IMDb (Part 1)
Relational Model, ER Modelling

Big Data Engineering
(formerly Informationssysteme)

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

April 29, 2023

IMDb (Part 1)

Planned structure for each two-week lecture:

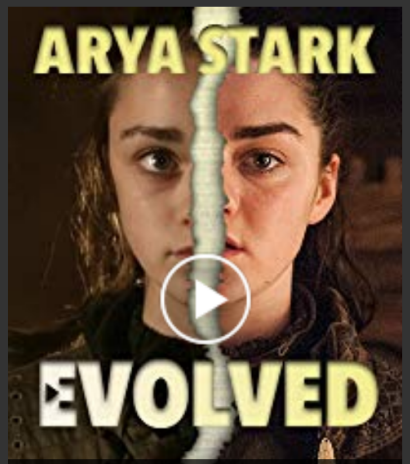
1. Concrete application: IMDb
2. What are the data management and analysis issues behind this?
3. Basics to be able to solve these problems
   (a) Slides
   (b) Jupyter/Python/SQL Hands-on
4. Transfer of the basics to the concrete application

IMDb (Part 1)

1. Concrete application: IMDb

- IMDb: Internet Movie Database
- https://www.IMDb.com
- data on film and television productions
- since 1990
- as of March 2023: more than 14 million titles, see https://www.imdb.com/pressroom/stats/
- short demo

Shazam! Soars to the Top
Weekend Box Office

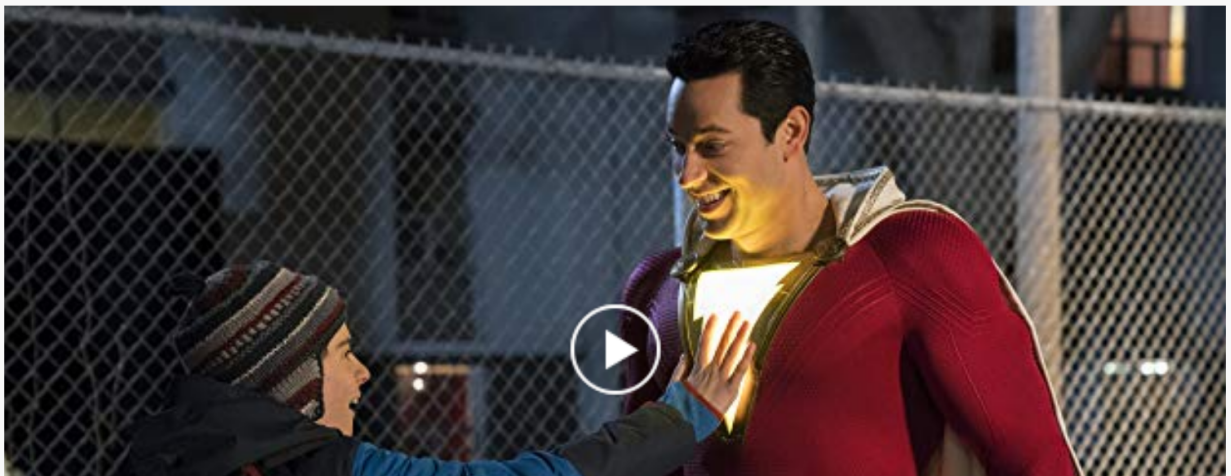


The Evolution of Arya Stark
From Misfit to Assassin

From 'Holby City' to 'Killing Eve'
The Rise of Jodie Comer

Browse trailers »

Zachary Levi Reveals His 'Shazam!' Suit Struggles



Opening This Week

- Hellboy - Call of Darkness
- Mister Link - Ein fellig verrücktes Abenteuer
- After Passion
- Little
- High Life
- Les filles du soleil
- Sauvage — Opens Apr. 10

See more opening this week »

Get Showtimes »

Now Playing (Box Office)

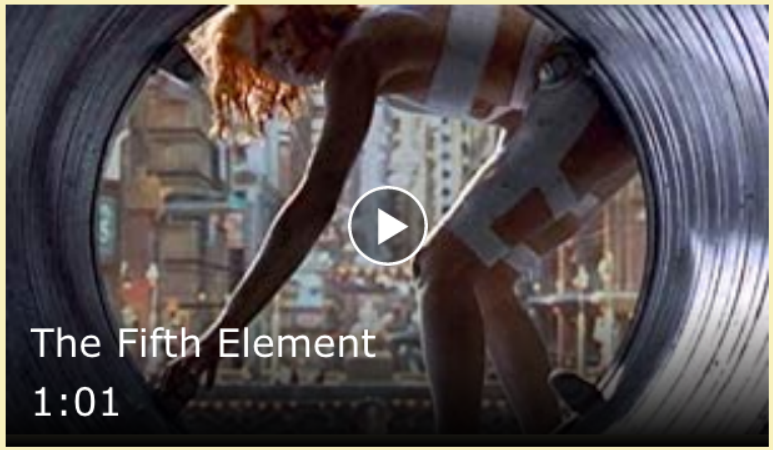
Shazam! $53.5M	Showtimes
Friedhof der Kuscheltiere $25.0M	Showtimes
Dumbo $18.2M	Showtimes

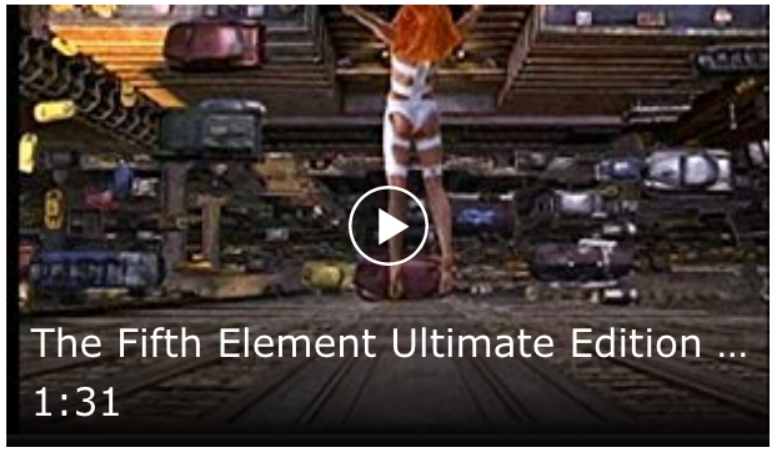
IMDb

the fith element

All

The Fifth Element (1997)
Bruce Willis, Milla Jovovich





The Fifth Element (1998)
Action, Adventure, Sci-Fi

Elementary: Season 5 - The Fifth Elementary (2017)
Nelsan Ellis, Jon Michael Hill

The Fifth Element (2002)

Beatboxing: The Fifth Element of Hip Hop (2011)

Shazam! Soars
Top
Weekend Box Office

Browse trailers »

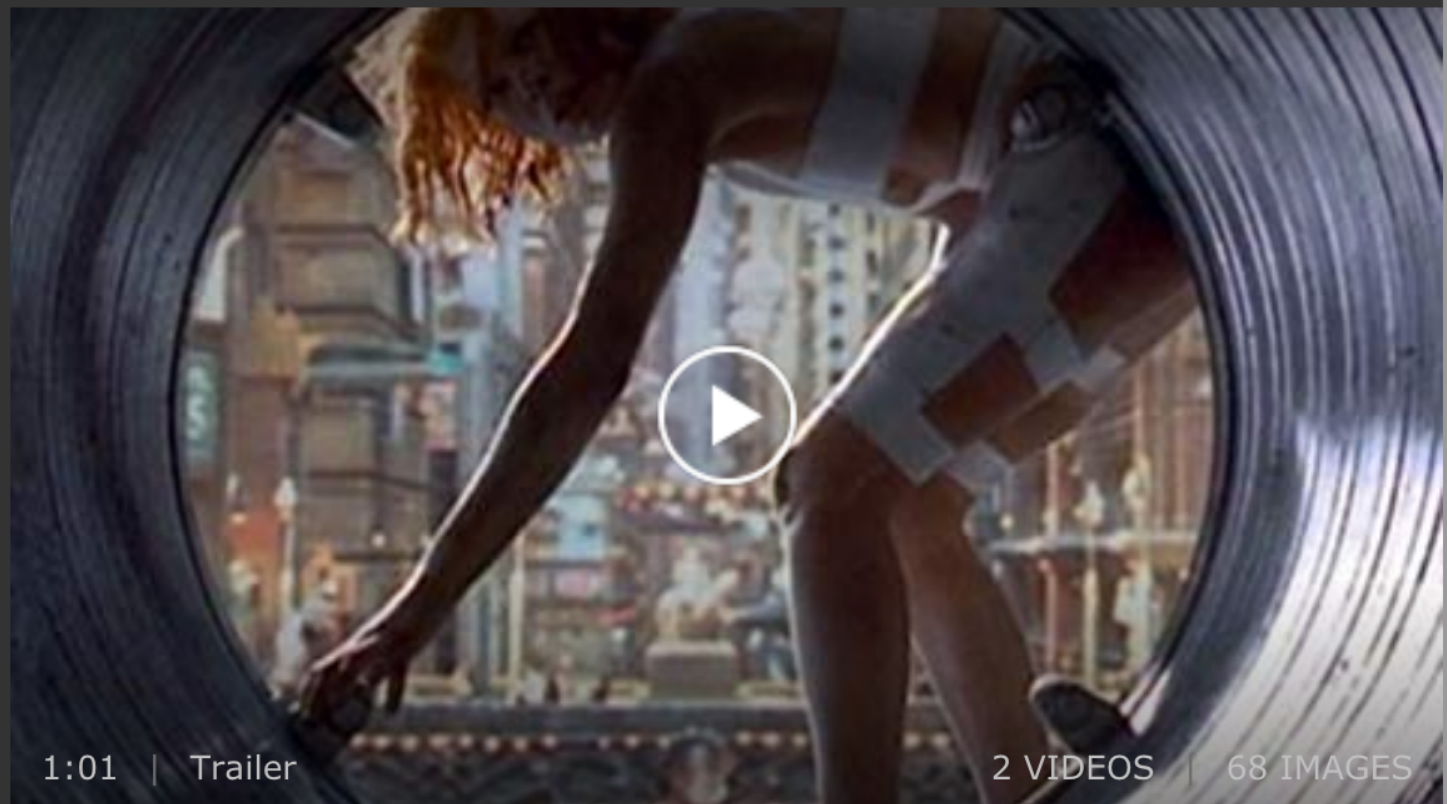
Zachary Le

+ Das fünfte Element (1997)

7,7/10 399.500 Rate This

The Fifth Element *(original title)*

12 | 2h 6min | Action, Adventure, Sci-Fi | 28 August 1997 (Germany)



Cast

Edit

Cast overview, first billed only:

	Bruce Willis	...	Korben Dallas
	Gary Oldman	...	Zorg
	Ian Holm	...	Cornelius
	Milla Jovovich	...	Leeloo
	Chris Tucker	...	Ruby Rhod
	Luke Perry	...	Billy
	Brion James	...	General Munro
	Tommy 'Tiny' Lister	...	President Lindberg (as Tommy 'Tiny' Lister Jr.)
	Lee Evans	...	Fog

IMDb (Part 1)

2. What are the data management and analysis issues behind this?

Question 1

How is the data on films, actors, directors etc. modelled and stored in IMDb?

