DATA INTEGRATION EXERCISE – Tv Shows

An important TV director wants to integrate the database of his company, containing only data about TV shows, with the data of another database concerning a wider variety of shows. The final integrated database must contain information related exclusively to TV shows coming from both the sources.

The first source is a standard relational database:

SHOW (Title, Date, Type)
CASTMEMBER (Code, Firstname, Lastname, Address, FeePerHour, AgentCode)
CAST (ShowTitle, ShowDate, CastMemberCode, AppearanceDuration)
AGENT (Code, Firstname, Lastname, Company, PhoneNumber)
ROLE (Name, Description)
CASTROLE (CastMemberCode, RoleName)

Remarks:

- The duration of the show is expressed in hours.
- The fees are expressed in dollars.
- Dates are expressed as yyyy-mm-dd.
- The possible roles are guest, stable-member and technician.
- Type of the show can be TvShow, Theater, …

The second source is that used in the web information system of the TV company, which relies on an XML database and contains only information about TV shows:

```
<!ELEMENT TvShowsDB (TvShow*)>
<!ELEMENT TvShow (Anchor+, Schedule)>
<!ELEMENT Anchor EMPTY>
<!ELEMENT Schedule (Day+)>
<!ELEMENT Day (Guest+)>
<!ELEMENT Guest EMPTY>
<!ATTLIST TvShow Title CDATA #REQUIRED
                 Edition CDATA #REQUIRED
                 Duration CDATA #REQUIRED>
<!ATTLIST Anchor Name CDATA #REQUIRED
                 EngagementFee CDATA #REQUIRED
                 AgentPhoneContact CDATA #IMPLIED>
<!ATTLIST Day Date CDATA #REQUIRED
              Special (0|1) "0">
<!ATTLIST Guest Name CDATA #REQUIRED
                EngagementFee CDATA #REQUIRED
                Address CDATA #IMPLIED>
```

An example of a valid XML file compliant with the schema is:

```
<TvShowsDB>
  <TvShow Title="The David Letterman Show" Edition="2016/2017" Duration="1">
    <Anchor Name="David$Letterman" EngagementFee="60,000"/>
    <Schedule>
      <Day Date="2016-09-14" Special="0">
        <Guest Name="Mike$Tyson" EngagementFee="200,000"/>
      </Day>
      <Day Date="2016-09-21">
        <Guest Name="Hillary$Clinton" EngagementFee="30,000"/>
        <Guest Name="Donald$Trump" EngagementFee="30,000"/>
      </Day>
    </Schedule>
  </TvShow>
</TvShowsDB>
```

Remarks:

