

Analysis of the Behaviour of Online Transactional Relational Schema Transformations

Lesley Wevers

Menno Tammens

Matthijs Hofstra

Marieke Huisman

Maurice van Keulen

University of Twente

FMT lunchmeeting, 25 November 2014

Motivation

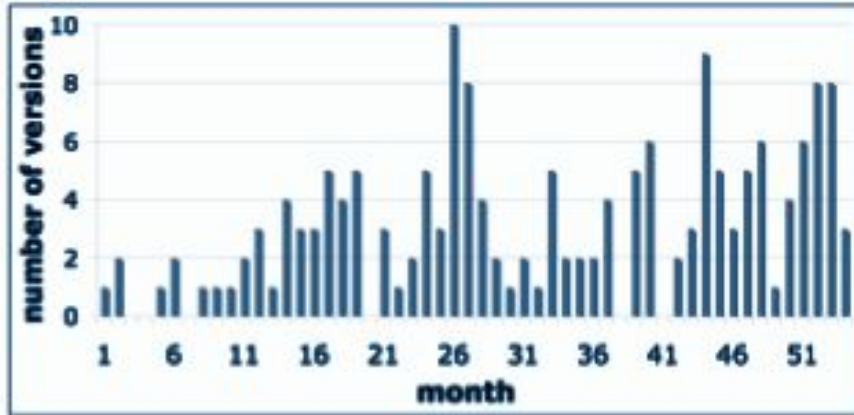
We often need to perform schema changes:

- Feature changes
- Improve performance

But, current database systems block concurrent transactions when transforming data!

Motivation

WikiMedia schema revisions:



- 90% require a write lock.
- Largest took 22 hours to complete for wikipedia.

Problem

Some systems can not go offline:

- Telecom, payment, airline reservation, online services

Ad-hoc solutions are insufficient:

- Faster hardware: Not scalable
- Splitting transformations: Non-transactional
- Lazy transformation: Difficult to get correct

The DBMS should solve this!

Goal

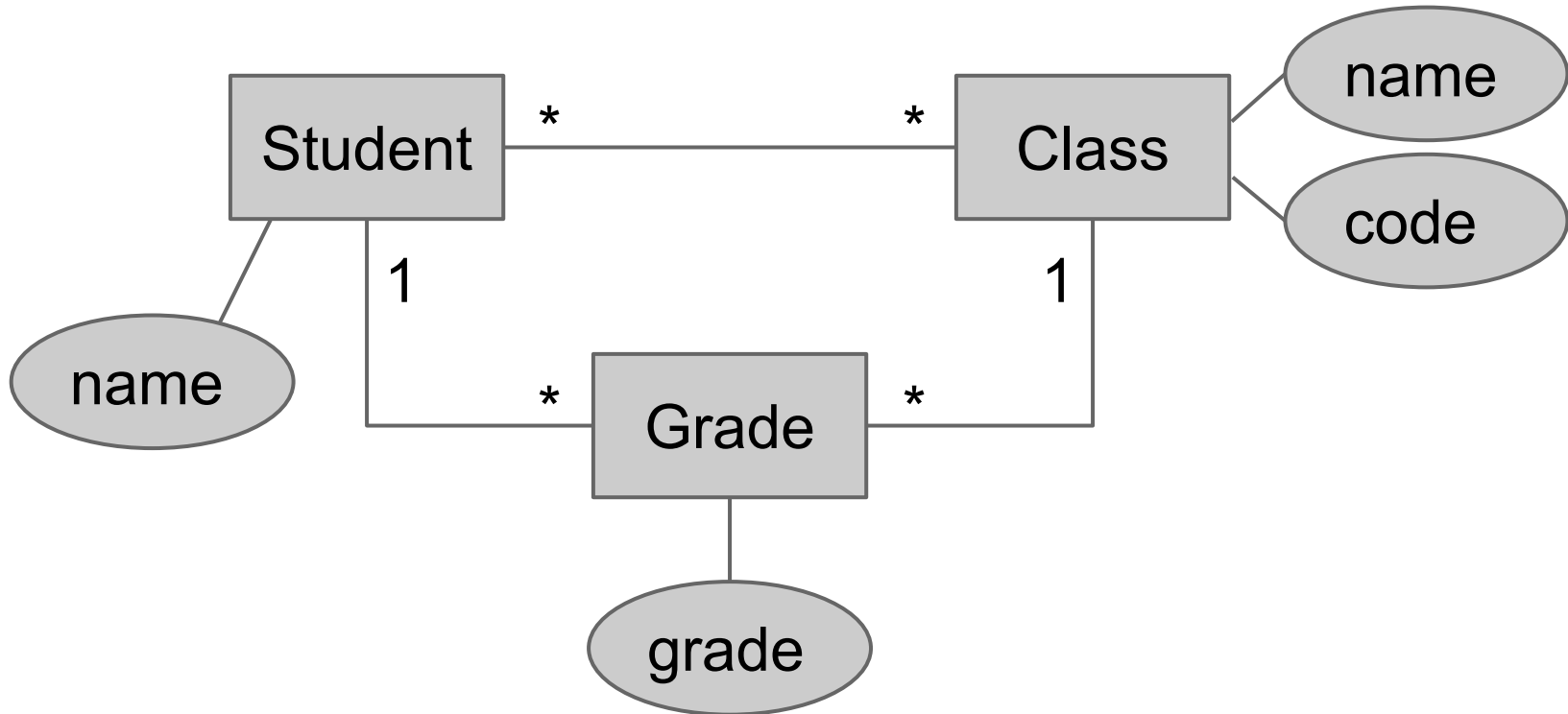
Drawing attention to the problem by investigating the extent of the problem in current database systems:

- **PostgreSQL**: no support for online schema changes.
- **MySQL**: claims to support online schema changes.
- **pt-online-schema-change**: representative for trigger based online schema transformation tools.

