Basics of database systems

Project – Database design

Lappeenranta-Lahti University of Technology LUT
Software Engineering

Basics of database systems
Spring 2022

TABLE OF CONTENTS

1 DEFINITION

Fish database for tracking fish

In project “Fish database”, database is developed for a customer, who tracks and does research on fish. The database will be possibly target for a large-scale activity in the future, so the database should be well optimized and updating it should be made easy. The database should allow the administration of individual fish. The fish are tagged and given individual fishID to make identification possible. Important fish information is fishID, type, previous catch date and catch date. The database also keeps track of the fish values; age, height, weight and condition (1 = poor, 2 = OK, 3 = Good). And finally, the location where the fish moves (PostalCode, city, coordinates of catch).

Possible database users: Administrator, Researcher, officials working with nature related stuff. Administrator can alter all the tables, queries, views and create new data and structures. Researcher can alter the data, so they can delete, add and change the data. Officials can only read the data.

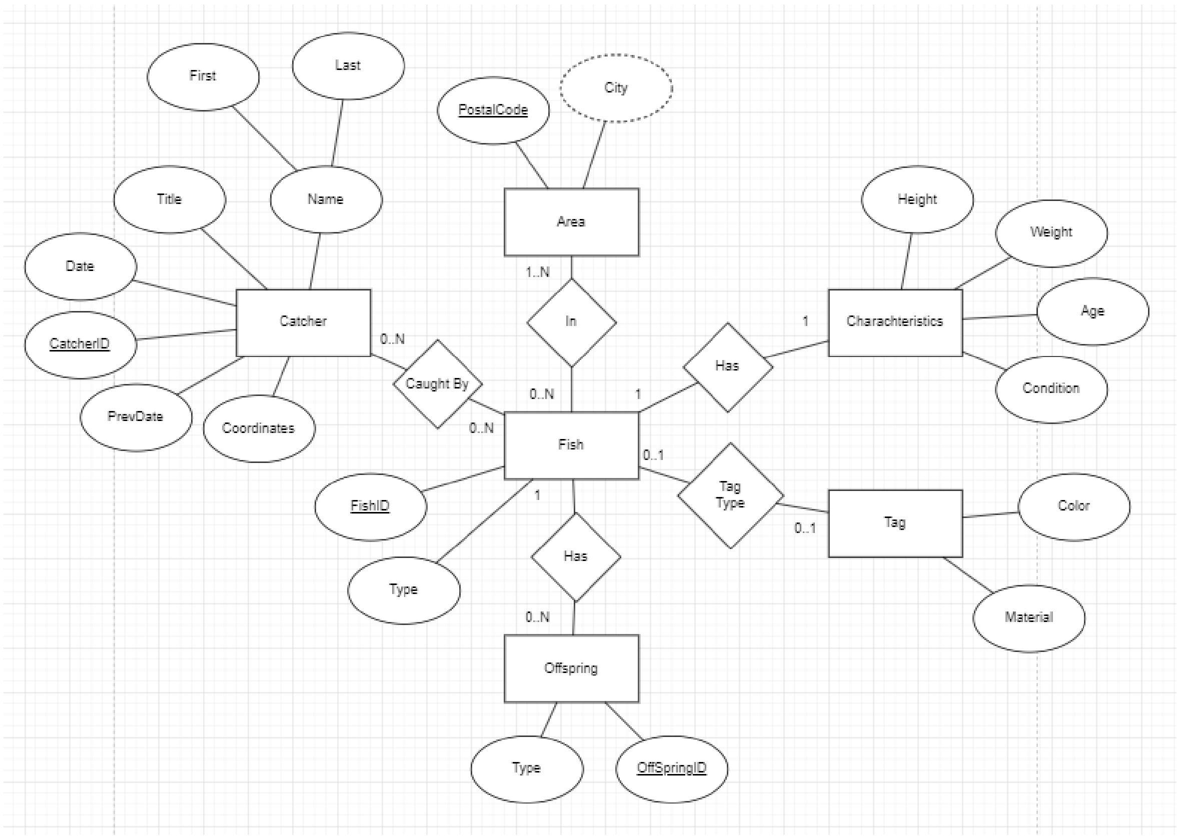
The following database queries have to be implemented: (1) List the number of fish based on their type. (Bokeh) (2) Update the information. (3) Insertion of new data. (4) Deletion of data. (5) List the fish on catch date, city and type.

