Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – February 10th, 2016

Available time: 1h 50m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliCo is a big company composed of many branches in various countries, and whose employees are often on travel for business reasons. Each business trip is performed by an employee and is associated with a certain expenditure item. The trip must be authorized by the employee who is responsible for that expenditure item.
Since business trips represent a very relevant expense for *PoliCo*, the management of the company asked you to design a data warehouse to analyze them.

The following is the schema of the *PoliCo* operational database:

BUSINESSTRIP (SSNEmployee, StartingDate, DurationInDays, FoodCost, LodgingCost, TransportationCost, DestinationCity, ExpenditureItemCode)
EMPLOYEE (SSN, Surname, GivenName, HomeCity, Role, BranchName) *// Role may assume the values "Clerk", "Manager" or "Technician".*
CITY (CityName, Country)
EXPENDITUREITEM (ExpenditureItemCode, Name, Supervisor, YearlyAvailableBudget)
// YearlyAvailableBudget represents the total expense which can be associated with the expenditure item in a year.
COMPANYBRANCH (BranchName, BranchCity)

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

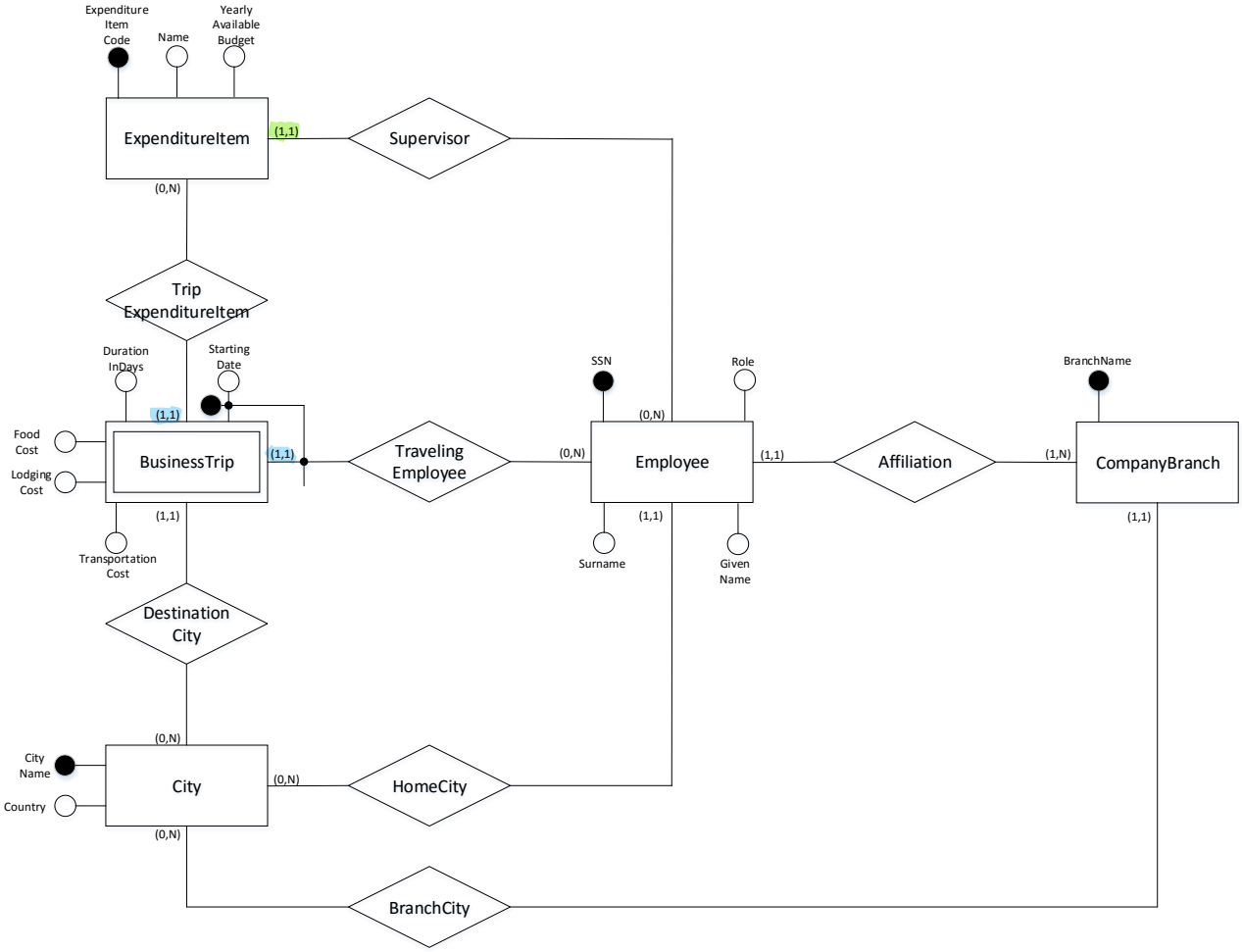
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2 points) Identify the measure(s) and produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (1.5 points) Find the average duration in days of the business trips performed by technicians grouped by departure month and city of the company branch to which the employee belongs.
    b. (1.5 points) Considering only the trips related to the Italian branches of *PoliCo*, find the total expense for each starting date, company branch and home country of the traveling employee. Include in the answer also the aggregations computed using only one or two of the three attributes.
    c. (2.5 points) Select SSN, given name and surname of the supervisor(s) having authorized the greatest number of trips with departure in the second quarter of 2015 and performed by employees residing in Milan.
    d. (2.5 points) Select for each expenditure item the Italian city/cities which has/have been the destination of the greatest number of trips with departure in 2014.

SOLUTION

1. Reverse engineering

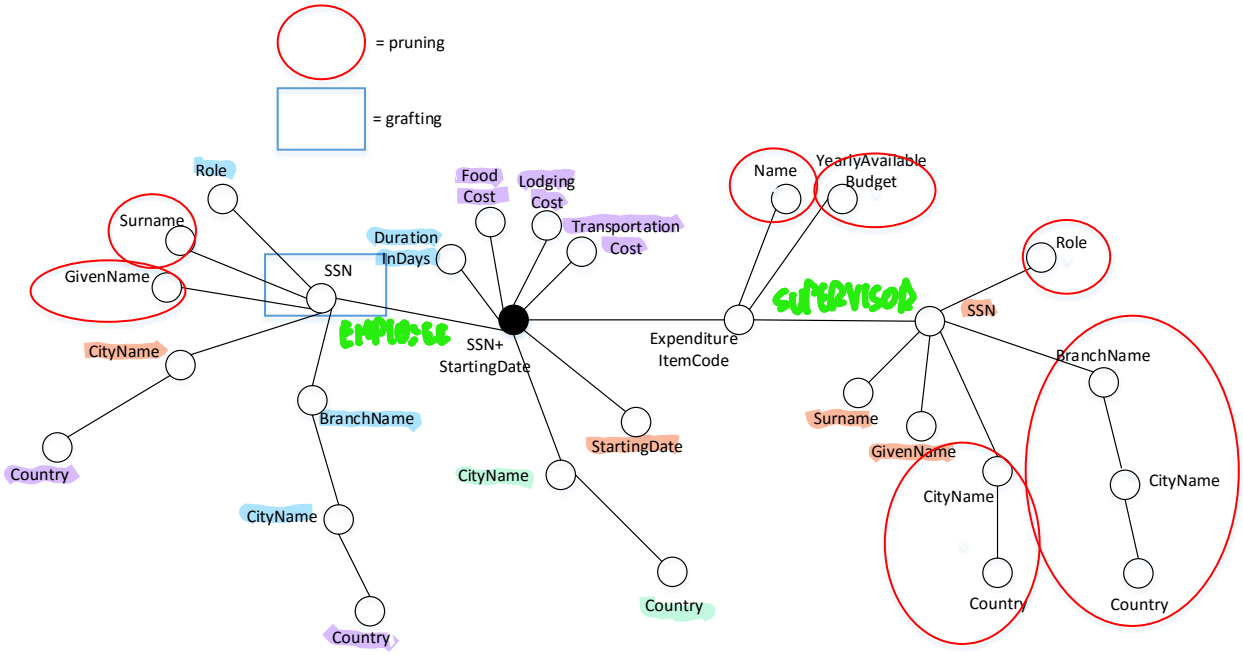


2. Conceptual design

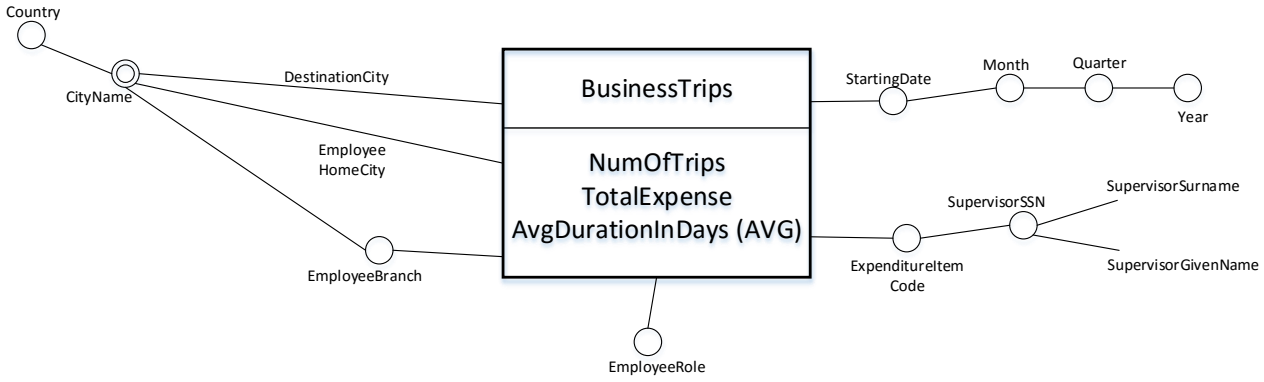
Fact: BusinessTrips (BusinessTrip entity)

RELEVAM FOR QUERIES
a b c d

2a) Attribute tree



2b) Fact schema



2c) Glossary

NumOfTrips

SELECT B.StartingDate, B.DestinationCity, B.ExpenditureItemCode, E.HomeCity, E.Role, E.BranchName,
COUNT(*)
FROM BusinessTrip **AS** B, Employee **AS** E
WHERE B.SSNEmployee=E.SSN
GROUP BY B.StartingDate, B.DestinationCity, B.ExpenditureItemCode, E.HomeCity, E.Role, E.BranchName

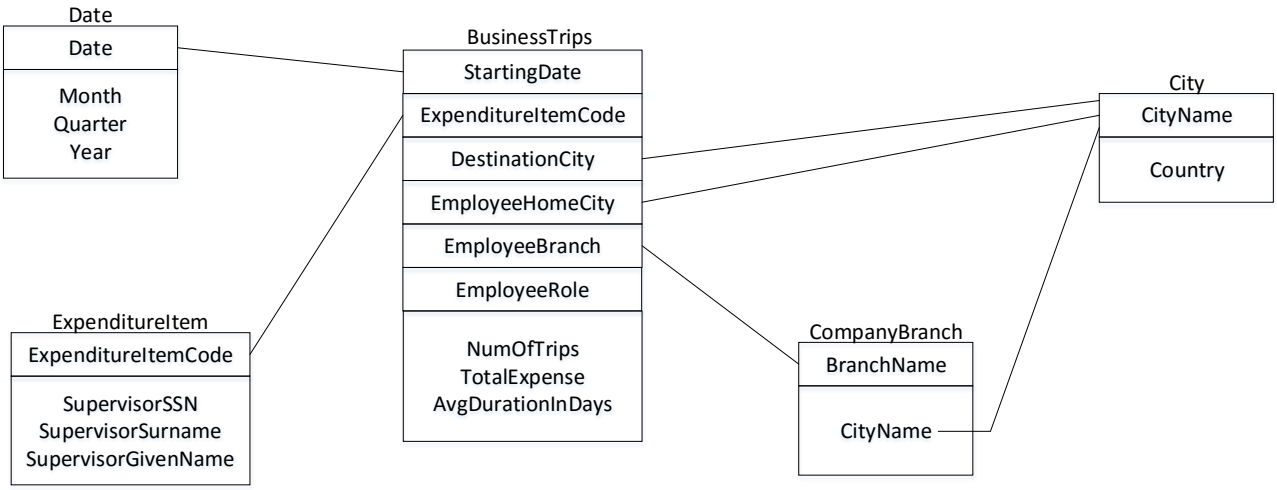
TotalExpense

SELECT B.StartingDate, B.DestinationCity, B.ExpenditureItemCode, E.HomeCity, E.Role, E.BranchName,
SUM(B.FoodCost+B.LodgingCost+B.TransportationCost)
FROM BusinessTrip **AS** B, Employee **AS** E
WHERE B.SSNEmployee=E.SSN
GROUP BY B.StartingDate, B.DestinationCity, B.ExpenditureItemCode, E.HomeCity, E.Role, E.BranchName

AvgDurationInDays

SELECT B.StartingDate, B.DestinationCity, B.ExpenditureItemCode, E.HomeCity, E.Role, E.BranchName,
AVG(B.DurationInDays)
FROM BusinessTrip **AS** B, Employee **AS** E
WHERE B.SSNEmployee=E.SSN
GROUP BY B.StartingDate, B.DestinationCity, B.ExpenditureItemCode, E.HomeCity, E.Role, E.BranchName

3. Logical design



4. Query answering

4a) Find the average duration in days of the business trips performed by technicians grouped by departure month and city of the company branch to which the employee belongs.

SELECT D.Month, C.CityName, **SUM**(B.NumOfTrips*B.AvgDurationInDays)/**SUM**(B.NumOfTrips)
FROM BusinessTrips **AS** B, Date **AS** D, CompanyBranch **AS** C
WHERE B.StartingDateDate=D.Date **AND** B.EmployeeBranch=C.BranchName **AND**
B.EmployeeRole='Technician'
GROUP BY D.Month, C.CityName

4b) Considering only the trips related to the Italian branches of *PoliCo*, find the total expense for each starting date, company branch and home country of the traveling employee. Include in the answer also the aggregations computed using only one or two of the three attributes.

```
SELECT B.StartingDate, B.EmployeeBranch, C-E.Country, SUM(B.TotalExpense)
FROM BusinessTrips AS B, City AS C-E, CompanyBranch AS C, City AS C-B
WHERE B.EmployeeHomeCity=C-E.CityName AND B.EmployeeBranch=C.BranchName AND
      C.CityName=C-B.CityName AND C-B.Country='Italy'
GROUP BY B.StartingDate, B.EmployeeBranch, C-E.Country WITH CUBE
```

4c) Select SSN, given name and surname of the supervisor(s) having authorized the greatest number of trips with departure in the second quarter of 2015 and performed by employees residing in Milan.

```
CREATE VIEW AuthorizedTrips (SupervisorSSN, SupervisorGivenName, SupervisorSurname, NumOfTrips) AS
(
      SELECT E.SupervisorSSN, E.SupervisorGivenName, E.SupervisorSurname, SUM(B.NumOfTrips)
      FROM BusinessTrips AS B, ExpenditureItem AS E, Date AS D
      WHERE B.ExpenditureItemCode=E.ExpenditureItemCode AND B.StartingDate=D.Date AND
            D.Quarter='2-2015' AND B.EmployeeHomeCity='Milan'
      GROUP BY E.SupervisorSSN, E.SupervisorGivenName, E.SupervisorSurname
)

SELECT SupervisorSSN, SupervisorGivenName, SupervisorSurname
FROM AuthorizedTrips
WHERE NumOfTrips = (
                        SELECT MAX(NumOfTrips)
                        FROM AuthorizedTrips
                  )
```

4d) Select for each expenditure item the Italian city/cities which has/have been the destination of the greatest number of trips with departure in 2014.

```
CREATE VIEW ExpItemCityTrips (ExpenditureItemCode, CityName, NumOfTrips) AS (
      SELECT B.ExpenditureItemCode, B.DestinationCity, SUM(B.NumOfTrips)
      FROM BusinessTrips AS B, City AS C, Date AS D
      WHERE B.DestinationCity=C.CityName AND B.StartingDate=D.Date AND C.Country='Italy' AND
            D.Year='2014'
      GROUP BY B.ExpenditureItemCode, B.DestinationCity
)

SELECT E.ExpenditureItemCode, E.CityName
FROM ExpItemCityTrips AS E
WHERE E.NumOfTrips = (
                        SELECT MAX(E2.NumOfTrips)
                        FROM ExpItemCityTrips AS E2
                        WHERE E2.ExpenditureItemCode=E.ExpenditureItemCode )
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – February 23rd, 2016

Available Time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliHighways and ***UniRoads*** are two companies managing roads. PoliHighways deals only with highways, i.e. roads with a toll to be paid by the drivers. UniRoads, on the contrary, manages a wider variety of roads, including both highways and free-of-charge roads; to manage the latter UniRoads receives a contribution by a funding body, typically a public administration. Each company relies on its own relational database describing the managed roads, the toll payments and the maintenance works.

The two companies adopt different systems for the toll payments. More in detail, PoliHighways adopts a closed system: vehicles have to cross an entrance tollbooth and an exit tollbooth, and the price to be paid is determined on the basis of the covered distance. On the contrary, UniRoads adopts an open system: some tollbooths are located in specific points of the road, and the vehicles that cross them have to pay a fixed amount only depending on the tollbooth. In addition, to use the PoliHighways highways the drivers have to subscribe a contract obtaining an electronic device employed for the payments; the device is associated with the customer and may be used with multiple vehicles, provided that all belong to the same category. On the contrary, neither electronic devices, nor contracts stipulated in advance are used by UniRoads for the payments.

The two companies have recently merged into a unique one named ***UniPoliRoads***. The UniPoliRoads ownership now asks you to integrate the two data sources into a unique one. You must perform the integration assuring to lose the least possible amount of information.

The original relational schemas of the two sources are reported below.

PoliHighways

HIGHWAY (HighwayCode, City1, City2) *// City1 and City2 are the cities where the highway begins and ends, respectively.*

TOLLBOOTH (TollboothCode, HighwayCode, City) *// Non-terminal tollbooths may also be located along the Highway.*

CONTRACT (DeviceNumber, PurchaseDate, CustomerSSN, VehicleCategory) *// VehicleCategory may be "Motorcycle", "Car" or "Truck".*

CUSTOMER (SSN, GivenName, Surname, BirthDate, HomeCity)

PAIDTOLL (DeviceNumber, Date, Time, PlateNumber, EntranceTollbooth, ExitTollbooth, PaidPrice)

MAINTENANCEWORK (HighwayCode, Date, KmPosition, Cost, CompanyName)

UniRoads

ROAD (RoadCode, Type, StartCity, EndCity, FundingBody*) *// The type of the road may assume values like "Highway", "Local road", … The roads that are free of charge have a funding body.*

FUNDINGBODY (Name, Type, ContactPerson)

TOLLBOOTH (TollboothNumber, RoadCode, City, Price)

PAIDTOLL (Date, Time, PlateNumber, TollboothNumber, RoadCode)

MAINTENANCEWORK (RoadCode, Date, KmPosition, Cost, CompanyName)

MAINTENANCECOMPANY (CompanyName, Address, City, Phone)

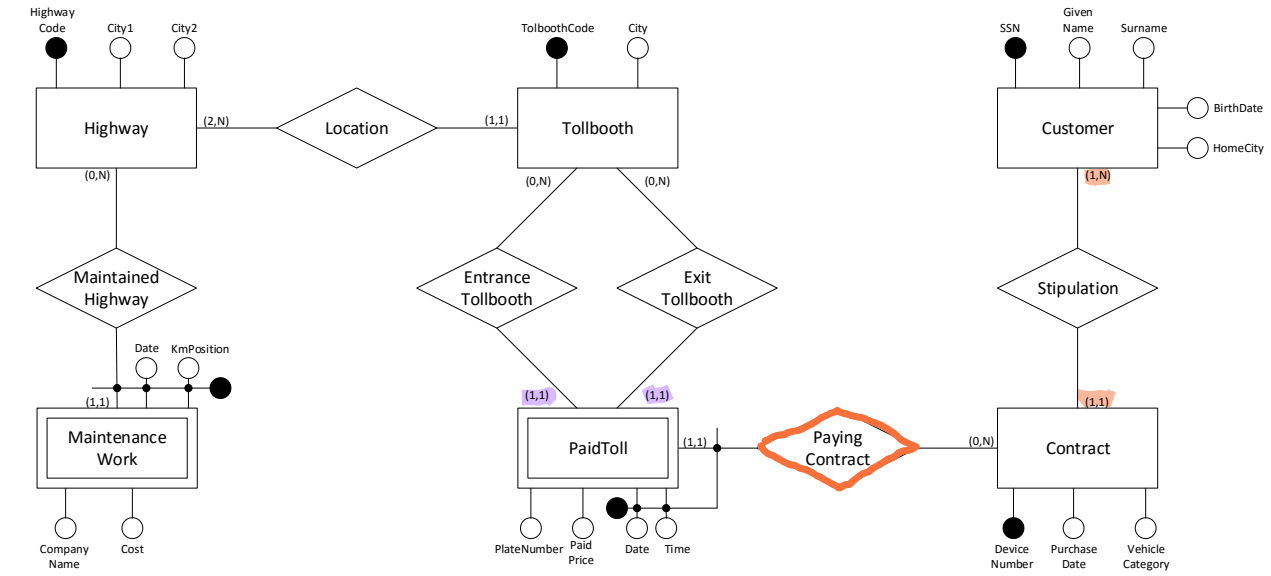
For simplicity, you can assume that the maintenance companies included in the PoliHighways database are disjoint from those in the UniRoads database.

1. **Source schema reverse engineering**. Provide, for each input data source, the reverse engineering from the logical schema to the conceptual model (ER graph). (5 points)
2. **Schema integration**. Design an integrated global conceptual schema (ER graph) for *UniPoliRoads* capturing all the data coming from both *PoliHighways* and *UniRoads*, and provide the corresponding global logical schema. In more detail, follow these steps:
    a. *Related concept identification and conflict analysis and resolution*. Write a table as shown in the exercise sessions, using the following columns: "PoliHighways concept", "UniRoads concept", "Conflict", "Solution". (3.5 points)
    b. *Integrated conceptual schema* (ER graph). (3.5 points)
    c. *Conceptual to logical translation of the integrated schema*. (2 points)
3. **Query answering and mapping definition**. Consider the query Q "Find date, time and plate number of the toll payments associated with tollbooths located in Milan on roads of type 'Highway', on which at least one maintenance work was performed in 2015. For the roads adopting a closed payment system consider just the entrance tollbooths."
    a. *Query formulation*. Consider query Q posed on the logical schema of *UniPoliRoads* and write it either in SQL or in Datalog. (1.5 points)
    b. *Mapping definition*. Write the GAV mappings between the schema of *UniPoliRoads* and the two sources either in SQL or in Datalog. Write the mappings only for the tables used to answer query Q**.** (4 points)
    c. *Query rewriting*. Show the rewriting of Q on the two data sources either in SQL or in Datalog. (2.5 points)

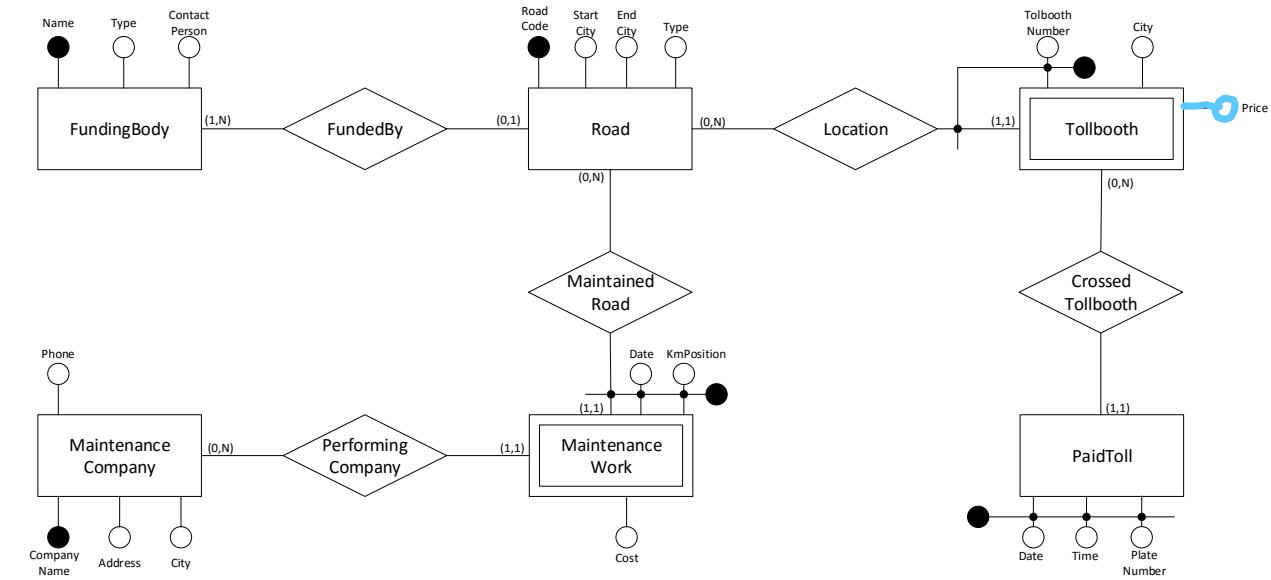
SOLUTION

1. Source schema reverse engineering

PoliHighways



UniRoads

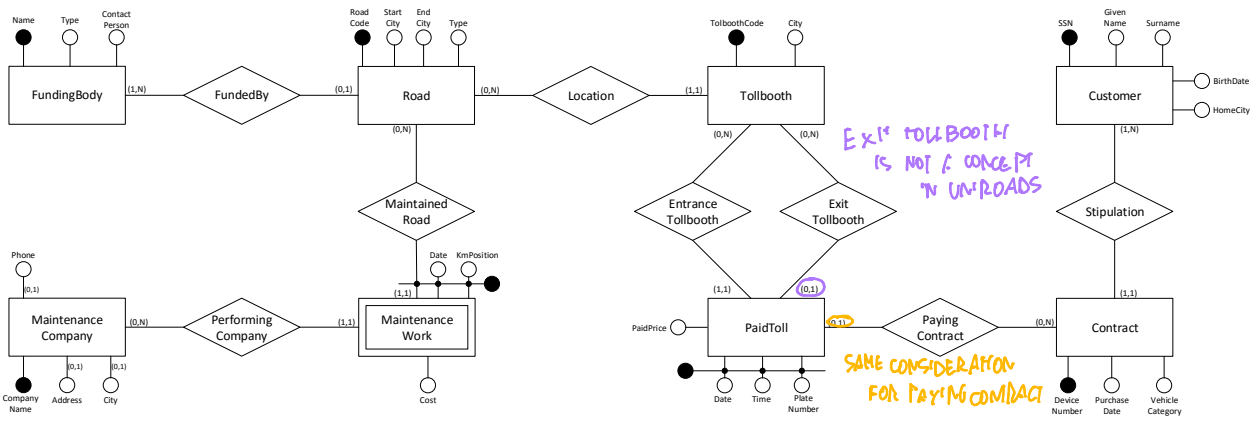


2. Schema integration

2a) Related concept identification + conflict analysis and resolution

PoliHighways	UniRoads	Conflict	Solution
Highway	Road	Name conflicts	
		- Entity name	Road
		- HighwayCode → RoadCode	RoadCode
		- City1 → StartCity	StartCity
		- City2 → EndCity	EndCity
Tollbooth	Tollbooth	Key conflict	
		- TollboothCode → RoadCode+TollboothNumber	TollboothCode
PaidToll	PaidToll	Key conflict	
		- Date + Time + DeviceNumber → Date + Time + PlateNumber	Date + Time + PlateNumber
		Structure conflicts	
		- Each payment is associated with two tollbooths → Each payment is associated with just one tollbooth	Each payment may be associated with two tollbooths
		- The paid price changes with each payment → The paid price is fixed for each tollbooth	The paid price changes with each payment
MaintenanceWork	MaintenanceWork	Structure conflicts	
		- The company is an attribute → The company is an entity	The company is an entity

2b) Global conceptual schema



2c) Conceptual to logical translation

Road (RoadCode, StartCity, EndCity, Type, FundingBody*)
FundingBody (Name, Type, ContactPerson)
Tollbooth(TollboothCode, RoadCode, City)
Contract (DeviceNumber, PurchaseDate, CustomerSSN, VehicleCategory)
Customer (SSN, GivenName, Surname, BirthDate, HomeCity)
PaidToll (Date, Time, PlateNumber, EntranceTollbooth, ExitTollbooth*, PaidPrice, DeviceNumber*)
MaintenanceWork (RoadCode, Date, KmPosition, Cost, CompanyName)
MaintenanceCompany (CompanyName, Address*, City*, Phone*)

3. Query answering and mapping definition

3a) Query formulation

Find date, time and plate number of the toll payments associated with tollbooths located in Milan on roads of type 'Highway' on which at least one maintenance work was performed in 2015. For the roads adopting a closed payment system consider just the entrance tollbooths.

