Information Integration – Exercise 6 – Gabriel Glaser

Task 1: LaV and the Bucket Algorithm

Sources:
S1: Company(CName, HQAddress)
S2: Product(PName, Category, Price, CName)
S3: Inventory(Product, Price)
S4: Inventag(ProduktName, Kategorie, FirmenName)
S5: CompanyProducts(CName, HQAddress, Product)

Global schema G
Firma(FirmenName, Sitz)
Produkt(ProduktName Kategorie, Preis, FirmenName)

a) Develop a LaV model for the scenario described above, using **Datalog** notation.

b) Given the following Datalog query over G, **how many buckets** does the bucket algorithm create to answer the query Q using you LaV model. Label these buckets (e.g., Bucket1, Bucket2, ...) for future reference.

```
Q(FName, PName, Price) :- Firma(FName, HQ), Produkt(PName, Cat,
    Price, FName), Cat = 'Book', Price < 100
```

c) **Which views are added to which buckets?** If a view is not added to a bucket, explain why.

d) **Provide the final rewriting of Q** using views in Datalog notation (hint: you will have a Datalog rule for **each relevant view combination**).

a)

- S1:

  S1(FirmenName, Sitz) :–
  Firma(FirmenName, Sitz)

  S1 exactly the same than relation “Firma”.

- S2:

  S2(ProduktName, Kategorie, Preis, FirmenName) :–
  Produkt(ProduktName, Kategorie, Preis, FirmenName)

  S2 exactly the same than relation “Produkt”.

- S3:

  S3(ProduktName, Preis) :–
  Produkt(ProduktName, Kategorie, Preis, FirmenName)

  Assumption: “ProduktName” and “PName” are the primary keys for their respective relation (Produkt / Product). Thus, they can be used as a foreign key to reference a product.

- S4:

  ```
  S4(ProduktName, Kategorie, FirmenName) :–
      Produkt(ProduktName, Kategorie, Preis, FirmenName)
  ```

  Basically S4 is the same table than "Produkt" but doesn't include the attribute "Preis".

- S5:

  ```
  S5(FirmenName, Sitz, ProduktName) :–
      Firma(FirmenName, Sitz),
      Produkt(ProduktName, Kategorie, Preis, FirmenName)
  ```

  Join both global schema tables according to "FirmenName" to get S5.

b) Given query Q over global schema G needs two buckets, because Q has two sub-goals. The bucket for *Firma* is called "$\text{Bucket}_{\text{Firma}}$" and the bucket for *Produkt* is called "$\text{Bucket}_{\text{Produkt}}$"

c)

- $\text{Bucket}_{\text{Firma}}$ corresponding to Firma(FName, HQ), all sub-goals of all views:
  - *S1.Firma(FirmenName, Sitz)* ✓
  - *S2.Produkt(ProduktName, Kategorie, Preis, FirmenName)* *X* (Doesn't contain HQ)
  - *S3.Produkt(ProduktName, Kategorie, Preis, FirmenName)* *X* (Doesn't contain HQ)
  - *S4.Produkt(ProduktName, Kategorie, Preis, FirmenName)* *X* (Doesn't contain HQ)
  - *S5.Firma(FirmenName, Sitz)* ✓
  - *S5.Produkt(ProduktName, Kategorie, Preis, FirmenName)* *X* (Doesn't contain HQ)

  ⇒ S1, S5

- $\text{Bucket}_{\text{Produkt}}$ corresponding to Produkt(PName, Cat, Price, FName), all sub-goals of all views:
  - *S1.Firma(FirmenName, Sitz)* *X* (Doesn't contain needed attributes)
  - *S2.Produkt(ProduktName, Kategorie, Preis, FirmenName)* ✓
  - *S3.Produkt(ProduktName, Kategorie, Preis, FirmenName)* *X* (Price and Cat not preserved by S3)
  - *S4.Produkt(ProduktName, Kategorie, Preis, FirmenName)* *X* (Price not preserved by S4)
  - *S5.Firma(FirmenName, Sitz)* *X* (Doesn't contain PName, Cat, Price)

– *S5.Produkt(ProduktName, Kategorie, Preis, FirmenName)* X (Price and Category not preserved by S4)

$\Rightarrow$ S2

d) Datalog queries (not optimized):

