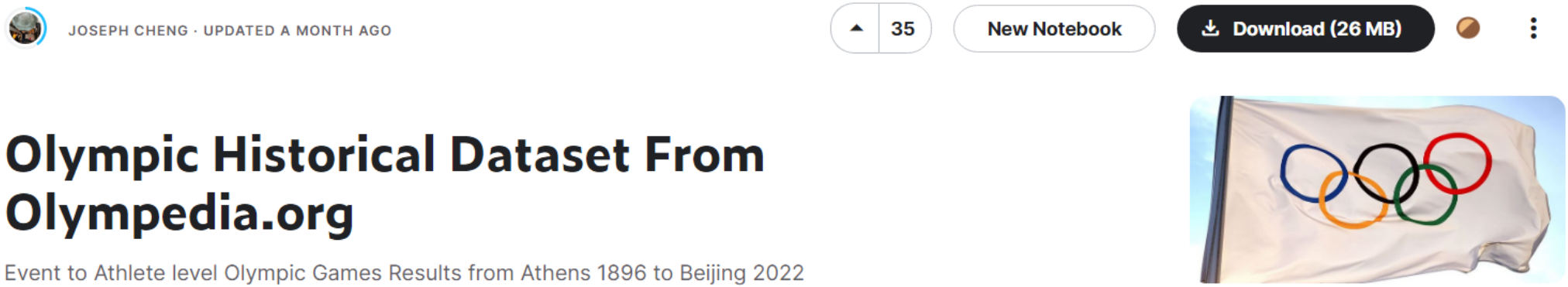
II: OlympicsDB

Dataset



This dataset is as an attempt to create up to date Olympic Event datasets (from 1986 to 2022 Olympics Game) for any sports/data enthusiast to use to visualise and create some insights on the Olympic Event dataset.

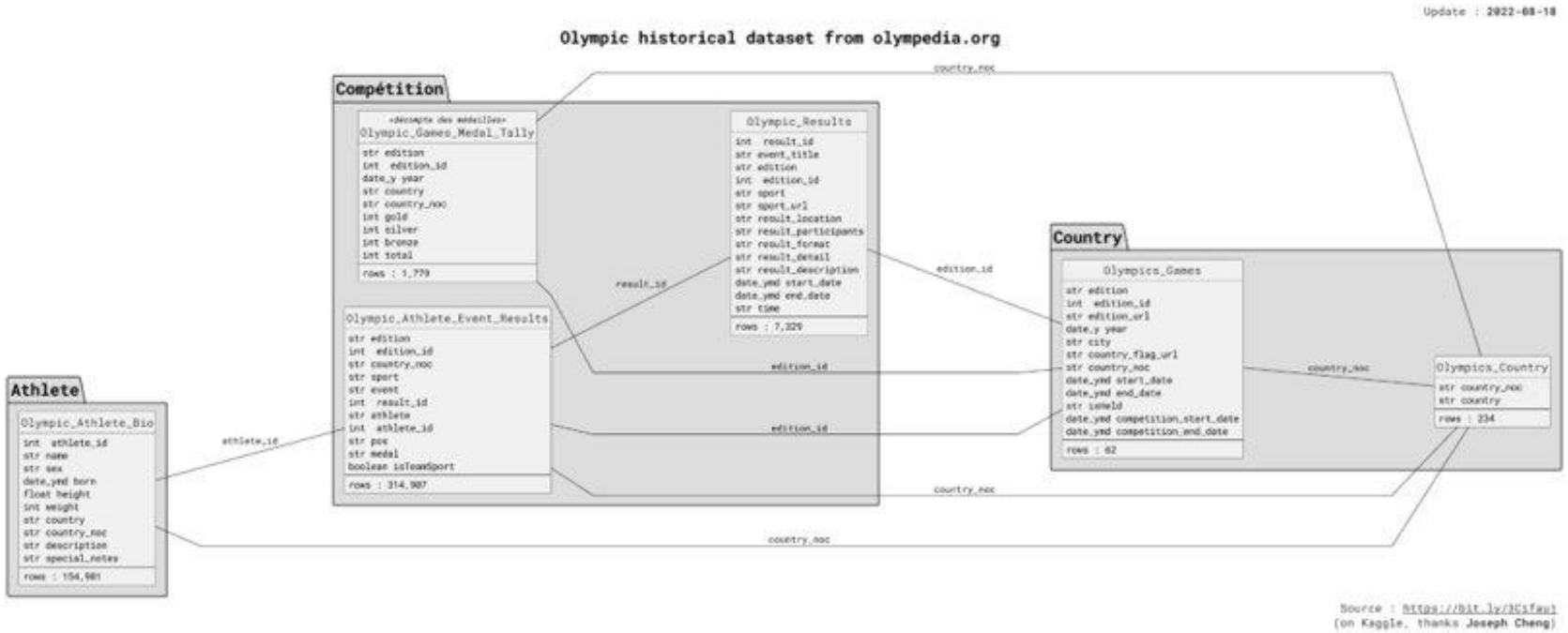
It contains ranking of each sporting event linked to specific country / athlete, which could be used to for any performance related analytics, and information about the athlete's bio, which could be useful in understanding more about the athlete.

