Modeling ER Features with Functional Dependencies

Functional Dependency Exercises

Objectives

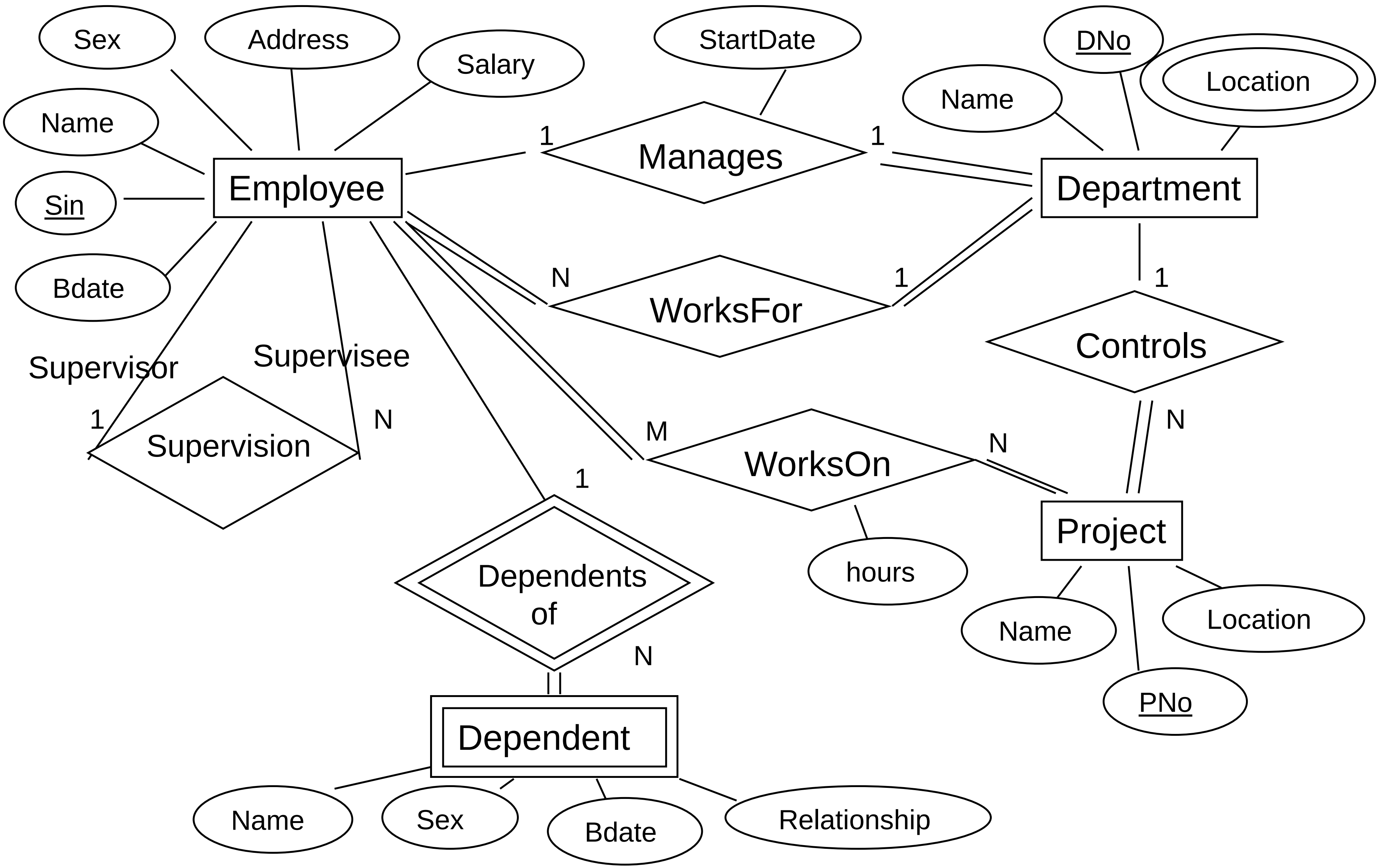
- Learn how Entity-Relationship features can be modeled with Functional Dependencies

- Some practice exercises involving functional dependencies

Topics

- Entities and Weak Entities
- 1:1, 1:N and N:N Relationships
- Multi-valued Attributes
- Inheritance

E-R diagram for Company Data (fig 3.2)



Elmasri Data Set

//Elmasri and Navathe Employee Dataset example

//As per 3005 Notes: Modeling ER with FD.

SIN -> Bdate,Name,Sex,Address,Salary

DNo->DName

PNo->Pname,Plocation

SIN,DepName -> DepSex,DepBdate,Relationship

DNo->Manages_SIN,Manages_StartDate

SIN->WorksFor_DNo

SIN->Supervisor_SIN

PNo->Control_DNo

SIN,PNo->WorksOn_Hours

DNo,Location->temp //remove column temp from final design

Minimal Cover

//Elmasri and Navathe Employee Dataset example

//As per 3005 Notes: Modeling ER with FD.

SIN -> Bdate,Name,Sex,Address,Salary,WorksFor_DNo,Supervisor_SIN

DNo -> DName,Manages_SIN,Manages_StartDate

PNo -> Pname,Plocation,Control_DNo

SIN,DepName -> DepSex,DepBdate,Relationship

SIN,PNo -> WorksOn_Hours

DNo,Location -> temp

3NF Tables

Dependency Preserving, 3NF tables

[<u>SIN</u> | Bdate,Name,Sex,Address,Salary,WorksFor_DNo,Supervisor_SIN]

[<u>DNo</u> | DName,Manages_SIN,Manages_StartDate]

[<u>PNo</u> | Pname,Plocation,Control_DNo]

[<u>SIN</u>,<u>DepName</u> | DepSex,DepBdate,Relationship]

[<u>SIN</u>,<u>PNo</u> | WorksOn_Hours]

[<u>DNo</u>,<u>Location</u> | ~~temp~~]

3NF Tables

Dependency Preserving, 3NF tables that can be joined into a single table (though not what you likely need)

[SIN | Bdate,Name,Sex,Address,Salary,WorksFor_DNo,Supervisor_SIN]

[DNo | DName,Manages_SIN,Manages_StartDate]

[PNo | Pname,Plocation,Control_DNo]

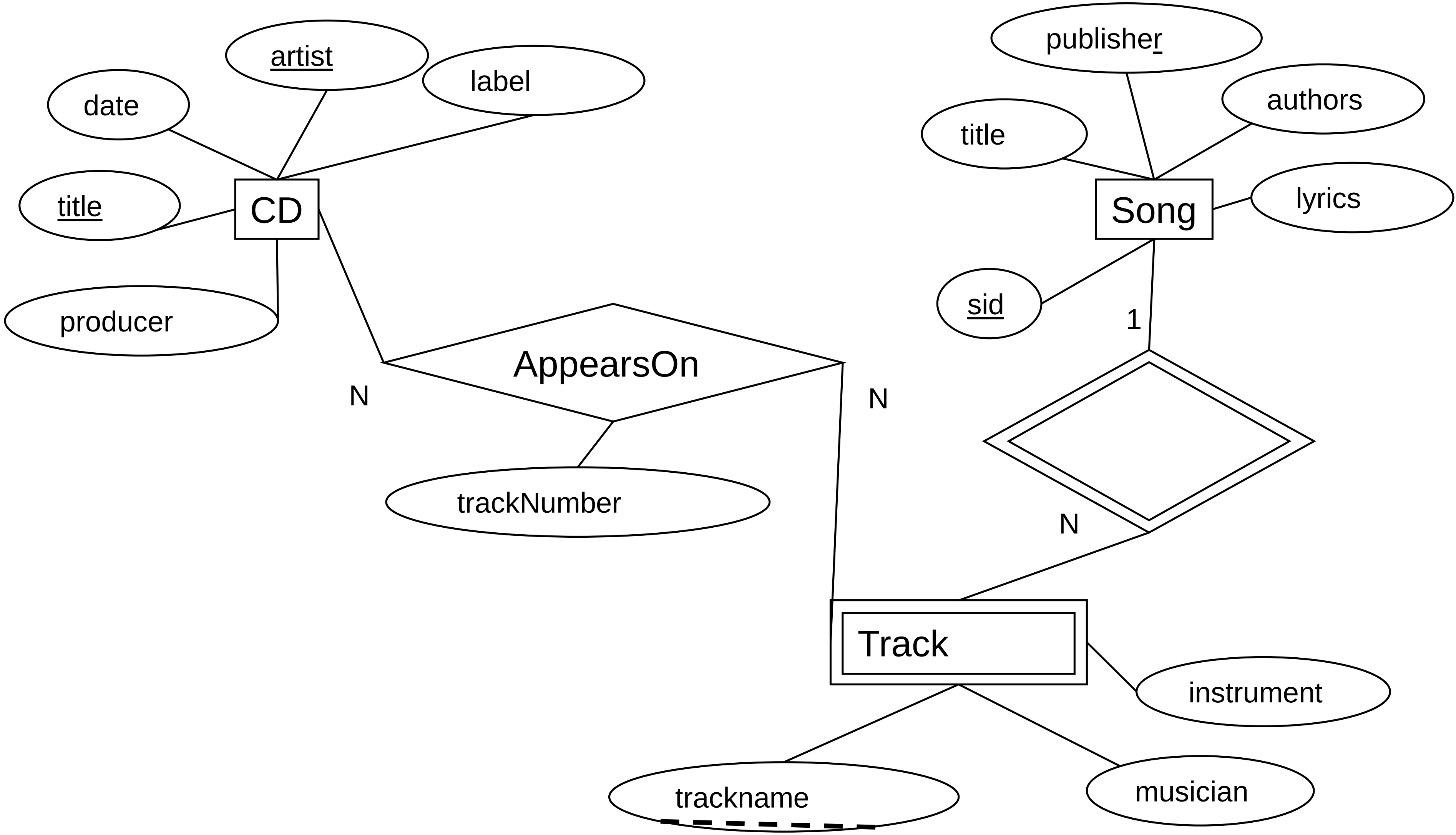
[SIN,DepName | DepSex,DepBdate,Relationship]

