Lessons Learned From Refactoring Multi Terabyte Databases

for the past 10 years

Sponsors & Organizers

What I Want to Talk About

Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

Martin Fowler

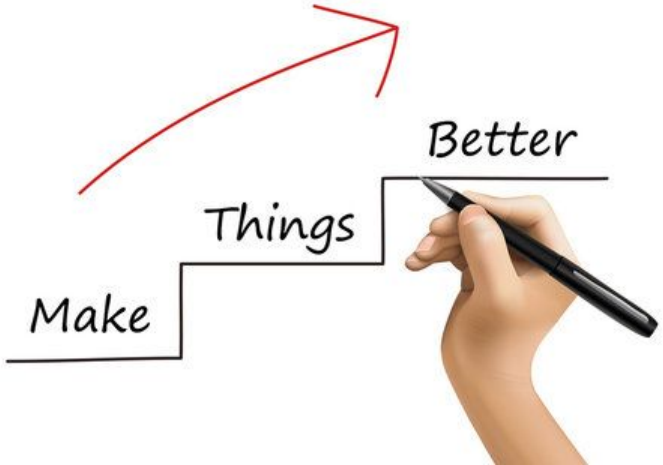


Image Source: https://www.regulation.org.uk/ob-failure_to_learn.html

What was the biggest database you've ever worked with?

About Me - Sergey Olontsev

Data Engineering Team Lead @ IPONWEB (a part of Criteo), ex-Kaspersky (Staff Engineer), ex-MVP.

18 years of experience with databases and high load systems (started with SQL Server 2000)

Worked with OLTP databases up to 50 TB, relational DWH up to 200 TB. Each, not in total! :)

100+ billions of rows in tables (OLTP).

https://sergeyolontsev.com

https://linkedin.com/in/sergeyolontsev

Microsoft CERTIFIED Master SQL Server® 2008

Microsoft CERTIFIED Solutions Master Charter - Data Platform

Very Large Database (VLDB) Challenges

Why to refactor?

Amount of Data

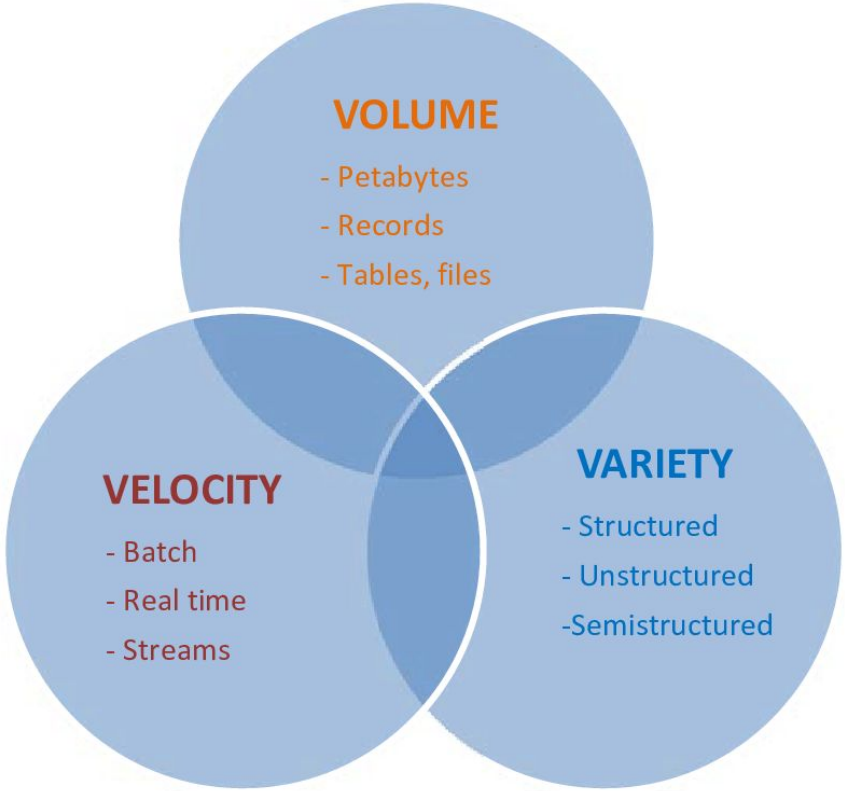
Transaction Rate

High Concurrency

Complex Logic

Number of Users

Developers / Hiring



Before You Start

Development Tools

You need a modern IDE + version control + CI/CD (Visual Studio + SSDT, IntelliJ DataGrip).

- Rename database objects / column names support
- Schema comparison / generate change script
- Better have dynamic SQL support / search in comments

Tests

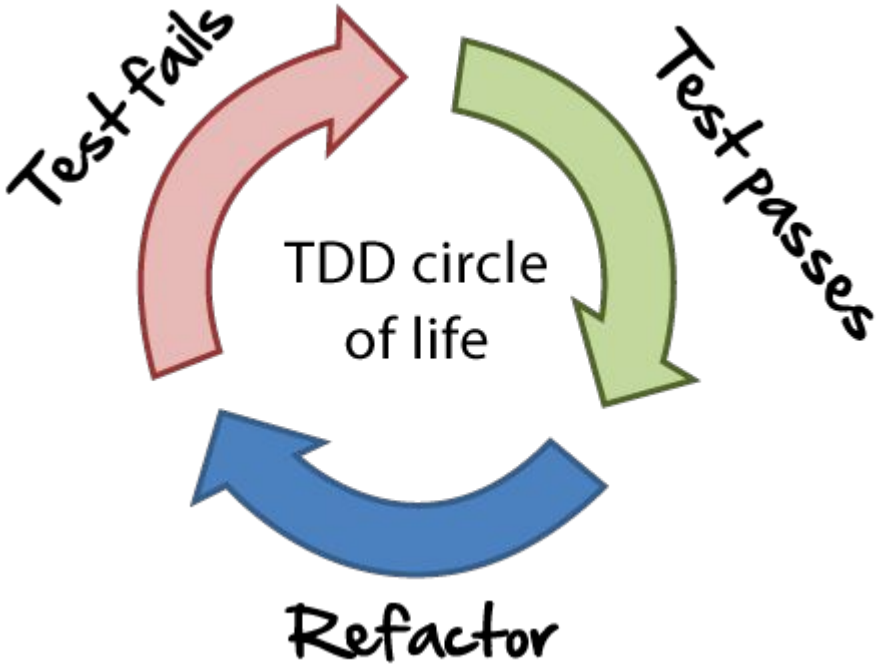


Image Source: https://medium.com/@nataprawiraf/test-driven-development-red-green-approach-3aa4231941c7

SQL Server Features Which Helps Refactoring

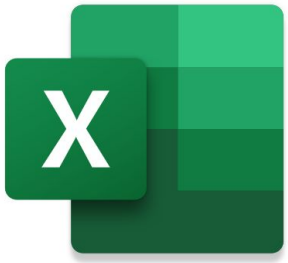
1. Add Column (NULL, default value)
2. Views (partitioned views, updatable views)
3. Synonyms (point to another DB objects for decoupling, renaming objects)
4. Triggers (data sync, triggers on views)
5. Stored Procedures (mapping to another SP)
6. DML / DDL Transactions

#1 Know Your Customer

Image Source: https://www.freepik.com/premium-vector/kyc-know-your-customer-with-business-verifying-identity-its-clients-concept_27204725.htm

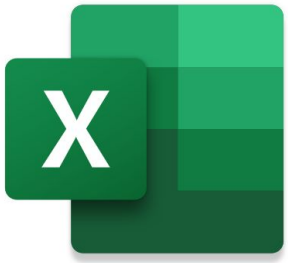
User Access Documentation

Do you have user access documentation in Excel or Confluence (Wiki)?



User Access Documentation

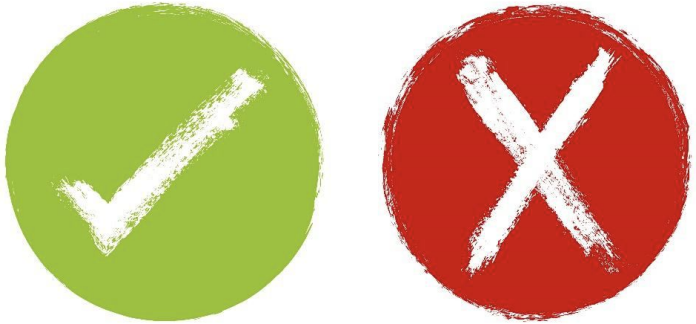
Do you have user access documentation in Excel or Confluence (Wiki)?



