Introduction to Databases

Exam of 27/06/2023

With Solutions

Diego Calvanese

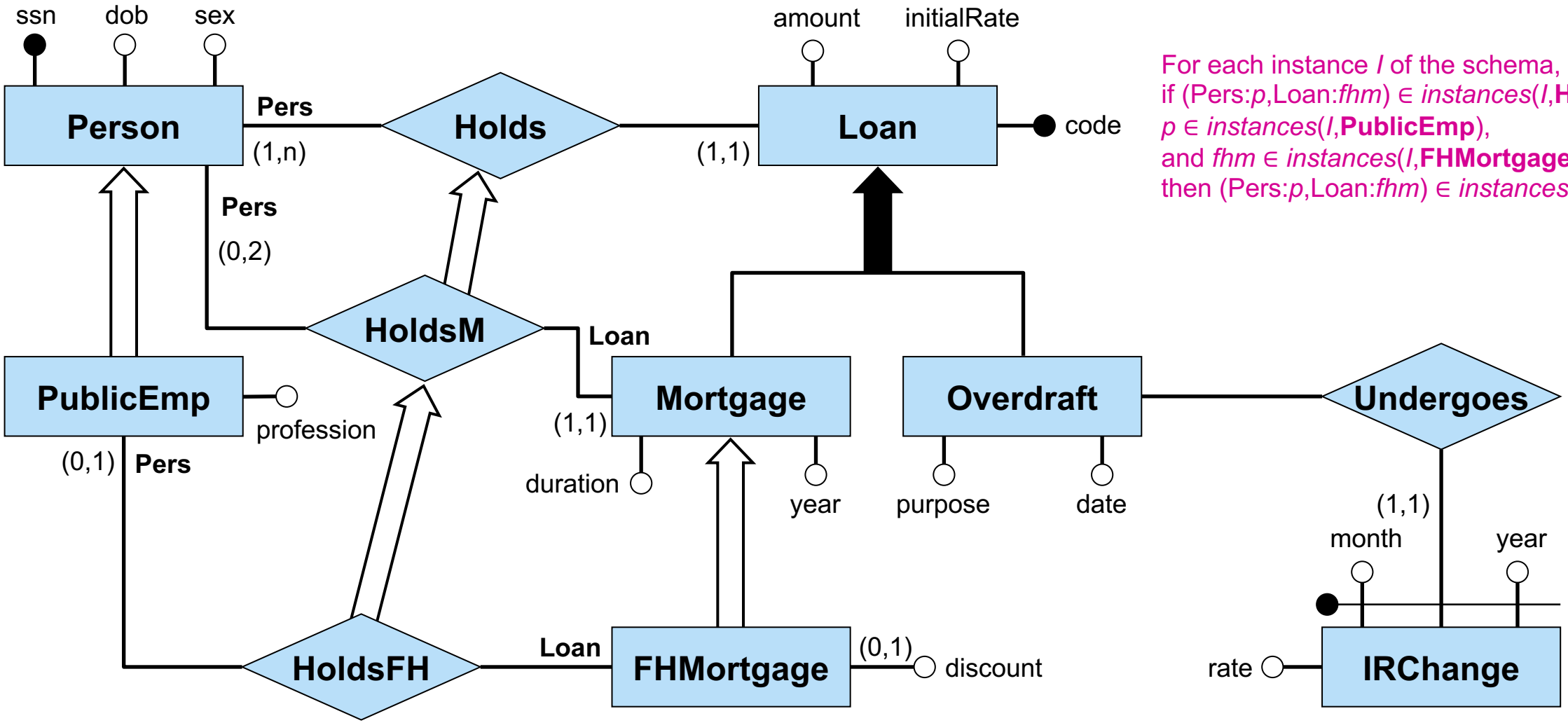
Bachelor in Computer Science
Faculty of Computer Science
Free University of Bozen-Bolzano

