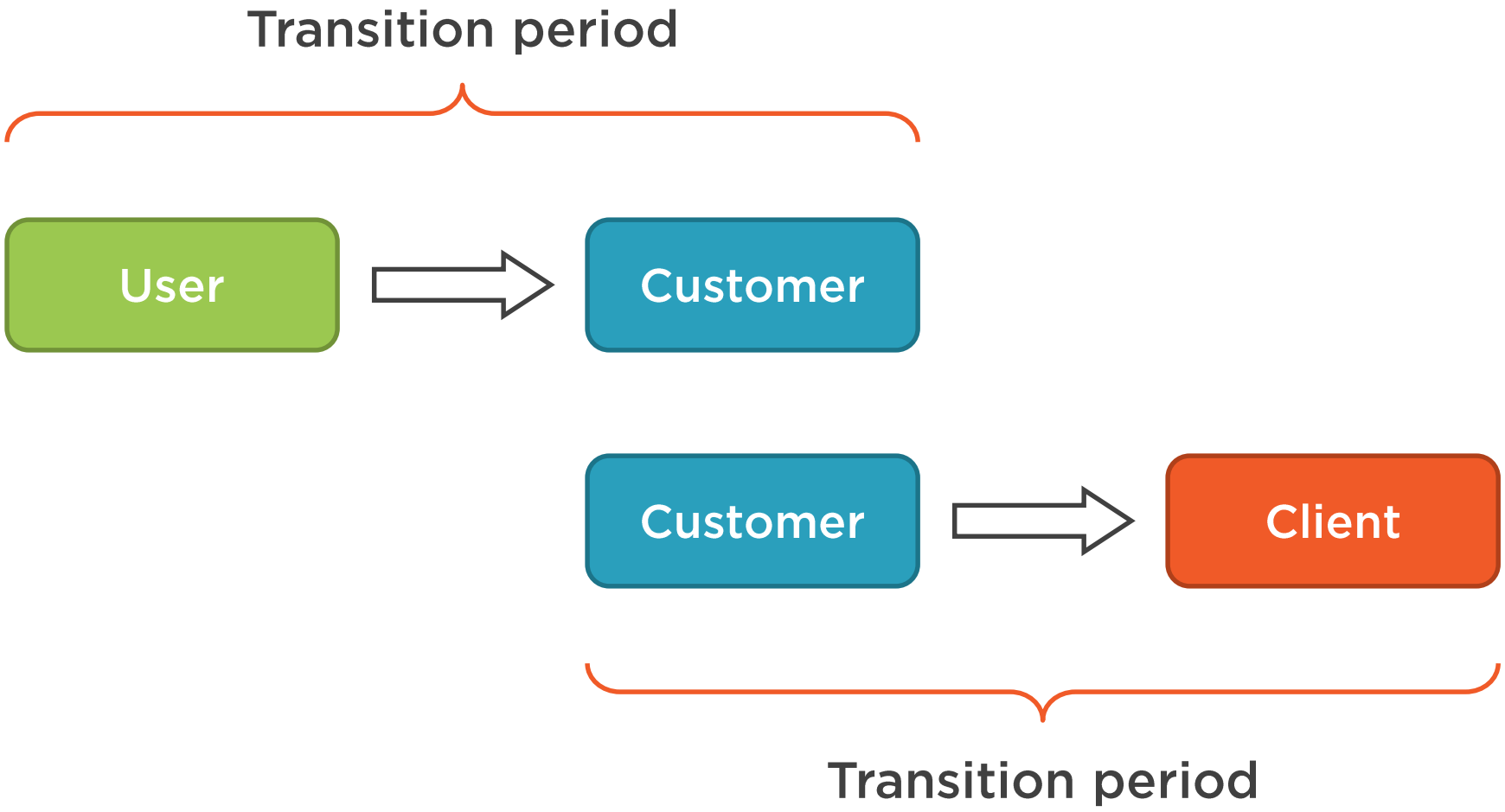
Easier to navigate through transition periods

