

НИТУ МИСИС
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОМПЬЮТЕРНЫХ НАУК

Мальцев Никита Алексеевич
Самостоятельная работа по предмету "Базы данных"

КОНТРОЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Научный руководитель:
ассистент, магистр
Валова Анастасия Александровна

Москва, 2023

Оглавление

1	Постановка задачи	3
2	Описание структуры БД	4
2.1	Вербальная модель	4
2.2	Реляционная модель	6
2.3	Анализ функциональных зависимостей	9
3	Заполнение БД информацией	11
4	Описание представлений	12
4.1	vSalaryList	14
4.2	vWeekSchedule	15
5	Описание функций	16
5.1	fn_GetGroupList	17
5.2	fn_TeacherLoad	18
6	Описание хранимых процедур	19
6.1	pr_TeacherInit	22
6.2	pr_ParentStudentInit	23
7	Описание триггеров	25
7.1	trigger_ReminderOnNumOfStudentsInGroup	26
7.2	trigger_SaveTheStory12	28
8	Примеры работы БД с использованием созданных объектов	29

1. Постановка задачи

Задача данной самостоятельной работы (в виде контрольного домашнего задания) состояла в разработке базы данных (далее БД) в соответствии с обозначенными преподавателем требованиями. Предметная область разработанной базы данных была выбрана индивидуально и согласована с преподавателем - это "Коммерческий центр по подготовке к экзаменам"(далее Центр).

Функциональное наполнение БД в свою очередь состоит из следующих частей:

- Часть, отвечающая за хранение информации о:
 - Сотрудниках
 - Учениках и их родителях
 - Преподаваемых предметах
 - Проводимых занятиях
 - Выплатах преподавателям
 - Результатах учеников на экзаменах
- Часть, отвечающая за агрегацию хранимой информации в виде:
 - Недельного расписания занятий
 - Бухгалтерского блока
 - Статистики результатов выпускившихся учеников
 - Ряд объектов, которые реализуют различные рабочие структуры и параметры работы Центра:
 - * Список групп
 - * Список преподавателей
 - * Список учеников
 - * Индивидуальное расписание преподавателей
 - * Загруженность преподавателей

2. Описание структуры БД

2.1. Вербальная модель

В соответствии с выбранной предметной областью и выбранной глубиной её описания были выделены следующие сущности:

- "Родитель" - сущность, содержащая контактные данные родителя ученика
- "Реальный ученик" - сущность, содержащая контактные данные ученика и связанная с сущностью "Родитель" при его наличии
- "Реальный преподаватель" - сущность, содержащая контактные данные преподавателя
- "Предмет" - сущность, содержащая данные о преподаваемых предметах
- "Ученик" - связка сущности "Реальный ученик" и сущности "Предмет" которому он хотел бы обучаться
- "Результаты" - сущность, содержащая данные о результатах учеников и связанная с сущностью "Ученик"
- "Преподаватель" - связка сущности "Реальный преподаватель" и сущности "Предмет" который он может вести
- "Группа преподавателя" - связка сущности "Преподаватель" и ряда дополнительных параметров, описывающих группу
- "Группа ученика" - связка сущности "Группа преподавателя" и "Ученик"
- "Расписание на неделю" недельное расписание, основанное на сущности "Группа" и параметрах, описывающих её место в расписании
- "Дни недели на русском" - сущность для выстраивания порядка дней недели, является вспомогательной для реализации инструментов по работе с БД

- "Список проведённых занятий" - список учёта действительно прошедших занятий, основанное на сущности "Расписание на неделю" и ряда дополнительных параметров, описывающих проведённое занятие
- "Оплата преподавателям" - сущность, содержащая информацию о выплатах преподавателям за их работу, связанная с сущностью "Реальный преподаватель"

2.2. Реляционная модель



Рис. 1. Даталогическая модель

ER-модель БД состоит из следующих таблиц с внутренними ключами (далее pk), связанных между собой с помощью внешних ключей (далее fk), с указанием источника внешнего ключа (from "источник"):

- Parents - "Родитель"
 - pk ParentsID
- Students - "Реальный ученик"
 - pk StudentsID
 - fk ParentsID from Parents
- Teachers - "Реальный преподаватель"
 - pk TeachersID
- Subjects - "Предмет"
 - pk SubjectsID
- TeachersSubjects - "Преподаватель"
 - pk TeachersSubjectsID
 - fk TeachersID from Teachers
 - fk SubjectsID from Subjects
- StudentsSubjects - "Ученик"
 - pk StudentsSubjectsID
 - fk StudentsID from Students
 - fk SubjectsID from Subjects
- FinalResults - "Результаты"
 - pk FinalResultsID
 - fk StudentsSubjectsID from StudentsSubjects
- GroupsTeachersSubjects - "Группа преподавателя"
 - pk GroupsTeachersSubjectsID

- fk TeachersSubjectsID from TeachersSubjects
- GroupStudentsSubjects - "Группа ученика"
 - pk GroupsStudentsSubjectsID
 - fk StudentsSubjectsID from StudentsSubjects
 - fk GroupsTeachersSubjectsID from GroupsTeachersSubjects
- WeekSchedule = "Расписание на неделю"
 - pk LessonsTimeTableID
 - fk GroupsTeachersSubjectsID from GroupsTeachersSubjects
 - fk DayOfTheWeek from WeekDaysInRussian
- WeekDaysInRussian - "Дни недели на русском"
 - pk DayID
- PastLessonsList - "Список проведённых занятий"
 - pk LessonsListID
 - fk WeekScheduleID from WeekSchedule
- PaidSalaryTransactions - "Оплата преподавателям"
 - pk PaidSalaryTransactionsID
 - fk TeachersID from Teachers

2.3. Анализ функциональных зависимостей

В текущей конфигурации БД находится в 3 нормальной форме.

