Оглавление

[Основные возможности 2](#_Toc3468416)

[Терминология 3](#_Toc3468417)

[Основные принципы построения системы 6](#_Toc3468418)

[Структура данных и связей 6](#_Toc3468419)

[История изменений 8](#_Toc3468420)

[Дополнительные связи 8](#_Toc3468421)

[Документы и версии документов 10](#_Toc3468422)

[Рабочие папки 11](#_Toc3468423)

[БАЗА ДАННЫХ 12](#_Toc3468424)

[Объекты 12](#_Toc3468425)

[Object\_classificator 12](#_Toc3468426)

[Object 12](#_Toc3468427)

[Object\_history 13](#_Toc3468428)

[Структура изделий (конструкторская документация) 13](#_Toc3468429)

[Structure 13](#_Toc3468430)

[Structure\_history 14](#_Toc3468431)

[Structure\_cross 14](#_Toc3468432)

[vObject 14](#_Toc3468433)

[Структура рабочего стола 15](#_Toc3468434)

[Версии документов 15](#_Toc3468435)

[Document\_object 16](#_Toc3468436)

[Document\_object\_history 16](#_Toc3468437)

[Document\_extra 16](#_Toc3468438)

[Document\_type 17](#_Toc3468439)

[Document\_inwork 17](#_Toc3468440)

[Document\_version 17](#_Toc3468441)

[Document\_complex 17](#_Toc3468442)

[vDocument 18](#_Toc3468443)

# Основные возможности

**Представление логики**

Навигация по системе производится с помощью двух представлений:

- **настраиваемое навигационное дерево**, содержащее в себе общие, типизированные разделы и отдельные изделия, позволяющее посмотреть наборы данных уточненных по различным параметрам. Дерево состоит из неизменяемой основы, видимой всем пользователям и дополнительных пользовательских элементов.

Например, можно добавить в дерево объект-комплекс, чтобы получить список всех входящих в него изделий. Для уточнения, можно вложить в этот объект подраздел и настроить его, чтобы он отображал только сборочные единицы. В данном режиме, объекты и настройки разделов позволяют организовать вложенную фильтрацию по различным признакам.

Дополнительные разделы и объекты навигационного дерева видны только создавшему их пользователю или другим, если владельцем настроены права доступа.

При фактическом удалении объекта, привязанного пользователем к дереву навигации он не удаляется, а остается доступным с пометкой об удалении. Так же, данное дерево не отслеживает добавления/изменения фактической структуры. Это не влияет на созданные разделы, но выдаваемые разделами данные будут меняться.

- **дерево изделий**, отображающее фактические связи изделий в комплексах, комплектах и сборочных единицах; их иерархию. Дерево строится автоматически, основываясь на объектах связей. Пользователь не может добавлять сюда персональные элементы. Все внесенные изменения влияют на актуальную рабочую структуру данных (удаленные элементы будут пропадать, добавленные - появляться). Так же, данное дерево можно просмотреть в состоянии на указанный момент времени и сравнить с актуальным состоянием.

**История изменений**

**Проекты**

Поскольку основная структура объектов программы представляет собой физическое строение механизмов в виде дерева с поддержкой истории изменений, возникают вопросы по экономии ресурсов сервера по хранении истории и безопасном внесении изменений.

Для решения данных вопросов введен механизм проектов, который, по сути, является копией части реального дерева, вынесенную в отдельную таблицу для свободной модификации. История внесенных изменений в проект не сохраняется и сам проект хранится после завершения в финальном его состоянии. Внесенные изменения можно отследить по хронологии событий проекта (отдельная таблица с сообщениями)

При завершении проекта, его данные накладываются на текущее актуальное дерево и создают его новое состояние с точкой изменения (используемой для пошаговой навигации по истории изменений).

Проект может содержать в себе несколько объектов со всем их содержимым, работать над ним может несколько человек с разными уровнями доступа.

*Подробнее смотреть в одноименном разделе ниже.*

# Терминология



**Детали** – отдельные части, изготавливаемые самим предприятием, либо заказанные по нашим чертежам у других предприятий (кооперация)

*Деталь - изделие, изготовленное из однородного по наименованию и марке материала, без применения сборочных операций, например, валик из одного куска металла; литой корпус; пластина из биметаллического листа; печатная плата; маховичок из пластмассы (без арматуры); отрезок кабеля или провода заданной длины. Эти же изделия, подвергнутые покрытиям (защитным или декоративным), независимо от вида, толщины и назначения покрытия, или изготовленные с применением местной сварки, пайки, склейки, сшивки и т.п., например, винт, подвергнутый хромированию; трубка, спаянная или сваренная из одного куска листового материала, коробка, склеенная из одного куска картона.*

**Стандартные изделия** – покупные изделия различной технологической сложности изготовленные на других предприятиях по сторонним чертежам и имеющие определение ГОСТ.

