Databases for data analytics

https://github.com/evidencebp/databases-course/

Data representation
and
Data Definition Language (DDL)

How tables are created?

- Some from others
  - create table MyMovies as select * from Movies;
- Some by word

```
IJS: CREATE TABLE `movies` (
  `id` int(11) NOT NULL DEFAULT 0,
  `name` varchar(100) DEFAULT NULL,
  `year` int(11) DEFAULT NULL,
  `rank` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `movies_name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_general_ci;
```

Definitions are great, here is another one

```
DROP TABLE IF EXISTS `movies`;

MovieLens: CREATE TABLE `movies` (

  `movieid` mediumint(8) unsigned NOT NULL DEFAULT 0,

  `year` int(11) NOT NULL,

  `isEnglish` enum('T','F') NOT NULL,

  `country` varchar(50) NOT NULL,

  `runningtime` int(11) NOT NULL,

  PRIMARY KEY (`movieid`),

  KEY `year` (`year`),

  KEY `isEnglish` (`isEnglish`),

  KEY `country` (`country`),

  KEY `runningtime` (`runningtime`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;
```

And another one

Full:

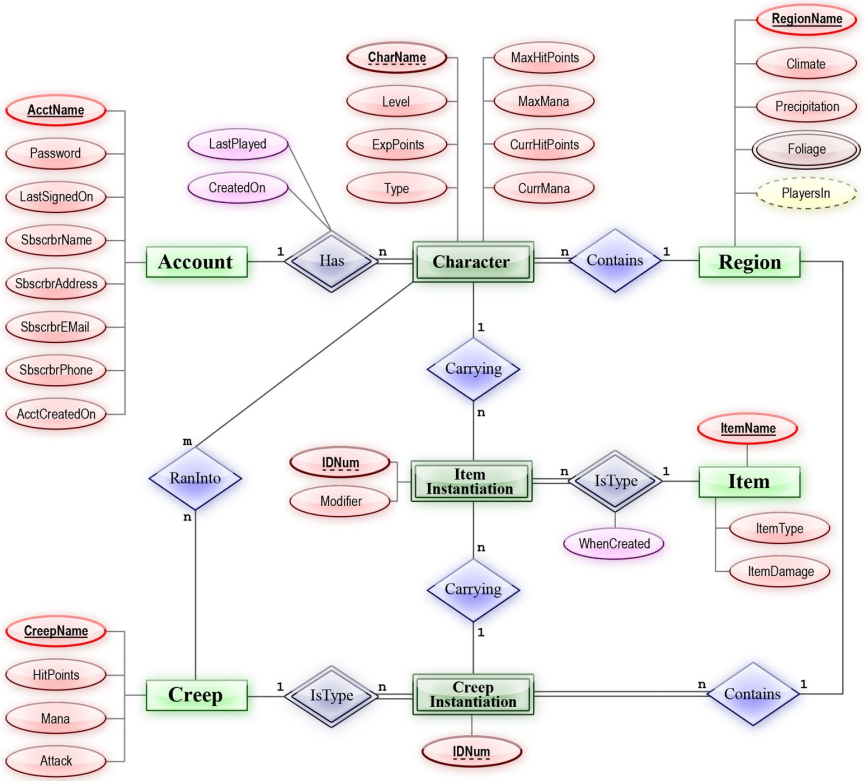
```
CREATE TABLE `movies` (
  `movieid` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(400) NOT NULL,
  `year` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`movieid`),
  KEY `title` (`title`(15)),
  FULLTEXT KEY `movies_title_fulltext` (`title`)
) ENGINE=InnoDB AUTO_INCREMENT=2593314 DEFAULT CHARSET=utf8mb3
COLLATE=utf8mb3_general_ci;
```

So, how should we represent our data?

