

Better Indexation  
without Performance  
Penalty

# Charities helping Ukraine 🇺🇦

- Ukrainian Foundation "Come Back Alive"

<https://savelife.in.ua/en/>




- Dutch Foundation "Sails of Freedom"

<https://zeilenvanvrijheid.nl/>



About me

# About me

 Passionate about open source projects

# About me



Passionate about open source projects



Expert in optimizing web applications

# About me



Passionate about open source projects



Expert in optimizing web applications



One of grand-parents of Magento

# About me



Passionate about open source projects



Expert in optimizing web applications



One of grand-parents of Magento



Experienced in putting out fires on production systems

# Why is indexation important?

# Why is indexation important?



Normalized database structure is slower to query

# Why is indexation important?



Normalized database structure is slower to query



Expensive calculations on listing pages

# Why is indexation important?



Normalized database structure is slower to query



Expensive calculations on listing pages



Text based search tokenization

What is offered by default?

# What is offered by default?



Update on Save

# What is offered by default?



Update on Save

- Does not take into account changes from import/export
- Prone to wait timeouts and deadlocks

# What is offered by default?



## Update on Save

- Does not take into account changes from import/export
- Prone to wait timeouts and deadlocks



## Update by Schedule

# What is offered by default?



## Update on Save

- Does not take into account changes from import/export
- Prone to wait timeouts and deadlocks



## Update by Schedule

- Named mview, but not a real materialized view
- Observes data modifications with triggers
- Litters up your database with {mview\_name}\_cl tables

Writign Good Triggers is Hard

# Writign Good Triggers is Hard

💥 2.2.x - Blows up large databases because core team did not understand how `<=>` operator works

# Writign Good Triggers is Hard

💥 2.2.x - Blows up large databases because core team did not understand how `<=>` operator works

- #23077 Triggers created by MView are triggered all the time

# Writign Good Triggers is Hard

💣 2.2.x - Blows up large databases because core team did not understand how  $\leq$  operator works

- #23077 Triggers created by MView are triggered all the time

🙄 2.4.x - Introduced better batching but made things worse

# Writing Good Triggers is Hard

💣 2.2.x - Blows up large databases because core team did not understand how  $\leq$  operator works


- #23077 Triggers created by MView are triggered all the time

🧐 2.4.x - Introduced better batching but made things worse


- #30012 Asynchronous indexing can break large servers
- #37367 Schedule index - entities processed multiple times


Not that Obvious Issues

# Not that Obvious Issues


 Every database write produces INSERT into the same set of tables within write transaction


# Not that Obvious Issues


 Every database write produces INSERT into the same set of tables within write transaction

 Catalog Staging introduces even more complex lookup logic on each write

# Not that Obvious Issues

 Every database write produces INSERT into the same set of tables within write transaction

 Catalog Staging introduces even more complex lookup logic on each write

 Dangerous when not configured properly in multi-origin replication

Can We do Better?

# Binary Log to the Rescue

# Binary Log to the Rescue



Enabled by default on AWS RDS

# Binary Log to the Rescue



Enabled by default on AWS RDS



Row based format is default since MySQL 8.x

# Binary Log to the Rescue



Enabled by default on AWS RDS



Row based format is default since MySQL 8.x



Experienced DevOps already use it for replication

# Binary Log to the Rescue



Enabled by default on AWS RDS



Row based format is default since MySQL 8.x



Experienced DevOps already use it for replication



Gives complete access to data change history

# Binary Log Configuration



/etc/mysql/conf.d/mysql-x-enable-binlog.cnf

```
[mysqld]
# Enables binary logging with default file paths
log_bin
# Make sure you row based replication is used
binlog-format = row
# Reduce size of binary log to not include text and binary fields if they are not c
binlog-row-image=noblob
# Nice to have automatic clean up of binlog files
expire_logs_days = 5
max_binlog_size = 100M
```

# Binary Log Configuration



/etc/mysql/conf.d/mysql-x-enable-binlog.cnf

```
[mysqld]
# Enables binary logging with default file paths
log_bin
# Make sure you row based replication is used
binlog-format = row
# Reduce size of binary log to not include text and binary fields if they are not c
binlog-row-image=noblob
# Nice to have automatic clean up of binlog files
expire_logs_days = 5
max_binlog_size = 100M
```

