report

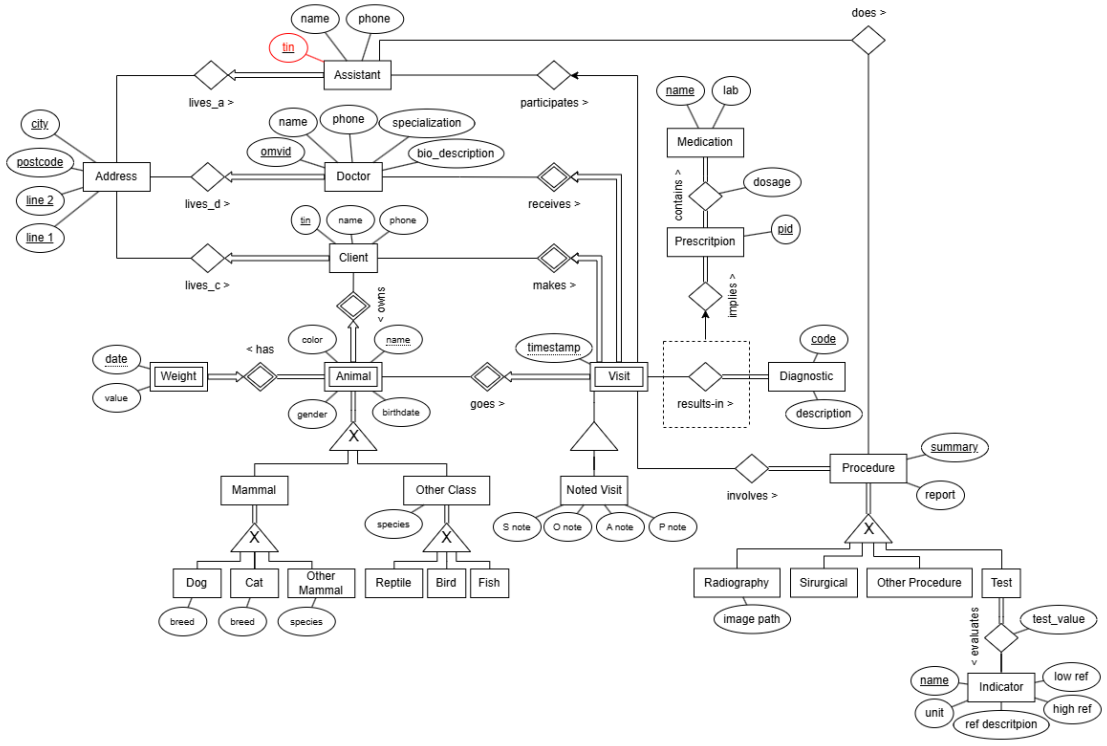
July 31, 2025

1 Small Company DB Project

1.1 PART I – E-R Model

1. Proposed database design

1.1.1 E-R Model



Notes - ER Model n this section, the reasoning behind certain design decisions in the model is explained.

For the Address entity, every attribute is used to form the primary key to ensure that no repetition occurs when some line, postal code, or city are the same, or when there is a combination of these elements.

For the Assistant entity, it is suggested that the client also store the TIN, as none of the known attributes are strong enough to be used as identifiers. Their uniqueness cannot be guaranteed. The

TIN is a standard way to identify a person, as each individual has a distinct TIN, which is also used for clients.

The Animal entity is modeled as a weak entity, since it only makes sense when associated with a client. This design ensures that a client may have several animals and different clients may have animals with the same name. Because an animal can be a mammal, reptile, bird, or fish—and within mammals, there is a desire to differentiate between dogs, cats, and other mammals—the model specializes animals into mammals and other categories. In the non-mammal categories, the species is stored. Within mammals, further specialization into dogs, cats, and other mammals is done, and for dogs and cats, the breed is stored. Although the non-mammal classes (reptiles, birds, and fish) do not have distinct attributes between them, they are specialized nonetheless, as they are considered 'first-class citizens' within the domain.

