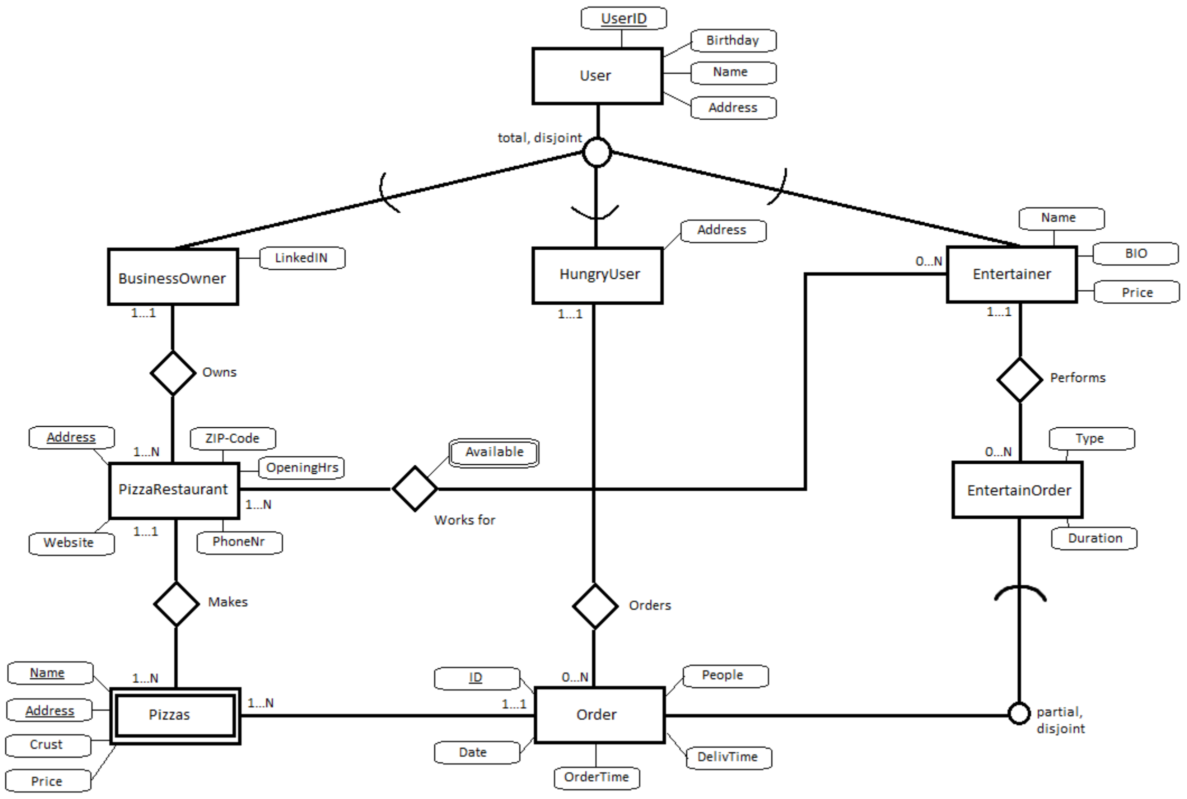
Assignment 2020-2021: Principles of Database Management

Daniel van Dijk – r0745998

Question 1



When a user enters the app, they're asked to share their birthdate, name and address. Next, a User is created with an ID, which is a key attribute as it uniquely identifies the user. The information given by the user is saved under normal attribute types shown in the model. Next the user has to pick one of three options: BusinessOwner, HungryUser or Entertainer. These are subclasses of the superclass User. This specialization is total and disjoint, because the user is forced to pick an option and can pick only one.

A businessowner must own at least one pizza restaurant in order to be eligible for the app, which is why there is a 1 to N relationship. Each pizza restaurant is managed by one app user and therefore has a 1 to 1 relationship. The key attribute for PizzaRestaurant is address as it can uniquely identify a single restaurant. We also save the zip-code, opening hours, website and phone number. Most pizzerias serve the same type of pizzas while we can all agree that one margarita is nothing like the other. In order to identify each and every unique pizza the key attribute types for Pizzas are Name and Address. The entity type Pizzas is therefore weak and dependent on PizzaRestaurant of whom it uses the attribute Address. For each pizza the price and crust type are saved.

The second option in the app is to select 'hungry customer' upon which the user must share their desired delivery address. The user can place an order for one or more pizzas, as shown in the model, and is asked to share a preferred delivery time and for how many people the order is. Each order gets a unique ID, and the date and time of the order are automatically saved. There is also an option for a special type of order: the entertainment order. As not every order is an entertainment order, the specialization between Order and EntertainOrder is partial. When the special order is picked the type of entertainment and duration must be given and is saved.

The third option for a user is to pick for Entertainer. An entertainer must share their stage name, bio and price per 30 minutes of entertaining. An Entertainer can be associated with 0 to many restaurants and vice versa. An entertainer can have non or a multitude of entertaining orders, whilst each order is executed by a single entertainer. Entertainers can work for several restaurants and they communicate to each when they are available. This is a multi-valued attribute type because an entertainer can be available multiple days of the week

There are a few semantics that cannot be enforced by the EER model. Temporal constraints cannot be enforced in the ER model so the attribute for the desired delivery time currently only serves as information. Domains are also not given in an ER model which means users can input illogical things for addresses, times, prices, etc. The ER model also doesn’t include any functions, so there are no options to determine the cheapest margarita pizza offered on the app.

Question 2

CITY (PostalCode, Name, Mayor, Province, Population)

LEGAL ENTITY(ID, Address, *PostalCode*)
- PostalCode refers to PostalCode in relation CITY: NOT NULL on delete/update cascade

BUSINESS(*ID*, Name, DateOfRegistration, BusinessID)
- ID refers to ID in relation LEGAL ENTITY: NOT NULL on delete/update cascade

INDIVIDUAL(*ID*, FirstName, LastName, DateOfBirth)
- ID refers to ID in relation LEGAL ENTITY: NOT NULL on delete/update cascade

BUILDING PERMIT(BP-ID, *Applicant*, Zone, Address)
- Applicant refers to ID in relation LEGAL ENTITY: NOT NULL on delete/update cascade

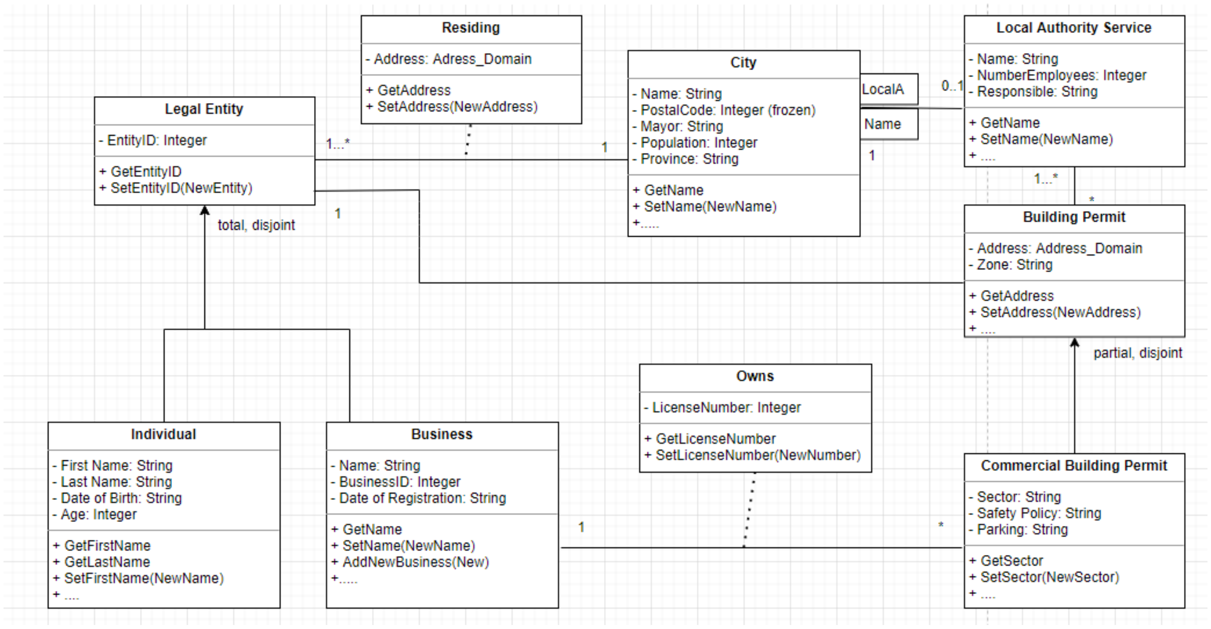
COMMERCIAL BUILDING PERMIT(*BP-ID*, *Business*, LicenseNumber, Sector, SafetyPolicy, Parking)
- BP-ID refers to BP-ID in relation BUILDING PERMIT: NOT NULL on delete/update cascade
- Business refers to ID in relation BUSINESS: NOT NULL on delete/update cascade

LOCAL AUTHORITY SERVICE(Name, *PostalCode*, Employees, Responsible)
- PostalCode refers to PostalCode in relation CITY: NOT NULL on delete/update cascade

ISSUES(*LAS*, *BP-ID*)
- LAS refers to Name in relation LOCAL AUTHORITY SERVICE: NOT NULL on delete/update cascade
- BP-ID refers to BP-ID in relation BUILDING PERMIT: NOT NULL on delete/update cascade

There are a few semantics that are lost in the relational database model. It is impossible to enforce the minimum cardinality of one to M options in the relational model. For example, we cannot enforce that every city belongs to at least one local authority service, or that each building permit is issued by at least one local authority service. It is also not possible to enforce that the two specializations in the model are disjoint.

Question 3



Both the dark arrows should be white ones to indicate a specialization, instead of the current black one which indicates a unidirectional association.

Question 4

(a)
```
SELECT A.PRODNR, A.SUPNR, A.PURCHASE_PRICE, B.SUPNR, B.PURCHASE_PRICE
FROM SUPPLIES A INNER JOIN SUPPLIES B
ON A.SUPNR < B.SUPNR
AND A.PRODNR = B.PRODNR
```

(b)
```
SELECT A.SUPNR, A.SUPSTATUS
FROM SUPPLIER A
WHERE A.SUPSTATUS IS NOT NULL
AND 10<
        (SELECT COUNT(*)
        FROM SUPPLIER B
        WHERE A.SUPSTATUS > B.SUPSTATUS)
```

(c)
```
SELECT A.SUPNAME, A.SUPADDRESS, A.SUPCITY
FROM SUPPLIER A AND PRODUCT B
WHERE NOT EXISTS
        (SELECT *
        FROM PRODUCT B
        WHERE B.AVAILABLE_QUANTITY > 0
```

(d)
```
SELECT A.SUPNAME
FROM SUPPLIER A
WHERE A.SUPSTATUS >= ALL
        (SELECT B.SUPSTATUS
        FROM SUPLIER B
        WHERE B.SUPSTATUS IS NOT NULL)
```

(e)
```
SELECT A.PRODNR, A.PRODNAME
        (SELECT SUM(QUANTITY)
        FROM PO_LINE B
        WHERE A.PRODNR = B.PRODNR) AS TOTAL
FROM PRODUCT A
WHERE TOTAL < ANY
        (SELECT SUM(QUANTITY)
        FROM PO_LINE C
        WHERE A.PRODNR = C.PRODNR
```