CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

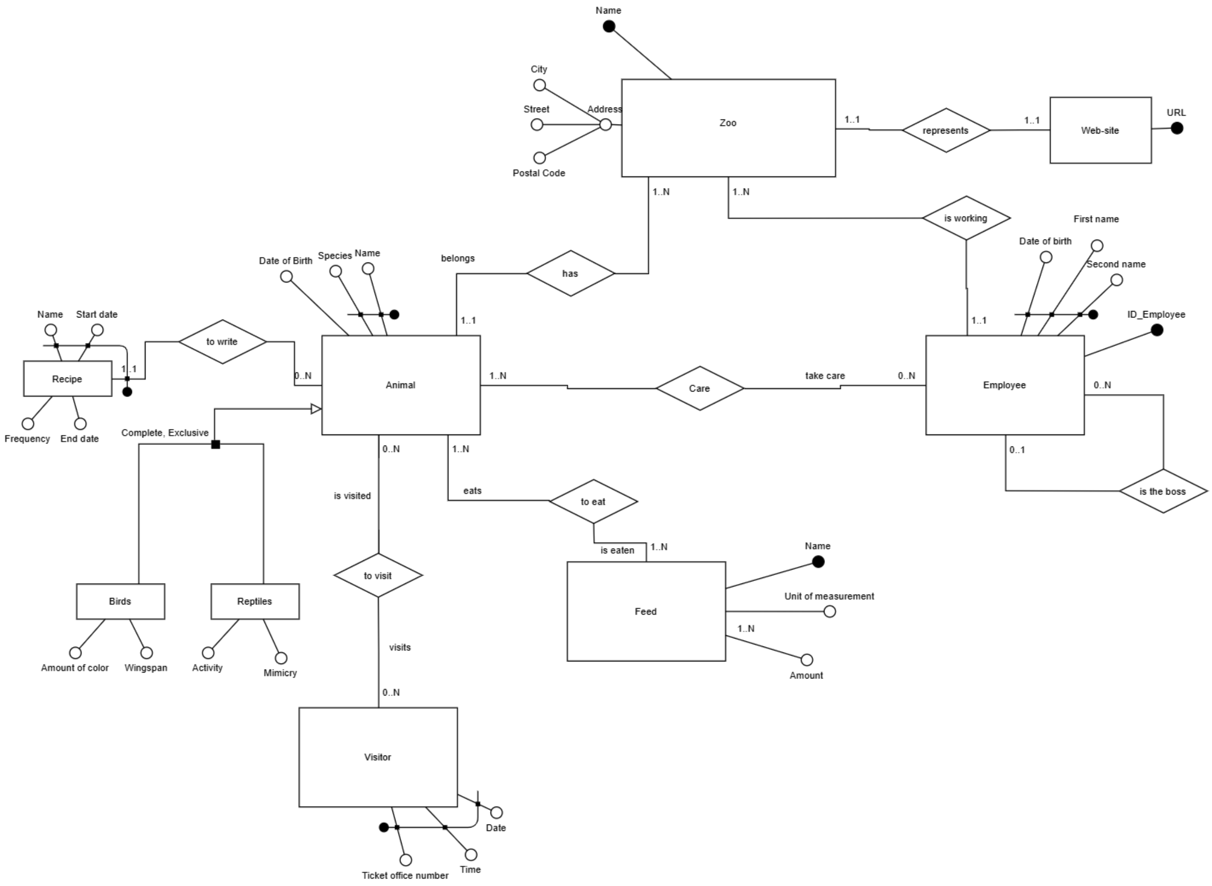
Semester Project

Subject: Database Systems
Student: Artur Shaimardanov

Conceptual Model

Objective: Create a relational conceptual model using ER Dia.
Requirements:

- Include 5-10 entity types without artificial identifiers.
- The model must include:
  - Simple, structured, and multi-valued attributes.
  - 1:1, 1
    , and N
    relationships with appropriate cardinalities.
  - Recursive or reflexive relationships.
  - Inheritance.
  - Weak entity types.
  - Relevant identifiers including composite and multiple identifiers for at least one entity type.



Topic: My Database is a Zoo

Entity Types:

1. Animal:
   - Identifier: ID_Animal
   - Attributes:
     - Age (Simple attribute)
     - Species (Multivalued attribute)
     - Weight (Simple attribute)
2. Zoo:
   - Identifier: ID_Zoo
   - Attributes:
     - Name (Simple attribute)
     - Address (Structured attribute with sub-attributes: Postal Code, Street, City)
     - Animal Types (Multivalued attribute)
3. Visitor:
   - Identifier: ID_Visitor
   - Attributes:
     - First Name (Simple attribute)
     - Age (Simple attribute)
     - Last Name (Simple attribute)
4. Food:
   - Identifier: ID_Food
   - Attributes:
     - Name (Simple attribute)

        - Size (Simple attribute)
        - Quantity (Multivalued attribute)

5. Employee:
    - Identifier: ID (Simple)
    - Attributes:
        - Composite identifier consisting of:
            - First Name (Simple attribute)
            - Last Name (Simple attribute)
6. Schedule: (Weak entity type)
    - Attributes:
        - Day of the Week (Simple attribute)
        - Start Time (Simple attribute)
        - End Time (Simple attribute)
        - Shift Type (Simple attribute)

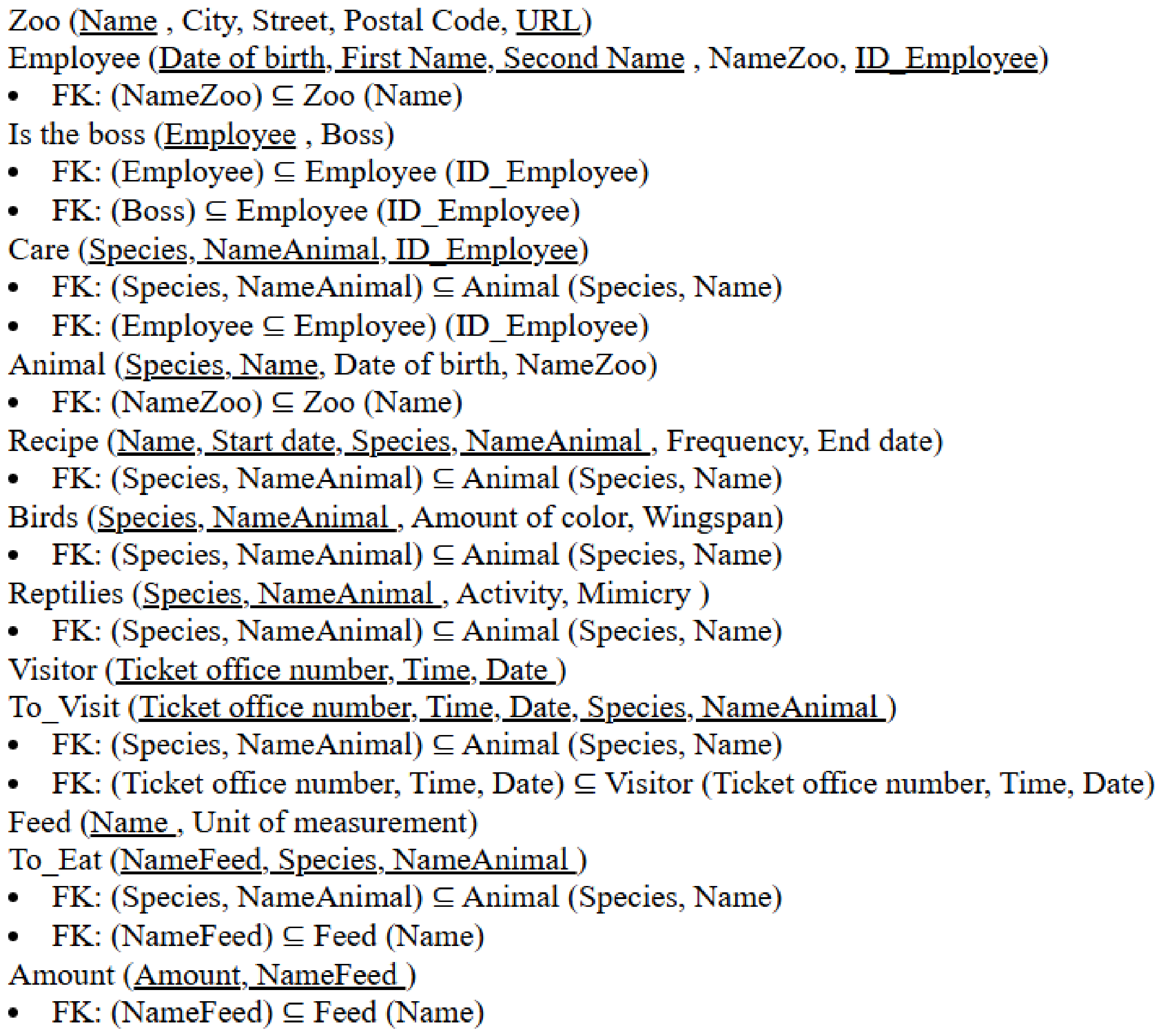
Relationships:

- Recursive Relationship: My supervisor’s supervisor is my supervisor.
- Reflexive Relationship: Visitor recommended.
- Inheritance:
    - Bird and Reptile inherit from Animal

Relational Model

Objective: Transform the conceptual model into a relational model.
Requirements:

- Use textual notation for tables including foreign keys.
- Do not consider NULL values.



