# *Zawansowane Modelowanie i Analiza Systemów Informatycznych*

***(l-6 short)***

**Polsko-Japońska Wyższa Szkoła Technik Komputerowych**
**Katedra Systemów Informacyjnych**
**2013**

**Transformation of ORM schema to the RDB**

## *Relational schema design*

**An ORM conceptual schema can be mapped into a relational database schema by the mapping process (sometimes also called transformation).**

**The mapping provides relations in Optimal Normal Form: a set of normalised tables to store the information permitted by the conceptual schema.**

# Relational schema design

**The input to the process of relational schema design is ORM conceptual schema**
**The output is the list of**

- **Table names**
- **for each table the list of its column names**
- **the list of columns that form table key(s)**

**The process is completed with the identification of foreign keys.**

**Alternative terminology**

| | |
|---|---|
| **Table** | **relation** |
| **Column** | **attribute** |

**Also we distinguish between simple (or basic) entity type and complex or compound entity types The latter correspond to nested fact types.**

# *Outline of the transformation algorithm*

- *By an attribute we understand an entity type (basic) combined with a role which it plays in a fact type.*
- *The names of attributes are (should be) typicaly derived from the semantics of the roles the entity types play in the fact types*
- *Each non-nested fact type generates a relation*
- *If a nested fact type plays a role in a non-nested fact type than it should be represented in the relation schema by all attributes 'contributing' to this nested fact type (possibly recursively)*
- *The names of the relations are generally determined by the combinations of key attributes*
- *Two relations with the same sets of keys should be combined into one relation.*

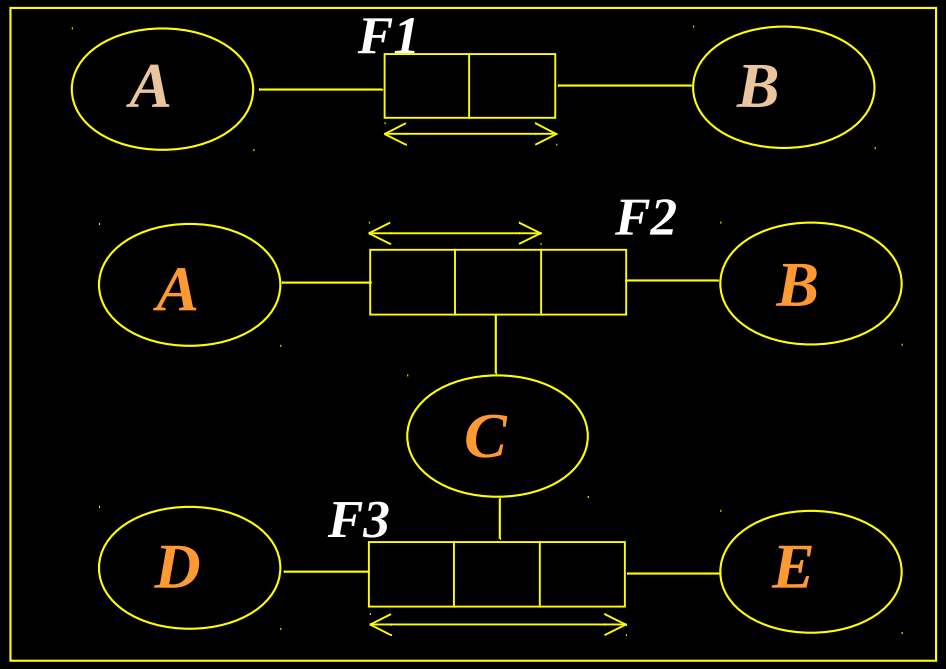
# *Procedure of RDB design*

***On the next slides, for the sake of simplicity, we assume that the names of entity types involved in fact types correspond to their roles in these fact types.***
***Note: Nested fact types are not considered yet***

- **Create a RELATION for each FLAT FACT TYPE**
- **The uniqueness constraint (UC) of the fact type is the key of the relation.**
- **The name of the fact type depends of the type of the uniqueness constraint:**
- **Multiple roles UC – Name of the fact type (Rule 1)**
- **Single role UC – Name of the entity type touching that role (rule 2)**

- **Keys could be shown by underlining their attributes**

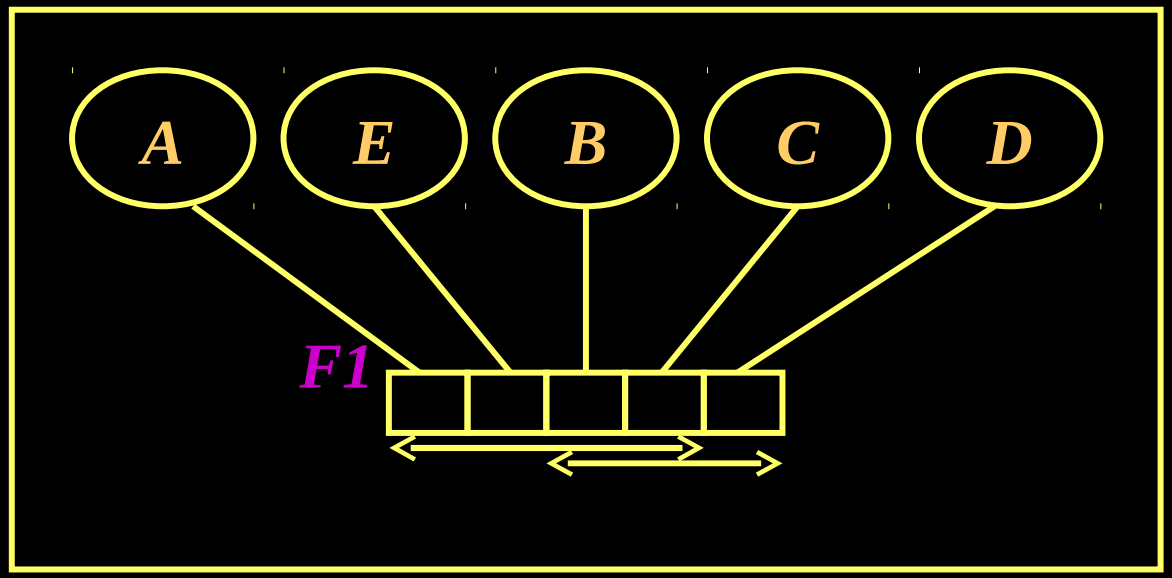
# In general




| | |
|---|---|
| F1(AB) | Key AB |
| F2(ABC) | Key AC |
| F3(CDE) | Key CDE |

**If there are more uniqueness constraints, then each of them corresponds to a key of the relation.**

***One more example***




***F1(ABCDE)*** ***Keys ABCE and BCD***

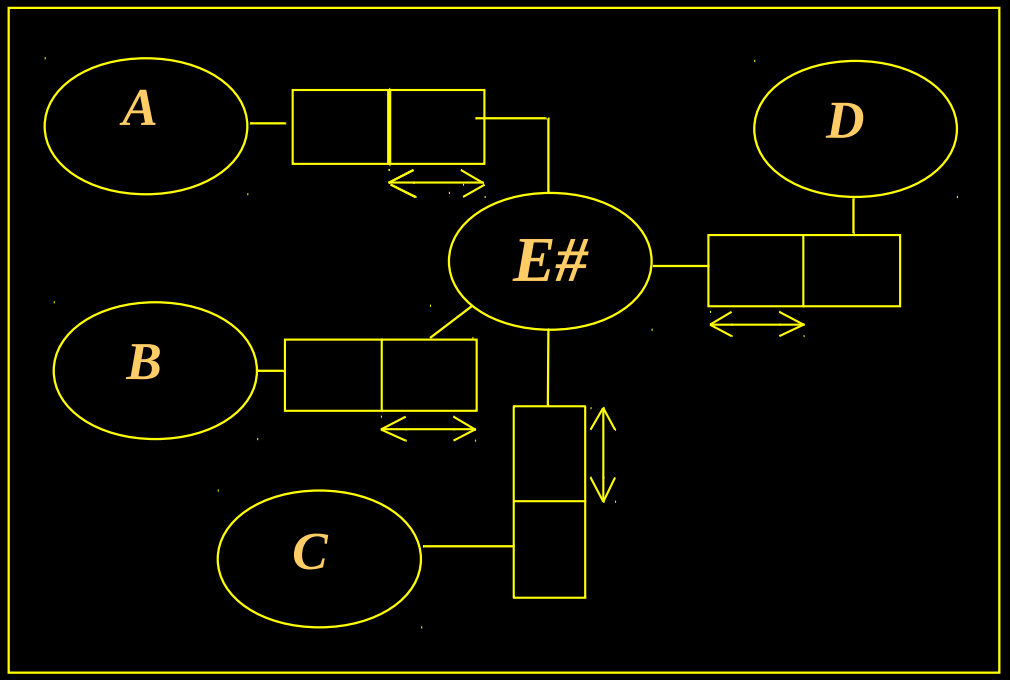
## *Single role uniqueness constraints (rule 2)*

**All binary fact types involving an entity type E, which have touching roles with that entity type covered by single UC, contribute to the relation which has**