Question 2

How are links between these data modelled and stored in IMDb?

next week:

Question 3

How do we query this data?

...

IMDb (Part 1)

3. Basics to be able to solve these problems
   (a) Slides
   (b) Jupyter/Python/SQL Hands-on

- Entity-relationship model
- Relational model

Main learning objectives

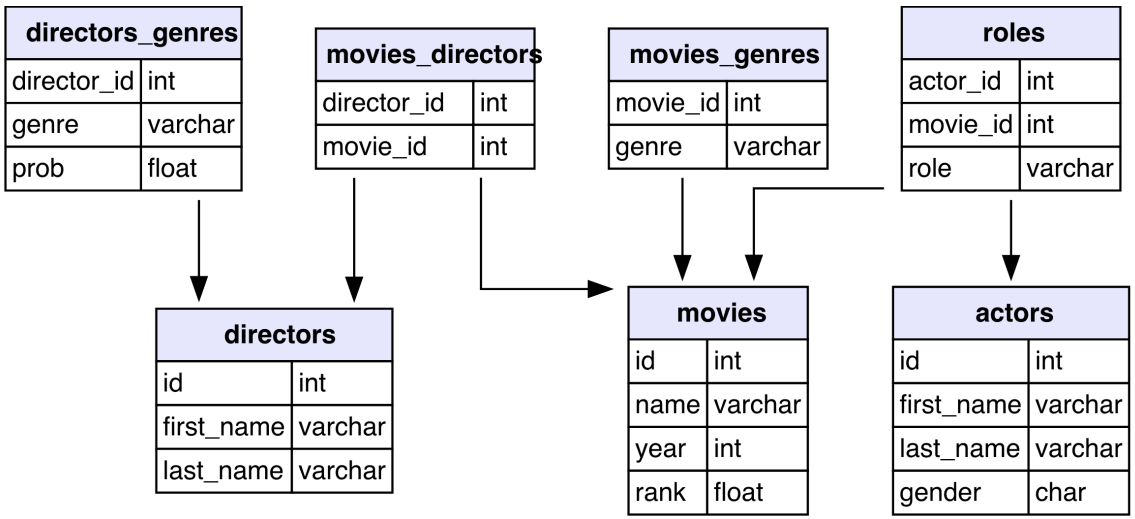
Entity Relationship Modelling

ER, Entity types, Relationship types (including n-ary), Domain, Attribute, Key, Functionalities (Chen notation), Functional Determined, N:M, 1:N, 1:1, N:M:P, Generalisation

Relational model

Relation, Tuple, Relational Schema, translating ER to RM, Key, Foreign Key

Initial Situation



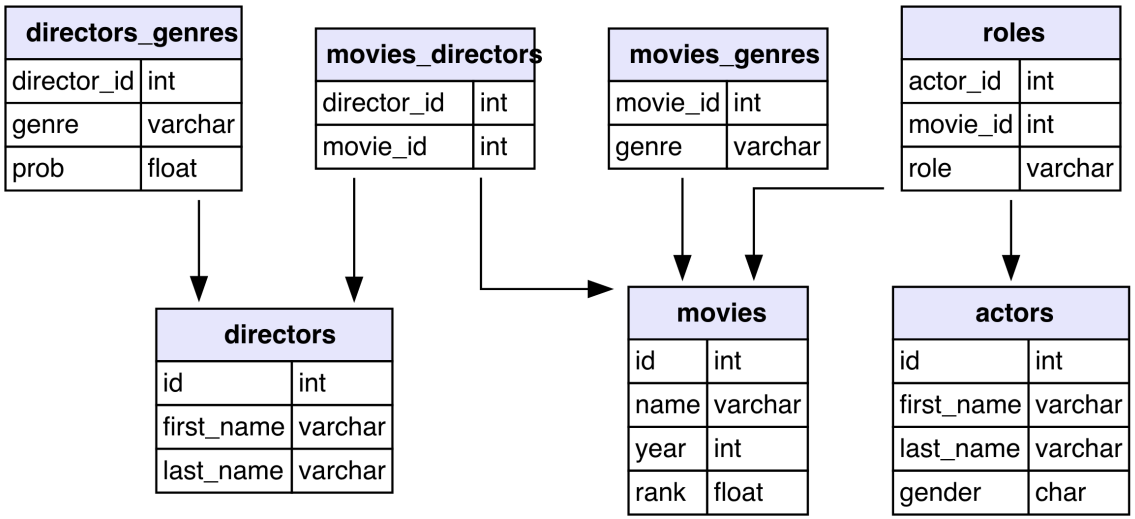
Caution

This is not a correct ER diagram!

- Table model as found on the website:
  https://relational.fit.cvut.cz/dataset/IMDb
- We will reconstruct (reverse engineer) the underlying entity-relationship data model (ER model) from this in the following.

IMDb Data simplified

[https://relational.fit.cvut.cz/dataset/IMDb]



Caution

This is not a correct ER diagram!

- 7 tables
- Header: Table name, left column: Attribute name, right column: attribute type
- Arrow from T1 → T2: Attribute from T1 references attributes in T2 (exact attribute is not specified)

Example: directors_genres → directors

means:

directors_genres.`director_id` → directors.`id`

IMDb Data basis [https://www.IMDb.com/interfaces/]

IMDb Datasets

Subsets of IMDb data are available for access to customers for personal and non-commercial use. You can hold local copies of this data, and it is subject to our terms and conditions. Please refer to the Non-Commercial Licensing and copyright/license and verify compliance.

Data Location

The dataset files can be accessed and downloaded from https://datasets.imdbws.com/. The data is refreshed daily.

IMDb Dataset Details

Each dataset is contained in a gzipped, tab-separated-values (TSV) formatted file in the UTF-8 character set. The first line in each file contains headers that describe what is in each column. A '\N' is used to denote that a particular field is missing or null for that title/name. The available datasets are as follows:

title.akas.tsv.gz - Contains the following information for titles:

- titleId (string) - a tconst, an alphanumeric unique identifier of the title
- ordering (integer) – a number to uniquely identify rows for a given titleId
- title (string) – the localized title
- region (string) - the region for this version of the title
- language (string) - the language of the title
- types (array) - Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd", "festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning
- attributes (array) - Additional terms to describe this alternative title, not enumerated
- isOriginalTitle (boolean) – 0: not original title; 1: original title

title.basics.tsv.gz - Contains the following information for titles:

- tconst (string) - alphanumeric unique identifier of the title
- titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc)
- primaryTitle (string) – the more popular title / the title used by the filmmakers on

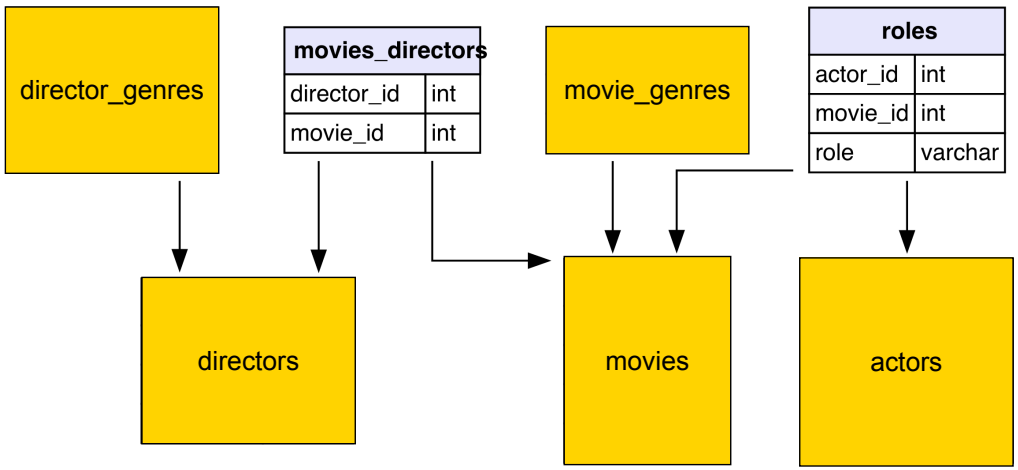
Semantics of tables

directors_genres:	genres assigned to directors
directors:	directors
movies_directors:	movies made by a director
movies:	movies
movies_genres:	genres assigned to movies
actors:	actors
roles:	actors which appeared in a movie and their role

Entity types: Intuition

Caution

This is not a correct ER diagram!



- Entity types: describes a **class** of entities that semantically represent the same concept.
- Notation: Rectangle (colour does not matter)

Entity types

Entity types

An entity type E_i describes and models a set of entities. Each entity type has a unique name (preferably plural). An entity type can be associated with any number of relationship types, but **never** directly with other entity types.