*изделия, примененные по категориям стандартов по ГОСТ*

**Прочие изделия** – аналогично стандартным, но без данных ГОСТ.

*изделия, примененные по техническим условиям, и импортные покупные изделия, примененные по сопроводительной технической документации зарубежных изготовителей (поставщиков)*

**Материалы** – исходные расходные для изготовления деталей.

*также материалы, необходимое количество которых не может быть определено разработчиком КД по размерам элементов изделия и вследствие этого устанавливается разработчиком технологической документации. К таким материалам относят, например: лаки, краски, клей, смазки, замазки, припои*

**Сборочные единицы** – сборные части из деталей или других сборочных (любой сложности), при этом являются частью более сложных сборочных единиц. Аналог ВП – блоки

*Сборочная единица - изделие, составные части (СЧ) которого подлежат соединению между собой на предприятии-изготовителе сборочными операциями (свинчиванием, сочленением, клепкой, варкой, пайкой, запрессовкой, развальцовкой, склеиванием, сшиванием, укладкой и т.п.), например, автомобиль, станок, телефонный аппарат, микромодуль, редуктор, сварной корпус, маховичок из пластмассы с металлической арматурой.*

*К сборочным единицам, при необходимости, также относят*

*а) изделия, для которых конструкцией предусмотрена разборка их на составные части*

*б) совокупность сборочных единиц и/или деталей, имеющих общее функциональное назначение и совместно устанавливаемых на предприятии-изготовителе в другой сборочной единице, например, электрооборудование станка, автомобиля, самолета; набор составных частей для установки врезного замка (замок, запорная планка, ключи)*

*в) совокупность сборочных единиц и/или деталей, имеющих общее функциональное назначение, совместно уложенных на предприятии-изготовителе в укладочные средства (футляр, коробку и т.п.), которые предусмотрено использовать вместе с уложенными в них изделиями, например, набор чертежных инструментов (готовальня) набор концевых плоскопараллельных мер длины*

*г) упаковочную единицу, представляющую изделие, создаваемое в результате соединения упаковываемой продукции с упаковкой.*

**Комплекс** – сборочная единица высшего уровня (базовое изделие)

*Комплекс - два и более специфицированных изделия, не соединенных на предприятии-изготовителе сборочными операциями, но предназначенных для выполнения взаимосвязанных эксплуатационных функций.*

*Каждое из этих специфицированных изделий, входящих в комплекс, служит для выполнения одной или нескольких основных функций, установленных для всего комплекса, например, цех-автомат; завод-автомат, автоматическая телефонная станция, бурильная установка; изделие, состоящее из метеорологической ракеты, пусковой установки и средств управления; корабль.*

*В комплекс, кроме изделий, выполняющих основные функции, могут входить детали, сборочные единицы, комплекты, предназначенные для выполнения вспомогательных функций, например, детали и сборочные единицы, предназначенные для монтажа комплекса на месте его эксплуатации; комплект запасных частей укладочных средств, тары и др.*

**Комплекты** – все объекты, идущие дополнительно к изделию (упаковочный лист). Состоит из набора объектов любых типов.

*ведомость эксплуатационных документов, ведомость документов для ремонта (при необходимости) и применяемые по КД комплекты, которые непосредственно входят в специфицируемое изделие и поставляются вместе с ним, а также упаковку, предназначенную для изделия*

*Комплект - два и более изделия, не соединенных на предприятии-изготовителе сборочными операциями и представляющих набор изделий, имеющих общее эксплуатационное назначение вспомогательного характера. например: комплект запасных частей, комплект инструмента и принадлежностей, комплект измерительной аппаратуры, комплект упаковочной тары и т.п. К комплектам также относят сборочную единицу или деталь, поставляемую вместе с набором других сборочных единиц и/или деталей, предназначенных для выполнения вспомогательных функций при эксплуатации этой сборочной единицы или детали, например осциллограф в комплекте с укладочным ящиком, запасными частями, монтажным инструментом, сменными частями.*

**Документация**

*В раздел "Документация" следует вносить документы, составляющие основной комплект КД*

*специфицируемого изделия, кроме его электронная спецификация, ведомости эксплуатационных документов и ведомости документов*

*для ремонта, а также документы основного комплекта записываемых в ЭСП неспецифицируемых составных*

*частей (деталей), кроме их рабочих электронных чертежей или электронных моделей*

В программе, на подобие системы Search, каждый объект имеет базовый, вспомогательный тип и документ (или несколько документов, но один всегда основной).

# Основные принципы построения системы

## Структура данных и связей

База данных строится по принципу разделения объектов и структуры.

- Объекты являются большим справочником реальных объектов различных типов, которые никак между собой не связаны.

- Структура – набор связей между отдельными объектами в виде дерева.

Данный подход позволяет минимизировать набор данных, поскольку требуется всего один экземпляр объекта, чтобы любое количество раз использовать его в любом количестве структур.

Существует естественное ограничение, что объект не может являться собственным потомком (рекурсивное вхождение). Данное ограничение отслеживается методом добавления связи между объектами (хранимая процедура)

Структура представляет собой скорее граф, чем дерево, поскольку объект может ссылаться на любое количество других (с соблюдением правила отсутствия рекурсии).

Главные преимущества:

– автоматическая расширение структуры при привязке

- автоматическое обновление всех упоминаний объекта во всех структурах

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Объекты в базе** | |  |  |  |
| A  B  C  E  D  F   |  | | --- | |  | |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | **Таблица ссылок создает структуру:** | | | |  |
|  | 1. (D -> E) |  |  |  |  |
|  | 2. (C -> D) |  |  |  |  |
|  |  |  |  |  |  |
|  | **Получаем:** | |  |  |  |
|  | D  E  C | | | |  |
|  |  |
|  |  |
|  |  |
|  | Имеющиеся структуры автоматически выстраиваются при добавлении новых ссылок: | | | | |
|  |  |  |  |  |  |
|  | 3. (E -> A) |  |  |  |  |
|  | 4. (E -> B) |  |  |  |  |
|  |  |  |  |  |  |
|  | **Получаем:** | |  |  |  |
| A  D  E  C  B  D  E  C   |  | | --- | |  | |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | и при расширении вложенной структуры изменения коснутся всех использований: | | | | |
|  |  |  |  |  |  |
|  | 5. (F -> D) |  |  |  |  |
|  |  |  |  |  |  |
|  | **Получаем:** | |  |  |  |
|  |  |  |  |  |  |
| A  D  E  C  F   |  | | --- | |  | |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| B  D  E  C  F   |  | | --- | |  | |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | Изменение данных работает так же: | | | |  |
|  |  |  |  |  |  |
|  | A -> A2 |  |  |  |  |
|  | C -> C2 |  |  |  |  |
| C2  A2  D  E  F   |  | | --- | |  | |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| B  D  E  C2  F   |  | | --- | |  | |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## История изменений

Для отслеживания изменений и возможности восстановления состояния структуры объектов на любой момент времени, используется подход разделения актуального состояния структуры от набора изменений каждой записи.