For an animal's most recent weight, a weak entity is also used, as weight only has meaning when associated with a specific animal. Weight can change throughout an animal's life, and different animals can have the same weight. To ensure the most recent weight is stored, the date of the measurement is included.

The Visit entity is modeled as a weak entity of client, doctor, and animal, because a visit involves all three and only makes sense when associated with a specific record of each. However, different visits may involve the same doctor, client, and animal. Since the client may not necessarily be the owner of the animal, the Client must be modeled as a strong entity related to the visit. The visit is identified using the date as a partial key, but this is stored as a timestamp, as there are scenarios where the same doctor, client, and animal might meet multiple times in the same day. Because the visit is a weak entity relative to the other three entities, no separate tables are created for the associations "goes", "receives", or "makes". Each visit may have some SOAP notes associated with it; since these notes are not mandatory, they are modeled as a specialization of the visit.

Each visit may result in one or more diagnostics, and when there is at least one diagnostic, there may be a prescription of medications. Therefore, the prescription is modeled as an optional leg, where all prescriptions must be associated with a visit-diagnostic association. Each instance of this association may only be linked to one prescription.

Since the model relates to the activities of a veterinary clinic, only the medications used in prescriptions are stored. All medications must appear in at least one prescription.

For the Procedures, the summary is used as the primary key, based on the assumption that the summary will always be unique for each procedure performed in the clinic. Using the full report as a primary key would be too costly.

1.2 PART II – Relational Model

1.2.1 Database Schema

1. Create the tables and integrity constraints corresponding to the relational database schema obtained.

[ ]:
```
%load_ext sql
%sql postgresql+psycopg://db:db@postgres/db
```

```
Connecting to 'postgresql+psycopg://db:***@postgres/db'
```

DROP TABLES

[ ]:
```
%%sql

DROP TABLE IF EXISTS client CASCADE;
DROP TABLE IF EXISTS doctor CASCADE;
DROP TABLE IF EXISTS assistant CASCADE;
DROP TABLE IF EXISTS adress CASCADE;
DROP TABLE IF EXISTS animal CASCADE;
DROP TABLE IF EXISTS weight;
DROP TABLE IF EXISTS mammal CASCADE;
DROP TABLE IF EXISTS dog;
DROP TABLE IF EXISTS cat;
DROP TABLE IF EXISTS other_mammal;
DROP TABLE IF EXISTS other_class CASCADE;
DROP TABLE IF EXISTS reptile CASCADE;
DROP TABLE IF EXISTS bird CASCADE;
DROP TABLE IF EXISTS fish CASCADE;
DROP TABLE IF EXISTS visit CASCADE;
DROP TABLE IF EXISTS noted_visit;
DROP TABLE IF EXISTS diagnostic CASCADE;
DROP TABLE IF EXISTS prescription CASCADE;
DROP TABLE IF EXISTS medication CASCADE;
DROP TABLE IF EXISTS procedures CASCADE;
DROP TABLE IF EXISTS radiography;
DROP TABLE IF EXISTS sirurgical;
DROP TABLE IF EXISTS other_procedure;
DROP TABLE IF EXISTS test CASCADE;
DROP TABLE IF EXISTS test_indicator CASCADE;
```

```
Running query in 'postgresql+psycopg://db:***@postgres/db'
```

[ ]:
```
++
||
++
++
```

[ ]:
```
%%sql

DROP TABLE IF EXISTS lives_a;
DROP TABLE IF EXISTS lives_d;
DROP TABLE IF EXISTS lives_c;
DROP TABLE IF EXISTS participates;
DROP TABLE IF EXISTS does;
DROP TABLE IF EXISTS results_in CASCADE;
DROP TABLE IF EXISTS implies CASCADE;
DROP TABLE IF EXISTS contains CASCADE;
DROP TABLE IF EXISTS involves CASCADE;
DROP TABLE IF EXISTS evaluates CASCADE;
```

```
Running query in 'postgresql+psycopg://db:***@postgres/db'
```