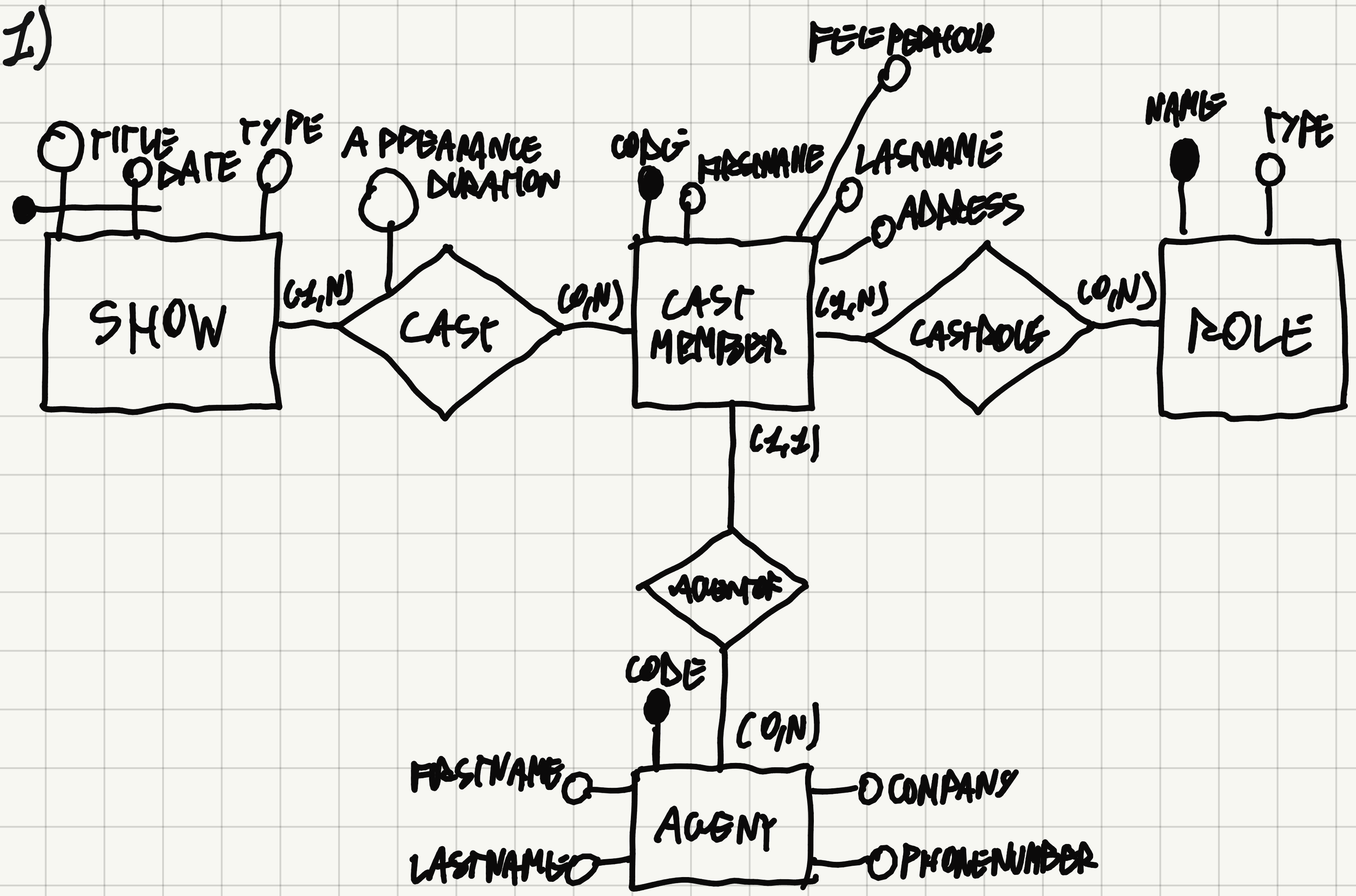
- The duration of the show is expressed in hours.
- The fees are expressed in dollars.
- The fee of an anchor or a guest may change for each edition of the show or date.
- Dates are expressed as yyyy-mm-dd.
- The anchor and guest names are in the form "*Firstname$Lastname"*.
- Suppose that there are no duplicates in the names of anchors and guests.

NOTE: you can suppose that the people in the two data sources are disjoint, and that the shows in the two data sources are disjoint.

Integrate the two data sources, following these steps:

1) Source schema reverse engineering
2) Schema integration
    a. Related concept identification + Conflict analysis and resolution
    b. Production of the global conceptual schema
    c. Conceptual to logical translation of the global schema
3) Query formulation and mapping definition
    a. Write in SQL the following query Q on the global schema: *"Select first name, last name and address of the cast members of TV shows that have been on air in special occasions"*
    b. GAV mapping definition (only for the tables required to answer the query Q at point 3a)
    c. Show the rewriting of Q on the data sources in SQL

1)



2) WE LOOK AT THE DDT DOCUMENT IN ORDER TO FIND THE INFORMATIO TO WRITE THE RELATIONAL SCHEMA

! XML DOCUMENT HAS NO PRIMARY KEYS. WE HAVE TO CHOOSE THEM OURSELVES

TVSHOWSDB

|*(≥0)

