Lecture Notes # 3

It is the aim of this script to introduce relational data models and to develop skills in designing a data model.[1].

Data Model: Different Schemas

Data Model, a notation for describing data or information.

Different schemas are based on different concepts.

ER Diagram	CONCEPTUAL SCHEMA
	⇓
Relation	LOGICAL SCHEMA
	⇓
	PHYSICAL SCHEMA

Basics of the Relational Model

The relational model is based on a single data–modelling concept: the **relation**.

- Relation is a **mathematical concept**:

  relation **R** over $S_1, ..., S_n$ is a subset $\mathbf{R} \subseteq S_1 \times ... \times S_n$.

  For instance, given S_1 = "name" = {tomato, onion, carrot, pepperoni, celery}, $|S_1| = 5$, and given S_2 = "colour" = {red, white, yellow, orange}, $|S_2| = 4$, then R = "vegetable" = {(onion, red), (pepperoni, red), (pepperoni, yellow), (carrot, orange)}, $|R| = 4$, is a relation over $S_1 \times S_2$.

- Relation is a **two dimensional table**:

vegetable	
name	**colour**
onion	red
pepperoni	red
pepperoni	yellow
carrot	orange

- **Tables look like mathematical relations**.

- The **relational model** describes data in one two dimensional table called **relation**.

[1]The script is mainly based on "A First Course in Database Systems", J. Ullman, J. Widom. Mostly examples and exercises are created by the author of this script.

Relational Database Modeling

Attribute

Looking at the relation as a table, *Attributes* of a relation are the names of the columns of the table. An attribute describes the meaning of the entries in the column. For instance in the relation "vegetable", "name" and "colour" are attributes.

Schema

(a) The name of a relation and its set of attributes for the relation is called the **Schema** for that relation (or **relation schema**). The schema for the relation consists in the relation name followed by parenthesized set of attributes, comma separated (the attributes in a relation schema are a set, not a list).

(b) In the relational model, a database consists of one ore more relations. The set of schemas for the relations of a database is called a relational database schema, or just a **Database Schema**.

Hence, the resulting product consists in a **Database Schema**.

Course(name, courseType, difficulty, preparationTime)

Ingredient(name, description, category)

Recipe(Course.courseName, Ingredient.ingredientName, quantity)

Figure 1: Example 1: Book - Recipe

Menu(name, description, main)

Food(name, unit, weight, label, price, startDate, endDate, Menu.main_name)

GiftBasket(name, description)

BasketCombines(GiftBasket.name, Food.name, Food.unit, Food.weight)

Figure 2: Example 2: Online Market

Tuple

Looking at the relation as a table, the rows of a relation, excluding the header row containing the attribute names, are called **tuples**.

- A tuple has one value for each attribute of the relation. A single tuple, not in the relation, is written separating values by commas, within parenthesis, for instance `(pepperoni, yellow)`;
- Tuple uses the order in which the attributes are listed in the relation schema;
- Relation are set of tuples, not list of tuples;
- The set of tuples for a relation is called an *instance* for that relation;
- Even the *schema* is relatively immutable an *instance* may change frequently.

Domain

A domain is associated with attributes of a relation, that is some **elementary type** (numeric, char, date, ...). Hence the components of any tuple of the relation must have, in each component, a value that belongs to the domain of the corresponding column. **However domain could be limited to subset of elementary data type (integer $\geq$ 18, date of the current year, ...).**

In order to represent a missing or unknown value the special value "NULL" is added to the domain.

The domain can be introduced in the relation schema by the colon, as follows:

Vegetable(name: string, colour: string)

Domain: SQL Data Types

Primitive data types that are supported by SQL - Data Definition Language:

(1) **Character string:**

  (a) `CHAR(n)` denotes a **fixed** length string of up to `n` characters (short strings are padded by trailing blanks to make `n` characters).

  (b) `VARCHAR(n)` denotes a **varying** length string of up to `n` characters.

(2) **Integer number:**

  (a) `INT` or `INTEGER` (these names are synonyms) denotes integer numbers.

  (b) `SHORTINT` also denotes integers, but the identifiable values are less than `INT`.

(3) **Floating-point number:**

  (a) `FLOAT` or `REAL` (these names are synonyms) denotes floating-point numbers.

  (b) `DECIMAL(n, d)` or `NUMERIC(n, d)` (these names are synonyms) denotes floating-point numbers that consist of `n` decimal digits, with the decimal point assumed to be `d` position from the right. **n denotes the total allowed digits, hence not considering dot separator.**

(4) **Dates and Times:**

  (a) `DATE` denotes dates [`DATETIME`].

  (b) `TIME` denotes times [`DATETIME`].

  Date and time are character strings of special form. Date value is declared by `DATE` followed by a quoted string of special form, for example `DATE '2023-11-19'`, similarly time value is declared by `TIME` followed by a quoted string of special form, for example `TIME '14:30:00'`

(5) **Boolean**: `BOOLEAN` denotes a logical values `TRUE`, `FALSE` and `UNKNOWN`.

damiano.somenzi@unibo.it
Università Alma Mater - Stats&Maths

Key

A set of attributes forms a key for a relation if we do not allow two tuples in a relation instance to have the same values in all the attributes of the key. We indicate the attribute or attributes that form a key for a relation by underlining the key attribute(s). In many real-world databases artificial keys are used (i.e. `ID`) to have an attribute whose purpose is to serve as a key, when doubting about the values of the attributes.