Gathering information just before significant refactoring usually leads to fail!

Document Interfaces Through Tests

1. Stored Procedures
2. Table Variables
3. Views
4. Direct Table Access



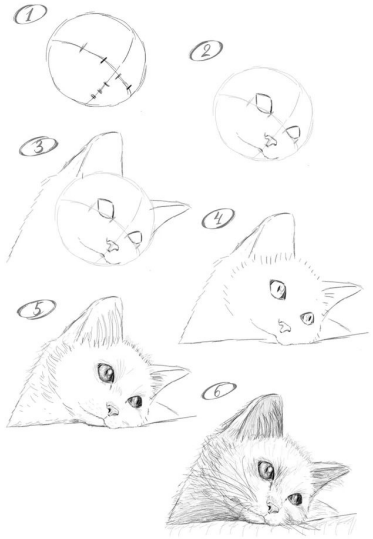
If someone has access to a table or user defined table variable, make a test, which will fail if a new column will be added, changed or removed.

No Direct Access

Encapsulate data access. Stored procedures are a common way to encapsulate access to your database.

Decouple application code from database tables. Stored procedures are an effective way to decouple applications from database tables. They enable you to change database tables without changing application code.

Implement entity-based security access control (SAC). Instead of directly accessing source tables, applications instead invoke the relevant stored procedures.



#2 Step-By-Step

Image Source: https://www.reddit.com/r/Drawing_Tutorials/comments/vcutzg/how_to_draw_a_sad_kitten_or_cat_stepbystep/

Step-By-Step Examples

Merge Tables

1. Move column 1
2. Move column 2
3. …
4. Move column N
5. Drop old table

Step-By-Step Examples

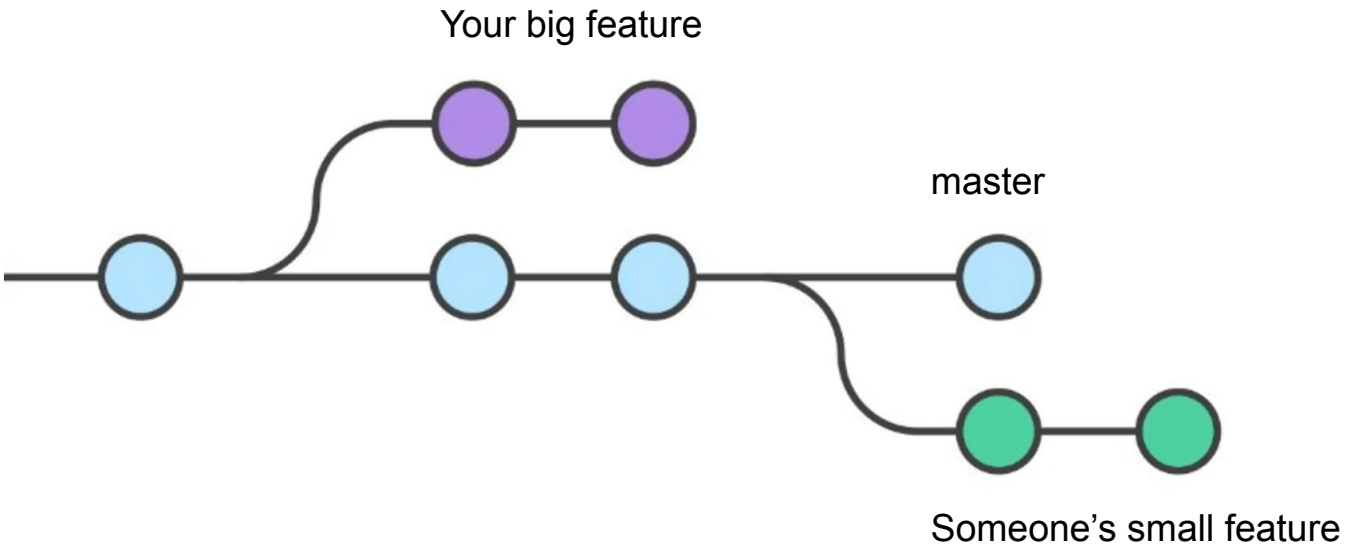
Merge Tables

1. Move column 1
2. Move column 2
3. …
4. Move column N
5. Drop old table

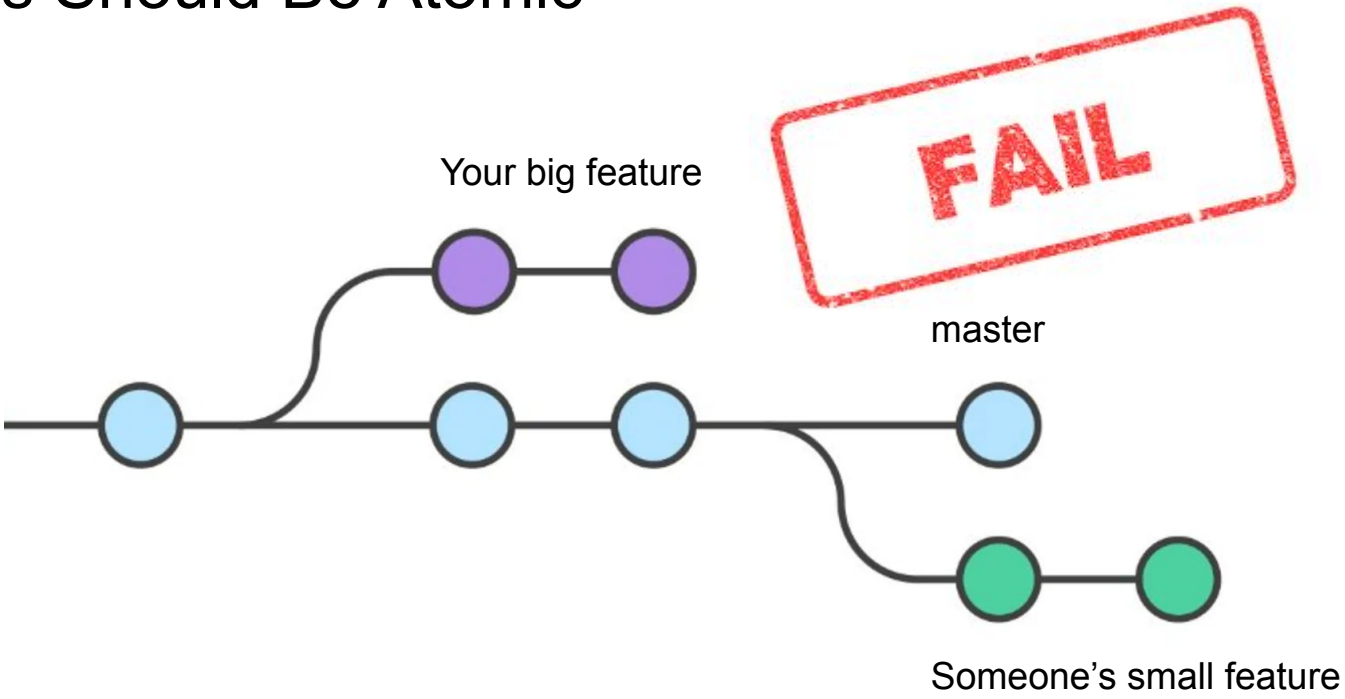
Move Column

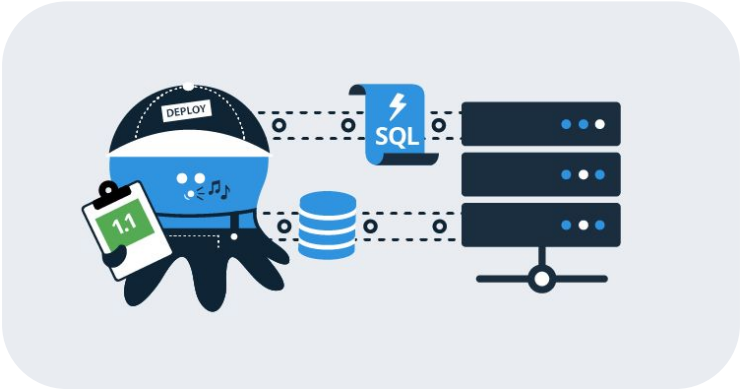
1. Create a new column
2. Move / sync data
3. Change queries
4. Drop old column