Zoo (Name , City, Street, Postal Code, URL)
Employee (Date of birth, First Name, Second Name , NameZoo, ID_Employee)
- FK: (NameZoo) ⊆ Zoo (Name)

Is the boss (Employee , Boss)
- FK: (Employee) ⊆ Employee (ID_Employee)
- FK: (Boss) ⊆ Employee (ID_Employee)

Care (Species, NameAnimal, ID_Employee)
- FK: (Species, NameAnimal) ⊆ Animal (Species, Name)
- FK: (Employee ⊆ Employee) (ID_Employee)

Animal (Species, Name, Date of birth, NameZoo)
- FK: (NameZoo) ⊆ Zoo (Name)

Recipe (Name, Start date, Species, NameAnimal , Frequency, End date)
- FK: (Species, NameAnimal) ⊆ Animal (Species, Name)

Birds (Species, NameAnimal , Amount of color, Wingspan)
- FK: (Species, NameAnimal) ⊆ Animal (Species, Name)

Reptilies (Species, NameAnimal , Activity, Mimicry)
- FK: (Species, NameAnimal) ⊆ Animal (Species, Name)

Visitor (Ticket office number, Time, Date)
To_Visit (Ticket office number, Time, Date, Species, NameAnimal)
- FK: (Species, NameAnimal) ⊆ Animal (Species, Name)
- FK: (Ticket office number, Time, Date) ⊆ Visitor (Ticket office number, Time, Date)

Feed (Name , Unit of measurement)
To_Eat (NameFeed, Species, NameAnimal)
- FK: (Species, NameAnimal) ⊆ Animal (Species, Name)
- FK: (NameFeed) ⊆ Feed (Name)

Amount (Amount, NameFeed)
- FK: (NameFeed) ⊆ Feed (Name)

SQL - Database Creation and Data Queries

Objective: Convert the relational model into an ER model and generate SQL queries to create and query the database.
Requirements:

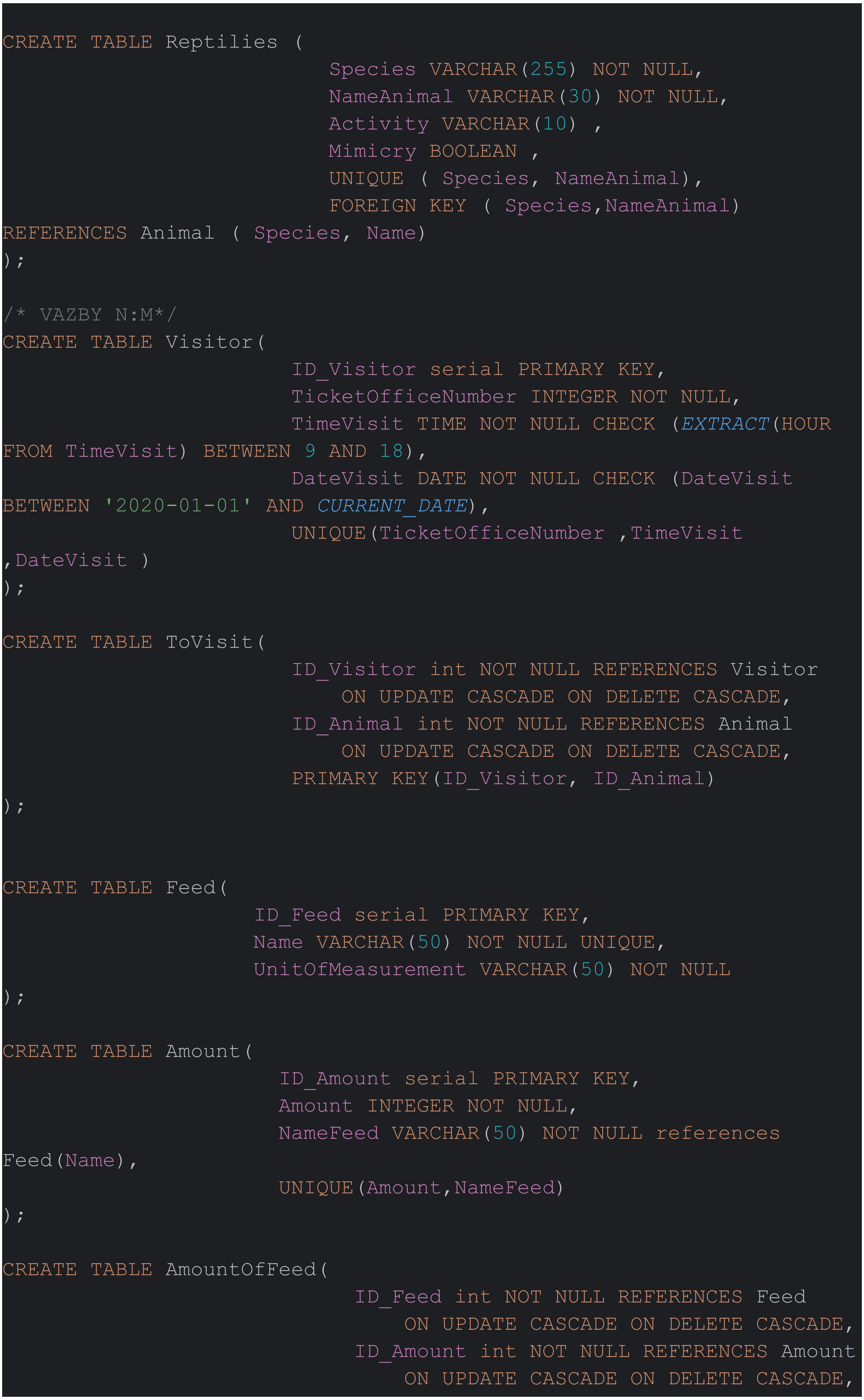
- Create all tables in the student database on `slon.felk.cvut.cz`.
- Ensure all SQL queries are executable on the specified server.
- Populate tables with relevant data, including a key table with approximately 32k operational data entries.
- Avoid using `ALTER TABLE` for adding integrity constraints.

1-Type Declaration :

```
Reptilies Mimicry BOOLEAN ( True - has Mimicry, False - doesn't have
)
Visitor has a CHECK on the date, which follows from the fact that
the opening date of the Zoo is 2020-01-01
```

