Data and Process Modelling

3. Object-Role Modeling - CSDP Step 4

Marco Montali

KRDB Research Centre for Knowledge and Data
Faculty of Computer Science
Free University of Bozen-Bolzano

Uniqueness Constraints

CSDP Step 4

Add uniqueness constraints and check the arity of fact types.

1. Model uniqueness constraints (UCs): each base fact type must be assigned at least one UC.
   - UC: at most one fact of a certain type is allowed.
     (**`Each Person has at most one Weight`**).
   - Identify keys for the fact types.
2. Use UCs to evaluate the arity of fact types.
   - Uniqueness check to determine if a fact type is elementary or to be split.

Uniqueness Constraints

CSDP Step 4

Add uniqueness constraints and check the arity of fact types.

1. Model uniqueness constraints (UCs): each base fact type must be assigned at least one UC.
   - UC: at most one fact of a certain type is allowed. (**`Each Person has at most one Weight`**).
   - Identify keys for the fact types.
2. Use UCs to evaluate the arity of fact types.
   - Uniqueness check to determine if a fact type is elementary or to be split.

Remember: fact types like `Person has Weight` are snapshot fact types: instances belong to a single database state.

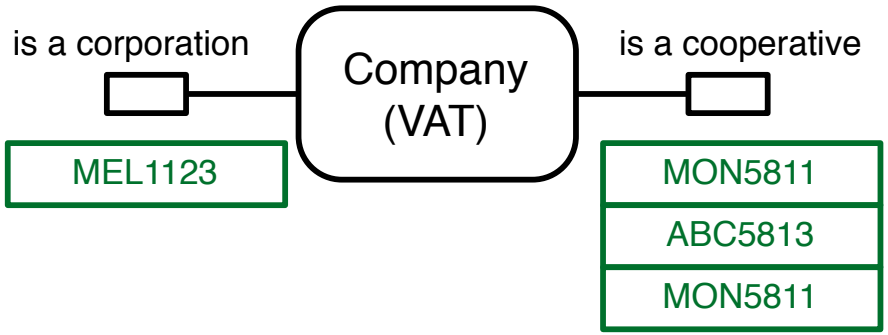
- Historical fact types can be modeled by explicitly referring to time: addition of a temporal role (`Person had Weight on Date`).

UC and Unary Fact Type

What about UC of a unary fact type? Which possibilities do we have?

UC and Unary Fact Type

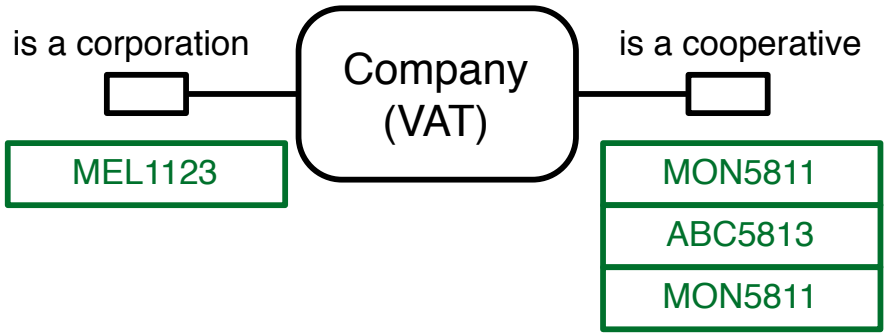
What about UC of a unary fact type? Which possibilities do we have?



UC and Unary Fact Type

What about UC of a unary fact type? Which possibilities do we have?

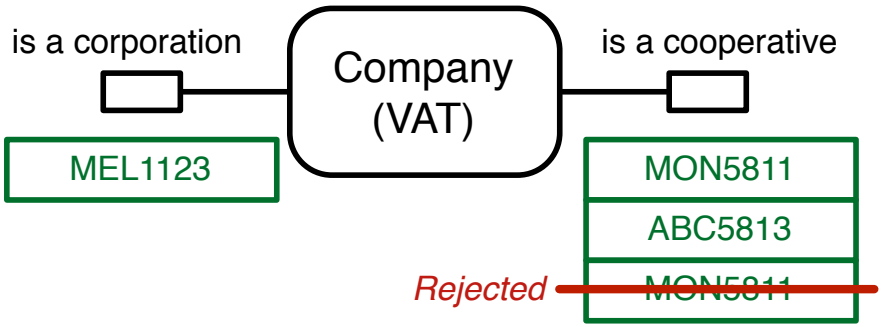
- Remember: the population of an information base is a **set** of individuals.
- Redundancy is sometimes accepted in the database, but *never* for elementary facts in the conceptual information base.



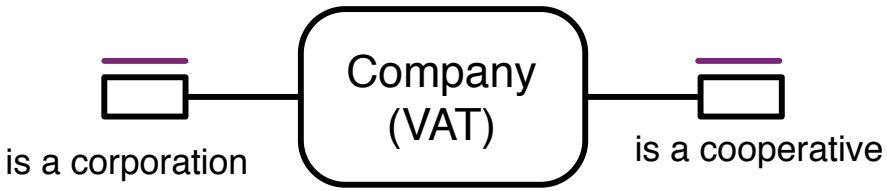
UC and Unary Fact Type

What about UC of a unary fact type? Which possibilities do we have?

- Remember: the population of an information base is a **set** of individuals.
- Redundancy is sometimes accepted in the database, but *never* for elementary facts in the conceptual information base.



- No choice for unary fact types: every unary fact type is (implicitly) associated to an UC.
- UC represented as a bar above/below the single role: simple UC.



UC and Binary Fact Type

How many possibilities?

UC and Binary Fact Type

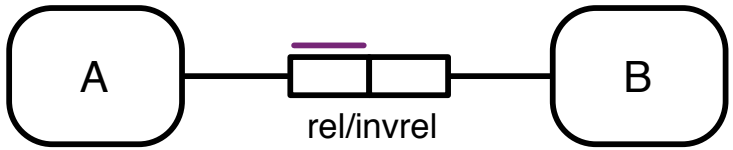
How many possibilities?

1. Many-to-one (n:1): each `A` in relation `rel` with **at most one** `B`; each `B` in relation `rel` with **many (0+)** `A`s. (1 simple UC)
2. One-to-many (1:n): each `A` in relation `rel` with **many (0+)** `B`s; each `B` in relation `rel` with **at most one** `A`. (1 simple UC)
3. One-to-one (1:1): each `A` in relation `rel` with **at most one** `B`, and **vice versa**. (2 simple UCs)
4. Many-to-many (n:m): each `A` in relation `rel` with **many (0+)** `B`, and **vice versa**. (1 composite UC)

UC and Binary Fact Type

How many possibilities?

1. Many-to-one (n:1): each `A` in relation `rel` with **at most one** `B`; each `B` in relation `rel` with **many (0+)** `A`s. (1 simple UC)
   - Each entry in the first role's fact column is unique.
   - Entries in the second role's fact column may be repeated.
   - `rel` is a partial function of `A`.

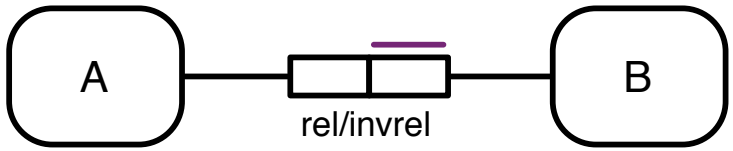


2. One-to-many (1:n): each `A` in relation `rel` with **many (0+)** `B`s; each `B` in relation `rel` with **at most one** `A`. (1 simple UC)
3. One-to-one (1:1): each `A` in relation `rel` with **at most one** `B`, and **vice versa**. (2 simple UCs)
4. Many-to-many (n:m): each `A` in relation `rel` with **many (0+)** `B`, and **vice versa**. (1 composite UC)

UC and Binary Fact Type

How many possibilities?

1. Many-to-one (n:1): each `A` in relation `rel` with **at most one** `B`; each `B` in relation `rel` with **many (0+)** `A`s. (1 simple UC)
2. One-to-many (1:n): each `A` in relation `rel` with **many (0+)** `B`s; each `B` in relation `rel` with **at most one** `A`. (1 simple UC)
   - Each entry in the second role's fact column is unique.
   - Entries in the first role's fact column may be repeated.
   - `invrel` is a partial function of `B`.

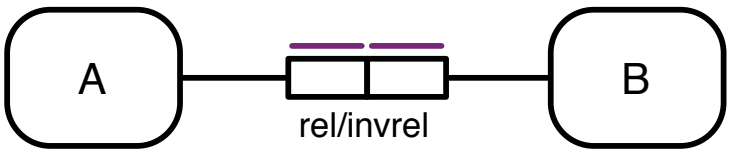


3. One-to-one (1:1): each `A` in relation `rel` with **at most one** `B`, and **vice versa**. (2 simple UCs)
4. Many-to-many (n:m): each `A` in relation `rel` with **many (0+)** `B`, and **vice versa**. (1 composite UC)

UC and Binary Fact Type

How many possibilities?