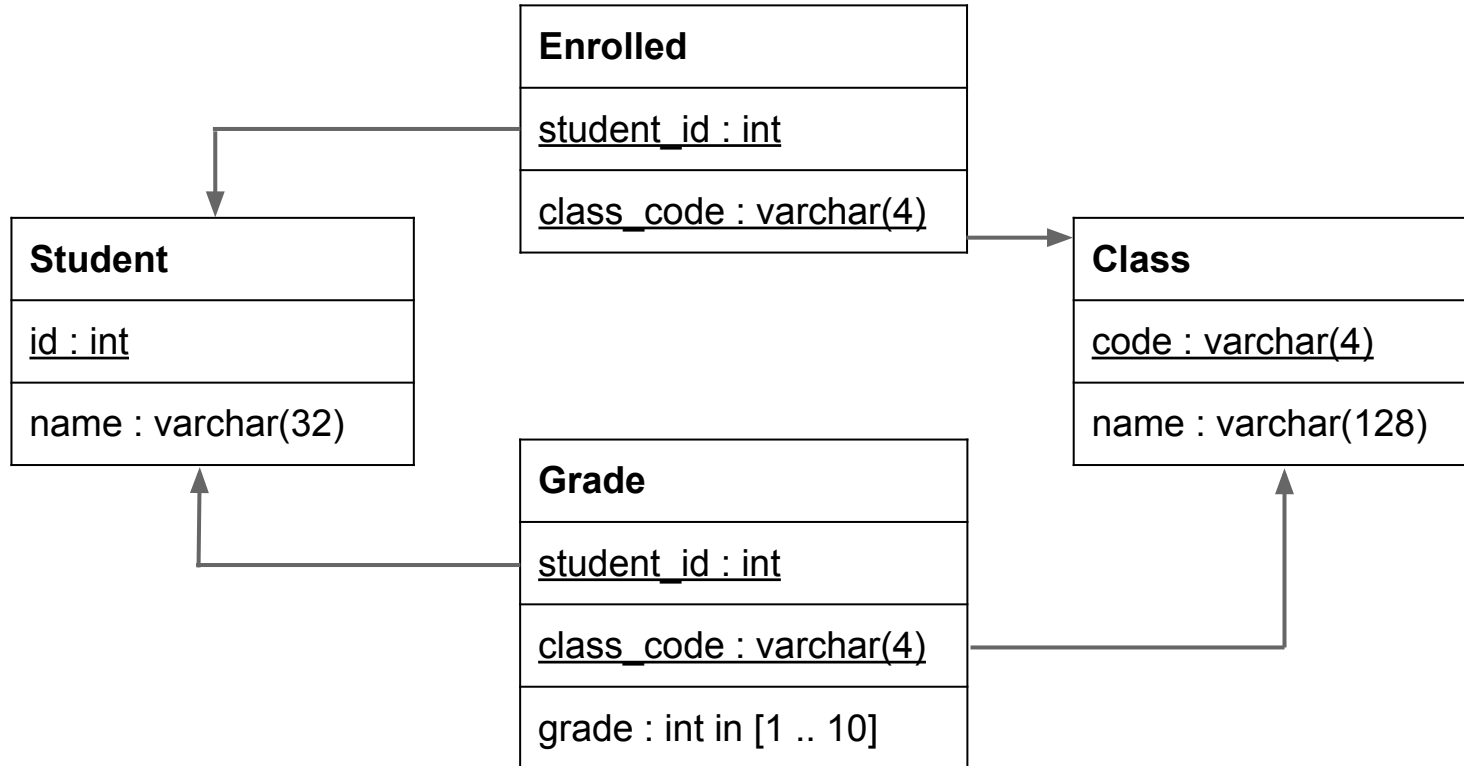
Contributions

- Criteria for online schema change mechanisms.
- Experimental investigation of existing systems.
- Proposal for a more fundamental solution.

Entity-Relationship Model



Relational Implementation



Functional Criteria

Schema transformation mechanisms must:

- Allow simple and complex:
 - ER-level transformations
 - Implementation transformations
- Provide data in new schema upon commit
- Have transactional semantics:
 - Serializable
 - Failure atomic
 - Composable
 - Support for upgrading applications that use the database

Performance Criteria

Impact on concurrent transactions:

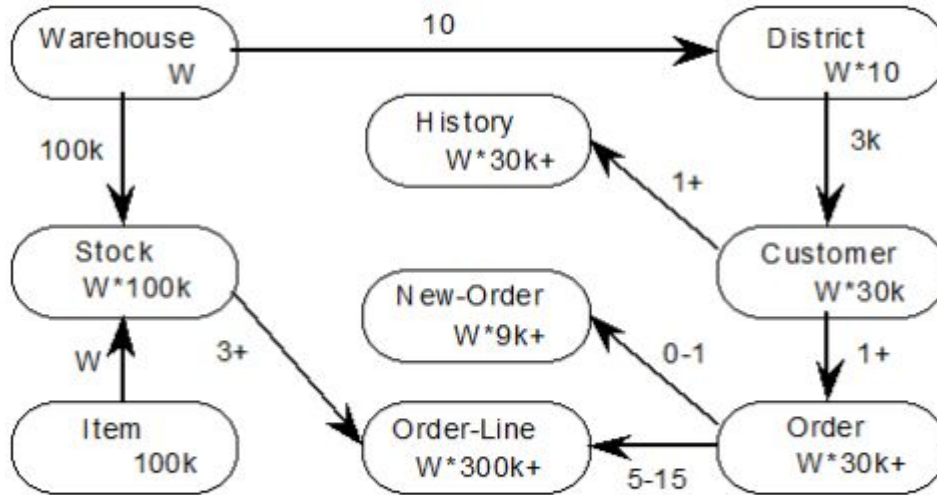
- Blocking
- Aborts
- Slowdown

Performance of schema transformations:

- No aborts
- Time to commit

Experimental Setup

TPC-C



New order

Payment

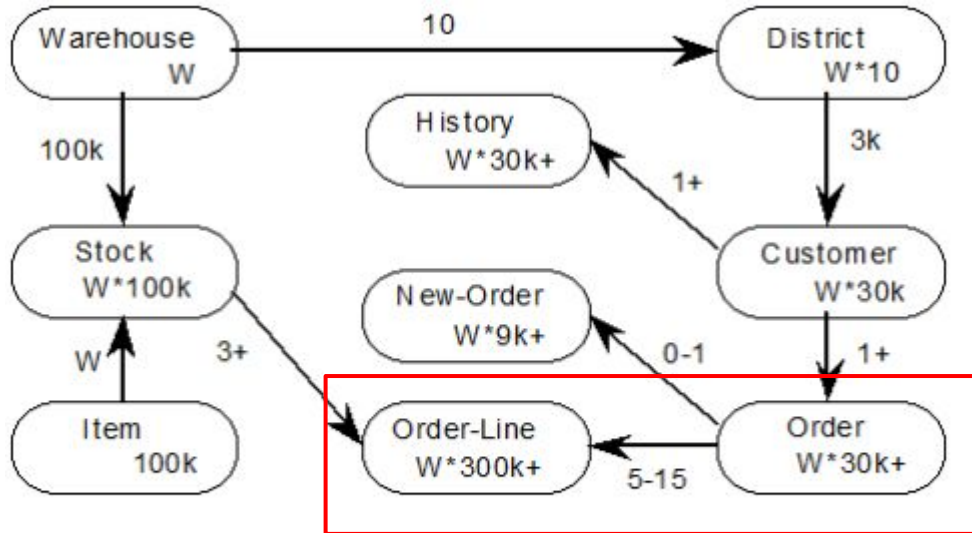
Order status

Delivery

Stock level

Experimental Setup

TPC-C



New order

Payment

Order status

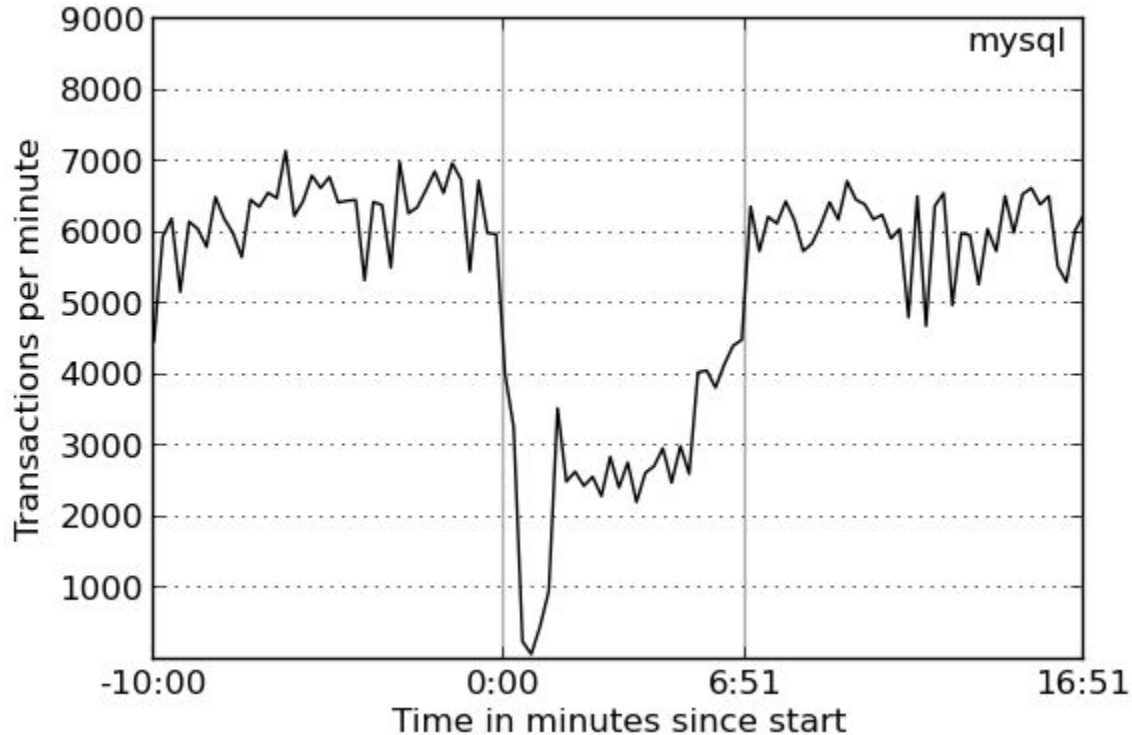
Delivery

Stock level

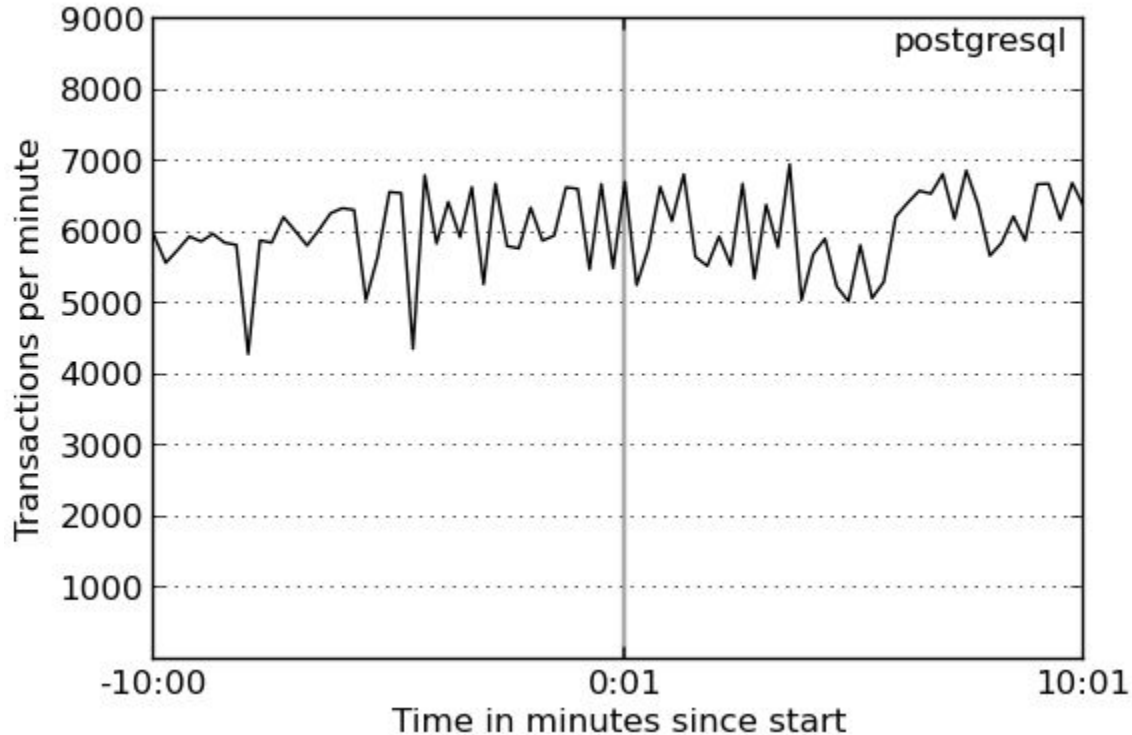
Experiment 1

Add a column to an existing table.

