Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт электронных и информационных систем

Кафедра информационных технологий и систем

**Зачисление в институт**

**курсовая работа по дисциплине «Базы данных»**

направления подготовки 09.04.01 «Информатика и вычислительная техника»

Преподаватель:

\_\_\_\_\_\_\_\_\_\_\_\_ / Телина И.C

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил студент группы 8091:

\_\_\_\_\_\_\_\_\_\_ / Васильев И.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Великий Новгород

2020

Содержание:

[**Введение**](#_7lcxk0d0666w) **3**

[**Структурный анализ**](#_v5iqqgwtei2e) **4**

[Анализ предметной области](#_bzof3n5lben5) 4

[Анализ требований к информационной системе](#_qg5cf4v6fe05) 5

[Инфологическая модель данных.](#_vv5j31q3pl35) 8

[**Физическая структура реляционной БД**](#_kg07arsih6s6) **9**

[Разработка таблиц БД и обеспечение целостности данных](#_wh0jzrpr7fx3) 9

[**Разработка запросов**](#_gg0vayesylwz) **10**

[**Транзакции**](#_3llbodhcnxyz) **11**

[Создание представлений](#_nrmdq9hqq18w) 12

[Создание индексов](#_q7b97muot556) 13

[**Разработка основных программных объектов базы данных**](#_wiwza5yn0v7i) **14**

[Разработка триггеров](#_6vys4m78ua3) 14

[**Заключение**](#_crre47fqe3c) **14**

# Введение

Базы данных — это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия.

Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять.

Каждая запись является набором именованных полей, или ячеек, которые могут хранить информацию. Однотипные поля разных записей образуют столбец таблицы.

Создав одну таблицу, вы уже получаете полноценную базу данных. Однако в реальной жизни структуры баз данных, а соответственно и способы их создания, намного сложнее.

Для решения проблем избыточной информации, в которой иногда трудно сориентироваться и выбрать нужные сведения, применяются автоматизированные базы данных (СУБД).

# 

# 

# Структурный анализ

## Анализ предметной области

Необходимо спроектировать базу данных для сценария зачисления абитуриентов в институт разных потоков, на группы разных факультетов. Из текста задания можно выделить 5 сущностей:

1. Абитуриент
2. Группа
3. Поток
4. Факультет
5. Институт

Для удобства, я введу словарь, который сможет дать определения вводимых мною слов, используемых в ходе работы.

Словарь:

Абитуриентом является человек, который хочет поступить на определенную специальность в институт. Для этого ему необходимо иметь при себе результаты ЕГЭ и подать необходимые документы на ту специальность, которую он выбрал.

Институтом является организация, предоставляющее образования для студентов, состоит из факультетов.

Студентом является человек, который был абитуриентом и прошёл отбор в институт. Он может получать услуги образования. Каждый студент прикрепляется к соответствующей группе по специальности, которую он выбрал.

Факультетом является подразделение института, на котором осуществляется подготовка [студентов](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%83%D0%B4%D0%B5%D0%BD%D1%82) по специальностям, которые определены в конкретном факультете.

Группой является объединение студентов, которые учатся на определенно одной специальности. Каждая группа имеет свой номер, идентифицирующий данную группу. Номер группы формируется по принципу: номер потока + номер кафедры + номер специальности.

Потоком является все абитуриентом, которые подали свои заявления в определённый год. Номером потока, является числовое значение, равное остатку от деления года поступления на 10

Экзамены, необходимые для поступления, являются критерием, который отбирает из всех абитуриентов потока только тех, кто имеет в сумме больше баллов, чем остальные. Ограничением для поступления служит целевой набор на специальность.

Целевым набором является количество студентов, которых может взять институт на определенную специальность.

Конец словаря.

## Анализ требований к информационной системе

Что конкретно должна уметь моя база данных ? Для ответа на этот вопрос я воспользуюсь методом проектирования программного обеспечения под названием сценарий использования, в котором постараюсь объяснить принцип работы моей системы и соответствующие требования к ней.

Имя сценария: зачисления в институт.