Examples:

- directors
- movies
- actors
- The entity type "directors" contains the entity "Quentin Tarantino".

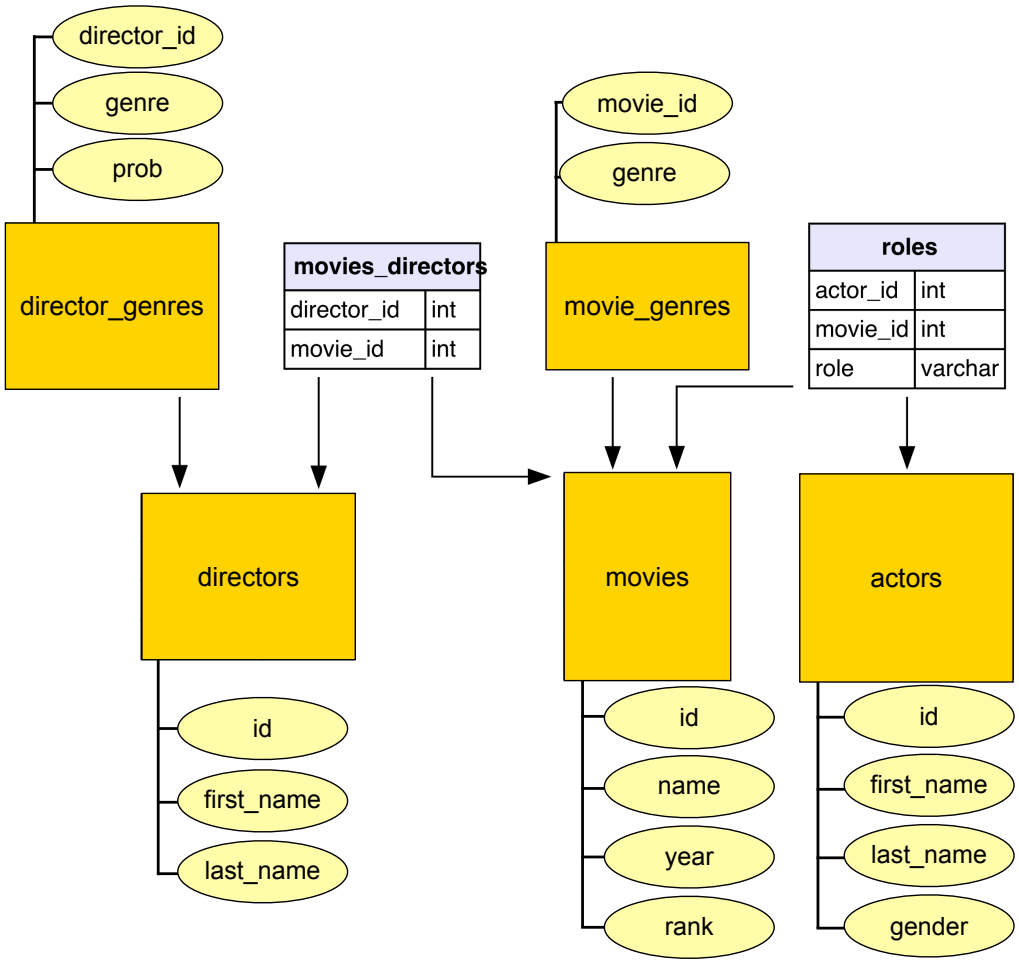
Compare:

Class and Object (in object-oriented modelling)

vs

Entity type and entity (in Entity-Relationship Modelling)

Attributes: Intuition



Caution

This is not a correct ER diagram!

- Attribute: An attribute describes an aspect of an entity or relationship type.
- Notation: Ellipse (colour does not matter)

Attributes

Attributes

An attribute describes an aspect of an entity type or relationship type. An attribute belongs to exactly one entity type or relationship type. For an entity type E_i, its attributes are named $A_{i,1}, \ldots, A_{i,k_i}$. For a relation type B, its attributes are named $A_1^B, \ldots, A_{k_B}^B$.

Examples:

- directors has the attributes *first name* and *last name*
- movies has the attribute *year of release*

Domain and attribute values

Domain (aka value range, type)

A domain is a set of **atomic** values. These values must not be structured (i.e. further divisible). Domains are notated as D, e.g. integer, float, string, etc. The domain of the attribute is usually not specified in ER, but can be additionally annotated.

Attribute values

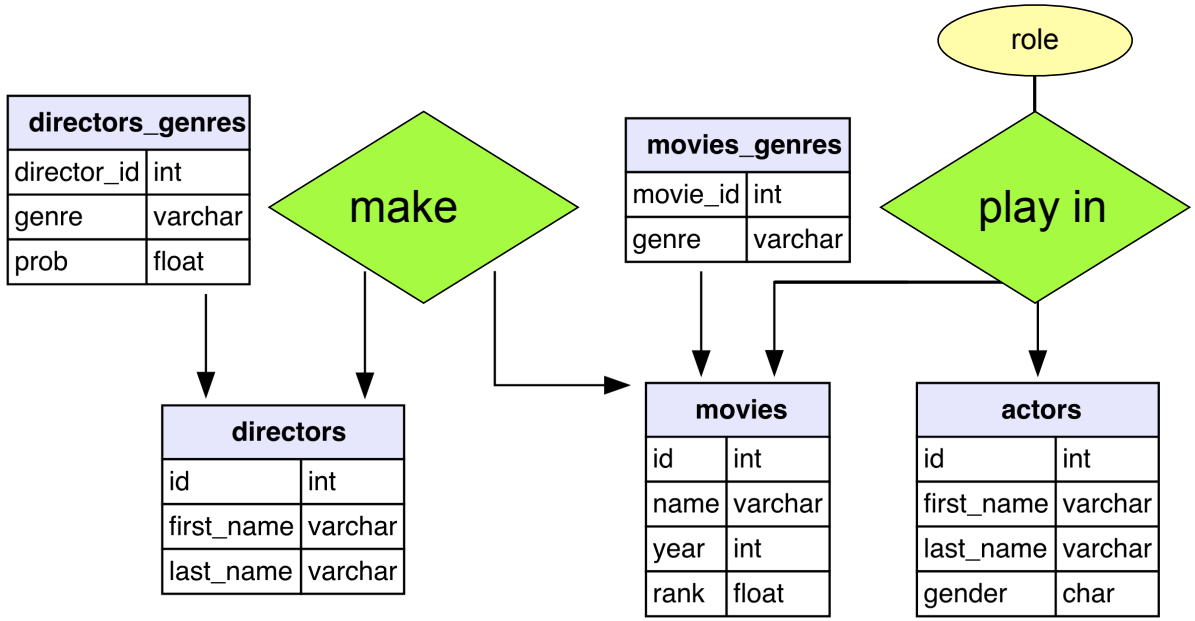
An attribute value is a concrete instance of the attribute's domain.

Example:
Entity type: lecture, attribute: duration
Entity: BDE, attribute value: 90 min

Relationship types: Intuition

Caution

This is not a correct ER diagram!



- Relationship types: express that two entity types are connected (related) to each other.
- Notation: diamond (colour does not matter)
- Attention: Reading direction not always clear but often the convention "from left to right" applies

Examples: directors make movies; actors play in movies

Relationship types

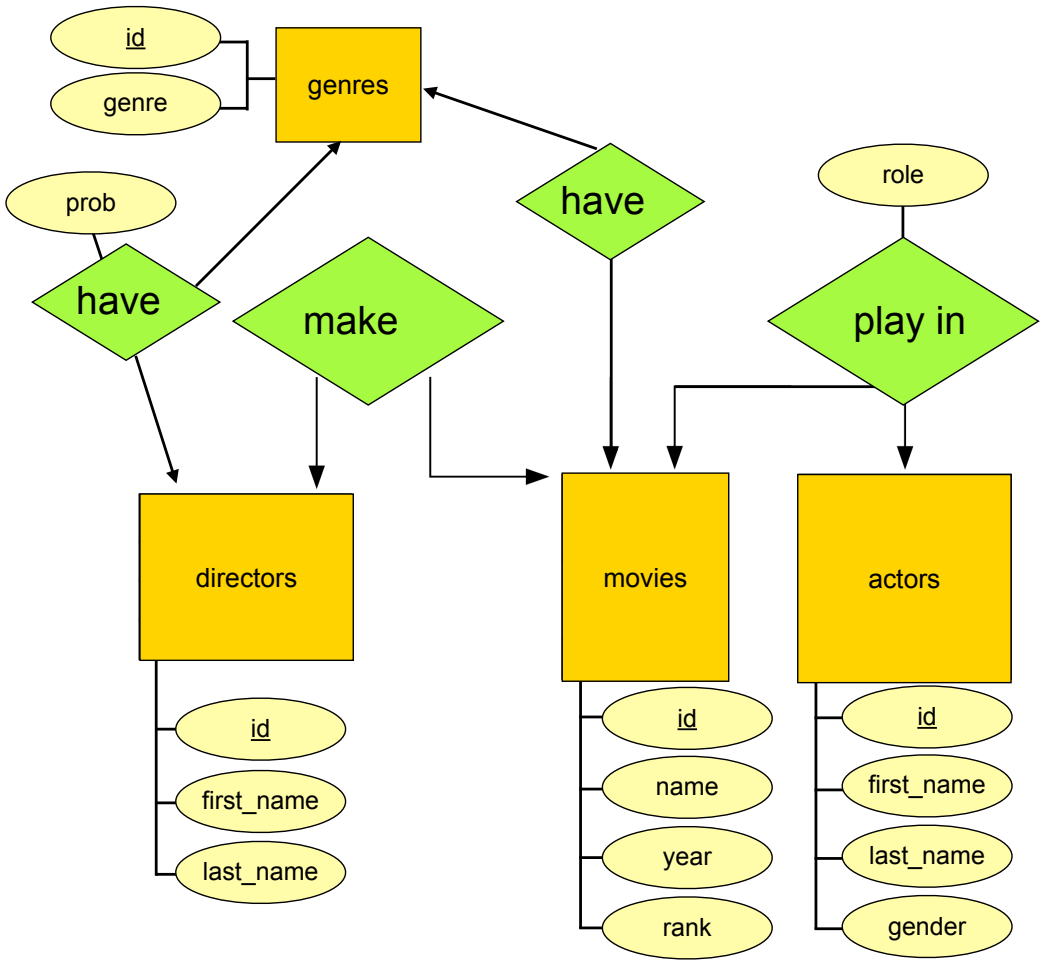
Relationship types

A relationship type $B \subseteq E_1 \times \ldots \times E_n$ describes a set of relationships. Each relationship type in an ER diagram has a unique name (preferably plural). A relationship type can be associated with any number $n \geq 2$ of entity types but **never** directly with other relationship types. Relationship types associated with $n = 2$, $n = 3$, $n \geq 4$, entity types are called binary, ternary, n-ary relationship types, respectively. In addition, a relationship type can have its own attributes $A_1^B, \ldots, A_{k_B}^B$.