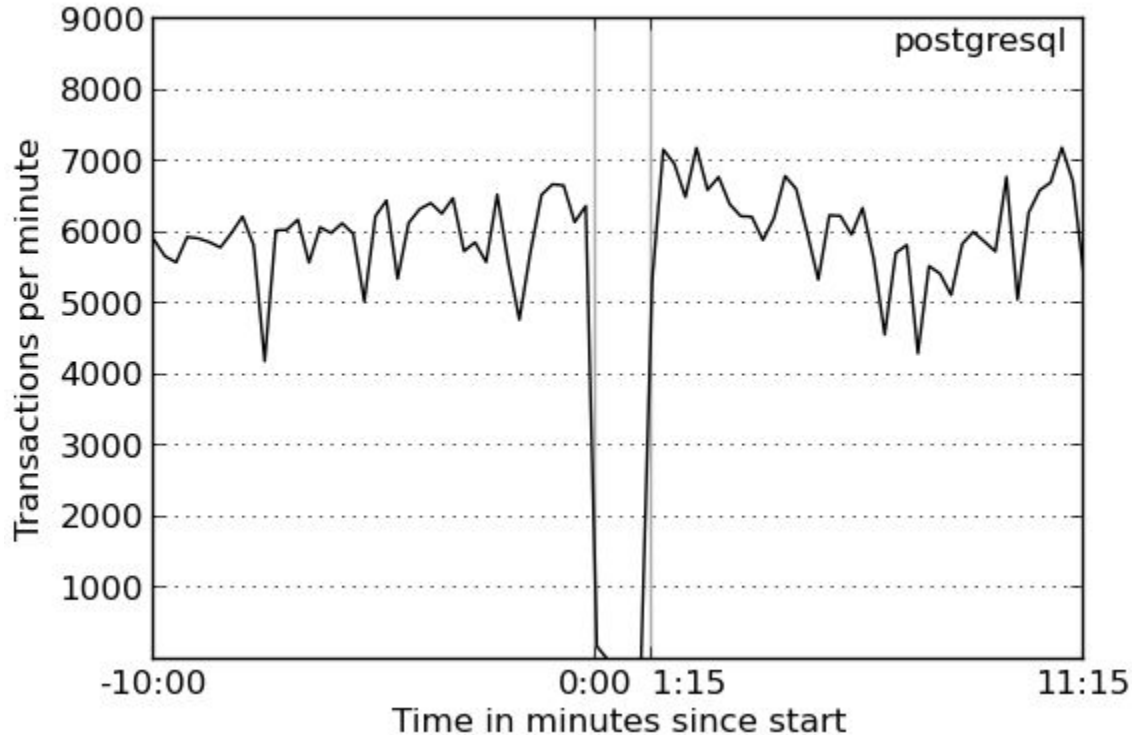
Results: Add Column



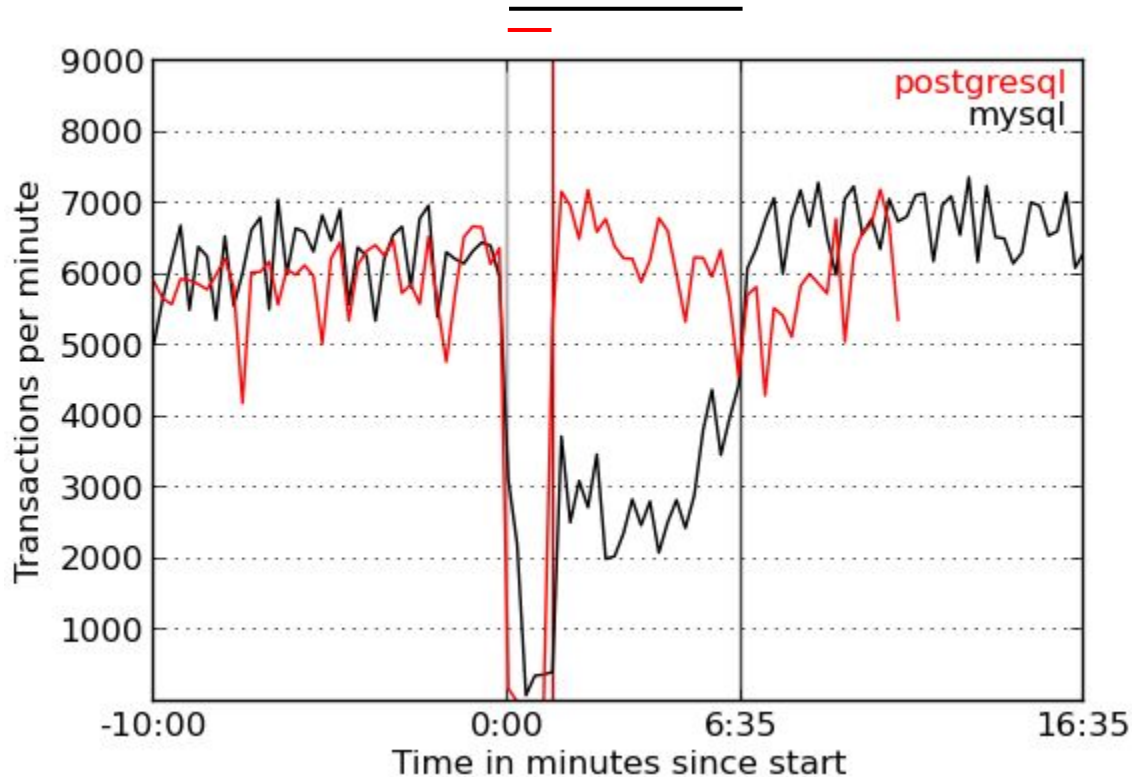
Results: Add Column



Results: Add Column with Default



Result: Add Column

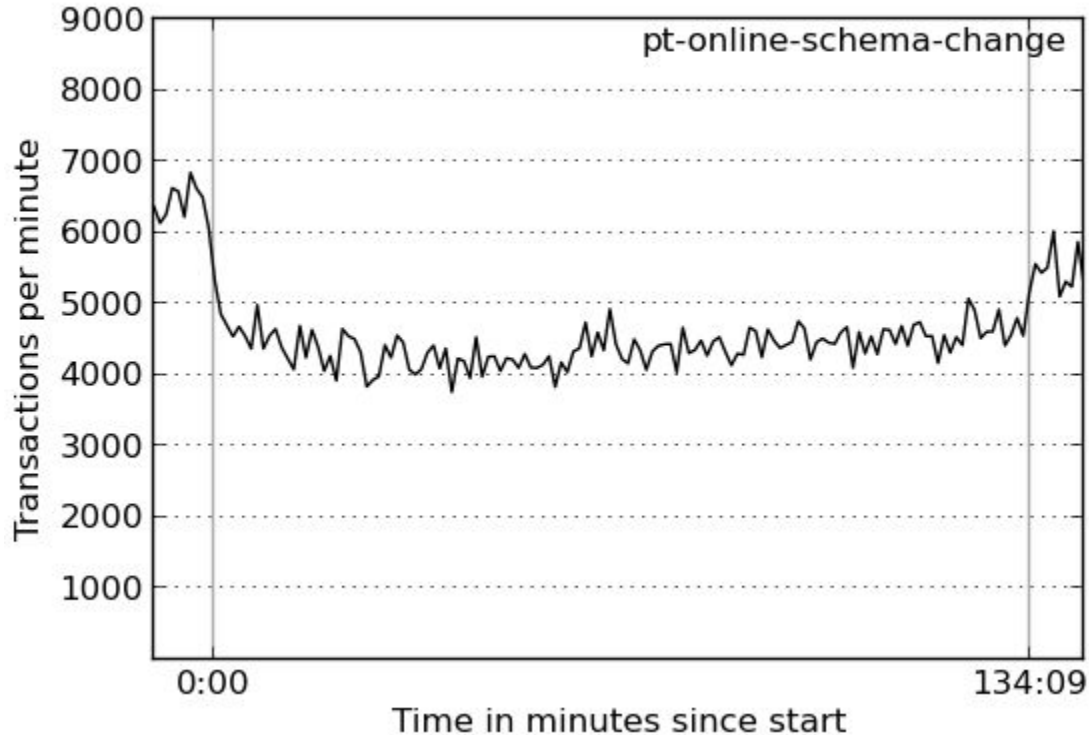


pt-online-schema-change

