Introduction to Databases

Exam of 25/08/2020

With Solutions

Diegoo Calvanese

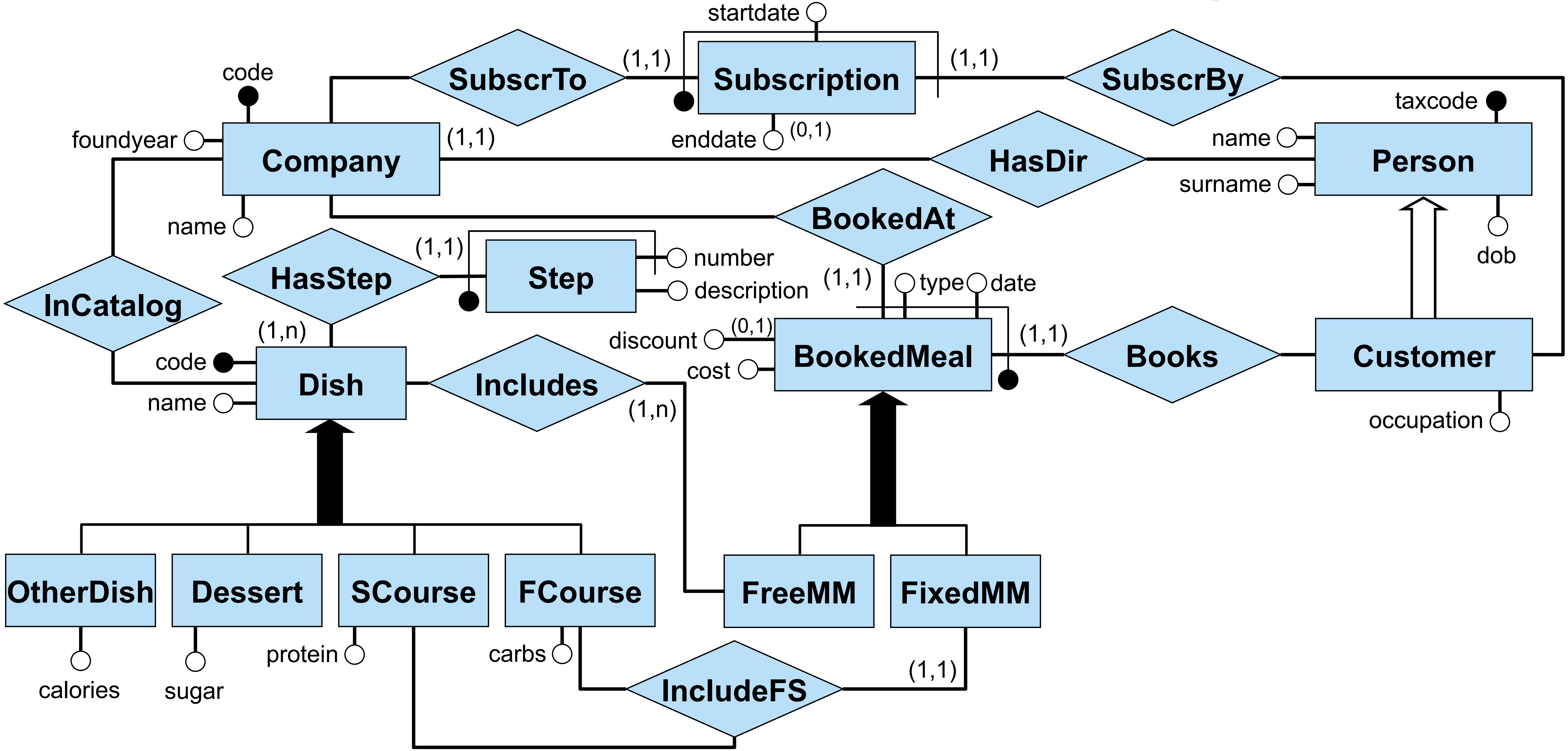
Bachelor in Computer Science
Faculty of Computer Science
Free University of Bozen-Bolzano