Vegetable(name: string, colour: string)

One or more attributes $A_1A_2...A_n$ are a key for a relation R if:

- The attributes functionally determines all other attributes of the relation, this means that it is impossible for two distinct tuples of R to agree on all of $A_1A_2...A_n$;
- No proper subset of $A_1A_2...A_n$ functionally determines all other attributes of R, that is a key, must be **minimal**.

Exercises

Ex. #1 "The UNESCO Institute for Statistics (UIS) is the official and trusted source of internationally comparable data on education, science, culture and communication. Aiming to evaluate how international trade in cultural goods is relevant for Italian economics, we have selected three main domain for trade in the year **2016**."

Data are stored in an XML file.

```xml
<cultural-trade>
   <country>
      <name> Italy </name>
      <year> 2016 </year>
      <domain>
         <name> Performance and celebration goods </name>
         <export currency = "USD"> 204163335 </export>
         <import currency = "USD"> 482868210 </import>
      </domain>
      <domain>
         <name> Visual arts and crafts goods </name>
         <export currency = "USD"> 7370364440  </export>
         <import currency = "USD"> 2385773808 </import>
      </domain>
      <domain>
         <name> Books and press goods </name>
         <export currency = "USD"> 702612304  </export>
         <import currency = "USD"> 376969744 </import>
      </domain>
   </country>
</cultural-trade>
```

Assume to converge the data into a single "relation", that could be seen as **a data set**. Write **the relation** specifying for each identified attribute the corresponding **data domain** (refer to the SQL data types). [**Tip:** *Export* and *Import* header shall be assumed to be values not variables of the data set].

```
Trade(country: VARCHAR(50),
         year: INT,
       domain: VARCHAR(100),
        trade: VARCHAR(10),  [domain = {Import, Export}]
        value: NUMERIC(12,0),
     currency: CHAR(3)
     )
```

Ex. #2 "Every year, millions of people visit museums, from the most famous to the lesser known, representing a significant industry of a country's economy. The table compare the **number of museums and visitors in 2017** in some countries of the European Union, with different historical, cultural and geographical characteristics."

		# Museums			
Nation	Population	Art, Archaeology, History	Science and Technology	Others	Visitors
France	64.694.497	805	341	78	63.199.181
Italy	60.589.445	2.656	1.184	1.136	50.263.520

Table 1: Cultural Statistics - 2017

Assume to converge **# Museums** data into a single "relation" and write **the relation** specifying for each identified attribute the corresponding **data domain** (refer to the SQL data types).

```
StatMuseums(year: CHAR(4),
          nation: VARCHAR(50),
     description: VARCHAR(100), [Population, Visitors, ... ]
          number: NUMERIC(8,0),
)
```

Ex. #3 "Geo-databases are widely used to store geographical data. Differently to traditional DBMS a geo-database allows to manage geographic entities that can be represented as sets of points, lines, and polygons, imagery and rasters, continuous surfaces using terrain datasets."

Assume that in the process of database design the entity **parcel** of terrain is defined: attributes and general types.

Attribute Name	Data Type
Object ID	Numeric
Shape	**Geometry**
Parcel ID	Char
Parcel Name	Char
Parcel Type	Numeric
Shape Length	Numeric
Shape Area	Numeric

Write **the relation** specifying for each identified attribute the corresponding **data domain** (refer to the SQL data types). Observe the type **geometry**, which is specific of geo-databases, so select among **POINT**, **LINE**, **POLYGON** data type.

```
Parcel(object_ID: NUMERIC(3,0),
           shape: POLYGON,
       parcel_ID: CHAR(6),
     parcel_name: VARCHAR(30),
     parcel_type: INTEGER,
    shape_length: NUMERIC(10,5),
      shape_area: NUMERIC(10,5),
)
```

Ex. #4 "Currently the Web offers a variety of data, even if it is embodied into HTML code". Marketing needs to compare flight offers of different companies. Data is described by means of the XML language. [The second flight is active, and the economy options are displayed.]