Справка: Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение. Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация производится поэтапно - от приведения БД к первой форме к следующим.

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов. В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ, чего в текущей конфигурации БД не происходит.

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа. Функционально полная зависимость означает, что если потенциальный ключ является составным, то атрибут зависит от всего ключа и не зависит от его частей. Важно упомянуть, что если составные ключи отсутствуют в любом из отношений, то при указанных условиях база данных будет находиться во 2 нормальной форме по умолчанию, что и происходит в текущей конфигурации БД.

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых. Более нативно следующее определение: каждый неключевой атрибут «должен предоставлять информацию о

ключе, полном ключе и ни о чём, кроме ключа». Данное условие также выполняется для каждой из сущностей БД в текущей конфигурации, что в итоге и приводит нас к 3 нормальной форме для БД.

3. Заполнение БД информацией

Источником данных для заполнения БД послужил импорт данных из моей головы.

Заполненность каждого из отношений:

- Parents составляет 3 элемента
- Students составляет 10 элементов
- Teachers составляет 4 элемента
- Subjects составляет 10 элементов
- TeachersSubjects составляет 7 элементов
- StudentsSubjects составляет 21 элемент
- FinalResults составляет 1 элемент
- GroupsTeachersSubjects составляет 7 элементов
- GroupStudentsSubjects составляет 20 элементов
- WeekSchedule составляет 19 элементов
- WeekDaysInRussian составляет 7 элементов
- PastLessonsList составляет 8 элементов
- PaidSalaryTransactions составляет 3 элемента

4. Описание представлений

БД содержит следующие представления:

- vWeekSchedule
 - Недельное расписание в формате:
 - * День
 - * Время
 - * Номер группы
 - * Имя и фамилия преподавателя
 - * Номер кабинета
- vSalaryList
 - Таблица бухгалтерского учёта в формате:
 - * Имя, фамилия и отчество преподавателя
 - * Количество денег, полученное Центром за занятия, проведенные данным преподавателем
 - * Доля преподавателя
 - * Выплаты наличными
 - * Выплаты на карту
 - * Остаток по выплате
- vSubjectsStatistic
 - Статистика по предметам в формате, собранная по результатам экзаменов:
 - * Количество преподавателей по данному предмету
 - * Количество студентов, сдавших предмет
 - * Название предмета
 - * Тип предмета (ЕГЭ, ОГЭ, Текущее)
 - * Средний результат
- vTeachersSubjectsInfo

- Список преподаватель - предмет для создания потенциальных групп
- vStudentsSubjectsInfo
 - Список ученик - предмет для добавления в группу
- vGroupTeachersSubjectsInfo
 - Список групп
- vGroupStudentsSubjectsInfo
 - Список ученик - группа
- vEmptyGroups
 - Список групп, которые ещё не содержат учеников, но открыты для заполнения
- vStudentsWithParents
 - Список студентов с указанием данных родителей.

4.1. vSalaryList

```

create view vSalaryList as
select TutorOffice.Teachers.Name,
TutorOffice.Teachers.SecondName,
TutorOffice.Teachers.LastName,
FullIncome, FullIncome*0.5 as TutorsIncome,
AllPaidByCash,AllPaidByCardMethod,
(FullIncome*0.5-AllPaidByCash-AllPaidByCardMethod) as ToPay
from (select sum(PaidInCash) as AllPaidByCash,
sum(PaidInCardMethod) as AllPaidByCardMethod,
TutorOffice.Teachers.Name as Name,
TutorOffice.Teachers.TeachersID as ID
from TutorOffice.PaidSalaryTransactions
join TutorOffice.Teachers
on PaidSalaryTransactions.TeachersID=Teachers.TeachersID
group by TutorOffice.Teachers.TeachersID,
TutorOffice.Teachers.Name) as PSTT
join (select sum(Cash) as FullIncome,
Teachers.TeachersID as ID
from TutorOffice.PastLessonsList
join TutorOffice.WeekSchedule
on PastLessonsList.WeekScheduleID
=WeekSchedule.LessonsTimeTableID
join TutorOffice.GroupsTeachersSubjects as GTS on
WeekSchedule.GroupsTeachersSubjectsID
=GTS.GroupsTeachersSubjectsID
join TutorOffice.TeachersSubjects as TS
on GroupsTeachersSubjects.TeachersSubjectsID
=TS.TeachersSubjectsID
join TutorOffice.Teachers
on TeachersSubjects.TeachersID=Teachers.TeachersID
group by Teachers.TeachersID) as PTSD
on PSTT.ID=PTSD.ID
join TutorOffice.Teachers on TeachersID=PTSD.ID

```

4.2. vWeekSchedule

```

create view vWeekSchedule as
select DayName, StartTime,
TutorOffice.WeekSchedule.GroupsTeachersSubjectsID as GroupID,
TutorOffice.Teachers.Name as TutorsName,
TutorOffice.Teachers.SecondName as TutorsSername,
ClassroomNumber
from TutorOffice.WeekSchedule
join TutorOffice.WeekDaysInRussian
on WeekSchedule.DayOfTheWeek
= WeekDaysInRussian.DayID
join TutorOffice.GroupsTeachersSubjects
on WeekSchedule.GroupsTeachersSubjectsID
= GroupsTeachersSubjects.GroupsTeachersSubjectsID
join TutorOffice.TeachersSubjects
on GroupsTeachersSubjects.TeachersSubjectsID
= TeachersSubjects.TeachersSubjectsID
join TutorOffice.Teachers
on TeachersSubjects.TeachersID
= Teachers.TeachersID
where TutorOffice.GroupsTeachersSubjects.GroupsTeachersSubjectsID
=TutorOffice.WeekSchedule.GroupsTeachersSubjectsID and Status=1
order by TutorOffice.WeekDaysInRussian.DayOrderNumber,
TutorOffice.WeekSchedule.StartTime ASC OFFSET 0 ROWS