Для этого, у каждой таблицы есть \_history версия, где хранятся неактуальные версии записей с дополнительными данными:

- дата удаления записи

- id пользователя, сделавшего изменение

- комментарий к изменению (обоснование, проставляемое автоматически или пользователем)

- id операции из таблицы [operation], для экономии места хранения архива

Такой подход позволяет:

- работать только с актуальными данными в текущей работе (не затрагивается \_history)

- получать состояние структуры и состояния объектов на любой момент времени, совмещая данные из актуальной и \_history таблиц выборкой по конкретной дате.

- быстро получать данные об истории изменений отдельного объекта/связи вверх или вниз по временной шкале.

Не ключевые таблицы не имеют таблицы истории, либо история чистится при достижении некоего конечного результата (завершения проекта).

## Дополнительные связи

Вспомогательный механизм, реализованный в таблицах «\_cross», для таблиц связей.

Хранит набор всех связок для всех родителей конкретной связки вплоть до корня дерева. Это позволяет по id выбранного в дереве объекта получить id всех вложенных в него объектов одним не рекурсивным запросом.

Таблица содержит три ключевых столбца:

- родитель (перебираются от связки вверх по дереву)

- потомок исходной связки (с которой начался поиск вверх по дереву)

- id исходной связки

Этих данных достаточно, чтобы простым запросом выполнить операции:

- Поиск всех потомков объекта X:

**SELECT DISTINCT(child) FROM table WHERE parent = X**

- Поиск всех предков объекта Х:

**SELECT DISTINCT(parent) FROM table WHERE child = X**

- Удаление допсвязей неактуальной связи c id = X:

**DELETE FROM table WHERE base\_link = X**

Например, имеем структуру:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **Таблица связей** | |  |
|  |  |  |  |  | **id** | **parent** | **child** |
| A  B  C  D  E  F  **1**  **2**  **3**  **4**  **5**  **6**  D  **7**   |  | | --- | |  | |  |  |  |  | 1 | A | B |
|  |  |  |  |  | 2 | B | C |
|  |  |  |  |  | 3 | B | D |
|  |  |  |  |  | 4 | C | E |
|  |  |  |  |  | 5 | D | E |
|  |  |  |  |  | 6 | D | F |
|  |  |  |  |  | 7 | C | D |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  | **Таблица допсвязей** | | |
|  |  |  |  |  | **parent** | **child** | **base\_link** |
|  |  |  |  |  | A | B | 1 |
|  |  |  |  |  | B | C | 2 |
|  |  |  |  |  | A | C | 2 |
|  |  |  |  |  | B | D | 3 |
|  |  |  |  |  | A | D | 3 |
|  |  |  |  |  | C | E | 4 |
|  |  |  |  |  | B | E | 4 |
|  |  |  |  |  | A | E | 4 |
|  |  |  |  |  | D | E | 5 |
|  |  |  |  |  | B | E | 5 |
|  |  |  |  |  | A | E | 5 |
|  |  |  |  |  | D | F | 6 |
|  |  |  |  |  | B | F | 6 |
|  |  |  |  |  | A | F | 6 |
|  |  |  |  |  | C | D | 7 |
|  |  |  |  |  | B | D | 7 |
|  |  |  |  |  | A | D | 7 |

Алгоритм генерации допсвязей:

- перебираем все связи

- для каждой:

- запоминаем id и child

- перебираем родителей рекурсивными запросами и формируем для каждого запись: текущий\_родитель, child, id

Генерацией допсвязей занимается хранимая процедура pdm\_CREATE\_CROSS\_LINKS, которая рекурсивно вызывает сама себя, создавая курсоры вида сХХХ для обработки каждой родительской связки, где XXX – id текущей обрабатываемой связки.

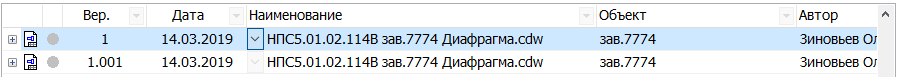
Работа через курсоры достаточно меделенная, но побочным полезным эффектом является автоматическая отсечка цикличных ссылок в дереве. В этом случае будет создан курсор с уже существующим именем и процедура упадет с соответствующей ошибкой. Это используется в программе, чтобы отменить операцию привязки с ошибкой: «Не удалось привязать объект» .

## Документы и версии документов

К каждому объекту можно привязать один или сколько угодно документов.

Документы в системе – так же объекты. Каждый файл может иметь несколько версий, которые являются отдельными объектами.

Версии документа могут быть полными (не находящиеся в редактировании) и рабочими (редактируются в данный момент).

Рабочей версией документа делается наличием связки в таблице [document\_inwork], при этом к номеру версии будет добавляться через точку номер подверсии. Например, “3.006”. При удалении связки рабочей версии к исходной в [document\_inwork], автоматически сделает ее полной со следующим номером. 

Рабочая версия документа позволяет сохранять предварительную версию в базу PDM, чтобы с ней могли ознакомиться другие пользователи. При сохранении рабочего документа, он обновляется в файловом хранилище (новые версии не создаются). Но каждая полная версия хранится в базе.