Changes Should Be Atomic



Changes Should Be Atomic





#3 Deploy

Image Source: https://octopus.com/blog/database-deployment-automation-adhoc-scripts

Deploy Scripts Versioning

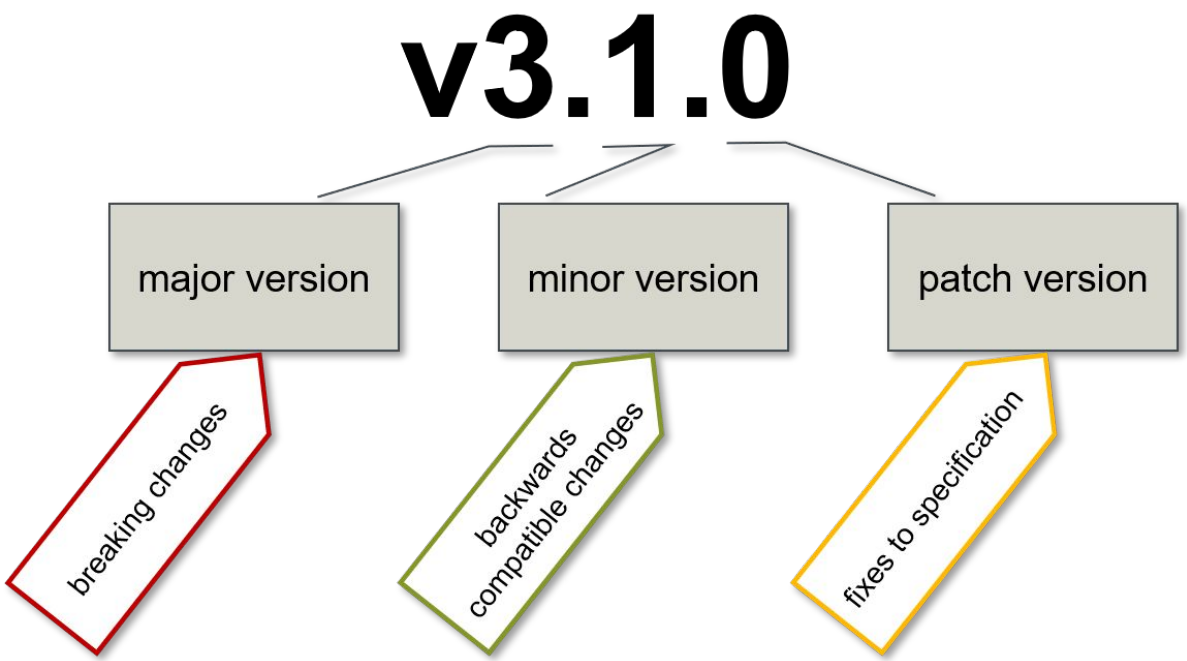
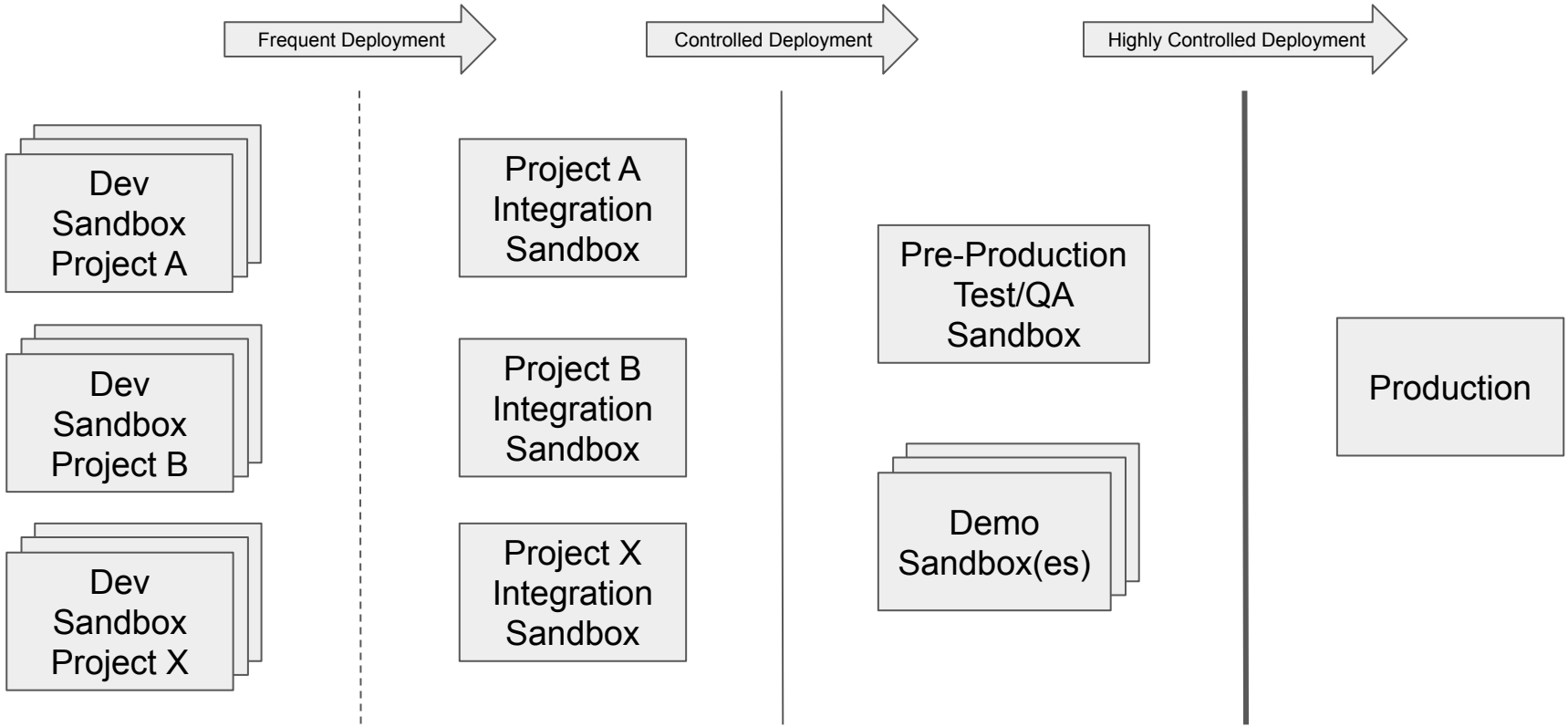


Image Source: https://documentation.mindsphere.io/MindSphere/concepts/concept-api-versioning.html

Benefits of Deploy Scripts

1. Scripts are immutable.
2. If you have too many scripts, you can collapse old ones into one.
3. No need to deploy every script on production.
4. Easy to deploy to as many servers as needed.
5. Any application can easily add a DB structure inside it’s projects of specific version. A story comes…

Sandboxes

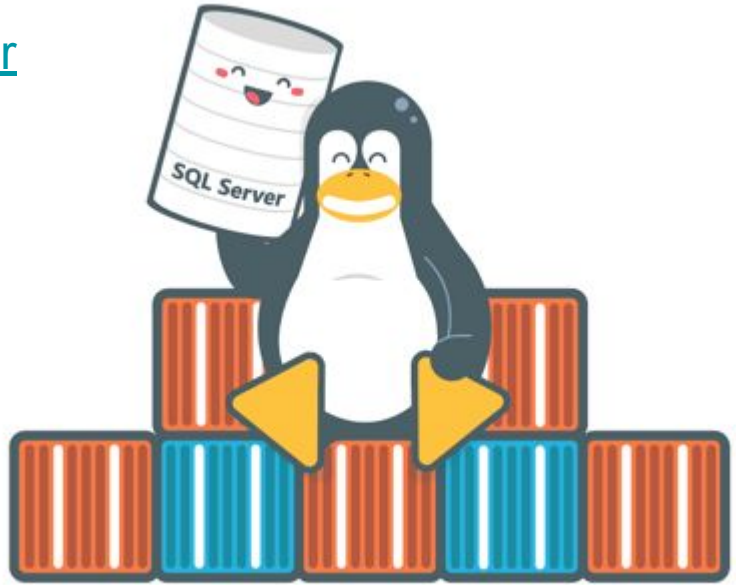

Frequent Deployment
Controlled Deployment
Highly Controlled Deployment
Dev Sandbox Project A
Dev Sandbox Project B
Dev Sandbox Project X
Project A Integration Sandbox
Project B Integration Sandbox
Project X Integration Sandbox
Pre-Production Test/QA Sandbox
Demo Sandbox(es)
Production

Sandboxes

SQL Server in Docker was truly a game changer in software development!

