Tutorial: Entity-Relationship Modelling

The Varsity International Network of Oenology wishes to computerise the management of the information about its members as well as to record the information they gather about various wines. Your company, Apasaja Private Limited, is commissioned by the Varsity International Network of Oenology to design and implement the relational schema of the database application. The organisation is big enough so that there could be several members with the same name. A card with a unique number is issued to identify each drinker. The contact address of each member is also recorded for the mailing of announcements and calls for meetings.

At most once a week, VINO organises a tasting session. At each session, the attending members taste several bottles. Each member records for each bottle his or her evaluation of the quality (very good, good, average, mediocre, bad, very bad) of each wine that she or he tastes. The evaluation may differ for the same wine from one drinker to another. Actual quality and therefore evaluation also varies from one to another bottle of a given wine. Every bottle that is opened during the tasting session is finished during that session.

Each wine is identified by its name ("Parade D'Amour"), appellation ("Bordeaux") and vintage (1990). Other information of interest about the wine is the degree of alcohol (11.5), where and by whom it has been bottled ("Mis en Bouteille par Amblard-Larolphie Negociant-Eleveur a Saint Andrede Cubzac (Gironde) - France"), the certification of its appellation if available ("Appellation Bordeaux Controlée"), and the country it comes from (produce of "France").

Generally, there are or have been several bottles of the same wine in the cellar. For each wine, the bottles in the wine cellar of VINO are numbered. For instance, the cellar has 20 bottles numbered 1 to 20 of a Semillon from 1996 named Rumbalara. For documentation purposes VINO may also want to record wines for which it does not own bottles. The bottles are either available in the cellar or they have been tasted and emptied.

We first want to design an entity-relationship schema that most correctly and most completely captures the constraints expressed in the above description of the VINO application.

1. Entity-relationship design.

   (a) Identify the entity sets. Justify your choice by quoting the sentences in the text that support it.

   > **Solution:** There are three entity sets: `member`, `bottle`, and `wine`.

   (b) Identify the relationship sets and the entity sets that they associate. Justify your choice by quoting the sentences in the text that support it.

   > **Solution:** There are two relationship sets: `taste` and `contain`. The relationship set taste links the entity set `member` with the entity set `bottle`. The relationship set contain links the entity set `wine` with the entity set `bottle`.

   (c) For each entity set and relationship set identify its attributes. Justify your choice by quoting the sentences in the text that support it.

Solution: See the solution.

(d) For each entity set, identify its keys.

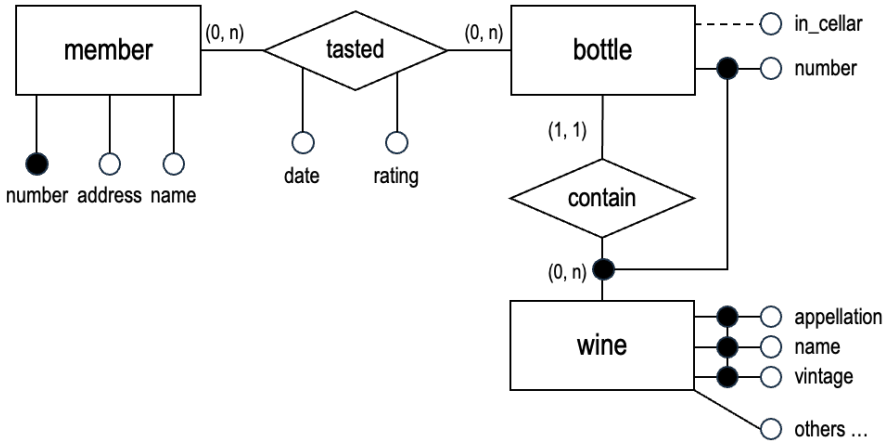
Solution: See the solution.

(e) For each entity set and each relationship set in which it participates, indicate the minimum and maximum participation constraints.

Solution: The relationship taste is an optional many-to-many relationship: both pairs of cardinality constraints are `(0, n)`.

The relationship `contain` is a one-to-many relationship from `wine` to `bottle`. It is mandatory for the bottle entities and optional for the optional for the `wine` entities. The participation constraints for the entity set `wine` is `(0, n)`. The participation constraints for the entity set `bottle` is `(1, 1)`. It could not be otherwise since the entity set `bottle` is a weak entity (is weakly identified under the relationship set `contain` and the dominant entity set `wine`).

(f) raw the corresponding entity-relationship diagram with the key and participation constraints. Indicate in English the constraints that cannot be captured, if any.

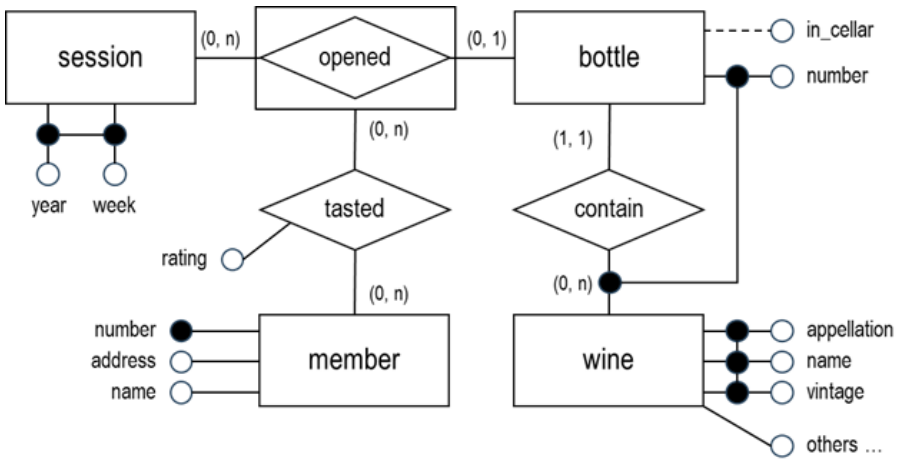


Solution:

The dotted line for the `in_cellar` attribute indicates that it is *calculated* (a bottle is in cellar if it was not drunk). It can be implemented as a **view**. If it is materialised the corresponding integrity constraints must be implemented. There are other constraints not captured such as the following.