[SIN,PNo | WorksOn_Hours]

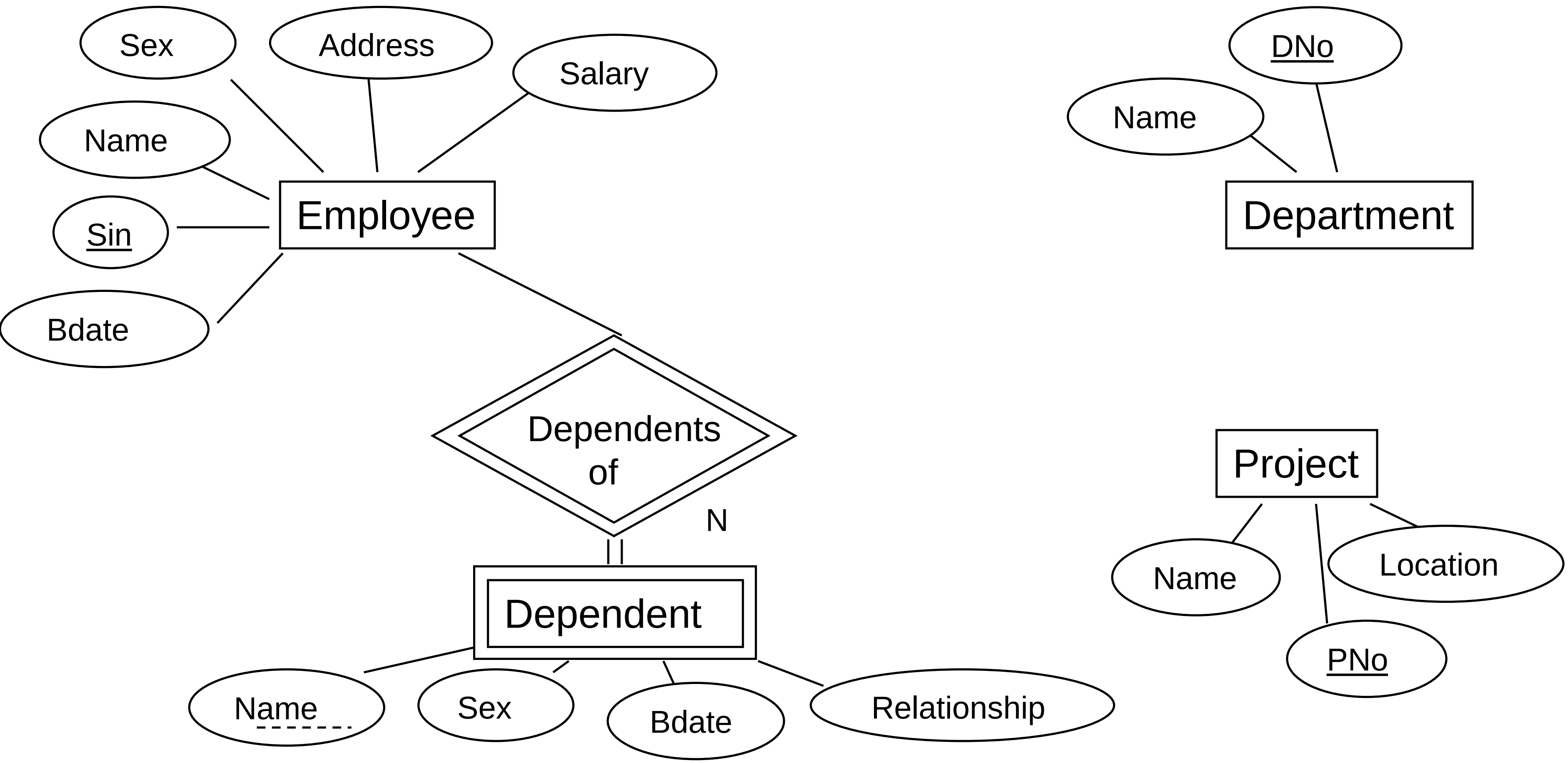
[DNo,Location | ~~temp~~]

[DNo,DepName,Location,PNo,SIN |]

A2, Q1 Possible Solution (others are possible as well)

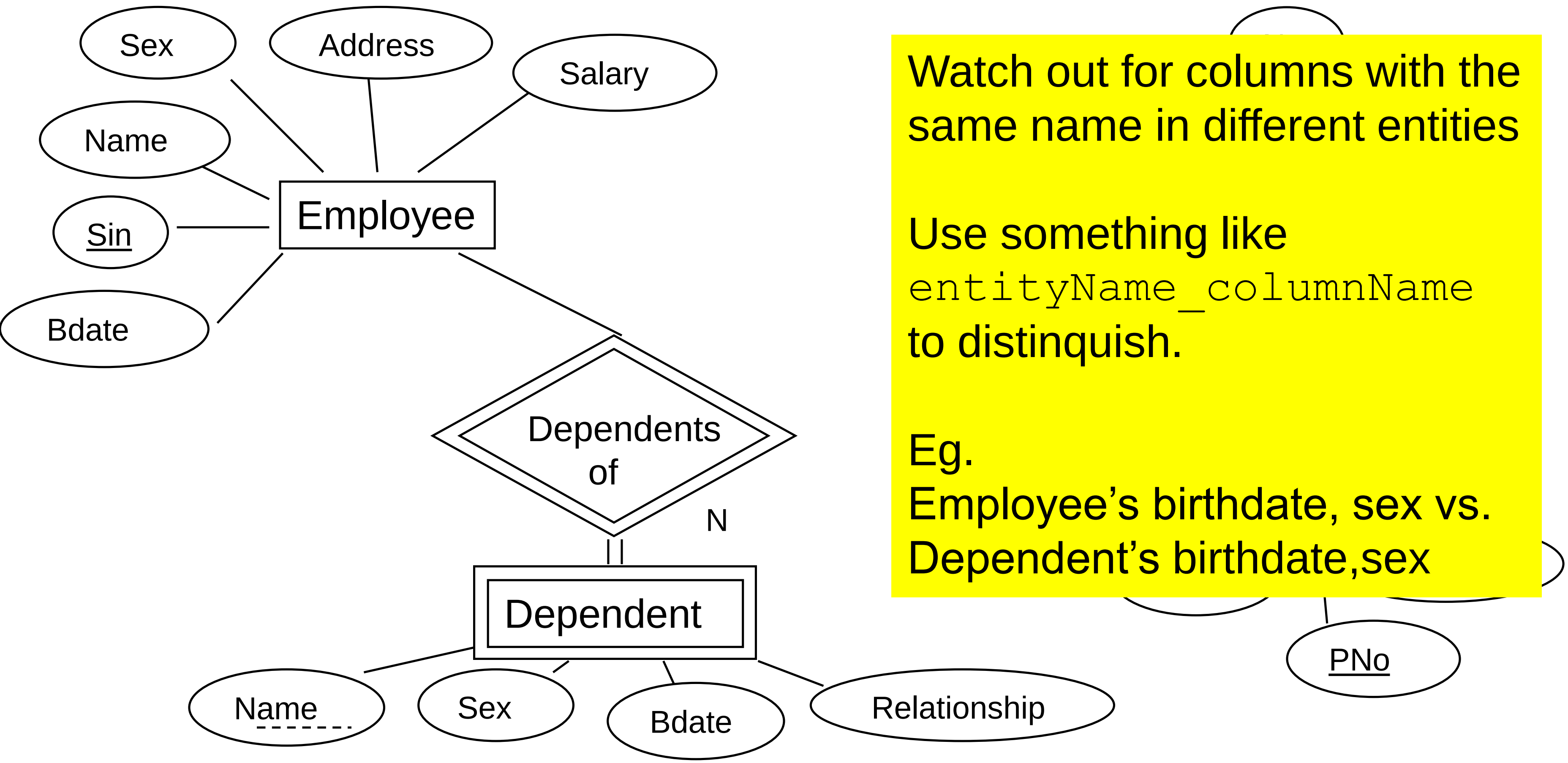


Entities and Weak Entities



SIN -> Bdate,Name,Sex,Address,Salary
DNo->DName
PNo->Pname,Plocation
SIN,DepName -> Dep_Sex,Dep_Bdate,Relationship

