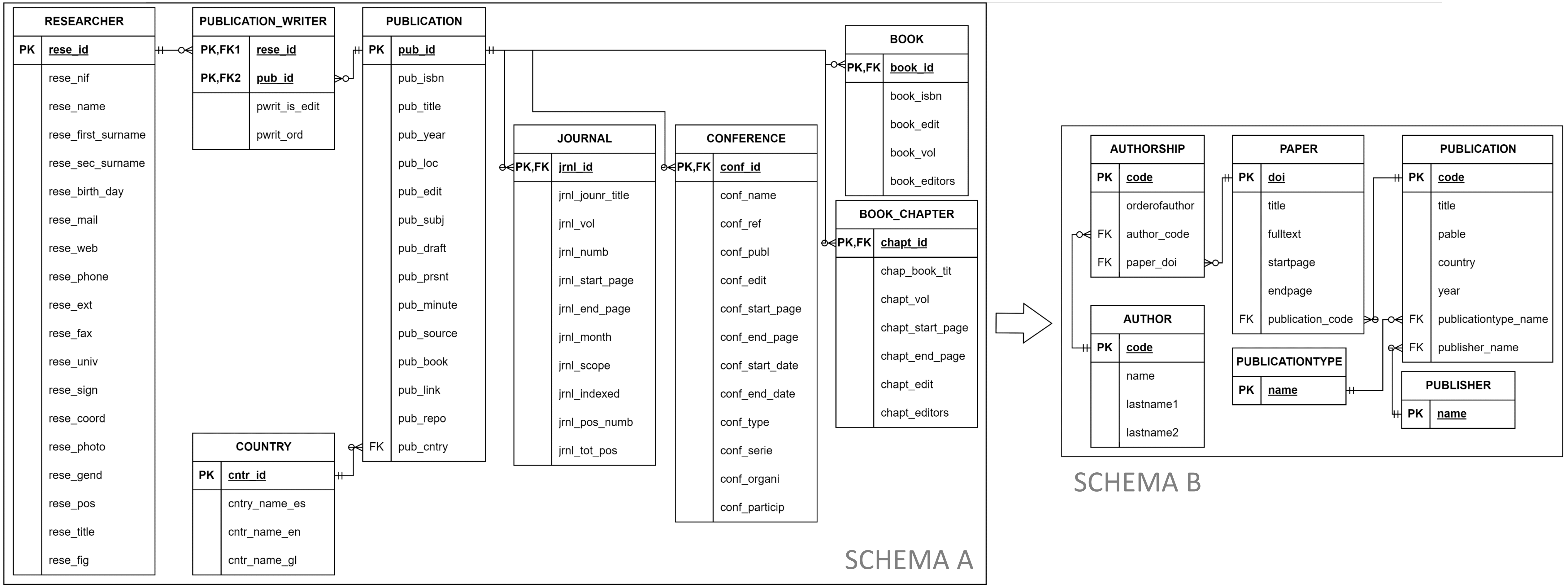
DAMI-Framework: A DSL for Data Migration

USABILITY EVALUATION 2024

GOAL: MIGRATE DATA FROM SCHEMA A TO SCHEMA B



SCHEMA B

DEFINE THE MIGRATION PRODUCT

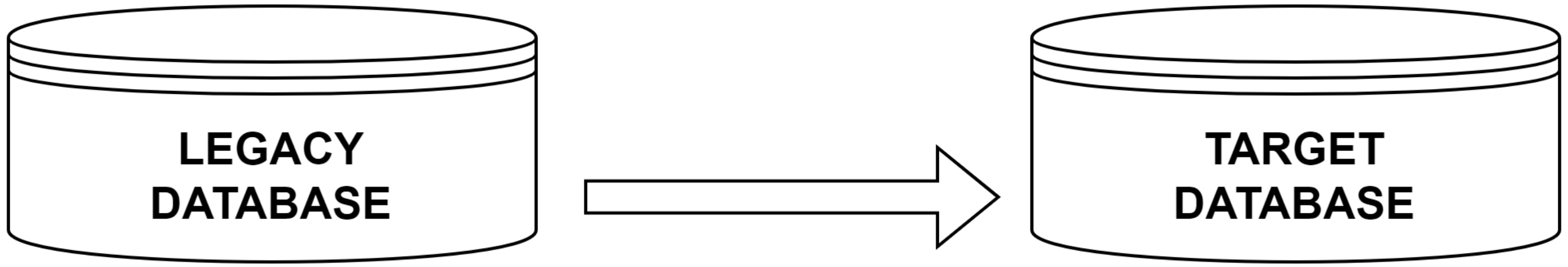
1. DEFINE PRODUCT IN THE DSL:

- `CREATE PRODUCT dsl_experiment;`

Defines an object in JavaScript called “dsl_experiment” which will store:

- Connection details
- Schemas
- Entities
  - Name
  - Columns
  - Mappings
- Relationships

