MidPoint 4.4 – Native PostgreSQL Repository

Richard Richter / January 2022

Agenda

Evolveum

- What is midPoint Repository
- Why the new Native repository
- Native repository anatomy
- Using and tuning Native repository
- Native audit, partitioning, migration

What is midPoint repository?

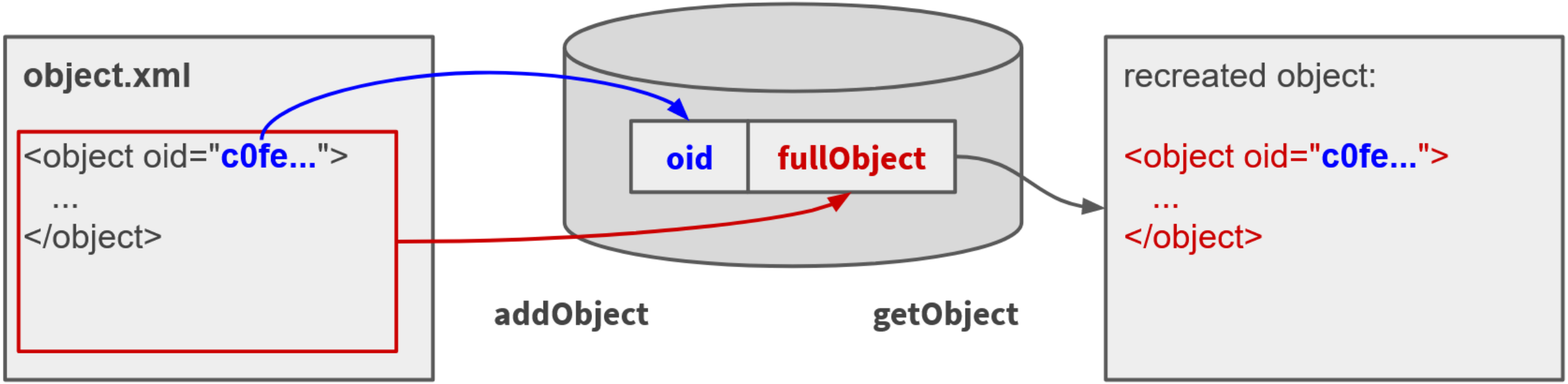
Evolveum

- **MidPoint Repository keeps midPoint objects persistent.**
  - MidPoint can be restarted and its state is preserved.
  - Objects don’t have to be in memory, if midPoint doesn’t work with them.
- Must support basic create/read/update/delete (CRUD) operations.
- For ages now, midPoint has been using an SQL database as a repository.

Minimal repository

Evolveum

- **addObject** writes **fullObject** document under its OID (generated if necessary)
- **getObject** uses OID to retrieve the **fullObject** and deserializes it



- Add update and delete and we're done! Or are we?

Repository must be searchable

Evolveum

- Repository must support fast object retrieval by the OID.
  - But what if we don’t know the OID?
- Repository must support efficient search for objects.
  - Internal hard-coded searches vs custom searches
    - All use midPoint Query API
  - Iterative search for processing many results
- In some cases we want to search for containers.
- Searchable properties must be available outside an opaque **fullObject**.

Not your DB for common information system

Evolveum

- MidPoint repository is still primarily “document” storage.
- MidPoint objects are the “documents” it stores.
- MidPoint objects are extensible.
- All the exploded columns are used only for object search, not for object retrieval.*

*There are exceptions, but let’s ignore them in this webinar.

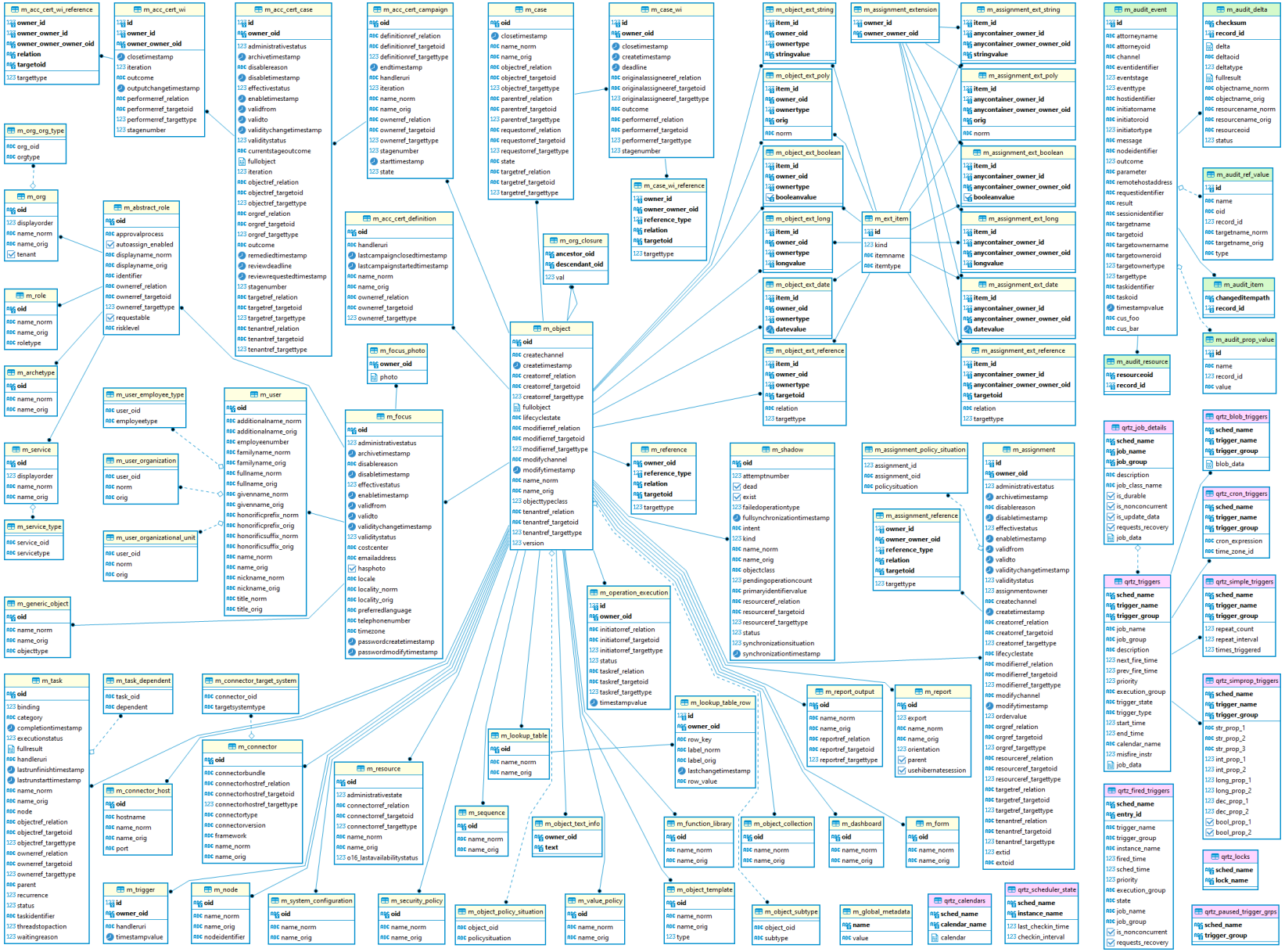
So instead of this...