[ ]:
```
++
||
++
++
```

Entities

[ ]:
```
%%sql

CREATE TABLE client (
    tin NUMERIC(9),
    name VARCHAR(80) NOT NULL,
    phone NUMERIC(9) NOT NULL,

    PRIMARY KEY (tin)
);

CREATE TABLE doctor (
    omvid INTEGER,
    name VARCHAR(80) NOT NULL,
    phone NUMERIC(9) NOT NULL,
    specialization VARCHAR(50) NOT NULL,
    bio_description TEXT NOT NULL,

    PRIMARY KEY (omvid)
);

CREATE TABLE assistant (
    tin_a NUMERIC(9),
    name VARCHAR(80) NOT NULL,
    phone NUMERIC(9) NOT NULL,

    PRIMARY KEY (tin_a)
);

CREATE TABLE adress (
    line_1 VARCHAR(255),
    line_2 VARCHAR(255),
    postcode VARCHAR(12),
    city VARCHAR(30),

    PRIMARY KEY (line_1, line_2, postcode, city)
);

CREATE TABLE animal (
    tin NUMERIC(9),
```

```
    name VARCHAR(80),
    color VARCHAR(50) NOT NULL,
    gender VARCHAR(20) NOT NULL,
    birthdate DATE NOT NULL,

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin) REFERENCES client(tin)

    -- Every animal must exist either in the table 'mammal' or in the table␣
 ↪'other_class'.
    -- No animal can exist at the same time in the both the table 'mammal' or␣
 ↪in the table 'other_class'.
);

CREATE TABLE weight (
    tin NUMERIC(9),
    name VARCHAR(80),
    date DATE,
    w_value NUMERIC(5,2) NOT NULL,

    PRIMARY KEY (tin, name, w_value),
    FOREIGN KEY (tin, name) REFERENCES animal(tin, name)
);

CREATE TABLE mammal (
    tin NUMERIC(9),
    name VARCHAR(80),

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES animal(tin, name)

    -- Every mammal must exist either in the table 'dog' or in the table 'cat'␣
 ↪or in the table 'other_mammal'.
    -- No mammal can exist at the same time in more than one of the tables␣
 ↪'dog', 'cat' or 'other_mammal'.
);

CREATE TABLE dog (
    tin NUMERIC(9),
    name VARCHAR(80),
    breed VARCHAR (50) NOT NULL,

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES mammal(tin, name)
);

CREATE TABLE cat (
```

```
    tin NUMERIC(9),
    name VARCHAR(80),
    breed VARCHAR (50) NOT NULL,

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES mammal(tin, name)
);

CREATE TABLE other_mammal (
    tin NUMERIC(9),
    name VARCHAR(80),
    species VARCHAR(50) NOT NULL,

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES mammal(tin, name)
);

CREATE TABLE other_class (
    tin NUMERIC(9),
    name VARCHAR(80),
    species VARCHAR(50) NOT NULL,

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES animal(tin, name)

    -- Every other class animal must exist either in the table 'reptile' or in␣
 ↪the table 'bird' or in the table 'fish'.
    -- No other class animal can exist at the same time in more than one of the␣
 ↪tables 'reptile', 'bird' or 'fish'.
);

CREATE TABLE reptile (
    tin NUMERIC(9),
    name VARCHAR(80),

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES other_class(tin, name)
);

CREATE TABLE bird (
    tin NUMERIC(9),
    name VARCHAR(80),