1. Many-to-one (n:1): each `A` in relation `rel` with **at most one** `B`; each `B` in relation `rel` with **many (0+)** `A`s. (1 simple UC)
2. One-to-many (1:n): each `A` in relation `rel` with **many (0+)** `B`s; each `B` in relation `rel` with **at most one** `A`. (1 simple UC)
3. One-to-one (1:1): each `A` in relation `rel` with **at most one** `B`, and **vice versa**. (2 simple UCs)
   - `invrel` is a the *inverse function* of `rel`: `invrel(rel(A)) = A`.
   - Used for reference types (abbreviated for the preferred reference mode).

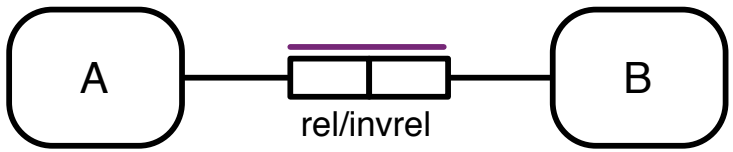


4. Many-to-many (n:m): each `A` in relation `rel` with **many (0+)** `B`, and **vice versa**. (1 composite UC)

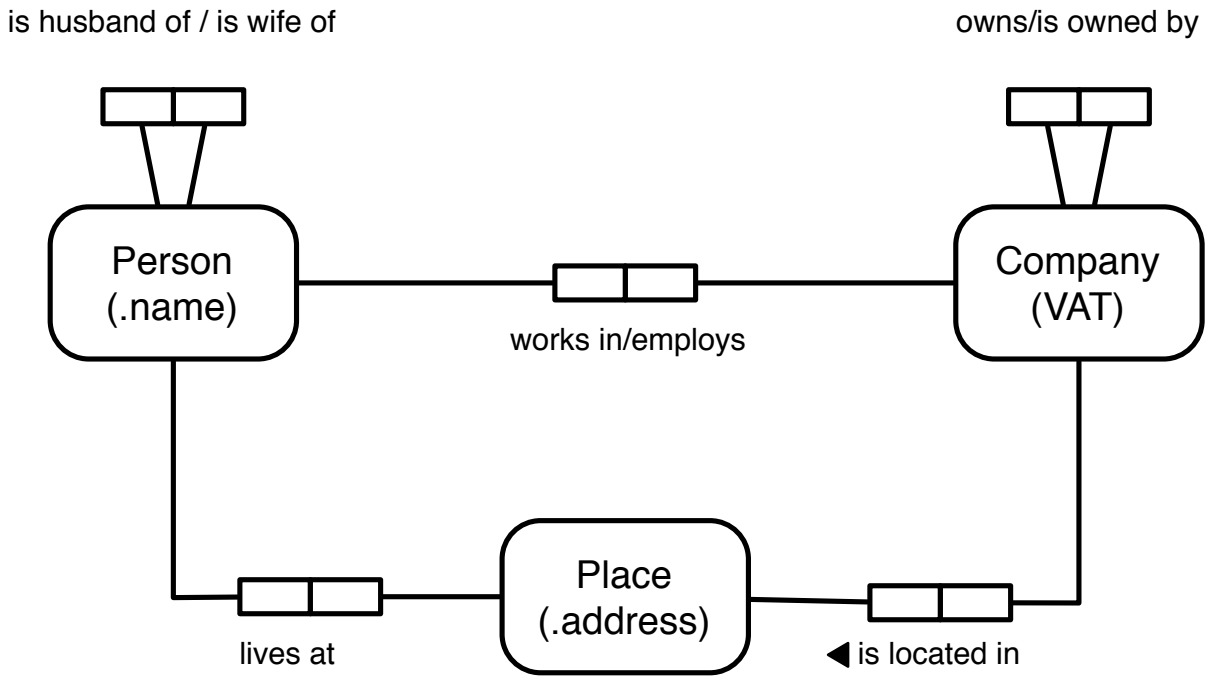
UC and Binary Fact Type

How many possibilities?

1. Many-to-one (n:1): each `A` in relation `rel` with **at most one** `B`; each `B` in relation `rel` with **many (0+)** `A`s. (1 simple UC)
2. One-to-many (1:n): each `A` in relation `rel` with **many (0+)** `B`s; each `B` in relation `rel` with **at most one** `A`. (1 simple UC)
3. One-to-one (1:1): each `A` in relation `rel` with **at most one** `B`, and **vice versa**. (2 simple UCs)
4. Many-to-many (n:m): each `A` in relation `rel` with **many (0+)** `B`, and **vice versa**. (1 composite UC)
   - Spanning uniqueness constraint.
   - Always true (set semantics).
   - Most general: $(1.) \vee (2.) \vee (3.) \rightarrow (4.)$.
   - Verify with domain experts if bags are supported $\rightarrow$ ternary relation
     - ★ Temporization is a common case.