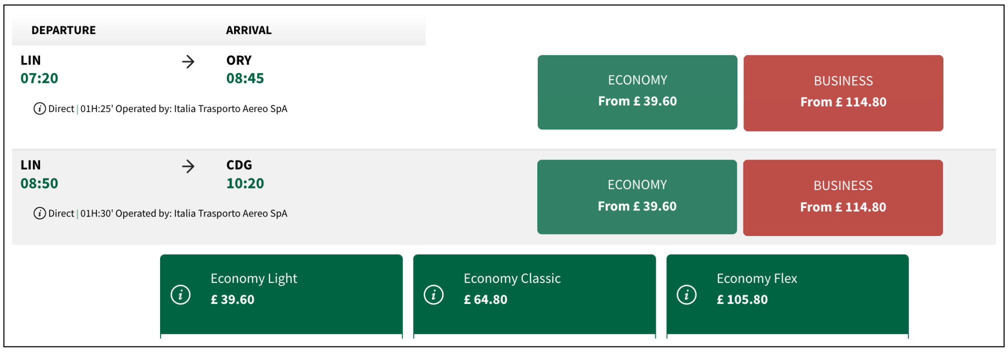


Figure 3: source: **www.itaspa.com**:

```
<flight-offers>
   <flight>
      <departure>
         <airport> LIN </airport>
         <time format="hh:mm"> 8:50 </time>
      </departure>
      <arrival>
         <airport> CGD </airport>
         <time format="hh:mm"> 10:20 </time>
      </arrival>
      <fare>
         <type> Economy </type>
         <offer>
            <name> Economy Light </name>
            <price unit="£"> 39.60 </price>
         </offer>
         <offer>
            <name> Economy Classic </name>
            <price unit="£"> 64.80 </price>
         </offer>
         <offer>
            <name> Economy Flex </name>
            <price unit="£"> 105.80 </price>
         </offer>
      </fare>
   </flight>
</flight-offers>
```

Now translate XML (semi-structured data model) into a table (structured data model). **Specify carefully for each identified attribute the corresponding data domain (refer to the SQL data types).**

```
Flight-offers(departure: CHAR(3),
                   time: TIME,
                arrival: CHAR(3),
                   time: TIME,
                   type: VARCHAR(20), [Economy, Business]
                   fare: VARCHAR(20), [Economy Light, Economy Classic, Economy Flex]
                  price: NUMERIC(6,2),
               currency: CHAR(3)
)
```

Ex. #5 Consider the following questions:

1. Given the relation that aims to hold data about kid cloths, identity for each attribute the corresponding data domain (refer to the SQL data type). **In braces are listed the acceptable values.**

```
Cloth(genre: {man, woman, kid}, type, date, price,
      save: {yes, no}, percentage {default 0},
      appreciation: {high, medium, low})
```

```
Cloth(genre: VARCHAR(5),
      type: VARCHAR(50),
      date: DATE,
      price: NUMERIC(5,2),
      save: BOOLEAN,
      percentage: NUMERIC(4,2),
      appreciation; VARCHAR(6),
```

2. Given the relation that aims to hold data about language courses, identity for each attribute the corresponding data domain (refer to the SQL data type). **In braces are listed the acceptable values.**

```
Course(title, language: {French, English, German},
      start: {date + time},
      end: {date + time}
      open: {yes, no} default no,
      level: {1, 2, 3, 4})
```

```
Course(title: VARCHAR(100),
       language: VARCHAR(7),
       start: DATETIME,
       end: DATETIME,
       open: BOOLEAN,
       level: INTEGER
)
```

Ex. #6 "Currently the Web shows a variety of data in effective way". Marketing needs to compare one night stay offers in luxury hotels. Consider the following data of a room availability:

Figure 4: source: **https://grandhotelmajestic.duetorrihotels.com/**

We have to register into a table mandatory: **Room nr.**, **year**, **month**, **dayOfWeek**, **day**, **workingDay**, **price**, and **availability**. Identify attributes data types (by reference to SQL datatypes).

```
Calendar(room_nr: INT,
         year: CHAR(4),
         month: VARCHAR(15),
         dayOfWeek: VARCHAR(10),
         day: INT,
         workingDay: BOOLEAN,
         price: NUMERIC(8,2),
         availability: BOOLEAN
)
```