Заложен механизм комплексных документов. Это файлы, включающие в себя дополнительные отдельные файлы. Например, чертеж, содержащий картинку и другой чертеж в качестве вставки.

Для работы требуется, чтобы для новой версии комплексного документа все вложенные файлы подтягивались в ту же рабочую папку, что и основной файл. Главной сложностью является подмена путей до вложенных файлов в файле КОМПАС, поскольку, в зависимости от ситуации, пути до них могут быть различными. Файлы КОМПАС на поддерживают относительные пути, только полные. Реализация данного типа документов пока отложена, до выяснения возможности через API КОМПАС подменять пути до вложений в файле.

Сами файлы привязанных документов хранятся в файловом хранилище, в базе [FilesDB], таблице [PDMFiles]. Ключевым полем в таблице хранилища явлется Name. Для связи из базы PDM, в таблице [document\_extra][fullname] хранится текстовый идентификатор из имени файла и комплексного идентификатора.

Например, **НПС5.01.02.114В зав.7774 Диафрагма.cdw{1054418\_1054459\_1}**

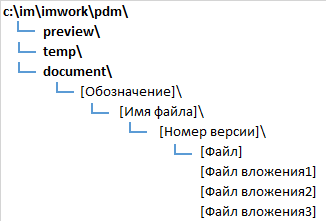
Где первое значение в фигурных скобках – id объекта-владельца документа, второй – id самого документа, третий – номер версии.

Как таковой, номер версии не обязателен, но добавлен для наглядности (поиска нужной версии)

Для защиты от одинаковых версий, для каждого файла хранится хэш и программа не позволяет сохранять файлы, которые не были изменены.

## Рабочие папки

Каждый пользователь на своей рабочей машине имеет набор рабочих папок PDM (создаются/обновляются автоматически при старте программы).



**Preview**

содержит миниатюры содержимого файлов, которые пользователь когда-либо просматривал. Миниатюры не хранятся в базе, а генерятся на лету при привязке документа (для показа в окне привязки) или при принудительном просмотре при перемещении по списку документов объекта.

Содержимое папки принудительно не очищается.

**Temp**

Хранилище различных временных файлов. В частности, документов, выгруженных из PDM открытых для просмотра.

При запуске или завершении программы – очищается.

**Document**

Папка работы над версиями документов. При взятии документа в работу, для него создается цепочка подпапок: *\[Обозначение]\[Имя файла]\[Версия]\*

Где,

*Обозначение* – уникальное обозначение родительского объекта (к которому привязан документ),

*Имя файла* – полное имя файла взятого в работу

*Версия* – номер новой версии. В данную папку выгружается основной документ и все привязанные к нему в качестве вложенных. При этом, основной файл редактируется – в нем через API КОМПАС переписываются пути до всех вложений до данной папки.

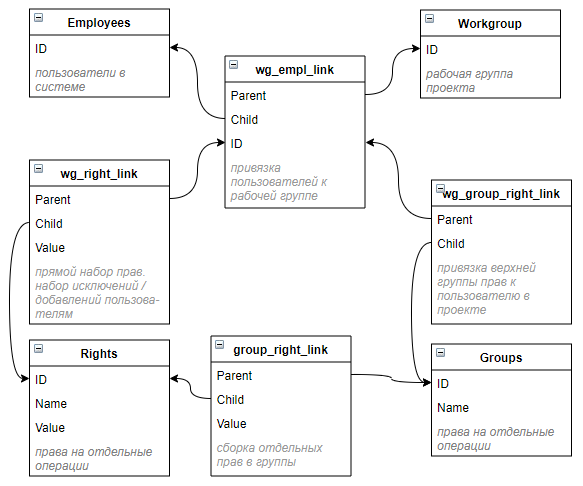
## Система прав

Позволяет установить разрешения и исключения на любое действие пользователя в системе. На данный момент, подразумевается набор прав для работы в рамках проекта.

Все строится на отдельном справочнике прав [Rights], где, в общем случае, указывается наименование операции и значение поумолчанию (это может быть значение разрешения или какого-то исходного статуса). Желательно, чтобы имя правила отображало сферу применения, поскольку допусается привязка отдельный прав к пользователю в рамках рабочей группы в обход механизма групп прав, имеющих общее пояснение. Для облегчения сортировки и выбоки прав, в таблице [Rights] присутствует поле [tags] в котором может содержатиься любое количество тематических меток (по сути, замещающих/дополняющих группы)

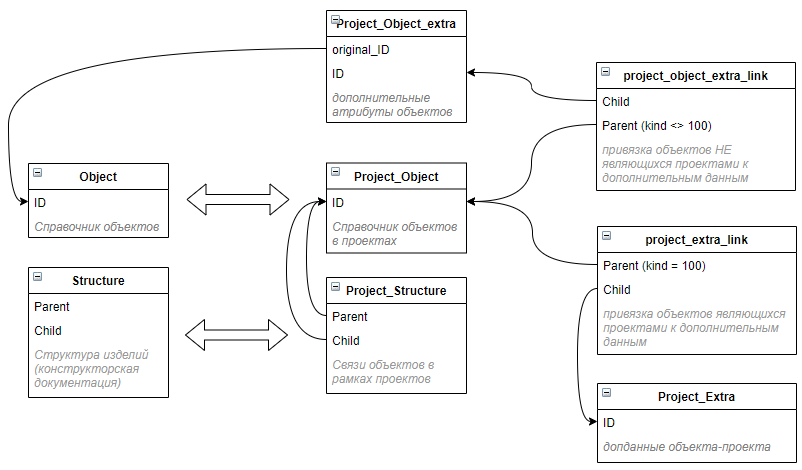
Отдельные права (любое количество) привязаны к группам [Groups], которые позволяют собирать различные конфигурации, причем, таблица связей с группами [group\_right\_link] имеет поле [value], перекрывающее значение [Rights][value], если не null. Что позволяет более тонко настроить каждую роль, если ее значение может варьироваться.