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES other_class(tin, name)
);
```

```
CREATE TABLE fish (
    tin NUMERIC(9),
    name VARCHAR(80),

    PRIMARY KEY (tin, name),
    FOREIGN KEY (tin, name) REFERENCES other_class(tin, name)
);

CREATE TABLE visit (
    tin NUMERIC(9),
    omvid INTEGER,
    name VARCHAR(80),
    date TIMESTAMP,
    tin_owner NUMERIC(9),

    PRIMARY KEY (tin, omvid, name, date, tin_owner),
    FOREIGN KEY (tin_owner, name) REFERENCES animal(tin, name),
    FOREIGN KEY (tin) REFERENCES client(tin),
    FOREIGN KEY (omvid) REFERENCES doctor(omvid)
);

CREATE TABLE noted_visit (
    tin NUMERIC(9),
    omvid INTEGER,
    name VARCHAR(80),
    date TIMESTAMP,
    tin_owner NUMERIC(9),
    s_note TEXT NOT NULL,
    o_note TEXT NOT NULL,
    a_note TEXT NOT NULL,
    p_note TEXT NOT NULL,

    PRIMARY KEY (tin, omvid, name, date, tin_owner),
    FOREIGN KEY (tin, omvid, name, date, tin_owner) REFERENCES visit(tin,␣
  ↪omvid, name, date, tin_owner)
);

CREATE TABLE diagnostic (
    code VARCHAR(20),
    description TEXT NOT NULL,

    PRIMARY KEY (code)

    -- Every diagnostic has to participate in table 'results_in' at least once
);

CREATE TABLE prescription (
```

```
    pid INTEGER,

    PRIMARY KEY (pid)

    -- Every prescription has to participate in table 'implies'
    -- Every prescription has to participate in table 'contains'
);

CREATE TABLE medication (
    name VARCHAR(40),
    lab VARCHAR(200) NOT NULL,

    PRIMARY KEY (name)

    -- Every medication has to participate in table 'contains'
);

CREATE TABLE procedures (
    summary VARCHAR(255) NOT NULL,
    report TEXT NOT NULL,

    PRIMARY KEY (summary)

    -- Every procedure must exist either in the table 'radiagrophy', in the␣
 ↪table 'sirurgical', in the table 'test' or in the table 'other_procedure'.
    -- No procedure can exist at the same time in more than one of tables␣
 ↪'radiagrophy', 'sirurgical', 'test' or 'other_procedure'.
    -- Every procedure must participate in table 'involves'
);

CREATE TABLE radiography (
    summary VARCHAR(255),
    image_path TEXT NOT NULL,

    PRIMARY KEY (summary),
    FOREIGN KEY (summary) REFERENCES procedures(summary)
);

CREATE TABLE sirurgical (
    summary VARCHAR(255),

    PRIMARY KEY (summary),
    FOREIGN KEY (summary) REFERENCES procedures(summary)
);

CREATE TABLE other_procedure (
    summary VARCHAR(255),
```

```
    PRIMARY KEY (summary),
    FOREIGN KEY (summary) REFERENCES procedures(summary)
);

CREATE TABLE test (
    summary VARCHAR(255),

    PRIMARY KEY (summary),
    FOREIGN KEY (summary) REFERENCES procedures(summary)
);

CREATE TABLE test_indicator (
    name VARCHAR(50),
    unit VARCHAR(10) NOT NULL,
    low_ref NUMERIC(6,2) NOT NULL,
    high_ref NUMERIC(6,2) NOT NULL,
    ref_description TEXT NOT NULL,

    PRIMARY KEY (name)
);
```

```
Running query in 'postgresql+psycopg://db:***@postgres/db'
```

