CS 425 Fall 2004
Exam 1 solutions

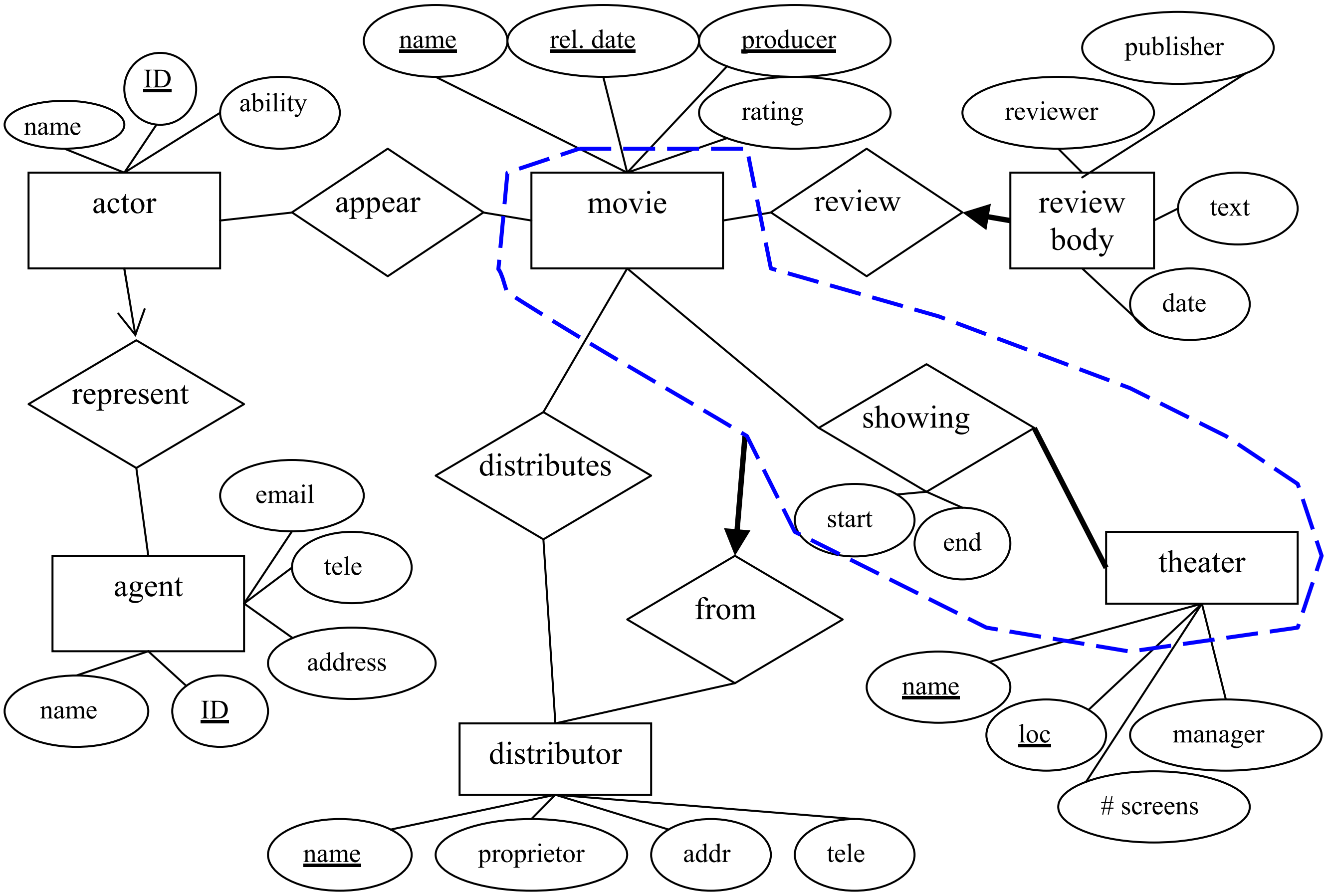
Problem 1:
Entity key constraints:

For movie: name, producer, release date
For theater: name, location
For distributor: business name
For actor: Equity ID
For agent: taxpayer ID

Other constraints:

Numbers of screens ≥ 1.
There is only one distributor for any one movie in one theater.
The number of movies showing in a theater is equal to the number of screens.
Each actor has at most one agent.

