Zawansowane Modelowanie i Analiza Systemów Informatycznych

(l- 4)

Polsko-Japońska Wyższa Szkoła Technik Komputerowych

Katedra Systemów Informacyjnych

2012

Overwiev

- **The CSDP steps 4-6 ,**

NOTE !!!!

This lecture covers LOTS of material – students' study will be necessary!!!!

The 7 steps of CSDP

Step 1: transform familiar information examples into elementary facts, and apply quality checks

Step 2: draw the fact types, and apply a population check

Step 3: check for entity types that should be combined, and note any arithmetic derivations

Step 4: add uniqueness constraints, arity of fact types, splitting of fact types.

Step 5: add mandatory role constraints, and check for logical derivations

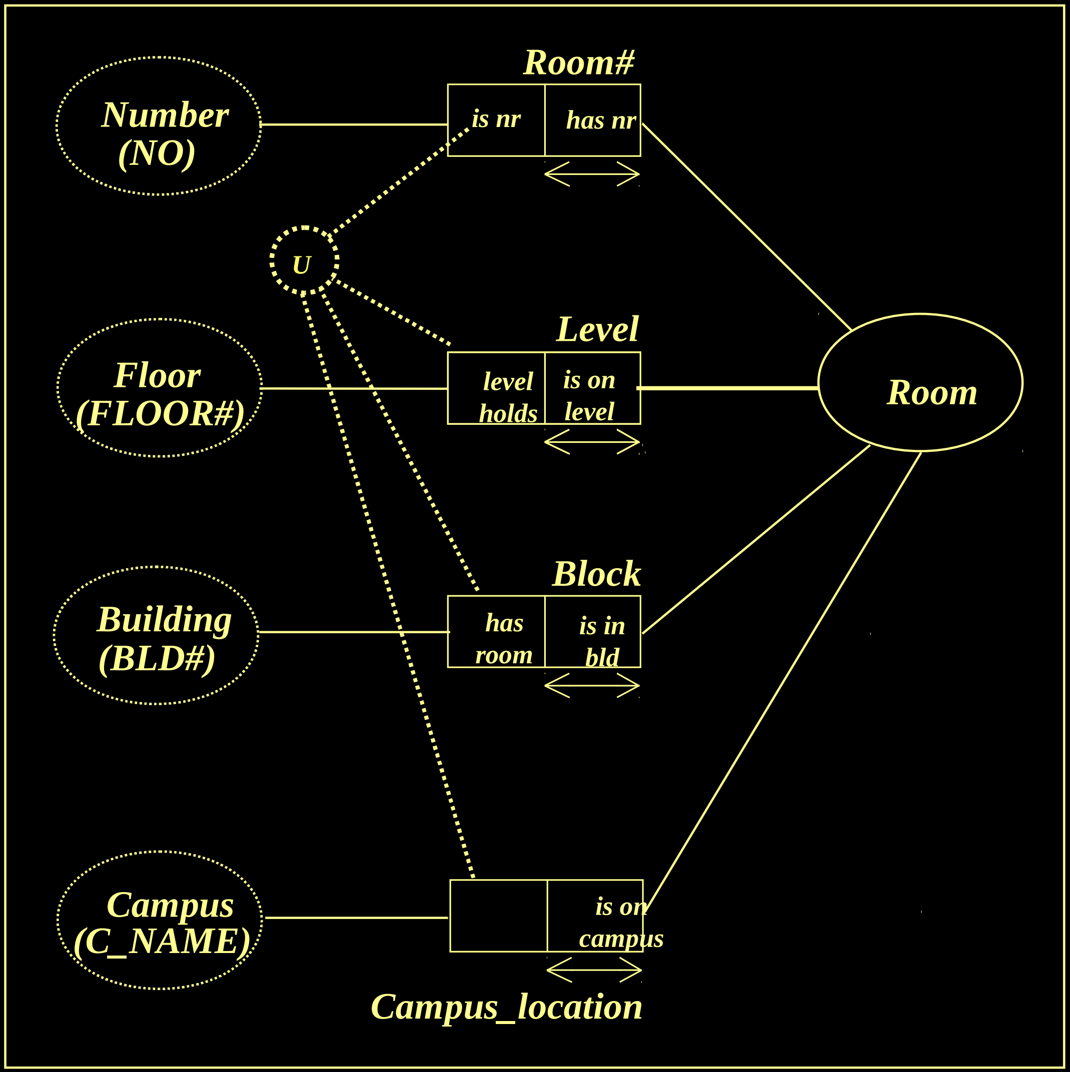
Step 6: add value, set comparison and subtyping constraints

Step 7: add other constraints and perform final quality checks (e.g., populating fact type instances)

Step 4 -Uniqueness constraints across several fact types

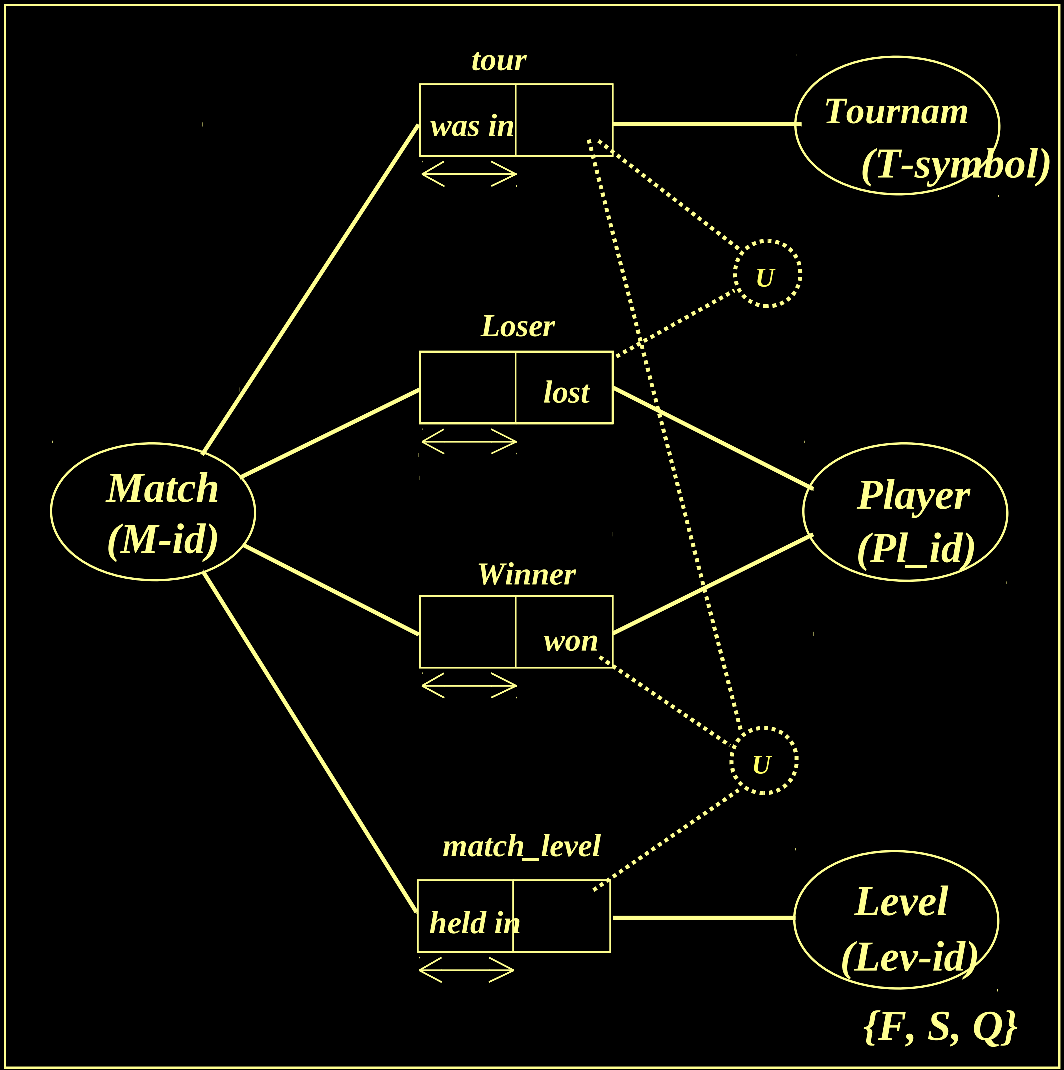
A room is identified by
- *Campus,*
- *Building,*
- *Level,*
- *Room number*

Example:
- *(Gardens, A, 3, 11)*



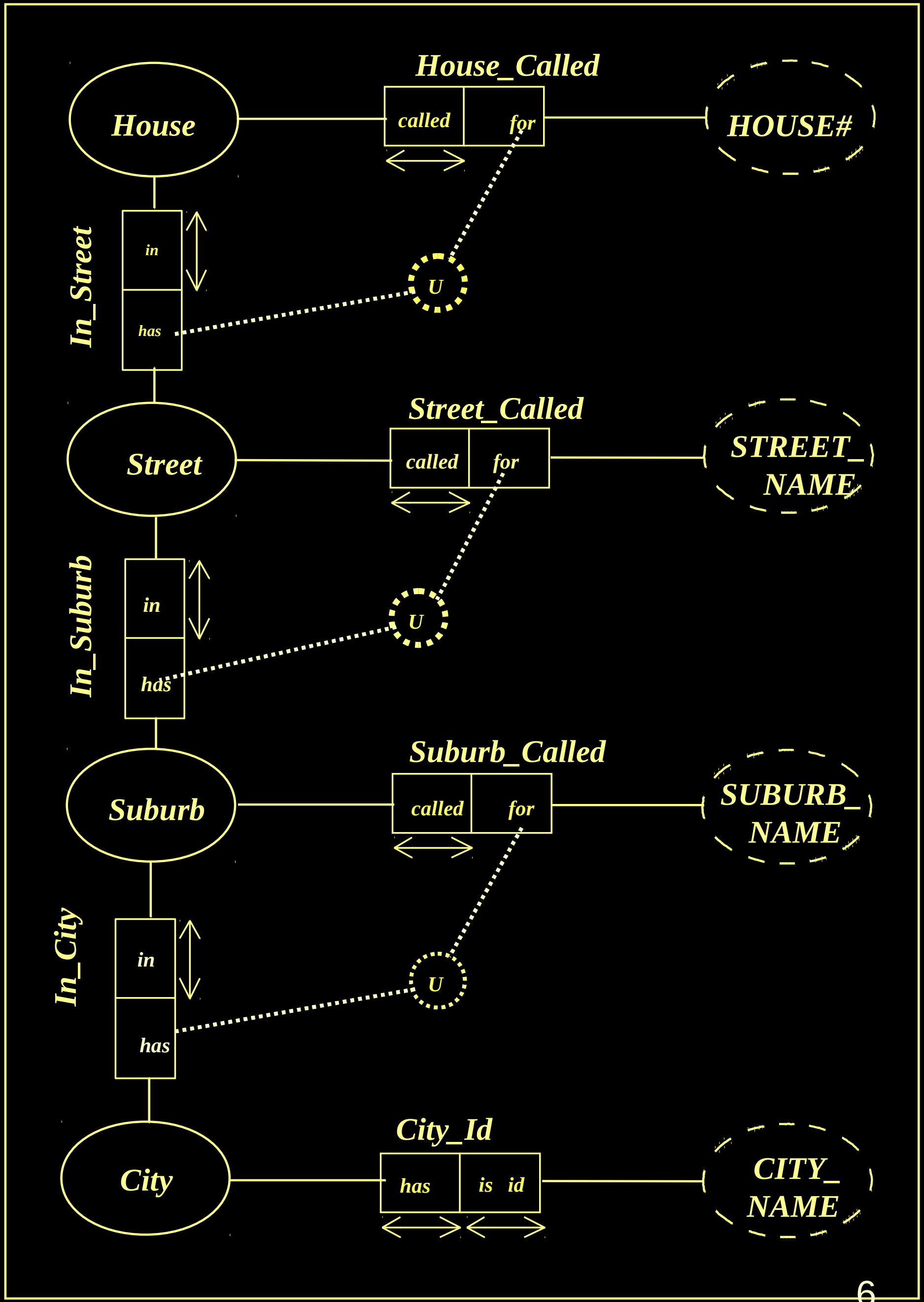
Uniqueness constraints across several fact types 'Tennis Tournaments'

M-id	Tour	Lev	Win	Los
001	Wi	Q	A	H
002	Wi	Q	B	G
003	Wi	Q	C	F
004	Wi	Q	D	E
005	Wi	S	A	D
006	Wi	S	B	C
007	Wi	F	A	B
008	AO	Q	B	D
009	AO	S	D	J
....				



F – final, S –semi-final, Q- quarter-final

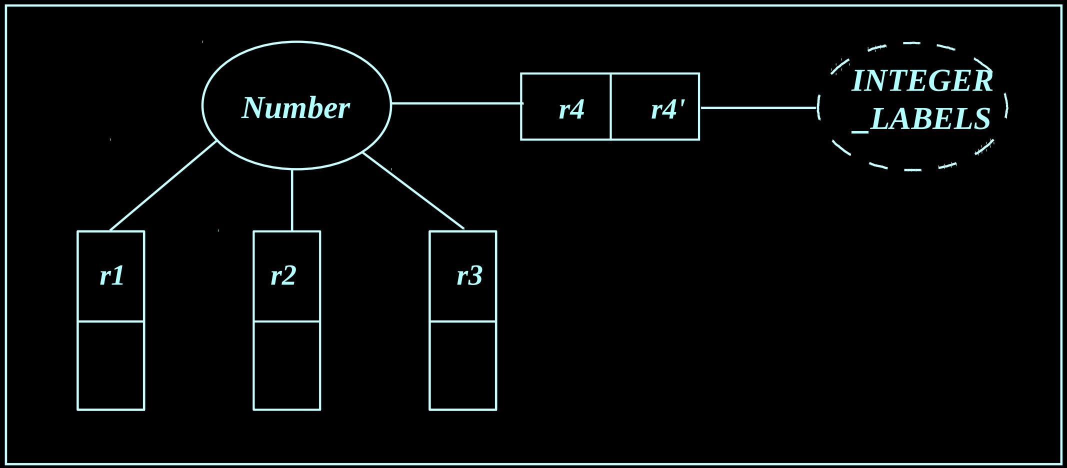
Hierarchical Structures for entity type identification



The distinction between entity types and label types:

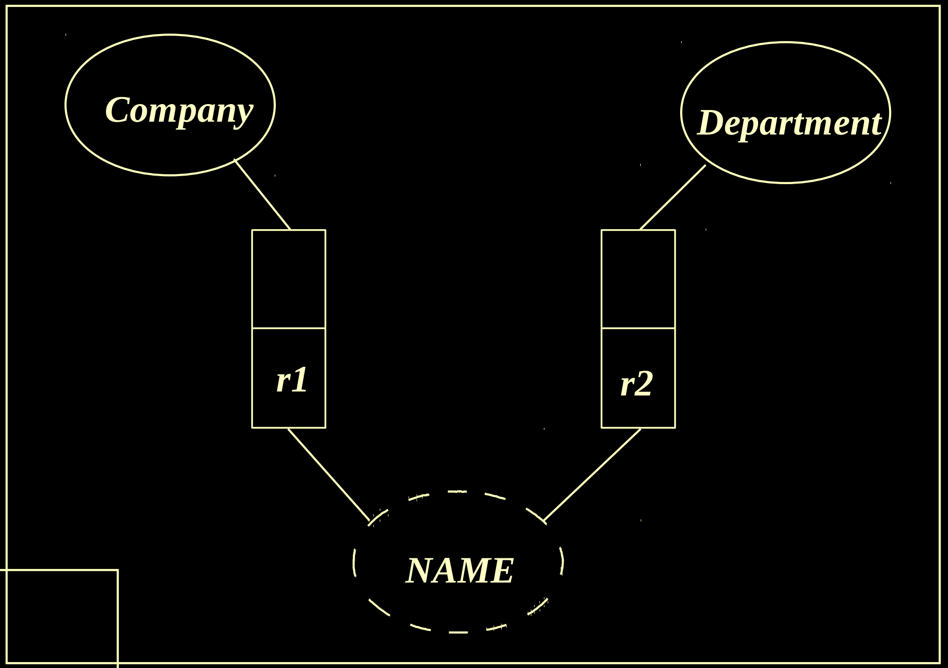
The distinguishing characteristic of a label type is that it is involved in <u>ONLY ONE ROLE</u>

Illustration:

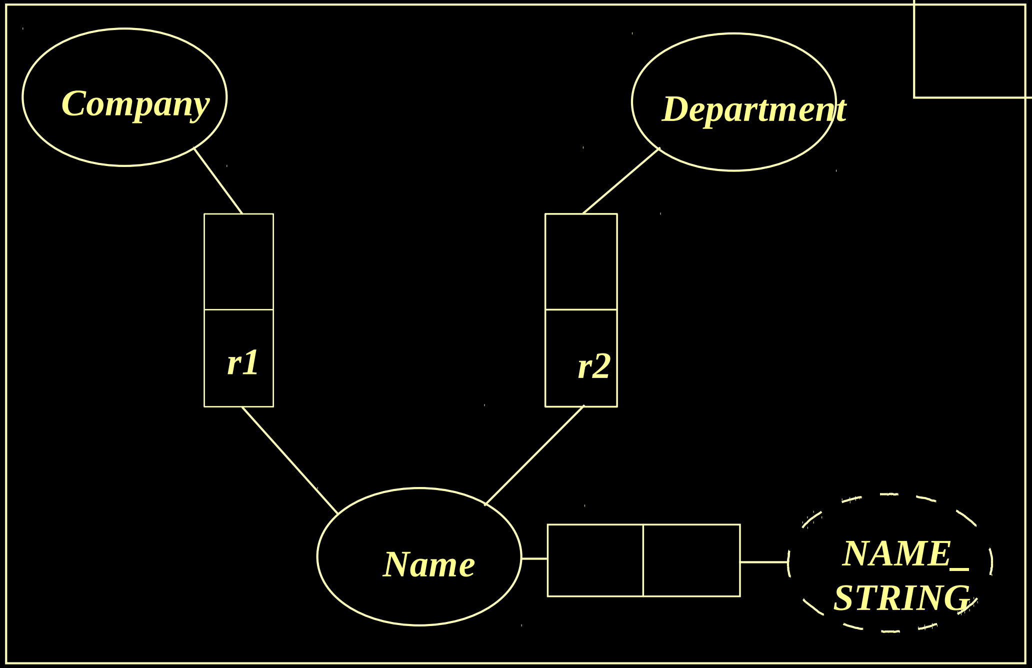


Number may be involved in many roles (r1, r2, r3, r4) and so is an entity type, whereas INTEGER is and will only ever be involved in one role r4'.