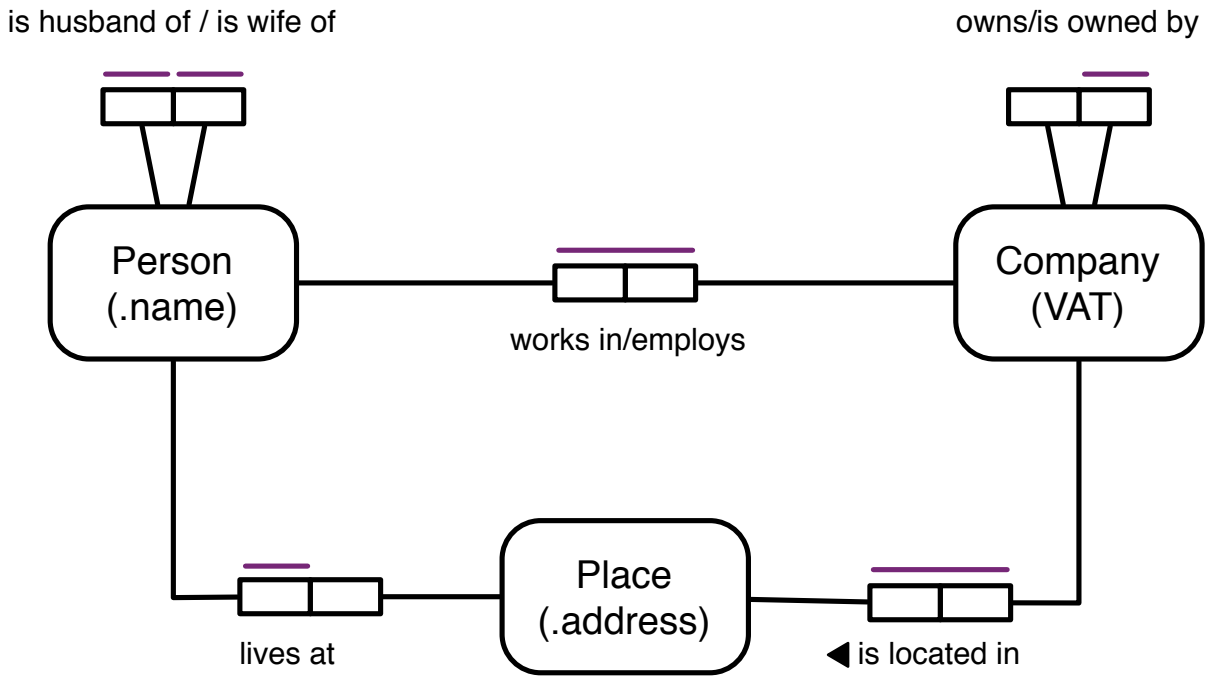


Constraints Elicitation



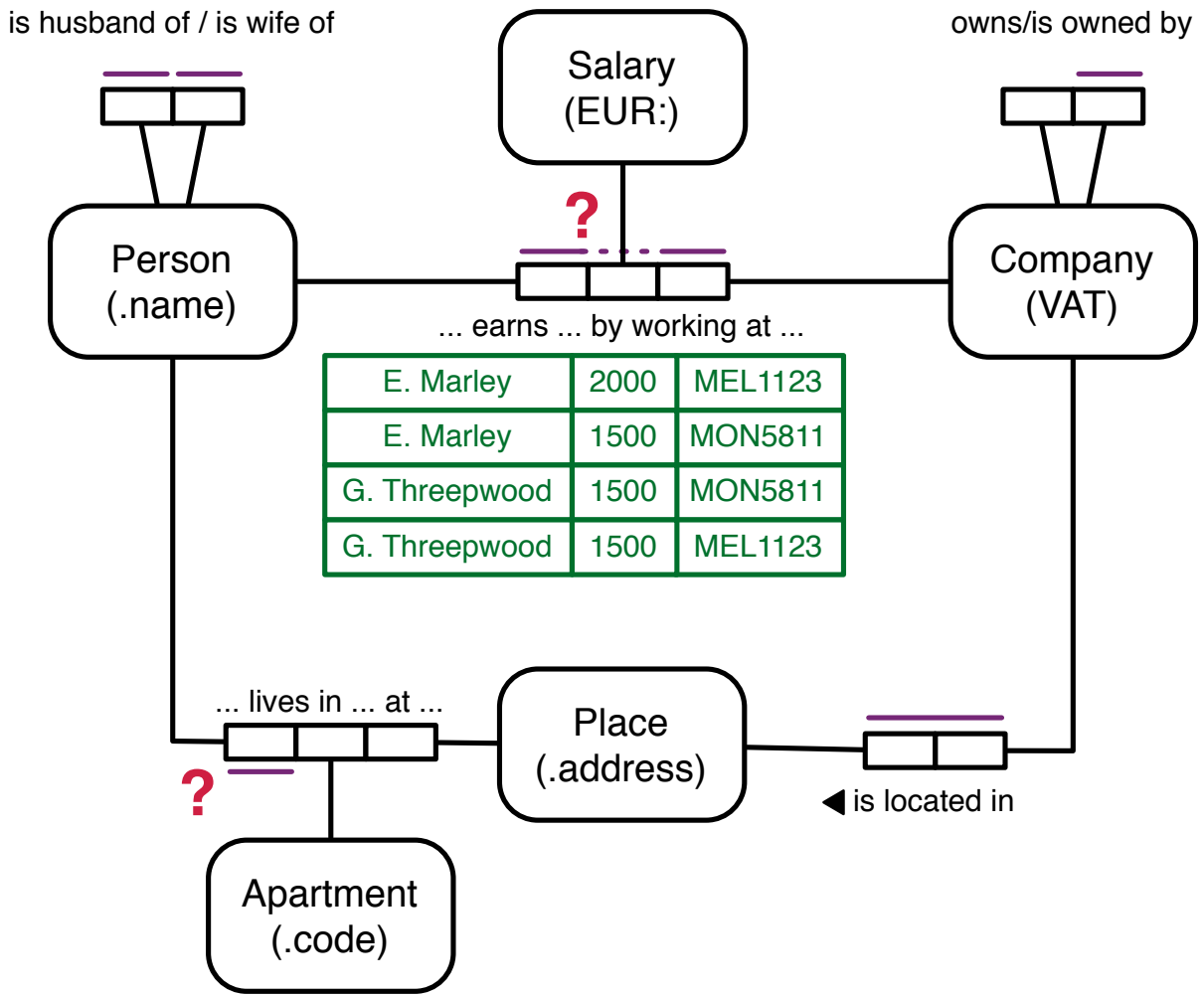
- Interaction with domain experts.
- Question each constraint in English, eliciting **counter-examples**.
  - Is it possible for a Person to live in more than one Place?
    Is it possible for a Company to be owned by more than one Company?
- Remember: a conceptual model is only an *approximation* of reality!
  - UCs should be at least as strong as those that apply in the real world.

Constraints Elicitation



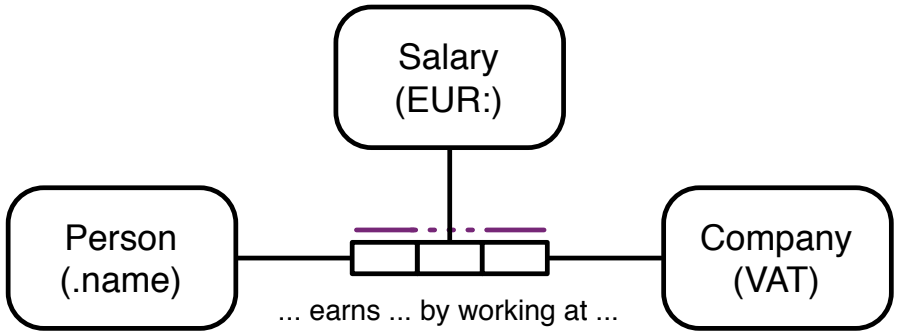
- Interaction with domain experts.
- Question each constraint in English, eliciting **counter-examples**.
  - Is it possible for a Person to live in more than one Place?
    Is it possible for a Company to be owned by more than one Company?
- Remember: a conceptual model is only an *approximation* of reality!
  - UCs should be at least as strong as those that apply in the real world.

UC and Ternary Fact Types



UC and Ternary Fact Types

- Deep case analysis using the available data (incomplete knowledge).
- Does a constraint make sense?
  - Very unlikely that Company+Salary univocally determines a Person.

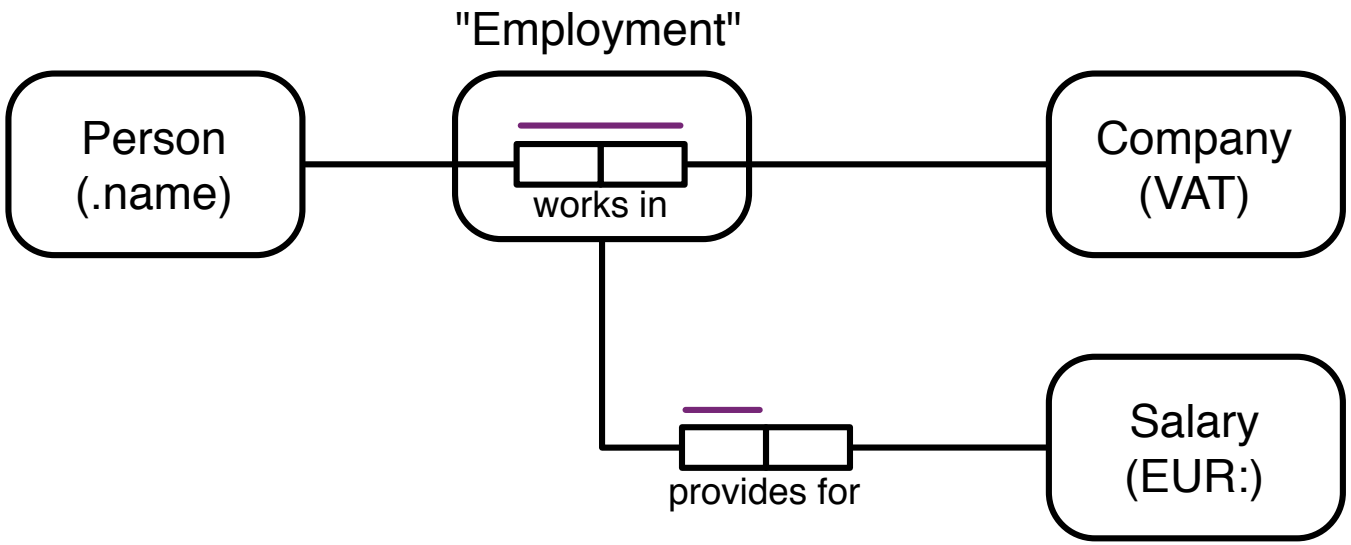


E. Marley	2000	MEL1123
E. Marley	1500	MON5811
G. Threepwood	1500	MON5811
G. Threepwood	1500	MEL1123

- Usage of divided constraint bar to write UCs over non-contiguous roles.

Ternary vs Objectified Association

Corresponding objectified diagram
(equivalent only if `provides for` is mandatory for `Employment`).



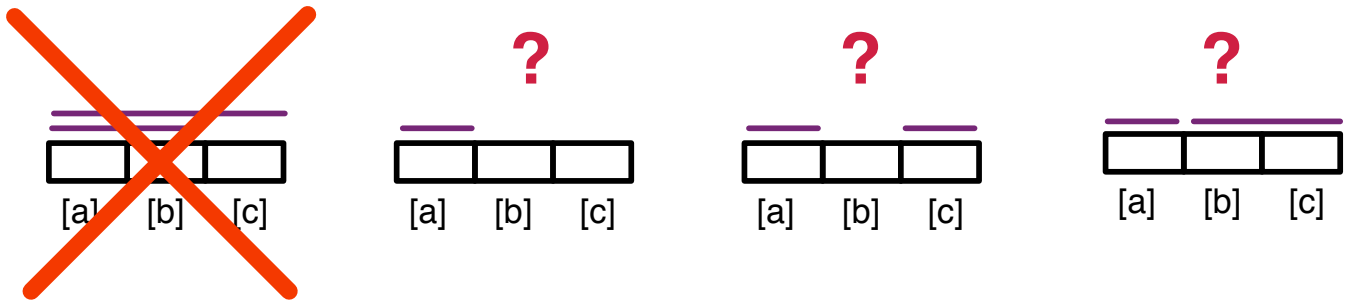
- Objectified associations must have a spanning UC (objectification introduces a reference to a combination of objects).
- If this is not true, refactor around the entity type(s) subject to the UC.
- Simple constraint on `provides for` predicate: each Employment (i.e., each pair (Person, Company)) is associated to at most one Salary.