CONNECTING THE DATABASES



1. SPECIFY CONNECTION DATA:

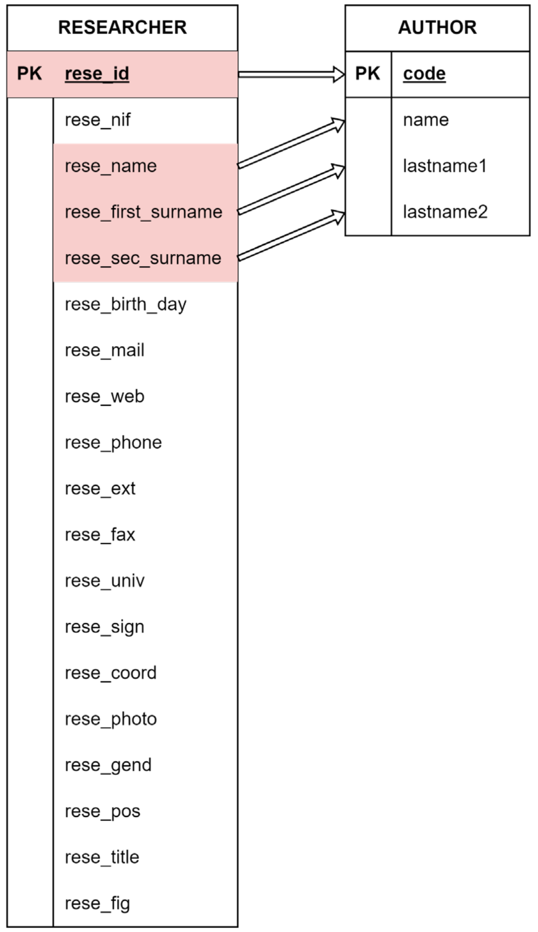
```
CREATE CONNECTION FROM (dbname dsl_bd, host chronos.lbd.org.es, port 5432, user
 postgres, pwd postgres, schema legacy);

CREATE CONNECTION TO (dbname dsl_bd, host chronos.lbd.org.es, port 5432, user
 postgres, pwd postgres, schema target);
```

CONNECTING THE DATABASES

2. SQL CODE FROM DSL TO STABLISH A CONNECTION ON POSTGRESQL:

```
CREATE EXTENSION IF NOT EXISTS dsl_fdw;

CREATE SERVER dsl_server FOREIGN DATA WRAPPER dsl_fdw OPTIONS (host
 'chronos.lbd.org.es', dbname 'dsl_bd', port '5432');

CREATE USER MAPPING FOR CURRENT_USER SERVER dsl_server OPTIONS (user 'postgres',
 password 'postgres');

CREATE SCHEMA legacy;

IMPORT FOREIGN SCHEMA legacy FROM SERVER dsl_server INTO legacy;

CREATE SCHEMA IF NOT EXISTS target AUTHORIZATION postgres;

CREATE SCHEMA IF NOT EXISTS aux AUTHORIZATION postgres;
```

MAPPING FROM LEGACY TABLE TO TARGET TABLE



1. DEFINE MAPPING IN DSL:

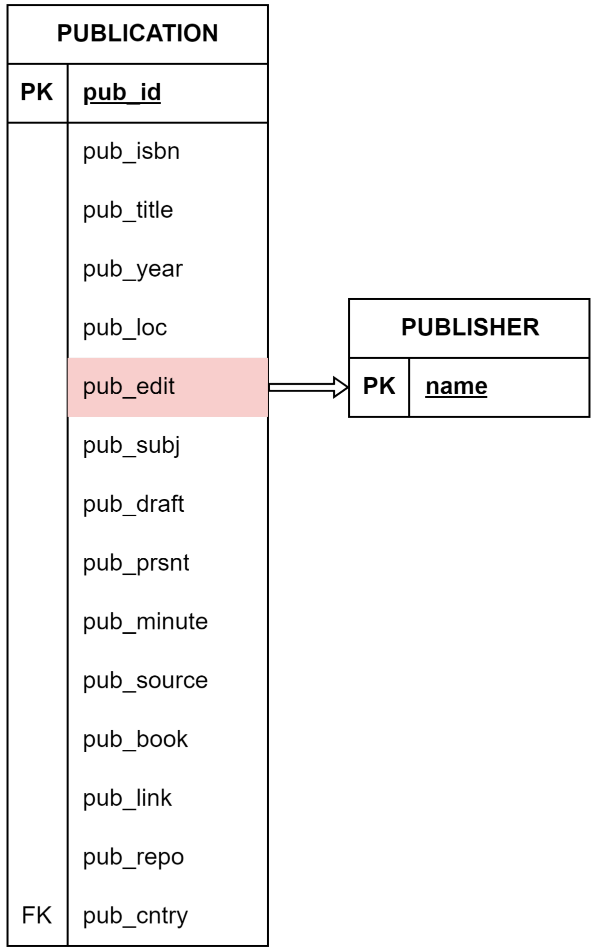
```
MAP researcher TO author (
    rese_id TO code,
    rese_name TO name,
    rese_first_surname TO lastname1,
    rese_sec_surname TO lastname2
);
```

PK(rese_id) = PK(code)

2. SQL CODE GENERATED:

```
INSERT INTO target.author(code,name,lastname1,lastname2)
    SELECT rese_id,rese_name,rese_first_surname,rese_sec_surname
FROM legacy.researcher;
```