Evolveum

m_object

oid UUID PRIMARY KEY
fullObject BLOB

...we got to this



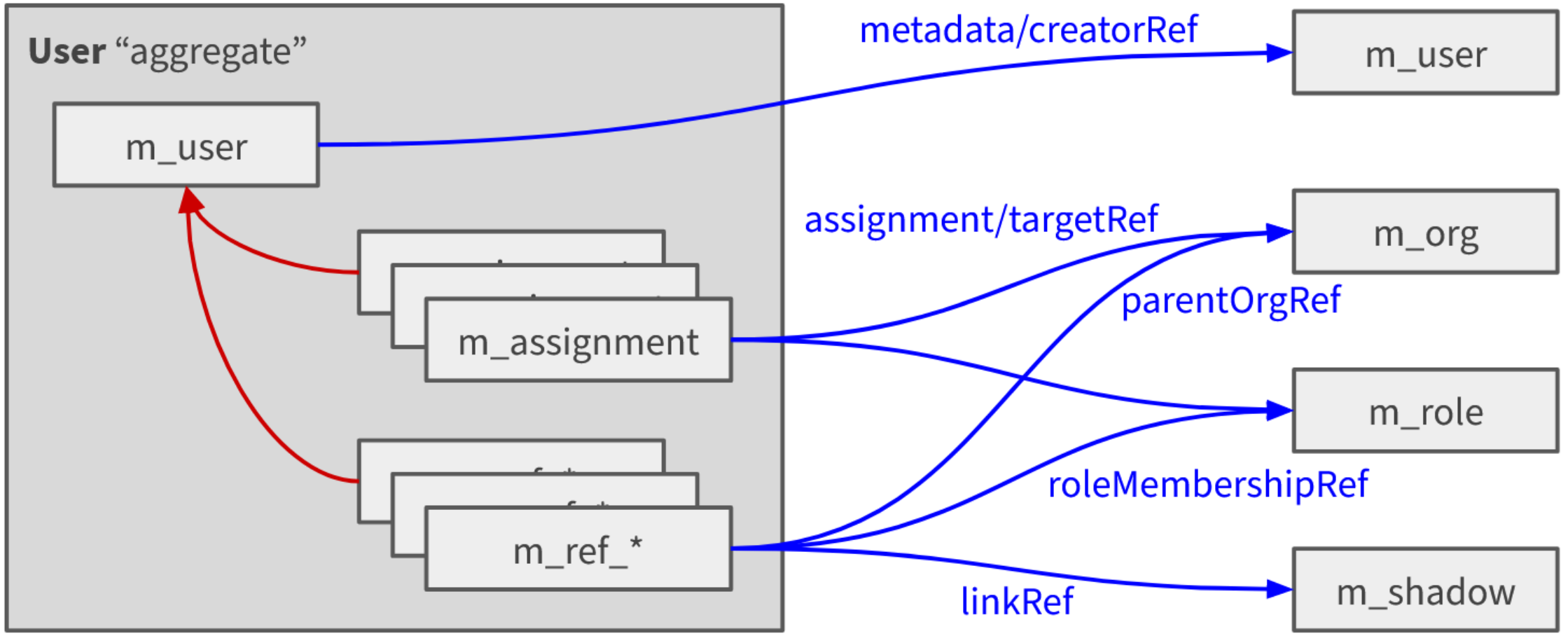
What is fullObject anyway?

Evolveum

- Repository stores serialized form of the object.
- One of typical midPoint serializations (XML, JSON) is used but technically **fullObject serialization is repository implementation detail!**
- When inserting XML object to midPoint it's deserialized first and then re-serialized again in the repository (different formats can be used).
- Object is modified by the repository before it's actually stored:
  - Container IDs are generated, OID is generated if missing as well.
  - Version number is set.

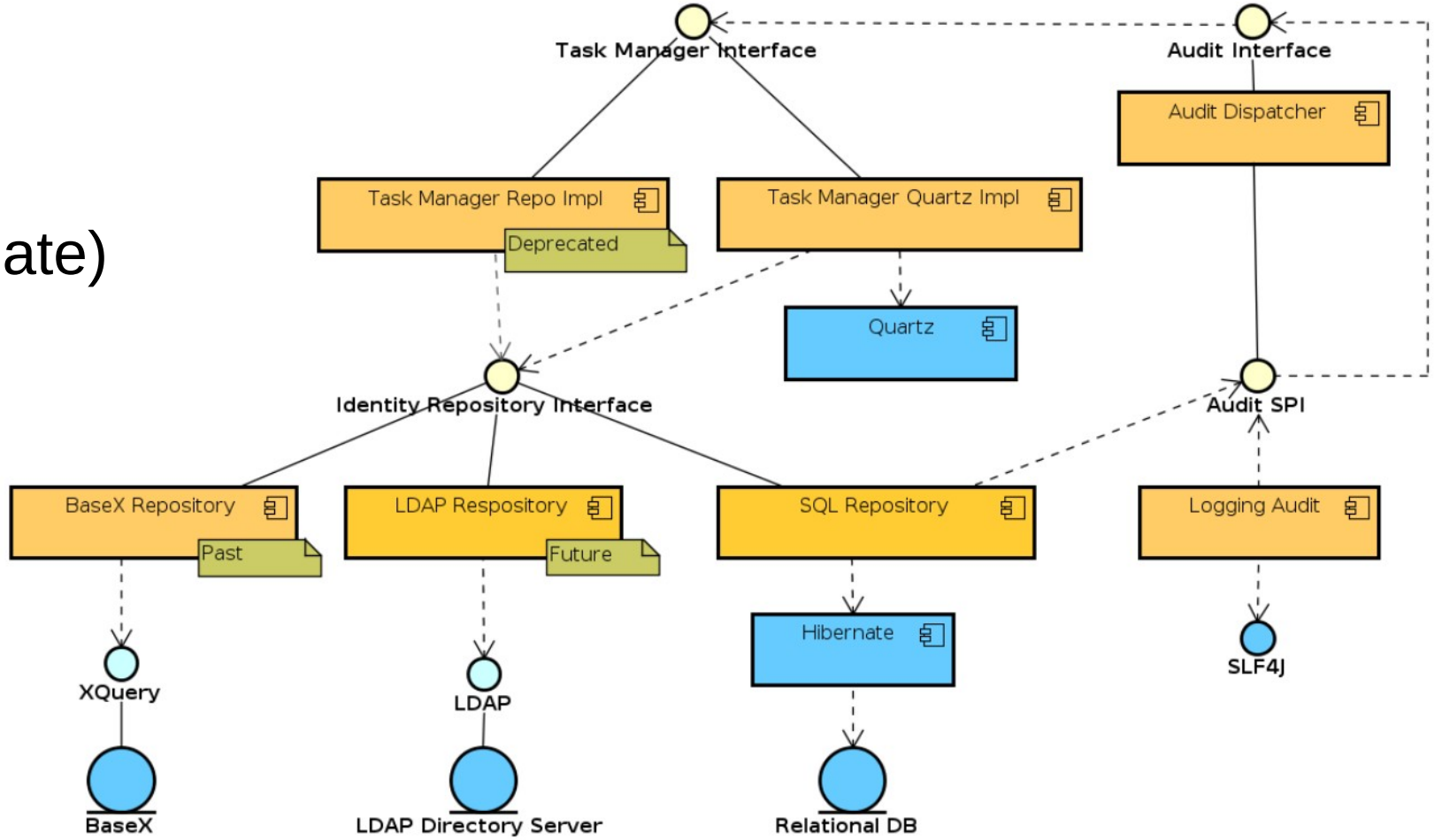
How is object exploded?

- Repository does not maintain strict referential integrity between objects (blue lines).
- It only maintains referential integrity inside the single object (red lines).



Looking back... (with very old picture)

- In 1.8 (Aug 2011)
  XML repository
- In 2.0 (Jun 2012)
  SQL repository (Hibernate)
- In 4.4 (Nov 2021)
  SQL repository reborn



Repository API vs implementation

Evolveum

- Other parts of midPoint depend only on the **Repository API**.
  - Nothing in the midPoint depends on the implementation details.
- Repository implementation depends only on low-level base parts of midPoint, its schema, Prism, etc.
- This design allows for repository implementation switch.
  - It may not be easy, but the boundaries are clear.

Repository vs Model

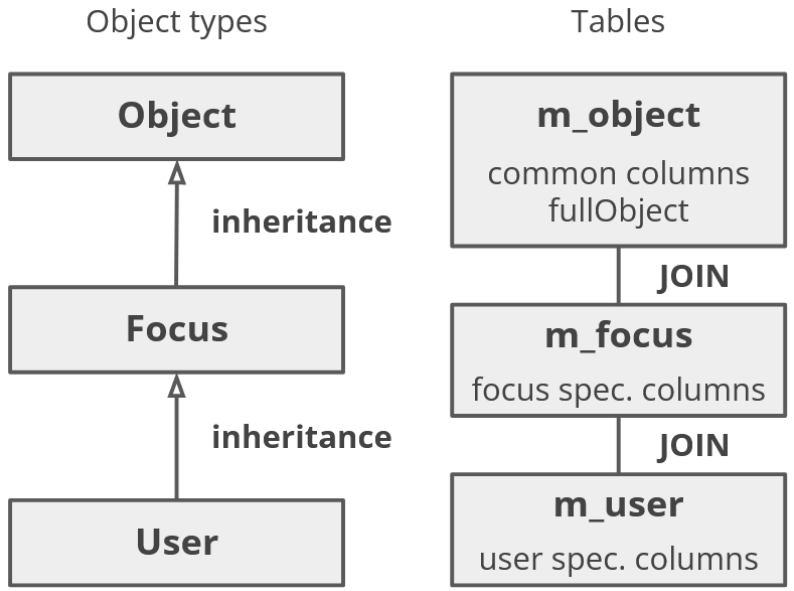
Evolveum

- User rarely talks directly to the Repository API.
- Instead, **Model** API is used in most cases.
  - For example, importing objects calls model, not repository directly.
- Repository API is quite low-level compared to the midPoint Model API.
- But Repository is not totally dumb either:
  - Fills in missing infrastructure information in the object (IDs).
  - Search uses Query API.
  - Updates use Prism deltas.