Recap: Renaming the User Table

Not all RDBMS support updatable views

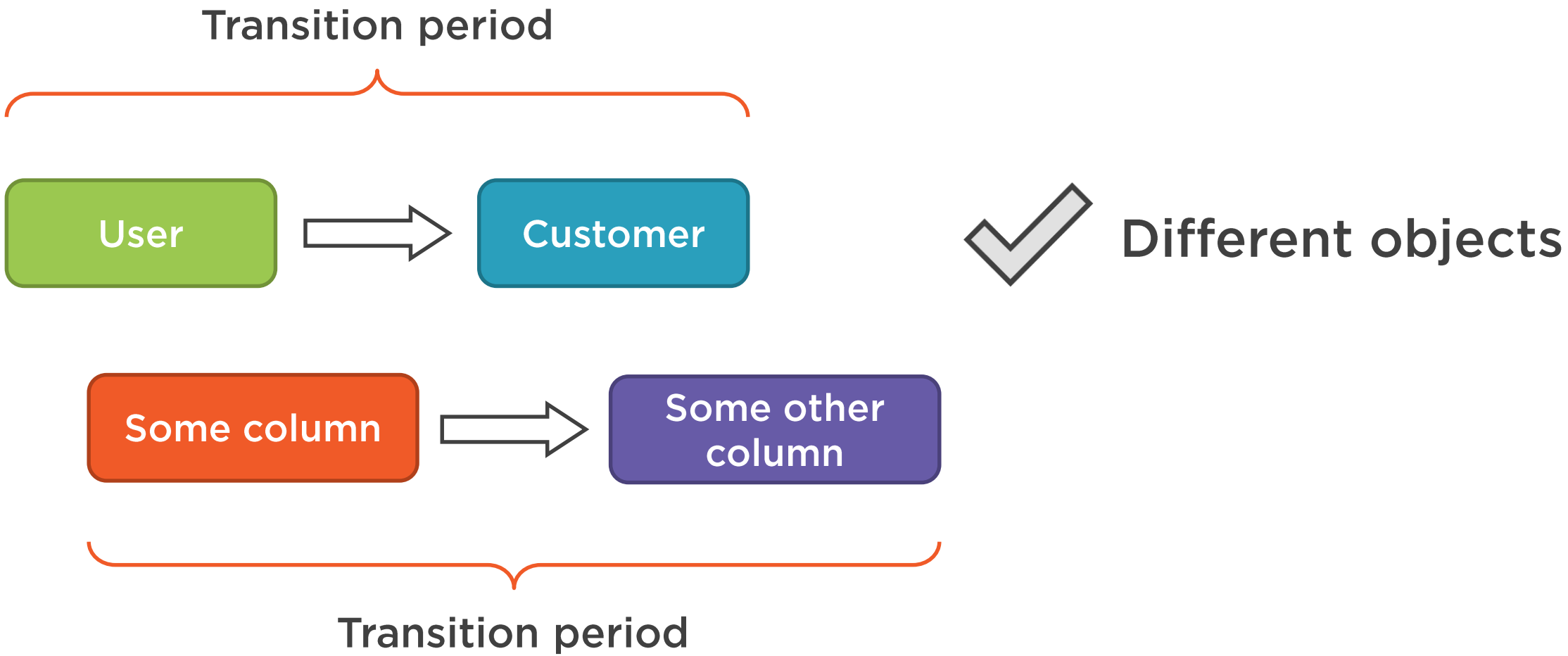
In such cases, you will have to keep both tables