UC and Ternary Fact Types: Possibilities

- Single UCs.

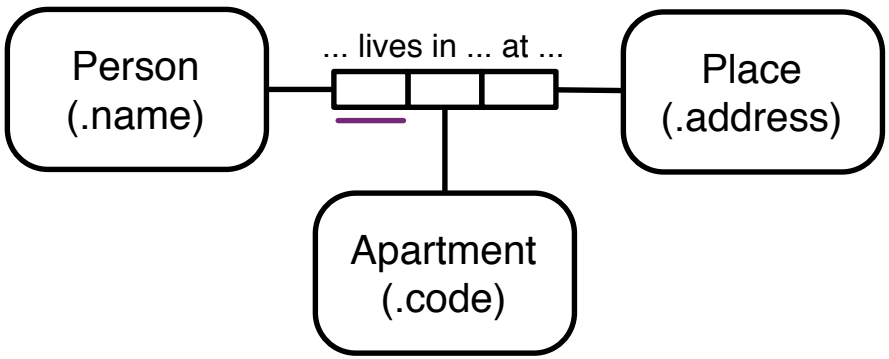
- Combined UCs.

- What about other possibilities?



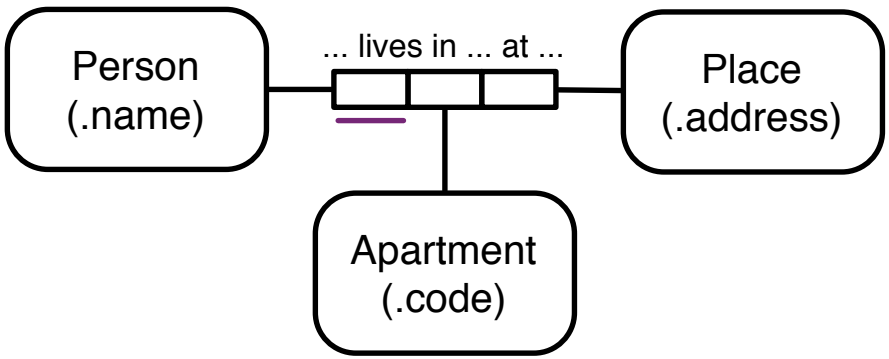
UCs and Elementary Facts

E. Marley	123	Palace Street 1, Meleé Island
G. Threepwood	123	Palace Street 1, Meleé Island
S. Stanman	431	Central Square 12, Booty Island



UCs and Elementary Facts

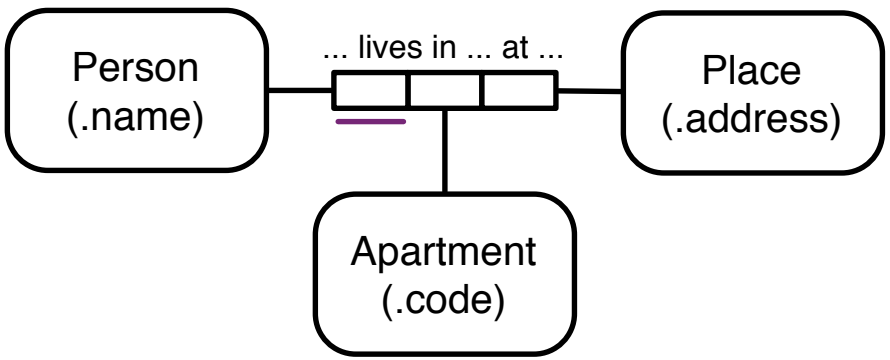
E. Marley	123	Palace Street 1, Meleé Island
G. Threepwood	123	Palace Street 1, Meleé Island
S. Stanman	431	Central Square 12, Booty Island



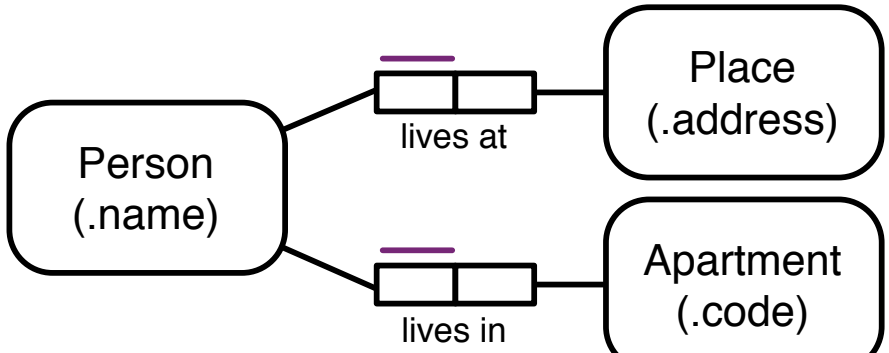
- Person acts as a pivot: determines both Apartment and Place.
- The UC *reveals* that the predicate is **non-elementary**.

UCs and Elementary Facts

E. Marley	123	Palace Street 1, Meleé Island
G. Threepwood	123	Palace Street 1, Meleé Island
S. Stanman	431	Central Square 12, Booty Island



E. Marley	Palace Street 1, Meleé Island
G. Threepwood	Palace Street 1, Meleé Island
S. Stanman	Central Square 12, Booty Island



E. Marley	123
G. Threepwood	123
S. Stanman	431

- Person acts as a pivot: determines both Apartment and Place.
- The UC *reveals* that the predicate is **non-elementary**.

Handling Non-Elementary Fact Types

Identification of non-elementary fact types in the model $\rightarrow$ decomposition.

- Key length check: UCs to identify predicates with too many roles.
  - *Sufficient* condition for splitting.
- Projection-join check: split and recombine information checking if there is information-loss.
  - Refine the sufficient condition for key length check.

Key Length Check

- Key: minimal combination of roles spanned by an UC.
  - Simple key: spans one role only.
- Predicates of wrong arity.
  - Too long: non-elementary fact type $\rightarrow$ to be split.
    - ★ Person lives in Apartment at Place
      $\rightarrow$ Person lives in Apartment; Person lives at Place.
  - Too short: information-loss $\rightarrow$ recombination.
    - ★ Lecturer teaches Course; Lecturer teaches at Faculty
      $\rightarrow$ Lecturer teaches Course at Faculty.
- Major issue: too long predicates.

Key Length Check

- Key: minimal combination of roles spanned by an UC.
  - Simple key: spans one role only.
- Predicates of wrong arity.
  - Too long: non-elementary fact type $\rightarrow$ to be split.
    - ★ Person lives in Apartment at Place
      $\rightarrow$ Person lives in Apartment; Person lives at Place.
  - Too short: information-loss $\rightarrow$ recombination.
    - ★ Lecturer teaches Course; Lecturer teaches at Faculty
      $\rightarrow$ Lecturer teaches Course at Faculty.
- Major issue: too long predicates.

n-1 rule

Each n-ary fact type has a key length of at least n-1.

- Elementary fact type of arity n:
  1. has exactly one key of length n;
  2. has one or more keys of length n-1.
- Ternary fact type to be split if it has a simple key $\rightarrow$ split on the key.

Functional Dependencies

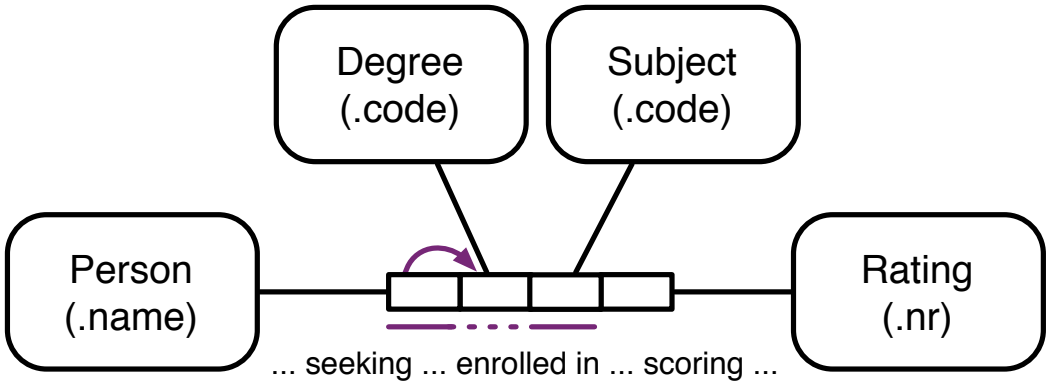
- Functional dependencies help in the decision of how to split.