http://www.inf.unibz.it/~calvanese/teaching/exams/idb/

Problem 1

Design the Entity-Relationship schema of an information system relating to food supply companies. Of each **company**, we are interested in the code (identifier), the name, the year of foundation, the person who directs it and the dishes that the company has in its catalog. Each **dish** is identified by a code and is characterized by the name and by the steps of the recipe used to prepare it (at least one). Each *recipe step* of a dish is identified by a sequence number and has associated a description. There are exactly 4 types of dishes: first course, second course, dessert, and other. Of each **first course** we are interested in the amount of carbohydrates it contains, of each **second** *course* in the type of protein it contains, of each **dessert** in the amount of sugar it contains, and of each *other dish* in the amount of calories it contains. Each **person** is identified by the tax code and are characterized by their name, surname, and date of birth. **Customers** are persons who book meals with food supply companies, and we are interested in their current occupation. When a customer books a meal of a certain type (for example, lunch, dinner, snack, aperitif, etc.) with a company for a certain date, the company establishes the cost of the **booked meal**. Please note that a client cannot book more than one meal of the same type with the same company and for the same date. There are two types of booked meals, depending on the set of dishes they are composed of: fixed menu meals and free menu meals. The **fixed menu meals** include a first course and a second course, while the **free menu meals** can include any set of dishes (at least one). A customer can subscribe to a company. Of each **subscription**, the start date is of interest, and also the end date, but only if the subscription has been terminated. Notice that a customer can subscribe at most once to the same company in a given period. For meals booked by customers who are currently subscribers of the company at which they booked the meal, we are interested in the discount applied to the meal.

Problem 1: Conceptual schema – Diagram



Problem 1: Conceptual schema – External constraints

1) Constraint on whether the value of the optional attribute **discount** is defined or not:
For each instance ***I*** of the schema and for each $m \in$ *instances*(I,BookedMeal),
there is (m,d) $\in$ *instances*(I,discount)
if and only if
there are $p \in$ *instances*(I,Customer), $s \in$ *instances*(I,Subscription), and $c \in$ *instances*(I,Company) such that
(Customer:p, BookedMeal:m) $\in$ *instances*(I,Books), (Subscription:s, Customer:p) $\in$ *instances*(I,SubscrBy),
(Subscription:s, Company:c) $\in$ *instances*(I,SubscrTo), and
(BookedMeal:m, Company:c) $\in$ *instances*(I,BookedAt).
Note: As far as the exam is concerned, it suffices to specify the above constraint in natural language.

2) Constraint expressing that a customer can subscribe at most once to the same company in a given period:
For each instance ***I*** of the schema, there are no instances s_1 and s_2 in *instances*(I,Subscription) such that
(Subscription:s_1, Customer:p) and (Subscription:s_2, Customer:p) are in *instances*(I,SubscrBy), for some p,
(Subscription:s_1, Company:c) and (Subscription:s_2, Company:c) are in *instances*(I,SubscrTo), for some c, and
(s_1,sd_1) and (s_2,sd_2) in *instances*(I,startdate), for some sd_1 *and* sd_2 such that $sd_1 \leq sd_2$ and where
either (s_1,ed_1) $\in$ *instances*(I,enddate), for some $ed_1 \geq sd_2$, or there is no (s_1,ed_1) $\in$ *instances*(I,enddate).
Note: As far as the exam is concerned, it suffices to specify the above constraint in natural language.

Problem 2

Carry out the logical design of the database, producing the complete relational schema with constraints, taking into account the following indications:

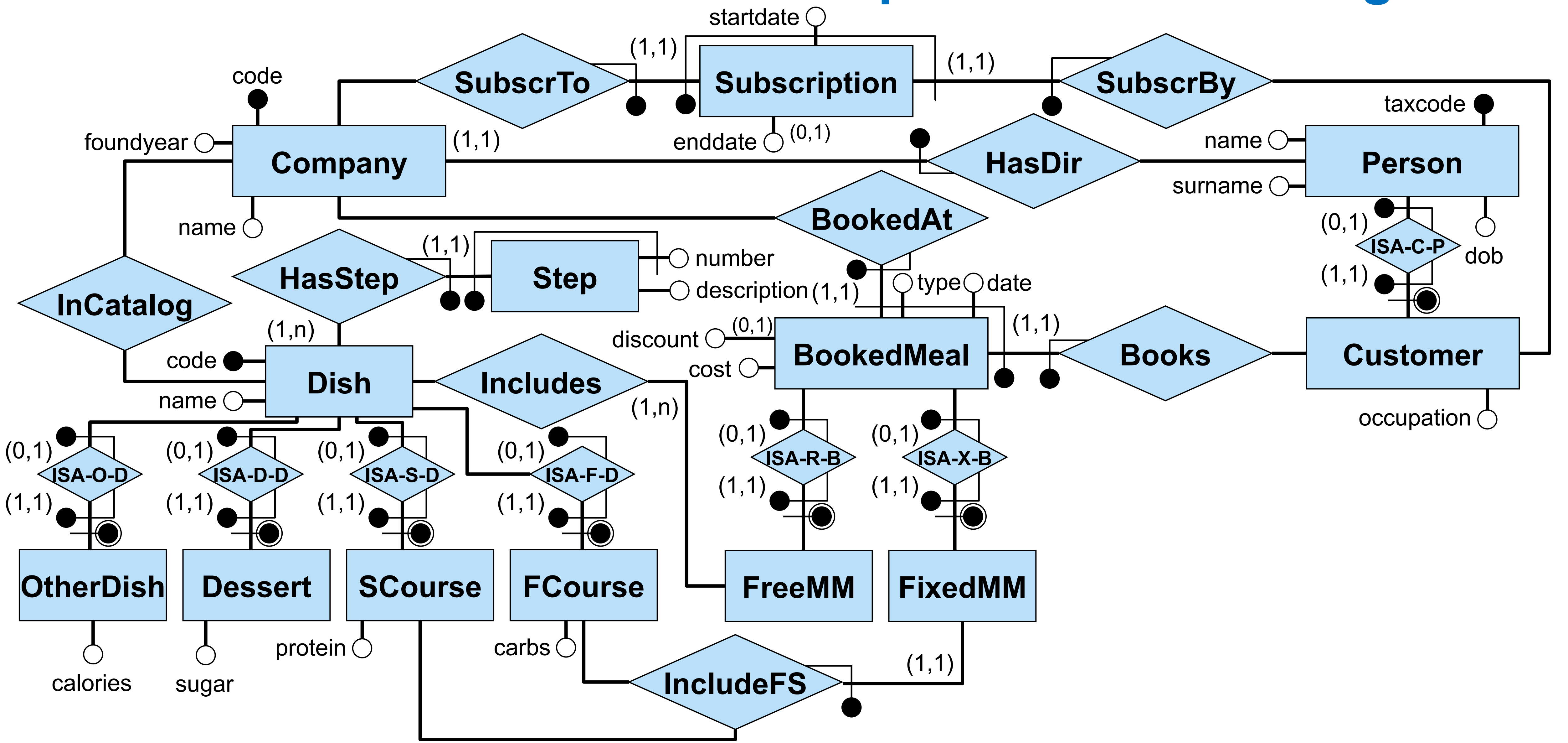
1. We want to avoid null values in the database.
2. When accessing a fixed menu meal, we always want to know which are the corresponding first course and second course.

As steps in your design you should produce:

- the restructured ER schema (possibly with external constraints),
- the direct translation into the relational model (possibly with external constraints), and
- the restructured relational schema (again with constraints).

Motivate explicitly how the above indications affect your design.

Problem 2: Restructured conceptual schema – Diagram



Problem 2: Restructured conceptual schema – External constraints

1) The constraint concerning whether the value of the optional attribute **discount** is defined or not does not need to be reformulated in the restructured schema, and can be kept identical.

2) The constraint expressing that a customer can subscribe at most once to the same company in a given period does not need to be reformulated in the restructured schema, and can be kept identical.

3) Generalization constraint resulting from the elimination of the complete generalization hierarchy for **BookedMeal**:
   For each instance of the conceptual schema, each instance of **BookedMeal** participates either to **ISA-R-B** or to **ISA-X-B**, but not to both.