Наличие поля [todata] типа даты, позволяет назначать временные права, котрые будут удалены при достижении даты, указанной в поле, что позволяет реализовать механизм временного замещения сотрудника или временного расширения прав.



## Проект

Отдельный режим работы, один из ключевых, в котором группа пользователей работают над копией части дерева структуры, после чего изменения вливаются в оригинальную структуру.



Общая логика работы:

- для реализации проектов существуют рабочие копии основных таблиц, куда, при создании проекта копируются существующие объекты

- сами проекты, так же являются объектом и находятся только в рабочей таблице объектов, являясь корнем структуры проекта

- при создании проекта-доработки, реально существующие объекты копируются в рабочие таблицы (с новыми id и сохранением оригинальных в дополнительной таблице данных объекта в проекте). Так же в рабочую таблицу переносятся исходные связи объектов, при этом они строятся по рабочим id объектов (не совпадающих с оригинальными из [object]).

- при редактировании объектов их предыдущие состояния общим механизмом помещаются в историю рабочей таблицы проектов [project\_object\_history], что позволят видеть изменения объектов в рамках проекта и откатиться до нужного состояния

- при добавлении объектов, они добавляются в рабочую таблицу и участвуют в связях в рамках проекта под своим рабочим id, при этом оригинальный id в дополнительной таблице [project\_object\_extra] будет пуст, что и является признаком нового объекта, к тому же, позволяющим его свободно править в рамках проекта.

- при добавлении связей, они создаются по рабочим id объектов. Так же работает и механизм допсвязей.

- при удалении связи, она помещается в архивную таблицу рабочей таблицы связей [project\_structure\_history]

-при отмене проекта доступ к его редактированию закрывается, полностью сохраняется его текущее состояние, а в исходной структуре никаких изменений не происходит. По сути это проект, работа с которым просто остановлена и при снятии признака архивного, работу над ним можно возобновить. При возобновлении проекта все заблокированные позиции обновляются.

- при успешном завершении проекта, он так же становится архивным, но при этом происходит объединение исходной структуры со всеми произведенными в рамках проекта изменениями. При этом, сканируются все остальные проекты и проверяется включение данной спецификации. Если есть, ее данные в том проекте обновляются. Это позволяет содержать все проекты в актуальном состоянии без необходимости вручную запрашивать апдейт проекта, если что-то внешнее изменилось.

- к дополнительным возможностям проекта относятся:

- система прав на весь спектр действий в проекте, определяемых на каждого участника

- система событий при изменении объектов или структуры, рассылаемых всем участникам рабочей группы и всем, кто подписался на проект (добавил его в подписку в дереве рабочего стола, в разделе проектов)

- система сообщений, позволяющая централизованно общаться рабочей группой

- система заданий, которые можно назначать друг-другу в рамках рабочей группы и отслеживать выполнение, что влияет на статистику завершения проекта

Строго отслеживается работа с уже существующими объектами. При добавлении в структуру объекта они получают статус не редактируемых, т.е. нельзя менять их характеристики и состав (набор вложенных элементов). Исключение в данном правиле делается только для корневой спецификации в проекте-доработке и действует только на список непосредственных потомков.

При загрузке уже существующих спецификаций в проект их содержимое не подгружается полностью, а только непосредственные потомки (наполнение 1-го уровня).

**Вопрос 1**: *если правка существующих заблокирована, но допустима «по извещению», следует ли открывать редактирование в текущем проекте или только в новом, где эта позиция будет корневой?*

*Ответ: редактируется вне проекта (в рамках другого, именно для этой позиции), после чего обновить данные в текущем.*

Любые добавленные в рамках проекта объекты могут редактироваться как угодно и строить любые структуры между собой.

Для редактирования новой позиции в рамках проекта (его и его состав 1-го уровня) назначается исполнитель, при этом позиция получает статус «в работе». Без этого она находится в статусе «просмотр». Назначение исполнителя на заблокированную позицию (скопированную из согласованной структуры) не дает возможности его редактирования. Исполнитель на позицию может быть назначен только один.

Исполнитель в рамках своей работы наполняет позицию (1-й уровень). Для более глубокого редактирования он может назначить на вложенный элемент исполнителем себя или другого участника группы.

По завершении наполнения, исполнитель выставляет позиции статус «на проверку» и назначенный проверяющим получает извещение о необходимости провести ее. При этом, если есть вложенные разрабатываемые другими пользователями позиции, они получают извещение, что работа над верхним уровнем завершена и требуется подтвердить готовность, иначе контролер не сможет принять этот объект. При этом, вместо «на проверку», позиция получает статус «в ожидании», который будет автоматически возвращен в «на проверку», как только все вложенные позиции так же перейдут в статус «на проверку»

При корректном наполнении проверяемой позиции, проверяющий выставляет статус «завершено». Нет необходимости принимать каждую из вложенных позиций. Можно принять самую верхнюю и все вложенные автоматически станут принятыми и завершенными. При этом все задействованные исполнители получат извещение о принятии позиции. Все задачи, связанные с данной позицией, так же переходят в статус завершения.

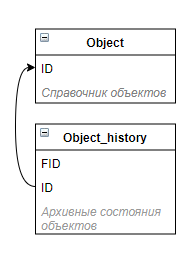
При неудовлетворительном результате проверки контролер возвращает позиции статус «в работе» и указывает необходимые для доработки моменты в виде нового задания, привязанного к редактору позиции.

На завершенную позицию больше на нее нельзя назначать исполнителей, но, при необходимости, возможен сброс состояния контролером в «просмотр» (когда требуется незапланированная доработка) и назначение исполнителя.

В рамках проекта, где исполнитель всего один, при попытке редактирования позиции, исполнитель назначается автоматически (не требуется ручного назначения), что облегчит работу с мелкими индивидуальными проектами с небольшими доработками.

# БАЗА ДАННЫХ

## Объекты



### Object\_classificator

Описание всех используемых в программе типов (kind) объектов, используемых как для построения навигации, так и фактического строения изделий, сборочных единиц, комплектов.

**Name** – общее наименование типа

**Icon** – имя общей иконки типа при отображении в дереве

**Descript** – общее описание типа

**Kind** – титульный тип объекта. Используется повсеместно.

**Extratable** – имя таблицы, содержащей дополнительные данные для данного типа объектов (справочное поле, облегчающее понимание структуры базы)

### Object

Все существующие и актуальные на данный момент объекты. По сути – большой справочник, не отражающий ни связей между объектами, ни существующего количества экземпляров.

**Kind** (int) – один из типов [Object\_classificator].

**Icon** (int) – имя персональной иконки

**Created** (datetime) – дата и время создания объекта

**Uid** (int) – id создавшего пользователя.

**Name** (varchar(1000)) – наименование

**Mass** (float) – масса в килограммах

**Mark** (varchar(100)) – конструкторское обозначение вида, схожего с: «2019.01.02.0000». Является (должно быть) уникальным в рамках всей таблицы. Позволяет определить структуру вложения объектов друг в друга (на практике используется, как дополнительное наименование, а связи содержатся в дополнительных таблицах)

**Submark** (varchar(500)) – дополнительное обозначение

**Comment** (varchar(max)) – дополнительное описание

**Material\_id** (int) – прямая ссылка на объект типа Материал в этой-же таблице, если данный объект является деталью

**Realization** (varchar(50)) – префикс типа '-01', '-02' и т.п. если данный объект является модификацией. по сути, он является отдельной копией оригинального (с таким же обозначением, без префикса) с небольшими изменениями.

**Has\_docs** (int) – признак (1/0) имеет ли объект хоть один привязанный документ. Используется для ускорения построения списка, чтобы не опрашивать каждый раз таблицу связей с документами.

Следует заметить, что дата создания содержит не момент фактического создания объекта в базе, а момент последнего изменения. То же самое и про UID – это идентификатор последнего изменившего пользователя.

Объекты никогда фактически не удаляются из базы. Признак удаления позволяет отсеять объекты при выборках и сохранить привязку истории изменений к удаленному объекту.

Так же, является таблицей источником для показа справочника в интерфейсе.

### Object\_history

Все неактуальные состояния объектов на любой момент времени с начала запуска программы. Включает все поля таблицы Object и некоторые дополнительные:

**deleted** (datetime) – момент создания именно этой архивной версии записи.

**huid (int)** – [history user id] пользователь, действием которого объект помещен в архив

**hcomment** (varchar(max)) – подробное описание действия, создавшего запись в архиве (если не обходимо)

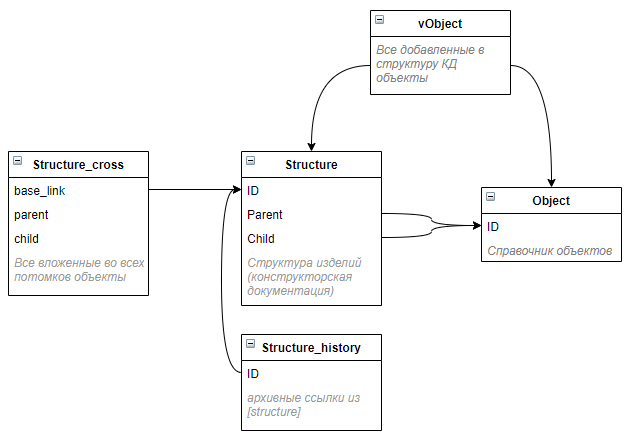
**operation\_id** (int) – ссылка на таблицу [operation], содержащая индекс стандартной операции, чтобы не забивать архив излишним текстом.

**fid** (int) – [false id] уникальный для архива id. Оригинальный id объекта сохраняется в одноименном поле, что позволяет быстро получить все его состояния

**…** - далее следуют все поля таблицы [object]

## Структура изделий (конструкторская документация)

Обеспечивает показ древовидной структуры изделий в интерфейсе и работу с ней.



### Structure

Все актуальные связи между объектами (objects), составляющие дерево существующих изделий.

**Child** (int) – привязанный объект

**Parent** (int) – объект, к которому привязан, если является корневым = 0

**Count** (float) – количество привязанных объектов в текущей системе исчисления

**Created** (datetime) – дата создания связи

**Uid** (int) – юзер, создавший связь

### Structure\_history

Все неактуальные состояния связей объектов на любой момент времени с начала запуска программы. Содержит все поля таблицы Link и дополнительно:

**deleted** (datetime) – момент создания именно этой архивной версии записи.

**huid (int)** – [history user id] пользователь, действием которого объект помещен в архив

**hcomment** (varchar(max)) – подробное описание действия, создавшего запись в архиве (если не обходимо)

**operation\_id** (int) – ссылка на таблицу [operation], содержащая индекс стандартной операции, чтобы не забивать архив излишним текстом.

**fid** (int) – [false id] уникальный для архива id. Оригинальный id объекта сохраняется в одноименном поле, что позволяет быстро получить все его состояния

**…** - далее следуют все поля таблицы [structure]

### Structure\_cross

Содержит набор дополнительных ссылок, для каждой связки из [structure], указывающих на всех вложенных в нее потомков всех уровней. Используется в интерфейсе для показа списка всех вложенных объектов при выборе элемента дерева структуры.

Позволяет избегать рекурсивных медленных запросов по всем потомкам ветки дерева и ограничиться только данной таблицей, которая по base\_link\_id вернет все связки, которые ведут до нее от корневого/корневых элементов.

Для понимания смысла и функционала таблицы см. раздел **Дополнительные ссылки.**

**base\_link** (int) – id ссылки из [structure], для которой сформирована эта дополнительная ссылка

**parent** (int) – объект-родитель (один из предков, в переборе вверх по дереву)

**child** (int) – объект-предок из ссылки base\_link

*Подробно о механизме дополнительных ссылок .*

### vObject

вьюшка, объединяющая таблицы [Object] и [Structure]. Используется для построения дерева изделий.

Привязка идет по полю **[structure][child]**. Т.е. все поля данных объекта отражают [child] объект, от родителя только [parent].

Данная таблица содержит все элементы [object], участвующие в связках, игнорируя никуда не привязанные.

dbo.object.name - наименование

dbo.object.kind – тип объекта из [object\_classificator]

dbo.object.mass - масса

dbo.object.mark – обозначение

dbo.object.icon – иконка подтипа

dbo.structure.id AS lId – id связи

dbo.object.uid AS oUid – пользователь создавший объект

dbo.structure.uid AS lUid – пользователь создавший связь

dbo.object.created AS oCreated – время создания объекта

dbo.structure.created AS lCreated – время создания связи

dbo.structure.count – количество объектов в связи

dbo.object.has\_docs – признак наличия привязанных доков у объекта

dbo.object.comment – описание объекта

dbo.structure.parent – **предок объекта**

dbo.structure.child – **сам объект**

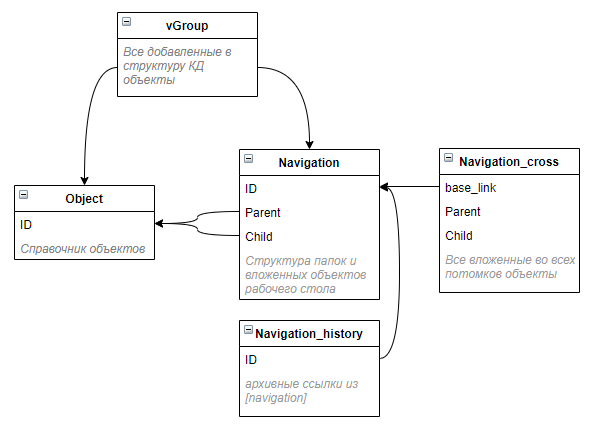
dbo.object.material\_id – материал, если объект - деталь

dbo.object.realization – префикс исполнения, если это исполнение

dbo.object.submark – альтернативное обозначение

## Структура рабочего стола

Обеспечивает показ древовидной структуры папок-разделов в интерфейсе рабочего стола и работу с ней.

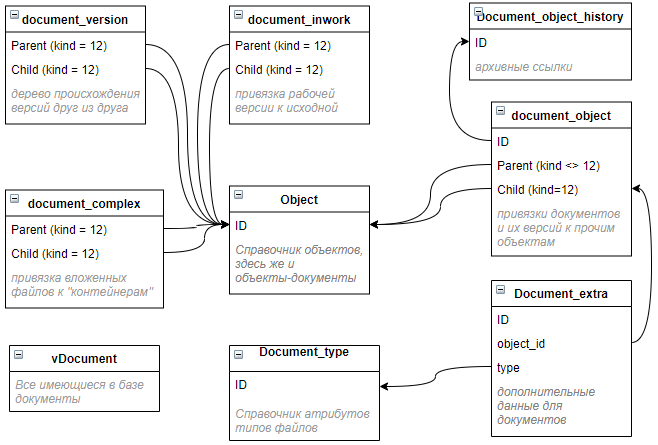


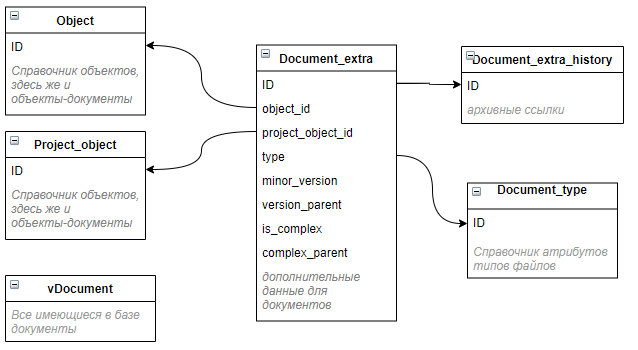
Логика и структура таблиц полностью повторяет ***структуру изделий***, только данная гораздо менее объемна и преимущественно состоит из объектов-папок.

## Версии документов

К каждому объекту можно привязать один или сколько угодно документов.

Документы в системе – так же объекты. Каждый файл может иметь несколько версий, которые являются отдельными объектами.

**Устарело**

**Актуально**

### Document\_object

Таблица, связывающая объекты-документы со своими владельцами (объектами, не являющимися документами). Каждая версия документа является отдельным объектом и имеет собственную ссылку на владельца.

**Child** (int) – привязанный объект-документ

**Parent** (int) – объект, к которому привязан

**Created** (datetime) – дата создания связи

**Uid** (int) – юзер, создавший связь (является владельцем данной версии документа)

### Document\_object\_history

Хранит все удаленные ссылки на документы. Удаленный документ не удаляется из базы, а просто удаляется рабочая ссылка, перемещаясь в данный архив.

**deleted** (datetime) – момент создания именно этой архивной версии записи.

**huid (int)** – [history user id] пользователь, действием которого объект помещен в архив

**hcomment** (varchar(max)) – подробное описание действия, создавшего запись в архиве (если не обходимо)

**operation\_id** (int) – ссылка на таблицу [operation], содержащая индекс стандартной операции, чтобы не забивать архив излишним текстом.

**fid** (int) – [false id] уникальный для архива id. Оригинальный id объекта сохраняется в одноименном поле, что позволяет быстро получить все его состояния

**…** - далее следуют все поля таблицы [document\_object]

### Document\_extra

Таблица с дополнительными данными документа, присущему только данному типу объектов.

**object\_id** [int] – ссылка на объект-документ из [object]

**is\_agree** [int] - признак, что версия документа утверждена

**is\_complex** [int] – признак того, что документ является комплексным

**is\_actual** [int] – признак, что это актуальная версия. Может быть только одна среди всех версий.

**is\_support** [int] – признак того, что это вложенный файл

**filename** [varchar](100) – имя файла с расширением

**type** [int] - ссылка на таблицу [document\_type]

**fullname** [varchar](200) – полное наименование, совпадающее с именем в хранилище файлов.

**hash** [varchar](50) – хэш текущей версии файла в хранилище.

**version** [float] – текущая минорная версия

### Document\_type

Справочник, описывающий атрибуты типов файлов.

**name** [varchar](100) - человеческое наименование типа файла

**ext** [varchar](5) – расширение с точкой

**program** [varchar](50) – имя запускающей программы (?)

**icon** [int] – идентификатор иконки. По этому индексу в программе берется картинка из imagelist c иконками типов файлов. В самой базе иконки не хранятся

### Document\_inwork

Привязка рабочих версий к исходным. Исходная и рабочая версия документа, по сути, ничем не отличаются и только наличие связки в данной таблице делает подчиненный документ рабочей версией. Удаление ссылки сделает подчиненный документ полноценной версией.

**parent** [int] – исходная версия документа

**child** [int] – рабочая версия

**uid** [int] – владелец рабочей версии

**created** [datetime] – вреся создания рабочей версии

**minor\_version** [float] – текущий минорный номер версии

### Document\_version

Связи, образующие дерево формирования версий. Какая из какой была создана.

**parent** [int] – исходная версия документа

**child** [int] – следующая наследная версия

**uid** [int] – владелец рабочей версии

**created** [datetime] – вреся создания рабочей версии

### Document\_complex

Связи, образующие комплексный документ. Каждая версия имеет собственный набор привязок ко вложенным файлам.

**parent** [int] – основной документ

**child** [int] – вложенный документ

**uid** [int] – владелец вложенного документа

**created** [datetime] – время привязки

### vDocument

Представление, показывающее все имеющиеся в системе документы. Полные версии определяются значением NULL в полях: [work\_fio], [edited], [minor\_version]

dbo.document\_extra.is\_agree – флаг согласования

dbo.document\_extra.is\_complex – является ли комплексным файлом

dbo.document\_extra.is\_actual – является ли актуальным, одним из всех версий

dbo.document\_extra.is\_support – является ли вложенным в комплексный файл

dbo.document\_extra.type – ссылка на [document\_type]

dbo.document\_extra.fullname – идентификатор для таблицы файлового хранилища

dbo.document\_extra.hash – хэш файла, хранящегося в базе

dbo.document\_extra.filename – имя файла с расширением

dbo.document\_extra.version – версия документа

dbo.document\_object.id AS link\_id, - id записи связки документов

dbo.document\_object.parent – id объекта-предка (предыдущей версии)

dbo.document\_object.child – id данного документа

dbo.document\_type.ext – расширение файла документа

dbo.object.name AS object\_name – наименование объекта-владельца

dbo.object.created AS created – время создания документа

dbo.object.comment – комментарий к документу

dbo.object.name AS doc\_name – наименование документа

full\_version – версия документа. Если рабочая версия – с минорной частью

dbo.employees.fio AS autor\_fio – владелец исходной версии

dbo.employees.id AS autor\_fio – владелец исходной версии

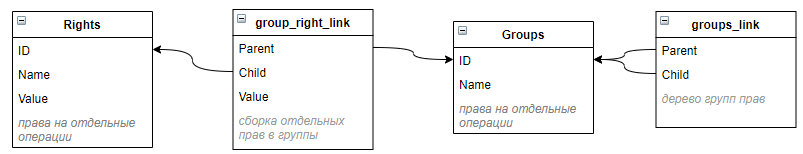
dbo.employees.fio AS work\_fio – владелец рабочей версии

dbo.document\_inwork.created AS edited – дата создания рабочей версии

dbo.document\_inwork.minor\_version – минорная часть версии, если это рабочая

dbo.document\_inwork.id AS inwork\_link\_id – id ссылки рабочей версии (ее наличие делает версию рабочей)

### Система прав



**Rights**

**ОБЪЕКТЫ**

Определение набора типов объектов, используемых как для построения навигационных связей, так и структуры физических объектов.

Актуальный список находится в базе, в таблице object\_classificator.

**Раздел**

Служит для формирования разделов навигационного дерева.  
Может содержать в себе любые типы разделов. При выборе данного раздела, объекты будут показаны в рабочем окне. Возможен выбор нескольких типов объектов, что влияет на доступный список при привязке объекта к этому разделу.

Атрибуты:

- отображаемое наименование

- иконка типа

- персональная иконка (настраиваемая)

- отображаемые типы объектов (если не указано – все типы)

- тип отображаемых данных: **список всех вложенных объектов / список объектов указанных типов**

**Объект**