MAPPING WITH SQL OPERATION



1. DEFINE MAPPING IN DSL:

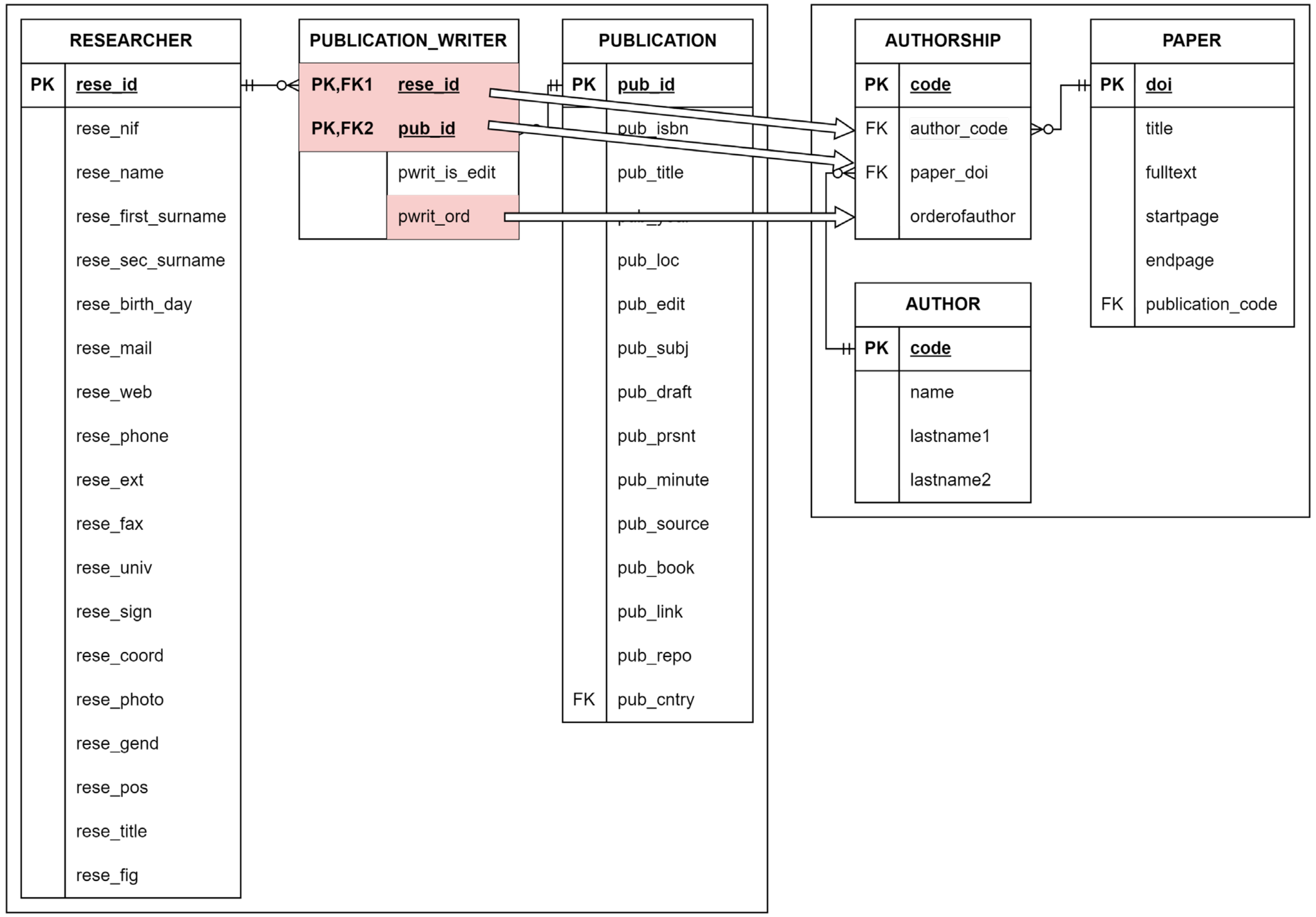
PURE SQL STATEMENT

```
MAP publication TO publisher (
    SQL: "distinct pub_edit" TO name
);
```

2. SQL CODE GENERATED:

```
INSERT INTO target.publisher(name)
    SELECT distinct pub_edit
FROM legacy.publication;
```

MAPPING INTERMEDIATE TABLE



1. DEFINE MAPPING IN DSL:

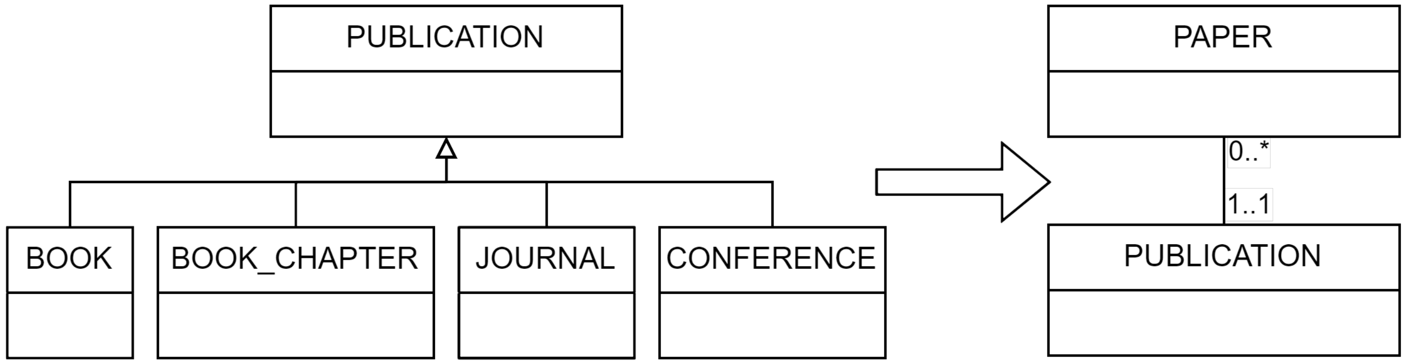
```
MAP publication_writer TO authorship (
    pub_id TO paper_doi,
    rese_id TO author_code,
    pwrit_ord TO orderofauthor
);
```

2. SQL CODE GENERATED:

```
INSERT INTO target.authorship (paper_doi,
    author_code, orderofauthor)
    SELECT pub_id, rese_id, pwrit_ord
FROM legacy.publication_writer;
```