The data is scrapped from www.olympedia.org which has the latest up to date olympic data set. Web Scrapping Project is provided via the source code in github using Python's BeautifulSoup.

Dataset

This dataset contains:

- 154,902 unique athletes and their biological information i.e. height, weight, date of birth
- All Winter / Summer Olympic games from 1896 to 2022
- 7326 unique results (result for a specific event played at an Olympic game)
- 314,726 rows of athlete to result data which includes both team sports and individual sports
- 235 distinct countries (some existing from the past)



Dataset Schema

Relation names and (named) attributes

ATHLETE_BIO(athlete_id, name, sex, born, height, weight, country, country_noc, description, notes)

ATHLETE_EVENT_RESULTS(edition, edition_id, country, sport, event, result_id, athlete, athlete_id, pos, medal, isTeamSport)

GAMES_MEDAL_TALLY(edition, edition_id, year, country, country_noc, gold, silver, bronze, total)

RESULTS(result_id, event, edition, edition_id, sport, sport_url, location, participants, format, detail, description, start_date, end_date, time)

COUNTRY(country_noc, country)

GAMES(edition, edition_id, edition_url, season, year, city, flag_url, country_noc, start_date, end_date, isHeld, competition_start_date, competition_end_date)

Relational alphabet

A_S = { $ATHLETE\_BIO_{/10}$, $ATHLETE\_EVENT\_RESULTS_{/11}$, $GAMES\_MEDAL\_TALLY_{/9}$, $RESULTS_{/14}$, $COUNTRY_{/2}$, $GAMES_{/13}$ }

Instance (example)

D(ATHLETE_BIO) = { (1, Jean-François Blanchy, Male, 1886-12-12, na, na, France, FRA, …, na), ... }

Queries to perform

Get the medals won by the athletes who competed in a given sport (e.g., swimming)

Get the number of medals won by a country in the edition it hosted

Get the italian athletes who competed in a given edition (e.g., in 2008)

Get the athletes who competed in an edition hosted by their home countries

Get the athletes who won all the medals of their home country in one edition

Get the countries that have not won any gold medals in a given season (e.g., winter)

Get the age of each fencer in relation to the competitions in which he or she took part

Get the women who competed in more than one edition

Classification

Scope of the integration

Domain-Based Information Integration System (general purpose system)

Mapping specification

Declarative mapping specification (using First-Order Logic)
Sound-mapping semantics
Global-As-View mapping, plus some considerations about (G)LAV mapping

Tools for the implementation

Talend Open Studio, plus PostgreSQL and ad-hoc Python code for pre-processing

Data Replication and Timeliness

Materialized representation of the integration (lazy loading), plus some considerations about virtualization

Preprocessing

Phase 1 (Python)

- Cleaning data
- Deleting useless columns
- Refactoring attributes and domains
- Managing "null" values
- ...

Phase 2 (Talend and PostgreSQL)

- Creating the relation ATHLETES_MEDAL_TALLY$_{/7}$ from ATHLETE_EVENT_RESULTS
  This relation is created following GAMES_MEDAL_TALLY
  It fulfills the need for a comprehensive view of how many madals each athlete has won
- Creating the relation ATHLETES_AGE$_{/3}$ from ATHLETE_EVENT_RESULTS, ATHLETE_BIO and RESULTS
  This relation gives us the age of the athletes with respect to each competition they have taken part in
  It can be useful for gathering other information (such as BMI)
- Setting up the postgres database