- S1, S2:

```
Q(FirmenName, ProduktName, Preis) :–
    S1(FirmenName, Sitz),
    S2(ProduktName, Kategorie, Preis, FirmenName)
```

- S5, S2:

```
Q(FirmenName, ProduktName, Preis) :–
    S5(FirmenName, Sitz, ProduktName),
    S2(ProduktName, Kategorie, Preis, FirmenName)
```

Task2: Bucket Algorithm (Sample exam question)

Given the following query Q and the four views V_1 to V_4, apply the **bucket algorithm** step by step by answering the questions below.

$$
\begin{aligned}
Q(ID, Dir) &:- Movie(ID, Title, Year, Genre), Revenues(ID, Amount), Director(ID, Dir), \\
&\quad Amount \geq 100M \\
V_1(I, Y) &:- Movie(I, T, Y, G), Revenues(I, A), I \geq 5000, A \geq 200M \\
V_2(I, A) &:- Movie(I, T, Y, G), Revenues(I, A) \\
V_3(I, A) &:- Revenues(I, A), A \leq 50M \\
V_4(I, D, Y) &:- Movie(I, T, Y, G), Director(I, D), I \leq 3000
\end{aligned}
$$

a) **How many buckets** does the bucket algorithm create to answer the query Q using the available views. Label these buckets (e.g., B1, B2, etc.) for future reference.

b) **Which views are added to which buckets**? If a view is not added to a bucket, explain why.

c) Given the views in each bucket, which **view combinations** need to be considered? Are any of these irrelevant? If so, explain why.

d) For the remaining combinations, write the query Q in Datalog referring to the views in the body of the rule. Be careful to perform the correct joins.

e) Can any of the combinations be further **simplified**? If so, explain why.

f) Write down the **final rewriting** for the query Q in **SQL**.

a) The bucket algorithm needs *three* buckets, B_{Movie}, B_{Revenues} and B_{Director}.

b)

- B_{Movie} with Movie(ID, Title, Year, Genre):
  - V1: 1. Movie(ID, Title, Year, Genre) = Movie(I, T, Y, G), 2. I $\geq$ 5000 doesn't contradict, 3. Variables of Q which are in Q.Movie are also in V1.

  - V2: 1. Movie(ID, Title, Year, Genre) = Movie(I, T, Y, G), 2. No contradictions, 3. Variables of Q which are in Q.Movie are also in V2.
  - not V3 (wrong relation)
  - V4: 1. Movie(ID, Title, Year, Genre) = Movie(I, T, Y, G), 2. I ≤ 3000 doesn't contradict, 3. Variables of Q which are in Q.Movie are also in V1.

  $\Rightarrow$ V1, V2, V4
- B_{Revenues} with Revenues(ID, Amount):
  - V1: 1. Revenues(ID, Amount) = Revenues(I, A), 2. predicates don't contradict, 3. Variables of Q which are in Q.Revenues (i.e., ID) are also in V2 (i.e., I).
  - V2: ✓
  - not V3, because predicates contradict.
  - not V4 (wrong relation)

  $\Rightarrow$ V1, V2
- B_{Director} with Director(ID, Dir):
  - not V1 (wrong relation)
  - not V2 (wrong relation)
  - not V3 (wrong relation)
  - V4 ✓

  $\Rightarrow$ V4

c) Combinations:
- V1, V1, V4: I $\leq 3000 \wedge$ I ≥ 5000 X (V1 contradicts with V4)
- V1, V2, V4: X
- V2, V1, V4: X
- V2, V2, V4:
- V4, V1, V4: X
- V4, V2, V4:

d) Remaining query plans in Datalog:
- V2, V2, V4:

```
Q(ID, Dir) :–
    V2(ID, Y),
    V2(ID, Y),
    V4(ID, Dir, Y)
```

- V4, V2, V4:

```
Q(ID, Dir) :–
    V4(ID, Dir, Y),
    V2(ID, Y),
    V4(ID, Dir, Y)
```

e) Both query plans contain the same view twice which is not necessary, because joining a second time doesn't change the results. Then, it can be easily seen that both are equivalent, i.e., only one needs to be executed to retrieve all results.

f) SQL query:

```
SELECT V2.ID, V4.ID, V4.Dir
FROM V2, V4
WHERE V2.ID = V4.ID;
```