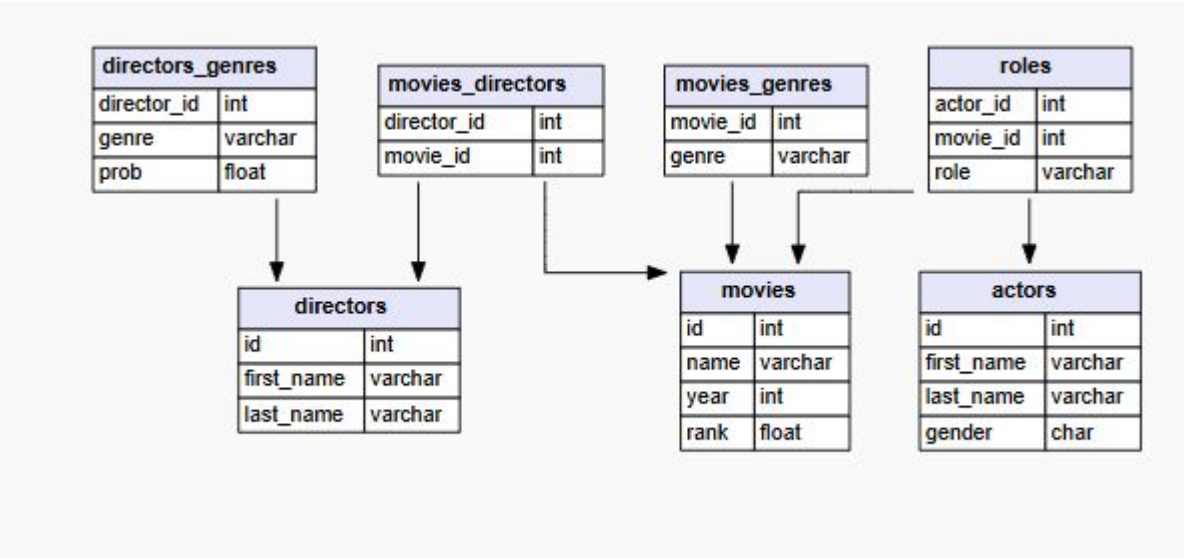
- A need should always come first
  - What are the goals of the representation?
  - What is not needed to be represented?
  - What is not needed now but likely to be needed?

Entity–relationship model

- The data contains entities
- Entities have properties
- Entities have relations among them
- *The Entity Relationship Model – Towards a Unifie*



Representation in tables



Cardinality

- Entities might have 1 to 1, 1 to many, and many to many relations
- 1 to 1 relationtionsip
  - For example citizen id and employee id, person and birth certificate
- 1 to many
  - For example, person and children, customer and orders, city and citizens
- Many to many
    - For example, students and courses, customers and programs
- The direction depends on the side one chooses for the entities - order and customers are many to one
- Relations does not have to be binary and might involve more entities (e.g., A Physician prescribed a Medication to a Patient at a Date)

Relationship representation in a database

- In 1 to 1 relations, the “enhancing” entity can be stored in the main entity table
- For example, a citizen id column in employees table (whose PK is employee id)
- In case of 1 to many, the many key should be unique in the relations table
- For example, cities table and a city_id column in customers table, customer_id column in orders table. The key column in the many side should have a foreign key constraint on the PK of its entity table.
- In many to many relation, we build a new relationship table that will have the keys of both entities. For example, customer_categories table will have both customer_id and category_id. Each key will have a foreign key constraint on the PK of its entity table.

Terms

- Superkey - unique identifier
- Candidate key - minimal unique identifier
- Functional dependency - $X \rightarrow Y$ if each X value is associated with precisely one Y value. Usually X is a key and Y are properties of the entity identified by X.