[ ]:
```
++
||
++
++
```

Associations

[ ]:
```
%%sql

CREATE TABLE lives_a (
    line_1 VARCHAR(255),
    line_2 VARCHAR(255),
    postcode VARCHAR(12),
    city VARCHAR(30),
    tin_a NUMERIC(9),

    PRIMARY KEY (line_1, line_2, postcode, city, tin_a),
    FOREIGN KEY (line_1, line_2, postcode, city) REFERENCES adress (line_1,␣
 ↪line_2, postcode, city),
    FOREIGN KEY (tin_a) REFERENCES assistant (tin_a)
);

CREATE TABLE lives_d (
    line_1 VARCHAR(255),
```

```
    line_2 VARCHAR(255),
    postcode VARCHAR(12),
    city VARCHAR(30),
    omvid INTEGER,

    PRIMARY KEY (line_1, line_2, postcode, city, omvid),
    FOREIGN KEY (line_1, line_2, postcode, city) REFERENCES adress (line_1,␣
 ↪line_2, postcode, city),
    FOREIGN KEY (omvid) REFERENCES doctor (omvid)
);

CREATE TABLE lives_c (
    line_1 VARCHAR(255),
    line_2 VARCHAR(255),
    postcode VARCHAR(12),
    city VARCHAR(30),
    tin NUMERIC(9),

    PRIMARY KEY (line_1, line_2, postcode, city, tin),
    FOREIGN KEY (line_1, line_2, postcode, city) REFERENCES adress (line_1,␣
 ↪line_2, postcode, city),
    FOREIGN KEY (tin) REFERENCES client (tin)
);

CREATE TABLE participates (
    tin_a NUMERIC(9),
    tin NUMERIC(9),
    omvid INTEGER,
    name VARCHAR(80),
    date TIMESTAMP,
    tin_owner NUMERIC(9),

    PRIMARY KEY (tin, omvid, name, date, tin_owner),
    FOREIGN KEY (tin, omvid, name, date, tin_owner) REFERENCES visit(tin,␣
 ↪omvid, name, date, tin_owner),
    FOREIGN KEY (tin_a) REFERENCES assistant (tin_a)
);

CREATE TABLE does (
    summary VARCHAR(255),
    tin_a NUMERIC(9),

    PRIMARY KEY (summary, tin_a),
    FOREIGN KEY (summary) REFERENCES procedures (summary),
    FOREIGN KEY (tin_a) REFERENCES assistant (tin_a)
);
```

```
CREATE TABLE results_in (
    tin NUMERIC(9),
    code VARCHAR(20),
    omvid INTEGER,
    name VARCHAR(80),
    date TIMESTAMP,
    tin_owner NUMERIC(9),

    PRIMARY KEY (tin, omvid, name, date, tin_owner, code),
    FOREIGN KEY (tin, omvid, name, date, tin_owner) REFERENCES visit(tin,␣
 ↪omvid, name, date, tin_owner),
    FOREIGN KEY (code) REFERENCES diagnostic(code)
);

CREATE TABLE implies (
    code VARCHAR(20),
    tin NUMERIC(9),
    omvid INTEGER,
    name VARCHAR(80),
    date TIMESTAMP,
    tin_owner NUMERIC(9),
    pid INTEGER,

    PRIMARY KEY (tin, omvid, name, date, tin_owner, code),
    FOREIGN KEY (tin, omvid, name, date, tin_owner, code) REFERENCES␣
 ↪results_in(tin, omvid, name, date, tin_owner, code),
    FOREIGN KEY (pid) REFERENCES prescription(pid)
);

CREATE TABLE contains (
    pid INTEGER,
    name VARCHAR(40),
    dosage VARCHAR(20) NOT NULL,

    PRIMARY KEY (pid, name),
    FOREIGN KEY (pid) REFERENCES prescription (pid),
    FOREIGN KEY (name) REFERENCES medication (name)
);

CREATE TABLE involves (
    summary VARCHAR(255),
    tin NUMERIC(9),
    omvid INTEGER,
    name VARCHAR(80),
    date TIMESTAMP,
    tin_owner NUMERIC(9),
```

```
    PRIMARY KEY (tin, omvid, name, date, tin_owner, summary),
    FOREIGN KEY (tin, omvid, name, date, tin_owner) REFERENCES visit(tin,␣
 ↪omvid, name, date, tin_owner),
    FOREIGN KEY (summary) REFERENCES procedures(summary)
);

CREATE TABLE evaluates (
    summary VARCHAR(255),
    name VARCHAR(50),
    test_value NUMERIC(6,2) NOT NULL,

    PRIMARY KEY (summary, name),
    FOREIGN KEY (summary) REFERENCES test (summary),
    FOREIGN KEY (name) REFERENCES test_indicator(name)
);
```

```
Running query in 'postgresql+psycopg://db:***@postgres/db'
```

[ ]:
```
++
||
++
++
```