UNIVERSIDADE DA CORUÑA

MAPPING HIERARCHY INTO SINGLE TABLE

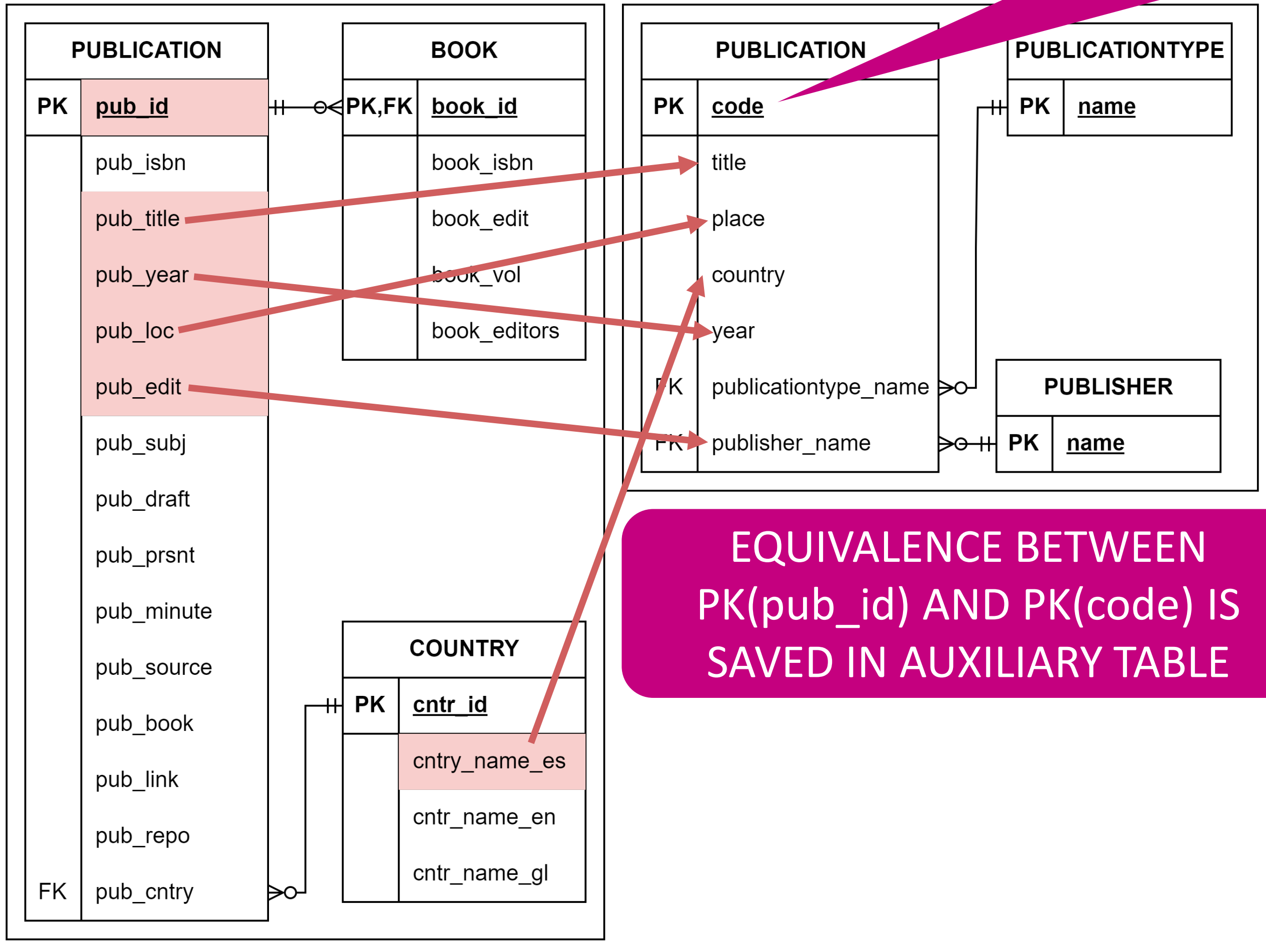


The legacy schema required creating a new tuple for each publication (paper), even if it was in a conference or journal where we had already been published before, leading to duplicated information.

We want to split this inheritance to have papers in one table and publications (journals, conferences, books) in another.

MAP 2 TABLES AND STORE PK REFERENCE

PK(pub_id) != PK(code)
PK(code) = new generated value



EQUIVALENCE BETWEEN PK(pub_id) AND PK(code) IS SAVED IN AUXILIARY TABLE

1. DEFINE MAPPING IN DSL:

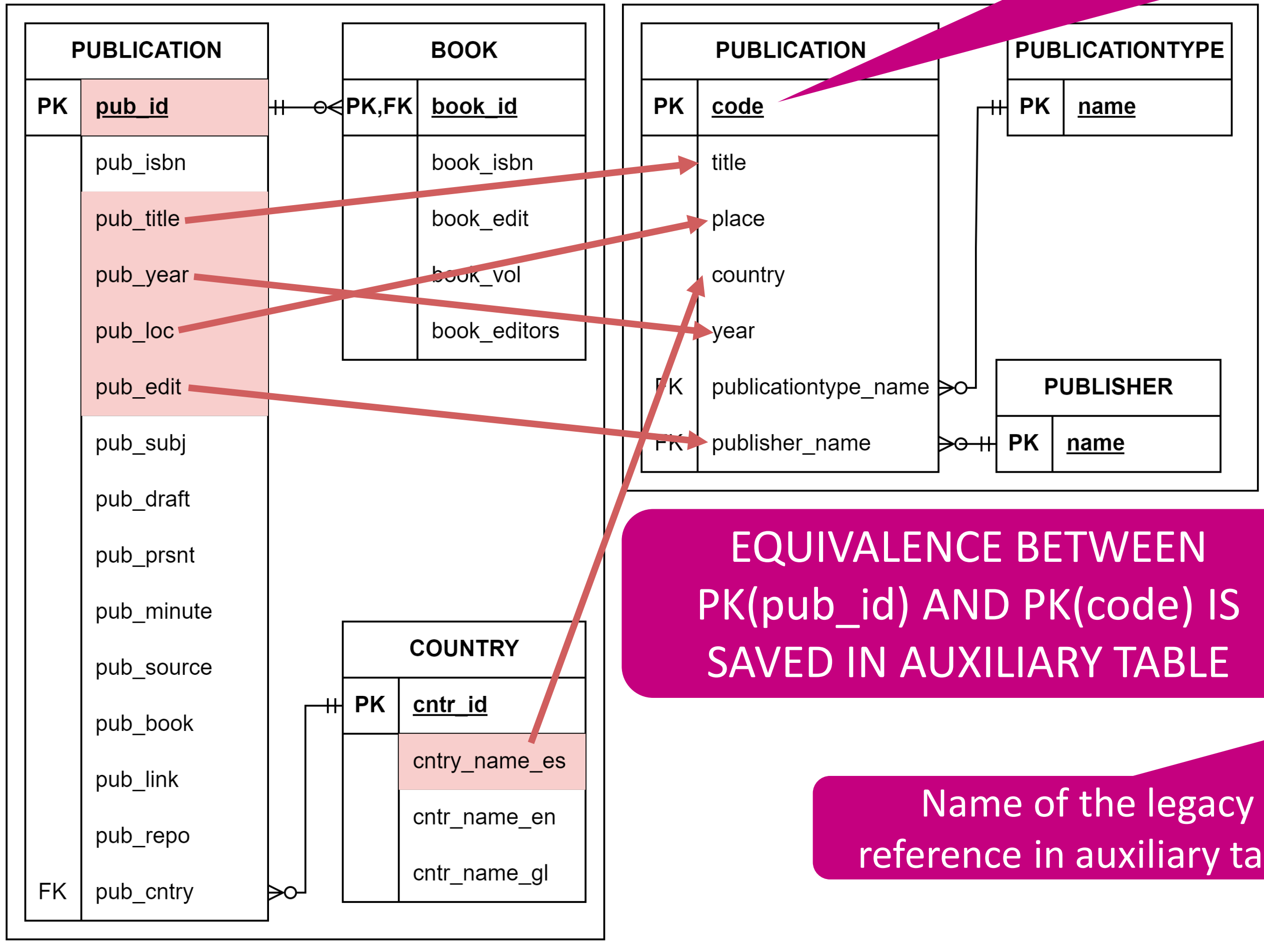
```
MAP publication, country TO publication (
    pub_title TO title,
    pub_loc TO place,
    pub_year TO year,
    'LIBRO' TO publicationtype_name,
    pub_edit TO publisher_name,
    cntry_name_es TO country,
    SAVE RELATION publication.pub_id AS
id_publication int EQUALS publication.code int
) WHERE (pub_cntry=cntry_id);
```