Examples:

- make: binary
- play in (alternatively: have): binary
- attribute role is an attribute of relationship type play in

Keys: Intuition



- Key(-attributes) uniquely identify the entity type's entities
- Notation: underline key attributes

Keys

Keys

A subset of the attributes of an entity type, also called key attributes, that uniquely identify each entity of the entity type by these attributes. The subset is marked by underlining.

Minimality of Keys

1. If we can reduce the subset of attributes and still conceptually identify all entities with it: then the current subset **does not** form a key, i.e. we reduce the subset of attributes.
2. If the subset of attributes cannot conceptually distinguish all entities: then the subset of attributes also forms **no** key, i.e. we enlarge the subset of attributes in an appropriate way.

Keys: Formal Definition

Keys

A key $\mathcal{S}_i \subseteq \{A_{i,1}, \ldots, A_{i,k_i}\}$ of an entity type E_i is a minimal, non-empty subset of the attributes of E_i such that any **conceivable** entity is uniquely determinable using the corresponding attribute values. The functions $a_{\mathcal{S}_i}, a_T$ assign to each entity from E_i a tuple of the attribute values for the attributes in $\mathcal{S}, T$. Thus, the key property is formally:

$$\forall e_1, e_2 \in E_i.\ a_{\mathcal{S}_i}(e_1) = a_{\mathcal{S}_i}(e_2) \Rightarrow e_1 = e_2$$

and

$$\nexists T \subset \mathcal{S}_i.\ \forall e_1, e_2 \in E_i.\ a_T(e_1) = a_T(e_2) \Rightarrow e_1 = e_2$$

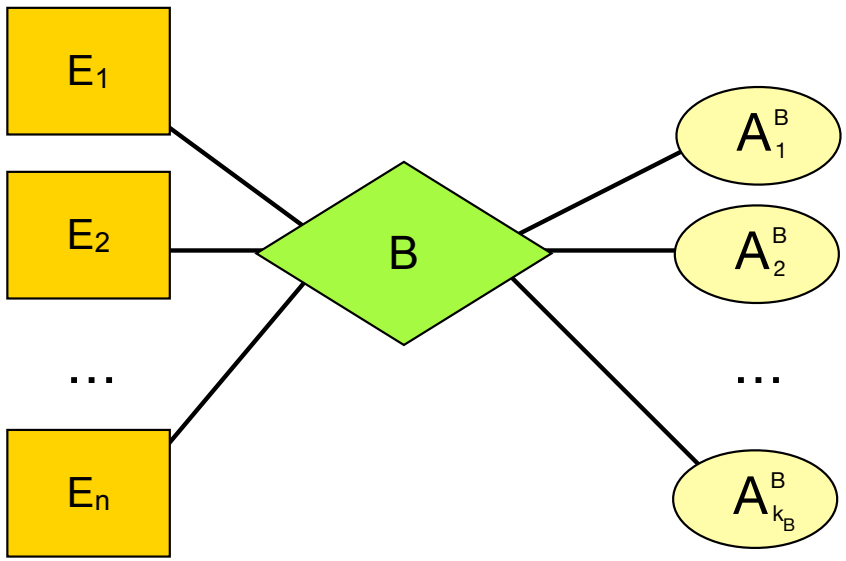
Interplay of Relationship Types and Entity Types

Interplay of Relationship Types and Entity Types

Suppose we have a relationship type B with n participating entity types $E_1, \ldots, E_n$. B has k_B attributes $A_1^B, \ldots, A_{k_B}^B$. Let each attribute have an associated domain: $A_1^B \in D_1^B, \ldots, A_{k_B}^B \in D_{k_B}^B$.

Similarly, we interpret the entity types $E_1, \ldots, E_n$ as domains with $e_i \in E_i$.

Then, B can be written as a $(n + k_B)$-ary tuple:
$(e_1, \ldots, e_n, a_1^B, \ldots, a_{k_B}^B) \in E_1 \times \ldots \times E_n \times D_1^B \times \ldots \times D_{k_B}^B$.

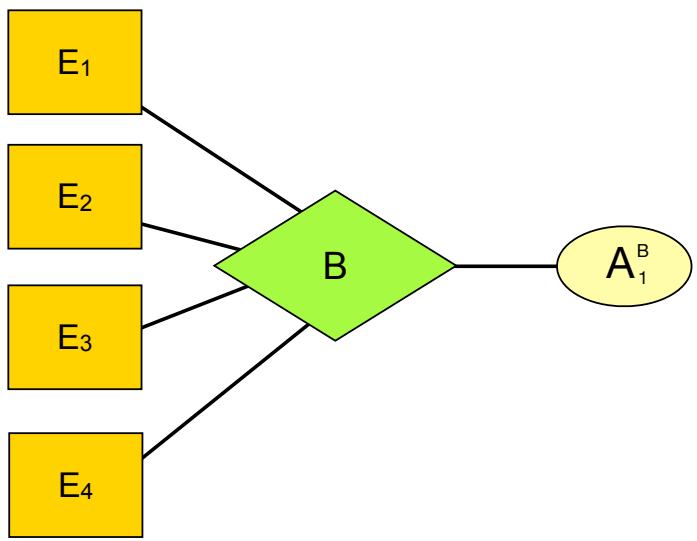


Example: n-ary Relationship Type with Entity Types as Domains

Relationship type B with $n = 4$ participating entity types $E_1, \ldots, E_4$. B has its own attribute ($k_B = 1$). This instance of B contains 5 relationships $b_0, \ldots, b_4$:

Example:

B					
	E_1	E_2	E_3	E_4	A_1^B
b_0	e_1^4	e_2^8	e_3^1	e_3^9	c
b_1	e_1^6	e_2^4	e_3^1	e_3^3	d
b_2	e_1^8	e_2^6	e_3^2	e_3^9	j
b_3	e_1^1	e_2^8	e_3^1	e_3^9	c
b_4	e_1^1	e_2^8	e_3^7	e_3^9	d



Interplay of Relationship Types and Entity Types with Keys

Interplay of Relationship Types and Entity Types **without** Keys

Definition from above:
"Similarly, we interpret the entity types $E_1, \ldots, E_n$ as domains with $e_i \in E_i$.
Then, B can be written as a $(n + k_B)$-ary tuple:
$(e_1, \ldots, e_n, a_1^B, \ldots, a_{k_B}^B) \in E_1 \times \ldots \times E_n \times D_1^B \times \ldots \times D_{k_B}^B$."

We still kind of do that (in our heads), but rather than using the entity as an attribute value we just use the key of that entity as an attribute value!

Interplay of Relationship Types and Entity Types **with** Keys

Let's assume that the key attribute of each entity type E_i is "id" and has an integer type.

new:

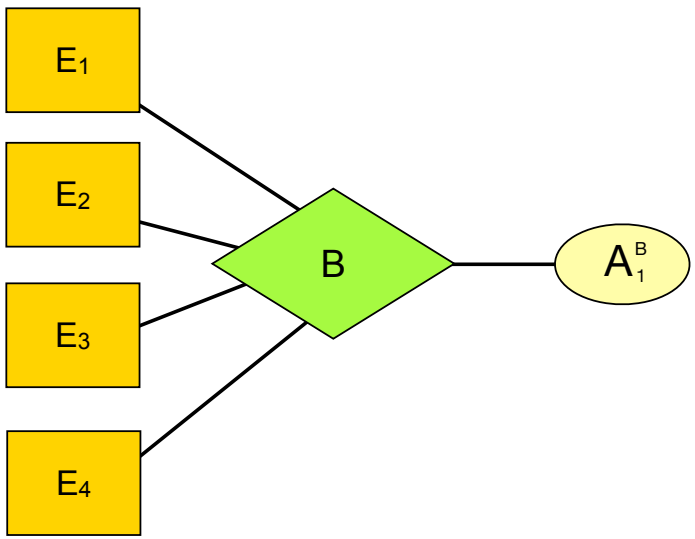
Similarly, we interpret the attributes $E_1, \ldots, E_n$ as domains with $e_i \in E_i.\mathbf{id}$.
Then, B can be written as a $(n + k_B)$-ary tuple:
$(e_1, \ldots, e_n, a_1^B, \ldots, a_{k_B}^B) \in E_1 \times \ldots \times E_n \times D_1^B \times \ldots \times D_{k_B}^B$.