It’s fully customizable.

https://hub.docker.com/_/microsoft-mssql-server

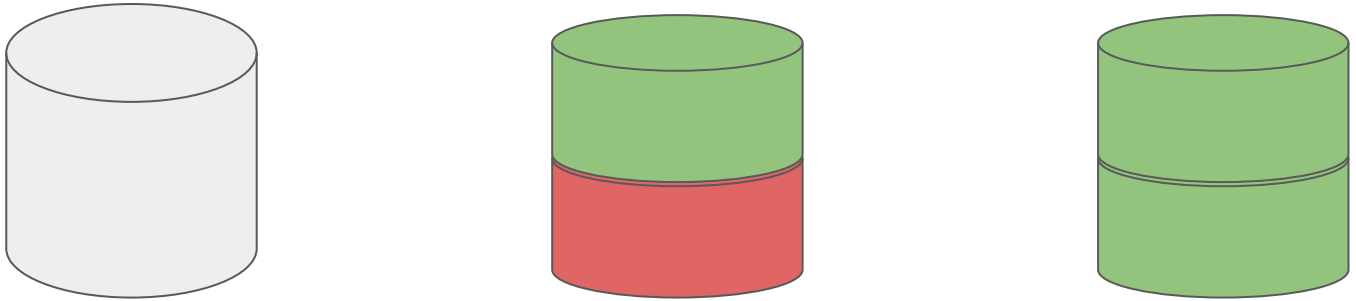


#4 Backup and Recovery

Backups / Restore / HA

Piecemeal Restores

https://learn.microsoft.com/en-us/sql/relational-databases/backup-restore/piecemeal-restores-sql-server?view=sql-server-ver16

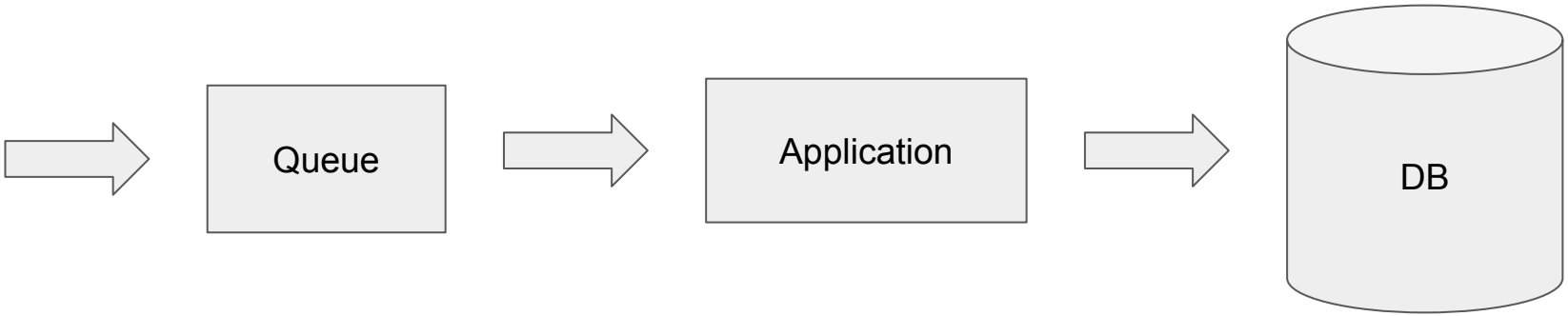


Full Restore

Step 1

Step 2

A Queue Before Database



Does It Look Good?

```
delete from b

from [dbo].[SampleTable] as b

where b.[insert_dt] < '2022-01-01'
```

Long-Running Transactions

```
delete from b

from [dbo].[SampleTable] as b

where b.[insert_dt] < '2022-01-01'
```

Controlled Batch Operations

```
declare @batch_size int = 10000
while 1 = 1
begin
        delete top (@batch_size) from b
        from [dbo].[SampleTable] as b
        where b.[insert_dt] < '2022-01-01'

        if @@rowcount = 0 break
end
```

#5 Move Column

Image Source: https://allegiancemovingandstorage.com/

Move Unused Columns To Separate Table

id	name	description
1	Alex	Some Long Text 1
2	Bob	Some Long Text 2
3	Mary	Some Long Text 3
4	Kate	Some Long Text 4

Imagine: 90% of your queries do not use description column.

What about memory utilization?

Move Unused Columns To Separate Table

id	name
1	Alex
2	Bob
3	Mary
4	Kate

id	description
1	Some Long Text 1
2	Some Long Text 2
3	Some Long Text 3
4	Some Long Text 4

Memory Utilization Improvement

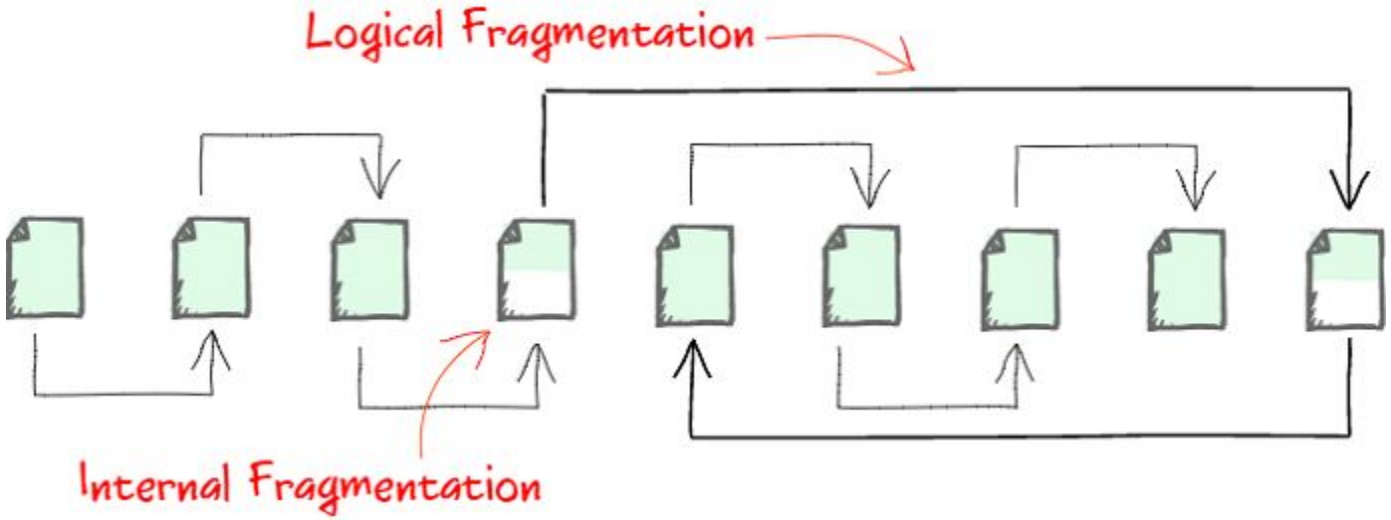
10% → 11.11% **20% → 25%** **30% → 42.86%** **35% → 53.85%**

Add Column → Join Elimination

id	name	has_children
1	Alex	true
2	Bob	false
3	Mary	false
4	Kate	true

~2x performance increase.

Less cache is used for the second table.