**Attributes – Entity types playing roles not covered by UC and identifier of E**

**Key - identifier of E**

**Name - The name of the entity type is to be considered as the name for the relation**



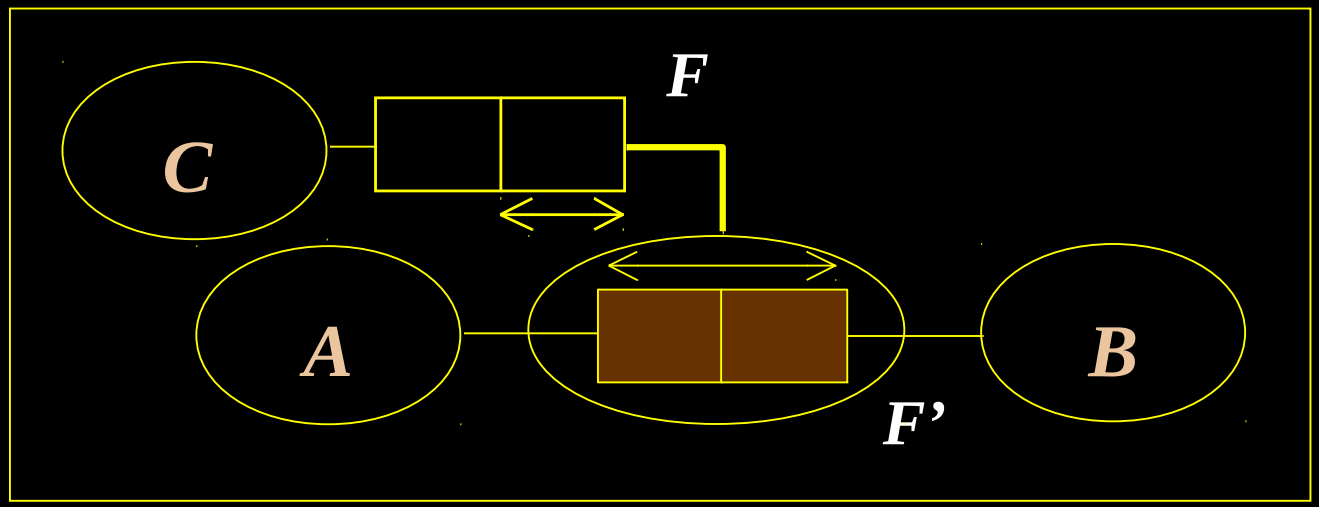

*E (E# A B C D)  Key  E#*

**If a nested fact F' type plays a role in the flat fact type F then**

**Create a relation for that flat fact type as that role would be 'played' by simple entity type**

**Use temporarily the name of that nested fact type, F', as one of the attributes of that relation**

**Substitute F' with the attributes that are generated by F'**




***Relation F' C***

***.. and after substitution → A B C***

**If a nested fact type F' plays a role covered with single uniquenes constraint in F then**

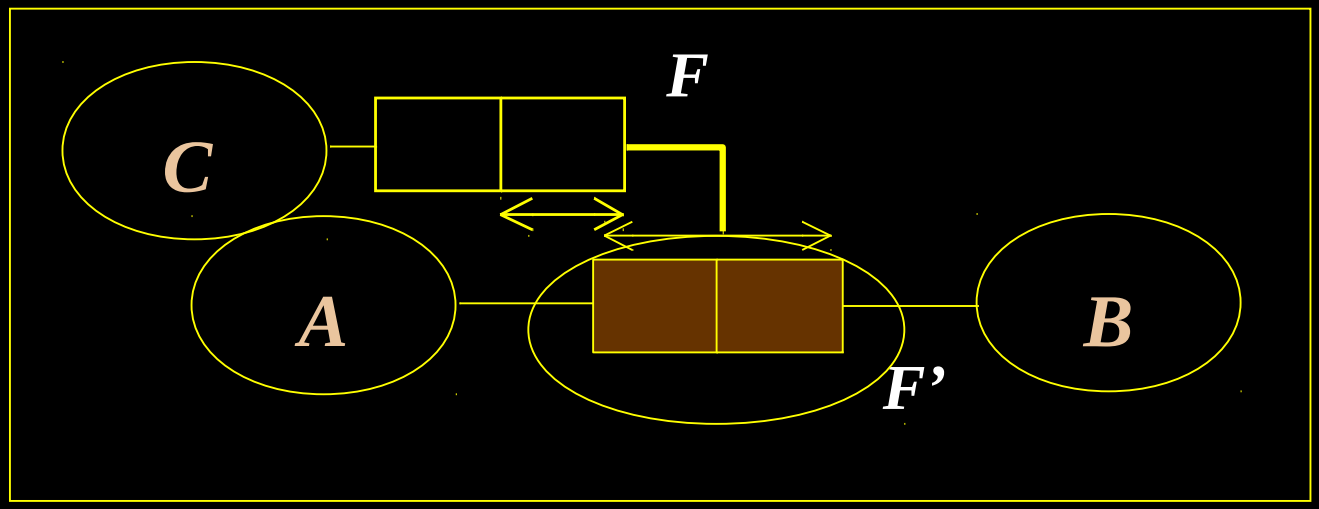
**The name of the relation is corresponding to F'**

**The key(s) of that relation are uniqueness constraints of F'**

**If a role of nested fact type F' covered with a multiple role uniquenes constraints in F then**

**The name of the produced relation is corresponding to F**

**The key(s) of that relation are determined as concatenations of uniqueness constraints of F' and uniqueness constraints of F**



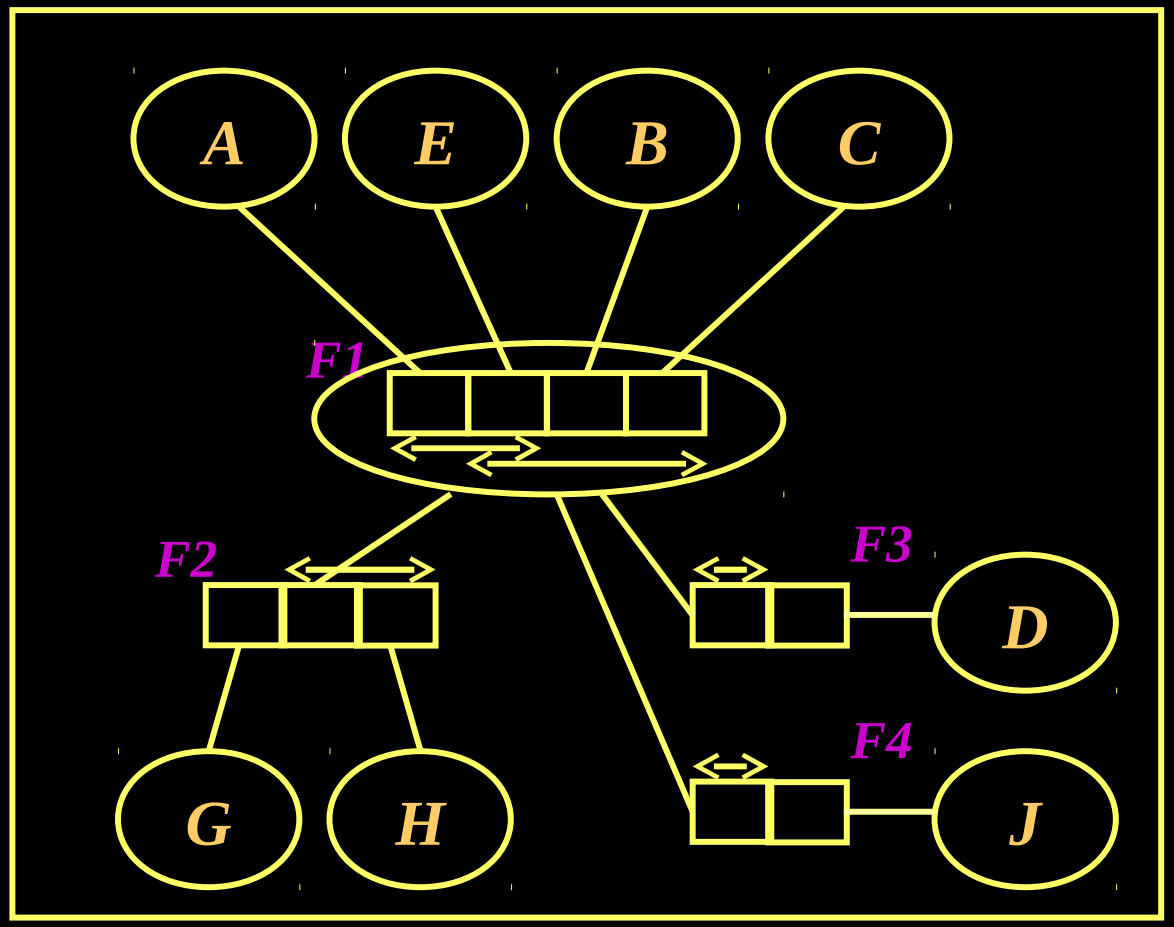

***Relation produced F' (A B C) with key AB***

## *One more example*

*F3: F1 (F1, D) key F1*
*F4: F1 (F1, J) key F1*
*Combine above into one table (the same key)*
*F1 (F1, D, J) key F1*