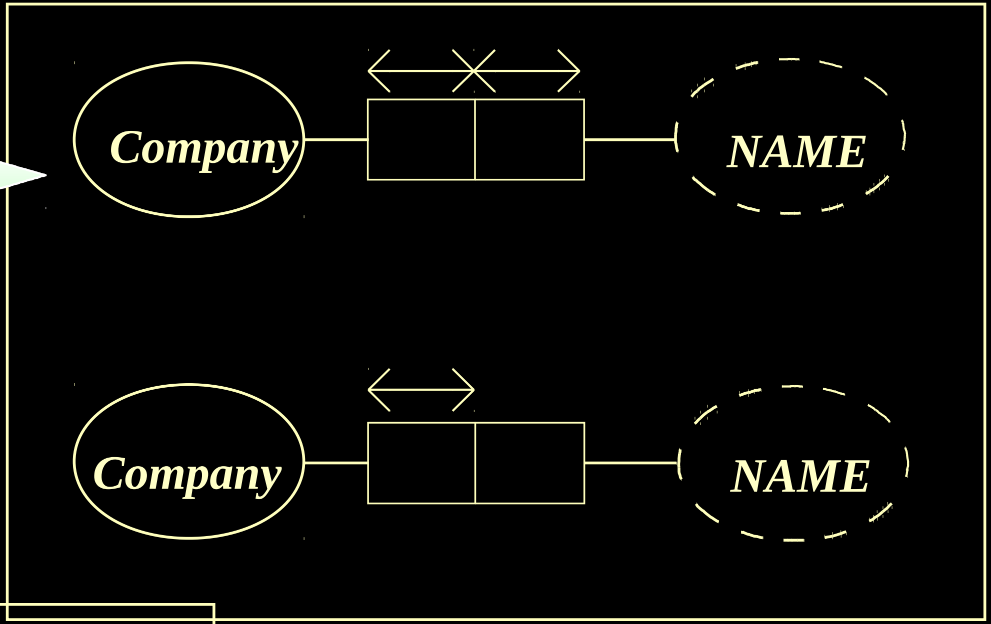
The following diagram is syntactically incorrect



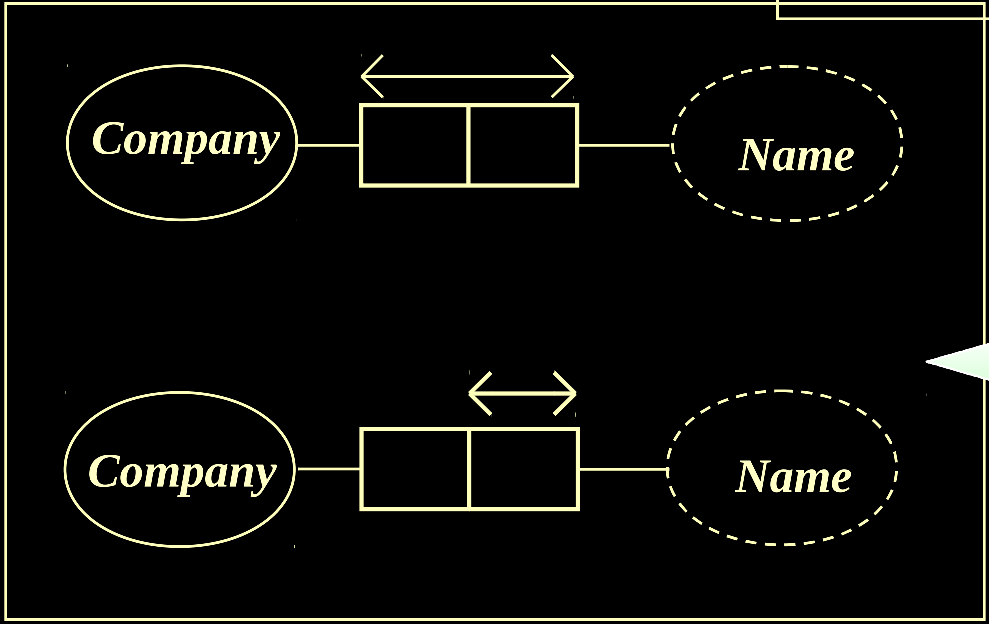
NAME is involved in two roles. Would the following declarations shown below be correct?



Valid situations for UCons in declaration of label types



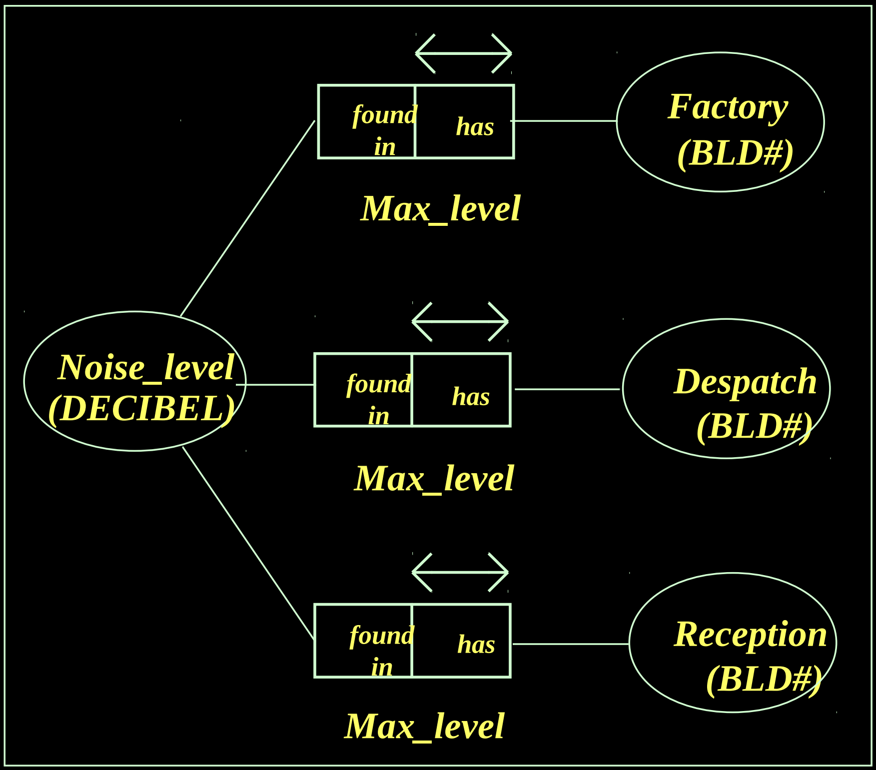
In the following situations NAME has to be an entity type, not a label type:



Incorrect schema

Example:

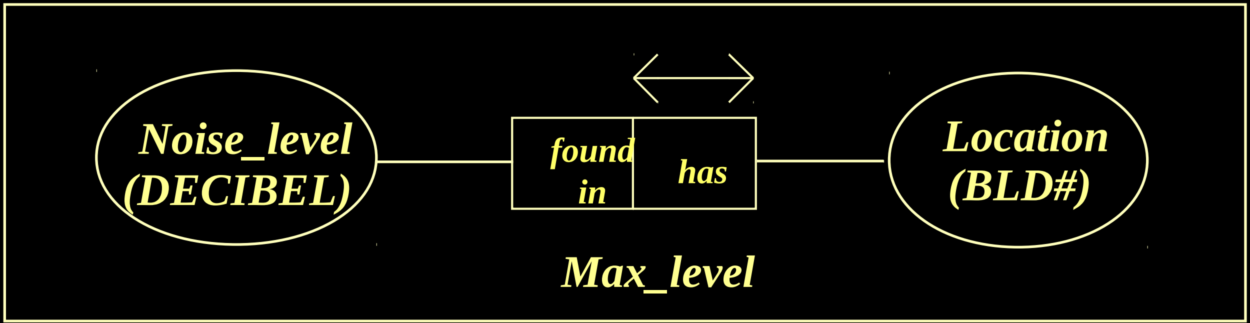
TYPES AND INSTANCES



The failure of drawing the distinction between an entity type and an instance of an entity type is a common mistake.

Factory, Despatch and Reception are instances of an entity type.
The fact type Max_level occurs several times.

The correct schema should be:

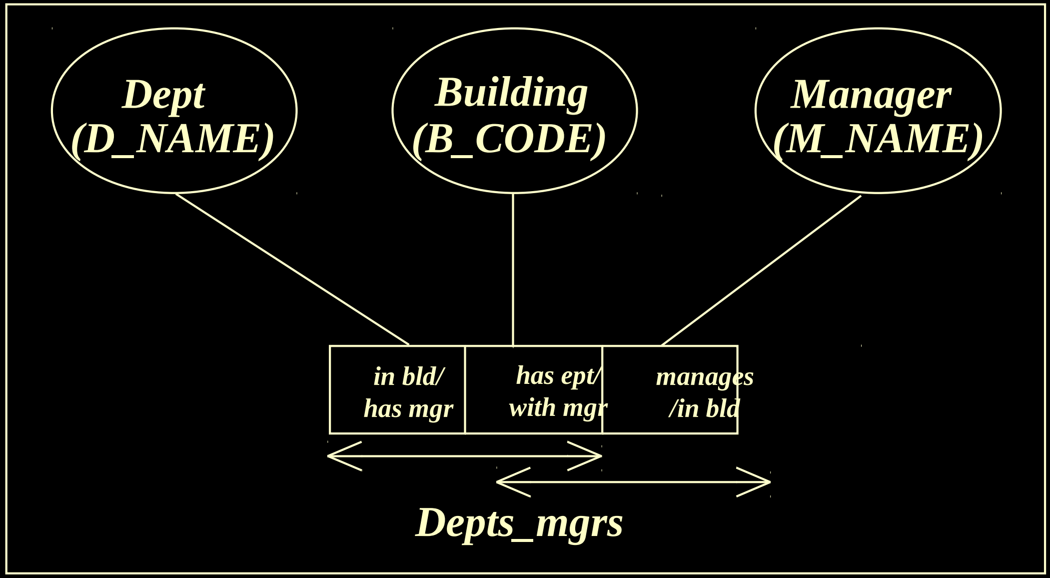


Let us consider a more complex example for UCons construction – remember the close relationship between the step 1 of the CSDP and configurations of Ucons.

Example 1
Departments

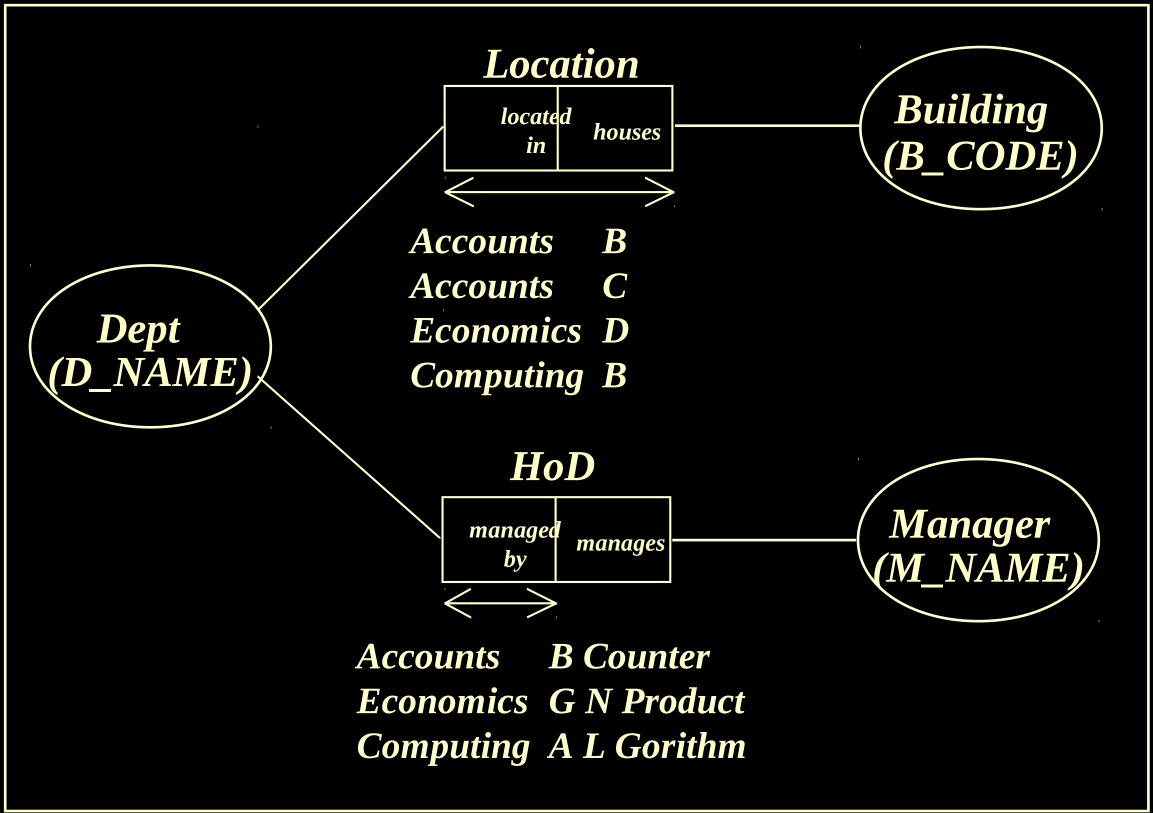
Notice the redundancy in this output report in all columns!!!

Department	*Building*	*Head of Dept*
Accounts	*B*	*B Counter*
Accounts	*C*	*B Counter*
Economics	*D*	*G N Product*
Computing	*B*	*A L Gorithm*



The schema is incorrect!

The output report should be modelled as follows:



In this example, Dept, Building, and Manager are not all semantically related together.

Dept is associated with Building, and INDEPENDENTLY Dept is also associated with Manager.

Manager is ‘partially’ dependent on the pair (Dept, Building)

Shall we add one more Ucon for the fact type HoD?

The above schema containing 2 elementary fact types (with shown UCons) correctly represents the UoD illustrated by the output report.

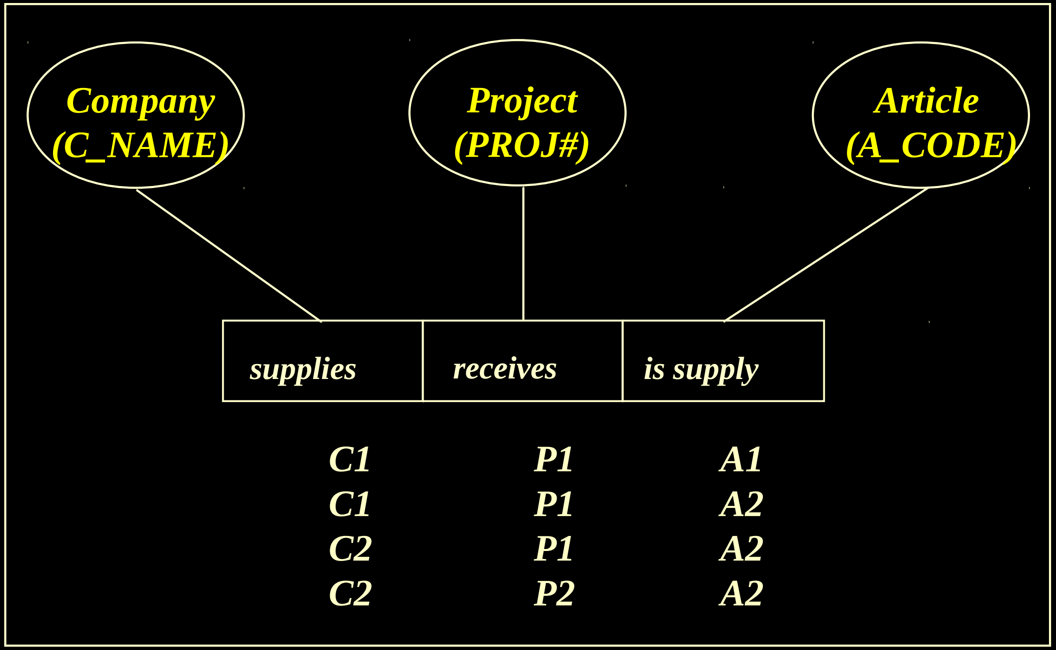
Determine the length of elementary fact types:

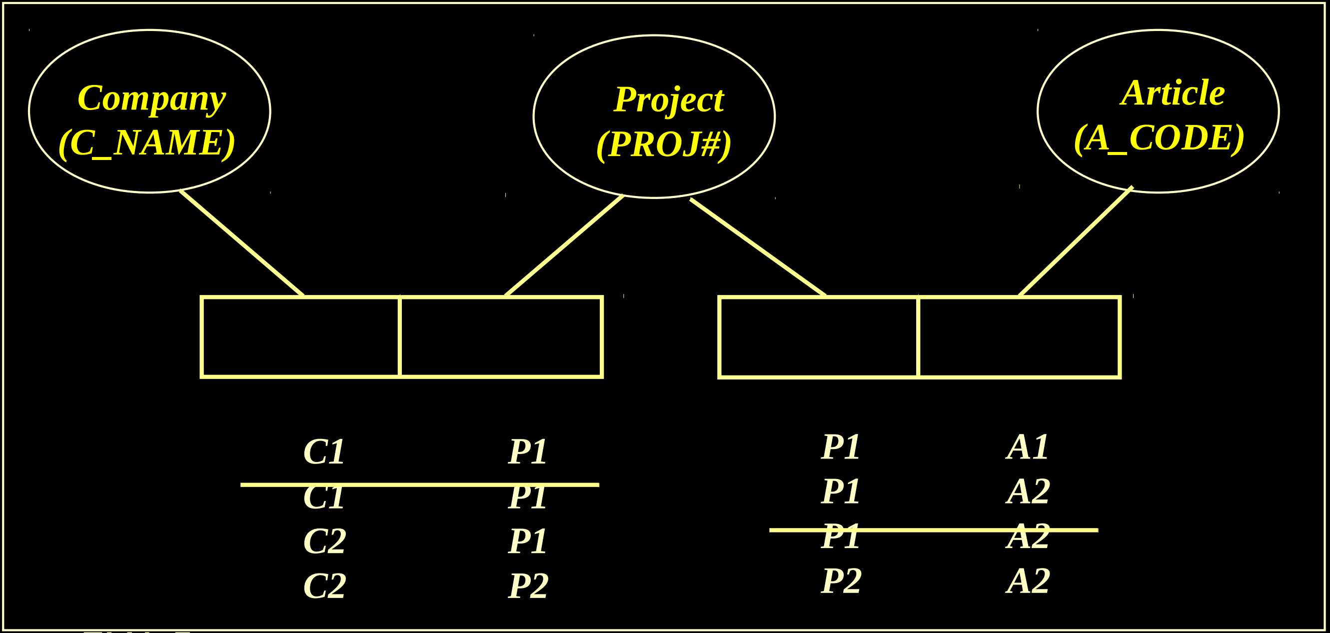
4A: perform <u>reducibility</u> check, and if necessary, correct the conceptual schema diagram; the fact type was too 'long'.