Functional Dependency

Given a combination of columns X and a column Y in a fact table, Y functionally depends on X ($X \rightarrow Y$) if for each value of X there is at most one value of Y.

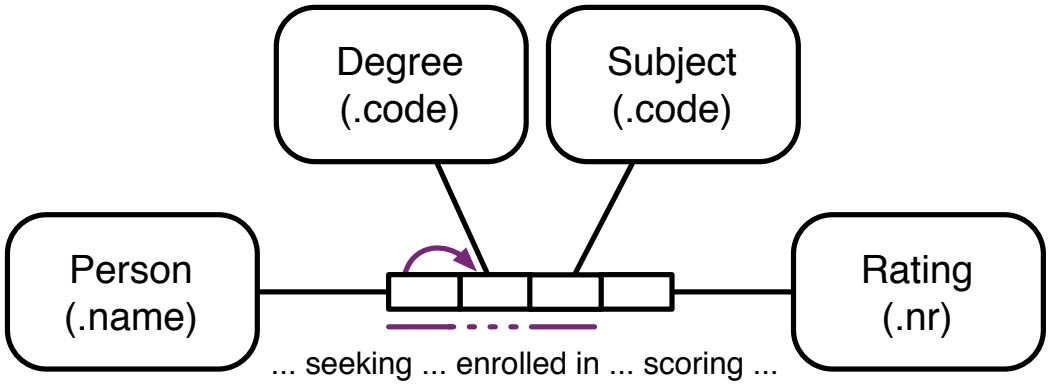
- Correct ORM schema: all FDs captured by UCs.
- $X \rightarrow Y$: there is a many-to-one relationship between X and Y.
- Suppose relation of arity n has key of size $< n - 1$.
  - Then there are at least two columns that functionally depend on key.
  - Split can be done by pairing each FD source with the corresponding target.
- Difficult to be exhaustively spotted: they are many and each one requires verbalization with the domain experts.
  - To be combined with human knowledge about the UoD.

FDs and Decomposition



- FDs shown only in a "temporary" model.
- Is the model correct?

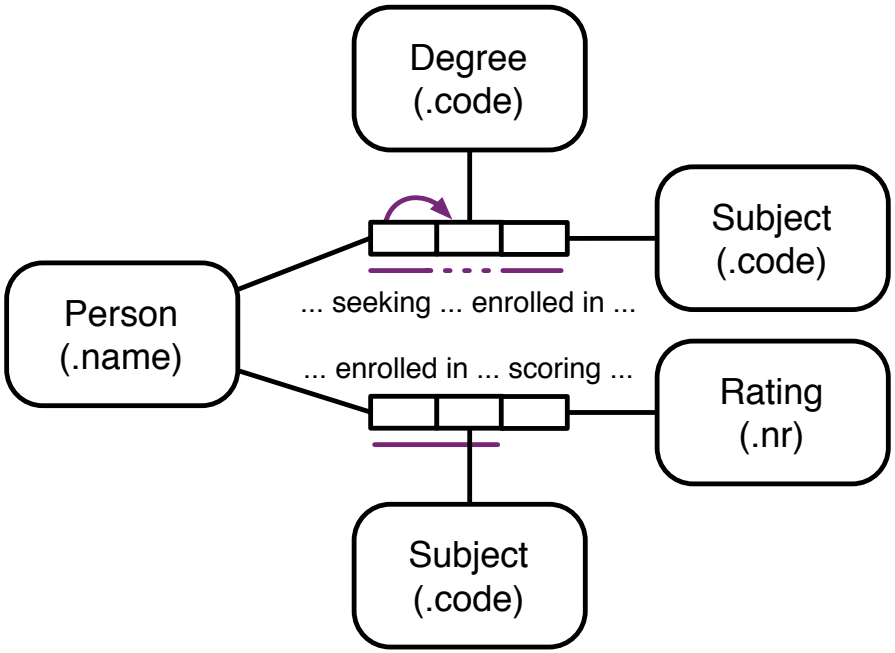
FDs and Decomposition



- FDs shown only in a "temporary" model.
- Is the model correct? Violation of the n-1 rule!

FDs and Decomposition

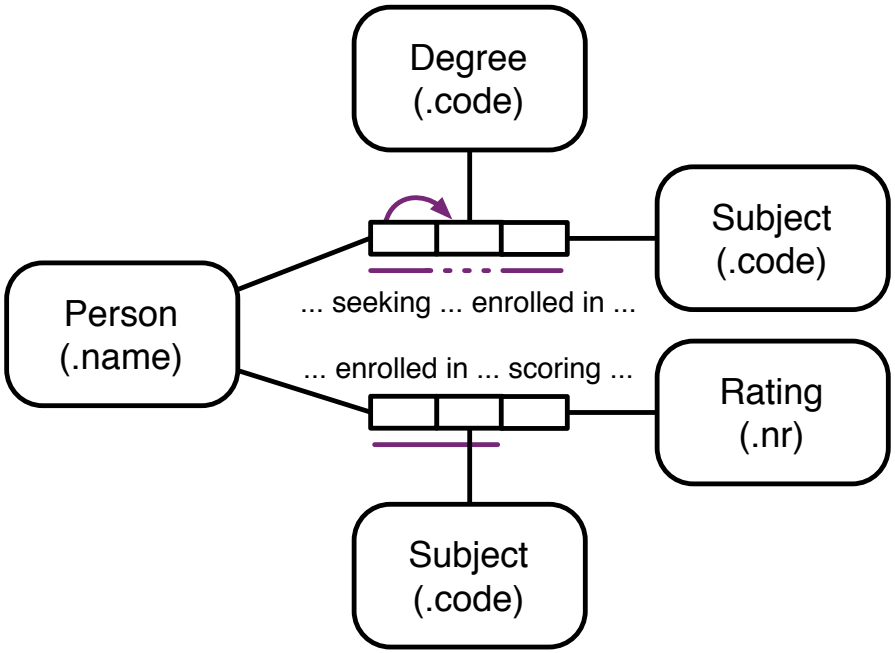
Decomposition using the UC as a pivot.



- Is the model correct?

FDs and Decomposition

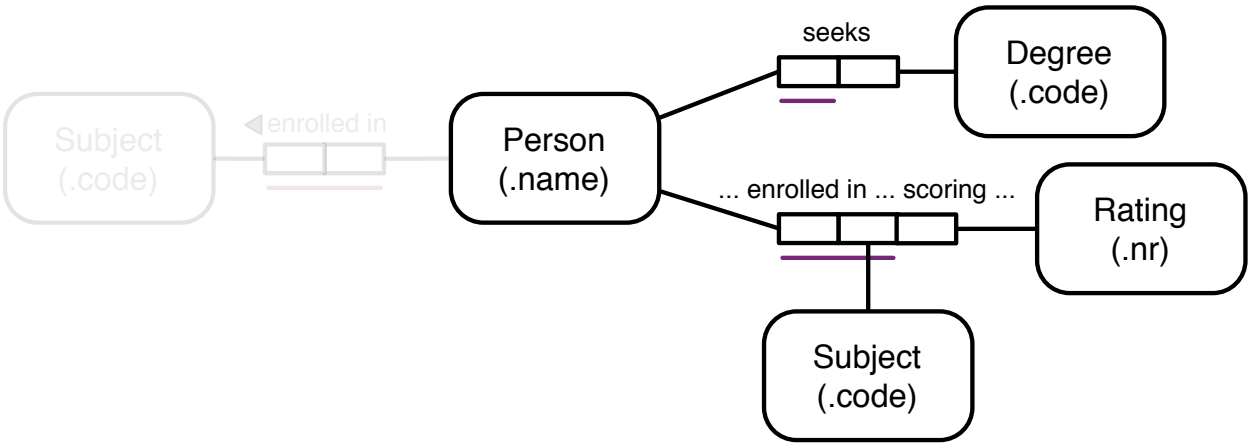
Decomposition using the UC as a pivot.



- Is the model correct? Relation is non-elementary due to the FD.
- The n-1 rule is a *sufficient* condition for splitting, but *not a necessary* one.

FDs and Decomposition

Decomposition using the source of FD as a pivot.



- Part of the decomposition is redundant $\rightarrow$ not included.

Conceptual Projection

Extraction of a portion of a fact table, obtained by maintaining only the desired columns.