MAP LITERAL TO ATTRIBUTE

JOIN CONDITION FOR PUBLICATION AND COUNTRY

MAP 2 TABLES AND STORE PK REFERENCE

PK(pub_id) != PK(code)
PK(code) = new generated value



1. DEFINE MAPPING IN DSL:

```
MAP publication, country TO publication (
    pub_title TO title,
    pub_loc TO place,
    pub_year TO year,
    'LIBRO' TO publicationtype_name,
    pub_edit TO publisher_name,
    cntry_name_es TO country,
    SAVE RELATION publication.pub_id AS
id_publication int EQUALS publication.code int
) WHERE (pub_cntry=cntry_id);
```

EQUIVALENCE BETWEEN PK(pub_id) AND PK(code) IS SAVED IN AUXILIARY TABLE

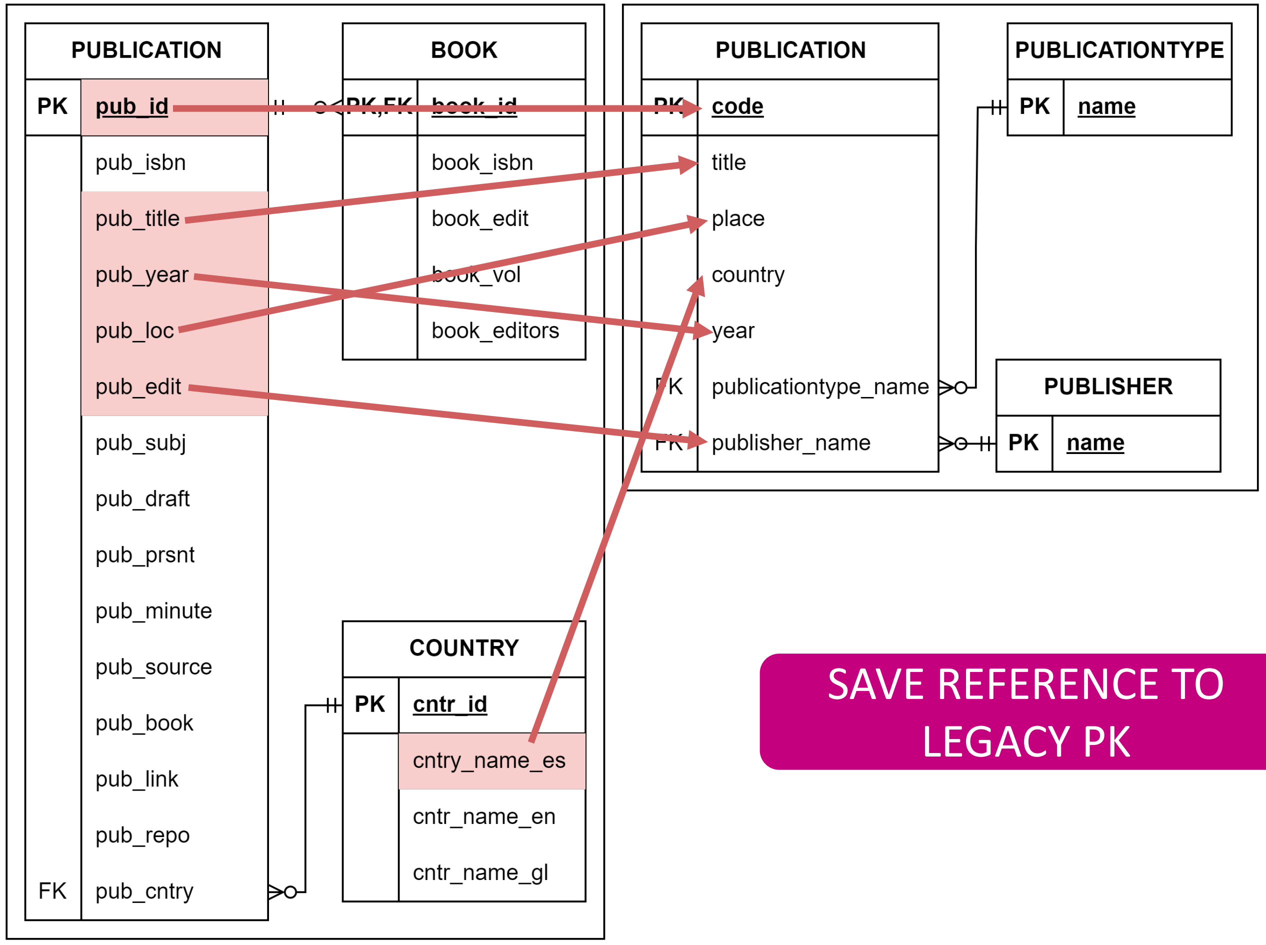
Name of the auxiliary table

Legacy PK to save

New PK to save

Name of the legacy reference in auxiliary table

MAP 2 TABLES AND STORE PK REFERENCE



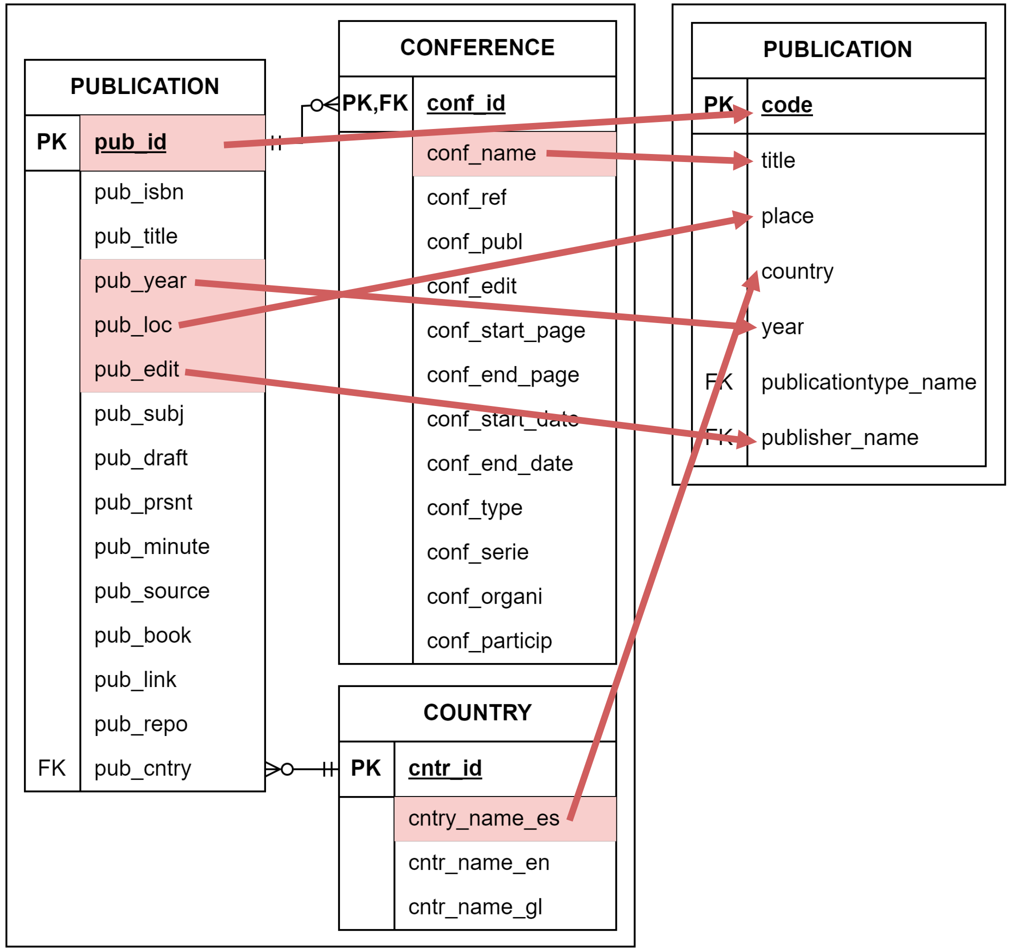
1. SQL CODE GENERATED:

```
ALTER TABLE target.publication ADD id_publication int;
CREATE TABLE IF NOT EXISTS aux.publication(
    publication_code int,
    id_publication int
);
INSERT INTO target.publication(title, place, year,
   publicationtype_name, publisher_name, country,
   id_publication)
    SELECT pub_title, pub_loc, pub_year, 'LIBRO’,
   pub_edit, cntry_name_es, pub_id
FROM legacy.publication, legacy.book, legacy.country
   WHERE pub_id=book_id AND pub_cntry=cntry_id;
INSERT INTO aux.publication(publication_code,
   id_publication)
    SELECT code, id_publication
FROM target.publication;
ALTER TABLE public.publication DROP id_publication;
```