4) Generalization constraint resulting from the elimination of the complete generalization hierarchy for **Dish**:
   For each instance of the conceptual schema, each instance of **Dish** participates to exactly one of **ISA-F-D**, **ISA-S-D**, **ISA-D-D**, or **ISA-O-D**.

Problem 2: Direct translation (1/3)

Company(code, name, foundyear)
  foreign key: Company[code] $\subseteq$ HasDir[company]

Dish(code, name)
  inclusion: Dish[code] $\subseteq$ Step[dish]
  generalization constraint: Dish[code] $\subseteq$ FCourse[code] $\cup$ SCourse[code] $\cup$ Dessert[code] $\cup$ OtherDish[code]

FCourse(code, carbs)
  foreign key: FCourse[code] $\subseteq$ Dish[code]

SCourse(code, protein)
  foreign key: SCourse[code] $\subseteq$ Dish[code]

Dessert(code, sugar)
  foreign key: Dessert[code] $\subseteq$ Dish[code]

OtherDish(code, calories)
  foreign key: OtherDish[code] $\subseteq$ Dish[code]

Step(dish, number, description)
  inclusion: Dish[code] $\subseteq$ Step[dish]

Problem 2: Direct translation (2/3)

InCatalog(<u>company</u>, <u>dish</u>)
  foreign key: InCatalog[company] ⊆ Company[code]
  foreign key: InCatalog[dish] ⊆ Dish[code]

Person(<u>taxcode</u>, name, surname, dob)

HasDir(<u>company</u>, <u>person</u>)
  foreign key: HasDir[company] ⊆ Company[code]
  foreign key: HasDir[person] ⊆ Person[taxcode]

Customer(<u>taxcode</u>, occupation)
  foreign key: Customer[taxcode] ⊆ Person[taxcode]

Subscription(<u>customer</u>, <u>company</u>, <u>startdate</u>, enddate*)
  foreign key: Subscription[customer] ⊆ Customer[taxcode]
  foreign key: Subscription[company] ⊆ Company[code]

BookedMeal(<u>customer</u>, <u>company</u>, <u>type</u>, <u>date</u>, cost, discount*)
  foreign key: BookedMeal[customer] ⊆ Customer[taxcode]
  foreign key: BookedMeal[company] ⊆ Company[code]
  generalization constraint: BookedMeal[customer,company,type,date] ⊆
    FixedMM[customer,company,type,date] ∪ FreeMM[customer,company,type,date]

Problem 2: Direct translation (3/3)

FixedMM(<u>customer</u>, <u>company</u>, <u>type</u>, <u>date</u>)

foreign key: FixedMM[customer,company,type,date] $\subseteq$ BookedMeal[customer,company,type,date]

foreign key: FixedMM[customer,company,type,date] $\subseteq$ IncludeFS[customer,company,type,date]

IncludeFS(<u>customer</u>, <u>company</u>, <u>type</u>, <u>date</u>, fcourse, scourse)

foreign key: IncludeFS[customer,company,type,date] $\subseteq$ FixedMM[customer,company,type,date]

foreign key: IncludeFS[fcourse] $\subseteq$ FCourse[code]

foreign key: IncludeFS[scourse] $\subseteq$ SCourse[code]

FreeMM(<u>customer</u>, <u>company</u>, <u>type</u>, <u>date</u>)

foreign key: FreeMM[customer,company,type,date] $\subseteq$ BookedMeal[customer,company,type,date]

Includes(<u>customer</u>, <u>company</u>, <u>type</u>, <u>date</u>, <u>dish</u>)

foreign key: Includes[customer,company,type,date] $\subseteq$ FreeMM[customer,company,type,date]

foreign key: Includes[dish] $\subseteq$ Dish[code]

Problem 2: Direct translation – External constraints (1/2)

1) The constraint on whether the value of the optional attribute **discount** is defined or not, can be expressed through the following relational algebra query, which must evaluate to **false** (i.e., it retrieves violations of the constraint):

