Phase IV - Database Model
Group #7 - Liam Marquis, Jenna Weldon, Tyler Davis
3/29/2024

Task 2: Normalization

Demonstrate that all the relations in the relational schema are normalized to Boyce–Codd normal form (BCNF).

- **For each table, specify whether it is in BCNF or not, and explain why.**
- **For each table that is not in BCNF, show the complete process that normalizes it to BCNF**

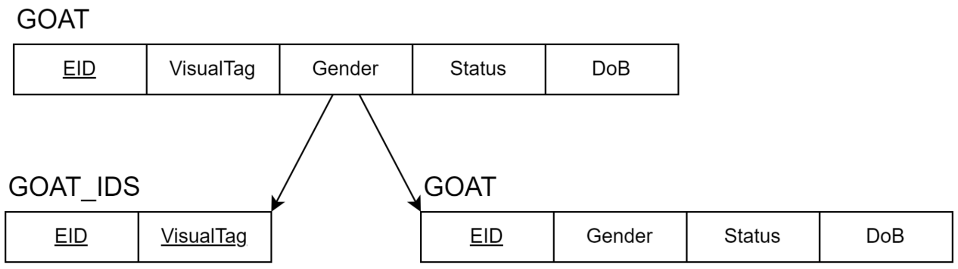
Relations:

1. GOAT

GOAT

EID	VisualTag	Gender	Status	DoB

While GOAT currently satisfies 2nd and 3rd normal forms, it does not satisfy BCNF. This issue we identified is that while gender, status and DoB cannot give us the value of EID on their own, VisualTag might be able to. In order to normalize GOAT to BCNF, we need to split the GOAT table in two with one new relation for GOAT_IDS which includes both EID and VisualTag as its candidate key and GOAT which maps EID to Gender, Status, and DoB as shown below.



After splitting the GOAT table into a relation of GOAT_IDS and GOAT, it is now normalized as it still satisfies 2nd and 3rd normal forms, and it now is BCNF with no violations regarding VisualTag.

2. WEIGHT

WEIGHT

WeightID	WeightValue	Date	Type

This is BCNF as it satisfies 2nd and 3rd normal form operating under a unique WeightID per weight session. And there are no cases in which the ID can be determined from the weightvalue, date, or type of weight.

3. NOTE

NOTE

NoteID	NoteText	CreatedDate

It satisfies 2nd and 3rd normal form and BCNF, and NoteID cannot be determined by the NoteText nor CreatedDate as the NoteID is unique and assigned to each note regardless of the date or its text.

4. PARENT_OF

PARENT_OF

ParentID	KidID	DamOrSire

Satisfies 2nd and 3rd normal form and BCNF. The ID's cannot be determined by whether or not the goat is a Dam or Sire. As there is no unique indicator in the id that is specific to the gender of the goat, it is a sequence of numbers.

5. WEIGHED

WEIGHED

GoatID	WeightID

There are no non-prime attributes in this relation, it satisfies 2nd normal form, and 3rd normal form; Therefore it is BCNF there's no cases where the ID's will have a form of dependency.

6. HAS

HAS

GoatID	NoteID

There are no non-prime attributes in this relation, so it satisfies 2nd normal form and 3rd normal form. Therefore it is BCNF because there is no form of dependencies.

Task 3: SQL Views

Define the different SQL views (virtual tables) required. For each view, list the data and transaction requirements. Give a few examples of queries, in English, to illustrate.

1. GOAT
    a. SQL

```
CREATE VIEW goat (eid, gender, status, dob) AS
SELECT rfid, sex, status, dob
FROM animal
```

    b. Requirements

For the data in the goat view, it is important that eid, as the primary key, remains unique for every goat. Additionally, gender and status must both only contain values from a set of possible values. For gender, this includes female, male, wether (a sterilized male), unknown and for status this includes current, sold, dead, off_farm. Finally, dob must be a valid date. It is important that this table remain consistent with the underlying animal table it is sourced from.

    c. Example Queries

- Display information for the goat with an EID of 964001009986941.
- List all female goats
- List all current goats born after 1/1/2016

2. GOAT_IDS
    a. SQL

```
CREATE VIEW goat_ids (eid, visualtag) AS
SELECT rfid, tag
FROM animal
```

    b. Requirements

For the transaction requirements of goat ids, it is important that for both the eid and visualtag, that these are distinct and unique values that are assigned to a particular goat. There should be no types of dependencies associated with these values such as particular numbers or letters associating with for example the gender of the goat. No two goats will share the same visualtag or eid.

    c. Example Queries

Find the VisualTag of the goat with an EID of 964001009986941.

3. WEIGHT
    a. SQL

```
CREATE VIEW weight (weightvalue, date, type) AS
SELECT alpha_value, when_measured, trait_code
FROM sessionanimaltrait
WHERE trait_code in {'53','357', '405','952','963','970'}
```

    b. Requirements

For the requirements of weight, it is important that the weightid, as the primary key, remains unique for each session of weighing and each goat. No two goats should share a weightid at any point, and no goat should have multiple weighing sessions fall under the same weight id. Additionally, value must contain a number (ex: 50, as in the goat weighs 50 lbs for example), and date must contain a specified date. Also, type must be from the specified weight types consisting of birth, weaning, winter, and sale weight.

    c. Example Queries

Determine the average live weight of goats measured on 9/14/2004

4. GNOTE
    a. SQL

```
CREATE VIEW note(notetext, createddate) AS
SELECT note, created
FROM note
```

    b. Requirements

The requirements of a note are that this note in particular has a unique id, unlike any other note no matter the context. It will also include when the note in particular was documented/created, and

    c. Example Queries

Determine when the contents of the note with id 12345, and the date that this note was created.