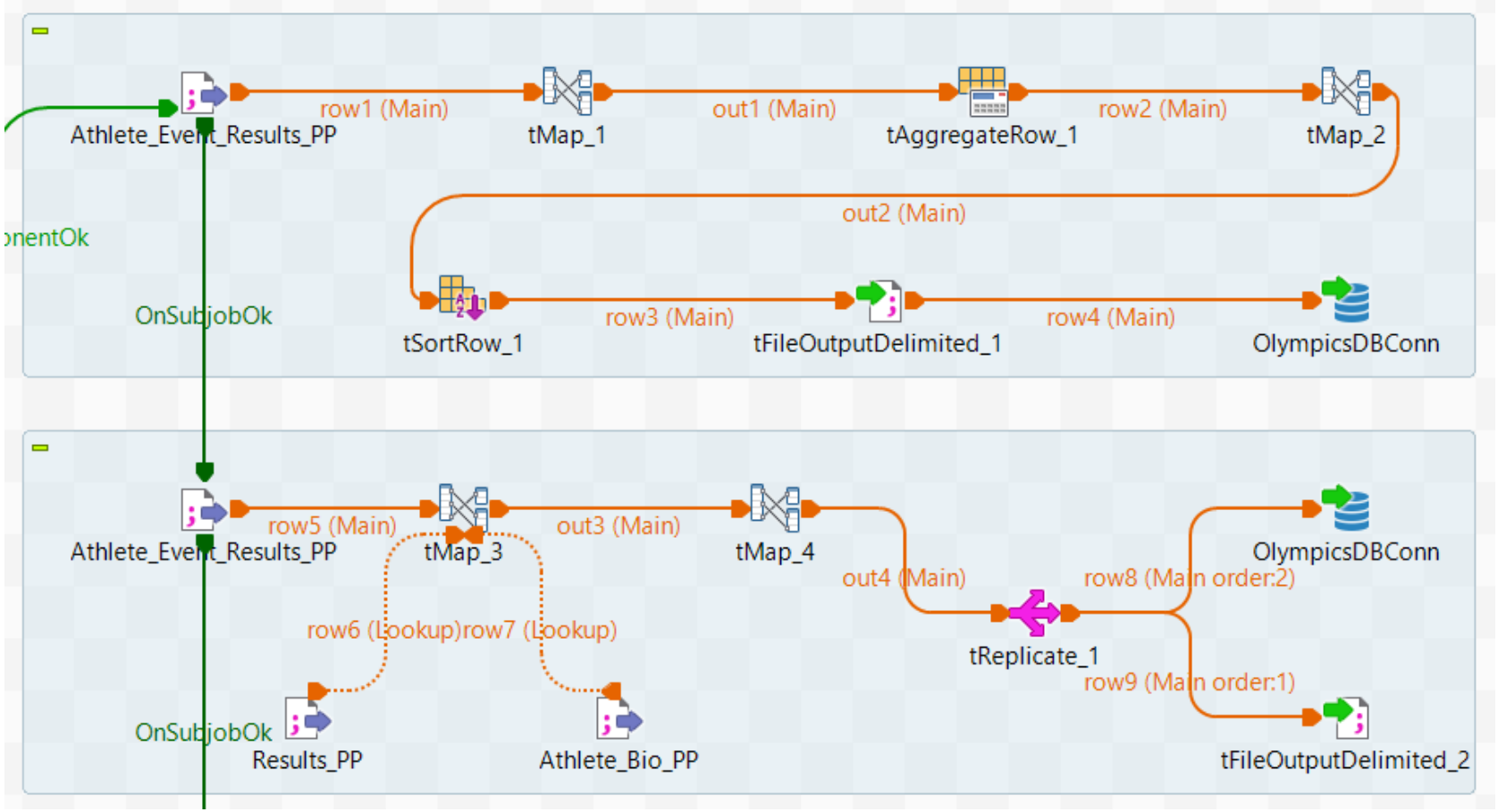
Preprocessing

Phase 1 (example)

```
with open('/content/drive/MyDrive/Games.csv')
      as csv_file:
  csv_reader = csv.reader(csv_file, delimiter=',')
  line_count = 0

  for row in csv_reader:
   line_count += 1

   del row[2]
   del row[5]
   del row[6]

   if line_count != 1:
    if row[6] == 'na':
     row[6] = True
    else:
     row[6] = False
   if line_count == 1:
    row[-2] = row[-2][12:]
    row[-1] = row[-1][12:]

   for i in range(9):
    if row[i] == 'na':
     row[i] = None

   rows.append(row)

with open('/content/drive/MyDrive/Games_PP.csv',
      mode='w') as csv_file:
  writer = csv.writer(csv_file, delimiter=';')

  for row in rows:
   writer.writerow(row)
```

Preprocessing

Phase 2



Source Schema

Relation names and (named) attributes

ATHLETE_BIO(athlete_id, name, sex, born, height, weight, country, country_noc)

ATHLETES_AGE(result_id, athlete_id, age)

ATHLETE_EVENT_RESULTS(edition, edition_id, country, sport, event, result_id, athlete, athlete_id, pos, medal)

ATHLETES_MEDAL_TALLY(edition_id, athlete_id, bronze, silver, gold, total)

GAMES_MEDAL_TALLY(edition, edition_id, year, country, country_noc, gold, silver, bronze, total)

COUNTRY(country_noc, country)

GAMES(edition, edition_id, season, year, city, country_noc, isHeld, start_date, end_date)

RESULTS(result_id, event, edition, edition_id, sport, location, participants, start_date, end_date, time)

Relational alphabet

A_S = { $ATHLETE\_BIO_{/8}$, $ATHLETES\_AGE_{/3}$, $ATHLETE\_EVENT\_RESULTS_{/10}$, $ATHLETES\_MEDAL\_TALLY_{/6}$, $GAMES\_MEDAL\_TALLY_{/9}$, $COUNTRY_{/2}$, $GAMES_{/9}$, $RESULTS_{/10}$ }

Instance (example)

D(ATHLETE_BIO) = { (1, Jean-François Blanchy, Male, 1886-12-12, NULL, NULL, France, FRA), ... }

Global Schema

Relation names and (named) attributes

ATHLETES_HISTORY(edition, sport, event, athlete, sex, age, medal)

GAMES_HISTORY(edition, season, hosting_country, participating_country, gold, silver, bronze, total)

COMPETITIONS_HISTORY(edition, year, hosting_country, event, athlete, ath_country, medal)

MEDALS_HISTORY(edition, athlete, ath_medals, country, c_medals)

Relational alphabet

A_G = { $ATHLETES\_HISTORY_{/7}$, $GAMES\_HISTORY_{/8}$, $COMPETITIONS\_HISTORY_{/7}$, $MEDALS\_HISTORY_{/5}$ }

Information Integration System

J = < G, M, S >, with S = source schema (logical theory over A_S) and G = global schema (logical theory over A_G)

$sem^D(J) = \{C \mid C \vDash G \text{ and } q_S^D \subseteq q_G^C \text{ for each } \langle q_S^D, q_G^C \rangle \in M\}$

$cert(q, J, D) = \{\bar{a} \mid \bar{a} \in q(B) \text{ for each } B \in sem^D(J)\}$

Mapping

Global-As-View mapping assertions

m_1 : ⟨{(ed, sp, ev, a, sex, age, med) | ∃ed_id. ∃c. ∃r_id. ∃a_id. ∃p. ∃n. ∃b. ∃h. ∃w. ∃co. ∃noc.
ATHLETE_EVENT_RESULTS(ed, ed_id, c, sp, ev, r_id, a, a_id, p, med) ∧
ATHLETE_BIO(a_id, n, sex, b, h, w, co, noc) ∧ ATHLETES_AGE(r_id, a_id, age)},
{(ed, sp, ev, a, sex, age, med) | ATHLETES_HISTORY(ed, sp, ev, a, sex, age, med)}⟩

m_1 : ∀ed, sp, ev, a, sex, age, med. ∃ed_id, c, r_id, a_id, p, n, b, h, w, co, noc.
ATHLETE_EVENT_RESULTS(ed, ed_id, c, sp, ev, r_id, a, a_id, p, med) ∧
ATHLETE_BIO(a_id, n, sex, b, h, w, co, noc) ∧ ATHLETES_AGE(r_id, a_id, age) →
ATHLETES_HISTORY(ed, sp, ev, a, sex, age, med)

m_2 : ∀ed, sn, c, co, g, s, b, t. ∃ed_id, y, cn, e, yr, ct, noc, ih, st, end.
GAMES_MEDAL_TALLY(ed, ed_id, y, co, cn, g, s, b, t) ∧
GAMES(e, ed_id, sn, yr, ct, noc, ih, st, end) ∧ COUNTRY(noc, c) →
GAMES_HISTORY(ed, s, c, co, g, s, b, t)

Mapping

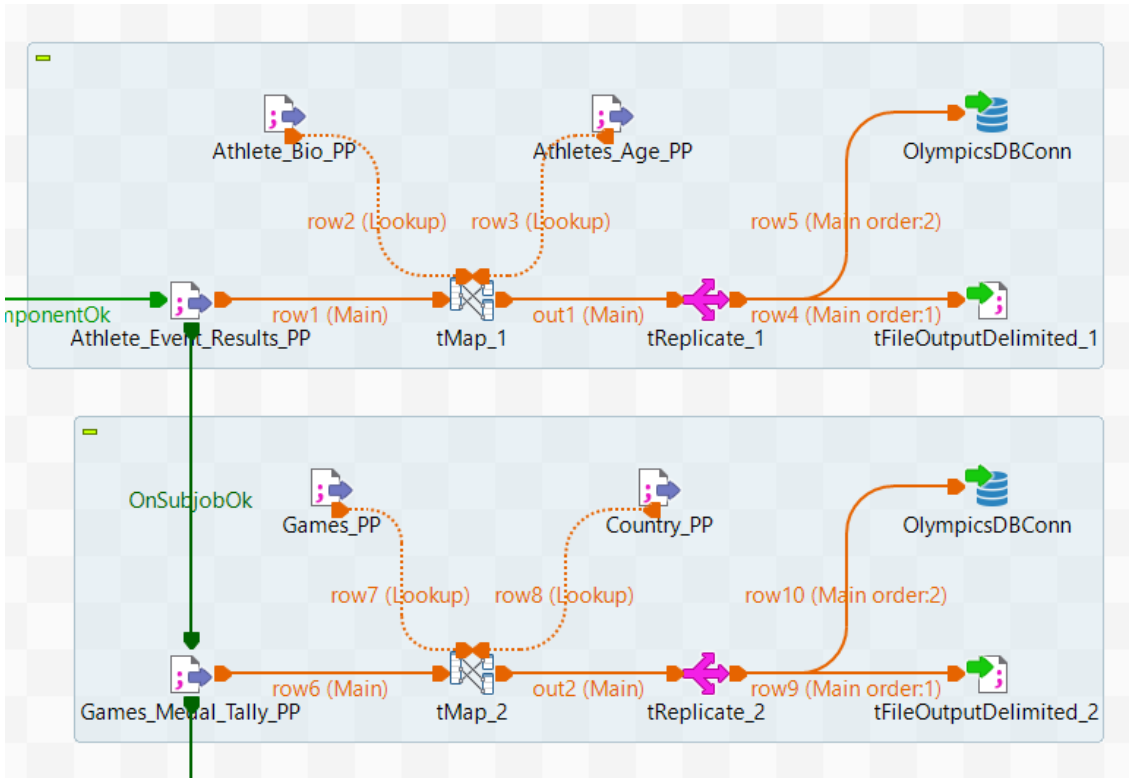
Global-As-View mapping assertions

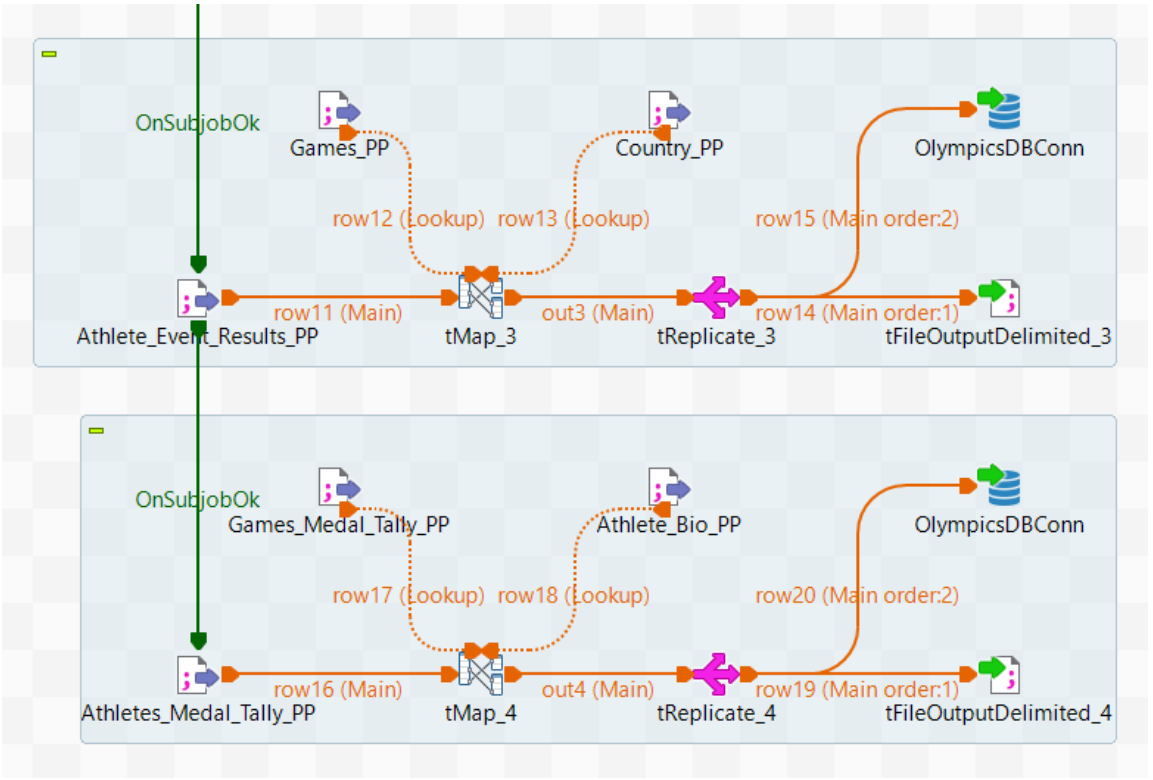
m_3 : $\forall ed, y, co, ev, a, c, med.\ \exists ed\_id, sp, r\_id, a\_id, p, e, s, ct, noc, ih, st, end.$
ATHLETE_EVENT_RESULTS(ed, ed_id, c, sp, ev, r_id, a, a_id, p, med) $\wedge$
GAMES(e, ed_id, s, y, ct, noc, ih, st, end) $\wedge$ COUNTRY(noc, co) $\rightarrow$
COMPETITIONS_HISTORY(ed, y, co, ev, a, c, med)

m_4 : $\forall ed, n, at, co, ct.\ \exists ed\_id, a\_id, ag, as, ab, x, b, h, w, c, noc, y, cg, cs, cb.$
ATHLETES_MEDAL_TALLY(ed_id, a_id, ag, as, ab, at) $\wedge$
ATHLETE_BIO(a_id, n, x, b, h, w, c, noc) $\wedge$
GAMES_MEDAL_TALLY(ed, ed_id, y, co, noc, cg, cs, cb, ct) $\rightarrow$
MEDALS_HISTORY(ed, n, at, co, ct)

Mapping

Mapping in Talend





Queries

Get the medals won by the athletes who competed in a given sport (e.g., swimming)

q_1 : {(ed, ev, a, med) | $\exists$sex. $\exists$age. ATHLETES_HISTORY(ed, Swimming, ev, a, sex, age, med)}

Get the number of medals won by a country in the edition it hosted

q_2 : {(ed, pc, tot) | $\exists$sn. $\exists$hc. $\exists$g. $\exists$s. $\exists$b. GAMES_HISTORY(ed, sn, hc, pc, g, s, b, tot) $\wedge$ (hc = pc)}

Get the italian athletes who competed in a given edition (e.g., in 2008)

q_3 : {(a, ev) | $\exists$ed. $\exists$hc. $\exists$med. COMPETITIONS_HISTORY(ed, 2008, hc, ev, a, Italy, med)}

Get the athletes who competed in an edition hosted by their home countries

q_4 : {(ed, ev, a, ac) | $\exists$y. $\exists$hc. $\exists$med. COMPETITIONS_HISTORY(ed, y, hc, ev, a, ac, med) $\wedge$ (hc = ac)}

Queries

Get the athletes who won all the medals of their home country in one edition

q_5 : {(ed, c, a, cm) | $\exists$am. MEDALS_HISTORY(ed, a, am, c, cm) $\wedge$ (am = cm)}

Get the countries that have not won any gold medals in a given season (e.g., winter)

q_6 : {(ed, pc) | $\exists$hc. $\exists$s. $\exists$b. $\exists$tot. GAMES_HISTORY(ed, Winter, hc, pc, 0, s, b, tot)}

Get the age of each fencer in relation to the competitions in which he or she took part

q_7 : {(ev, a, age, med) | $\exists$ed. $\exists$sex. ATHLETES_HISTORY(ed, Fencing, ev, a, sex, age, med)}

Get the women who competed in more than one edition

q_8 : {(ed, a) | $\exists$sp. $\exists$ev. $\exists$age. $\exists$med. $\exists$ed′. $\exists$sp′. $\exists$ev′. $\exists$age′. $\exists$med′.
ATHLETES_HISTORY(ed, sp, ev, a, Female, age, med)
$\wedge$ ATHLETES_HISTORY(ed′, sp′, ev′, a, Female, age′, med′)
$\wedge \neg$(ed = ed′)}

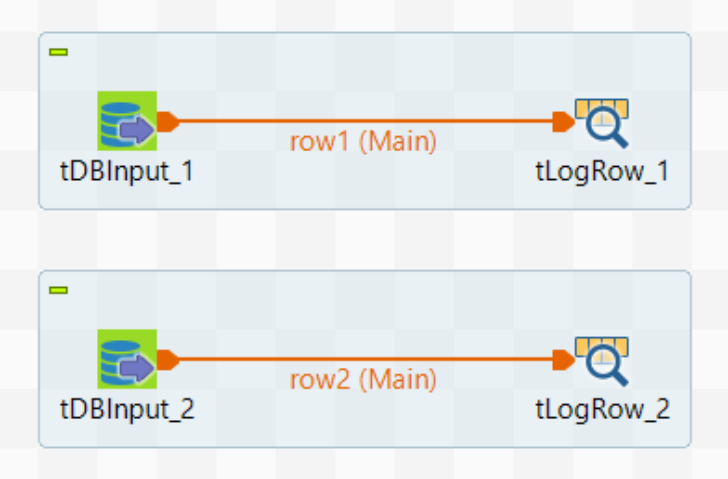
Query Answering

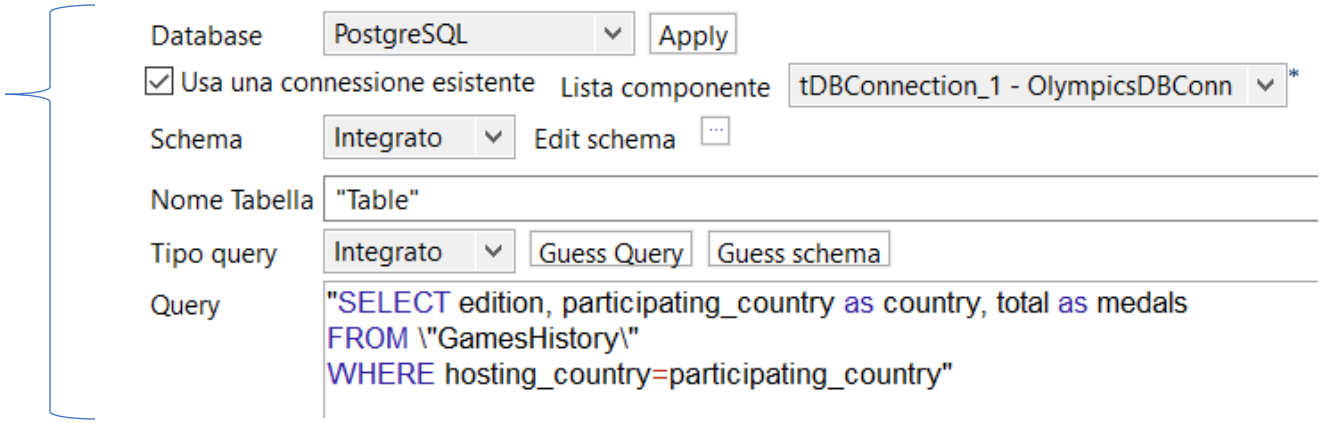
Materialization → Retrieved Global Database (realized using Talend)

$M(D) = \{g(\bar{c}) \mid \langle q_S, g\rangle \in M \; and \; \bar{c} \in q_S(D)\}$

Given a UCQ q, $\bar{c} \in cert(q, J, D)$ if and only if $\bar{c} \in q(M(D))$

Query in Talend





Query Answering

Virtualization → Query Unfolding

Given a query q, a perfect L-rewriting of q w.r.t. J is a query s.t. $\bar{a} \in cert(q, J, D)$ if and only if $\bar{a} \in rew_L^{(q,J)}(D)$

Given the unfolding of a UCQ query q w.r.t. M $\mathrm{unf_M(q)}$, it is a perfect UCQ-rewriting $rew_{UCQ}^{(q,J)}$

Example: get the medals won by the athletes that competed in a given sport (e.g., swimming)

q : {(ed, ev, a, med) | ∃sex. ∃age. ATHLETES_HISTORY(ed, Swimming, ev, a, sex, age, med)}

m_1 : ∀ed, sp, ev, a, sex, age, med. ∃ed_id, c, r_id, a_id, p, n, b, h, w, co, noc.
ATHLETE_EVENT_RESULTS(ed, ed_id, c, sp, ev, r_id, a, a_id, p, med) ∧
ATHLETE_BIO(a_id, n, sex, b, h, w, co, noc) ∧ ATHLETES_AGE(r_id, a_id, age) →
ATHLETES_HISTORY(ed, sp, ev, a, sex, age, med)

q' : {(ed, ev, a, med) | ∃sex. ∃age. ∃ed_id′. ∃c′. ∃r_id′. ∃a_id′. ∃p′. ∃n′. ∃b′. ∃h′. ∃w′. ∃co′. ∃noc′.
ATHLETE_EVENT_RESULTS(ed, ed_id′, c′, Swimming, ev, r_id′, a, a_id′, p′, med) ∧
ATHLETE_BIO(a_id′, n′, sex, b′, h′, w′, co′, noc′) ∧ ATHLETES_AGE(r_id′, a_id′, age)}

Query Answering

GLAV Mapping

m_1 : $\forall ed, sp, ev, a, sex, age, med.\ \exists ed\_id, c, r\_id, a\_id, p, n, b, h, w, co, noc.$
ATHLETE_EVENT_RESULTS(ed, ed_id, c, sp, ev, r_id, a, a_id, p, med) ∧
ATHLETE_BIO(a_id, n, sex, b, h, w, co, noc) ∧ ATHLETES_AGE(r_id, a_id, age) →
$\exists bmi.$ ATHLETES_HISTORY(ed, sp, ev, a, sex, age, bmi, med)

Materialization → Naïve Chase

Given a query q and a universal solution $U_{(J,D)}$, $\bar{a} \in cert(q, J, D)$ if and only if $\bar{a} \in cert(q, U_{(J,D)})$

The naïve chase algorithm computes a universal solution $U_{(J,D)}$

Example

D = { ATHLETE_EVENT_RESULTS(1908 Summer Olympics, 5, Australasia, Athletics, 3500 m Race Walk, 56421, Harry Kerr, 64719, 3, Bronze), ATHLETE_BIO(64719, Harry Kerr, Male, 1879-01-28, 184, 76, Australasia, ANZ), ATHLETES_AGE(56421, 64719, 29), …}