```

5. Описание функций

БД содержит следующие функции:

- `fn_GetGroupList`
 - Данная функция является табличной. Она принимает в качестве аргумента номер группы и позволяет получить список учеников, которые в данной группе состоят.
- `fn_TeacherLoad`
 - Данная функция является скалярной. Она принимает в качестве аргументов данные преподавателя и позволяет узнать текущую нагрузку этого преподавателя. Размерность значений данной функции - количество занятий в неделю.
- `fn_TeacherWeekLessons`
 - Данная функция является табличной. Она принимает в качестве аргументов данные о преподавателе и позволяет получить его личное недельное расписание.

5.1. fn_GetGroupList

```

create function fn_GetGroupList(@GroupID as int)
returns table as return
(select TutorOffice.Students.Name as Name,
TutorOffice.Students.SecondName as SecondName,
TutorOffice.Students.LastName as LastName,
TutorOffice.StudentsSubjects.StartLevel as StudentsLevel,
TutorOffice.GroupsTeachersSubjects.Level as GroupLevel
from TutorOffice.Students
join TutorOffice.StudentsSubjects
on Students.StudentsID = StudentsSubjects.StudentsID
join TutorOffice.GroupsStudentsSubjects
on StudentsSubjects.StudentsSubjectsID
= GroupsStudentsSubjects.StudentsSubjectsID
join TutorOffice.GroupsTeachersSubjects
on GroupsStudentsSubjects.GroupsTeachersSubjectsID
= GroupsTeachersSubjects.GroupsTeachersSubjectsID
where GroupsStudentsSubjects.GroupsTeachersSubjectsID=@GroupID)

```

5.2. fn_TeacherLoad

```

create function fn_TeacherLoad
(@TeacherName as nvarchar(50),
@TeacherSecondName as nvarchar(50),
@TeacherLastName as nvarchar(50))
returns int
with execute as caller as
begin
declare @Load int = (select
count(TutorOffice.WeekSchedule.LessonsTimeTableID) as Load
from TutorOffice.WeekSchedule
join TutorOffice.GroupsTeachersSubjects
on WeekSchedule.GroupsTeachersSubjectsID
= GroupsTeachersSubjects.GroupsTeachersSubjectsID
join TutorOffice.TeachersSubjects
on GroupsTeachersSubjects.TeachersSubjectsID
= TeachersSubjects.TeachersSubjectsID
join TutorOffice.Teachers
on TeachersSubjects.TeachersID = Teachers.TeachersID
where TutorOffice.Teachers.Name=@TeacherName
and TutorOffice.Teachers.SecondName=@TeacherSecondName
and TutorOffice.Teachers.LastName=@TeacherLastName
and TutorOffice.WeekSchedule.Status=1)
return @Load
end

```

6. Описание хранимых процедур

БД содержит следующие хранимые процедуры:

- pr_TeacherInit
 - Данная процедура принимает в качестве аргументов данные о преподавателе и позволяет инициировать его появление в БД, т.е. создать учителя.
- pr_InitSubject
 - Данная процедура принимает в качестве аргументов данные о преподаваемом предмете и позволяет инициировать его появление в БД, т.е. создать преподаваемый предмет.
- pr_ConnectSubjectAndTeacher
 - Данная процедура принимает в качестве аргументов данные о преподавателе и предмете, который он будет преподавать и позволяет инициировать появление этой связки в БД, т.е. создать учителя, который преподаёт конкретный предмет.
- pr_ParentStudentInit
 - Данная процедура принимает в качестве аргументов данные о будущем ученике и его родителе и позволяет инициировать появление этой связки в БД, т.е. создать ученика с родителем или без, если ученик пришёл в центр самостоятельно.
- pr_ConnectionSubjectAndStudent
 - Данная процедура принимает в качестве аргументов данные о ученике и предмете, которому он хотел бы обучаться, и позволяет инициировать появление этой связки в БД, т.е. создать ученика, который будет обучаться определенному предмету.
- pr_AddFinalResultForStudent

- Данная процедура принимает в качестве аргументов данные об ученике и его результатах по итогу обучения, и позволяет инициировать появление этой связки в БД, т.е. создать учтённый результат работы с учеником.
- pr_GroupInit
 - Данная процедура принимает в качестве аргументов данные о связке преподаватель-предмет и ряд дополнительных параметров для создания группы, и позволяет инициировать появление этой связки параметров в БД, т.е. создать группу, т.е. создать группу по определенному предмету с определенным преподавателем, который её ведёт.
- pr_AddStudentWithSubjectToGroup
 - Данная процедура принимает в качестве аргументов данные о связке ученик-предмет и группе, в которой он мог бы заниматься, и позволяет инициировать появление этой этой связки параметров в БД, т.е. создать ученика, который хотел бы обучаться определенному предмету, связанного с группой, в которой он мог бы обучаться.
- pr_AddNewLessonToSchedule
 - Данная процедура принимает в качестве аргументов данные о группе, времени проведения занятия и позволяет инициировать появление этой связки в БД, т.е. создать занятие для определенной группы, включённое в текущее недельное расписание.
- pr_AddPastLessonToListOfLessons
 - Данная процедура принимает в качестве аргументов данные о занятии из расписания, а также количество человек, пришедших на занятие, и позволяет инициировать появление этой связки в БД, т.е. создать запись о проведённом занятии.
- pr_AddPaidTransaction

- Данная процедура принимает в качестве аргументов данные о преподавателе и количестве денег, которое было данному преподавателю выплачено в качестве зарплаты, и позволяет инициировать появление этой связки в БД, т.е. создать запись о выдаче зарплаты определённому преподавателю.
- `pr_DeprecatesGroup`
 - Данная процедура принимает в качестве аргументов данные о группе и позволяет вывести эту групп из активного оборота, т.е. обновить данные о группе, дабы она считалась закрытой.

