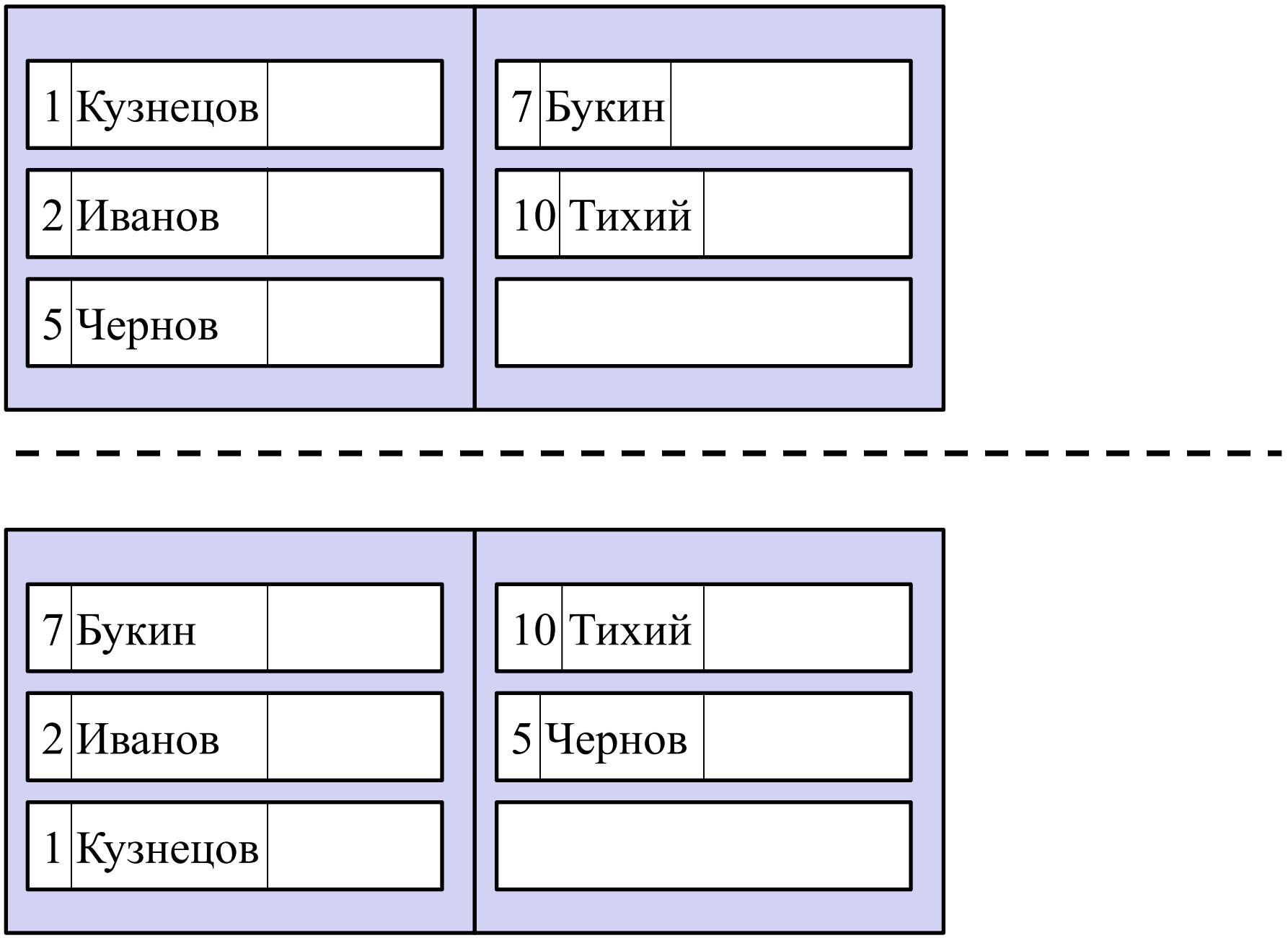
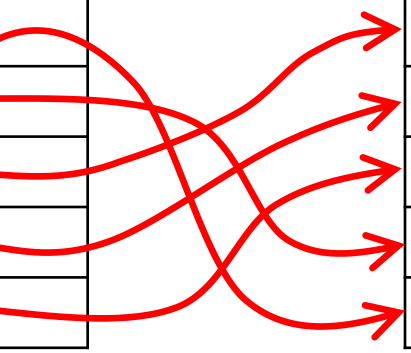
Индексы


1 Кузнецов
2 Иванов
5 Чернов
7 Букин
10 Тихий
7 Букин
2 Иванов
1 Кузнецов
10 Тихий
5 Чернов