TVSHOW ENTITY

+(≥1) / \ WITHOUT ATTRIBUTES

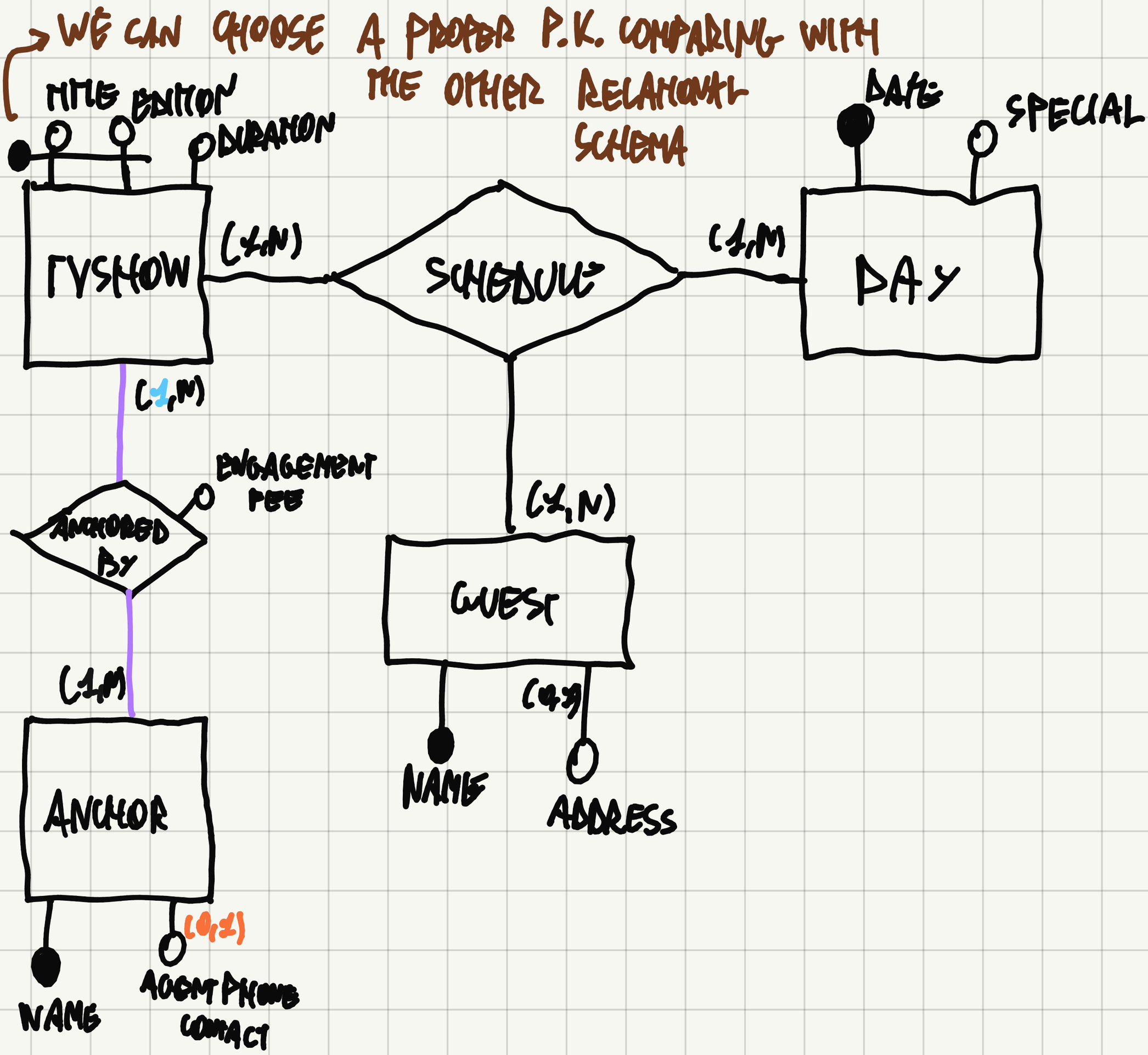
ANCHOR ENTITY — SCHEDULE RELATIONSHIP — ONLY ELEMENT

| +

DAY ENTITY

| +

GUEST ENTITY



LOGICAL SCHEMA

TVSHOW(TITLE, EDITION, DURATION)

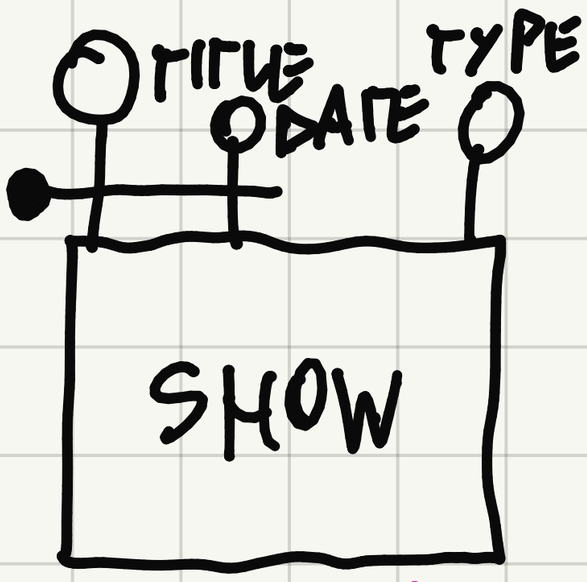
ANCHOR(NAME, AGENTPHONECONTACT*)