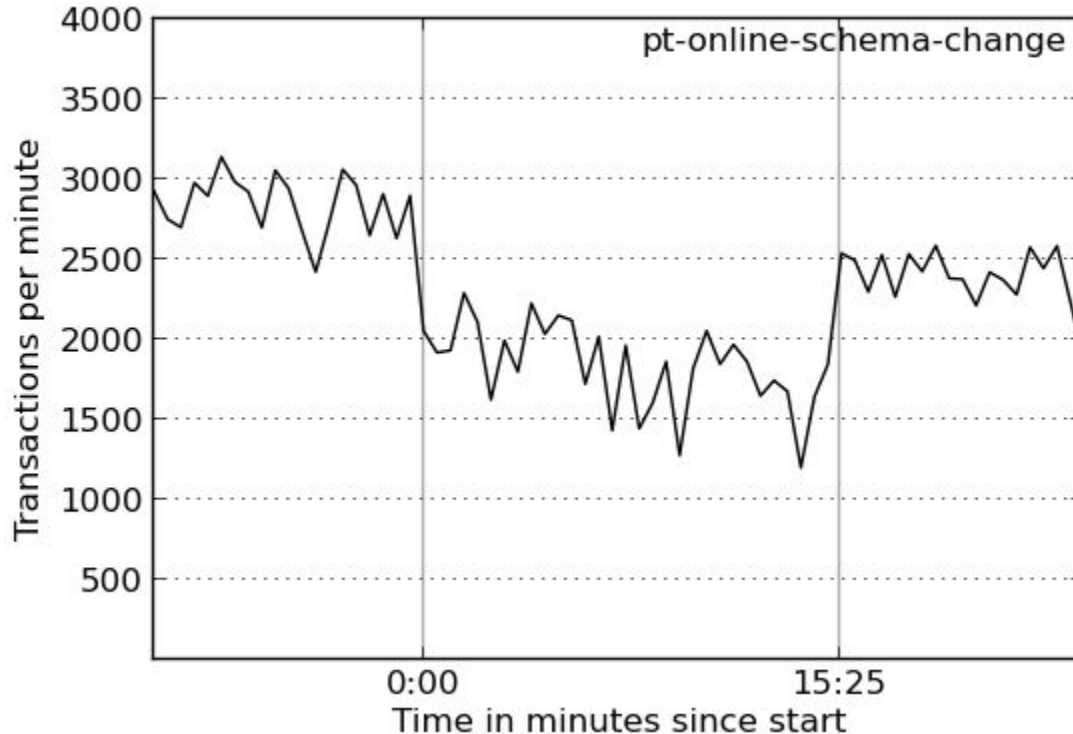
Trigger based transformations:

- Create a table in the new schema
- Copy data to the new table in chunks
- Database triggers are used to synchronize updated on the old table to the new table.