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

Problem 1

Design the Entity-Relationship schema of an application related to loans granted by a bank to persons. Of each **loan**, we are interested in the code (identifier), the amount, the initial interest rate, and the person who holds the loan. Each loan is of one of the following two types: overdraft and mortgage. Of each **overdraft**, we are interested in the date it was granted, the purpose for which it was requested, and the changes in the interest rate that were applied to it, knowing that this interest rate cannot change more than once per month. Of each **mortgage**, we are interested is the year in which it began and in its duration. Some mortgages are **for the first home** and of these we are interested in the discount applied to the interest rate, if any. Of each **person** holding a loan, we are interested in the social security number (identifier), the date of birth, and the sex. Of each such person who is a **public employee** we are also interested in the profession. Finally, consider that each person cannot hold more than two mortgages and each public employee cannot hold more than one first-home mortgage.

Problem 1: Conceptual schema



For each instance *I* of the schema,
if (Pers:*p*,Loan:*fhm*) ∈ *instances*(*I*,**Holds**),
p ∈ *instances*(*I*,**PublicEmp**),
and *fhm* ∈ *instances*(*I*,**FHMortgage**),
then (Pers:*p*,Loan:*fhm*) ∈ *instances*(*I*,**HoldsFH**).

Problem 2

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

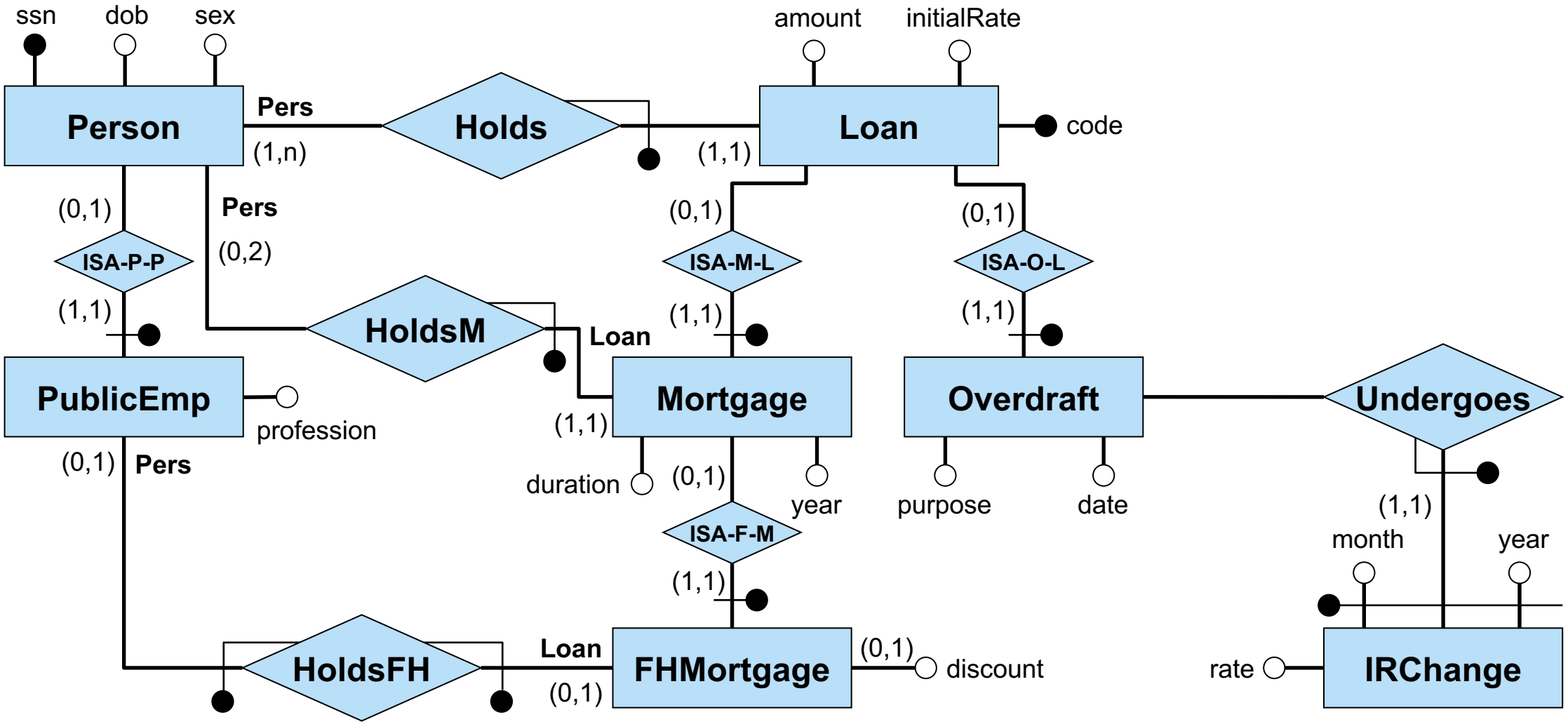
1. Loans are always accessed by choosing to access either mortgages or overdrafts, never both.
2. whenever we access a loan, we always want to know whether it is a first-home loan or not and if so we always want to know the interest rate discount, if any.