Цель: человек хочет начать учиться в институте на определенную специальность.

Акторы: человек, абитуриент, студент.

Предварительные условия: институт получается целевой набор, определяя количество новый студентов на специальности.

Активаторы: после окончания учебного года, начинается формироваться поток абитуриентов, которые хотят учиться.

Порядок событий:

1. Человек подаёт заявление в институт, получая статус абитуриент
2. Ожидание даты окончания приема документов
3. Сформирован список абитуриентов, которые теперь должны пройти конкурс на место.
4. Студенты,уже учащиеся в институте, переходят на следующий курс
5. Формирование списка новых студентов из списка абитуриентов путем отбора
6. Формирование новых групп, в которых будут учится новые студент

Таблица 1 - Таблица сущностей

|  |  |  |
| --- | --- | --- |
| Название сущности | Идентификатор сущности | Назначение сущности |
| Студент | Student | Содержит сведения о студентах |
| Группа | Groups | Содержит сведения о группах |
| Специальность | Specialty | Содержит сведения о специальностях |
| Факультет | Department | Содержит сведения о факультетах |
| Баллы | Points | Содержит сведения о результатов экзаменов |
| Предмет | Academic\_subject | Содержит сведения об академических дисциплинах |

Отношение “Студент”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Содержимое описание | Тип данных | Размерность | Область допустимых значений | Возможность значения  Null | Роль | Пример |
| student\_book\_number | Номер зачётной книжки | символьный | - | ‘0-9’,‘-’ | нет | PK | 011-1 |
| last\_name | Имя | символьный | 30 | ‘А-я’ | нет |  | Васильев |
| first\_name | Фамилия | символьный | 15 | ‘А-я’ | нет |  | Иван |
| patronymic | Отчество | символьный | 20 | ‘А-я’ | нет |  | Владимирович |
| date\_of\_dirth | Дата рождения | дата | 8 | 01.01.1980 – 31.12.2100 | нет |  | 2000-08-20 |
| group\_id | номер группы | символьный | - | ‘0-9’ | нет | FK | 011 |
| course | номер курса | целый | - | ‘0-6’ | нет |  | 1 |

Отношение “Группа”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Содержимое описание | Тип данных | Размерность | Область допустимых значений | Возможность значения  Null | Роль | Пример |
| group\_id | номер группы | символьный | - | ‘0-9’ | нет | PK | 011 |
| specialty\_id | Номер специальности | целый | - | ‘0-9’ | нет | FK | 1 |

Отношение “Специальность”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Содержимое описание | Тип данных | Размерность | Область допустимых значений | Возможность значения  Null | Роль | Пример |
| specialty\_id | Номер специальности | целый | - | ‘0-9’ | нет | PK | 1 |
| department\_id | Номер факультета | целый | - | ‘0-9’ | нет | FK | 1 |
| specialty\_name | Название специальности | символьный | - | ‘А-я’ | нет |  | Прикладная математика и информатика |
| target\_set | целевой набор | целый | - | ‘0-9’ | нет |  | 5 |

Отношение “Факультет”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Содержимое описание | Тип данных | Размерность | Область допустимых значений | Возможность значения  Null | Роль | Пример |
| department\_id | Номер факультета | целый | - | ‘0-9’ | нет | PK | 1 |
| department\_name | название факультета | символьный | - | ‘A-Я’ | нет |  | ИЭИС |

Отношение “Баллы”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Содержимое описание | Тип данных | Размерность | Область допустимых значений | Возможность значения  Null | Роль | Пример |
| applicant\_id | Номер абитуриента | целый | - | ‘0-9’ | нет | FK | 1 |
| Academic\_subject\_id | Номер предмета | целый | - | ‘0-9’ | нет | FK | 2 |
| Point\_count | баллы | целый | - | ‘0-9’ | нет |  | 60 |