4B: perform <u>information loss</u> check. Bring all the information of the output report into facts permitted by the conceptual schema and check that the original reports can be reconstructed without loss of information; the fact type was too restrictive.

Example 2
Project Needs

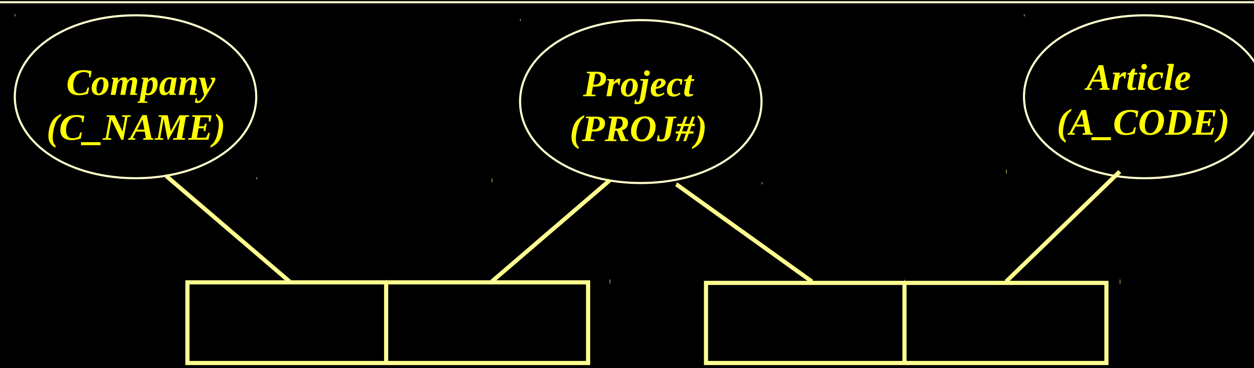
Splitting of this ternary fact type into two binary ones (on Project)





In this example, Company, Project, and Article are inter-related.

Grouping of two binary fact types.

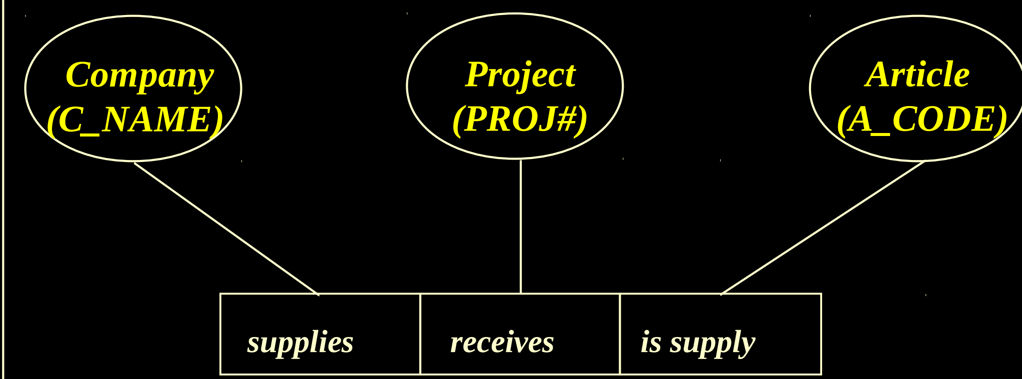


C1	P1	A1
C1	P1	A2
C2	P1	A2
C2	P2	A2

C1	P1
~~C1~~	~~P1~~
C2	P1
C2	P2

P1	A1
P1	A2
~~P1~~	~~A2~~
P2	A2

After Join:

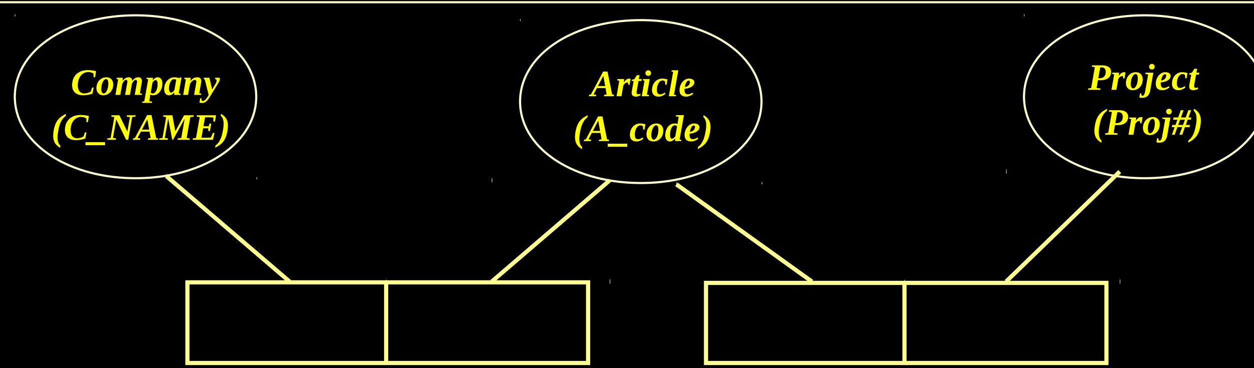


C1	P1	A1
C1	P1	A2
C2	P1	A1
C2	P1	A2
C2	P2	A2

Non-fact (row C2 P1 A1)

Conclusion:
The ternary fact type CANNOT be split in the manner tried. Further splits should be tried: see next

Split on Article.

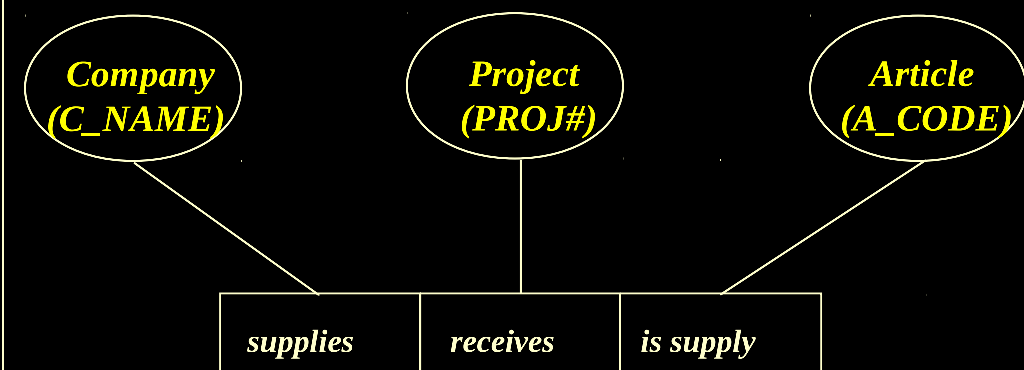


C1	P1	A1
C1	P1	A2
C2	P1	A2
C2	P2	A2

C1	A1
C1	A2
C2	A2
~~C2~~	~~A2~~

A1	P1
A2	P1
~~A2~~	~~P1~~
A2	P2

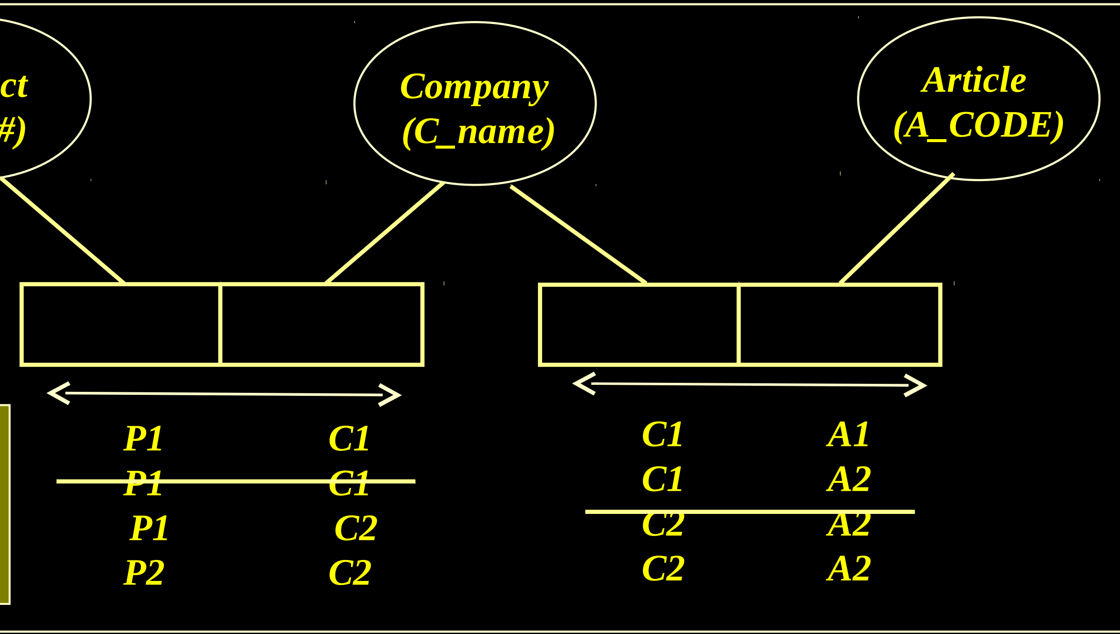
After Join:



C1	P1	A1
C1	P1	A2
C1	P2	A2
C2	P1	A2
C2	P2	A2

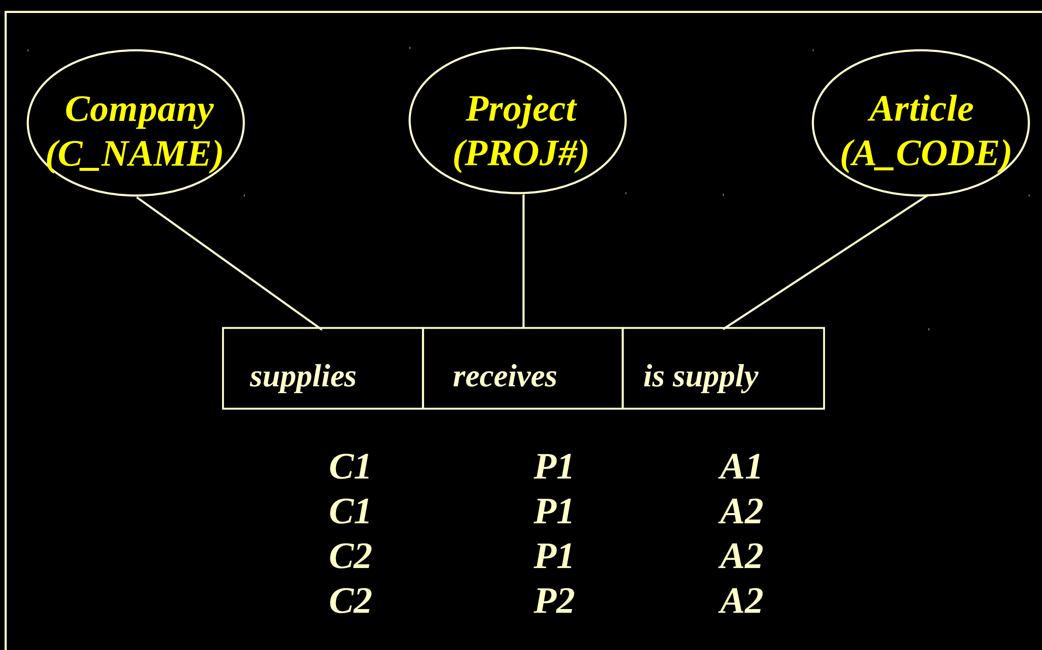
Conclusion:
Again, the ternary fact type <u>CANNOT</u> be split in the manner tried. Further splits should be tried: see next..

Another split, on Company.



C1	P1	A1
C1	P1	A2
C2	P1	A2
C2	P2	A2

After Join:



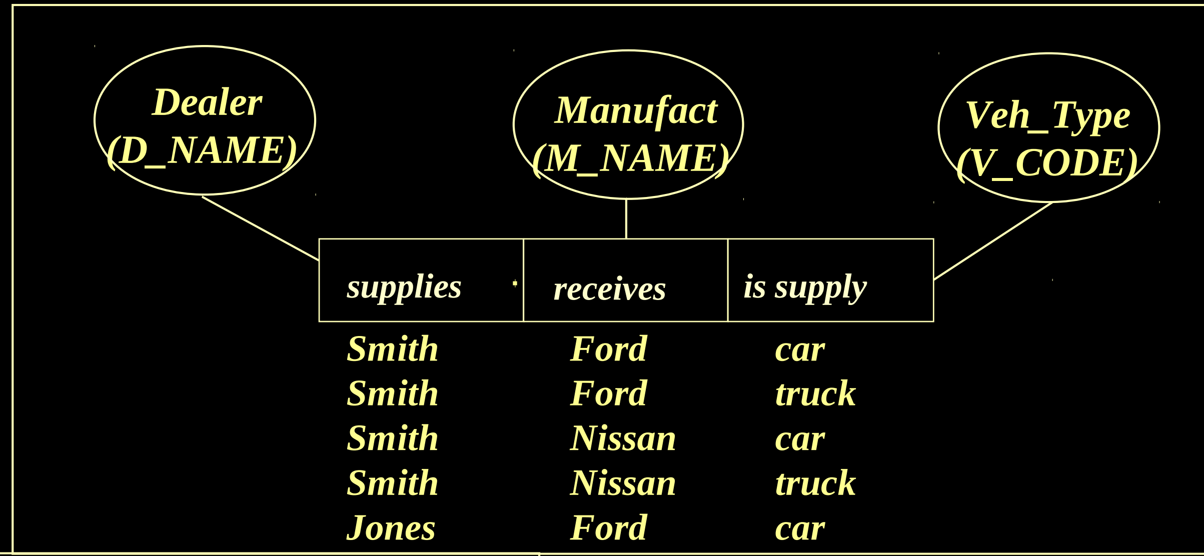
Join Produces original population

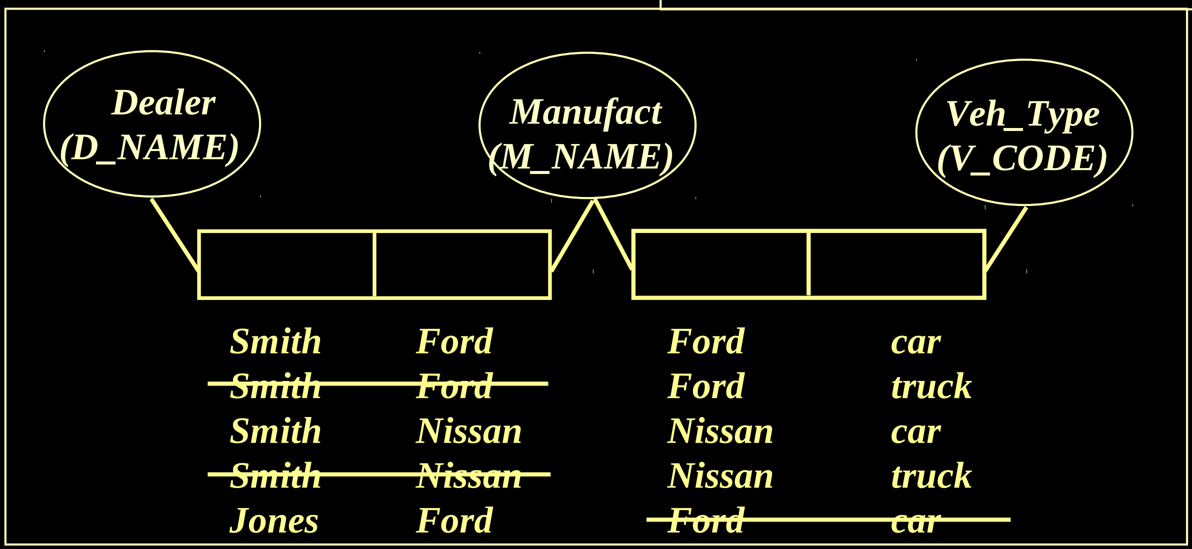
Conclusion:

The ternary fact type CAN be split in the manner tried. No further splits should be tried in this case BUT WE MUST BE SURE about the data representativenes used for the splits