Entities and Weak Entities



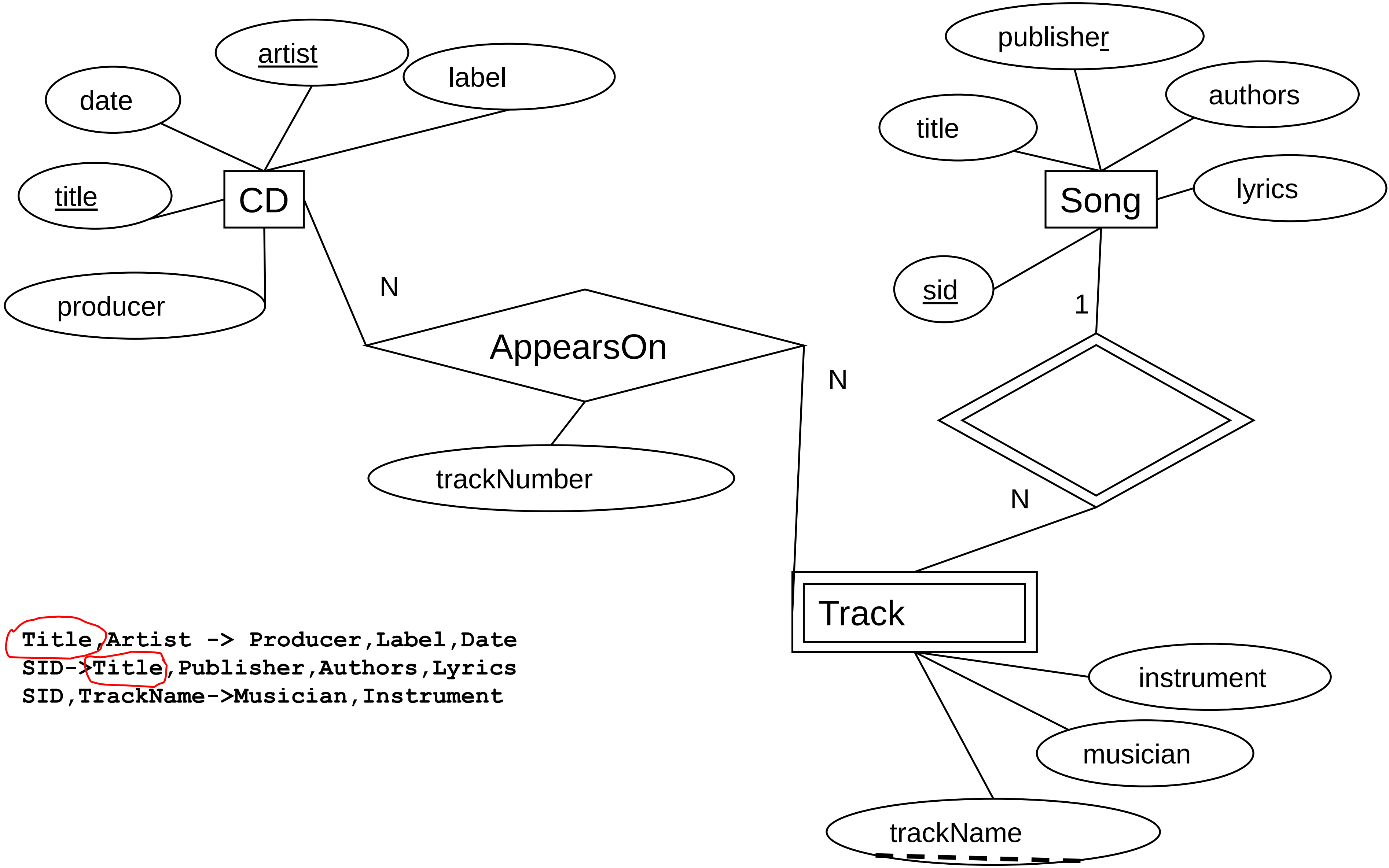
Watch out for columns with the same name in different entities

Use something like `entityName_columnName` to distinquish.

Eg.
Employee's birthdate, sex vs.
Dependent's birthdate,sex

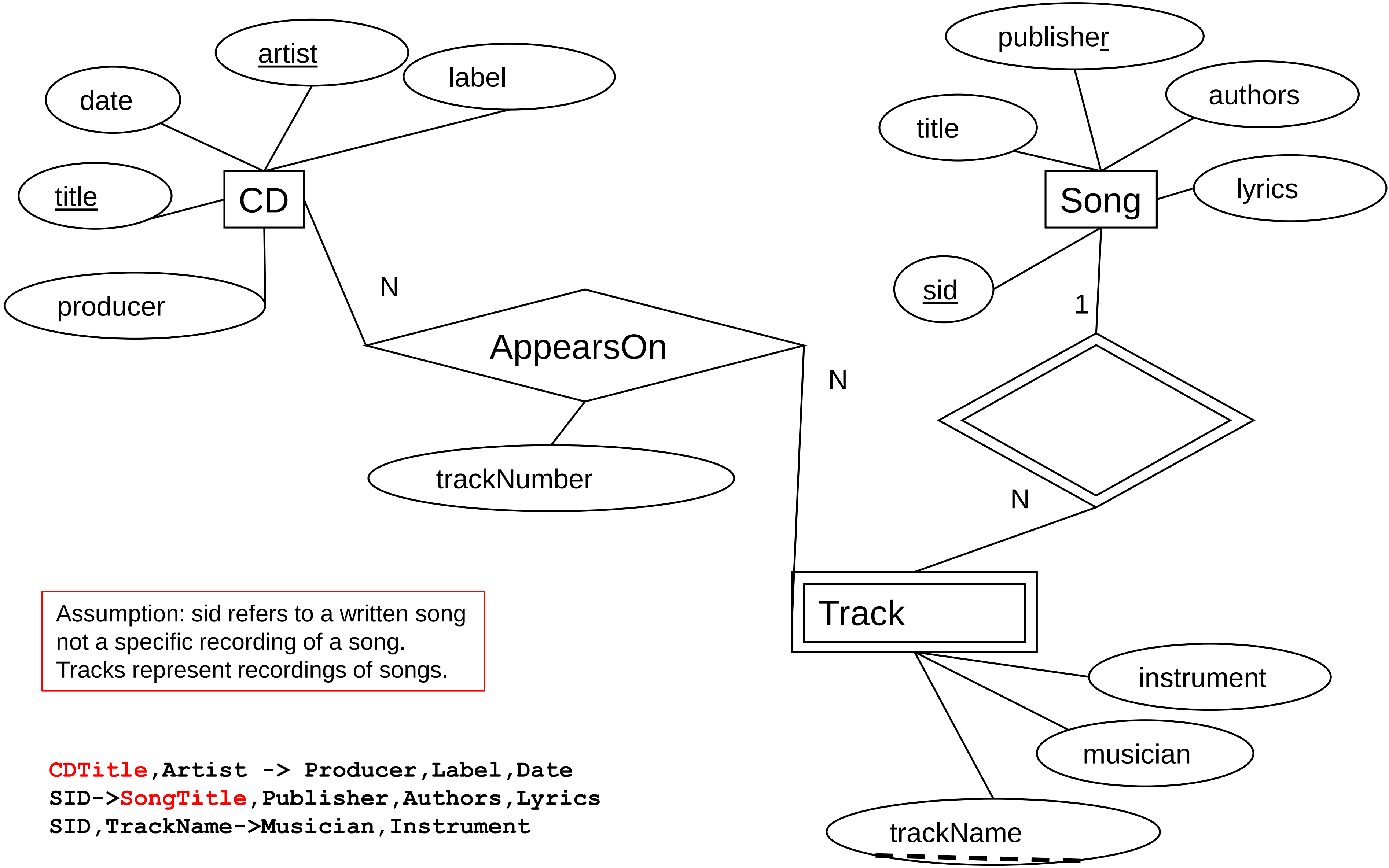
```
SIN -> Bdate,Name,Sex,Address,Salary
DNo->DName
PNo->Pname,Plocation
SIN,DepName -> Dep_Sex,Dep_Bdate,Relationship
```