# Binary Log Configuration



/etc/mysql/conf.d/mysql-x-enable-binlog.cnf

```
[mysqld]
# Enables binary logging with default file paths
log_bin
# Make sure you row based replication is used
binlog-format = row
# Reduce size of binary log to not include text and binary fields if they are not c
binlog-row-image=noblob
# Nice to have automatic clean up of binlog files
expire_logs_days = 5
max_binlog_size = 100M
```

# Binary Log Configuration



/etc/mysql/conf.d/mysql-x-enable-binlog.cnf

```
[mysqld]
# Enables binary logging with default file paths
log_bin
# Make sure you row based replication is used
binlog-format = row
# Reduce size of binary log to not include text and binary fields if they are not c
binlog-row-image=noblob
# Nice to have automatic clean up of binlog files
expire_logs_days = 5
max_binlog_size = 100M
```

# Binary Log Configuration



/etc/mysql/conf.d/mysql-x-enable-binlog.cnf

```
[mysqld]
# Enables binary logging with default file paths
log_bin
# Make sure you row based replication is used
binlog-format = row
# Reduce size of binary log to not include text and binary fields if they are not c
binlog-row-image=noblob
# Nice to have automatic clean up of binlog files
expire_logs_days = 5
max_binlog_size = 100M
```

# User Permissions

Make sure you have the following permission for your MySQL user

```
GRANT SELECT, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO '[USER]@'%';
```

Not needed if you have all **GRANT ALL**, but it is good practice to create separate user with only this permissions.

# Mage-OS Database Changelog

# Mage-OS Database Changelog



Written in Rust as static binary

# Mage-OS Database Changelog



Written in Rust as static binary



No root permissions needed

# Mage-OS Database Changelog



Written in Rust as static binary



No root permissions needed



Blazingly fast

# How it Works

# How it Works



Connects to MySQL as Replication Client

# How it Works



Connects to MySQL as Replication Client



Analyzes Binary Log since last run

# How it Works



Connects to MySQL as Replication Client



Analyzes Binary Log since last run



Transforms Binary Events into Row Changes

# How it Works



Connects to MySQL as Replication Client



Analyzes Binary Log since last run



Transforms Binary Events into Row Changes



Maps Row Changes into Domain Updates

# How it Works



Connects to MySQL as Replication Client



Analyzes Binary Log since last run



Transforms Binary Events into Row Changes



Maps Row Changes into Domain Updates



Aggregate reduces Domain Update into Log Output

# How it Works



Connects to MySQL as Replication Client



Analyzes Binary Log since last run



Transforms Binary Events into Row Changes



Maps Row Changes into Domain Updates



Aggregate reduces Domain Update into Log Output



Magento processes Log Output and updates MView

Easy to Customize

# Easy to Customize



Available also as Cargo library

# Easy to Customize



Available also as Cargo library



Easy to create own Mappers

# Easy to Customize



Available also as Cargo library



Easy to create own Mappers



Sample Application Skeleton (coming)

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Example of Custom Mapper

```
pub struct MyAwesomeTableMapper;

impl ChangeLogMapper<ProductChange> for MyAwesomeTableMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ProductChange>, Error> {
        Ok(match event {
            Event::UpdateRow(row) ⇒ FieldUpdate
                ::new(row.parse("product_id", schema)?)
                    .process_field_update("not_awesome_field", row, schema)
                    .process_field_update("super_awesome_field", row, schema)
                    .into_change_log()
            _ ⇒ None,
        })
    }
}
```

# Adding Table Name Mapper

```
pub struct MyAwesomeTableNameMapper;

impl ChangeLogMapper<ItemChange> for MyAwesomeTableNameMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ItemChange>, Error> {
        Ok(match schema.table_name() {
            "my_awesome_table_for_product" ⇒ MyAwesomeTableMapper
                .map_event(event, schema)?.map(ItemChange::ProductChange),
            _ ⇒ None
        })
    }
}
```

# Adding Table Name Mapper

```
pub struct MyAwesomeTableNameMapper;

impl ChangeLogMapper<ItemChange> for MyAwesomeTableNameMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ItemChange>, Error> {
        Ok(match schema.table_name() {
            "my_awesome_table_for_product" ⇒ MyAwesomeTableMapper
                .map_event(event, schema)?.map(ItemChange::ProductChange),
            _ ⇒ None
        })
    }
}
```