5. PARENT_OF
    a. SQL

```
CREATE VIEW parent_of (parent_id, kid_id, dam_or_sire) AS
(
SELECT dam, rfid, 'dam'
FROM animal
WHERE dam IS NOT NULL
)
```

```
UNION ALL
(
SELECT sire, rfid,'sire'
FROM animal
WHERE sire IS NOT NULL
)
```

b. Requirements

In determining the parent of a goat, there must be a unique parent id that is unique to only that parent goat, and there must be a unique kid id that is unique to only that kid. If known, then the dam of the sire will be included as well, the dam will have a unique id as well that it doesn't share with any other goat.

c. Example Queries

Get all the kids of the goat with EID = 123456789

6. WEIGHED

a. SQL

```
CREATE VIEW weighed (goatid,weightid) AS
SELECT animal_id, session_id
FROM sessionanimalactivity
```

b. Requirements

In the act of weighing a goat, requirements stand that a goat has an id that is unique to only that goat, and that this session of weighing has an id that is unique to only that session and goat.

c. Example Queries

Get all live weights for the goat with EID = 12345789

7. HAS

a. SQL

```
CREATE VIEW has(goatid,noteid) AS
SELECT animal_id, created
FROM note
```

b. Requirements

The requirements are that a goat has a unique id to itself that no other goat may share with it, and that it has a unique noteid associated with it as well that is unique to itself and that no other goat may share with it.

c. Example Queries

Get all notes for the goat with EID = 123456789

Task 4: SQL Queries

Design a complete set of SQL queries to satisfy the transaction requirements identified in the previous phases, using the relational schema and views defined in tasks 2 and 3 above. For each query, list the data and transaction requirements.

1. GOAT
    a. Display information for the goat with an EID of 964001009986941.

Requirements:

- EID must be unique for each goat

```
SELECT *
FROM goat
WHERE eid = '964001009986841';
```

    b. List all female goats

Requirements:

- Gender must have a valid value

```
SELECT *
FROM goat
WHERE gender = 'female';
```

    c. List all current goats born after 1/1/2016

Requirements

- dob must be a valid datetime
- status must have a valid value

```
SELECT *
FROM goat
WHERE status = 'current' AND dob > '1/1/2016';
```

2. GOAT_IDS
    a. Find the VisualTag of the goat with an EID of 964001009986941.

Requirements:

- All EIDs must be unique

```
SELECT visualtag
FROM goat_ids
WHERE eid = '964001009986941';
```

3. WEIGHT
    a. Determine the average live weight of goats measured on 9/14/2004

Requirements:

- weighttype must have a valid value for all tuples
- date must be valid datetime

```
SELECT AVG(weightvalue)
FROM weight
WHERE weighttype = '53' AND date = '9/14/2004';
```

4. NOTE
    a. Determine the contents of the note with id 12345, and the date that this note was created

Requirements:

- Date note created must be a valid date
- Must exist some form of contents for note with id of 12345.

```
SELECT *
FROM note
WHERE noteid = '12345'
```

5. PARENT_OF
    a. Get all the kids of the goat with EID = 123456789

Requirements:

- Each value for parent_id and kid_id must be valid EIDs for a tuple in the goat table.

```
SELECT *
FROM parent_of
WHERE parent_id = '123456789'
```

6. WEIGHED
    a. Get all weights for the goat with EID = 12345789

Requirements:

- Each value of animal_id must be a valid EID of a tuple from goat
- Each value of weight_id must be a valid weight_id of a tuple from weight

```
(
SELECT weight_id
FROM weighed
WHERE animal_id = '123456789'
)
JOIN weight;
```

7. HAS
    a. Get all notes for the goat with EID = 123456789

Requirements:

- Each value of goatid must be a valid EID of a tuple from goat
- Each value of noteid must be a valid EID of a tuple from note

```
SELECT *
FROM note
WHERE goatid = '123456789';
```

Task 5: Discussion

Explain how your views and queries defined in tasks 3 and 4 above implement your use cases.

Our use case “View Progeny Report’ allows the user to view a report of information about a selected goat. In order for us to deliver this feature to them we need the ability to retrieve this information from our database. All the above views and queries contribute to creating a report with the tables goat, goat_ids, weight, and note containing the bulk of the information while parent_of, weighed, and has connected those views together. Utilizing queries like the ones provided in task 4, we can gather up the information that we need before formatting it for the report.

Our other use case “View Genetic Trends” requires a bit more work than the previous use case. The function described by this use case is the ability to view how certain values such as weight matchup between family members and are visualized using our family tree. In order to build the family tree we make use of the goat and parent_of tables to build out the family tree of all goats. Connections between goats can be found using queries such as the given in task 4 for parent_of which returns all the tuples containing the kids of a certain goat and whether that goat was the dam or sire. Once the family tree is constructed the user will be able to select the metric they would like to view between all other goats in the family tree. Based on the connections that have been established with previous queries, we can use queries to locate the desired data from goat, weight, and note using the parent_of, weighed, and has views.