*F2: F2 (F1, G, H)*
*with key F1 H*

*After substitution of F1 with contributing attributes*

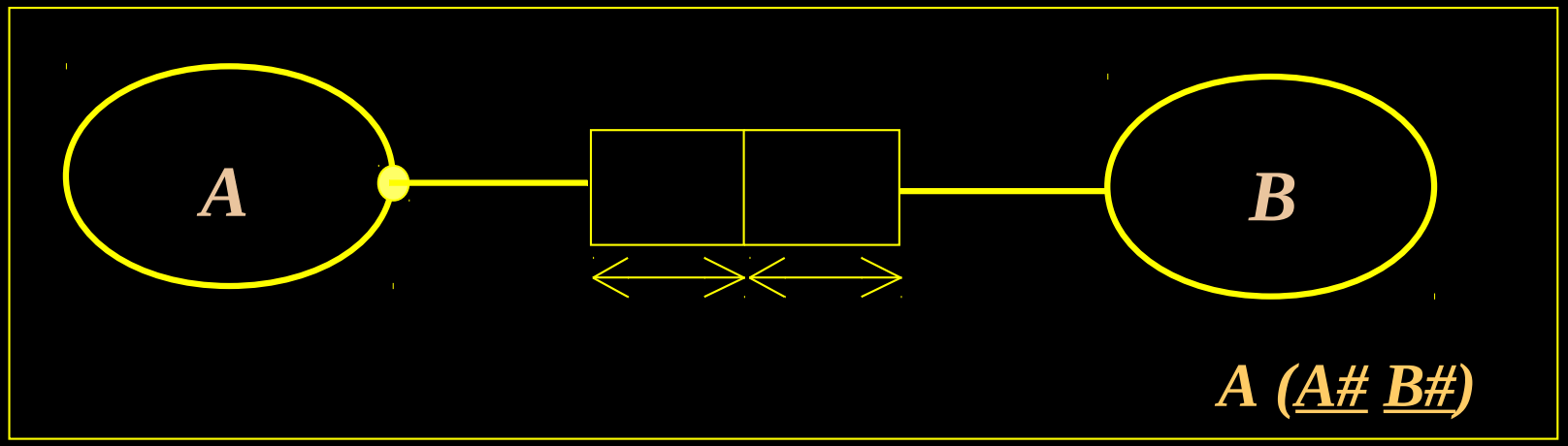



*Resulting tables are*

| | |
|---|---|
| *F1(A B C E D J)* | *keys AE and BCE* |
| *F2 ( A B C E G H)* | *keys A E H and B C E H* |

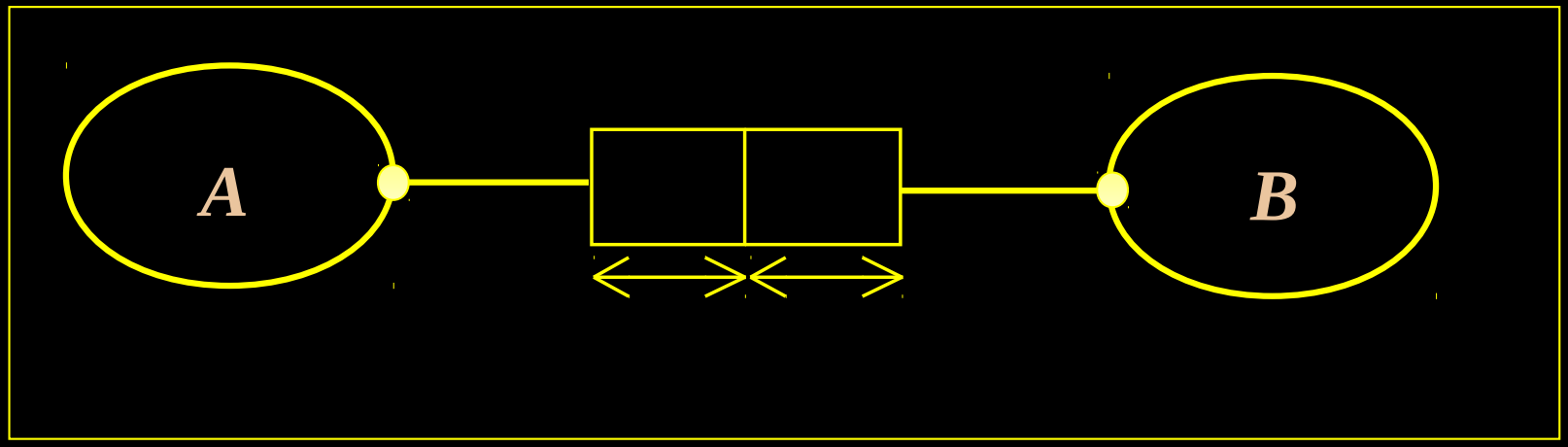
**Special cases:**

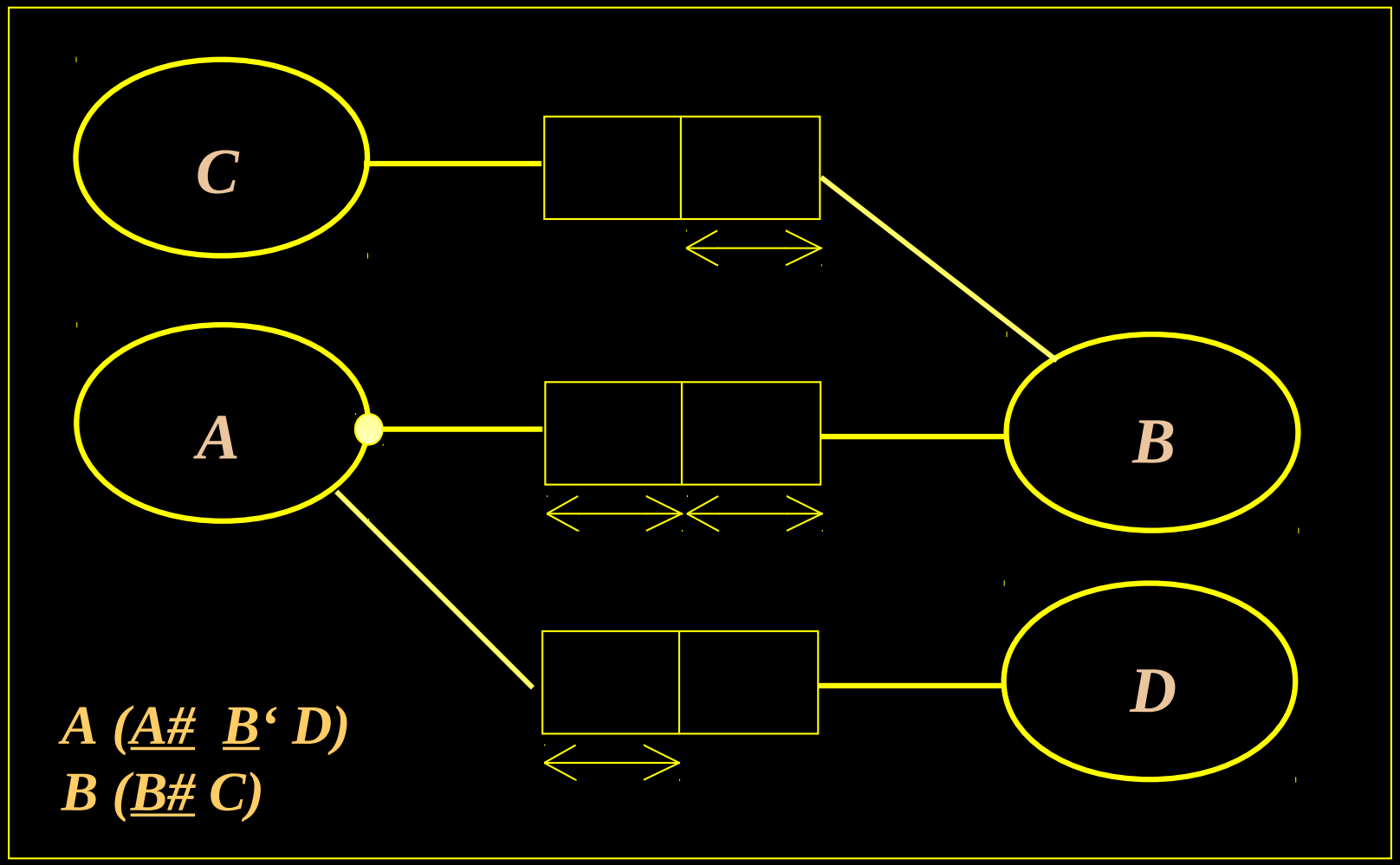
**For each binary fact type with both roles covered with simple uniqueness constraint, include this fact in only one relation. Include it in the entity type that has a total role. Failing this, include it in the entity type which minimises the number of relations in the schema.**