Overlapping Transition Periods

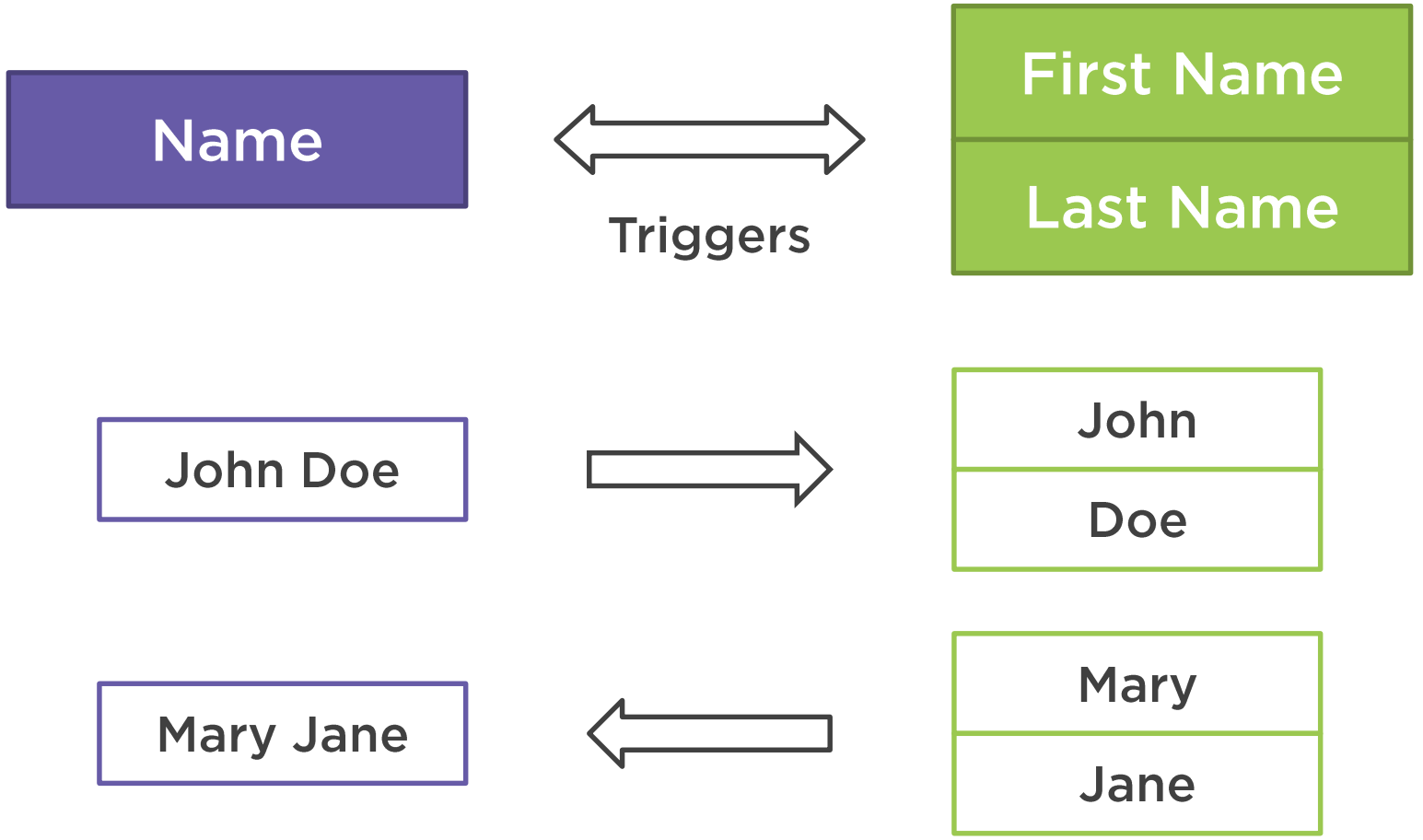


Same object

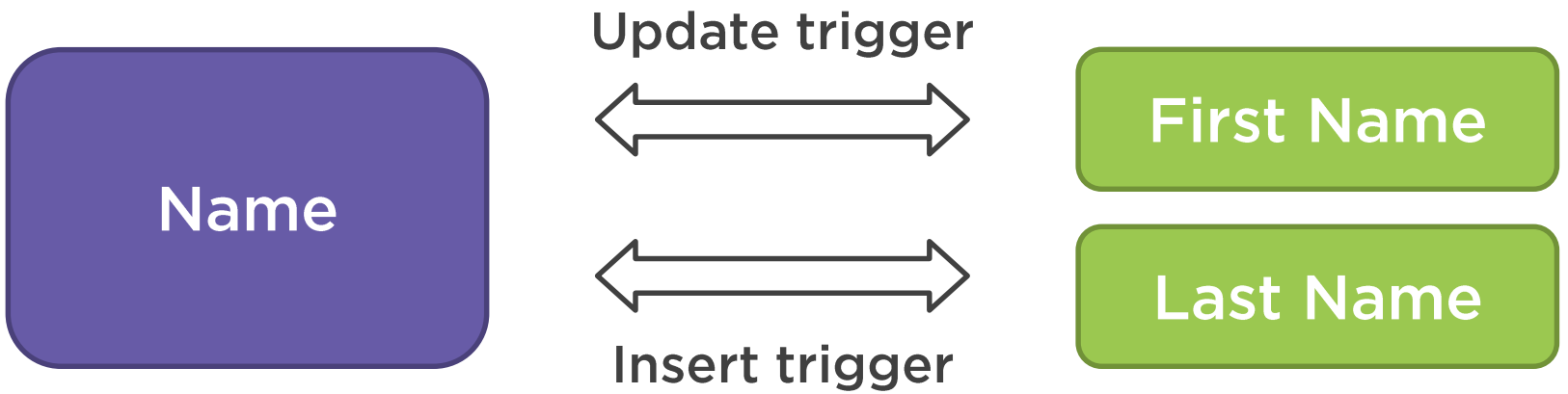
Overlapping Transition Periods



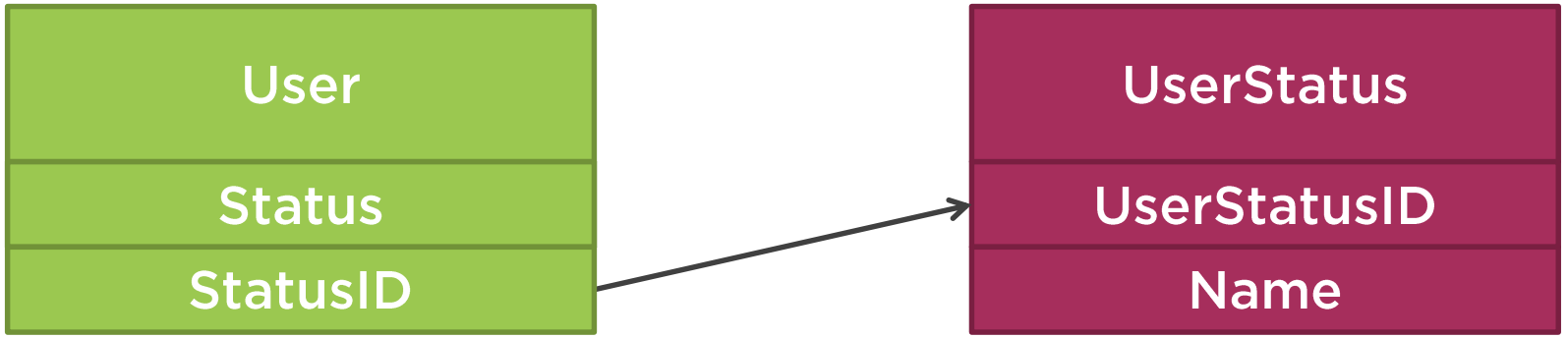
Splitting the Name Column



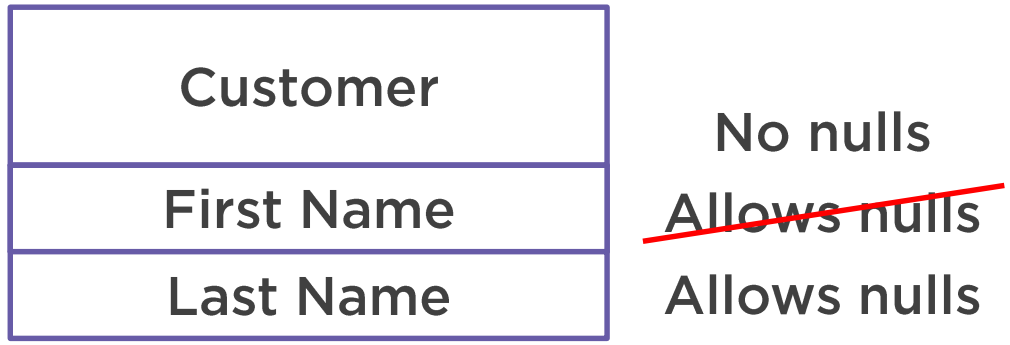
Recap: Splitting the Name Column



Recap: Splitting the Name Column

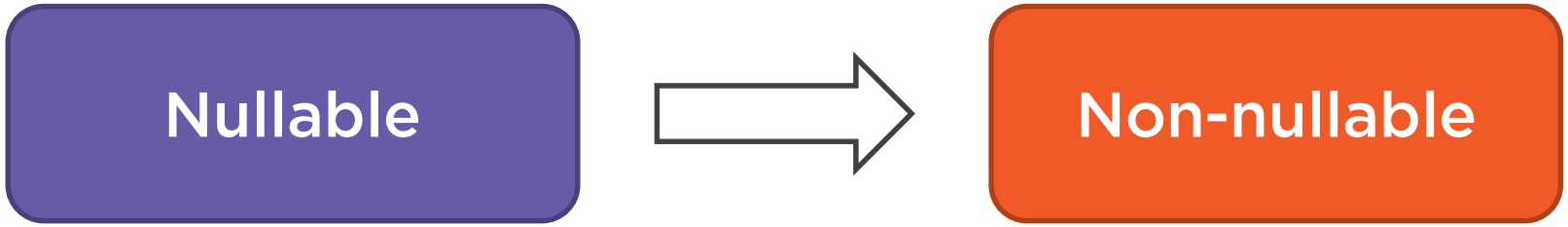


Making a Nullable Column Non-nullable



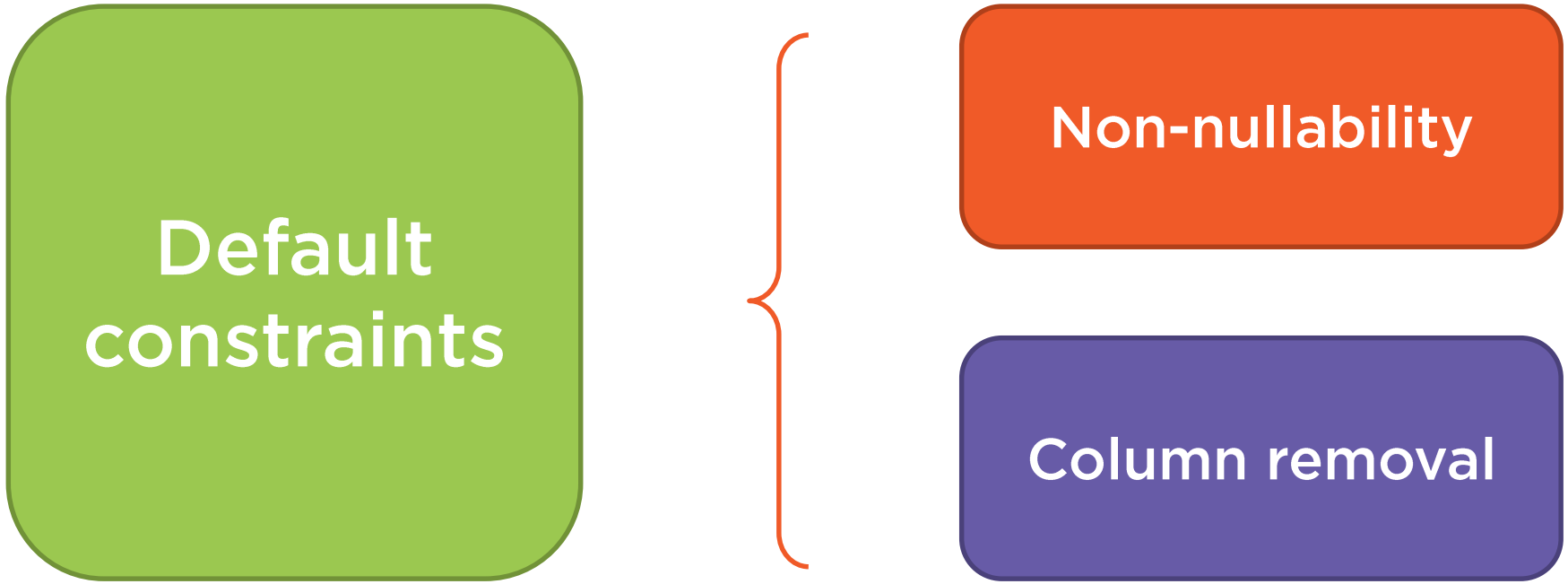
Use default constraints

Recap: Making a Nullable Column Non-nullable



Default constraint

Recap: Making a Nullable Column Non-nullable



Integration Databases Refactoring

1-on-1 views