6.1. pr_TeacherInit

```
create procedure pr_TeacherInit
(@Name nvarchar(50),
@SecondName nvarchar(50),
@LastName nvarchar(50),
@Age int, @Phone varchar(50),
@Mail varchar(50), @WebLink varchar(50)) AS
if not EXISTS
(select * from TutorOffice.Teachers
where Name = @Name and SecondName
= @SecondName and LastName
= @LastName and Phone = @Phone)
insert into TutorOffice.Teachers(Name, SecondName,
LastName, Age, Phone, Mail, WebLink)
VALUES (@Name, @SecondName, @LastName,
@Age, @Phone, @Mail, @WebLink)
```

6.2. pr_ParentStudentInit

```

create procedure pr_ParentStudentInit
(@IfParentExistWrite1Else0 int,
@StudentName nvarchar(50),
@StudentSecondName nvarchar(50),
@StudentLastName nvarchar(50),
@StudentPhone varchar(50),
@StudentMail varchar(50) = NULL,
@StudentWebLink varchar(50) = NULL,
@ParentName nvarchar(50) = NULL,
@ParentSecondName nvarchar(50) = NULL,
@ParentLastName nvarchar(50) = NULL,
@ParentPhone varchar(50) = NULL,
@ParentMail varchar(50) = NULL,
@ParentWebLink varchar(50) = NULL)
AS
if @IfParentExistWrite1Else0 = 1
begin
if not EXISTS(select *
from TutorOffice.Parents where Name=@ParentName and
SecondName=@ParentSecondName and Phone=@ParentPhone)
begin
insert into TutorOffice.Parents (Name, SecondName,
LastName, Phone, Mail, WebLink)
values (@ParentName,@ParentSecondName,@ParentLastName,
@ParentPhone,@ParentMail,@ParentWebLink)
declare @CurrParentID int = ident_current('TutorOffice.Parents')
if not EXISTS(select * from TutorOffice.Students
where ParentsID=@CurrParentID and
Name=@StudentName
and SecondName=@StudentSecondName
and Phone=@StudentPhone)
begin
insert into TutorOffice.Students (Name, secondname,

```

```

lastname, phone, mail, weblink,parentsid)
values (@StudentName,@StudentSecondName,@StudentLastName,
@StudentPhone,@StudentMail,@Student,
WebLink,@CurrParentID) end end
if EXISTS(select * from TutorOffice.Parents
where Name=@ParentName and
SecondName=@ParentSecondName and Phone=@ParentPhone)
begin
declare @CurrParentID2 int
=(select TutorOffice.Parents.ParentsID
from TutorOffice.Parents
where Name=@ParentName
and SecondName=@ParentSecondName
and Phone=@ParentPhone)
if not EXISTS(select *
from TutorOffice.Students
where ParentsID=@CurrParentID2 and
Name=@StudentName and SecondName=@StudentSecondName
and Phone=@StudentPhone)
begin
insert into TutorOffice.Students (Name, secondname,
lastname, phone, mail, weblink, parentsid)
values (@StudentName,@StudentSecondName,@StudentLastName,
@StudentPhone,@StudentMail,@Student,WebLink,@CurrParentID2)
end end end
if @IfParentExistWrite1Else0 = 0
begin
if not EXISTS(select *
from TutorOffice.Students where Name=@StudentName
and SecondName=@StudentSecondName
and LastName=@StudentLastName)
insert into TutorOffice.Students (Name, secondname, lastname,
phone, mail, weblink, parentsid)
values (@StudentName,@StudentSecondName,@StudentLastName,
@StudentPhone,@StudentMail,@StudentWebLink,Null) end

```


7. Описание триггеров

- TutorOffice.trigger_ReminderOnNumOfStudentsInGroup
 - Данный триггер не позволяет добавлять в группу избыточное количество учеников и действует на заполнение в таблице GroupsStudentsSubjects("Группа ученика"). В данной конфигурации в следствии работы этого триггера в БД в группе может быть максимум 6 человек. Также триггер не позволяет добавить ученика в группу, которая создана для индивидуальных занятий. Небольшое пояснение: различия занятий с одним учеников в группе и индивидуальной группой состоит в первую очередь в цене, для индивидуального занятия она больше. Идея в том, что один ученик в не индивидуальной группе - это проблема для Центра, соответственно, как только появится возможность такую группу дополнить новыми участниками, это будет сделано. А вот тот, кто занимался индивидуально будет продолжать работу в таком режиме до конца обучение. Так что потенциальный убыток от таких занятий минимален. Плюс, всегда есть возможность повременить с включением таких "неудобных" занятий в расписание до момента определенного наполнения, если это возможно. Данный вопрос потенциально должен решается не на уровне БД, так как зависит в большей степени от человеческого фактора.
- TutorOffice.trigger_SaveTheStory[1-11]
 - Данная серия триггеров, в совокупности с триггером trigger_SaveTheStory12, не позволяет удалять данные из базы данных.
- TutorOffice.trigger_SaveTheStory12
 - Данный триггер не позволяет вносить любые изменения в таблицу WeekDaysInRussian, так как там хранятся данные о днях недели и их порядке, которые никогда не должны изменяться в процессе работы БД.