Example 3: Dealers

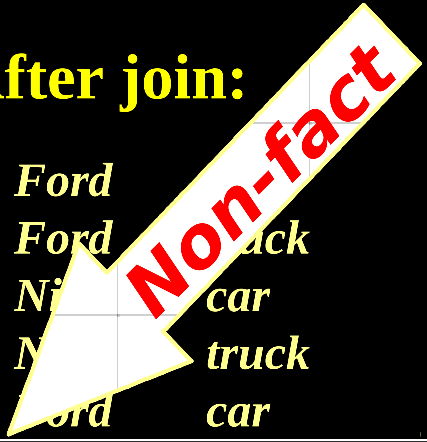
Split and join
(on manufacturer)





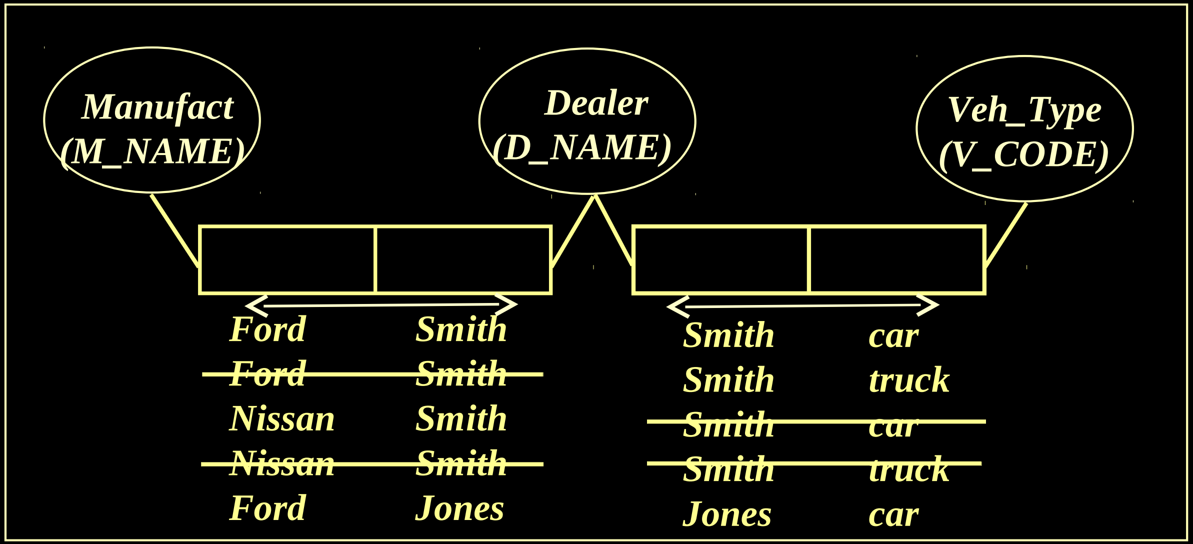
After join:

Smith	Ford	[illegible]
Smith	Fo[illegible]d	[illegible]uck
Smith	Ni[illegible]	car
Smith	N[illegible]	truck
Jones	[illegible]ord	car
Jones	Ford	truck



A non-fact has been generated. Therefore, this split is not correct!

Consider another splitting possibility (there is a third one):



The original significant population has been recovered.

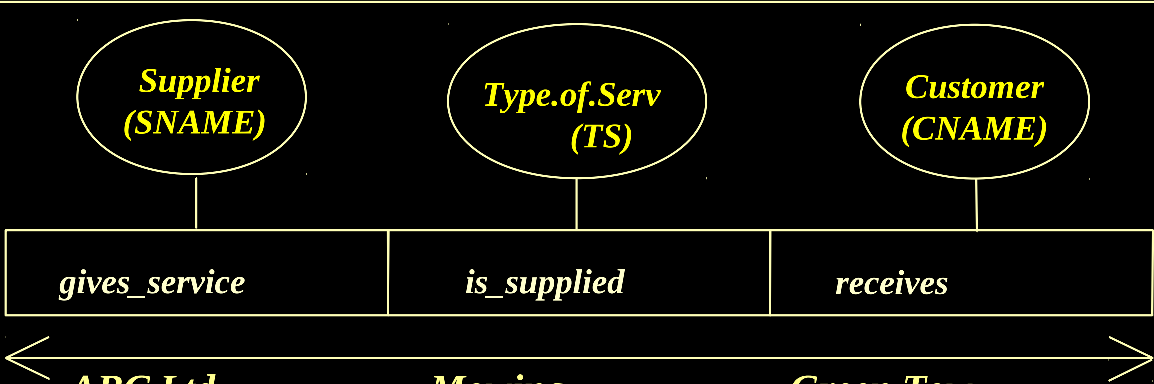
Therefore, the ternary is splittable on Dealer. This is an important result Redundancy is eliminated.

Join the two data samples:

Smith	*Ford*	*car*
Smith	*Ford*	*truck*
Smith	*Nissan*	*car*
Smith	*Nissan*	*truck*
Jones	*Ford*	*car*

Example 4:

Services



ABC Ltd	*Mowing*	*Green Tow*
ABC Ltd	*Pool maint*	*Waterford*
Better Staff	*Mowing*	*Green Tow*
Better Staff	*Mowing*	*Light Plaza*
Better Staff	*Windows*	*Light Plaza*
Care4U	*Mowing*	*Light Plaza*

<u>Theoretically ALL COMBINATIONS</u> should be tried before jumping to a conclusion:

1. **(s,t) join (t,c), (split on t)**
2. **(s,t) join (s,c) (split on s) and**
3. **(t,c) join (s,c) (split on c)**

Before attempting the three way join

4. **(s,t) join (t,c) join (s,c).**

Projections

ST *(Supplier – Type of service)*

ABC Ltd	*Mowing*
ABC Ltd	*Pool maint*
Better Staff	*Mowing*
Better Staff	*Mowing*
Better Staff	*Windows*
Care4U	*Mowing*

SC

ABC Ltd	*Green Tow*
ABC Ltd	*Waterford*
Better Staff	*Green Tow*
~~*Better Staff*~~	~~*Light Plaza*~~
Better Staff	*Light Plaza*
Care4U	*Light Plaza*

TC

Mowing	*Green Tow*
Pool maint	*Waterford*
Mowing	*Green Tow*
Mowing	*Light Plaza*
Windows	*Light Plaza*
Mowing	*Light Plaza*

Joins of any two tables differ from the original output report .

ST Join SC:

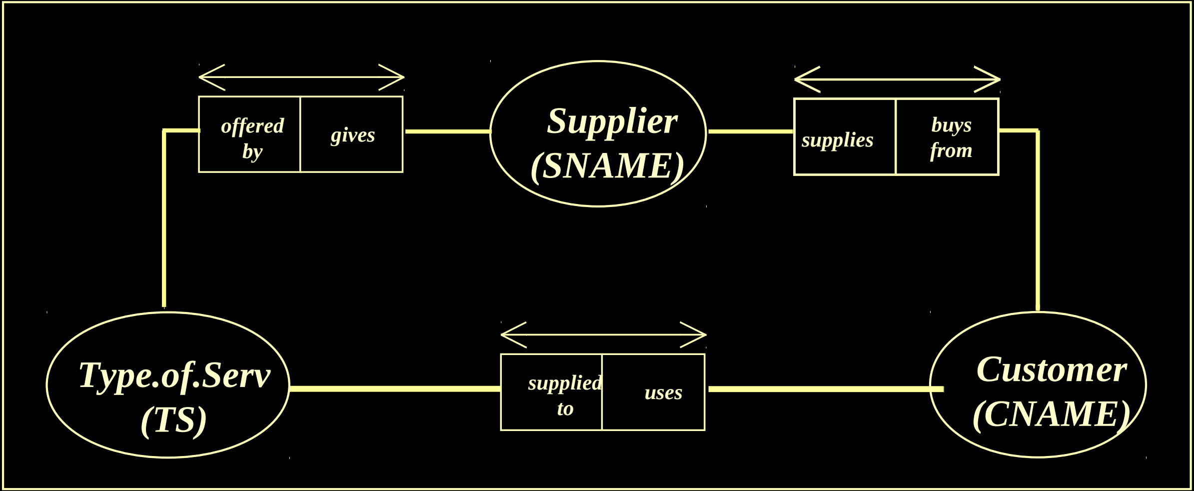
ABC Ltd	*Mowing*	*Green Tow*
ABC Ltd	*Mowing*	*Waterford*
ABC Ltd	*Pool maint*	*Green Tow*
ABC Ltd	*Pool maint*	*Waterford*
Better Staff	*Mowing*	*Green Tow*
Better Staff	*Mowing*	*Light Plaza*
Better Staff	*Windows*	*Green Tow*
Better Staff	*Windows*	*Light Plaza*
Care4U	*Mowing*	*Light Plaza*

(ST Join SC) Join TC:

ABC Ltd	*Mowing*	*Green Towers*
ABC Ltd	*Pool maint*	*Waterford*
Better Staff	*Mowing*	*Green Towers*
Better Staff	*Mowing*	*Light Plaza*
Better Staff	*Windows*	*Light Plaza*
Care4U	*Mowing*	*Light Plaza*

Original table

The original population is recovered from the three binary fact types. The correct schema is shown below to replace the 3-ary fact type on slide 20.



IMPORTANT OBSERVATIONS!!!!

Even though an n-ary fact type should have a uniqueness constraint spanning at least n-1 roles it may be splittable into several smaller fact types.

All possible splits have to be considered before a fact type can be considered unsplittable. In practice the analysis should be simplified by designer understanding of terms and relatioships in the considered UoD.

Advices and summary for beginners:

1. You only model semantically <u>inter-related</u> entities (N>= 3) as ternary or higher order fact types;

2. Refine your model by the splitability check;

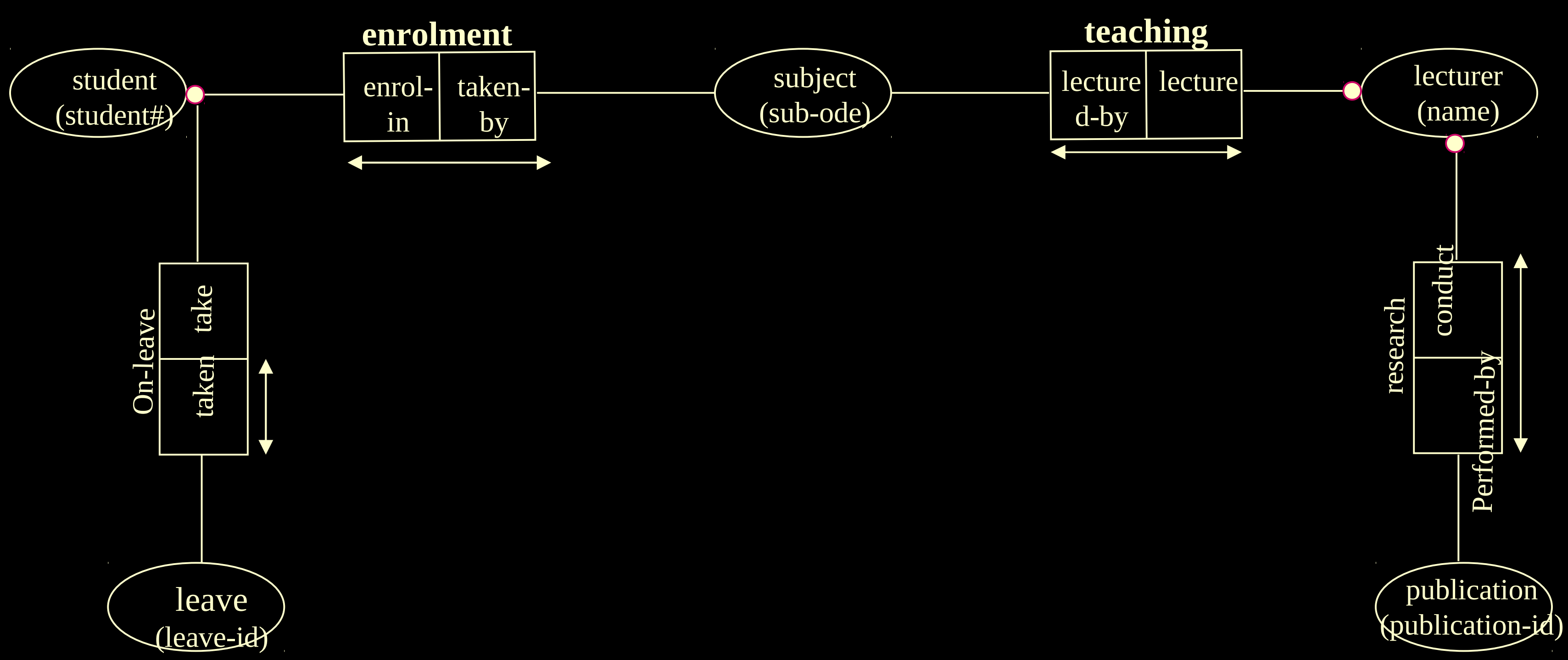
3. In real life UoDs we might have higher order fact types present. So we should be able to identify them;

4. Nested FT is useful when it participates in another non-nested FT(s). Otherwise there is no need to use it. So, use nesting with care!

Step 5 - Add mandatory role constraint – (each instance of an entity MUST participate in the role)

Example : **Each lecturer must conduct teaching *and* research.**

A student is either enrolling in subjects *or* on-leave.

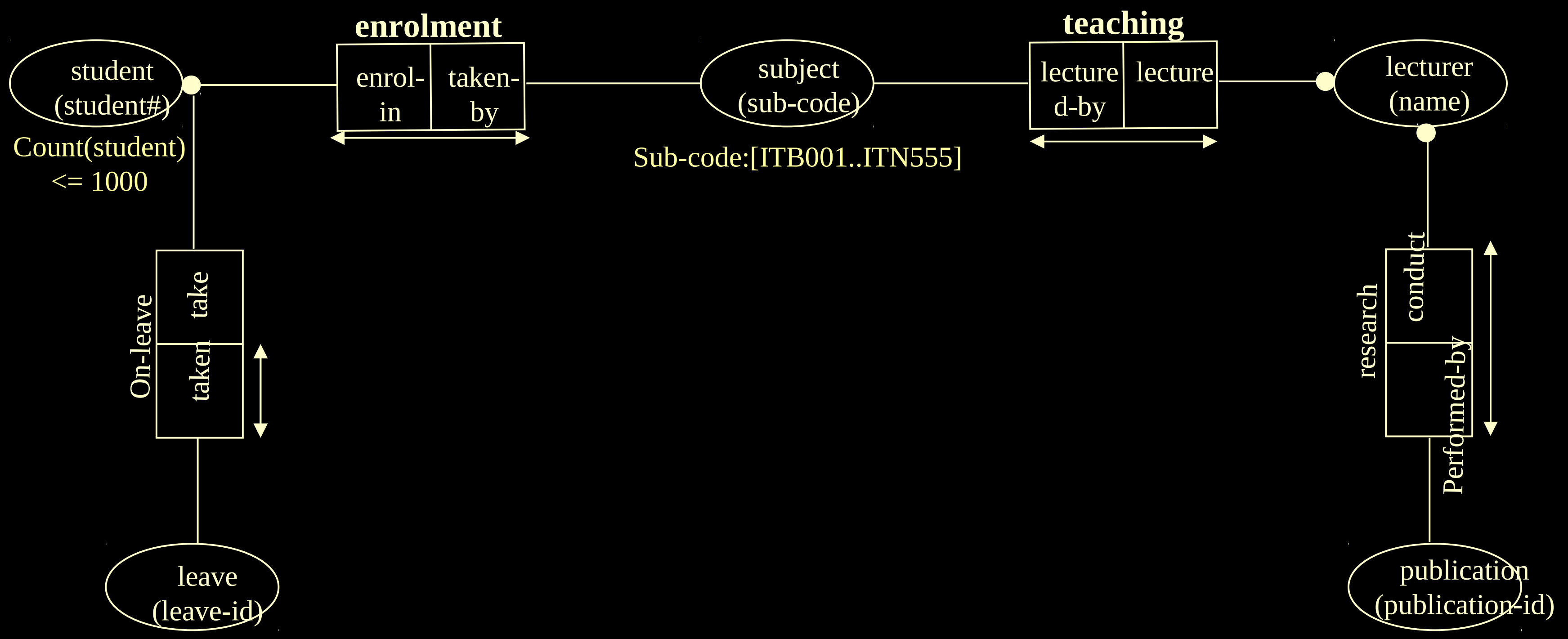


Step 5 (con't)

2. Add cardinality entity/label constraints.

Entity constraint: No. of students is less than or equal to one thousand.