Why the new Native repository?

Problems of the old repository

Evolveum



- Too many supported databases!
  - We can hardly be experts on all of them
  - The code has many annoying **if**s
- Hibernate (object-relational mapping)
  - It helped to support all the DBs, but...
  - ORM is The Vietnam of Computer Science after all
- Generic support of multiple databases can’t use strength of any of them.
- SQL schema required some reorganization.

More problems with the old repository!

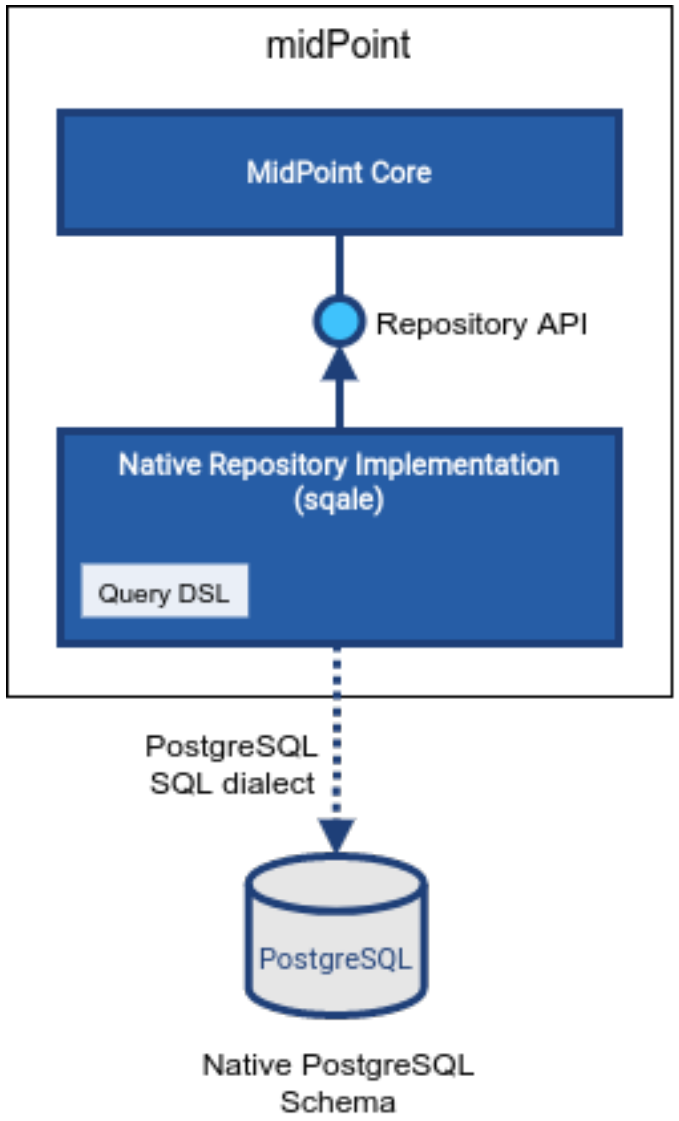
Evolveum

- In modify-heavy deployments transaction serialization is an issue
- **m_object** table had all the **fullObject**s and was more contended
- Many generated queries are inefficient on larger deployments
  - E.g. validity scanner or correlation queries using extensions/attributes
  - Query interpreter generated HQL, final SQL often looked much worse
- Exists filter is tricky, translated as SQL **JOIN**, not **EXISTS**
  - **NOT EXISTS** does not work properly in the old repository
  - **DISTINCT** is often required to remove duplicated results

Between revolution and evolution

Evolveum

- It's still an SQL database – but it's **PostgreSQL** only
  - It's the most advanced open source RDBMS
- Hibernate (ORM/JPA) is gone
  - Querydsl is used for direct SQL query generation
- Table structure uses PostgreSQL inheritance
- We can utilize Postgres types like JSONB and arrays

