Relational schema (Task management application)

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

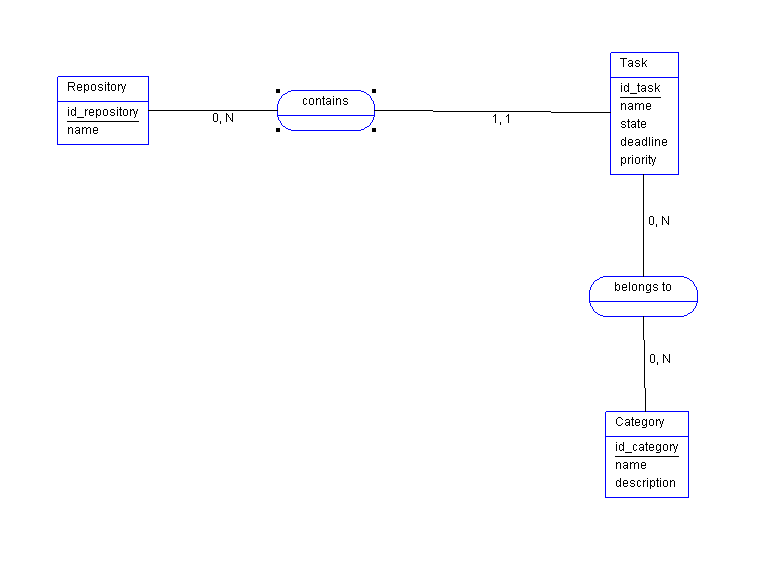
# Context:

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)

1. 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.**

1. 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)

1. 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)