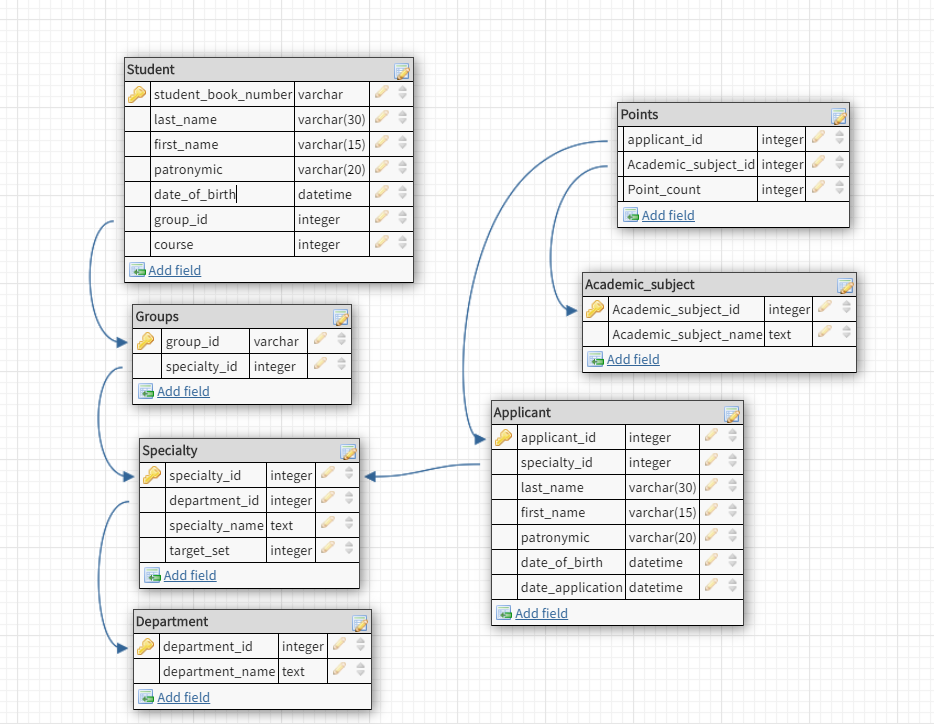
Отношение “Предмет”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Содержимое описание | Тип данных | Размерность | Область допустимых значений | Возможность значения | Роль | Пример |
| Academic\_subject\_id | Номер предмета | целый | - | ‘0-9’ | нет | PK | 1 |
| Academic\_subject\_name | Название предмета | текст | - | ‘A-Я’ | нет |  | Русский язык |

Отношение “Абитуриент”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Содержимое описание | Тип данных | Размерность | Область допустимых значений | Возможность значения | Роль | Пример |
| applicant\_id | Номер абитуриента | целый | - | ‘0-9’ | нет | PK | 1 |
| specialty\_id | номер специальности | целый | - | ‘0-9’ | нет | FK | 2 |
| last\_name | Фамилия | символьный | 30 | ‘A-Я’ | нет |  | Бочкарёв |
| first\_name | Имя | символьный | 15 | ‘A-Я’ | нет |  | Борис |
| patronymic | Отчество | символьный | 20 | ‘A-Я’ | нет |  | Анатольевич |
| date\_of\_dirth | Дата рождения | дата | - | 01.01.1980 – 31.12.2100 | нет |  | 2000-08-20 |
| date\_application | Дата подачи документов | дата | - | 01.01.1980 – 31.12.2100 | нет |  | 2020-08-20 |

## Инфологическая модель данных.

В этом разделе я предоставлю ER - диаграмму, которая содержит информацию о сущностях системы и способах их взаимодействия, включает идентификацию объектов, важных для предметной области, свойств этих объектов и их отношений другими объектами.   