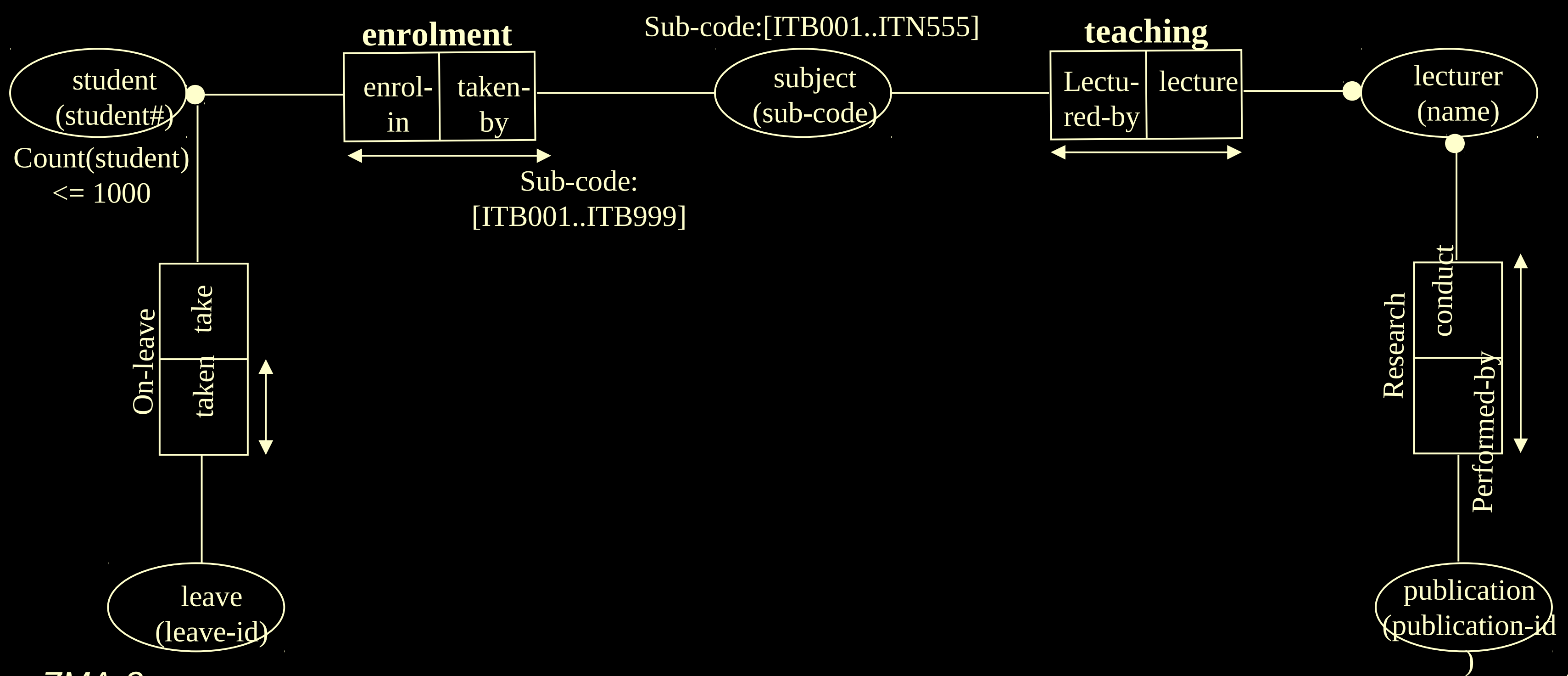
Label constraint: Valid subject codes are between ITB001 and ITN555.



Step 5 (con't)

3. Add role constraints.

Role constraint: Only subjects with subject-codes between ITB001 and ITB999 *can be taken by students*.

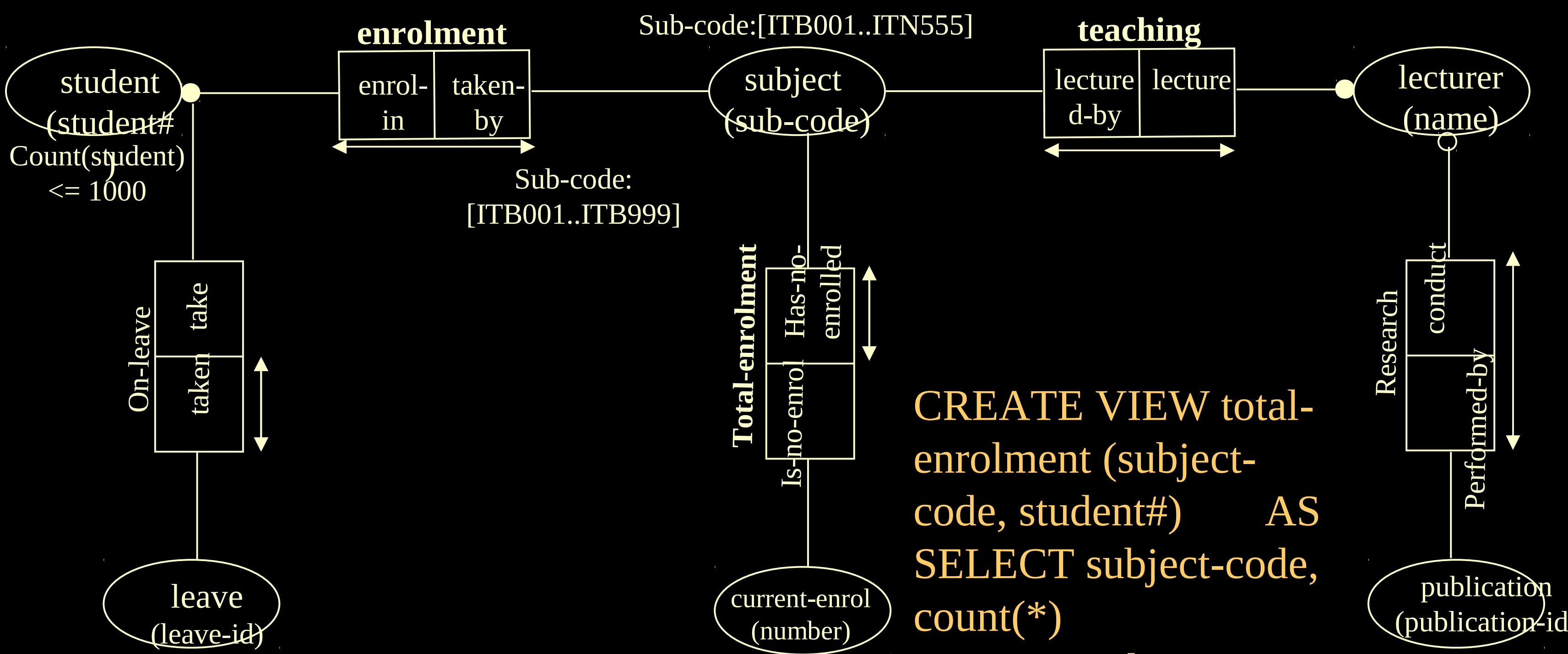


Step 5 (con't)

4. Check logical derivation.

The **total-enrolment** fact type can be derived from other facts:

Therefore, total-enrolment should be removed at Step 5!

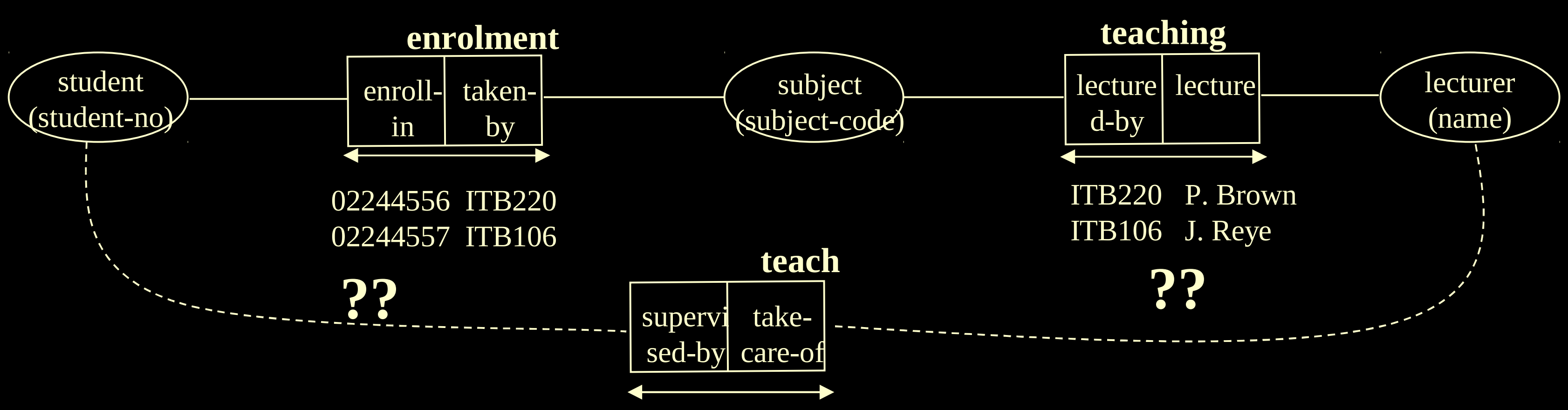


CREATE VIEW total-enrolment (subject-code, student#) AS SELECT subject-code, count(*) FROM enrolment GROUP BY subject-code

Step 5 (con't)

Is the TEACH fact type logically derived from the the teaching fact type and enrolment fact types?

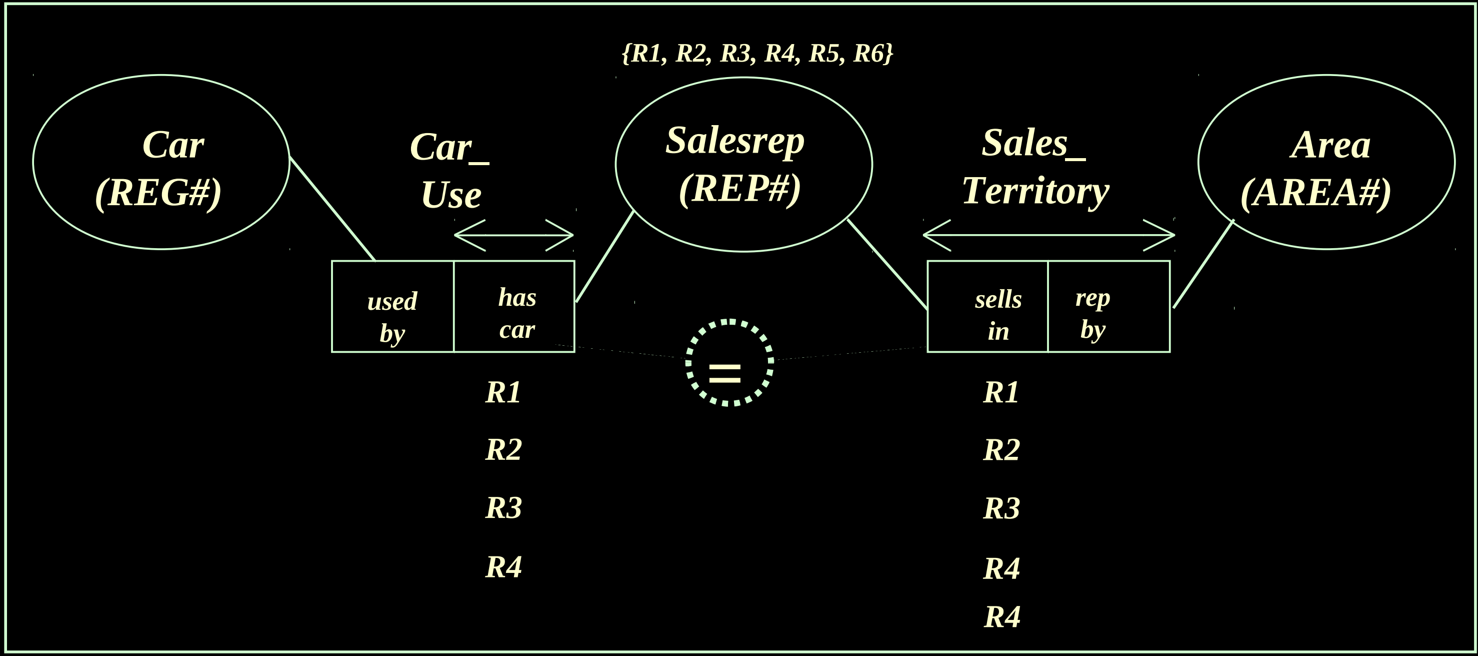
Answer: Yes. So, it should be deleted from the conceptual schema. However, if a supervision relationship (e.g. thesis supervision) exists independent of normal teaching (course-work) relationship, a thesis-supervision fact type may be added.



Equality constraint - SET EQUALITY

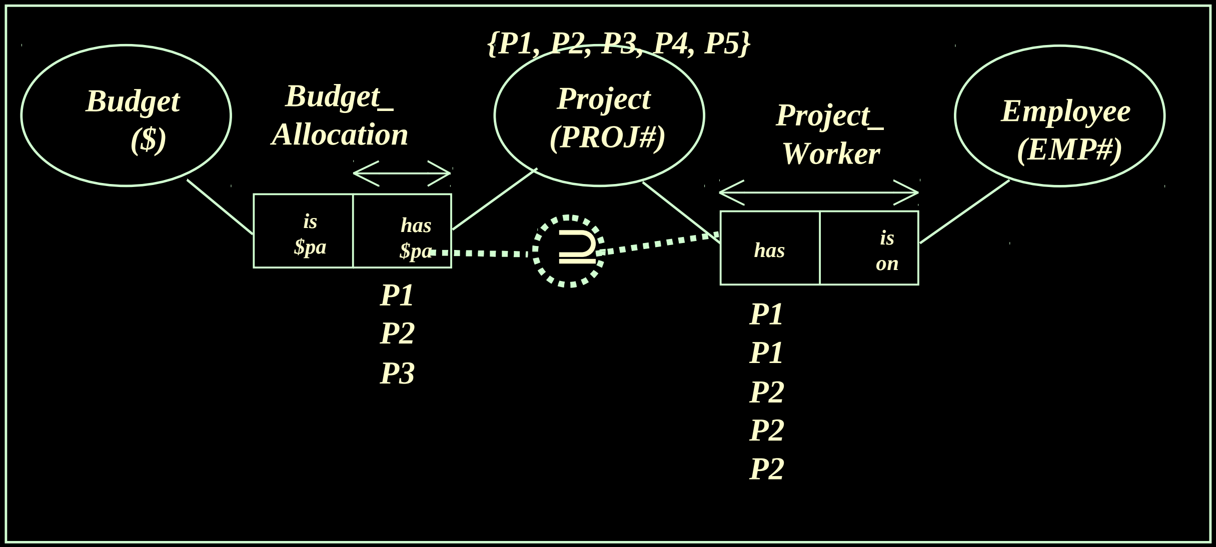
The set of entity instances involved in one role is exactly the same as a set of the entity instances involved in a second role.

Example



Subset constraint

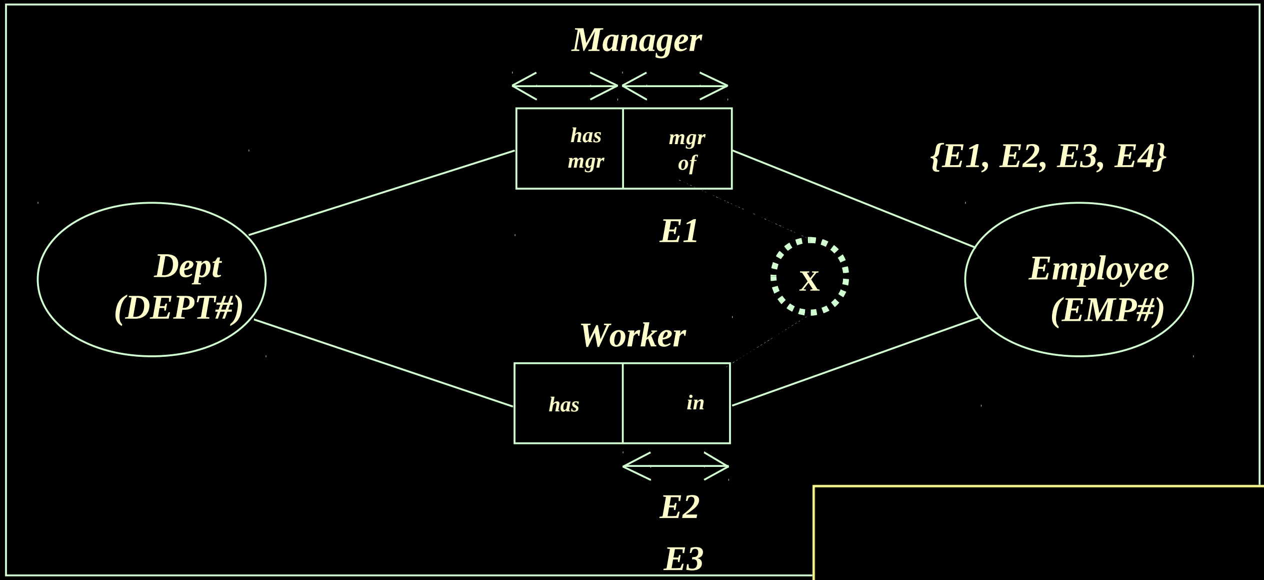
The set of entity instances involved in one role is a subset of the entity instances involved in a second role.

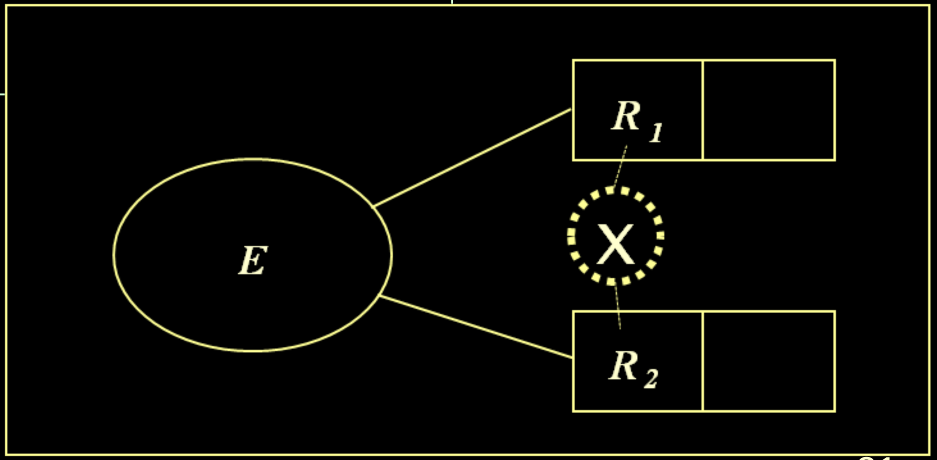


Each Project that can employ workers must have a budget; some of these project can have no a worker allocated yet, but other projects can have many workers allocated.

Set Constraints

Exclusion constraint If an entity instance is involved in one role it is excluded from involvement in another role.