GUEST(NAME, ADDRESS*)

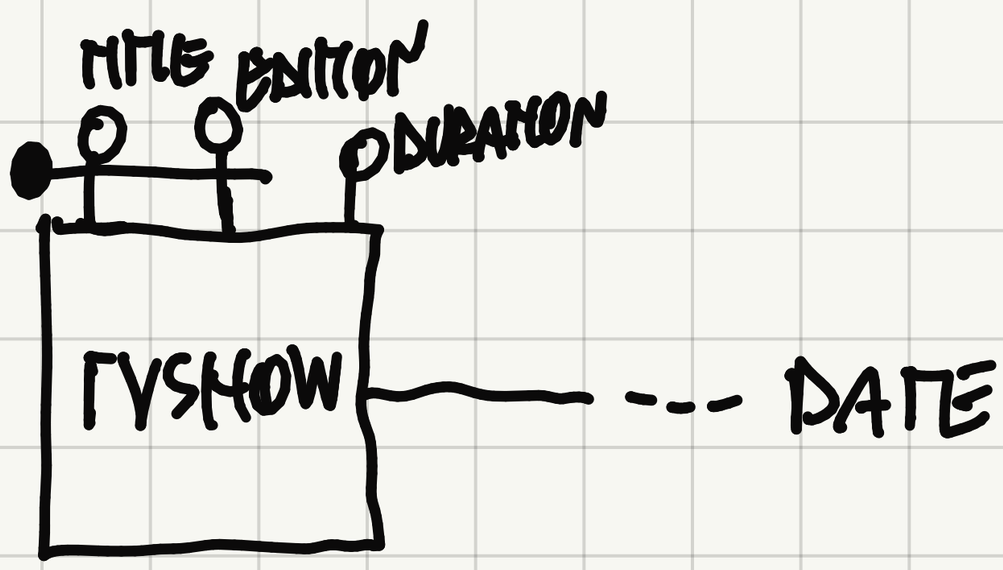
DAY(DATE, SPECIAL)

ANCHOREDBY(TVSHOWTITLE, TVSHOWEDITION, ANCHORNAME, ENGAGEMENTFEE)

SCHEDULE(TVSHOWTITLE, TVSHOWEDITION, DATE, GUESTNAME, GUESTENGAGEMENT FEE)