A2,Q1: Entities and Weak Entities




```
Title,Artist -> Producer,Label,Date
SID->Title,Publisher,Authors,Lyrics
SID,TrackName->Musician,Instrument
```

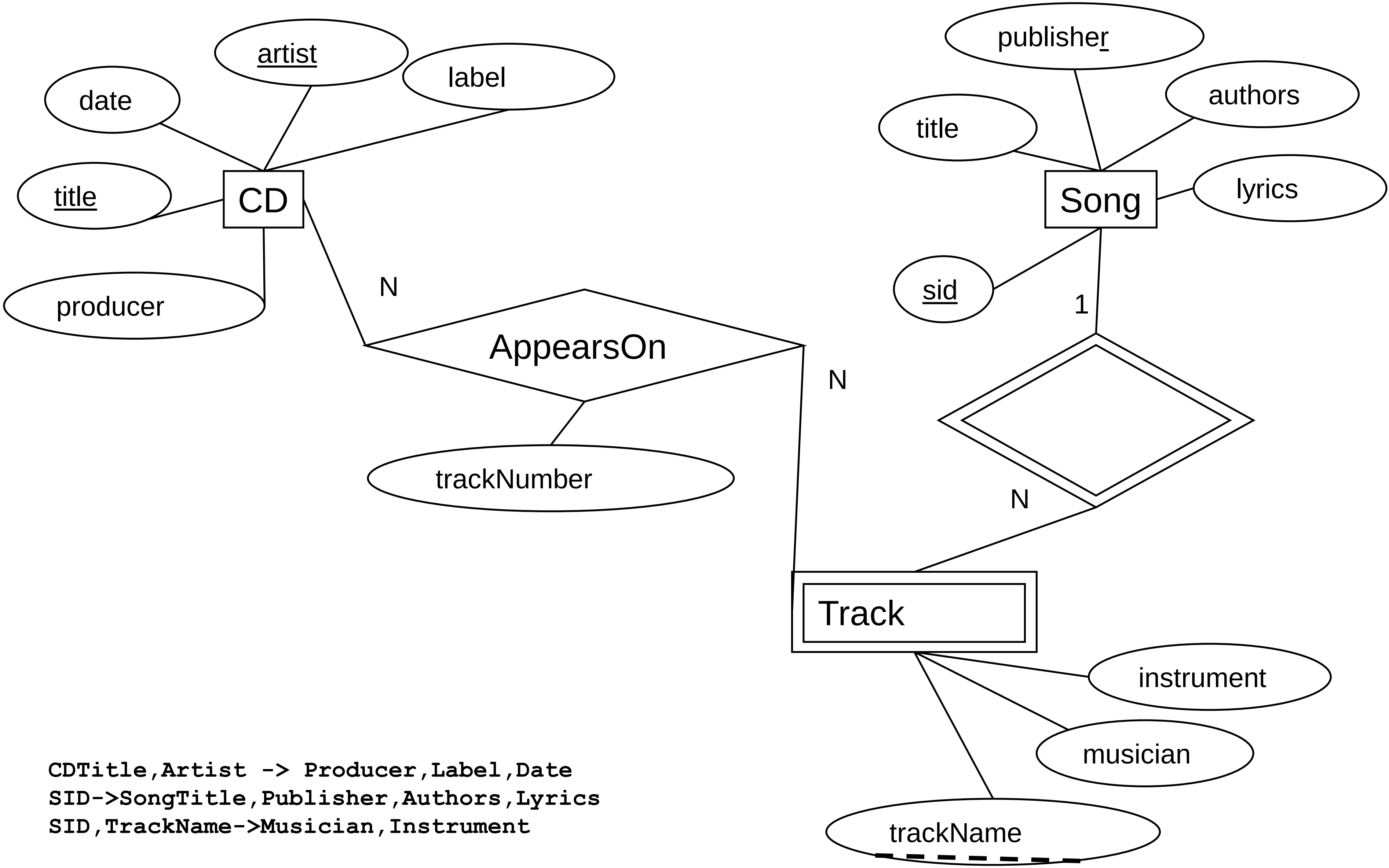
A2,Q1: Entities and Weak Entities



Assumption: sid refers to a written song not a specific recording of a song. Tracks represent recordings of songs.

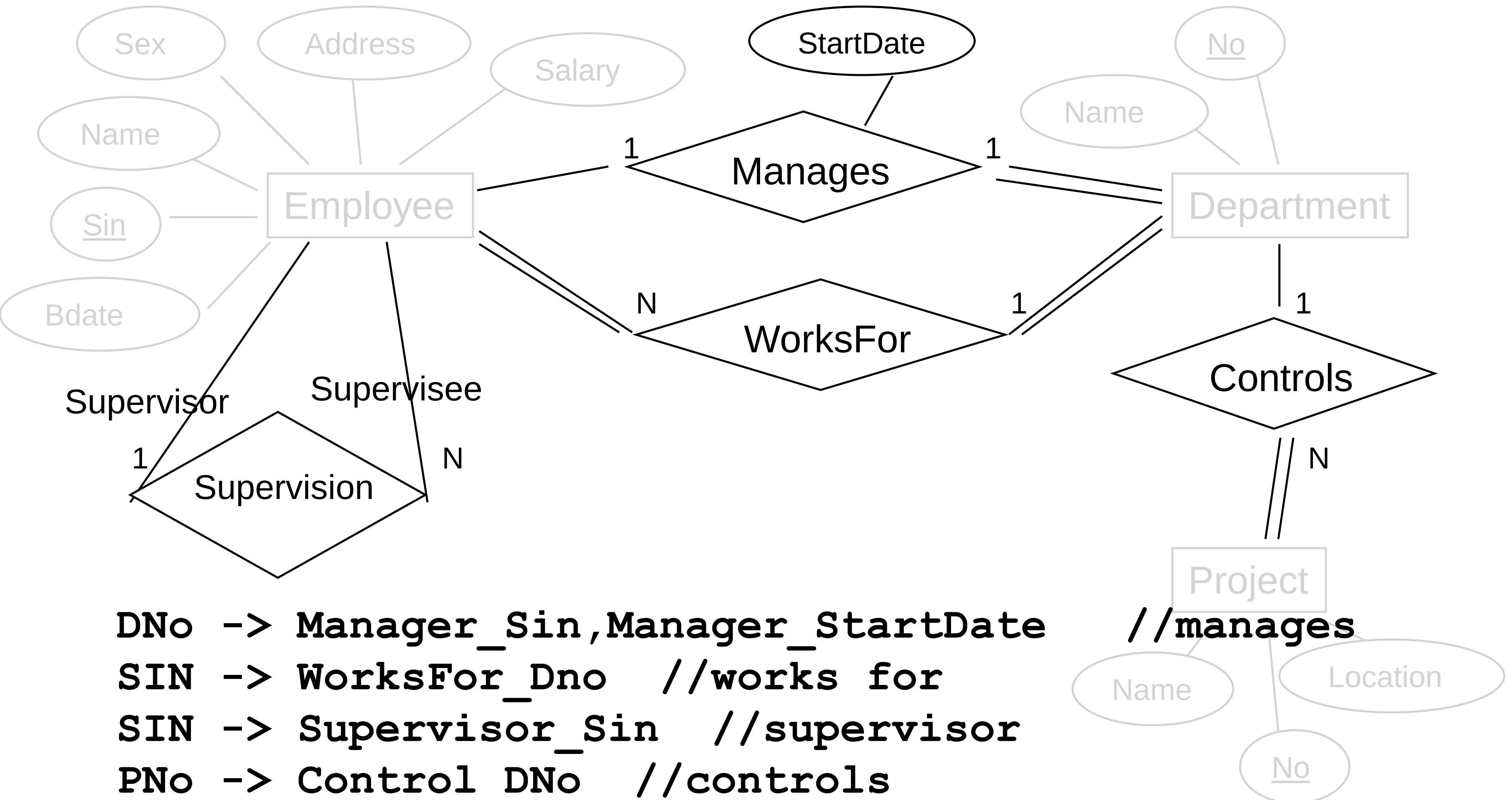
CDTitle,Artist -> Producer,Label,Date
SID->SongTitle,Publisher,Authors,Lyrics
SID,TrackName->Musician,Instrument

A2,Q1: Entities and Weak Entities




```
CDTitle,Artist -> Producer,Label,Date
SID->SongTitle,Publisher,Authors,Lyrics
SID,TrackName->Musician,Instrument
```

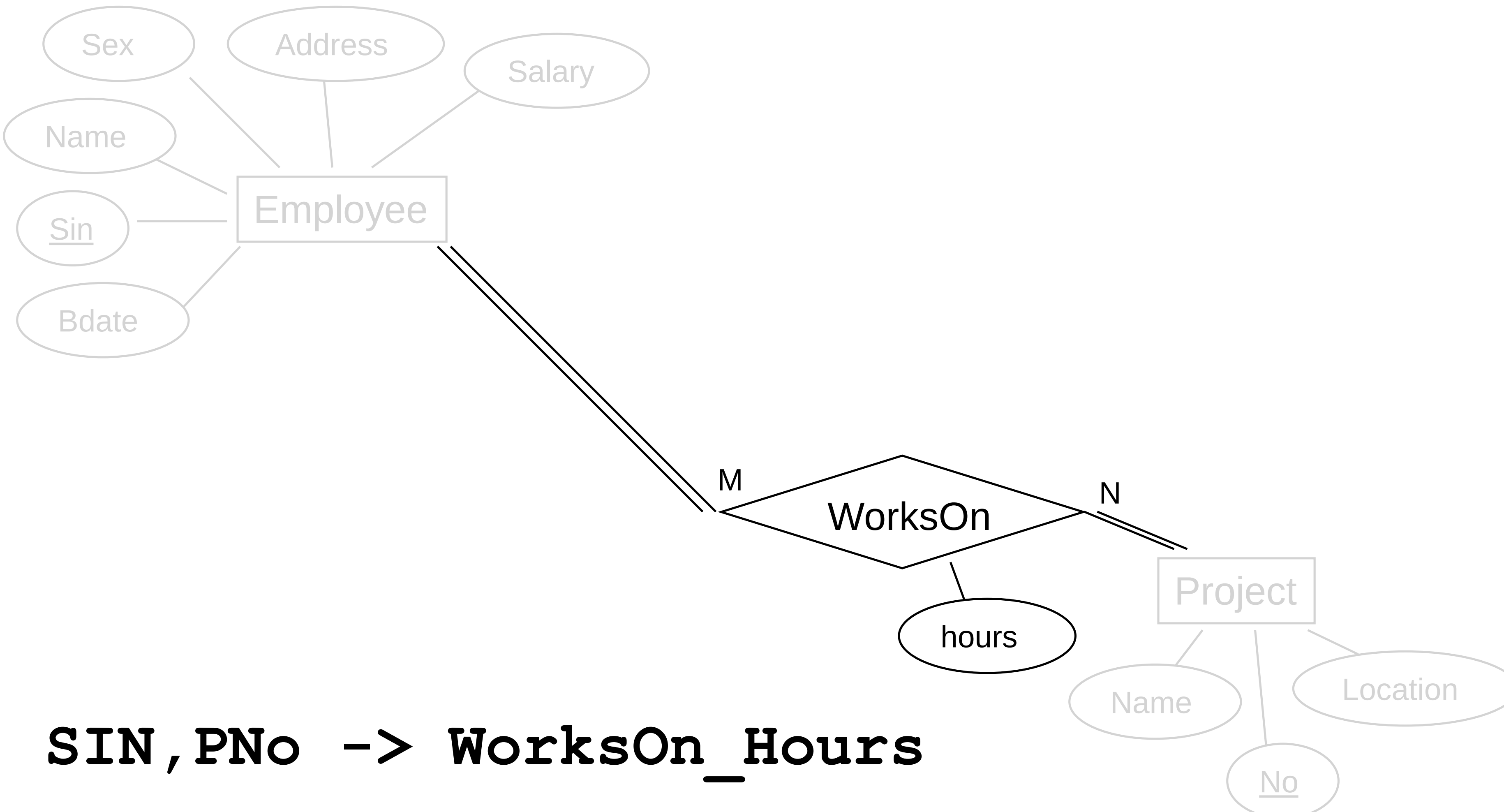
1:1 and 1:N Relationships



DNo -> Manager_Sin,Manager_StartDate //manages
SIN -> WorksFor_Dno //works for
SIN -> Supervisor_Sin //supervisor
PNo -> Control_DNo //controls

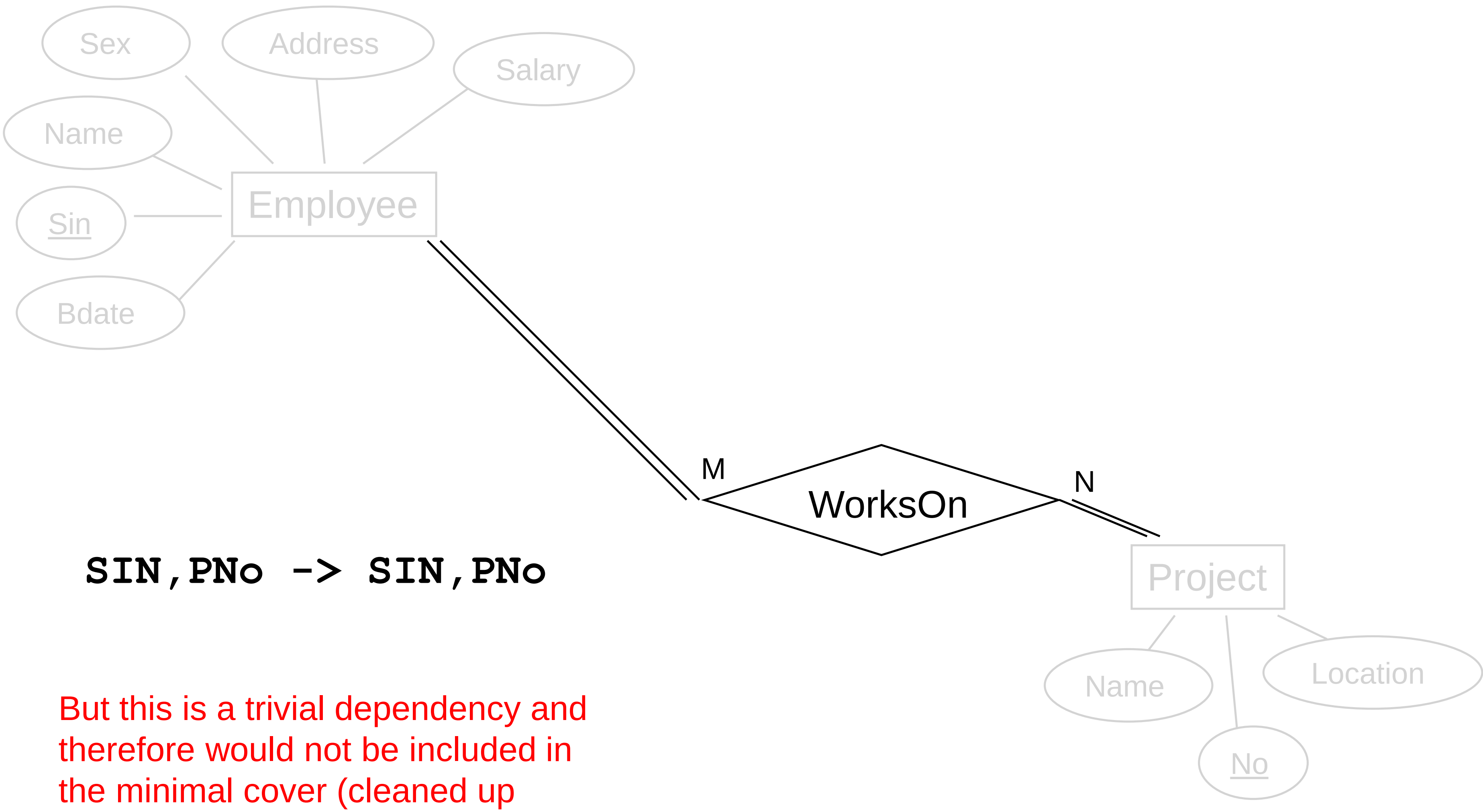
Here Underscore attributes to distinguish from previous Entity attributes

N:N Relationships –with relationship attributes



`SIN,PNo -> WorksOn_Hours`

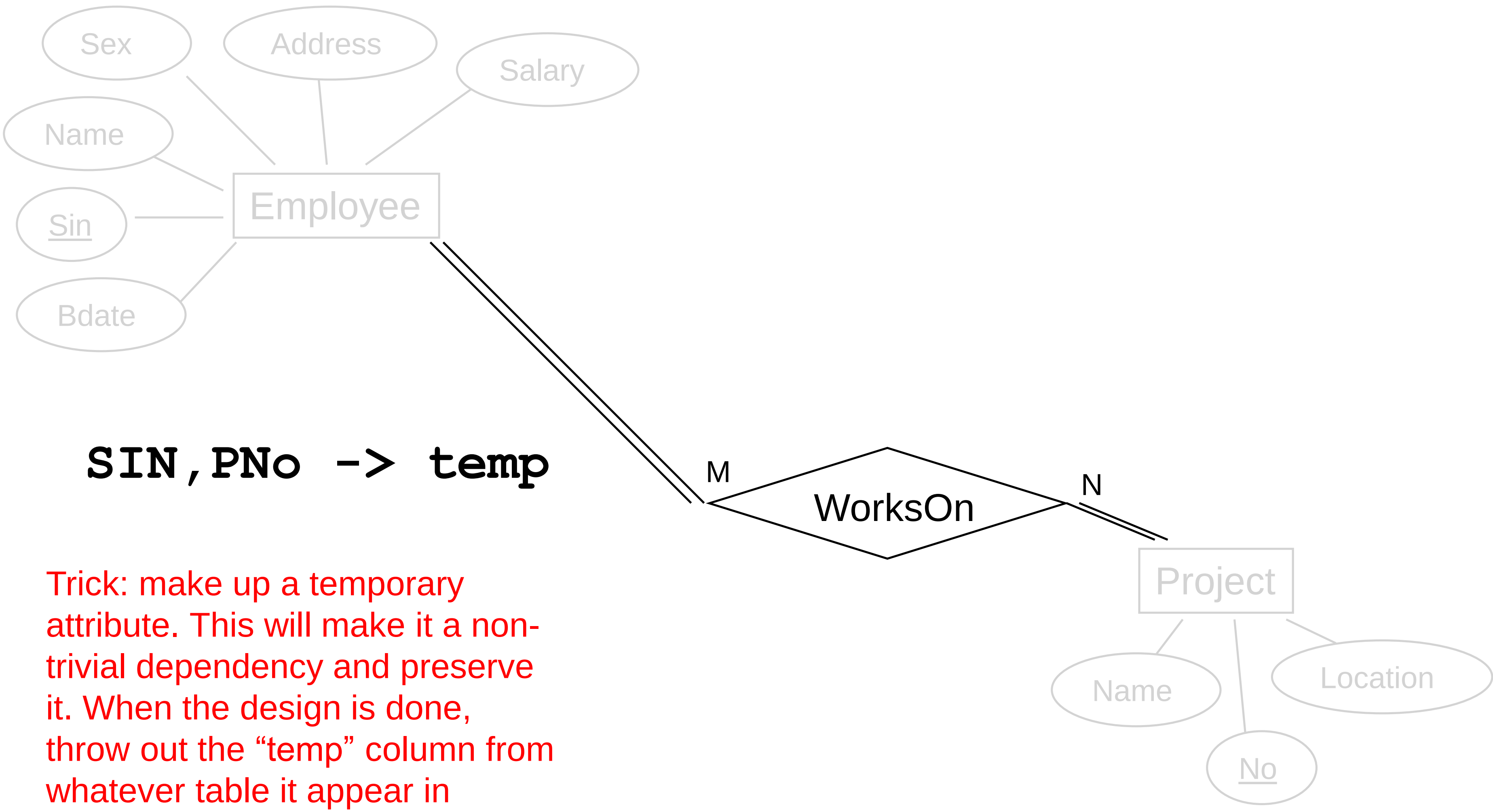
N:N Relationships –without relationship attributes



`SIN,PNo -> SIN,PNo`

But this is a trivial dependency and therefore would not be included in the minimal cover (cleaned up version) of F

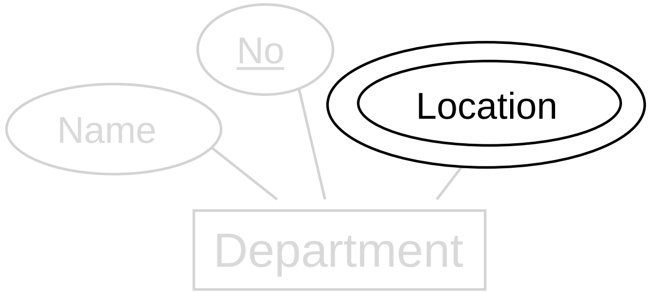
N:N Relationships –without relationship attributes



`SIN,PNo -> temp`

Trick: make up a temporary attribute. This will make it a non-trivial dependency and preserve it. When the design is done, throw out the “temp” column from whatever table it appear in

E-R diagram for Company Data (fig 3.2)

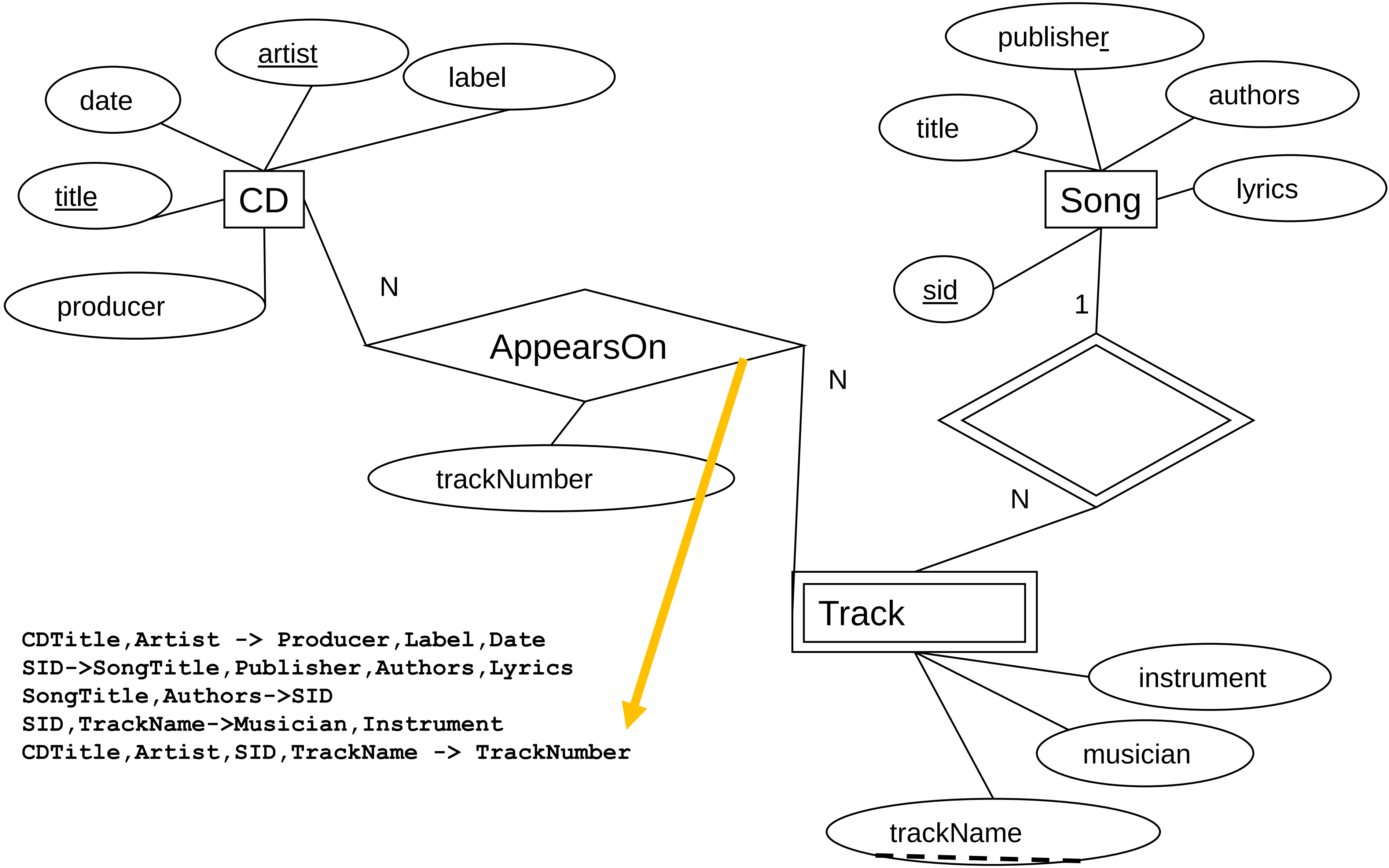


On the surface this is a 1:N relationship between department and locations (a department can have many locations)

But if more than one department can have the same location, then this is really a N:N relationship between departments and locations

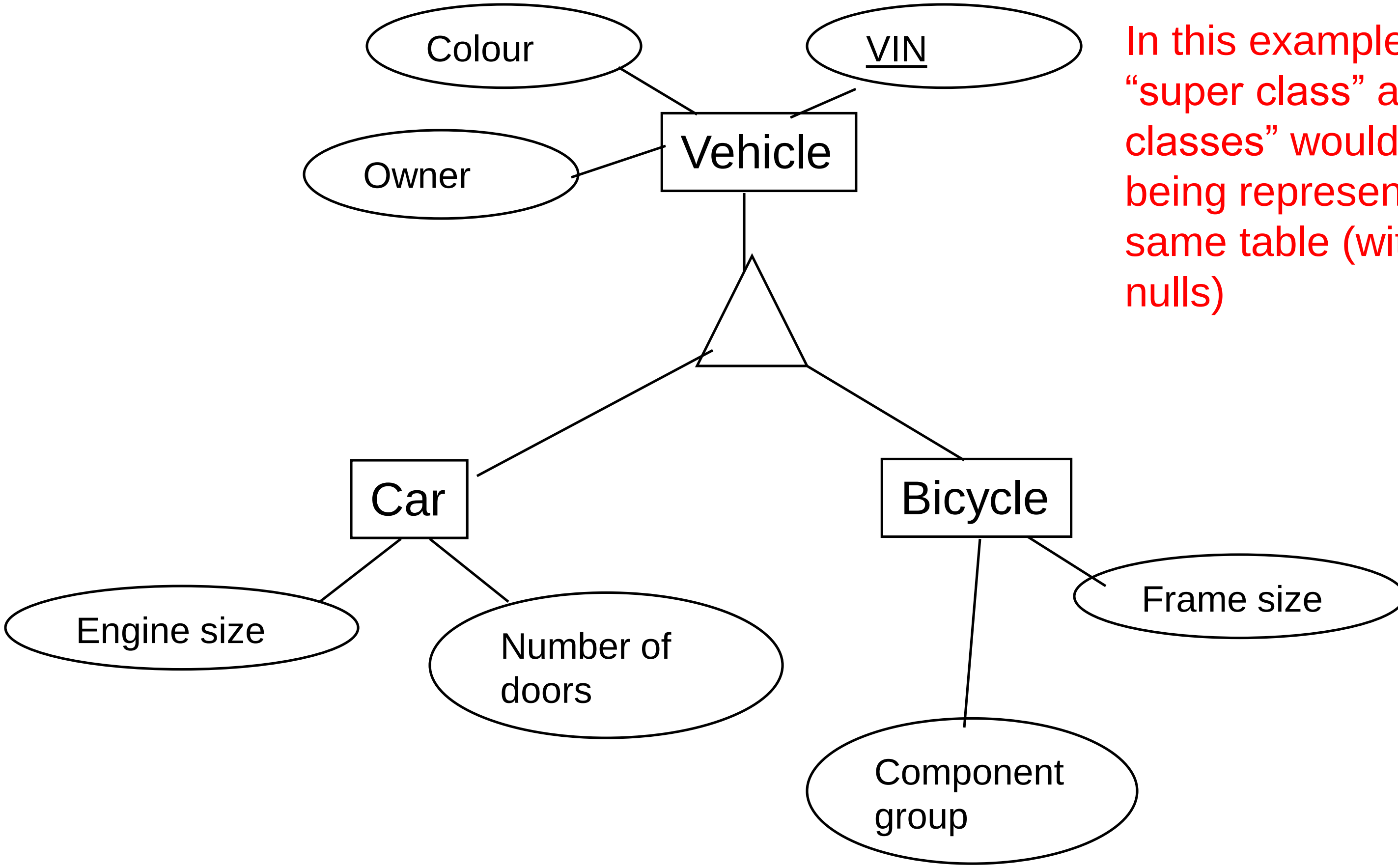
```
DNo,Location -> Dno,Location  //trivial dependency

Or

Dno,Location -> temp2  //non trivial
```

A2,Q1: Entities and Weak Entities




```
CDTitle,Artist -> Producer,Label,Date
SID->SongTitle,Publisher,Authors,Lyrics
SongTitle,Authors->SID
SID,TrackName->Musician,Instrument
CDTitle,Artist,SID,TrackName -> TrackNumber
```

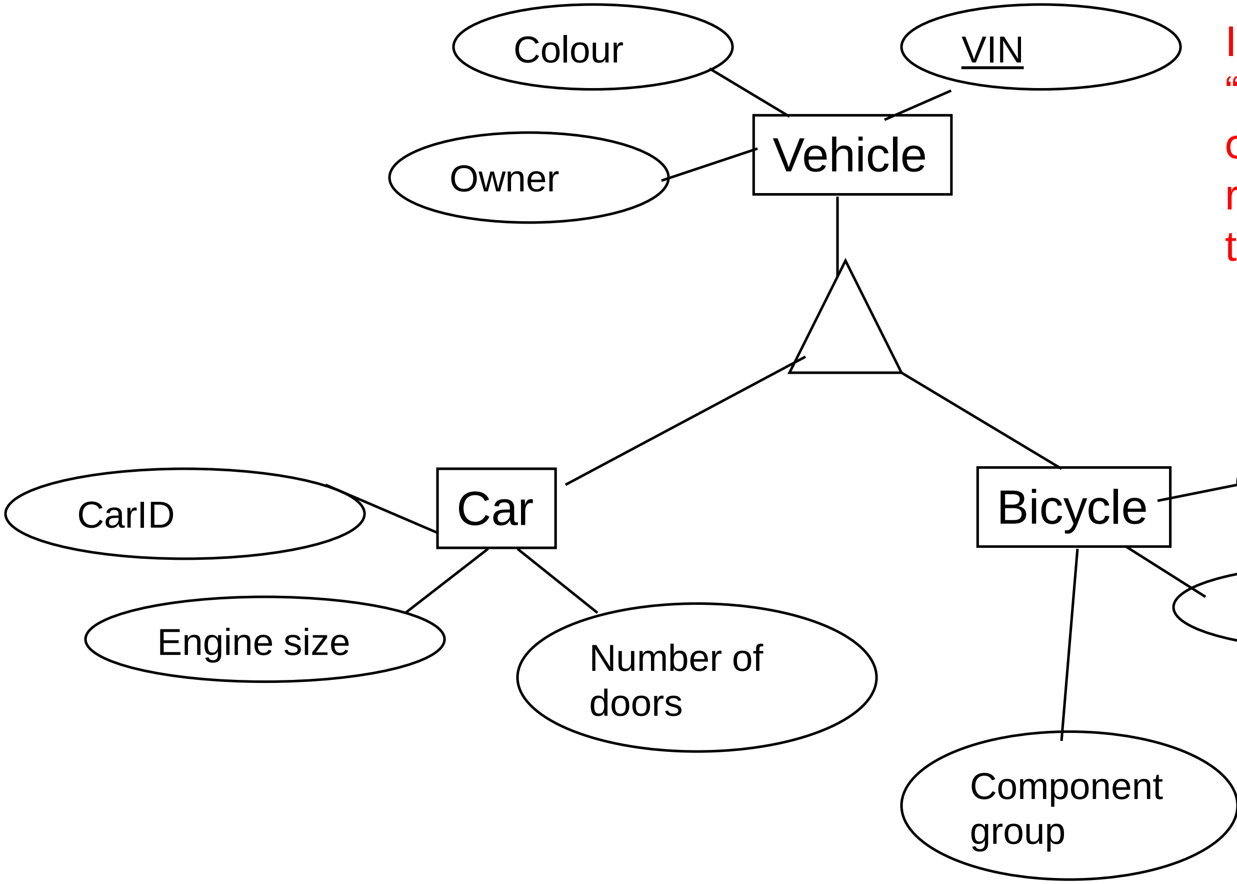
Inheritance



In this example the "super class" and "sub-classes" would end up being represented in the same table (with lots of nulls)