2.Create Table

```
CREATE TABLE Zoo (
                    ID_ZOO serial PRIMARY KEY,
                    Name VARCHAR(255) Not NULL UNIQUE,
                    City VARCHAR(255) Not NULL,
                    Street VARCHAR(255) Not NULL,
                    PostalCode INTEGER Not NULL,
                    URL VARCHAR(255) Not NULL UNIQUE
);

CREATE TABLE Employee (
                         ID_Employee serial PRIMARY KEY,
                         RodneCislo VARCHAR(11) Not NULL UNIQUE,
                         DateOfBirth DATE Not NULL ,
                         FirstName VARCHAR(255) Not NULL ,
                         SecondName VARCHAR(255)Not NULL ,
                         NameZoo VARCHAR(255) REFERENCES Zoo (Name),
                         Boss INTEGER REFERENCES Employee
(ID_Employee) ON UPDATE CASCADE ON DELETE CASCADE,
                         UNIQUE (DateOfBirth, FirstName, SecondName)
);


CREATE TABLE Animal (
                       ID_Animal serial PRIMARY KEY,
                       Species VARCHAR(255) NOT NULL,
                       DateOfBirth DATE Not NULL,
                       Name VARCHAR(30) NOT NULL,
                       UNIQUE (Species, Name),
                       NameZoo VARCHAR(255) REFERENCES Zoo (Name)
);
/* slaba entita*/
CREATE TABLE Recipe (
                       ID_Animal int4 NOT NULL
                           REFERENCES Animal ON UPDATE CASCADE ON
DELETE CASCADE,
                       Name VARCHAR(30) NOT NULL,
                       StartDate DATE NOT NULL,
                       EndDate DATE NOT NULL,
                       Frequency INTEGER NOT NULL,
                       PRIMARY KEY (ID_Animal, Name, StartDate)
);

CREATE TABLE Birds (
                      Species VARCHAR(255) NOT NULL,
                      NameAnimal VARCHAR(30) NOT NULL,
                      AmountOfColor INTEGER CHECK( AmountOfColor >
0),
                      Wingspan INTEGER CHECK(Wingspan > 0),
                      UNIQUE ( Species, NameAnimal),
                      FOREIGN KEY ( Species,NameAnimal) REFERENCES
Animal ( Species, Name)
);
```

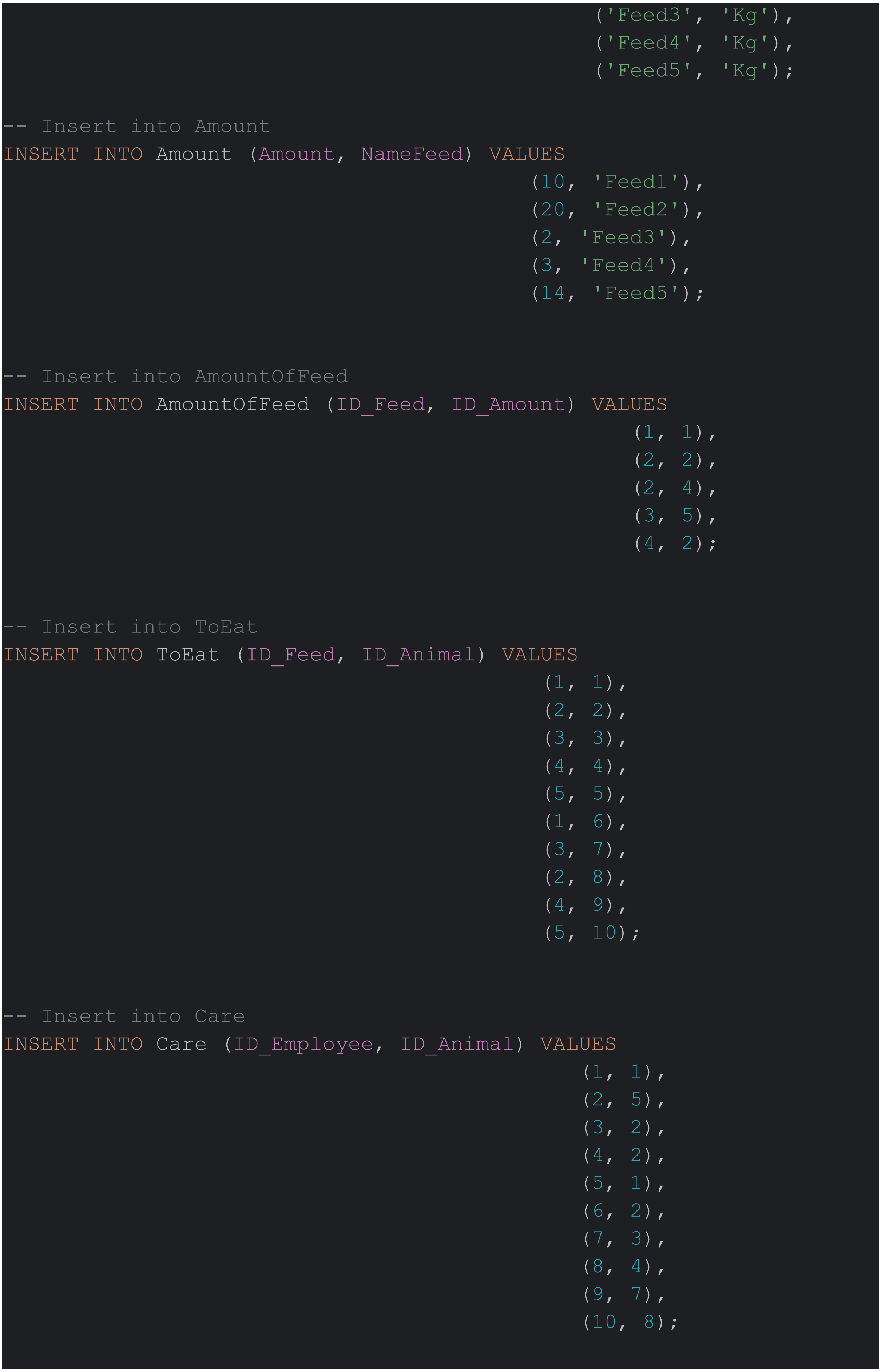


```
CREATE TABLE Reptilies (
                          Species VARCHAR(255) NOT NULL,
                          NameAnimal VARCHAR(30) NOT NULL,
                          Activity VARCHAR(10) ,
                          Mimicry BOOLEAN ,
                          UNIQUE ( Species, NameAnimal),
                          FOREIGN KEY ( Species,NameAnimal)
REFERENCES Animal ( Species, Name)
);

/* VAZBY N:M*/
CREATE TABLE Visitor(
                       ID_Visitor serial PRIMARY KEY,
                       TicketOfficeNumber INTEGER NOT NULL,
                       TimeVisit TIME NOT NULL CHECK (EXTRACT(HOUR
FROM TimeVisit) BETWEEN 9 AND 18),
                       DateVisit DATE NOT NULL CHECK (DateVisit
BETWEEN '2020-01-01' AND CURRENT_DATE),
                       UNIQUE(TicketOfficeNumber ,TimeVisit
,DateVisit )
);

CREATE TABLE ToVisit(
                       ID_Visitor int NOT NULL REFERENCES Visitor
                           ON UPDATE CASCADE ON DELETE CASCADE,
                       ID_Animal int NOT NULL REFERENCES Animal
                           ON UPDATE CASCADE ON DELETE CASCADE,
                       PRIMARY KEY(ID_Visitor, ID_Animal)
);


CREATE TABLE Feed(
                    ID_Feed serial PRIMARY KEY,
                    Name VARCHAR(50) NOT NULL UNIQUE,
                    UnitOfMeasurement VARCHAR(50) NOT NULL
);

CREATE TABLE Amount(
                      ID_Amount serial PRIMARY KEY,
                      Amount INTEGER NOT NULL,
                      NameFeed VARCHAR(50) NOT NULL references
Feed(Name),
                      UNIQUE(Amount,NameFeed)
);

CREATE TABLE AmountOfFeed(
                            ID_Feed int NOT NULL REFERENCES Feed
                                ON UPDATE CASCADE ON DELETE CASCADE,
                            ID_Amount int NOT NULL REFERENCES Amount
                                ON UPDATE CASCADE ON DELETE CASCADE,
                            PRIMARY KEY(ID_Feed, ID_Amount)
);

CREATE TABLE ToEat(
                     ID_Feed int NOT NULL REFERENCES Feed
                         ON UPDATE CASCADE ON DELETE CASCADE,
                     ID_Animal int NOT NULL REFERENCES Animal
                         ON UPDATE CASCADE ON DELETE CASCADE,
                     PRIMARY KEY(ID_Feed, ID_Animal)
);

CREATE TABLE Care (
                     ID_Employee int NOT NULL REFERENCES Employee
                         ON UPDATE CASCADE ON DELETE CASCADE,
                     ID_Animal int NOT NULL REFERENCES Animal
                         ON UPDATE CASCADE ON DELETE CASCADE,
                     PRIMARY KEY(ID_Employee, ID_Animal)
);
```