C = { ATHLETES_HISTORY(1908 Summer Olympics, Athletics, 3500 m Race Walk, Harry Kerr, Male, 29, x, Bronze), …}

Query Answering

GLAV Mapping

m_1 : $\forall ed, sp, ev, a, sex, age, med.\ \exists ed\_id, c, r\_id, a\_id, p, n, b, h, w, co, noc.$
ATHLETE_EVENT_RESULTS(ed, ed_id, c, sp, ev, r_id, a, a_id, p, med) ∧
ATHLETE_BIO(a_id, n, sex, b, h, w, co, noc) ∧ ATHLETES_AGE(r_id, a_id, age) →
$\exists bmi.$ ATHLETES_HISTORY(ed, sp, ev, a, sex, age, bmi, med)

Virtualization → Perfect Rewriting + Query Unfolding

If $\langle q_S, q_G \rangle$ is a GLAV mapping assertion and p a new predicate, $\langle q_S, q_G \rangle$ is equivalent to $\langle q_S, p \rangle \cup \langle p, q_G \rangle$

To rewrite a query q, we use the PR algorithm w.r.t. LAV assertions and we unfold the result w.r.t. GAV assertions

Example

m_1^G : $\forall ed, sp, ev, a, sex, age, med.\ \exists ed\_id, c, r\_id, a\_id, p, n, b, h, w, co, noc.$
ATHLETE_EVENT_RESULTS(ed, ed_id, c, sp, ev, r_id, a, a_id, p, med) ∧
ATHLETE_BIO(a_id, n, sex, b, h, w, co, noc) ∧ ATHLETES_AGE(r_id, a_id, age) →
TEMP(ed, sp, ev, a, sex, age, med)

m_1^L : $\forall ed, sp, ev, a, sex, age, med.$ TEMP(ed, sp, ev, a, sex, age, med) →
$\exists bmi.$ ATHLETES_HISTORY(ed, sp, ev, a, sex, age, bmi, med)

Query Answering

Virtualization → Perfect Rewriting

A sound-rewriting of q w.r.t. M is a query q' such that, for every database D, if $\bar{a} \in q'(D)$ then $\bar{a} \in cert(q, J, D)$

A maximally-sound rewriting q' is s.t., for every sound-rewriting q'', the set $q'(D)$ is not contained in the set $q''(D)$

Given $\langle R, q_R \rangle \in M$, the expansion $exp^{(q_S, M)}$ is obtained by replacing each atom $R(\bar{z})$ in q_S with its definition

q_S is a sound rewriting of q_G if and only if $exp^{(q_S, M)}(D) \subseteq q_G(D)$ for every database D

If q_m is the the union of all the maximally-sound CQ rewritings of q w.r.t. J, then q_m is a perfect UCQ-rewriting of q

Given q with m conjuncts, the PR algorithm outputs the union of all sound-rewriting q' with at most m conjuncts

Example

q_1 : {(ed, ev, a, med) | ∃sex. ∃age. ∃bmi. ATHLETES_HISTORY(ed, Swimming, ev, a, sex, age, bmi, med)}

q' : {(ed, ev, a, med) | ∃sex. ∃age. TEMP(ed, Swimming, ev, a, sex, age, med)}

$exp^{(q', m_1^L)}$: {(ed, ev, a, med) | ∃sex. ∃age. ∃bmi′. ATHLETES_HISTORY(ed, Swimming, ev, a, sex, age, bmi′, med)}

Global Schema Constraints

Axiom

$a_1 : \forall a.\, \exists ed, sp, ev, sex, age, med.\ \text{ATHLETES\_HISTORY}(ed, sp, ev, a, sex, age, bmi, med) \rightarrow$
$\exists c.\ \text{COACHES\_HISTORY}(a, c)$

Repairing R.G.D. → Chase Algorithm

Axioms in the global schema let us infer new knowledge

We can materialize the inferred knowledge in some cases (e.g., with acyclic constraints)

Example

D = { ATHLETE_EVENT_RESULTS(1908 Summer Olympics, 5, Australasia, Athletics, 3500 m Race Walk, 56421, Harry Kerr, 64719, 3, Bronze), ATHLETE_BIO(64719, Harry Kerr, Male, 1879-01-28, 184, 76, Australasia, ANZ), ATHLETES_AGE(56421, 64719, 29), …}

M(D) = { ATHLETES_HISTORY(1908 Summer Olympics, Athletics, 3500 m Race Walk, Harry Kerr, Male, 29, Bronze), …}

$M^A(D)$ = { COACHES_HISTORY(Harry Kerr, c_1), …}