`VIN -> Colour,Owner,EngineSize,Ndoors,Group,FrameSize`

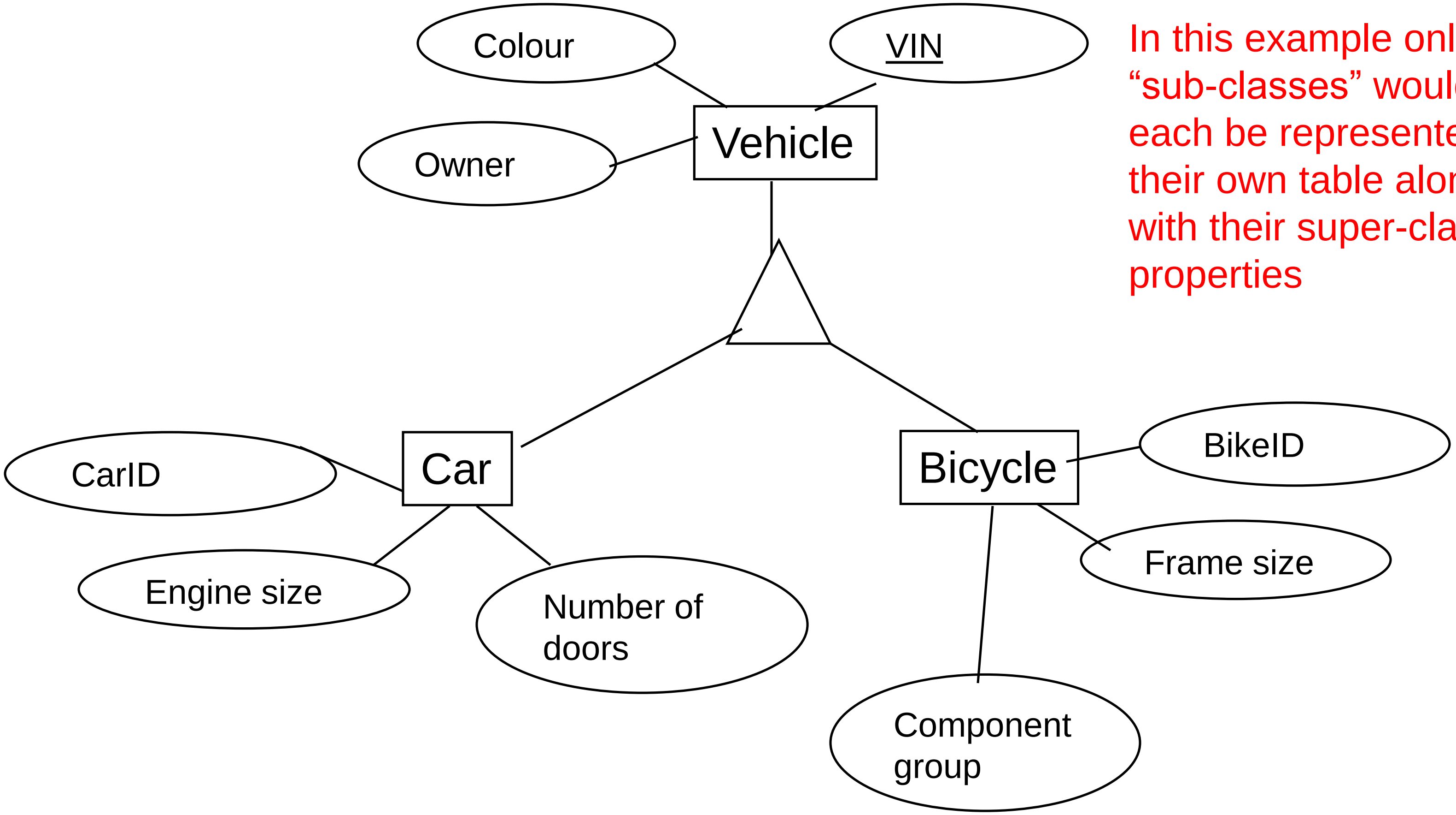
Inheritance



In this example the "super class" and "sub-classes" would each be represented in their own table

```
VIN -> Colour,Owner,CarID,BikeID
CarID -> EngineSize,Ndoors
BikeID -> Group,FrameSize
```

Inheritance



In this example only the “sub-classes” would each be represented in their own table along with their super-class properties

```
CarID -> VIN, CarColour,CarOwner,EngineSize,Ndoors
BikeID -> VIN,BikeColour,BikeOwner,Group,FrameSize
```

Exercise 1

Consider the set of attributes R={A,B,C,D,E,F} and the following set of functional dependencies proposed by the table designer.

F_1= {ABD->AC, B->E, BA->E, C->BE, AD->FB, C->E }

A colleague suggests that they use the following dependency set instead

F_2 = { AD->CF, C->B, B->E}

Determine if this is a reasonable suggestion

Approach 1

If we can show the sets are equivalent then F2 can be used instead of F1

F1 and F2 are equivalent if F1+ = F2+

To show this we must show that each functional dependency in F1 is implied by the set F2 and vice versa each functional dependency in F2 is implied by the set F1

Approach 1

As illustration we will show that:

ABD->AC from F1 is implied by set F2

and

AD->CF from F2 is implied by set F1

The same would have to be done for each functional dependency in each set, but for illustration here we only show the above two.

Approach 1

AD->CF from F2 is implied by set F1

Proof

AD>-CF implied because :

AD->BF, given in F1
AD->F decomposition rule

ABD->AC given in F1
ABD->C decompostion rule

AD->B given in in F1
AD->ABD augmentation rule
AD- >C transitive rule : AD- >ABD, ABD->C

AD->CF union rule, AD->C, AD->F

F_1= {
ABD->AC,
B->E,
BA->E,
C->BE,
AD->BF,
C->E }

F_2 = {
AD->CF,
C->B,
B->E}

Approach 1

ABD->AC from F1 is implied by set F2

Proof

ABD>-AC implied because :

AD->CF, given in F2

AD- >ACF augmentation rule (add A to both sides)

ABD->ABCF augmentation rule (add B to both sides)

ABD- >AC decomposition rules (ABD->AC, ABD->BF)

F_1= {
ABD->AC,
B->E,
BA->E,
C->BE,
AD->FB,
C->E }

F_2 = {
AD->CF,
C->B,
B->E}

Approach 2

We can find a minimal cover F_{m1} of F_1 and a minimal cover F_{m2} of F_2 and show that these minimal covers are equivalent

Again we would have to show that $F_{m1}^+ = F_{m2}^+$ but the hope is that this would be trivial by inspection, or an easier problem than working with the "raw" dependency sets.

Approach 2

Find a minimal cover of F1

F_1= {
ABD->AC,
B->E,
BA->E,
C->BE,
AD->FB,
C->E
}

=

F_1= {
ABD->A //decomposition
ABD->C //decomposition
B->E,
BA->E,
C->B, //decomposition
C->E, //decomposition
AD->F, //decomposition
AD->B, //decomposition
C->E
}

Approach 2

F_1= {

~~ABD->A~~ //reflexive

ABD->C

B->E,

~~BA->E~~, //implied by

//previous rule

C->B,

C->E,

AD->F,

AD->B,

~~C->E~~ //duplicate

}

=

F_1= {

ABD->C

B->E,

C->B,

C->E,

AD->F,

AD->B,

}

Approach 2

F_1= {

A~~B~~D->C //since AD->B

B->E,

C->B,

~~C->E~~, //transitive

AD->F,

AD->B,

}

=

F_1= {

AD->C

B->E,

C->B,

AD->F,

AD->B,

}

Approach 2

F_1= {

AD->C

B->E,

C->B,

AD->F,

~~AD->B~~, //transitive

}

=

F_1= {

AD->C

B->E,

C->B,

AD->F,

}

F_{m1} = {AD->C, AD->F, B->E, C->B}

Approach 2

Find a minimal cover of F2

F_2= {

AD->CF,

C->B,

B->E

}

=

F_2= {

AD->C, //decomposition

AD->F, //decomposition

C->B,

B->E

}

F_{m2} = { AD->C, AD->F, C->B, B->E}

Approach 2

F_{m1} = {AD->C, AD->F, B->E, C->B}

F_{m2} = { AD->C, AD->F, C->B, B->E}

By inspection $F_{m1} = F_{m2}$ so the sets F_1 and F_2 are equivalent

Exercise 2

- **For each of the following cases a relation R has been defined over attributes A,B,C,D,E,F along with a set of functional dependencies that apply to them.**

- **Find all the candidate keys for the relation**
- **State the highest normal form the table R=ABCDEF would currently satisfy**
- **Decompose the table until all resulting tables are in BCNF form**

Exercise 2

F={ AB-> CDEF, EF ->C }

Exercise 2

F={ AB-> CDEF, EF ->C }

Candidate keys: AB

Current Normal Form: = 2nd NF since EF->C violates 3rd NF.

Decomposition:

<u>AB</u>DEF

<u>EF</u>C

Exercise 2

F={ AB-> CDEF, EF->B , D->B }

Exercise 2

F={ AB-> CDEF, EF->B , D->B }

Candidate keys: AB, AD, AEF

Current Normal Form: = 3rd NF since D->B violates BCNF.

Decomposition:

<u>AD</u>EF

<u>D</u>B

Exercise 2

F={ AB-> CDEF, BC->D, }

Exercise 2

F={ AB-> CDEF, BC->D, }

Candidate keys: AB

Current Normal Form: = 2^{nd} NF since BC->D violates 3^{rd} NF.

(Recall: Y->A is a transitive dependency if Y is neither a superkey of R nor a proper subset of a key of R)

Decomposition:

<u>AB</u>CEF

<u>BC</u>D

Exercise 2

F={ AB-> CDEF, B->C , D->C}

Exercise 2

F={ AB-> CDEF, B->C , D->C}

Candidate keys: AB

Current Normal Form: = 1st NF since B->C violates 2nd NF.

Decomposition:

<u>AB</u>DEF

<u>B</u>C

<u>D</u>C