$\text{PROJ}_{\text{customer,company,type,date}}\ (\text{SEL}_{\text{discount IS NULL}}\ ($
$\quad ((\text{BookedMeal JOIN}_{\text{customer=taxcode}}\ \text{Customer})\ \text{JOIN}_{\text{taxcode=cus}}$
$\qquad \text{REN}_{\text{cus} \leftarrow \text{customer, com} \leftarrow \text{company}}\ (\text{Subscription}))\ \text{JOIN}_{\text{com=code AND code=company}}\ \text{Company}))$
$\cup\ \text{PROJ}_{\text{customer,company,type,date}}\ (\text{SEL}_{\text{discount IS NOT NULL}}\ ($
$\qquad \text{PROJ}_{\text{customer,company,type,date,discount}}\ (\text{BookedMeal})\ –$
$\qquad \text{PROJ}_{\text{customer,company,type,date,discount}}\ ($
$\qquad\quad ((\text{BookedMeal JOIN}_{\text{customer=taxcode}}\ \text{Customer})\ \text{JOIN}_{\text{taxcode=cus}}$
$\qquad\qquad \text{REN}_{\text{cus} \leftarrow \text{customer, com} \leftarrow \text{company}}\ (\text{Subscription}))\ \text{JOIN}_{\text{com=code AND code=company}}\ \text{Company})))$

Note: As far as the IDB exam is concerned, it is sufficient to specify that the constraint needs to be expressed on the relational schema, but there is no need to specify it in terms of a relational algebra query.

Problem 2: Direct translation – External constraints (2/2)

2) The constraint that a customer can subscribe at most once to the same company in a given period, can be expressed through the following relational algebra query, which must evaluate to **false** (i.e., it retrieves violations of the constraint):

$\text{SEL}_{\text{startdate} \leq \text{start2 AND (start2} \leq \text{end OR end IS NULL)}}($
$\quad\text{Subscription JOIN}_{\text{customer = cus2, company = com2}}$
$\quad\text{REN}_{\text{cus2} \leftarrow \text{customer, com2} \leftarrow \text{company, start2} \leftarrow \text{startdate, end2} \leftarrow \text{enddate}}(\text{Subscription}))$

Note: As far as the IDB exam is concerned, it is sufficient to specify that the constraint needs to be expressed on the relational schema, but there is no need to specify it in terms of a relational algebra query.

3-4) The generalization constraints resulting from the elimination of the complete hierarchies for **BookedMeal** and for **Dish** are already expressed in the schema.

Problem 2: Restructuring of the relational schema

1. We want to avoid null values in the database.
2. When accessing a fixed menu meal, we always want to know which are the corresponding first course and second course.

We take into account the above indications in the following way:

- We take into account indication 1 by:
  - performing a horizontal partition of **Subscription** into two relations containing respectively the subscriptions that have and have not been terminated. We need also to reformulate foreign key constraints towards **Subscription** into foreign keys towards the two new relations.
  - performing a horizontal partition of **BookedMeal** into two relations containing respectively the booked meals that have and do not have a discount applied to them.
- We take into account indication 2 by merging the relation **FixedMM** with the relation **IncludeFS**. The mutual foreign keys between **FixedMM** and **IncludeFS** disappear, and the foreign keys in **IncludeFS** to **FCource** and **SCourse** become foreign keys in **FixedMM**.

Problem 2: Restructured relational schema (1/2)

We specify here only the relations with their constraints that have been changed with respect to the schema obtained through the direct translation.

SubscriptionNT(customer, company, startdate, enddate)
- foreign key: SubscriptionNT[customer] $\subseteq$ Customer[taxcode]
- foreign key: SubscriptionNT[company] $\subseteq$ Company[code]

SubscriptionT(customer, company, startdate)
- foreign key: SubscriptionT[customer] $\subseteq$ Customer[taxcode]
- foreign key: SubscriptionT[company] $\subseteq$ Company[code]
- disjointness: SubscriptionNT[customer,company,startdate] $\cap$ SubscriptionT[customer,company,startdate] = $\emptyset$