CREATE AUXILIAR TABLE FOR PUBLICATION

SAVE REFERENCE TO LEGACY PK

MAP 3 TABLES AND STORE PK REFERENCE

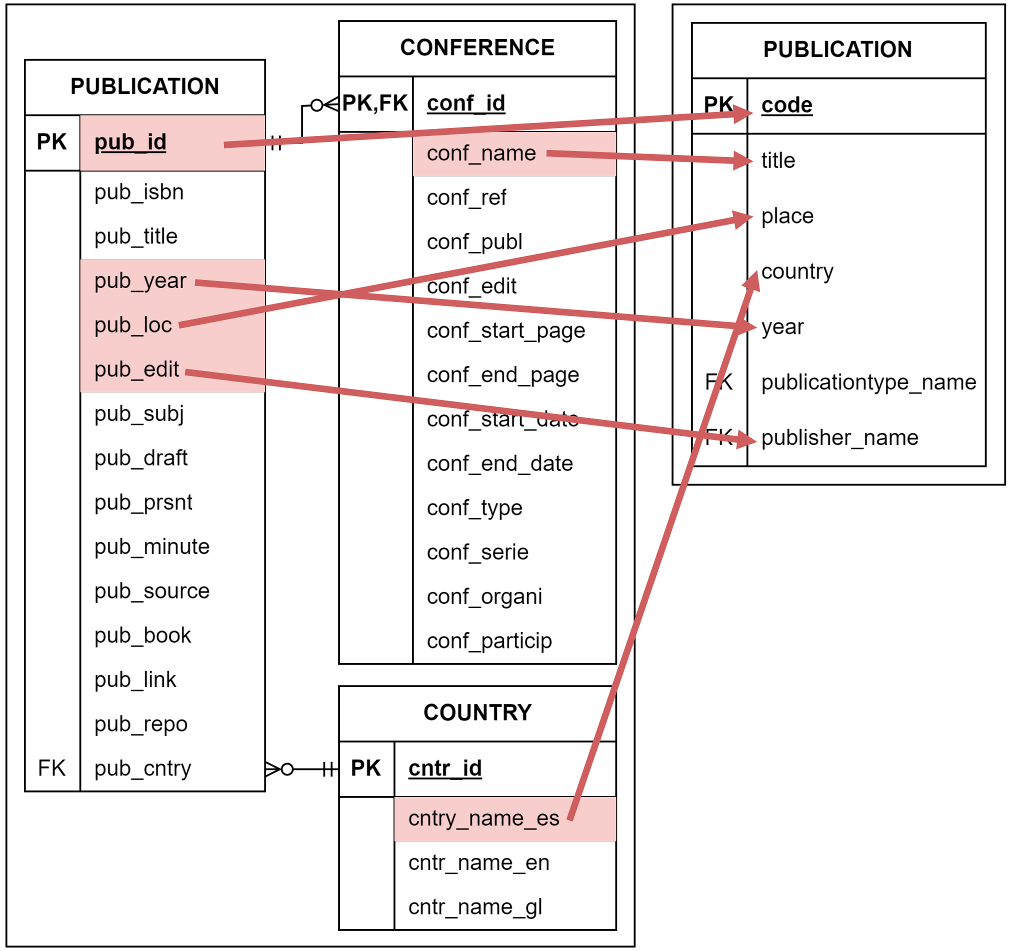


1. DEFINE MAPPING IN DSL:

```
MAP publication, conference, country TO publication
   (conf_name TO title,
    pub_loc TO place,
    pub_year TO year,
    'CONGRESO' TO publicationtype_name,
    pub_edit TO publisher_name,
    cntry_name_es TO country,
    SAVE RELATION publication.pub_id AS id_publication int
   EQUALS publication.code int
) WHERE (pub_id=conf_id AND pub_cntry=cntry_id);
```

