A Benchmark for Non-Blocking Schema Transformations

Lesley Wevers

Menno Tammens

Matthijs Hofstra

Marieke Huisman

Maurice van Keulen

University of Twente

DATA 2015

Why change the structure of data?

Why change the structure of data?

Logical changes:

- Add or remove columns
- Add constraints
- Change the cardinality of a relationship
- Use surrogate keys instead of natural keys
- ...

Why change the structure of data?

Logical changes:

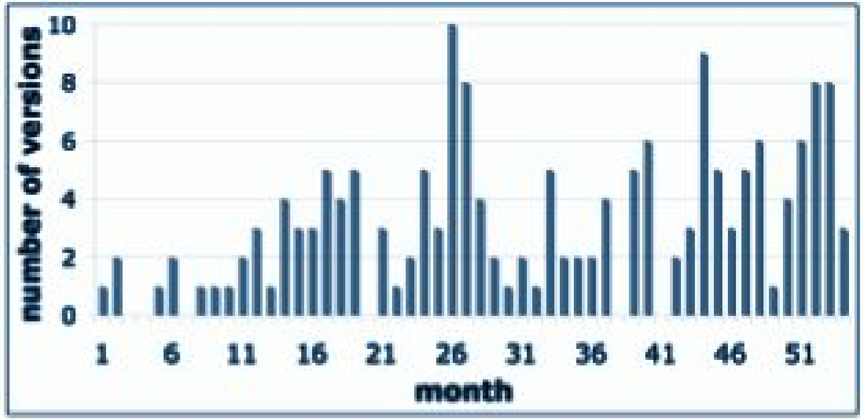
- Add or remove columns
- Add constraints
- Change the cardinality of a relationship
- Use surrogate keys instead of natural keys
- ...

Improve performance:

- Add indices
- Precompute aggregates
- Change normalization
- ...

What is the problem?

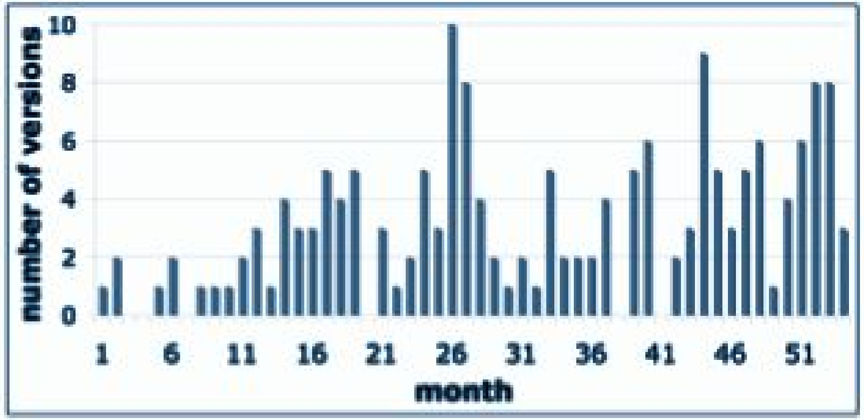
WikiMedia schema revisions:



Source: http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Schema_Evolution_Benchmark

What is the problem?

WikiMedia schema revisions:

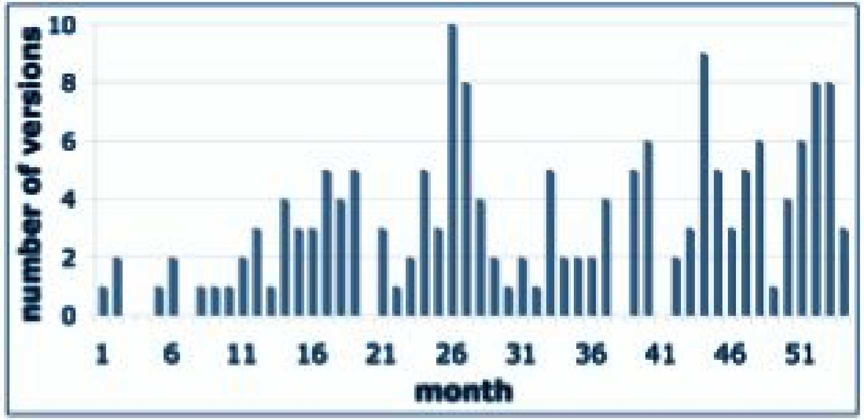


- 90% require a write lock.