- Set semantics: no repetitions inside the filtered table.
- Notation: $T[role_1,\ldots,role_n]$.
  - Corresponds to $\pi_{role_1,\ldots,role_n}(T)$ in relational algebra.
- Example:

T

EMPLOYEE	SALARY	COMPANY
E. Marley	2000	MEL1123
E. Marley	1500	MON5811
G. Threepwood	1500	MON5811
G. Threepwood	1500	MEL1123

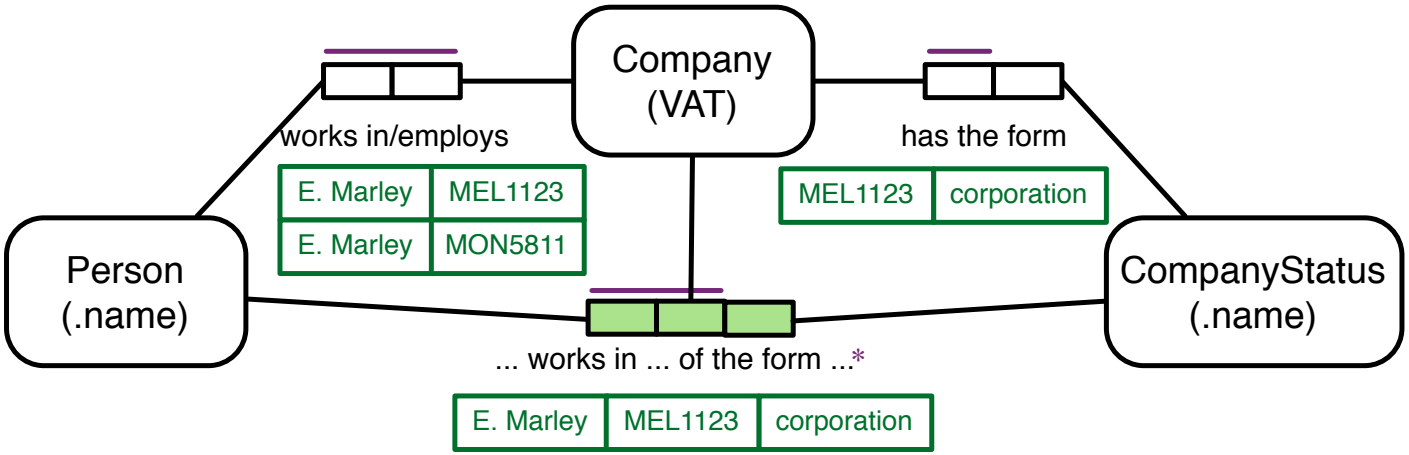
T[salary,company]

SALARY	COMPANY
2000	MEL1123
1500	MON5811
1500	MEL1123

Conceptual Join

Traversal of the conceptual schema from one fact type to another, passing through an object type.

Intuition for the navigation: when applying the conceptual join to concrete objects, the object of the join type is fixed.



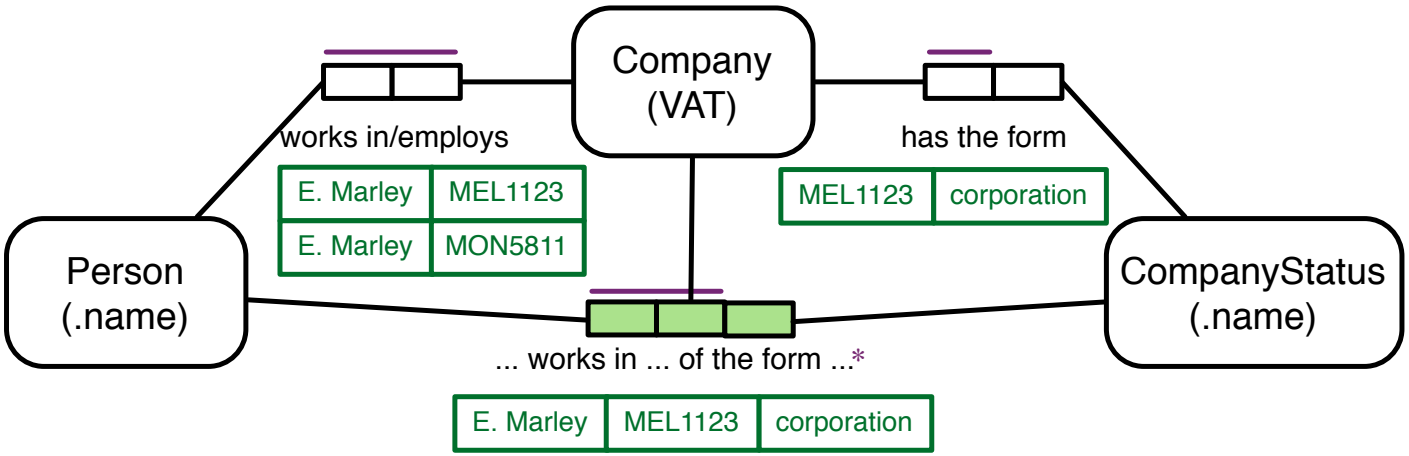

```
* define Person works in Company of the form CompanyStatus as
        Person works in Company that has the form CompanyStatus
```

The conceptual join gives raise to a *compound fact type*.

- obtained from the combination of different predicates by imposing **equivalence** among objects playing a certain role in them.

Types of Conceptual Join

- Conceptual inner join: join object must be the *same*.
- Left (right, full) outer join: also keep those facts for which the join object only appears in just the left (right, one of) fact table.
  - ? to denote the absence of a value (NULL).
  - Left outer join in the example: addition of `E. Marley MON5811 ?`.



* **define** Person works in Company of the form CompanyStatus **as**
Person works in Company **that** has the form CompanyStatus

Projection-Join Check

Tests whether a fact type is compound (hence splittable).

1. Provide a significant fact table for the fact type.
   - Must cover all the possible cases!
2. Split this table into two or more projections.
3. Recombine by conceptual (inner) join.
   - By construction no NULL entry.
4. The fact type is splittable in this way **if and only if** the result is the same as the original.

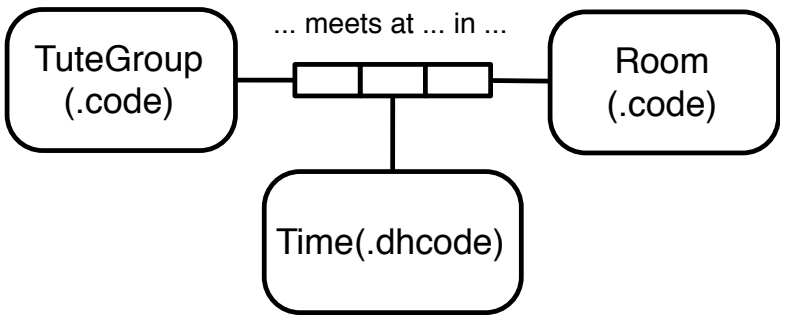
N.B.: usually having a significant fact table already supports the right choice.

Projection-Join Check

Suppose the fact table covers all possible cases.

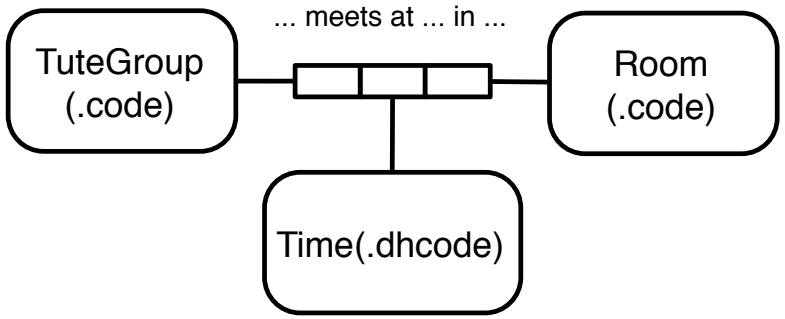
- Is this fact type splittable?

A	Mon. 3 p.m.	CS-718
B1	Tue. 2 p.m.	E-B18
C	Mon. 3 p.m.	E-B18



- And this version?

A	Mon. 3 p.m.	CS-718
B1	Tue. 2 p.m.	E-B18
B1	Wed. 2 p.m.	E-B15
C	Mon. 3 p.m.	E-B18

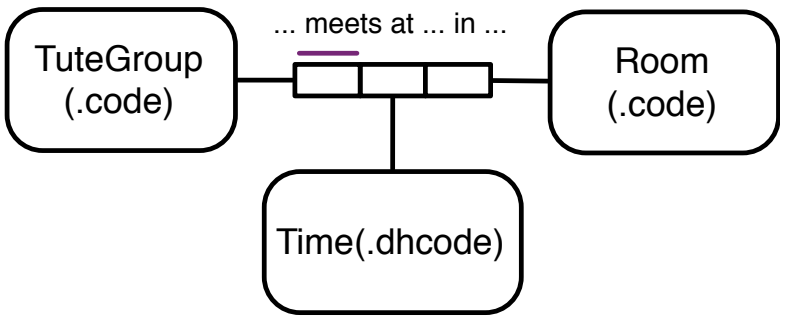


Projection-Join Check

Suppose the fact table covers all possible cases.