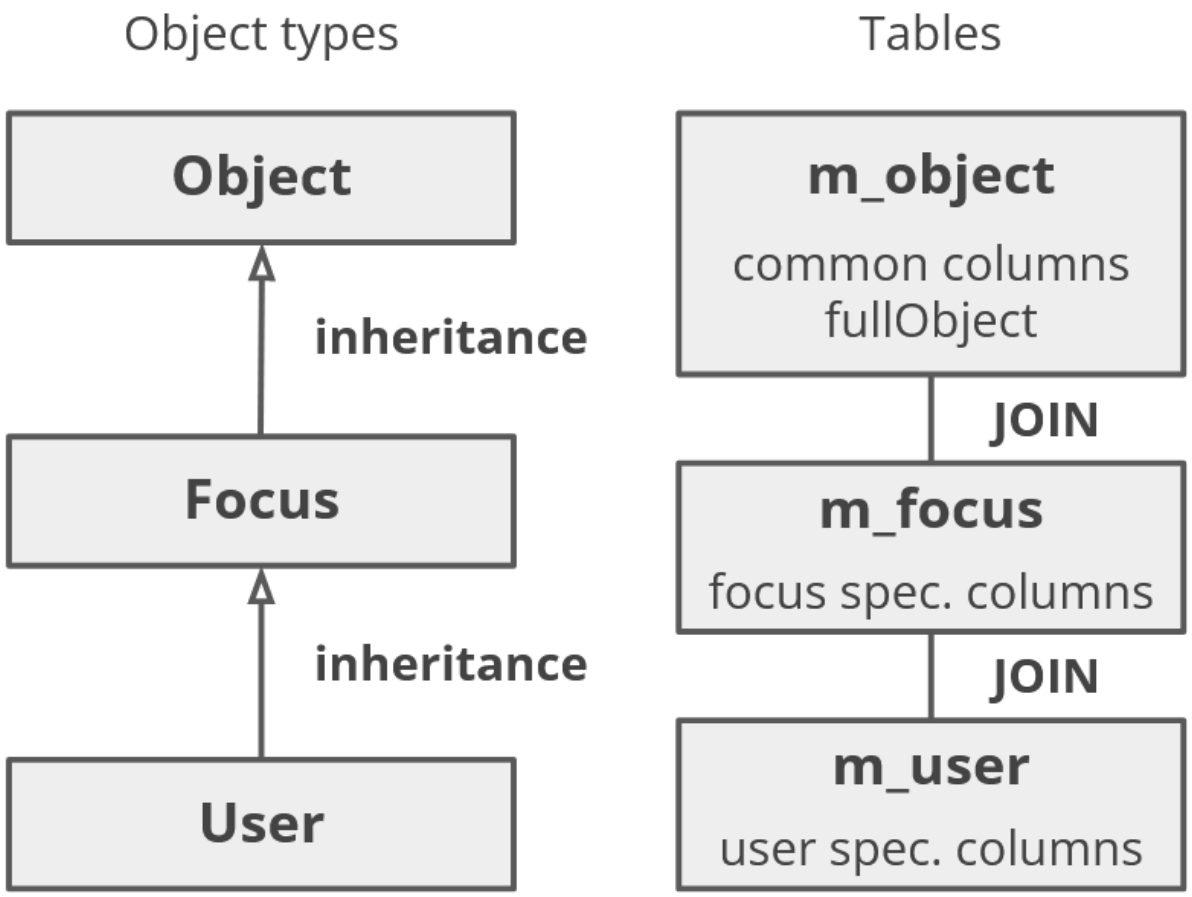


Native repository anatomy

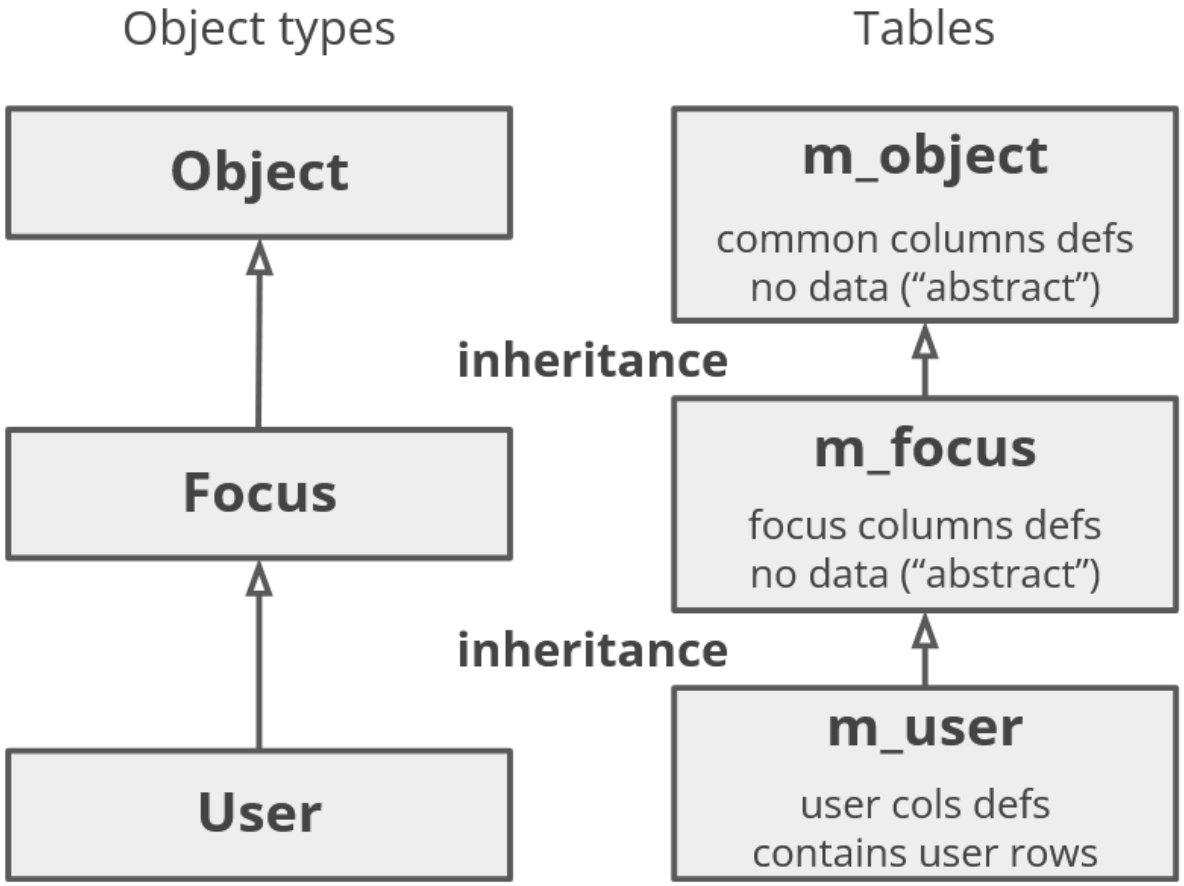
Table structure comparision

Evolveum

- Old repository:



- New repository:



Query example

Evolveum

```
<q:filter>
 <q:exists>
  <q:path>c:assignment</q:path>
   <q:filter>
    <q:or>
     <q:and>
      <q:greater>
       <q:path>c:activation/c:validFrom</q:path>
       <q:value xsi:type="xsd:dateTime">2021-01-01T00:00:00.000Z</q:value>
      </q:greater>
      <q:lessOrEqual>
       <q:path>c:activation/c:validFrom</q:path>
       <q:value xsi:type="xsd:dateTime">2021-06-01T00:00:00.000Z</q:value>
      </q:lessOrEqual>
     </q:and>
     <q:and>
      <q:greater>
       <q:path>c:activation/c:validTo</q:path>
       <q:value xsi:type="xsd:dateTime">2021-01-01T00:00:00.000Z</q:value>
      </q:greater>
      <q:lessOrEqual>
       <q:path>c:activation/c:validTo</q:path>
       <q:value xsi:type="xsd:dateTime">2021-06-01T00:00:00.000Z</q:value>
      </q:lessOrEqual>
     </q:and>
    </q:or>
   </q:filter>
  </q:exists>
 </q:filter>
```

Query comparision

Evolveum

- Old repository:

```
SELECT
  ruser0_.oid AS col_0_0_,
  ruser0_2_.fullobject AS col_1_0_
FROM m_user ruser0_
  INNER JOIN m_focus ruser0_1_
    ON ruser0_.oid = ruser0_1_.oid
  INNER JOIN m_object ruser0_2_
    ON ruser0_.oid = ruser0_2_.oid
  LEFT OUTER JOIN m_assignment assignment1_
    ON ruser0_.oid = assignment1_.owner_oid
      AND ( assignment1_.assignmentowner =:1 )
WHERE assignment1_.validfrom >:2
  AND assignment1_.validfrom <=:3
  OR assignment1_.validto >:4
  AND assignment1_.validto <=:5
order by nlssort(ruser0_.oid,'NLS_SORT=BINARY_AI') asc
fetch first :6 rows only
```

- New repository:

```
select
  u.oid,
  u.fullObject
from m_user u



where exists (select 1
  from m_assignment a
  where u.oid = a.ownerOid
    and a.containerType = $1
    and (a.validFrom > $2
      and a.validFrom <= $3
      or a.validTo > $4
      and a.validTo <= $5))
order by u.oid asc
limit $6
```

Schema files (`docs/config/sql`)

Evolveum

- Old repository (**`generic-old`**):
  - Initialize and upgrade scripts
  - Various DBs

…

```
oracle-4.4-all.sql
oracle-upgrade-4.0-4.4.sql
oracle-upgrade-4.3-4.4.sql
postgresql-4.4-all.sql
postgresql-upgrade-4.0-4.4.sql
postgresql-upgrade-4.3-4.4.sql
```

…

- New repository (**`native-new`**):
  - Initialize and upgrade scripts
  - Single DB, various repo parts

```
postgres-new.sql
postgres-new-audit.sql
postgres-new-quartz.sql
postgres-new-upgrade.sql
postgres-new-upgrade-audit.sql
```

- Useful comments inside!

https://github.com/Evolveum/midpoint/tree/master/config/sql

Repository, audit and scheduler tables (or DBs!)

Evolveum

- There are three distinct parts of the repository:
  - *The* repository, or main repository, storing midPoint objects
  - Audit tables for SQL audit trail
  - Scheduler (Quartz) tables
- By default, MidPoint creates a single connection pool for all parts of the repository.
  - Doesn’t require so many connections in total, better control.
- Each part can be separated in its own database, even separate servers.
  - But main and audit repository must be PostgreSQL.

List of connections with single connection pool

Evolveum

```
midpoint=# select pid, datname, usename, application_name, client_addr, backend_start, state
midpoint-# from pg_stat_activity
midpoint-# where client_addr is not null
midpoint-# order by datname, usename, backend_start;

 pid  | datname  | usename  | application_name | client_addr  |         backend_start         | state
------+----------+----------+------------------+--------------+-------------------------------+--------
 1501 | midpoint | midpoint | mp-repo          | 192.168.56.1 | 2022-01-11 21:52:23.210279+00 | idle
 1505 | midpoint | midpoint | mp-repo          | 192.168.56.1 | 2022-01-11 21:52:52.475241+00 | idle
 1507 | midpoint | midpoint | mp-repo          | 192.168.56.1 | 2022-01-11 21:52:52.564232+00 | idle
```

- New repository also sets nice **application_name** for the connection.

List of connections with separate audit DB

Evolveum

```
midpoint=# select pid, datname, usename, application_name, client_addr, backend_start, state
midpoint-# from pg_stat_activity
midpoint-# where client_addr is not null
midpoint-# order by datname, usename, backend_start;
```

pid	datname	usename	application_name	client_addr	backend_start	state
1791	**midaudit**	**midaudit**	**mp-audit**	192.168.56.1	2022-01-11 22:15:59.053225+00	idle
1792	midaudit	midaudit	mp-audit	192.168.56.1	2022-01-11 22:15:59.185445+00	idle
1793	midaudit	midaudit	mp-audit	192.168.56.1	2022-01-11 22:15:59.20636+00	idle
1794	midaudit	midaudit	mp-audit	192.168.56.1	2022-01-11 22:15:59.222589+00	idle
1795	midaudit	midaudit	mp-audit	192.168.56.1	2022-01-11 22:15:59.240416+00	idle
1796	midaudit	midaudit	mp-audit	192.168.56.1	2022-01-11 22:15:59.257741+00	idle
1797	midaudit	midaudit	mp-audit	192.168.56.1	2022-01-11 22:15:59.275098+00	idle
1798	midaudit	midaudit	mp-audit	192.168.56.1	2022-01-11 22:15:59.289891+00	idle
1790	**midpoint**	**midpoint**	**mp-repo**	192.168.56.1	2022-01-11 22:15:58.28119+00	idle
1802	midpoint	midpoint	mp-repo	192.168.56.1	2022-01-11 22:16:26.895899+00	idle
1803	midpoint	midpoint	mp-repo	192.168.56.1	2022-01-11 22:16:26.923571+00	idle
1804	midpoint	midpoint	mp-repo	192.168.56.1	2022-01-11 22:16:26.983387+00	idle

New repository – main differences

Evolveum

- Works only with PostgreSQL – but utilizes more of its features.
- Scales better and produces more efficient queries.
- Uses PG inheritance for tables, more about schema on the next slide.
- Many filter interpretation improvements
  - **NOT EXISTS** works properly
  - Multi-value **EQ** support improvements, both left and right side
  - Query conditions use subqueries (**EXISTS**) instead of **JOIN**, which does not require **DISTINCT** that much anymore.
- Single iterative search method is used, **iterationMethod** is ignored.