Source: http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Schema_Evolution_Benchmark

What is the problem?

WikiMedia schema revisions:



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

Source: http://yellowstone.cs.ucla.edu/schema-evolution/index.php/Schema_Evolution_Benchmark

What is the problem?

Some systems can not go offline:

- Telecom, payment, airline reservation, online services

What is the problem?

Some systems can not go offline:

- Telecom, payment, airline reservation, online services

Ad-hoc solutions are insufficient:

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

What is the problem?

Some systems can not go offline:

- Telecom, payment, airline reservation, online services

Ad-hoc solutions are insufficient:

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

The DBMS should provide a solution!

Support for Online Schema Changes

DBMS support:

- PostgreSQL: Partial Instantaneous DDL
- MySQL: Partial Online DDL
- Oracle: Parallel copy

Third party tools (for MySQL):

- pt-online-schema-change
- oak-online-alter-table
- online-schema-change

Support for Online Schema Changes

DBMS support:

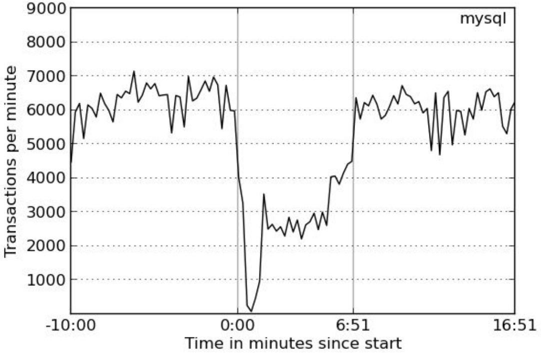
- PostgreSQL: Partial Instantaneous DDL
- MySQL: Partial Online DDL
- Oracle: Parallel copy

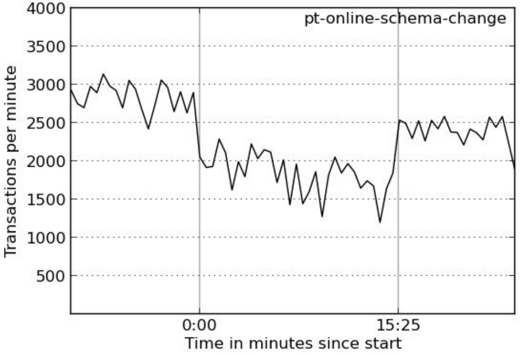
Third party tools (for MySQL):

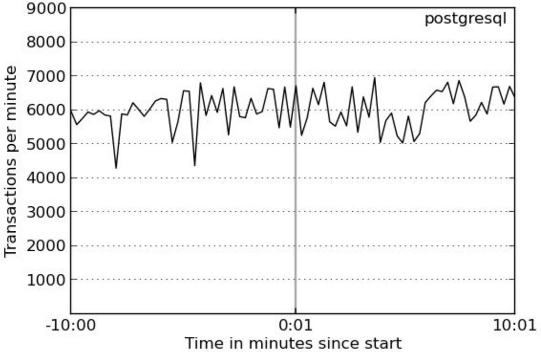
- pt-online-schema-change
- oak-online-alter-table
- online-schema-change

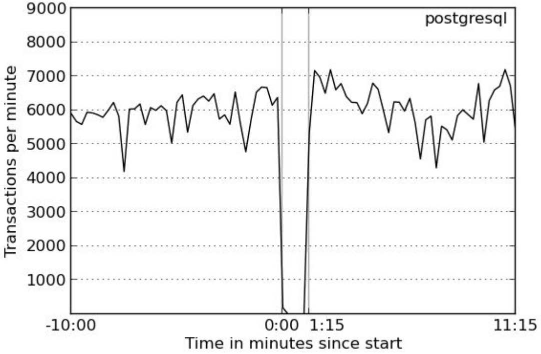
To what degree do these solutions work?