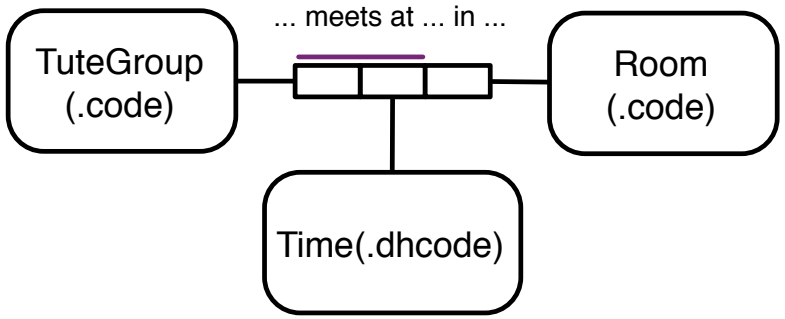
- Is this fact type splittable?

A	Mon. 3 p.m.	CS-718
B1	Tue. 2 p.m.	E-B18
C	Mon. 3 p.m.	E-B18



- And this version?

A	Mon. 3 p.m.	CS-718
B1	Tue. 2 p.m.	E-B18
B1	Wed. 2 p.m.	E-B15
C	Mon. 3 p.m.	E-B18

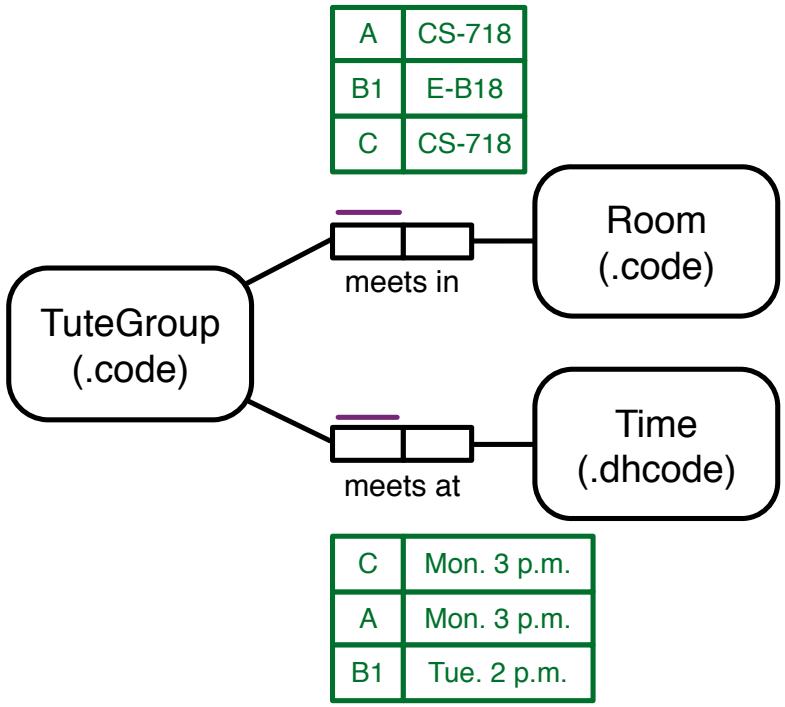


External Uniqueness Constraints

Apply to roles from different predicates.

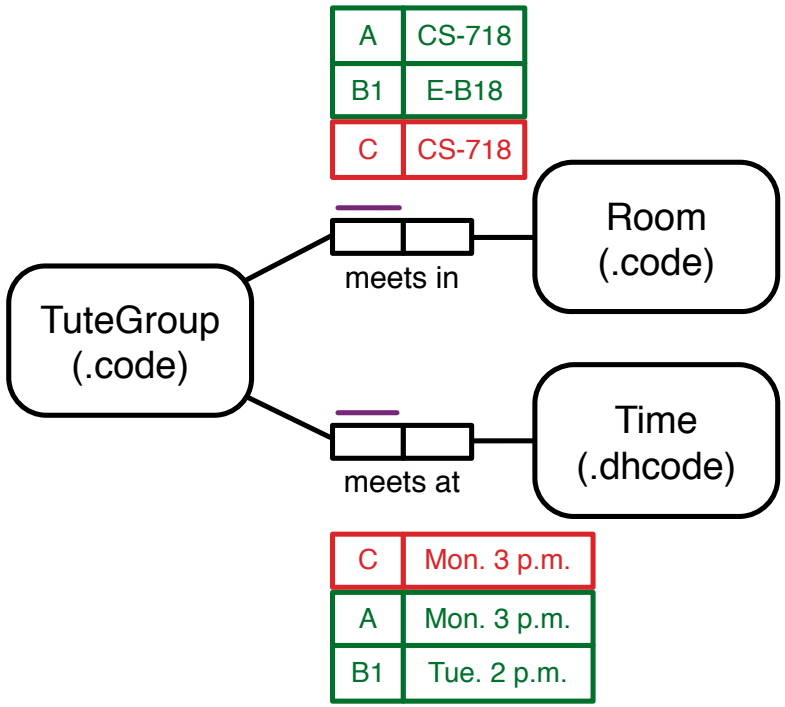
External Uniqueness Constraints

Apply to roles from different predicates.



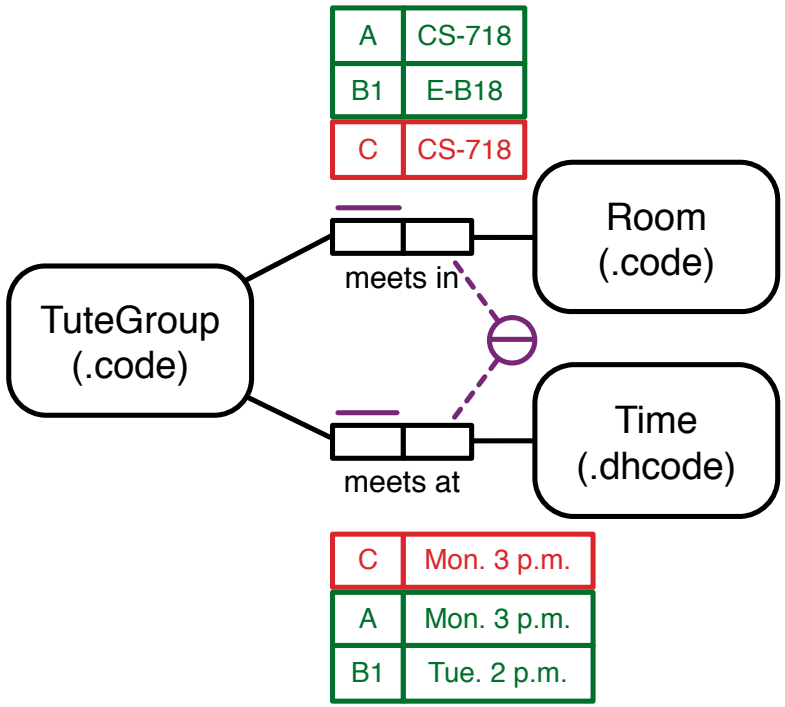
External Uniqueness Constraints

Apply to roles from different predicates.



External Uniqueness Constraints

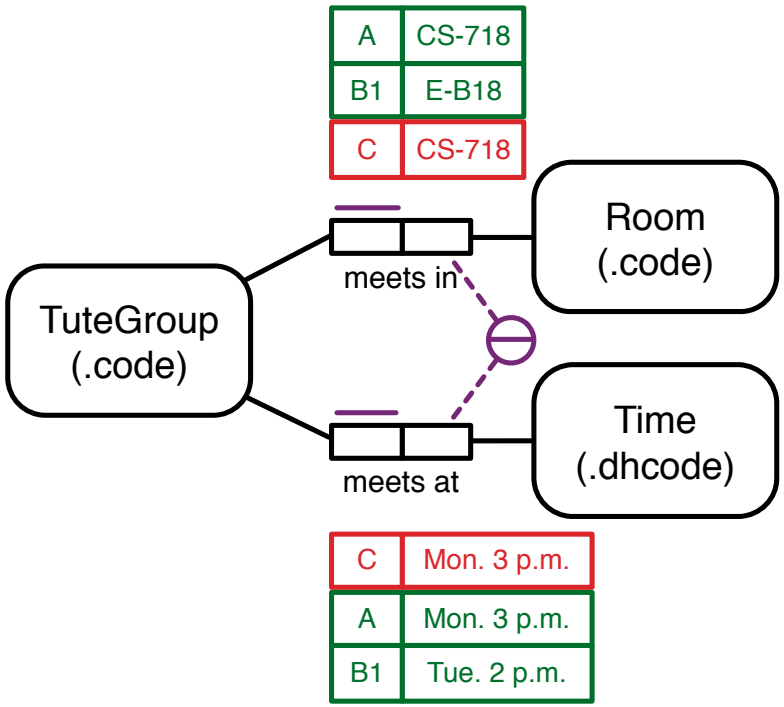
Apply to roles from different predicates.