Example: n-ary Relationship Type with Entity Types Key's Domains

Relationship type B with $n = 4$ participating entity types $E_1, \ldots, E_4$. B has its own attribute ($k_B = 1$). This instance of B has 5 relationships $b_0, \ldots, b_4$:

Example:

B					
	E_1	E_2	E_3	E_4	A_1^B
b_0	4	8	1	9	c
b_1	6	4	1	3	d
b_2	8	6	2	9	j
b_3	1	8	1	9	c
b_4	1	8	7	9	d

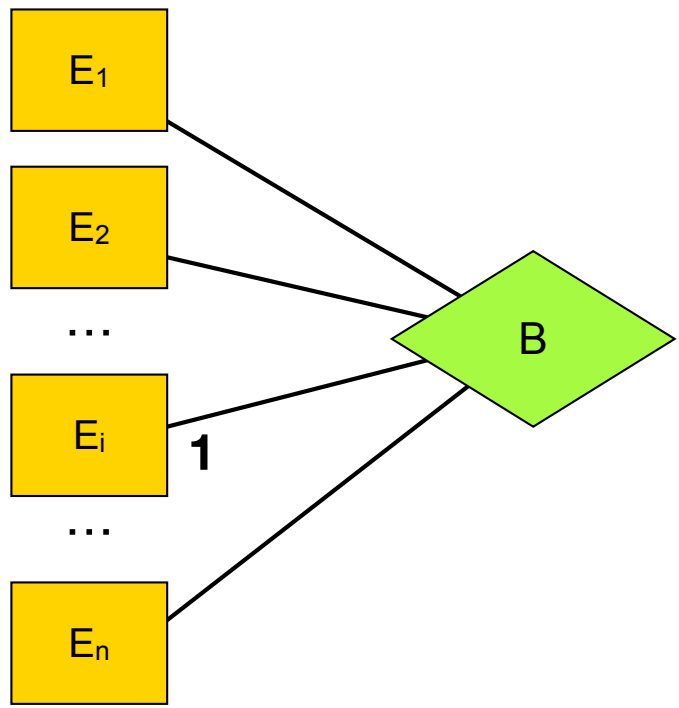


Functionally Determined

Functionally Determined

Suppose we have a relationship type $B \subseteq E_1 \times \ldots \times E_n$. Furthermore, at entity type $E_i, 1 \leq i \leq n$ stands a "1". Then for each pair of relationships $b_1 = (e_1, \ldots, e_n) \in B$ and $b_2 = (e'_1, \ldots, e'_n) \in B$ it must hold: If $(\bigwedge_{\{1 \leq k \leq n | k \neq i\}} e_k = e'_k) \Rightarrow e_i = e'_i$. Since B is a set, it follows that $b_1 = b_2$.

In other words:
if in both relationships b_1 and b_2 all entities of entity types $E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_n$ (i.e. all except E_i) are equal, then this implies that e_i and e'_i are equal and thus must reference the same entity.



How to read

"$E_1, \ldots, E_{i-1}, E_{i+1}, \ldots, E_n$ determine E_i" or simply: "E_i is determined functionally by the other entity types."

Functionally Determined: Formal Definition

Functionally Determined

Let $B \subseteq E_1 \times \ldots \times E_n$ be a relationship type. Assume $C_B = (c_1, \ldots, c_n)$ with $c_1, \ldots, c_n \in \{1, A, B, \ldots, Z\}$ describes the cardinalities associated with B, where the values of all c_i in C_B not equal to 1 are pairwise different.
E_i is functionally determined if $c_i = 1$. In this case,

$$\forall (e_1, \ldots, e_n), (e'_1, \ldots, e'_n) \in B. \left(\bigwedge_{\{1 \leq j \leq n | j \neq i\}} e_j = e'_j \right) \Rightarrow e_i = e'_i$$

must hold. We denote this by a "1" at E_i in the graphical representation. If the entity type is not functionally determined, we write a letter, usually starting with "N", "M", ...

Example: Four-ary Relationship Type with Functionalities

Relationship type B with four participating entity types $E_1, \ldots, E_4$ and 5 relationships each. B has no attributes of its own.

Example: ($i = 3$)

B			
E_1	E_2	E_3	E_4
4	8	1	9
6	4	1	3
8	6	2	9
1	8	**1**	9
1	8	**7**	9

Let us assume that there is a 1 at E_3 in the corresponding ER, so E_3 is functionally determined according to the ER.
The data contradicts this: $(1, 8, 1, 9)$ and $(1, 8, 7, 9)$ have the same value for all attributes except E_3.

Example: ($i = 2$)

B			
E_1	E_2	E_3	E_4
4	8	1	9
6	4	1	3
8	6	2	9
1	8	**1**	9
1	8	**7**	9

Let us assume that there is a 1 at E_2 in the ER, so E_2 is functionally determined according to the ER.
The data does not contradict this.

Functionally Determined and Chen Notation in an ER Diagram

Functionality (also called Chen notation)

Functionalities are notated between entity type and relationship type. The reading direction of this notation is towards the entity type, i.e. this notation makes a statement about whether other entity types participating in this relationship type functionally determine this entity type.

Caution

There are quite different notations for functionally determinedness, an important alternative that we will not use in the context of this lecture is the (min,max) notation.

Further information see https://youtu.be/oxcaqMQdGyw (in German).

Functionally Determined or not in an ER diagram

Functionally not determined in an ER diagram

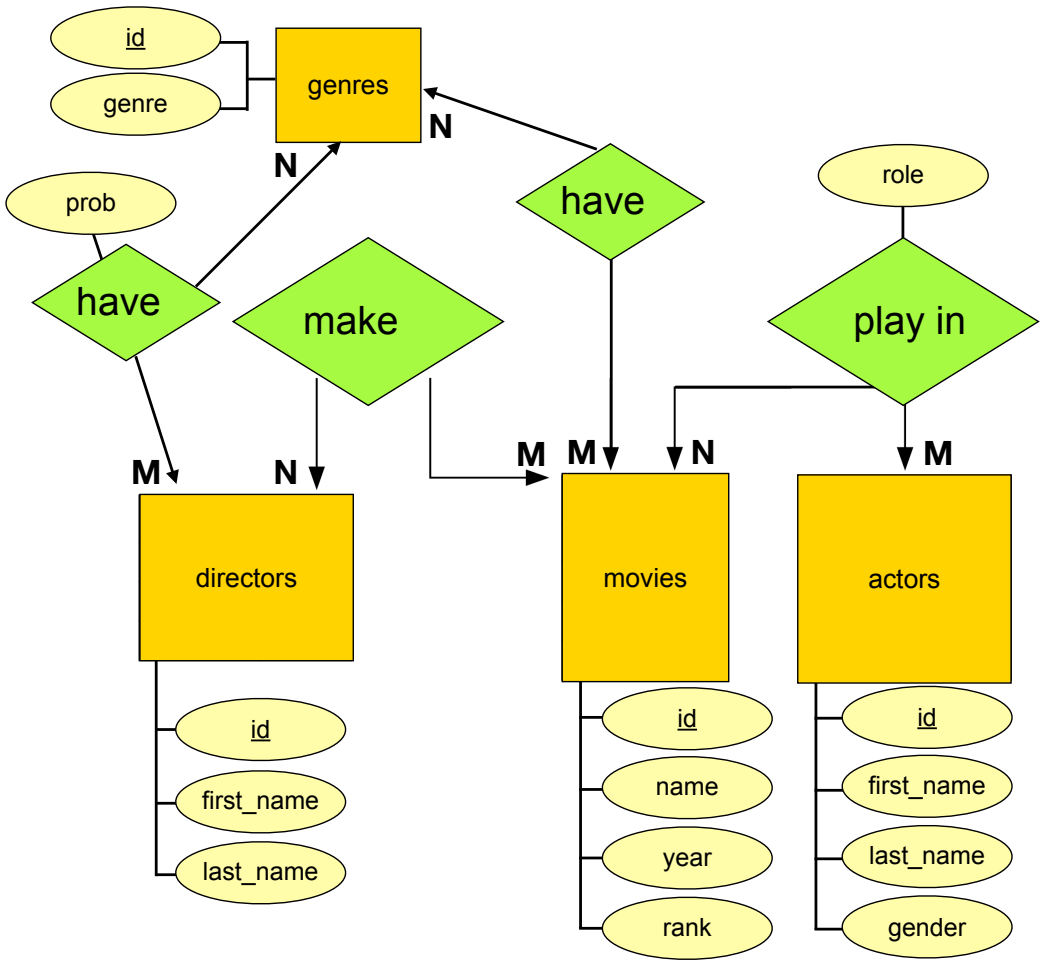
An ‘N’ or ‘M’ (or other letter) means: The entities of this entity type are **not functionally determined** by the other entity types.

Functionally determined in an ER diagram

A ‘1’ means: The entities of this entity type are **functionally determined** by the other entity types.

In ER we call these annotations “functionalities”.