New repository – SQL schema differences

Evolveum

- PG inheritance is used for object and container tables.
- Concrete object table (e.g. **m_user**) now contains all the columns and its data (with related containers and refs, of course), including **fullObject**.
- Different reference types are in separate tables, not in a single table.
- Extensions are stored in JSONB **ext** columns (inline).
  - There are fewer tables, but they may be larger (but TOAST may help).
  - Future may bring other storage options for extensions/attributes.
- Many simple multi-values are stored inline as arrays or **JSONB**.

New repository – column type differences

Evolveum

- **OID column is now UUID, not VARCHAR!**
  - UUID represents 16 bytes/128 bits label, only **hexadecimal chars** (and dashes) can appear in its string representation: https://en.wikipedia.org/wiki/Universally_unique_identifier
- **TEXT** is used instead of all limited **VARCHAR** columns, PG is fine with it.
- Custom enumeration types are used, much easier to read.
- Repeated URIs are stored in **m_uri** table (OK, harder to read).

```
select oid, objecttype, nameorig, administrativestatus, createchannelid from m_focus;
+------------------------------------+----------+------------------------------+--------------------+---------------+
|oid                                 |objecttype|nameorig                      |administrativestatus|createchannelid|
+------------------------------------+----------+------------------------------+--------------------+---------------+
|00000000-0000-0000-0000-000000000002|USER      |administrator                 |ENABLED             |1              |
|00000000-0000-0000-0000-000000000004|ROLE      |Superuser                     |                    |1              |
|00000000-0000-0000-0000-000000000600|SERVICE   |Internal                      |                    |1              |
|00000000-0000-0000-0000-000000000300|ARCHETYPE |System user                   |                    |1              |
```

Object items vs columns and tables

Evolveum

- Columns on object tables (e.g. **m_user**, including those defined in **m_focus** and **m_object**) are single-value items (properties or references) of the object itself.
  - Multiple columns can cover single property (e.g. poly-string like **nameOrig** and **nameNorm**) or reference.
  - Items of nested singleton containers are also inside object table, e.g. column **createTimestamp** for **metadata/createTimestamp**.
- Multi-value containers are stored in separate tables, e.g. **m_assignment**.
- Multi-value properties can be stored in array or JSONB columns (inline).

New repository – serialized form differences

Evolveum

- Default serialized object form is unformatted JSON.
  - This affects various **fullObject** columns (and **delta** in audit).
- No compression of serialized forms on the application side.
- Easier to access the uncompressed data via SQL.
  - But the form is still considered an implementation detail. ;-)
- Postgres compresses the data transparently depending on the size threshold.
  - It also stores the data „out of line“, if necessary, see TOAST for more.
    https://www.postgresql.org/docs/current/storage-toast.html

Special m_object columns

Evolveum

- **version** for optimistic locking concurrency control, it is stored in the prism object, but is managed strictly by repository which increments it during modifications
- **cid_seq** internal sequence for container IDs, assigned by the repository
- **ext** stores searchable indexed/extension attributes; can be implemented as indexed JSONB column or by additional tables (entity-attribute-value, EAV model)
- **db_created/modified** purely database managed columns, not accessible by the application
- **objectType** designates object type (read-only column)

Describe m_user

Evolveum

```
midpoint=# \d m_user
         Column          |            Type             | Nullable |       Default
-------------------------+-----------------------------+----------+---------------------
 oid                     | uuid                        | not null |
 objecttype              | objecttype                  | not null |
                                    generated always as ('USER'::objecttype) stored
 nameorig                | text                        | not null |
 namenorm                | text                        | not null |
 fullobject              | bytea                       |          |
 tenantreftargetoid      | uuid                        |          | -- three cols per ref
 tenantreftargettype     | objecttype                  |          |
 tenantrefrelationid     | integer                     |          |
 lifecyclestate          | text                        |          |
 cidseq                  | bigint                      | not null | 1
 version                 | integer                     | not null | 1
 policysituations        | integer[]                   |          |
 subtypes                | text[]                      |          |
 fulltextinfo            | text                        |          |
 ext                     | jsonb                       |          |
...
 createchannelid         | integer                     |          |
 createtimestamp         | timestamp with time zone    |          |
...
 db_created              | timestamp with time zone    | not null | CURRENT_TIMESTAMP
 db_modified             | timestamp with time zone    | not null | CURRENT_TIMESTAMP
 costcenter              | text                        |          | -- m_focus columns
 emailaddress            | text                        |          |
 photo                   | bytea                       |          |
...
 passwordcreatetimestamp | timestamp with time zone    |          |
 passwordmodifytimestamp | timestamp with time zone    |          |
 administrativestatus    | activationstatustype        |          |
...
 validitystatus          | timeintervalstatustype      |          |
 validfrom               | timestamp with time zone    |          |
 validto                 | timestamp with time zone    |          |
 validitychangetimestamp | timestamp with time zone    |          |
 archivetimestamp        | timestamp with time zone    |          |
 lockoutstatus           | lockoutstatustype           |          |
 additionalnameorig      | text                        |          | -- m_user
 additionalnamenorm      | text                        |          |
 employeenumber          | text                        |          |
...
 titleorig               | text                        |          |
 titlenorm               | text                        |          |
 organizations           | jsonb                       |          | -- polys
 organizationunits       | jsonb                       |          | -- polys
Indexes:
    "m_user_pkey" PRIMARY KEY, btree (oid)
    "m_user_employeenumber_idx" btree (employeenumber)
    "m_user_ext_idx" gin (ext)
    "m_user_ext_org_unit" btree ((ext ->> '14'::text)) -- custom index!
    "m_user_familynameorig_idx" btree (familynameorig)
    "m_user_fullnameorig_idx" btree (fullnameorig)
    "m_user_fulltextinfo_idx" gin (fulltextinfo gin_trgm_ops)
    "m_user_givennameorig_idx" btree (givennameorig)
    "m_user_namenorm_key" UNIQUE, btree (namenorm)
    "m_user_nameorig_idx" btree (nameorig)
    "m_user_organizations_idx" gin (organizations)
    "m_user_organizationunits_idx" gin (organizationunits)
    "m_user_policysituation_idx" gin (policysituations gin__int_ops)
    "m_user_subtypes_idx" gin (subtypes)
...
Foreign-key constraints: -- m_object_oid works as a unique OID pool
    "m_user_oid_fkey" FOREIGN KEY (oid) REFERENCES m_object_oid(oid)
...
Inherits: m_focus
```

Select m_user

Evolveum

```
midpoint=# select oid, objecttype, nameorig,
substring(convert_from(fullobject, 'UTF8'), 1, 100) fullobject, -- making it readable in psql
pg_column_size(fullobject) fo_size, length(fullobject) fo_len,
ext, subtypes, emailaddress, length(photo) photo_len,
createtimestamp, effectiveStatus, validityStatus, db_modified
from m_user limit 1;
-[ RECORD 1 ]----+-----------------------------------------------------------------------
oid              | 2018557c-4f30-4b31-8550-c61c05bdaecb
objecttype       | USER
nameorig         | n04881d
fullobject       | {"user":{"oid":"2018557c-4f30-4b31-8550-c61c05bdaecb","version":"8",
  "name":"n04881d","subtype":"default","extension":"givenNameAccented":"Michelle","familyName
fo_size          | 1551 -- obviously compressed
fo_len           | 3946
ext              | {"5": "2021-10-09T23:43:03.055Z", "7": "2021-10-09T23:43:03.055Z", "8": "a",
                   "12": "0", "13": "n", "14": "4347914", "15": "Michelle", "16": ["FARRELL"]}
subtypes         | {default} -- array
emailaddress     |
photo_len        |
createtimestamp  | 2021-10-10 05:08:38.357+00
effectivestatus  | ENABLED
validitystatus   |
db_modified      | 2021-10-10 05:08:38.88718+00
```