# Adding Table Name Mapper

```
pub struct MyAwesomeTableNameMapper;

impl ChangeLogMapper<ItemChange> for MyAwesomeTableNameMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ItemChange>, Error> {
        Ok(match schema.table_name() {
            "my_awesome_table_for_product" ⇒ MyAwesomeTableMapper
                .map_event(event, schema)?.map(ItemChange::ProductChange),
            _ ⇒ None
        })
    }
}
```

# Adding Table Name Mapper

```
pub struct MyAwesomeTableNameMapper;

impl ChangeLogMapper<ItemChange> for MyAwesomeTableNameMapper {
    fn map_event(&self, event: &Event, schema: &impl TableSchema)
        → Result<Option<ItemChange>, Error> {
        Ok(match schema.table_name() {
            "my_awesome_table_for_product" ⇒ MyAwesomeTableMapper
                .map_event(event, schema)?.map(ItemChange::ProductChange),
            _ ⇒ None
        })
    }
}
```

# Add to App Skeleton

```
Application::parse()?
    .with_mapper(MyAwesomeTableNameMatchMapper)
    .run().await?;
```



Eager for Nmbrs?

# Updating Product Data with Triggers

```
checks.....: 100.00% ✓ 1512      ✗ 0
data_received.....: 11 MB    19 kB/s
data_sent.....: 1.5 MB    2.4 kB/s
dropped_iterations.....: 28      0.046447/s
http_req_waiting.....: avg=352.83ms min=101.32ms med=383.35ms
http_reqs.....: 1512     2.508118/s
```

```
running (10m02.8s), 0/1 VUs, 72 complete and 0 interrupted iterations
default ✗ [=====] 1 VUs  10m02.8s/10m0s  072/100 shared iters
```

```
Product Category MView Time: 30.49039s
```

# Updating Product Data with Triggers

```
checks.....: 100.00% ✓ 1512      ✗ 0
data_received.....: 11 MB    19 kB/s
data_sent.....: 1.5 MB    2.4 kB/s
dropped_iterations.....: 28      0.046447/s
http_req_waiting.....: avg=352.83ms min=101.32ms med=383.35ms
http_reqs.....: 1512     2.508118/s
```

```
running (10m02.8s), 0/1 VUs, 72 complete and 0 interrupted iterations
default ✗ [=====] 1 VUs  10m02.8s/10m0s  072/100 shared iters
```

Product Category MView Time: 30.49039s

# Updating Product Data with Triggers

```
checks.....: 100.00% ✓ 1512      ✗ 0
data_received.....: 11 MB    19 kB/s
data_sent.....: 1.5 MB    2.4 kB/s
dropped_iterations.....: 28      0.046447/s
http_req_waiting.....: avg=352.83ms min=101.32ms med=383.35ms
http_reqs.....: 1512     2.508118/s
```

```
running (10m02.8s), 0/1 VUs, 72 complete and 0 interrupted iterations
default ✗ [=====] 1 VUs  10m02.8s/10m0s  072/100 shared iters
```

Product Category MView Time: 30.49039s

# Updating Product Data with Triggers

```
checks.....: 100.00% ✓ 1512      ✗ 0
data_received.....: 11 MB    19 kB/s
data_sent.....: 1.5 MB    2.4 kB/s
dropped_iterations.....: 28      0.046447/s
http_req_waiting.....: avg=352.83ms min=101.32ms med=383.35ms
http_reqs.....: 1512     2.508118/s
```

```
running (10m02.8s), 0/1 VUs, 72 complete and 0 interrupted iterations
default ✗ [=====] 1 VUs  10m02.8s/10m0s  072/100 shared iters
```

Product Category MView Time: 30.49039s

# Updating Product Data with Triggers

```
checks.....: 100.00% ✓ 1512      ✗ 0
data_received.....: 11 MB    19 kB/s
data_sent.....: 1.5 MB    2.4 kB/s
dropped_iterations.....: 28      0.046447/s
http_req_waiting.....: avg=352.83ms min=101.32ms med=383.35ms
http_reqs.....: 1512      2.508118/s
```

```
running (10m02.8s), 0/1 VUs, 72 complete and 0 interrupted iterations
default ✗ [=====] 1 VUs  10m02.8s/10m0s  072/100 shared iters
```