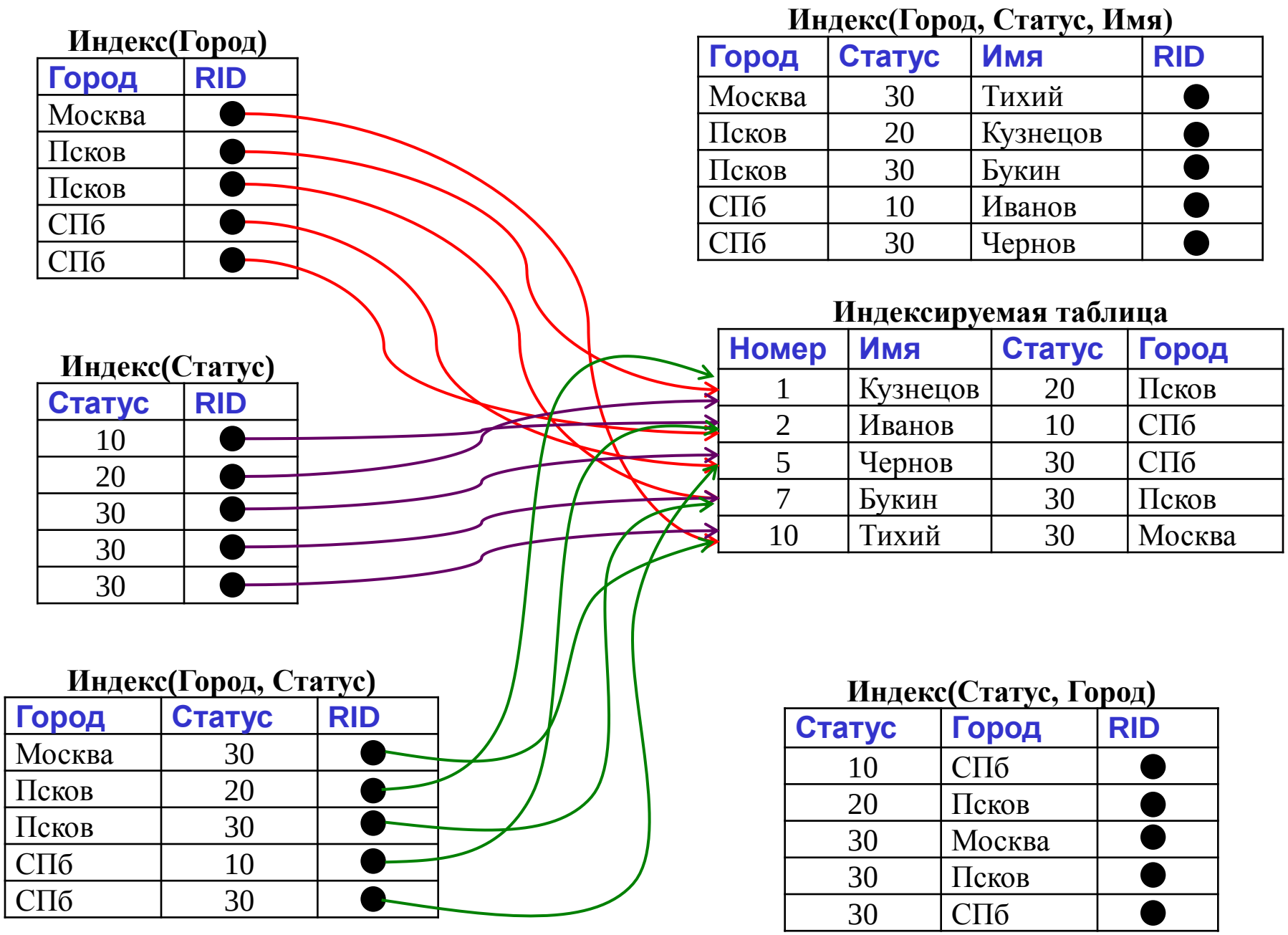
Индекс(Город)

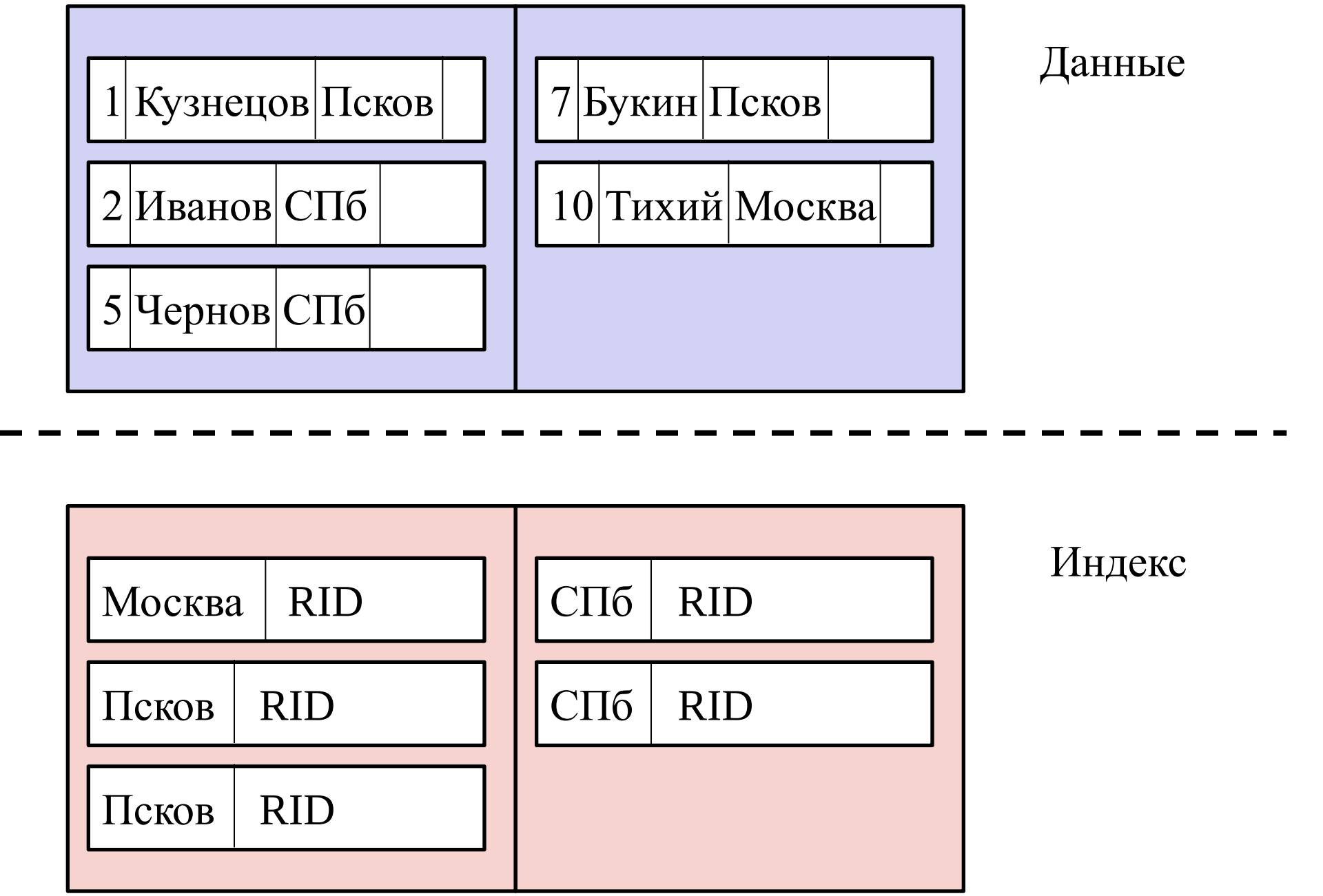
Город	RID
Москва	●
Псков	●
Псков	●
СПб	●
СПб	●