7.1. trigger_ReminderOnNumOfStudentsInGroup

```

create trigger TutorOffice.trigger_ReminderOnNumOfStudentsInGroup
on TutorOffice.GroupsStudentsSubjects
after insert as
if (ROWCOUNT_BIG() = 0)return;
declare @tmpGroupID int = (select inserted.GroupsTeachersSubjectsID
from inserted)
declare @tmpStudSubjID int = (select inserted.StudentsSubjectsID
from inserted)
if ((select
count(TutorOffice.GroupsStudentsSubjects.GroupsStudentsSubjectsID)
from TutorOffice.GroupsStudentsSubjects
join inserted
on GroupsStudentsSubjects.GroupsTeachersSubjectsID
= inserted.GroupsTeachersSubjectsID
where TutorOffice.GroupsStudentsSubjects.GroupsTeachersSubjectsID
= inserted.GroupsTeachersSubjectsID)> 5)
begin
raiserror (N'В данную группу добавить нового ученика нельзя,
она полностью заполнена.',16,1)
delete from TutorOffice.GroupsStudentsSubjects
where GroupsTeachersSubjectsID=@tmpGroupID
and StudentsSubjectsID=@tmpStudSubjID end
if ((select count(TutorOffice.GroupsStudentsSubjects.GroupsStudentsSubj
from TutorOffice.GroupsStudentsSubjects
join inserted
on GroupsStudentsSubjects.GroupsTeachersSubjectsID
= inserted.GroupsTeachersSubjectsID
join TutorOffice.GroupsTeachersSubjects
on GroupsStudentsSubjects.GroupsTeachersSubjectsID
= GroupsTeachersSubjects.GroupsTeachersSubjectsID
where TutorOffice.GroupsStudentsSubjects.GroupsTeachersSubjectsID
= inserted.GroupsTeachersSubjectsID) = 1)

```

```
and EXISTS(select * from TutorOffice.GroupsTeachersSubjects
where TutorOffice.GroupsTeachersSubjects.GroupsTeachersSubjectsID
= @tmpGroupID and PricePerLesson>700)
begin
raiserror (N'В данную группу добавить нового ученика нельзя,
так как это группа для индивидуальных занятий.',16,1)
delete from TutorOffice.GroupsStudentsSubjects
where GroupsTeachersSubjectsID=@tmpGroupID
and StudentsSubjectsID=@tmpStudSubjID end
```

7.2. trigger_SaveTheStory12

```
create trigger TutorOffice.trigger_SaveTheStory12
on TutorOffice.WeekDaysInRussian
instead of insert, delete, update
as
if (ROWCOUNT_BIG() = 0)
return;
RAISERROR (N'Необходимо удалить триггер
TutorOffice.trigger_SaveTheStory12
чтобы удалить данные из базы данных', 10, 1)
```

8. Примеры работы БД с использованием созданных объектов

Для демонстрации возможностей БД, произведём серию действий для обработки следующей рабочей ситуации.

В центр приходит новый ученик Василий Простой со своим отцом Николаем Владимировичем. Он хочет подготовиться к экзаменам по математике и русскому языку в 11 классе. Оставляет свои данные, а также упоминает, что хотел бы заниматься в группе, а не индивидуально. Далее, он пишет наш внутренний тест для определения уровня. Выходит, что по результатам теста он имеет средний уровень по математике (2 в нашей системе оценки) и слабый по русскому (1 в нашей системе оценки). Далее, он отправляется домой, в ожидании нашего ответа.

Чтобы обработать данную заявку на обучение, в первую очередь внесём информацию о нашем потенциальном ученике и его родителе в БД. Для этого воспользуемся процедурой `pr_ParentStudentInit`. Состояние БД до заполнения проиллюстрировано на рисунке 2, а состояние после на рисунке 3. Те же данные можно получить в том же виде через представление `vStudentsWithParents`.

TutorOfficeSetupconsole_1ParentsStudentsTx: AutoDDL

WHEREORDER BY

	StudentsID	Name	SecondName	LastName	Phone	Mail	WebLink	ParentsID
1	14	Степан	Чесноков	Кириллович	87465876775	raz@gaf.com	@tgraz	<null>
2	15	Иван	Маслов	Аркадьевич	86476535521	<null>	<null>	<null>
3	17	Марина	Тамбовская	Кириллович	87422876775	raz@gaf.com	@tgraz	<null>
4	19	Карина	Власова	Кириллович	87422876775	raz@gaf.com	@tgraz	3
5	23	Таисия	Путина	Васильевна	85874763631	<null>	<null>	<null>
6	24	Дарья	Осталопова	Марковна	87422876733	razdvatri@gaf.com	@tgdvz	7
7	25	Алексей	Осталопов	Маркович	87433876733	razd123tri@gaf.com	@tgqevz	7
8	26	Юлия	Осталопова	Марковна	82233876733	ra123123tri@gaf.com	@tgqevz5836	7
9	27	Анна	Гончарова	Даниловна	82244476733	5433tri@gaf.com	@tgqevz583340	8
10	33	Иван	Тестов	Иванович	87654345432	<null>	<null>	<null>

TutorOfficeSetupconsole_1ParentsStudentsTx: AutoDDL

WHEREORDER BYParentsID DESC

	ParentsID	Name	SecondName	LastName	Phone	Mail	WebLink
1	8	Алиса	Гончарова	Павловна	88883333565	dvladneum@kss.com	@tgkoko
2	7	Анастасия	Осталопова	Егорова	86473333565	dvamum@kss.com	@tgkoko
3	3	Жанна	Власова	Егорова	86473712565	razmum@kss.com	@vkko

Рис. 2. Состояние до внесения

```
exec pr_ParentStudentInit
```

```

@IfParentExistWrite1Else0 = 1, @StudentName = Василий,
@StudentSecondName = Простой, @StudentLastName = Николаевич,
@StudentPhone = 89876789876, @StudentMail = 'prost@mail.com',
@StudentWebLink = '@tgprost', @ParentName = Николай,
@ParentSecondName = Простой, @ParentLastName = Владимирович,
@ParentPhone = 87564737564, @ParentMail = Null,
@ParentWebLink = Null

```

The screenshot shows a database management tool with two tables displayed. The top table is 'Students' and the bottom table is 'Parents'.

