Relational schema (Task management application)

This document presents the relational schema of a task management application.

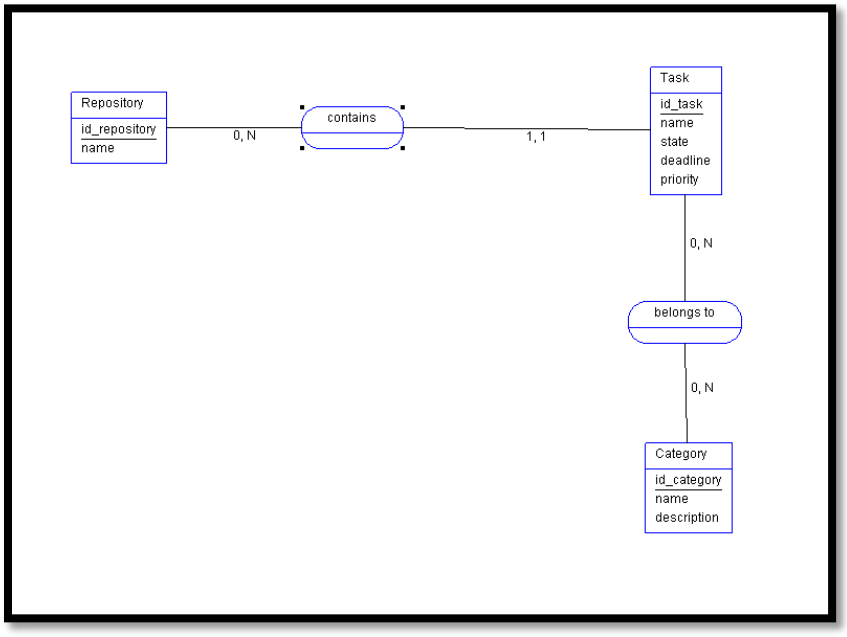
Context:

The application allows users to manage their personal tasks by organizing them into their personal repository and categories.

So, I need to design a task management database. I have the following information:

- There is a single repository but to ensure compatibility with future updates, a repository has a code and a name
- Each task has a code, a name, a state, a deadline and a priority
- Each category has a code, a name and a description
- A repository can have multiple tasks
- A task can be contained in only one repository
- A category can have multiple tasks, and a task can belong to multiple categories

Entity-relationship model:



Translation of the entity-association model into a relational schema:

Rule 1: each entity becomes a relation

Repository (id_repository, name)

Task (id_task, name, state, deadline, priority)

Category (id_category, name, description)

Rule 2:

Task (id_task, name, state, deadline, priority, #id_repository)

Rule 3:

belongsTo (#id_task, #id_category)

Final relational schema :

Repository (id_repository, name)

Task (id_task, name, state, deadline, priority, #id_repository)

Category (id_category, name, description)

belongsTo (#id_task, #id_category)

Functional dependencies :

By the decomposition algorithm, we found these results :

1) Universal relation

   Universal(id_repository, name_repository, id_task, name_task, state, deadline, priority, id_category, name_category, description)

2) All functional dependencies

Id_repository -> name_repository	1
Id_task -> name_task, state, deadline, priority, id_repository	2
Id_category -> name_category, description	3

Demonstrate that id_task, id_category -> id_repository, , name_repository, id_task, name_task, state, deadline, priority, name_category, description :

2 + augmentation : id_task, id_category -> name_task, state, deadline, priority, id_repository, id_category 4

4 + decomposition : id_task, id_category -> name_task, state, deadline, priority, id_repository
(id_task, id_category -> id_category) 5

3 + augmentation: id_category, id_task -> name_category, description, id_task 6

6 + decomposition : id_category, id_task -> name_category, description
(id_category, id_task -> id_task) 7

5 + 7 + union : id_task, id_category -> name_task, state, deadline, priority, id_repository, name_category, description 8

1 + decomposition : id_task -> id_repository
(id_task -> name_task, state, deadline, priority) 9

9 + transitivity : id_task -> name_repository 10

10 + augmentation : id_task, id_category -> name_repository, id_category 11

11 + decomposition : id_task, id_category -> name_repository
(id_task, id_category -> id_category) 12

8 + 12 + union : id_task, id_category -> name_task, state, deadline, priority, id_repository, name_category, description, name_repository

13

Which was to be proven.

3) Now, we deduct the final relational schema :
   Repository (id_repository, name)
   Task (id_task, name, state, deadline, priority, #id_repository)
   Category (id_category, name, description)
   belongsTo (#id_task, #id_category)

4) The relations are already in 3rd normal form because there are no transitive dependencies.

Conclusion

According to the entity-relationship model and the decomposition algorithm, we can deduce that the correct relational schema is as follows because both results match :

Repository (id_repository, name)
Task (id_task, name, state, deadline, priority, #id_repository)
Category (id_category, name, description)
belongsTo (#id_task, #id_category)