Using and tuning Native repository

Getting started

Evolveum

- Follow our document **Using Native PostgreSQL Repository**.
  https://docs.evolveum.com/midpoint/reference/repository/native-postgresql/usage/
- Typical post-installation configuration
- Decide if you want audit and main repository together or separate.
- Use **doc/config/config-native.xml** as a starting point for **config.xml**.
- Examples are examples, use Repository Configuration document to finish your configuration.
  https://docs.evolveum.com/midpoint/reference/repository/configuration/

Default config.xml

Evolveum

```
<configuration>
  <midpoint>
    <webApplication>
      <importFolder>${midpoint.home}/import</importFolder>
    </webApplication>
    <repository>
      <!--
      Uncomment this section to use the new Native repository (and comment the rest).
      For more see: https://docs.evolveum.com/midpoint/reference/repository/configuration/
      Don't forget to switch Sql/Sqale audit service factory accordingly (lower in this config).
      <type>native</type>
      <jdbcUrl>jdbc:postgresql://localhost:5432/midpoint</jdbcUrl>
      <jdbcUsername>midpoint</jdbcUsername>
      <jdbcPassword>password</jdbcPassword>
      -->

      <!-- Old Generic repository configured for embedded H2 for quick start. -->
      <repositoryServiceFactoryClass>com.evolveum.midpoint.repo.sql.SqlRepositoryFactory</repositoryServiceFactoryClass>
      <baseDir>${midpoint.home}</baseDir>
      <asServer>true</asServer>
    </repository>
...
```

Default config.xml

Evolveum

```
...
<audit>
  <auditService>
    <auditServiceFactoryClass>com.evolveum.midpoint.audit.impl.LoggerAuditServiceFactory</auditServiceFactoryClass>
  </auditService>
  <auditService>
    <!-- Use SqlAuditServiceFactory for old generic repository and SqaleAuditServiceFactory for new Native one. -->
    <auditServiceFactoryClass>com.evolveum.midpoint.repo.sql.SqlAuditServiceFactory</auditServiceFactoryClass>
    <!--
    <auditServiceFactoryClass>com.evolveum.midpoint.repo.sqale.audit.SqaleAuditServiceFactory</auditServiceFactoryClass>
    -->
  </auditService>
</audit>
```

config.xml for Native repository

Evolveum

```
<configuration>
  <midpoint>
    <webApplication>
      <importFolder>${midpoint.home}/import</importFolder>
    </webApplication>
    <repository>
      <type>native</type>
      <jdbcUrl>jdbc:postgresql://localhost:5432/midpoint</jdbcUrl>
      <jdbcUsername>midpoint</jdbcUsername>
      <jdbcPassword>password</jdbcPassword>
    </repository>
    <audit>
      <auditService>
         <auditServiceFactoryClass>com.evolveum.midpoint.audit.impl.LoggerAuditServiceFactory</auditServiceFactoryClass>
      </auditService>
      <auditService>
         <auditServiceFactoryClass>com.evolveum.midpoint.repo.sqale.audit.SqaleAuditServiceFactory</auditServiceFactoryClass>
      </auditService>
    </audit>
...
  </midpoint>
</configuration>
```

Sizing the database server is... complicated

Evolveum

- Basic recommendations:
  https://docs.evolveum.com/midpoint/install/system-requirements/#sizing-of-database-system
- Sizing is disk sizing, performance sizing (CPU, memory, IO)
  - Virtualization may give some flexibility.
- Postgres configuration tweaks depending on the size:
  - Default PG settings are very conservative – on the low-end.
  - Use your PG experts if possible.
  - Use some calculator as a starting point, for example:
    https://pgtune.leopard.in.ua/ (use OLTP or Mixed as „DB Type“)

Using calculator from pgtune.leopard.in.ua

Evolveum

Parameters of your system

DB version — what is this?
14

OS Type — what is this?
Linux

DB Type — what is this?
Mixed type of application

Total Memory (RAM) — what is this?
16 GB

Number of CPUs — what is this?
8

Number of Connections — what is this?
100

Data Storage — what is this?
SSD storage

Generate

postgresql.conf | ALTER SYSTEM

Add/modify this settings in **postgresql.conf** and restart database

```
# DB Version: 14
# OS Type: linux
# DB Type: mixed
# Total Memory (RAM): 16 GB
# CPUs num: 8
# Connections num: 100
# Data Storage: ssd

max_connections = 100
shared_buffers = 4GB
effective_cache_size = 12GB
maintenance_work_mem = 1GB
checkpoint_completion_target = 0.9
wal_buffers = 16MB
default_statistics_target = 100
random_page_cost = 1.1
effective_io_concurrency = 200
work_mem = 5242kB
min_wal_size = 1GB
max_wal_size = 4GB
max_worker_processes = 8
max_parallel_workers_per_gather = 4
max_parallel_workers = 8
max_parallel_maintenance_workers = 4
```

Sizing the storage

Evolveum

- Use your current DB storage size info.
- Native repository should be a bit smaller.
- Don’t just use XML sizes, use actual database storage size.
- Indexes take a lot of room too.
- Often, audit tables take most of the space.
  - That is also a good reason to use a separate audit database.

Table and database size example

Evolveum

```
midpoint=# SELECT oid, table_schema, table_name, row_estimate, pg_size_pretty(total_bytes) AS total,
  pg_size_pretty(table_bytes) AS table, pg_size_pretty(toast_bytes) AS toast, pg_size_pretty(index_bytes) AS index
FROM (SELECT *, total_bytes - index_bytes - COALESCE(toast_bytes, 0) AS table_bytes
    FROM (SELECT c.oid, nspname AS table_schema, relname AS table_name, c.reltuples::bigint AS row_estimate,
      pg_total_relation_size(c.oid) AS total_bytes, pg_indexes_size(c.oid) AS index_bytes, pg_total_relation_size(reltoastrelid) AS toast_bytes
        FROM pg_class c LEFT JOIN pg_namespace n ON n.oid = c.relnamespace WHERE relkind = 'r') a ) b
WHERE table_schema = 'public'
ORDER BY total_bytes DESC limit 10;
  oid   | table_schema |       table_name        | row_estimate |  total   |  table   |   toast    |  index
--------+--------------+-------------------------+--------------+----------+----------+------------+---------
 18526 | public       | ma_audit_delta_default  |    187152384 | 508 GB   | 260 GB   | 233 GB     | 15 GB
 17337 | public       | m_shadow                |     87694848 | 165 GB   | 146 GB   | 32 kB      | 19 GB
 17105 | public       | m_user                  |     27546152 | 77 GB    | 38 GB    | 27 GB      | 12 GB
 18156 | public       | m_assignment            |    127124184 | 33 GB    | 15 GB    | 8192 bytes | 18 GB
 17064 | public       | m_ref_projection        |     82952200 | 17 GB    | 6482 MB  |            | 11 GB
 18509 | public       | ma_audit_event_default  |     19523042 | 12 GB    | 8260 MB  | 8192 bytes | 3663 MB
 16877 | public       | m_object_oid            |    116446608 | 9427 MB  | 4932 MB  |            | 4495 MB
 17026 | public       | m_ref_role_membership   |     38737040 | 6271 MB  | 2925 MB  |            | 3346 MB
 17270 | public       | m_ref_object_parent_org |      9381595 | 1566 MB  | 741 MB   |            | 825 MB
 17153 | public       | m_role                  |         5006 | 12 MB    | 11 MB    | 56 kB      | 1368 kB

midpoint=# SELECT pg_size_pretty(pg_database_size('midpoint'));
 pg_size_pretty
----------------
 829 GB
```

Main repo 309 GB, 116M objects, each object takes ~27 KB on average. Not many assignments here, it can easily be over 100KB.