Students Table:

StudentsID	Name	SecondName	LastName	Phone	Mail	WebLink	ParentsID
1	14	Степан	Чесноков	Кириллович	87465876775	raz@gaf.com	<null>
2	15	Иван	Маслов	Аркадьевич	86476535521	<null>	<null>
3	17	Марина	Тамбовская	Кириллович	87422876775	raz@gaf.com	<null>
4	19	Карина	Власова	Кириллович	87422876775	raz@gaf.com	3
5	23	Таисия	Путина	Васильевна	85874763631	<null>	<null>
6	24	Дарья	Осталопова	Марковна	87422876733	razdvatri@gaf.com	7
7	25	Алексей	Осталопов	Маркович	87433876733	razd123tri@gaf.com	7
8	26	Юлия	Осталопова	Марковна	82233876733	ra123123tri@gaf.com	7
9	27	Анна	Гончарова	Даниловна	82244476733	5433tri@gaf.com	8
10	33	Иван	Тестов	Иванович	87654345432	<null>	<null>
11	34	Василий	Простой	Николаевич	89876789876	prost@mail.com	@tgprost

Parents Table:

ParentsID	Name	SecondName	LastName	Phone	Mail	WebLink
1	9	Николай	Простой	Владимирович	87564737564	<null>
2	8	Алиса	Гончарова	Павловна	88883333565	dvldneum@kss.com
3	7	Анастасия	Осталопова	Егорова	86473333565	dvaum@kss.com
4	3	Жанна	Власова	Егорова	86473712565	raznum@kss.com

Рис. 3. Состояние после внесения

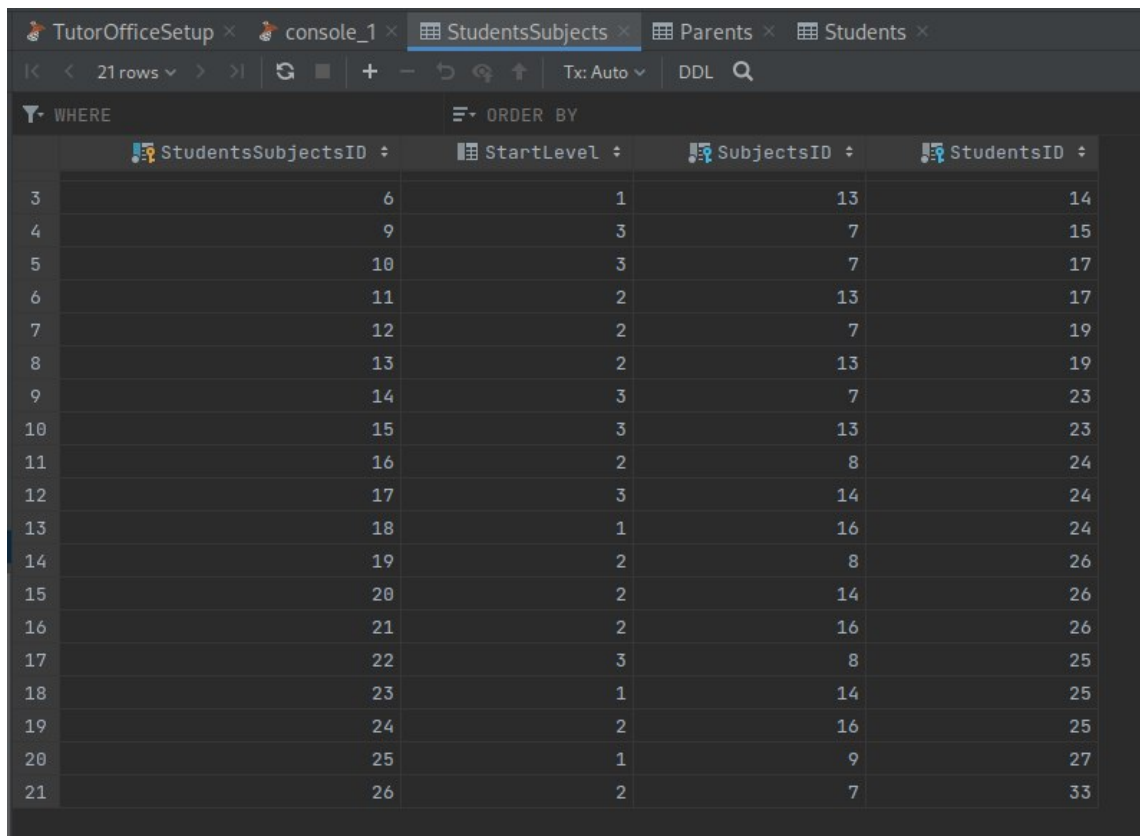
Далее, занесем данные в БД о том, чему конкретно хотел бы обучаться Вася в нашем центре с помощью процедуры `pr_ConnectionSubjectAndStudent`. Состояние БД до заполнения проиллюстрировано на рисунке 4, а состояние после на рисунке 6.

```

exec pr_ConnectionSubjectAndStudent @StudentName = Василий,
@StudentSecondName = Простой, @SubjectName = Математика,
@SubjectType = ЕГЭ, @StartedLvl = 2
exec pr_ConnectionSubjectAndStudent @StudentName = Василий,
@StudentSecondName = Простой, @SubjectName = РусскийЯзык,
@SubjectType = ЕГЭ, @StartedLvl = 1

```

Далее проверим возможные группы для добавления, оценив информацию в представлении `vGroupsTeachersSubjectsInfo`. Результаты поиска на рисунке 7.