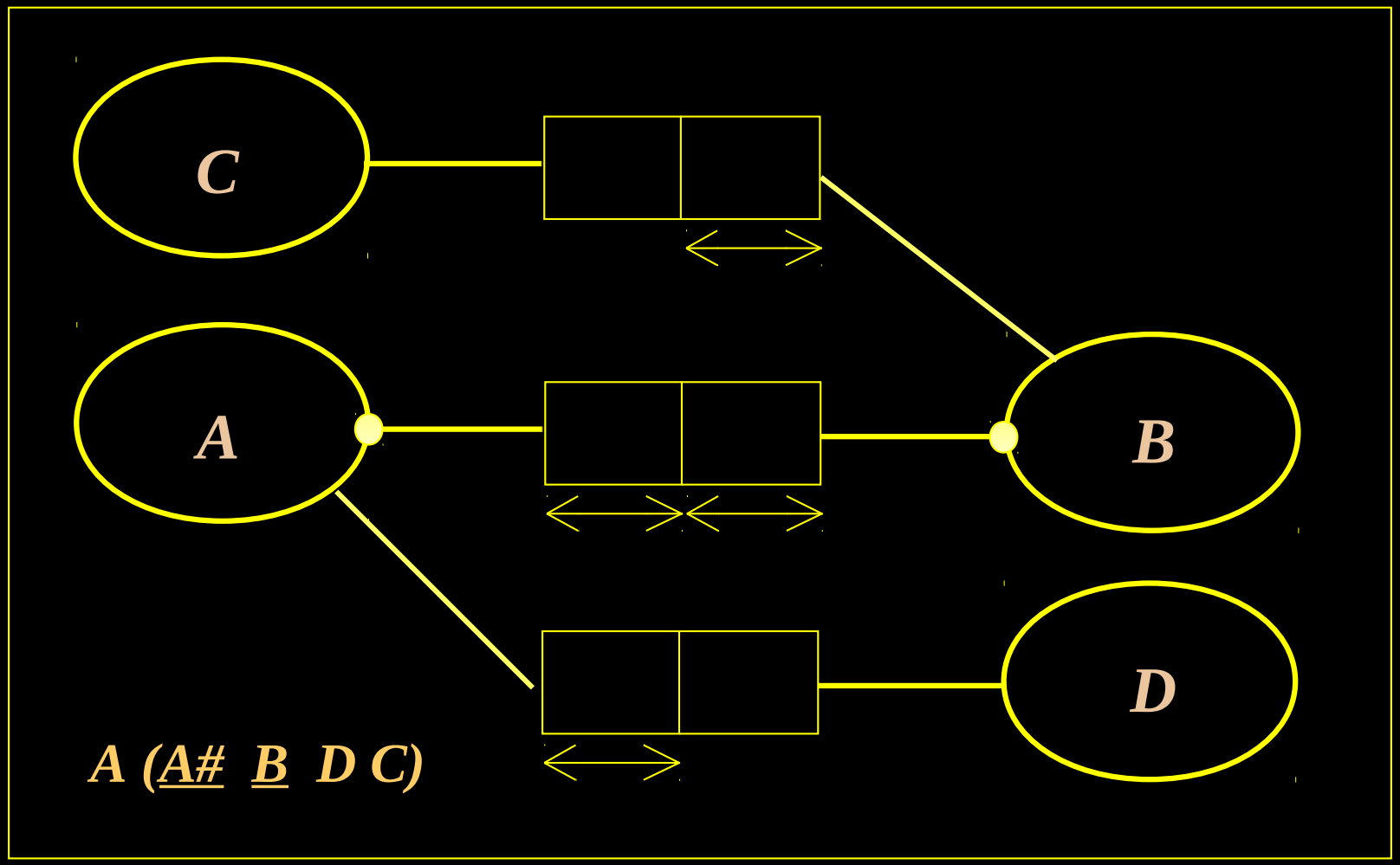
**Special cases (cont):**

**If each role in a binary fact type has a single uniqueness constraint on it, and both roles have total roles include this fact in only one entity type. The identifier of either entity type serves as the key of the resulting relation.**



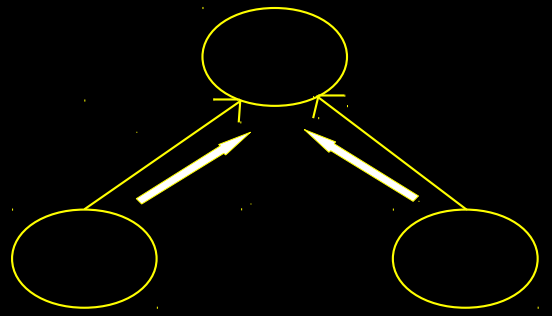



*Note that the attribute B’ in the first relation has different meaning than B# in the second one*
*Eg A- department , B’ – Manager, B - employee*

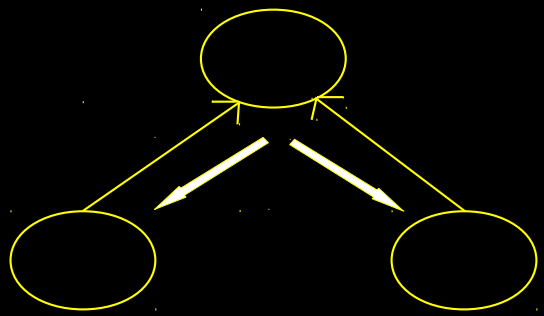

C
A
B
D
A (A# B D C)

## Special cases:

*Subtypes have many possibilities. At one extreme, treat the subtype structure as if it were one entity type*

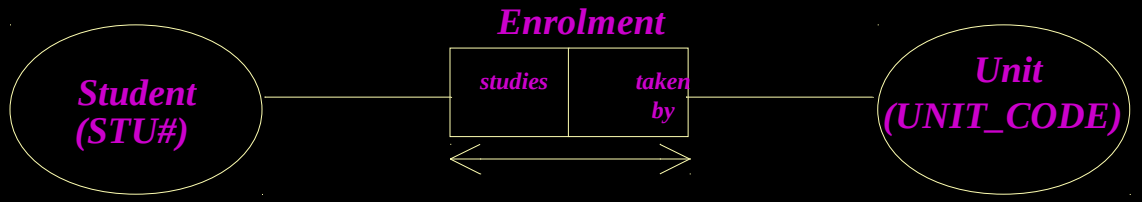


*At another extreme, create a relation for each subtype i.e. 'pulling down' the supertype's fact types into each subtype. Any compromise between the two extremes can be used.*



# SCHEMA TRANSFORMATION EXAMPLES

## Example 1.




**The relation Enrolment is formed with two attributes:**

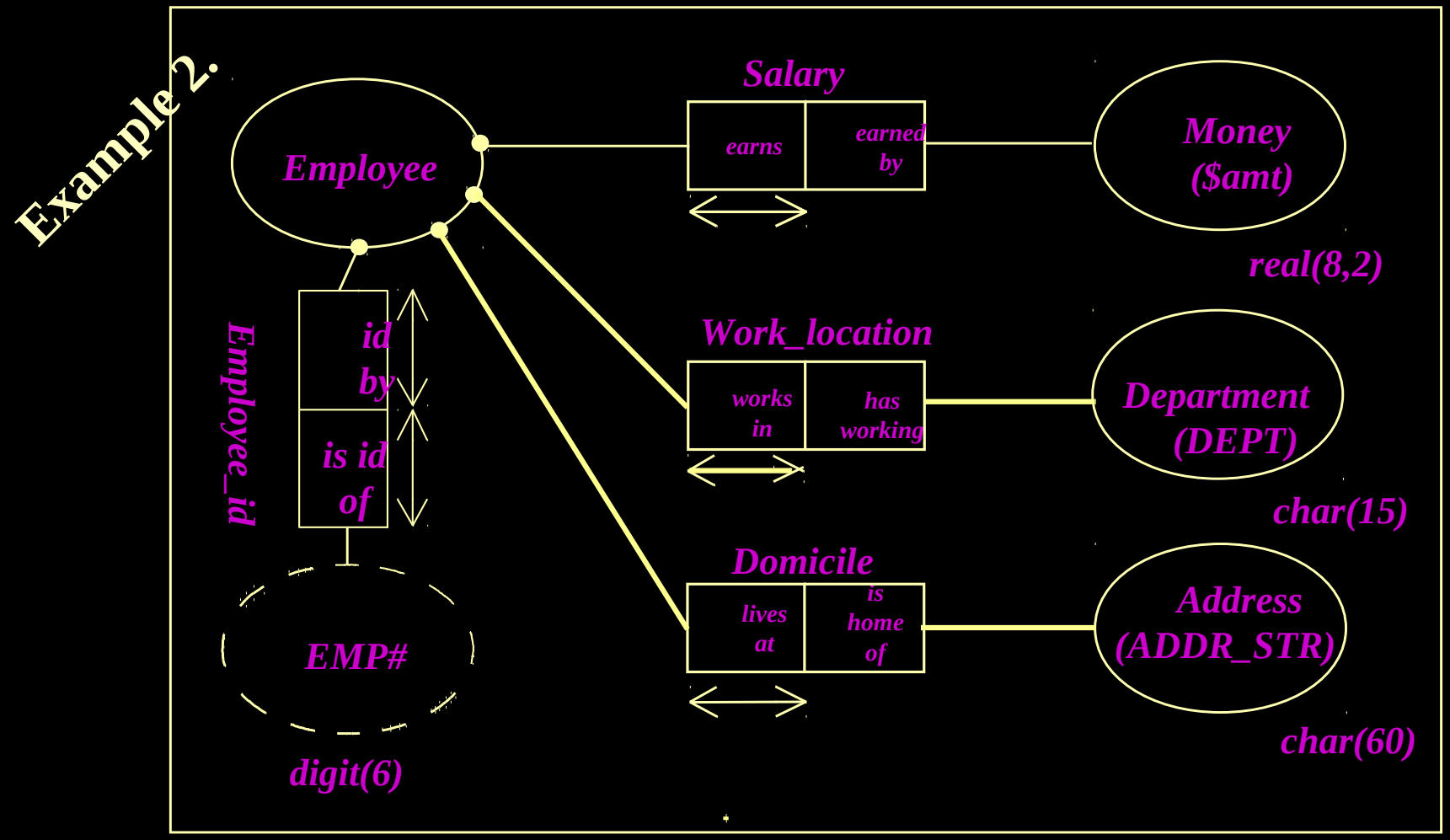
**Student number and Unit code**

**Enrolment (St#, Unit_Code)**

**The key for the relation is the combination**

**St#, Unit_Code**

| Enrolment St# | Unit_Code |
|---|---|
| | |




**Build a relation around Employee**

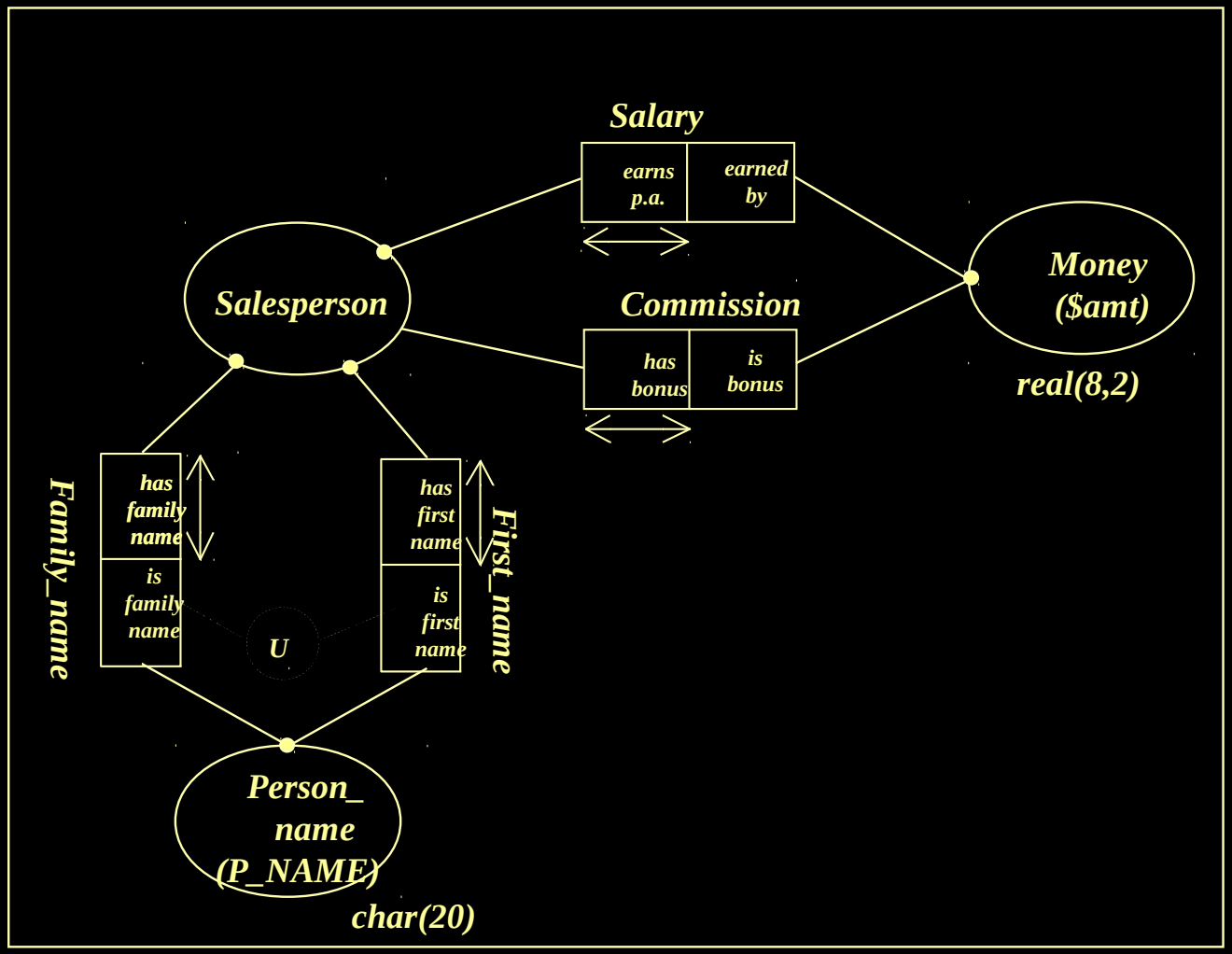
## *The Semantics of Relation:*

***All these binary FT describe certain properties of employees.***
***The natural name for the relation is Employee.***
***Its identifier is key is Employee Id***
***Other attributes are describing employee:***

- ***Empl Address,***
- ***Employing Department and***
- ***Salary***

The relation created is :
Employee (Empl# , Dept, Salary, H_Addr)  Key:  Emp#

*Example 3.*




***Build a relation around Salesperson As in example 2***

***SALESPERSON:***

| *sman identity* | *earned by* | *is bonus (opt)* |
|---|---|---|

*SALESPERSON: (SalesPerson ID, Salary, Bonus)*
*Key would be* ***SalesPerson ID 'if existed'***

***Notice, That SALESPERSON entity type is not identified in 1-1 way by a single label type SalesPerson ID – no such label type)***
***However the schema provisdes identification of SalesPerson using a combination of names***

***The resulting relation is***
***SALESPERSON: (SP_Fname SP_Lname, Salary, Bonus)***
***With key SP_Fname SP_Lname***

***SALESPERSON:***

| Salesperson | | Salary | Bonus (opt) |
|---|---|---|---|
| SP first name | SP last name | | |

**Two points:**

1. **the key extends over two roles,**
2. **the is bonus column is an optional column; null values are allowed. This is deduced from the lack of a total role constraint.**

**COLUMN NAMES:**

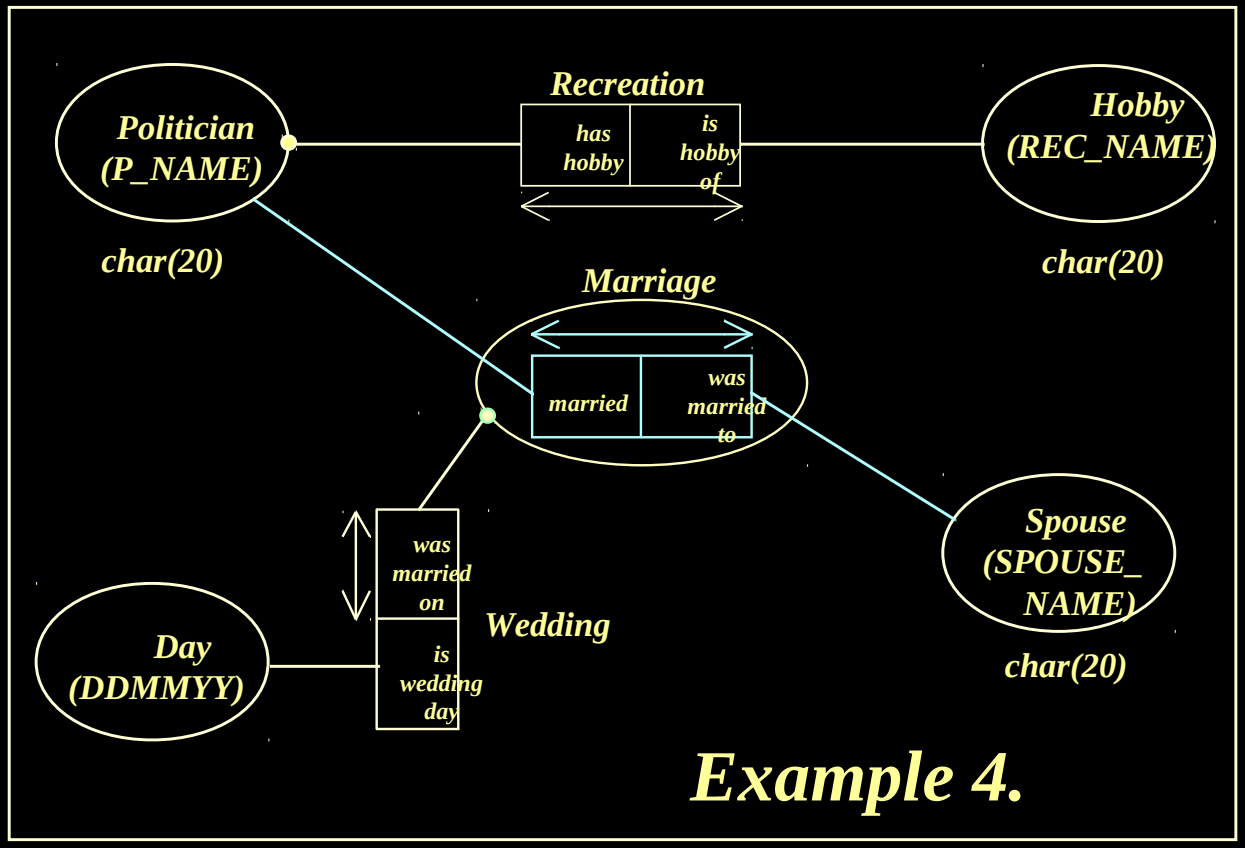
**Column names have to be chosen with care. The role name is the semantically correct choice of name, but other candidates can be the fact type name, label type name, or even the entity type name. The designer should make a sensible choice.**

**Preferred column names in this course:**

**Entity types or their Label names with roles in which they are involved.**

**RELATION NAMES:**

**Derived either from fact type name or from the entity type name. depending on the length of the key.**




*Recreation →*

*(PolName, Hobby)* *Key* *PolName, Hobby*

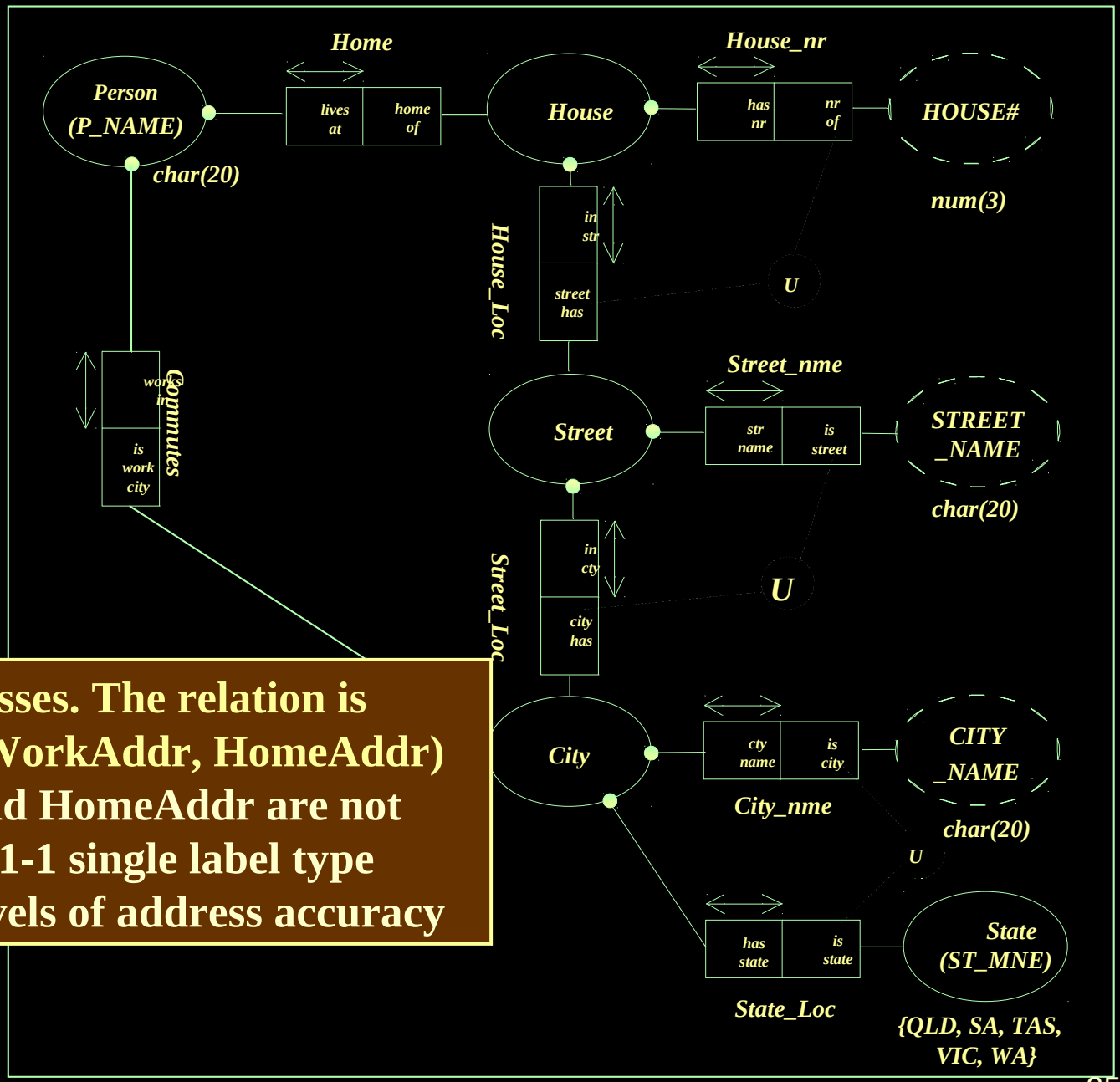
*Wedding →*

*Marriage (Marriage ,WeddingDate) →*

*Marriage (PolName, SpouseName ,WeddingDate)*

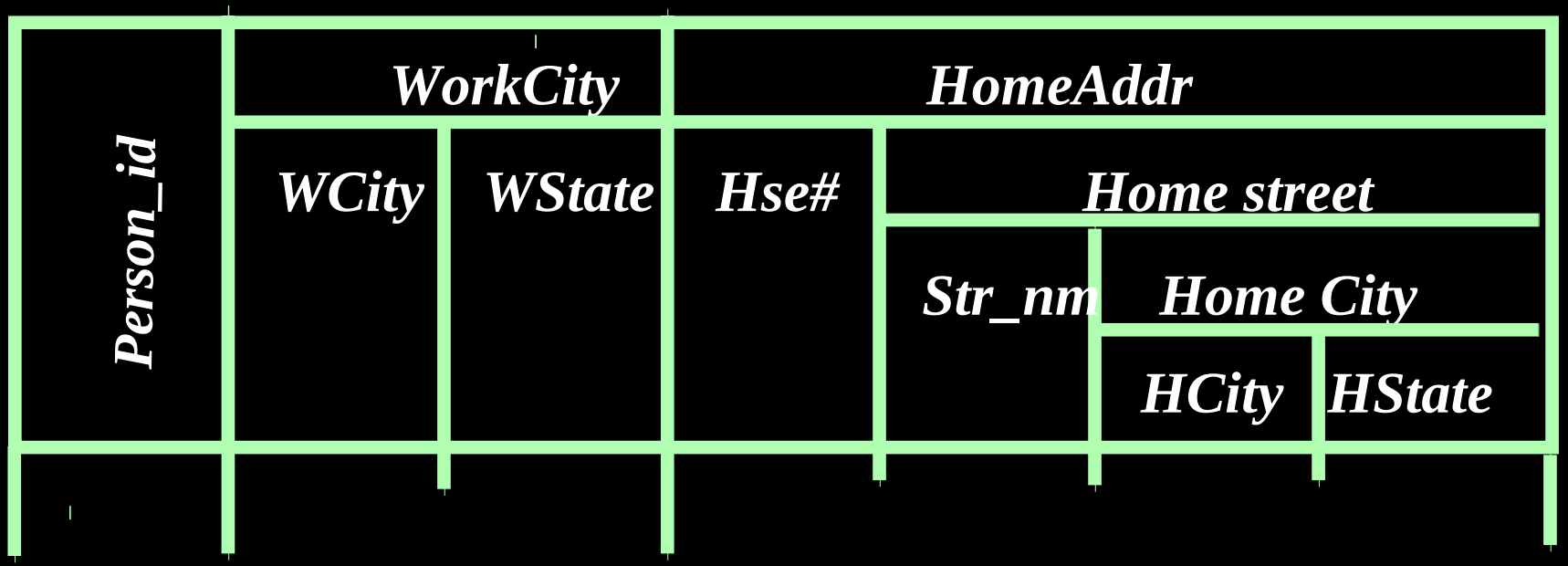
*Key is PolName. SpouseName*

# *Example 5.*




**Person addresses. The relation is Person(Pname, WorkAddr, HomeAddr) Work Addr and HomeAddr are not identified by 1-1 single label type Note different levels of address accuracy**

## *The recursive view on the addresses Ids*

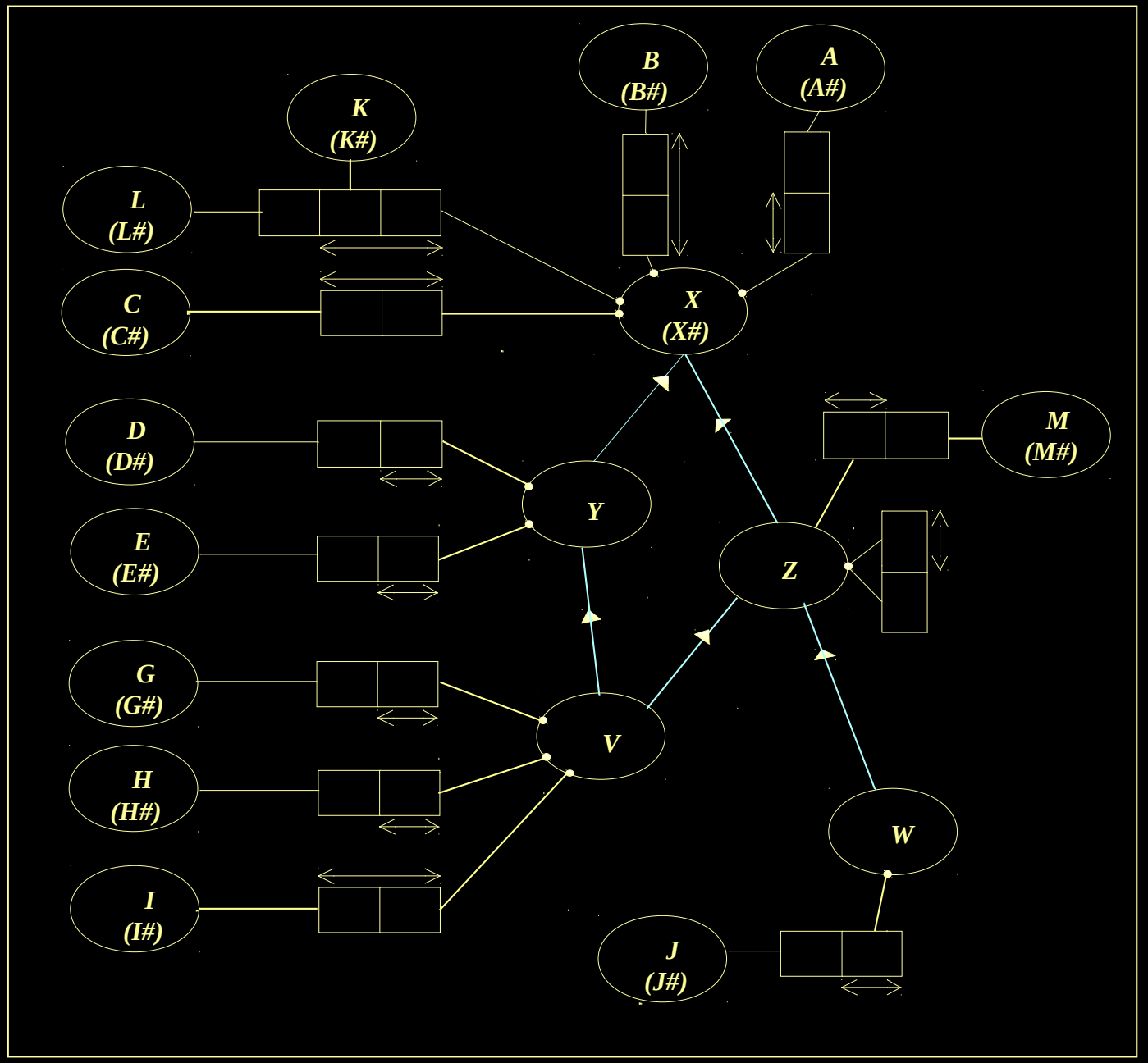



*The resulting relation is as follows*

*Person (PersName, WorkCity, WorkState, Hse#, Hstr, Hcity,Hstate)*
*With key PersName*

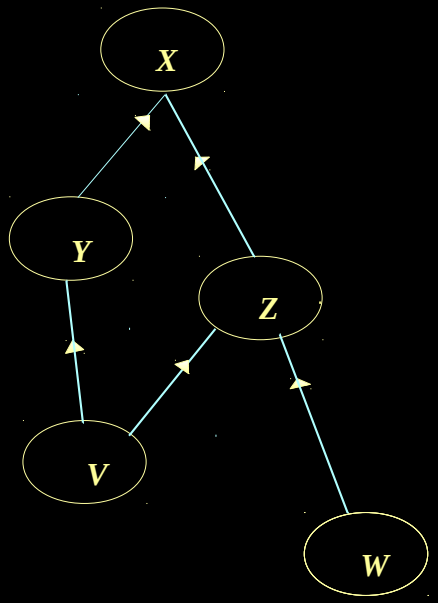
# Example 6: Subtyping

| | |
|---|---|
| R1 | X B |
| R2 | X K L |
| R3 | X C |
| R4 | V I |
| R5 | X A |
| R6 | Y D E |
| R7 | Z M Z' |
| R8 | V G H |
| R9 | W J |

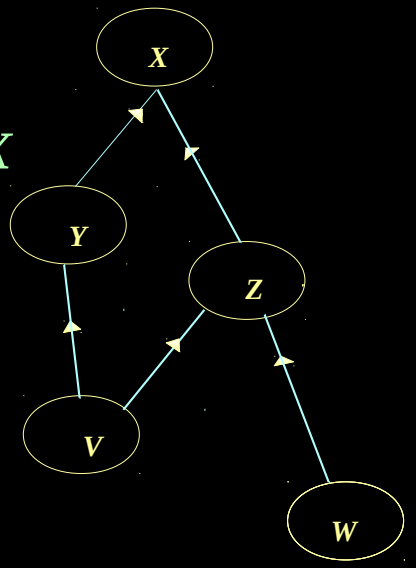


## *ABSORBED:*

| | *Y by X* | *Z by X* | *Y,Z by X* |
|---|---|---|---|
| *X B* | *X B* | *X B* | *X B* |
| *X K L* | *X K L* | *X K L* | *X K L* |
| *X C* | *X C* | *X C* | *X C* |
| *V I* | *V I* | *V I* | *V I* |
| *X A* | *X A D E* | *X A M X'* | *X A D E M X'* |
| *Y D E* | | *Y D E* | |
| *Z M Z'* | *Z M Z'* | | |
| *V G H* | *V G H* | *V G H* | *V G H* |
| *W J* | *W J* | *W J* | *W J* |



## *ABSORBED:*




| *W by Z* | *Z by X and V by Y* | *V, Y, Z by X* |
|---|---|---|
| ***X B*** | ***X B*** | ***X B*** |
| ***X K L*** | ***X K L*** | ***X K L*** |
| ***X C*** | ***X C*** | ***X C*** |
| ***V I*** | ***V I*** | ***Y I*** |
| ***X A*** | ***X A*** | ***X A M X'*** |
| ***Y D E*** | ***Y D E*** | ***Y D E G H*** |
| ***Z M Z'*** | ***Z M Z' J*** | |
| ***V G H*** | ***V G H*** | |
| ***W J*** | | ***W J*** |

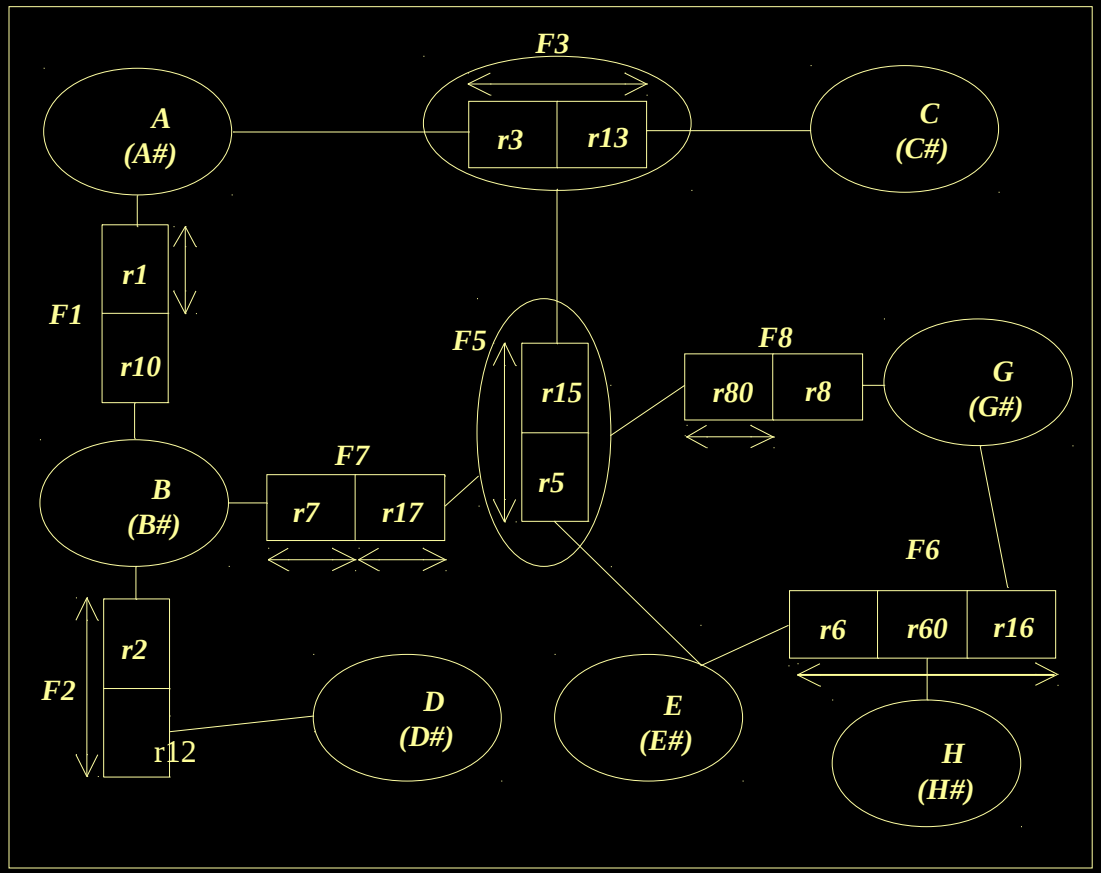
| *X B* |
|---|
| *X K L* |
| *X C* |
| *X I* |
| *X A D E M X' G H* |
| *W J* |