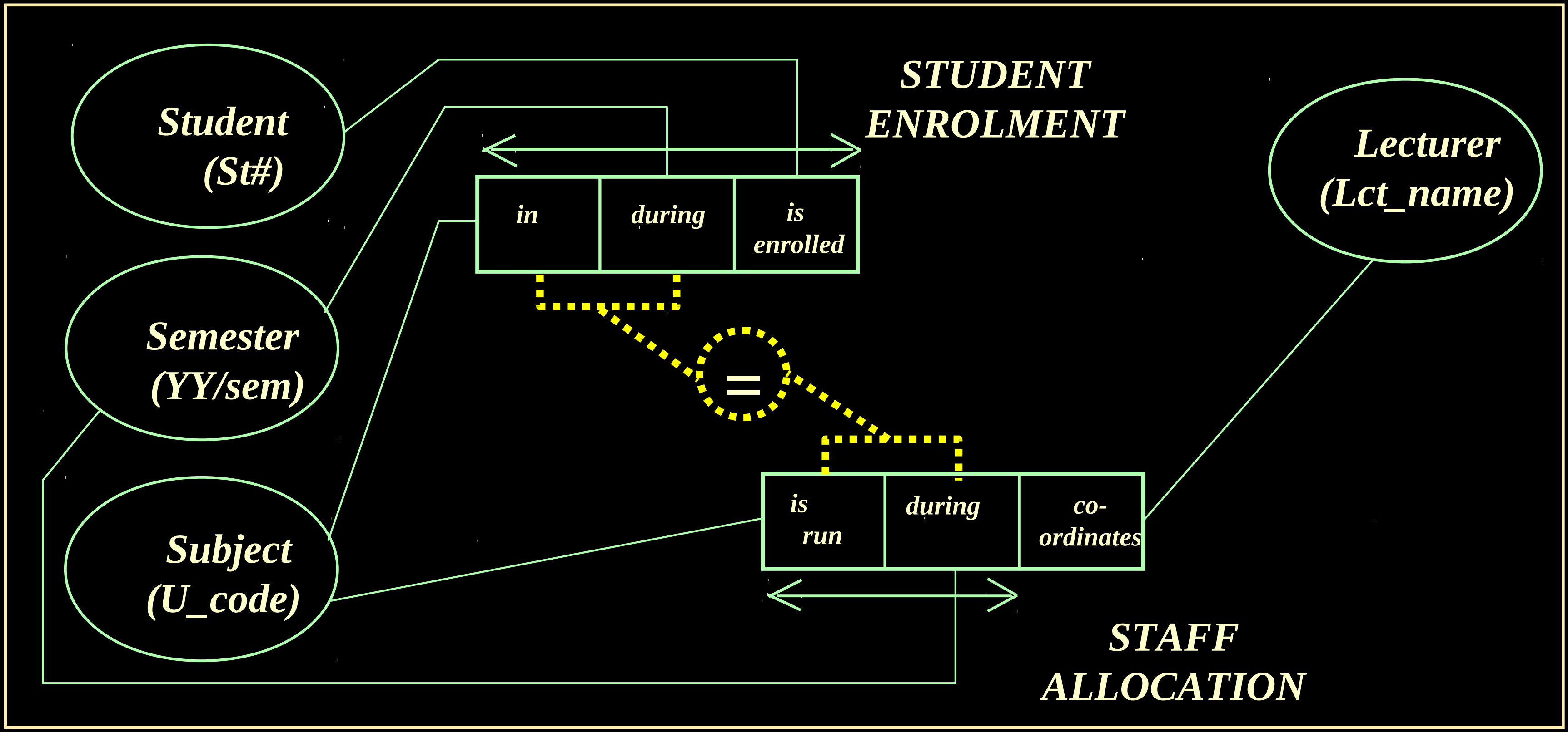




Multiple-role Equality constraint

The set of combinations of entity instances of two or more entity types involved in two fact types are exactly the same.

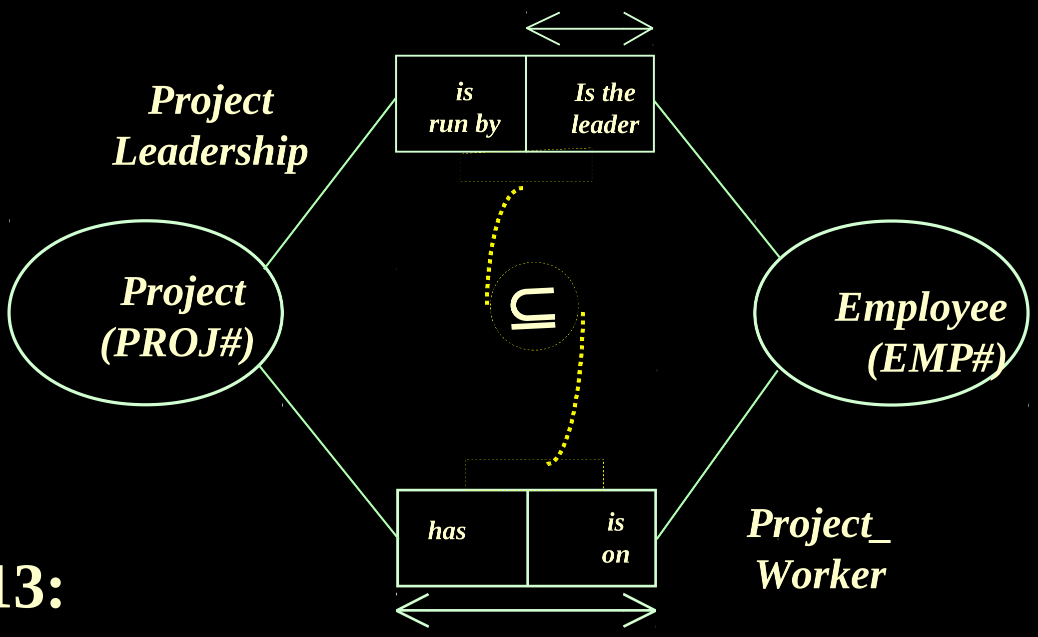
Example 12



Multiple-role Subset constraint

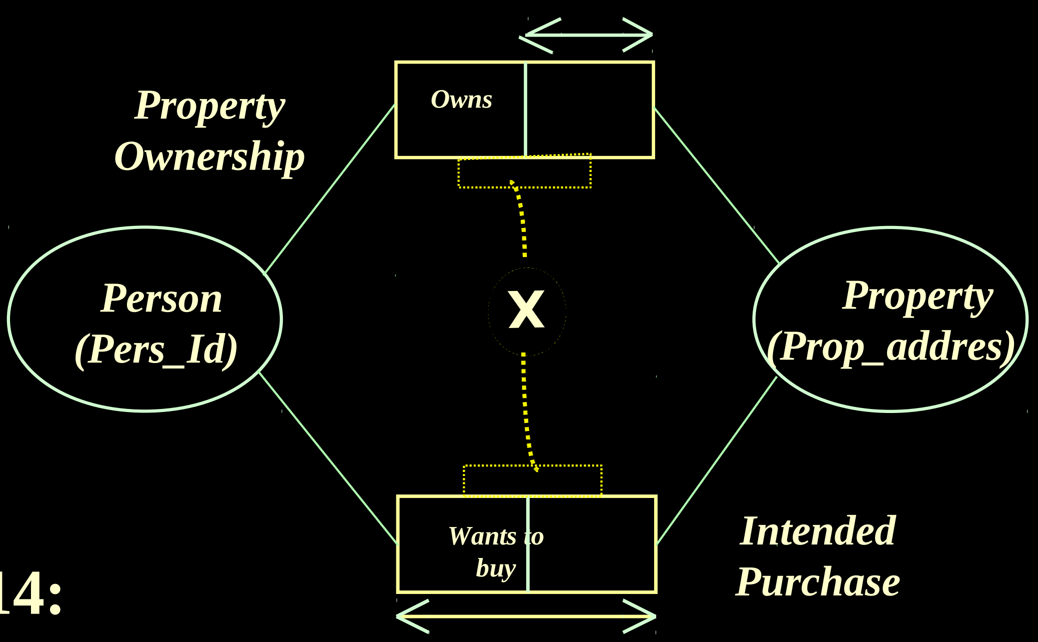
The sets of combinations of entity instances of the same entities involved in two fact types are in a inclusion relationship.

Example 13:



Multiple-role Exclusion constraint

The sets of combinations of entity instances of the same entities involved in two fact types are in an exclusion relationship.



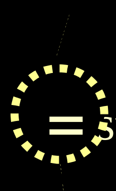
Example 14:

General definitions (more formal)

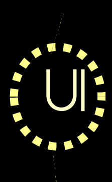
Consider two Fact Types F and F' that have subsets of their roles

$$R = \{R_{k1}, R_{k2}, \ldots, R_{km}\} \text{ and } R' = \{R'_{h1}, R'_{h2}, \ldots, R'_{hm}\}$$

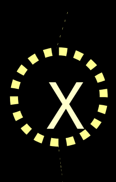
where in each pair of corresponding roles R_{ki} and R'_{hi} (i=1..m) are defined on the same entity type. We say that these sets of roles satisfy a set constraint, if the population of any instance of the fact type F restricted to the set R of roles , pop(R), is involved in the same set_relationship with the population of F' restricted to R', pop(R'). In particular;



Fact types F and F' satisfy equality constraint on R and R' if for any stance of the UoD pop(R) = pop(R')



Fact types F and F' are involved in the subset constraint on R and R' if for any instance of the UoD pop(R) $\subseteq$ pop(R')



Fact types F and F' are involved in the exclusion constraint on R and R' if for any instance of the UoD pop(R) $\cap$ pop(R') = $\varnothing$ (are disjoint)

Example to illustrate complex configurations of constraints.

The first purpose of the presented example is to show how the set constraints could be identified from the rules valid in a larger domain. The other purpose is to identify a possibly complete set of constraints that will guarantee the consistency of data with the rules that must be enforced in the particular domain. The example is concerned with the UOD ‘Tennis tournaments’. For simplicity we consider only data from one year and we introduce limitation to cover only events from one stream (say men singles) and from quarter-finals upwards.

The rules of any tournament organisation are well known. Two players play a match and the winner advances to the next level. No other players than winners in Quarter-final and Semi-finals, can play in Semi-finals and Final of a given tournament respectively. To check if the set of constraints on the schema is sufficient to enforce consistency of data with the rules of the UoD, we may introduce an amendement of the output report by some corrupted data that violate UOD rules but satisfy constraints that are introduced so far. We will identify the constraint(s) that could prevent presence of such data.

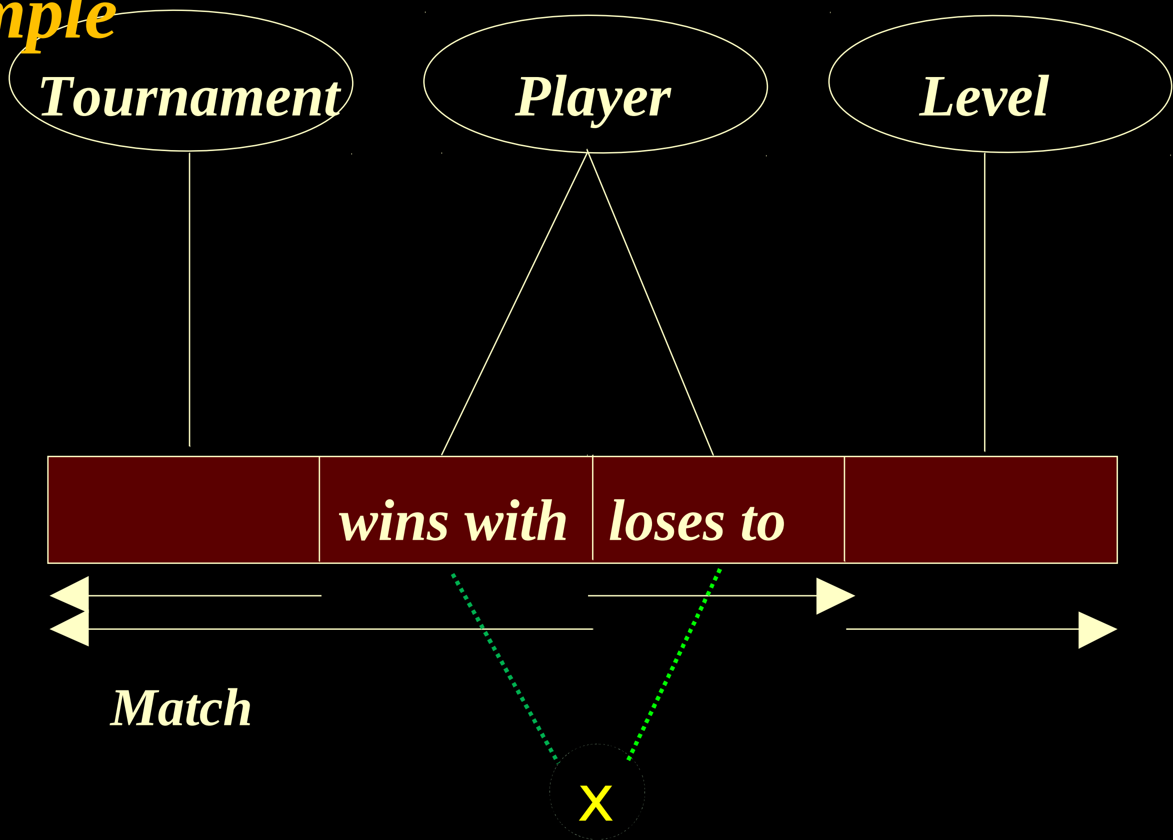
The abbreviations used are;
Q, S, F (in the column Lev(el)) – Quarter-final, Semi-final, Final
A, B, C, ….(in the columns Win(ner), Los(er)) – Synonyms for Player Ids
Wi, AO – examples of tornament shortened names (Wimbledon, Australia Open)

Set Constraints –Example

Tennis Tournaments

Tour	Lev	Win	Los
Wi	Q	A	H
Wi	Q	B	G
Wi	Q	C	F
Wi	Q	D	E
Wi	S	A	D
Wi	S	B	C
Wi	F	A	B
AO	Q	B	D
AO	S	D	J

For simplicity, label types are omitted here.



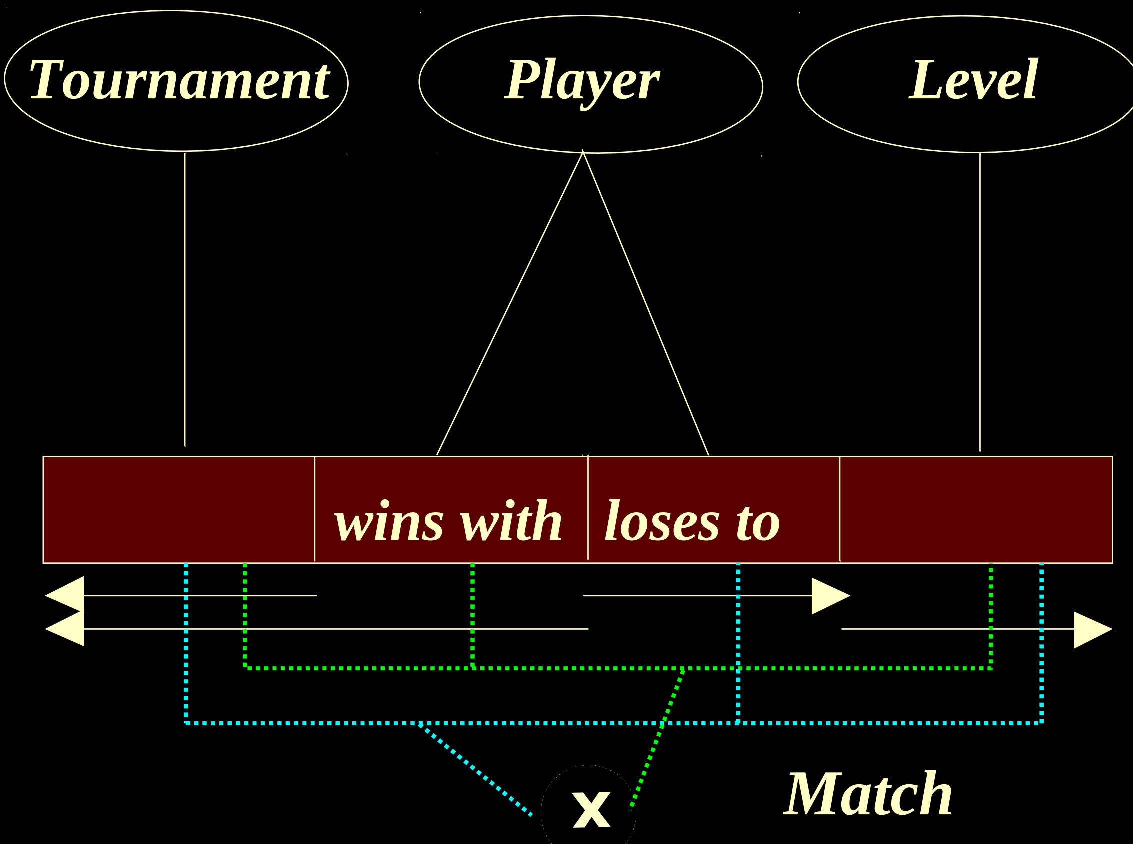
1. *During any tournament one can be defeated only once*

2. *During any tournament one can win only once in any stage (Q,S,F)*

Is the exclusion constraints valid for this UoD? Obviously NOT.

Tennis Tournaments

Tour	Lev	Win	Los
Wi	Q	A	H
Wi	Q	B	G
Wi	Q	C	F
Wi	Q	D	E
Wi	S	A	D
Wi	S	B	C
Wi	F	A	B
Ao	Q	B	D
Ao	S	D	J
....			



During no stage (Q,S,F) of a tournament one can win a match and lose another match.