2 MODELING

2.1 Concept model

From the definition we can identify 6 entities. The fish, individual fish which is the subject of the research and in the middle of the er-model. To catch the fish and get the catch date etc. there has to be a catcher. The relation is many-to-many, as the catcher can catch multiple or no fish and vice versa, the fish can be not caught or caught by multiple different catchers. The fish can have offspring, from multiple to none. All the fishes either have a tag or they don’t. The relation between area and fish is as well many-to-many as there can be multiple fish in one area, and one fish can be in multiple areas. Either way there has to always be area, because otherwise the fish would not have been caught. Finally, the fish has characteristics with one-to-one relation, which could have as well been attributes.

2.1.1 ER-Model



2.2 Relational model

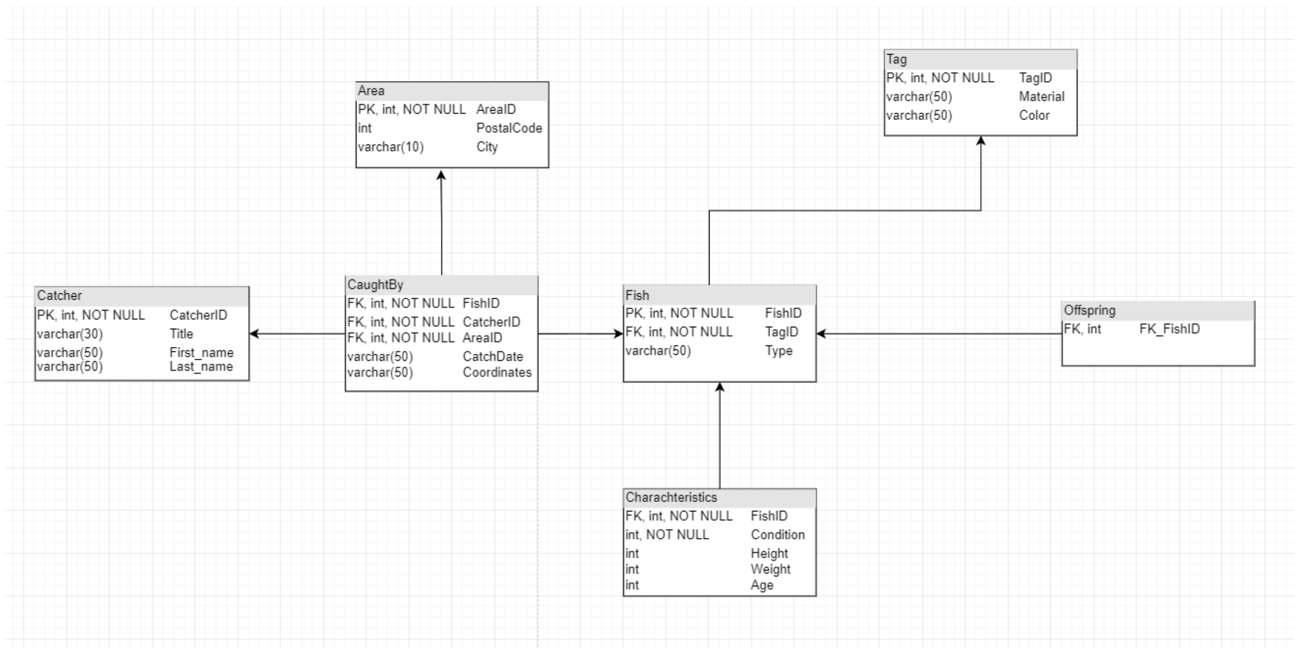
There has been made alterations to the relational model, so the result (freeform UML-style table) doesn’t look exactly like it should. In the one-to-to relationships, the foreign key has been placed on one of the sides of the relations.