Different view on DB object sizes

Evolveum

```
midpoint=# SELECT t.oid, CASE WHEN tft.relname IS NOT NULL THEN tft.relname || ' (TOAST)' ELSE t.relname END AS object,
    pg_size_pretty(pg_relation_size(t.oid)) AS size, t.relkind, t.reltuples::bigint as row_estimate, t.relname as object_name
FROM pg_class t
    INNER JOIN pg_namespace ns ON ns.oid = t.relnamespace
    LEFT JOIN pg_class tft ON tft.reltoastrelid = t.oid -- table for toast
    LEFT JOIN pg_namespace tftns ON tftns.oid = tft.relnamespace
WHERE 'public' IN (ns.nspname, tftns.nspname)
ORDER BY pg_relation_size(t.oid) DESC
LIMIT 15;
```

oid	object	size	relkind	row_estimate	object_name
18526	ma_audit_delta_default	260 GB	r	187152384	ma_audit_delta_default
18531	ma_audit_delta_default (TOAST)	229 GB	t	168837440	pg_toast_18526
17337	m_shadow	146 GB	r	87694848	m_shadow
17105	m_user	38 GB	r	27546152	m_user
17114	m_user (TOAST)	26 GB	t	12554565	pg_toast_17105
18156	m_assignment	15 GB	r	127124184	m_assignment
18529	**ma_audit_delta_default_pkey**	**15 GB**	**i**	**191936144**	**ma_audit_delta_default_pkey**
17371	**m_shadow_primidval_objcls_resrefoid_key**	**9457 MB**	**i**	**87694848**	**m_shadow_primidval_objcls_resrefoid_key**
18509	ma_audit_event_default	8257 MB	r	19523042	ma_audit_event_default
18163	**m_assignment_pkey**	**6724 MB**	**i**	**127025472**	**m_assignment_pkey**
17069	**m_ref_projection_pkey**	**6541 MB**	**i**	**82952200**	**m_ref_projection_pkey**
17064	m_ref_projection	6480 MB	r	82952200	m_ref_projection
16877	m_object_oid	4930 MB	r	116446608	m_object_oid
16881	**m_object_oid_pkey**	**4495 MB**	**i**	**116521240**	**m_object_oid_pkey**
17076	**m_ref_projection_targetoidrelationid_idx**	**4456 MB**	**i**	**82952200**	**m_ref_projection_targetoidrelationid_idx**

https://docs.evolveum.com/midpoint/reference/repository/native-postgresql/db-maintenance/

Indexing

Evolveum

- Indexes are great, but come at a price (example from our 309 GB DB):

```
midpoint=# SELECT pg_size_pretty(sum(pg_relation_size(t.oid)))
FROM pg_class t INNER JOIN pg_namespace ns ON ns.oid = t.relnamespace
WHERE ns.nspname = 'public' and t.relkind='i' and t.relname like 'm\_%';
 pg_size_pretty
----------------
 68 GB -- that's 22%
```

- Size is the least problem, but updates need to refresh indexes, they need to be vacuumed too, etc.
- Most important columns have B-tree indexes or other suitable indexes.
  - Not all columns have indexes though… but they are still searchable.
- Identify slow queries for your cases and add indexes accordingly.

Use pg_stat_statements to identify slow queries

Evolveum

```
-- list of selects using the most time, change order to get other avg/max/calls to top
-- NOTE: postgresql.conf must have (+restart): shared_preload_libraries = 'pg_stat_statements'
-- Also first, to see pg_stat_statements table: CREATE EXTENSION IF NOT EXISTS pg_stat_statements;
-- To reset collected statistics: select pg_stat_statements_reset();
select
    (total_exec_time / 1000 / 60)::numeric(20,4) as total_min, -- min as minutes here
    mean_exec_time::numeric(20,2) as avg_ms,
    max_exec_time::numeric(20,2) as max_ms,
    calls,
    (rows / calls)::numeric(20) as avg_rows,
    (100.0 * shared_blks_hit / nullif(shared_blks_hit + shared_blks_read, 0))::numeric(20,2) AS hit_percent,
    query
from pg_stat_statements
-- optional where to limit it to one database, if needed (e.g. shared/test DB)
-- where dbid = (select oid from pg_database where datname = 'midpoint')
order by 1 desc -- change order as necessary
limit 50;
```

Extension indexing

Evolveum

- Extensions and shadow attributes are stored in JSONB columns.
- By default, **ext** column and **attributes** in **m_shadow** use only GIN index.
  - This is fine for EQ filter which covers most of the cases.
- For other cases (comparison, substrings) index needs to be created:
  - Applicable only to single-value extensions or attributes
  - Use index on **ext->'id'** for non-string properties or **ext->>'id'** for strings.
  - Consult **m_ext_item** table to find the id of the extension/attribute item.
  - Use normal B-tree index for comparisons or trigram index for substrings.

Extension indexing example

Evolveum

- Query filter:

```
<q:filter><q:substring>
  <q:matching>stringIgnoreCase</q:matching>
  <q:path>c:extension/string</q:path>
  <q:value xsi:type="xsd:string">VaLuE</q:value>
  <q:anchorEnd>true</q:anchorEnd>
</q:substring></q:filter>
```

- Select:

```
select u.oid, u.fullObject from m_user u where u.ext->>'195' ilike $1 -- $1 = '%VaLuE'
```

- Good index:

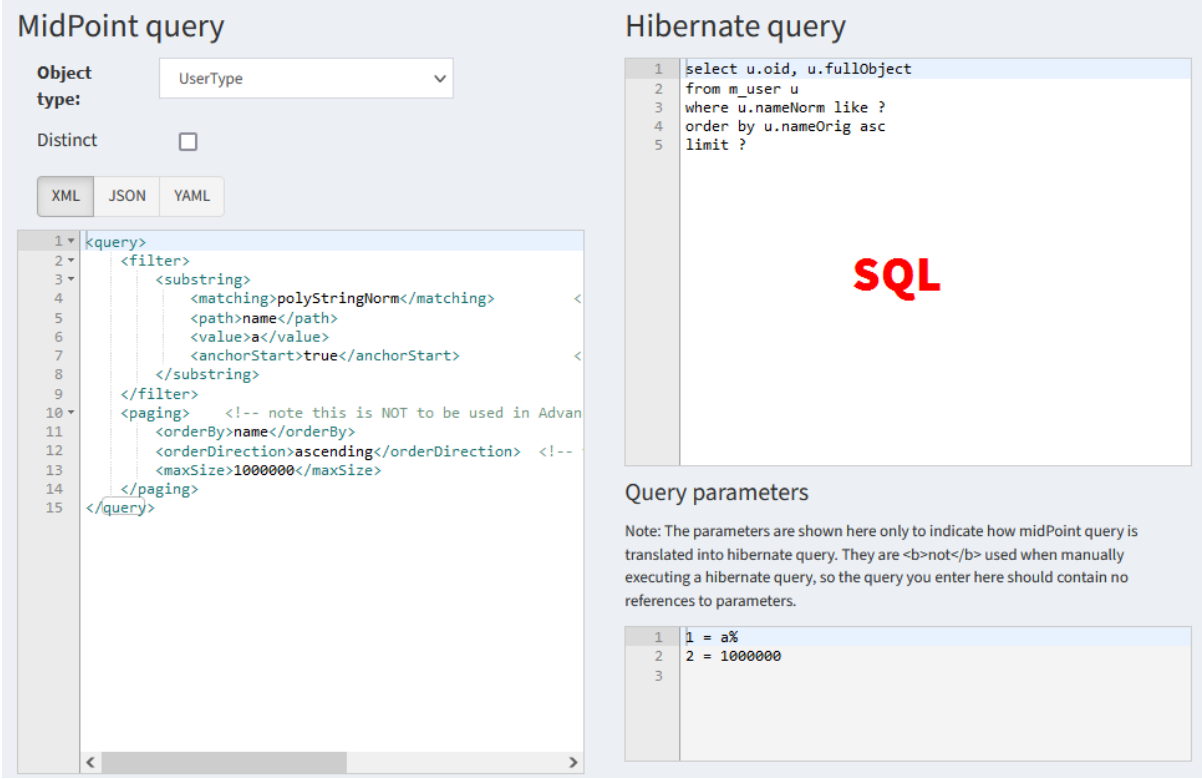
```
CREATE INDEX m_user_ext_string_trgm_idx ON m_user USING gin((ext->>'195') gin_trgm_ops);
```

- “How do I get that 195?”

```
select * from m_ext_item where itemname like '%#string';
|id |itemname                    |valuetype                                |holdertype|cardinality|
+---+----------------------------+-----------------------------------------+----------+-----------+
|195|https://example.com/p#string|http://www.w3.org/2001/XMLSchema#string|EXTENSION |SCALAR     |
```

Query API tips

- Experiment with Query playground
- Prefer concrete types queries
  - User for users, not Focus or Object
  - Generic queries are less efficient
- Limit result count, e.g.: `<paging><b><maxSize>10</maxSize></b></paging>`
  - Native repository uses implicit limit 10,000 if none is provided for sanity
  - Higher number can be provided explicitly with **maxSize**… but why?!
- Use iterative search mechanisms for queries with longer result lists



Schema upgrade

Evolveum

- Custom procedure is used for schema changes:
  - **apply_change** for main repository
  - **apply_audit_change** for the audit schema
- Upgrade script can be re-run, it applies only the missing changes
- Schema version is now sequential and not semantic
  - Upgrade script can be checked for version comments
- There is no automatic DB upgrade or check for the Native repository
  - Simply run the upgrade scripts from the MP distribution you run

Migration to Native PostgreSQL Repository

Evolveum

- Upgrade to 4.4 using original repository
  - Upgrade possible from 4.0.4 (LTS to LTS) or 4.3.2
- Export existing data using Ninja
- Initialize native repository
- Change midPoint configuration for native repository
- Import previously exported data back to midPoint using Ninja
- Start midPoint with new configuration
- Audit migration with midPoint already up and running
- More in the next webinar!