#6 Fragmentation

Image Source: https://blog.devart.com/sql-server-index-fragmentation-in-depth.html

Internal Fragmentation

FILLFACTOR

https://learn.microsoft.com/en-us/sql/relational-databases/indexes/specify-fill-factor-for-an-index?view=sql-server-ver16

Page fill **70%** → **90%**, memory utilization increase **28.57%**

Page fill **60%** → **90%**, memory utilization increase **50%**

Page fill **55%** → **80%**, memory utilization increase **45.45%**

External Fragmentation

There is one simple solution here…

#7 Data Synchronization

Data Migration / Synchronization Techniques

1. Triggers (on tables, on temp views)
2. Bulk Load
3. Bulk Load + Replay Recent Events
4. Parallel Run
5. Unified Data Load & Processing
    a. Low Priority Queue for Bulk Load
    b. Ability to overwrite existing data
6. Problems With Delete (ghost records, use PAGLOCK hint)

DELETE WITH PAGLOCK

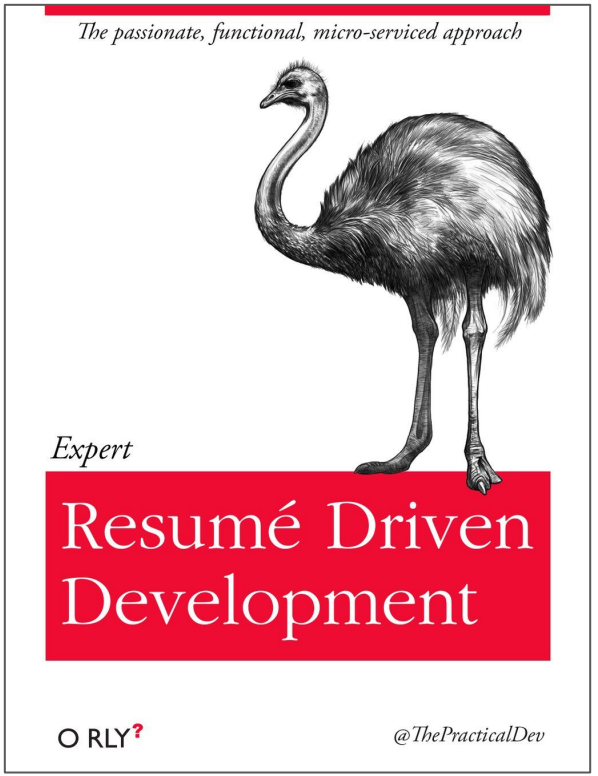
```
declare @batch_size int = 10000
while 1 = 1
begin
	delete top (@batch_size) from b with (paglock)
	from [dbo].[SampleTable] as b
	where b.[insert_dt] < '2022-01-01'

	if @@rowcount = 0 break
end
```

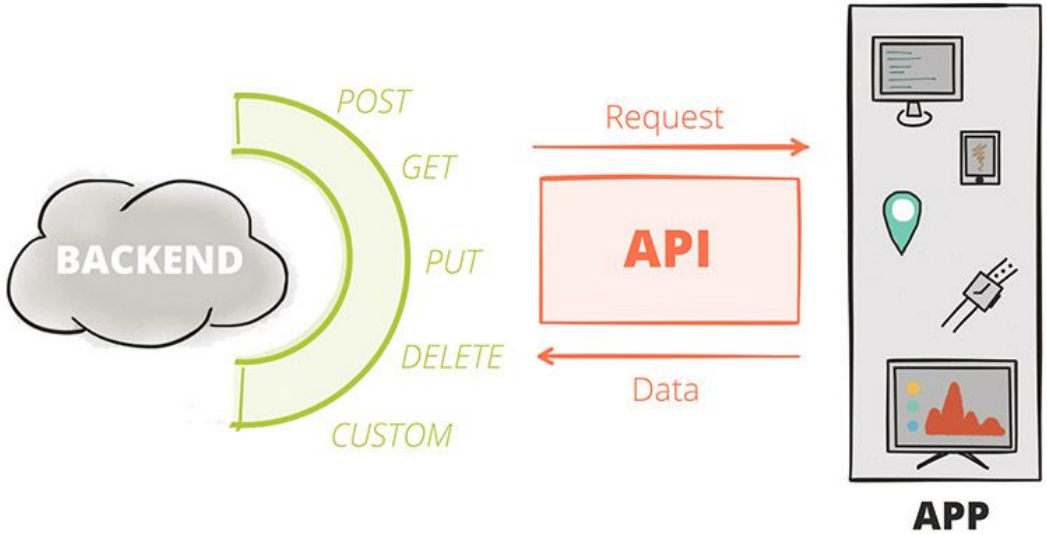
#8 Stale Data

Image Source: https://www.cioinsight.com/it-strategy/does-data-collection-waste-time-and-resources/

Remove Stale Data & Structures

1. Unused Data
2. Unused Indexes
3. Unused Columns
4. Unused Tables



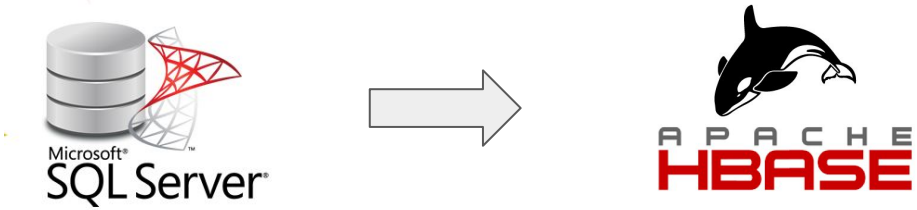


#9 Application API

Image Source: https://lvivity.com/what-is-an-api-and-how-does-it-work

External API Pros

1. Lookup tables in application (periodically refresh, no extra joins).
2. Benefit from external cache.
3. Not necessarily and application API.
   a. Example: separate DB + Service Broker.
4. Easy to change DB engine.
5. Turn your database into a single application database.
   a. Easier internal refactoring.
   b. Split database.
   c. Easy to change DB engine.



#10 Data Integrity

Image Source: https://blazent.com/three-key-requirements-achieve-data-integrity/

Constraints

Foreign key constraints reduce performance within your database because the existence of the row in the foreign table will be verified whenever the source row is updated.

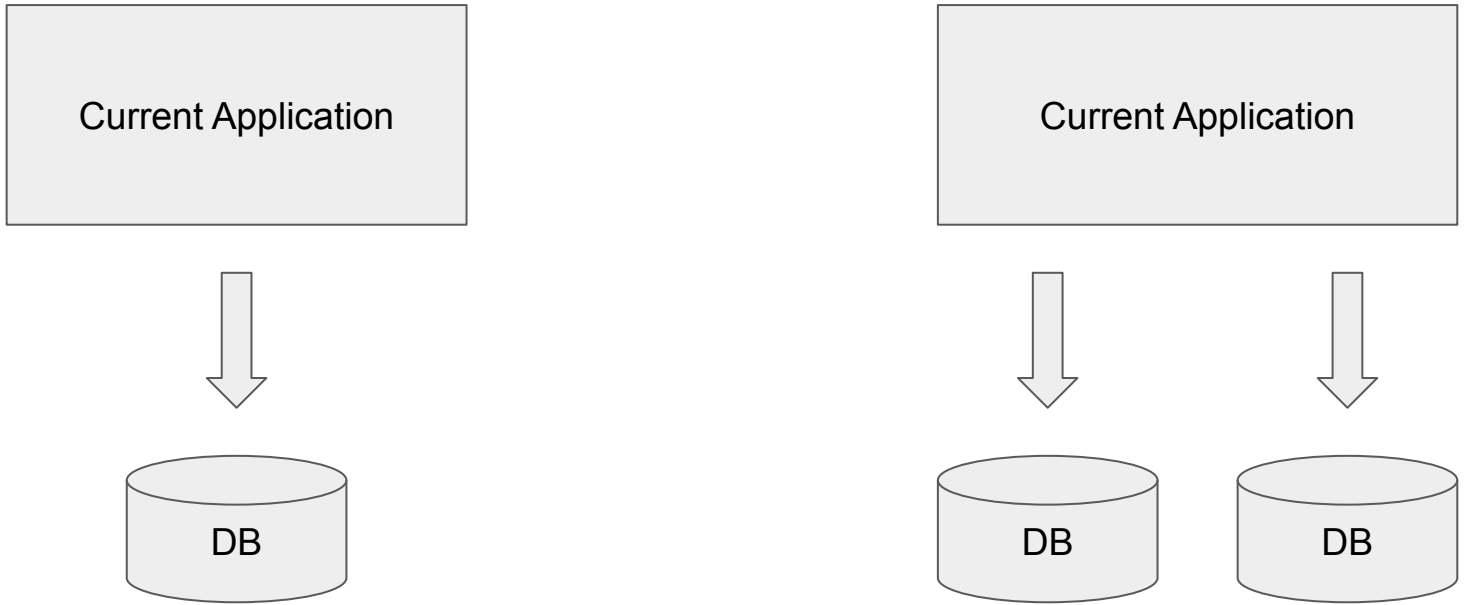
If you drop foreign keys, you'll need better testing of your code and (probably) some regular external data checks.

