ERD说明

用户和团队之间是一个一对多和一个多对多的关系，一个用户可以参与多个团队，一个团队可以有多个参与者；一个用户可以管理多个团队，一个团队只有一个管理者。

用户和项目之间是一个一对多和一个多对多的关系，一个用户可以参与多个项目，一个项目可以有多个参与者；一个用户可以管理多个项目，一个项目只有一个管理者。

团队和项目之间是一对多的关系，一个团队可以拥有多个项目，但一个项目只能属于一个团队。

用户和子任务之间是一个一对多和一个多对多的关系，一个用户可以参与多个子任务，一个子任务可以有多个参与者；一个用户可以管理多个子任务，一个子任务只有一个管理者。

项目和子任务之间是一对多的关系，一个项目可以包含多个子任务，但一个子任务只能属于一个项目。

用户和文件之间是一对多的关系，一个用户可以拥有多个文件，但一个文件只能属于一个用户。

团队和文件之间是一对多的关系，一个团队可以拥有多个文件，但一个文件只能属于一个团队。

项目和文件之间是一对多的关系，一个项目可以拥有多个文件，但一个文件只能属于一个项目。

关系模式，主键加粗，外键斜体

用户(**用户id**，用户名，密码，权限)

团队(**团队id**，团队名，团队描述)

项目(**项目id**，项目名称，项目描述，项目截止日期，项目状态)

子任务(**子任务id**，子任务名称，子任务描述，子任务截止日期，子任务状态)

文件(**文件id**，文件名字，文件地址，文件权限，文件状态)

用户-参与-团队(*用户id*,*团队id*)

用户-管理-团队(*用户id*,*团队id*)

团队-拥有-项目(*团队id，项目id*)

用户-管理-项目(*用户id*,*项目id*)

用户-参与-项目(*用户id*,*项目id*)

用户-管理-子任务(*用户id*,*子任务id*)

用户-参与-子任务(*用户id*,*子任务id*)

用户-拥有-文件(*用户id*,*文件id*)

团队-拥有-文件(*团队id**文件id*)

项目-拥有-文件(*项目id*,*文件id*)

关系模式的规范化处理

首先，我们可以把用户，团队，项目，子任务和文件这五个实体分别作为一个关系，它们的主键分别是用户id，团队id，项目id，子任务id和文件id。这样就可以消除这些实体之间的部分依赖和传递依赖，达到第二范式。

然后，我们把用户-参与-团队，用户-管理-团队，团队-拥有-项目，用户-管理-项目，用户-参与-项目，用户-管理-子任务，用户-参与-子任务，用户-拥有-文件，团队-拥有-文件和项目-拥有-文件这十个关系作为一个关系集合，它们的主键都是由两个属性组成的复合主键。这样就可以消除这些关系之间的非主属性对主键的部分函数依赖和传递函数依赖，达到第三范式。

因此，该数据库属于第三范式。

数据库完整性约束

实体完整性：用户id，团队id，项目id，子任务id和文件id都是主键，必须是唯一的，不为空的，不重复的。

参照完整性：用户-管理-团队，用户-参与-团队，团队-拥有-项目，用户-管理-项目，用户-参与-项目，用户-管理-子任务，用户-参与-子任务，用户-拥有-文件，团队-拥有-文件和项目-拥有-文件中的所有属性都是外键，要求非空且与其参照的主键相匹配。

域完整性：已在上文的数据字典中给出。

用户定义完整性：

项目截止日期和子任务截止日期必须大于当前日期。

一个用户不能同时参与和管理同一个团队，项目或子任务。

一个项目不能包含两个相同名称的子任务。

一个用户不能拥有两个相同名字的文件。

项目的管理者和参与者必须是参与的团队的管理者或者成员。

子项目的管理者和参与者必须是项目的参与人员。