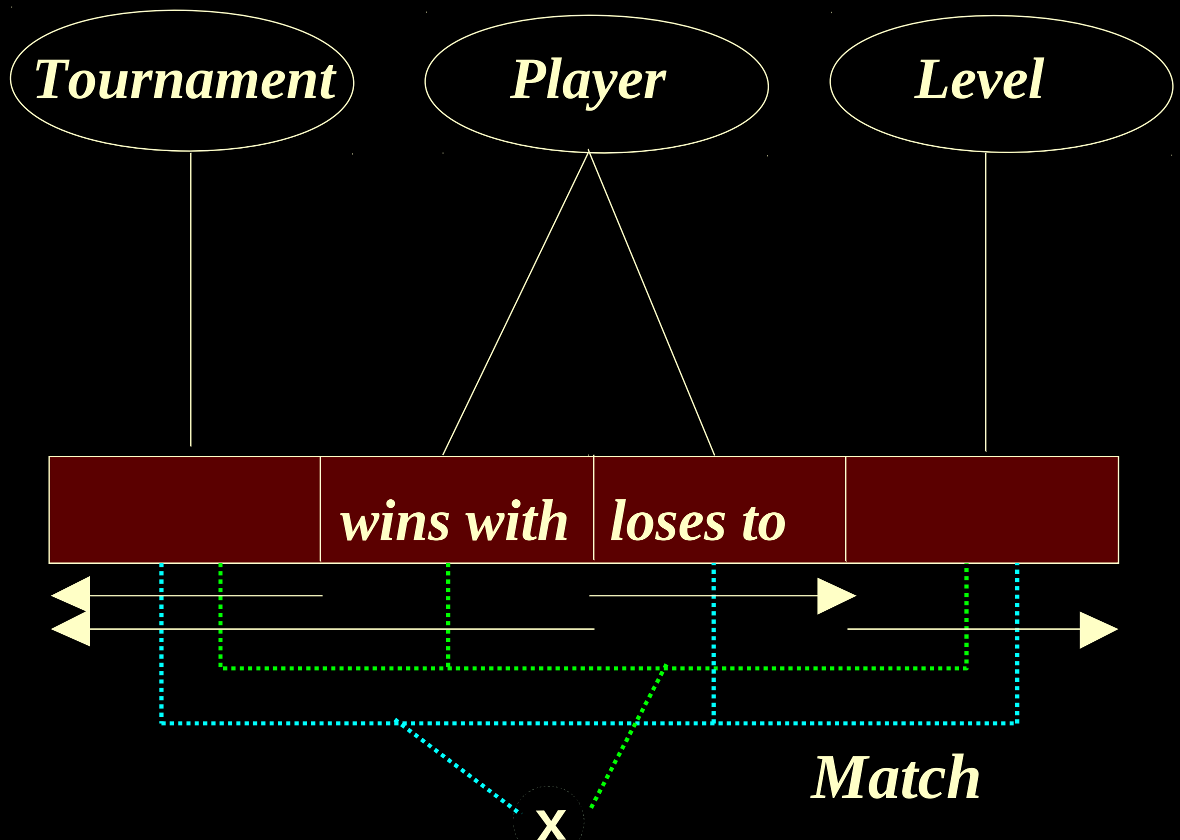
Wi A Q
Wi B Q
Wi C Q
Wi D Q
...

Wi H Q
Wi G Q
Wi F Q
Wi E Q
...

IT is because above projections of any sample data are disjoint.

Tennis Tournaments

Tour	Lev	Win	Los
Wi	Q	A	H
Wi	Q	B	G
Wi	Q	C	F
Wi	Q	D	E
Wi	S	E	D
Wi	S	F	C
Wi	F	G	K
Ao	Q	B	D
Ao	S	D	J
....			

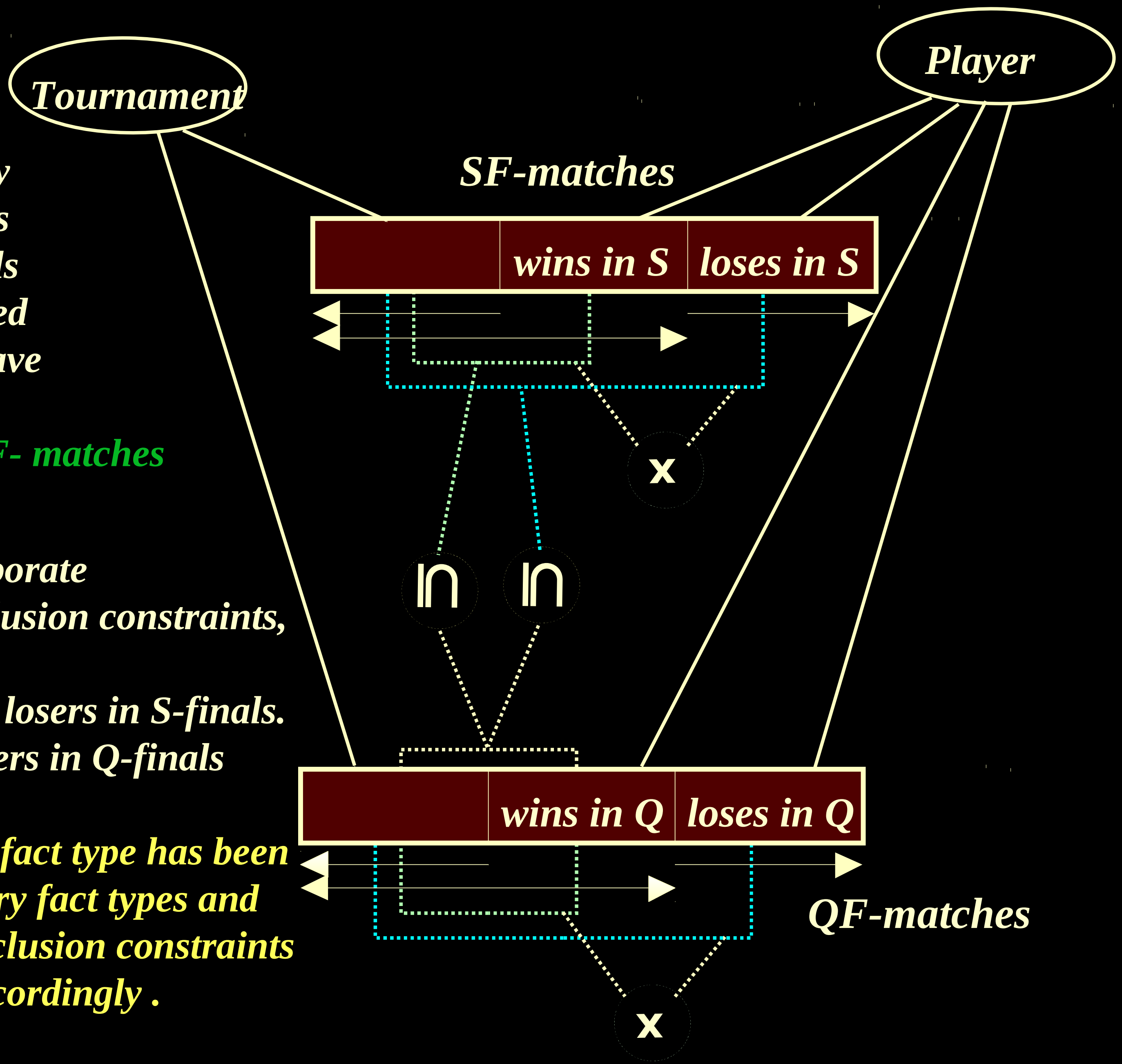


Note, that ***the above constraints are valid also for data which do not satsfy 'tournament rules"***. *The compromised data (in green) still satisfy the constraints but they show that it is still possible for the losers in Q to win in S and/or in F. Also, it is not prevented to have other players at the higher stages. The constraints on the diagram do not enforce tournament logistics.*

To guarantee, that only winners from Q-finals will advance to S-finals and this will be enforced by our Schema, we have to separate QF- matches and SF- matches

Now it is easy to incorporate in the schema two inclusion constraints, one for winners in S-finals and one for losers in S-finals. All of them are winners in Q-finals

Note that the fortnary fact type has been transformed into ternary fact types and the uniqueness and exclusion constraints have been modified accordingly .

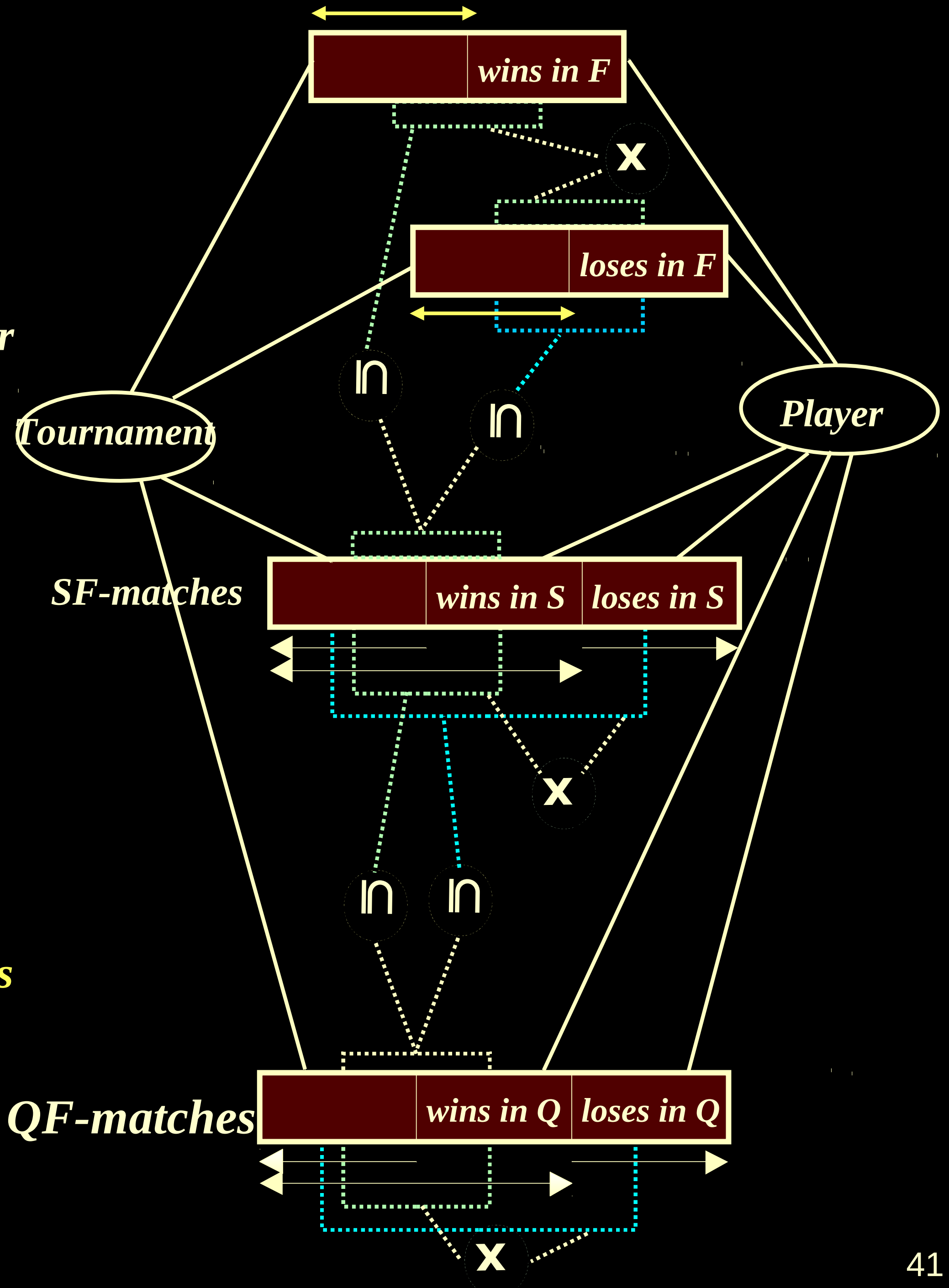


Next we have to ensure that only winners from S-finals advance to Finals.
The relevant Fact Types on the schema differ from the one for Q- or S-finals, since there is only one winner and one loser in Finals of each tournament (*)

Now, we incorporate an exclusion constraint

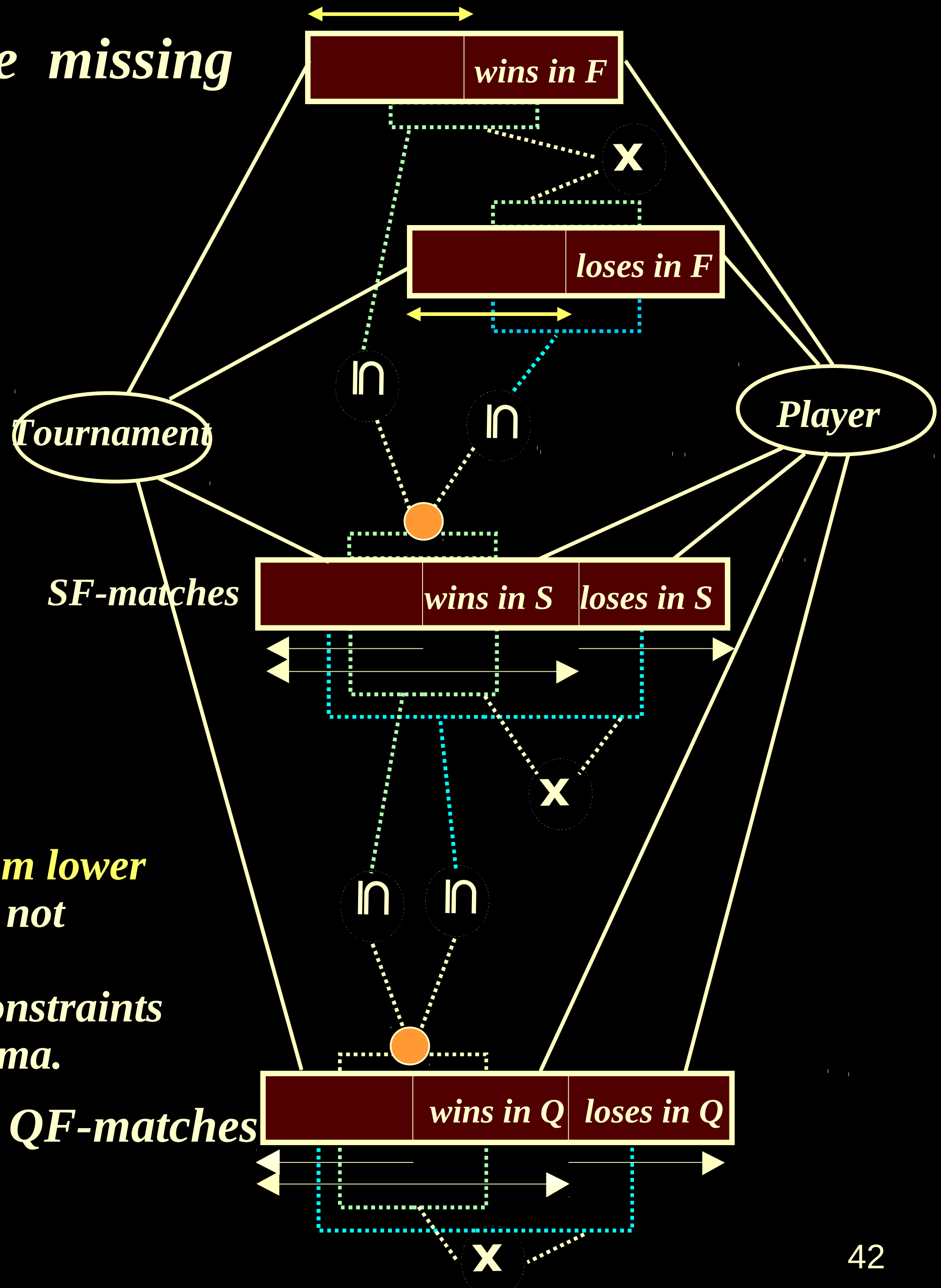
...and two inclusion constraints, one for winner in Finals and one for losers in Finals

(*) Note that one could try to set a ternary fact type for finals, but in this case the uniquenes constraint could be on only one role - for Tournament, and that ternary fact type had to be split



Still… some constraints are missing

Tour	Lev	Win	Los
Wi	Q	A	H
Wi	Q	B	G
Wi	Q	C	F
Wi	Q	D	E
Wi	Q	X	Z
WI	Q	Y	W
Wi	S	A	D
Wi	S	B	C
Wi	S	X	Y
Wi	F	A	B



The requirement: ONLY all winners from lower stage have advanced to the next stage is not supported
The above 'corrupted' data satisfy all constraints shown currently on the conceptual schema.