Check constraints are good.

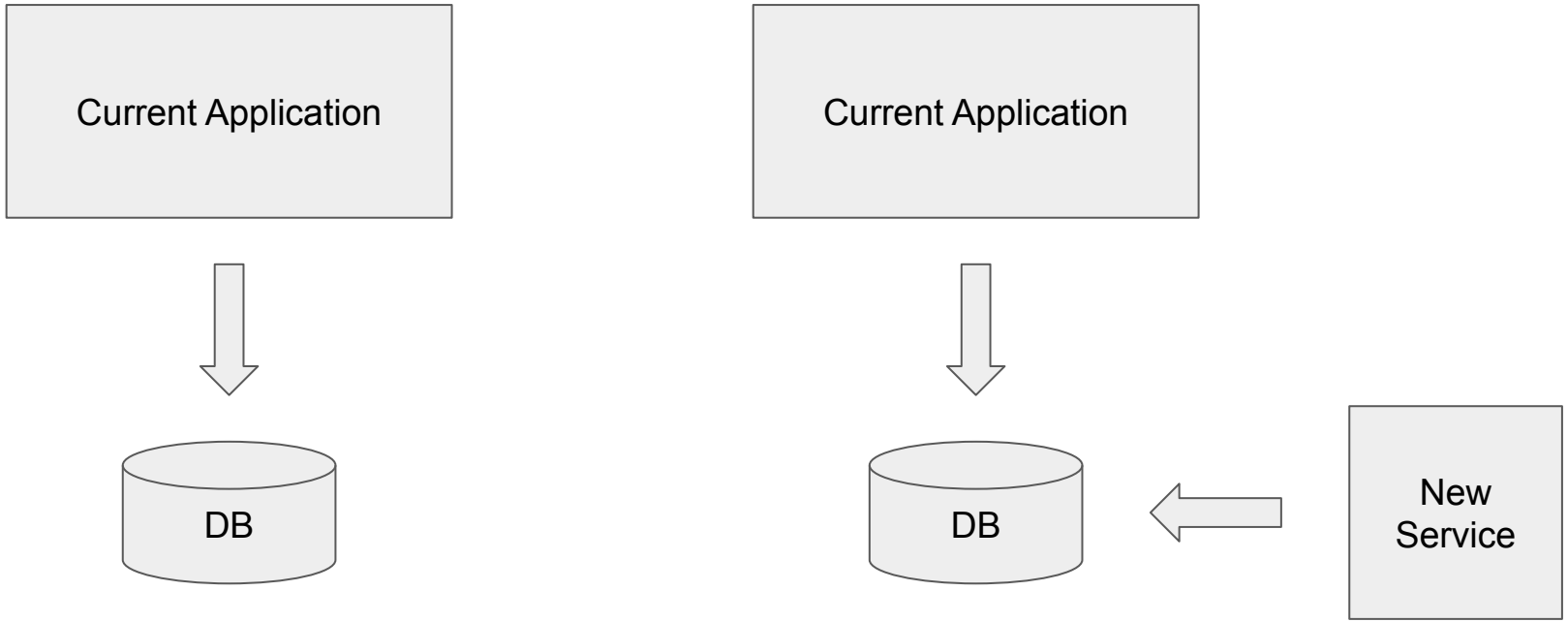
#11 Split Database

Image Source: https://diygarden.co.uk/log-splitters/best-log-splitting-axe

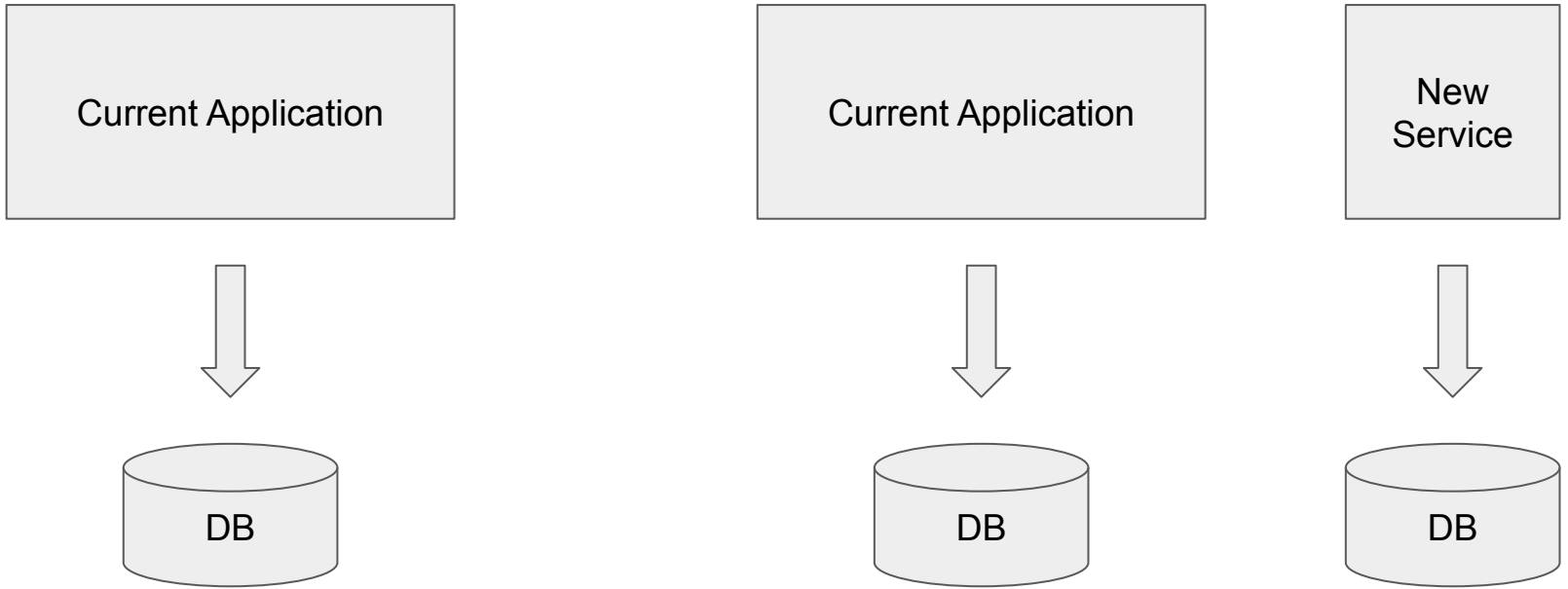
Split Database First



Split Code First



Split Database and Code Together



Strangler Fig

Incrementally migrate a legacy system by gradually replacing specific pieces of functionality with new applications and services. As features from the legacy system are replaced, the new system eventually replaces all of the old system's features, strangling the old system and allowing you to decommission it.