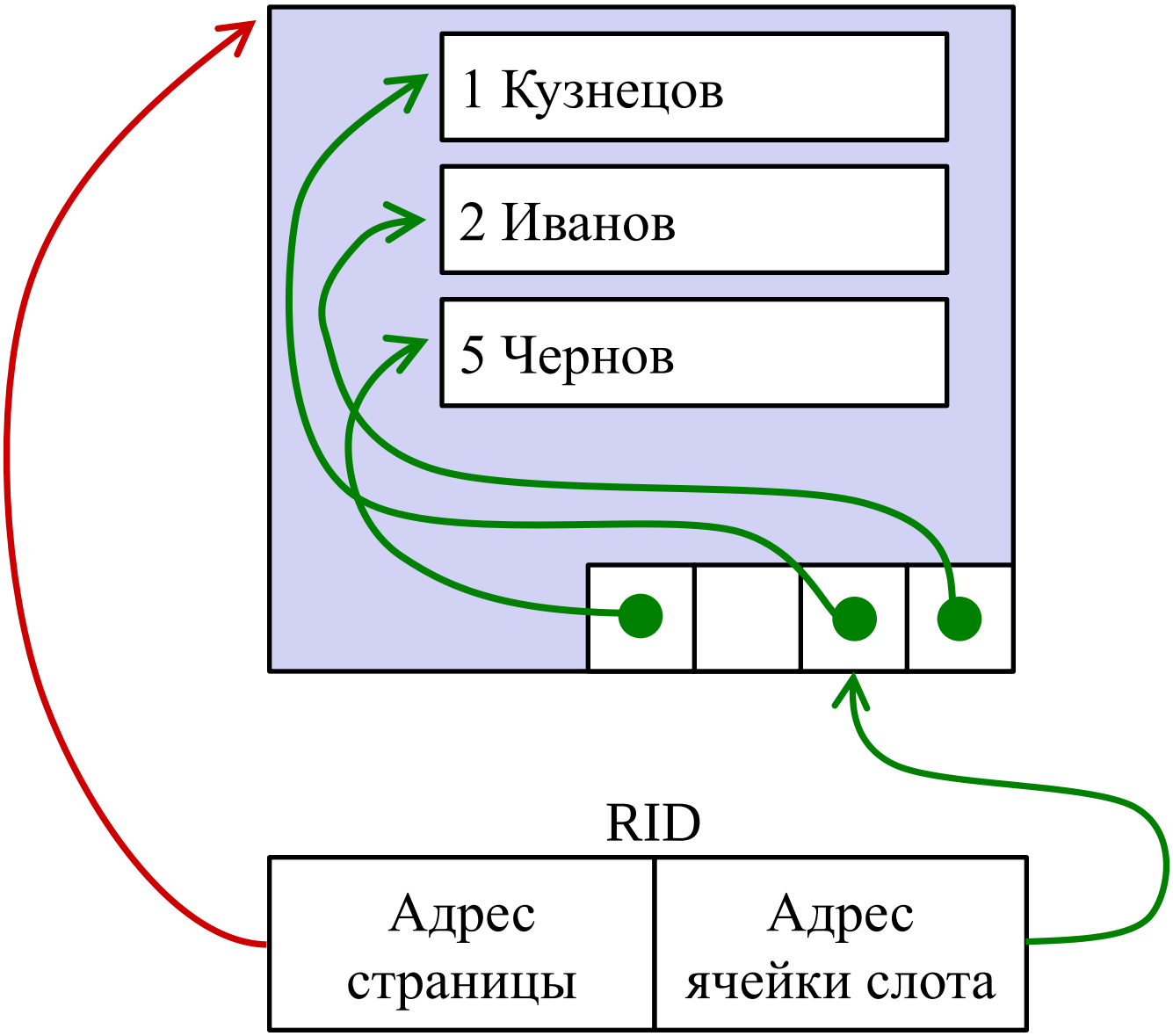


Индексируемая таблица

Номер	Имя	Статус	Город
1	Кузнецов	20	Псков
2	Иванов	10	СПб
5	Чернов	30	СПб
7	Букин	30	Псков
10	Тихий	30	Москва