2 JOIN CONDITIONS FOR 3 TABLES

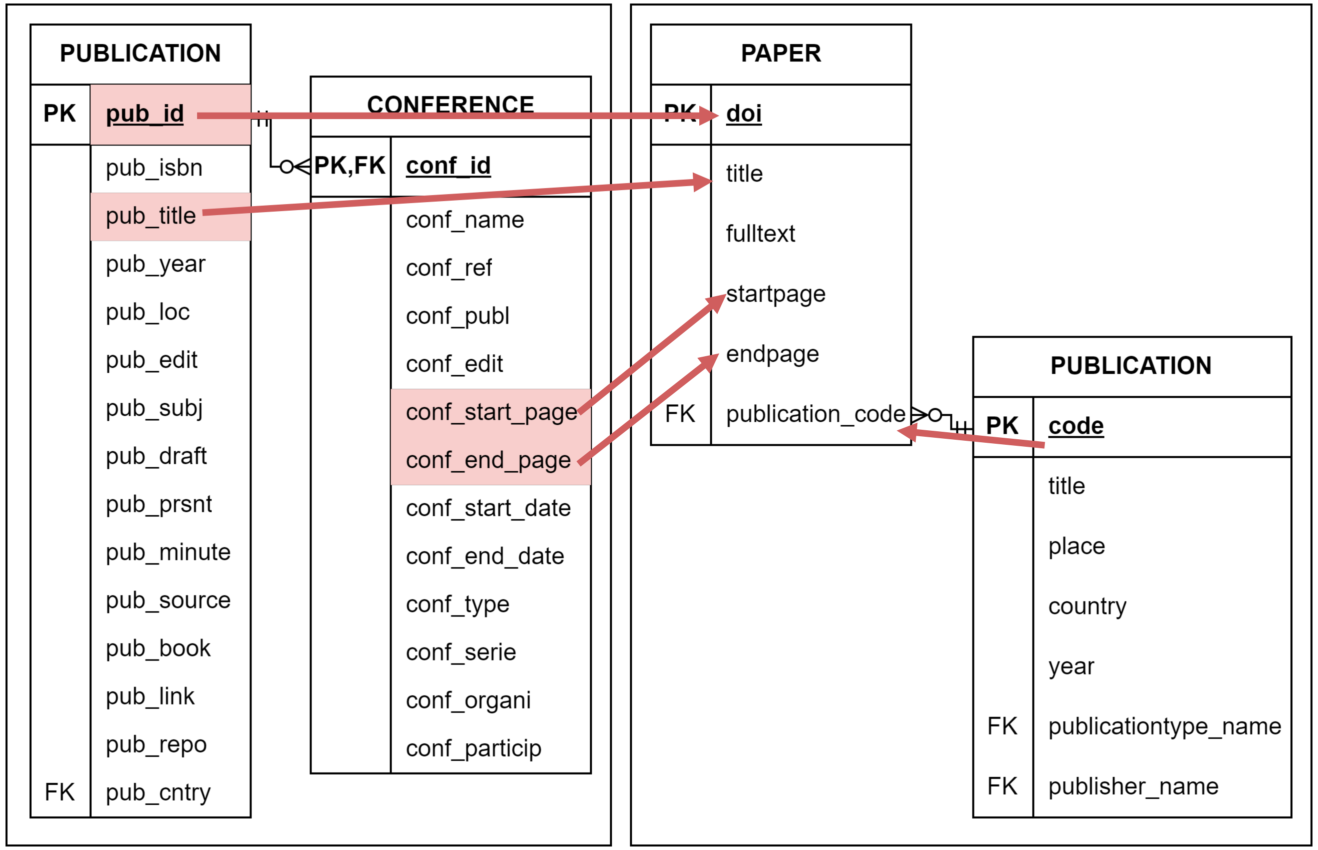
MAP FROM 3 TABLES AND STORE REFERENCE



2. SQL CODE GENERATED:

```
ALTER TABLE target.publication ADD id_publication int;
CREATE TABLE IF NOT EXISTS aux.publication(
    publication_code int,
    id_publication int
);
INSERT INTO target.publication (title, place, year,
publicationtype_name, publisher_name, country, id_publication)
    SELECT conf_name, pub_loc, pub_year, 'CONGRESO’, pub_edit,
cntry_name_es, pub_id
FROM legacy.publication, legacy.conference, legacy.country
WHERE pub_id=jrnl_id AND pub_cntry=cntry_id;
INSERT INTO aux.publication(publication_code, id_publication)
    SELECT code, id_publication
FROM target.publication;
ALTER TABLE public.publication DROP id_publication;
```

MAP AND RETRIEVE STORED REFERENCE

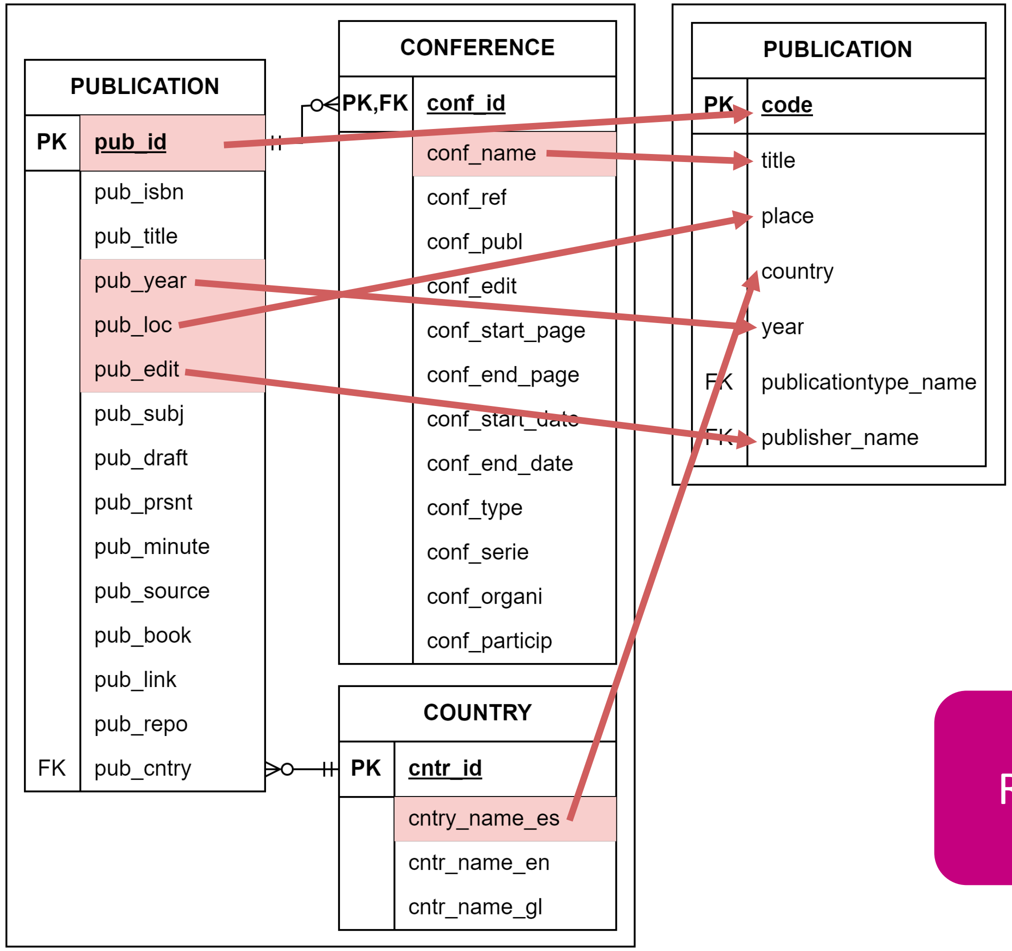


1. DEFINE MAPPING IN DSL:

```
MAP publication, conference TO paper (
    pub_title TO title,
    pub_id TO doi,
    conf_start_page TO startpage,
    conf_end_page TO endpage
) WHERE (pub_id=conf_id)
GET publication_code FROM publication.code WHEN
   id_publication=pub_id;
```

SAME NAME DEFINED IN THE “SAVE RELATION” STATEMENT

