**Пояснительная записка к базе данных**

1. Спорные решения:
   1. Исторические данные — отдельно создана универсальная таблица для накопления данных о действиях юзера. Но, запись истории данных ведется только по 4 таблицам, которые были указаны в проекте.
   2. Большое количество справочных таблиц — была выполнена большая степень нормализации таблиц.
   3. Для модели типа: «один элемент таблицы может быть включен в многие элементы других таблиц», сделаны отношения: основная таблица-таблица включающая ИД главной таблицы-вторая главаная таблица.
   4. Во многих ключевых таблицах, кроме Primary key формата integer создан уникальный ключ формата uniqueidentifier (некластерный индекс). Второй ключ уникальности, не является обязательным. Также существует отдельный уникальный некластерный индекс по основным названиям ключевых таблиц.
   5. Диаграмма базы данных — таблица исторических данных (history log) не соединена через foreign keys с основными таблицами базы, так как она является универсальной таблицей, которая принимает все данные.
   6. Табличный пользовательский формат данных — создан такой формат для использования в триггерах. Основная цель использования вместо тех же временных таблиц — продолжительность жизни инофрмации для передачи как аргумента процедуре. Для таких форматов была создана отдельная схема безопасности user
   7. Триггеры — на 4 основных таблицах задействованы тригерры. Суть их работы заключается в том, что при операции INSERT или UPDATE (именно в SQL, не в логике) данные об операции назначаются переменной, которая имеет табличный пользовательский формат и передается Stored Procedure как аргумент.
2. Политика имен — выбрал политику имен, которая относится к SQL naming\_conventions. Выбрал эту политику, так как на начало создания проекта не до конца понимал плюсы Camel Case, а в роли API используются Stored procedures.
3. API — как основной инструмент были выбраны stored procedures, для них также были созданы функции для упрощения кода.
   1. Stored procedures — были задействованы такие процедуры:
      1. [dbo].[usp\_accounting\_of\_time\_operations] — процедура для работы с записями учета времени.
      2. [dbo].[usp\_groups] — то же самое для групп.
      3. [dbo].[usp\_historical\_data\_log\_operations] — процедура для отображения определенных типов отчетов по историчности данных.
      4. [dbo].[usp\_logs] — процедура для занесения данных в историю.
      5. [dbo].[usp\_project\_operations] — процедура для работы с проектами
      6. [dbo].[usp\_roles] — то же самое для ролей.
      7. [dbo].[usp\_task\_operations] — работа с задачами.
      8. [dbo].[usp\_user\_month\_position] — вспомагательная процедура для работы с назначениями пользователей.
      9. [dbo].[usp\_user\_operations] — процедура для работы с пользователями.
   2. Functions — были использованы скалярные функции:
      1. [dbo].[ufn\_is\_integer] — для определения является ли переменная целым числом.
      2. [dbo].[ufn\_is\_it\_admin] — является ли пользователь админом
      3. [dbo].[ufn\_is\_it\_project\_manager] — является ли позьвователь project managerom.
      4. [dbo].[ufn\_is\_word] — является ли переменная словом, а не набором символов.
4. Запланированные улучшений — если бы это был реальный проект, то я бы сделал:
   1. Я бы спроектировал исторчиность всех таблиц, а не только указанных.
   2. После реализации базы данных систему истрочbности делал бы все-таки без использования триггеров, а попытался бы максимум использовать Stored Procedures, чтобы улучшить контроль производительности системы. Задействовал бы информацию формата: SELECT \* FROM INSERTED
   3. Как минимум задействовал бы SQL Agent для создания списка backup, Update statistics и Rebuild indexes.
   4. В зависимости от количества данных подумал бы над задействованием partitions of tables.
5. Сложности при разработке — были такие сложности:
   1. Оптимальное создание индексов на таблице, в которой происходит как SELECT так и INSERT, UPDATE.
   2. Выбор функций между скалярными и табличными.
   3. Большой блок по тестированию и отладке получился по триггерам.
   4. Нюансы с типами данных binary, varbinary и uniqueidentifier при использовании их в типах переменных, при работе с данными типа varchar и при написании и создании динамических запросов, которые выполянются через тот же EXECUTE sp\_executesql.