	StudentsSubjectsID	StartLevel	SubjectsID	StudentsID
3	6	1	13	14
4	9	3	7	15
5	10	3	7	17
6	11	2	13	17
7	12	2	7	19
8	13	2	13	19
9	14	3	7	23
10	15	3	13	23
11	16	2	8	24
12	17	3	14	24
13	18	1	16	24
14	19	2	8	26
15	20	2	14	26
16	21	2	16	26
17	22	3	8	25
18	23	1	14	25
19	24	2	16	25
20	25	1	9	27
21	26	2	7	33

Рис. 4. Состояние до внесения

Окажется, что подходящая группа существует, у неё номер 1. Теперь добавим туда нового ученика с помощью процедуры `pr_AddStudentWithSubjectToGroup`. Состояние БД после на рисунке 8.

```
exec pr_AddStudentWithSubjectToGroup
@GroupID = 1, @StudentName = Василий,
@StudentSecondName = Простой,
@SubjectName = Математика, @SubjectType = ЕГЭ
```

Результат будет не совсем тот, которого мы хотели достиг. Данная процедура не повлияет на состояние базы данных, так как триггер `trigger_ReminderOnNumOfStudentsInGroup` не позволит добавить в группу 7 человека из-за переполнения.

Других групп с необходимыми параметрами, с уже проводящимися в данный момент занятиями, нет. Выхода из сложившейся ситуации два. Либо существует группа, но там ещё нет зарегистрированных участников, т.е. она в процессе первоначального набора (подобные группы мы можем найти в представлении `vEmptyGroups`), либо можно

предложить ученику индивидуальные занятия. Но сейчас нам везёт и в списке пустых групп существует группа (с номером 4), подходящая под параметры. Результаты поиска на рисунке 9.

Советуемся с преподавателем данной группы. Представим, что он согласен на работу с данным учеником, а значит мы можем действовать дальше. Теперь добавим Василия в эту группу с помощью процедуры `pr_AddStudentWithSubjectToGroup`:

```
exec pr_AddStudentWithSubjectToGroup
@GroupID = 4, @StudentName = Василий,
@StudentSecondName = Простой,
@SubjectName = Математика, @SubjectType = ЕГЭ
```

Результат увидим в представлении `vGroupTeacherSubjectsInfo`, в котором появится запись о новой группе (с номером 4), где есть один участник. Результат проиллюстрирован на рисунке 10

Далее представим, что преподаватель готов начать занятия с группой из одного человека. В таком случае, нам необходимо вставить занятие в расписание. Проверим свободные места в расписании с помощью представления `vWeekSchedule`. Результат проиллюстрирован на рисунке 11.

Далее выбираем подходящее нам время и вносим занятия в расписание с помощью процедуры `pr_AddNewLessonToSchedule`. Результаты проиллюстрированы на рисунке 12.

```
exec pr_AddNewLessonToSchedule
@StartTime = '16:30', @DayOfTheWeek = N'Понедельник',
@ClassroomNumber = 1, @GroupID = 4
exec pr_AddNewLessonToSchedule
@StartTime = '16:30', @DayOfTheWeek = N'Среда',
@ClassroomNumber = 1, @GroupID = 4
exec pr_AddNewLessonToSchedule
@StartTime = '16:30', @DayOfTheWeek = N'Пятница',
@ClassroomNumber = 1, @GroupID = 4
```

И, последний шаг. Представим, что ученик начал свои занятия на второй неделе, после получения информации о времени их проведе-

ния. Внесём в список проведённых первое проведённое занятие, прошедшее в понедельник. Для это используем процедуру `pr_AddPastLessonToListOfLessons` и заполним информацию о занятиях, прошедших в понедельник.

```
exec pr_AddPastLessonToListOfLessons
@StartTime = '16:30', @DayOfTheWeekName = Понедельник,
@WeekNumber = 2, @GroupID = 4, @NumOfMembers = 1
exec pr_AddPastLessonToListOfLessons
@StartTime = '18:00', @DayOfTheWeekName = Понедельник,
@WeekNumber = 2, @GroupID = 1, @NumOfMembers = 4
exec pr_AddPastLessonToListOfLessons
@StartTime = '19:30', @DayOfTheWeekName = Понедельник,
@WeekNumber = 2, @GroupID = 8, @NumOfMembers = 2
```

Изменения по зарплате для преподавателей, в результате проведенных занятий, отражаются в представлении `vSalaryList`. Иллюстрацию работы на рисунках до 13 и после 14.

В результате был разобран один из самых популярных сценариев по работе в данной БД, с детальной иллюстрацией пошаговых изменений для наглядной визуализации функционала разработатной БД.