MAPPING HIERARCHY INTO SINGLE TABLE



2. SQL CODE GENERATED:

```
INSERT INTO target.paper(title, doi, startpage, endpage)
    SELECT pub_title, pub_id, conf_start_page, conf_end_page
FROM legacy.publication, legacy.conference
WHERE pub_id=jrnl_id;

UPDATE target.paper SET publication_code = (
    SELECT publication_code FROM aux.publication t
      WHERE t.id_publication = conference.pub_id);
```

RETRIEVE SAVED REFERENCE FROM AUXILIAR TABLE

TASKS TO COMPLETE

TASK 1:

Provided the reference migration and the SQL code, complete the SQL definition required to:

Populate the target table “Paper” with all the papers corresponding to the legacy tables “Journal”, “Conference”, “Book”, and “Book Chapter”.

TASK 2:

Provided the reference migration and the DSL code, please complete the DSL definition required to:

Populate the target table “Paper” with all the papers corresponding to the legacy tables “Journal”, “Conference”, “Book”, and “Book Chapter”.

FORM: https://forms.gle/K2MoPxpCZvxVVUox5

LINKS

FORM:

https://forms.gle/K2MoPxpCZvxVVUox5

TASK 1 (DSL):

https://shorturl.at/ndIrq

TASK 2 (SQL):

https://shorturl.at/K2U78