Migration scripts for OAI-PMH

Populate `complete_updated_date` column in instance table

0. Disable trigger

Trigger should be disabled to not override historical data to be populated in `complete_updated_date` column

```
ALTER TABLE {tenant}_mod_inventory_storage.instance DISABLE TRIGGER updatecompleteupdatedate_instance;
```

1. Prepare step

On this step `uuid_range` table is created in `public` schema to split UUIDs in different partitions (16 partitions in total) and track their status later (completed column).

```
CREATE TABLE public.uuid_range
(
  id serial PRIMARY KEY,
  partition uuid,
  subrange_start uuid,
  subrange_end uuid,
  completed boolean default false
);

DO
$$
DECLARE
  partition_start UUID;
  subrange_start UUID;
  subrange_end UUID;
BEGIN
  FOR i IN 0..15
  LOOP
    partition_start := (rpad(to_hex(i), 8, '0') || '-0000-0000-0000-000000000000')::UUID;
    subrange_start := partition_start;
    FOR j IN 1..4096
    LOOP
      IF i < 15 OR (i = 15 AND j < 4096) THEN
        subrange_end := (to_hex(i)::text || rpad(lpad(to_hex(j), 3, '0'), 7, '0') ||
          '-0000-0000-0000-000000000000')::UUID;
      ELSE
        subrange_end := 'ffffffff-ffff-ffff-ffff-ffffffffffff'::UUID; -- upper bound for last subrange in last partition
      END IF;
    END LOOP;
    INSERT INTO public.uuid_range (partition, subrange_start, subrange_end) VALUES (partition_start, subrange_start, subrange_end);
    subrange_start := subrange_end;
  END LOOP;
END;
$$
LANGUAGE plpgsql;
```

2. Update
Update step should be executed for each partition created from step #1. Update script can be executed concurrently (tests results can be found in PERF-255 - Test OAI-PMH inventory update script without users impact) to understand how many partitions can be processed at the same time on specific env without users impact.

Params to be updated

- tenant - tenant name
- partition - partition value from public.uuid_range table

```sql
set search_path = "{tenant}.mod_inventory_storage", "public";

do
$$
declare
    partition_to_process uuid := '{partition}'::uuid;
    range_rec public.uuid_range%rowtype;
begin
    for range_rec in select ur.*
        from public.uuid_range ur
        where ur.partition = partition_to_process
        and ur.completed = false
        loop
            -- start transaction;
            UPDATE {tenant}.mod_inventory_storage.instance i
            SET complete_updated_date = (jsonb -> 'metadata' ->> 'updatedDate')::timestamp with time zone
            where i.id between range_rec.subrange_start and range_rec.subrange_end;

            update public.uuid_range
            set completed = true
            where id = range_rec.id;
            -- raise notice 'subrange id: % completed', range_rec.id;
            commit;
        end loop;
$$;
```

### 3. Monitor

Next script can be used to monitor how many sub-partitions in partitions are left to be processed

```sql
select partition, count(partition)
from public.uuid_range
where completed = false
group by partition
order by partition;
```

### 4. Post-migration

As initially trigger was disabled, it should be enabled again, additionally index should be created for better performance

```sql
ALTER TABLE {tenant}.mod_inventory_storage.instance ENABLE TRIGGER updatecompleteupdateddate_instance;
CREATE INDEX IF NOT EXISTS idx_instance_complete_updated_date
ON {tenant}.mod_inventory_storage.instance(complete_updated_date);
```