- A session is held at most once a week.
- Every bottle that is opened during the tasting session is finished during that session (i.e., a bottle can only be opened in at most one session).

These constraints can be captured if we use **aggregate** below.



2. Logical design and SQL DDL code.

   (a) Translate your entity-relationship diagram into a relational schema. Give the SQL DDL statements to create the schema. Declare the necessary integrity constraints. Indicate in English the constraints that cannot be captured, if any.

Solution: The translation for solution **without aggregate** is shown below.

```
CREATE TABLE member (
  card_number  CHAR(10)     PRIMARY KEY,
  address      VARCHAR(64)  NOT NULL,
  name         VARCHAR(32)  NOT NULL
);
```

```
CREATE TABLE wine (
  name             VARCHAR(32),
  appellation      VARCHAR(32),
  vintage          DATE,
  alcohol_degree   NUMERIC NOT NULL,
  bottled          VARCHAR(128) NOT NULL,
  certification    VARCHAR(64),
  country          VARCHAR(32) NOT NULL,
  PRIMARY KEY (name, appellation, vintage)
);
```

```
CREATE TABLE bottle (
  wine_name        VARCHAR(32),
  appellation      VARCHAR(32),
  vintage          DATE,
  number           INTEGER NOT NULL CHECK (number > 0),
  PRIMARY KEY (number, wine_name, appellation, vintage),
  FOREIGN KEY (wine_name, appellation, vintage)
    REFERENCES wine (name, appellation, vintage)
);
```

```
CREATE TABLE tasted (
  wine_name       VARCHAR(32),
  appellation     VARCHAR(32),
  vintage         DATE,
  bottle_number   INTEGER,
  member          CHAR(10),
  tasting_date    DATE NOT NULL,
  rating          VARCHAR(32) NOT NULL,
    REFERENCES member (card_number),
  PRIMARY KEY (member, bottle_number, wine_name, appellation, vintage),
  FOREIGN KEY (bottle_number, wine_name, appellation, vintage)
    REFERENCES bottle (bottle_number, wine_name, appellation, vintage)
);
```

The translation for solution **with aggregate** is shown below.

```
CREATE TABLE member (
  card_number  CHAR(10)     PRIMARY KEY,
  address      VARCHAR(64)  NOT NULL,
  name         VARCHAR(32)  NOT NULL
);
```

```
CREATE TABLE wine (
  name             VARCHAR(32),
  appellation      VARCHAR(32),
  vintage          DATE,
  alcohol_degree   NUMERIC NOT NULL,
  bottled          VARCHAR(128) NOT NULL,
  certification    VARCHAR(64),
  country          VARCHAR(32) NOT NULL,
  PRIMARY KEY (name, appellation, vintage)
);
```

```
CREATE TABLE session (
  year  INTEGER,
  week  INTEGER,
  PRIMARY KEY (year, week)
);
```

```
CREATE TABLE bottle (
  wine_name       VARCHAR(32),
  appellation     VARCHAR(32),
  vintage         DATE,
  number          INTEGER CHECK (number > 0),
  PRIMARY KEY (number, wine_name, appellation, vintage),
  FOREIGN KEY (wine_name, appellation, vintage)
    REFERENCES wine (name, appellation, vintage)
);
```

```
CREATE TABLE opened (
  wine_name       VARCHAR(32),
  appellation     VARCHAR(32),
  vintage         DATE,
  bottle_number   INTEGER,
  session_year    INTEGER NOT NULL,
  session_week    INTEGER NOT NULL,
  PRIMARY KEY (bottle_number, wine_name, appellation, vintage),
  FOREIGN KEY (session_year, session_week)
    REFERENCES session (year, week),
  FOREIGN KEY (bottle_number, wine_name, appellation, vintage)
    REFERENCES bottle (number, wine_name, appellation, vintage)
);
```

```
CREATE TABLE tasted (
  wine_name      VARCHAR(32),
  appellation    VARCHAR(32),
  vintage        DATE,
  bottle_number  INTEGER,
  member         CHAR(10),
  rating         VARCHAR(32) NOT NULL,
    REFERENCES member (card_number),
  PRIMARY KEY (member, bottle_number, wine_name, appellation, vintage),
  FOREIGN KEY (bottle_number, wine_name, appellation, vintage)
    REFERENCES opened (bottle_number, wine_name, appellation, vintage)
);
```

References

[1] P. Atzeni, S. Ceri, S. Paraboschi, and R. Torlone. *Database Systems - Concepts, Languages and Architectures*. `http://dbbook.dia.uniroma3.it/`. Visited on 29 December 2021.

[2] S. Bressan and B. Catania. *Introduction to Database Systems*. McGraw-Hill Education, 2006.

[3] H. Garcia-Molina, J.D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Pearson international edition. Pearson Prentice Hall, 2009.

[4] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, 2002.