CONFLICT



RESOLUTION

ENTITY NAME

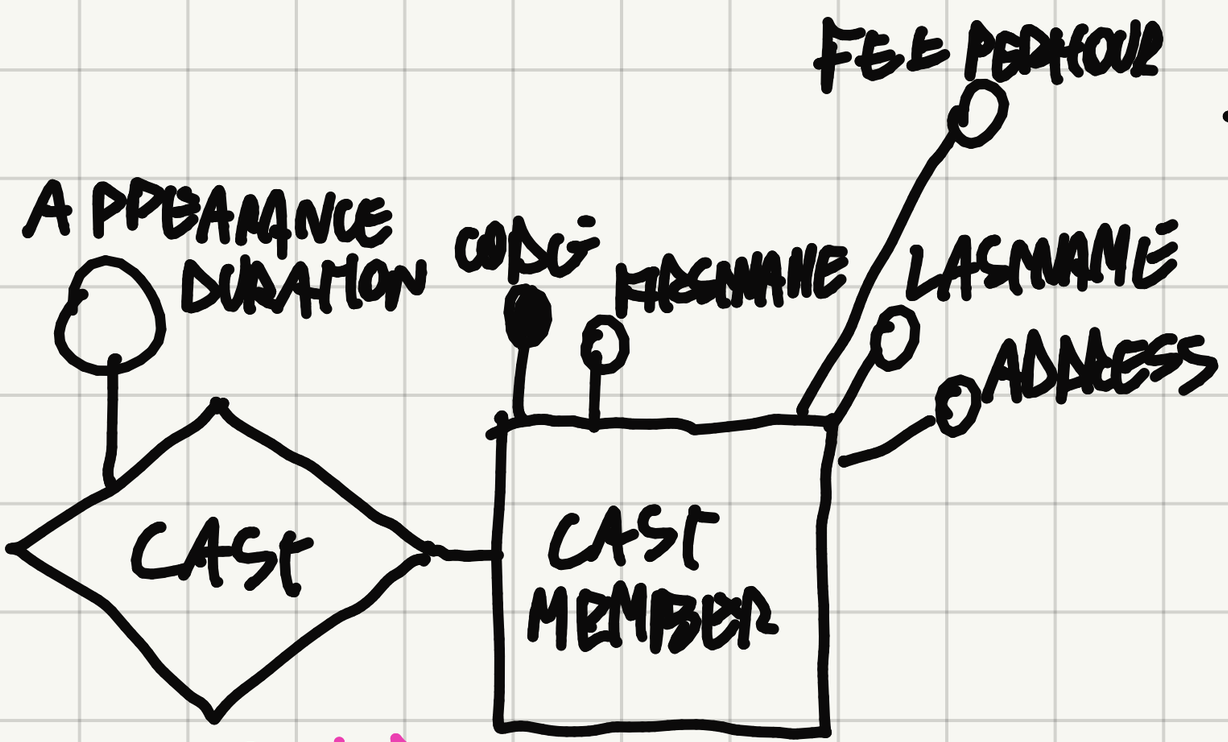
TVSHOW

STRUCTURE CONFLICTS:

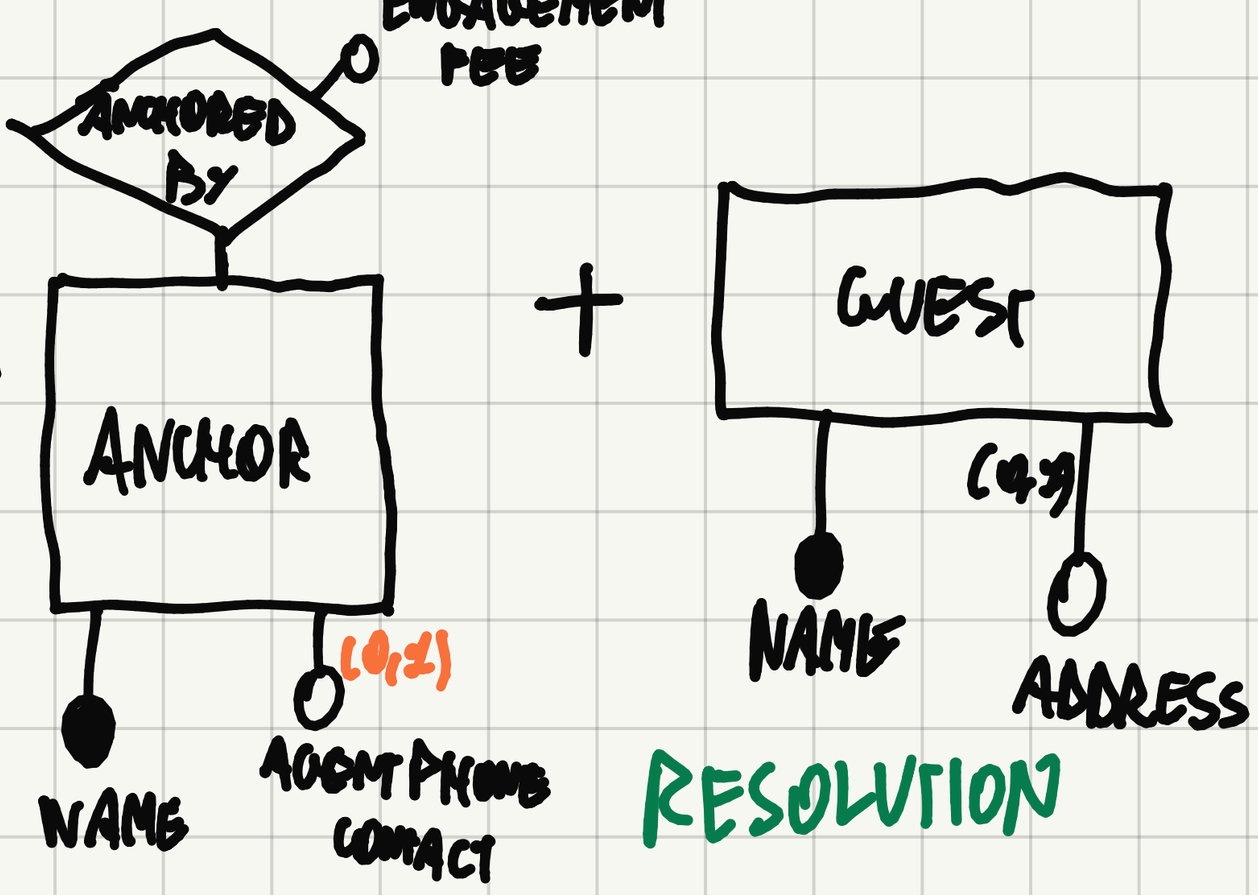
- DATE AS ATTRIBUTE, DATE AS ENTITY — DATE AS ATTRIBUTE

KEY CONFLICT:

TITLE+DATE; TITLE+EDITION — TITLE+DATE



CONFLICT



RESOLUTION

ENTITY NAME:

CAST MEMBER, ANCHOR+GUEST — CASTMEMBER (C1 TO REMAIN AS GENERAL AS POSSIBLE)

DATA SEMANTIC CONFLICTS:

APPEARANCE = FEEPERHOUR * DURATION

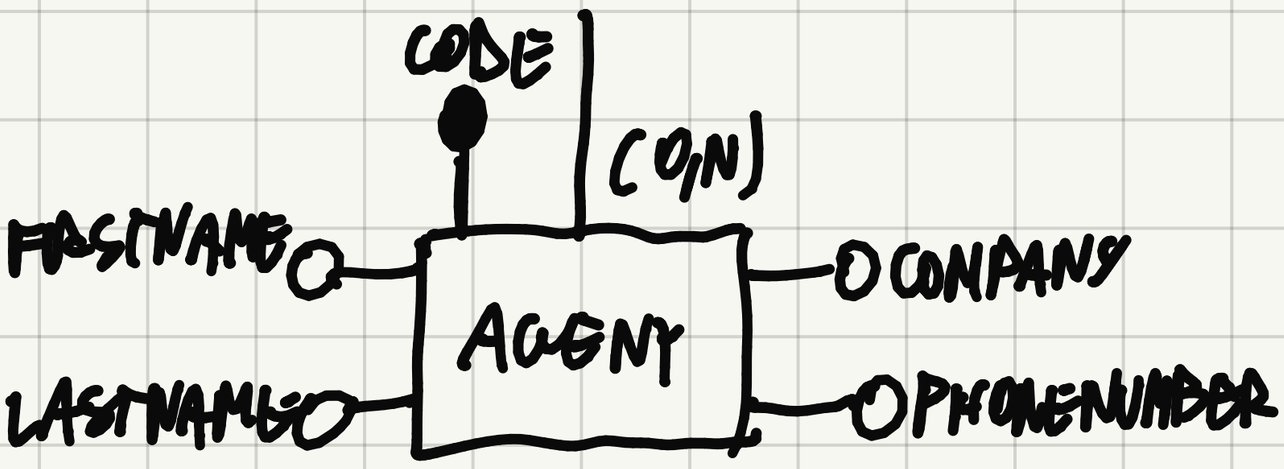
FEEPERHOUR($/h), ENGAGEMENTFEE($) — ENGAGEMENTFEE($)

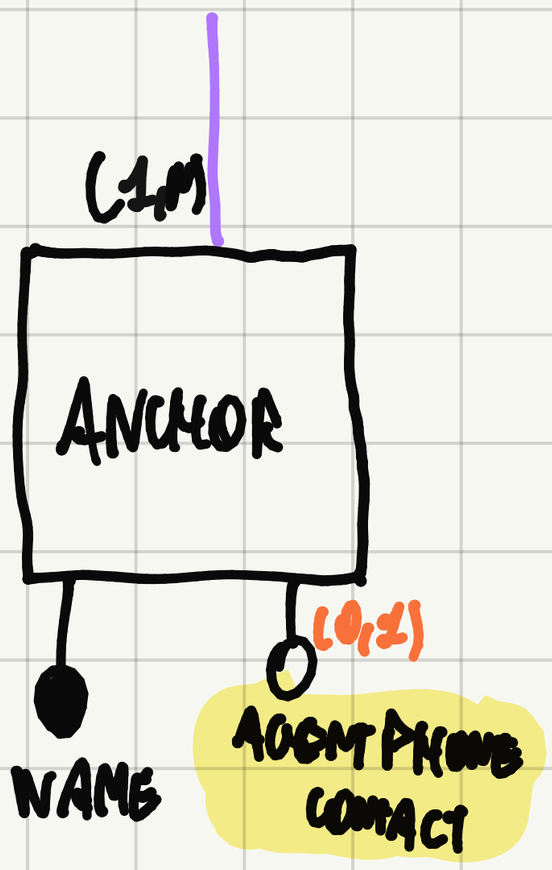
STRUCTURE CONFLICT:

- LASTNAME+FIRSTNAME, NAME(... $...) — LASTNAME+FIRSTNAME

- FEEPERHOUR(UNIQUE V PERSON), ENGAMBITFEE(DEPENDING ON TV SHOW)

DEPENDING ON TV SHOW



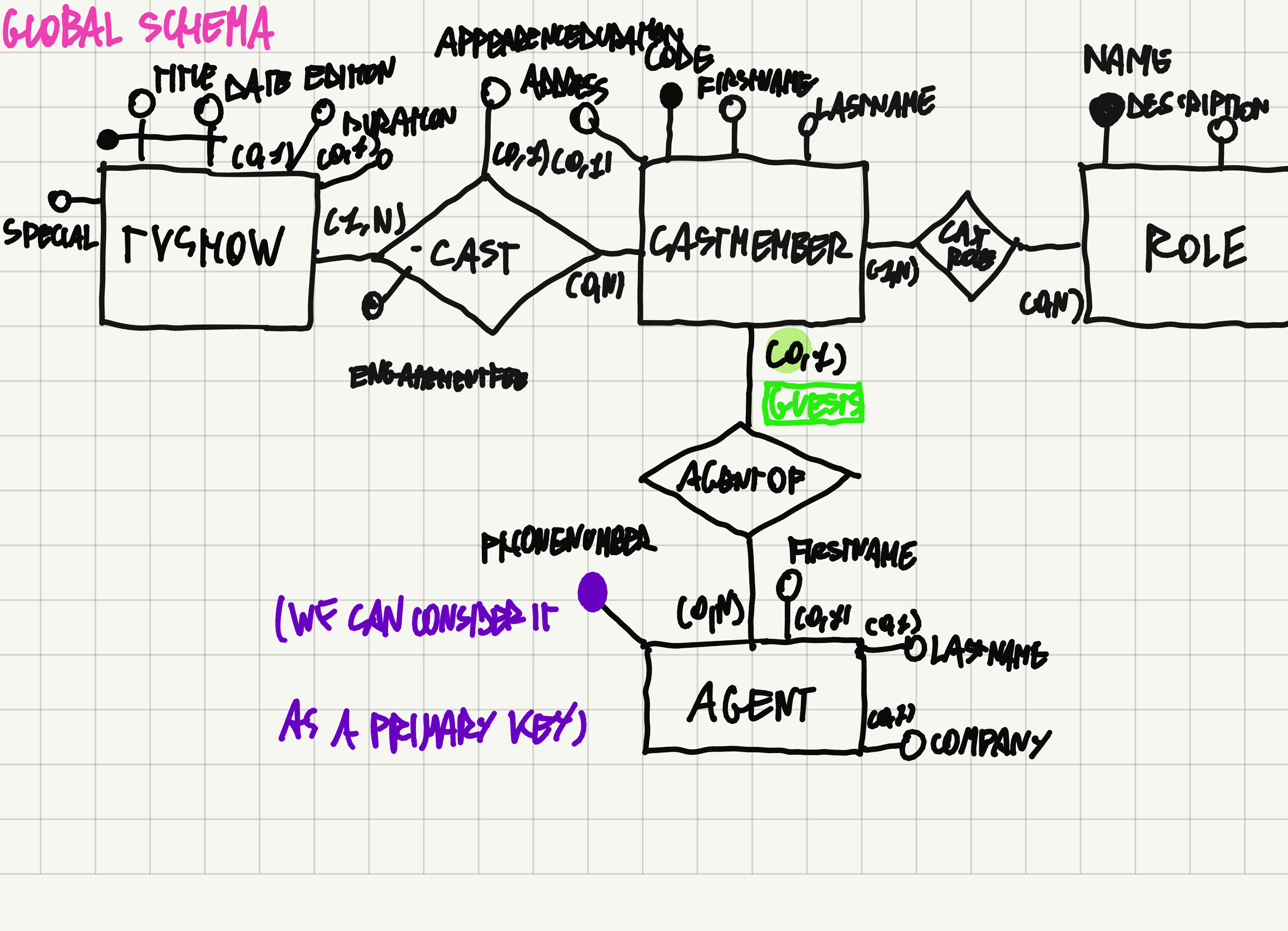


WHOLE ENTITY

- KEYCONFLICT(CODE, NAME)

