Relazione di Base di dati

Oldstyle rental

Gruppo 9
PETER MAHROUS LOKA 223530

ANALISI DEL PROGETTO:

Questo progetto serve a realizzare una base di dati in grado di soddisfare i bisogni di una catena di negozi fisici specializzati nel noleggio di film.

PRIMA FASE:

Nella prima fase dobbiamo fare l'analisi dei dati. Viste le ambiguità che il linguaggio formale possa offrire, abbiamo dovuto effettuare delle modifiche per poter rimuovere queste ambiguità. Poi vengono tradotti i requisiti in un modello entità/relazioni, questi passaggi ci portano alla fine alla fase di progettazione concettuale.

PROGETTAZIONE CONCETTUALE:

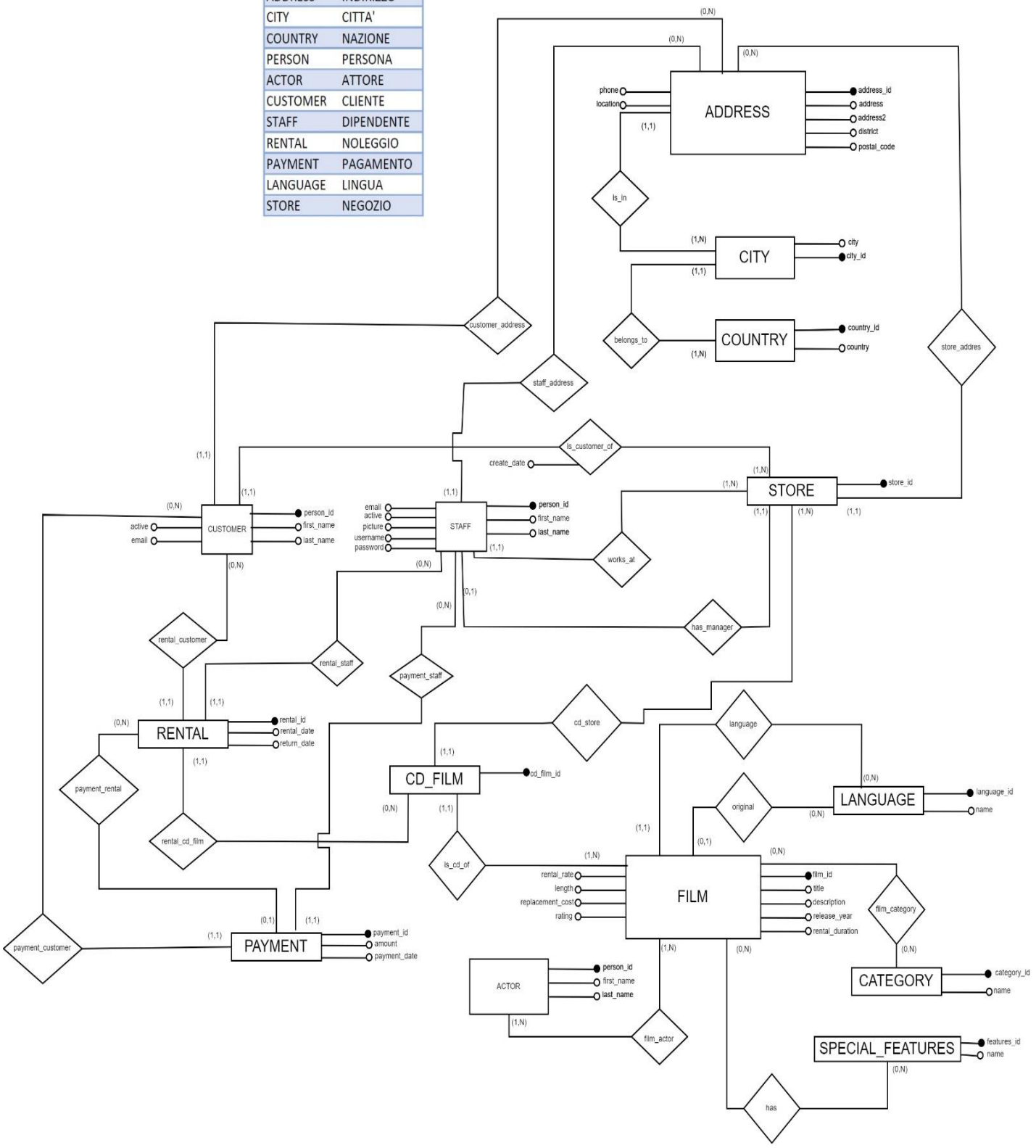
In questa progettazione concettuale vediamo le entità le relazioni del nostro database "old style rental", ci sono state alcune modifiche sulla progettazione concettuale.

-abbiamo eliminato alcune generalizzazioni per poter fare lo schema logico.

- l'Entità Category: prima era stata messa come un attributo nell'entità film, poi vedendo che un film può avere più di una categoria abbiamo dovuto creare l'entità category.

- L'entità staff aveva l'attributo booleano is_manager che indicava se quel membro dello staff era un manager dello store in cui lavorava o meno adesso invece abbiamo aggiunto la relazione has_manager sull'indicazione che abbiamo avuto a lezione.