Functionalities for the IMDb Example

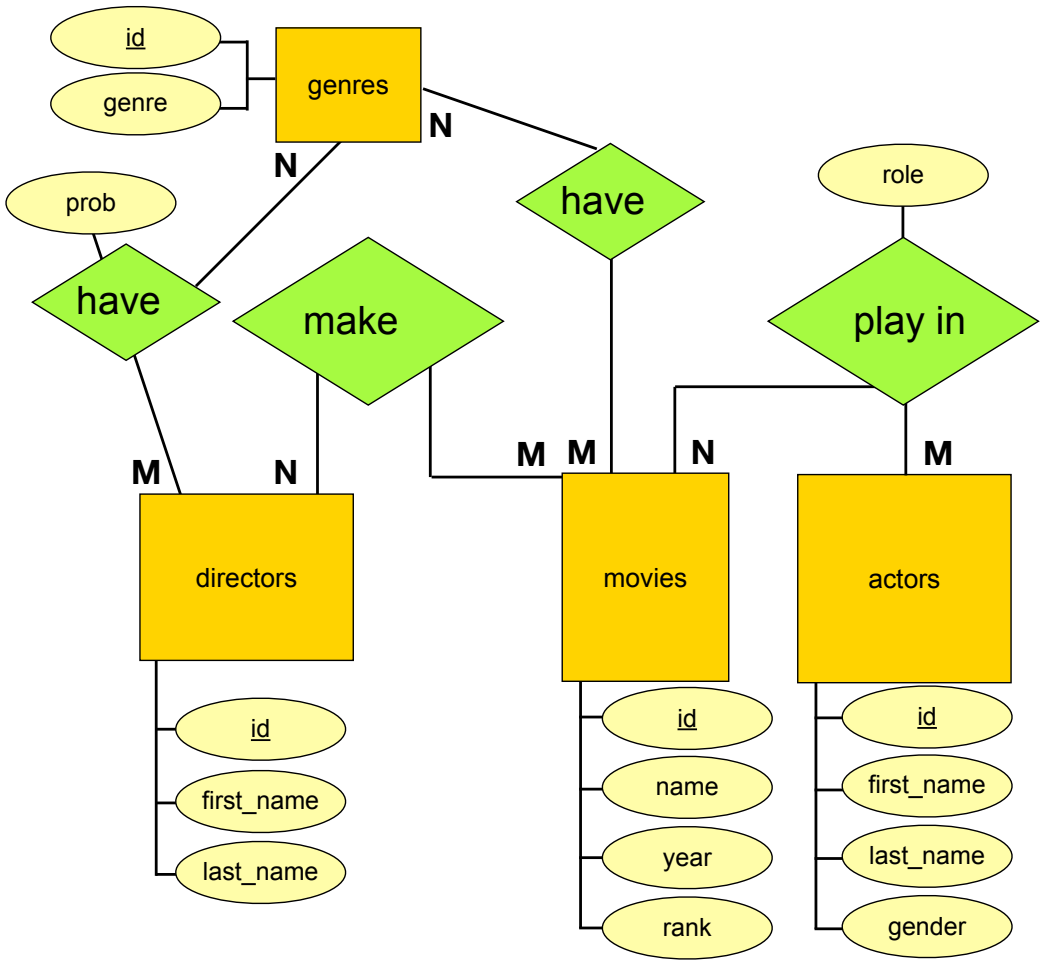


Caution

This is not a correct ER diagram!

Functionalities indicate whether the entities of an entity type are functionally determined by other entity types. Only Ns and Ms are here, i.e. nothing is functionally determined here.

Functionalities (notation without arrows)

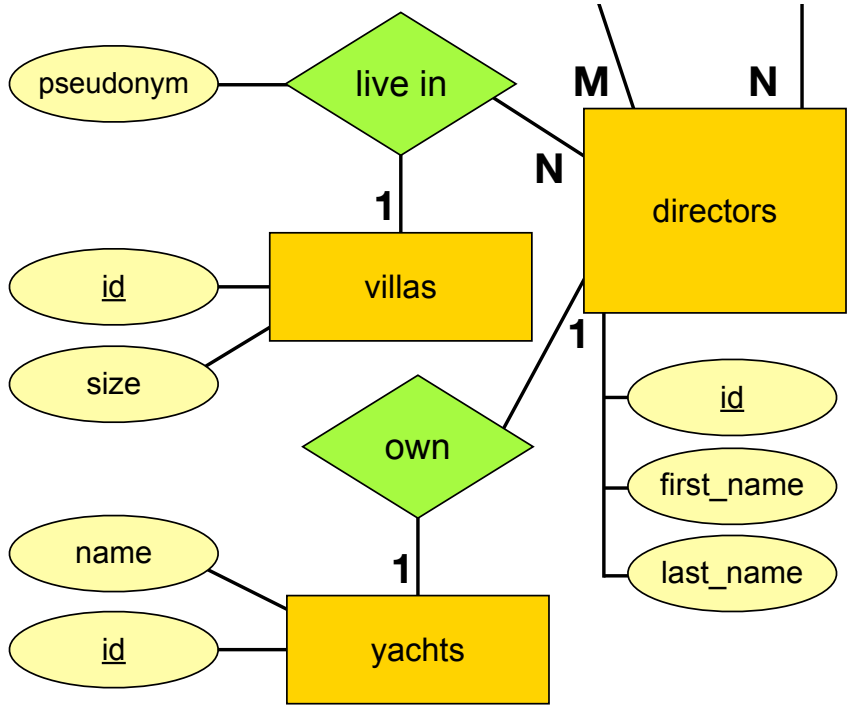


All good.

This **is** a correct ER diagram.

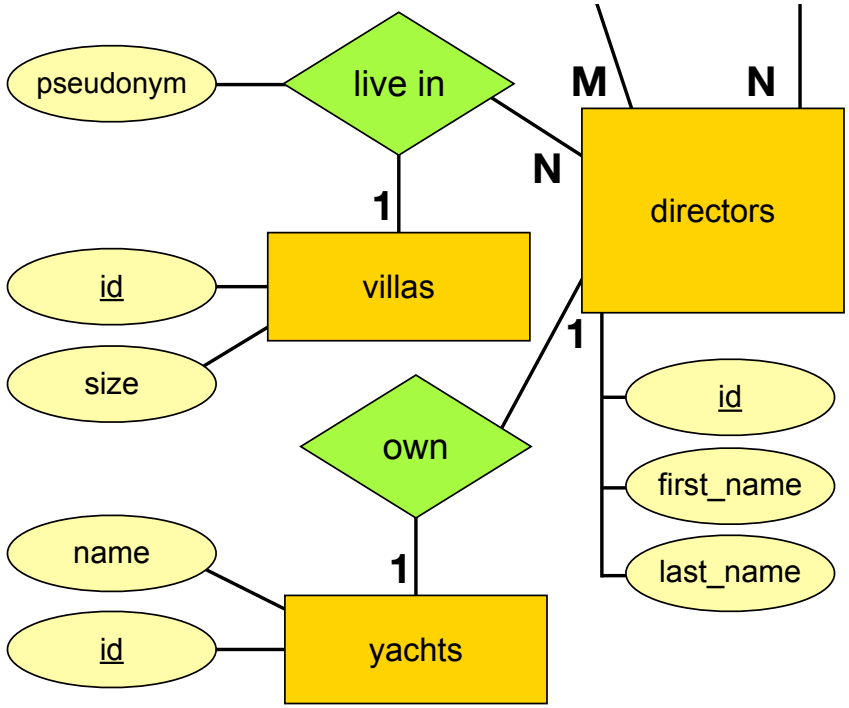
We do not use arrows in the following, as they could otherwise lead to confusion with other notation elements and one of the zillion variants of ER.

1:N-Relationship Examples



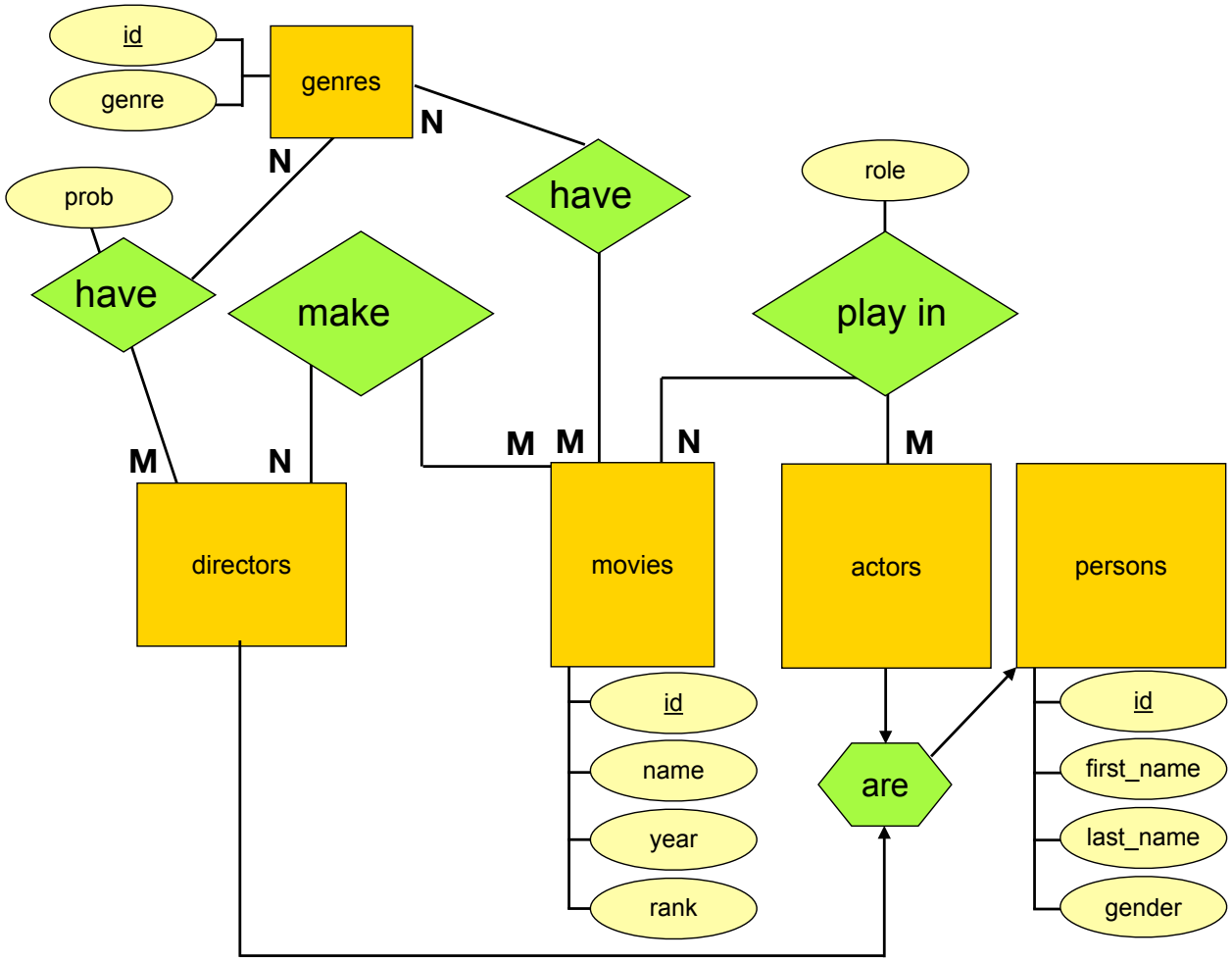
- Directors determine villas: if I have an entity of directors, I know if and if so in which villa he/she lives.
- in other words: A director lives in **maximum** one villa.
- Villas **do not** determine directors: if I have an entity of villas, I don't know which director lives there.
- in other words: more than one director can live in the same villa.
- Villas are allowed to be uninhabited and directors are allowed to be homeless!