In one-to-many relationships, the foreign key has been placed on the side of the ‘many’, however the only relationship of this kind is with ‘Offspring’, which as well is kind of a relationship to itself. In the practical portion, the FK is probably moved within the ‘Fish’ table. The FK ‘TagID’ based on the ER-model could technically be NULL, but it doesn’t really make sense, as the tag is the only thing that allows us to track the fish, A.K.A all the fish in the table ‘Fish’ are tagged.

In many-to-many relationships, a linking/interim relation is formed that stores foreign keys to both relations as well as attributes related to the relationship. There is two of these kind of relationships in the ER-model, with ‘Area’ and ‘Catcher’. Here have been made many alterations, but basically the interim table ‘CaughtBy’ has been made (many-to-many relationship). And again, technically the FKs in table ‘CaughtBy’ could be NULLs based on the ER-model, but again it wouldn’t really make sense, as for them to be in the database, these values can’t be NULL. The idea on the ER-model was that in real life not all fish is necessary caught.

Relationship ‘CaughtBy’ has been given attributes as it is more fitted, as ‘Area’ is as well moved to the same relationship. So, it is now described as an entity. All primary keys are unique.

2.2.1 Freeform UML style table structure



Area	
PK, int, NOT NULL	AreaID
int	PostalCode
varchar(10)	City

Tag	
PK, int, NOT NULL	TagID
varchar(50)	Material
varchar(50)	Color

Catcher	
PK, int, NOT NULL	CatcherID
varchar(30)	Title
varchar(50)	First_name
varchar(50)	Last_name

CaughtBy	
FK, int, NOT NULL	FishID
FK, int, NOT NULL	CatcherID
FK, int, NOT NULL	AreaID
varchar(50)	CatchDate
varchar(50)	Coordinates

Fish	
PK, int, NOT NULL	FishID
FK, int, NOT NULL	TagID
varchar(50)	Type

Offspring	
FK, int	FK_FishID

Charachteristics	
FK, int, NOT NULL	FishID
int, NOT NULL	Condition
int	Height
int	Weight
int	Age

3 DATABASE IMPLEMENTATION

In general:

Some of the transformations during the transformation of the concept model to relational model have already been discussed, i.e., how NULL values have been changed. Also as briefly mentioned, the relationship ‘CaughtBy’ has been made into entity and entity ‘Area’ was moved to the relationship as it’s basically the same relationship / strongly linked to it. Attributes ‘CatchDate’ and ‘Coordinates’ have been moved to the new entity ‘CaughtBy’ as it is more appropriate location for them and previously, they were misplaced.

‘Offspring’ has been deleted and the FK within it has been moved to ‘Fish’, as it is a relationship / instance of itself. ‘Fish’ attribute ‘Type’ has been changed to ‘Kind’ as ‘Type’ is pre-set keyword in SQL. ‘Characteristics’ attribute ‘Weight’ has been done the same to ‘Mass’. PostalCode was changed to VARCHAR (10).

The queries were quite loosely defined in the beginning, so no changes have been made. The final queries are as follows.

(1) Update catcher title
(2) Delete dead fish
(3) Insert new area
(4) Print all caught fish, with (ID, Type, Catch date, City)
(5) Show how many of each fish type there is (Bokeh -visualization)

Implementation:

Simple Python program was written, that showed the above-mentioned queries. Reading is demonstrated before and after the queries, by showing the changes.

Testing:

Nothing to note.

4 DISCUSSION

The need for reasonable indices was completely missed and ignored. However, an index has been implemented for table ‘CaughtBy’ values ‘FishID’, ‘CatcherID’ and ‘AreaID’. As ID is mostly used for identification, there is no need for other indences. However, we could make catcher identifiable by last name and a index ‘CREATE INDEX LastNameIndex ON Catcher (Last_name);’, but this is not done in the project.

In reflection, some changes were made between the relational model and the implementation, which is not ideal.