	StudentsSubjectsID	StartLevel	SubjectsID	StudentsID
8	13	2	13	19
9	14	3	7	23
10	15	3	13	23
11	16	2	8	24
12	17	3	14	24
13	18	1	16	24
14	19	2	8	26
15	20	2	14	26
16	21	2	16	26
17	22	3	8	25
18	23	1	14	25
19	24	2	16	25
20	25	1	9	27
21	26	2	7	33
22	27	2	7	34
23	28	1	13	34

Рис. 5. Состояние после внесения через состояние таблицы

	StudentsSubjectsID	StartLevel	SubjectsID	StudentsID
8	13	2	13	19
9	14	3	7	23
10	15	3	13	23
11	16	2	8	24
12	17	3	14	24
13	18	1	16	24
14	19	2	8	26
15	20	2	14	26
16	21	2	16	26
17	22	3	8	25
18	23	1	14	25
19	24	2	16	25
20	25	1	9	27
21	26	2	7	33
22	27	2	7	34
23	28	1	13	34

Рис. 6. Состояние после внесения, выраженное через состояние более наглядного представления vStudentsSubjectsInfo

	GroupsTeachersSubjectsID	Name	SecondName	SubjectName	Type	PricePerLesson	Num
1	1	Никита	Мальцев	Математика	ЕГЭ	500.0000	6
2	3	Даниил	Тарасов	Программирование	Текущее	600.0000	3
3	5	Никита	Мальцев	Математика	ОГЭ	500.0000	3
4	8	Мария	Самусева	РусскийЯзык	ЕГЭ	500.0000	4
5	9	Мария	Самусева	РусскийЯзык	ОГЭ	500.0000	3

Рис. 7. Текущие работающие группы

```
TestOfficeSetup> exec pr_AddStudentWithSubjectToGroup @GroupID = 1, @StudentName = Василий, @StudentSecondName = Простой, @SubjectName = Математика, @SubjectType =
[2022-12-07 21:10:19] [50001][50000] Line 13: В данную группу добавить нового ученика нельзя, она полностью заполнена.
```

Рис. 8. Попытка записать человека в полностью заполненную группу

	GroupsTeachersSubjectsID	Name	SecondName	SubjectName	Type	Level	PricePerLesson
1	4	Никита	Мальцев	Математика	ЕГЭ	1	500.0000

Рис. 9. Список пустых групп

	GroupsTeachersSubjectsID	Name	SecondName	SubjectName	Type	PricePerLesson	Num
1	1	Никита	Мальцев	Математика	ЕГЭ	500.0000	6
2	4	Никита	Мальцев	Математика	ЕГЭ	500.0000	1
3	5	Никита	Мальцев	Математика	ОГЭ	500.0000	3
4	3	Даниил	Тарасов	Программирование	Текущее	600.0000	3
5	8	Мария	Самусева	РусскийЯзык	ЕГЭ	500.0000	4
6	9	Мария	Самусева	РусскийЯзык	ОГЭ	500.0000	3

Рис. 10. Обновленный список рабочих групп

	DayName	StartTime	GroupID	TutorsName	TutorsSername	ClassroomNumber
1	Понедельник	18:00:00	1	Никита	Мальцев	1
2	Понедельник	19:30:00	8	Мария	Самусева	1
3	Вторник	16:30:00	3	Даниил	Тарасов	2
4	Вторник	18:00:00	5	Никита	Мальцев	4
5	Вторник	18:00:00	9	Мария	Самусева	1
6	Среда	18:00:00	1	Никита	Мальцев	1
7	Среда	19:30:00	8	Мария	Самусева	1
8	Четверг	16:30:00	3	Даниил	Тарасов	2
9	Четверг	18:00:00	5	Никита	Мальцев	4
10	Четверг	18:00:00	9	Мария	Самусева	1
11	Пятница	18:00:00	1	Никита	Мальцев	1
12	Пятница	19:30:00	8	Мария	Самусева	1

Рис. 11. Недельное расписание до добавления новых занятий

	DayName	StartTime	GroupID	TutorsName	TutorsSername	ClassroomNumber
1	Понедельник	18:00:00	1	Никита	Мальцев	1
2	Среда	18:00:00	1	Никита	Мальцев	1
3	Пятница	18:00:00	1	Никита	Мальцев	1
4	Четверг	16:30:00	3	Даниил	Тарасов	2
5	Вторник	16:30:00	3	Даниил	Тарасов	2
6	Понедельник	16:30:00	4	Никита	Мальцев	1
7	Среда	16:30:00	4	Никита	Мальцев	1
8	Пятница	16:30:00	4	Никита	Мальцев	1
9	Четверг	18:00:00	5	Никита	Мальцев	4
10	Вторник	18:00:00	5	Никита	Мальцев	4
11	Понедельник	19:30:00	8	Мария	Самусева	1
12	Среда	19:30:00	8	Мария	Самусева	1
13	Пятница	19:30:00	8	Мария	Самусева	1
14	Четверг	18:00:00	9	Мария	Самусева	1
15	Вторник	18:00:00	9	Мария	Самусева	1

Рис. 12. Недельное расписание после добавления новых занятий

	Name	SecondName	LastName	FullIncome	TutorsIncome	AllPaidByCash	AllPaidByCardMethod	ToPay
1	Никита	Мальцев	Алексеевич	6000.0000	3000.00000	1000.0000	500.0000	1500.00000
2	Даниил	Тарасов	Игоревич	2600.0000	1300.00000	0.0000	0.0000	1300.00000
3	Мария	Самусева	Игоревна	5000.0000	2500.00000	1000.0000	2000.0000	-500.00000

Рис. 13. Список проведённых занятий после добавления занятий за понедельник

	Name	SecondName	LastName	FullIncome	TutorsIncome	AllPaidByCash	AllPaidByCardMethod	ToPay
1	Никита	Мальцев	Алексеевич	8500.0000	4250.00000	1000.0000	500.0000	2750.00000
2	Даниил	Тарасов	Игоревич	2600.0000	1300.00000	0.0000	0.0000	1300.00000
3	Мария	Самусева	Игоревна	6000.0000	3000.00000	1000.0000	2000.0000	0.00000

Рис. 14. Список проведённых занятий после добавления занятий за понедельник