***How the design looks like if ALL subtypes are ABSORBED by supertype X?***

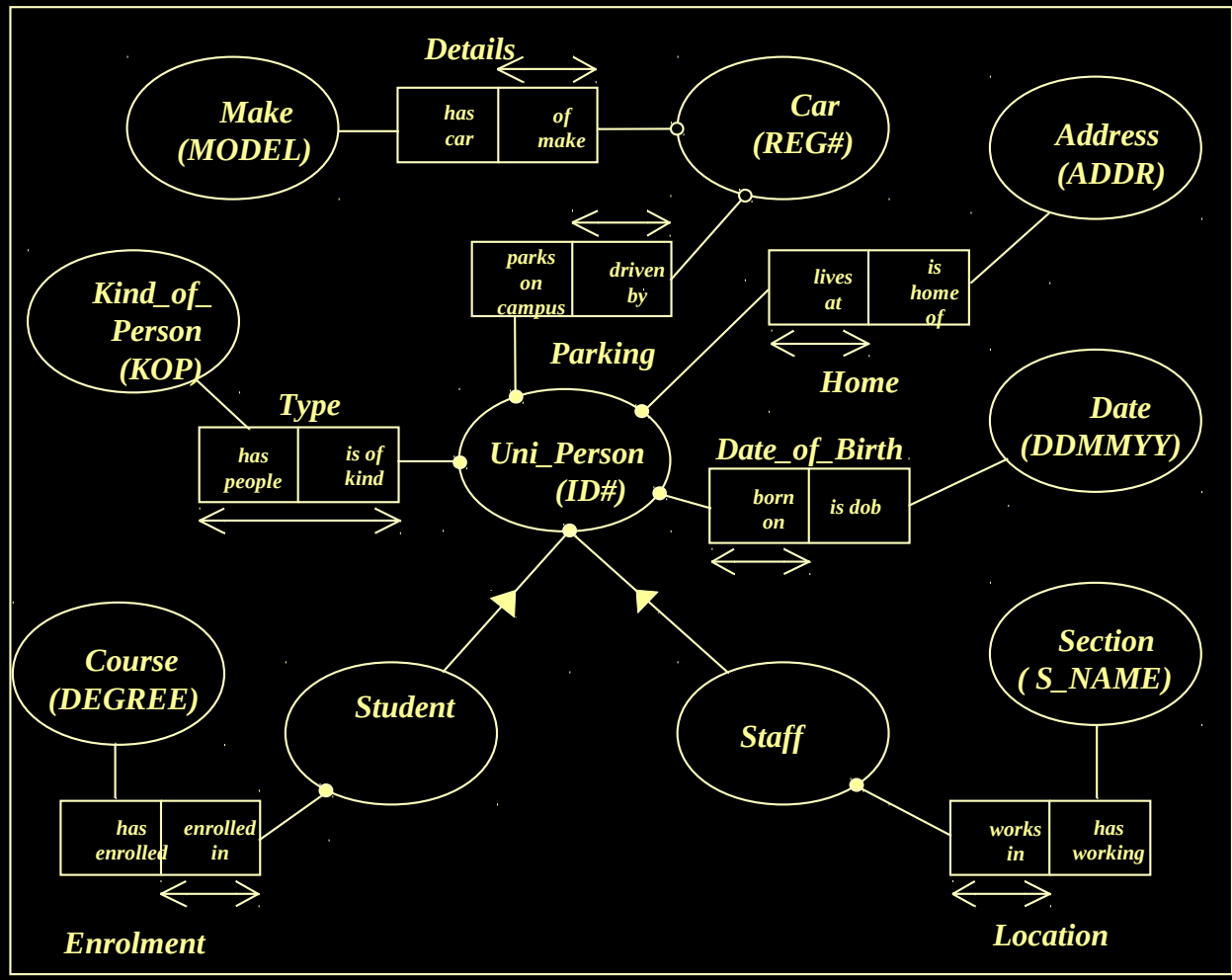
# Example 7

F1 → A (A# , B10)
F2 → F2 (B2, D12)
F6 → F6 (E6,H60,G16)
F8 → F5 (F5, G8) =
F5 (F3, E5, G8) =
F5 (A3, C13, E5, G8)
F7 → F5 (B#, F5) =
F5 (B#, A3, C13, E5)
With B# as alternative key

After combining with previous relation with the same key:
→→ F5 (A3, C13, E5, G8, B#)
With B# as alternative key



# Example 8: further foreign keys




**Type:** | ID# | Kind_of_person |

ID# references Uni_person

**Car:** | REG# | MODEL | ID# |

ID# references Uni_person

**Uni_person:** | ID# | Dob | Home |

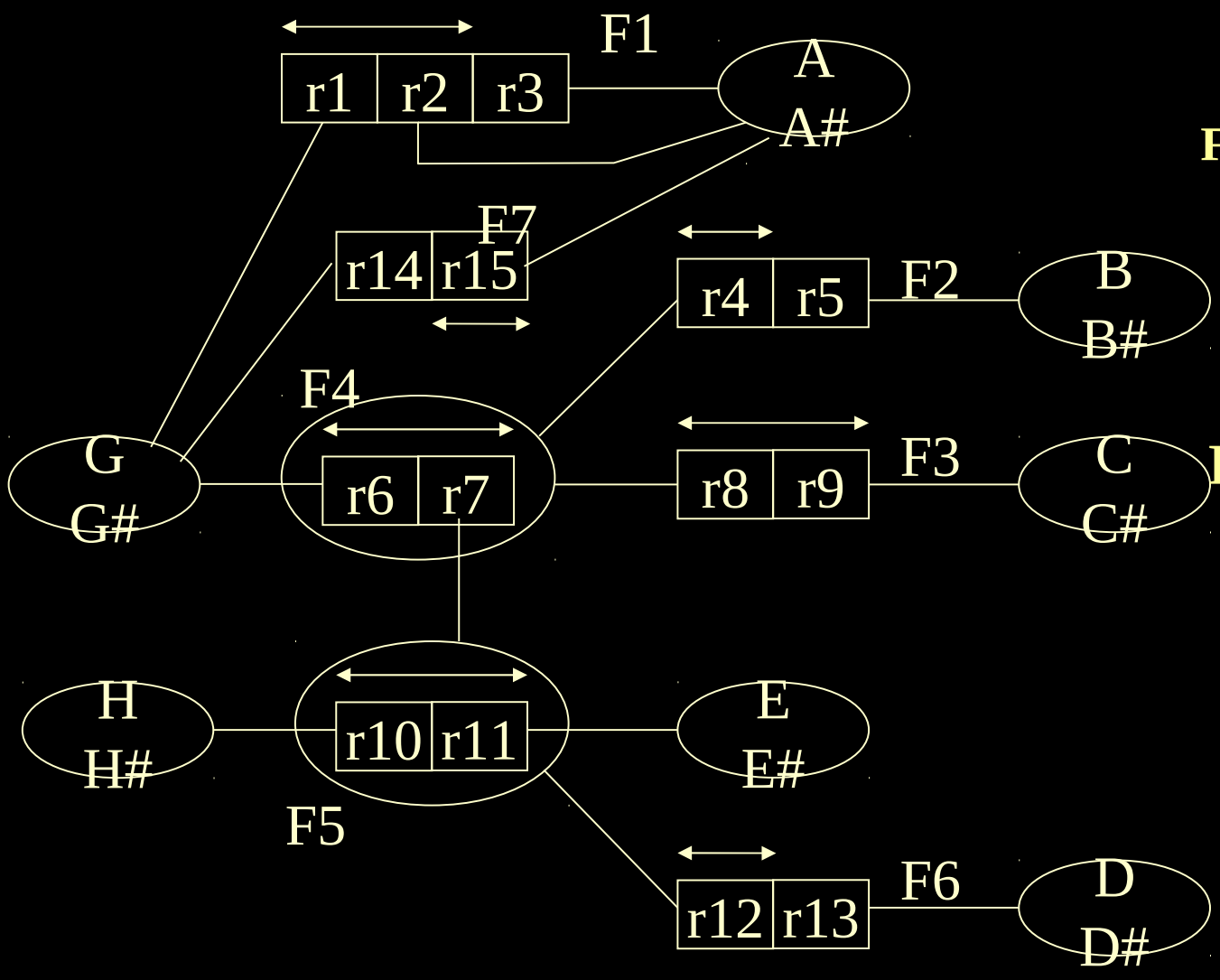
**Student:** | ID# | DEGREE |

ID# references Uni_person

**Staff:** | ID# | S_NAME |

ID# references Uni_person

# *Example*




**F1: (G1 A2 A3)**

**F2: F4 (F4 B5) →**

**F4 (G6 F5 B5) →**

**F4 (G6 H10 E11 B5)**

**F3: F3(F4 C9) →**

**F3( G6 F5 C9)**

**F3( G6 H10 E11 C9)**

**F6: F5 (F5 D13)**

**F5( H10 E11 D13)**

**F7: A (A#, G14)**

## *Summary*

**This lecture covered detailed transformation of ORM schema to the RDB.**