From E/R Diagram to Relations

E/R model: E/R diagram $\Longrightarrow$ **Relational Model**: Relations

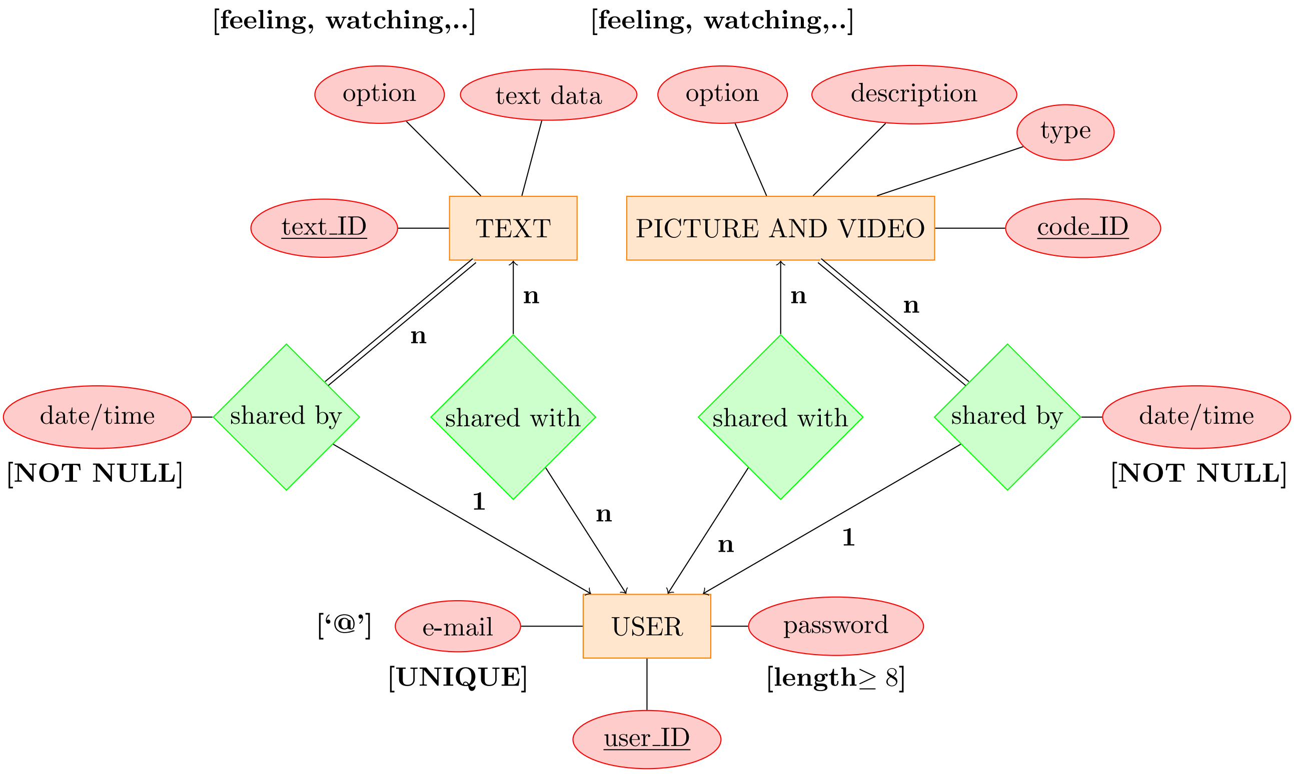


Figure 5: "Social" - E/R Diagram

From E/R Entity Set to Relation

- For each entity set depicted in the E/R diagram is created a relation schema of the same name and with the same set of attributes;
- In this phase the relation has not any indication of the relationships in which the entity set participates;
- For instance the entity set **USER** becomes User(user_ID, e-mail, password).

From E/R Relationships to Relation

E/R Relationships come to relations.

1. If the relationship is **many-one**, then all the attributes of the one side are uniquely identified by the key of the entity set, so the corresponding relation schema can consists in:

   (1) adding to the relation schema of the entity set of the *many* side, the key attributes of the entity set of the *one* side;

   (2) moreover any attributes belonging to the relationships;

   Notice: for an entity of the many side that is not related to any entity of the one side, then the attributes (1) and (2) have NULL values within the tuple;

2. If the relationship is **many-many**, then a relation schema is defined and consists in:

   (1) the key attributes of each entity set involved in the relationship;

   (2) moreover any attributes belonging to the relationship;

3. If the relationship is **one-one**, then all attributes of both sides can be determined uniquely by the key of each entity set, so the corresponding relation schema can consists in:

   (1) adding to the relation of the entity set of one side, the key attributes of the entity set of the other side or vice versa;

   (2) moreover any attributes belonging to the relationship;

4. A set of attributes in a relation schema that exactly matches the **key** in another relation is called the **foreign key** of the relation schema (the names of the attributes could not be exactly the same). We indicate the attribute or attributes that form the foreign key for a relation by dash-underlining the foreign key attribute(s).

5. If one entity set is involved several times in a relationship in different roles (E/R roles), then its key attributes appear in the relation schema many times as there are roles, **but renamed to avoid name duplication**.

6. For a **weak entity set** all its keys attributes must appear, including its "discriminator", and those of the other entity set that contribute to weak entity set key into a **Relation Schema**.

User(<u>user_ID</u>, email, password)

Text(<u>text_ID</u>, data, option, datetime, user.user_ID)

Picture_video(<u>code_ID</u>, description, option, type, datetime, user.user_ID)

Shared_text(text.text_ID, user.user_ID)

Shared_picture_video(picture_video.code_ID, user.user_ID)

The E/R diagram "Social" relates to the given database schema.

Conventionally relation names start with a capital letter, and attribute names begin with lower-case letter. Keys are underlined, while references are dash-underlined.

Example: “book - recipe”

1. **Conceptual Schema [E/R Diagram]**

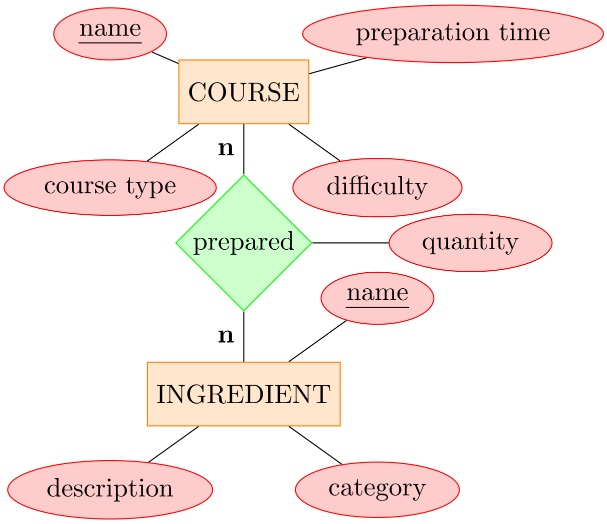


Figure 6: Recipe database schema

2. **Relational Model [Tables]**

   Course(name, courseType, difficulty, preparationTime)