https://learn.microsoft.com/en-us/azure/architecture/patterns/strangler-fig

Strangler Fig

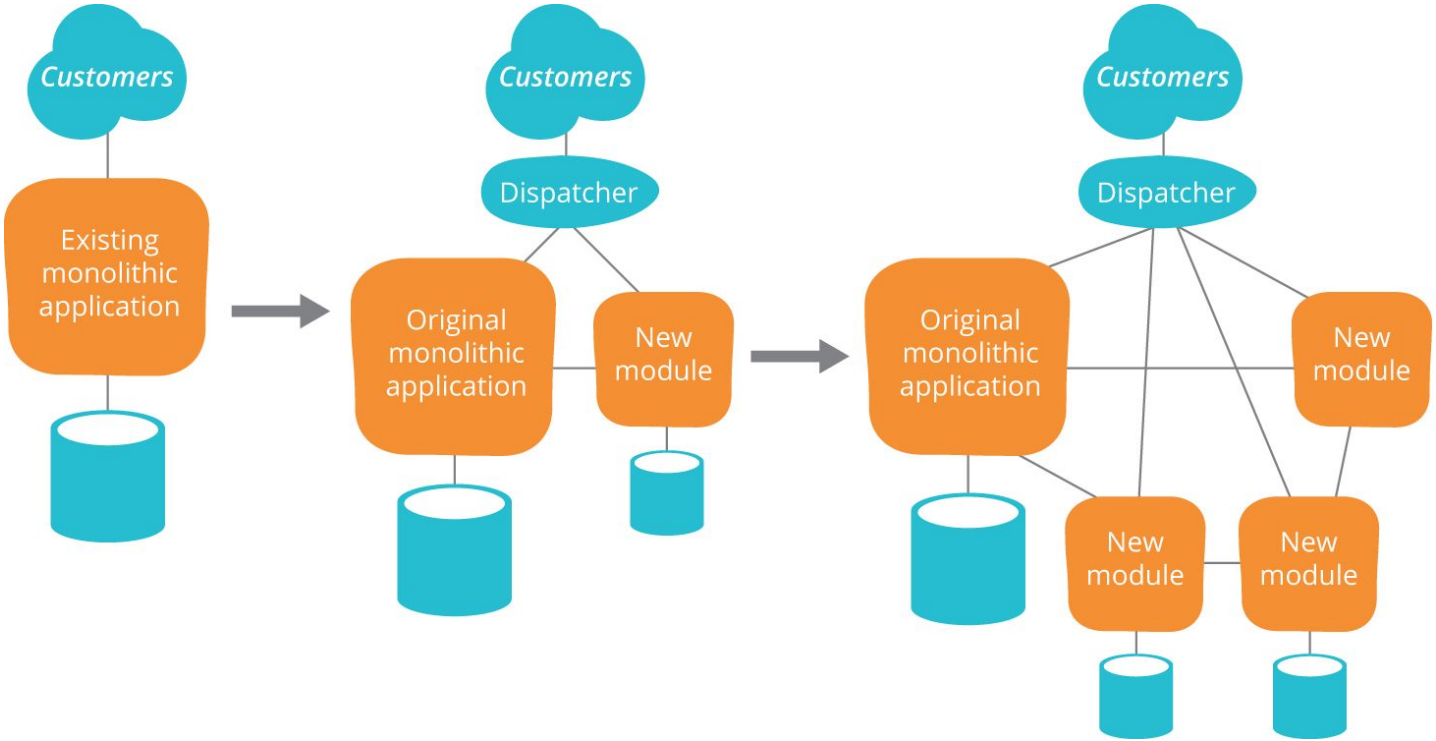
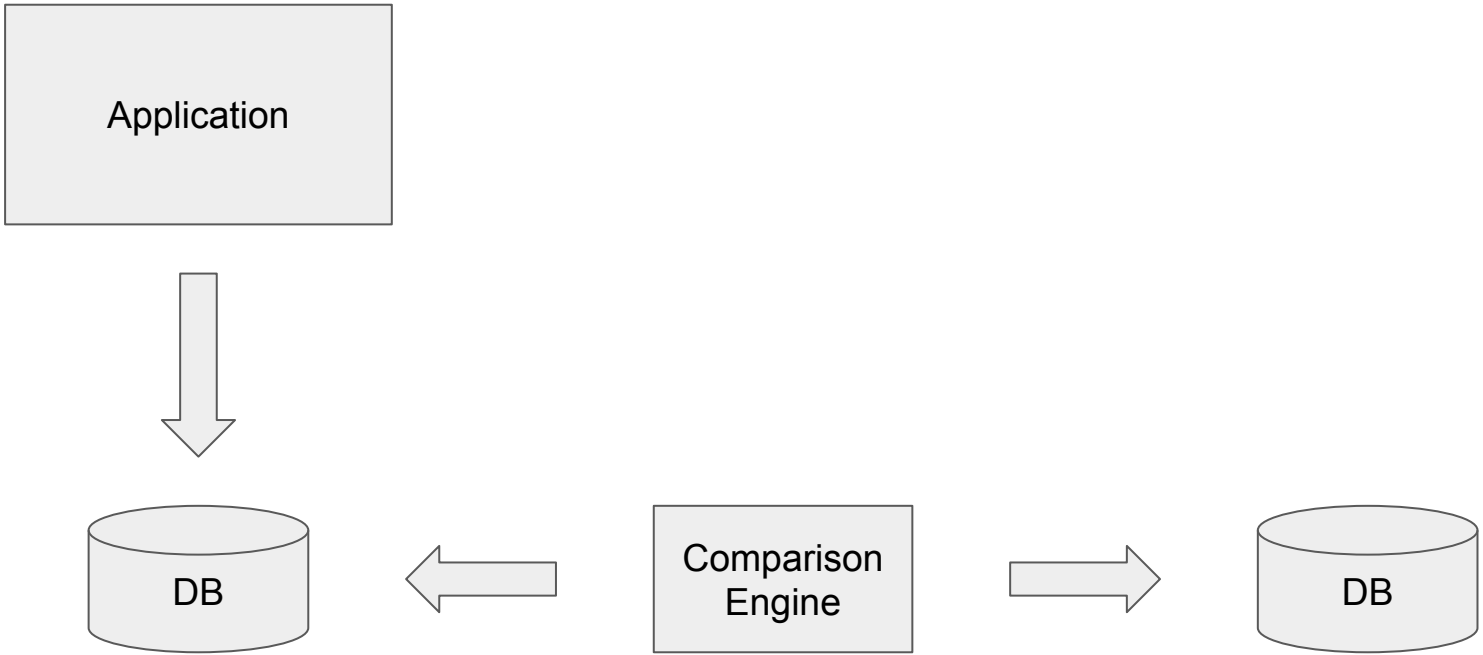
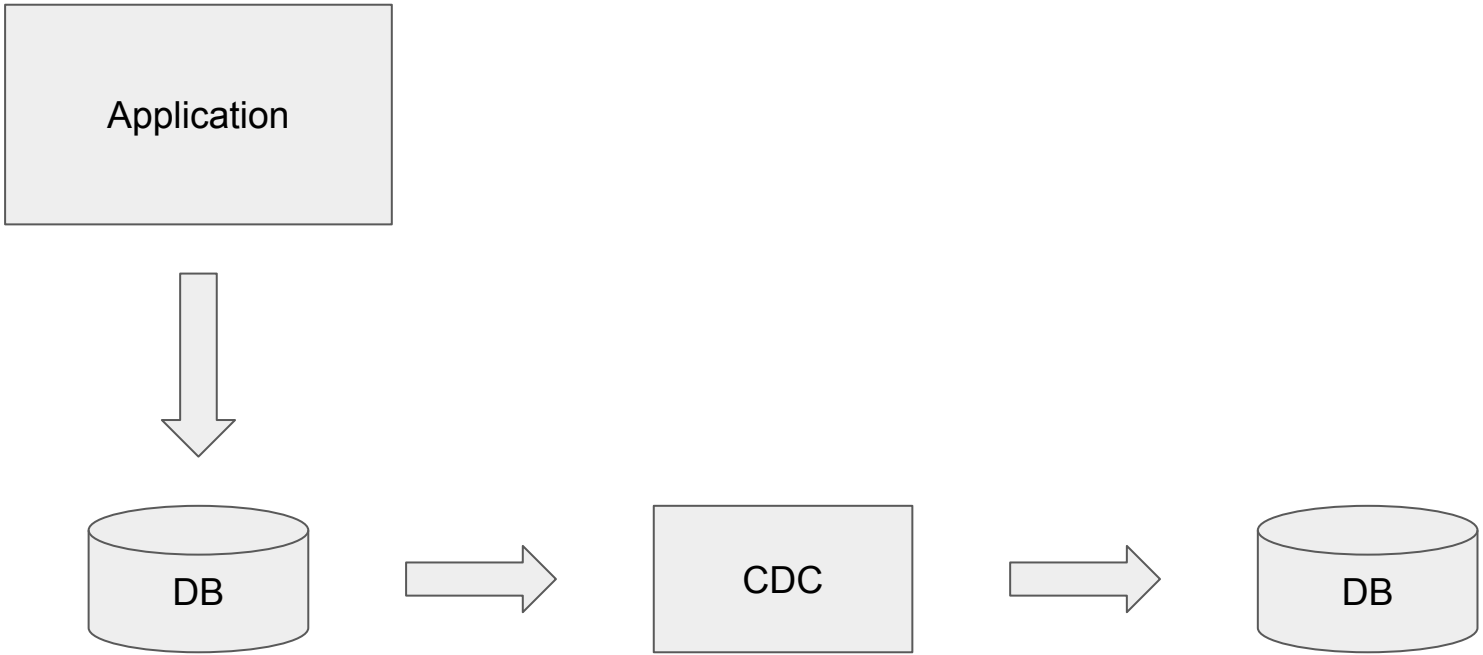