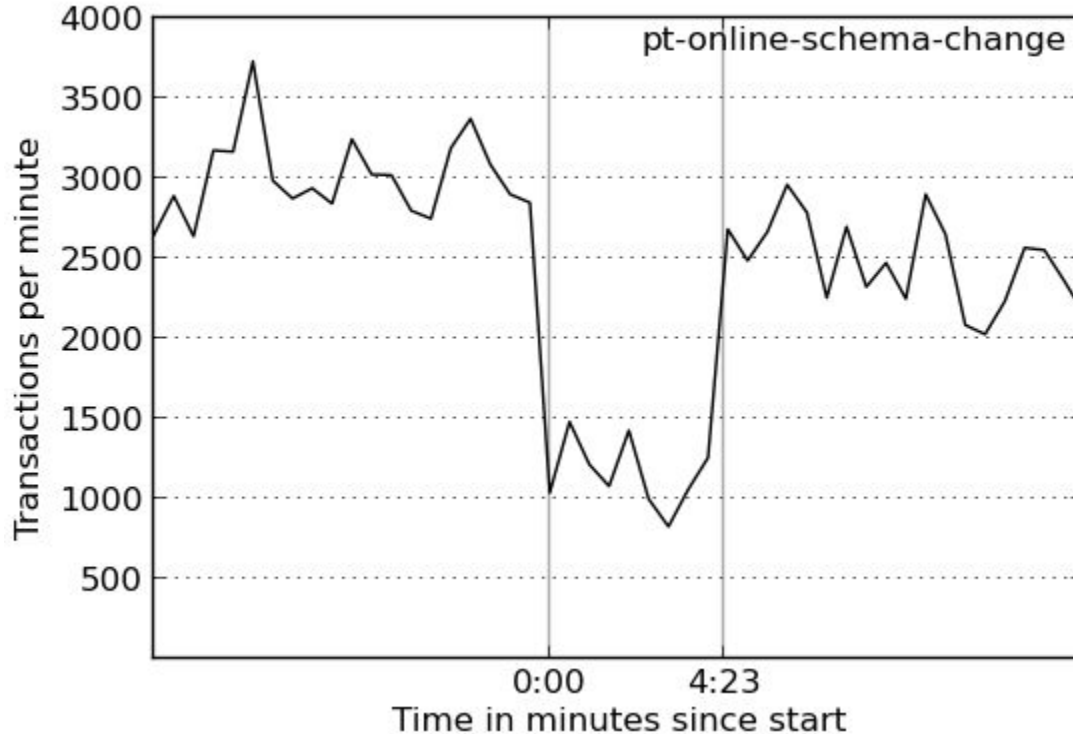
Result: Add Column with Default



Result: Add Column with Default



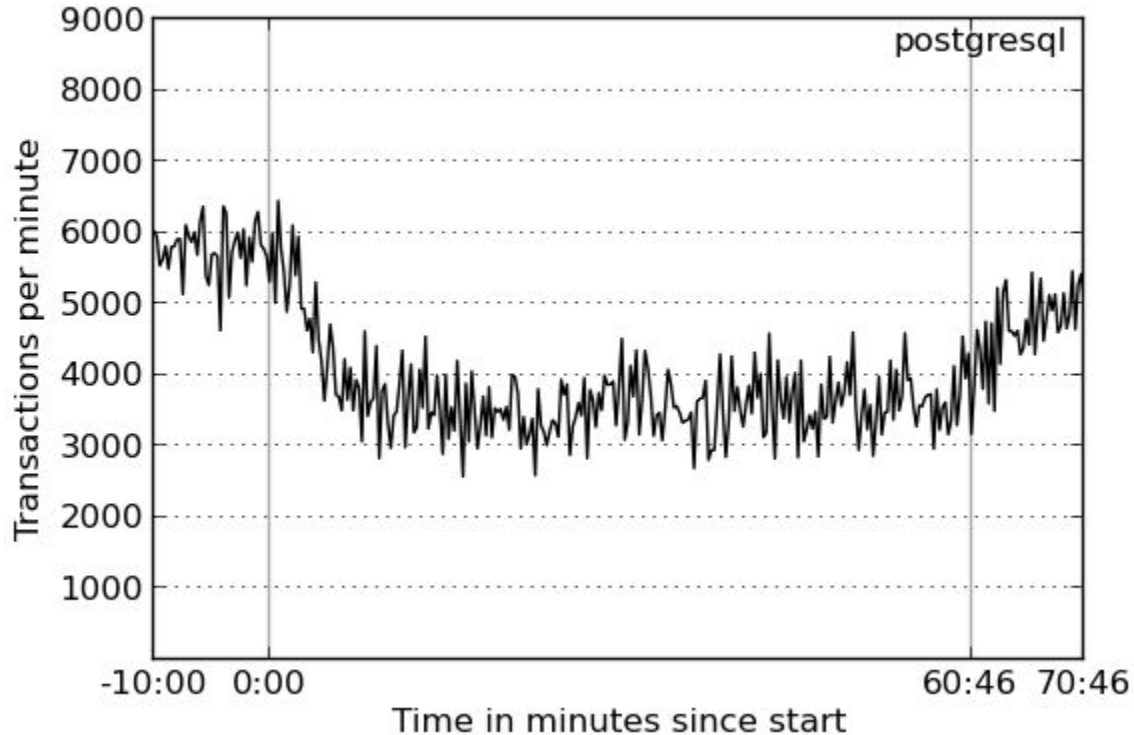
Result: Add Column with Default



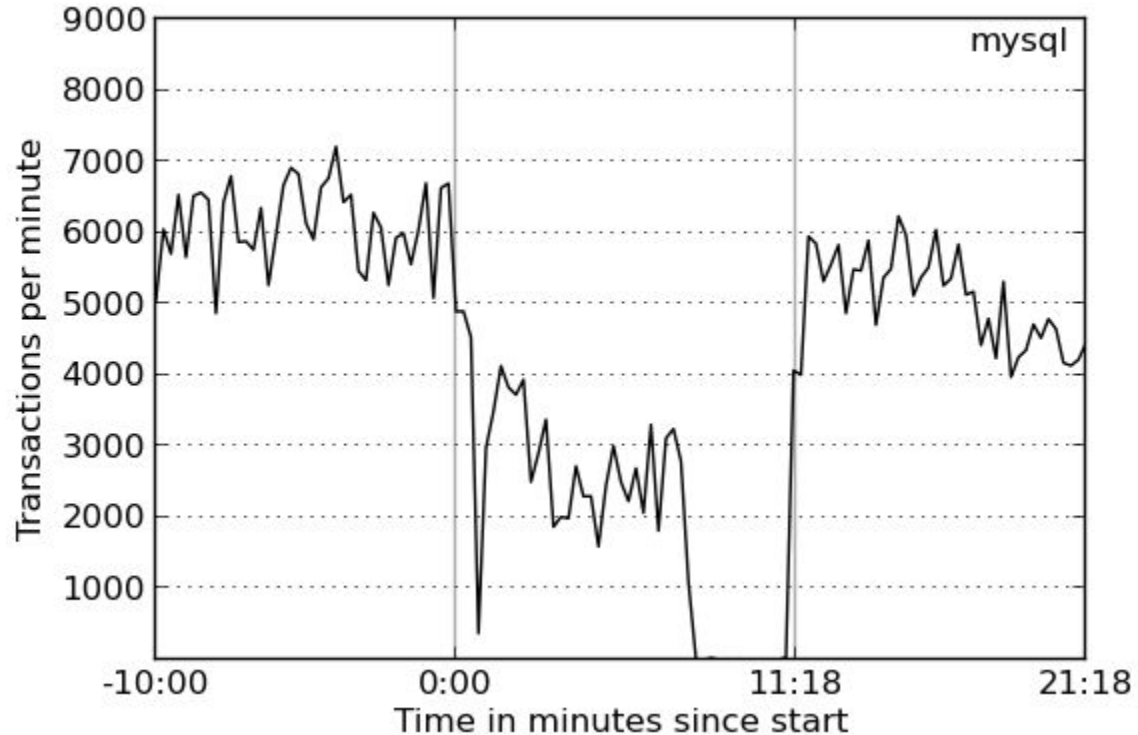
Experiment 2

Create an index.