Entità	Concetto
ADDRESS	INDIRIZZO
CITY	CITTA'
COUNTRY	NAZIONE
PERSON	PERSONA
ACTOR	ATTORE
CUSTOMER	CLIENTE
STAFF	DIPENDENTE
RENTAL	NOLEGGIO
PAYMENT	PAGAMENTO
LANGUAGE	LINGUA
STORE	NEGOZIO



PROGETTAZIONE LOGICA:

actor (actor_id, first_name, last_name)

country (country_id, country)

category (category_id, name)

special_features(feature_id, name)

language(language_id, name)

staff (staff_id, first_name, last_name, address_id*, picture, e-mail, store_id*, active, username, password) con vincolo di integrità referenziale tra l'attributo store_id e la relazione store

city (city_id, city, country_id*) con vincolo di integrità referenziale tra l'attributo country_id e la relazione country

address(address_id, address, address2, district, city_id*, postal_code, phone, location) con vincolo di integrità referenziale tra l'attributo city_id e la relazione city

store (store_id, manager_staff_id*, address_id*) con vincolo di integrità referenziale tra l'attributo manager_staff_id e la relazione staff e tra l'attributo address_id e la relazione address

customer (customer_id, store_id*, first_name, last_name, e-mail, address_id*, active, create_date) con vincolo di integrità referenziale tra l'attributo store_id e la relazione store e tra l'attributo address_id e la relazione address

film (film_id, title, description, release_year, language_id*, original_language_id*, rental_duration, rental_rate, lenght, replacement_cost, rating) con vincolo di integrità referenziale tra gli attributi language _id e original_language_id la relazione language

cd_film (cd_film_id, film_id*, store_id*) con vincolo di integrità referenziale tra l'attributo store_id e la relazione store e tra l'attributo film_id e la relazione film

rental (rental_id, rental_date, cd_film_id*, customer_id*, return_date, staff_id*) con vincolo di integrità referenziale tra l'attributo customer_id e la relazione customer e tra l'attributo cd_film_id e la relazione cd_film e tra l'attributo staff_id e la relazione staff

payment (payment_id, customer_id*, staff_id*, rental_id*, amount, payment_date) con vincolo di integrità referenziale tra l'attributo customer_id e la relazione customer e tra l'attributo staff_id e la relazione staff e tra l'attributo rental_id e la relazione rental
film_actor (actor_id*, film_id*) con vincolo di integrità referenziale tra l'attributo actor_id e la relazione actor e tra l'attributo film_id e la relazione film

film_category (film_id*, category_id*) con vincolo di integrità referenziale tra l'attributo category_id e la relazione category e tra l'attributo film_id e la relazione film

LABORATORIO:

dopo aver fatto lo schema logico possiamo procedere alla creazione, popolazione, modifica e interrogazione al nostro database, usando il linguaggio SQL

esempio per la creazione della entità country:

```
CREATE TABLE `country` (
  `country_id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
  `country` varchar(50) NOT NULL,
  `last_update` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp(),
  PRIMARY KEY (`country_id`))
```

esempio per la popolazione della entità country:

```
INSERT INTO `country` VALUES (1,'Egitto','1996-10-29 00:00:00'),
                             (2,'Itaia','2006-02-15 03:44:00')
```

Dopo aver creato il database per completo e abbiamo popolato le entità abbiamo iniziato ad interrogare il DB per capire meglio come si fanno le query in SQL.
È stato richiesto di svolgere alcune query nel documento dei requisiti.

1. Popolamento ed aggiornamento delle tabelle.

2. Trovare i negozi locati in Italia.

```
select S.store_id
from store S, address A, city C, country CT
where S.address_id=A.address_id and A.city_id=C.city_id and
        C.country_id=CT.country_id and
        CT.country="Italy"
```

3. Trovare i titoli dei film noleggiati nel 2020.

```
select f.title
from film as f , cd_film as cd, rental as r
where f.film_id=cd.cd_film_id and
        cd.cd_film_id=r.cd_film_id and
        year(rental_date) ="2020”
```

4. Trovare i titoli dei film con il maggior numero di noleggi di sempre.

```
select F.title, count(rental_id)
from film as f, rental as r, cd_film as cf
where f.film_id= cf.film_id and
         cf.cd_film_id=r.cd_film_id
```

```
group by F.title
```

5. Trovare, per ogni categoria, il numero di noleggi totali.

```
select count(r.rental_id) as noleggi, c.name
from rental as r ,category as c, film_category as fc, cd_film as cd
where r.cd_film_id=cd.cd_film_id and
        cd.film_id=fc.film_id and
        fc.category_id=c.category_id
group by c.name
```

6. Calcolare il totale dei pagamenti incassati da ciascun negozio.

```
select sum(p.amount) as totale ,s.store_id
from store as s, staff as st, rental as r, payment as p
where s.store_id=st.store_id and
        st.staff_id=r.staff_id and
        r.rental_id=p.rental_id
group by s.store_id;
```