BookedMeal(customer, company, type, date, cost)
- foreign key: BookedMeal[customer] $\subseteq$ Customer[taxcode]
- foreign key: BookedMeal[company] $\subseteq$ Company[code]

BookedMealDiscounted(customer, company, type, date, cost, discount)
- foreign key: BookedMeal[customer] $\subseteq$ Customer[taxcode]
- foreign key: BookedMeal[company] $\subseteq$ Company[code]
- generalization constraint:
  BookedMeal[customer,company,type,date] $\cup$ BookedMealDiscounted[customer,company,type,date] $\subseteq$
  FixedMM[customer,company,type,date] $\cup$ FreeMM[customer,company,type,date]

Problem 2: Restructured relational schema (2/2)

FixedMM(<u>customer</u>, <u>company</u>, <u>type</u>, <u>date</u>, fcourse, scourse)

generalized inclusion constraint: FixedMM[customer,company,type,date] ⊆
BookedMeal[customer,company,type,date] ∪
BookedMealDiscounted[customer,company,type,date]

foreign key: FixedMM[fcourse] ⊆ FCourse[code]

foreign key: FixedMM[scourse] ⊆ SCourse[code]

FreeMM(<u>customer</u>, <u>company</u>, <u>type</u>, <u>date</u>)

generalized inclusion constraint: FreeMM[customer,company,type,date] ⊆
BookedMeal[customer,company,type,date] ∪
BookedMealDiscounted[customer,company,type,date]

Problem 2: Restructured relational schema – External constraints (1/2)

1) The constraint on whether the value of the optional attribute **discount** is defined or not, needs to be reformulated over the restructured relational schema, again through a relational algebra query that must evaluate to **false**:

$$
\begin{aligned}
&\mathrm{PROJ}_{customer,company,type,date}\ (\\
&\quad ((\mathrm{BookedMeal\ JOIN}_{customer=taxcode}\ \mathrm{Customer})\ \mathrm{JOIN}_{taxcode=cus}\\
&\qquad \mathrm{REN}_{cus \leftarrow customer,\ com \leftarrow company}\ (\mathrm{Subscription}))\ \mathrm{JOIN}_{com=code\ AND\ code=company}\ \mathrm{Company})\\
&\cup\ \mathrm{PROJ}_{customer,company,type,date}\ (\\
&\qquad \mathrm{PROJ}_{customer,company,type,date,discount}\ (\mathrm{BookedMealDiscounted})\ -\\
&\qquad \mathrm{PROJ}_{customer,company,type,date,discount}\ (\\
&\qquad\quad ((\mathrm{BookedMealDiscounted\ JOIN}_{customer=taxcode}\ \mathrm{Customer})\ \mathrm{JOIN}_{taxcode=cus}\\
&\qquad\qquad \mathrm{REN}_{cus \leftarrow customer,\ com \leftarrow company}\ (\mathrm{Subscription}))\ \mathrm{JOIN}_{com=code\ AND\ code=company}\ \mathrm{Company}))
\end{aligned}
$$

Note: As far as the IDB exam is concerned, it is sufficient to specify that the constraint needs to be reformulated on the restructured relational schema, but there is no need to specify it in terms of a relational algebra query.

Problem 2: Restructured relational schema – External constraints (2/2)

2) The constraint that a customer can subscribe at most once to the same company in a given period, needs to be reformulated over the restructured relational schema, again through a relational algebra query that must evaluate to **false**:

$SEL_{startdate \leq start2 \text{ AND } start2 \leq end}($
$\quad SubscriptionT\ JOIN_{customer = cus2,\ company = com2}\ (REN_{cus2 \leftarrow customer,\ com2 \leftarrow company,\ start2 \leftarrow startdate}($
$\quad\quad SubscriptionT \cup PROJ_{customer,\ company,\ startdate}(SubscriptionNT))))$
$\cup\ SEL_{startdate \leq start2}($
$\quad SubscriptionNT\ JOIN_{customer = cus2,\ company = com2}\ (REN_{cus2 \leftarrow customer,\ com2 \leftarrow company,\ start2 \leftarrow startdate}($
$\quad\quad SubscriptionT \cup PROJ_{customer,\ company,\ startdate}(SubscriptionNT))))$

Note: As far as the IDB exam is concerned, it is sufficient to specify that the constraint needs to be reformulated on the restructured relational schema, but there is no need to specify it in terms of a relational algebra query.