# Физическая структура реляционной БД

## Разработка таблиц БД и обеспечение целостности данных

CREATE TABLE Student

(

student\_book\_number varchar PRIMARY KEY,

last\_name varchar(30) NOT NULL,

first\_name varchar(15) NOT NULL,

patronymic varchar(20) NOT NULL,

date\_of\_dirth data(8) NOT NULL,

group\_id varchar REFERENCES Groups(group\_id),

course int NOT NULL

);

CREATE TABLE Groups

(

group\_id varchar PRIMARY KEY,

specialty\_id int REFERENCES Specialty(specialty\_id)

);

CREATE TABLE Specialty

(

specialty\_id INTEGER PRIMARY KEY AUTOINCREMENT,

department\_id INTEGER REFERENCES Department(department\_id),

specialty\_name text NOT NULL,

target\_set int NOT NULL

);

CREATE TABLE Department

(

department\_id INTEGER PRIMARY KEY AUTOINCREMENT,

department\_name text NOT NULL

);

CREATE TABLE Points

(

applicant\_id INTEGER REFERENCES Applicant(applicant\_id),

Academic\_subject\_id REFERENCES Academic\_subject(Academic\_subject\_id),

Point\_count int NOT NULL

CHECK ((Academic\_subject\_id = 1 AND Point\_count >= 36) OR

(Academic\_subject\_id = 2 AND Point\_count >= 27) OR

(Academic\_subject\_id = 3 AND Point\_count >= 36) OR

(Academic\_subject\_id = 4 AND Point\_count >= 36) OR

(Academic\_subject\_id = 5 AND Point\_count >= 36) OR

(Academic\_subject\_id = 6 AND Point\_count >= 42))

);

CREATE TABLE Academic\_subject

(

Academic\_subject\_id INTEGER PRIMARY KEY AUTOINCREMENT,

Academic\_subject\_name text

);

CREATE TABLE Applicant

(

applicant\_id INTEGER PRIMARY KEY AUTOINCREMENT,

specialty\_id INTEGER REFERENCES Specialty(specialty\_id),

last\_name varchar(30) NOT NULL,

first\_name varchar(15) NOT NULL,

patronymic varchar(20) NOT NULL,

date\_of\_dirth data NOT NULL,

date\_application data NOT NULL

);

## Разработка запросов

1. Запрос выводит всех студентов определенной группы

SELECT \*

FROM Student

WHERE group\_id='011';

1. Запрос выводит всех студентов определенной специальности

SELECT \*

FROM Students\_info

WHERE Специальность = 'Маркетинг';

1. Запрос выводит прирост абитуриентов по специальностям и годам, ответ выдаёт в процентах.

SELECT DISTINCT Специальность,

Count,

COALESCE( round(100 \* (Count - LAG(Count) OVER(partition by Специальность order by strftime('%Y',Дата\_заявления))) / LAG(Count) OVER(partition by Специальность order by strftime('%Y',Дата\_заявления)) ,2)||'%', 'Firs year, not increase') AS increase,

strftime('%Y',Дата\_заявления)

FROM (

SELECT DISTINCT

Специальность,

Дата\_заявления,

COUNT() OVER(PARTITION BY Специальность ORDER BY Специальность, strftime('%Y',Дата\_заявления)) AS Count

FROM Applicants\_info)

ORDER BY Специальность;

1. Запрос выводит всех абитуриентов, которые не смогли поступить

SELECT Фамилия,

Имя,

Отчество,

Дата\_рождения,

Специальность

FROM Applicants\_info

EXCEPT

SELECT Фамилия,

Имя,

Отчество,

Дата\_рождения,

Специальность

FROM Students\_info;

## Транзакции

Транзакция - это логическая группа операций, которая может быть выполнена только целиком.Я использовал транзакцию в зачисление студентов.

BEGIN TRANSACTION;

UPDATE Student

SET Course = Course +1;

INSERT INTO Student

SELECT CAST(CAST(stream AS INT) as TEXT) || CAST(department\_id AS TEXT) || CAST(specialty\_id AS TEXT) || '-' || rank,

last\_name,

first\_name,

patronymic,

date\_of\_dirth,

CAST(CAST(stream AS INT) as TEXT) || CAST(department\_id AS TEXT) || CAST(specialty\_id AS TEXT) AS group\_id,