Benchmark: Adding a Column

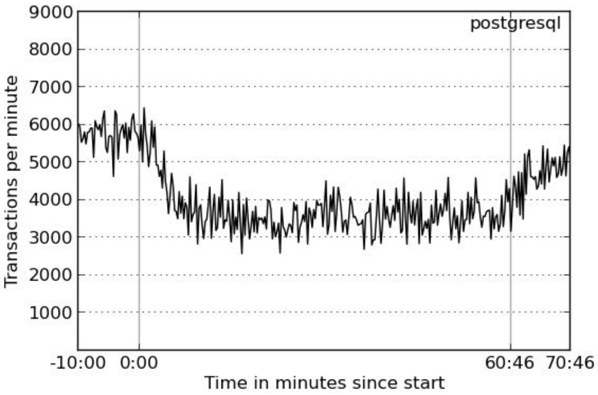


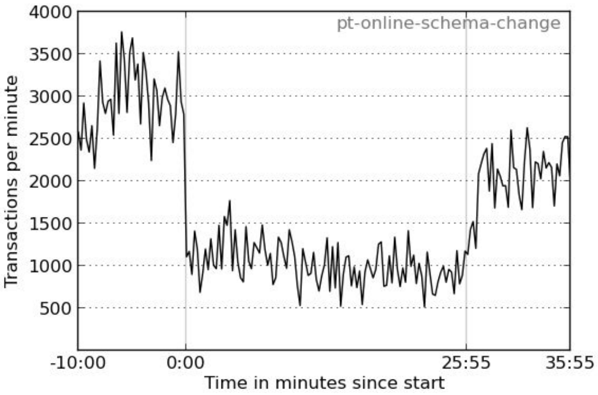


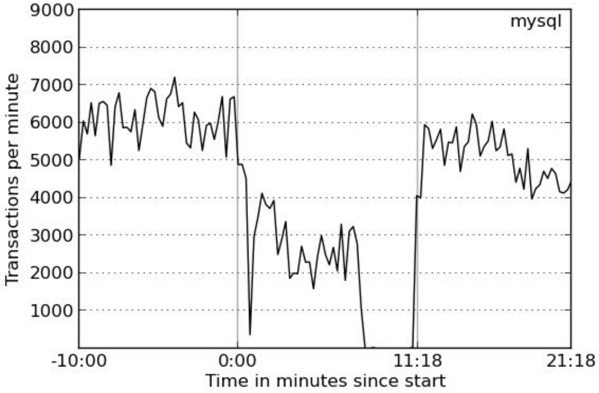




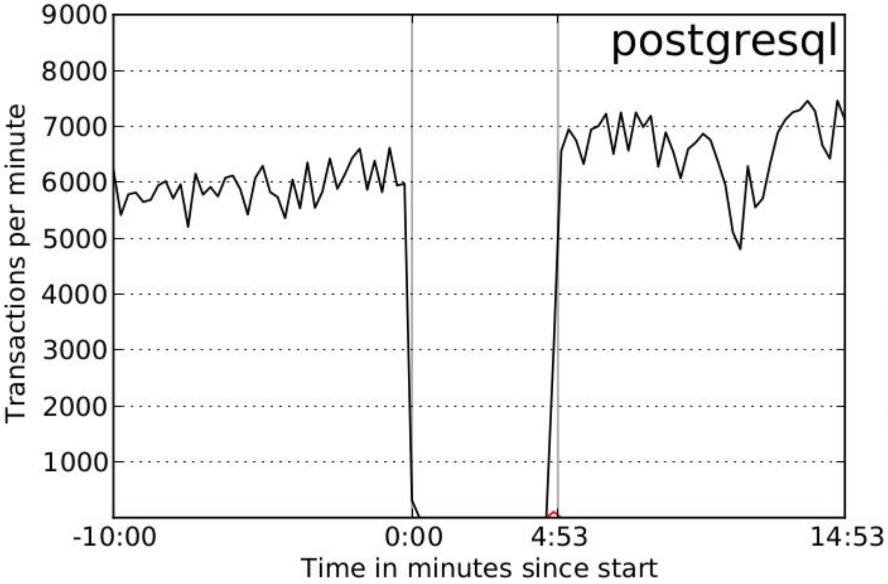
Benchmark: Creating an Index

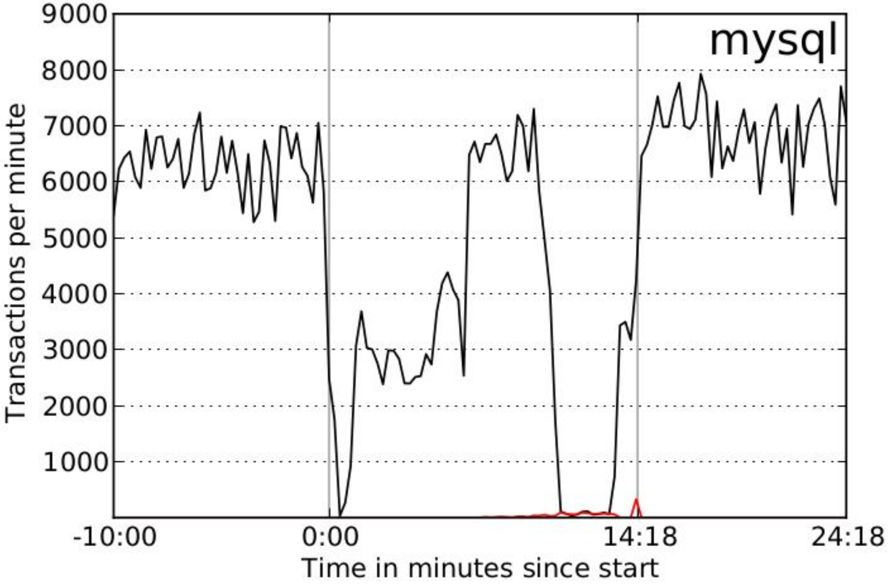






Complex Transformations





No support by third party tools

Current Situation

- Mixed results for basic (DDL) transformations:
    - Columns
    - Indices
    - Constraints
    - Data transformations

Current Situation

- Mixed results for basic (DDL) transformations:
  - Columns
  - Indices
  - Constraints
  - Data transformations

- Support for complex online transformations is mostly absent:
  - Change a primary key
  - Splitting and merging of tables
  - Changing the cardinality of a relationship
  - ...

Contributions

- Criteria for evaluating online schema change mechanisms in general, and for the relational model in particular.
- A concrete benchmark based on TPC-C to:
  - Compare existing solutions
  - Challenge the DB community to find solutions

Contributions

- Criteria for evaluating online schema change mechanisms in general, and for the relational model in particular.

Criteria for Online Transformations

We have defined criteria for:

- Functionality of OST
- Performance of OST

We define:

- Ideal behaviour
- Acceptable behaviour

Based on characteristics of state of the art solutions.

Functional Criteria

A mechanism for schema transformations should:

- Allow simple and complex transformations
- Provide data in new schema upon commit
- Satisfy the ACID properties
- Be declarative
- Support online upgrading of database applications

Performance Characteristics

Impact on concurrent transactions:

- Blocking
- Aborts
- Slowdown

Performance Characteristics

Impact on concurrent transactions:

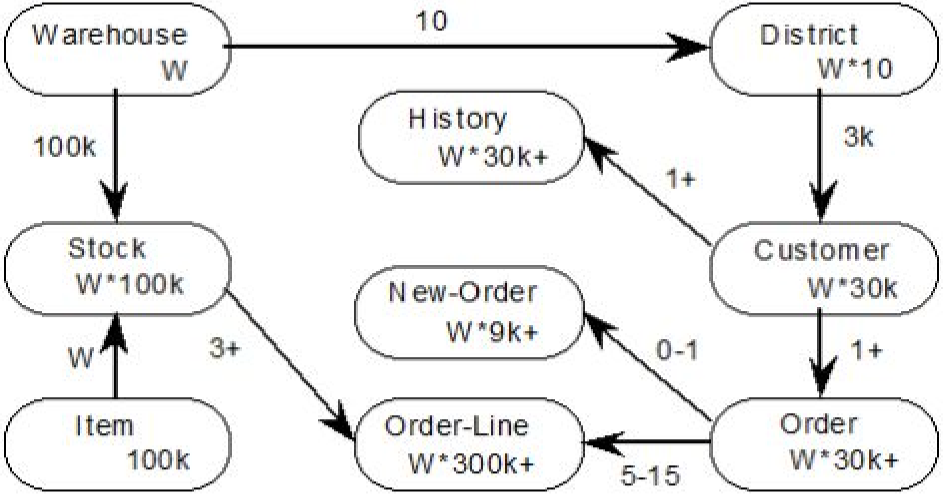
- Blocking
- Aborts
- Slowdown

Performance of schema transformations:

- No aborts
- Time to commit

Benchmark

TPC-C



New order

Payment

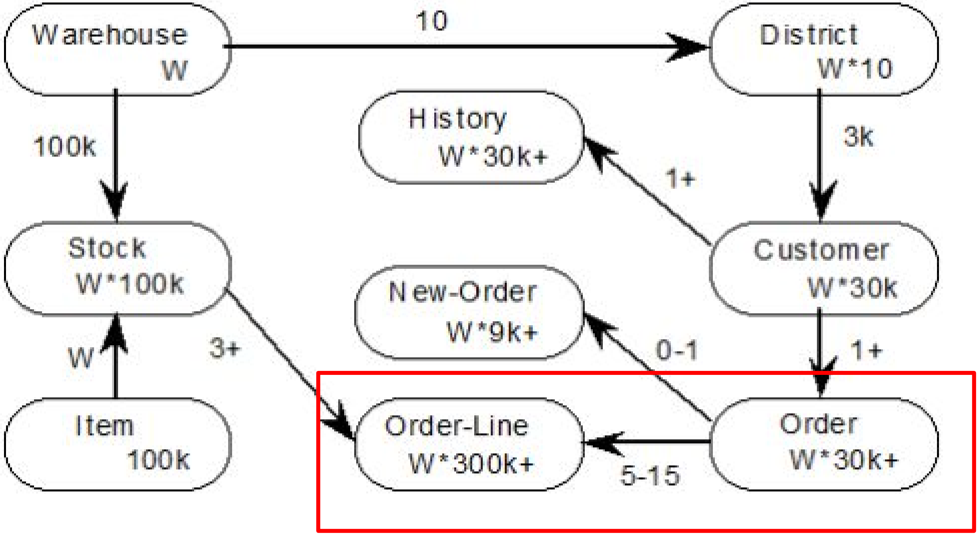
Order status

Delivery

Stock level

Benchmark

TPC-C



New order

Payment

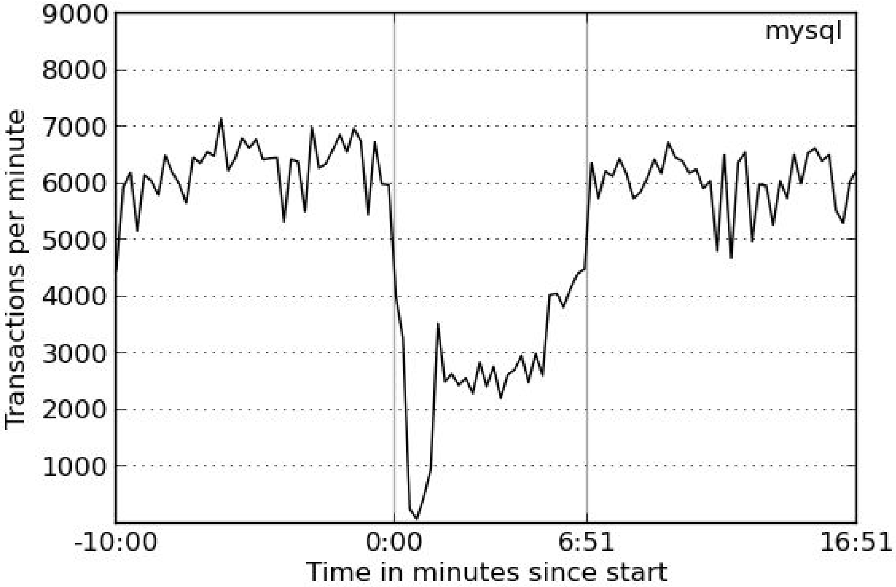
Order status

Delivery

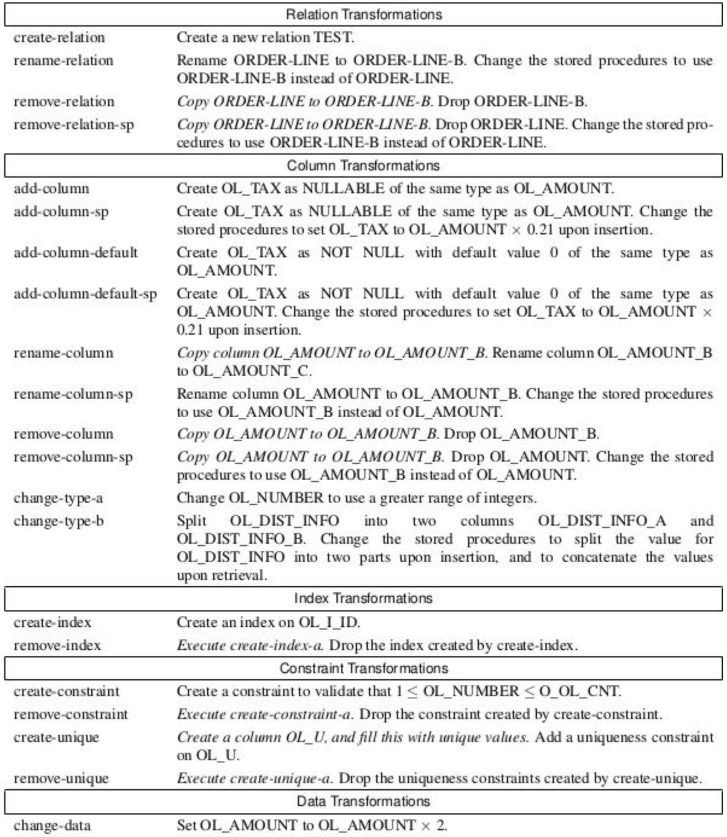
Stock level

Benchmark Process

- Setup database
- Start TPC-C
- Intro period
- Transform:
    - Schema
    - Stored procedures
- Outro period
- Stop TPC-C