3-4) The generalization constraints resulting from the elimination of the complete hierarchies for **BookedMeal** and for **Dish** are already expressed in the schema.

Problem 3

Consider the relation `Votes(newspaper,player,day,score)`, each tuple of which records the score assigned by a newspaper to a player in a certain day of the championship.

1. Let us call "top player" a player who in all the days of the championship had an average score (average calculated on all newspapers in that day) equal or higher than the average of the other players in the same day. Write a SQL query that calculates all top players.
2. Describe in words what the following query computes:

```
SELECT DISTINCT V1.player
FROM Votes V1
WHERE 7 <= ALL(SELECT score  FROM Votes V2  WHERE V2.player = V1.player) AND
      NOT EXISTS (SELECT V3.player  FROM Votes V3
                  WHERE V3.day = V1.day AND V3.newspaper = V1.newspaper AND
                        V3.score < V1.score)
```

Problem 3: Solution (1/2)

Votes(<u>newspaper</u>,<u>player</u>,<u>day</u>,score)

1. Let us call “top player” a player who in all the days of the championship had an average score (average calculated on all newspapers in that day) equal or higher than the average of the other players in the same day. Write a SQL query that calculates all top players.

```
SELECT DISTINCT V1.player
FROM Votes V1
WHERE NOT EXISTS
  (SELECT V2.day  FROM Votes V2
   WHERE V2.player = V1.player
   GROUP BY V2.day
   HAVING AVG(V2.score) < (SELECT AVG(V3.score)
                           FROM Votes V3
                           WHERE V3.day = V2.day AND
                                 V3.player <> V2.player))
```

Problem 3: Solution (2/2)

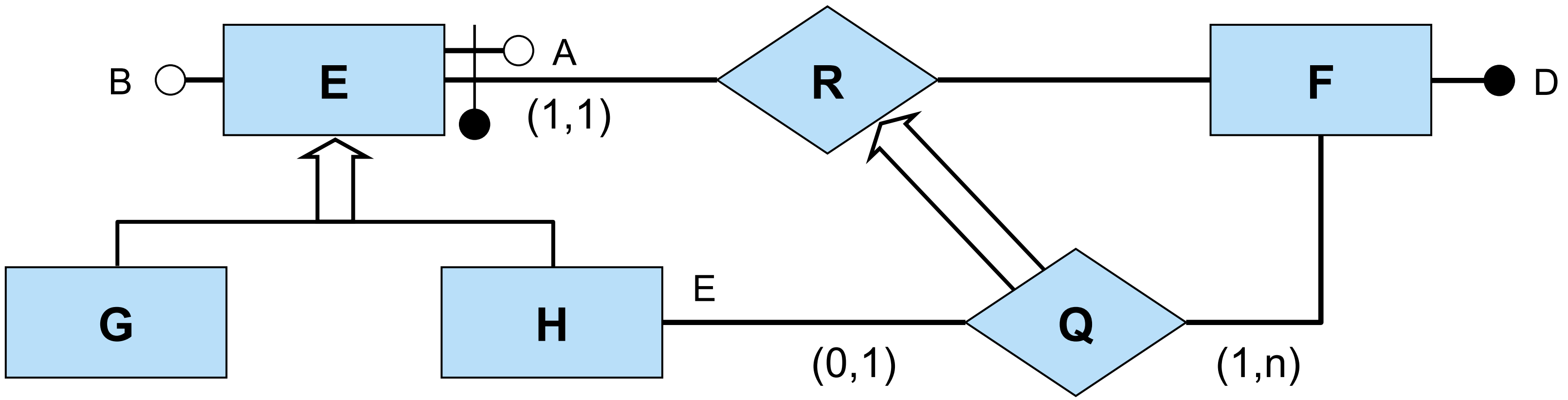
`Votes(newspaper,player,day,score)`

2. Describe in words what the following query computes:

```
SELECT DISTINCT V1.player
FROM Votes V1
WHERE 7 <= ALL(SELECT score  FROM Votes V2  WHERE V2.player = V1.player) AND
      NOT EXISTS (SELECT V3.player  FROM Votes V3
                  WHERE V3.day = V1.day AND V3.newspaper = V1.newspaper AND
                        V3.score < V1.score)
```