Индекс(Город)
Город
RID
Москва
Псков
Псков
СПб
СПб
Индекс(Статус)
Статус
RID
10
20
30
30
30
Индекс(Город, Статус)
Город
Статус
RID
Москва
30
Псков
20
Псков
30
СПб
10
СПб
30
Индекс(Город, Статус, Имя)
Город
Статус
Имя
RID
Москва
30
Тихий
Псков
20
Кузнецов
Псков
30
Букин
СПб
10
Иванов
СПб
30
Чернов
Индексируемая таблица
Номер
Имя
Статус
Город
1
Кузнецов
20
Псков
2
Иванов
10
СПб
5
Чернов
30
СПб
7
Букин
30
Псков
10
Тихий
30
Москва
Индекс(Статус, Город)
Статус
Город
RID
10
СПб
20
Псков
30
Москва
30
Псков
30
СПб


1 Кузнецов Псков
2 Иванов СПб
5 Чернов СПб
7 Букин Псков
10 Тихий Москва
Данные
Москва RID
Псков RID
Псков RID
СПб RID
СПб RID
Индекс


1 Кузнецов
2 Иванов
5 Чернов
RID
Адрес страницы
Адрес ячейки слота

Индекс(Город)

Город	RID
Москва	●
Псков	●
Псков	●
СПб	●
СПб	●

Индексируемая таблица

Номер	Имя	Статус	Город
1	Кузнецов	20	Псков
2	Иванов	10	СПб
5	Чернов	30	СПб
7	Букин	30	Псков
10	Тихий	30	Москва

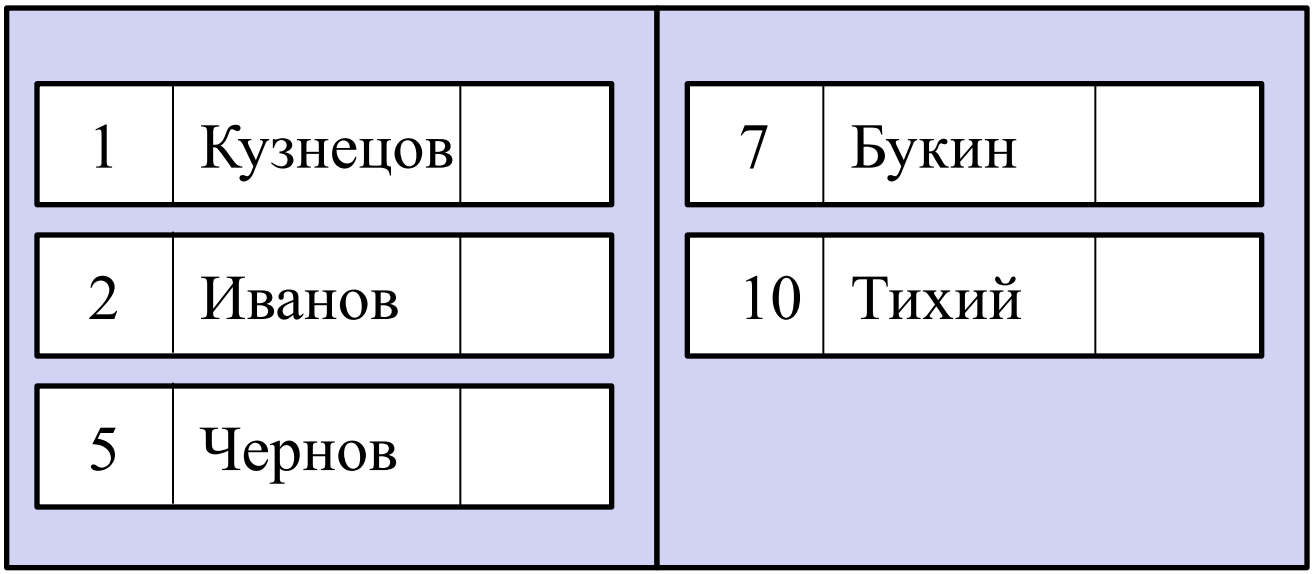

Индекс(Город)
Город
RID
Москва
Псков
Псков
СПб
СПб
Ростов
Индексируемая таблица
Номер
Имя
Статус
Город
1
Кузнецов
20
Псков
2
Иванов
10
СПб
5
Чернов
30
СПб
7
Букин
30
Псков
10
Тихий
30
Москва
8
Кашин
20
Ростов