Benchmark Cases



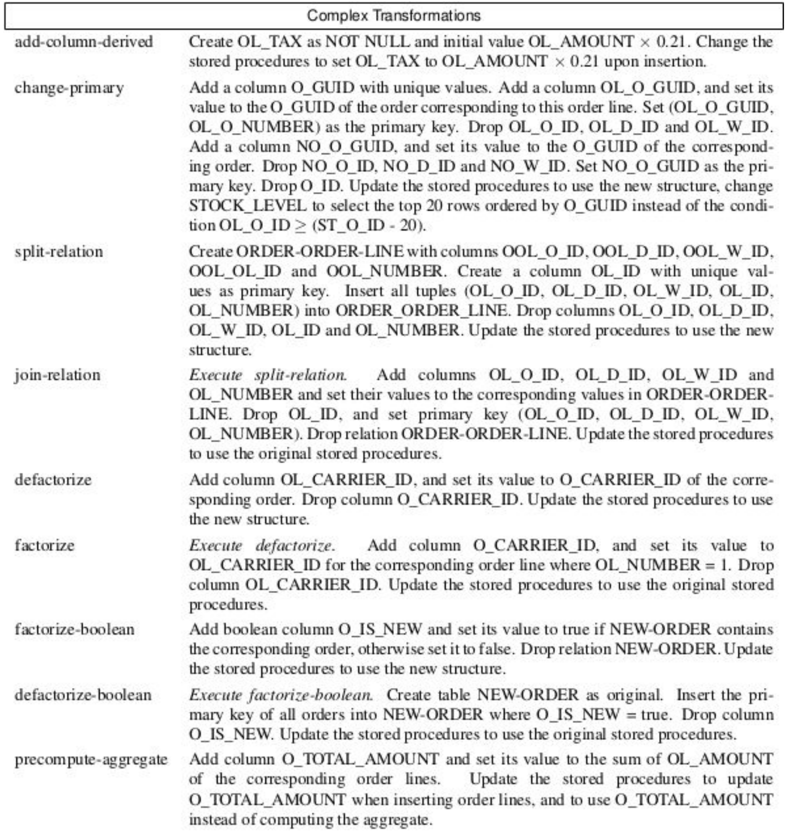
Relation Transformations	
create-relation	Create a new relation TEST.
rename-relation	Rename ORDER-LINE to ORDER-LINE-B. Change the stored procedures to use ORDER-LINE-B instead of ORDER-LINE.
remove-relation	*Copy ORDER-LINE to ORDER-LINE-B.* Drop ORDER-LINE-B.
remove-relation-sp	*Copy ORDER-LINE to ORDER-LINE-B.* Drop ORDER-LINE. Change the stored procedures to use ORDER-LINE-B instead of ORDER-LINE.

Column Transformations	
add-column	Create OL_TAX as NULLABLE of the same type as OL_AMOUNT.
add-column-sp	Create OL_TAX as NULLABLE of the same type as OL_AMOUNT. Change the stored procedures to set OL_TAX to OL_AMOUNT × 0.21 upon insertion.
add-column-default	Create OL_TAX as NOT NULL with default value 0 of the same type as OL_AMOUNT.
add-column-default-sp	Create OL_TAX as NOT NULL with default value 0 of the same type as OL_AMOUNT. Change the stored procedures to set OL_TAX to OL_AMOUNT × 0.21 upon insertion.
rename-column	*Copy column OL_AMOUNT to OL_AMOUNT_B.* Rename column OL_AMOUNT_B to OL_AMOUNT_C.
rename-column-sp	Rename column OL_AMOUNT to OL_AMOUNT_B. Change the stored procedures to use OL_AMOUNT_B instead of OL_AMOUNT.
remove-column	*Copy OL_AMOUNT to OL_AMOUNT_B.* Drop OL_AMOUNT_B.
remove-column-sp	*Copy OL_AMOUNT to OL_AMOUNT_B.* Drop OL_AMOUNT. Change the stored procedures to use OL_AMOUNT_B instead of OL_AMOUNT.
change-type-a	Change OL_NUMBER to use a greater range of integers.
change-type-b	Split OL_DIST_INFO into two columns OL_DIST_INFO_A and OL_DIST_INFO_B. Change the stored procedures to split the value for OL_DIST_INFO into two parts upon insertion, and to concatenate the values upon retrieval.

Index Transformations	
create-index	Create an index on OL_I_ID.
remove-index	*Execute create-index-a.* Drop the index created by create-index.

Constraint Transformations	
create-constraint	Create a constraint to validate that 1 ≤ OL_NUMBER ≤ O_OL_CNT.
remove-constraint	*Execute create-constraint-a.* Drop the constraint created by create-constraint.
create-unique	*Create a column OL_U, and fill this with unique values.* Add a uniqueness constraint on OL_U.
remove-unique	*Execute create-unique-a.* Drop the uniqueness constraints created by create-unique.