```
SELECT DISTINCT P.Date, P.Time, P.PlateNumber
FROM PaidToll AS P, Tollbooth AS T, Road AS R, MaintenanceWork AS M
WHERE P.EntranceTollbooth=T.TollboothCode AND T.RoadCode=R.RoadCode AND
      R.RoadCode=M.RoadCode AND T.City='Milan' AND R.Type='Highway' AND M.Date BETWEEN
      '1/1/2015' AND '31/12/2015'
```

3b) GAV mapping definition

The KeyGen(_, _) functions generate univocal identifiers.

```
CREATE VIEW UniPoliRoads.Road (RoadCode, StartCity, EndCity, Type, FundingBody) AS (
      SELECT KeyGenRoad(HighwayCode, 'PoliHighways'), City1, City2, 'Highway', null
      FROM PoliHighways.Highway

      UNION

      SELECT KeyGenRoad(RoadCode, 'UniRoads'), StartCity, EndCity, Type, FundingBody
      FROM UniRoads.Road
)
```

```
CREATE VIEW UniPoliRoads.Tollbooth (TollbothCode, RoadCode, City) AS (
      SELECT KeyGenTollbooth(TollboothCode, 'PoliHighways'), KeyGenRoad(HighwayCode,
            'PoliHighways'), City
      FROM PoliHighways.Tollbooth
```

```
        UNION

        SELECT KeyGenTollbooth(TollboothNumber||RoadCode, 'UniRoads'), KeyGenRoad(RoadCode,
                'UniRoads'), City
        FROM UniRoads.Tollbooth
)


CREATE VIEW UniPoliRoads.PaidToll (Date, Time, PlateNumber, EntranceTollbooth, ExitTollbooth,
PaidPrice) AS (
        SELECT Date, Time, PlateNumber, KeyGenTollbooth(EntranceTollbooth, 'PoliHighways'),
                KeyGenTollbooth(ExitTollbooth, 'PoliHighways') , PaidPrice
        FROM PoliHighways.PaidToll

        UNION

        SELECT P.Date, P.Time, P.PlateNumber, KeyGenTollbooth(P.TollboothNumber||P.RoadCode,
                'UniRoads'), null,  T.Price
        FROM UniRoads.PaidToll AS P, UniRoads.Tollbooth AS T
        WHERE P.TollboothNumber=T.TollboothNumber AND P.RoadCode=T.RoadCode
)


CREATE VIEW UniPoliRoads.MaintenanceWork (RoadCode, Date, KmPosition, Cost, CompanyName) AS (
        SELECT KeyGenRoad(HighwayCode, 'PoliHighways'), Date, KmPosition, Cost, CompanyName
        FROM PoliHighways.MaintenanceWork

        UNION

        SELECT KeyGenRoad(RoadCode, 'UniRoads'), Date, KmPosition, Cost, CompanyName
        FROM UniRoads.MaintenanceWork
)
```

3c) Query rewriting

```
SELECT P.Date, P.Time, P.PlateNumber
FROM PoliHighways.PaidToll AS P, PoliHighways.Tollbooth AS T, PoliHighways.MaintenanceWork AS M
WHERE P.EntranceTollbooth=T.TollboothCode AND T.HighwayCode=M.HighwayCode AND T.City='Milan'
        AND M.Date BETWEEN '1/1/2015' AND '31/12/2015'

UNION

SELECT P.Date, P.Time, P.PlateNumber
FROM UniRoads.PaidToll AS P, UniRoads.Tollbooth AS T, UniRoads.Road AS R, UniRoads.MaintenanceWork
        AS M
WHERE P.TollboothNumber=T.TollboothNumber AND P.RoadCode=T.RoadCode AND
        T.RoadCode=R.RoadCode AND M.RoadCode=R.RoadCode AND T.City='Milan' AND
        R.Type='Highway' AND M.Date BETWEEN '1/1/2015' AND '31/12/2015'
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – June 27th, 2016

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliTours is an agency organizing tours in specific cities of Lombardy, relying on a mobile application to promote the tours. A tour is composed of several places of interest to be visited, all located in the same city; each place of interest features a set of landmarks, like monuments or historical buildings. When a user is in a certain city, the PoliTours server delivers to the user's smartphone an XML document describing the tours available for that city, and the mobile application running on the device provides a user-friendly representation of the content of the document. The information system of PoliTours is very primitive: they just keep on their server the collection of XML documents associated with the cities.

UniTours is a similar agency, offering comparable services in Piedmont. A UniTours tour may envisage the visit of points of interest like monuments or historical buildings, but also the visit of companies producing typical local foods. UniTours relies on a better organized data store, constituted by a relational database.

PoliTours and UniTours have recently merged into a unique company named ***UniPoliTours***. The ownership of UniPoliTours wants now to build an integrated relational database storing all the data contained in the PoliTours XML documents and in the UniTours relational database. You must perform the integration assuring to lose the least possible amount of information.

The following is the DTD of the XML documents representing the cities in the PoliTours data store:

```
<!ELEMENT City (Tour+, ReferenceAgency)>
<!ELEMENT Tour (PlaceOfInterest+)>
<!ELEMENT PlaceOfInterest (Landmark+)>
<!ELEMENT ReferenceAgency EMPTY>
<!ATTLIST City    Name CDATA #REQUIRED
                  Province CDATA #REQURED>
<!ATTLIST Tour    Number CDATA #REQUIRED
                  Guide CDATA #REQUIRED
                  DurationInHours CDATA #REQUIRED
                  Price CDATA #REQUIRED>
<!ATTLIST PlaceOfInterest Name CDATA #REQUIRED>
<!ATTLIST Landmark        Name CDATA #REQUIRED
                          Description CDATA #REQUIRED
                          Type CDATA #REQUIRED>
<!ATTLIST ReferenceAgency         Name CDATA #REQUIRED
```

Address CDATA #REQUIRED
City CDATA #REQUIRED
OpeningHours CDATA #IMPLIED>

- Each city is associated with a reference agency, which is not necessarily located in the city itself. A certain agency may be the reference agency for more than one city.
- The names of cities, landmarks, places of interest and reference agencies are unique.
- Each tour is associated with a progressive tour number, unique for each city.
- The guide is represented by a string in the form *Firstname%Lastname*. The company is small, and you can assume that there are no guides with the same firstnames and lastnames.

The following is a portion of the XML document associated with the city of Milan:

```
<City Name="Milan" Province="Milan">
    <Tour Number="1" DurationInHours="5" Guide="Carlo%Rossi" Price="100">
        <PlaceOfInterest Name="Piazza del Duomo">
            <Landmark Name="Duomo of Milan" Description="…" Type="Church"/>
            <Landmark Name="Royal Palace of Milan" Description="…" Type="Palace"/>
        </PlaceOfInterest>
        <PlaceOfInterest Name="Piazza della Scala">
            <Landmark Name="Teatro alla Scala" Description="…" Type="Theater"/>
        </PlaceOfInterest>
    </Tour>
    <Tour Number="2" DurationInHours="7" Guide="Luca%Bianchi" Price="50">
        …
    </Tour>
    <ReferenceAgency Name="Agency1" Address="Via Roma, 5" City="Sesto San Giovanni" OpeningHours="9-
        18"/>
</City>
```

The schema of the relational database of UniTours is as follows:

CITY (<u>CityName</u>, Province, NumberOfInhabitants)
POINTOFINTEREST (<u>POIName</u>, Description, Type, HistoricalPeriod*, CityName) *// Type may assume values like 'Theater', 'Monument', 'Palace', 'Church', …*
FOODCOMPANY (<u>VATNumber</u>, Name, Address, Phone, Email, CompanyType, CityName) *// CompanyType may assume values like 'Wine producer', 'Cheese factory', …*
PRODUCT (<u>ProductName</u>, Year*, Description, FoodCompanyVATNumber) *// This table contains a selection of typical products. Year assumes a value for some kinds of products, like the wines.*
GUIDE (<u>SSN</u>, Firstname, Lastname, Address, BirthDate)
TOUR (<u>TourName</u>, SSNGuide, DurationInMinutes, Cost)
POINTOFINTERESTTOUR (<u>POIName</u>, <u>TourName</u>)
FOODCOMPANYTOUR (<u>FoodCompanyVATNumber</u>, <u>TourName</u>)

<u>Note</u>: You may assume that the names of places of interest, landmarks and points of interests in the two data sources are disjoint. E.g., you can assume that you will not find any point of interest named "Royal Palace of Milan" in Piedmont.

1. **Source schema reverse engineering**. Provide, for each input data source, the reverse engineering from the logical schema to the conceptual model (ER graph). For the XML source *PoliTours* provide also the relational translation of the ER schema. (5 points)

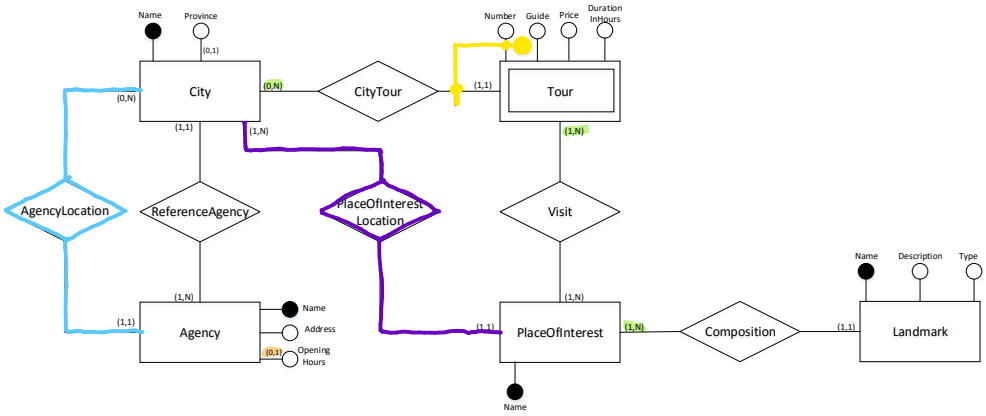
2. **Schema integration**. Design an integrated global conceptual schema (ER graph) for *UniPoliTours* capturing all the data coming from both *PoliTours* and *UniTours*, and provide the corresponding global logical (relational) schema. In more detail, follow these steps:
    a. *Related concept identification and conflict analysis and resolution*. Write a table as shown in the exercise sessions, using the following columns: "PoliTours concept", "UniTours concept", "Conflict", "Solution". (3.5 points)
    b. *Integrated conceptual schema* (ER graph). (3.5 points)
    c. *Conceptual to logical translation of the integrated schema*. (2 points)

3. **Query answering and mapping definition**. Consider the query Q: "Find firstname, lastname and (when available) birth date of the guides who supervise tours envisaging at least one visit in a theater".
    a. *Query formulation*. Consider query Q posed on the logical schema of *UniPoliTours* and write it either in SQL or in Datalog. (1.5 points)
    b. *Mapping definition*. Write the GAV mappings between the schema of *UniPoliTours* and the two sources either in SQL or in Datalog. For *PoliTours*, write the mappings between the *UniPoliTours* integrated relational schema and the *PoliTours* relational schema defined at point 1. Write the mappings only for the tables used to answer query Q**.** (4 points)
    c. *Query rewriting*. Show the rewriting of Q on the two data sources either in SQL or in Datalog. Again, for *PoliTours* consider the relational schema defined at point 1. (2.5 points)

SOLUTION

1. Source schema reverse engineering

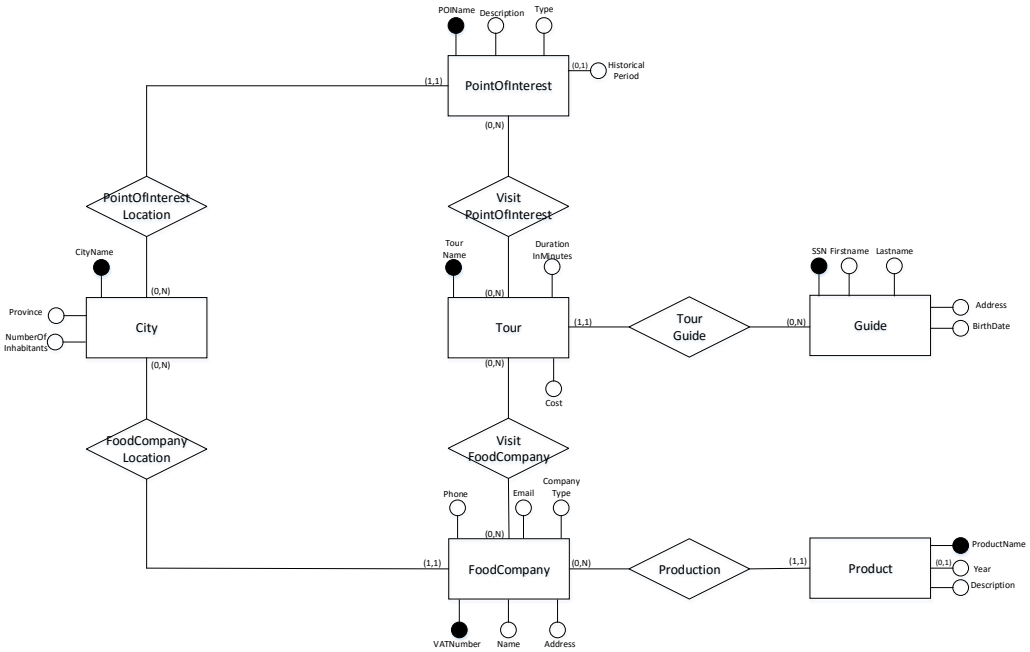
PoliTours



Relational schema
City (Name, Province*, ReferenceAgency)
Agency (Name, Address, OpeningHours*, CityName)
PlaceOfInterest (Name, CityName)
Tour (CityName, Number, Guide, Price, DurationInHours, CityName)
Landmark (Name, Description, Type, PlaceOfInterestName)
PlaceOfInterestTour (PlaceOfInterestName, TourNumber, CityName)

NOTE: In this course we do not study how to implement wrappers. Therefore, we suppose the existence of wrappers guaranteeing the correspondence between the XML source and its relational counterpart, without entering in details.

UniTours

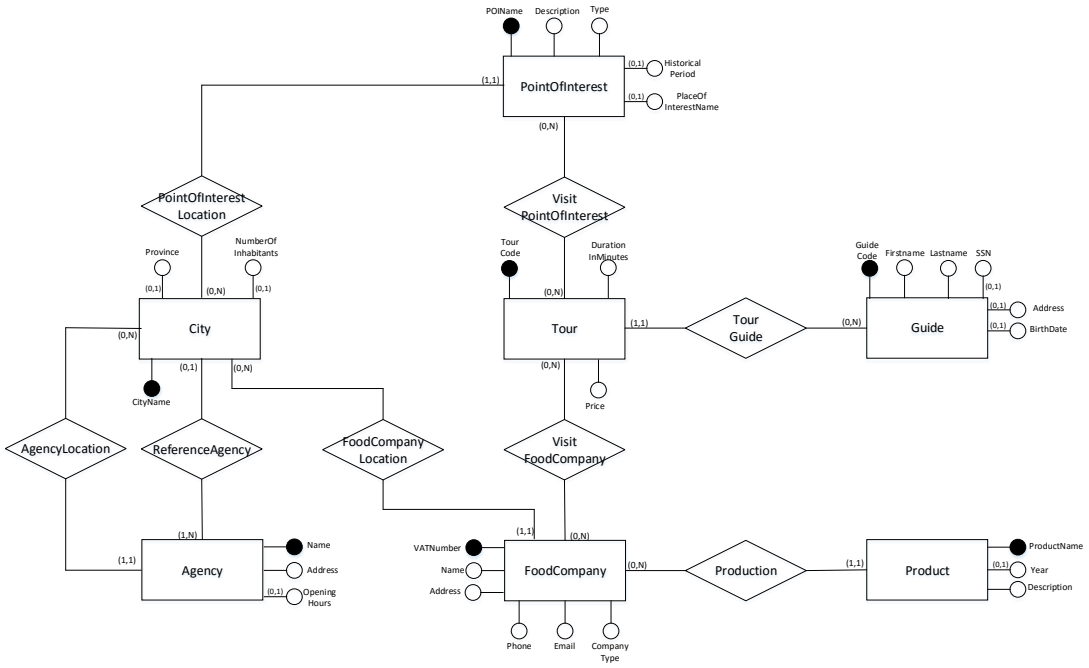


2. Schema integration

2a) Related concept identification + conflict analysis and resolution

PoliTours	UniTours	Conflict	Solution
City	City	Name conflicts	
		- Name → CityName	CityName
Landmark	PointOfInterest	Name conflicts	
		- Entity name	PointOfInterest
		- Name → POIName	POIName
Tour	Tour	Name conflicts	
		- Price → Cost	Price
		Key conflict	
		- CityName+Number → TourName	Generate a TourCode
		Data semantics conflicts	
		- DurationInHours → DurationInMinutes	DurationInMinutes
		Structure conflicts	
		- The tour is associated with places of interest, in their turn associated with landmarks → The tour is directly associated with points of interest (corresponding to landmarks)	Associate the tour directly with landmarks/points of of interest
		Cardinality conflicts	
		- The tour is associated with landmarks in just one city → A tour may be associated with points of interest located in more cities	A tour may be associated with landmarks/points of interest in more cities (the city of the tour will not be specified)
Guide (attribute)	Guide	Structure conflicts	
		- The guide is an attribute → The guide is an entity	The guide is an entity, with a new identifier since SSN is missing in PoliTours
		- Firstname and lastname are represented in the same attribute, separated by '%' → Firstname and lastname are distinct attributes	Firstname and lastname are distinct attributes

2b) Global conceptual schema



2c) Conceptual to logical translation

City (<u>CityName</u>, Province*, NumberOfInhabitants*, ReferenceAgency*)
Agency (<u>Name</u>, Address, OpeningHours*, CityName)
PointOfInterest (<u>POIName</u>, Description, Type, HistoricalPeriod*, PlaceOfInterestName*, CityName)
FoodCompany (<u>VATNumber</u>, Name, Address, Phone, Email, CompanyType, CityName)
Product (<u>ProductName</u>, Year*, Description, FoodCompanyVATNumber)
Guide (<u>GuideCode</u>, Firstname, Lastname, SSN*, Address*, BirthDate*)
Tour (<u>TourCode</u>, GuideCode, DurationInMinutes, Price)
PointOfInterestTour (<u>POIName</u>, <u>TourCode</u>)
FoodCompanyTour (<u>FoodCompanyVATNumber</u>, <u>TourCode</u>)

3. Query answering and mapping definition

3a) Query formulation

Find firstname, lastname and (when available) birth date of the guides who supervise tours envisaging at least one visit in a theater.

SELECT DISTINCT G.Firstname, G.Lastname, G.BirthDate
FROM Guide **AS** G, Tour **AS** T, PointOfInterestTour **AS** PT, PointOfInterest **AS** P
WHERE G.GuideCode=T.GuideCode **AND** T.TourCode=PT.TourCode **AND** PT.POIName=P.POIName **AND**
P.Type='Theater'

3b) GAV mapping definition

The KeyGen(_, _) functions generate univocal identifiers.

```
CREATE VIEW UniPoliTours.Guide (GuideCode, Firstname, Lastname, SSN, Address, BirthDate) AS (
        SELECT KeyGenGuide(Guide, 'PoliTours'), left(Guide, position('%' in Guide)-1),
                right(Guide, length(Guide)-position('%' in Guide)), null, null, null
        FROM PoliTours.Tour

        UNION

        SELECT KeyGenGuide(SSN, 'UniTours'), Firstname, Lastname, SSN, Address, BirthDate
        FROM UniTours.Guide
)


CREATE VIEW UniPoliTours.Tour (TourCode, GuideCode, DurationInMinutes, Price) AS (
        SELECT KeyGenTour(CityName||Number, 'PoliTours'), KeyGenGuide(Guide, 'PoliTours'),
                DurationInHours*60, Price
        FROM PoliTours.Tour

        UNION

        SELECT KeyGenTour(TourName, 'UniTours'), KeyGenGuide(SSNGuide, 'UniTours'),
                DurationInMinutes, Cost
        FROM UniTours.Tour
)


CREATE VIEW UniPoliTours.PointOfInterestTour (POIName, TourCode) AS (
        SELECT L.Name, KeyGenTour(P.TourCity||P.TourNumber, 'PoliTours')
        FROM PoliTours.PlaceOfInterestTour AS P, PoliTours.Landmark AS L
        WHERE P.PlaceOfInterestName=L.PlaceOfInterestName

        UNION

        SELECT POIName, KeyGenTour(TourName, 'UniTours')
        FROM UniTours.PointOfInterestTour
)


CREATE VIEW UniPoliTours.PointOfInterest (POIName, Description, Type, HistoricalPeriod,
        PlaceOfInterest, CityName) AS (
        SELECT L.Name, L.Description, L.Type, null, P.Name, P.CityName
        FROM PoliTours.Landmark AS L, PoliTours.PlaceOfInterest AS P
        WHERE L.PlaceOfInterestName=P.Name

        UNION
```

```
        SELECT POIName, Description, Type, HistoricalPeriod, null, CityName
        FROM UniTours.PointOfInterest
)
```