Breaking repository changes

Evolveum

- New repository does not support H2, you need to install the DB.
  - H2 is unsupported and for testing only anyway.
- OID must be in UUID format – but this was always strongly encouraged!
- Group by filter is not supported (and probably meaningless).
  - And will be removed from Query API.
- Audit/dashboards do not support SQL/HQL queries anymore (since 4.3).
  - Now it uses Query API, just like the main Repository API.
- ...and that's it! Ninja tool will take the care for the rest!

Native audit, partitioning, migration

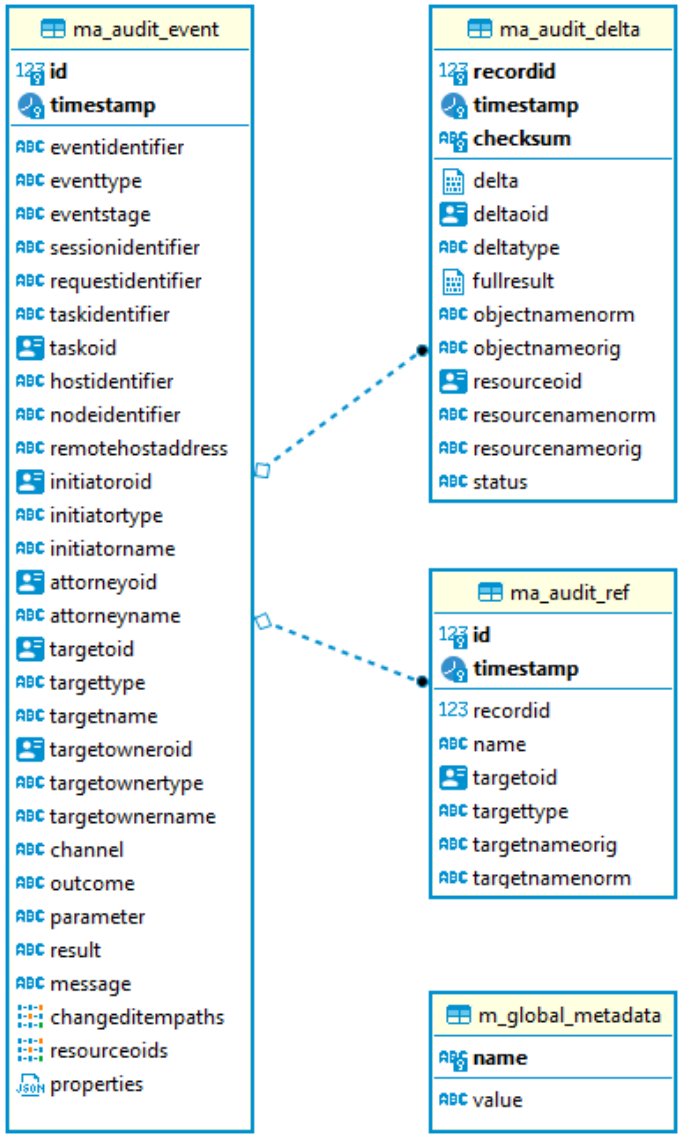
SQL audit overview

Evolveum

- Designed as insert only, no updates!
- Previously, Audit API had reindex operation, this is gone now.
- Insert-only table is much more efficient, no VACUUM needed.
  - Until the cleanup executes some **delete** that is, more on that later.
- It is, after all, an audit trail – but searchable.

Audit tables

- Stores audit event records (containers since 4.2)
  - Tables similar to the old repository, changes similar to those in the main repository
  - Prefix changed to **ma_** for clearer separation.
- No dependency on the main portion of the repo.
  - E.g. **channel** is now **TEXT**, no reference to **m_uri**.
- All tables are partitionable by **timestamp**.
- Delete from **ma_audit_event** cascades to details.



Separate audit database – config.xml

Evolveum

```
<configuration>
  <midpoint>
...
    <audit>
...
      <auditService>
        <auditServiceFactoryClass>
          com.evolveum.midpoint.repo.sqale.audit.SqaleAuditServiceFactory
        </auditServiceFactoryClass>
        <jdbcUrl>jdbc:postgresql://192.168.56.33:5432/midaudit</jdbcUrl>
        <jdbcUsername>midaudit</jdbcUsername>
        <jdbcPassword>password</jdbcPassword>
        <!-- specifying custom application name (available in connection list)
        <jdbcUrl>jdbc:postgresql://192.168.56.33:5432/midaudit?ApplicationName=audit</jdbcUrl>
        or tweaking connection pool
        <maxPoolSize>6</maxPoolSize>
        -->
      </auditService>
    </audit>
...
  </midpoint>
</configuration>
```

SQL audit partitioning

Evolveum

- All three audit tables are partitionable by **timestamp** column.
- By default, only one ***_default** partition is created for each table.
- Run **audit_create_monthly_partitions** procedure to create partitions:
  - Example, 10 years ahead: **call audit_create_monthly_partitions(120);**
  - Or 5 years back (migration): **call audit_create_monthly_partitions(-60);**
  - Currently, partitions are not created automatically.
- Partitions are not query performance solution!

Partitioning for fast audit cleanup

Evolveum

- Partitions are solution for fast data cleanup (drop or detach partition).
- You probably want to remove **auditRecords** from **cleanupPolicy** in **SystemConfiguration** object.
- Using partitions as the sole cleanup mechanism also means that each partition (which is a table) is strictly insert-only.
  - No VACUUM is needed.
- Drawback: Currently the partition management, including cleanup, is manual only.

https://docs.evolveum.com/midpoint/reference/repository/native-audit/#cleanup-task-vs-partitions

Audit migration example

Evolveum

- 1 mil. audit events migrated from Generic PG to Native PG
- Ninja supports audit migration in midPoint 4.4.1
- Use `-z` to zip the output files
- Run multiple ninjas in parallel for export with `repoId` filter. (~1000/s)
- Run ninja with multiple threads, e.g. `-l 4`, for import. (~400/s)
- Original size 5.0 GB with gzipped deltas, new size 3.9 GB with plain deltas (transparently compressed by PG).

https://docs.evolveum.com/midpoint/reference/repository/native-audit/#audit-migration-from-other-database
https://docs.evolveum.com/midpoint/reference/deployment/ninja/

Conclusion

Main takeaways

Evolveum

- There is this new Native PostgreSQL repository.
- It’s better. Consider using it.
  - You still should test it in non-production environment first, of course.
- There is new SQL audit trail. It can be partitioned!
- Repository and Query API documentation was massively updated.

Resources

- MidPoint Repository
  https://docs.evolveum.com/midpoint/reference/repository/
- Native PostgreSQL Repository
  https://docs.evolveum.com/midpoint/reference/repository/native-postgresql/
  https://docs.evolveum.com/midpoint/reference/repository/native-postgresql/usage/
- Native PostgreSQL Audit Trail
  https://docs.evolveum.com/midpoint/reference/repository/native-audit/
- Repository Database Support (Generic vs Native repo explanation)
  https://docs.evolveum.com/midpoint/reference/repository/repository-database-support/
- Query API
  https://docs.evolveum.com/midpoint/reference/concepts/query/query-api/

MidScale

Evolveum

- Target:
  - Tens of millions of identities
- Key results:
  - Improved scalability of midPoint
  - Improved visibility, diagnostic and reliability of midPoint
  - Improved performance and user experience of midPoint user interface

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the NGI_TRUST grant agreement no 825618.

Webinars

Evolveum

- Upgrade Guide, January 27, 2022 by Anton Tkáčik
- Tasks, February 3, 2022 by Pavol Mederly
- Customizing GUI, February 10, 2022 by Katarína Bolemant
- Native reports, February 16, 2022 by Lukáš Škublík

Thank you for your time!

See other talks at https://docs.evolveum.com

Also **follow us** on our social media for further information!

 /Evolveum

 /Evolveum

 /Evolveum

 @Evolveum

 /Evolveum

Evolveum