1

FROM(

SELECT applicant\_id,

last\_name,

first\_name,

specialty\_id,

patronymic,

date\_of\_dirth,

date\_applic - 10 \* round((date\_applic/10)-0.5,0) AS stream,

Sum\_total,

target\_set,

ROW\_NUMBER() OVER(PARTITION BY specialty\_id ORDER BY Sum\_total DESC) AS rank,

department\_id

FROM (

SELECT DISTINCT applicant\_id,

last\_name,

first\_name,

patronymic,

specialty\_id,

date\_of\_dirth,

CAST(strftime('%Y',date\_application) as double) AS date\_applic,

SUM(Point\_count) OVER(PARTITION BY applicant\_id) AS Sum\_total,

target\_set,

department\_id

FROM Applicant

JOIN Points USING(applicant\_id)

JOIN Specialty USING(specialty\_id)

JOIN Department USING(department\_id)

WHERE date\_applic = (SELECT MAX(strftime('%Y',date\_application)) FROM Applicant)

ORDER BY specialty\_id, SUM(Point\_count) OVER(PARTITION BY applicant\_id) desc,applicant\_id ASC))

WHERE rank <= target\_set;

ROLLBACK;

COMMIT;

## Создание представлений

Представления - сохраненный запрос в виде объекта базы данных (виртуальная таблица). Обращение к представлениям такое же, как и к обычным таблицам.  
Я использовал для удобства два представления с выводами полной информации об абитуриентах и студентах.

CREATE VIEW Applicants\_info AS

SELECT DISTINCT applicant\_id AS ID\_Абитуриента,

last\_name AS Фамилия,

first\_name AS Имя,

patronymic AS Отчество,

date\_of\_dirth AS Дата\_рождения,

date\_application AS Дата\_заявления,

specialty\_id AS id\_Специальности,

specialty\_name AS Специальность,

GROUP\_CONCAT(Academic\_subject\_name|| '-' || Point\_count) OVER(PARTITION BY applicant\_id) AS Баллы\_ЕГЭ

FROM Applicant

JOIN Specialty USING(specialty\_id)

JOIN Points USING(applicant\_id)

JOIN Academic\_subject USING(Academic\_subject\_id);

CREATE VIEW Students\_info AS

SELECT student\_book\_number AS Зачётка,

group\_id AS Группа,

last\_name AS Фамилия,

first\_name AS Имя,

patronymic AS Отчество,

date\_of\_dirth AS Дата\_рождения,

course AS Курс,

specialty\_name AS Специальность

FROM Student

JOIN Groups USING(group\_id)

JOIN Specialty USING(specialty\_id);

## Создание индексов

Индексы - это структура данных, ускоряющая выборку данных из таблиц, за счёт дополнительных операций записи и пространства надиске, используемых для хранения структуры данных и поддержания её в актуальном состоянии. Я использовал 4 индекса для таблиц: Абитуриент, Студент, Баллы, Группы.

CREATE INDEX IND\_Applicant\_applicant\_id

ON Applicant(applicant\_id);

CREATE UNIQUE INDEX IND\_Student\_student\_book\_number

ON Student(student\_book\_number);

CREATE INDEX IND\_Point\_applicant\_id

ON Points(applicant\_id);

CREATE INDEX IND\_Groups\_group\_id

ON Groups(group\_id);

# Разработка основных программных объектов базы данных

## Разработка триггеров

Триггер - это объект, который реагирует на действия и назначает действия на те или иные события. Я использовал триггер на поддержание целостности данных в таблице студент, где до добавления нового студента срабатывает триггер и сначала создается группа для этого студента. Второй триггер на удаление абитуриента, в котором до удаления, сначала удаляются его данные по экзаменам в таблице баллы.

CREATE TRIGGER create\_new\_group

BEFORE INSERT ON Student

WHEN new.group\_id NOT IN(SELECT group\_id FROM Groups) and new.course = 1

BEGIN

INSERT INTO Groups(group\_id) VALUES(new.group\_id);

END;

CREATE TRIGGER delete\_applicant

BEFORE DELETE ON Applicant

BEGIN

DELETE FROM Points

WHERE Points.applicant\_id = old.applicant\_id;

END;

# 

# 

# Заключение

Результатом выполнения курсовой работы стало разработанное приложение баз данных, отвечающее всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации.