db

перенести в описание всей системы. А этот файл закрыть.

Contents

[db 1](#_Toc90316889)

[to do 1](#_Toc90316890)

[Иметь ввиду 1](#_Toc90316891)

[Описание 1](#_Toc90316892)

# db

## to do

* Сделать библиотеку работы с базой данных
  + Сделать
  + Оттестировать
  + Сохранить
* Вставить поле дата создания в таблицу задачи
* Продумать как сочетается существование задач в контексте проектов
  + Может сделать таблицы проектов вместо указания к какой категории относится задача.
    - Следует учесть, что я могу закрывать направление или проект. Другие направления могут быть открытыми всегда. Получается это всё-таки отдельная сущность, с которой происходят какие-то манипуляции.
      * Можно учесть то, что я написал в скелете программы прошлой разработки.
    - Сравнить эти два подхода и подумать о других подходах.
  + Учесть, что у меня система состоит из дел и проектов (и систем). Правда там всё-таки направления, такие как семья, например.
  + Учесть подзадачи.
    - Думаю меньше, чем подзадачи не делать. Это избыточно. Усложняет систему сразу со всех сторон. На практике, как мне кажется достаточно только подзадач.
  + Учесть, что подзадачи могут появляться в процессе работы над задачей. И это может быть интересно посмотреть в аналитике.
* Определить экспериментально размер строкового поля для
  + Имени
  + Описания
  + Результата
* Пересмотреть возможность null для обоих таблиц
* Обновить описание обоих таблиц
* Если будут подпункты у задачи, и они будут выполнены в разные дни – отразить это как-то в таблице назначений
* Вынести все таблицы назначений в одну таблицу
* Добавить полям выполнено и отменено даты
* Добавить поле индекс в другой системе. Если задача была заимствована из другой системы следует иметь представление, где она там находится.

# Концепции

## Об универсальности всех типов проекта

Так же, как и задачи подзадачи следует свести в отдельную таблицу. Либо для каждой задачи делать отдельную таблицу подзадач. Но множество таблиц в базе данных засоряет её. Не хочется этого. Либо писать всё в самой задаче. Но это плохо с точки зрения теории баз данных.

Кроме того, на данный момент придерживаюсь принципа трёх уровней:

* Проекты и направления
* Задачи
* Подзадачи
* (ещё существует корневая таблица, содержащая перечень проектов)

Больше уровней усложнит всю систему на всех направлениях, в том числе на принципиальных, например ухудшит моё понимание ситуации с делами. Поэтому три уровня.

Однако почему бы не заложить возможность быстрой и качественной изменении концепции. Это не только не составит большого труда, но и будет соответствовать всем принципам построения систем. Речь вот о чём. Сделать и проекты, и задачи, и подзадачи одним типом сущностей. Это упрощает понимание всей системы. Одно входит в другое. Всё является одним и тем же типом.

Тип, который может содержать объекты такого же типа.

Этот подход

* Сделает упростит архитектуру проекта, сохранив и улучшив его функционал
* Сделает структуру проекта расширяемой. На случай необходимости в будущем пересмотра количества уровней

Разделение уровней происходит на уровне пользовательского интерфейса.

Тогда сделаем следующую структуру базы данных.

* Таблица индексов
  + Первая колонка ключ, вторая – индекс текущей задачи, третья – индекс задачи, входящей в текущую
* Таблица с задачами
  + Содержит первое ключевое поле – индекс задачи, по которому, в том числе, определяется какая задача в какую входит

## О в выделении направлений в отдельную единицу вместо поля в задаче

Попробуем оценить на сколько важна самостоятельность направлений.

Вообще-то видно, что важна. С одной стороны, это сферы деятельности. С другой стороны, я не принимаю существование сфер деятельности, но существуют потребности людей.

Плюсом к самостоятельности является т, что часть направлений является проектами.

Они могут обладать своими качествами. Начало, завершение. Название. О как много в этом звуке.

Думаю да – это важно, чтобы они были самостоятельными единицами.

Тогда переходим к структуре с проектами.

## О необходимости таблицы или таблиц для назначений дела

Каждое дело может быть назначено в разные дни. В какие-то дни оно ни на сколько не будет выполнено – до него просто не дойдут. В какие-то дни оно может быть выполнено частично. В какой-то день оно будет завершено или отменено. Назначения на разные дни имеет смысл сохранять информацию в какие дни что было сделано с делом – оно было назначено, выполнено частично или полностью или отменено. Эта информация может быть удобна для просмотра прогресса – как я выполнял дело и для аналитики – сколько дел я назначаю и сколько выполняю.

Поскольку информация о назначении дел довольно обширна её следует сохранять. Сохранять можно тремя основными способами.

* Всё помещать в ту же строку в общей таблице. Такой способ неудобен и противоречит основным принципам построения базы данных.
* Создать отдельную таблицу для каждого дела, если оно хоть раз назначалось и сохранять в неё всю информацию. Хороший способ. Соответствует принципам построения баз данных. Однако обладает одним недостатком – захламляется база данных. Появляется куча новых таблиц.
* Сделать одну таблицу для всех дел, в которую помещать информацию об их назначении. Каждая строка – одно назначение. При необходимости делать выборку из это таблицы. Этот подход наиболее удобен. База данных не захламляется. Подход соответствует принципам построения баз данных.
  + Предполагаю в моём случае не будет сильной просадки в производительности в этом случае. А может будет и выигрыш. Пока не знаю.
  + Таблица не будет изменяться пользователем. В другие таблицы могут вноситься правки. Например, если пользователь решил отредактировать описание. В эту таблицу нет. Поэтому при парсинге не возникнет проблемы обратной связи.

# Описание

Все задачи сведены в одну таблицу tasks.

Такие имена позволяют разместить таблицу задач в начале списка таблиц в базе данных, а таблицы назначений в конце. Получается наглядно. После t остаются буквы uvwxyz, на которые навряд ли появятся таблицы начинающиеся, а эти буквы. Предполагается самое большое количество таблиц будет у таблиц назначений. Но их можно все свести в одну таблицу и делать по ней выборку.

Для раздела, к которому относится задача предусмотрена колонка section. Алгоритм выбран не распределение задач по таблицам разделов и выборка всех таблиц разделов. А выбор одной таблицы и последующая сортировка по секциям. Так выглядит проще. Если будет сказывать на производительности переделаю. Но, судя по всему, не будет. Хотя посмотрим, как всё реализуется в процессе.

Для типов процентов выбран tinyint. Всё-равно больше 100 нельзя установить.

Для указания в какие дни назначалась задача и сколько в эти дни она делалась в процентах и абсолютном значении создаётся отдельная таблица для каждой задачи, если она назначалась хотя бы в один день. Если никогда не назначалась такой таблицы не создаётся. Сответствующее поле пустое. Примерный вид таблицы назначений приведён внизу. Формат названия таблицы tasks\_i\_appointment, где i = Id в таблице tasks.

Таблица 1. tasks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Id | int | False |  |
|  | section | nvarchar(30) | True |  |
|  | name | nvarchar(30) | True |  |
|  | description | nvarchar(300) | True |  |
|  | importance | tinyint | True |  |
|  | urgency | tinyint | True |  |
|  | scope | tinyint | True |  |
|  | done | bit | True |  |
|  | percent | tinyint | True |  |
|  | canceled | bit | True |  |
|  | result | nvarchar(300) | True |  |
|  | periodicity | bit | True |  |
|  | periodicity\_value | nvarchar(300) | True |  |
|  | source | nvarchar(300) | True |  |
|  | table\_appointment\_date | nvarchar(30) | True |  |

Таблица 2. tasks\_i\_appointment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Id | int | False |  |
|  | data | data | False |  |
|  | percent | tinyint | False |  |
|  | absolute | nvarchar(300) | True |  |