3c) Query rewriting

```
SELECT left(Guide, position('%' in Guide)-1), right(Guide, length(Guide)-position('%' in Guide)), null
FROM PoliTours.Tour AS T, PoliTours.PlaceOfInterestTour AS PT, PoliTours.Landmark AS L
WHERE T.CityName=PT.CityName AND T.Number=PT.TourNumber AND
        PT.PlaceOfInterestName=L.PlaceOfInterestName AND L.Type='Theater'

UNION

SELECT G.Firstname, G.Lastname, G.BirthDate
FROM Guide AS G, Tour AS T, PointOfInterestTour AS PT, PointOfInterest AS P
WHERE G.SSN=T.SSNGuide AND T.TourName=PT.TourName AND PT.POIName=P.POIName AND
        P.Type='Theater'
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – September 6th, 2016

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

Polimusic.com is a website allowing its customers to listen to songs by paying a fee for each listening. Each song has a specific price. 60% of the income derived by a listening is retained by Polimusic.com, while the remaining 40% is paid to the artists featuring the song. A song may feature more than one artist, and in this case the money is divided among the featuring artists on the basis of a weight.
The management of *Polimusic.com* has now hired you to design a data warehouse to analyze the listenings.

The following is the schema of the *Polimusic.com* operational database:

ARTIST (ArtistCode, Name, RecordLabel) *// An artist may be either a single person or a band.*
SONG (SongCode, Title, Genre, Language, DurationInSeconds, Price)
SONGARTIST (SongCode, ArtistCode, WeightForProfits) *// The weights assigned to the artists featuring a certain song sum to 1.*
USER (UserCode, Surname, GivenName, BirthDate, HomeCountry, RegistrationDate)
COUNTRY (CountryName, Continent)
LISTENING (UserCode, SongCode, StartDate, StartTime, HighResolution) *// HighResolution is a boolean attribute which is true if the listening was performed using high-resolution audio.*

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

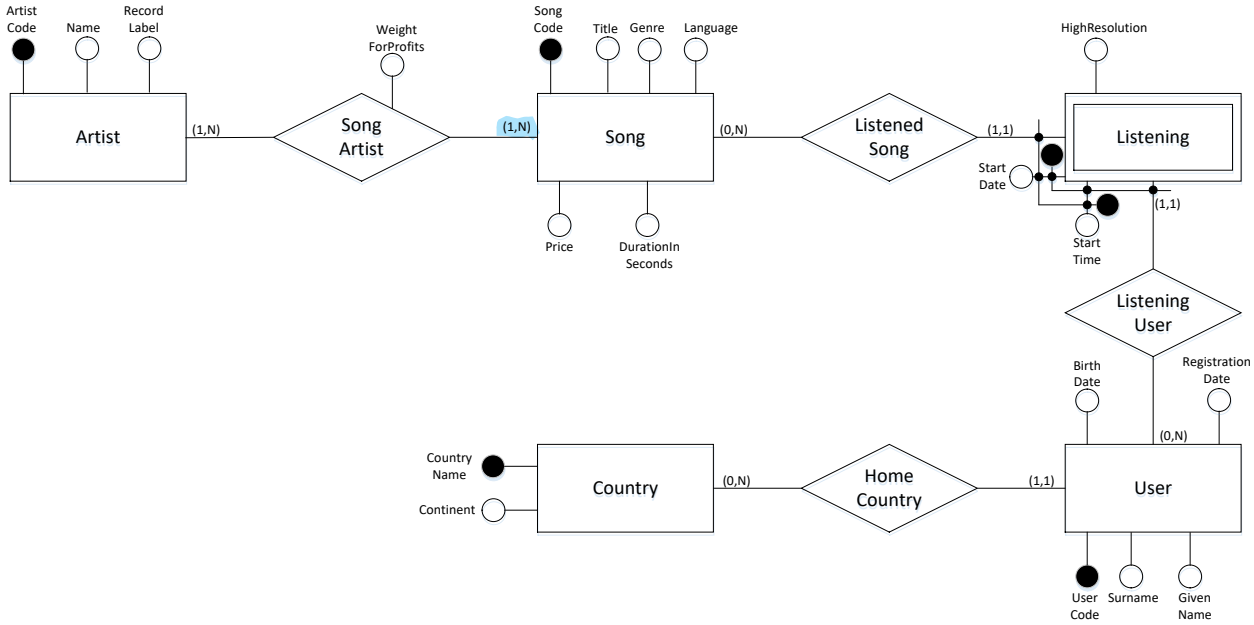
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2 points) Identify the measure(s) and produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (1.5 points) Considering only the listenings performed on Sundays by users born in 1990, find the total income realized by the website (therefore excluding the quota for the artists) grouped by month, user (specify code, surname and given name), type of listening (high resolution or not) and genre of the song.
    b. (1.5 points) For each artist, find the name and the number of listenings realized on 3rd July 2014.
    c. (2.5 points) Find the record label whose artists obtained the greatest total income in 2015.
    d. (2.5 points) Select for each continent the code and the title of the song with the greatest total listening time (note: all the listenings to a song can be considered as complete).

SOLUTION

1. Reverse engineering

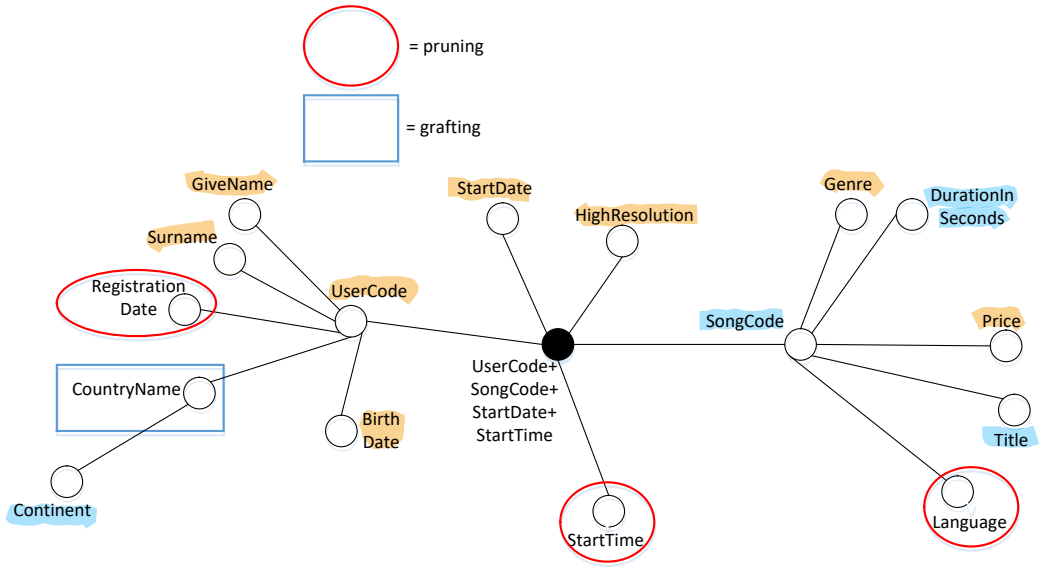


2. Conceptual design

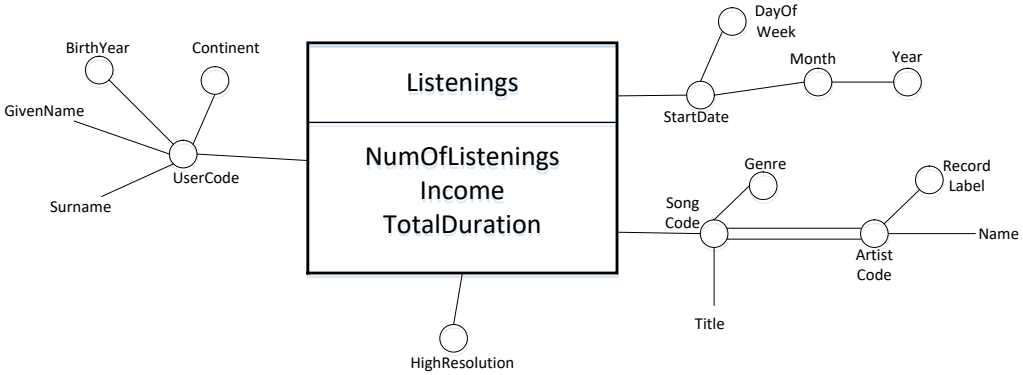
RELEVANT FOR QUERIES a d

Fact: Listenings (Listening entity)

2a) Attribute tree



2b) Fact schema



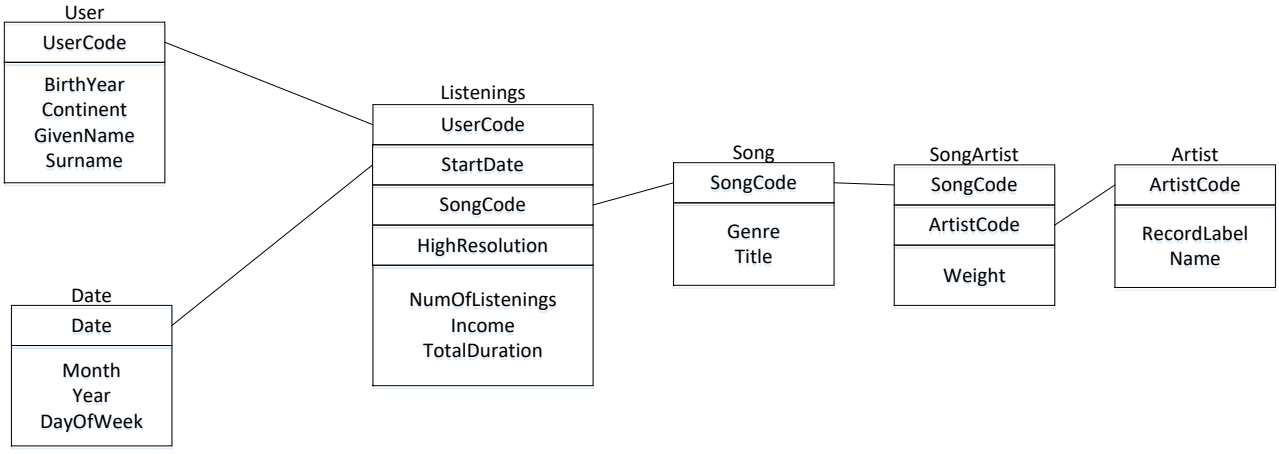
2c) Glossary

NumOfListenings
SELECT L.UserCode, L.StartDate, L.SongCode, L.HighResolution, **COUNT**(*)
FROM Listening **AS** L
GROUP BY L.UserCode, L.StartDate, L.SongCode, L.HighResolution

Income
SELECT L.UserCode, L.StartDate, L.SongCode, L.HighResolution, **SUM**(S.Price)
FROM Listening **AS** L, Song **AS** S
WHERE L.SongCode=S.SongCode
GROUP BY L.UserCode, L.StartDate, L.SongCode, L.HighResolution

TotalDuration
SELECT L.UserCode, L.StartDate, L.SongCode, L.HighResolution, **SUM**(S.DurationInSeconds)
FROM Listening **AS** L, Song **AS** S
WHERE L.SongCode=S.SongCode
GROUP BY L.UserCode, L.StartDate, L.SongCode, L.HighResolution

3. Logical design



4. Query answering

4a) Considering only the listenings performed on Sundays by users born in 1990, find the total income realized by the website (therefore excluding the quota for the artists) grouped by month, user (specify code, surname and given name), type of listening (high resolution or not) and genre of the song.

```
SELECT D.Month, U.UserCode, U.Surname, U.GivenName, L.HighResolution, S.Genre,
        SUM(0.6*L.Income)
FROM Listenings AS L, Date AS D, User AS U, Song AS S
WHERE L.StartDate=D.Date AND L.UserCode=U.UserCode AND L.SongCode=S.SongCode AND
        D.DayOfWeek='Sunday' AND U.BirthYear='1990'
GROUP BY D.Month, U.UserCode, U.Surname, U.GivenName, L.HighResolution, S.Genre
```

4b) For each artist, find the name and the number of listenings realized on 3rd July 2014.

```
SELECT A.ArtistCode, A.Name, SUM(L.NumOfListenings)
FROM Listening AS L, SongArtist AS SA, Artist AS A
WHERE L.SongCode=SA.SongCode AND SA.ArtistCode=A.ArtistCode AND L.StartDate='2014-07-03'
GROUP BY A.ArtistCode, A.Name
```

4c) Find the record label(s) whose artists obtained the greatest total income in 2015.

```
CREATE VIEW LabelIncome (RecordLabel, Income) AS (
        SELECT A.RecordLabel, SUM(0.4*SA.Weight*L.Income)
        FROM Listenings AS L, Date AS D, SongArtist AS SA, Artist AS A
        WHERE L.StartDate=D.Date AND L.SongCode=SA.SongCode AND SA.ArtistCode=A.ArtistCode AND
                D.Year='2015'
        GROUP BY A.RecordLabel
)

SELECT RecordLabel
FROM LabelIncome
WHERE Income = (
                        SELECT MAX(Income)
                        FROM LabelIncome
                )
```

4d) Select for each continent the code and the title of the song(s) with the greatest total listening time (note: all the listenings to a song can be considered as complete).

```
CREATE VIEW ContinentSongDuration (Continent, SongCode, Title, TotalDuration) AS (
        SELECT U.Continent, S.SongCode, S.Title, SUM(L.TotalDuration)
        FROM Listenings AS L, User AS U, Song AS S
        WHERE L.UserCode=U.UserCode AND L.SongCode=S.SongCode
        GROUP BY U.Continent, S.SongCode, S.Title
)

SELECT C.Continent, C.SongCode, C.Title
FROM ContinentSongDuration AS C
WHERE C.TotalDuration = (
                                SELECT MAX(C2.TotalDuration)
                                FROM ContinentSongDuration AS C2
                                WHERE C.Continent=C2.Continent
                        )
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – September 21st, 2016

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliRestaurants is a restaurant chain selling home-delivery meals in Italy through its website. Each order is associated with just one dish, and the customers may choose the restaurant to which the delivery is requested. Each dish has a full price, but a discount may be applied when the customer completes the order; the restaurant chain proposes promotions varying continuously, therefore each order is associated with its own discount.
The management of *PoliRestaurants* has now hired you to design a data warehouse to analyze the orders.

The following is the schema of the *PoliRestaurants* operational database:

RESTAURANT (RestaurantName, Address, CityName, SupervisorSSN, NumberOfEmployees)
SUPERVISOR (SupervisorSSN, Surname, GivenName)
CITY (CityName, Region)
DISHSUBCATEGORY (Subcategory, Category) *// The available categories are 'Food' and 'Beverage'. Possible subcategories are 'AlcoholicDrink', 'Meat', 'Fish', 'Pasta', …*
DISH (DishName, Description, Subcategory, Price)
CUSTOMER (CustomerId, Surname, GivenName, BirthYear)
ORDER (CustomerId, Date, Time, DishName, RestaurantName, Discount, DeliveryAddress, DeliveryCity) *// The discount is a percentage expressed as a number between 0 and 1.*

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

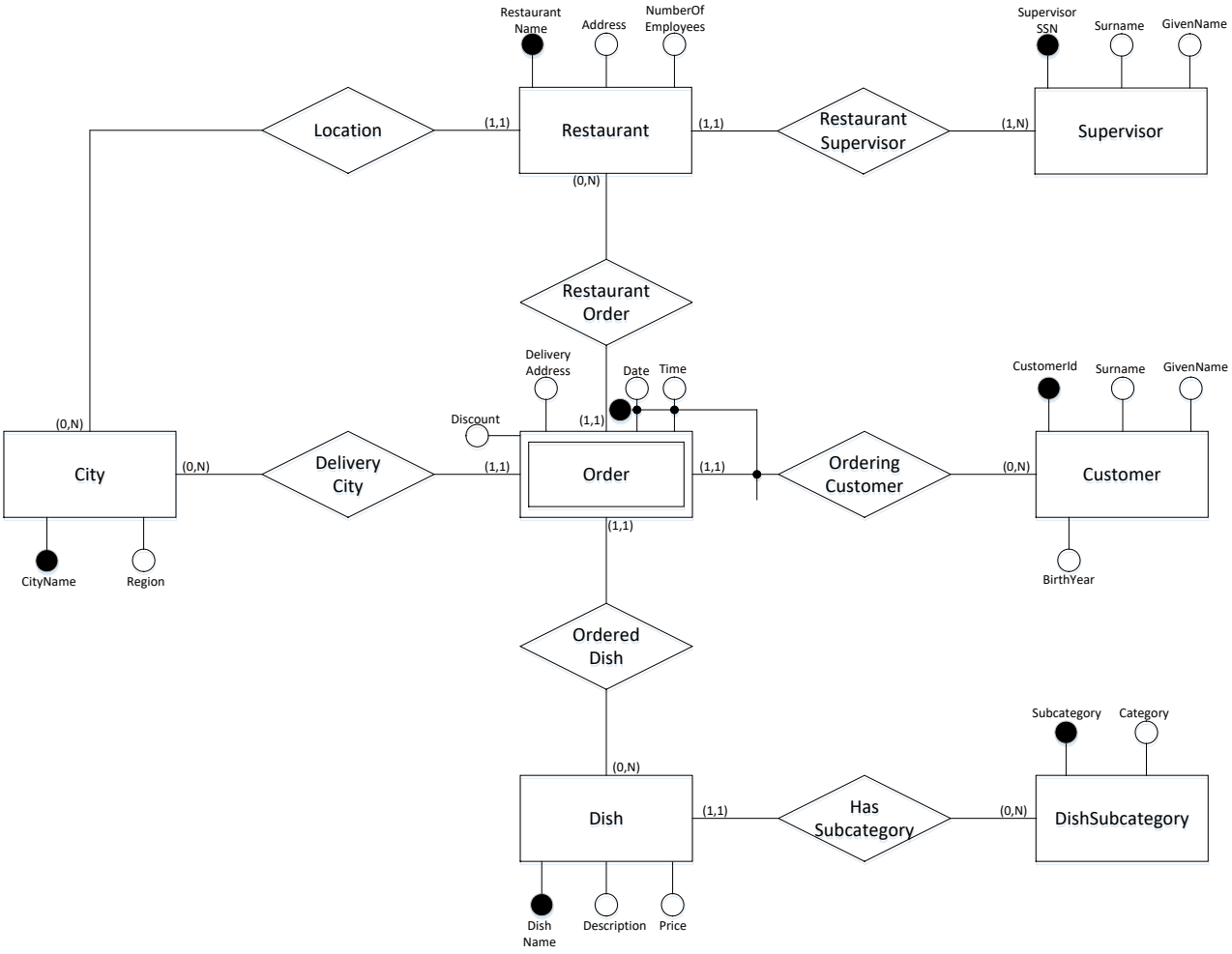
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2 points) Identify the measure(s) and produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (2 points) Considering only the orders with category ‘Food’, find the average discount by customer birth year, dish subcategory and restaurant city.
    b. (2 points) Considering only the restaurants located in Lombardy, find the total number of orders by restaurant, delivery city and date. Include in the answer also the aggregations computed using only one or two of the three attributes.
    c. (2 points) Find SSN, surname and given name of the supervisors who in 2015 have increased their income at least of 30% with respect to 2014.
    d. (2 points) For each delivery region, find the dish(es) with the greatest number of orders.

SOLUTION

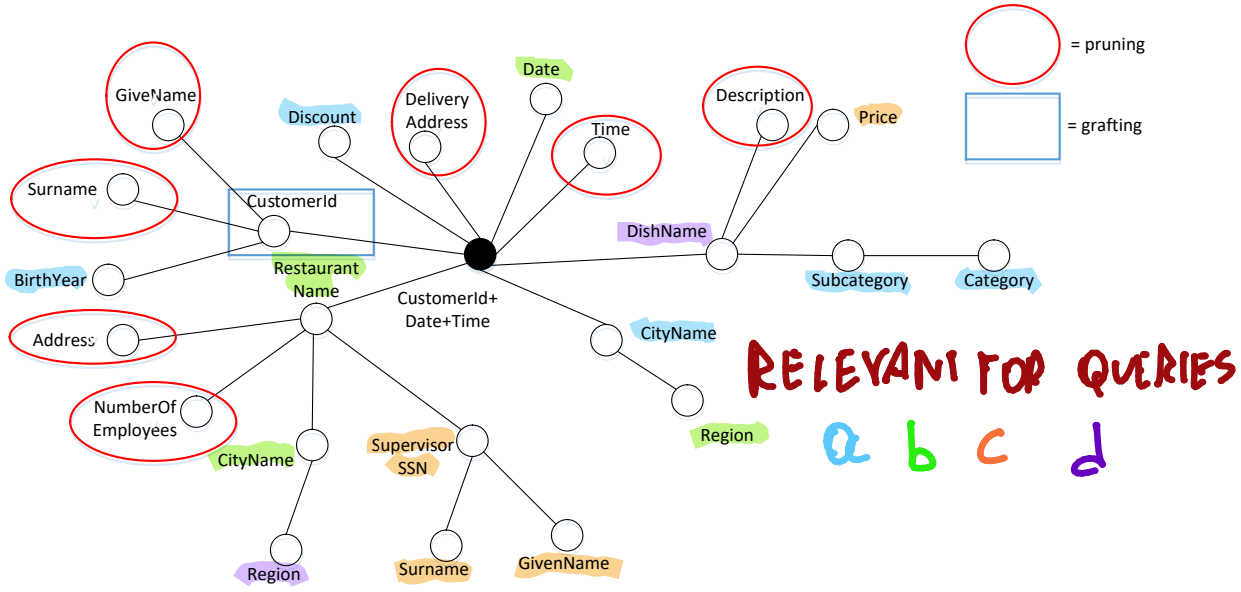
1. Reverse engineering



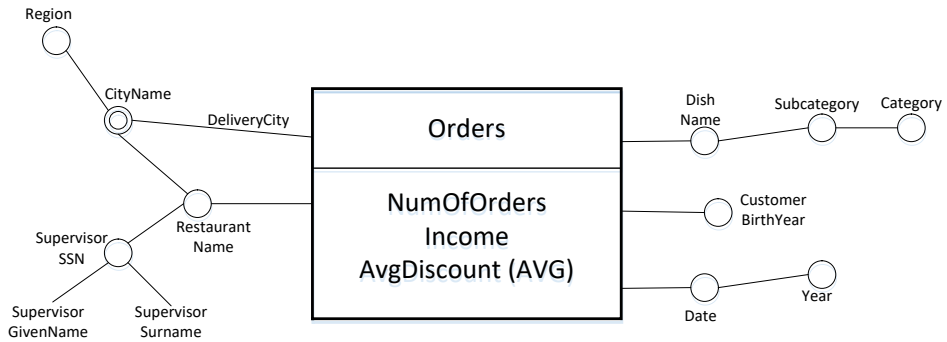
2. Conceptual design

Fact: Orders (Order entity)

2a) Attribute tree



2b) Fact schema



2c) Glossary

<u>NumOfOrders</u>

SELECT O.Date, C.BirthYear, O.DishName, O.RestaurantName, O.DeliveryCity, **COUNT**(*)
FROM Order **AS** O, Customer **AS** C
WHERE O.CustomerId=C.CustomerId
GROUP BY O.Date, C.BirthYear, O.DishName, O.RestaurantName, O.DeliveryCity

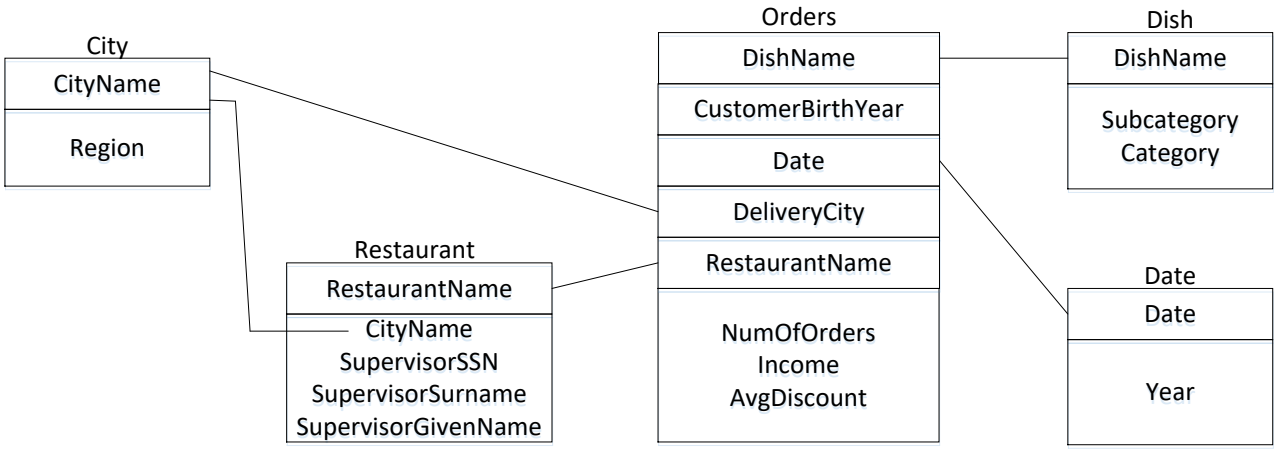
<u>Income</u>

SELECT O.Date, C.BirthYear, O.DishName, O.RestaurantName, O.DeliveryCity,
SUM(D.Price-D.Price*O.Discount)
FROM Order **AS** O, Customer **AS** C, Dish **AS** D
WHERE O.CustomerId=C.CustomerId **AND** O.DishName=D.DishName
GROUP BY O.Date, C.BirthYear, O.DishName, O.RestaurantName, O.DeliveryCity

AvgDiscount

SELECT O.Date, C.BirthYear, O.DishName, O.RestaurantName, O.DeliveryCity, **AVG**(O.Discount)
FROM Order **AS** O, Customer **AS** C
WHERE O.CustomerId=C.CustomerId
GROUP BY O.Date, C.BirthYear, O.DishName, O.RestaurantName, O.DeliveryCity

3. Logical design



4. Query answering

4a) Considering only the orders with category 'Food', find the average discount by customer birth year, dish subcategory and restaurant city.

SELECT O.CustomerBirthYear, D.Subcategory, R.CityName,
SUM(O.NumOfOrders*O.AvgDiscount)/**SUM**(O.NumOfOrders)
FROM Orders **AS** O, Dish **AS** D, Restaurant **AS** R
WHERE O.DishName=D.DishName **AND** O.RestaurantName=R.RestaurantName **AND** D.Category='Food'
GROUP BY O.CustomerBirthYear, D.Subcategory, R.CityName

4b) Considering only the restaurants located in Lombardy, find the total number of orders by restaurant, delivery city and date. Include in the answer also the aggregations computed using only one or two of the three attributes.

SELECT R.RestaurantName, O.DeliveryCity, O.Date, **SUM**(O.NumOfOrders)
FROM Orders **AS** O, Restaurant **AS** R, City **AS** C
WHERE O.RestaurantName=R.RestaurantName **AND** R.CityName=C.CityName **AND** C.Region='Lombardy'
GROUP BY R.RestaurantName, O.DeliveryCity, O.Date **WITH CUBE**

4c) Find SSN, surname and given name of the supervisors who in 2015 have increased their income at least of 30% with respect to 2014.

```
SELECT R.SupervisorSSN, R.SupervisorSurname, R.SupervisorGivenName
FROM Orders AS O, Restaurant AS R, Date AS D
WHERE O.RestaurantName=R.RestaurantName AND O.Date=D.Date AND D.Year='2015'
GROUP BY R.SupervisorSSN, R.SupervisorSurname, R.SupervisorGivenName
HAVING SUM(O.Income)>=1.3*(
                SELECT SUM(O.Income)
                FROM Orders AS O2, Restaurant AS R2, Date AS D2
                WHERE O2.RestaurantName=R2.RestaurantName AND O2.Date=D2.Date AND
                        D2.Year='2014' AND R2.SupervisorSSN=R.SupervisorSSN
                )
```