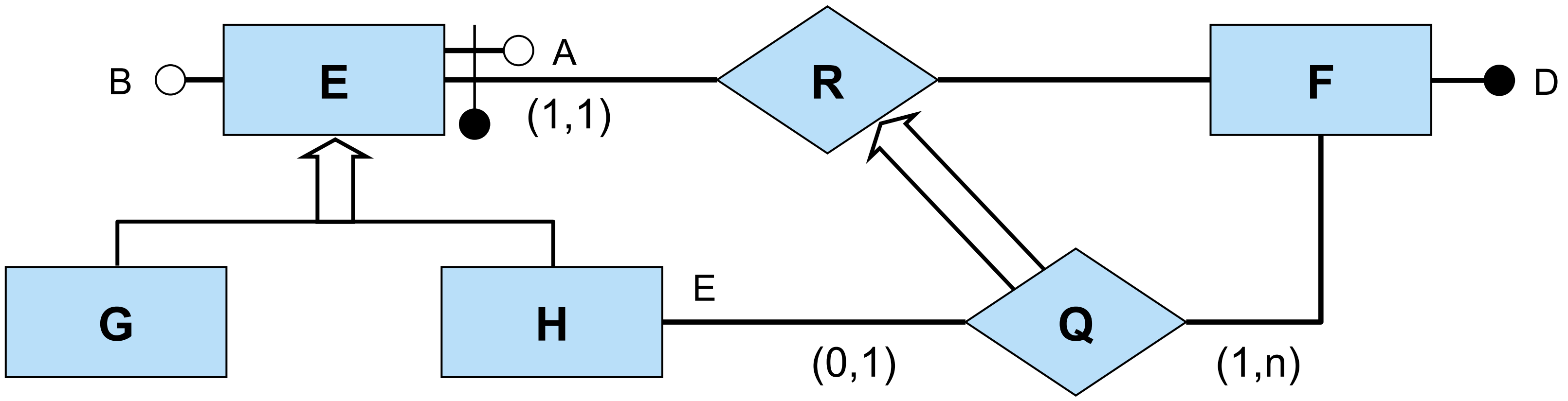
The query computes the set of all players *p* all of whose scores are at least 7, and moreover on some day for some newspaper no player had a score lower than that of *p.*

Problem 4



Consider the conceptual diagram *S* shown above. Provide the list of all pairs ⟨e_1,e_2⟩ of entities of *S* such that e_1 and e_2 are disjoint, i.e., such that in each instance *I* of schema *S*, the set *instances*(*I*,e_1) of instances of e_1 in *I* is disjoint from the set *instances*(*I*,e_2) of instances of e_2 in *I*.

Problem 4: Solution



The following pairs of entities of *S* are disjoint:

- **⟨*G*,*H*⟩**, since they are siblings in a generalization hierarchy.
- **⟨*E*,*F*⟩**, since they do not have any common ancestor along the ISA hierarchy.
- **⟨*G*,*F*⟩**, **⟨*H*,*F*⟩**, since *G* and *H* are subentities of *E*, and *E* and *F* are disjont.
- **⟨*G*,*E*⟩**, assuming that we consider only finite instances *I* of *S*. Indeed, the cardinality constraints on *R* and *Q* imply that *E* and *H* have the same number of instances in *I*, for every instance *I* of *S*. If *I* is finite, together with *instances*(*I*,*H*) ⊆ *instances*(*I*,*E*), this implies that *instances*(*I*,*H*) = *instances*(*I*,*E*). Hence, considering that *G* and *H* are disjoint and that *instances*(*I*,*G*) ⊆ *instances*(*I*,*E*), we have that *instances*(*I*,*G*) is empty. And an empty entity is disjoint from every other entity, in particular from *E*.