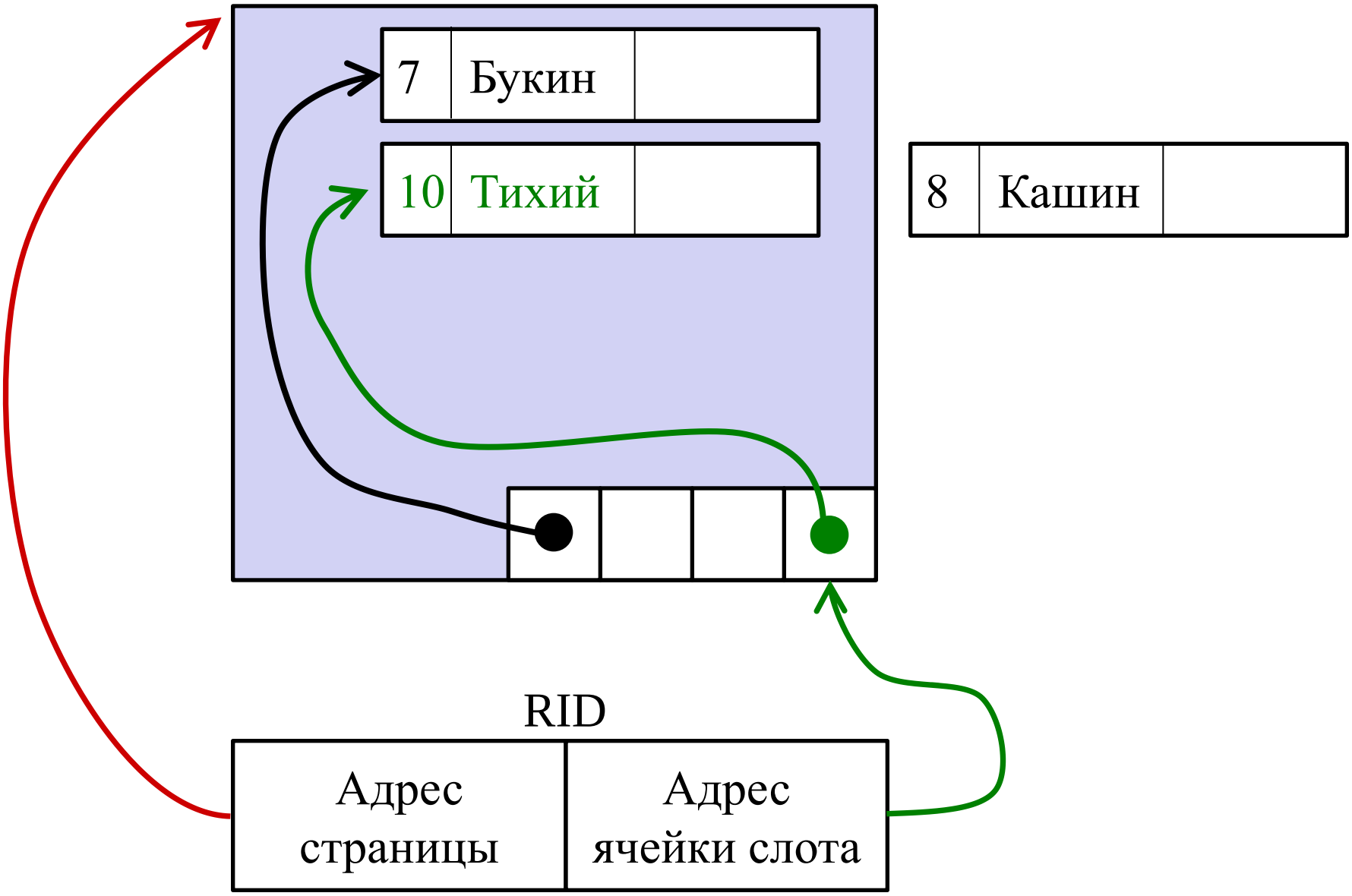
Индекс(Город)

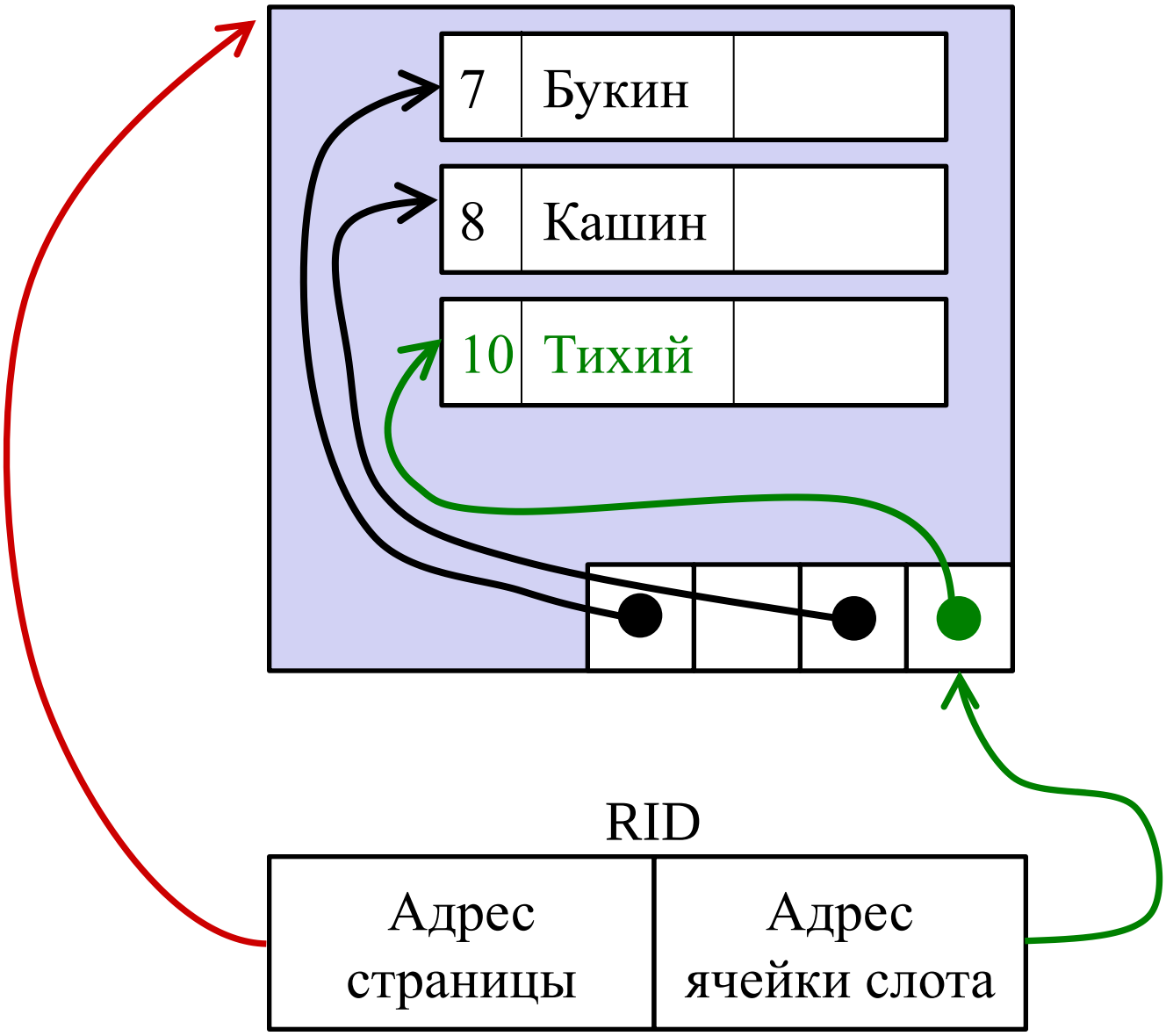
Город	RID
Москва	●
Псков	●
Псков	●
Ростов	●
СПб	●
СПб	●