Product Category MView Time: 30.49039s

# Updating Product Data with Changelog

```
checks.....: 100.00% ✓ 1680      ✗ 0
data_received.....: 12 MB    21 kB/s
data_sent.....: 1.6 MB    2.7 kB/s
dropped_iterations.....: 20      0.033187/s
http_req_waiting.....: avg=316.38ms min=107.47ms med=335.25ms
http_reqs.....: 1680     2.787712/s
```

```
running (10m02.6s), 0/1 VUs, 80 complete and 0 interrupted iterations
default ✗ [=====] 1 VUs  10m02.6s/10m0s  080/100 shared iters
```

```
Product Category MView Time: 7.19157s
```

# Updating Product Data with Changelog

```
checks.....: 100.00% ✓ 1680      ✗ 0
data_received.....: 12 MB    21 kB/s
data_sent.....: 1.6 MB    2.7 kB/s
dropped_iterations.....: 20      0.033187/s
http_req_waiting.....: avg=316.38ms min=107.47ms med=335.25ms
http_reqs.....: 1680     2.787712/s
```

```
running (10m02.6s), 0/1 VUs, 80 complete and 0 interrupted iterations
default ✗ [=====→—————] 1 VUs  10m02.6s/10m0s  080/100 shared iters
```

Product Category MView Time: 7.19157s

# Updating Product Data with Changelog

```
checks.....: 100.00% ✓ 1680      ✗ 0
data_received.....: 12 MB    21 kB/s
data_sent.....: 1.6 MB    2.7 kB/s
dropped_iterations.....: 20      0.033187/s
http_req_waiting.....: avg=316.38ms min=107.47ms med=335.25ms
http_reqs.....: 1680      2.787712/s
```

```
running (10m02.6s), 0/1 VUs, 80 complete and 0 interrupted iterations
default ✗ [=====>————] 1 VUs  10m02.6s/10m0s  080/100 shared iters
```

Product Category MView Time: 7.19157s

# Updating Product Data with Changelog

```
checks.....: 100.00% ✓ 1680      ✗ 0
data_received.....: 12 MB    21 kB/s
data_sent.....: 1.6 MB    2.7 kB/s
dropped_iterations.....: 20      0.033187/s
http_req_waiting.....: avg=316.38ms min=107.47ms med=335.25ms
http_reqs.....: 1680      2.787712/s
```

```
running (10m02.6s), 0/1 VUs, 80 complete and 0 interrupted iterations
default ✗ [=====>—————] 1 VUs  10m02.6s/10m0s  080/100 shared iters
```

Product Category MView Time: 7.19157s

# Updating Product Data with Changelog

```
checks.....: 100.00% ✓ 1680      ✗ 0
data_received.....: 12 MB    21 kB/s
data_sent.....: 1.6 MB    2.7 kB/s
dropped_iterations.....: 20      0.033187/s
http_req_waiting.....: avg=316.38ms min=107.47ms med=335.25ms
http_reqs.....: 1680      2.787712/s
```

```
running (10m02.6s), 0/1 VUs, 80 complete and 0 interrupted iterations
default ✗ [=====→—————] 1 VUs  10m02.6s/10m0s  080/100 shared iters
```

Product Category MView Time: 7.19157s

# Output Example from Load Test

```
{  
  "attributes":{  
    "106":[ 31, 14, 27, 58 ],  
    "124":[ 16, 76, 10, 2 ],  
    "97":[ 646, 403, 773, 41553, 306, 756 ],  
    "99":[ 712, 45, 334, 690, 426 ]  
  },  
  "entity":"product",  
  "metadata":{  
    "file":"47de4c575982-bin.000001",  
    "position":3086928,  
    "timestamp":1683532855  
  }  
}
```

# Output Example from Load Test

```
{
  "attributes":{
    "106":[ 31, 14, 27, 58 ],
    "124":[ 16, 76, 10, 2 ],
    "97":[ 646, 403, 773, 41553, 306, 756 ],
    "99":[ 712, 45, 334, 690, 426 ]
  },
  "entity":"product",
  "metadata":{
    "file":"47de4c575982-bin.000001",
    "position":3086928,
    "timestamp":1683532855
  }
}
```

# Output Example from Load Test

```
{  
  "attributes":{  
    "106":[ 31, 14, 27, 58 ],  
    "124":[ 16, 76, 10, 2 ],  
    "97":[ 646, 403, 773, 41553, 306, 756 ],  
    "99":[ 712, 45, 334, 690, 426 ]  
  },  
  "entity":"product",  
  "metadata":{  
    "file":"47de4c575982-bin.000001",  
    "position":3086928,  
    "timestamp":1683532855  
  }  
}
```