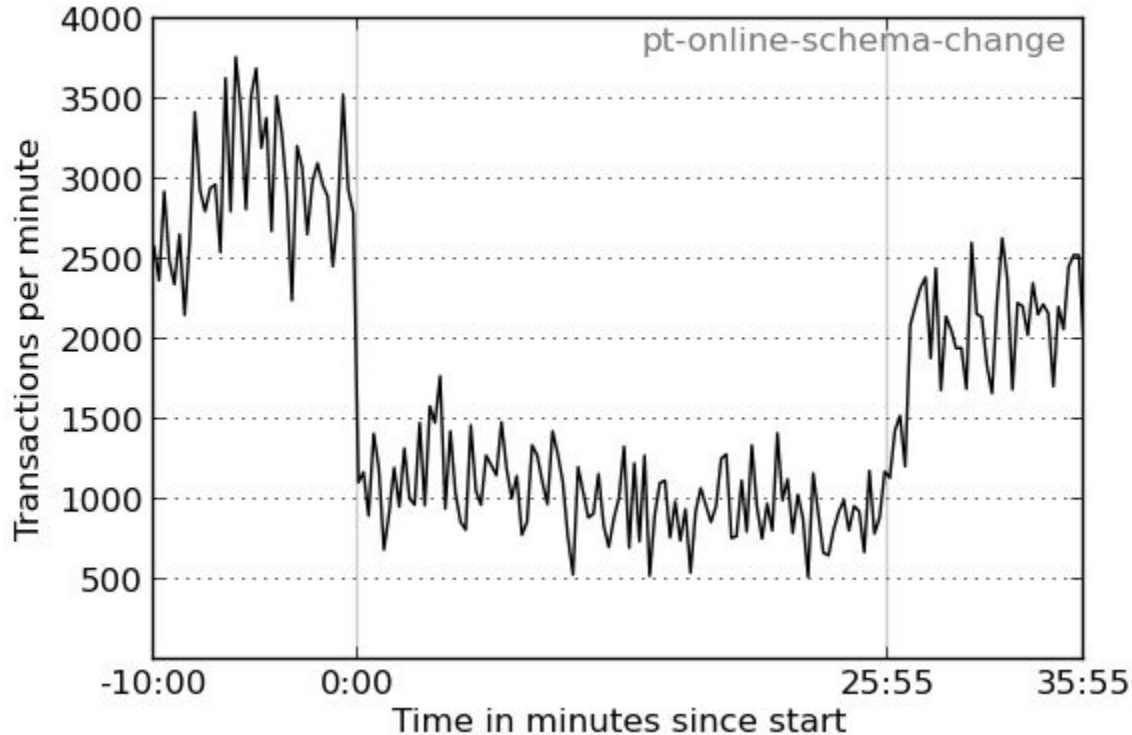
Results: Create Index



Results: Create Index



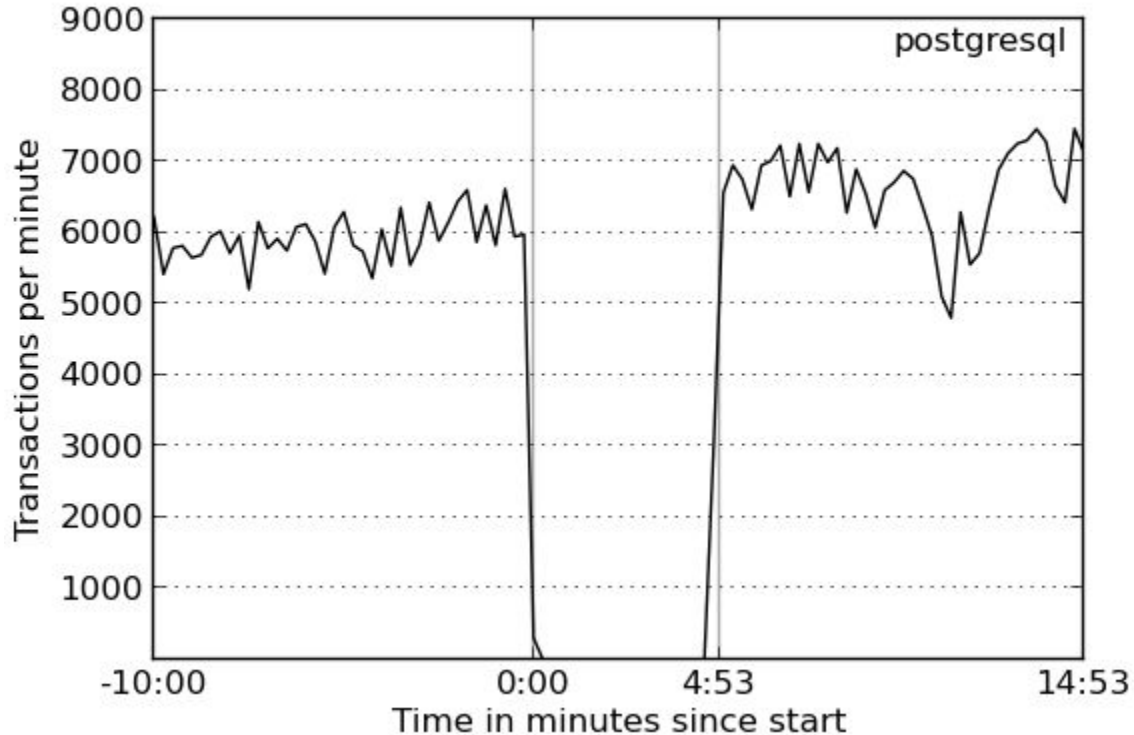
Results: Create Index



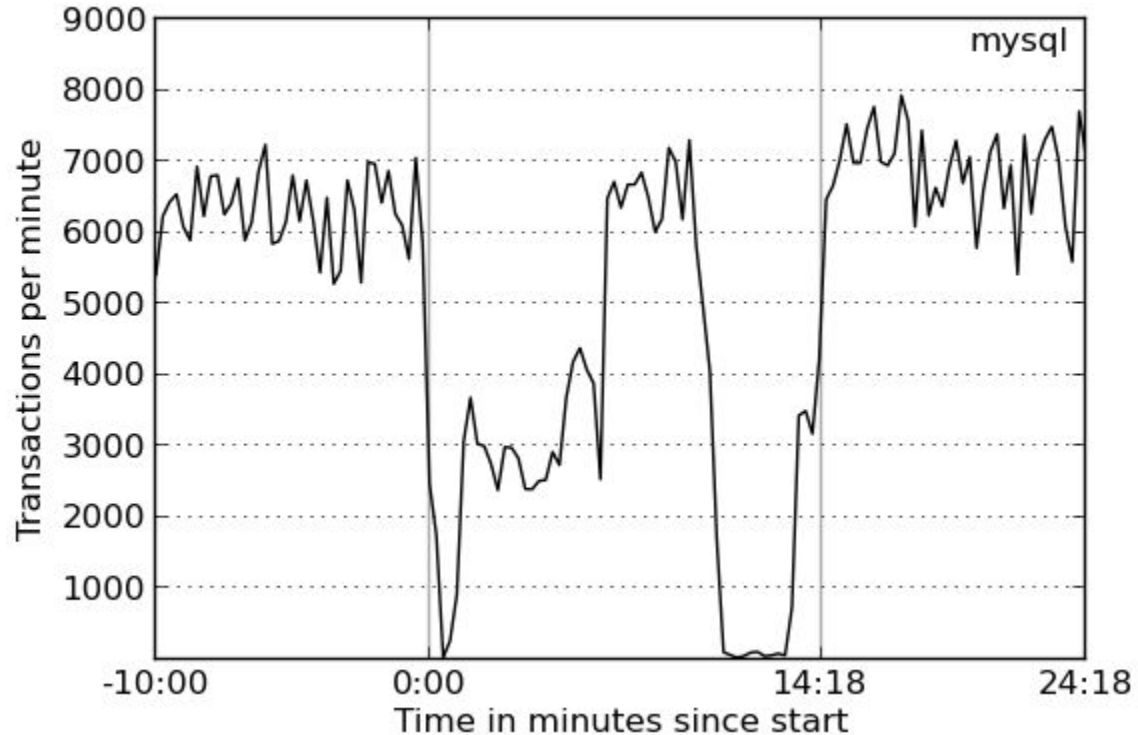
Experiment 3

Instead of a single carrier per ORDER, make every ORDER-LINE have a different carrier.

Results: Move Attribute



Results: Move Attribute



Results: Move Attribute

pt-online-schema-change:

No support for complex transformations

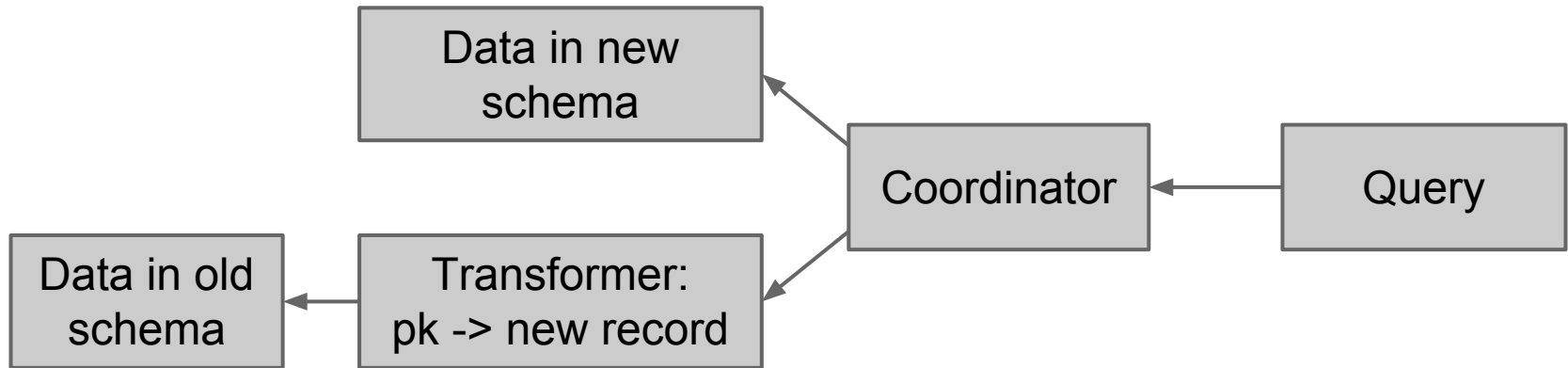
Experimental Results

	PostgreSQL	MySQL	pt-osc
Simple Changes	Mixed results	Mixed results	Good*
Complex Changes	Correct but blocking	Online but incorrect	No support

Solution Direction

Lazy schema transformations:

- Transform data on demand when accessed.



Lazy Schema Transformations

How this better meets the requirements:

- Updates are immediately visible.
- Lazy schema changes are composable.

Literature shows promising results for:

- Object databases
- Simple relational transformations

Challenges in Complex Lazy Transformations

Limitations to lazy transformations:

- **Blocking transformations:**
 - Creating indices (sorting)
 - Checking constraints
- **Inherently sequential operations:**
 - Generation of sequential keys

Does this perform well compared to existing systems?

A Real Fundamental Solution

Database based on functional core:

- Many opportunities for optimizing transactions:
 - Bottom up support for lazy evaluation.
 - Rewriting transactions.
 - Parallel execution.
- Not far from declarative queries:
 - XQuery is purely functional

Conclusion

- Requirements: A DBMS should allow complex **online** and **transactional** schema changes.
- Experimental evaluation of existing systems:
 - Mixed results for simple transformations.
 - Composed transformations either block or do not satisfy the ACID properties.
- Proposal: perform transformations lazily.