1:1-Relationship Examples



- Directors determine yachts: if I have an entity of directors, I know if and if so which yacht he/she owns.
- in other words: A director owns **maximum** one yacht.
- Yachts determine directors: if I have an entity of yachts, I know if and if so which director it belongs to.
- in other words: the same yacht belongs to **maximum** one director.
- Yachts are allowed to be director-less and directors are allowed to be yacht-less!

Generalisation



- Similar to object-oriented modelling, overlapping concepts should not be modelled independently. This avoids redundancies.
- Subclasses inherit all attributes of their superclass.

Inheritance: Formal Definition

Inheritance

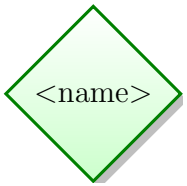
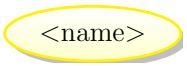
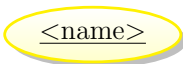
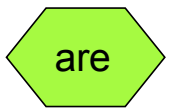
Two entity types E_1, E_2 can be in an inheritance relation $V \subseteq \mathcal{E} \times \mathcal{E}$.
$(E_1, E_2) \in V$ means, E_1 is the subtype (subclass, specialisation) and E_2 is the supertype (superclass). The following properties then apply:
All entities of the subtype E_1 are also contained in the supertype E_2:

$$E_1 \subseteq E_2$$

The subtype E_1 has all attributes of the supertype E_2:

$$\{A_{E_1,1}, A_{E_1,2}, \ldots, A_{E_1,k_{E_1}}\} \supseteq \{A_{E_2,1}, A_{E_2,2}, \ldots, A_{E,k_{E_2}}\}$$

Overview of Entity Relationship Modelling Elements

Symbol	Meaning
	entity type
	relationship type
1, N or M	functionality (Chen notation)
	attribute
	key (-attribute)
	inheritance

The colours of the symbols do not matter.
The inheritance “are” is also sometimes called “is a” (which is inconsistent because of the plural of the entity types).

ER-Modell: Formal Definition

ER-Modell

An ER model $(\mathcal{E}, \mathcal{B}, \mathcal{V}, \mathcal{A}, \mathcal{C})$ consists of a tuple formed by

- a set of entity types $(\mathcal{E})$,
- relationship types $(\mathcal{B})$,
- a set of inheritances $\mathcal{V}$,
- the attributes $\mathcal{A}$ associated with the entity, and
- relationship types and the functionalities (aka cardinalities) $\mathcal{C}$ associated with the relationship types.

The Relational Model[1]

Relation, Unnamed Relational Schema, and Instance

A relation $R = (\{R\}, [R])$ consists of

- an unnamed relational schema $[R]$ and
- an instance $\{R\}$.

The concrete types and number of domains $\subseteq D_1 \times \ldots \times D_n$ are called the unnamed relational schema, i.e. the attributes do not have names.
$\{R\} \subseteq D_1 \times \ldots \times D_n$ is any subset of the cartesian product over n domains and called the instance.

Tuple and Attribute Values

Each element $t = (a_1, \ldots, a_n) \in \{R\}, a_{1 \leq i \leq n} \in D_i$ is called a tuple. The individual values of a_i are called **attribute values**.

[1] The notation follows the textbook "Datenbanksysteme" by Kemper&Eickler.

(Named) Relational schema

A relation defines the schema only along its domains but not along attribute names. Let's fix that.

(Named) Relational schema

A relational schema specifies domains **and** attribute names for each relation, i.e. we specify not only the domains but also the attribute names.
A relational schema is notated as **sequence** $[R] : \{[A_1 : D_1, \ldots, A_n : D_n]\}$.
The **key attributes** are indicated by underlining.

Examples:

[movies] : {[<u>id:int</u>, name:varchar, year:int , rank:float]}
[actors] : {[<u>id:int</u>, first_name:varchar, last_name:varchar, gender:char]}

Order of Attributes and Tuples

The order in which we define the attributes of a relation schema **does not matter**. Likewise, the "order" of the tuples in an instance does not matter: in a set, there is no notion of order.
In summary, the order of attributes and tuples is **not** information-bearing.

Translating ER to the Relational Model (1/2)

Translation of Entity Types

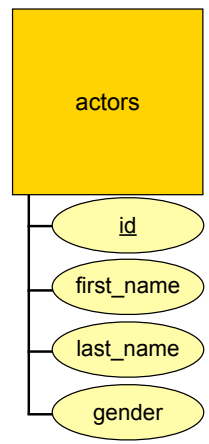
Each entity type E_i with attributes $A_{i,1}, \ldots, A_{i,k_i}$ is translated as follows. Let $\text{keyset}(E_i) \subseteq \{A_{i,1}, \ldots, A_{i,k_i}\}$ be the key attributes of E_i and the complement $\text{nonkeyset}(E_i) = \{A_{i,1}, \ldots, A_{i,k_i}\} \setminus \text{keyset}(E_i)$ be the non-key attributes of E_i. Then E_i is translated into a relational schema as follows:

$$[E_i] : \{[\ \underbrace{\underbrace{a \text{ for } a \in \text{keyset}(E_i)}_{\text{key attributes of } E_i}}_{\text{keyset of } [E_i]}, \quad \underbrace{a \text{ for } a \in \text{nonkeyset}(E_i)}_{\text{non-key attributes of } E_i}\ ]\}.$$

The attribute domains $D_1, \ldots, D_n$ are to be chosen appropriately.

Here, the notation "a for a in S" emphasises that the attributes contained in attribute set S are unnested.

Example: Entity type actors:
[actors] : {[id:int, first_name:varchar, last_name:varchar, gender:char]}



Translating ER to the Relational Model (2a/2)

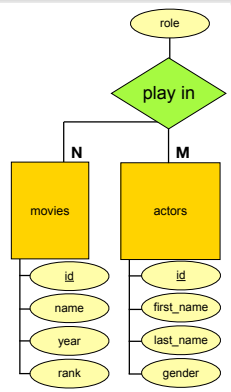
Translation of Relationship Types (Variant without Foreign Key)

Any relationship type B with n participating entity types $E_1, \ldots, E_n$ as well as k_B attributes $A_1^B, \ldots, A_{k_B}^B$ can be translated in very general terms as follows: Let $\text{keyset}(E_i) \subseteq \{A_{i,1}, \ldots, A_{i,k_i}\}$ be the key attributes of E_i. Then the relational schema $[B]$ is given by:

$$[B] : \{[\ \underbrace{\underbrace{\underline{a \text{ for } a \in \text{keyset}(E_1)}}_{\text{key attributes of } E_1}, \ldots, \underbrace{\underline{a \text{ for } a \in \text{keyset}(E_n)}}_{\text{key attributes of } E_n}}_{CKS=\text{candidate key set of } [B]}, \underbrace{A_1^B : D_1^B, \ldots, A_{k_B}^B : D_{k_B}^B}_{\text{attributes of } B}\ ]\}$$

Note that depending on the functionalities and the attributes of entity type B CKS may still have to be increased or decreased.

Example: "Movies **have** actors":
modelling as explained above: (Incorrect: keeps CKS as is)
[have] : {[<u>movie_id:int, actors_id:int</u>, role:str]}
or alternative modelling: (Correct: this increases CKS)
[have] : {[<u>movie_id:int, actors_id:int, role:str</u>]}



Edit

Dr. Seltsam oder: Wie ich lernte, die Bombe zu lieben (1964)

Full Cast & Crew

See agents for this cast & crew on IMDbPro

Directed by

Stanley Kubrick

Writing Credits

Stanley Kubrick ...
Terry Southern ...
Peter George ...

Peter George ...

Cast (in credits order) verified as complete

Peter Sellers	...	Group Capt. Lionel Mandrake / President Merkin Muffley / Dr. Strangelove
George C. Scott	...	Gen. 'Buck' Turgidson
Sterling Hayden	...	Brig. Gen. Jack D. Ripper
Keenan Wynn	...	Col. 'Bat' Guano
Slim Pickens	...	Maj. 'King' Kong

Dr. Seltsam oder: Wie ich lernte, die Bombe zu lieben

Details

Full Cast and Crew
Release Dates

Caution

Peter Sellers had **three** different roles in Dr. Strangelove!
That is why we need alternative modelling here:

[have] : {[movie_id: (movies→id), actors_id: (actors→id), role:str]}

Since this can of course occur in principle in any film, the relation 'have' must be modelled in this way.

Related lists from IMDb users

1960s
a list of 29 titles
created 29 Oct 2017

must see
a list of 37 titles
created 1 day ago

The 25 Best War Movies of All Time
a list of 25 titles

Foreign key: Formal Definition

Foreign keys

Let R be a relation with the schema

$$[R] : \{[A_{R_1} : D_{R_1}, \ldots, A_{R_m} : D_{R_m}]\}$$

and S be a relation with the schema

$$[S] : \{[\underbrace{A_{S_1} : D_{S_1}, \ldots, A_{S_k} : D_{S_k}}_{\text{key attributes of } S}, A_{S_k+1} : D_{S_k+1}, \ldots, A_{S_n} : D_{S_n}]\}.$$

In addition, let the key attributes $A_{S_1}, \ldots, A_{S_k}$ as $A_{R_{f(1)}}, \ldots, A_{R_{f(k)}}$ also be attributes of the relation R, where f maps the indices of the attributes of S to the corresponding indices of the attributes in R. Then we write in the schema of $[R]$ for each of these attributes:

$$A_{R_{f(i)}} : (S \to A_{S_i})$$

instead of $A_{R_{f(i)}} : D_{R_{f(i)}}$, to make this recognisable as a *foreign key from R to S*.