CODE

GLOBAL SCHEMA



LOGICAL SCHEMA

TVSHOW(TITLE, DATE, EDITION*, DURATION*, SPECIAL)

CASTMEMBER(CODE, FIRSTNAME, LASTNAME, ADDRESS*, AGENTPHONENUMBER*)

CAST(TVSHOWTITLE, TVSHOWDATE, CASTMEMBERCODE, APPEARENCEDURATION*, ENGAGEMENTFEE)

AGENT(PHONENUMBER, FIRSTNAME*, LASTNAME*, COMPANY*)

ROLE(NAME, DESCRIPTION)

CASTROLE(CASTMEMBERCODE, ROLENAME)

QUERY

```
SELECT DISTINCT  CM.FIRSTNAME, CM.LASTNAME, CM.ADDRESS
FROM    CASTMEMBER AS CM, CAST AS C, TVSHOW AS T
WHERE   CM.CODE = C.CASTMEMBERCODE AND C.TVSHOWTITLE = T.TITLE
        AND C.TVSHOWDATE = T.DATE
        AND T.SPECIAL = TRUE
```

VIEWS

```
CREATE VIEW GS.TVSHOW (TITLE, DATE, EDITION, DURATION, SPECIAL) AS
(
    SELECT TITLE, DATE, NULL, NULL, ISSPECIAL(DATE)
    FROM DS1.SHOW
    WHERE TYPE="TV SHOW"
UNION
    SELECT T.TITLE, D.DATE, T.EDITION, T.DURATION, D.SPECIAL
    FROM DS2.TVSHOW AS T, DS2.SCHEDULE AS S,
         DS2.DAY AS D
    WHERE T.TITLE=S.TVSHOW AND T.EDITION=S.TVSHOWEDITION
        AND S.DATE=D.DATE
)
```

```
CREATE VIEW GS.CASTMEMBER(CODE,FIRSTNAME,LASTNAME,ADDRESS,
                                        AGENTPHONENUMBER) AS (
        SELECT KEYGENCASTMEMBER(CM.CODE,'DS1'),CM.FIRSTNAME,
                CM.LASTNAME,CM.ADDRESS,CM.AGENTPHONENUMBER
        FROM   DS1.CASTMEMBER AS CM,DS1.AGENT AS A,
               DS1.CAST AS A,DS1.SHOW AS S
        WHERE  CM.AGENTCODE=A.CODE AND CM.CODE=C.CASTMEMBERCODE
               AND C.SHOWTITLE=S.TITLE AND C.SHOWDATE=S.DATE
               AND S.TYPE="TVSHOW"
UNION
                                        AGENTPHONE
                                EXTRACTSURNAME(NAME),NULL,NUMBER
        SELECT KEYGENCASTMEMBER(NAME,'DB2'),EXTRACTNAME(NAME),
        FROM   DS2.ANCHOR
UNION
                                EXTRACTSURNAME(NAME),ADDRESS,NULL
        SELECT KEYGENCASTMEMBER(NAME,'DB2'),EXTRACTNAME(NAME),
FROM   DS2.GUEST
```

```
CREATE VIEW GS.CAST(TVSHOWTITLE, TVSHOWDATE, CASTMEMBERCODE,
    APPEARENCEDURATION, ENGAGEMENTFEE) AS (
    SELECT (.SHOWTITLE, C.DATE, KEYGENCASTMEMBER(C.CASTMEMBERCODE,
        'DS2'), C.APPEARENCEDURATION, CM.FEEPERHOUR * C.APPEARENCE
                                                    DURATION
    FROM DS2.CAST AS C, DS1.CASTMEMBER AS CM, DS2.SHOW AS S
    WHERE C.CASTMEMBERCODE = CM.CODE AND C.SHOWTITLE=S.TITLE
        AND C.DATE=S.DATE AND S.TYPE="TVSHOW"
UNION
    SELECT A.TVSHOWTITLE, S.DATE, KEYGENCASTMEMBER(A.ANCHORNAME,
        'DS2'), NULL, A.ENGAGEMENTFEE
    FROM DS2.ANCHOR AS A, DS2.SCHEDULE AS S
    WHERE A.TVSHOWTITLE=S.TVSHOWTITLE AND A.TVSHOWEDITION=
        =S.TVSHOWEDITION
UNION
    SELECT TVSHOWTITLE, TVSHOWDATE, KEYGENCASTMEMBER (GUESTNAME, 'DS2')
    FROM DS2.SCHEDULE AS S
)
```