3. INSRET

```
-- Insert into Zoo
INSERT INTO Zoo (Name, City, Street, PostalCode, URL) VALUES
                                                         ('Kazan
Zoo', 'Kazan', 'Dekabristov 102', 12345, 'http://Kazanzoo.com'),
                                                         ('Moscow
Zoo', 'Moscow', 'Frankova 7', 23456, 'http://Moscowzoo.com'),
                                                         ('Tokyo
Zoo', 'Tokyo', 'Toshiba 12/1 ', 34567, 'http://TokyoZoo.com'),
                                                         ('Delhi
Zoo', 'Delhi', 'Tvero 9', 45678, 'http://Delhizoo.com'),
                                                         ('Shanghai
Zoo', 'Shanghai', 'Stroiva 22', 56789, 'http://Shanghaizoo.com'),
                                                         ('Dhaka
Zoo', 'Dhaka', 'Dewewa 8', 67890, 'http://Dhakazoo.com'),
                                                         ('Prague
Zoo', 'Prague', 'Zikova 12', 78901, 'http://Praguezoo.com'),
                                                         ('Osaka
Zoo', 'Osaka', 'Koriha 54', 89012, 'http://Osakazoo.com'),
                                                         ('Karachi
Zoo', 'Karachi', 'Jopin 2/12', 90123, 'http://Karachizoo.com'),
                                                         ('Lagos
Zoo', 'Lagos', 'Omigo 89', 10234, 'http://Lagoszoo.com');

-- Insert into Employee
INSERT INTO Employee (RodneCislo, DateOfBirth, FirstName,
SecondName, NameZoo) VALUES

('12345678901', '1980-01-01', 'Michael', 'Smith', 'Kazan Zoo'),

('23456789012', '1981-02-02', 'Sarah', 'Jones', 'Kazan Zoo'),
```

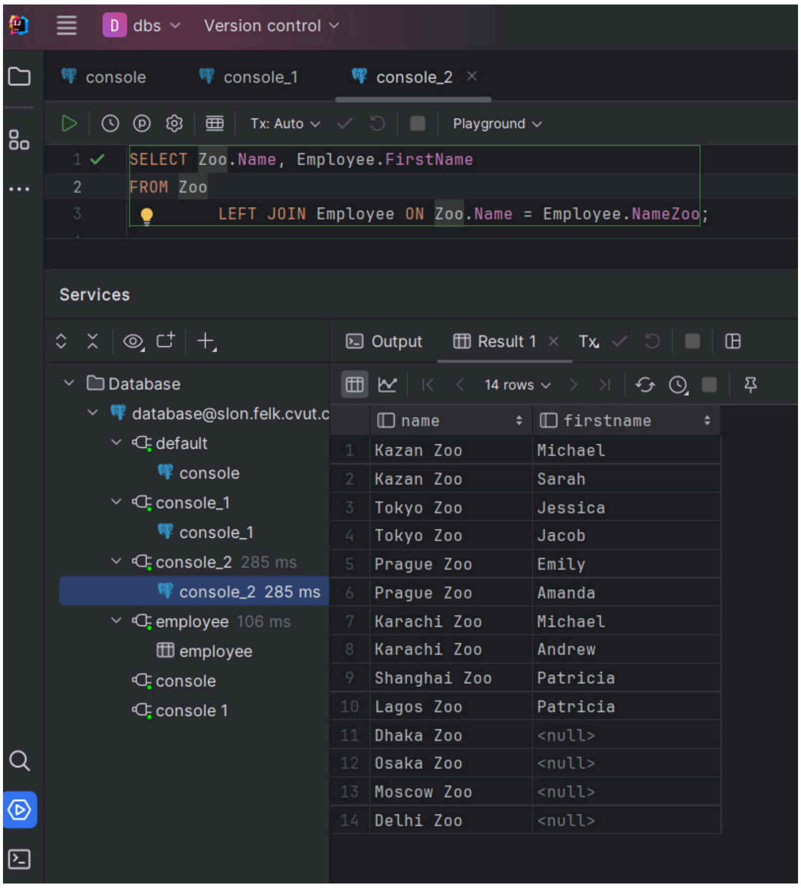
```sql
('34567890123', '1982-03-03', 'Jessica', 'Davis', 'Tokyo Zoo'),

('45678901234', '1983-04-04', 'Jacob', 'Williams', 'Tokyo Zoo'),

('56789012345', '1984-05-05', 'Emily', 'Brown', 'Prague Zoo'),

('67890123456', '1985-06-06', 'Amanda', 'Rodriguez', 'Prague Zoo'),

('78901234567', '1986-07-07', 'Michael', 'Ivanov', 'Karachi Zoo'),

('89012345678', '1987-08-08', 'Andrew', 'Williams', 'Karachi Zoo'),

('90123456789', '1988-09-09', 'Patricia', 'Martinez', 'Shanghai
Zoo'),

('10234567890', '1989-10-10', 'Patricia', 'Smith', 'Lagos Zoo');

-- Insert into Animal
INSERT INTO Animal (Species, DateOfBirth, Name, NameZoo) VALUES

('Birds', '2020-01-01', 'Name1', 'Moscow Zoo'),

('Reptilies', '2020-02-02', 'Name2', 'Prague Zoo'),

('Reptilies', '2020-03-03', 'Name3', 'Moscow Zoo'),

('Reptilies', '2020-04-04', 'Name4', 'Prague Zoo'),

('Birds', '2020-05-05', 'Name5', 'Kazan Zoo'),

('Reptilies', '2020-06-06', 'Name6', 'Kazan Zoo'),

('Reptilies', '2020-07-07', 'Name7', 'Moscow Zoo'),

('Birds', '2020-08-08', 'Name8', 'Kazan Zoo'),

('Reptilies', '2020-09-09', 'Name9', 'Kazan Zoo'),

('Birds', '2020-10-10', 'Name10', 'Kazan Zoo');

-- Insert into Recipe
INSERT INTO Recipe (ID_Animal, Name, StartDate, EndDate, Frequency)
VALUES

(1, 'Recipe1', '2022-01-01', '2022-12-31', 7),

(2, 'Recipe2', '2022-02-02', '2022-12-31', 14),

(3, 'Recipe3', '2022-03-03', '2022-12-31', 21),
(4, 'Recipe4', '2022-04-04', '2022-12-31', 28),

(5, 'Recipe5', '2022-05-05', '2022-12-31', 35),

(6, 'Recipe6', '2022-06-06', '2022-12-31', 42),

(7, 'Recipe7', '2022-07-07', '2022-12-31', 49),

(8, 'Recipe8', '2022-08-08', '2022-12-31', 56),

(9, 'Recipe9', '2022-09-09', '2022-12-31', 63),

(10, 'Recipe10', '2022-10-10', '2022-12-31', 70);

-- Insert into Birds
INSERT INTO Birds (Species, NameAnimal, AmountOfColor, Wingspan)
VALUES

('Birds', 'Name1', 3, 100),

('Birds', 'Name5', 6, 500),

('Birds', 'Name8', 7, 600),

('Birds', 'Name10', 7, 600);

INSERT INTO Reptilies (Species, NameAnimal, Activity, Mimicry)
VALUES

('Reptilies', 'Name2', 'Active', TRUE),

('Reptilies', 'Name3', 'Inactive', FALSE),

('Reptilies', 'Name4', 'Active', TRUE),

('Reptilies', 'Name6', 'Inactive', FALSE),

('Reptilies', 'Name7', 'Active', TRUE),

('Reptilies', 'Name9', 'Inactive', FALSE);

INSERT INTO ToVisit (ID_Visitor, ID_Animal) VALUES
                                               (3, 1),
                                               (2, 2),
                                               (4, 3),
                                               (5, 5);

INSERT INTO Feed (Name, UnitOfMeasurement) VALUES
                                              ('Feed1', 'Kg'),
                                              ('Feed2', 'Kg'),
```



```
                                              ('Feed3', 'Kg'),
                                              ('Feed4', 'Kg'),
                                              ('Feed5', 'Kg');

-- Insert into Amount
INSERT INTO Amount (Amount, NameFeed) VALUES
                                         (10, 'Feed1'),
                                         (20, 'Feed2'),
                                         (2, 'Feed3'),
                                         (3, 'Feed4'),
                                         (14, 'Feed5');


-- Insert into AmountOfFeed
INSERT INTO AmountOfFeed (ID_Feed, ID_Amount) VALUES
                                                 (1, 1),
                                                 (2, 2),
                                                 (2, 4),
                                                 (3, 5),
                                                 (4, 2);


-- Insert into ToEat
INSERT INTO ToEat (ID_Feed, ID_Animal) VALUES
                                          (1, 1),
                                          (2, 2),
                                          (3, 3),
                                          (4, 4),
                                          (5, 5),
                                          (1, 6),
                                          (3, 7),
                                          (2, 8),
                                          (4, 9),
                                          (5, 10);


-- Insert into Care
INSERT INTO Care (ID_Employee, ID_Animal) VALUES
                                             (1, 1),
                                             (2, 5),
                                             (3, 2),
                                             (4, 2),
                                             (5, 1),
                                             (6, 2),
                                             (7, 3),
                                             (8, 4),
                                             (9, 7),
                                             (10, 8);
```