Translating ER to the Relational Model (2b/2)

Translation of Relationship Types (variant using foreign key notation)

Each relationship type B with n participating entity types $E_1, \ldots, E_n$ as well as k_B attributes $A_1^B, \ldots, A_{k_B}^B$ can be implemented quite generally as follows: Let keyset$(E_i) \subseteq \{A_{i,1}, \ldots, A_{i,k_i}\}$ be the key attributes of E_i **as foreign key**. Then the relational schema $[B]$ is given by:

$$[B] : \{[\underbrace{\left.\begin{array}{l} \underline{A^B_{Fk_{1,1}} : (E_1 \to A_{1,s_1}), \ldots, A^B_{Fk_{1,s_{k_1}}} : (E_1 \to A_{1,s_{k_1}}),} \\ \ldots, \\ \underline{A^B_{Fk_{n,1}} : (E_n \to A_{n,s_1}), \ldots, A^B_{Fk_{n,s_{k_n}}} : (E_n \to A_{n,s_{k_n}}),} \end{array}\right\} \text{foreign key on } E_1, \ldots, E_n}_{\text{attributes of } B} \underbrace{A_1^B : D_1^B, \ldots, A_{k_B}^B : D_{k_B}^B}]\}$$

Note that depending on the functionalities and the attributes of entity type B the keyset of $[B]$ may still have to be increased or decreased.

Translating ER to the Relational Model (2c/2)

Example: (from above, converted to use foreign key notation)

"Movies **have** actors":
[have] : {[movie_id: (movies→id), actors_id: (actors→id), role:str]}

or alternative modelling: (allowing actors to have multiple roles in the same movie)
[have] : {[movie_id: (movies→id), actors_id: (actors→id), role:str]}

Shorter Foreign Key Notation

In case the entity type E being referenced has only a single key attribute, i.e. not a composed key, we can omit denoting the key of E in the foreign key. We will use this notation in the following wherever possible.

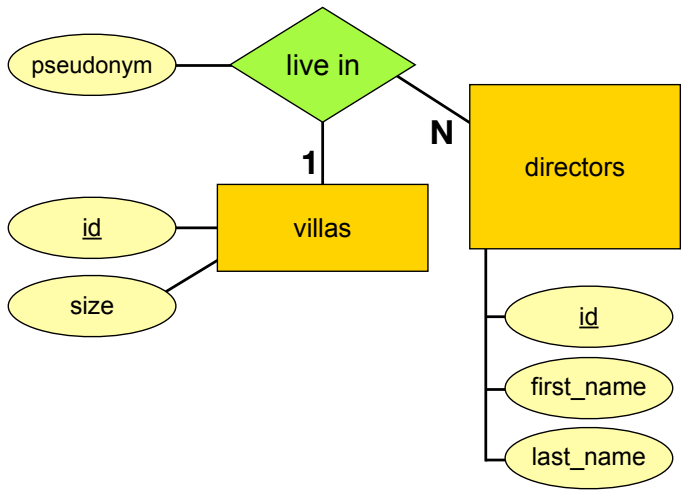
Example:
[have] : {[movie_id: (movies), actors_id: (actors), role:str]}
[have] : {[movie_id: (movies), actors_id: (actors), role:str]}

Simplified Translation of 1:N Relationship Types

Translation of 1:N Relationship Types and Foreign Keys

Any binary relationship type B with participating entity types E_1, E_2, functionality 1:N, and k_B attributes $A_1^B, \ldots, A_{k_B}^B$ can be implemented quite generally without creating a separate relation for B. The relation of the **determining** entity type E_2 is extended by a **foreign key** on E_1 and the attributes $A_1^B, \ldots, A_{k_B}^B$.

Example:
"Directors **live in** villas":
[directors] : {[id:int, first_name:varchar, last_name:varchar, villa_id:(villas), pseudonym: str]}



In the example, villa_id:(villas) is a foreign key on the key id of relation villas.

Simplified Translation of 1:1 Relationship Types

Translation of 1:1 Relationship Types

This is a special case of implementing 1:N relationship types. Either the entity type E_2 is extended by a **foreign key** on E_1 and the attributes $A_1^B, \ldots, A_{k_B}^B$ **or** E_1 by a **foreign key** on E_2 and the attributes $A_1^B, \ldots, A_{k_B}^B$.

Example:

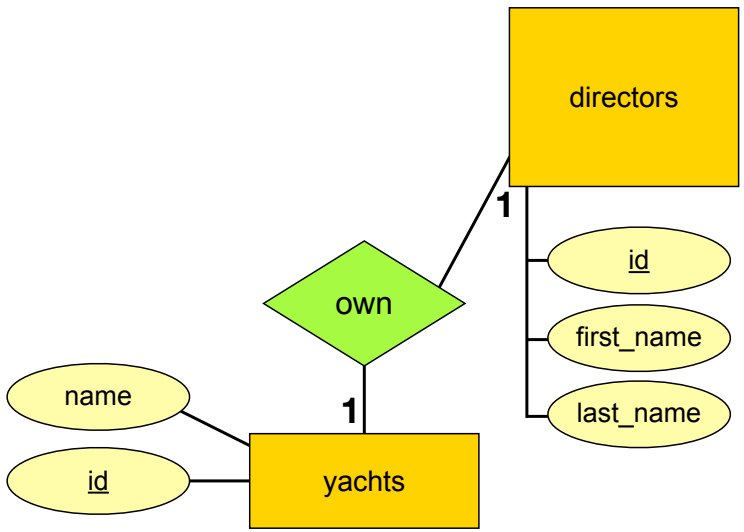
Either:

"Directors **own** yachts":

[directors] : {[id:int, first_name:varchar, last_name:varchar, yacht_id:(Yachts)]}

or:

[Yachts] : {[id:int, name:varchar, director_id:(Directors)]}



But not both!

The Relational Model in Python (relation.py from us)

The Relational Model in Python

Copyright Jens Dittrich & Marcel Maltry, Big Data Analytics Group, CC-BY-SA

```
In [1]: from ra.utils import load_csv
        from ra.relation import Relation
```

The `Relation` class

The class `Relation` is implemented in `ra.relation` and implements the following methods:

- `add_tuple(tup)` : Adds the tuple `tup` if the tuple's schema is valid.
- Several print methods that are showcased below for the IMDb dataset.

Remember: Neither the order of rows nor the order of columns carry any meaning in a relation!

```
In [2]: foo = Relation('foo', [('id', int), ('name', str)])
        foo.add_tuple( (2,'Hello') )
        foo.add_tuple( (7,'World') )
        foo.add_tuple( (1,'!') )

        foo

Out[2]: ---
        foo
        --------------
        id      name
        --------------
        1       !
        7       World
        2       Hello
```

- https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/RelationalModel.ipynb
- Note: we use the original version of the data from https://relational.fit.cvut.cz/dataset/IMDb and not our improved version of the model.

IMDb (Part 1)

4. Transfer of the basics to the concrete application

see various examples on IMDb above, summarised:

Question 1

How is the data on films, actors, directors etc. modelled and stored in IMDb?

Modelling: entity-relationship model, relational model
Data storage: This is a degree of freedom about which we do not have to make a decision at the moment: Any form of data storage that conforms to the relational model can be chosen!

This property is called **physical data independence**, i.e. as a user we don't need to know anything about how the data is physically stored and represented. In other words, we do not have to worry about how the relational model is implemented[2].

[2] For performance considerations, the physical implementation can then matter a lot after all. More on this later and in the core lecture Database Systems

IMDb (Part 1)

Question 2

How are links between these data modelled and stored in IMDb?

Modelling: through foreign key relationships in the relational model
Data storage: Don't worry... (see physical data independence)

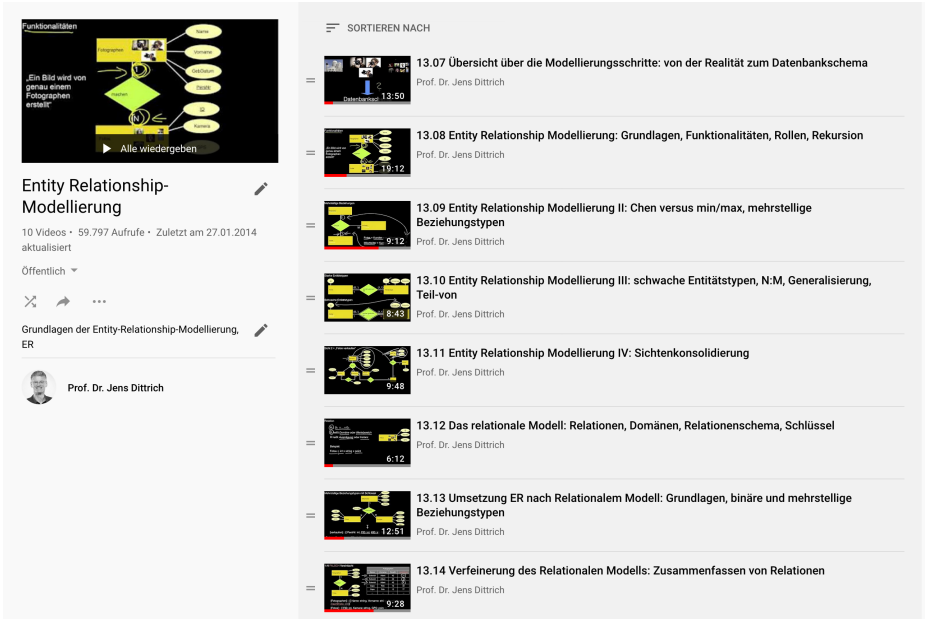
next week:

Question 3

How do we query this data?

...

Further Material and Literature (in German and English)



- Youtube videos by Prof. Jens Dittrich on ER modelling with other examples in German
- Video on ER basics with other examples in English
- Chapters 2&3 in Kemper&Eickler in German
- Chapters 3–5 in Elmasri&Navathe in German or in English
- Big Data Engineering Lecture recordings from 2022 in German
- Recommended literature for this lecture (Semesterapparat)