Problem 2:



This is one of several correct ER diagrams. Most variations are different ways of trying to capture constraints listed in Problem 1 that are not entity key constraints. Of those four constraints, only “Each actor has at most one agent.” is easily captured - as a key constraint. The solution here captures “There is only one distributor for any one movie in one theater.” by using aggregation to relate a “movie showing in a theater” to a unique distributor with a key and participation constraint from the movie-theater pair. However, the aggregation results in a consistency constraint that cannot be captured: the distributor related to the “movie showing in theater” pair through the *from* relationship must be related to that movie in the *distributes* relationship. Constraints “Numbers of screens ≥ 1” and “The number of movies showing in a theater is equal to the number of screens.” cannot be captured (they are constraints relating values of entities), but these constraints do imply the total participation constraint of *theater* in *showing.* Note that this solution interprets a movie review as a prose critique; one would expect the text to be specific to a certain movie. Other interpretations were not penalized if represented correctly.

Problem 3:

```
create table movie (
        name char(30),
        producer char(30),
        rel_date char(8),
        rating char,
        primary key (name, producer, rel_date)  )

create table theater (
        name char(30),
        loc char(30),
        #_screens integer,
        manager char(50),
        primary key (name, loc.),
        check ( #_screens >= 1)   )

create table distributor (
        name char(30),
        proprietor char(50),
        addr char(100),
        tele char(10)
        primary key (name) )

create table actor (
        name char(50),
        ID char(10),
        ability integer,
        agent_ID char(20),
        primary key (ID),
        foreign key (agent_ID) references agent  )

create table agent (
        name char(50),
        ID char(20),
        address char(100),
        tele char(10),
        email char(50),
        primary key (ID)   )

create table review (
        reviewer char(50),
        publisher char(50),
        text char(5000),
        date char(8),
        name char(30) not null,
        producer char(30) not null,
        rel_date char(8) not null,
        primary key (reviewer, publisher, text, date),
        foreign key (name, producer, rel_date) references movie  )

create table distributes (
        name char(30),
        producer char(30),
        rel_date char(8),
        distrib_name char(30),
        primary key (name, producer, rel_date, distrib_name),
        foreign key (name, producer, rel_date) references movie,
        foreign key (distrib_name) references distributor )

create table showing (
        name char(30),
        producer char(30),
        rel_date char(8),
        t_name char(30),
        t_loc char(30),
        distrib_name char(30) not null,
        start char(8),
        end char(8),
        primary key( name, producer, rel_date, t_name, t_loc )
        foreign key (name, producer, rel_date, distrib_name) references distributes,
        foreign key (t_name, t_loc) reference theater  )
```

```
create table appear (
        name char(30),
        producer char(30),
        rel_date char(8),
        ID char(10),
        foreign key (name, producer, rel_date) references movie,
        foreign key(ID) references actor  )

create assertion all_screens
check ( not exists   (
                        select T.name, T.loc
                        from theater T
                        where  T.#_screens != (
                                                select count(*)
                                                from showing S
                                                where (S.t_name = T.name) AND
                                                        (S.t_loc = T.loc)
                                                )
                        )
        )
```

Problem 4:

A.

$$\begin{array}{l} \Pi_{ID,\ Birthdate} \ (\\ \quad (\Pi_{DogID} \sigma_{Impairment = 'total'} \\ \qquad (Client\_Dog\_Relation \bowtie_{ClientName = Name\ AND\ ClientAddress = Address} Client) \ ) \\ \quad \bowtie_{DogID=ID} Dog \ ) \end{array}$$

B.

$$\begin{array}{l} (\Pi_{CertificationTrainerSSN} (Trained\_Dog) \ ) \quad \cup \\ (\Pi_{SSN} \ \sigma_{NumberYearsService > 2} \ (Trainer) \ ) \end{array}$$

C.

$$\begin{array}{l} \Pi_{SSN,\ Name,\ Address} (\quad Trainer \ \bowtie_{SSN=TrainerSSN} \ (\\ \ (\Pi_{TrainerSSN,\ Breed} (Trainer\_Dog\_Relation \bowtie_{DogID=ID} Dog) \ ) \ / \ (\Pi_{Breed} Dog \ ) \\ \qquad\qquad\qquad\qquad\qquad\qquad) \ \ ) \end{array}$$

Problem 5:
A.

```
{ < S,N,A > | EXISTS(Y,L) (   ( < S,N,A,Y,L> ε Trainer ) AND
                          FORALL(B) (
                               ( EXISTS($I_1$,$T_1$) (<$I_1$,B,$T_1$ > ε Dog ) ) =>
                               EXISTS($I_2$,$T_2$) ( (<$I_2$,B,$T_2$ > ε Dog ) AND
                                          (<S,$I_2$ > ε Trainer_Dog_Relation)  )
                                    )
                                 )                                          }
```

B.

```
select  count(*) as count_in_breed, D.breed
from Trained_Dog  T, Dog D
where T.ID = D.ID
grouped by D.Breed
```