Индексируемая таблица

Номер	Имя	Статус	Город
1	Кузнецов	20	Псков
2	Иванов	10	СПб
5	Чернов	30	СПб
7	Букин	30	Псков
8	**Кашин**	**20**	**Ростов**
10	Тихий	30	Москва


1 Кузнецов
2 Иванов
5 Чернов
7 Букин
10 Тихий
8 Кашин


7
Букин
10
Тихий
8
Кашин
RID
Адрес страницы
Адрес ячейки слота

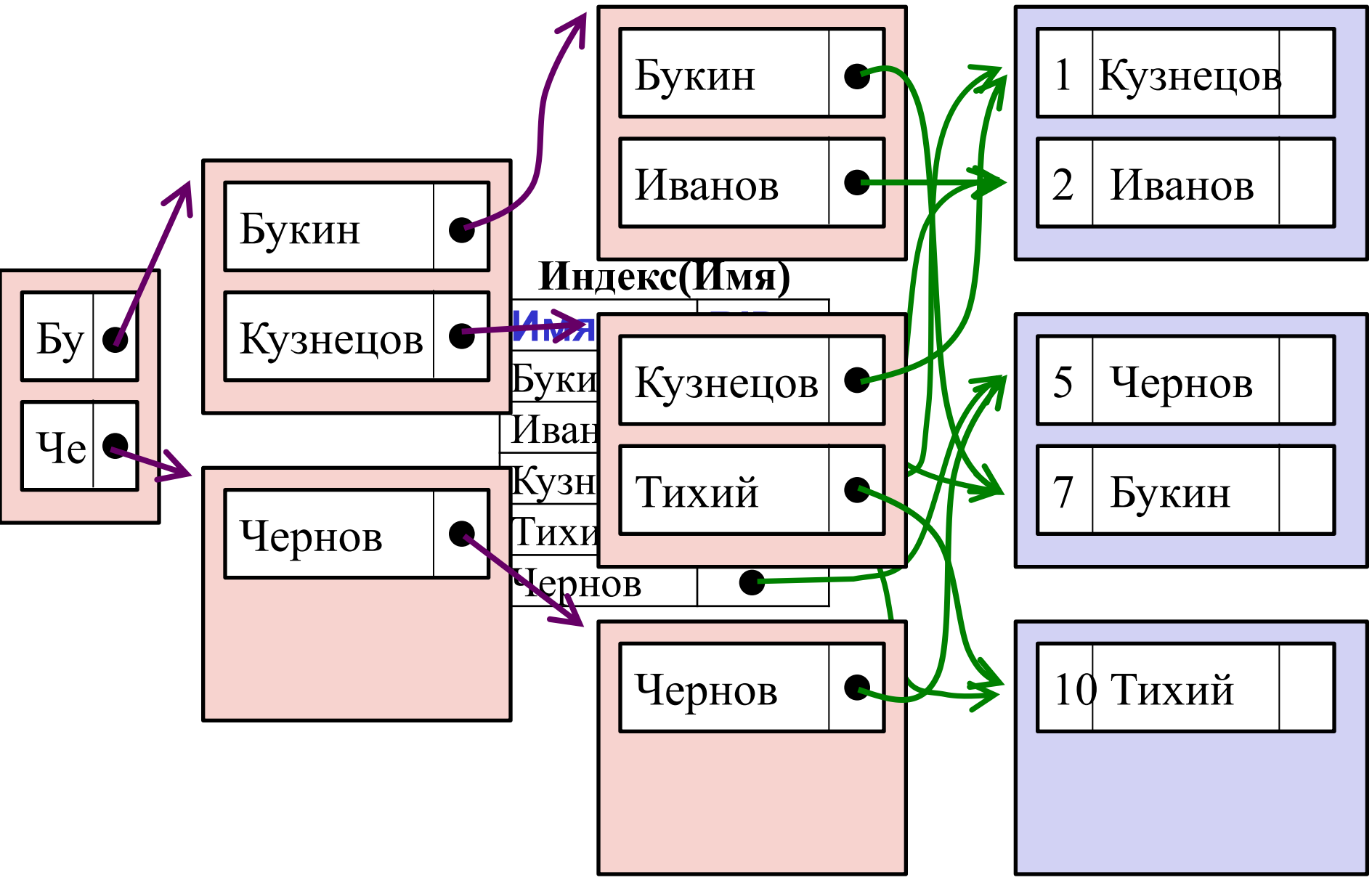

7
Букин
8
Кашин
10
Тихий
RID
Адрес страницы
Адрес ячейки слота

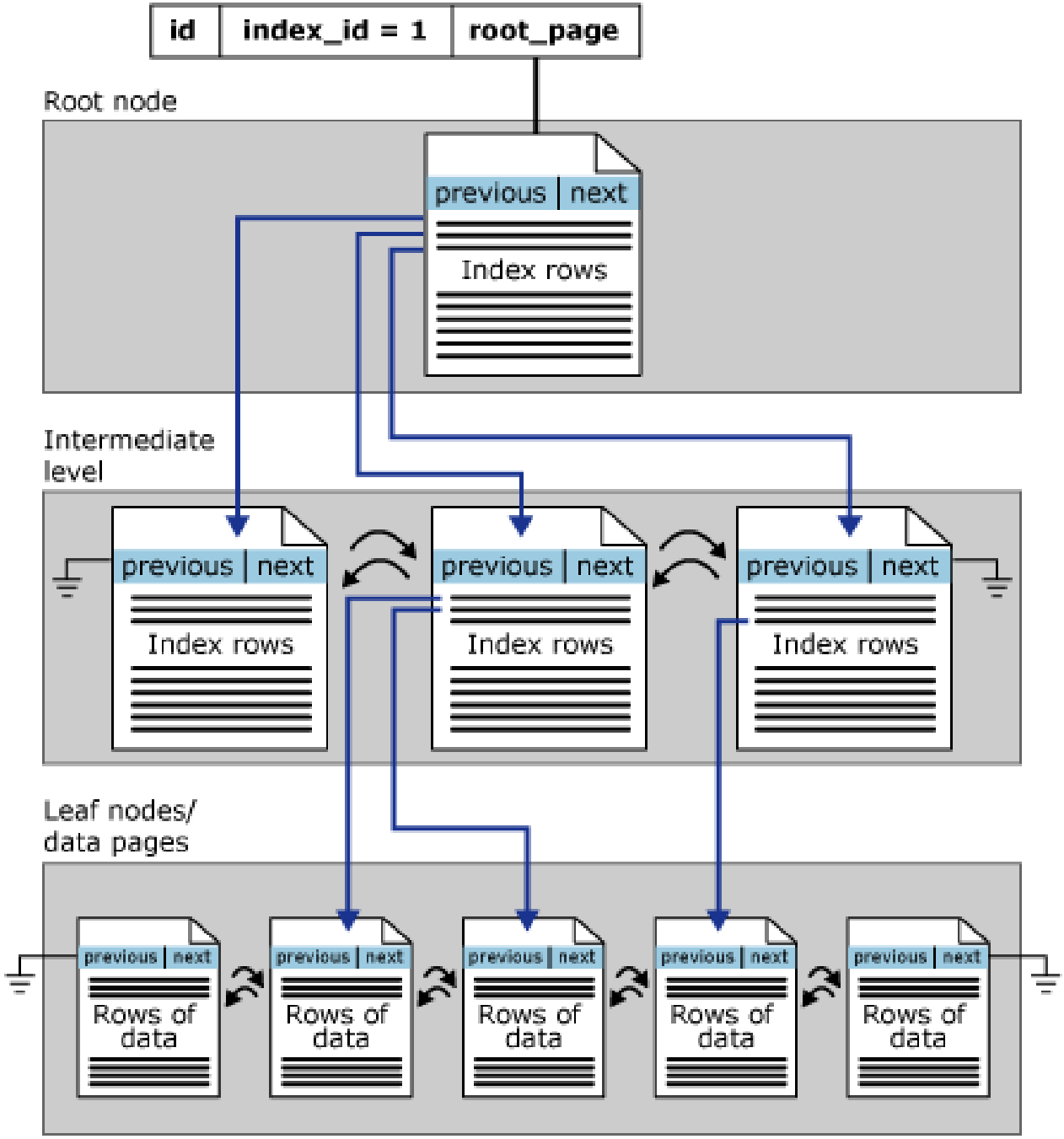
Индекс(Город)

Город	RID
Москва	●
Псков	●
Псков	●
Ростов	●
СПб	●
СПб	●

Индексируемая таблица

Номер	Имя	Статус	Город
1	Кузнецов	20	Псков
2	Иванов	10	СПб
5	Чернов	30	СПб
7	Букин	30	Псков
8	**Кашин**	**20**	**Ростов**
10	**Тихий**	**30**	**Москва**


Бу
Че
Букин
Кузнецов
Чернов
Индекс(Имя)
Имя
Буки
Иван
Кузн
Тихи
Чернов
Букин
Иванов
Кузнецов
Тихий
Чернов
1 Кузнецов
2 Иванов
5 Чернов
7 Букин
10 Тихий


id
index_id = 1
root_page
Root node
previous
next
Index rows
Intermediate level
previous
next
Index rows
previous
next
Index rows
previous
next
Index rows
Leaf nodes/ data pages
previous
next
Rows of data
previous
next
Rows of data
previous
next
Rows of data
previous
next
Rows of data
previous
next
Rows of data

Описание команды

CREATE [unique] [{asc | desc}] **INDEX** <*имя*> **ON** <*таблица*> (<*столбец*>,…)

