Exam Simulation

An important TV director wants to set up a new TV show with a great anchor and VIP guests. Unfortunately, the TV company database contains only data about TV shows and the director needs to integrate them with the data of another database concerning a wider variety of shows.

The first source is a standard relational database describing various kinds of shows with the following schema:

DS1:

SHOW (title, date, type)
CAST (*show-title, show-date, cast-member-name*, appearance-duration)
CAST-MEMBER (name, address, fee-per-hour, *agent-name*)
AGENT (name, company, phone-number)
CAST-ROLE (*cast-member-name, role-name*)
ROLE (name, type)

(Remarks: Primary keys are underlined, while foreign keys are italicized.
The duration of the show is expressed in hours.
The fees are expressed in dollars.
Dates are expressed as yyyy-mm-dd.
Type can be (*guest*|*cast-member*|*technician*)

The second source is the modern web information system of the TV company which relies on a XML database. The data are described by the following DTD:

DS2:

```
<!DOCTYPE tv-shows-DB SYSTEM "showsDB.dtd">
<!ELEMENT shows-DB (tv-show*)>
<!ELEMENT tv-show (anchor+, schedule+)>
<!ELEMENT anchor EMPTY>
<!ELEMENT schedule (day+)>
<!ELEMENT day (guest+)>
<!ELEMENT guest EMPTY>

<!ATTLIST tv-show title CDATA #REQUIRED>
<!ATTLIST tv-show edition CDATA #REQUIRED>
<!ATTLIST tv-show duration CDATA #IMPLIED>

<!ATTLIST anchor name CDATA #REQUIRED>
<!ATTLIST anchor engagement-fee CDATA #IMPLIED>
<!ATTLIST anchor agent-phone-contact CDATA #IMPLIED>

<!ATTLIST day date CDATA #REQUIRED>
<!ATTLIST day special CDATA (0,1) “0”>

<!ATTLIST guest name CDATA #REQUIRED>
<!ATTLIST guest role CDATA #IMPLIED>
<!ATTLIST guest engagement-fee CDATA #IMPLIED>
<!ATTLIST guest address CDATA #IMPLIED>
```

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

```
<tv-shows-DB>
  <tv-show title=”The David Lettermann Show” edition=”2007/2008” duration=”1”>
    <anchor name=”David Lettermann” engagement-fee=”60,000”/>
    <schedule>
      <day date=”2007-09-14” special=”0”>
        <guest name=”Mike Tyson” engagement-fee=”200,000”/>
      </day>
      <day date=”2007-09-21”>
        <guest name=”Hillary Clinton” engagement-fee=”30,000”/>
        <guest name”Monika Lewinsky” engagement-fee=”100,000”/>
      </day>
    </schedule>
  </tv-show>
</tv-shows-DB>
```

(Remarks: Consider the required attributes as primary keys.
The duration of the show is expressed in hours.
The fees are expressed in dollars.
Dates are expressed as yyyy-mm-dd.)

1) Propose a data integration solution which is able to represent the data of both datasources without information loss, discussing which integration technique you suggest for this scenario (GAV, LAV, GLAV, etc.). You may assume the stability of the datasources.

2) Reengineer the sources to obtain their conceptual models, listing the most relevant mismatches between the two schemata and propose a solution.

3) Propose a global conceptual model as integration solution.

4) Present the mappings (in terms of SQL or Xquery views) needed to map the two data sources to the merged schema you have proposed.

5) (OPTIONAL) Write the following query on the merged schema and show its translation in terms of the data sources:

“Select all the guests and their fee per hour of TV shows that have been on air in special occasions”.

Proposed solution:

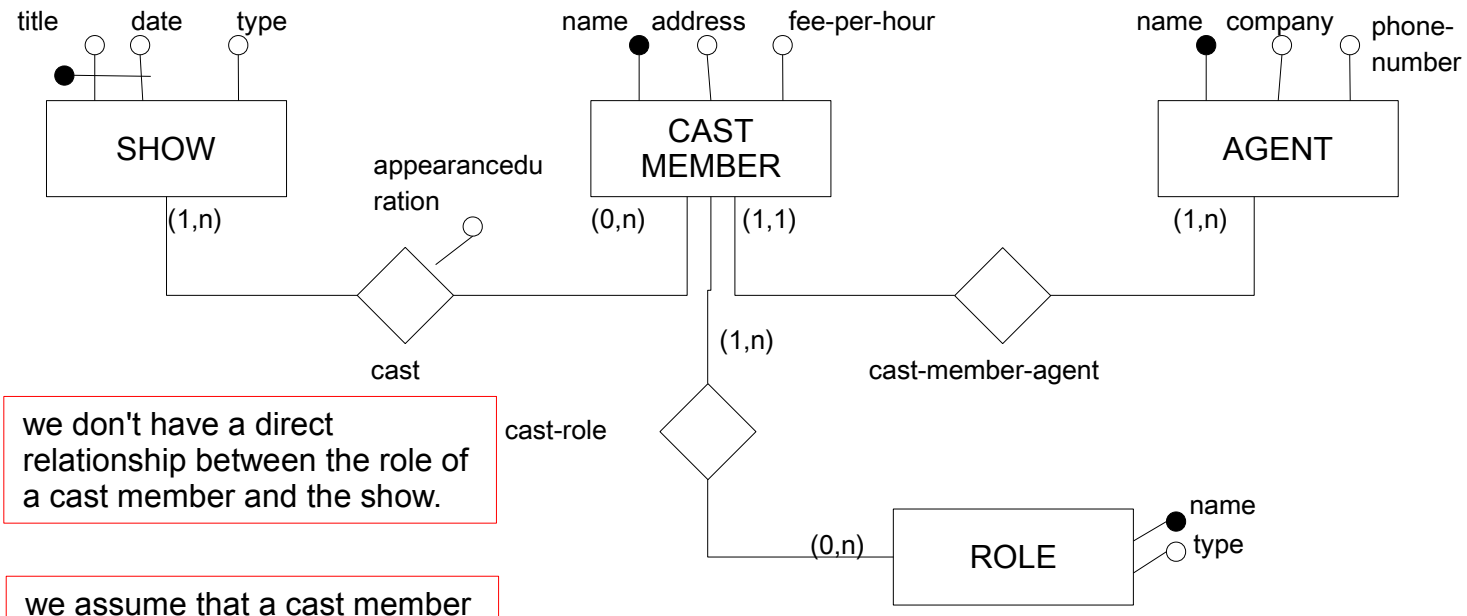
DISCLAIMER: Your teachers are quite expert in data integration. However, it is possible that some of the information you have been provided are incomplete or ambiguous. Discuss all the hypothesis you have made to ease the correction of the exam.

1) We propose a data integration solution based on a merged-schema expressed in the relational data model. As the datasources are known and stable we will build the DIS (Data integration system) using GAV mappings. This will also ease the query processing for the query proposed at point 5.

2) Schemata reverse engineering.