Guidelines

- Each entity should be identified by a **minimal** primary key
  - A “natural” key is prefered
  - A single key is preferred
- Bill Kent: "[every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key".
  - Extra credit - provide a direct fact on the key, without transitive functionality
- All properties of the entity should be stored in a single table
- Only properties of the entity should be in its table
- Avoid redundancy - they might lead to discrepancies
- Break complex data into simple part (e.g., person’s name, address)

Normalization

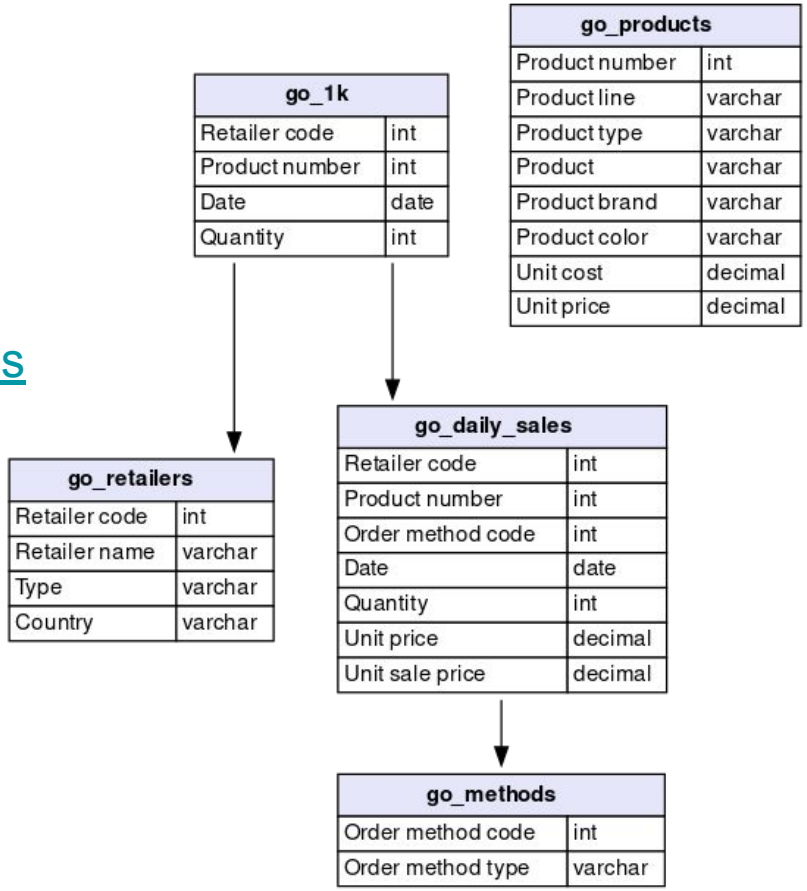
Constraint (informal description in parentheses)	UNF (1970)	1NF (1970)	2NF (1971)	3NF (1971)	EKNF (1982)	BCNF (1974)	4NF (1977)	ETNF (2012)	5NF (1979)	DKNF (1981)	6NF (2003)
Unique rows (no duplicate records)[4]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Scalar columns (columns cannot contain relations or composite values)[5]	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-prime attribute has a full functional dependency on each candidate key (attributes depend on the *whole* of every key)[5]	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with a prime attribute (attributes depend *only* on candidate keys)[5]	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with an elementary prime attribute (a stricter form of 3NF)	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	—
Every non-trivial functional dependency begins with a superkey (a stricter form of 3NF)	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	—

Normalization process

- First normal form
- Second normal form
- Third normal form
- General rule - given a table with a violation, fix it.

Exercise - Sales database

- https://relational.fel.cvut.cz/dataset/GOSales
- Makes sense but what about?
  - 1+1 sales?
  - Different branches?
  - VAT?
  - Payment in different currencies?



Exercise 1

- Each student should pair twice with others
- One of the student should be a customer and the other is the designer
- The customer should provide a use case for sales databases, requirements, and their justification.
- The designer should build a datasat and explain how it fits the requirements
- Each student should be with different people and in different role in each pair

Exercise 2

- A database system for basketball games present per game
  - The teams
  - The winning team and the score
  - The highest scorer per team
- Build a normalized database the can present these results
- Write a query on that database that present the results