# Output Example from Load Test

```
{
  "attributes":{
    "106":[ 31, 14, 27, 58 ],
    "124":[ 16, 76, 10, 2 ],
    "97":[ 646, 403, 773, 41553, 306, 756 ],
    "99":[ 712, 45, 334, 690, 426 ]
  },
  "entity":"product",
  "metadata":{
    "file":"47de4c575982-bin.000001",
    "position":3086928,
    "timestamp":1683532855
  }
}
```

# But there is more

```
enum AggregateKey {  
    Created,  
    Deleted,  
    Field(&'static str),  
    Attribute(usize),  
    WebsiteAll,  
    WebsiteSpecific(usize),  
    CategoryAll,  
    CategorySpecific(usize),  
    Link(usize),  
    Composite,  
    MediaGallery,  
    TierPrice,  
}
```

# But there is more

```
enum AggregateKey {  
    Created,  
    Deleted,  
    Field(&'static str),  
    Attribute(usize),  
    WebsiteAll,  
    WebsiteSpecific(usize),  
    CategoryAll,  
    CategorySpecific(usize),  
    Link(usize),  
    Composite,  
    MediaGallery,  
    TierPrice,  
}
```

# But there is more

```
enum AggregateKey {  
    Created,  
    Deleted,  
    Field(&'static str),  
    Attribute(usize),  
    WebsiteAll,  
    WebsiteSpecific(usize),  
    CategoryAll,  
    CategorySpecific(usize),  
    Link(usize),  
    Composite,  
    MediaGallery,  
    TierPrice,  
}
```

# But there is more

```
enum AggregateKey {  
    Created,  
    Deleted,  
    Field(&'static str),  
    Attribute(usize),  
    WebsiteAll,  
    WebsiteSpecific(usize),  
    CategoryAll,  
    CategorySpecific(usize),  
    Link(usize),  
    Composite,  
    MediaGallery,  
    TierPrice,  
}
```

# But there is more

```
enum AggregateKey {  
    Created,  
    Deleted,  
    Field(&'static str),  
    Attribute(usize),  
    WebsiteAll,  
    WebsiteSpecific(usize),  
    CategoryAll,  
    CategorySpecific(usize),  
    Link(usize),  
    Composite,  
    MediaGallery,  
    TierPrice,  
}
```

# But there is more

```
enum AggregateKey {  
    Created,  
    Deleted,  
    Field(&'static str),  
    Attribute(usize),  
    WebsiteAll,  
    WebsiteSpecific(usize),  
    CategoryAll,  
    CategorySpecific(usize),  
    Link(usize),  
    Composite,  
    MediaGallery,  
    TierPrice,  
}
```

# But there is more

```
enum AggregateKey {  
    Created,  
    Deleted,  
    Field(&'static str),  
    Attribute(usize),  
    WebsiteAll,  
    WebsiteSpecific(usize),  
    CategoryAll,  
    CategorySpecific(usize),  
    Link(usize),  
    Composite,  
    MediaGallery,  
    TierPrice,  
}
```

# Roadmap

# Roadmap

 Support for Category Entity

# Roadmap



Support for Category Entity



Support for Staging in Adobe Commerce

# Roadmap



Support for Category Entity



Support for Staging in Adobe Commerce



Support for Custom Tables via Configuration File

# Roadmap



Support for Category Entity



Support for Staging in Adobe Commerce





Support for Custom Tables via Configuration File





Support for Magento 1.x

# Roadmap

 Support for Category Entity

 Support for Staging in Adobe Commerce

 Support for Custom Tables via Configuration File

 Support for Magento 1.x

 Plug-and-Play Composer based Module with Binary

Get Invovled

# Get Invovled

♥ Join Mage-OS Community [mage-os.org](https://mage-os.org)

# Get Invovled

🧡 Join Mage-OS Community [mage-os.org](https://mage-os.org)

👤 Follow on GitHub [EcomDev/mage-os-database-changelog](https://github.com/EcomDev/mage-os-database-changelog)

Bright Future Ahead

# Bright Future Ahead



Security monitoring tool based on database changelog

# Bright Future Ahead



Security monitoring tool based on database changelog



Blazingly fast data streams for even better indexation and data export

# Thank You



Want to learn more about performance?



Join performance training this July in Barcelona, Spain



[bit.ly/mage-performance-training](https://bit.ly/mage-performance-training)