Data Transformations	
change-data	Set OL_AMOUNT to OL_AMOUNT × 2.



Complex Transformations	
add-column-derived	Create OL_TAX as NOT NULL and initial value OL_AMOUNT × 0.21. Change the stored procedures to set OL_TAX to OL_AMOUNT × 0.21 upon insertion.
change-primary	Add a column O_GUID with unique values. Add a column OL_O_GUID, and set its value to the O_GUID of the order corresponding to this order line. Set (OL_O_GUID, OL_O_NUMBER) as the primary key. Drop OL_O_ID, OL_D_ID and OL_W_ID. Add a column NO_O_GUID, and set its value to the O_GUID of the corresponding order. Drop NO_O_ID, NO_D_ID and NO_W_ID. Set NO_O_GUID as the primary key. Drop O_ID. Update the stored procedures to use the new structure, change STOCK_LEVEL to select the top 20 rows ordered by O_GUID instead of the condition OL_O_ID ≥ (ST_O_ID - 20).
split-relation	Create ORDER-ORDER-LINE with columns OOL_O_ID, OOL_D_ID, OOL_W_ID, OOL_OL_ID and OOL_NUMBER. Create a column OL_ID with unique values as primary key. Insert all tuples (OL_O_ID, OL_D_ID, OL_W_ID, OL_ID, OL_NUMBER) into ORDER_ORDER_LINE. Drop columns OL_O_ID, OL_D_ID, OL_W_ID, OL_ID and OL_NUMBER. Update the stored procedures to use the new structure.
join-relation	*Execute split-relation.* Add columns OL_O_ID, OL_D_ID, OL_W_ID and OL_NUMBER and set their values to the corresponding values in ORDER-ORDER-LINE. Drop OL_ID, and set primary key (OL_O_ID, OL_D_ID, OL_W_ID, OL_NUMBER). Drop relation ORDER-ORDER-LINE. Update the stored procedures to use the original stored procedures.
defactorize	Add column OL_CARRIER_ID, and set its value to O_CARRIER_ID of the corresponding order. Drop column O_CARRIER_ID. Update the stored procedures to use the new structure.
factorize	*Execute defactorize.* Add column O_CARRIER_ID, and set its value to OL_CARRIER_ID for the corresponding order line where OL_NUMBER = 1. Drop column OL_CARRIER_ID. Update the stored procedures to use the original stored procedures.
factorize-boolean	Add boolean column O_IS_NEW and set its value to true if NEW-ORDER contains the corresponding order, otherwise set it to false. Drop relation NEW-ORDER. Update the stored procedures to use the new structure.
defactorize-boolean	*Execute factorize-boolean.* Create table NEW-ORDER as original. Insert the primary key of all orders into NEW-ORDER where O_IS_NEW = true. Drop column O_IS_NEW. Update the stored procedures to use the original stored procedures.
precompute-aggregate	Add column O_TOTAL_AMOUNT and set its value to the sum of OL_AMOUNT of the corresponding order lines. Update the stored procedures to update O_TOTAL_AMOUNT when inserting order lines, and to use O_TOTAL_AMOUNT instead of computing the aggregate.

Implementation

Benchmark scripts available for:

- MySQL
- PostgreSQL
- Oracle (partially implemented)
- pt-online-schema-change (only basic cases)

Based on the HammerDB TPC-C implementation.

Conclusion

- Criteria for online schema changes:
  - Clarify the problem of OST
  - Identify ideal characteristics of a solution
- We have developed a benchmark to:
  - Show the extend of the problem
  - Compare performance of solutions
  - Challenge the DB community to find solutions for:
    - Better support for basic transformations
    - Support for complex transformations

For more info

Read the paper:

- A Benchmark for Non-blocking Schema Transformations

Download the benchmark implementations:

- http://wwwhome.ewi.utwente.nl/~weversl2/?page=ost

ADBIS 2015 paper:

- Analysis of the Blocking Behaviour of Schema Transformations in Relational Database Systems