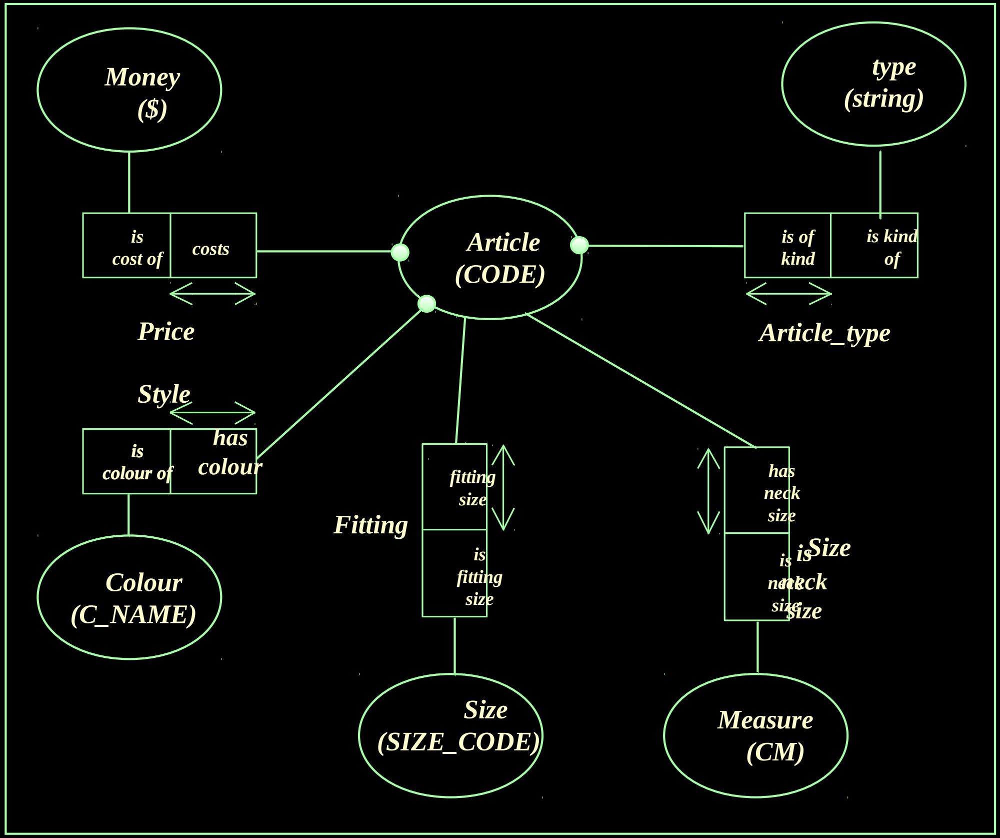
And the medicine is ……

Step 6 cont - Subtype construction

- **The need for subtyping**
- **Method of construction**
  - **Entity-Role matrix**
- **Subtype notation**

Consider the following output report:

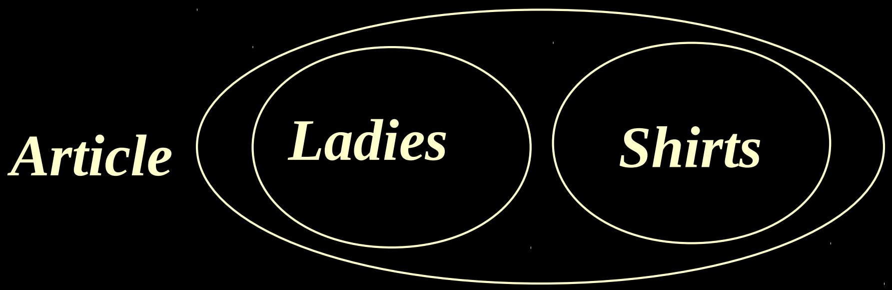
Article	Code	Colour	Price	Size	Neck
Dress	134-6	Red	50.00	14	-
Dress	214-5	Yellow	45.00	12	-
Skirt	712-0	Red	34.95	12	-
Shirt	615-8	White	25.00	-	90
Shirt	547-2	White	50.00	-	85
Shirt	615-9	White	25.00	-	90



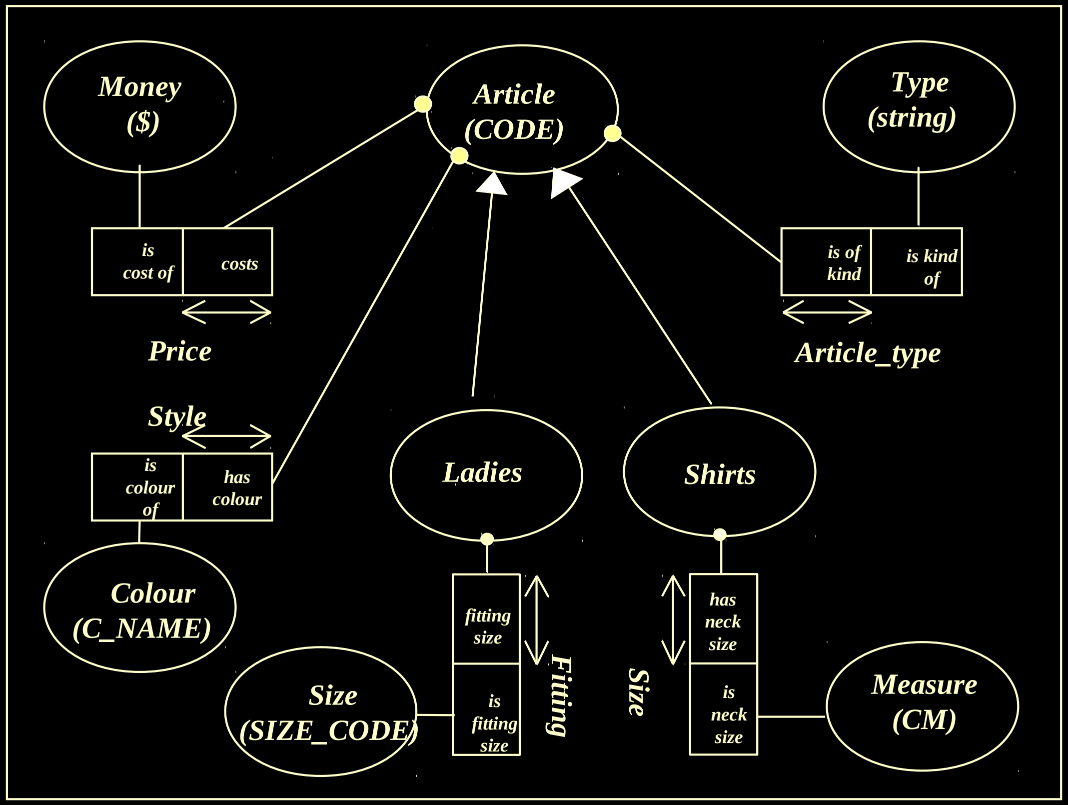
The conceptual schema diagram

From the output report it is evident that shirts have a neck size and skirts and dresses have a fitting size.

Therefore, although we are concerned with one type of thing, article of clothing, there are variations in the UoD.



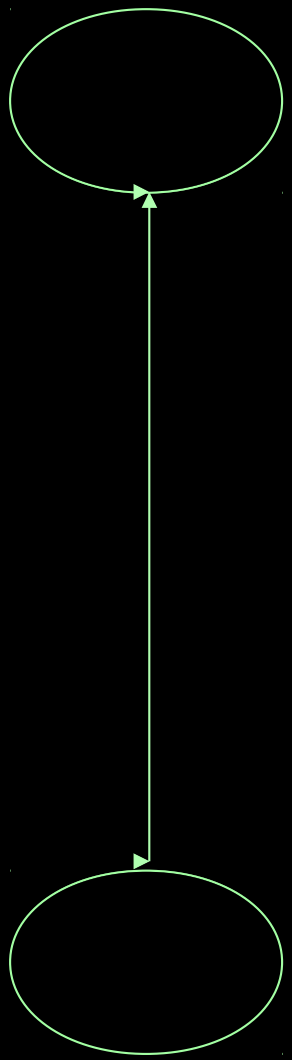
This is shown as follows:



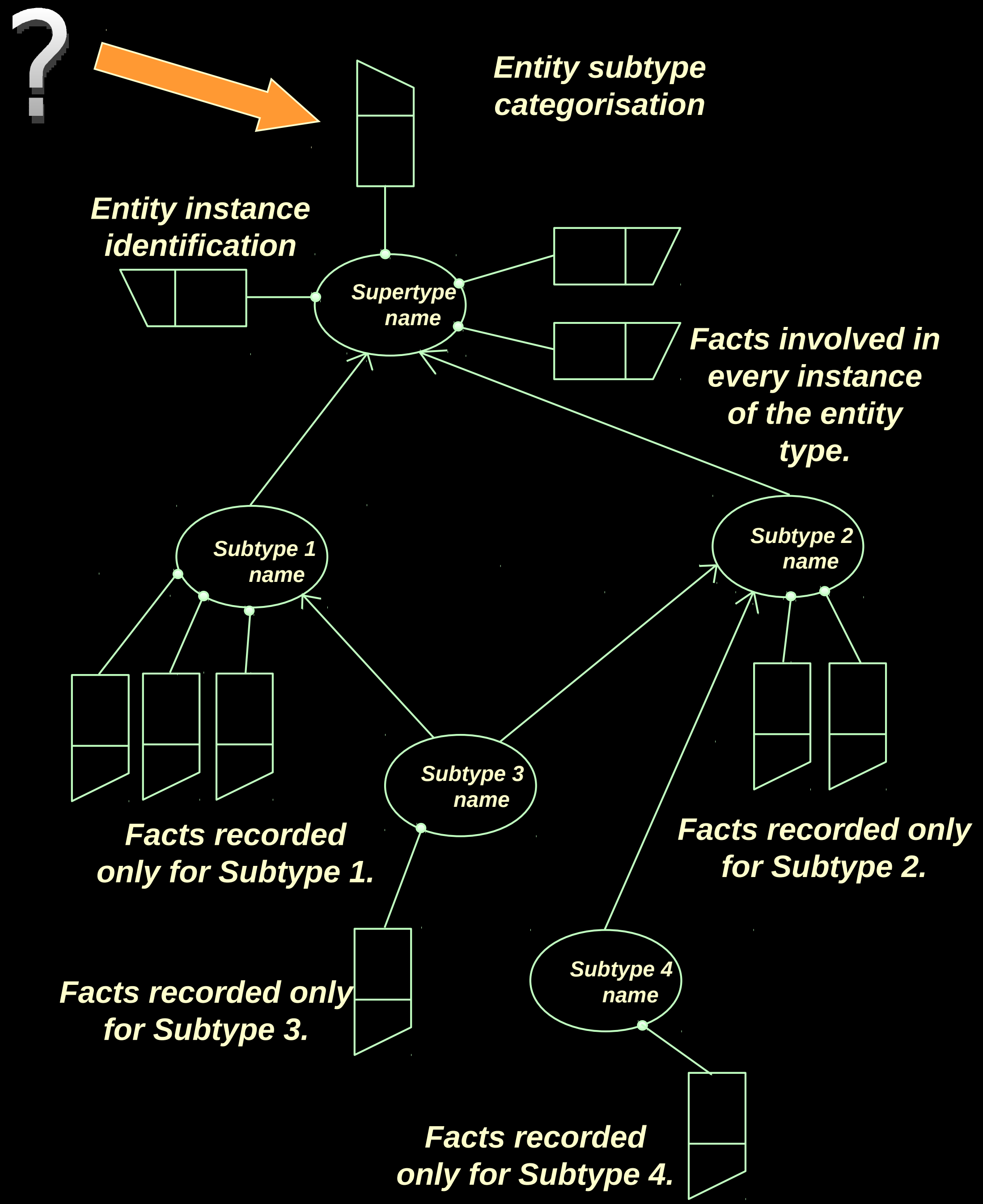
Supertype: contains the entity type name, and facts giving entity instance identification and subtype categorisation information.

The supertype is involved in facts which are to be recorded about every instance of the entity type (total role). The subtype is involved in facts which are to be recorded about every instance of that particular subtype, and no other

Subtype: contains the name of the subtype and the facts that relate only to that subtype.



General subtype characteristics



Entity/Role Matrix

1. **Record all *roles* related to an entity type as *columns* in the matrix.**
2. **Record each *instance* of the entity type as a *row* in the matrix.**
3. **For each instance, if it participates in a role (i.e. it has a particular property), put ‘x’ in the corresponding cell; otherwise leave blank.**
4. **Remove repeating rows.**
5. **Mark columns (roles) with the same pattern of values ‘x’/blank in constructed matrix, and assign to each such pattern an entity sub-type name.**
6. **In general, an entity type Y is a sub-type of entity type Z iff for each row where Y =‘x’, Z = ‘x’.**
7. **Draw the sub-type graph and delete any transitively implied arcs.**

Example (cont)

Article	Code	Colour	Price	Size	Neck
Dress	134-6	Red	50.00	14	-
Dress	214-5	Yellow	45.00	12	-
Skirt	712-0	Red	34.95	12	-
Shirt	615-8	White	25.00	-	90
Shirt	547-2	White	50.00	-	85
Shirt	615-9	White	25.00	-	90

Article	Code	Colour	Price	Size	Neck
X	X	X	X	X	-
X	X	X	X	X	-
X	X	X	X	X	-
X	X	X	X	-	X
X	X	X	X	-	X
X	X	X	X	-	X

Article	Code	Colour	Price	Size	Neck
X	X	X	X	X	-
X	X	X	X	-	X
A				B	C

In general -

If for every cross in the column representing a set Y there exists a cross in the same rows in a column representing the set Z, then we say that there is a subtype relationship between Y and Z, more precisely: Y is a subtype of Z (notation: Y→Z).

X	X
	X
X	X
	X
X	X
X	X
Y	Z

Y is a subtype of Z.

Y→ Z

Each cross in column Y has a coresponding cross on the same level in the Z column

X	X
X	
X	X
X	X
	X
	X
X	X
Y	Z

Y is not a subtype of Z.

Z is not a subtype of Y

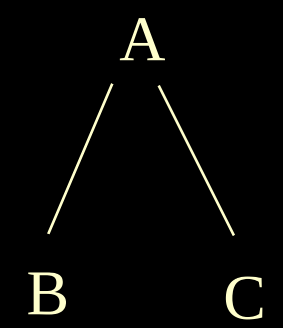
The cross in the second row in column Y has no a coresponding cross on the same level in the column Z. Crosses in the fifth and sixth row in the column C have no corresponding crosses in the column Y

Example (cont)

Article	*Code*	*Colour*	*Price*	*Size*	*Neck*
X	*X*	*X*	*X*	*X*	-
X	*X*	*X*	*X*	-	*X*
A				B	C

The subtype relationships are:

B → A and C→ A



Processing of the object role matrix

Example

Role/ Instance	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10
1	x	x	x	x						
2	x	x	x	x						
3	x	x			x	x	x	x	x	x
4	x	x			x	x			x	x
5	x	x			x	x	x	x	x	
6	x	x								x

Repeating rows are removed

Example

Role/ Group	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10
(Instance 1&2) 1	x	x	x	x						
2	x	x			x	x	x	x	x	x
3	x	x			x	x			x	x
4	x	x			x	x	x	x	x	
5	x	x								x

Columns with the same pattern are identified … *Example*

Role/ Group	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10
(Instance 1&2) 1	1	1	1	1						
2	1	1			1	1	1	1	1	1
3	1	1			1	1			1	1
4	1	1			1	1	1	1	1	
5	1	1								1

A B C D E

Step 5

Example

Role/ Group	r1	r2	r3	r4	r5	r6	r9	r7	r8	r10
(Instance 1&2) 1	x	x	x	x						
2	x	x			x	x	x	x	x	x
3	x	x			x	x	x			x
4	x	x			x	x	x	x	x	
5	x	x								x
	A	A	B	B	C	C	C	D	D	E

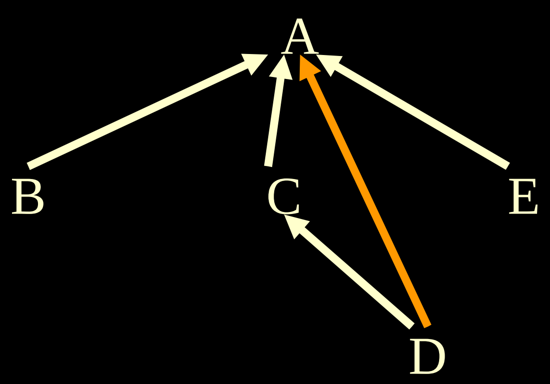
… and grouped (for convenience

r1	r2	r3	r4	r5	r6	r9	r7	r8	r10
x	x	x	x						
x	x			x	x	x	x	x	x
x	x			x	x	x			x
x	x			x	x	x	x	x	
x	x								x
A		B		C			D		E

Relatonships between column patterns identified and consequently relationships between subtypes

B → A, C → A, D → A, E → A, D → C

Every instance (row) with a set of roles B also has a set of roles A.
Every instance (row) with a set of roles C also has a set of roles A.
Every instance (row) with a set of roles D also has a set of roles C
Every instance (row) with a set of roles D also has a set of roles A.
Every instance (row) with a set of roles E also has a set of roles A.

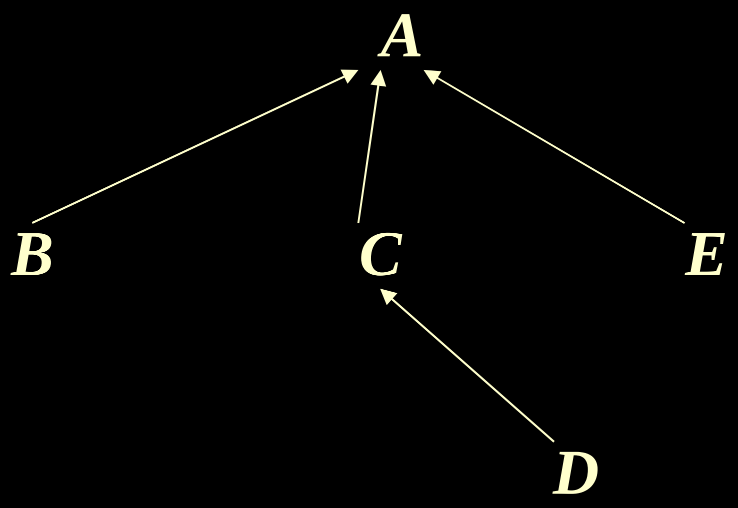


r1	r2	r3	r4	r5	r6	r9	r7	r8	r10
x	x	x	x						
x	x			x	x	x	x	x	x
x	x			x	x	x			x
x	x			x	x	x	x	x	
x	x								x
A		B		C			D		E

The relationship D → A is redundant as it can be inferred from D → C and C → A (transitivity)

Example (cont)

Draw the sub-type graph and delete any transitively implied arcs.



Summary

- **This lecture covered construction of additional constraints:**
  – Complex uniqueness constraints,
  – Subsets constraints and
  – Subtype constraints.
  –
- **The next lecture will continue discussion on sub-typing construction and the final checks.**