DS1: relational
DS2: XML

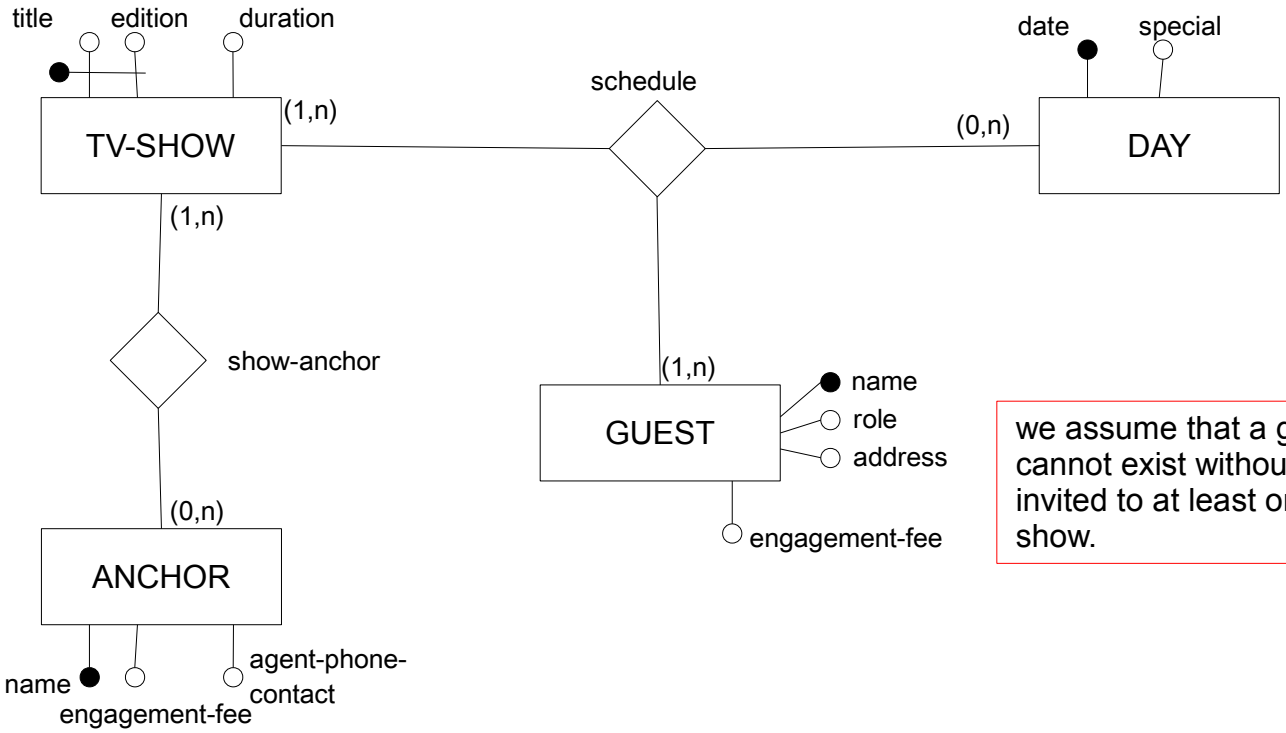
DS1 conceptual schema



we don't have a direct relationship between the role of a cast member and the show.

we assume that a cast member may be in the database even if (s)he has never been involved in a show. Similar assumption for the role.

DS2 conceptual schema



we assume that a guest cannot exist without being invited to at least one show.

Conflicts identification:

1. The whole DS2 represent a subset of the reality represented by DS1 (TV-SHOWS are a subset of SHOWS). Since we are interested only in tv shows we do not need all the data contained in DS1!

2. The way DS1 is designed is however more general, we use that conceptual model as a basis for the integrated schema. For example, the anchor is a cast member as the others.

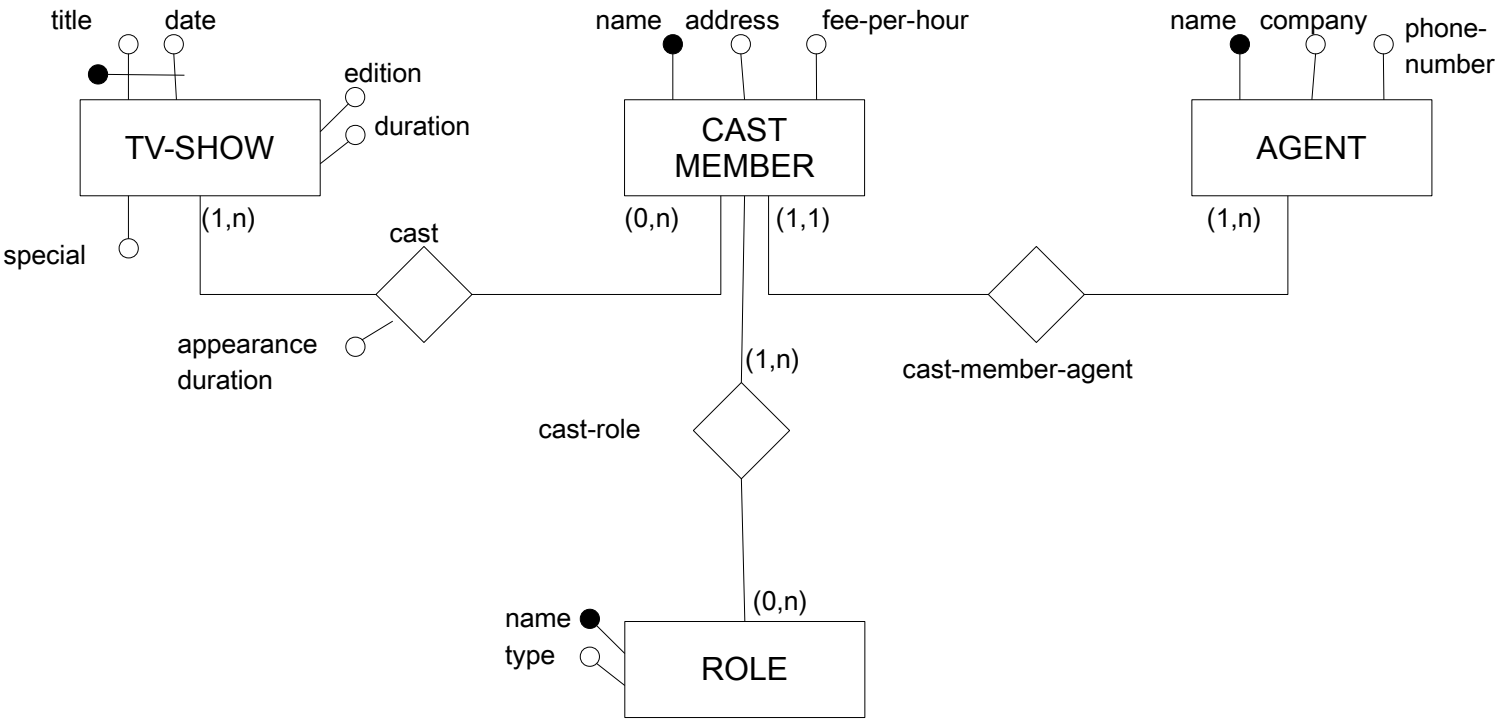
3. Fees are represented as fees per hour in DS1 and in fees per engagement in DS2. We need an appropriate conversion.

