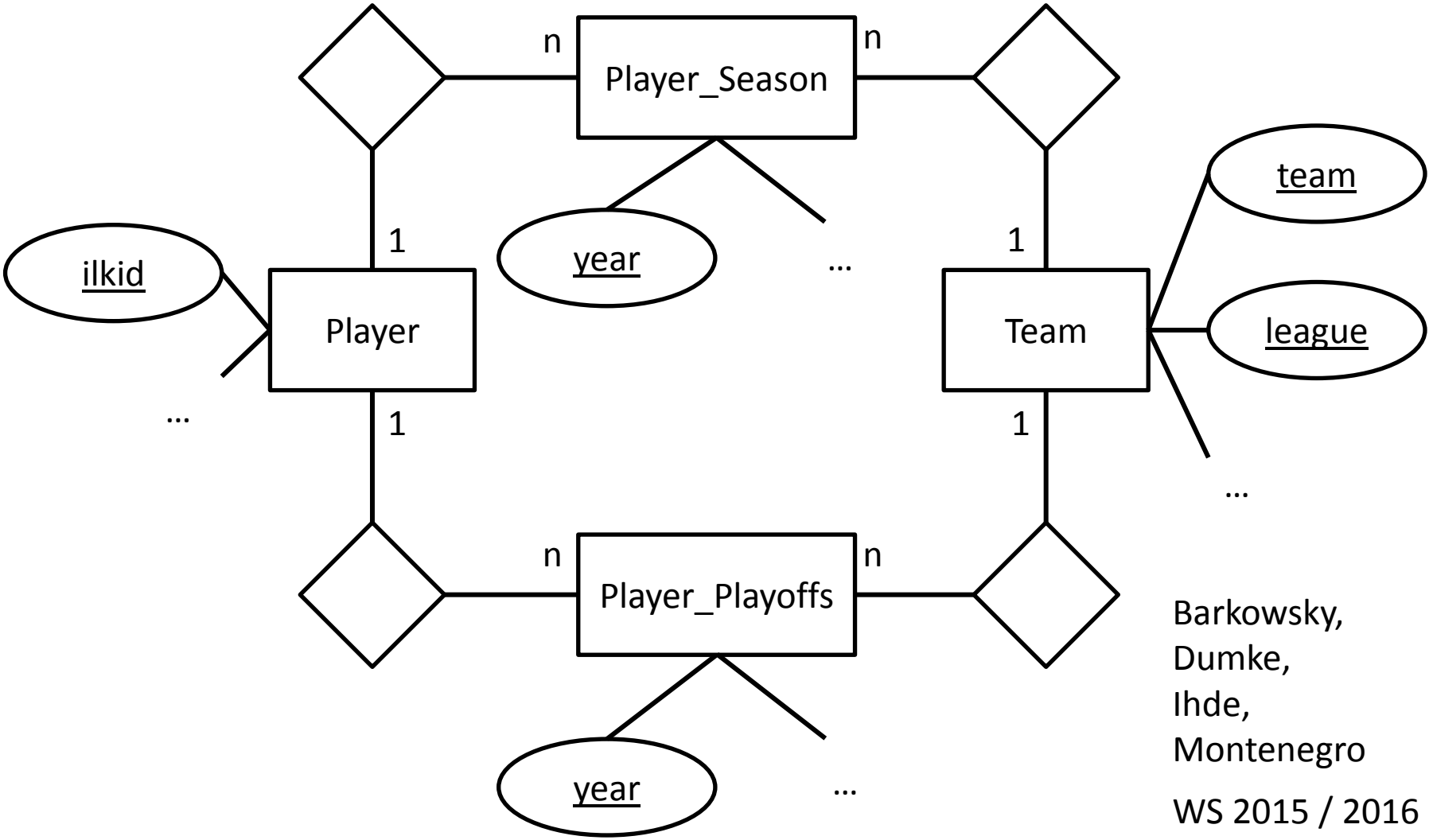
databaseBasketball: Source

- databaseBasketball.com
  - Part of databaseSports.com (contains databases about many different sports)
  - Contains basketball players, teams and statistics per season from the NBA and ABA
  - About 4000 Players from 100 Teams, 30000 player season statistics

databaseBasketball: ER-Diagram



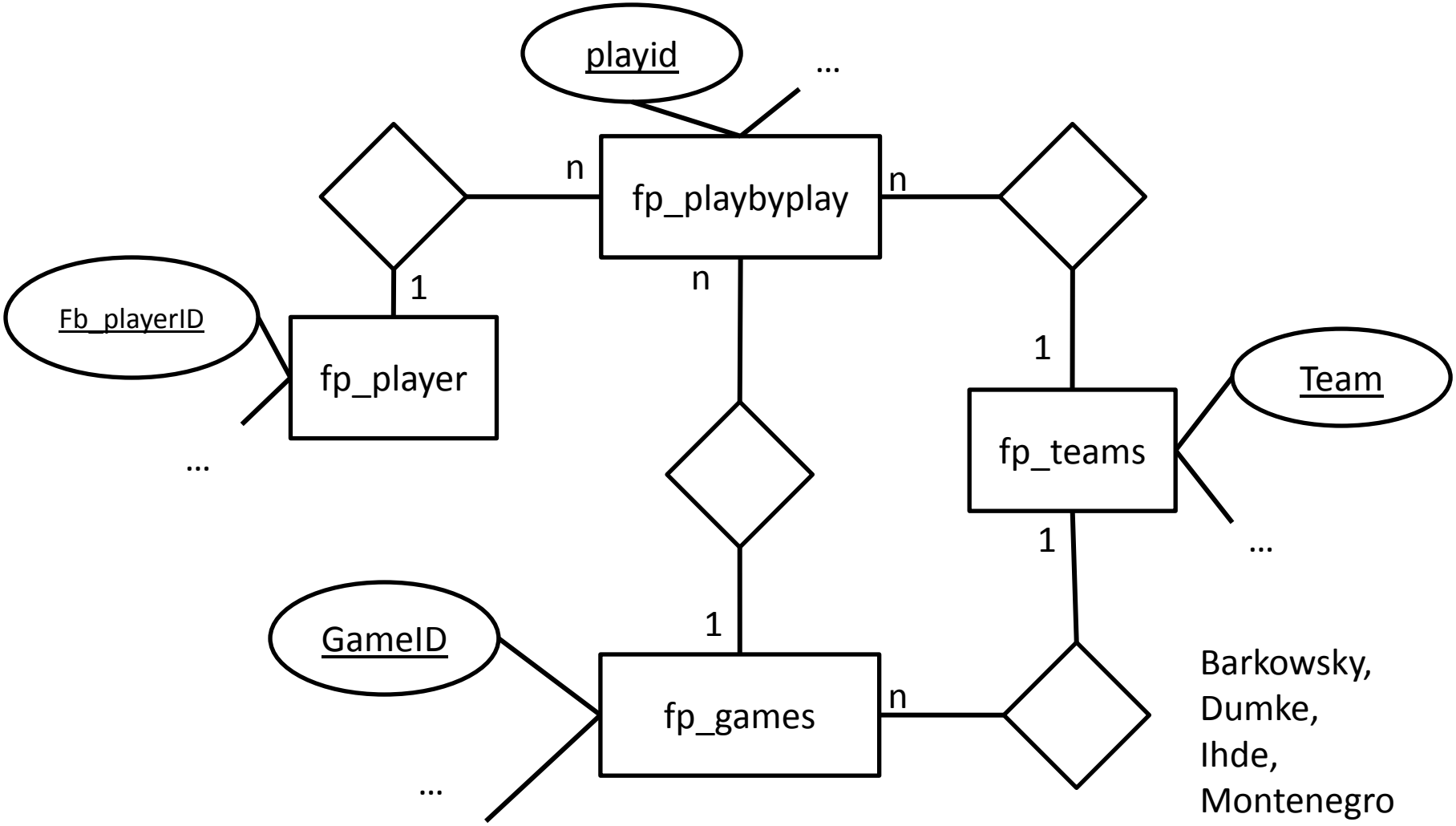
Barkowsky,
Dumke,
Ihde,
Montenegro

WS 2015 / 2016

footballProject: Source

- footballProject.com
  - Goal of the DB is to create play-by-play data for porfessional football games since 1920
  - Contains football players, teams, games and play-by-play information mostly from the NFL
  - About 4000 Players from 32 Teams in 267 games with close to 44000 play-by-play data
  - Tables were imported from .csv files

footballProject: ER-Diagram



Barkowsky,
Dumke,
Ihde,
Montenegro

WS 2015 / 2016

databaseBasketball: Schema

```
CREATE TABLE player
(
ilkid CHAR(9) PRIMARY KEY,
firstname VARCHAR(255),
lastname VARCHAR(255),
position CHAR(1),
firstseason SMALLINT,
lastseason SMALLINT,
h_feet SMALLINT,
h_inches FLOAT,
weight SMALLINT,
college VARCHAR(255),
birthdate DATE
);
```

```
CREATE TABLE team
(
team CHAR(3),
location VARCHAR(255),
name VARCHAR(255),
league CHAR(1),
PRIMARY KEY(team, league)
);
```

databaseBasketball: Schema

```
CREATE TABLE player_regular_season
(
ilkid CHAR(9),
year SMALLINT,
firstname VARCHAR(255),
lastname VARCHAR(255),
team CHAR(3),
league CHAR(1),
gp SMALLINT,
minutes SMALLINT,
pts SMALLINT,
oreb SMALLINT,
dreb SMALLINT,
reb SMALLINT,
asts SMALLINT,
stl SMALLINT,
blk SMALLINT,
turnover SMALLINT,
pf SMALLINT,
fga SMALLINT,
fgm SMALLINT,
fta SMALLINT,
ftm SMALLINT,
tpa SMALLINT,
tpm SMALLINT,
PRIMARY KEY(ilkid, team, league, year),
FOREIGN KEY(ilkid) REFERENCES player(ilkid),
FOREIGN KEY(team, league) REFERENCES team(team, league)
);
```

```
CREATE TABLE player_playoffs
(
ilkid CHAR(9),
year SMALLINT,
firstname VARCHAR(255),
lastname VARCHAR(255),
team CHAR(3),
league CHAR(1),
gp SMALLINT,
minutes SMALLINT,
pts SMALLINT,
dreb SMALLINT,
oreb SMALLINT,
reb SMALLINT,
asts SMALLINT,
stl SMALLINT,
blk SMALLINT,
turnover SMALLINT,
pf SMALLINT,
fga SMALLINT,
fgm SMALLINT,
fta SMALLINT,
ftm SMALLINT,
tpa SMALLINT,
tpm SMALLINT,
PRIMARY KEY(ilkid, team, league, year),
FOREIGN KEY(ilkid) REFERENCES player(ilkid),
FOREIGN KEY(team, league) REFERENCES team(team, league)
);
```

Barkowsky,
Dumke,
Ihde,
Montenegro

WS 2015 / 2016

footballProject: Schema

```
CREATE TABLE fb_player
(
  "fp_playerID" character varying(100) NOT NULL,
  "LName" character varying(50),
  "FName" character varying(50),
  "Position" character varying(50),
  CONSTRAINT fb_player_pkey PRIMARY KEY ("fp_playerID")
);
```

```
CREATE TABLE fp_playbyplay
(
  playid bigint NOT NULL,
  gameid character varying(100),
  period smallint,
  clocktime smallint,
  sequence smallint,
  offense character varying(50),
  defense character varying(50),
  scrimmage character varying(50),
  down smallint,
  distance smallint,
  playtype character varying(50),
  playsubtype character varying(50),
  formation character varying(50),
  playerid character varying(100),
  yds smallint,
  passerid character varying(100),
  playloc character varying(50),
  playresult character varying(50),
  returnteam character varying(50),
  returnid character varying(100),
  returnloc character varying(50),
  returnyds smallint,
  noadvance character varying(50),
  tackleid character varying(100),
  tackleid2 character varying(100),
  tackleid3 character varying(100),
  blockid character varying(100),
  outofbonds character varying(50),
  firstdown character varying(50),
  touchback character varying(50),
  onsidekick character varying(50),
  aborted character varying(50),
  noplay character varying(50),
  scoreteam character varying(50),
  scoretype character varying(50),
  scoreid character varying(50),
  scorenull character varying(50),
  description character varying(255),
  note character varying(255),
  "Empty" character varying(50),
  CONSTRAINT fp_playbyplay_pkey PRIMARY KEY (playid),
  CONSTRAINT fp_playbyplay_blockid_fkey FOREIGN KEY (blockid)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_defense_fkey FOREIGN KEY (defense)
      REFERENCES fp_teams ("Team") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_gameid_fkey FOREIGN KEY (gameid)
      REFERENCES fp_games ("GameID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_offense_fkey FOREIGN KEY (offense)
      REFERENCES fp_teams ("Team") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_passerid_fkey FOREIGN KEY (passerid)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_playerid_fkey FOREIGN KEY (playerid)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_returnid_fkey FOREIGN KEY (returnid)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_returnteam_fkey FOREIGN KEY (returnteam)
      REFERENCES fp_teams ("Team") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_scoreid_fkey FOREIGN KEY (scoreid)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_scoreteam_fkey FOREIGN KEY (scoreteam)
      REFERENCES fp_teams ("Team") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_tackleid2_fkey FOREIGN KEY (tackleid2)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_tackleid3_fkey FOREIGN KEY (tackleid3)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fp_playbyplay_tackleid_fkey FOREIGN KEY (tackleid)
      REFERENCES fb_player ("fp_playerID") MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

Barkowsky,
Dumke,
Ihde,
Montenegro

WS 2015 / 2016

footballProject: Schema

```
CREATE TABLE fp_games
(
  "Season" smallint,
  "Lg" character varying(50),
  "Week" smallint,
  "Post" character varying(50),
  "GameID" character varying(100) NOT NULL,
  "Date" character varying(50) NOT NULL,
  "VTm" character varying(50),
  "VPts" smallint,
  "HTm" character varying(50),
  "HPts" smallint,
  "NeztralSite" character varying(50),
  "Notes" character varying(255),
  "ReleaseDate" character varying(50),
  CONSTRAINT fp_games_pkey PRIMARY KEY ("GameID"),
  CONSTRAINT "fp_games_HTm_fkey" FOREIGN KEY ("HTm")
     REFERENCES fp_teams ("Team") MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "fp_games_VTm_fkey" FOREIGN KEY ("VTm")
     REFERENCES fp_teams ("Team") MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

```
CREATE TABLE fp_teams
(
  "Year" smallint,
  "Fra" character varying(50),
  "Team" character varying(50) NOT NULL,
  "Lg" character varying(50),
  "Conf" character varying(50),
  "Div" character varying(50),
  "TmFull" character varying(100),
  "G" smallint,
  "W" smallint,
  "L" smallint,
  "T" smallint,
  "PW" smallint,
  "PL" smallint,
  CONSTRAINT fp_teams_pkey PRIMARY KEY ("Team")
);
```

Barkowsky,
Dumke,
Ihde,
Montenegro

WS 2015 / 2016

Information Integration

Exercise 1 – Dataset Introduction

Beck, Dücker, Mattfeld, Schneider

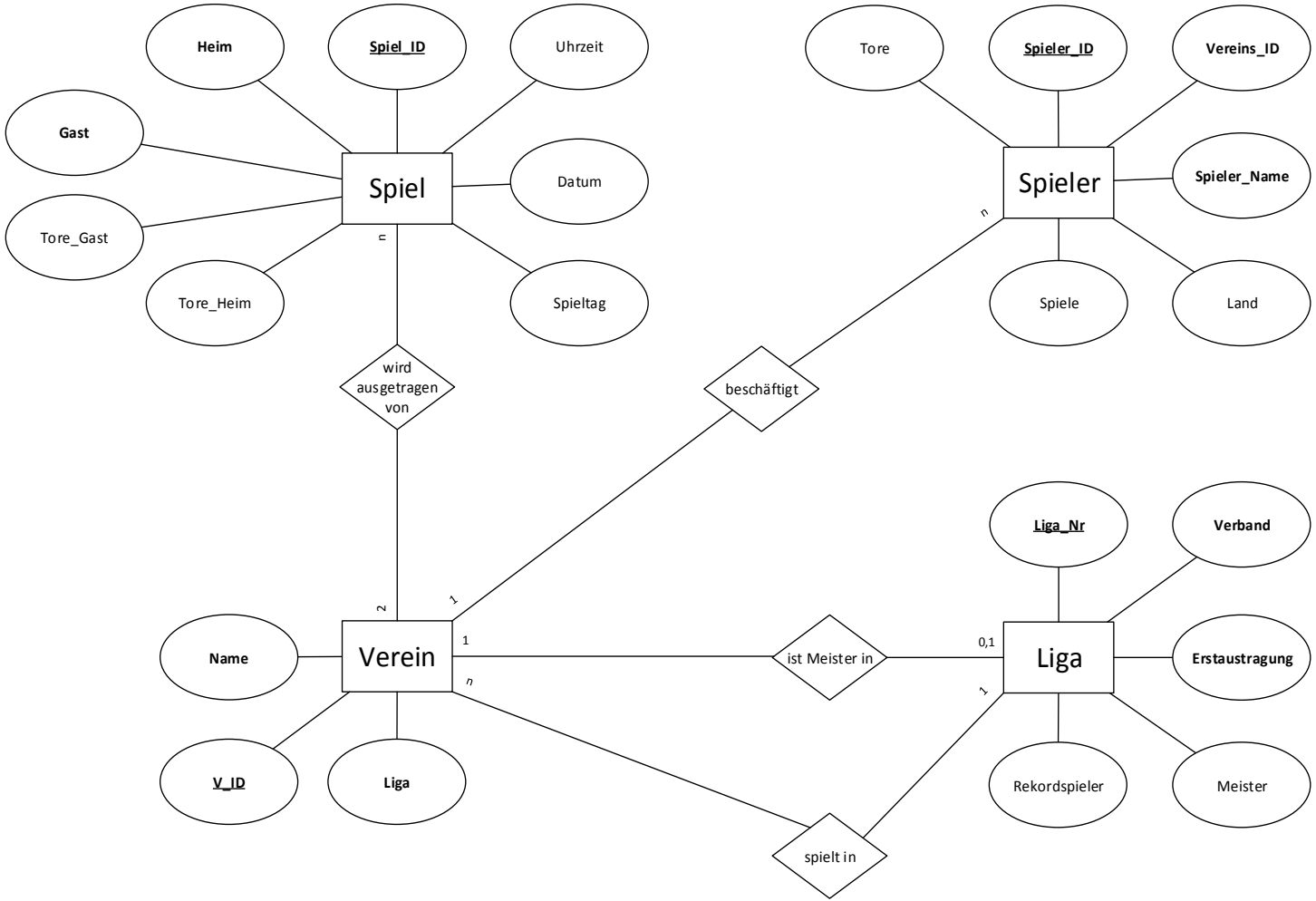
HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Dataset 1 – Bundesliga
Introduction

- 1., 2., 3. Fußball-Bundesliga
- Aktuelle Saison
- 56 Vereine
- 1000 Spiele
- 1.500 Spieler

- Link:
  http://dbup2date.uni-bayreuth.de/bundesliga.html

- MySQL-Download, Skript-Anpassung



Information Integration Exercise 1

Beck, Dücker, Mattfeld, Schneider

Chart **3**

Dataset 1 – Bundesliga
CREATE TABLE – Liga

```
CREATE TABLE "Liga" (
        "Liga_Nr" integer PRIMARY KEY,
        "Verband" varchar(255) NOT NULL,
        "Erstaustragung" date NOT NULL,
        "Meister" integer REFERENCES "Verein"("V_ID"),
        "Rekordspieler" varchar(255) DEFAULT NULL,
        "Spiele_Rekordspieler" integer DEFAULT NULL
) ;
```

Dataset 1 – Bundesliga
CREATE TABLE – Spiel

```
CREATE TABLE "Spiel" (
        "Spiel_ID" SERIAL PRIMARY KEY,
        "Spieltag" integer DEFAULT NULL,
        "Datum" date DEFAULT NULL,
        "Uhrzeit" time DEFAULT NULL,
        "Heim" integer REFERENCES "Verein"("V_ID"),
        "Gast" integer REFERENCES "Verein"("V_ID"),
        "Tore_Heim" integer DEFAULT NULL,
        "Tore_Gast" integer DEFAULT NULL
) ;
```

Dataset 1 – Bundesliga
CREATE TABLE – Spieler

```
CREATE TABLE "Spieler" (
        "Spieler_ID" SERIAL PRIMARY KEY,
        "Vereins_ID" integer REFERENCES "Verein"("V_ID"),
        "Trikot_Nr" integer DEFAULT NULL,
        "Spieler_Name" varchar(255) NOT NULL,
        "Land" varchar(255) DEFAULT NULL,
        "Spiele" integer DEFAULT NULL,
        "Tore" integer DEFAULT NULL,
        "Vorlagen" integer DEFAULT NULL
) ;
```

Dataset 1 – Bundesliga
CREATE TABLE – Verein

```
CREATE TABLE "Verein" (
        "V_ID" SERIAL PRIMARY KEY,
        "Name" varchar(255) NOT NULL UNIQUE,
        "Liga" integer REFERENCES "Liga"("Liga_Nr")
);
```

Dataset 2 – Hockey
Introduction

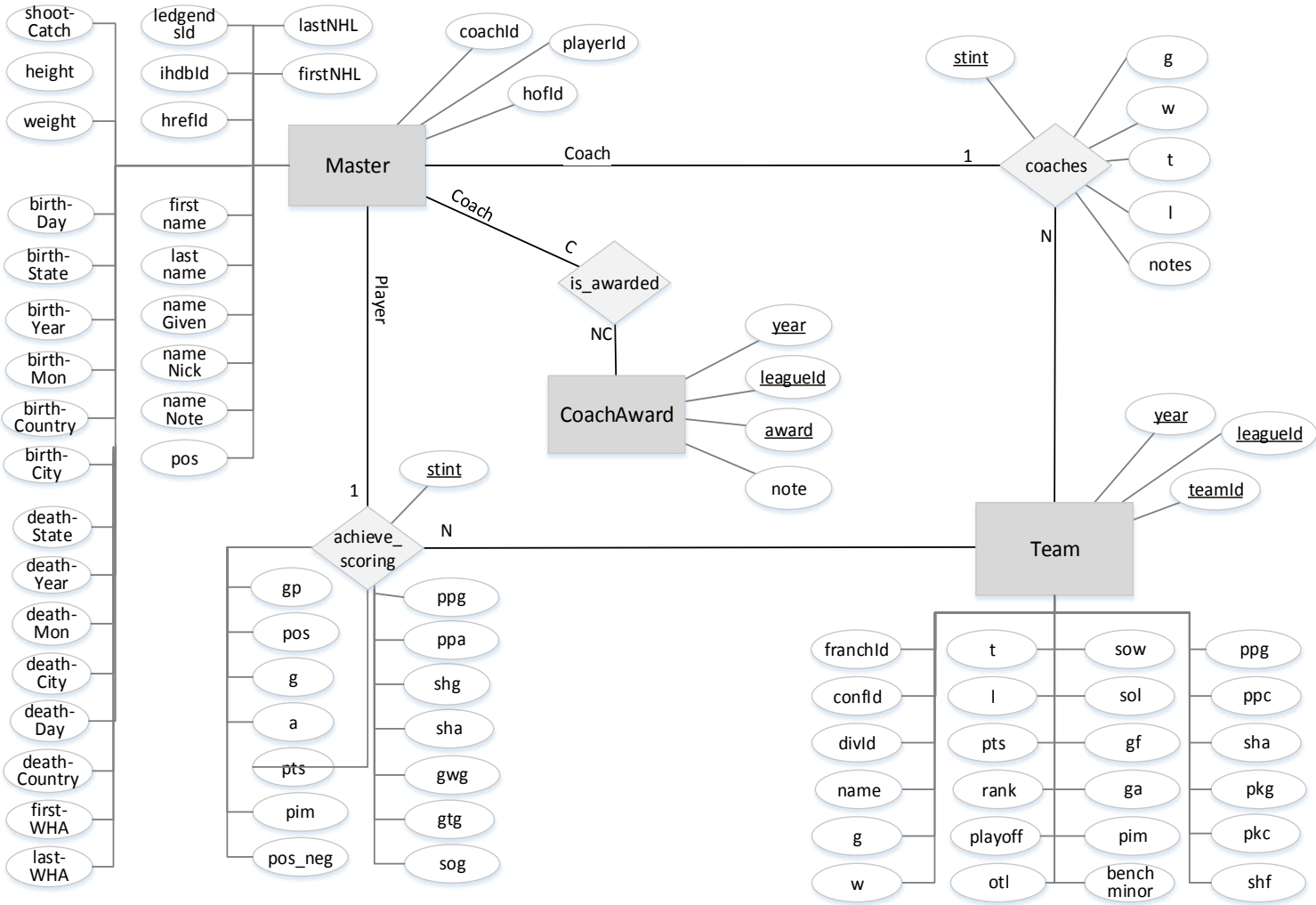
- leagues: NHL, NHA, PCHA, WCHL
- seasons since 1909 – 2012

- names and biographical informations
- individuell players, coaches and team statistics

- 7.800 players and coaches
- 1.500 team scorings per league and year
- 46.000 player scorings per team and year

Dataset 2 – Hockey
Introduction

- http://www.opensourcesports.com/hockey/
- Dataset download (23 CSV files)
- Data import via pgAdmin



Information Integration Exercise 1

Beck, Dücker, Mattfeld, Schneider

Chart **10**

Dataset 2 – Hockey
CREATE TABLE - Master

```
CREATE TABLE "master" (
   "playerid" VARCHAR(10) UNIQUE, "coachid" VARCHAR(10) UNIQUE,
   "hofid" VARCHAR(10) UNIQUE, "firstName" VARCHAR(20),
   "lastName" VARCHAR(30) NOT NULL, "nameNote" TEXT,
   "nameGiven" VARCHAR(50), "nameNick" VARCHAR(50),
   "height" INTEGER, "weight" INTEGER, "shootCatch" VARCHAR(1),
   "legendid" VARCHAR(10) UNIQUE, "ihdbid" VARCHAR(10) UNIQUE,
   "hrefid" VARCHAR(10) UNIQUE, "firstNHL" INTEGER, "lastNHL" INTEGER,
   "firstWHA" INTEGER, "lastWHA" INTEGER, "pos" VARCHAR(3),
   "birthYear" VARCHAR(4), "birthMon" INTEGER, "birthDay" INTEGER,
   "birthCountry" VARCHAR(50), "birthState" VARCHAR(50),
   "birthCity" VARCHAR(50), "deathYear" INTEGER, "deathMon" INTEGER,
   "deathDay" INTEGER, "deathCountry" VARCHAR(50),
   "deathState" VARCHAR(50), "deathCity" VARCHAR(50)
);
```

Dataset 2 – Hockey
CREATE TABLE - Coaches

```
CREATE TABLE "coaches" (
   "coachid" VARCHAR(10), "year" INTEGER, "teamid" VARCHAR(3),
   "leagueId" VARCHAR(4), "stint" INTEGER, "notes" TEXT,
   "g" INTEGER, "w" INTEGER, "l" INTEGER, "t" INTEGER,
   "postg" INTEGER, "postw" INTEGER, "postl" INTEGER, "postt" INTEGER,
   FOREIGN KEY("year", "leagueId", "teamid")
   REFERENCES team("year", "leagueId", "teamid"),
   FOREIGN KEY("coachid") REFERENCES master("coachid"),
   PRIMARY KEY("coachid", "year", "stint", "teamid", "leagueId")
);
```

Dataset 2 – Hockey
CREATE TABLE – Coach Award

```
CREATE TABLE "coachaward" (
   "coachid" VARCHAR(10),
   "award" VARCHAR(50),
   "year" INTEGER,
   "leagueId" VARCHAR(4),
   "note" TEXT,
   FOREIGN KEY("coachid") REFERENCES master("coachid"),
   PRIMARY KEY("coachid", "year", "award")
);
```

Dataset 2 – Hockey
CREATE TABLE - Team

```
CREATE TABLE "team" (
   "year" INTEGER, "leagueId" VARCHAR(4), "teamid" VARCHAR(3),
   "franchid" VARCHAR(3), "confid" VARCHAR(2), "divid" VARCHAR(2),
   "rank" INTEGER, "playoff" VARCHAR(5), "g" INTEGER,
   "w" INTEGER, "l" INTEGER, "t" INTEGER, "otl" INTEGER,
   "pts" INTEGER, "sow" INTEGER, "sol" INTEGER, "gf" INTEGER,
   "ga" INTEGER, "name" VARCHAR(50), "pim" INTEGER,
   "benchminor" INTEGER, "ppg" INTEGER, "ppc" INTEGER,
   "sha" INTEGER, "pkg" INTEGER, "pkc" INTEGER, "shf" INTEGER,
   PRIMARY KEY("year", "leagueId", "teamid")
);
```

Dataset 2 – Hockey
CREATE TABLE - Scoring

```
CREATE TABLE "scoring" (
   "playerid" VARCHAR(10), "year" INTEGER, "stint" INTEGER, "teamid" VARCHAR(3),
   "leagueId" VARCHAR(4), "pos" VARCHAR(3), "gp" INTEGER, "g" INTEGER,
   "a" INTEGER, "pts" INTEGER, "pim" INTEGER, "pos_neg" INTEGER, "ppg" INTEGER,
   "ppa" INTEGER, "shg" INTEGER, "sha" INTEGER, "gwg" INTEGER, "gtg" INTEGER,
   "sog" INTEGER,  "postgp" INTEGER, "postg" INTEGER, "posta" INTEGER, "postpts" INTEGER,
   "postpim" INTEGER, "postpos_neg" INTEGER, "postppg" INTEGER, "postppa" INTEGER,
   "postshg" INTEGER, "postsha" INTEGER, "postgwg" INTEGER, "postsog" INTEGER,
   FOREIGN KEY("year", "leagueId", "teamid")
   REFERENCES team("year", "leagueId", "teamid"),
   FOREIGN KEY("playerid") REFERENCES master("playerid"),
   PRIMARY KEY("playerid", "year", "stint", "teamid")
);
```

IMDb Dataset

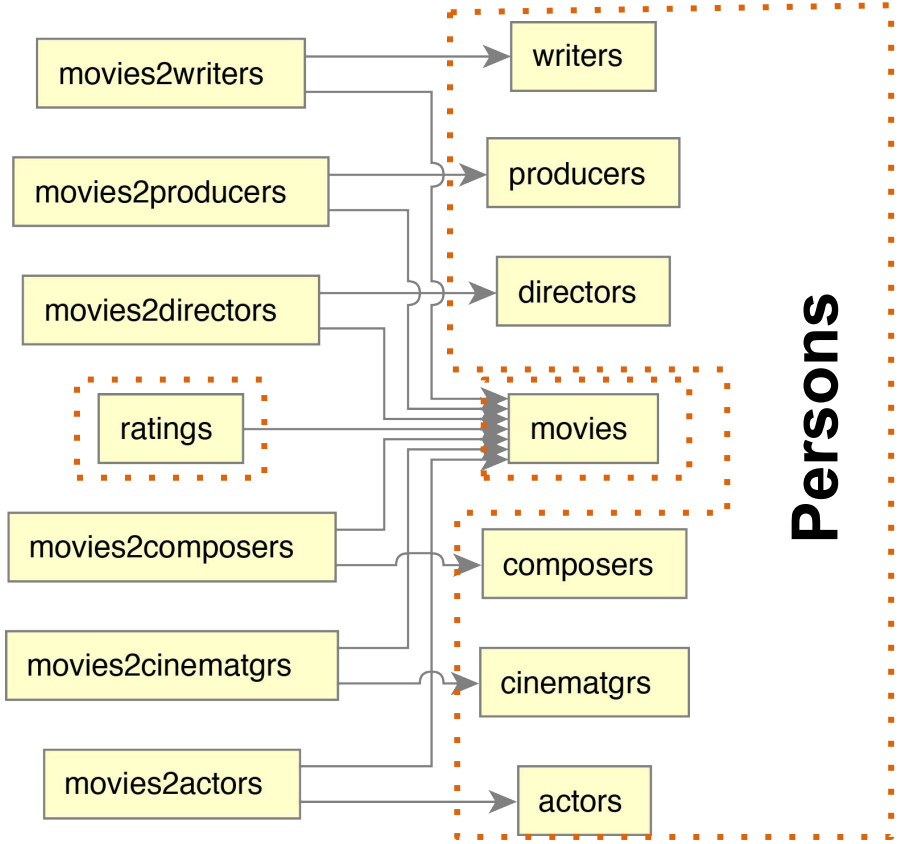
- **Topic:** Movies
- **Entities:** Movies, Persons, Ratings
- **Source:** http://www.imdb.com/interfaces
- **Extraction:**

Java Movie Database

Exercise 1
Extraction

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

IMDb Dataset



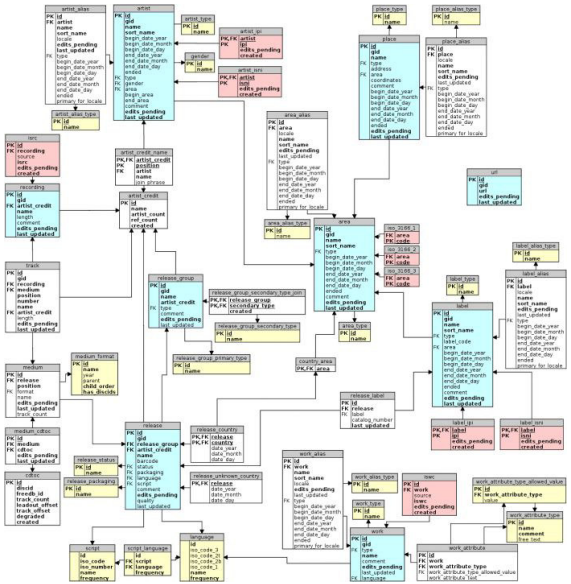
- 3 entities
- 39 attributes
- e.g. 3,485,565 actors
- ~ 4 GB

Exercise 1
Extraction

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

MusicBrainz Dataset

- **Topic:** Music
- **Entities:** Artist, Release, Label, Area, Release Group
- **Source:** http://www.musicbrainz.org

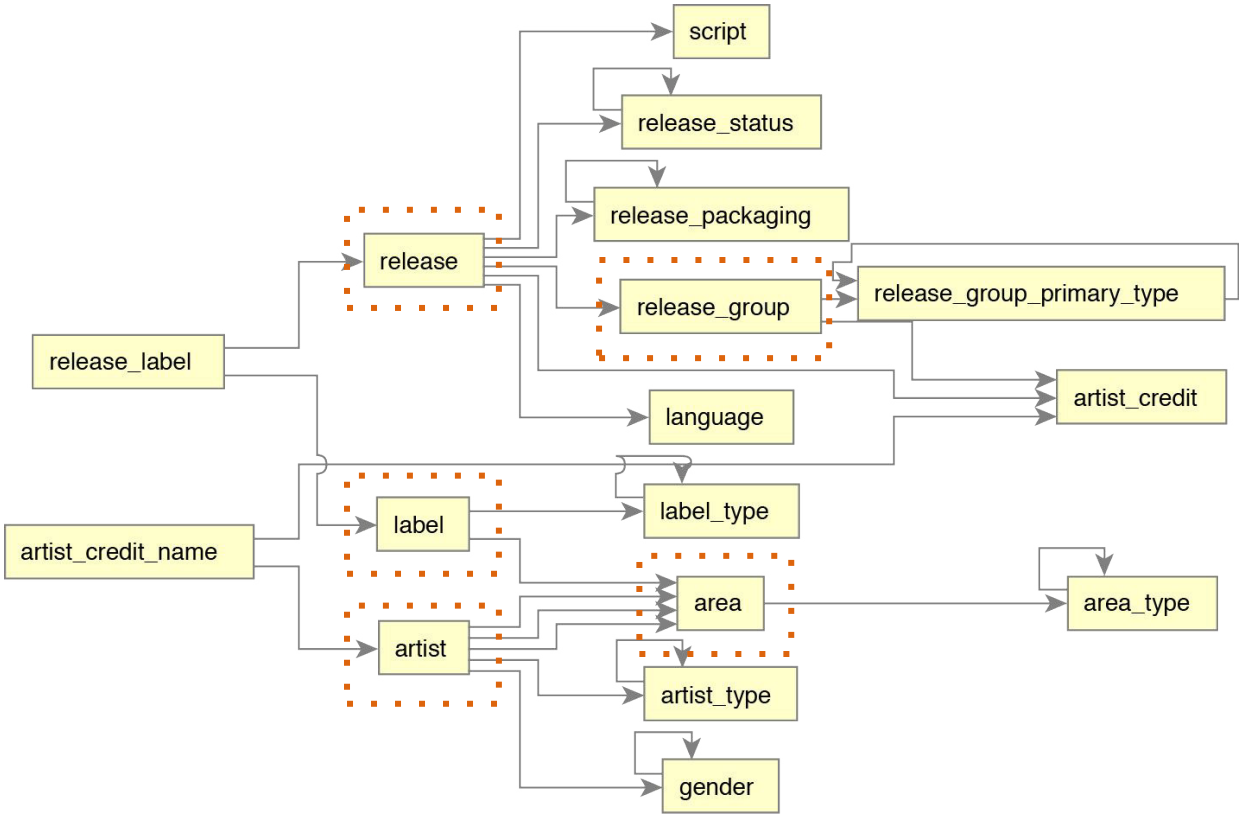


- **Omitted:** Recording, Track, Medium, Work, URL, Place, …
- **Extraction:**
  - CSV-like format
  - Import with COPY FROM command

Exercise 1
Extraction

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

MusicBrainz Dataset



- 5 entities
- 133 attributes
- e.g. 1,003,001 artists
- ~ 1 GB

Exercise 1
Extraction

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

Chart **4**

IMDB CREATE TABLE Statements

- CREATE TABLE actors (
  actorid integer NOT NULL,
  name character varying(250) NOT NULL,
  sex character(1));
- CREATE TABLE cinematgrs (
  cinematid integer NOT NULL,
  name character varying(250) NOT NULL);
- CREATE TABLE composers (
  composerid integer NOT NULL,
  name character varying(250) NOT NULL);
- CREATE TABLE directors (
  directorid integer NOT NULL,
  name character varying(250) NOT NULL);
- CREATE TABLE movies (
  movieid integer NOT NULL,
  title character varying(400) NOT NULL,
  year character varying(100),
  imdbid character varying(10));

IMDB CREATE TABLE Statements

- CREATE TABLE movies2actors (
  movieid integer NOT NULL,
  actorid integer NOT NULL,
  as_character character varying(1000));
- CREATE TABLE movies2cinematgrs (
  movieid integer NOT NULL,
  cinematid integer NOT NULL,
  addition character varying(1000));
- CREATE TABLE movies2composers (
  movieid integer NOT NULL,
  composerid integer NOT NULL,
  addition character varying(1000));
- CREATE TABLE movies2directors (
  movieid integer NOT NULL,
  directorid integer NOT NULL,
  addition character varying(1000));
- CREATE TABLE movies2producers (
  movieid integer NOT NULL,
  producerid integer NOT NULL,
  addition character varying(1000));

IMDB CREATE TABLE Statements

- CREATE TABLE movies2writers (
  movieid integer NOT NULL,
  writerid integer NOT NULL,
  addition character varying(1000));
- CREATE TABLE producers (
  producerid integer NOT NULL,
  name character varying(250) NOT NULL);
- CREATE TABLE ratings (
  movieid integer NOT NULL,
  rank character(4) NOT NULL,
  votes integer,
  distribution character(10) NOT NULL);
- CREATE TABLE writers (
  writerid integer NOT NULL,
  name character varying(250) NOT NULL);

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

IMDB CONSTRAINT Statements

- ALTER TABLE ONLY actors
  ADD CONSTRAINT actors_pkey PRIMARY KEY (actorid);
- ALTER TABLE ONLY cinematgrs
  ADD CONSTRAINT cinematgrs_pkey PRIMARY KEY (cinematid);
- ALTER TABLE ONLY composers
  ADD CONSTRAINT composers_pkey PRIMARY KEY (composerid);
- ALTER TABLE ONLY directors
  ADD CONSTRAINT directors_pkey PRIMARY KEY (directorid);
- ALTER TABLE ONLY movies
  ADD CONSTRAINT movies_pkey PRIMARY KEY (movieid);
- ALTER TABLE ONLY producers
  ADD CONSTRAINT producers_pkey PRIMARY KEY (producerid);
- ALTER TABLE ONLY writers
  ADD CONSTRAINT writers_pkey PRIMARY KEY (writerid);

IMDB CONSTRAINT Statements

- ALTER TABLE ONLY movies2actors
  ADD CONSTRAINT movies2actors_actorid_fkey FOREIGN KEY (actorid) REFERENCES actors(actorid);
- ALTER TABLE ONLY movies2actors
  ADD CONSTRAINT movies2actors_movieid_fkey FOREIGN KEY (movieid) REFERENCES movies(movieid);
- ALTER TABLE ONLY movies2cinematgrs
  ADD CONSTRAINT movies2cinematgrs_cinematid_fkey FOREIGN KEY (cinematid) REFERENCES cinematgrs(cinematid);
- ALTER TABLE ONLY movies2cinematgrs
  ADD CONSTRAINT movies2cinematgrs_movieid_fkey FOREIGN KEY (movieid) REFERENCES movies(movieid);
- ALTER TABLE ONLY movies2composers
  ADD CONSTRAINT movies2composers_composerid_fkey FOREIGN KEY (composerid) REFERENCES composers(composerid);
- ALTER TABLE ONLY movies2composers
  ADD CONSTRAINT movies2composers_movieid_fkey FOREIGN KEY (movieid) REFERENCES movies(movieid);
- ALTER TABLE ONLY movies2directors
  ADD CONSTRAINT movies2directors_directorid_fkey FOREIGN KEY (directorid) REFERENCES directors(directorid);

IMDB CONSTRAINT Statements

- ALTER TABLE ONLY movies2directors
  ADD CONSTRAINT movies2directors_movieid_fkey FOREIGN KEY (movieid) REFERENCES movies(movieid);
- ALTER TABLE ONLY movies2producers
  ADD CONSTRAINT movies2producers_movieid_fkey FOREIGN KEY (movieid) REFERENCES movies(movieid);
- ALTER TABLE ONLY movies2producers
  ADD CONSTRAINT movies2producers_producerid_fkey FOREIGN KEY (producerid) REFERENCES producers(producerid);
- ALTER TABLE ONLY movies2writers
  ADD CONSTRAINT movies2writers_movieid_fkey FOREIGN KEY (movieid) REFERENCES movies(movieid);
- ALTER TABLE ONLY movies2writers
  ADD CONSTRAINT movies2writers_writerid_fkey FOREIGN KEY (writerid) REFERENCES writers(writerid);
- ALTER TABLE ONLY ratings
  ADD CONSTRAINT ratings_movieid_fkey FOREIGN KEY (movieid) REFERENCES movies(movieid);

MusicBrainz CREATE TABLE Statements

- CREATE TABLE release_packaging (-- replicate

```
id              SERIAL primary key,
name            VARCHAR(255) NOT NULL,
parent          INTEGER references release_packaging (id),
child_order     INTEGER NOT NULL DEFAULT 0,
description     TEXT);
```

- CREATE TABLE release_status (-- replicate

```
id              SERIAL primary key,
name            VARCHAR(255) NOT NULL,
parent          INTEGER references release_status (id),
child_order     INTEGER NOT NULL DEFAULT 0,
description     TEXT);
```

- CREATE TABLE language (-- replicate

```
id              SERIAL primary key,
iso_code_2t     CHAR(3), -- ISO 639-2 (T)
iso_code_2b     CHAR(3), -- ISO 639-2 (B)
iso_code_1      CHAR(2), -- ISO 639
name            VARCHAR(100) NOT NULL,
frequency       INTEGER NOT NULL DEFAULT 0,
iso_code_3      CHAR(3)  -- ISO 639-3);
```

MusicBrainz CREATE TABLE Statements

- CREATE TABLE script (-- replicate
  - id serial primary key,
  - iso_code CHAR(4) NOT NULL, -- ISO 15924
  - iso_number CHAR(3) NOT NULL, -- ISO 15924
  - name VARCHAR(100) NOT NULL,
  - frequency INTEGER NOT NULL DEFAULT 0);
- CREATE TABLE release_group_primary_type (-- replicate
  - id serial primary key,
  - name VARCHAR(255) NOT NULL,
  - parent INTEGER references release_group_primary_type (id),
  - child_order INTEGER NOT NULL DEFAULT 0,
  - description TEXT);
- CREATE TABLE artist_credit (-- replicate
  - id serial primary key,
  - name VARCHAR NOT NULL,
  - artist_count SMALLINT NOT NULL,
  - ref_count INTEGER DEFAULT 0,
  - created TIMESTAMP WITH TIME ZONE DEFAULT NOW());

Exercise 1
Extraction

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

MusicBrainz CREATE TABLE Statements

- CREATE TABLE release_group (-- replicate (verbose)

```
id              serial primary key,
gid             UUID NOT NULL,
name            VARCHAR NOT NULL,
artist_credit   INTEGER NOT NULL references artist_credit (id),
type            INTEGER references release_group_primary_type (id),
comment         VARCHAR(255) NOT NULL DEFAULT '',
edits_pending   INTEGER NOT NULL DEFAULT 0 CHECK (edits_pending >= 0),
last_updated    TIMESTAMP WITH TIME ZONE DEFAULT NOW());
```

- CREATE TABLE gender (-- replicate

```
id              serial primary key,
name            VARCHAR(255) NOT NULL,
parent          INTEGER references gender (id),
child_order     INTEGER NOT NULL DEFAULT 0,
description     TEXT);
```

- CREATE TABLE artist_type (-- replicate

```
id              serial primary key,
name            VARCHAR(255) NOT NULL,
parent          INTEGER references artist_type (id),
child_order     INTEGER NOT NULL DEFAULT 0,
description     TEXT);
```

MusicBrainz CREATE TABLE Statements

- CREATE TABLE area_type (-- replicate
  id serial primary key, -- PK
  name VARCHAR(255) NOT NULL,
  parent INTEGER references area_type (id),
  child_order INTEGER NOT NULL DEFAULT 0,
  description TEXT);
- CREATE TABLE area (-- replicate (verbose)
  id serial primary key, -- PK
  gid uuid NOT NULL,
  name VARCHAR NOT NULL,
  type INTEGER references area_type (id),
  edits_pending INTEGER NOT NULL DEFAULT 0 CHECK (edits_pending >=0),
  last_updated TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  begin_date_year SMALLINT,
  begin_date_month SMALLINT,
  begin_date_day SMALLINT,
  end_date_year SMALLINT,
  end_date_month SMALLINT,
  end_date_day SMALLINT,
  ended BOOLEAN NOT NULL DEFAULT FALSE);

MusicBrainz CREATE TABLE Statements

```
CREATE TABLE artist ( -- replicate (verbose)
    id                  serial primary key,
    gid                 UUID NOT NULL,
    name                VARCHAR NOT NULL,
    sort_name           VARCHAR NOT NULL,
    begin_date_year     SMALLINT,
    begin_date_month            SMALLINT,
    begin_date_day      SMALLINT,
    end_date_year       SMALLINT,
    end_date_month              SMALLINT,
    end_date_day        SMALLINT,
    type                INTEGER  references artist_type (id),
    area                INTEGER  references area (id),
    gender              INTEGER  references gender (id),
    comment             VARCHAR(255) NOT NULL DEFAULT '',
    edits_pending       INTEGER NOT NULL DEFAULT 0 CHECK (edits_pending >= 0),
    last_updated        TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    ended               BOOLEAN NOT NULL DEFAULT FALSE
    begin_area          INTEGER references area (id),
    end_area            INTEGER references area (id));
```

Exercise 1
Extraction

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

MusicBrainz CREATE TABLE Statements

- CREATE TABLE label_type (-- replicate
  id serial primary key,
  name VARCHAR(255) NOT NULL,
  parent INTEGER references label_type (id),
  child_order INTEGER NOT NULL DEFAULT 0,
  description TEXT);
- CREATE TABLE label (-- replicate (verbose)
  id serial primary key,
  gid UUID NOT NULL,
  name VARCHAR NOT NULL,
  begin_date_year SMALLINT,
  begin_date_month SMALLINT,
  begin_date_day SMALLINT,
  end_date_year SMALLINT,
  end_date_month SMALLINT,
  end_date_day SMALLINT,
  label_code INTEGER CHECK (label_code > 0 AND label_code < 100000),
  type INTEGER references label_type (id),
  area INTEGER references area (id),
  comment VARCHAR(255) NOT NULL DEFAULT '',
  edits_pending INTEGER NOT NULL DEFAULT 0 CHECK (edits_pending >= 0),
  last_updated TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  ended BOOLEAN NOT NULL DEFAULT FALSE);

MusicBrainz CREATE TABLE Statements

- CREATE TABLE release (-- replicate (verbose)

```
    id              serial primary key,
    gid             UUID NOT NULL,
    name            VARCHAR NOT NULL,
    artist_credit   INTEGER NOT NULL references artist_credit (id),
    release_group   INTEGER NOT NULL references release_group (id),
    status          INTEGER references release_status (id),
    packaging       INTEGER references release_packaging (id),
    language        INTEGER references language (id),
    script          INTEGER references script (id),
    barcode         VARCHAR(255),
    comment         VARCHAR(255) NOT NULL DEFAULT '',
    edits_pending   INTEGER NOT NULL DEFAULT 0 CHECK (edits_pending >= 0),
    quality         SMALLINT NOT NULL DEFAULT -1,
    last_updated    TIMESTAMP WITH TIME ZONE DEFAULT NOW());
```

- CREATE TABLE release_label (-- replicate (verbose)

```
    id              SERIAL Primary key,
    release         INTEGER NOT NULL references release (id),
    label           INTEGER references label (id),
    catalog_number  VARCHAR(255),
    last_updated    TIMESTAMP WITH TIME ZONE DEFAULT NOW());
```

Exercise 1
Extraction

Tobias Bleifuß
Susanne Bülow
Tim Draeger
Tsun Yin Wong

MusicBrainz CREATE TABLE Statements

- CREATE TABLE artist_credit_name (-- replicate (verbose)
  artist_credit INTEGER NOT NULL references artist_credit (id),
  position SMALLINT NOT NULL, -- PK
  artist INTEGER NOT NULL references artist (id),
  name VARCHAR NOT NULL,
  join_phrase TEXT NOT NULL DEFAULT '');

Information Integration
Übung 1

Rosa Braatz, Henriette Dinger,
Sören Oldag, Dominic Sauer

05.11.15

Datensätze

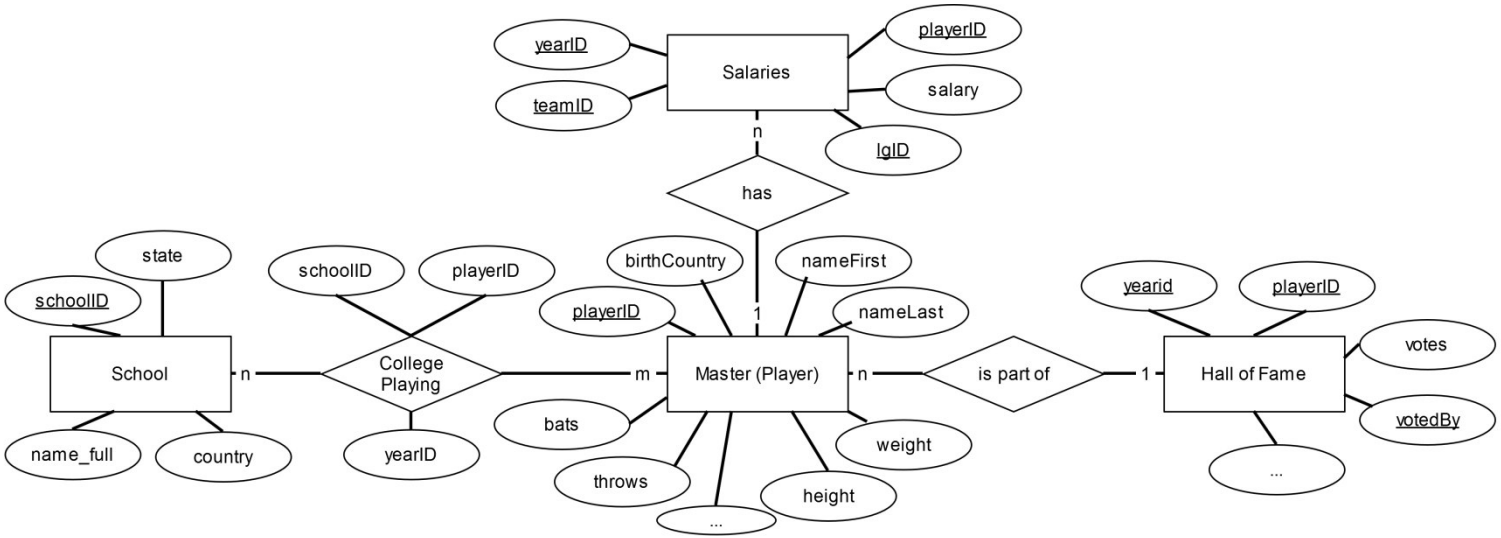
Sean Lahman – Baseballarchiv

- 31MB
- Datenformat: CSV, SQL
- 24 Tabellen über Baseballspieler und ihre Statistiken
- Beschränkung auf Spieler, Gehalt, Hall of Fame und Schule
- http://seanlahman.com/baseball-archive/statistics/

GND – Deutsche Normdatei

- 13,1GB
- Datenformat: XML/Marc21, XMLRDF
- Beschränkung auf Personen, geographische Orte, Werke
- http://datendienst.dnb.de/cgi-bin/mabit.pl?userID=GNDxml&pass=vupabe&cmd=login

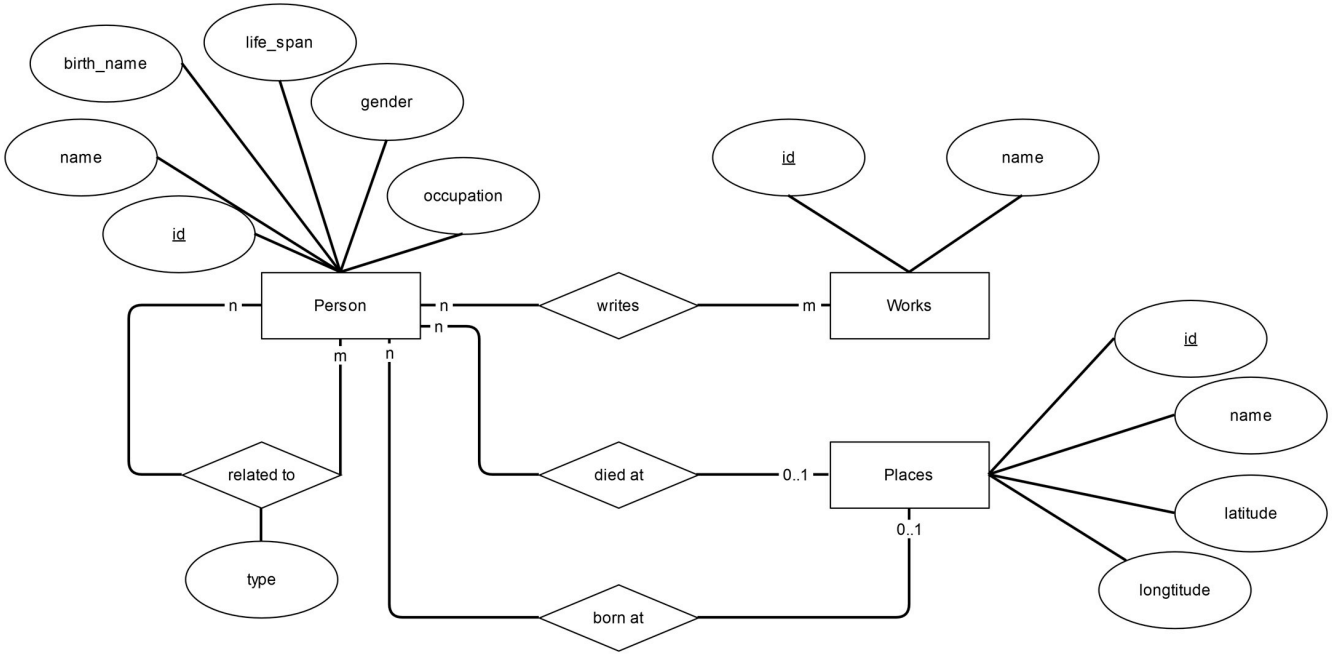
Sean Lahman - Baseballarchiv



Information Integration – Übung 1

Rosa Braatz,
Henriette Dinger,
Sören Oldag,
Dominic Sauer

GND – Deutsche Normdatei



Information Integration – Übung 1

Rosa Braatz,
Henriette Dinger,
Sören Oldag,
Dominic Sauer

Sean Lahman - Baseballarchiv

- „Create-Table statements“ und darin enthaltene „add constraint statements“ können unter folgendem Link gefunden werden: https://github.com/soldag/InformationIntegration
- Da die Daten die Fremdschlüsselbeziehung verletzen, ist unter https://github.com/soldag/InformationIntegration auch ein Script, das invalide Daten entfernt.

GND – Deutsche Normdatei

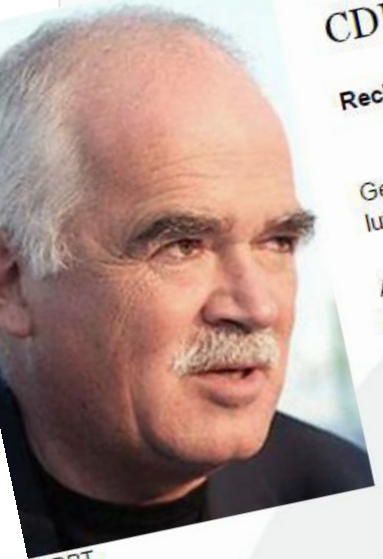
- „Create-Table statements“ und darin enthaltene „add constraint statements“ können unter folgendem Link gefunden werden: https://github.com/soldag/InformationIntegration
- Unter https://github.com/soldag/InformationIntegration ist ein Extractor zu finden, der die Daten aus der GND extrahiert und in die Datenbank schreibt. Da die Daten die Fremdschlüsselbeziehung verletzen, werden invalide Daten entfernt.

bundestag.de

Name	Fraktion	Angaben	Einkünfte	Mindesthöhe der Einkünfte	Stufen*									
					1	2	3	4	5	6	7	8	9	10
Dr. Peter Gauweiler	CDU/CSU	41	41	1 888 500,00 €	7	7	6	6	5	2	1	1	2	4
Albert Stegemann	CDU/CSU			902 500,00 €	2	1	1	2	2	1				3
Hans-Georg von der Marwitz	CDU/CSU			587 500,00 €	2	1	6		3	2		2	1	
Philipp Mißfelder				380 000,00 €					1	1		3		
Rudolf				336 000,00 €		26	35							

Dr. Peter Gauweiler, CDU/CSU*

Rechtsanwalt, Publizist, Staatsminister a. D.



© DBT

Geboren am 22. Juni 1949 in München; evangelisch-lutherisch; verheiratet, vier Kinder.

Abitur. Studium der Rechte, 1978 Promotion zum Dr. jur.

Rechtsanwalt und Autor in München.

Ehrenfreund der Stadt Tel Aviv, Ehrenbürger der Stadt Marktredwitz, Ehrenoffizier der Gebirgsschützenkompanie Traunstein, Ehrenmitglied der bayerischen Heimat- und Trachtenvereine Isargau, Ehrenmitglied im Trachtenverein München-Großhadern; 1986 Verleihung der Medaille "München leuchtet - den Freunden Münchens" in Gold, 1989 Verleihung des Verdienstkreuzes am Bande des Verdienstordens der Bundesrepublik Deutschland, 1991 Verleihung des Bayerischen Verdienstordens.

Informationen zur Fra

CDU CS

Fraktion im Deutschen Bun

Reden des Md

Neue Red

Hinweise

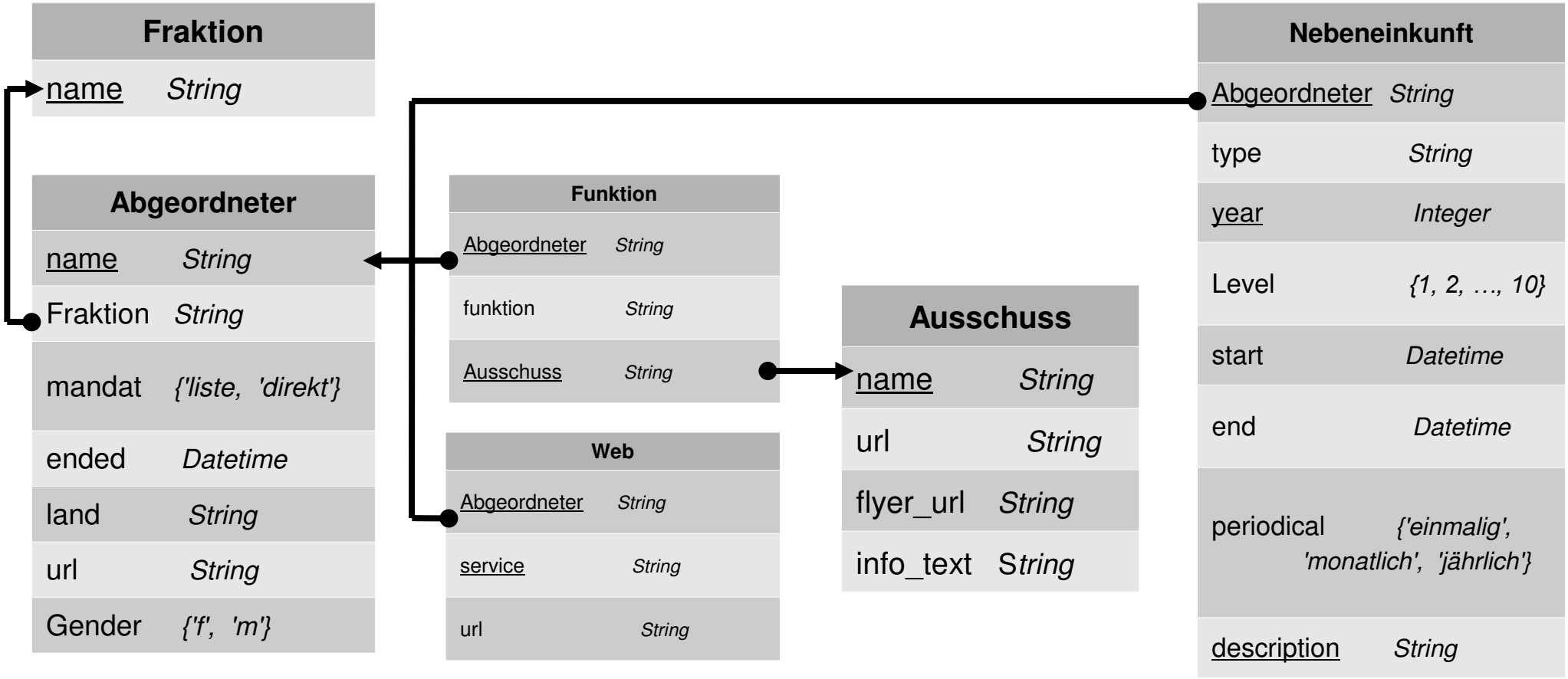
Namen

Abstin

Entgeltliche Tätigkeiten neben dem Mandat

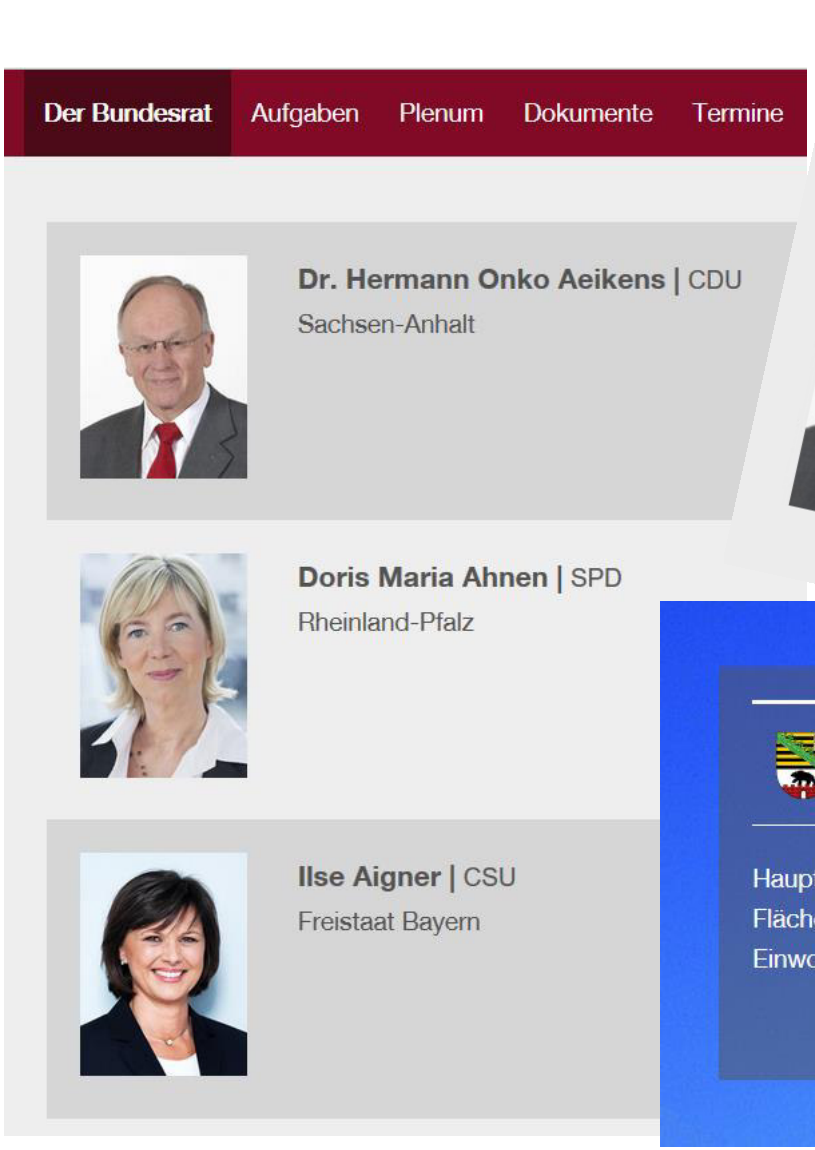
Rechtsanwälte Bub, Gauweiler & Partner, München, Rechtsanwalt
Mandant 01, 2013, Stufe 5; 2014, Stufe 10
Mandant 02, 2013, Stufe 10; 2014, Stufe 1
Mandant 03, 2013, Stufe 2; 2014, Stufe 5
Mandant 04, 2013, Stufe 1; 2014, Stufe 9
Mandant 05, 2013, Stufe 4; 2014, Stufe 10
Mandant 06, 2013, Stufe 6; 2014, Stufe 4
Mandant 07, 2013, Stufe 5; 2014, Stufe 2
Mandant 08, 2013, Stufe 5; 2014, Stufe 8
Mandant 09, 2013, Stufe 1; 2014, Stufe 9
Mandant 10, 2013, Stufe 3; 2014, Stufe 5
Mandant 11, 2013, Stufe 4; 2014, Stufe 7
Mandant 12, 2013, Stufe 1; 2014, Stufe 4

bundestag.de – Structure

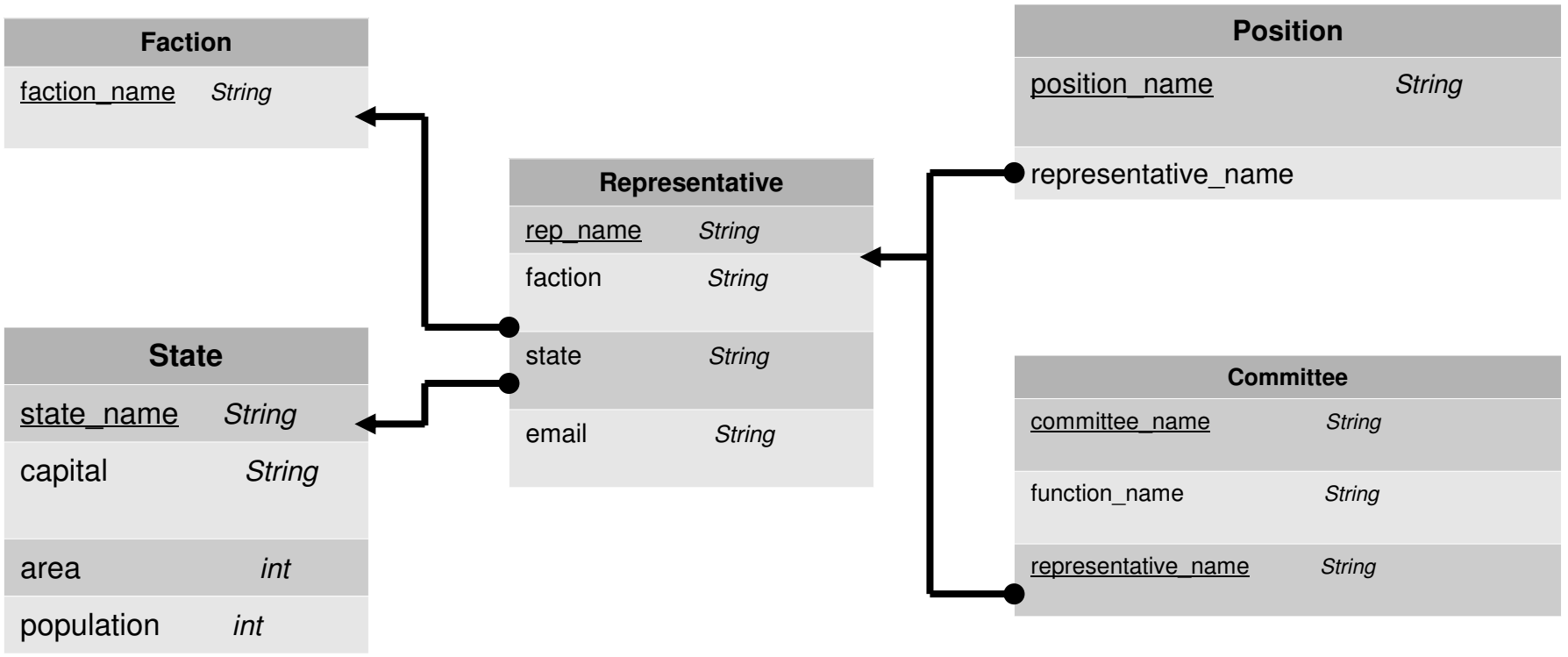


- Some data already parsed by https://apps.opendatacity.de/nebeneinkuenfte/ (available as JSON)
- Remaining data manually downloaded/parsed from http://bundestag.de

bundesrat.de



bundesrat.de - Structure



- All data manually downloaded/parsed from http://bundesrat.de
- Schema designed manually

bundestag.de – SQL

```
CREATE TABLE Fraktion
( name VARCHAR(50) PRIMARY KEY );


CREATE TABLE Abgeordneter
( name VARCHAR(50) PRIMARY KEY,
  Fraktion VARCHAR(50) REFERENCES
Fraktion (name),
  mandat VARCHAR(50),
  ended DATETIME,
  land VARCHAR(30),
  url VARCHAR(255),
  gender VARCHAR(1) );


CREATE TABLE Nebeneinkunft
( Abgeordneter VARCHAR(50) REFERENCES
Abgeordneter (name),
  type VARCHAR(30),
  year INT,
  level INT,
  start TIMESTAMP,
  end TIMESTAMP,
  periodical VARCHAR(10),
  description VARCHAR(1),
  PRIMARY KEY (Abgeordneter, year,
description) );
```

```
CREATE TABLE Web
( Abgeordneter VARCHAR(50) REFERENCES
Abgeordneter (name),
  service VARCHAR(100),
  url VARCHAR(255),
  PRIMARY KEY (Abgeordneter, service) );


CREATE TABLE Ausschuss
( name VARCHAR(255) PRIMARY KEY,
  url VARCHAR(255),
  flyer_url VARCHAR(255),
  info_text VARCHAR(255) );


CREATE TABLE Funktion
( Abgeordneter VARCHAR(50) REFERENCES
Abgeordneter (name),
  funktion VARCHAR(30),
  Ausschuss VARCHAR(50) REFERENCES
Ausschuss (name),
  PRIMARY KEY (Abgeordneter, Ausschuss)
);
```

bundesrat.de – SQL

```
CREATE TABLE Faction
( faction_name VARCHAR(255) PRIMARY KEY
);


CREATE TABLE State
( state_name VARCHAR(50) PRIMARY KEY,
  capital VARCHAR(30),
  area INTEGER,
  population INTEGER );


CREATE TABLE Representative
( rep_name VARCHAR(255) PRIMARY KEY,
  faction VARCHAR(255) REFERENCES
Faction (faction_name),
  state VARCHAR(255) REFERENCES State
(state_name),
  email VARCHAR(100) );
```

```
CREATE TABLE Position
( position_name VARCHAR(255) PRIMARY KEY,
  representative_name VARCHAR(255)
REFERENCES Representative (rep_name) );


CREATE TABLE Committee
( committee_name VARCHAR(255),
  function_name VARCHAR(255),
  representative_name VARCHAR(255)
REFERENCES Representative (rep_name),
  PRIMARY KEY (committee_name,
representative_name) );
```

Information Integration

Exercise 1

Fabian Eckert, Dustin Gläser, Manuel Hegner, Martin Zabel

Data Sets

- **Encyclopaedia Metallum**
  - http://www.metal-archives.com/
- **Goodreads**
  - https://www.goodreads.com/api/

Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Schema

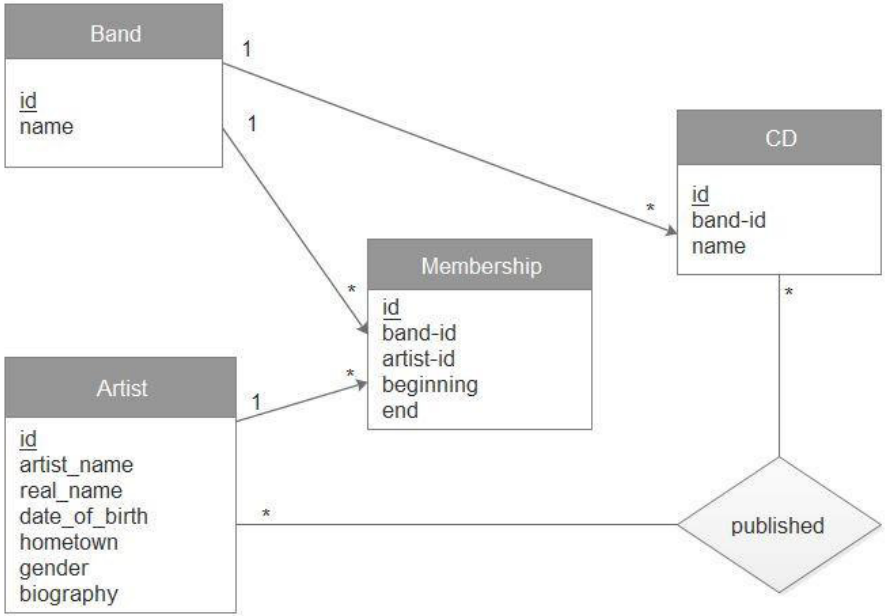
Metal Archives

- **Main Entities**
  - **Artists**
    - **id, artist_name, real_name**
    - **date_of_birth, hometown**
    - **gender, biography**
  - **Bands**
    - **id, name**
  - **CDs**
    - **band_id, name**

Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Schema

Metal Archives



Information Integration
Exercise 1

Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Schema

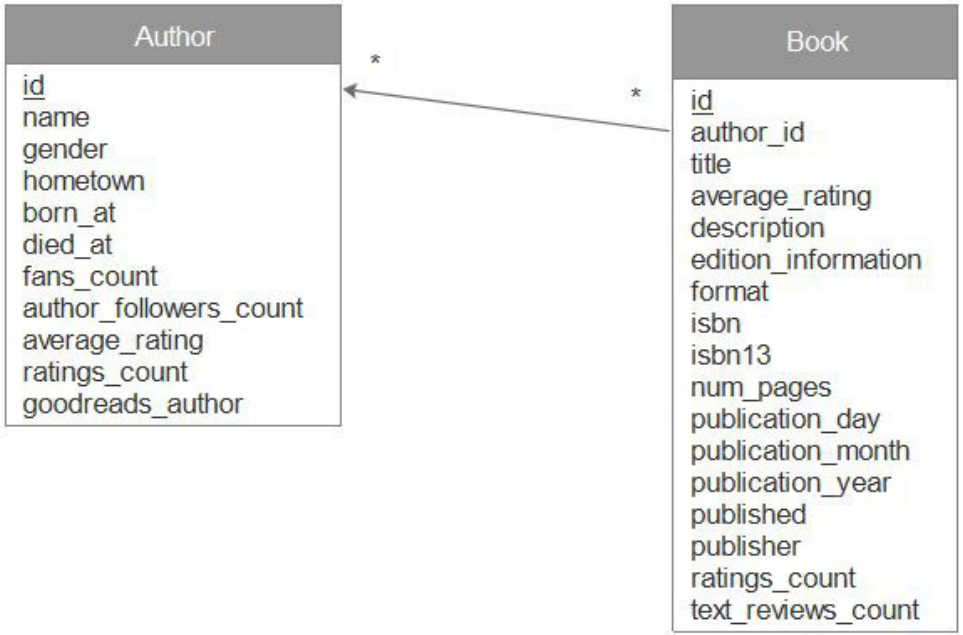
Goodreads

- **Main Entities**
  - **Authors**
    - **ID, name, gender**
    - **hometown, place of birth, place of death**
    - **fans and followers**
    - **ratings**
  - **Books**
    - **ID, title, publisher, page number**
    - **edition information, format, ISBN-10, ISBN-13**
    - **publication date**
    - **ratings**

Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Schema

Goodreads



Information Integration
Exercise 1

Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Appendix

Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Schema

Metal Archives

```
CREATE TABLE IF NOT EXISTS `metallum_artists` (
  `id` int(8) unsigned NOT NULL,
  `artist_name` varchar(100) NOT NULL,
  `real_name` varchar(100) NOT NULL,
  `date_of_birth` date DEFAULT NULL,
  `hometown` varchar(100) DEFAULT NULL,
  `gender` set('male','female') DEFAULT NULL,
  `biography` mediumtext
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `metallum_artists`
  ADD PRIMARY KEY (`id`);
```

Schema

Metal Archives

```
CREATE TABLE IF NOT EXISTS `metallum_membership` (
  `id` int(8) unsigned NOT NULL,
  `band-id` int(8) unsigned NOT NULL,
  `artist-id` int(8) unsigned NOT NULL,
  `beginning` date DEFAULT NULL,
  `end` date DEFAULT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `metallum_membership`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `metallum_membership`
ADD CONSTRAINT `artist_membership` FOREIGN KEY (`artist-id`) REFERENCES `metallum_artist` (`id`);

ALTER TABLE `metallum_membership`
ADD CONSTRAINT `band_membership` FOREIGN KEY (`band-id`) REFERENCES `metallum_band` (`id`);
```

Schema

Metal Archives

```
CREATE TABLE IF NOT EXISTS `metallum_band` (
  `id` int(8) unsigned NOT NULL,
  `name` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `metallum_band`
  ADD PRIMARY KEY (`id`);

CREATE TABLE IF NOT EXISTS `metallum_cd` (
  `id` int(8) unsigned NOT NULL,
  `band-id` int(8) unsigned NOT NULL,
  `name` varchar(100) NOT NULL,
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `metallum_cd`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `metallum_cd`
ADD CONSTRAINT `band_cd` FOREIGN KEY (`band-id`) REFERENCES `metallum_band` (`id`);
```


Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Schema

Metal Archives

```
CREATE TABLE IF NOT EXISTS `metallum_published` (
  `artist-id` int(8) unsigned NOT NULL,
  `cd-id` int(8) unsigned NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `metallum_published`
ADD CONSTRAINT `artist_published` FOREIGN KEY (`band-id`) REFERENCES `metallum_artist` (`id`);

ALTER TABLE `metallum_publshed`
ADD CONSTRAINT `cd_published` FOREIGN KEY (`band-id`) REFERENCES `metallum_cd` (`id`);
```


Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Schema

Goodreads

```
CREATE TABLE IF NOT EXISTS `goodreads_authors` (
  `id` int(8) unsigned NOT NULL,
  `name` varchar(100) NOT NULL,
  `gender` set('male','female') DEFAULT NULL,
  `hometown` varchar(100) DEFAULT NULL,
  `born_at` char(10) DEFAULT NULL,
  `died_at` char(10) DEFAULT NULL,
  `works_count` smallint(5) unsigned NOT NULL,
  `fans_count` mediumint(6) NOT NULL,
  `author_followers_count` mediumint(6) NOT NULL,
  `average_rating` decimal(2,1) unsigned NOT NULL,
  `ratings_count` tinyint(1) unsigned NOT NULL,
  `goodreads_author` set('true','false') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `goodreads_authors`
  ADD PRIMARY KEY (`id`);
```

Schema

Goodreads

```
CREATE TABLE IF NOT EXISTS `goodreads_books` (
  `id` int(8) unsigned NOT NULL,
  `author_id` int(8) unsigned NOT NULL,
  `title` varchar(255) NOT NULL,
  `average_rating` decimal(3,2) DEFAULT NULL,
  `description` mediumtext,
  `edition_information` varchar(132) DEFAULT NULL,
  `format` varchar(58) DEFAULT NULL,
  `isbn` char(10) DEFAULT NULL,
  `isbn13` char(13) DEFAULT NULL,
  `num_pages` int(5) DEFAULT NULL,
  `publication_day` tinyint(4) DEFAULT NULL,
  `publication_month` tinyint(4) DEFAULT NULL,
  `publication_year` smallint(6) DEFAULT NULL,
  `published` smallint(5) unsigned DEFAULT NULL,
  `publisher` varchar(100) DEFAULT NULL,
  `ratings_count` mediumint(9) NOT NULL,
  `text_reviews_count` mediumint(8) unsigned NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
ALTER TABLE `goodreads_books`
  ADD PRIMARY KEY (`id`), ADD KEY `author_id` (`author_id`);

ALTER TABLE `goodreads_books`
ADD CONSTRAINT `book_author` FOREIGN KEY (`author_id`)
REFERENCES `goodreads_authors` (`id`);
```


Fabian Eckert
Dustin Gläser
Manuel Hegner
Martin Zabel

Information Integration Exercise 1

Clemens Frahnow, Maximilian Grundke, Jan Lindemann, Jan Philipp Sachse

Personal data like

- Birthday and birthplace
- Gender
- Nationality

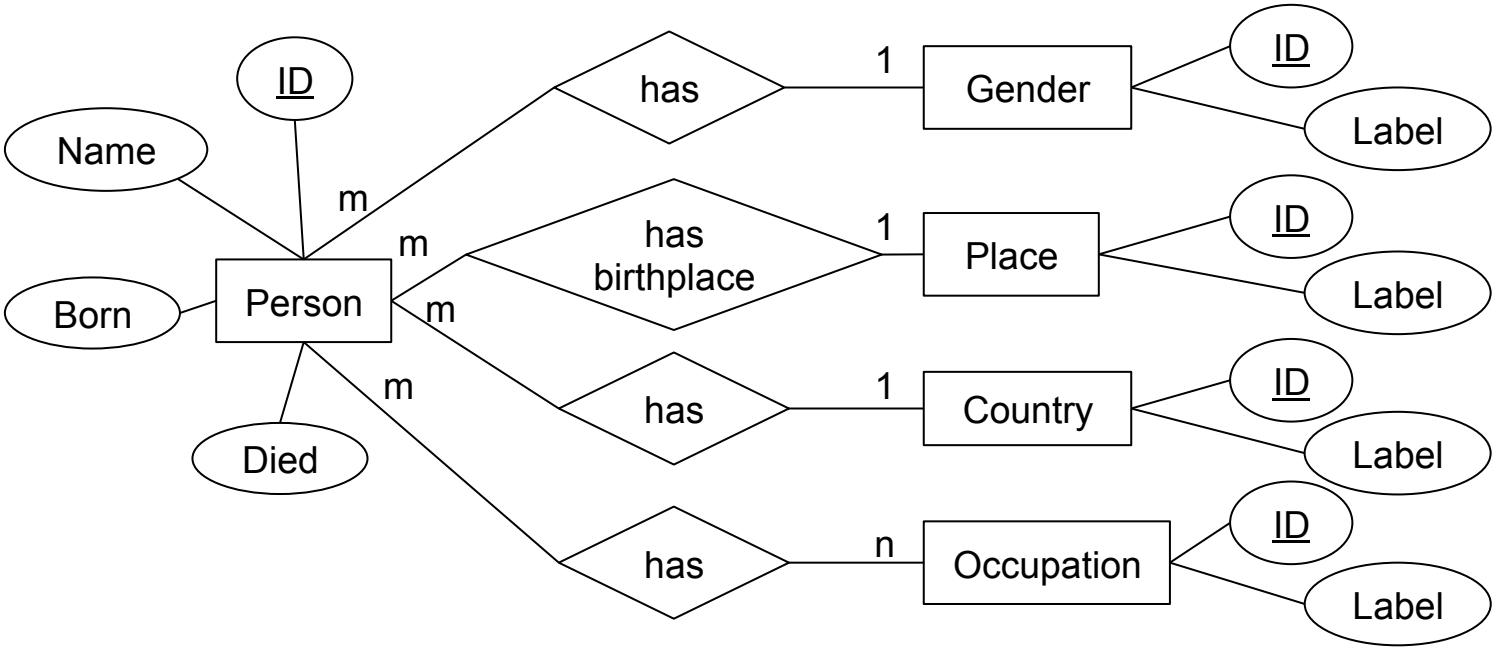
Extracted from Wikidata

- linked open data, organized in triples
- SPARQL query and Python script for extraction

Size

- 15,068,495 general entries
- 2,932,736 entries about humans
- limited to 10,000 in importer script

Clemens Frahnow, Maximilian Grundke, Jan Lindemann, Jan Philipp Sachse



Information Integration
Exercise 1

Clemens Frahnow, Maximilian Grundke, Jan Lindemann, Jan Philipp Sachse

Create Table Statements:

```
CREATE TABLE Person (ID VARCHAR NOT NULL, Name VARCHAR, Gender VARCHAR, Born VARCHAR, Birthplace VARCHAR, Country VARCHAR, Died VARCHAR, Occupation VARCHAR);

CREATE TABLE Gender (ID VARCHAR PRIMARY KEY, Label VARCHAR);

CREATE TABLE Occupation (ID VARCHAR PRIMARY KEY, Label VARCHAR);

CREATE TABLE Place (ID VARCHAR PRIMARY KEY, Label VARCHAR);

CREATE TABLE Country (ID VARCHAR PRIMARY KEY, Label VARCHAR);
```

Add Constraint Statements:

```
ALTER TABLE Person ADD CONSTRAINT person_gender_fk FOREIGN KEY (Gender) REFERENCES Gender (ID) MATCH FULL;

ALTER TABLE Person ADD CONSTRAINT person_birthplace_fk FOREIGN KEY (Birthplace) REFERENCES Place (ID) MATCH FULL;

ALTER TABLE Person ADD CONSTRAINT person_country_fk FOREIGN KEY (Country) REFERENCES Country (ID) MATCH FULL;

ALTER TABLE Person ADD CONSTRAINT person_occupation_fk FOREIGN KEY (Occupation) REFERENCES Occupation (ID) MATCH FULL;
```

Cinemalytics

Indian movie (Bollywood) database

- contains data about movies like title, genre, rating, release date
- actors that participated

Extracted from Cinemalytics

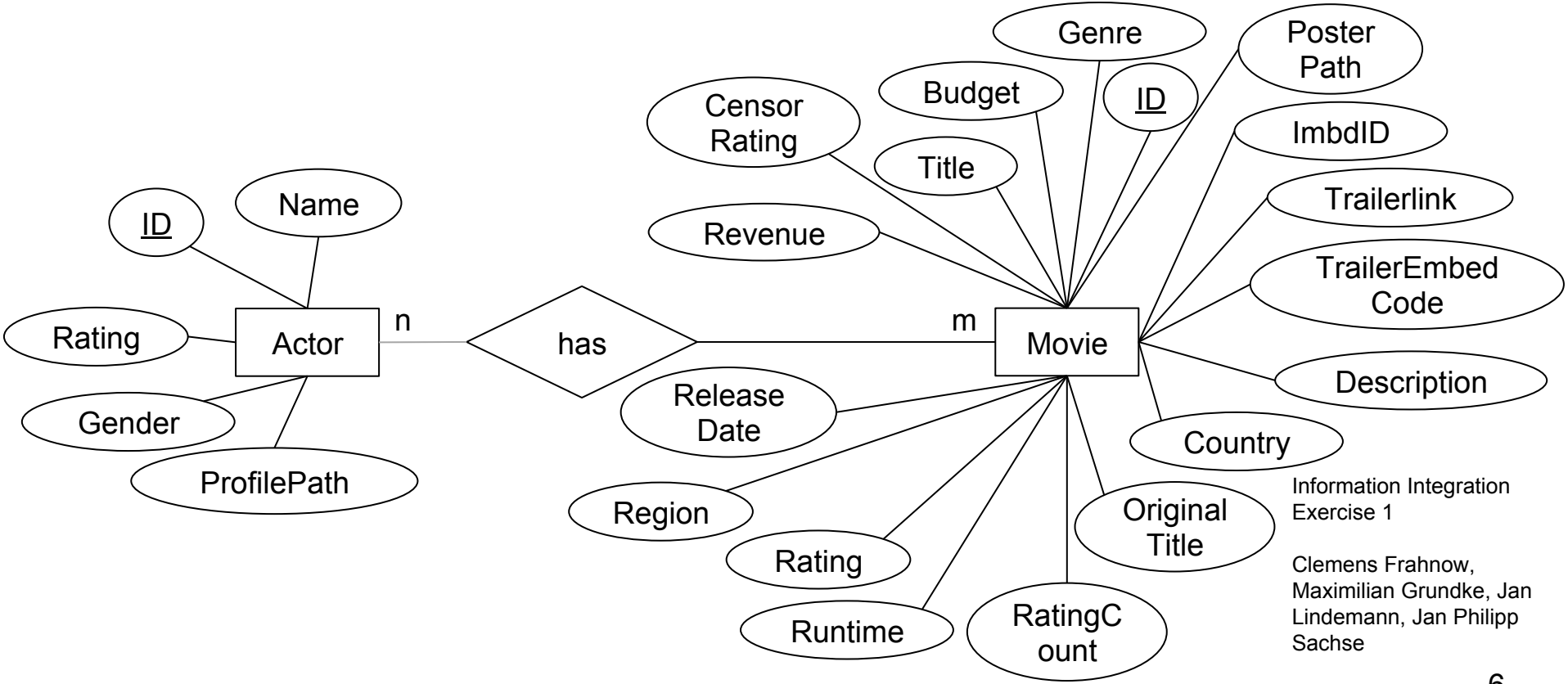
- provides REST API, returns json content
- python script
  - get all actor names, fetch IDs for actor names, fetch movies for actor IDs
  - problem: correctly encode actor names

Size

- 884 actors
- 1900 movies
- 4584 roles in the movies

Clemens Frahnow, Maximilian Grundke, Jan Lindemann, Jan Philipp Sachse

Cinemalytics



Information Integration Exercise 1

Clemens Frahnow, Maximilian Grundke, Jan Lindemann, Jan Philipp Sachse

Cinemalytics

Create Table Statements:

```
CREATE TABLE Actors (Id VARCHAR PRIMARY KEY, Name VARCHAR, Gender VARCHAR, ProfilePath VARCHAR, Rating NUMERIC(4, 2));

CREATE TABLE Actor_Movies (Actor_Id VARCHAR, Movie_Id VARCHAR);

CREATE TABLE Movies (Id VARCHAR PRIMARY KEY, ImdbId VARCHAR, OriginalTitle VARCHAR, Title VARCHAR, Description VARCHAR, TrailerLink VARCHAR, TrailerEmbedCode VARCHAR, Country VARCHAR, Region VARCHAR, Genre VARCHAR, RatingCount INTEGER, Rating NUMERIC(4, 2), CensorRating VARCHAR, ReleaseDate VARCHAR, Runtime REAL, Budget REAL, Revenue REAL, PosterPath VARCHAR);
```

Add Constraint Statements:

```
ALTER TABLE Actor_Movies ADD PRIMARY KEY (Actor_Id, Movie_Id);

ALTER TABLE Actor_Movies ADD CONSTRAINT actorfk FOREIGN KEY (Actor_Id) REFERENCES Actors (Id) MATCH FULL;

ALTER TABLE Actor_Movies ADD CONSTRAINT moviefk FOREIGN KEY (Movie_Id) REFERENCES Movies (Id) MATCH FULL;
```


Clemens Frahnow, Maximilian Grundke, Jan Lindemann, Jan Philipp Sachse

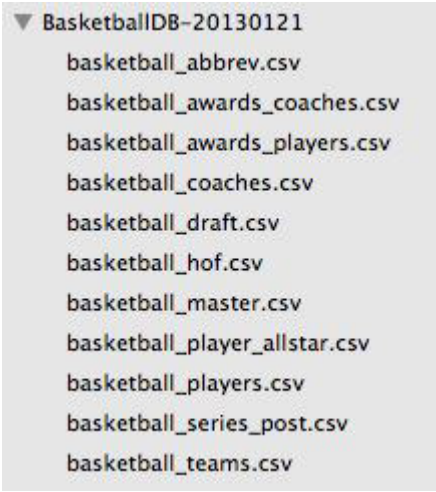
Information Integration 2015

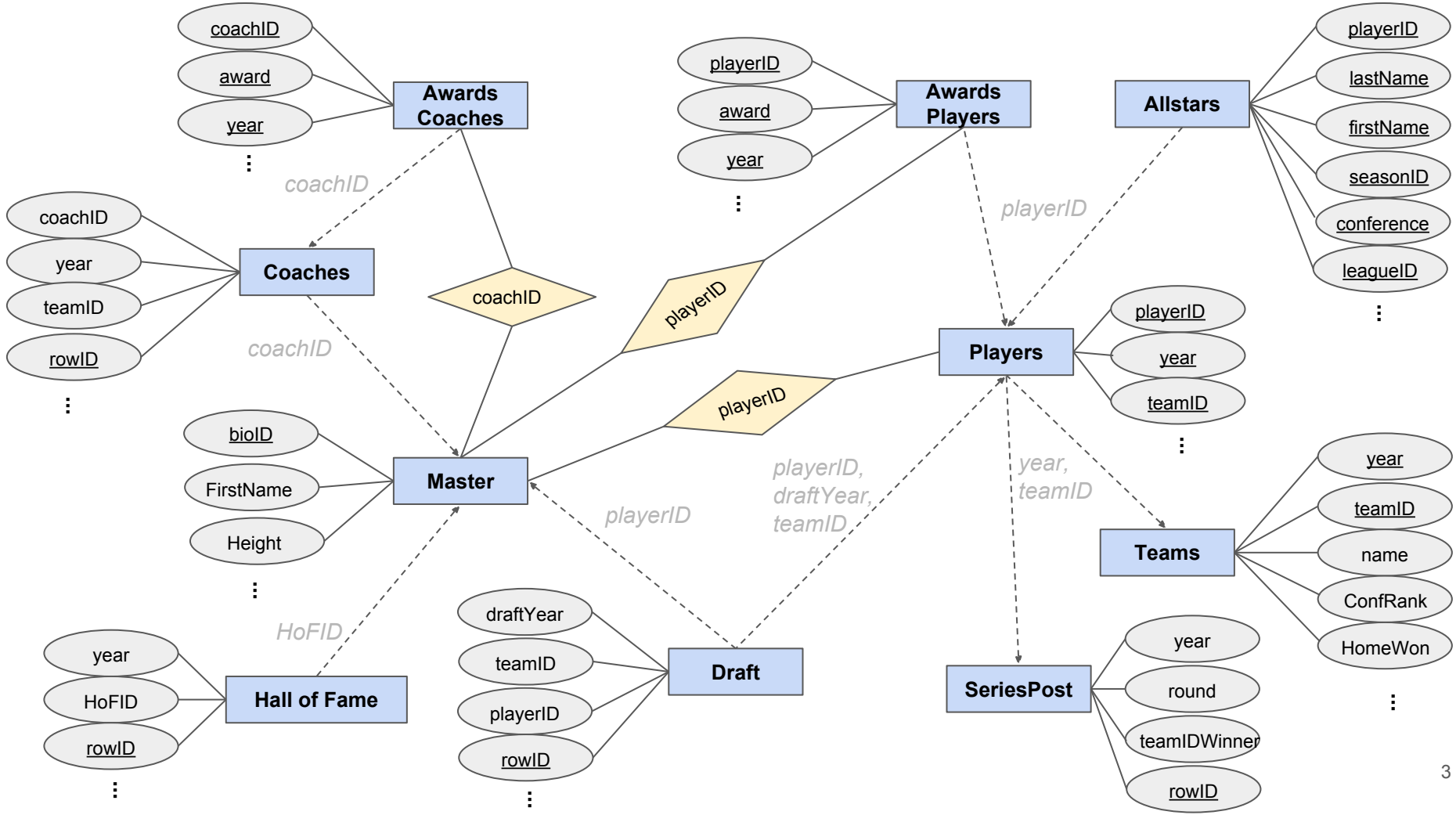
Exercise 1

Johannes Jasper, Norman Rzepka,
Thomas Werkmeister, Florian Moritz

Dataset 1 opensourcesports.com/basketball/

- U.S. basketball data (1937 - 2011)
- NBA, ABA leagues
- focus on player and team statistics
  - points scored, balls stolen, three-pointers scored
  - season statistics
- data is split over several CSV files
- 45k entries total, 5k persons thereof




coachID
award
year
Awards Coaches
coachID
coachID
year
teamID
rowID
Coaches
coachID
coachID
playerID
award
year
Awards Players
playerID
playerID
playerID
lastName
firstName
seasonID
conference
leagueID
Allstars
playerID
year
teamID
Players
bioID
FirstName
Height
Master
playerID
playerID,
draftYear,
teamID
year,
teamID
year
teamID
name
ConfRank
HomeWon
Teams
HoFID
year
HoFID
rowID
Hall of Fame
draftYear
teamID
playerID
rowID
Draft
year
round
teamIDWinner
rowID
SeriesPost

```
CREATE TABLE bb_master (
    "bioID" character varying(9) NOT NULL,
    "useFirst" character varying(128),
    "firstName" character varying,
    "middleName" character varying,
    "lastName" character varying,
    "nameGiven" character varying,
    "fullGivenName" character varying,
    "nameSuffix" character varying,
    "nameNick" character varying,
    "pos" character varying(128),
    "firstseason" integer,
    "lastseason" integer,
    "height" numeric(5,2),
    "weight" numeric(5,2),
    "college" character varying,
    "collegeOther" character varying,
    "birthDate" character varying(10),
    "birthCity" character varying,
    "birthState" character varying(128),
    "birthCountry" character varying(128),
    "highSchool" character varying,
    "hsCity" character varying,
    "hsState" character varying,
    "hsCountry" character varying(128),
    "deathDate" character varying(10),
    "race" character varying(128)
);
```

tables

```
CREATE TABLE bb_teams (
    "year" integer NOT NULL,
    "lgID" character varying(4),
    "tmID" character varying(3) NOT NULL,
    "franchID" character varying(3),
    "confID" character varying(2),
    "divID" character varying(2),
    "rank" integer,
    "confRank" integer,
    "playoff" character varying(2),
    "name" character varying,
    "o_fgm" integer,
    "o_fga" integer,
    "o_ftm" integer,
    "o_fta" integer,
    "o_3pm" integer,
    "o_3pa" integer,
    "o_oreb" integer,
    "o_dreb" integer,
    "o_reb" integer,
    "o_asts" integer,
    "o_pf" integer,
    "o_stl" integer,
    "o_to" integer,
    "o_blk" integer,
    "o_pts" integer,
    "d_fgm" integer,
    "d_fga" integer,
    "d_ftm" integer,
    "d_fta" integer,
    "d_3pm" integer,
    "d_3pa" integer,
    "d_oreb" integer,
    "d_dreb" integer,
    "d_reb" integer,
    "d_asts" integer,
    "d_pf" integer,
    "d_stl" integer,
    "d_to" integer,
    "d_blk" integer,
    "d_pts" integer,
    "o_tmRebound" integer,
    "d_tmRebound" integer,
    "homeWon" integer,
    "homeLost" integer,
    "awayWon" integer,
    "awayLost" integer,
    "neutWon" integer,
    "neutLoss" integer,
    "confWon" integer,
    "confLoss" integer,
    "divWon" integer,
    "divLoss" integer,
    "pace" integer,
    "won" integer,
    "lost" integer,
    "games" integer,
    "min" integer,
    "arena" character varying,
    "attendance" integer,
    "bbtmID" character varying(3)
);
```

```sql
CREATE TABLE bb_coaches (
    "coachID" character varying(9),
    "year" integer,
    "tmID" character varying(3),
    "lgID" character varying(4),
    "stint" integer,
    "won" integer,
    "lost" integer,
    "post_wins" integer,
    "post_losses" integer,
    "rowID" integer NOT NULL
);

CREATE TABLE bb_series_post (
    "year" integer,
    "round" character varying(4),
    "series" character varying(1),
    "tmIDWinner" character varying(3),
    "lgIDWinner" character varying(4),
    "tmIDLoser" character varying(3),
    "lgIDLoser" character varying(4),
    "W" integer,
    "L" integer,
    "rowID" integer NOT NULL
);
```

```sql
CREATE TABLE bb_awards_coaches (
    "year" integer NOT NULL,
    "coachID" character varying(9) NOT NULL,
    "award" character varying NOT NULL,
    "lgID" character varying(3),
    "note" character varying
);

CREATE TABLE bb_awards_players (
    "playerID" character varying(9) NOT NULL,
    award character varying NOT NULL,
    year integer NOT NULL,
    "lgID" character varying(4),
    note character varying,
    pos character varying(3)
);

CREATE TABLE bb_hof (
    "year" integer,
    "hofID" character varying(9),
    "name" character varying,
    "category" character varying(128),
    "rowID" integer NOT NULL
);
```

```
CREATE TABLE bb_players (
    "playerID" character varying(9) NOT NULL,
    "year" integer NOT NULL,
    "stint" integer,
    "tmID" character varying(3) NOT NULL,
    "lgID" character varying(4),
    "GP" integer,
    "GS" integer,
    "minutes" integer,
    "points" integer,
    "oRebounds" integer,
    "dRebounds" integer,
    "rebounds" integer,
    "assists" integer,
    "steals" integer,
    "blocks" integer,
    "turnovers" integer,
    "PF" integer,
    "fgAttempted" integer,
    "fgMade" integer,
    "ftAttempted" integer,
    "ftMade" integer,
    "threeAttempted" integer,
    "threeMade" integer,
    "PostGP" integer,
    "PostGS" integer,
    "PostMinutes" integer,

...
```

```
...
    "PostPoints" integer,
    "PostoRebounds" integer,
    "PostdRebounds" integer,
    "PostRebounds" integer,
    "PostAssists" integer,
    "PostSteals" integer,
    "PostBlocks" integer,
    "PostTurnovers" integer,
    "PostPF" integer,
    "PostfgAttempted" integer,
    "PostfgMade" integer,
    "PostftAttempted" integer,
    "PostftMade" integer,
    "PostthreeAttempted" integer,
    "PostthreeMade" integer,
    "note" character varying
);
```

```
CREATE TABLE bb_player_allstars (
    "playerID" character varying(9) NOT NULL,
    "last_name" character varying NOT NULL,
    "first_name" character varying NOT NULL,
    "seasonID" integer NOT NULL,
    "conference" character varying NOT NULL,
    "leagueID" character varying(4) NOT NULL,
    "games_played" integer,
    "minutes" integer,
    "points" integer,
    "o_rebounds" integer,
    "d_rebounds" integer,
    "rebounds" integer,
    "assists" integer,
    "steals" integer,
    "blocks" integer,
    "turnovers" integer,
    "personal_fouls" integer,
    "fg_attempted" integer,
    "fg_made" integer,
    "ft_attempted" integer,
    "ft_made integer",
    "three_attempted" integer,
    "three_made" integer
);
```

```
CREATE TABLE bb_draft (
    "draftYear" integer,
    "draftRound" integer,
    "draftSelection" integer,
    "draftOverall" integer,
    "tmID" character varying(3),
    "firstName" character varying,
    "lastName" character varying,
    "suffixName" character varying,
    "playerID" character varying(9),
    "draftFrom" character varying,
    "lgID" character varying(4),
    "rowID" integer NOT NULL
);
```

primary keys

```
ALTER TABLE ONLY bb_awards_coaches
    ADD CONSTRAINT bb_awards_coaches_pkey PRIMARY KEY (year, "coachID", award);

ALTER TABLE ONLY bb_awards_players
    ADD CONSTRAINT bb_awards_players_pkey PRIMARY KEY ("playerID", award, year);

ALTER TABLE ONLY bb_coaches
    ADD CONSTRAINT bb_coaches_pkey PRIMARY KEY ("rowID");

ALTER TABLE ONLY bb_draft
    ADD CONSTRAINT bb_draft_pkey PRIMARY KEY ("rowID");

ALTER TABLE ONLY bb_master
    ADD CONSTRAINT bb_master_pkey PRIMARY KEY ("bioID");

ALTER TABLE ONLY bb_player_allstars
    ADD CONSTRAINT bb_player_allstars_pkey PRIMARY KEY ("playerID", last_name, first_name, "seasonID",
    conference, "leagueID");

ALTER TABLE ONLY bb_players
    ADD CONSTRAINT bb_players_pkey PRIMARY KEY ("playerID", year, "tmID");

ALTER TABLE ONLY bb_series_post
    ADD CONSTRAINT bb_series_post_pkey PRIMARY KEY ("rowID");

ALTER TABLE ONLY bb_teams
    ADD CONSTRAINT bb_teams_pkey PRIMARY KEY (year, "tmID");
```

foreign keys

```
ALTER TABLE ONLY bb_awards_coaches
    ADD CONSTRAINT "bb_awards_coaches_coachID_fkey" FOREIGN KEY ("coachID") REFERENCES bb_master("bioID");

ALTER TABLE ONLY bb_awards_players
    ADD CONSTRAINT "bb_awards_players_playerID_fkey" FOREIGN KEY ("playerID") REFERENCES bb_master("bioID");

ALTER TABLE ONLY bb_players
    ADD CONSTRAINT "bb_players_playerID_fkey" FOREIGN KEY ("playerID") REFERENCES bb_master("bioID");
```

Dataset 1 opensourcesports.com/basketball/

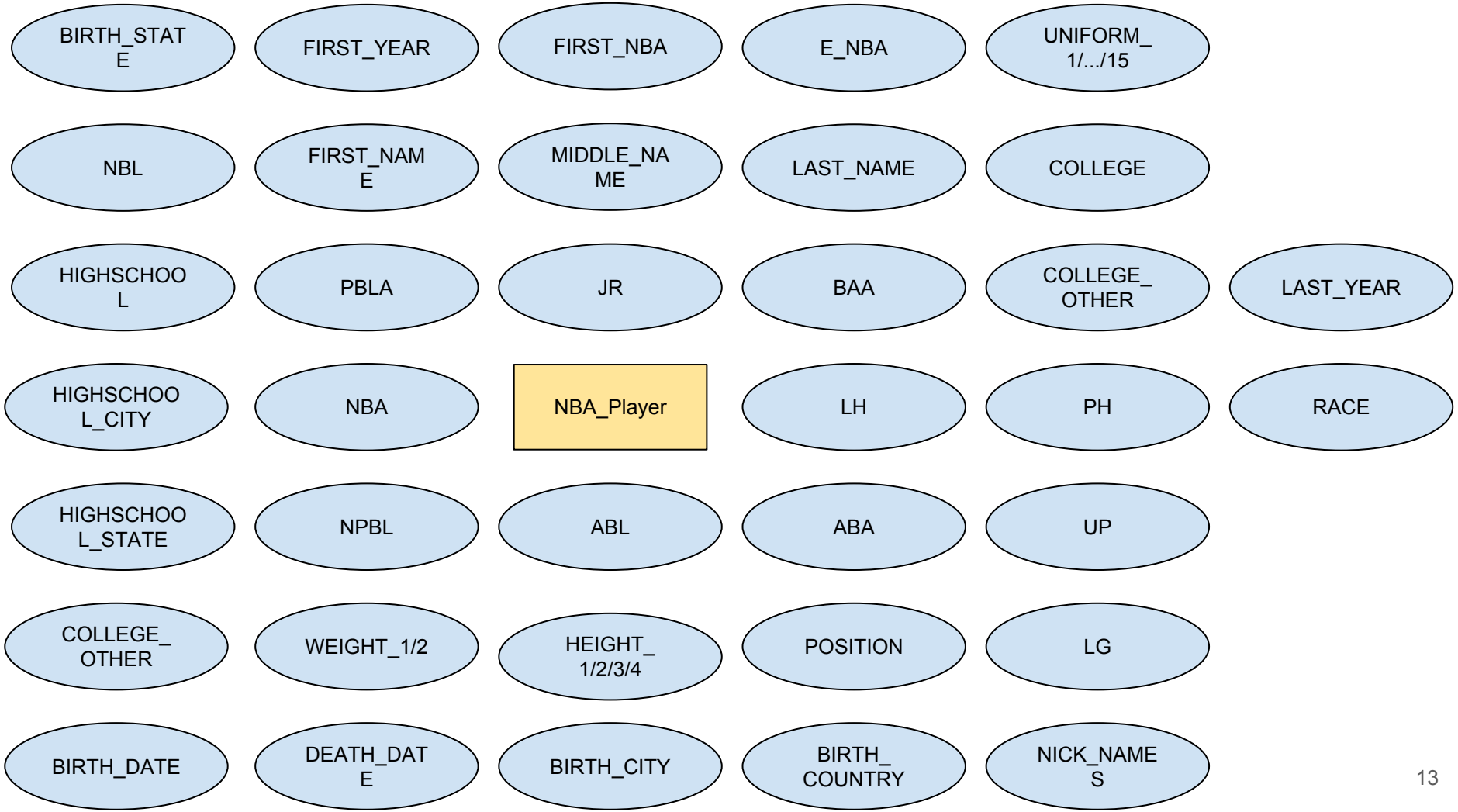
- data set needs cleaning:
  - several intended primary keys not working
    - empty values
    - duplicates
    - wrong values
  - we introduced rowIDs as surrogate primary keys several times
  - intended foreign keys not possible in this state

Examples

(Coaches → Master)	Key (coachID)=(dehnere01) is not present in table "bb_master".
(Drafts → Players)	Key (draftYear, tmID, playerID)=(1967, ANA, hardyda01) is not present in table "bb_players".
(Players → Master)	Key (playerID)=(athari01) is not present in table "bb_master".

Dataset 2

- U.S. basketball biographic data (1937 - 2011)
- NBA, ABA, BAA, PBLA leagues
- focus on player data
  - height, weight, nicknames, uniforms
  - education
- data is split over 2 CSV files in a single schema
- 4k entries total


BIRTH_STATE
FIRST_YEAR
FIRST_NBA
E_NBA
UNIFORM_1/.../15
NBL
FIRST_NAME
MIDDLE_NAME
LAST_NAME
COLLEGE
HIGHSCHOOL
PBLA
JR
BAA
COLLEGE_OTHER
LAST_YEAR
HIGHSCHOOL_CITY
NBA
NBA_Player
LH
PH
RACE
HIGHSCHOOL_STATE
NPBL
ABL
ABA
UP
COLLEGE_OTHER
WEIGHT_1/2
HEIGHT_1/2/3/4
POSITION
LG
BIRTH_DATE
DEATH_DATE
BIRTH_CITY
BIRTH_COUNTRY
NICK_NAMES

Dataset 2

Dataset needs normalization:

- all data is stored in one table
- empty cells
- “array” types

Examples

- uniforms are spread over 15 columns
- league assignments are binary columns
- height is spread over 4 columns (from/to range, foot/inch components)
- weight is spread over 2 columns (from/to range)

```
CREATE TABLE nba (
    lg text,
    nbl text,
    baa text,
    pbla text,
    nba text,
    npbl text,
    abl text,
    aba text,
    ph text,
    up text,
    first_name text,
    middle_name text,
    last_name text,
    jr text,
    height_1 numeric,
    height_2 numeric,
    height_3 numeric,
    height_4 numeric,
    lh text,
    weight_1 numeric,
    weight_2 numeric,
    "position" text,
    college text,
    college_other text,
    highschool text,
    highschool_city text,
    highschool_state text,
    highschool_country text,
    birth_date text,
    death_date text,
    birth_city text,
    birth_state text,
    birth_country text,
    nick_names text,
    first_year integer,
    first_nba integer,
    e_nba integer,
    last_year integer,
    race text,
    uniform_1 text,
    uniform_2 text,
    uniform_4 text,
    uniform_5 text,
    uniform_6 text,
    uniform_7 text,
    uniform_8 text,
    uniform_9 text,
    uniform_10 text,
    uniform_11 text,
    uniform_12 text,
    uniform_13 text,
    uniform_14 text,
    uniform_15 text
);
```

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

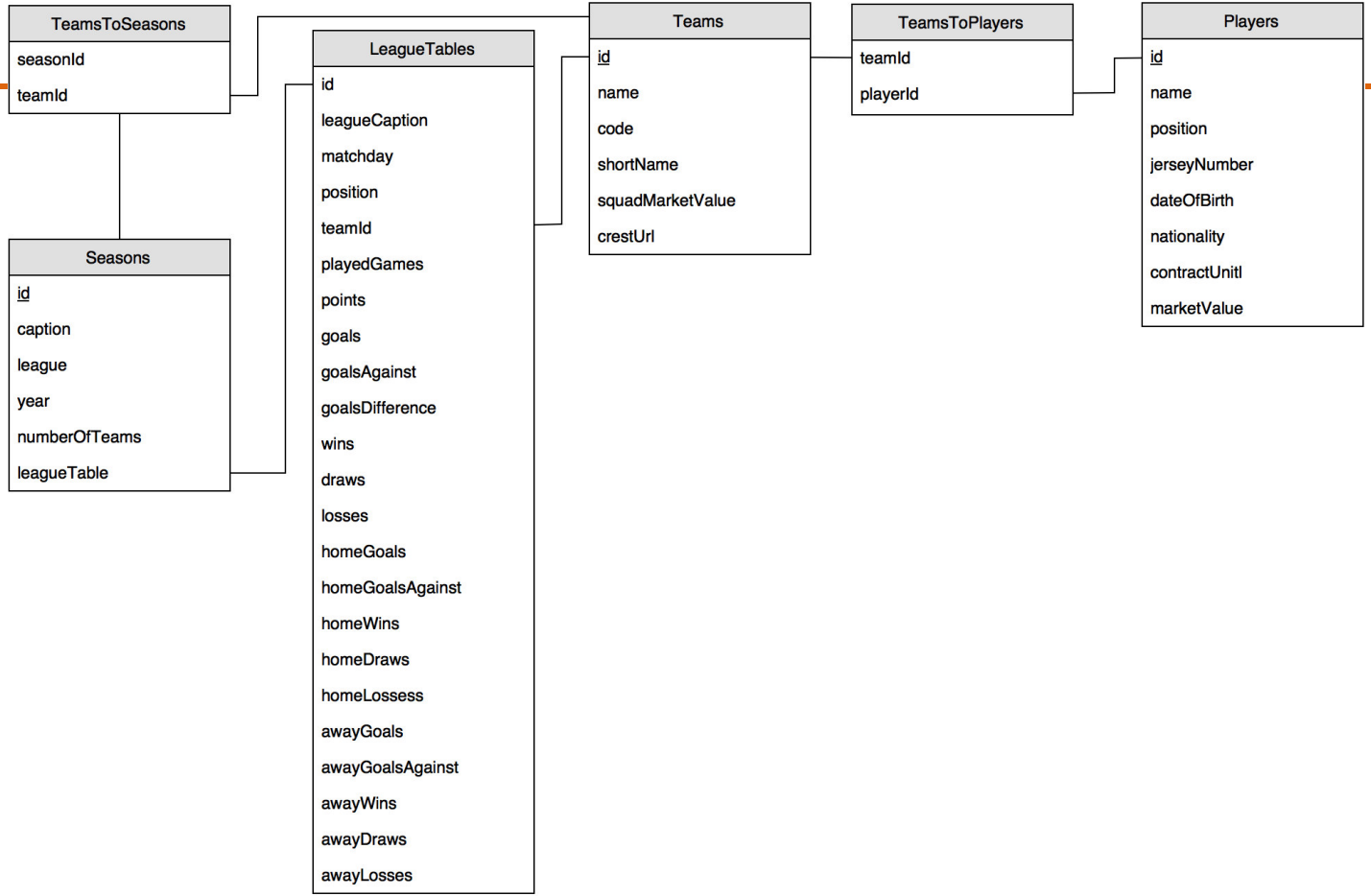
Information Integration
Übung 1

Sebastian Kliem,
Markus Petrykowski,
Sebastian Rehfeldt

football-data.org

- Website: http://api.football-data.org/index
- Alle größeren europäischen Ligen
- Extrahiert:
  - Spieler
  - Teams
  - Saisons
  - Spiel-Tabellen

football-data.org



HPI Hasso Plattner Institut

Übung 1

Sebastian Kliem,
Markus Petrykowski,
Sebastian Rehfeldt

football-data.org

```
CREATE TABLE Seasons(id INTEGER PRIMARY KEY,caption TEXT,league
TEXT,year INTEGER,numberOfTeams INTEGER,lastUpdated TEXT,leagueTable
INTEGER,FOREIGN KEY(leagueTable) REFERENCES LeagueTables(id));

CREATE TABLE Teams(id INTEGER PRIMARY KEY,name TEXT,code
TEXT,shortName TEXT,squadMarketValue TEXT,crestUrl TEXT);

CREATE TABLE Players(id INTEGER PRIMARY Key,name TEXT,position
TEXT,jerseyNumber INTEGER,dateOfBirth TEXT,nationality
TEXT,contractUntil TEXT,marketValue TEXT);

CREATE TABLE TeamsToPlayers(teamId INTEGER,playerId INTEGER,FOREIGN
KEY(teamId) REFERENCES Teams(id), FOREIGN KEY(playerId) REFERENCES
Players(id));
```

football-data.org

```
CREATE TABLE TeamsToSeasons(seasonId INTEGER,teamId INTEGER,FOREIGN
KEY(seasonId) REFERENCES Seasons(id), FOREIGN KEY(teamId) REFERENCES
Teams(id));

CREATE TABLE LeagueTables(id INTEGER,leagueCaption TEXT,matchday
INTEGER,position INTEGER,teamId INTEGER,playedGames INTEGER,points
INTEGER,goals INTEGER,goalsAgainst INTEGER,goalsDifference
INTEGER,wins INTEGER,draws INTEGER,losses INTEGER,homeGoals
INTEGER,homeGoalsAgainst INTEGER,homeWins INTEGER,homeDraws
INTEGER,homeLosses INTEGER,awayGoals INTEGER,awayGoalsAgainst
INTEGER,awayWins INTEGER,awayDraws INTEGER,awayLosses INTEGER,FOREIGN
KEY(teamId) REFERENCES Teams(id));
```

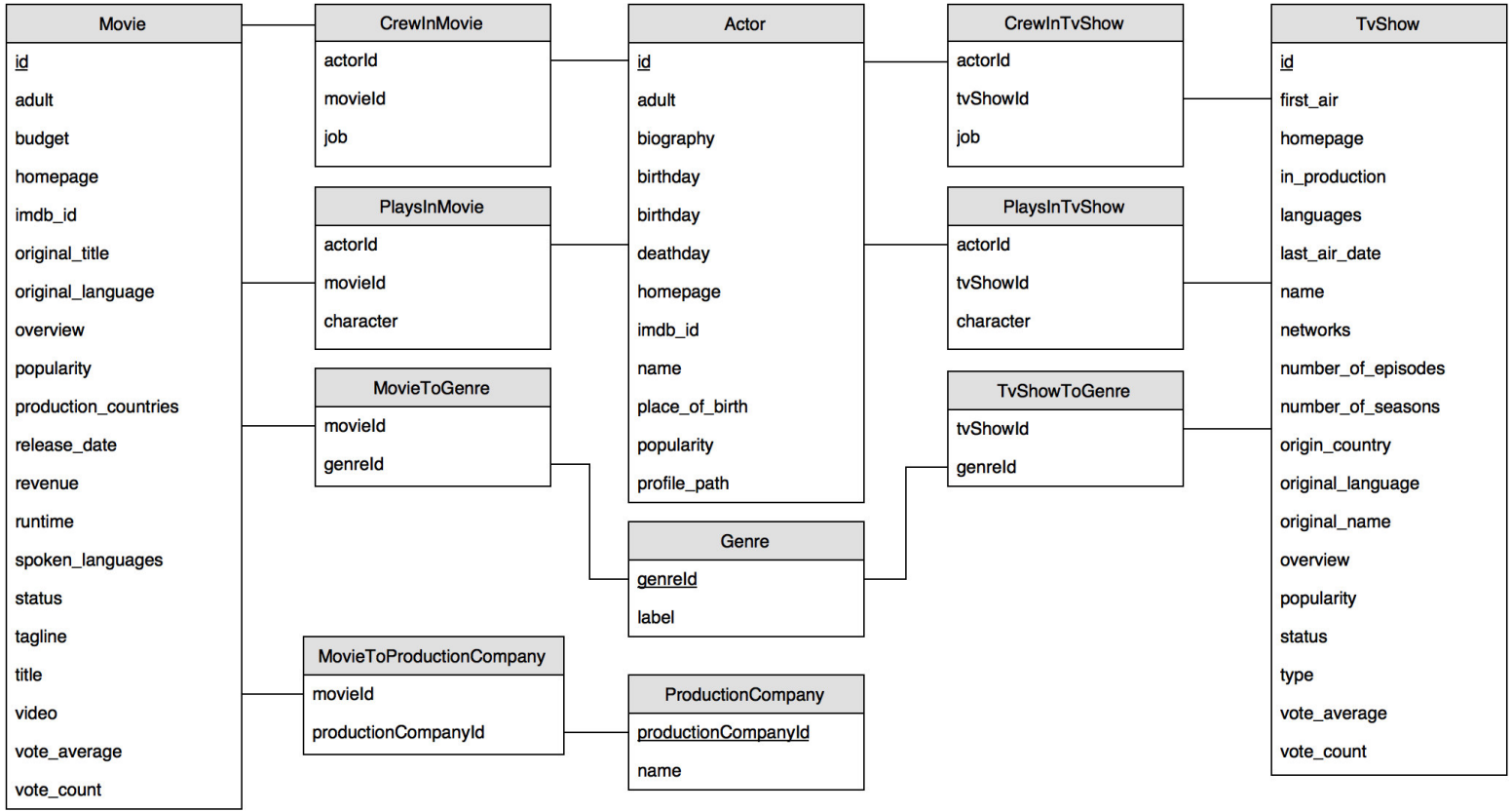
The Movie Database

- Website: https://www.themoviedb.org/
- Freie Filmdatenbank
- Gepflegt durch angemeldete Nutzer
- Daten über Filme, Serien, Personen
- Enthält Links zu IMDb-Entitäten
- Bestand: ca.
  - 256.000 Filme
  - 62.000 Serien
  - 1.090.000 Episoden
  - 644.000 Personen

Stand 05.11.2015:

- 162344 Schauspieler
- 161005 Filme
- 23354 Serien
- 34 Genres
- 33211 Produktionsfirmen
- Film mit meisten Credits: „Popieluszko. Wolnosc jest w nas“ – 183 Schauspieler
- Film mit größter Crew: „The Assassination of Richard Nixon” – 236 Crew-Mitglieder
- Größe der Datenbank: 168 MB

The Movie Database



Übung 1

Sebastian Kliem,
Markus Petrykowski,
Sebastian Rehfeldt

Chart **7**

The Movie Database

```
CREATE TABLE Actor (id INTEGER PRIMARY KEY, adult INTEGER, biography
TEXT, birthday TEXT, deathday TEXT, homepage TEXT, imdb_id TEXT, name
TEXT, place_of_birth TEXT, popularity REAL, profile_path TEXT);

CREATE TABLE PlaysInMovie (actorId INTEGER, movieId INTEGER,
character TEXT);

CREATE TABLE PlaysInTvShow (actorId INTEGER, tvShowId INTEGER,
character TEXT);

CREATE TABLE CrewInTvShow (actorId INTEGER, tvShowId INTEGER, job
TEXT);

CREATE TABLE CrewInMovie (actorId INTEGER, movieId INTEGER, job
TEXT);
```

The Movie Database

```
CREATE TABLE Movie (id INTEGER PRIMARY KEY, adult INTEGER, budget
INTEGER, homepage TEXT, imdb_id TEXT, original_title TEXT,
original_language TEXT, overview TEXT, popularity REAL,
production_countries TEXT, release_date TEXT, revenue INTEGER,
runtime INTEGER, spoken_languages TEXT, status TEXT, tagline TEXT,
title TEXT, video INTEGER, vote_average REAL, vote_count INTEGER);

CREATE TABLE TvShow (id INTEGER PRIMARY KEY, first_air_date TEXT,
homepage TEXT, in_production INTEGER, languages TEXT, last_air_date
TEXT, name TEXT, networks TEXT, number_of_episodes INTEGER,
number_of_seasons INTEGER, origin_country TEXT, original_language
TEXT, original_name TEXT, overview TEXT, popularity REAL, status
TEXT, type TEXT, vote_average REAL, vote_count INTEGER );

CREATE TABLE MovieToGenre (movieId INTEGER, genreId INTEGER);
```

The Movie Database

```
CREATE TABLE TvShowToGenre (tvShowId INTEGER, genreId INTEGER);

CREATE TABLE MovieToProductionCompany (movieId INTEGER,
productionCompanyId INTEGER);

CREATE TABLE Genre (genreId INTEGER PRIMARY KEY, label TEXT);

CREATE TABLE ProductionCompany (productionCompanyId INTEGER PRIMARY
KEY, name TEXT);


ALTER TABLE CrewInMovie ADD FOREIGN KEY (movieId) REFERENCES
Movie(id);

ALTER TABLE CrewInMovie ADD FOREIGN KEY (actorId) REFERENCES
Actor(id);
```

The Movie Database

```
ALTER TABLE PlaysInMovie ADD FOREIGN KEY (movieId) REFERENCES Movie(id);

ALTER TABLE PlaysInMovie ADD FOREIGN KEY (actorId) REFERENCES Actor(id);

ALTER TABLE MovieToGenre ADD FOREIGN KEY (movieId) REFERENCES Movie(id);

ALTER TABLE CrewInMovie ADD FOREIGN KEY (genreId) REFERENCES
Genre(genreId);

ALTER TABLE MovieToProductionCompany ADD FOREIGN KEY (movieId)
REFERENCES Movie(id);

ALTER TABLE MovieToProductionCompany ADD FOREIGN KEY
(productionCompanyId) REFERENCES ProductionCompany(productionCompanyId);

ALTER TABLE CrewInTvShow ADD FOREIGN KEY (tvShowId) REFERENCES
TvShow(id);

ALTER TABLE CrewInTvShow ADD FOREIGN KEY (actorId) REFERENCES Actor(id);
```

The Movie Database

```
ALTER TABLE PlaysInTvShow ADD FOREIGN KEY (tvShowId) REFERENCES
TvShow(id);

ALTER TABLE PlaysTvShow ADD FOREIGN KEY (actorId) REFERENCES
Actor(id);

ALTER TABLE TvShowToGenre ADD FOREIGN KEY (genreId) REFERENCES
Genre(genreId);
```

Information Integration

Data Extraction

Dennis Hempfing, Jaspar Mang, Sebastian Koall, Tobias Stengel

FIFA World Cup Data

- data concerning the FIFA World Cup 2014
- information about all players, the teams they are on and the clubs they are playing in
- source: https://docs.google.com/spreadsheets/d/1i7aUrjBcle7XtEbRUqQrpSLi6cGxIIE1MpJyPpfdg5Y/edit#gid=1425104045
- size: 733 player, 32 teams, 297 clubs

FIFA World Cup Data: ER-Diagram

FIFA World Cup Data: Create Table + Constraints

```
CREATE TABLE spieler
(
  id serial NOT NULL,
  "Nachname" character varying(50),
  "Vorname" character varying(50),
  "Geburtstag" date,
  "Captain" boolean,
  "Position" character varying(10),
  "Anzahl Einsaetze" integer,
  "TeamId" integer,
  "ClubId" integer,
  "Spielernummer" integer,
  CONSTRAINT spieler_pkey PRIMARY KEY (id),
  CONSTRAINT "spieler_ClubId_fkey" FOREIGN KEY ("ClubId")
      REFERENCES club (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "spieler_TeamId_fkey" FOREIGN KEY ("TeamId")
      REFERENCES teamsoccer (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

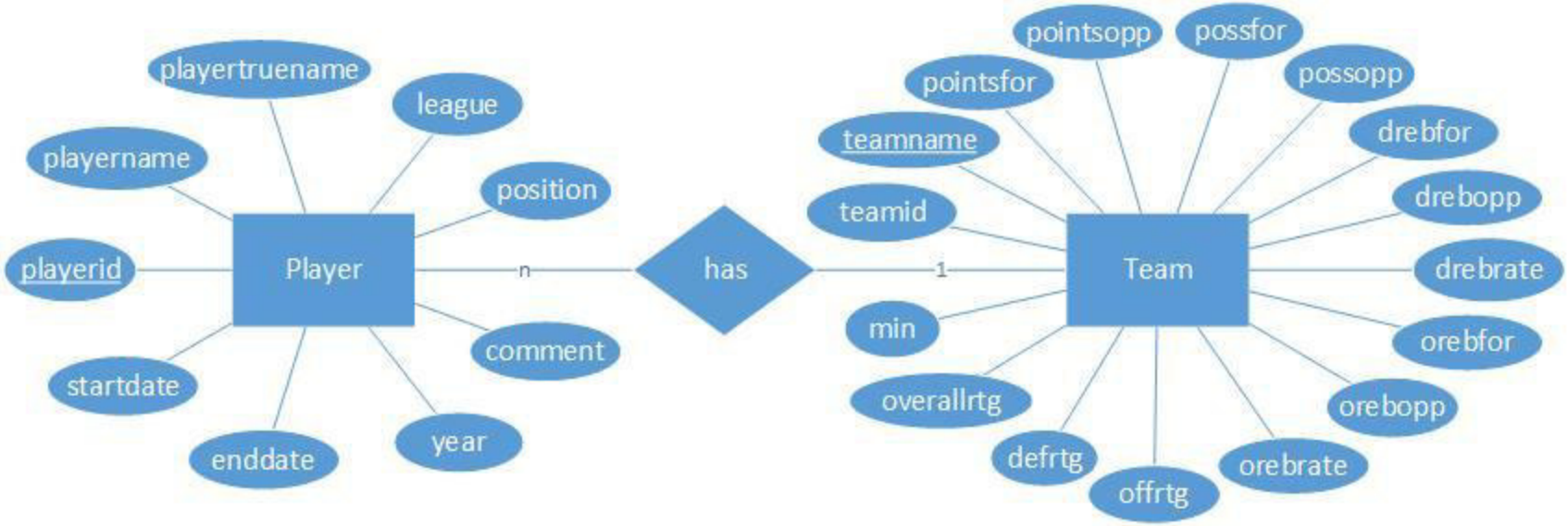
```
CREATE TABLE club
(
  id serial NOT NULL,
  "Name" character varying(50),
  "Liga" character varying(50),
  CONSTRAINT club_pkey PRIMARY KEY (id)
)
```

```
CREATE TABLE teamsoccer
(
  id serial NOT NULL,
  "Name" character varying(50),
  "Gruppe" character varying(20),
  CONSTRAINT teamsoccer_pkey PRIMARY KEY (id)
)
```

NBA Data

- data to perform statistical analysis of the NBA
- information about player and the teams they are on
- season 2011-2012
- source: http://basketballvalue.com/downloads.php
- size: 518 player, 30 teams

NBA Data: ER-Diagram

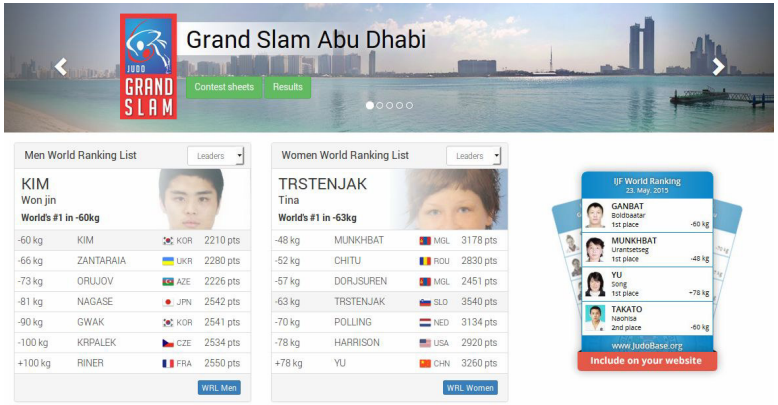


NBA Data: Create Table + Constraints

```
CREATE TABLE team
(
  teamid integer,
  teamname character varying(3) NOT NULL,
  min real,
  possfor integer,
  possopp integer,
  pointsfor integer,
  pointopp integer,
  offrtg real,
  defrtg real,
  overallrtg real,
  orebfor integer,
  orebopp integer,
  drebfor integer,
  drebopp integer,
  orebrate real,
  drebrate real,
  CONSTRAINT team_pkey PRIMARY KEY (teamname)
)
```

```
CREATE TABLE player
(
  playerid integer NOT NULL,
  playername character varying(50),
  playertruename character varying(50),
  league character varying(10),
  teamname character varying(50),
  year character varying(10),
  "position" real,
  comment character varying(20),
  startdate character varying(20),
  enddate character varying(20),
  CONSTRAINT player_pkey PRIMARY KEY (playerid),
  CONSTRAINT player_teamname_fkey FOREIGN KEY (teamname)
        REFERENCES team (teamname) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Judobase und Ergast



Judobase (data.judobase.org/api/get_json)

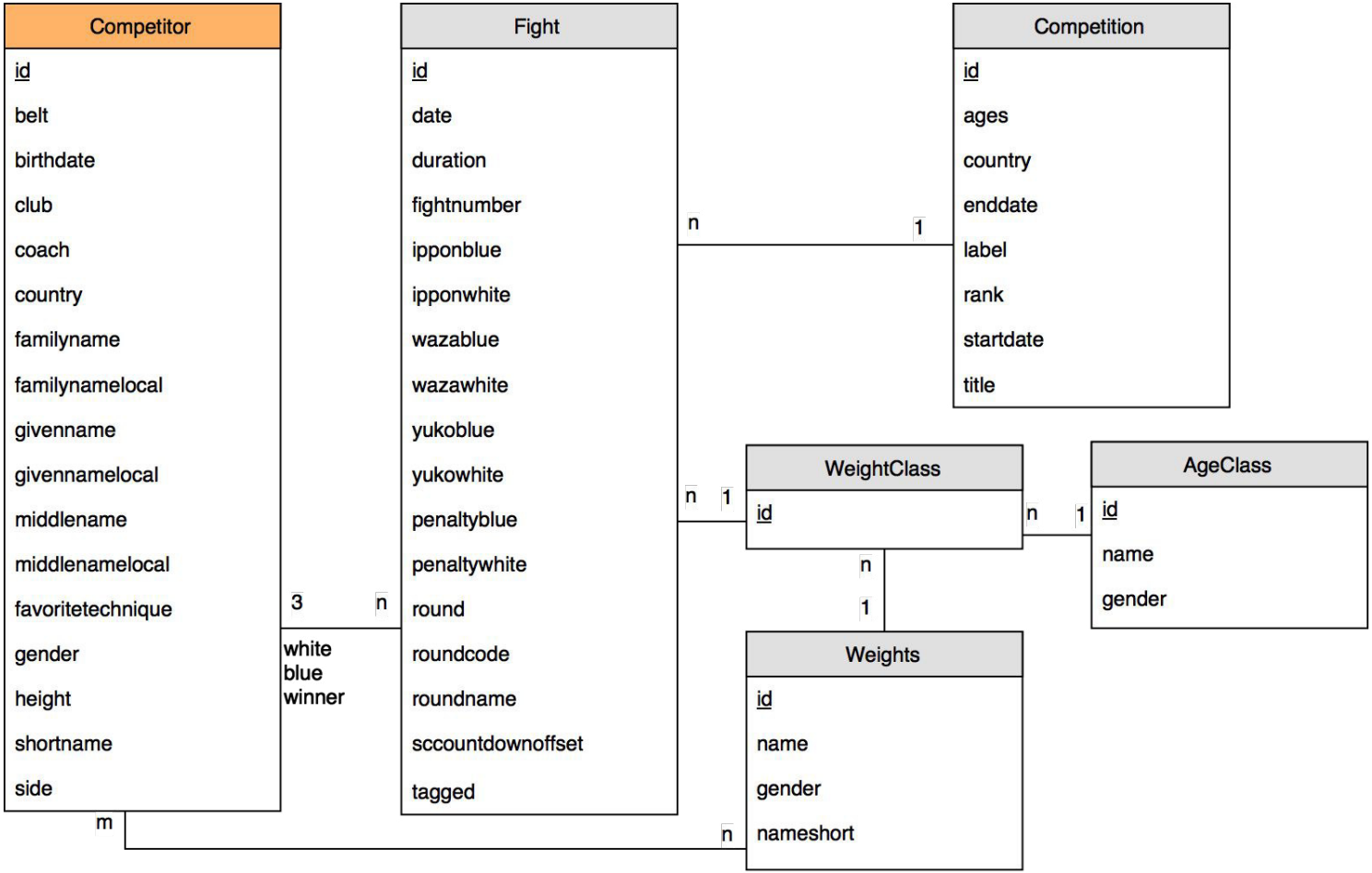
- Datenbank für int. Judowettkämpfe
- Crawlen der API (JSON)
- 28743 Kämpfer

Ergast (ergast.com/api/f1)

- Datenbank für Formel 1
- Datenbank-Dump (MySQL)
- 834 Fahrer
- https://github.com/AnatolyUss/FromMySqlToPostgreSql

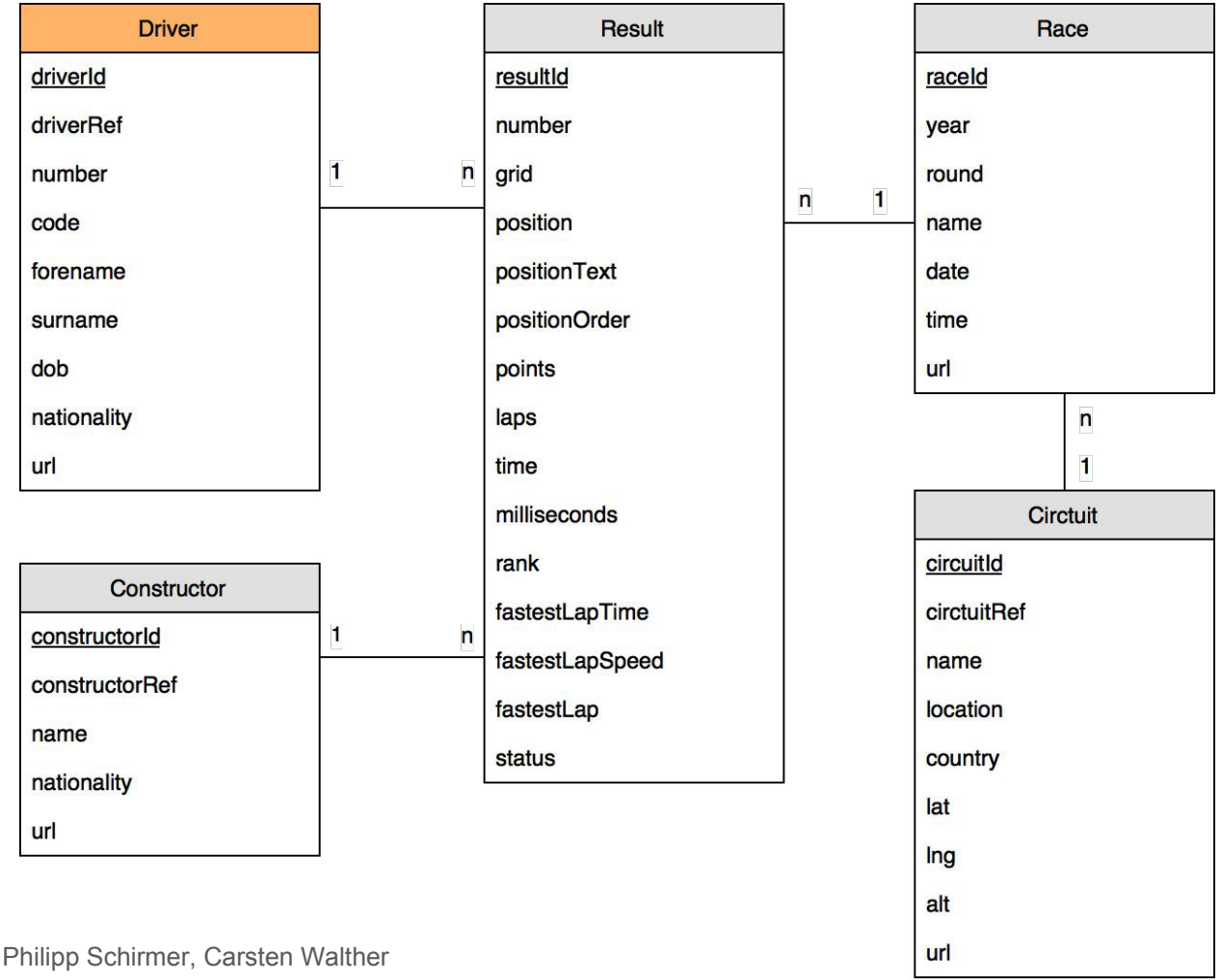
Pascal Lange, Balthasar Martin, Philipp Schirmer, Carsten Walther

Judobase



Pascal Lange, Balthasar Martin, Philipp Schirmer, Carsten Walther

Ergast



Pascal Lange, Balthasar Martin, Philipp Schirmer, Carsten Walther

Judobase

```
CREATE TABLE competitors
(
  id serial PRIMARY KEY,
  belt character varying(255),
  birthdate date,
  club character varying(255),
  coach character varying(255),
  country character varying(255),
  familyname character varying(255),
  familynamelocal character varying(255),
  favoritetechnique character varying(255),
  gender character varying(6),
  givenname character varying(255),
  givennamelocal character varying(255),
  height integer,
  middlename character varying(255),
  middlenamelocal character varying(255),
  shortname character varying(255),
  side character(1)
);
```

```
CREATE TABLE weights
(
  id serial PRIMARY KEY,
  gender character(6),
  name character varying(255),
  nameshort character varying(255)
);
```

```
CREATE TABLE competitorweights
(
  idcompetitor integer REFERENCES competitors (id),
  idweight integer REFERENCES weights (id),
  PRIMARY KEY (idcompetitor, idweight)
);
```

```
CREATE TABLE ageclasses
(
  id serial PRIMARY KEY,
  name character varying(255),
  gender character varying(6)
);
```

```
CREATE TABLE competitions
(
  id serial PRIMARY KEY,
  ages character varying(511),
  country character varying(255),
  enddate date,
  label character varying(255),
  rank character varying(255),
  startdate date,
  title character varying(255)
);
```

```
CREATE TABLE weightclasses
(
  id serial PRIMARY KEY,
  weight integer REFERENCES weights (id),
  age integer REFERENCES ageclasses (id)
);
```

```
CREATE TABLE fights
(
  id serial PRIMARY KEY,
  white integer REFERENCES competitors (id),
  blue integer REFERENCES competitors (id),
  winner integer REFERENCES competitors (id),
  competition integer REFERENCES competitions (id),
  date date,
  duration time without time zone,
  fightnumber integer,
  ipponblue integer,
  ipponwhite integer,
  penaltyblue integer,
  penaltywhite integer,
  round integer,
  roundcode character varying(255),
  roundname character varying(255),
  sccountdownoffset integer,
  tagged integer,
  wazablue integer,
  wazawhite integer,
  weight integer REFERENCES weightclasses(id),
  yukoblue integer,
  yukowhite integer
);
```

Pascal Lange, Balthasar Martin, Philipp Schirmer, Carsten Walther

Ergast

```sql
CREATE TABLE drivers
(
  driverid serial PRIMARY KEY,
  driverref character varying(255) NOT NULL,
  "number" integer,
  code character varying(3),
  forename character varying(255) NOT NULL,
  surname character varying(255) NOT NULL,
  dob date,
  nationality character varying(255),
  url character varying(255) NOT NULL
);
```

```sql
CREATE TABLE constructors
(
  constructorid serial PRIMARY KEY,
  constructorref character varying(255) NOT NULL,
  name character varying(255) NOT NULL,
  nationality character varying(255),
  url character varying(255) NOT NULL
);
```

```sql
CREATE TABLE circuits
(
  circuitid serial PRIMARY KEY,
  circuitref character varying(255) NOT NULL,
  name character varying(255) NOT NULL,
  location character varying(255),
  country character varying(255),
  lat real,
  lng real,
  alt integer,
  url character varying(255) NOT NULL
);
```

```sql
CREATE TABLE results
(
  resultid serial PRIMARY KEY,
  raceid integer NOT NULL REFERENCES races (raceid),
  driverid integer NOT NULL REFERENCES drivers (driverid),
  constructorid integer NOT NULL REFERENCES constructors (constructorid),
  "number" integer NOT NULL,
  grid integer NOT NULL,
  "position" integer,
  positiontext character varying(255) NOT NULL,
  positionorder integer NOT NULL,
  points real NOT NULL,
  laps integer NOT NULL,
  "time" character varying(255),
  milliseconds integer,
  fastestlap integer,
  rank integer,
  fastestlaptime character varying(255),
  fastestlapspeed character varying(255),
  status character varying(255)
);
```

```sql
CREATE TABLE races
(
  raceid serial PRIMARY KEY,
  circuitid integer NOT NULL REFERENCES circuits (circuitid),
  year integer NOT NULL,
  round integer NOT NULL,
  name character varying(255) NOT NULL,
  date date NOT NULL,
  "time" time without time zone,
  url character varying(255)
);
```

Pascal Lange, Balthasar Martin, Philipp Schirmer, Carsten Walther

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Information Integration - Exercise 1

Daniel Neuschäfer-Rube, Jaqueline Pollak, Jakob Reschke, Lars Rückert | 09.11.2015

Abgeordnetenwatch.de

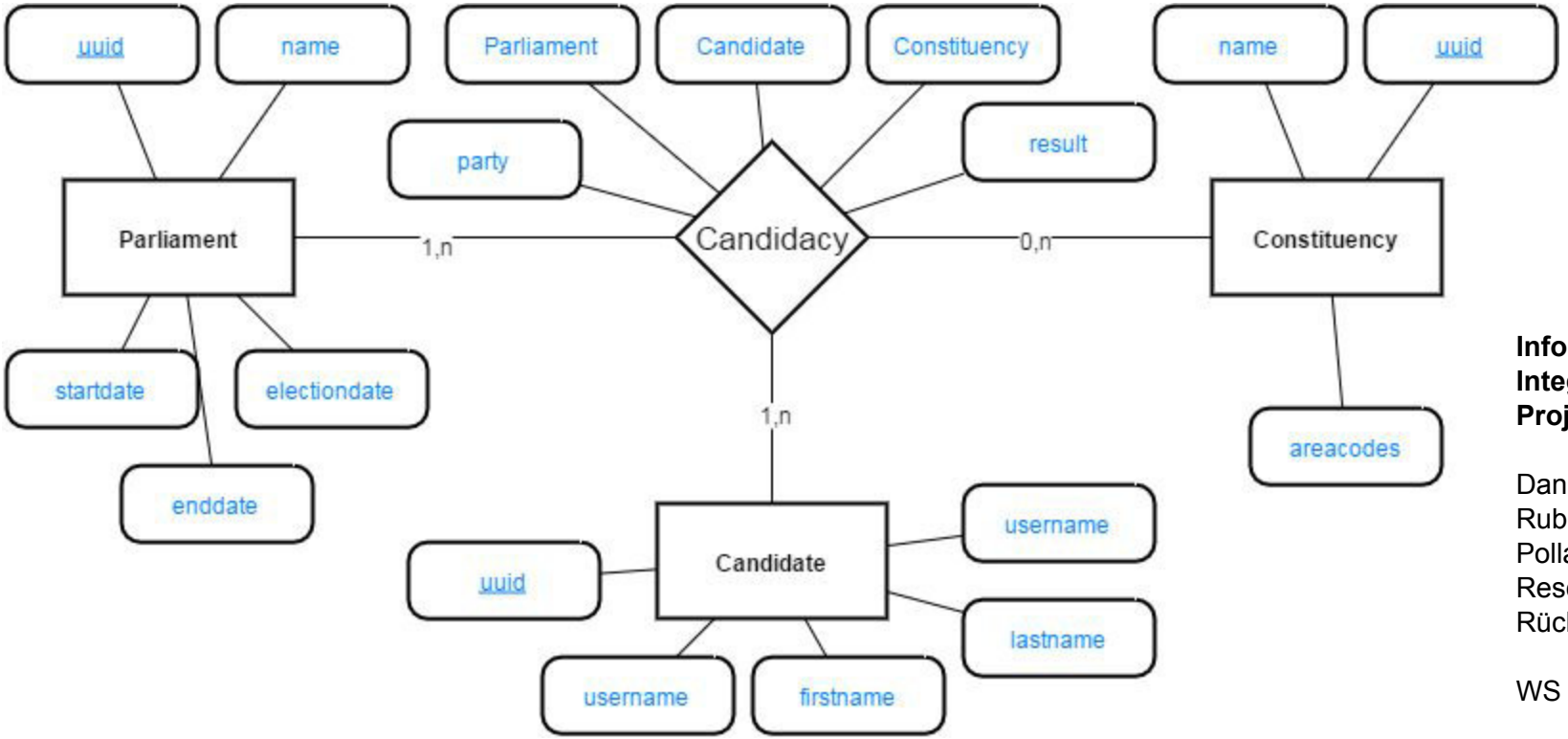
- Datenbank mit Daten über
  - aktuelle Parlamente
  - gewählte Politiker
  - Wahlkreise
- Daten sind unter Open Database License verfügbar
- JSON abgefragt mittels API (Betaphase)

Daten:

- 5144 Kandidaturen
- 3741 Kandidaten
- 2234 Wahlkreise
- 65 Parlamente

Daniel Neuschäfer-Rube, Jaqueline Pollak, Jakob Reschke, Lars Rückert

Abgeordnetenwatch.de



Information Integration Project

Daniel Neuschäfer-Rube, Jaqueline Pollak, Jakob Reschke, Lars Rückert

WS 2015/2016

DBpedia.org

- strukturierte Informationen von Wikipedia
- extrahiert um im Web frei verfügbar zu sein
- gespeichert als RDF-Graphen
- Abgefragt über das SPARQL Interface

SPARQL Explorer for http://dbpedia.org/sparql

SPARQL:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
SELECT DISTINCT?person ?firstname ?lastname ?birthDate
FROM <http://dbpedia.org/>
WHERE {
    ?person rdf:type dbo:Person.
    ?person foaf:givenName ?firstname.
    ?person foaf:surname ?lastname.
    ?person dbo:birthDate ?birthDate.
    ?person dbo:party ?party.
    ?party dbp:nativeName ?partyname.
```

Results: Browse Go! Reset

SPARQL results:

person	firstname	lastname	birthDate
:A._Amirthalingam	"A."@en	"Amirthalingam"@en	"1927-08-26"^^xsd:date
:A._D._Champika_Premadasa	"A. D. Champika"@en	"Premadasa"@en	"1948-11-04"^^xsd:date
:A._H._M._Fowzie	"A. H. M."@en	"Fowzie"@en	"1937-10-13"^^xsd:date
:A._J._M._Muzammil	"A. J. M."@en	"Muzammil"@en	"1949-01-05"^^xsd:date
:A._J._Ranasinghe	"A. J."@en	"Ranasinghe"@en	"1927-01-04"^^xsd:date
:A._M._M._Naushad	"A. M. M."@en	"Naoshaad"@en	"1958"^^xsd:gYear

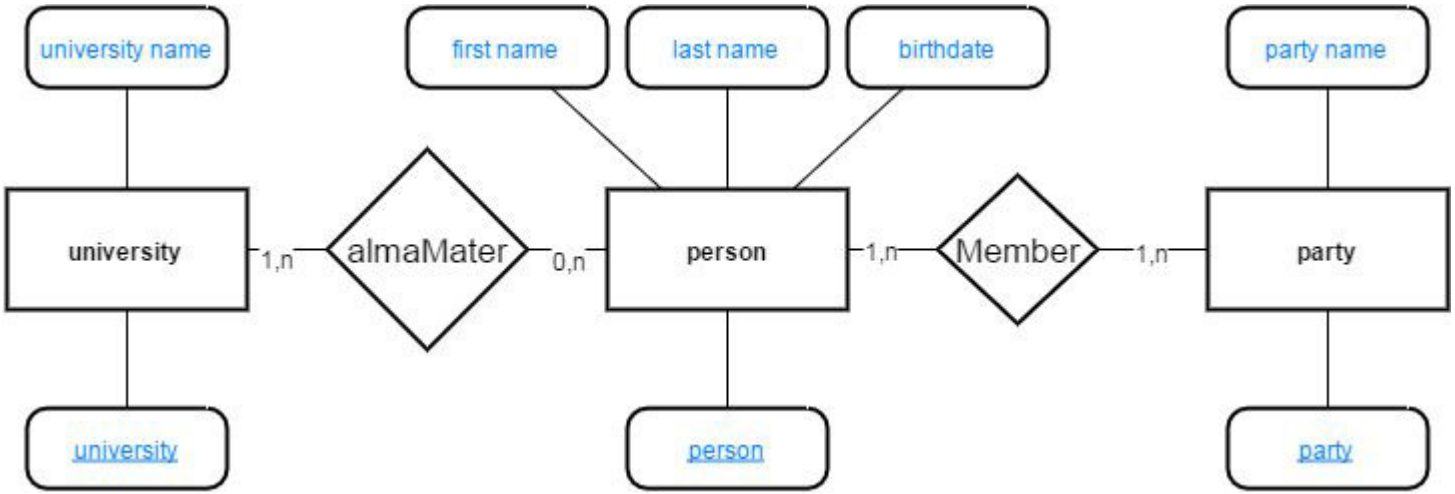
Information Integration Project

Daniel Neuschäfer-Rube, Jaqueline Pollak, Jakob Reschke, Lars Rückert

WS 2015/2016

DBpedia.org

- Fragen Daten über Personen ab, die einer Partei angehören
- ungefähr 4600 Personen abgefragt



Candidacy

```
CREATE TABLE candidacy
(
  parliament uuid NOT NULL references parliament,
  candidate uuid NOT NULL references candidates,
  constituency uuid references constituency,
  party text,
  num integer,
  result real,
  PRIMARY KEY (parliament, candidate)
)
```

Candidate

```
CREATE TABLE candidate
(
  uuid uuid NOT NULL PRIMARY KEY,
  username text,
  firstname text,
  lastname text,
  gender text,
  birthyear integer,
  education text,
  profession text,
  email text,
  twitter text,
  degree text,
  country text,
  county text,
  city text,
  postal_code integer,
  picture_url text
)
```

Constituency

```
CREATE TABLE constituency
(
  uuid uuid NOT NULL PRIMARY KEY,
  parliament uuid references parliament,
  name text,
  areacodes integer[]
)
```

Parliament

```
CREATE TABLE parliament
(
  uuid uuid NOT NULL PRIMARY KEY,
  name text,
  startdate date,
  enddate date,
  electiondate date
)
```

Person

```
CREATE TABLE person
(
  id text NOT NULL PRIMARY KEY,
  firstname text,
  lastname text,
  birthdate date
)
```

Party, University

```
CREATE TABLE party
(
  party text NOT NULL PRIMARY KEY,
  partyname text
)

CREATE TABLE uni
(
  university text NOT NULL PRIMARY KEY,
  universityname text,
)
```


Daniel Neuschäfer-Rube, Jaqueline Pollak, Jakob Reschke, Lars Rückert

Person-Party, Person-University

```
CREATE TABLE person_party
(
  person text references person,
  party text references party,
)

CREATE TABLE person_uni
(
  person text references person,
  university text references uni,
)
```

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

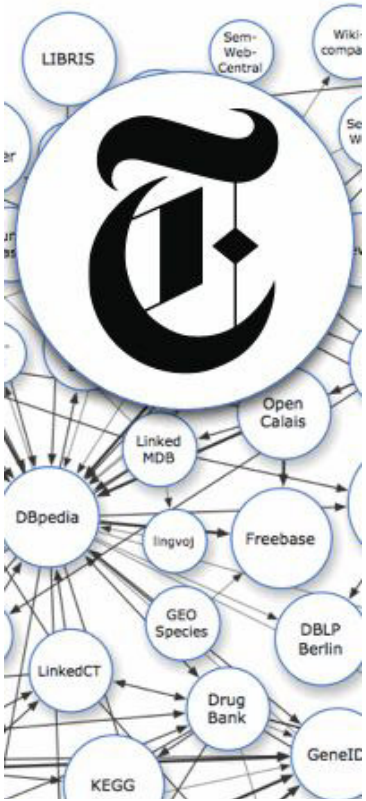
Task 1

Timo Djürken, Fabian Maschler, Mariya Perchyk, Julian Risch

Information Integration Project

WS 2015/2016

New York Times
Linked Open Data



Source:

+ New York Times articles, index for the published news vocabularies with entities:

  people, organizations

  locations and subject descriptors

Size:

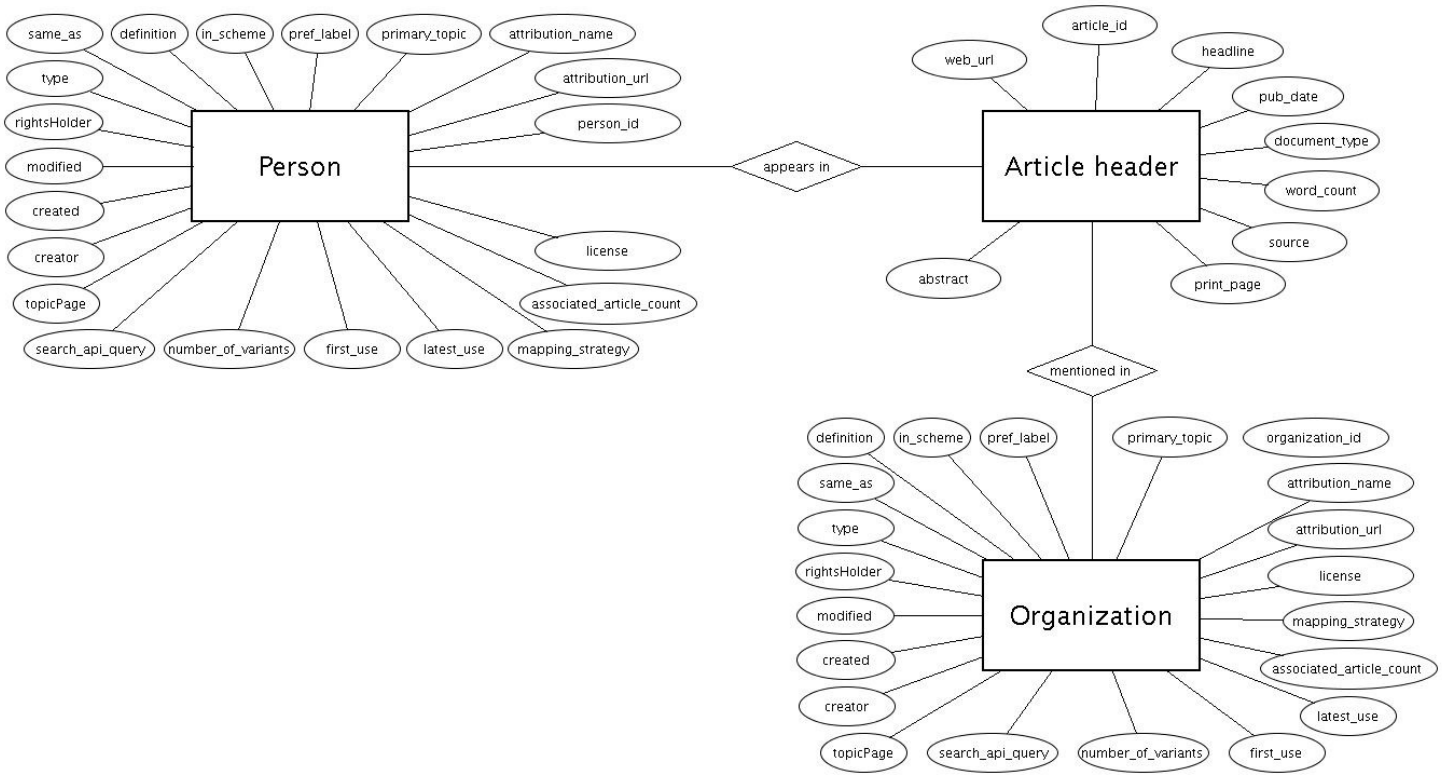
150 MB news articles

23.2 MB

Information Integration Project

WS 2015/2016

New York Times - Linked Open Data
Entity-Relationship Diagram



Information Integration Project

WS 2015/2016

New York Times - Linked Open Data
Create tables

```
CREATE TABLE people (
    person_id varchar(24),
    attribution_name varchar(100),
    attribution_url varchar(100),
    license varchar(100),
    associated_article_count integer,
    first_use date,
    latest_use date,
    mapping_strategy varchar(100),
    number_of_variants integer,
    search_api_query text,
    topicPage text,
    creator varchar(100),
    created date,
    modified date,
    rightsHolder varchar(200),
    type text,
    same_as text,
    definition text,
    in_scheme varchar(255),
    pref_label varchar(255),
    primary_topic varchar(255)
);
```

```
CREATE TABLE organizations (
    organization_id varchar(24),
    attribution_name varchar(100),
    attribution_url varchar(100),
    license varchar(100),
    associated_article_count integer,
    first_use date,
    latest_use date,
    mapping_strategy varchar(100),
    number_of_variants integer,
    search_api_query text,
    topicPage text,
    creator varchar(100),
    created date,
    modified date,
    rightsHolder varchar(200),
    type text,
    same_as text,
    definition text,
    in_scheme varchar(255),
    pref_label varchar(255),
    primary_topic varchar(255)
);
```

```
CREATE TABLE article_header (
    article_id varchar(24),
    headline text,
    pub_date date,
    document_type varchar(20),
    word_count integer,
    source varchar(250),
    print_page varchar(50),
    abstract text,
    web_url varchar(1000)
);
```

```
CREATE TABLE people_ref (
    article_id varchar(24),
    person varchar(100)
);

CREATE TABLE organizations_ref (
    article_id varchar(100),
    organization varchar(100)
);
```

New York Times - Linked Open Data
Understanding the data

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

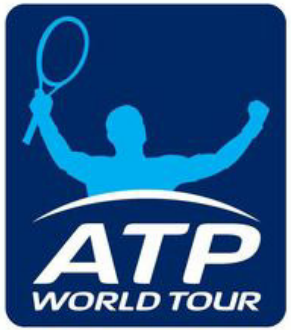
attributionName
The name the creator of a Work would like used when attributing re-use.
attributionURL
The URL the creator of a Work would like used when attributing re-use.
associated_article_count
is used to indicate that The New York Times has written a specified number of articles about the subject resource.
first_use
is used to indicate the first date on which an article in The New York Times was annotated as being about the subject resource.
latest_use
is used to indicate the most recent date on which an article in The New York Times was annotated as being about the subject resource.
mapping_strategy
is used to indicate the strategy used to associate the subject heading with external data resources. There are two strategies: manual and automatic
number_of_variants
is used to indicate that the subject resource is referred by at least the specified number of distinct but related terms in The New York Times.
topic_Page
is used to indicate that The New York Times has a Times Topics page about the subject resource at the URI specified by the object.

type - rdf:type
in_scheme - skos:inScheme
same_as - owl:sameAs
prefLabe - skos:prefLabel
definition - skos:definition
primary_topic - foaf:primaryTopic

New York Times - Linked Open Data
Constraints, Keys Foreign Keys


```
ALTER TABLE people ADD PRIMARY KEY(person_id);

ALTER TABLE organizations ADD PRIMARY KEY(organization_id);

ALTER TABLE article_header ADD PRIMARY KEY(article_id);


ALTER TABLE people_ref ADD  FOREIGN KEY(article_id) REFERENCES article_header(article_id);

ALTER TABLE organizations_ref ADD  FOREIGN KEY(article_id) REFERENCES article_header(article_id);

ALTER TABLE people_ref ADD  FOREIGN KEY(person_id) REFERENCES people(person_id);

ALTER TABLE organizations_ref ADD  FOREIGN KEY(organization_id) REFERENCES organizations(organization_id);
```

ATP Tennis Players and Rankings



Source:

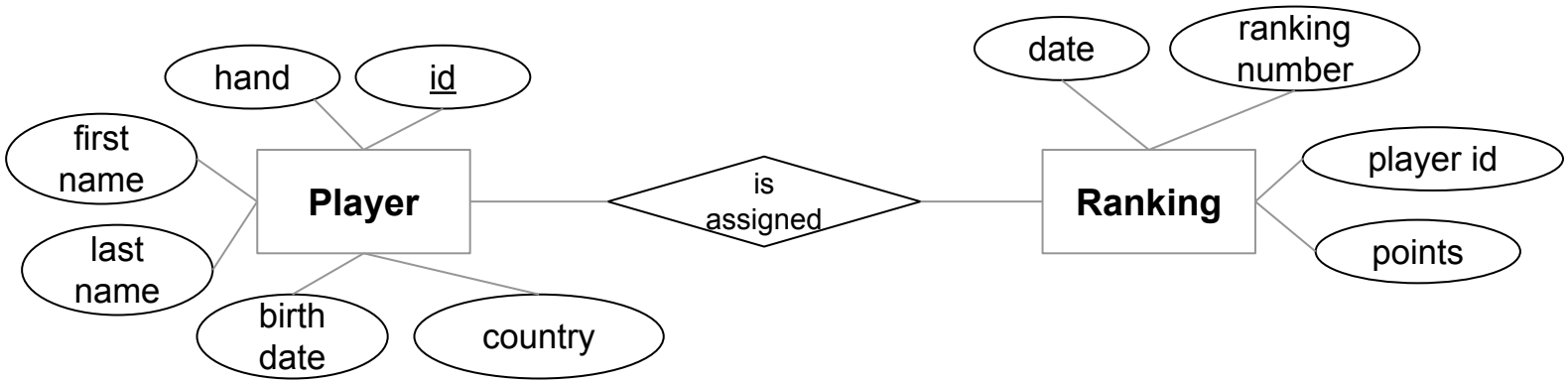
https://github.com/JeffSackmann/tennis_atp

contains player information, rankings, (matches)

Size:

3.11 MB

ATP Tennis Players and Rankings
Entity-Relationship Diagram



ATP Tennis Players and Rankings
Understanding the data

birth
is an 8-digit integer starting with the year, followed by month and then day

hand
is either R or L, indicating left- or right-hander

date
is an 8-digit integer starting with the year, followed by month and then day

country
is the three letter country code according to ISO 3166-1
https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3

pos
ranking position on a certain date

pts
points from the player in this ranking

ATP Tennis Players and Rankings
Create tables

```
CREATE TABLE players (
    id integer,
    firstName varchar(255),
    lastName varchar(255),
    hand char,
    birth integer,
    country varchar(3)
);
```

```
CREATE TABLE rankings (
    id serial PRIMARY KEY,
    date integer,
    pos integer,
    player_id integer,
    pts integer
);
```

no key! (yet?)

ATP Tennis Players and Rankings
Constraints, Keys Foreign Keys

Temporary store duplicates in second table

```
CREATE TABLE player_duplicates (
        id integer,
        firstName varchar(255),
        lastName varchar(255),
        hand char,
        birth integer,
        country varchar(3)
);
INSERT into player_duplicates  (SELECT * from players WHERE id IN
                (SELECT id FROM players GROUP BY id HAVING count(id)>1));
DELETE FROM players WHERE id IN (SELECT id FROM player_duplicates);

ALTER TABLE players ADD PRIMARY KEY(id);
```

ATP Tennis Players and Rankings
Constraints, Keys Foreign Keys

Temporary store unknown FKs in second table

```
CREATE TABLE rankings_violation (
        id serial PRIMARY KEY,
        date integer,
        pos integer,
        player_id integer,
        pts integer
);
INSERT INTO rankings_violation (SELECT * FROM rankings WHERE player_id NOT IN
                (SELECT id FROM players));
DELETE FROM rankings WHERE id IN (SELECT id FROM rankings_violation);



ALTER TABLE rankings ADD FOREIGN KEY (player_id) REFERENCES players (id);
```

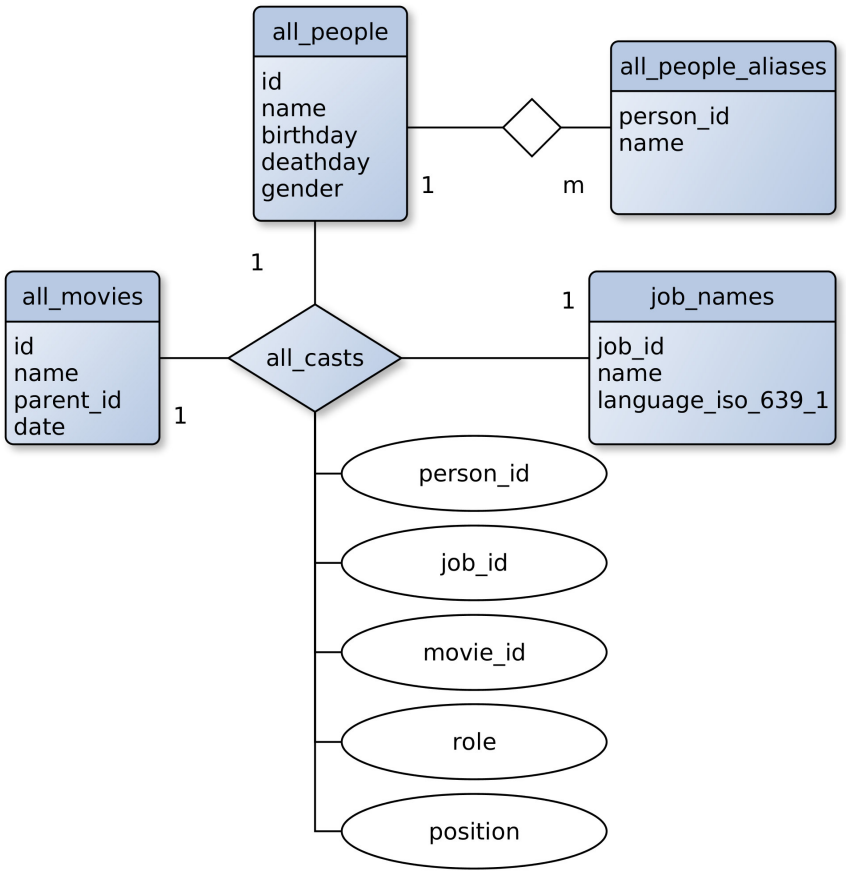
omdb

- http://www.omdb.org/content/Help:DataDownload
- movies, actors, casts
- entities:
  - people
  - aliases
  - jobs
  - movies
  - casts
- 925450 lines in csv files
- 33 MB

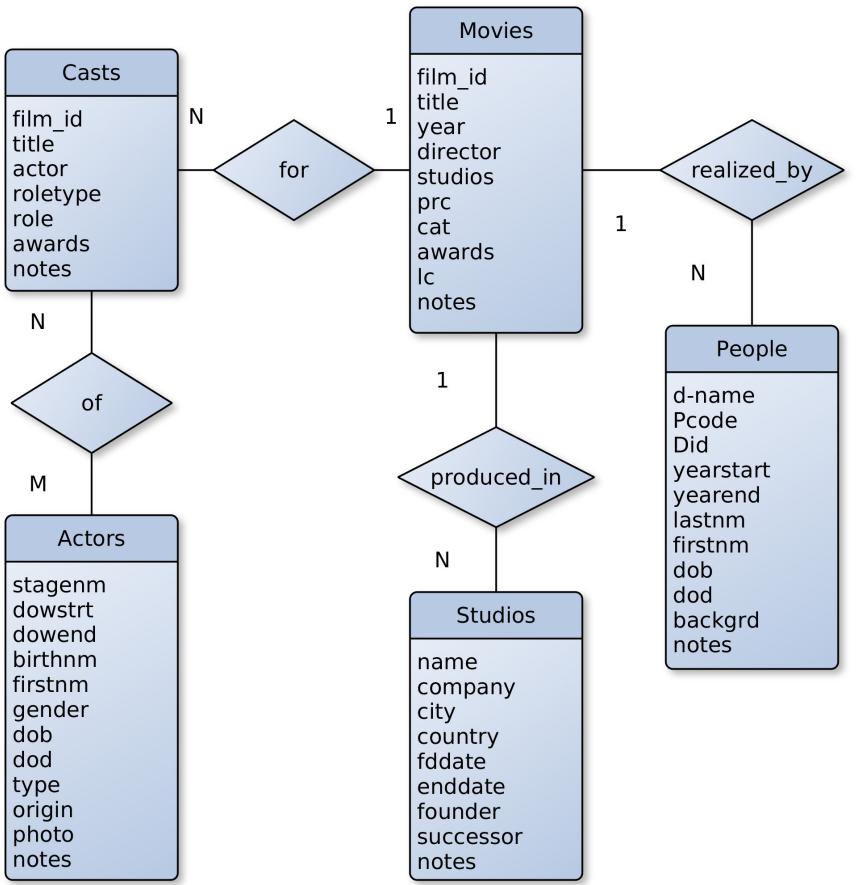
The UCI KDD Archive

- http://kdd.ics.uci.edu/databases/movies/movies.html
- movies, actors, studios
- entities:
  - people
  - actors
  - studios
  - casts
  - main
- 87900 lines in csv files
- 5MB

omdb



The UCI KDD Archive



Keßler, Kimmig, Kroschk, Stamm

omdb

```
create table all_movies (
  id varchar(12) not null PRIMARY KEY,
  name varchar(100) not null,
  parent_id varchar(35),
  date varchar(35));
COPY all_movies FROM '/path_to_data/omdb/all_movies.csv' HEADER DELIMITER ',' CSV;

create table job_names (
  job_id varchar(12) not null,
  name varchar(64) not null,
  language_iso_639_1 varchar(4) not null);
COPY job_names FROM '/path_to_data/omdb/job_names.csv' HEADER DELIMITER ',' CSV;
Delete from job_names where language_iso_639_1 != 'en';
alter table job_names add primary key (job_id);

create table all_people (
  id varchar(12) not null primary key,
  name varchar(64) not null,
  birthday varchar(12),
  deathday varchar(12),
  gender varchar(4));
COPY all_people FROM '/path_to_data/omdb/all_people.csv' HEADER DELIMITER ',' CSV;

create table all_people_aliases (
  person_id varchar(12) not null,
  name varchar(736) not null);
COPY all_people_aliases FROM '/path_to_data/omdb/all_people_aliases.csv' HEADER DELIMITER ',' CSV;
Delete from all_people_aliases where person_id IN (
SELECT person_id FROM all_people_aliases LEFT JOIN all_people on all_people_aliases.person_id=all_people.id WHERE all_people.id IS NULL);
ALTER TABLE all_people_aliases ADD CONSTRAINT person_id_fk FOREIGN KEY (person_id) REFERENCES all_people(id);

create table all_casts (
  movie_id varchar(12) not null,
  person_id varchar(12) not null references all_people(id),
  job_id varchar(12) not null references job_names(job_id),
  role varchar(256),
  position varchar(35));
COPY all_casts FROM '/path_to_data/omdb/all_casts.csv' HEADER DELIMITER ',' CSV;
Delete from all_casts where movie_id IN (
SELECT movie_id FROM all_casts LEFT JOIN all_movies on all_casts.movie_id=all_movies.id WHERE all_movies.id IS NULL);
ALTER TABLE all_casts ADD CONSTRAINT movie_id_fk FOREIGN KEY (movie_id) REFERENCES all_movies(id);
```

Keßler, Kimmig, Kroschk, Stamm

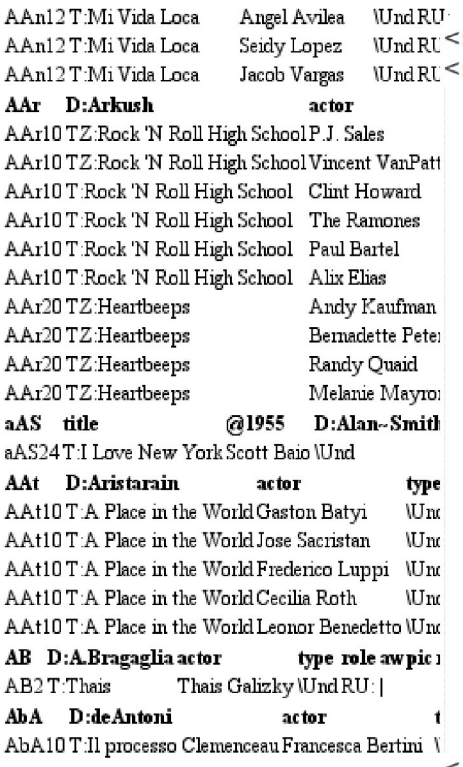
The UCI KDD Archive

```
create table actors (
 stagenm varchar(128),dow varchar(35),birthnm varchar(64),firstnm varchar(64),gender varchar(8),
 dob varchar(36),dod varchar(36),type2 varchar(128),origin varchar(128),photo Text,notes Text
  );

COPY actors FROM '/path_to_data/movies19/csv/actors.csv' DELIMITER ',' CSV;

create table studios (
 name varchar(128),company varchar(64),city varchar(64),country varchar(64),fddate varchar(24),
 fdend varchar(24),founder varchar(128),successor varchar(128),notes varchar(128)
 );
COPY studios FROM '/home/axel/Dokumente/uni/ii-ws2015/movies19/csv/studios.csv' DELIMITER ',' CSV;

create table people (
dname varchar(64) not null,Pcode varchar(35),Did varchar(35),yearstart varchar(35),yearend varchar(35),
lastnm varchar(35),firstnm varchar(35),dob varchar(35),dod varchar(35),notes Text
  );

COPY people FROM '/path_to_data/movies19/csv/people.csv' DELIMITER ',' CSV;

create table movies (
film_id varchar(128),title varchar(128),fyear varchar(35),director varchar(64),prd varchar(128),
studios varchar(64),prc varchar(35),cat varchar(35),awards varchar(64),lc varchar(128),notes text
);
COPY movies FROM '/path_to_data/movies19/csv/movies.csv' DELIMITER ',' CSV;

create table casts (
film_id varchar(64), director varchar(64),title varchar(128),actor varchar(64),
roletype varchar(64),role varchar(128),awards varchar(64),notes text
);

COPY casts FROM '/path_to_data/movies19/csv/casts.csv' DELIMITER ',' CSV;
```

mehrere HTML-Tabellen

JQuery

eigene, saubere HTML-Tabelle



AAn12	T:Mi Vida Loca	Angel Avilea	\Und	RU:	
AAn12	T:Mi Vida Loca	Seidy Lopez	\Und	RU:	
AAn12	T:Mi Vida Loca	Jacob Vargas	\Und	RU:	
AAr	**D:Arkush**		**actor**		
AAr10	T:Z:Rock 'N Roll High School	P.J. Sales			
AAr10	T:Z:Rock 'N Roll High School	Vincent VanPatt			
AAr10	T:Rock 'N Roll High School	Clint Howard			
AAr10	T:Rock 'N Roll High School	The Ramones			
AAr10	T:Rock 'N Roll High School	Paul Bartel			
AAr10	T:Rock 'N Roll High School	Alix Elias			
AAr20	T:Z:Heartbeeps	Andy Kaufman			
AAr20	T:Z:Heartbeeps	Bernadette Pete			
AAr20	T:Z:Heartbeeps	Randy Quaid			
AAr20	T:Z:Heartbeeps	Melanie Mayro			
aAS	**title**	**@1955**	**D:Alan~Smith**		
aAS24	T:I Love New York	Scott Baio	\Und		
AAt	**D:Aristarain**		**actor**		**type**
AAt10	T:A Place in the World	Gaston Batyi	\Un		
AAt10	T:A Place in the World	Jose Sacristan	\Un		
AAt10	T:A Place in the World	Frederico Luppi	\Un		
AAt10	T:A Place in the World	Cecilia Roth	\Un		
AAt10	T:A Place in the World	Leonor Benedetto	\Un		
AB	**D:A.Bragaglia**	**actor**	**type**	**role**	**awpic**
AB2	T:Thais	Thais Galizky	\Und	RU: |	
AbA	**D:deAntoni**		**actor**		
AbA10	T:Il processo Clemenceau	Francesca Bertini	\		

```
<script src="https://code.jquery.com/jquery-1.11.3.min.
<script type=text/javascript>
  $(document).ready(function () {
    var newTable = $('<table></table>');
    $('table').each(function (tableIndex, tableElement)
      var tableBody = tableElement.children[0];
      if (!tableBody) return;
      var rows = tableBody.children;
      if (rows.length == 0) return;
      if (rows[0].children.length < 2) return;
      $.makeArray(rows).splice(1).forEach(function (row
        var cells = row.children;
        if (!cells) return;
        if (cells.length < 2) return;
        var newRow = $('<tr></tr>');
        for (var i = 0; i < cells.length; i++) {
          newRow.append('<td>' + cells[i].textContent +
        }
        newTable.append(newRow);
      });
    });
    $('body').append($('<h1>D Gandse Tabell</h1>'));
    $('body').append(newTable);
  })
</script>
```

RuM37	D:Russ~Meyer	T:Beyond the Valley of the Dolls	Cynthia Meyers	\Und	RU:
RuM37	D:Russ~Meyer	T:Beyond the Valley of the Dolls	Marcia McBroom	\Und	RU:
RuM37	D:Russ~Meyer	T:Beyond the Valley of the Dolls	John LaZar	\Und	RU:
RuM37	D:Russ~Meyer	T:Beyond the Valley of the Dolls	Michael Blodgett	\Und	RU:
RuM37	D:Russ~Meyer	T:Beyond the Valley of the Dolls	Edy Williams	\Und	RU:
RuM40	D:Russ~Meyer	T:The Seven Minutes	Wayne Maunder	\Und	RZ:defending lawyer
RuM40	D:Russ~Meyer	T:The Seven Minutes	Marianne McAndrew	\Und	R:victim
RuM40	D:Russ~Meyer	T:The Seven Minutes	Yvonne deCarlo	\Und	R:actress, author
RuM40	D:Russ~Meyer	T:The Seven Minutes	Philip Carey	\Und	R:aggressive district attorney
RuM40	D:Russ~Meyer	T:The Seven Minutes	Jay C. Flippen	\Sus	RZ:suspect
RuM40	D:Russ~Meyer	T:The Seven Minutes	Edy Williams	\Und	R:friend
RuM40	D:Russ~Meyer	T:The Seven Minutes	John Carradine	\Und	RU:
RUn10	D:Underwood	T:City Slickers	Billy Crystal	\Und	R:has mid-life crisis "Mitch"
RUn10	D:Underwood	T:City Slickers	Patricia Wettig	\Und	R:fed-up wife
RUn10	D:Underwood	T:City Slickers	Bruno Kirby	\Und	R:daredevil buddy
RUn10	D:Underwood	T:City Slickers	Daniel Stern	\Und	R:buddy
RUn10	D:Underwood	T:City Slickers	Jack Palance	\Und	R:trail boss "Curly"