4.Select

1)SELECT Zoo.Name, Employee.FirstName
FROM Zoo
LEFT JOIN Employee ON Zoo.Name = Employee.NameZoo;

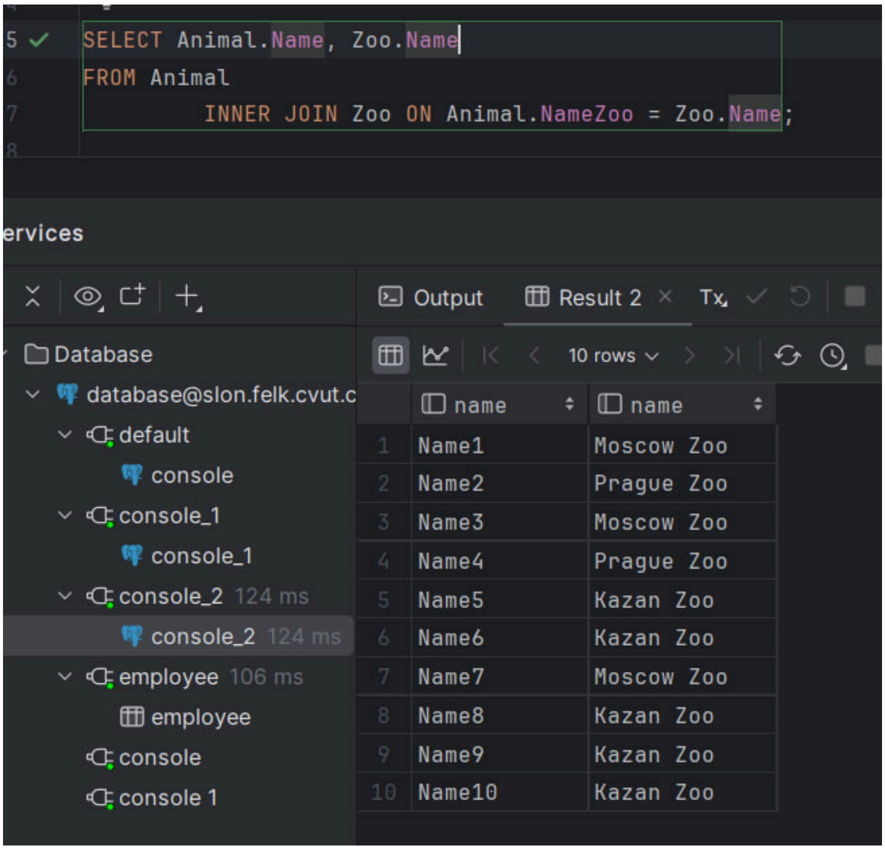


```
SELECT Zoo.Name, Employee.FirstName
FROM Zoo
        LEFT JOIN Employee ON Zoo.Name = Employee.NameZoo;
```

	name	firstname
1	Kazan Zoo	Michael
2	Kazan Zoo	Sarah
3	Tokyo Zoo	Jessica
4	Tokyo Zoo	Jacob
5	Prague Zoo	Emily
6	Prague Zoo	Amanda
7	Karachi Zoo	Michael
8	Karachi Zoo	Andrew
9	Shanghai Zoo	Patricia
10	Lagos Zoo	Patricia
11	Dhaka Zoo	<null>
12	Osaka Zoo	<null>
13	Moscow Zoo	<null>
14	Delhi Zoo	<null>

Description
This query selects the names of zoos and the names of employees. It performs a LEFT JOIN on the Zoo and Employee tables based on the Name field. If there is no employee for a zoo, the query will still return the name of the zoo, but the employee name will be NULL.

2)SELECT Animal.Name, Zoo.Name
FROM Animal
INNER JOIN Zoo ON Animal.NameZoo = Zoo.Name;

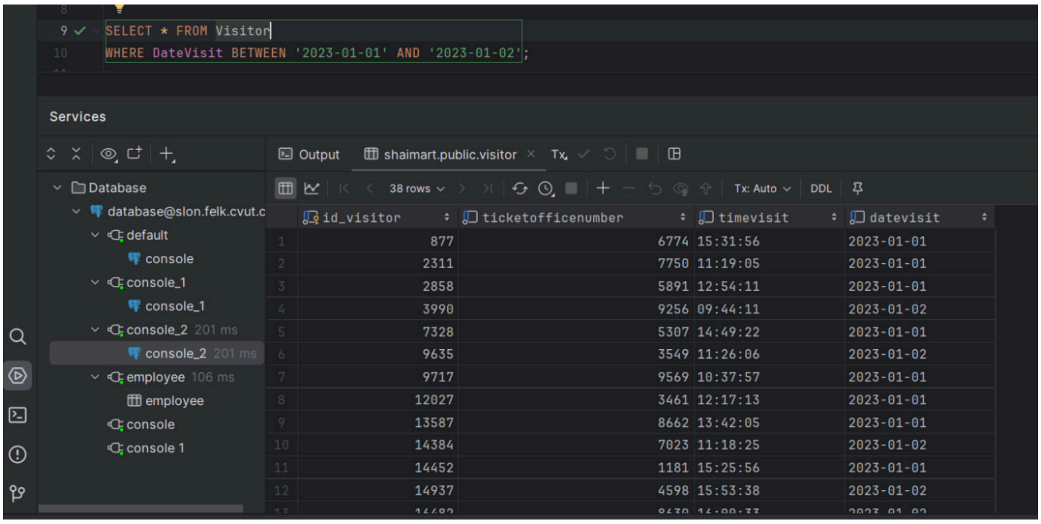


```
SELECT Animal.Name, Zoo.Name
FROM Animal
         INNER JOIN Zoo ON Animal.NameZoo = Zoo.Name;
```

	name	name
1	Name1	Moscow Zoo
2	Name2	Prague Zoo
3	Name3	Moscow Zoo
4	Name4	Prague Zoo
5	Name5	Kazan Zoo
6	Name6	Kazan Zoo
7	Name7	Moscow Zoo
8	Name8	Kazan Zoo
9	Name9	Kazan Zoo
10	Name10	Kazan Zoo

Description
This query selects the names of animals and the names of the zoos where they are located. It performs an INNER JOIN on the Animal and Zoo tables based on the Name field. If there is no zoo for an animal, it will not be included in the result.

3)SELECT * FROM Visitor
WHERE DateVisit BETWEEN '2023-01-01' AND '2023-01-02';



```
SELECT * FROM Visitor
WHERE DateVisit BETWEEN '2023-01-01' AND '2023-01-02';
```

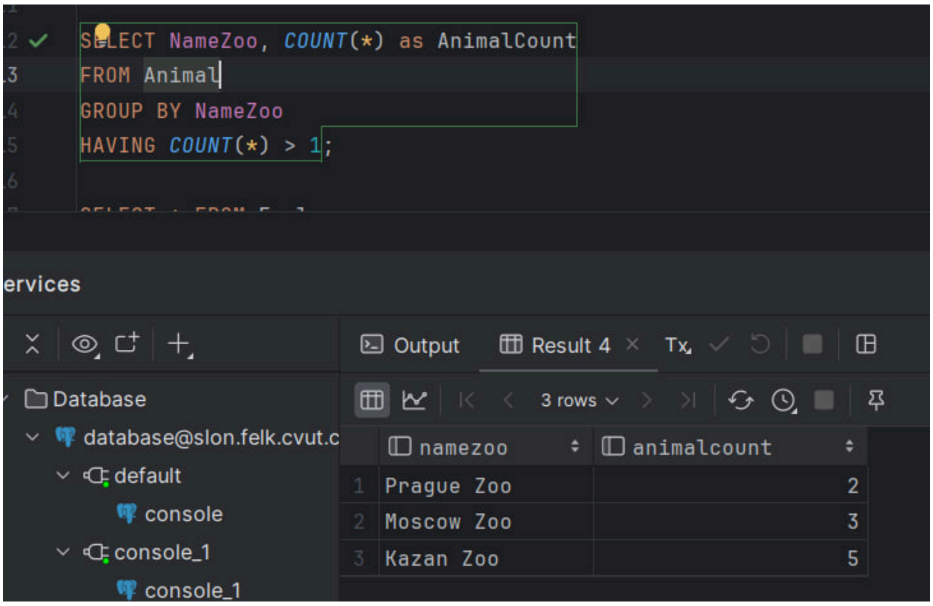
	id_visitor	ticketofficenumber	timevisit	datevisit
1	877	6774	15:31:56	2023-01-01
2	2311	7750	11:19:05	2023-01-01
3	2858	5891	12:54:11	2023-01-01
4	3990	9256	09:44:11	2023-01-02
5	7328	5307	14:49:22	2023-01-01
6	9635	3549	11:26:06	2023-01-02
7	9717	9569	10:37:57	2023-01-01
8	12027	3461	12:17:13	2023-01-01
9	13587	8662	13:42:05	2023-01-01
10	14384	7023	11:18:25	2023-01-02
11	14452	1181	15:25:56	2023-01-01
12	14937	4598	15:53:38	2023-01-02