Image Source: http://mycloudcomputing2017.blogspot.com/2017/02/what-is-strangler-application-pattern.html

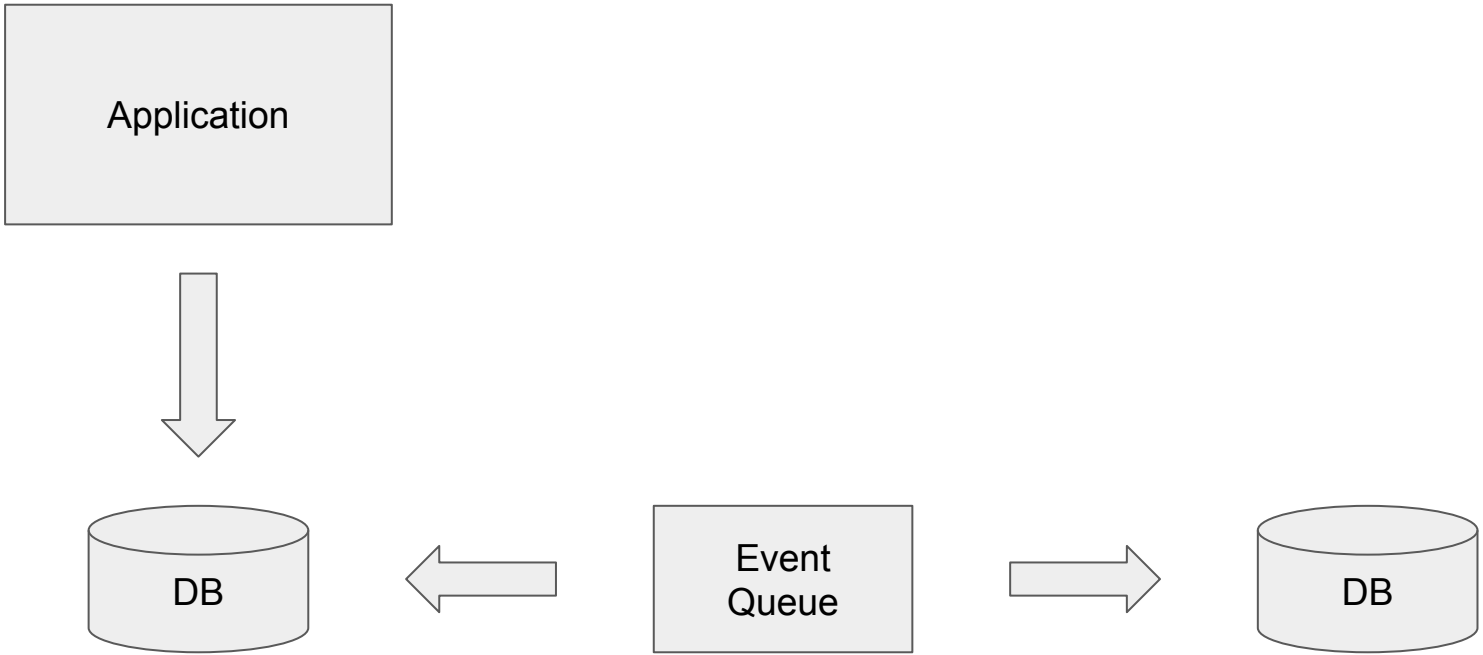
Parallel Run



Change Data Capture



Event Queue



#1 Story: Optimization

Image Source: https://itknowledgezone.com/top-it-cost-optimization-techniques/

Replace One-To-Many With Associative Table

user_id	name
1	Alessandro
2	Luigi
3	Isabella

user_id	pizza_id
1	1
1	2
2	2
3	1
3	2
3	3

pizza_id	name
1	Margherita
2	Pepperoni
3	Quattro Formaggi
4	Parma

Replace One-To-Many With Associative Table

user_id	name
1	Alessandro
2	Luigi
3	Isabella

set_id	pizzas
1	[1, 2]
2	[2]
3	[1, 2, 3]

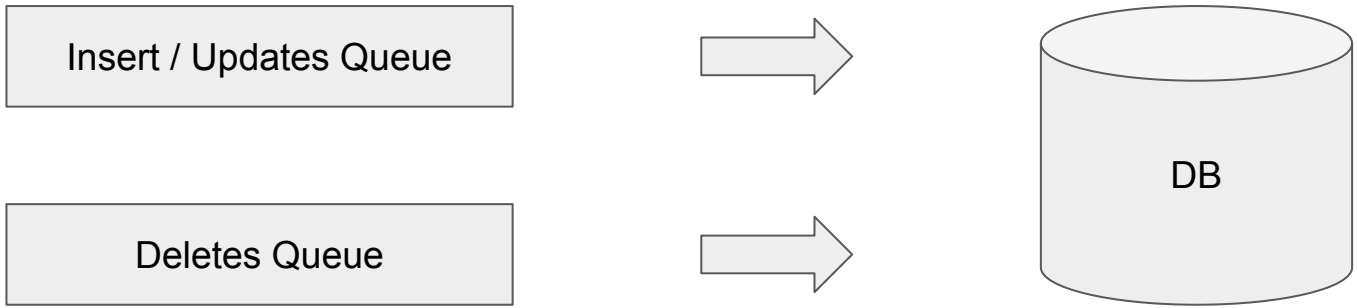
pizza_id	name
1	Margherita
2	Pepperoni
3	Quattro Formaggi
4	Parma

user_id	pizza_set_id
1	1
2	2
3	3

#2 Story: Hard vs. Soft Deletes

Image Source: https://www.starecta.com/hard-vs-soft-approach/

Given…



#3 Story: Plan In Advance

Image Source: https://valley.d51schools.org/counseling/life_after_valley/plan_ahead_

Given…

DB ~3 TB

Need to predict growth and plan resources.

3 tables each at about 600-800 GB. Fragmentation ~60-70%.

Fillfactor 100%. Free space in DB ~400GB.

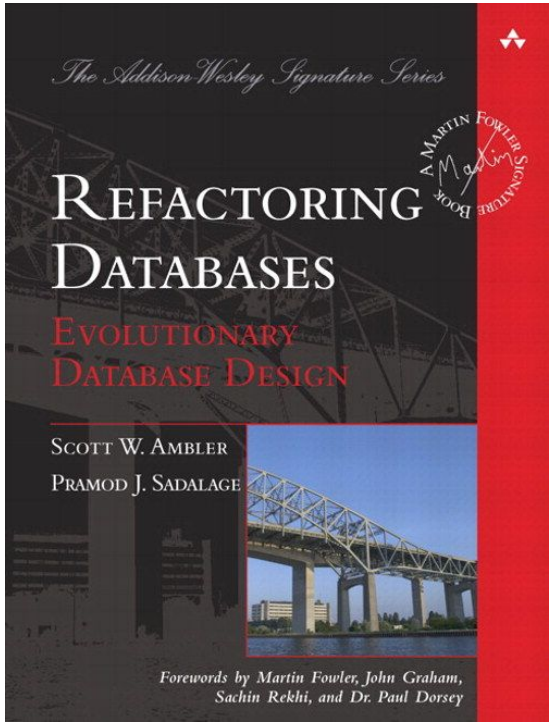
Index reorganize? Index Rebuild? Rebuild by partition?

Additional Resources

Refactoring Databases: Evolutionary Database Design

Scott W. Ambler (Author), Pramod J. Sadalage (Author), Martin Fowler (Foreword), John Graham (Foreword), Sachin Rekhi (Foreword), Paul Dorsey (Foreword)

https://www.amazon.com/gp/product/0321293533/



DATA SATURDAYS

engage LABS