4d) For each delivery region, find the dish(es) with the greatest number of orders.

```
CREATE VIEW DeliveryRegionDishOrders (DeliveryRegion, DishName, NumOfOrders) (
        SELECT C.Region, O.DishName, SUM(O.NumOfOrders)
        FROM Orders AS O, City AS C
        WHERE O.DeliveryCity=C.CityName
        GROUP BY C.Region, O.DishName
)

SELECT D.DeliveryRegion, D.DishName
FROM DeliveryRegionDishOrders AS D
WHERE D.NumOfOrders = (
                        SELECT MAX(D2.NumOfOrders)
                        FROM DeliveryRegionDishOrders AS D2
                        WHERE D2.DeliveryRegion=D.DeliveryRegion
                        )
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – February 1st, 2017

Available Time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliRestaurants is a chain of Italian restaurants operating in Lombardy which allows customers to order meals through its website. The customer needs to register, and then can navigate to the website, where he/she can choose one of the restaurants of the chain. At this point it is possible to order dishes and beverages from the chosen restaurant, specifying for each dish or beverage the required number of items. All the PoliRestaurants restaurants offer take-away service, and in some cities also home delivery is available.

UniRestaurants, on the contrary, is a restaurant chain featuring Chinese and Japanese cuisine, which offers comparable services in Piedmont and Liguria. Analogously to PoliRestaurants, all the UniRestaurants restaurants offer the take-away service. Home delivery is available just in some of them. In addition, UniRestaurants does not permit to order beverages.

The two companies have now merged into a unique one named ***UniPoliRestaurants***. The UniPoliRestaurants ownership asks you to integrate the relational databases of the two companies into a unique relational database. You must perform the integration ensuring to lose the least possible amount of information.

The original relational schemas of the two sources are reported below.

PoliRestaurants

CITY (CityName, HomeDeliveryService) *// HomeDeliveryService is a boolean attribute that is true if the restaurants in this city allow the home delivery.*

MANAGER (ManagerSSN, Surname, GivenName, PhoneNumber)

RESTAURANT (RestaurantCode, Name, Address, CityName, ManagerSSN) *// A manager may supervise multiple restaurants.*

CUSTOMER (CustomerSSN, Surname, GivenName, BirthDate)

ORDER (OrderCode, Timestamp, OrderType, ServiceDate, ServiceTime, DeliveryAddress*, RestaurantCode, CustomerSSN) *// OrderType is either 'TakeAway' or 'HomeDelivery'. The DeliveryAddress attribute is null for for take-away orders.*

DISH (DishCode, Name, Description, Cost, IsVegan, WeightGrams) *// IsVegan is a boolean attribute that is true if this dish is vegan. WeightGrams represents the weight of the dish expressed in grams.*

BEVERAGE (BeverageCode, Name, Description, Price, IsAlcoholic) // *IsAlcoholic is a boolean attribute that is true if this beverage is alcoholic.*

ORDEREDDISH (OrderCode, DishCode, Quantity)

ORDEREDBEVERAGE (OrderCode, BeverageCode, Quantity)

UniRestaurants

CITY (CityName, Region) *// Region is either 'Piedmont' or 'Liguria'.*

RESTAURANT (RestaurantCode, Name, Address, HomeDeliveryService, ManagerPhoneContact, CityName) *// HomeDeliveryService is a boolean attribute that is true if the restaurant allows the home delivery.*

CUSTOMER (CustomerRegistrationNr, CustomerSSN, LastName, FirstName, PhoneNumber)

COURSE (CourseCode, Name, Description, CuisineType, Price, WeightKilos) *// CuisineType is either 'Chinese' or 'Japanese'. WeightKilos represents the weight of the course expressed in kilograms.*

INGREDIENT (IngredientName, Category, Supplier) *// The ingredient category may be vegetable, fruit, dairy, …*

COURSEINGREDIENT (CourseCode, IngredientName, Quantity)

BOOKING (Timestamp, CustomerRegistrationNr, BookingType, ServiceDate, ServiceTime, DeliveryAddress*, CourseCode, RestaurantCode, Quantity) *// BookingType is either 'TakeAway' or 'HomeDelivery'. The DeliveryAddress attribute is null for take-away bookings.*

Notes:

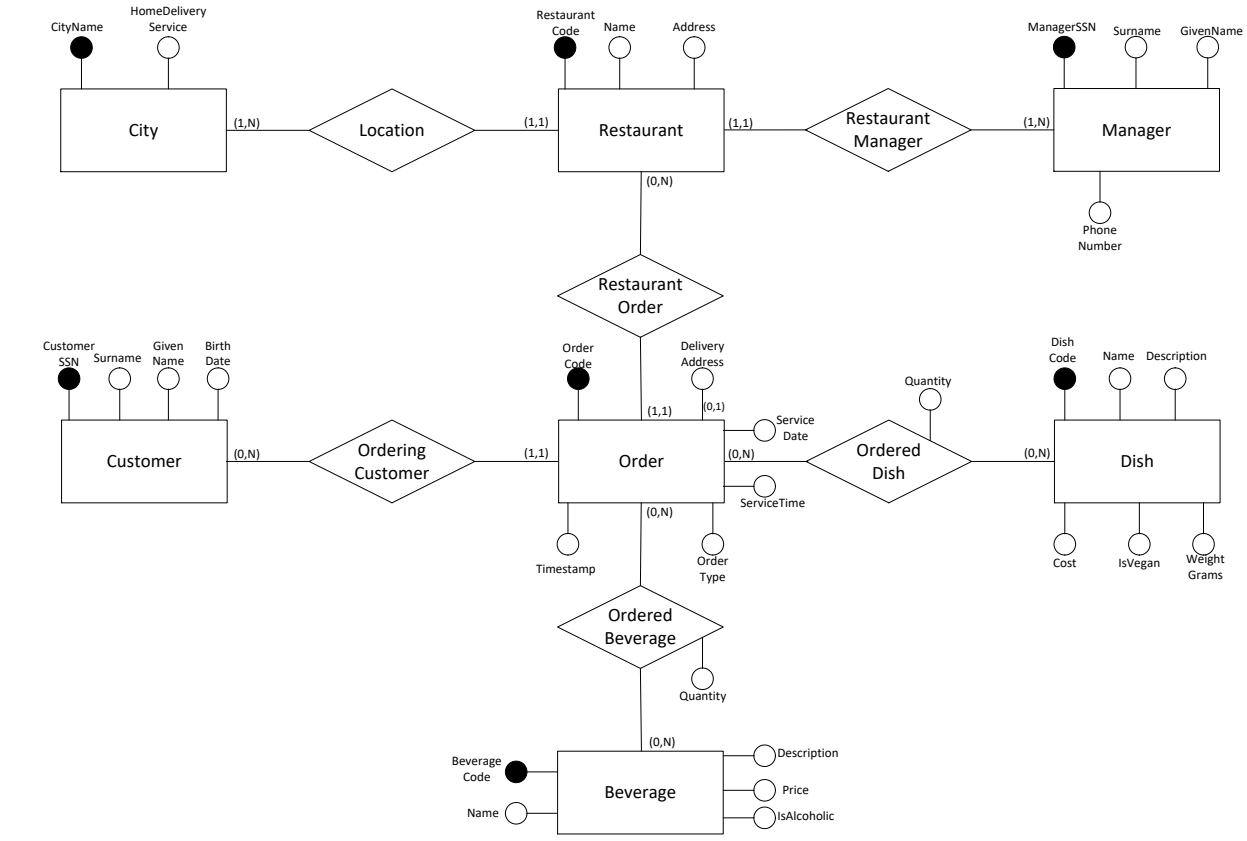
- Since PoliRestaurants and UniRestaurants operate in different geographic areas, you can assume that the customers in the two data sources are disjoint.
- Since PoliRestaurants and UniRestaurants deal with different kinds of cuisine, you can assume that the sets of dishes included in the two data sources are disjoint.

1. **Source schema reverse engineering**. Provide, for each input data source, the reverse engineering from the logical schema to the conceptual model (ER graph). (5 points)
2. **Schema integration**. Design an integrated global conceptual schema (ER graph) for *UniPoliRestaurants* capturing all the data coming from both *PoliRestaurants* and *UniRestaurants*, and provide the corresponding global logical schema. In more detail, follow these steps:
    a. *Related concept identification and conflict analysis and resolution*. Write a table as shown in the exercise sessions, using the following columns: "PoliRestaurants concept", "UniRestaurants concept", "Conflict", "Solution". (3.5 points)
    b. *Integrated conceptual schema* (ER graph). (3.5 points)
    c. *Conceptual to logical translation of the integrated schema*. (2 points)
3. **Query answering and mapping definition**. Consider the query Q "Find timestamp and customer SSN of the take-away orders to restaurants located in Milan or in Turin, which have involved at least one dish with weight greater than 200 grams and price greater than 20 euros".
    a. *Query formulation*. Consider query Q posed on the logical schema of *UniPoliRestaurants* and write it in SQL. (1.5 points)
    b. *Mapping definition*. Write the GAV mappings between the schema of *UniPoliRestaurants* and the two sources using SQL. Write the mappings only for the tables used to answer query Q**.** (4 points)
    c. *Query rewriting*. Show the rewriting of Q on the two data sources using SQL. (2.5 points)

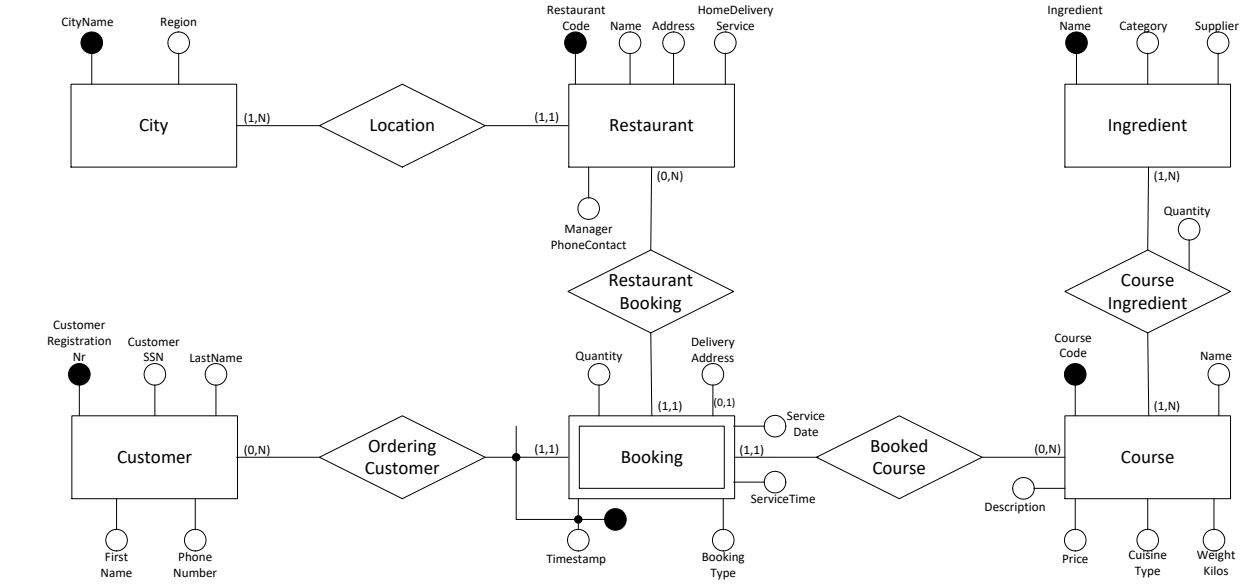
SOLUTION

1. Source schema reverse engineering

PoliRestaurants



UniRestaurants

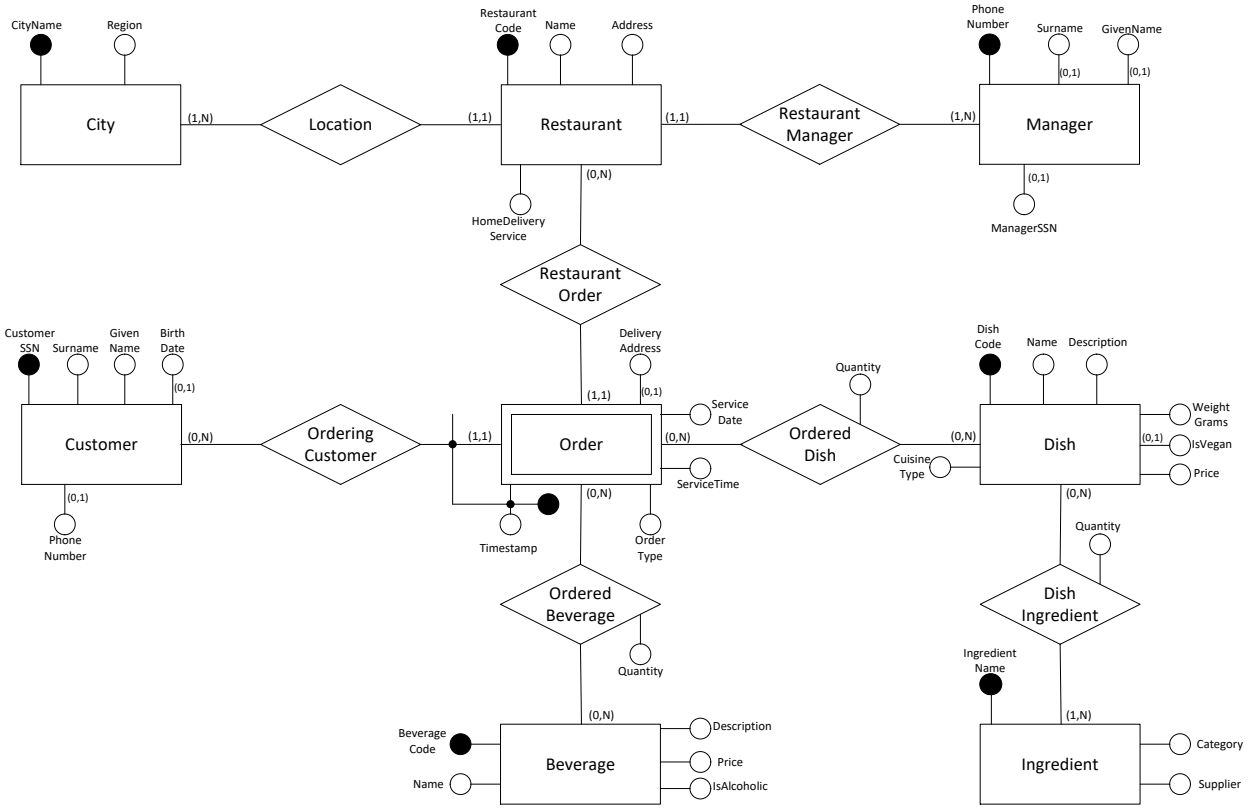


2. Schema integration

2a) Related concept identification + conflict analysis and resolution

PoliRestaurants	UniRestaurants	Conflict	Solution
City	City	Structure conflicts	
		- HomeDeliveryService is an attribute of City → HomeDeliveryService is an attribute of Restaurant	HomeDeliveryService is an attribute of Restaurant
Restaurant	Restaurant	Structure conflicts	
		- Manager is an entity → The manager (phone contact) is an attribute of Restaurant	Manager is an entity
Customer	Customer	Name conflicts	
		- Surname → LastName	Surname
		- GivenName → FirstName	GivenName
		Key conflict	
		- CustomerSSN → CustomerRegistrationNr	CustomerSSN
Dish	Course	Name conflicts	
		- Entity name	Dish
		- DishCode → CourseCode	DishCode
		- Cost → Price	Price
		Data semantics conflicts	
		- WeightInGrams → WeightInKilos	WeightInGrams
Order	Booking	Name conflicts	
		- Entity name	Order
		- OrderType → BookingType	OrderType
		Key conflict	
		- OrderCode →Timestamp + key of the Customer	Timestamp + key of the Customer
		Cardinality conflicts	
		- Each order may be associated with multiple dishes → Each booking is associated with just one course	Each order may be associated with multiple dishes

2b) Global conceptual schema



2c) Conceptual to logical translation

CITY (<u>CityName</u>, Region)
MANAGER (<u>PhoneNumber</u>, ManagerSSN*, Surname*, GivenName*)
RESTAURANT (<u>RestaurantCode</u>, Name, Address, HomeDeliveryService, CityName, ManagerPhoneNumber)
CUSTOMER (<u>CustomerSSN</u>, Surname, GivenName, BirthDate*, PhoneNumber*)
ORDER (<u>CustomerSSN</u>, <u>Timestamp</u>, OrderType, ServiceDate, ServiceTime, DeliveryAddress*, RestaurantCode)
DISH (<u>DishCode</u>, Name, Description, Price, IsVegan*, WeightGrams, CuisineType)
BEVERAGE (<u>BeverageCode</u>, Name, Description, Price, IsAlcoholic)
ORDEREDDISH (<u>CustomerSSN</u>, <u>Timestamp</u>, <u>DishCode</u>, Quantity)
ORDEREDBEVERAGE (<u>CustomerSSN</u>, <u>Timestamp</u>, <u>BeverageCode</u>, Quantity)
INGREDIENT (<u>IngredientName</u>, Category, Supplier)
DISHINGREDIENT (<u>DishCode</u>, <u>IngredientName</u>, Quantity)

3. Query answering and mapping definition

3a) Query formulation

Find timestamp and customer SSN of the take-away orders to restaurants located in Milan or in Turin, which have involved at least one dish with weight greater than 200 grams and price greater than 20 euros.

SELECT DISTINCT O.CustomerSSN, O.Timestamp
FROM Order **AS** O, Restaurant **AS** R, OrderedDish **AS** OD, Dish **AS** D
WHERE O.RestaurantCode=R.RestaurantCode **AND** O.OrderCode=OD.OrderCode **AND**
OD.DishCode=D.DishCode **AND** O.OrderType='TakeAway' **AND** (R.CityName='Milan' **OR**
R.CityName='Turin') **AND** D.WeightGrams>200 **AND** D.Price>20

3b) GAV mapping definition

The KeyGen(_, _) functions generate univocal identifiers.

CREATE VIEW UniPoliRestaurants.Restaurant (RestaurantCode, Name, Address, HomeDeliveryService, CityName, ManagerPhoneNumber) **AS** (
SELECT KeyGenRestaurant(R.RestaurantCode, 'PoliRestaurants'), R.Name, R.Address,
C.HomeDeliveryService, R.CityName, M.ManagerPhoneNumber
FROM PoliRestaurants.Restaurant **AS** R, PoliRestaurants.City **AS** C, PoliRestaurants.Manager **AS** M
WHERE R.CityName=C.CityName **AND** R.ManagerSSN=M.ManagerSSN

UNION

SELECT KeyGenRestaurants(RestaurantCode, 'UniRestaurants'), Name, Address,
HomeDeliveryService, CityName, ManagerPhoneContact
FROM UniRestaurants.Restaurant
)

CREATE VIEW UniPoliRestaurants.Dish (DishCode, Name, Description, Price, IsVegan, WeightGrams, CuisineType) **AS** (
SELECT KeyGenDish(DishCode, 'PoliRestaurants'), Name, Description, Cost, IsVegan, WeightGrams,
'Italian'
FROM PoliRestaurants.Dish

UNION

SELECT KeyGenDish(CourseCode, 'UniRestaurants'), Name, Description, Price, **null**,
WeightKilos*1000, CuisineType
FROM UniRestaurants.Course
)

```
CREATE VIEW UniPoliRestaurants.Order (CustomerSSN, Timestamp, OrderType, ServiceDate, ServiceTime,
DeliveryAddress, RestaurantCode) AS (
        SELECT CustomerSSN, Timestamp, OrderType, ServiceDate, ServiceTime, DeliveryAddress,
                KeyGenRestaurant(RestaurantCode, 'PoliRestaurants')
        FROM PoliRestaurants.Order

        UNION

        SELECT C.CustomerSSN, B.Timestamp, B.BookingType, B.ServiceDate, B.ServiceTime,
                B.DeliveryAddress, KeyGenRestaurant(B.RestaurantCode, 'UniRestaurants')
        FROM UniRestaurants.Booking AS B, UniRestaurants.Customer AS C
        WHERE B.CustomerRegistrationNr=C.CustomerRegistrationNr
)


CREATE VIEW UniPoliRoads.OrderedDish (CustomerSSN, Timestamp, DishCode, Quantity) AS (
        SELECT O.CustomerSSN, O.Timestamp, KeyGenDish(OD.DishCode, 'PoliRestaurants'), OD.Quantity
        FROM PoliRestaurants.OrderedDish AS OD, PoliRestaurants.Order AS O
        WHERE OD.OrderCode=O.OrderCode

        UNION

        SELECT C.CustomerSSN, B.Timestamp, KeyGenDish(B.CourseCode, 'UniRestaurants'), B.Quantity
        FROM UniRestaurants.Booking AS B, UniRestaurants.Customer AS C
        WHERE B.CustomerRegistrationNr=C.CustomerRegistrationNr
)
```

3c) Query rewriting

Find timestamp and customer SSN of the take-away orders to restaurants located in Milan or in Turin, which have involved at least one dish with weight greater than 200 grams and price greater than 20 euros.

```
SELECT O.CustomerSSN, O.Timestamp
FROM Order AS O, Restaurant AS R, OrderedDish AS OD, Dish AS D
WHERE O.RestaurantCode=R.RestaurantCode AND O.OrderCode=OD.OrderCode AND
        OD.DishCode=D.DishCode AND O.OrderType='TakeAway' AND (R.CityName='Milan' OR
        R.CityName='Turin') AND D.WeightGrams>200 AND D.Cost>20

UNION

SELECT Cu.CustomerSSN, B.Timestamp
FROM Booking AS B, Customer AS Cu, Restaurant AS R, Course AS Co
WHERE B.CustomerRegistrationNr=Cu.CustomerRegistrationNr AND B.RestaurantCode=R.RestaurantCode
        AND B.CourseCode=Co.CourseCode AND B.BookingType='TakeAway' AND (R.CityName='Milan' OR
        R.CityName='Turin') AND D.WeightKilos>(200/1000) AND D.Price>20
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – February 15th, 2017

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliAirways is an airline operating *exclusively direct flights* across Europe. In order to manage the reimbursements for the lost luggage items, PoliAirways acquires the luggage claim data associated with its own passengers from the airports, and inserts them into an operational database; this database is then completed with the amounts that the airline had to pay to the passengers as reimbursements and with the information related to flights and tickets. The reimbursement includes both the compensation for the value of the luggage items irretrievably lost and the compensation for the expenses faced by the passengers due to the delay in the luggage delivery (e.g., to buy clothes).

The management of PoliAirways has now hired you to design a data warehouse to analyze the claims.

The following is the schema of the operational database used by PoliAirways:

CITY (CityName, Country)
AIRPORT (AirportCode, AirportName, CityName)
FLIGHT (FlightCode, DepartTime, ArrivTime, DepartAirportCode, ArrivAirportCode)
TICKET (TicketNr, FlightCode, FlightDate) *// The ticket refers to a specific flight in a specific date.*
LUGGAGEITEM (ItemCode, Type, Weight, TicketNr) *// The ItemCode is unique worldwide, and allows for luggage tracking in any airport. It is always printed on a sticker tag attached to the luggage. The luggage type can be "Trolley", "Suitcase", "Bag", "Other". Attribute Weight has discrete values: Light (0-15 kg), Medium (16-25 kg), Large (26-40kg), and ExtraLarge (>40kg).*
CLAIM (ItemCode, ClaimDate, ClaimIssuer, IssuerPhoneNr, IssuerAddress, FoundDate*, PaidReimbursement) // *The FoundDate attribute is set to null for the luggage items that are irretrievably lost.*

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

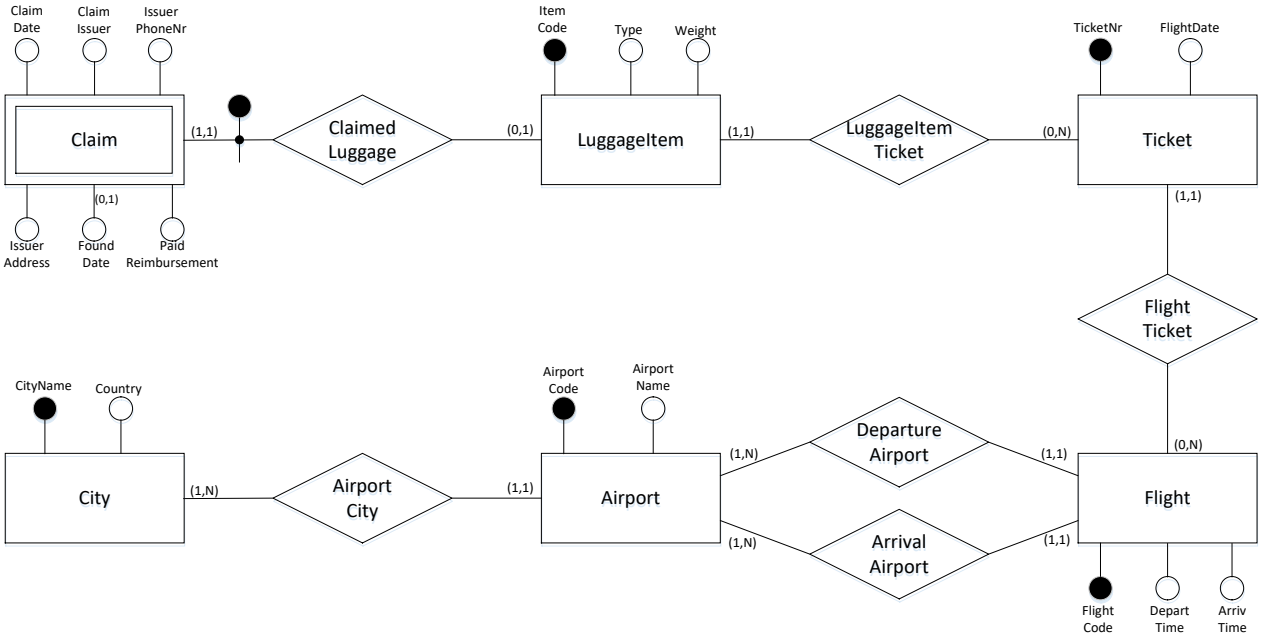
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2 points) Produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (2 points) Considering only the flights landing in Italy and only the luggage items that were finally found, compute the average number of days to retrieve the luggage item for each arrival airport (specify code and name), date, weight and type.
    b. (2 points) Considering only the flight with code YZ1234, aggregate the total reimbursement paid for the claims by month, quarter and year (include in the answer the aggregations computed only by month, only by quarter and only by year).
    c. (2 points) Compute the total number of claims for each day of week, departure city and arrival city. Include in the answer also the aggregations computed using only one and two of the three attributes.
    d. (2 points) Find, for each country, name and code of the departure airport(s) associated with the greatest number of claims.

SOLUTION

1. Reverse engineering



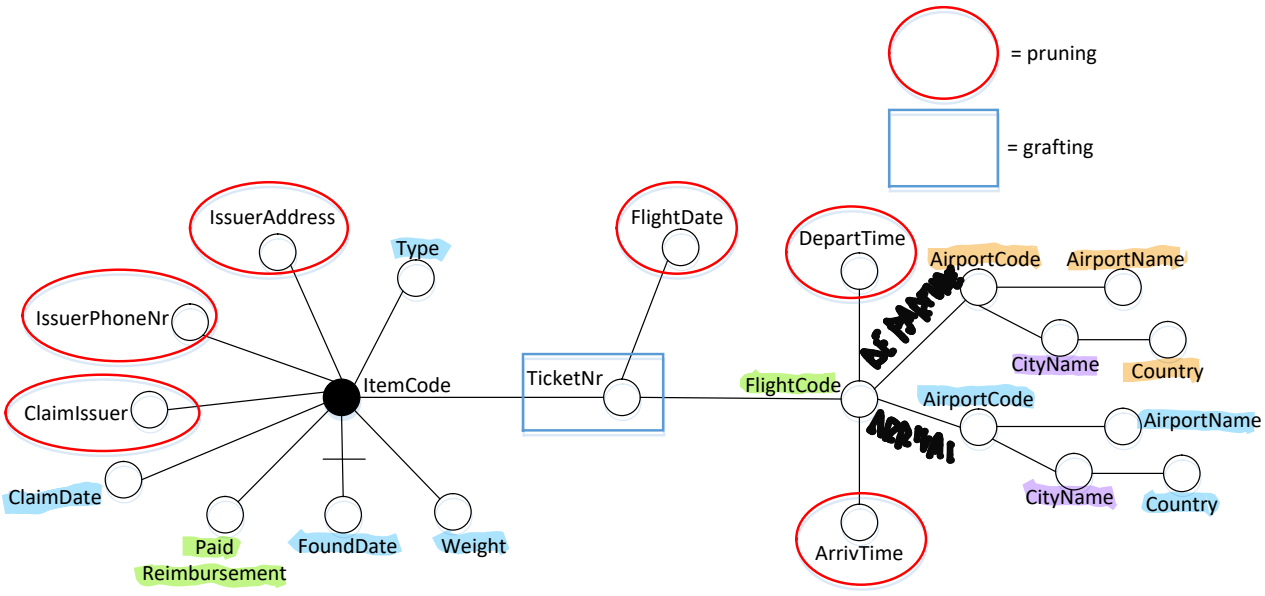
2. Conceptual design

RELEVANT FOR QUERIES

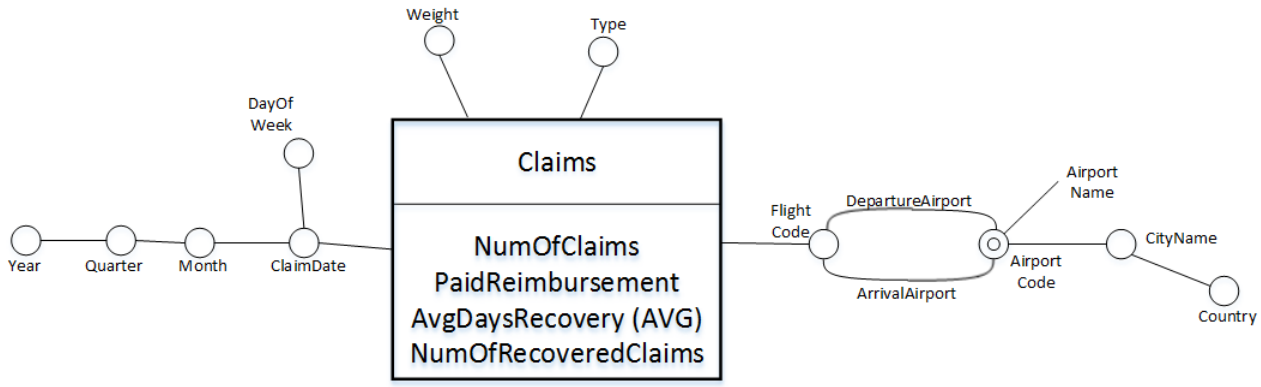
a b c d

Fact: Claims (Claim entity)

2a) Attribute tree



2b) Fact schema



NOTE: the measure NumOfRecoveredClaims is added because it is the support measure required for the aggregation of the measure AvgDaysRecovery using the AVG operator.

2c) Glossary

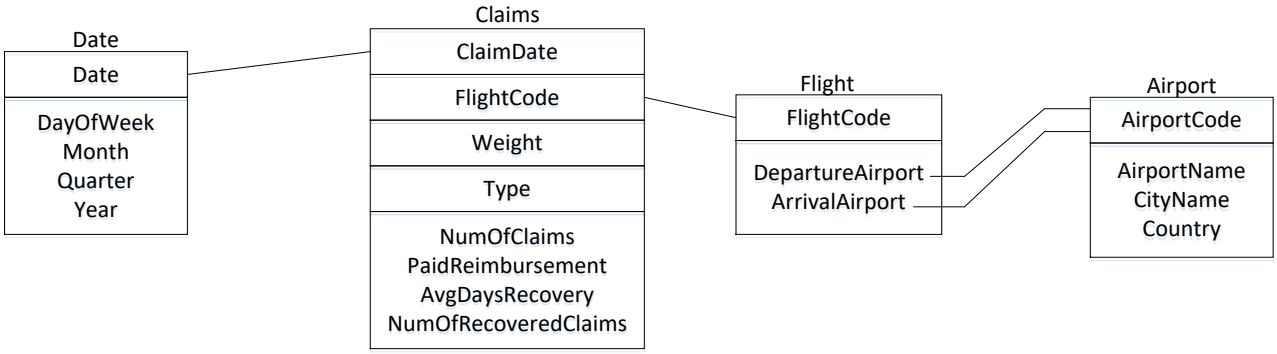
NumOfClaims
SELECT C.ClaimDate, T.FlightCode, L.Weight, L.Type, **COUNT**(*)
FROM Claim **AS** C, LuggageItem **AS** L, Ticket **AS** T
WHERE C.ItemCode=L.ItemCode **AND** L.TicketNr=T.TicketNr
GROUP BY C.ClaimDate, T.FlightCode, L.Weight, L.Type

PaidReimbursement
SELECT C.ClaimDate, T.FlightCode, L.Weight, L.Type, **SUM**(C.PaidReimbursement)
FROM Claim **AS** C, LuggageItem **AS** L, Ticket **AS** T
WHERE C.ItemCode=L.ItemCode **AND** L.TicketNr=T.TicketNr
GROUP BY C.ClaimDate, T.FlightCode, L.Weight, L.Type

AvgDaysRecovery
SELECT C.ClaimDate, T.FlightCode, L.Weight, L.Type, **AVG**(C.FoundDate-C.ClaimDate)
FROM Claim **AS** C, LuggageItem **AS** L, Ticket **AS** T
WHERE C.ItemCode=L.ItemCode **AND** L.TicketNr=T.TicketNr **AND** C.FoundDate **IS NOT NULL**
GROUP BY C.ClaimDate, T.FlightCode, L.Weight, L.Type

NumOfRecoveredClaims
SELECT C.ClaimDate, T.FlightCode, L.Weight, L.Type, **COUNT**(*)
FROM Claim **AS** C, LuggageItem **AS** L, Ticket **AS** T
WHERE C.ItemCode=L.ItemCode **AND** L.TicketNr=T.TicketNr **AND** C.FoundDate **IS NOT NULL**
GROUP BY C.ClaimDate, T.FlightCode, L.Weight, L.Type

3. Logical design



4. Query answering

4a) Considering only the flights landing in Italy and only the luggage items that were finally found, compute the average number of days to retrieve the luggage item for each arrival airport (specify code and name), date, weight and type.

```
SELECT A.AirportCode, A.AirportName, C.ClaimDate, C.Weight, C.Type,
        SUM(C.NumOfRecoveredClaims*C.AvgDaysRecovery)/SUM(C.NumOfRecoveredClaims)
FROM Claims AS C, Flight AS F, Airport AS A
WHERE C.FlightCode=F.FlightCode AND F.ArrivalAirport=A.AirportCode AND A.Country='Italy'
GROUP BY A.AirportCode, A.AirportName, C.ClaimDate, C.Weight, C.Type
```

4b) Considering only the flight with code YZ1234, aggregate the total reimbursement paid for the claims by month, quarter and year (include in the answer the aggregations computed only by month, only by quarter and only by year).

```
SELECT D.Year, D.Quarter, D.Month, SUM(C.PaidReimbursement)
FROM Claims AS C, Date AS D
WHERE C.ClaimDate=D.Date AND C.FlightCode='YZ1234'
GROUP BY D.Year, D.Quarter, D.Month WITH ROLLUP
```

4c) Compute the total number of claims for each day of week, departure city and arrival city. Include in the answer also the aggregations computed using only one and two of the three attributes.

```
SELECT D.DayOfWeek, A-D.CityName, A-A.CityName, SUM(C.NumOfClaims)
FROM Claim AS C, Date AS D, Flight AS F, Airport AS A-D, Airport AS A-A
WHERE C.ClaimDate=D.Date AND C.FlightCode=F.FlightCode AND F.DepartureAirport=A-D.AirportCode
        AND F.ArrivalAirport=A-A.AirportCode
GROUP BY D.DayOfWeek, A-D.CityName, A-A.CityName WITH CUBE
```

4d) Find for each country name and code of the departure airport(s) associated with the greatest number of claims.

```
CREATE VIEW AirportCountryNumOfClaims (AirportCode, AirportName, Country, Num) AS (
        SELECT A.AirportCode, A.AirportName, A.Country, SUM(C.NumOfClaims)
        FROM Claims AS C, Flight AS F, Airport AS A
        WHERE C.FlightCode=F.FlightCode AND F.DepartureAirport=A.AirportCode
        GROUP BY A.AirportCode, A.AirportName, A.Country
)

SELECT A.Country, A.AirportCode, A.AirportName
FROM AirportCountryNumOfClaims AS A
WHERE A.Num = (
                        SELECT MAX(A2.Num)
                        FROM AirportCountryNumOfClaims AS A2
                        WHERE A.Country=A2.Country
                )
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – July 3rd, 2017

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliMultiplex is a chain of multiplex movie theaters operating in Italy. In order to watch movies the customers must necessarily obtain a fidelity card, and each ticket issued by PoliMultiplex is associated with the customer that has purchased it and stored in the operational database of the company. The operational database contains also information about the theaters of the chain, the showings, and the programmed movies; each movie is also associated with the main actors that it features.
The management of PoliMultiplex has now hired you to design a data warehouse to analyze the issued tickets.
The following is the schema of the operational database used by PoliMultiplex:

CITY (CityName, Region)
CUSTOMER (CustomerId, Name, HomeCityName, BirthYear)
THEATER (TheaterId, TheaterName, CityName)
SHOWINGROOM (TheaterId, RoomNr, NrSeats)
MOVIE (MovieId, Title, Genre, DurationInMinutes, ProductionYear)
SHOWING (ShowingId, Date, Time, TheaterId, RoomNr, MovieId, Price) *// Each Showing may have a different price.*
ACTOR (ActorId, ActorName, Gender, BirthYear, HomeCountry) *// The attribute Gender may take the values 'M' or 'F'.*
STARRING (MovieId, ActorId)
TICKET (CustomerId, ShowingId)

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

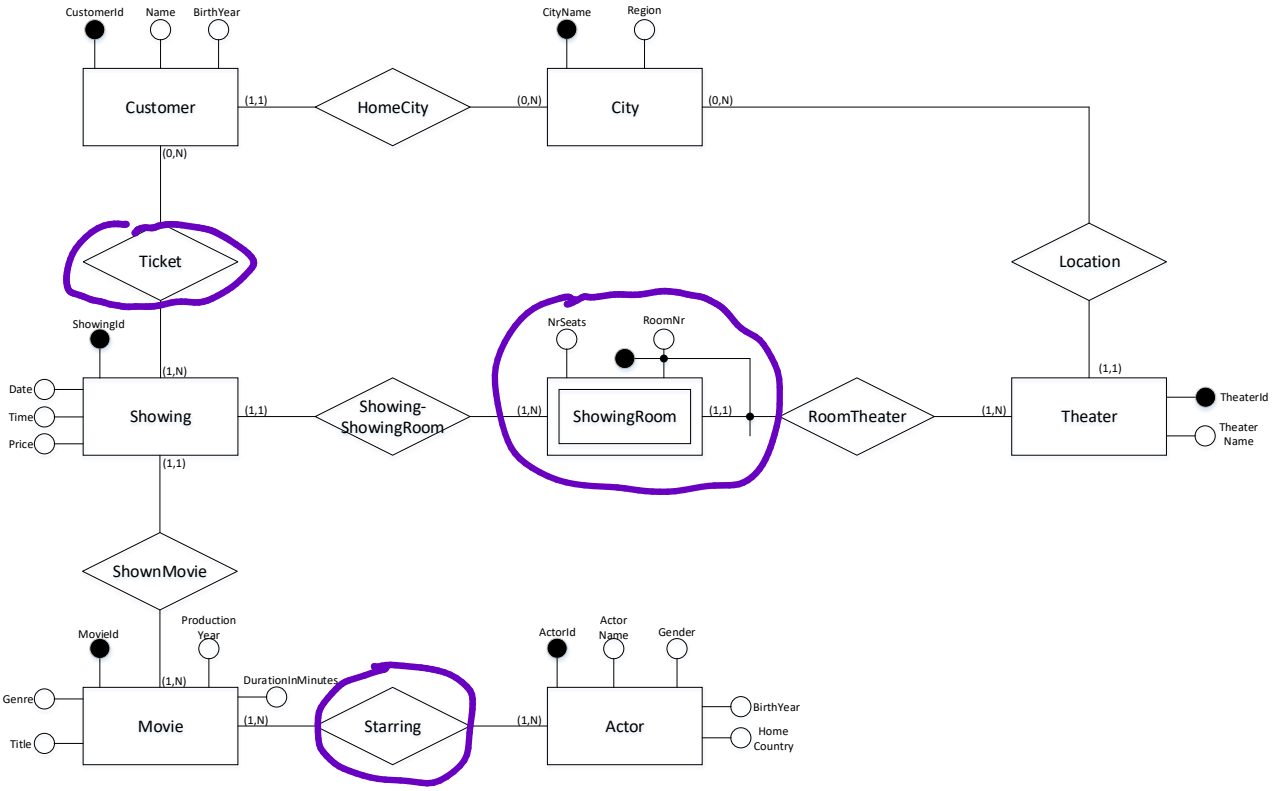
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2 points) Produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (2 points) Considering only the customers from Rome, aggregate the total income by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).
    b. (2 points) Compute the total number of tickets for each actor (specify id and name), customer home region, customer birth year and region of the theater.
    c. (2 points) Compute the total income realized by each movie (specify id and title), but considering only the movies starring at least one actress.
    d. (2 points) Considering only the theaters located in Milan, compute the genre(s) associated with the greatest number of tickets for each day of the week and theater (specify id and name).

SOLUTION

1. Reverse engineering

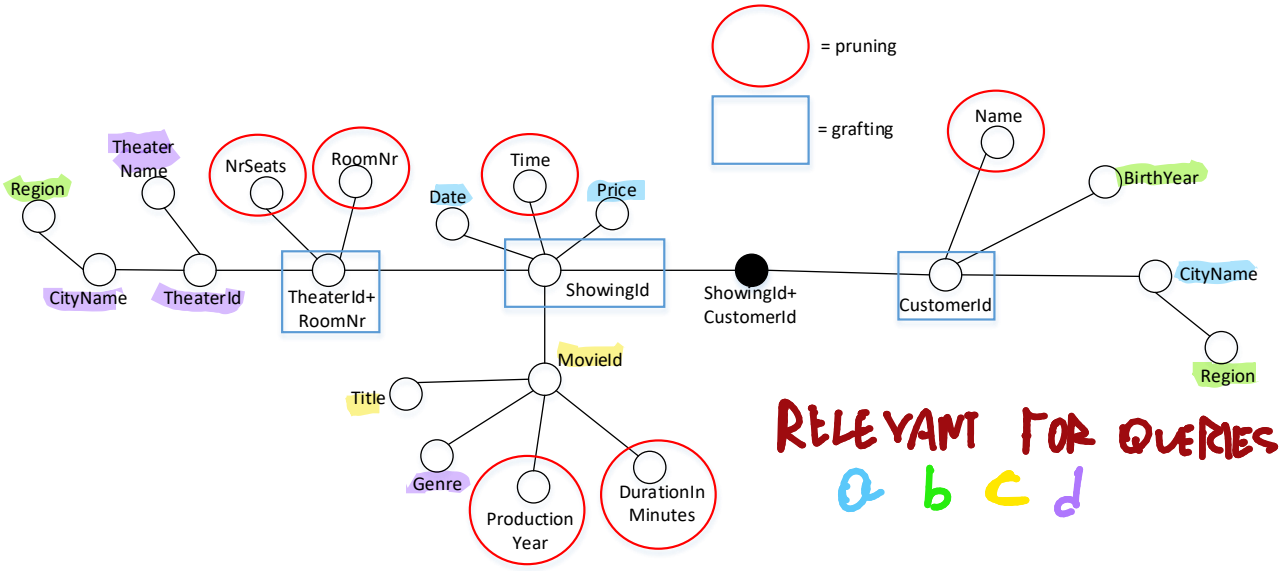


2. Conceptual design

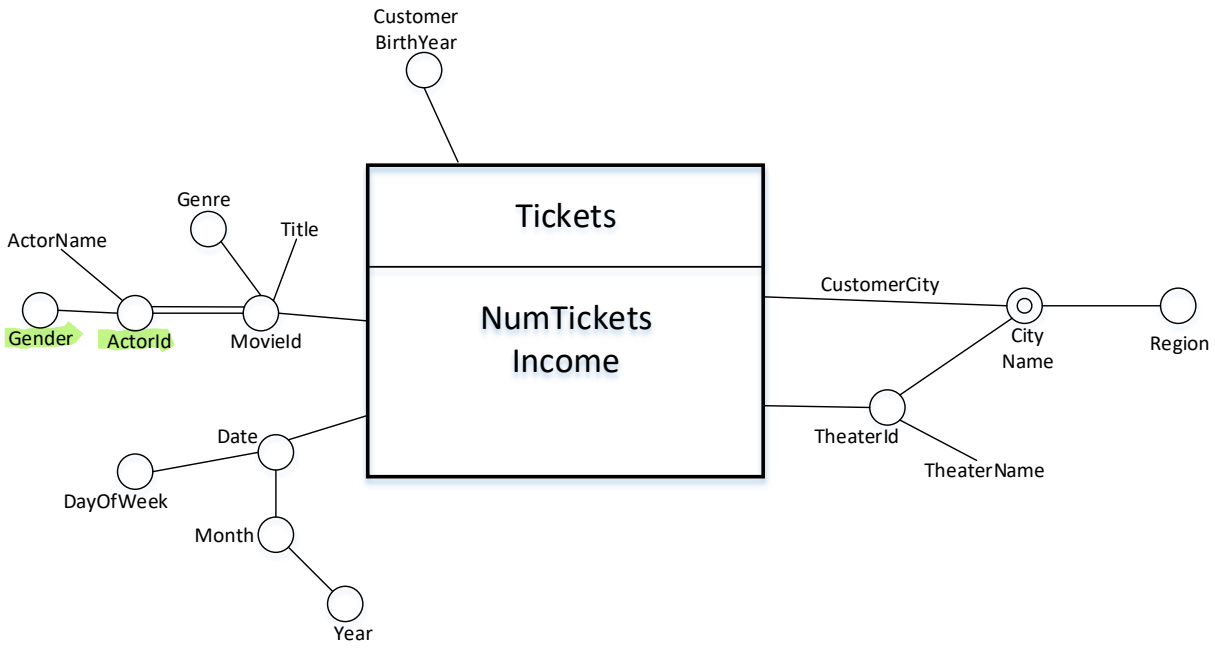
Fact: Tickets (Ticket relationship)

IN THIS EXERCISE. SEVERAL CONCEPTS ARE POTENTIAL FACTS.
A SUGGESTION IS TO READ CAREFULLY WHAT THE EXERCISE ASKS

2a) Attribute tree



2b) Fact schema



2c) Glossary

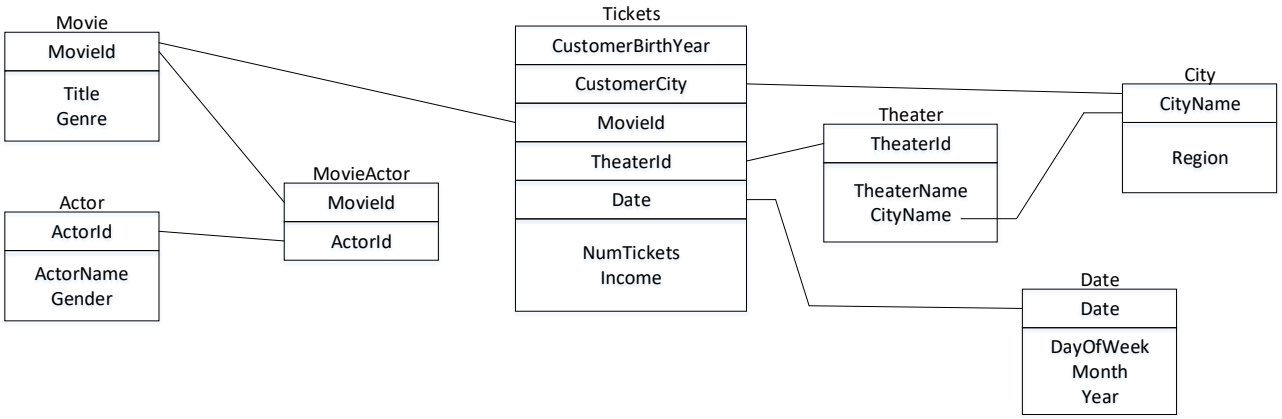
NumTickets

SELECT S.MovieId, S.Date, S.TheaterId, C.BirthYear, C.HomeCityName, **COUNT**(*)
FROM Ticket **AS** T, Showing **AS** S, Customer **AS** C
WHERE T.ShowingId=S.ShowingId **AND** T.CustomerId=C.CustomerId
GROUP BY S.MovieId, S.Date, S.TheaterId, C.BirthYear, C.HomeCityName

Income

SELECT S.MovieId, S.Date, S.TheaterId, C.BirthYear, C.HomeCityName, **SUM**(S.Price)
FROM Ticket **AS** T, Showing **AS** S, Customer **AS** C
WHERE T.ShowingId=S.ShowingId **AND** T.CustomerId=C.CustomerId
GROUP BY S.MovieId, S.Date, S.TheaterId, C.BirthYear, C.HomeCityName

3. Logical design



4. Query answering

4a) Considering only the customers from Rome, aggregate the total income by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).

```
SELECT D.Year, D.Month, D.Date, SUM(T.Income)
FROM Tickets AS T, Date AS D
WHERE T.Date=D.Date AND T.CustomerCity='Rome'
GROUP BY D.Year, D.Month, D.Date WITH ROLLUP
```

4b) Compute the total number of tickets for each actor (specify id and name), customer home region, customer birth year and region of the theater.

```
SELECT A.ActorId, A.ActorName, C-C.Region, T.CustomerBirthYear, C-T.Region, SUM(T.NumTickets)
FROM Tickets AS T, City AS C-C, Theater AS Th, City AS C-T, MovieActor AS MA, Actor AS A
WHERE T.CustomerCity=C-C.CityName AND T.TheaterId=Th.TheaterId AND Th.CityName=C-T.CityName
        AND T.MovieId=MA.MovieId AND MA.ActorId=A.ActorId
GROUP BY A.ActorId, A.ActorName, C-C.Region, T.CustomerBirthYear, C-T.Region
```

4c) Compute the total income realized by each movie (specify id and title), but considering only the movies starring at least one actress.

```
SELECT M.MovieId, M.Title, SUM(T.Income)
FROM Tickets AS T, Movie AS M
WHERE T.MovieId=M.MovieId AND M.MovieId IN (
                SELECT MA.MovieId
                FROM MovieActor AS MA, Actor AS A
                WHERE MA.ActorId=A.ActorId AND A.Gender='F'
                )
GROUP BY M.MovieId, M.Title
```

4d) Considering only the theaters located in Milan, compute the genre(s) associated with the greatest number of tickets for each day of the week and theater (specify id and name).

```
CREATE VIEW DowTheaterGenreTickets (DayOfWeek, TheaterId, TheaterName, Genre, NumTickets) AS (
        SELECT D.DayOfWeek, Th.TheaterId, Th.TheaterName, M.Genre, SUM(T.NumTickets)
        FROM Tickets AS T, Date AS D, Movie AS M, Theater AS Th
        WHERE T.Date=D.Date AND T.MovieId=M.MovieId AND T.TheaterId=Th.TheaterId AND
                Th.CityName='Milan'
        GROUP BY D.DayOfWeek, Th.TheaterId, Th.TheaterName, M.Genre
)

SELECT DayOfWeek, TheaterId, TheaterName, Genre
FROM DowGenreTickets AS D
WHERE D.NumTickets = (
                                SELECT MAX(D2.NumTickets)
                                FROM DowGenreTickets AS D2
                                WHERE D.DayOfWeek=D2.DayOfWeek AND D.TheaterId=D2.TheaterId
                        )
```

Technologies for Information Systems

Part II (22 points)

prof. L. Tanca – September 8th, 2017

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliConferences is a company organizing scientific conferences. The conferences are grouped in series, and for each series there is a conference in each year. Users may register to participate in conferences, and each conference defines some registration categories, each associated with its own price. The users register to the conferences selecting a specific registration category, and pay an amount depending on the registration category and on the discount that may be applied.
The management of PoliConferences has now hired you to design a data warehouse to analyze the registrations.

The following is the schema of the operational database used by PoliConferences:

COUNTRY (CountryName, Continent)

CONFERENCE (Series, Year, Country, StartDate, DurationInDays, Capacity) *// Capacity is the maximum number of registrants that can be accepted for this conference. An example of a conference series is: "Very Large DataBases"; every year there is a different conference of the series "Very Large DataBases".*

USER (UserCode, Name, Surname, CountryName)

REGISTRATIONCATEGORY (Series, Year, CategoryName, Price) *// Example of registration category names: "Early" registration, "Regular" registration, "Late" registration. All the prices are expressed in US dollars.*

REGISTRATION (UserCode, Series, Year, CategoryName, RegistrationDate, PercDiscount) *// The discount is a percentage expressed as a number between 0 and 100; each registration may be associated with a different discount.*

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).

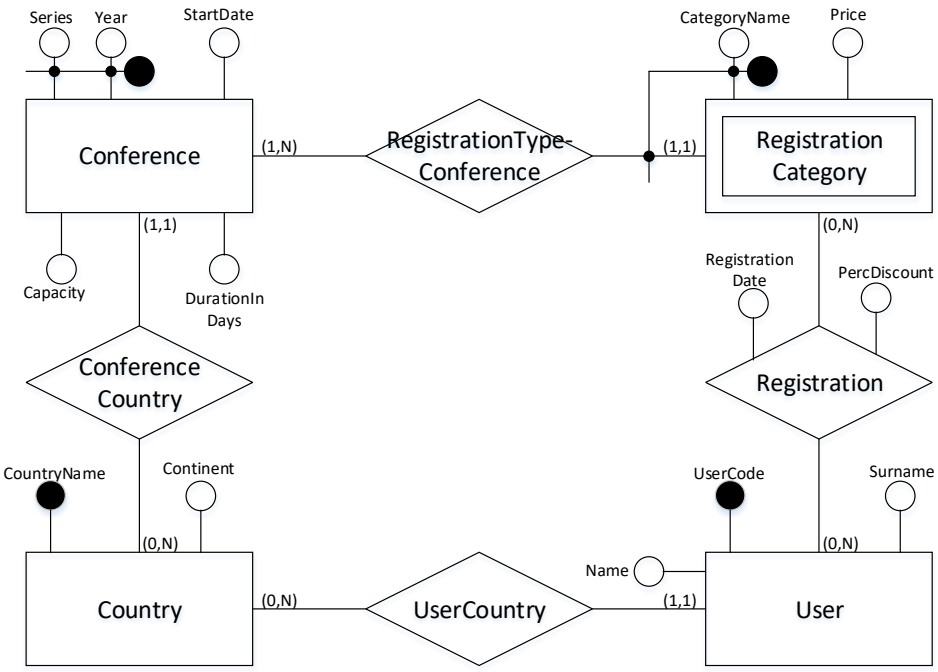
c. (2 points) Produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
   a. (2 points) Considering only the conferences located in Europe, compute the average discount for each continent of the user and day of the week.
   b. (2 points) Compute the total revenue for each conference country, registration date and user country. Include in the answer also the aggregations computed using only one and two of the three attributes.
   c. (2 points) For each series, compute the conference(s) (specify the year(s)) with the greatest number of registrants.
   d. (2 points) Find the conferences (specify series and year) that in October 2016 have collected from the registrations a revenue greater of more than 20% than that collected in September 2016.

SOLUTION

1. Reverse engineering

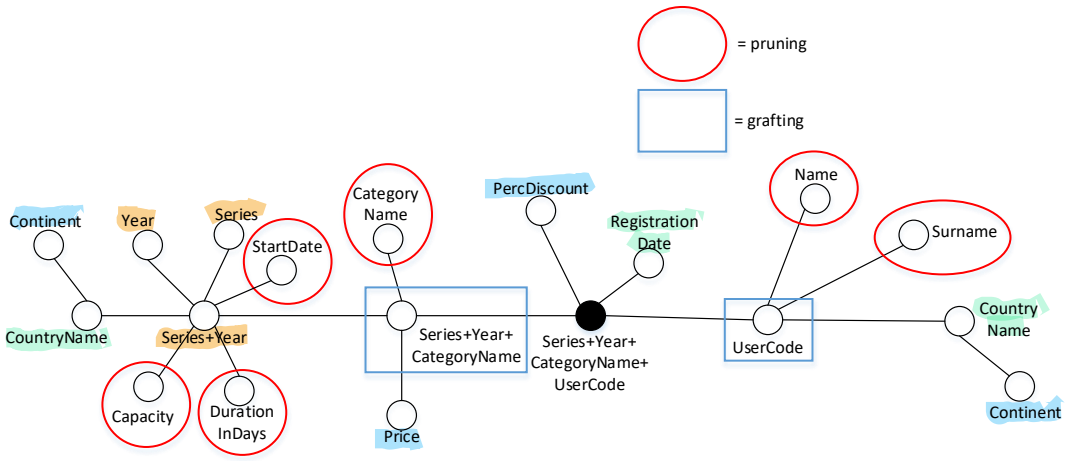


2. Conceptual design

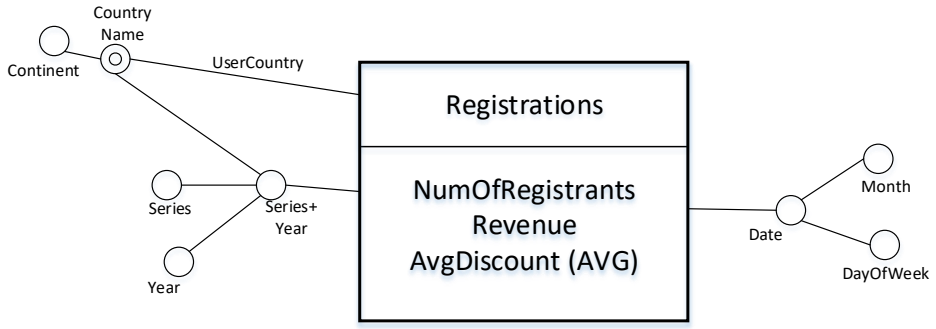
RELEVANT FOR QUERIES a b d

Fact: Registrations (Registration relationship)

2a) Attribute tree



2b) Fact schema



2c) Glossary

NumOfRegistrations

SELECT U.CountryName, R.Series, R.Year, R.RegistrationDate, **COUNT**(*)
FROM Registration **AS** R, User **AS** U
WHERE R.UserCode=U.UserCode
GROUP BY U.CountryName, R.Series, R.Year, R.RegistrationDate

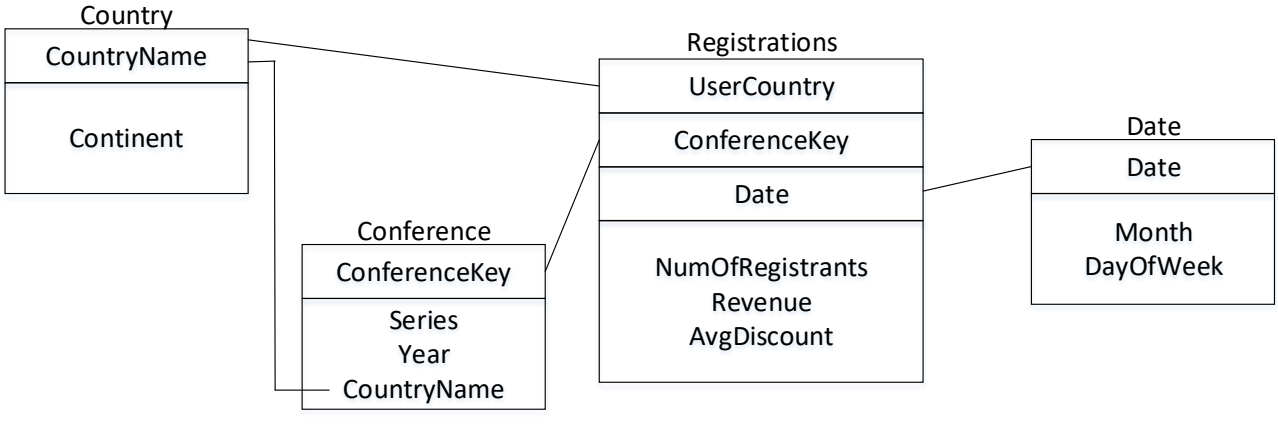
Revenue

SELECT U.CountryName, R.Series, R.Year, R.RegistrationDate,
    SUM(RC.Price - RC.Price*R.PercDiscount*0.01)
FROM Registration **AS** R, User **AS** U, RegistrationCategory **AS** RC
WHERE R.UserCode=U.UserCode **AND** R.Series=RC.Series **AND** R.Year=RC.Year **AND**
    R.CategoryName=RC.CategoryName
GROUP BY U.CountryName, R.Series, R.Year, R.RegistrationDate

AvgDiscount

SELECT U.CountryName, R.Series, R.Year, R.RegistrationDate, **AVG**(R.PercDiscount)
FROM Registration **AS** R, User **AS** U
WHERE R.UserCode=U.UserCode
GROUP BY U.CountryName, R.Series, R.Year, R.RegistrationDate

3. Logical design



4. Query answering

4a) Considering only the conferences located in Europe, compute the average discount for each continent of the user and day of the week.

```
SELECT C-U.Continent, D.DayOfWeek, SUM(R.NumOfRegistrants*R.AvgDiscount)/SUM(R.NumOfRegistants)
FROM Registrations AS R, Conference AS C, Country AS C-C, Country AS C-U
WHERE R.ConferenceKey=C.ConferenceKey AND C.CountryName=C-C.CountryName AND
        R.UserCountry=C-U.CountryName AND C-C.Continent='Europe'
GROUP BY C-U.Continent, D.DayOfWeek
```

4b) Compute the total revenue for each conference country, registration date and user country. Include in the answer also the aggregations computed using only one and two of the three attributes.

```
SELECT C.CountryName, R.Date, R.UserCountry, SUM(R.Revenue)
FROM Registrations AS R, Conference AS C
WHERE R.ConferenceKey=C.ConferenceKey
GROUP BY C.CountryName, R.Date, R.UserCountry WITH CUBE
```

4c) For each series, compute the conference(s) (specify the year(s)) with the greatest number of registrants.

```
CREATE VIEW ConferenceNumRegistrants (ConferenceKey, Series, Year, Num) AS (
        SELECT C.ConferenceKey, C.Series, C.Year, SUM(R.NumOfRegistrants)
        FROM Registrations AS R, Conference AS C
        WHERE R.ConferenceKey=C.ConferenceKey
        GROUP BY C.ConferenceKey, C.Series, C.Year
)
SELECT C.Series, C.Year
FROM ConferenceNumRegistrants AS C
WHERE C.Num =   (       SELECT MAX(C2.Num)
                        FROM ConferenceNumRegistrants AS C2
                        WHERE C2.Series=C.Series
                )
```

4d) Find the conferences (specify series and year) that in October 2016 have collected from the registrations a revenue greater of more than 20% than that collected in September 2016.

```
SELECT C.Series, C.Year
FROM Registrations AS R, Conference AS C, Date AS D
WHERE R.ConferenceKey=C.ConferenceKey AND R.Date=D.Date AND D.Month='Oct-2016'
GROUP BY C.ConferenceKey, C.Series, C.Year
HAVING SUM(R.Revenue) > 1.2 * (
                                        SELECT SUM(R2.Revenue)
```

```
        FROM Registrations AS R2, Date AS D2
        WHERE R2.Date=D2.Date AND D2.Month='Sep-2016' AND
              R2.ConferenceKey=R.ConferenceKey
)
```

Technologies for Information Systems

Part II (23 points)

prof. L. Tanca – January 19th, 2018

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliCourses is an online platform that offers courses and related exercises. Exercises are of different types, and each has a maximum score. The score obtained when solving an exercise is computed as the fraction of the maximum score corresponding to the level of completion, plus a possible bonus assigned by the system (e.g., because the student solved the exercise quickly). A student can try each specific exercise only once.
The management of PoliCourses has now hired you to design a data warehouse to analyze the results of the exercises performed by the students.

The following is the schema of the operational database used by PoliCourses:

UNIVERSITY (UniversityName, Country, NumOfStudents, Rector)
STUDENT (StudentId, Name, Surname, BirthDate, Nationality, UniversityName*) *// This table contains the students registered to the system. Registered students may be affiliated to a university (UniversityName is null if this is not the case).*
COURSE (CourseId, CourseTitle, Area) *// Sample areas: computer science, mathematics, physics, …*
EXERCISE (ExerciseId, Title, Text, Type, Topic, MaximumScore, CourseId) *// Sample types: multiple choice, check all that apply, fill in the blank, … Sample topics: database queries, derivatives, quantum mechanics, …*
EXERCISERESULT (ExerciseId, StudentId, Date, PercentageOfCompletion, Bonus)
// PercentageOfCompletion is a number between 0 and 100. The score obtained by the student for the performed exercise is computed applying the percentage of completion to the maximum score of that exercise, and then summing the bonus.

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

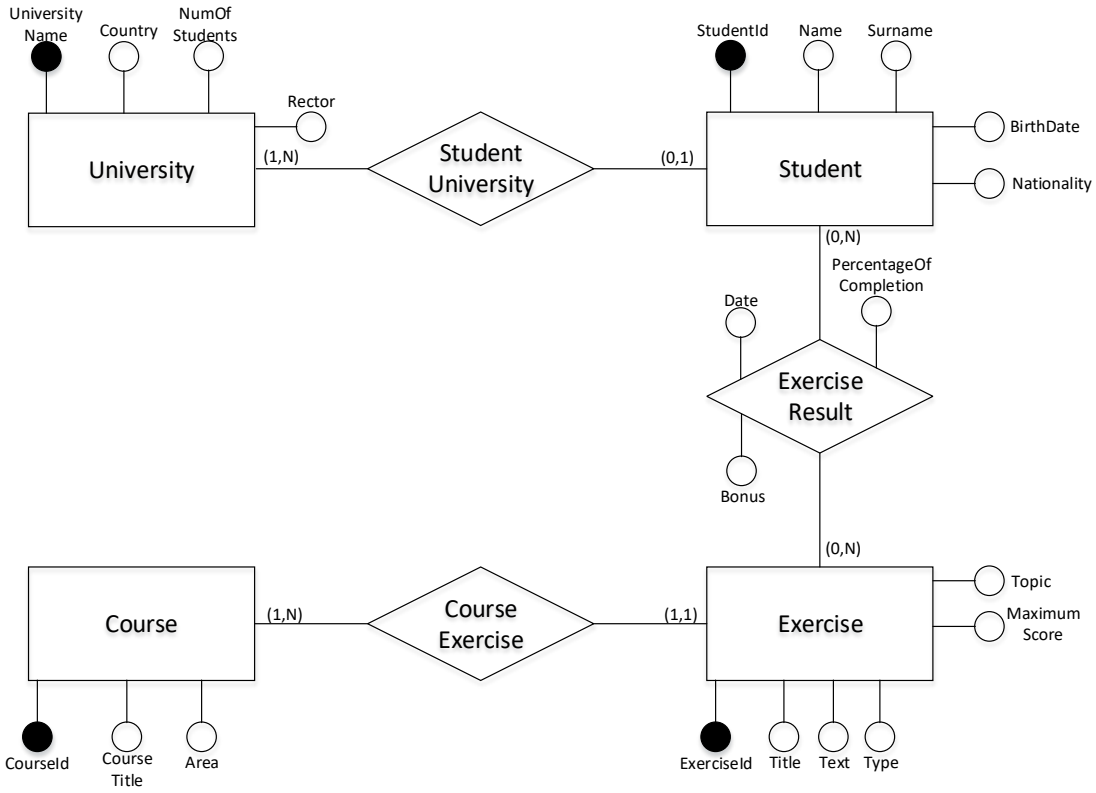
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2.5 points) Produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (2 points) Compute the total number of solved exercises for each university, type of exercise and area of the course. Include in the answer also the aggregations computed using only one and two of the three attributes.
    b. (2.5 points) Compute the average bonus assigned on Monday to students born in 1994 for each course (specify id and title), topic and country of the university.
    c. (2 points) Aggregate the total score by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).
    d. (2 points) Identify the Italian student(s) who have obtained the greatest total score (specify id, name and surname).

SOLUTION

1. Reverse engineering



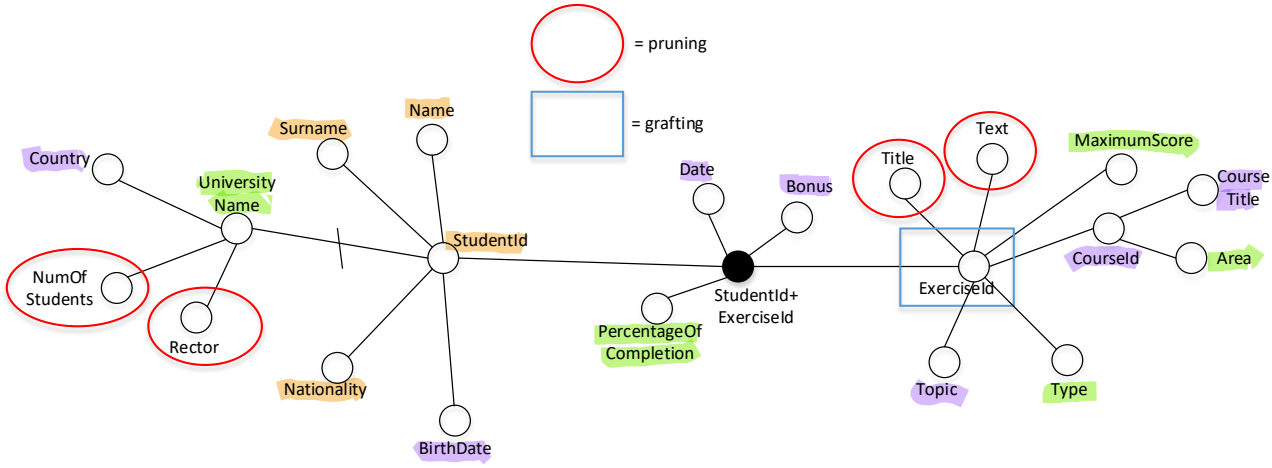
RELEVANT FOR QUERIES

a b d

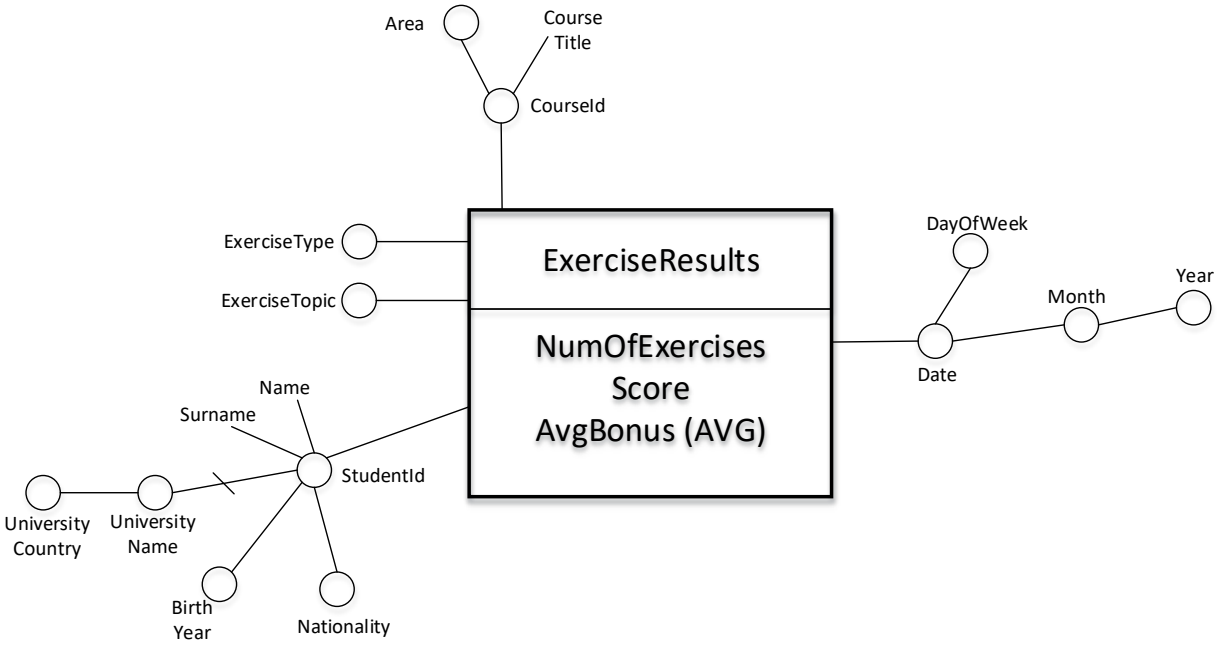
2. Conceptual design

Fact: ExerciseResults (ExerciseResult relationship)

2a) Attribute tree



2b) Fact schema



2c) Glossary

NumOfExercises

SELECT ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId, **COUNT**(*)
FROM ExerciseResult **AS** ER, Exercise **AS** E
WHERE ER.ExerciseId=E.ExerciseId
GROUP BY ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId

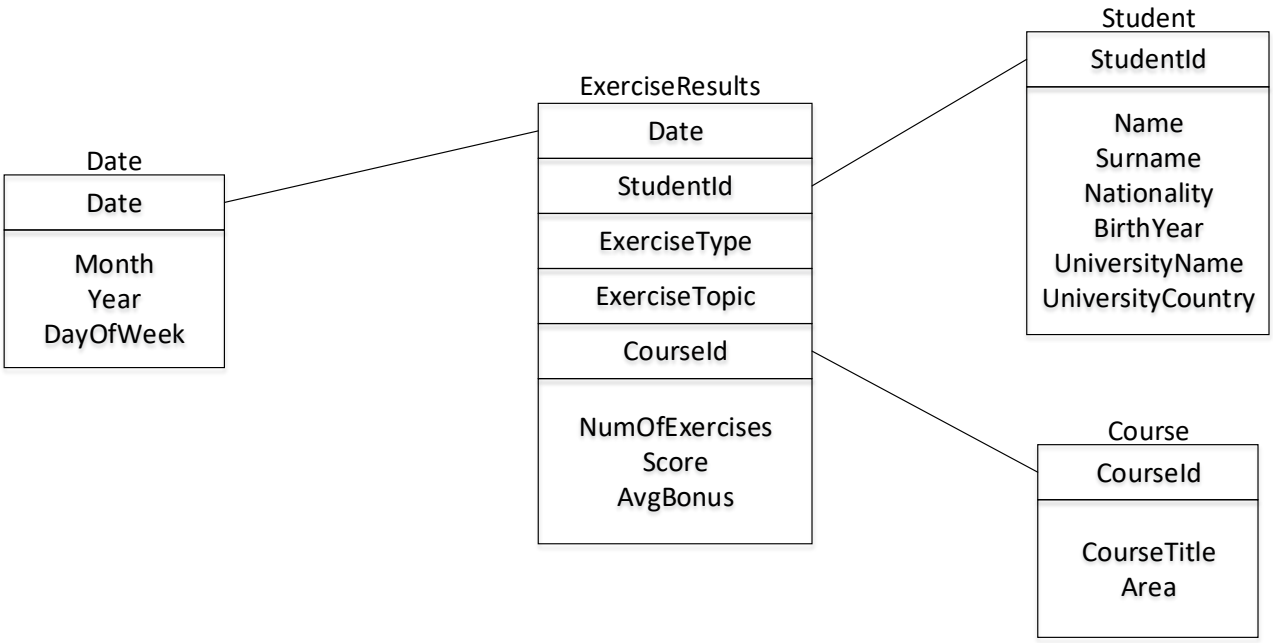
Score

SELECT ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId,
SUM(ER.Percentage*0.01*E.MaximumScore+ER.Bonus)
FROM ExerciseResult **AS** ER, Exercise **AS** E
WHERE ER.ExerciseId=E.ExerciseId
GROUP BY ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId

AvgBonus

SELECT ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId, **AVG**(ER.Bonus)
FROM ExerciseResult **AS** ER, Exercise **AS** E
WHERE ER.ExerciseId=E.ExerciseId
GROUP BY ER.StudentId, ER.Date, E.Type, E.Topic, E.CourseId

3. Logical design



4. Query answering

4a) Compute the total number of solved exercises for each university, type of exercise and area of the course. Include in the answer also the aggregations computed using only one and two of the three attributes.

```
SELECT S.UniversityName, E.ExerciseType, C.Area, SUM(E.NumOfExercises)
FROM ExerciseResults AS E, Student AS S, Course AS C
WHERE E.StudentId=S.StudentId AND E.CourseId=C.CourseId
GROUP BY S.UniversityName, E.ExerciseType, C.Area WITH CUBE
```

4b) Compute the average bonus assigned on Monday to students born in 1994 for each course (specify id and title), topic and country of the university.

```
SELECT C.CourseId, C.CourseTitle, E.Topic, S.UniversityCountry,
        SUM(E.NumOfExercises*E.AvgBonus)/SUM(E.NumOfExercises)
FROM ExerciseResults AS E, Student AS S, Course AS C, Date AS D
WHERE E.StudentId=S.StudentId AND E.CourseId=C.CourseId AND E.Date=D.Date AND
        D.DayOfWeek='Monday' AND S.BirthYear=1994
GROUP BY C.CourseId, C.CourseTitle, E.Topic, S.UniversityCountry
```

4c) Aggregate the total score by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).

```
SELECT D.Year, D.Month, D.Date, SUM(E.Score)
FROM ExerciseResults AS E, Date AS D
WHERE E.Date=D.Date
GROUP BY D.Year, D.Month, D.Date WITH ROLLUP
```

4d) Identify the Italian student(s) who have obtained the greatest total score (specify id, name and surname).

```
CREATE VIEW StudentTotalScore (StudentId, Name, Surname, Score) AS (
        SELECT S.StudentId, S.Name, S.Surname, SUM(E.Score)
        FROM ExerciseResults AS E, Student AS S
        WHERE E.StudentId=S.StudentId AND S.Nationality='Italy'
        GROUP BY S.StudentId, S.Name, S.Surname
)
SELECT StudentId, Name, Surname
FROM StudentTotalScore
WHERE Score = (         SELECT MAX(Score)
                        FROM StudentTotalScore
                )
```

Technologies for Information Systems

Part II (23 points)

prof. L. Tanca – February 20th, 2018

Available Time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliCoach is a transport company offering coach services in Italy. PoliCoach issues tickets for the travels; tickets are strictly personal, and may comprehend multiple travels. The price of a specific travel may vary depending on the other travels with which it is sold, therefore the total price of a ticket cannot be decomposed into the prices of the component travels.

UniCoach too is a transport company offering coach services, but it operates in France, Spain and Portugal. UniCoach issues personal and group tickets, and each ticket is valid for a single travel. The price of a travel for each person is fixed.

PoliCoach and UniCoach have now merged into a unique company named ***UniPoliCoach***. The UniPoliCoach ownership asks you to integrate the relational databases of the two companies into a unique relational database. You must perform the integration ensuring to lose the least possible amount of information.

The original relational schemas of the two sources are reported below.

PoliCoach

VEHICLE (PlateNumber, Manufacturer, Displacement, NrSeats)
TRAVEL (TravelId, Date, DepartureCity, DestinationCity, LengthMiles, VehiclePlateNumber)
CUSTOMER (CustomerId, SSN, BirthDate, Name) *// The Name attribute contains both first name and last name, separated by a blank space; within the Name attribute value the first name is always composed of a single word.*
TICKET (CustomerId, PurchaseTimestamp, Class, Price)
TICKETTRAVEL (CustomerId, PurchaseTimestamp, TravelId)

UniCoach

COACH (PlateNumber, Brand, NrSeats)
CITY (CityName, Country, NrInhabitants, Seaside) *// Seaside is a boolean attribute that is true if the city is on the sea.*
TRIP (TripId, Date, LengthKm, PricePerPerson, DepartureCity, DestinationCity, CoachPlateNumber)
PASSENGER (SSN, BirthDate, GivenName, Surname, Address)
TICKET (TicketId, PurchaseTimestamp, Class, TripId)
TICKETPASSENGER (TicketId, SSN)

Notes:

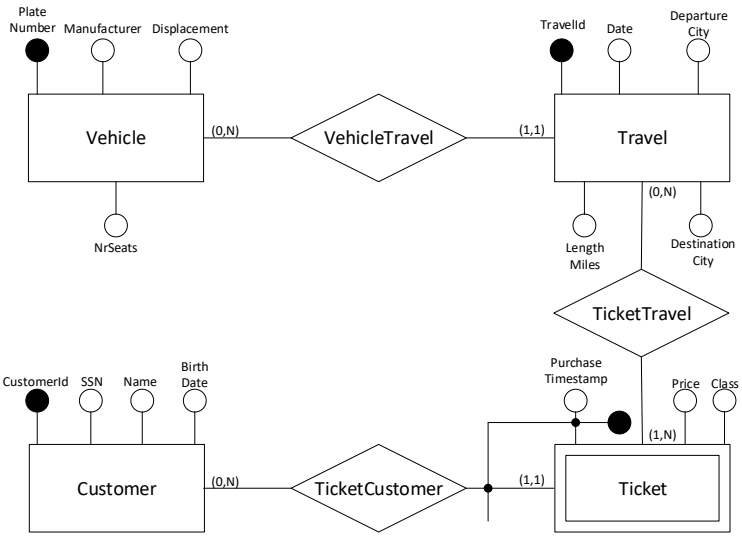
- You can assume that the people in the two data sources are disjoint.
- 1 mile is equal to 1.609 km.

1. **Source schema reverse engineering**. Provide, for each input data source, the reverse engineering from the logical schema to the conceptual model (ER graph). (5 points)
2. **Schema integration**. Design an integrated global conceptual schema (ER graph) for *UniPoliCoach* capturing all the data coming from both *PoliCoach* and *UniCoach*, and provide the corresponding global logical schema. In more detail, follow these steps:
    a. *Related concept identification and conflict analysis and resolution*. Write a table as shown in the exercise sessions, using the following columns: "PoliCoach concept", "UniCoach concept", "Conflict", "Solution". (3.5 points)
    b. *Integrated conceptual schema* (ER graph). (4 points)
    c. *Conceptual to logical translation of the integrated schema*. (2.5 points)
3. **Query answering and mapping definition**. Consider the query Q: "Find city and country of destination of the first-class tickets for travels with length greater than 300 km".
    a. *Query formulation*. Consider query Q posed on the logical schema of *UniPoliCoach* and write it in SQL. (1.5 points)
    b. *Mapping definition*. Write the GAV mappings between the schema of *UniPoliCoach* and the two sources using SQL. Write the mappings only for the tables used to answer query Q**.** (4 points)
    c. *Query rewriting*. Show the rewriting of Q on the two data sources using SQL. (2.5 points)

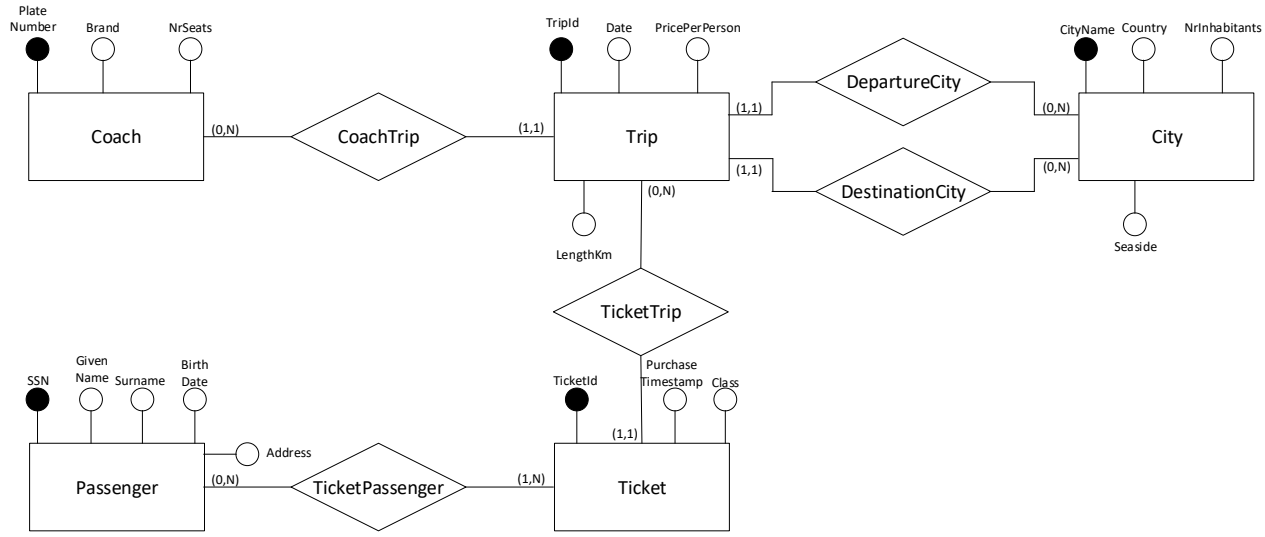
SOLUTION

1. Source schema reverse engineering

PoliCoach



UniCoach

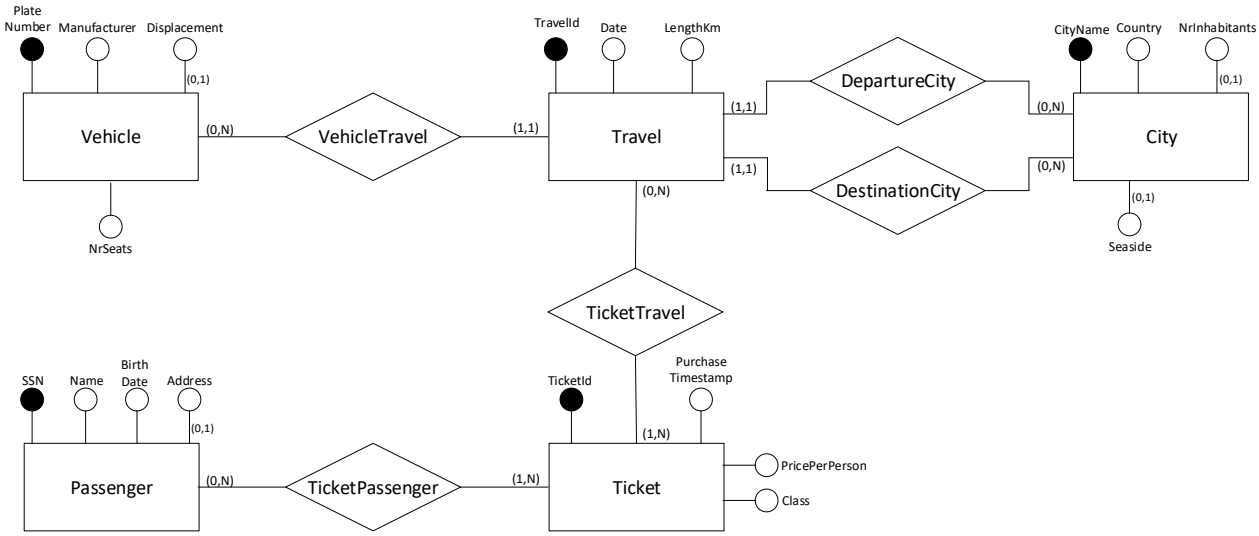


2. Schema integration

2a) Related concept identification + conflict analysis and resolution

PoliCoach	UniCoach	Conflict	Solution
Vehicle	Coach	Name conflicts	
		- Entity name	Vehicle
		- Manufacturer → Brand	Manufacturer
Travel	Trip	Name conflicts	
		- Entity name	Travel
		- TravelId → TripId	TravelId
		Data semantics conflicts	
		- LengthMiles → LengthKm	LengthKm
		Structure conflicts	
		- Departure and destination cities are represented as attributes → City is an entity	City is an entity
Customer	Passenger	Name conflicts	
		- Entity name	Passenger
		Key conflict	
		- CustomerId → SSN	SSN
		Structure conflicts	
		- Given name and surname in a unique attribute Name → Two distinct attributes	Just one attribute Name
Ticket	Ticket	Key conflict	
		- CustomerId+ PurchaseTimestamp → TicketId	TicketId
		Cardinality conflicts	
		- A ticket is associated with just one customer → A ticket may be associated with multiple passengers	A ticket may be associated with multiple passengers
		- A ticket may be associated with multiple travels → A ticket is associated with just one trip	A ticket may be associated with multiple travels
		Structure conflicts	
		- The price (per person) of a travel varies for each ticket → The price per person of a trip is fixed	The price (per person) of a travel varies for each ticket

2b) Global conceptual schema



2c) Conceptual to logical translation

VEHICLE (<u>PlateNumber</u>, Manufacturer, NrSeats, Displacement*)
CITY (<u>CityName</u>, Country, NrInhabitants*, Seaside*)
TRAVEL (<u>TravelId</u>, Date, LengthKm, DepartureCity, DestinationCity, VehiclePlateNumber)
PASSENGER (<u>SSN</u>, Name, Birthdate, Address*)
TICKET (<u>TicketId</u>, PurchaseTimestamp, Class, PricePerPerson)
TICKETPASSENGER (<u>TicketId</u>, <u>SSN</u>)
TICKETTRAVEL (<u>TicketId</u>, <u>TravelId</u>)

3. Query answering and mapping definition

3a) Query formulation

Find city and country of destination of the first-class tickets for travels with length greater than 300 km.

<u>NOTE</u>:
During the exam also the following alternate equivalent text of the query was provided to the students:
Find city and country of destination of the travels with length greater than 300 Km associated with first-class tickets.

SELECT C.CityName, C.Country
FROM City **AS** C, Travel **AS** Tr, TicketTravel **AS** TT, Ticket **AS** Ti
WHERE C.CityName=Tr.DestinationCity **AND** Tr.TravelId=TT.TravelId **AND** TT.TicketId=Ti.TicketId **AND**
Ti.Class=1 **AND** Tr.LengthKm>300

3b) GAV mapping definition

```
CREATE VIEW UniPoliCoach.City (CityName, Country, NrInhabitants, Seaside) AS (
        SELECT DepartureCity, 'Italy', null, null
        FROM PoliCoach.Travel

        UNION

        SELECT DestinationCity, 'Italy', null, null
        FROM PoliCoach.Travel

        UNION

        SELECT CityName, Country, NrInhabitants, Seaside
        FROM UniCoach.City
)


CREATE VIEW UniPoliCoach.Travel (TravelId, Date, LengthKm, DepartureCity, DestinationCity,
VehiclePlateNumber) AS (
        SELECT KeyGenTravel(TravelId, 'PoliCoach'), Date, DepartureCity, DestinationCity,
                LengthMiles*1.609, VehiclePlateNumber
        FROM PoliCoach.Travel

        UNION

        SELECT KeyGenTravel(TripId, 'UniCoach'), Date, DepartureCity, DestinationCity, LengthKm,
                CoachPlateNumber
        FROM UniCoach.Trip
)


CREATE VIEW UniPoliCoach.Ticket (TicketId, PurchaseTimestamp, Class, PricePerPerson) AS (
        SELECT KeyGenTicket(CustomerId||PurchaseTimestamp, 'PoliCoach'), PurchaseTimestamp, Class,
                Price
        FROM PoliCoach.Ticket

        UNION

        SELECT KeyGenTicket(Ti.TicketId, 'UniCoach'), Ti.PurchaseTimestamp, Ti.Class, Tr.PricePerPerson
        FROM UniCoach.Ticket AS Ti, UniCoach.Trip AS Tr
        WHERE Ti.TripId=Tr.TripId
)


CREATE VIEW UniPoliCoach.TicketTravel (TicketId, TravelId) AS (
        SELECT KeyGenTicket(CustomerId||PurchaseTimestamp, 'PoliCoach'), KeyGenTravel(TravelId,
                'PoliCoach')
        FROM PoliCoach.TicketTravel
```

```
    UNION

    SELECT KeyGenTicket(TicketId, 'UniCoach'), KeyGenTravel(TripId, 'UniCoach')
    FROM UniCoach.Ticket
)
```

3c) Query rewriting

Find city and country of destination of the first-class tickets for travels with length greater than 300 km.

```
SELECT Tr.DestinationCity, 'Italy'
FROM Travel AS Tr, TicketTravel AS TT, Ticket AS Ti
WHERE Tr.TravelId=TT.TravelId AND TT.CustomerId=Ti.CustomerId AND
    TT.PurchaseTimestamp=Ti.PurchaseTimestamp AND Ti.Class=1 AND Tr.LengthMiles*1.609>300

UNION

SELECT C.CityName, C.Country
FROM City AS C, Trip AS Tr, Ticket AS Ti
WHERE C.CityName=Tr.DestinationCity AND Tr.TripId=Ti.TripId AND Ti.Class=1 AND Tr.LengthKm>300
```

Technologies for Information Systems

Part II (23 points)

prof. L. Tanca – June 22nd, 2018

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliScience is a platform offering the online purchase and download of scientific journal papers in electronic format. Registered users can access the platform and download papers by paying a fee. Note that the system charges every download, so if a user wishes to downolad again a paper he/she has already downloaded in the past, he/she is required to pay the fee again.

The management of PoliScience has now hired you to design a data warehouse to analyze the downloads performed by the users.

The following is the schema of the operational database used by PoliScience:

COUNTRY (CountryName, Continent)
USER (UserId, GivenName, Surname, Affiliation*, CountryName) *// Registered users may be affiliated to an institution (e.g., a university); Affiliation is null if this is not the case.*
PAPER (PaperId, Title, NumOfPages, PublicationDate, JournalName, Price)
DOWNLOAD (UserId, Date, Time, PaperId)
SUBTOPIC (SubtopicName, TopicName) *// Sample subtopic/topic pair: ("Databases", "Computer Science").*
AUTHOR (AuthorId, GivenName, Surname, BirthYear, CountryName)
PAPERSUBTOPIC (PaperId, SubtopicName, Relevance) *// A paper may have multiple subtopics, each with its relevance; the relevances of the subtopics to a paper sum to 1. For instance, a paper may be associated with the subtopic "Databases" with relevance 0.7 and with the subtopic "Software Engineering" with relevance 0.3.*
PAPERAUTHOR (PaperId, AuthorId) *// A paper may be authored by multiple co-authors.*

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

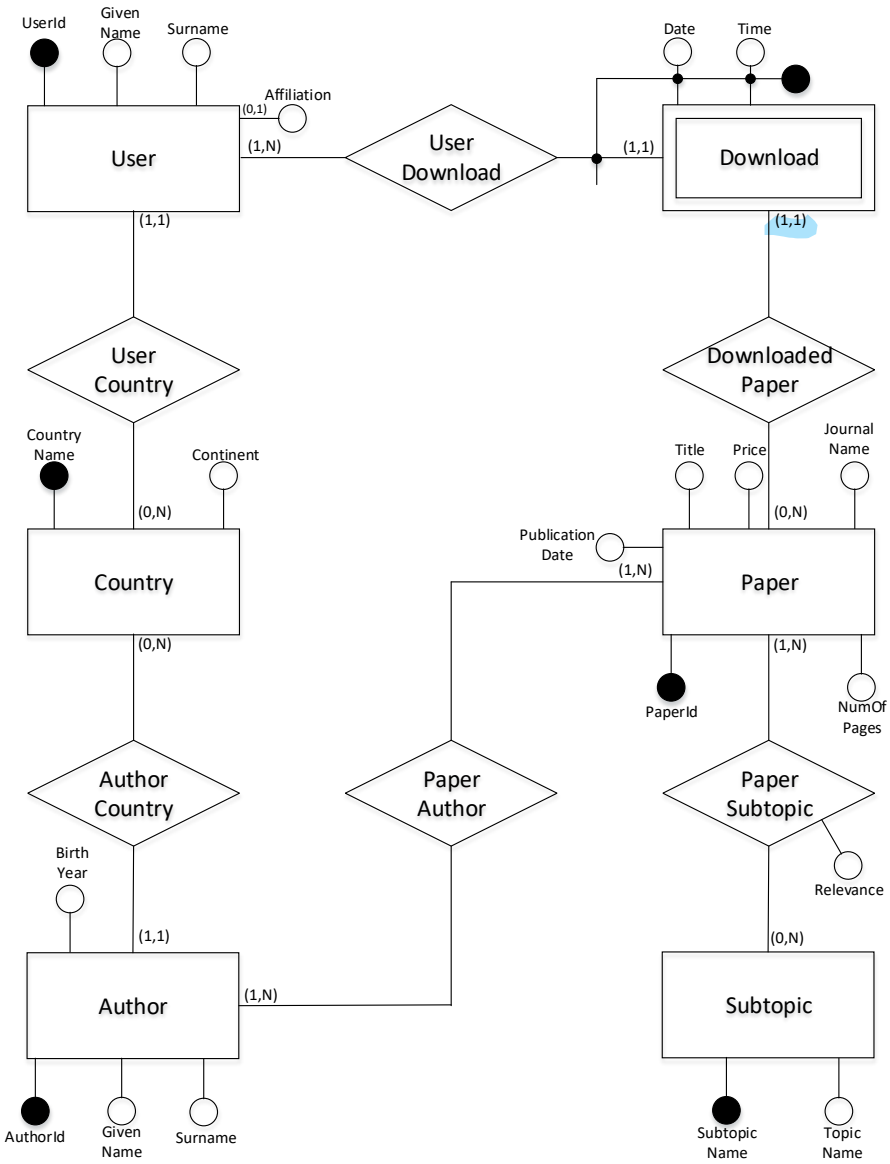
2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2.5 points) Produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

4. Write in SQL the following queries against the designed logical schema:
    a. (2 points) Aggregate the total income from European users by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).
    b. (2 points) Compute the number of downloads for each subtopic of “Computer Science”, weighted by the relevance of the subtopics to the papers.
    c. (2 points) Considering only Asian authors, compute the number of downloads for each author, user affiliation and journal name.
    d. (2.5 points) Compute the total income by paper (specify id and title) and user country, considering only the papers with at least one French author.

SOLUTION

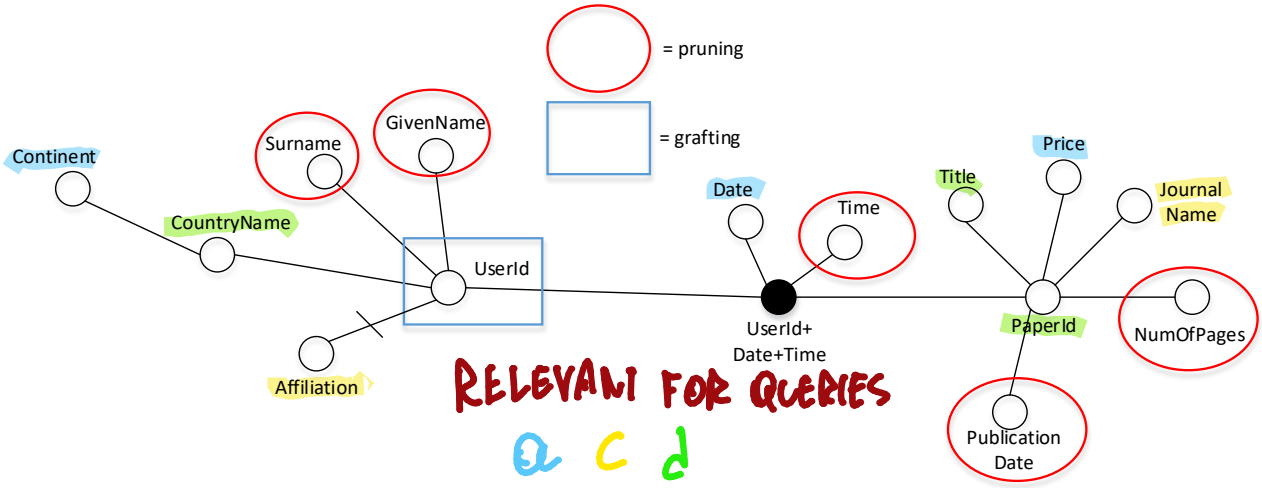
1. Reverse engineering



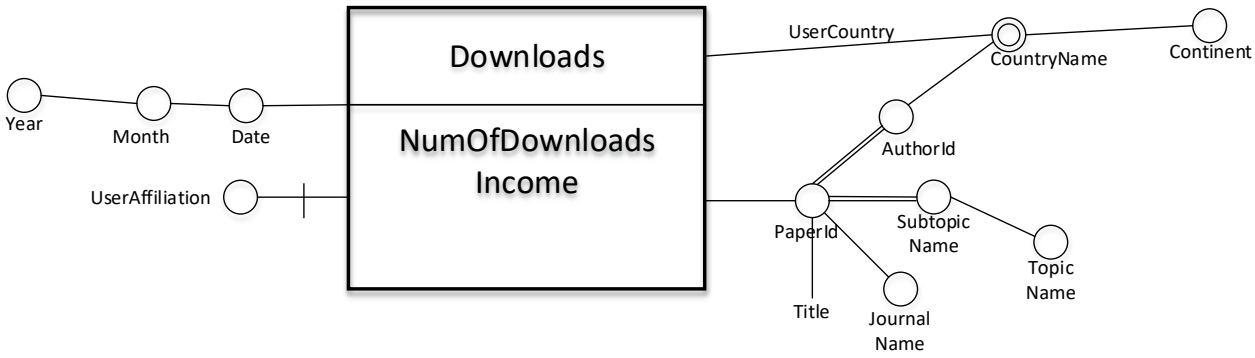
2. Conceptual design

Fact: Downloads (Download entity)

2a) Attribute tree



2b) Fact schema



2c) Glossary

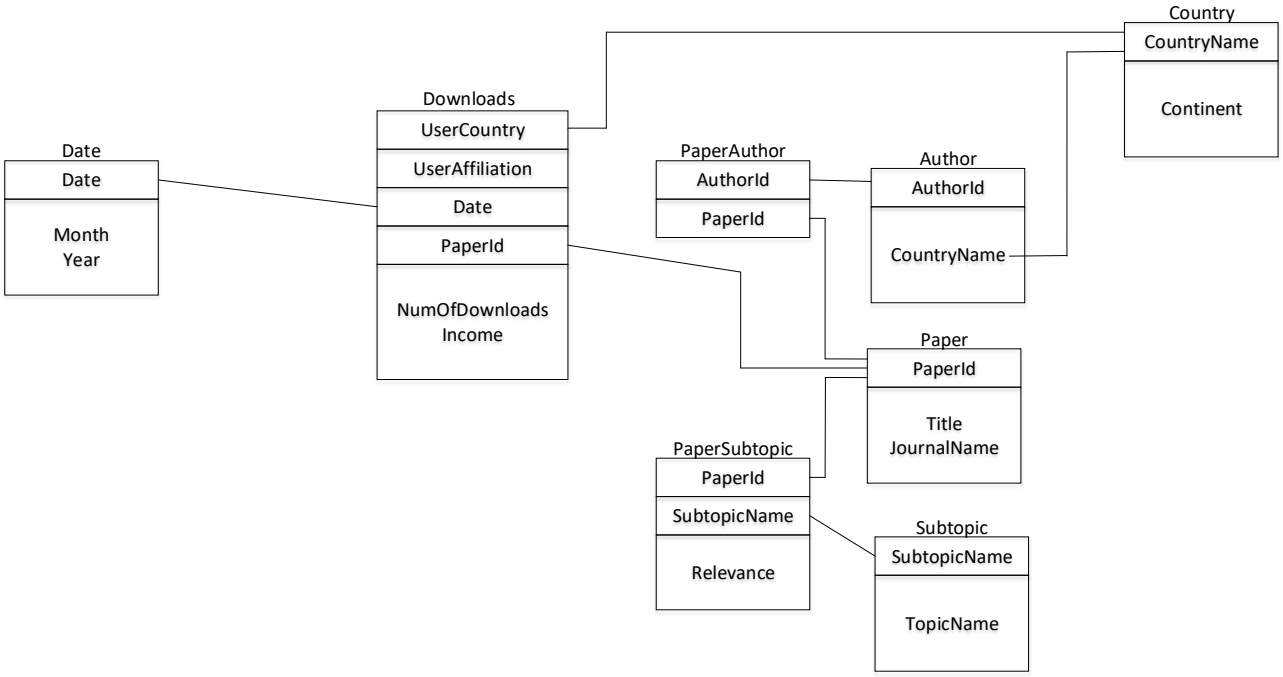
NumOfDownloads

SELECT D.PaperId, D.Date, U.Affiliation, U.CountryName, **COUNT**(*)
FROM Download **AS** D, User **AS** U
WHERE D.UserId=U.UserId
GROUP BY D.PaperId, D.Date, U.Affiliation, U.CountryName

Income

SELECT D.PaperId, D.Date, U.Affiliation, U.CountryName, **SUM**(P.Price)
FROM Download **AS** D, User **AS** U, Paper **AS** P
WHERE D.UserId=U.UserId **AND** D.PaperId=P.PaperId
GROUP BY D.PaperId, D.Date, U.Affiliation, U.CountryName

3. Logical design



4. Query answering

4a) Aggregate the total income from European users by date, month and year (include in the answer the aggregations computed only by date, only by month and only by year).

SELECT Da.Year, Da.Month, Da.Date, **SUM**(D.Income)
FROM Downloads **AS** D, Date **AS** Da, Country **AS** C
WHERE D.Date=Da.Date **AND** D.UserCountry=C.CountryName **AND** C.Continent='Europe'
GROUP BY Da.Year, Da.Month, Da.Date **WITH ROLLUP**

4b) Compute the number of downloads for each subtopic of "Computer Science", weighted by the relevance of the subtopics to the papers.

SELECT S.SubtopicName, **SUM**(D.NumOfDownloads*PS.Relevance)
FROM Downloads **AS** D, PaperSubtopic **AS** PS, Subtopic **AS** S
WHERE D.PaperId=PS.PaperId **AND** PS.SubtopicName=S.SubtopicName **AND** S.TopicName='Computer Science'
GROUP BY S.SubtopicName

4c) Considering only Asian authors, compute the number of downloads for each author, user affiliation and journal name.

```
SELECT A.AuthorId, D.UserAffiliation, P.JournalName, SUM(D.NumOfDownloads)
FROM Downloads AS D, Paper AS P, PaperAuthor AS PA, Author AS A, Country AS C
WHERE D.PaperId=P.PaperId AND P.PaperId=PA.PaperId AND PA.AuthorId=A.AuthorId AND
        A.CountryName=C.CountryName AND C.Continent='Asia'
GROUP BY A.AuthorId, D.UserAffiliation, P.JournalName
```

4d) Compute the total income by paper (specify id and title) and user country, considering only the papers with at least one French author.

```
SELECT P.PaperId, P.Title, D.UserCountry, SUM(D.Income)
FROM Downloads AS D, Paper AS P
WHERE D.PaperId=P.PaperId AND P.PaperId IN (
                                        SELECT PA.PaperId
                                        FROM PaperAuthor AS PA, Author AS A
                                        WHERE PA.AuthorId=A.AuthorId AND
                                                A.CountryName='France'
                                        )
GROUP BY P.PaperId, P.Title, D.UserCountry
```

Technologies for Information Systems

Part II (23 points)

prof. L. Tanca – July 9th, 2018

Available Time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliPatents is an organization granting technological patents to companies in America. Each patent may be assigned to multiple companies (i.e., the patent assignees) and is associated with one or more inventors; inventors are the people having participated in the invention that has been patented. PoliPatents stores the patent-related data in a relational database.

UniPatents is a similar organization operating in Europe. UniPatents too allows multiple inventors per patent but, differently from PoliPatents, allows just one assignee per patent. UniPatents stores the patent-related data in a big XML document.

The two organizations have now merged into a unique organization named ***UniPoliPatents***, and the management of UniPoliPatents asks you to integrate the two data sources into a unique **relational** database. You must perform the integration ensuring to lose the least possible amount of information.

The relational schema employed by PoliPatents is as follows:

PATENT (PatentId, Title, GrantDate, Abstract, CPCCategory) *// The id of the patent is an alphanumeric string always starting with 'PP' (e.g., 'PP12345678'). CPCCategory is the category of the patent on the basis of the Cooperative Patent Classification (CPC).*

CITY (CityName, Country)

ASSIGNEE (AssigneeId, Name, CityName)

INVENTOR (InventorId, Firstname, Lastname, CityName)

PATENTASSIGNEE (PatentId, AssigneeId)

PATENTINVENTOR (PatentId, InventorId)

CITATION (CitingPatent, CitedPatent) *// A row in this table indicates that the document describing the citing patent cites the cited patent.*

The following is the DTD of the UniPatents XML document:

```
<!ELEMENT PatentsDB (Patent*)>
<!ELEMENT Patent (Title, Summary, GrantDate, Assignee, Inventors, IPCCategories)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Summary (#PCDATA)>
<!ELEMENT GrantDate (#PCDATA)>
<!ELEMENT Assignee (Name, CityName, Country)>
<!ELEMENT Inventors (Inventor+)>
<!ELEMENT IPCCategories (IPCCategory+)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT CityName (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Inventor (Name, CityName, Country)>
<!ELEMENT IPCCategory EMPTY>
<!ATTLIST Patent PatentId CDATA #REQUIRED>
<!ATTLIST Assignee AssigneeId CDATA #REQUIRED>
<!ATTLIST Inventor InventorId CDATA #REQUIRED>
<!ATTLIST IPCCategory CatName CDATA #REQUIRED>
```

The following is a portion of a valid XML document according to the previous DTD:

```
<PatentsDB>
    <Patent PatentId="UP12345678">
        <Title>New algorithm to automatically solve university exams</Title>
        <Summary> … </Summary>
        <GrantDate>2017-12-15</GrantDate>
        <Assignee AssigneeId="10000001">
            <Name>Fake Corporation</Name>
            <CityName>Milan</CityName>
            <Country>Italy</Country>
        </Assignee>
        <Inventors>
            <Inventor InventorId="82000000">
                <Name>Mario#Rossi</Name>
                <CityName>Monza</CityName>
                <Country>Italy</Country>
            </Inventor>
            <Inventor InventorId="82000001"/>
                <Name>John#Smith</Name>
                <CityName>London</CityName>
                <Country>United Kingdom</Country>
            </Inventor>
        </Inventors>
        <IPCCategories>
            <IPCCategory CatName="Life-Saving"/>
            <IPCCategory CatName="Computer-Aided Design"/>
        </IPCCategories>
    </Patent>
    <Patent>
        …
    </Patent>
</PatentsDB>
```

Remarks:

- The id of the patent is an alphanumeric string always starting with 'UP' (e.g., 'UP12345678').
- IPCCategory is the category of the patent on the basis of International Patent Classification (IPC).
- Inventor names in UniPatents are represented in the form 'Firstname#Lastname'.
- UniPatents allows assignees and inventors to change city, so in different patents the same assignee/inventors may be associated with different cities.

NOTES:

- PoliPatents and UniPatents refer to different geographic areas, and you can assume that patents, assignees and inventors in the two data sources are disjoint.
- CPC categories and IPC categories are different.

1. **Source schema reverse engineering**. Provide, for each input data source, the reverse engineering from the logical schema to the conceptual model (ER graph). For the XML source *UniPatents* provide also the relational translation of the ER schema. (5 points)

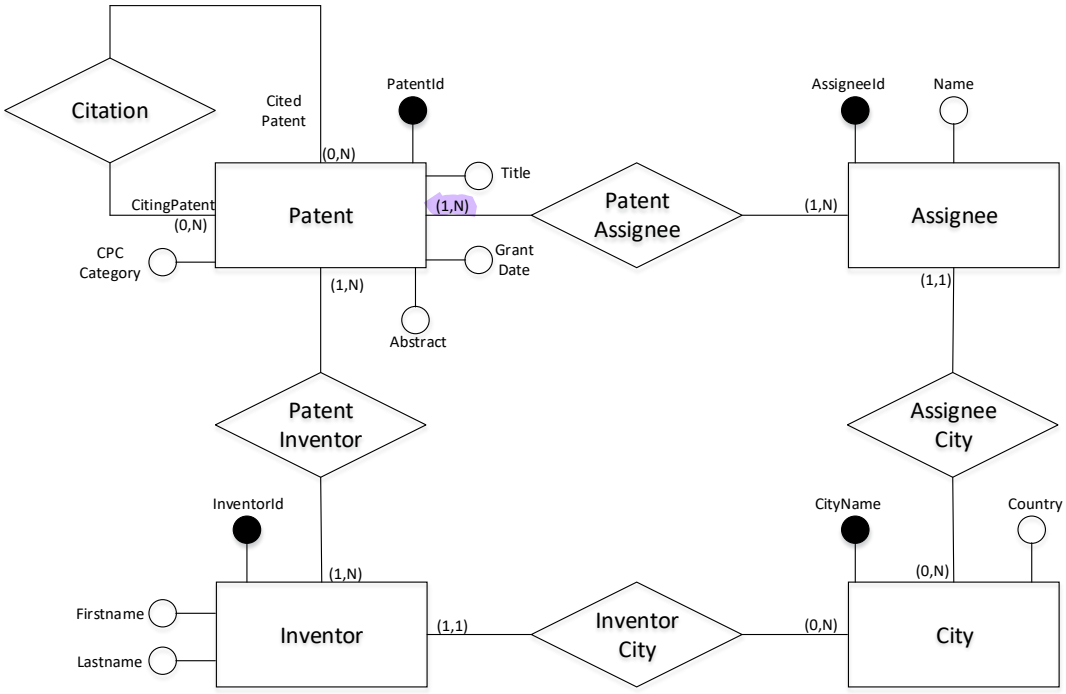
2. **Schema integration**. Design an integrated global conceptual schema (ER graph) for *UniPoliPatents* capturing all the data coming from both *PoliPatents* and *UniPatents*, and provide the corresponding global logical (relational) schema. In more detail, follow these steps:
    a. *Related concept identification and conflict analysis and resolution*. Write a table as shown in the exercise sessions, using the following columns: "PoliPatents concept", "UniPatents concept", "Conflict", "Solution". (3.5 points)
    b. *Integrated conceptual schema* (ER graph). (4 points)
    c. *Conceptual to logical translation of the integrated schema*. (2.5 points)

3. **Query answering and mapping definition**. Consider the query Q: "Find the pairs (id of the patent, name of the assignee) associated with assignees from Milan and patents granted in 2017".
    a. *Query formulation*. Consider query Q posed on the logical schema of *UniPoliPatents* and write it in SQL. (1.5 points)
    b. *Mapping definition*. Write the GAV mappings between the schema of *UniPoliPatents* and the two sources using SQL. For *UniPatents*, write the mappings between the *UniPoliPatents* integrated relational schema and the *UniPatents* relational schema defined at point 1. Write the mappings only for the tables used to answer query Q**.** (4 points)
    c. *Query rewriting*. Show the rewriting of Q on the two data sources using SQL. Again, for *UniPatents* consider the relational schema defined at point 1. (2.5 points)

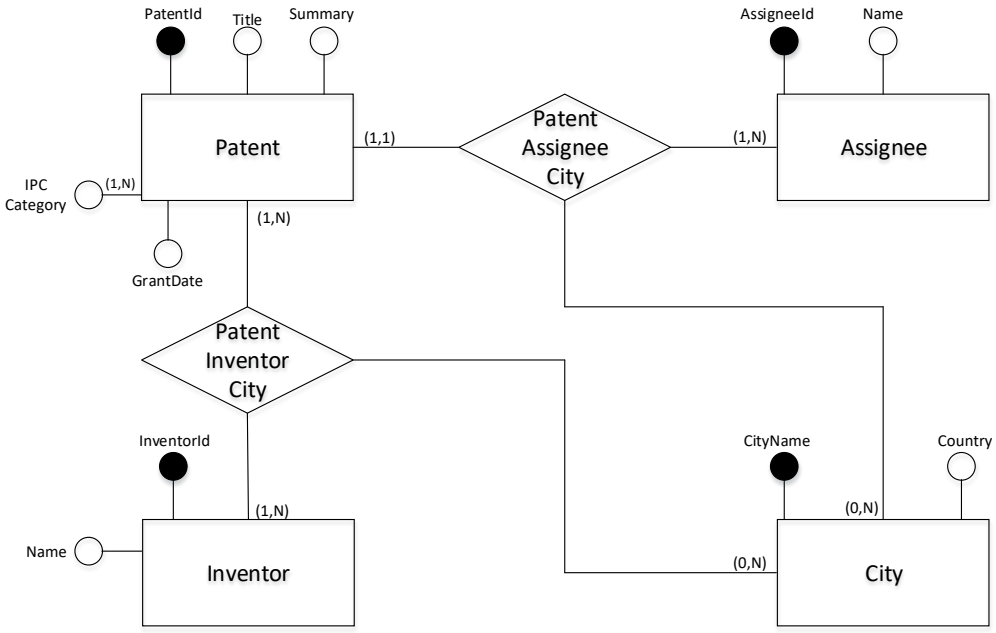
SOLUTION

1. Source schema reverse engineering

PoliPatents



UniPatents



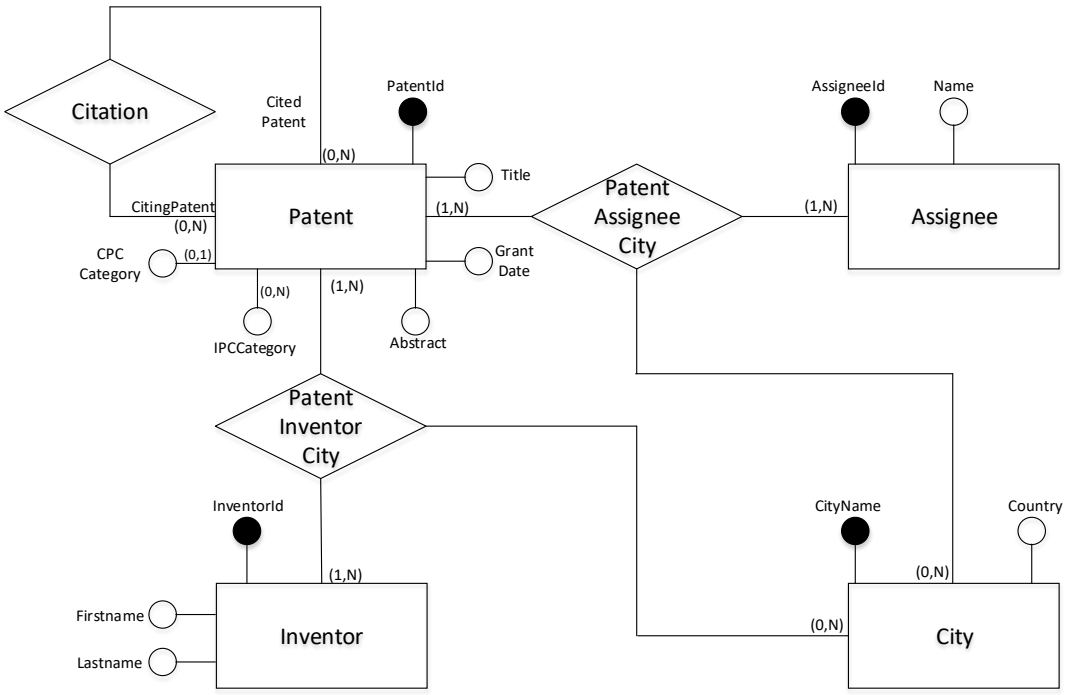
Relational schema
Patent (PatentId, Title, Summary, GrantDate, AssigneeId, AssigneeCityName)
PatentIPCCategory(PatentId, IPCCategoryName)
Assignee (AssigneeId, Name)
Inventor (InventorId, Name)
PatentInventorCity (PatentId, InventorId, CityName)
City (CityName, Country)

2. Schema integration

2a) Related concept identification + conflict analysis and resolution

PoliTwitter	UniTwitter	Conflict	Solution
Patent	Patent	Name conflicts	
		- Abstract → Summary	Abstract
		Cardinality conflicts	
		- A patent may have multiple assignees → A patent has just one assignee	A patent may have multiple assignees
Assignee	Assignee	Structure conflicts	
		- The city of the assignee is fixed → The city of the assignee may vary with the patent	The city of the assignee may vary with the patent
Inventor	Inventor	Structure conflicts	
		- The city of the inventor is fixed → The city of the inventor may vary with the patent	The city of the inventor may vary with the patent
		- Two distinct attributes for Firstname and Lastname → Just one attribute Name of the form *Firstname#Lastname*	Two distinct attributes for Firstname and Lastname

2b) Global conceptual schema



2c) Conceptual to logical translation

Patent (<u>PatentId</u>, Title, GrantDate, Abstract, CPCCategory*)
PatentIPCCategory(<u>PatentId</u>, <u>IPCCategoryName</u>)
City (<u>CityName</u>, Country)
Assignee (<u>AssigneeId</u>, Name)
Inventor (<u>InventorId</u>, Firstname, Lastname)
PatentAssigneeCity (<u>PatentId</u>, <u>AssigneeId</u>, CityName)
PatentInventorCity (<u>PatentId</u>, <u>InventorId</u>, CityName)
Citation (<u>CitingPatent</u>, <u>CitedPatent</u>)

3. Query answering and mapping definition

3a) Query formulation

Find the pairs (id of the patent, name of the assignee) associated with assignees from Milan and patents granted in 2017.

SELECT P.PatentId, A.Name
FROM Patent **AS** P, PatentAssigneeCity **AS** PAC, Assignee **AS** A
WHERE P.PatentId=PAC.PatentId **AND** PAC.AssigneeId=A.AssigneeId **AND** PAC.CityName=’Milan’ **AND** P.GrantDate **BETWEEN** ‘2017-01-01’ **AND** ‘2017-12-31’

3b) GAV mapping definition

```
CREATE VIEW UniPoliPatents.Patent (PatentId, Title, GrantDate, Abstract, CPCCategory) AS (
        SELECT PatentId, Title, GrantDate, Abstract, CPCCategory
        FROM PoliPatents.Patent

        UNION

        SELECT PatentId, Title, GrantDate, Summary, null
        FROM UniPatents.Patent
)


CREATE VIEW Assignee (AssigneeId, Name) AS (
        SELECT KeyGenAssignee(AssigneeId, 'PoliPatents'), Name
        FROM PoliPatents.Assignee

        UNION

        SELECT KeyGenAssignee(AssigneeId, 'UniPatents'), Name
        FROM UniPatents.Assignee
)


CREATE VIEW UniPoliPatents.PatentAssigneeCity (PatentId, AssigneeId, CityName) AS (
        SELECT PA.PatentId, KeyGenAssignee(A.AssigneeId, 'PoliPatents'), A.CityName
        FROM PoliPatents.PatentAssignee AS PA, PoliPatents.Assignee AS A
        WHERE PA.AssigneeId=A.AssigneeId

        UNION

        SELECT PatentId, KeyGenAssignee(AssigneeId, 'UniPatents'), AssigneeCityName
        FROM UniPatents.Patent
)
```

3c) Query rewriting

```
SELECT P.PatentId, A.Name
FROM PoliPatents.Patent AS P, PoliPatents.PatentAssignee AS PA, PoliPatents.Assignee AS A
WHERE P.PatentId=PA.PatentId AND PA.AssigneeId=A.AssigneeId AND A.CityName='Milan' AND
P.GrantDate BETWEEN '2017-01-01' AND '2017-12-31'

UNION

SELECT P.PatentId, A.Name
FROM UniPatents.Patent AS P, UniPatents.Assignee AS A
WHERE P.AssigneeId=A.AssigneeId AND P.AssigneeCityName='Milan' AND P.GrantDate BETWEEN '2017-
01-01' AND '2017-12-31'
```

Technologies for Information Systems

Part II (23 points)

prof. L. Tanca – September 7th, 2018

Available time: 2h 00m

Last Name _____

First Name _____

Student ID _____ **Signature**_____

PoliBeach is a company owning a chain of beach resorts. All the beach resorts owned by PoliBeach offer their customers the same set of packages, each associated with the rental of various objects like beach chairs and beach umbrellas. Each package has a daily rental cost, and the customer purchases the package for a certain number of days. The director of a beach resort may negotiate the sale price with the customers, deciding to apply a discount percentage that may be different for each sale.

The management of PoliBeach has now hired you to design a data warehouse to analyze the sales of the packages.

The following is the schema of the operational database used by PoliBeach:

CITY (CityName, Country)

BEACHRESORT (BeachResortId, BeachResortName, Director, CityName) *// A director may supervise multiple beach resorts.*

PACKAGE (PackageId, Description, Category, DailyPrice) *// The packages are classified in categories describing their luxury level (e.g., standard, premium, …). The daily price is expressed in EUR.*

AVAILABLERENTALOBJECT (ObjectName, Description) *// Examples of rental objects: beach chair, beach umbrella, …*

PACKAGECOMPOSITION (PackageId, RentalObjectName, Quantity) *// The packages are associated with the rental of various objects.*

CUSTOMER (CustomerId, Surname, GivenName, BirthDate, CityOfResidence)

SALE (CustomerId, PackageId, BeachResortId, StartDate, Duration, DiscountPercentage) *// Duration represents the number of days of the rental. DiscountPercentage is a number between 0 and 100.*

1. (3 points) Perform the reverse engineering of the given logical schema into a conceptual schema (Entity-Relationship model).

2. With respect to the produced ER diagram, discover the fact(s) that are useful specifically for answering the queries reported below. For each of these facts:
    a. (3 points) Produce the attribute tree (with pruning and grafting).
    b. (3 points) Produce the conceptual schema (fact schema).
    c. (2.5 points) Produce the glossary.

3. (3 points) Produce a logical schema consistent with the conceptual schema.

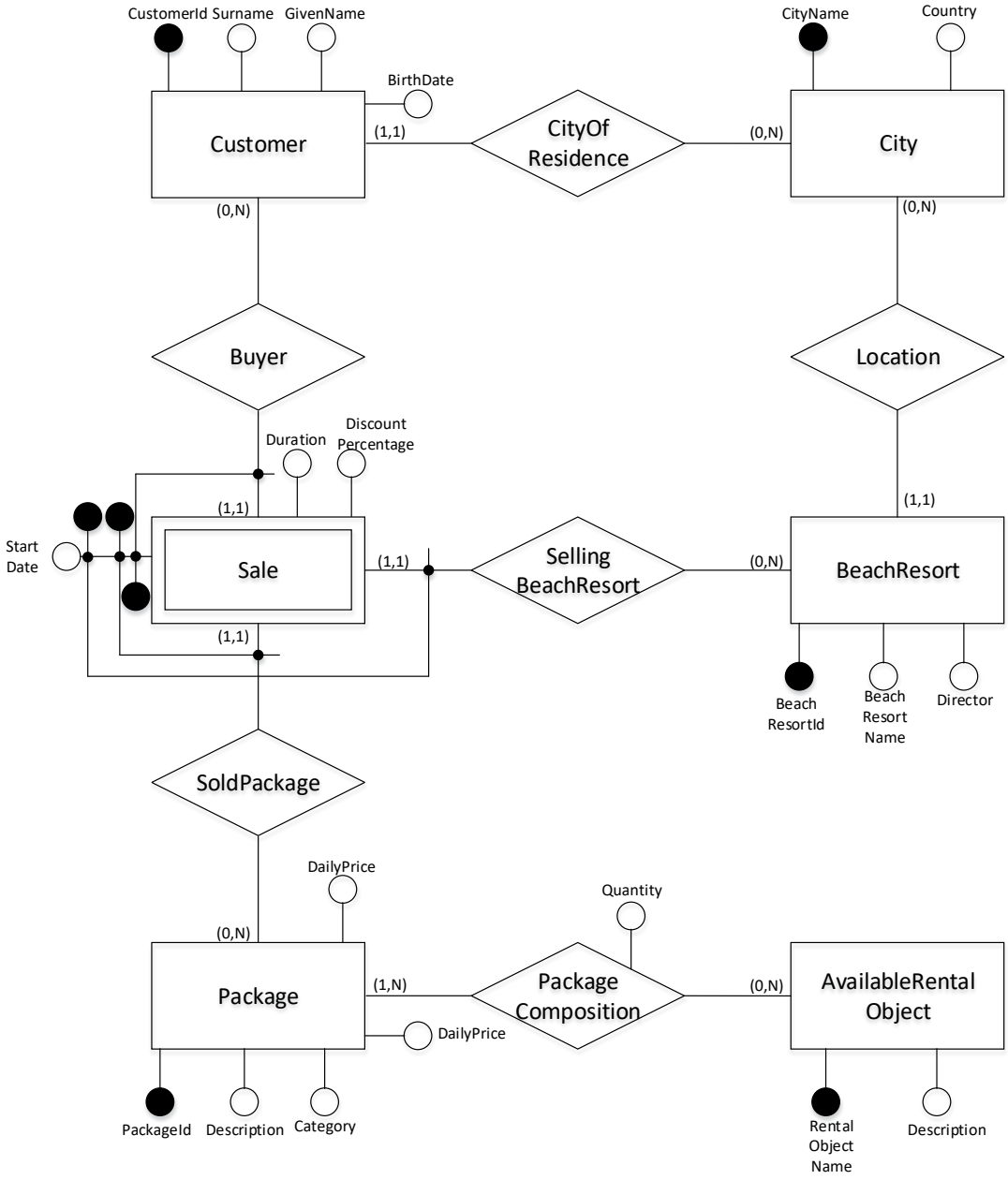
4. Write in SQL the following queries against the designed logical schema:
    a. (2 points) Compute the total number of sold packages for each package category, beach resort director and city of the customer. Include in the answer also the aggregations computed using only one and two of the three attributes.
    b. (2.5 points) Compute the average duration of the packages purchased by Italian customers born in 1989 for each city of the beach resort and package id.
    c. (2 points) Aggregate the total income by start date, month and year (include in the answer the aggregations computed only by start date, only by month and only by year).
    d. (2 points) Find for each country the beach resort(s) (specify id and name) having earned the greatest income.

NOTE

In SQL (PostgreSQL syntax), given a column of type Date named *d*, you can extract the year from the column using the function **EXTRACT**(**YEAR FROM** *d*).

SOLUTION

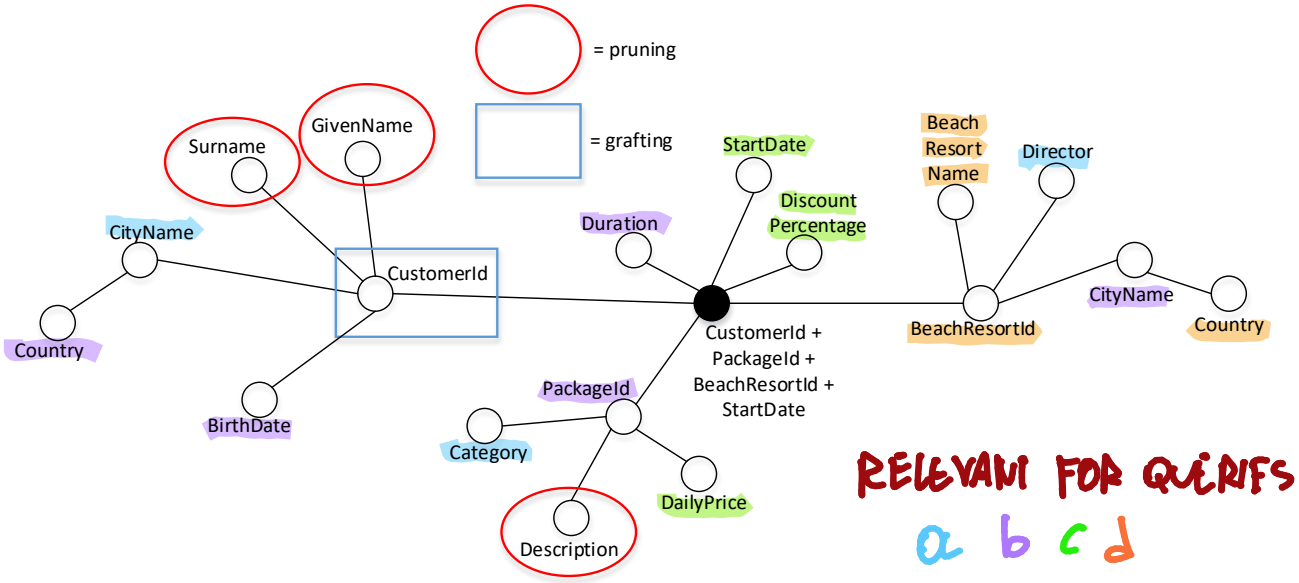
1. Reverse engineering



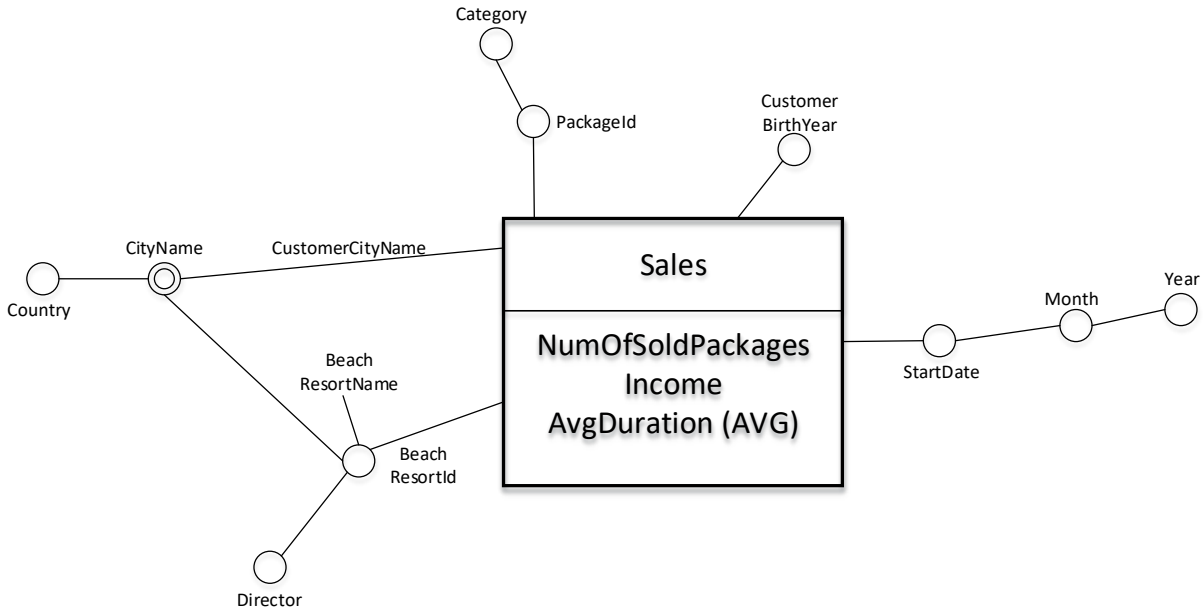
2. Conceptual design

Fact: Sales (Sale entity)

2a) Attribute tree



2b) Fact schema



2c) Glossary

NumOfSoldPackages

SELECT S.StartDate, S.BeachResortId, C.CityOfResidence, **EXTRACT**(**YEAR FROM** C.BirthDate), S.PackageId, **COUNT**(*)
FROM Sales **AS** S, Customer **AS** C
WHERE S.CustomerId=C.CustomerId
GROUP BY S.StartDate, S.BeachResortId, C. CityOfResidence, **EXTRACT**(**YEAR FROM** C.BirthDate), S.PackageId

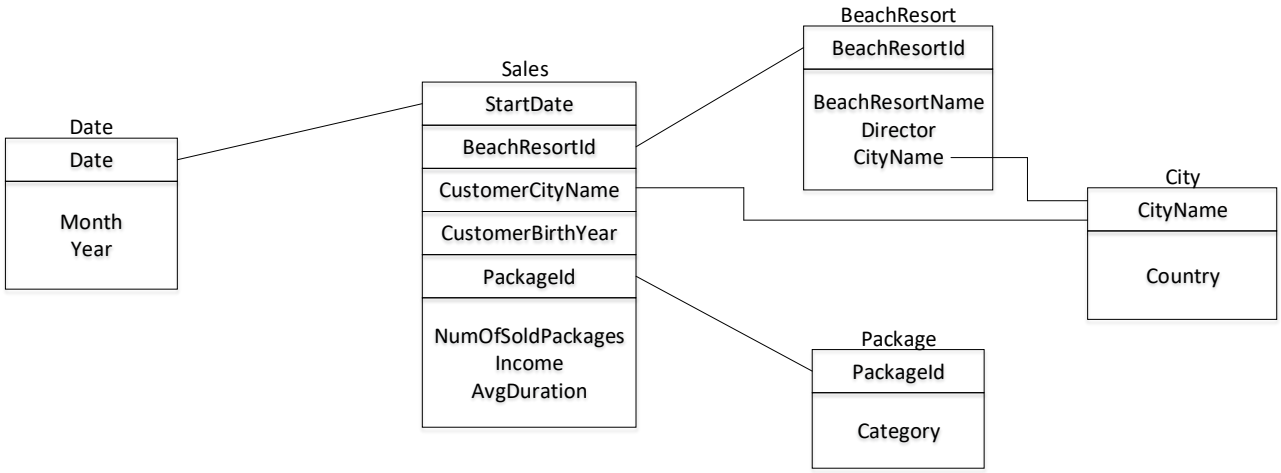
Income

```
SELECT S.StartDate, S.BeachResortId, C.CityOfResidence, EXTRACT(YEAR FROM C.BirthDate), S.PackageId,
        SUM(S.Duration*P.DailyPrice*(1-0.01*S.PercentageDiscount))
FROM Sales AS S, Customer AS C, Package AS P
WHERE S.CustomerId=C.CustomerId AND S.PackageId=P.PackageId
GROUP BY S.StartDate, S.BeachResortId, C. CityOfResidence, EXTRACT(YEAR FROM C.BirthDate),
        S.PackageId
```

AvgDuration

```
SELECT S.StartDate, S.BeachResortId, C. CityOfResidence, EXTRACT(YEAR FROM C.BirthDate), S.PackageId,
        AVG(S.Duration)
FROM Sales AS S, Customer AS C
WHERE S.CustomerId=C.CustomerId
GROUP BY S.StartDate, S.BeachResortId, C. CityOfResidence, EXTRACT(YEAR FROM C.BirthDate),
        S.PackageId
```

3. Logical design



4. Query answering

4a) Compute the total number of sold packages for each package category, beach resort director and city of the customer. Include in the answer also the aggregations computed using only one and two of the three attributes.

```
SELECT P.Category, B.Director, S.CustomerCityName, SUM(S.NumOfSoldPackages)
FROM Sales AS S, BeachResort AS B, Package AS P
WHERE S.BeachResortId=B.BeachResortId AND S.PackageId=P.PackageId
GROUP BY P.Category, B.Director, S.CustomerCityName WITH CUBE
```

4b) Compute the average duration of the packages purchased by Italian customers born in 1989 for each city of the beach resort and package id.

```
SELECT B.CityName, S.PackageId,
        SUM(S.NumOfSoldPackages *S.AvgDuration)/SUM(S.NumOfSoldPackages)
FROM Sales AS S, BeachResort AS B, City AS C
WHERE S.BeachResortId=B.BeachResortId AND S.CustomerCityName=C.CityName AND C.Country='Italy'
        AND S.CustomerBirthYear=1989
GROUP BY B.CityName, S.PackageId
```

4c) Aggregate the total income by start date, month and year (include in the answer the aggregations computed only by start date, only by month and only by year).

```
SELECT D.Year, D.Month, D.Date, SUM(S.Income)
FROM Sales AS S, Date AS D
WHERE S.StartDate=D.Date
GROUP BY D.Year, D.Month, D.Date WITH ROLLUP
```

4d) Find for each country the beach resort(s) (specify id and name) having earned the greatest income.

```
CREATE VIEW BeachResortIncome (BeachResortId, BeachResortName, Country, TotIncome) AS (
        SELECT B.BeachResortId, B.BeachResortName, C.Country, SUM(S.Income)
        FROM Sales AS S, BeachResort AS B, City AS C
        WHERE S.BeachResortId=B.BeachResortId AND B.CityName=C.CityName
        GROUP BY B.BeachResortId, B.BeachResortName, C.Country
)
SELECT B.Country, B.BeachResortId, B.BeachResortName
FROM BeachResortIncome AS B
WHERE B.TotIncome = (       SELECT MAX(B2.TotIncome)
                            FROM BeachResortIncome AS B2
                            WHERE B.Country=B2.Country
                    )
```