Description
This query selects all visitors who visited the zoo between '2023-01-01' and '2023-01-02'.
P.S. The Visitor table is populated with 32k records.
by using this script :

```
from faker import Faker
import psycopg2
import random
```

```
from datetime import datetime, timedelta, date

conn= psycopg2.connect(
    dbname="",
    user="",
    password="",
    host="slon.felk.cvut.cz",
    port="",
)
cur=conn.cursor()

fake = Faker()
num_records = 32000
for i in range(num_records):
    TicketOfficeNumber = random.randint(1, 10000)
    TimeVisit = (datetime.now().replace(hour=random.randint(9, 18),
minute=random.randint(0, 59), second=random.randint(0, 59))).time()
    DateVisit = fake.date_between(start_date=date(2020, 1, 1),
end_date='today')
    print(f'TicketOfficeNumber: {TicketOfficeNumber}, TimeVisit:
{TimeVisit}, DateVisit: {DateVisit}')
    cur.execute("INSERT INTO Visitor (TicketOfficeNumber, TimeVisit,
DateVisit) VALUES (%s, %s, %s) ON CONFLICT DO NOTHING",
            (TicketOfficeNumber, TimeVisit, DateVisit))

conn.commit()
```

4)SELECT NameZoo, COUNT(*) as AnimalCount
FROM Animal
GROUP BY NameZoo
HAVING COUNT(*) > 1;



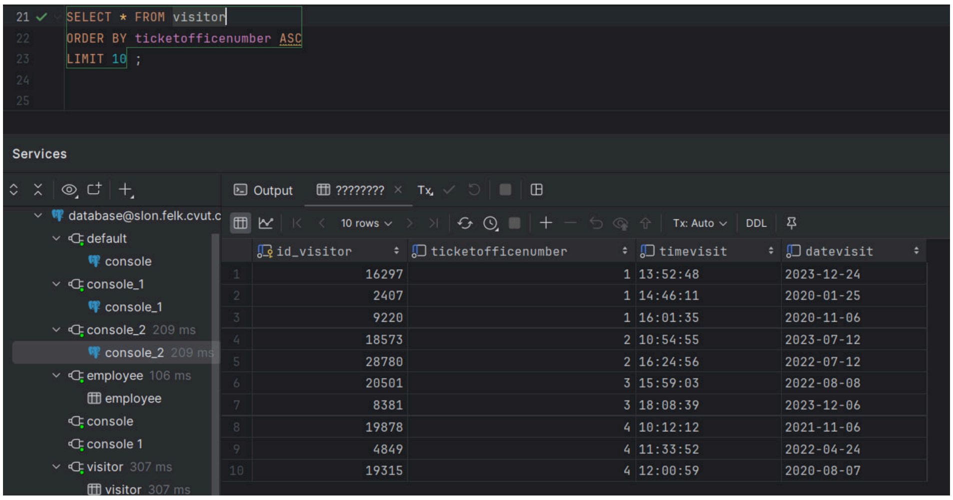
```
SELECT NameZoo, COUNT(*) as AnimalCount
FROM Animal
GROUP BY NameZoo
HAVING COUNT(*) > 1;
```

	namezoo	animalcount
1	Prague Zoo	2
2	Moscow Zoo	3
3	Kazan Zoo	5

Description

This query selects the names of zoos and the number of animals in each zoo. It groups the results by the NameZoo field and counts the number of animals in each group. It then selects only those groups where the number of animals is greater than 1.

5)SELECT * FROM visitor
ORDER BY ticketofficenumber ASC
LIMIT 10 ;



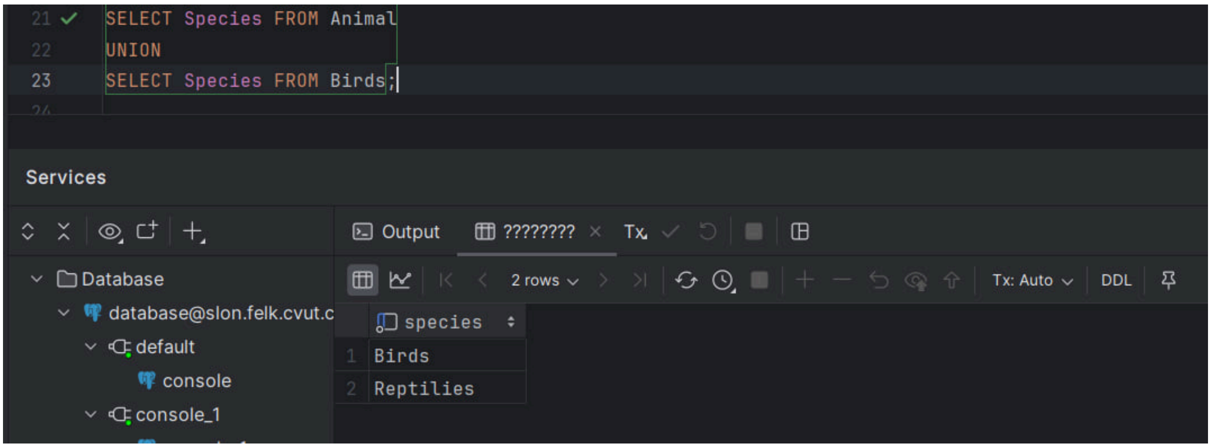
```
SELECT * FROM visitor
ORDER BY ticketofficenumber ASC
LIMIT 10 ;
```

	id_visitor	ticketofficenumber	timevisit	datevisit
1	16297	1	13:52:48	2023-12-24
2	2407	1	14:46:11	2020-01-25
3	9220	1	16:01:35	2020-11-06
4	18573	2	10:54:55	2023-07-12
5	28780	2	16:24:56	2022-07-12
6	20501	3	15:59:03	2022-08-08
7	8381	3	18:08:39	2023-12-06
8	19878	4	10:12:12	2021-11-06
9	4849	4	11:33:52	2022-04-24
10	19315	4	12:00:59	2020-08-07

Description

This query selects all visitors, sorts them in ascending order by ticket office number, and limits the result to the first 10 records.

6)SELECT Species FROM Animal
UNION
SELECT Species FROM Birds;

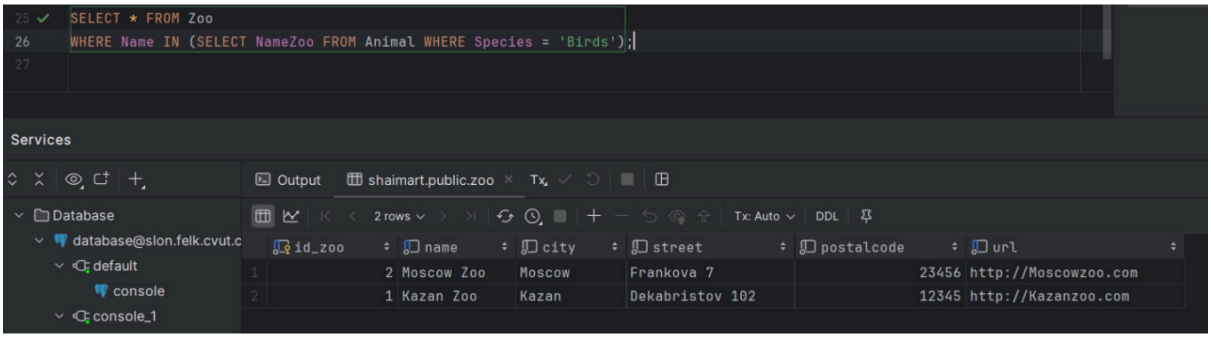


```
SELECT Species FROM Animal
UNION
SELECT Species FROM Birds;
```

	species
1	Birds
2	Reptilies

Description
This query selects all unique species that are present in both the Animal and Birds tables. However, since the Birds table contains only birds, the result will include species of birds and reptilies.