In your design you should follow the methodology adopted in the course, and 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).

You should motivate explicitly how the above indications affect your design.

Problem 2: Restructured conceptual schema



Problem 2: Restructured conceptual schema – External constraints

1. For each instance *I* of the schema, if (Pers:*p*,Loan:*fhm*) ∈ *instances*(*I*,**Holds**), *p* ∈ *instances*(*I*,**PublicEmp**), and *fhm* ∈ *instances*(*I*,**FHMortgage**), then (Pers:*p*,Loan:*fhm*) ∈ *instances*(*I*,**HoldsFH**).
2. Generalization constraint for the entity **Loan**:
   For each instance *I* of the schema, each instance of **Loan** in *I*, participates either to **ISA-M-L** or to **ISA-O-L**, but not to both.
3. Constraint resulting from the elimination of ISA between relationships **HoldsM** and **Holds**:
   For each instance *I* of the schema, if (Pers:*p*,Loan:*m*) ∈ *instances*(*I*,**HoldsM**) and *lo* is the (unique) instance of **Loan** such that (Mortgage:*m*,Loan:*lo*) ∈ *instances*(*I*,**ISA-M-L**), then (Pers:*p*,Loan:*lo*) ∈ *instances*(*I*,**Holds**).
4. Constraint resulting from the elimination of ISA between relationships **HoldsFH** and **HoldsM**:
   For each instance *I* of the schema, if (Pers:*pe*,Loan:*fhm*) ∈ *instances*(*I*,**HoldsFH**), *p* is the (unique) instance of **Person** such that (PublicEmp:*pe*,Person:*p*) ∈ *instances*(*I*,**ISA-P-P**), and *m* is the (unique) instance of **Mortgage** such that (FHMortgage:*fhm*,Mortgage:*m*) ∈ *instances*(*I*,**ISA-F-M**), then (Pers:*p*,Mortgage:*m*) ∈ *instances*(*I*,**HoldsM**).

Problem 2: Direct translation (1/2)

Person(ssn, dob, sex)
  inclusion: Person[ssn] $\subseteq$ Holds[person]

PublicEmp(ssn, profession)
  foreign key: PublicEmp[ssn] $\subseteq$ Person[ssn]

Loan(code, amount, initialRate)
  foreign key: Loan[code] $\subseteq$ Holds[loan]

Mortgage(code, duration, year)
  foreign key: Mortgage[code] $\subseteq$ Loan[code]
  foreign key: Mortgage[code] $\subseteq$ HoldsM[loan]

Overdraft(code, purpose, date)
  foreign key: Overdraft[code] $\subseteq$ Loan[code]

FHMortgage(code, discount*)
  foreign key: FHMortgage[code] $\subseteq$ Mortgage[code]

Problem 2: Direct translation (2/2)

IRChange(month, year, loan, rate)
  foreign key: IRChange[loan] $\subseteq$ Overdraft[code]

Holds(person, loan)
  foreign key: Holds[person] $\subseteq$ Person[ssn]
  foreign key: Holds[loan] $\subseteq$ Loan[code]

HoldsM(person, loan)
  foreign key: HoldsM[person,loan] $\subseteq$ Holds[person,loan]
  foreign key: HoldsM[loan] $\subseteq$ Mortgage[code]

HoldsFH(person, loan)
  foreign key: HoldsFH[person,loan] $\subseteq$ HoldsM[person,loan]
  foreign key: HoldsFH[person] $\subseteq$ PublicEmp[ssn]
  foreign key: HoldsFH[loan] $\subseteq$ FHMortgage[code]
  key: person

Problem 2: Direct translation – External constraints

1. Holds $\cap$ $\mathbf{REN}_{\text{person,loan} \leftarrow \text{ssn,code}}(\mathbf{PROJ}_{\text{ssn}}(\text{PublicEmp})$ **JOIN** $\mathbf{PROJ}_{\text{code}}(\text{FHMortgage})) \subseteq \text{HoldsFH}$
2. Generalization constraints for the entity **Loan**:
   $\text{Loan[code]} \subseteq \text{Mortgage[code]} \cup \text{Overdraft[code]}$
   $\text{Mortgage[code]} \cap \text{Overdraft[code]} = \emptyset$
3. The constraint resulting from the elimination of ISA between relationships **HoldsM** and **Holds** has already been expressed as a foreign key on **HoldsM**.
4. The constraint resulting from the elimination of ISA between relationships **HoldsFH** and **HoldsM** has already been expressed as a foreign key on **HoldsFH**.
5. Check constraint enforcing the maximum cardinality 2 constraint on the **Pers** role of **HoldsM**.

Problem 2: Restructuring of the relational schema

1. Loans are always accessed by choosing to access either mortgages or overdrafts, never both.
2. Whenever we access a loan, we always want to know whether it is a first-home loan or not and if so we always want to know the interest rate discount, if any.

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

- We take into account indication 1 by performing a horizontal decomposition of **Loan** into **LoanMortgage** and **LoanOverdraft**, and a subsequent merge of **LoanMortgage** into **Mortgage**, and of **LoanOverdraft** into **Overdraft**. We also perform a corresponding horizontal decomposition of **Holds** into **HoldsMort** and **HoldsO**, and a subsequent merge of **HoldsMort** into **HoldsM**.
- We take into account indication 2 by merging relation **FHMortgage** into **Mortgage**, adding a boolean flag **firsthome** to distinguish the instances of **FHMortgage**. We also remove the relation **HoldsFH**, and we reexpress the constraint stating that the **person** attribute is a key by means of a check constraint.

Problem 2: Restructured relational schema

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

The relations Loan, FHMortgage, Holds, and HoldsFH are removed.

Mortgage(<u>code</u>, amount, initialRate, duration, year, firsthome, discount*)
  foreign key: Mortgage[code] $\subseteq$ HoldsM[loan]

Overdraft(<u>code</u>, amount, initialRate, purpose, date)
  foreign key: Overdraft[code] $\subseteq$ HoldsO[loan]

HoldsM(person, <u>loan</u>)
  foreign key: HoldsM[person] $\subseteq$ Person[ssn]
  foreign key: HoldsM[loan] $\subseteq$ Mortgage[code]

HoldsO(person, <u>loan</u>)
  foreign key: HoldsO[person] $\subseteq$ Person[ssn]
  foreign key: HoldsO[loan] $\subseteq$ Overdraft[code]

Problem 2: Restructured schema – External constraints

All constraints can be removed, except for the following ones.

1. Disjointness part of the generalization constraints for the entity **Loan**: Mortgage[code] ∩ Overdraft[code] = ∅
2. Check constraint enforcing the maximum cardinality 2 constraint on the **Pers** role of **HoldsM**.
3. Check constraint enforcing that there are no two tuples of **HoldsM** coinciding on the **person** attribute, and such that the **loan** attributes of the two tuples are equal to the **code** attributes of two tuples in **Mortgage** for which the **firsthome** attribute is true.

Problem 3

Consider the relation `Deposit(code,customer,month,year,amount)`, which stores deposits that a customer makes on their bank account, with the identification code of the deposit (primary key), the customer holding the account, the month and year in which the deposit was made, and the amount in Euros of the deposit.

1. Express **in SQL** a query that, given a month *M* of a year *Y* (e.g., June 2023), returns for each customer the balance of the account in month *M* of year *Y* . Notice that the balance is the sum of all deposits made by that customer up to month *M* of year *Y*.
2. Express **in SQL** a query that, for each customer and for each year, returns the month(s) in which the maximum amount for that year was deposited.
3. Express **in relational algebra** a query that, for each customer, returns the month(s) in which the customer has made at least two deposits with the same amount (where obviously a month is identified by the month number and the year).

Problem 3: Solution (1/3)

`Deposit(code,customer,month,year,amount)`

1. Express **in SQL** a query that, given a month *M* of a year *Y* (e.g., June 2023), returns for each customer the balance of the account in month *M* of year *Y* . Notice that the balance is the sum of all deposits made by that customer up to month *M* of year *Y*.

```
SELECT customer, SUM(amount)
FROM Deposit
WHERE (year < Y) OR (year = Y AND month <= M)
GROUP BY customer
```

Problem 3: Solution (2/3)

`Deposit(code,customer,month,year,amount)`

2. Express **in SQL** a query that, for each customer and for each year, returns the month(s) in which the maximum amount for that year was deposited.

   Note: “maximum amount” is considered as the maximum total amount deposited in a month.

```
WITH TotalMonthlyDeposits AS
  (SELECT customer, year, month, SUM(deposit) AS monthlyDeposit
   FROM Deposit
   GROUP BY customer, year, month)
SELECT customer, year, month
FROM TotalMonthlyDeposits TMDO
WHERE monthlyDeposit >= ALL (SELECT TMD.monthlyDeposit
                             FROM TotalMonthlyDeposits TMD
                             WHERE TMD.customer = TMDO.customer AND
                                   TMD.year = TMDO.year)
```

Problem 3: Solution (3/3)

`Deposit(code,customer,month,year,amount)`

3. Express **in relational algebra** a query that, for each customer, returns the month(s) in which the customer has made at least two deposits with the same amount (where obviously a month is identified by the month number and the year).

$$\mathbf{PROJ}_{\text{customer,month,year}}\ (\mathbf{SEL}_{\text{code} <> \text{code2}}\ (\text{Deposit}\ \mathbf{JOIN}\ \mathbf{REN}_{\mathbf{code2} \leftarrow \text{code}}\ (\text{Deposit})))$$

Problem 4

R

A	B	C	D
6	6	6	1
4	null	3	2
6	6	5	3
5	6	6	4

S

E
5
6
7

Consider the relational database shown above and answer the following questions.

1. Which are the key constraints that are satisfied in the relation *R* (i.e., which are the keys of *R*)?
2. Which are the superkey constraints that are satisfied in the relation *R* (i.e., which are the superkeys of *R*)?
3. Which are the foreign key constraints that are satisfied in the database?

Problem 4: Solution

R

A	B	C	D
6	6	6	1
4	null	3	2
6	6	5	3
5	6	6	4

S

E
5
6
7

Consider the relational database shown above and answer the following questions.

1. Which are the keys of *R*? – The keys of *R* are: {D}, {A,C}
2. Which are the superkeys of *R*? – The superkeys of *R* are:
   {D}, {D,A}, {D,B}, {D,C}, {D,A,B}, {D,A,C} {D,B,C}, {D,A,B,C}, {A,C}, {A,B,C}
3. Which are the foreign key constraints that are satisfied in the database?
   The foreign key constraints satisfied in the database are:
   $R[B] \subseteq S[E]$, $R[B] \subseteq R[A]$, $R[B] \subseteq R[C]$