   Ingredient(name, description, category)

   Recipe(Course.courseName, Ingredient.ingredientName, quantity)

3. **Physical Schema [Script file]**

```sql
CREATE DATABASE recipe;
USE recipe;

CREATE TABLE Course(
    name VARCHAR(100) PRIMARY KEY,
    courseType VARCHAR(100),
    difficulty VARCHAR(10),
    preparationTime NUMERIC(4)
);

CREATE TABLE Ingredient(
    name VARCHAR(100) PRIMARY KEY,
    description VARCHAR(50),
    category VARCHAR(20)
);

CREATE TABLE Recipe(
    courseName VARCHAR(100),
    ingredientName VARCHAR(100),
    quantity NUMERIC(4),
    FOREIGN KEY (courseName) REFERENCES course(name),
    FOREIGN KEY (ingredientName) REFERENCES Ingredient(name)
);
```

Example: "Online Market"

1. **Conceptual Schema [E/R Diagram]**

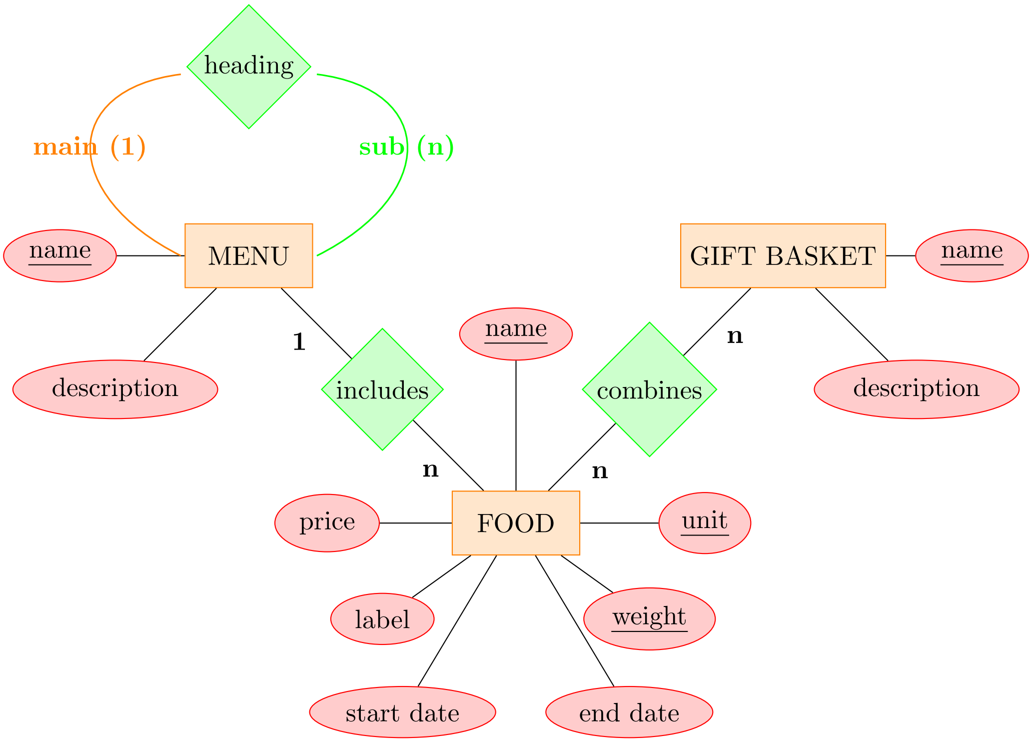


Figure 7: Online market database schema

2. **Relational Model [Tables]**

   Menu(name, description, main)

   Food(name, unit, weight, label, price, startDate, endDate, Menu.menu_name)

   GiftBasket(name, description)

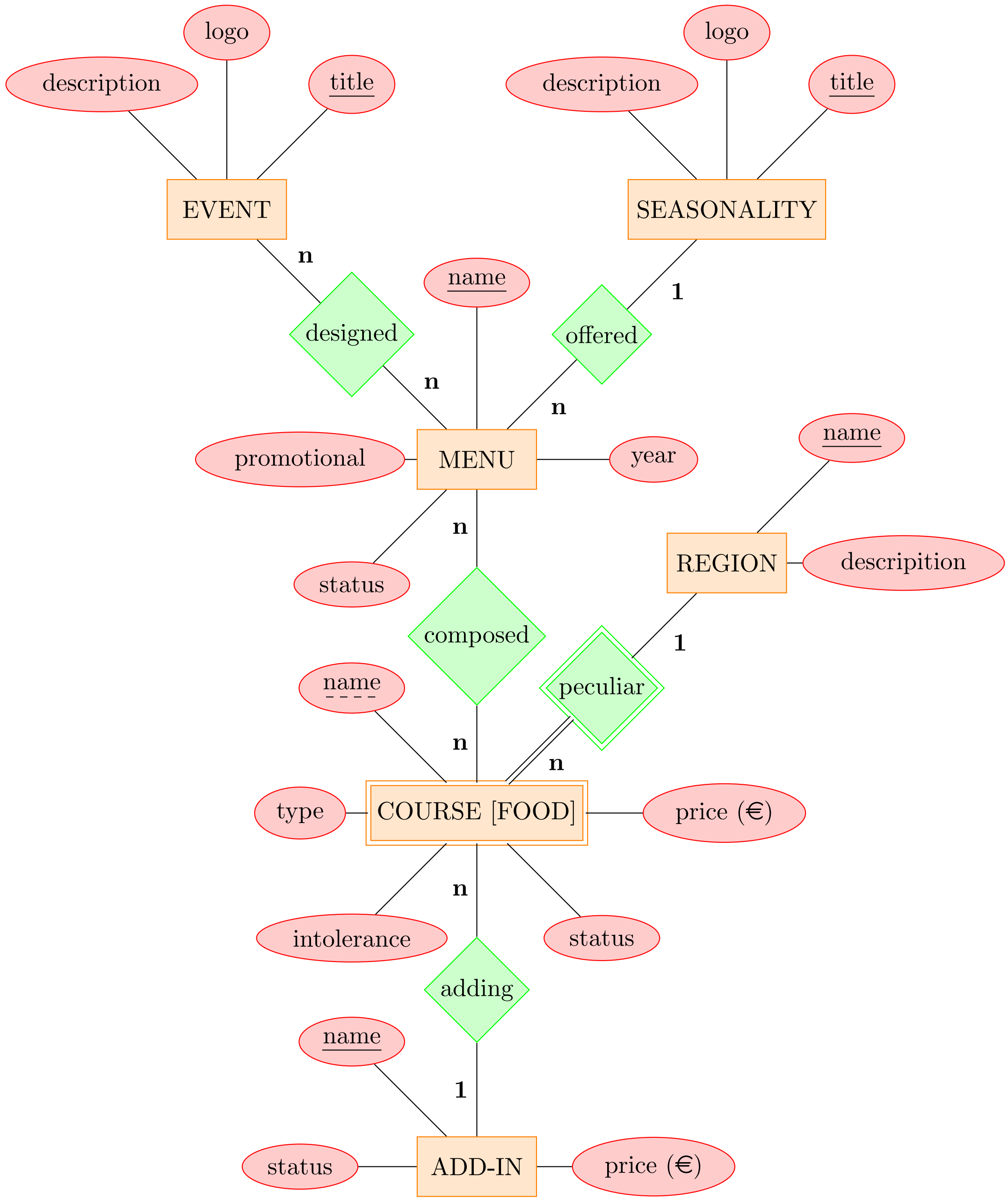
   BasketCombines(GiftBasket.name, Food.name, Food.unit, Food.weight)

Ex. # 1

(1) "*Gourmant-Food* is a modern restaurant which offers courses mainly based on traditional cooking food and regional products. Proposed menu - menù à la carte - changes depending on seasonality. However the restaurant offers special menù for particular events, in the restaurant or as catering service". The aim is to design a database to facilitate menù management.

Draw the E/R diagram that capture the requirements stated below:

1. The restaurant has created many different **menù**, assigning a name, sometimes a promotional - a short descriptive message -, when has been created - year -. The status specifies if the menù is available or not available anymore.
2. For each **seasonality** (springtime, estival, ...) there exist different menù, created selecting courses traditionally prepared in that season or simply using seasonal products. A season should be defined in the database by a title and further by a description and by an image - logo - that well identify the season. **A menù is typically related to one season**.
3. The restaurant designs special menù for particular **events**, for instance, anniversary party, congress buffet, ... An event should be defined in the database by a title and further by a description and an image - logo - that well identify the type of event.
4. A menù is designed identifying carefully courses which emphasize taste. A **course** has a name, a type (dessert, pasta, salad, ...) and a price, even intolerance information. The status specifies if the course is available or not available anymore.
5. Promoting traditional courses is the main restaurant goal, hence in the database **regions** are registered, with the name and general description.
6. Assuming that, a course is uniquely identified not only by means of its name, but also by the region it represents, model this fact by a suitable relationship.
7. Some courses can be "customized", adding no more than one additional ingredient (**add-in**: bread, french fries, oil dressing, whipped cream, ...). Each add-in has a name and a price. The status specifies if the ingredient is available or not available at the moment.


logo
description
title
EVENT
logo
description
title
SEASONALITY
n
designed
n
name
offered
1
n
promotional
MENU
year
name
n
status
REGION
descripition
composed
1
name
peculiar
n
n
type
COURSE [FOOD]
price (€)
n
intolerance
status
adding
name
1
status
ADD-IN
price (€)

(2) Translate the E/R digram into the corresponding "relations".

Seasonality(title, description, logo)

Menu(name, year, promotional, Seasonality.title)

Event(title, description, logo)

EventDesigned(Event.title, Menu.name)

Region(name, description)

AddIn(name, status, price)

Course(name, type, intolerance, status, price, Region.name, AddIn.name)

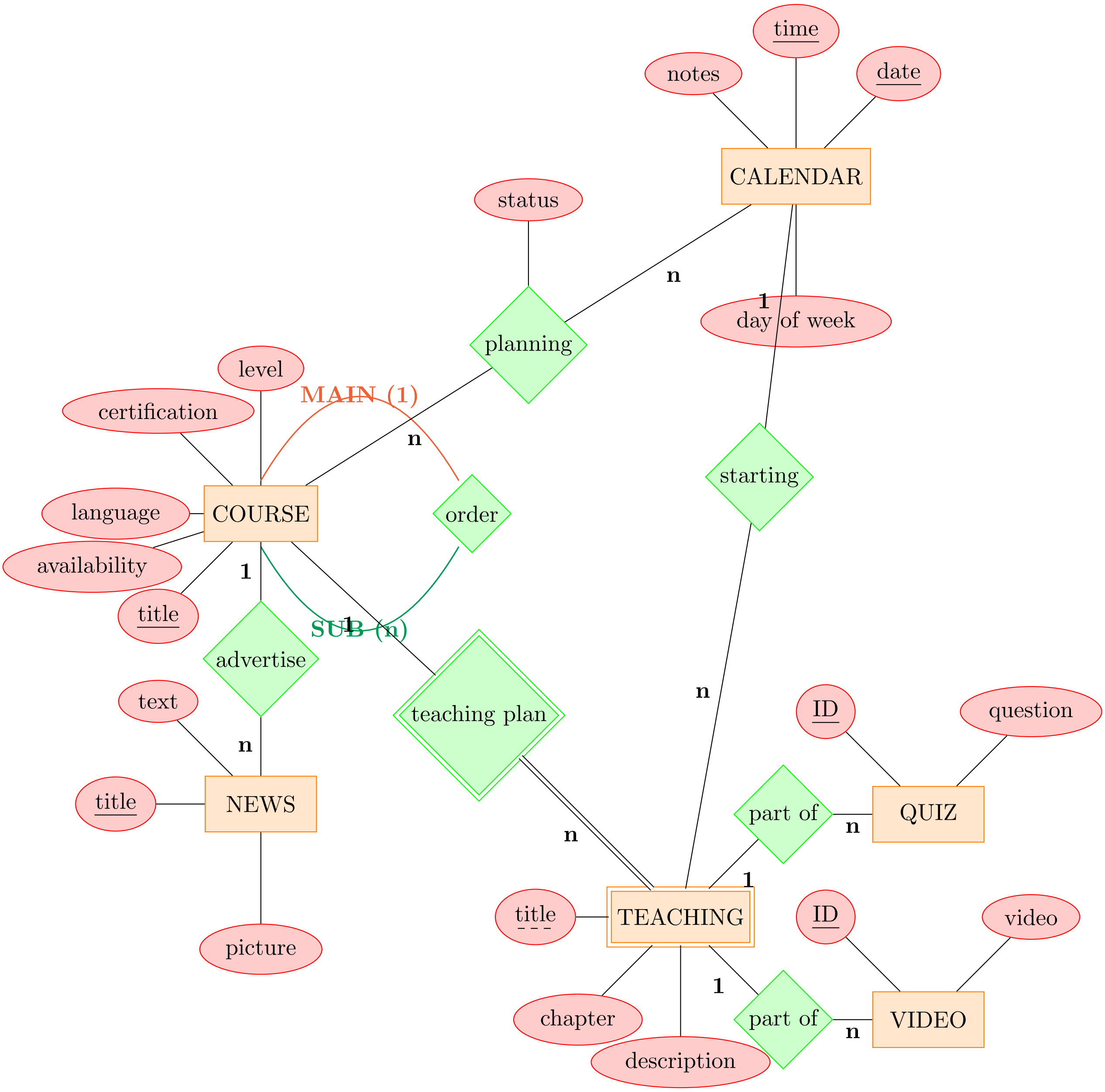
Composed(Menu.name, Course.name, Regione.name)

Ex. # 2

(1) "**openFUNlanguage**" application helps everybody interested in languages to learn and speak a language in few weeks. Open means that is free, and no registration is needed, just explore the available courses, select one and have fun to learn your favorite language. The goal is to design a database supporting the 'app'.

Draw the E/R diagram that capture the requirements stated below. **Use "ID" as key only if strictly necessary**.

1. A variety of language **courses** are available, given the title, the language offered, the level, if it leads to a certification or not, and status of availability {coming soon, open, on-going, archived}.
2. Generally courses are related themselves in order to define an order of suggested attendance (level: B1, B2, ...).
3. **News** could advertise a coming course, or report an announcement for the on-going ones. Generally news has a title, a text and a picture.
4. Each course has a planning based on a **calendar**. Assume an entity of a calendar as a tuple holding a date (**dd/mm/yyyy**), time, day of the week, notes.
5. A course is strictly related to time (calendar entity) in three different ways: (1) *starting*, when the course is becoming on-going (2) *ending*, when the course is becoming archived (3) *open*, 30 days before starting the user is entitled to browse most of the information (i.e. plan of teachings, type of resources, ...)
6. A plan of **teaching** activities: chapter, title, short description, is *weakly* released. For each chapter one or more **videos** and **quizzes** are available.
7. For each chapter is specified a starting time-point in the calendar.



(2) Write "relations" of the **Relational Data Model** which describe **News** and **Course**.

`Course(title, language, level, certification, availability, Course.title)`

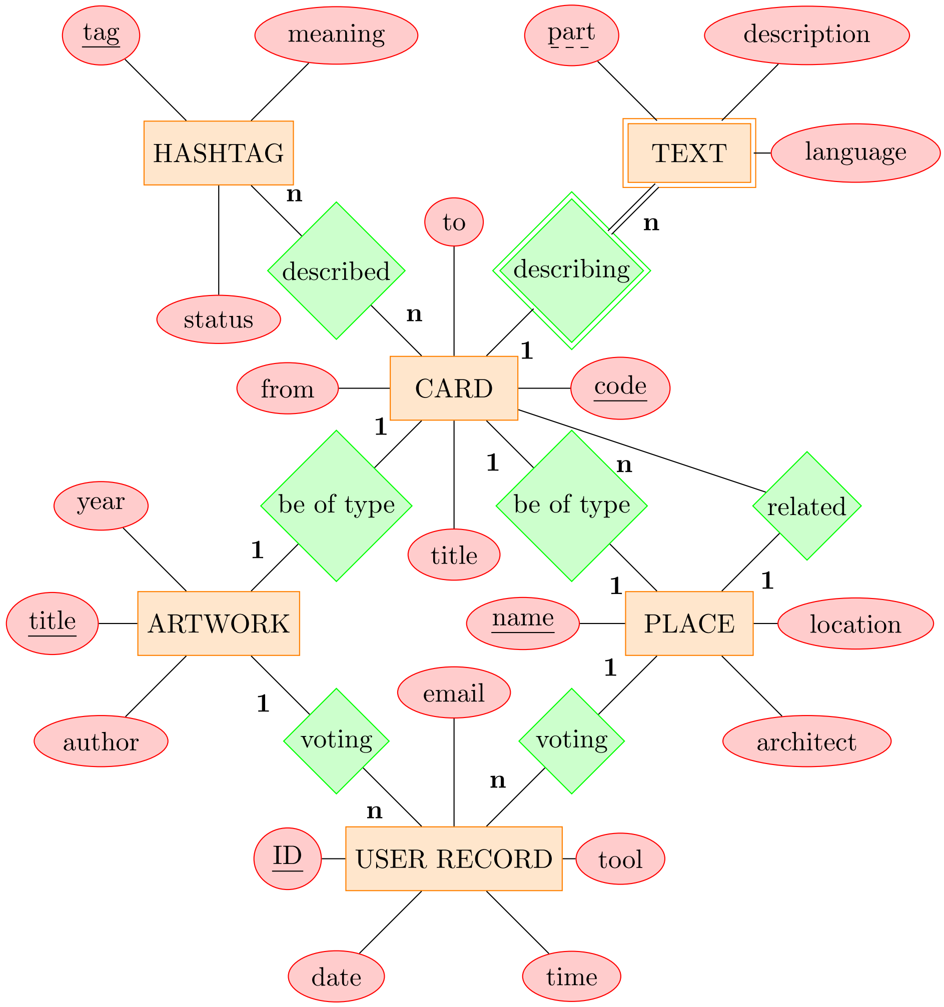
`News(title, text, picture, Course.title)`

Ex. # 3

(1) "Web, Socials, Whatsapp, are frequently used to collect opinions, song votes in a competition, customer satisfactions about a service, or generally appreciations, feelings,...". The goal is to promote historical buildings, gardens, artwork, enabling people to give a vote for 'appreciation, like, felling, ...' voting through a Web site.

Draw the E/R diagram that capture the requirements stated below. **Use "ID" as key only if strictly necessary**.

1. We generally submit people **cards** to vote, which are uniquely identified by a code, they have a title, and the period (from, to) people is allowed to vote it.
2. Each card could be identified by *one or more* **hashtag**, identified in the database, adding the meaning we have assigned to it and a status, that is if this hashtag is again in use or obsolete.
3. Cards could be for a **place**, like an historical building (castle, palace, church, ...) a garden, a natural park, described in the database by means of a common name, the location, the architect - if known.
4. Alternative cards could be for an **artwork**, described in the database by means of a title, author and a year of realization.
5. It could happen a card for a building be related to another item that is a garden or a painting or other ...
6. For all cards, is available a detailed **text** describing it. Specifically multi-language texts are allowed and available.
7. It could be allowed user to access to a Web site to vote by means of Facebook, Google profile (this implies a request of services to an external application!). Other solutions request to vote by means of Whatsapp, Facebook directly.
8. Any vote received through the listed tools is converted into a **user record**: an automatic ID is assigned, the tool used, and when (date + time). The relationships between a user record and a specific item: place or artwork represents **one vote**.



(2) Write the "relation" that describes all **texts**, then the SQL statement to `CREATE` the corresponding table.

1. Text(part: {title, desc., rank, nrVotes, ..}, description, language, cardCode)
2. ```
   CREATE TABLE Text(
       part VARCHAR(100),
       description VARCHAR(5000),
       language VARCHAR(30),
       cardCode CHAR(5),
       FOREIGN KEY (cardCode) REFERENCES Card(code),
       PRIMARY KEY (part, cardCode)
   );
   ```
   ```

References

1. J. Ullman, J. Widom, A First Course in Database Systems - Third Edition, Pearson - Prentice Hall
2. P. Atezeni et Altri, Basi di dati: modelli e linguaggi di interrogazione, McGraw Hill