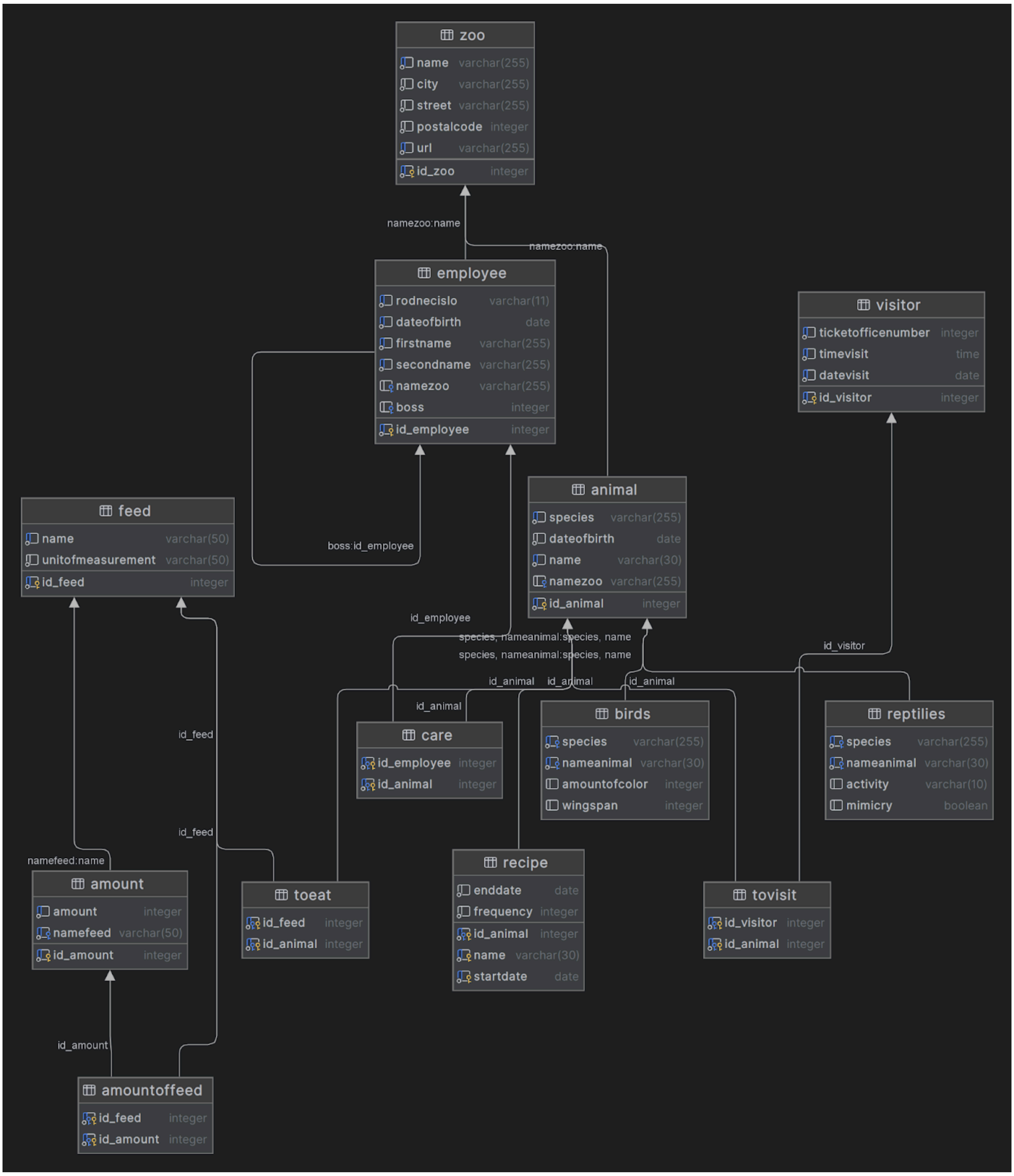
7) SELECT * FROM Zoo
WHERE Name IN (SELECT NameZoo FROM Animal WHERE Species = 'Birds');



```
SELECT * FROM Zoo
WHERE Name IN (SELECT NameZoo FROM Animal WHERE Species = 'Birds');
```

	id_zoo	name	city	street	postalcode	url
1	2	Moscow Zoo	Moscow	Frankova 7	23456	http://Moscowzoo.com
2	1	Kazan Zoo	Kazan	Dekabristov 102	12345	http://Kazanzoo.com

Description
This query selects all zoos that contain birds. It uses a nested query to select the names of zoos from the Animal table where the species is equal to 'Birds', and then selects all records from the Zoo table where the Name is present in the results of the nested query.



Advanced Database Technologies

Objective: Extend the database with advanced technologies.
Demonstrations:

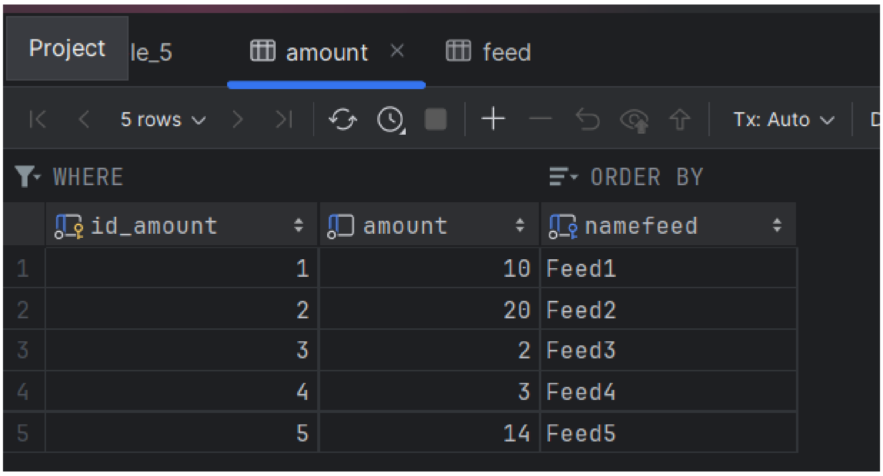
- Transaction call and query set with appropriate isolation levels and conflict explanation.
- Creation and usage of a view.
- Creation and usage of a trigger.
- Creation and usage of an index with an analysis of its benefits.

1. Invocation of a transaction and a set of queries including setting an appropriate isolation level, specify a conflict that could arise if a transaction were not used.

```
BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
UPDATE Amount
SET Amount = Amount + 10 WHERE namefeed = 'Feed4';
UPDATE Amount
SET Amount = Amount - 1 WHERE namefeed = 'Feed1';

COMMIT;
```

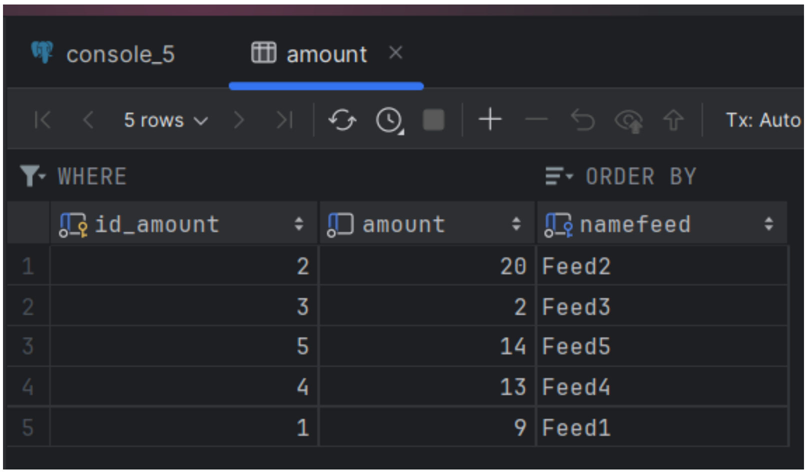
The following conflicts could arise:
If another process attempts to update the amount of feed (Amount) at the same time we are trying to increase it by 10 for Feed4 and decrease it by 1 for Feed1, it could lead to "dirty reads" or "non-repeatable reads." This means that another process may read inconsistent data or see that the data is changing during its operation.