4. In DS2 agent is missing we keep the modeling solution of DS1 and we need a function to generate a unique identifier.

5.Special date are present only in DS2. We can use DS2 tuples to reconstruct the missing information in DS1. We assume DS2:DAY as a “calendar” table containing all the days with indication of special days (e.g. Christmas).

6. The type attribute is no longer necessary. The global schema contains only tv shows.

3) Global conceptual schema:



Before creating the GAV mappings we need to translate the DS2 schema and the global schema in the relational model:

DS2: (Fake relational schema)

TV-SHOW (title, edition, duration)
ANCHOR (name, engagement-fee, agent-contact-phone)
GUEST (name, role, address, engagement-fee)
DAY (date, special)
SHOW-ANCHOR (anchor-name, show-name, show-edition)
SCHEDULE (show-name, show-edition, guest-name, show-date)

GS:

TV-SHOW (title, date, edition, duration, special)
CAST-MEMBER (name, address, fee-per-hour, *agent-name*)
AGENT (name, company, phone-number)
CAST (*show-title, show-date, cast-member-name*, appearance-duration)
CAST-ROLE(*cast-member-name, role-name*)
ROLE (name, type)

GAV MAPPINGS

```
create view GS:TV-SHOW (title, date, edition, duration, special) as (
  select DS1:SHOW.title, DS1:SHOW.date, null, null, DS2:DAY.special
  from DS1:SHOW, DS2:DAY
  where DS1:SHOW.type = “tv-show” AND DS1:SHOW.date = DS2:DAY.date
UNION
  select DS2:TV-SHOW.title, DS2:SCHEDULE.date, null, DS2:TV-SHOW.duration,
        DS2:DAY.special
  from DS2:TV-SHOW, DS2:SCHEDULE, DS2:DAY
  where DS2:SCHEDULE.show-name = DS2:TV-SHOW.title AND
       DS2:SCHEDULE.show-date = DS2:DAY.date
)

create view GS:CAST-MEMBER (name, address, fee-per-hour, agent-name) as (
  select DS1:CAST-MEMBER.name, DS1:CAST-MEMBER.address, DS1:CAST-MEMBER.fee-per-hour,
        DS1:CAST-MEMBER.agent-name
  from DS1:CAST-MEMBER, DS1:SHOW, DS1:CAST
  where DS1:CAST-MEMBER.name = DS1:CAST.cast-member-name AND
       DS1:CAST.show-title = DS1:SHOW.title AND
       DS1:CAST.show-date = DS1.SHOW.date AND
       DS1:SHOW.type = “tv-show”
UNION
  select DS2:GUEST.name, DS2:GUEST.address,
        DS2:GUEST.engagement-fee/DS2:TV-SHOW.duration, null
  from DS2:GUEST, DS2:TV-SHOW, DS2:SCHEDULE
  where DS2:GUEST.name = DS2:SCHEDULE.guest-name AND
       DS2:TV-SHOW.title = DS2:SCHEDULE.show-title AND
       DS2:TV-SHOW.date = DS2:SCHEDULE.show-date
UNION
  select DS2:ANCHOR.name, null, DS2:ANCHOR.engagement-fee/DS2:TV-SHOW.duration,
        f1(DS2:ANCHOR.agent-phone-contact)
  from DS2:ANCHOR, DS2:TV-SHOW, DS2:SHOW-ANCHOR
  where DS2:ANCHOR.name = DS2:SHOW-ANCHOR.anchor-name AND
       DS2:TV-SHOW.title = DS2:SHOW-ANCHOR.show-title AND
       DS2:TV-SHOW.date = DS2:SHOW-ANCHOR.show-date
)
```

The other views are left as an exercise.

5) Query on the global schema:

```
select GS:CAST-MEMBER.name, GS:CAST-MEMBER.fee-x-hour
from GS:CAST-MEMBER, GS:CAST-ROLE, GS:ROLE
where GS:CAST-MEMBER.name = GS:CAST-ROLE.cast-member-name AND
      GS:CAST-ROLE.role-name = GS:ROLE.name AND
      GS:ROLE.type = “guest” AND
      GS:CAST-MEMBER.name in (
        select GS:CAST.name
        from GS:CAST, GS:TV-SHOW
        where GS:CAST.show-title = GS:TV-SHOW.title AND
              GS:CAST.show-date = GS:TV-SHOW.date and
              GS:TV-SHOW.special = “1”
      )
```