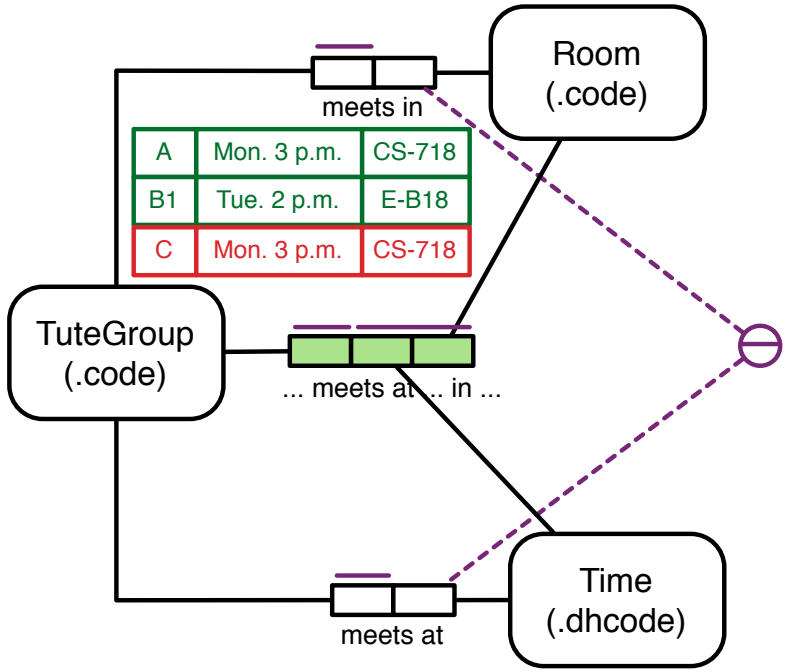


`Each combination of MeetingRoom and MeetingTime is paired with at most one TuteGroup`

External Uniqueness Constraints

Apply to roles from different predicates.

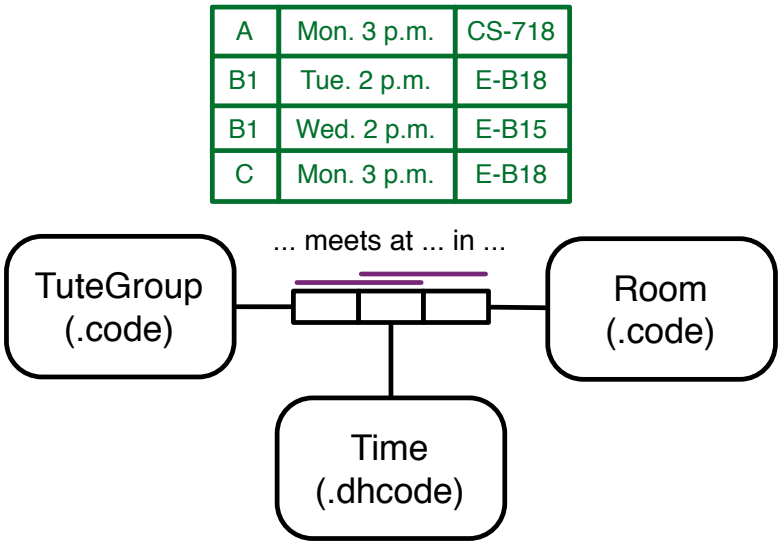


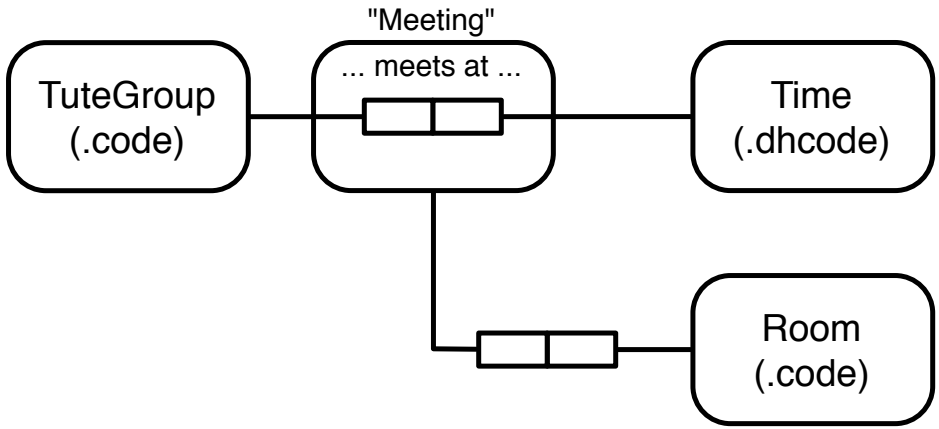


Each combination of MeetingRoom and MeetingTime is paired with at most one TuteGroup

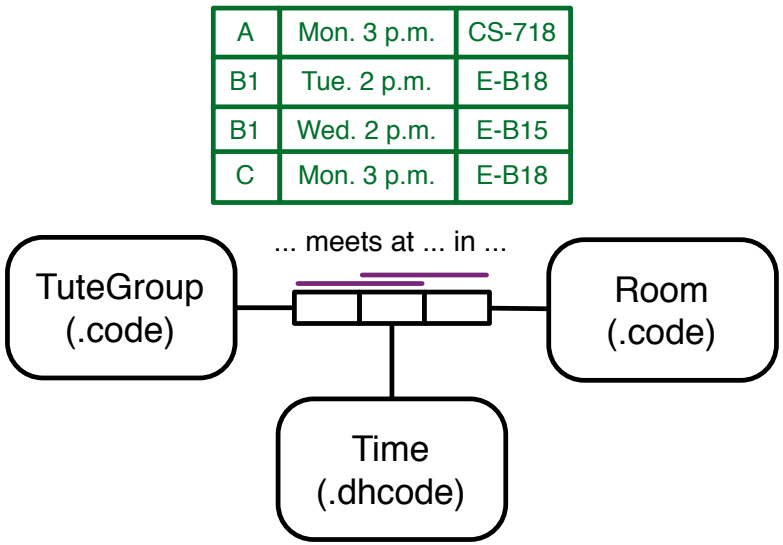
Each population of 'meets at' **join** 'meets in' has (Room,Time) unique

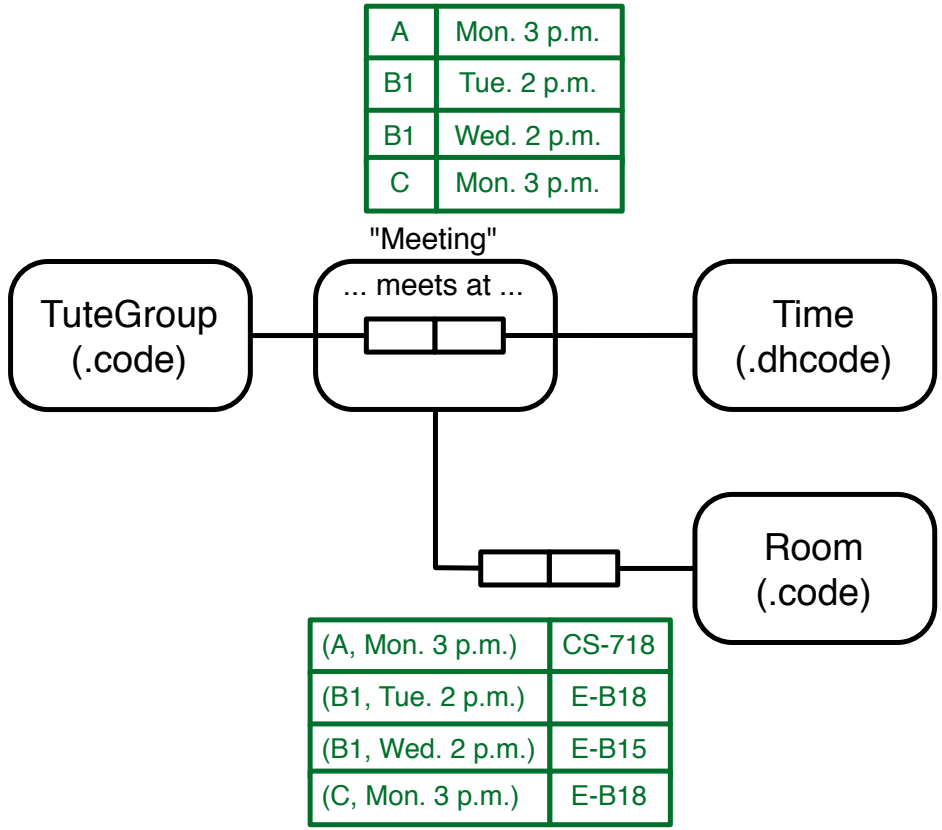
EUC and Objectification





EUC and Objectification





EUC and Objectification