Before:



	id_amount	amount	namefeed
1	1	10	Feed1
2	2	20	Feed2
3	3	2	Feed3
4	4	3	Feed4
5	5	14	Feed5

After:



	id_amount	amount	namefeed
1	2	20	Feed2
2	3	2	Feed3
3	5	14	Feed5
4	4	13	Feed4
5	1	9	Feed1

2. Creating and Using a View

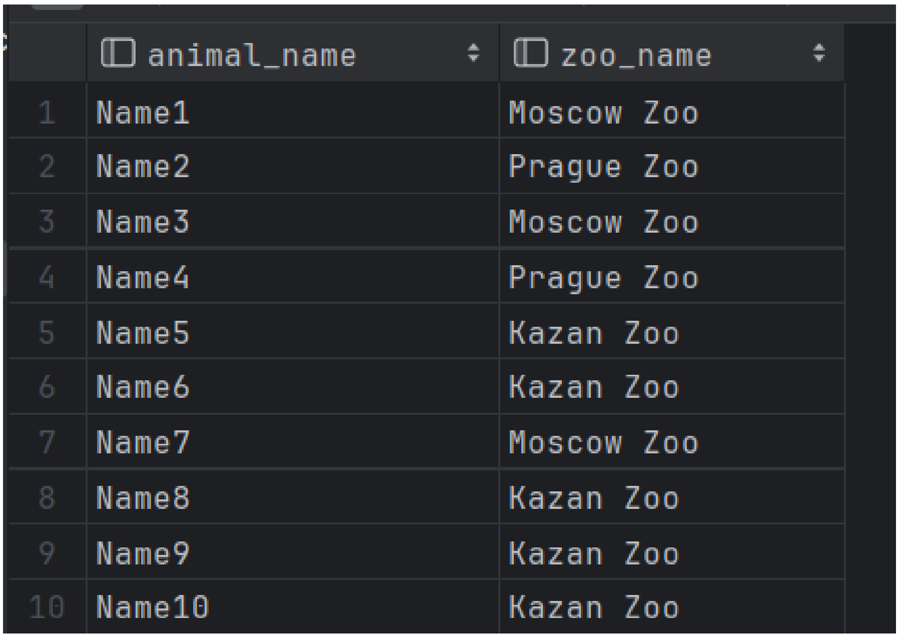
Creating

```
CREATE VIEW animalZoo AS
```

```
SELECT Animal.Name AS Animal_Name, Zoo.Name AS Zoo_Name
FROM Animal
INNER JOIN Zoo ON Animal.NameZoo = Zoo.Name;
```

Using

```
select * from animalZoo;
```



	animal_name	zoo_name
1	Name1	Moscow Zoo
2	Name2	Prague Zoo
3	Name3	Moscow Zoo
4	Name4	Prague Zoo
5	Name5	Kazan Zoo
6	Name6	Kazan Zoo
7	Name7	Moscow Zoo
8	Name8	Kazan Zoo
9	Name9	Kazan Zoo
10	Name10	Kazan Zoo

3. Creating and Using a Triggers

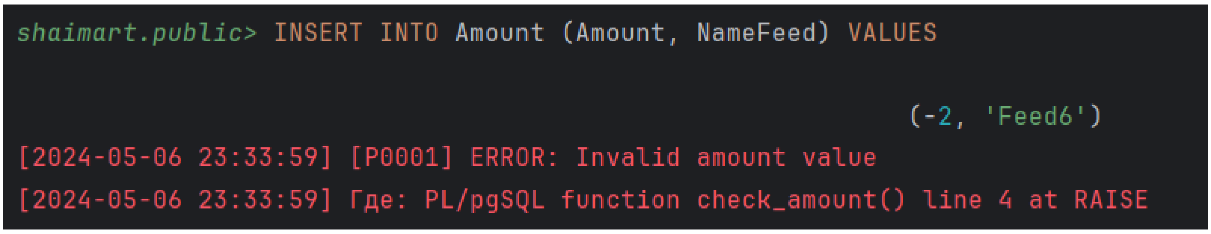
Creating

```
CREATE FUNCTION check_amount()

   RETURNS TRIGGER

AS $$

BEGIN

   IF ((NEW.amount IS NULL) OR (NEW.amount < 0)) THEN

       RAISE EXCEPTION 'Invalid amount value';

   END IF;

   RETURN NEW;

END

$$ LANGUAGE plpgsql;


CREATE TRIGGER amount_tg_check BEFORE INSERT OR UPDATE ON
amount
```

```
    FOR EACH ROW EXECUTE PROCEDURE check_amount();
```

Using

```
INSERT INTO Amount (Amount, NameFeed) VALUES


                                                                  (-2, 'Feed6');
```

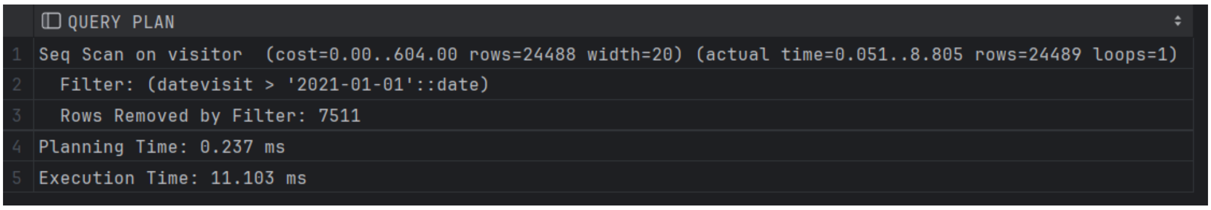


```
shaimart.public> INSERT INTO Amount (Amount, NameFeed) VALUES

                                                                  (-2, 'Feed6')
[2024-05-06 23:33:59] [P0001] ERROR: Invalid amount value
[2024-05-06 23:33:59] Где: PL/pgSQL function check_amount() line 4 at RAISE
```

4. Creating and Using a Index

Without

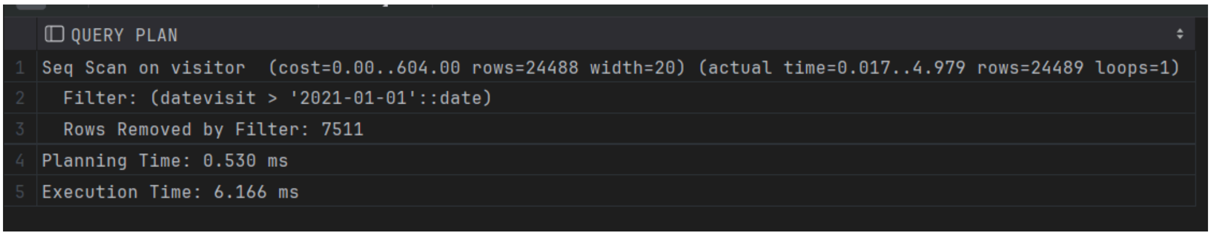
```
EXPLAIN ANALYZE
SELECT * FROM Visitor
WHERE DateVisit > '2021-01-01';
```



```
QUERY PLAN
Seq Scan on visitor  (cost=0.00..604.00 rows=24488 width=20) (actual time=0.051..8.805 rows=24489 loops=1)
  Filter: (datevisit > '2021-01-01'::date)
  Rows Removed by Filter: 7511
Planning Time: 0.237 ms
Execution Time: 11.103 ms
```

With

```
CREATE INDEX idx_date ON Visitor(DateVisit);
EXPLAIN ANALYZE
SELECT * FROM Visitor
WHERE DateVisit > '2021-01-01';
```



```
QUERY PLAN
Seq Scan on visitor  (cost=0.00..604.00 rows=24488 width=20) (actual time=0.017..4.979 rows=24489 loops=1)
  Filter: (datevisit > '2021-01-01'::date)
  Rows Removed by Filter: 7511
Planning Time: 0.530 ms
Execution Time: 6.166 ms
```

JPA Application Foundation

You can find this part on my GitHub in the "Code" section.
Where I added both automatically generated entities and Dao.
In my case, I implemented inheritance, transactions, and CRUD operations.