Help with table renaming

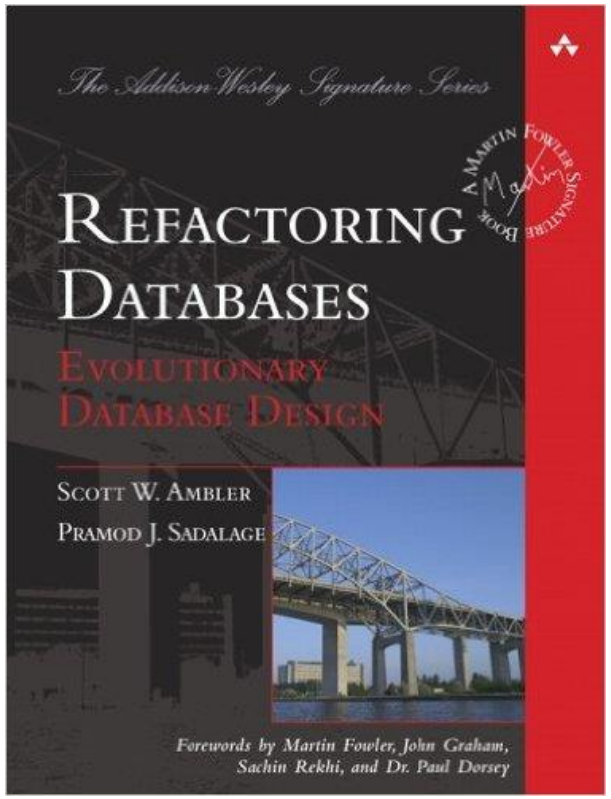
Triggers

Help with complex structure transformations

Default constraints

Non-nullability and column removal

Integration Databases Refactoring



Refactoring Databases: Evolutionary Database Design

by Scott Ambler and Pramodkumar Sadalage

http://bit.ly/database-refactoring

Module Summary

Refactoring integration databases

- All changes must be backward compatible

2-step approach

- Start a transition period where both schema versions are supported
- Remove the old version after all clients had adopted it

Prefer application databases over integration databases

Module Summary

Keep refactorings small

Don’t allow transition periods to overlap

Keep transition periods short

Perform integration testing in each step

- Old application within the transition period
- New application within the transition period
- New application outside of the transition period

Refactoring an integration database

- 1-on-1 view
- Update and Insert triggers
- Default constraint

Resource List

Database migration tool	https://github.com/vkhorikov/DatabaseUpgradeTool
	http://bit.ly/migration-tool
The elephant in the room: Continuous Delivery for Databases	https://vimeo.com/131637362
Refactoring Databases: Evolutionary Database Design	http://www.amazon.com/gp/product/0321293533/
	http://bit.ly/database-refactoring

Course Summary

Basic principles behind database delivery

- Keep your database in the source control system
- Refactor it the same way you refactor your application code

State-based approach: state is explicit

Migration-based: transitions are explicit

State-based vs migration-based approaches

- Large distributes team or lots of logic in DB -> state
- Small team or lots of structural changes -> migrations
- Multiple production DBs -> migrations

Combine the two approaches if possible

Contacts

vladimir.khorikov@gmail.com

@vkhorikov

http://enterprisecraftsmanship.com/