ALTER INDEX <*имя*> {active | inactive}

DROP INDEX <*имя*>

описание команды mySQL

CREATE TABLE <*таблица*> (<*столбец*>, **INDEX** <*имя*> (<*столбец*>), …)

ALTER TABLE <*таблица*> **ADD INDEX** <*имя*> (<*столбец*>)

пример mySQL

```
CREATE  TABLE  tblStudent
( txtName char(200),
INDEX  indStudentName (txtName(40)),
…)
```

описание команды
MS SQL Server

CREATE [unique] [{clustered | nonclustered}] **INDEX** <*имя*> **ON** <*таблица*> (<*столбец*> [{asc | desc}] ,…)

пример MS SQL Server

CREATE nonclustered **INDEX** indStudentName **ON** tblStudent (txtLastName, txtFirstName, txtSecondName)

Базы данных

Щеголева Л. В.

ПетрГУ

Аномалии схемы отношения

Запас(Товар, Поставщик, Адрес, Цена, Склад, Объем)

Аномалии схемы отношения

Запас(Товар, Поставщик, Адрес, Цена, Склад, Объем)

Запас

Товар	Поставщик	Адрес	Цена	Склад	Объем
Стол	Столплит	Лесной, 51	3000 руб.	2	50
Стол	Форт	Ровио, 3	3120 руб.	3	37
Шкаф	Форт	Ровио, 3	5000 руб.	3	37
Стул	Столплит	Лесной, 51	1200 руб.	2	50

Ограничения схемы отношений

Запас

Товар	Поставщик	Адрес	Цена	Склад	Объем
Стол	Столплит	Лесной, 51	3000 руб.	2	50
Стол	Форт	Ровио, 3	3120 руб.	3	37
Шкаф	Форт	Ровио, 3	5000 руб.	3	37
Стул	Столплит	Лесной, 51	1200 руб.	2	50
Шкаф	Столплит	Ровио, 3	1200 руб.	2	50

Ограничения схемы отношений

Запас

Товар	Поставщик	Адрес	Цена	Склад	Объем
Стол	Столплит	Лесной, 51	3000 руб.	2	50
Стол	Форт	Ровио, 3	3120 руб.	3	37
Шкаф	Форт	Ровио, 3	5000 руб.	3	37
Стул	Столплит	Лесной, 51	1200 руб.	2	50
Шкаф	Столплит	Лесной, 51	1200 руб.	2	100

Ограничения схемы отношений

Запас

Товар	Поставщик	Адрес	Цена	Склад	Объем
Стол	Столплит	Лесной, 51	3000 руб.	2	50
Стол	Форт	Ровио, 3	3120 руб.	3	37
Шкаф	Форт	Ровио, 3	5000 руб.	3	37
Стул	Столплит	Лесной, 51	1200 руб.	2	50
Стол	Столплит	Лесной, 51	3200 руб.	2	50

Ограничения предметной области

- Каждый поставщик имеет только один адрес
- Каждый поставщик поставляет товар по определенной цене
- Товары одного наименования, поставленные одним поставщиком, хранятся на одном складе
- Каждый склад имеет свой объем

Функциональная зависимость

$$r(R), \quad X, Y \in R$$

Функциональная зависимость

$$r(R), \quad X, Y \in R$$

$$X \to Y$$

Функциональная зависимость

$$r(R), \quad X, Y \in R$$

$$X \to Y$$

$$\pi_Y \left(\sigma_{X=x}(r)\right)$$

Ограничения предметной области

- Каждый поставщик имеет только один адрес
- Каждый поставщик поставляет товар по определенной цене
- Товары одного наименования, поставленные одним поставщиком, хранятся на одном складе
- Каждый склад имеет свой объем

Ограничения предметной области

- Каждый поставщик имеет только один адрес

$$\text{Поставщик} \rightarrow \text{Адрес}$$

- Каждый поставщик поставляет товар по определенной цене
- Товары одного наименования, поставленные одним поставщиком, хранятся на одном складе
- Каждый склад имеет свой объем

Ограничения предметной области

- Каждый поставщик имеет только один адрес

$$\text{Поставщик} \rightarrow \text{Адрес}$$

- Каждый поставщик поставляет товар по определенной цене

$$\{\text{Товар}, \text{Поставщик}\} \rightarrow \text{Цена}$$

- Товары одного наименования, поставленные одним поставщиком, хранятся на одном складе
- Каждый склад имеет свой объем

Ограничения предметной области

- Каждый поставщик имеет только один адрес

$$\text{Поставщик} \rightarrow \text{Адрес}$$

- Каждый поставщик поставляет товар по определенной цене

$$\{\text{Товар}, \text{Поставщик}\} \rightarrow \text{Цена}$$

- Товары одного наименования, поставленные одним поставщиком, хранятся на одном складе

$$\{\text{Товар}, \text{Поставщик}\} \rightarrow \text{Склад}$$

- Каждый склад имеет свой объем

Ограничения предметной области

- Каждый поставщик имеет только один адрес

$$\text{Поставщик} \rightarrow \text{Адрес}$$

- Каждый поставщик поставляет товар по определенной цене

$$\{\text{Товар}, \text{Поставщик}\} \rightarrow \text{Цена}$$

- Товары одного наименования, поставленные одним поставщиком, хранятся на одном складе

$$\{\text{Товар}, \text{Поставщик}\} \rightarrow \text{Склад}$$

- Каждый склад имеет свой объем

$$\text{Склад} \rightarrow \text{Объем}$$

Аксиомы вывода

Аксиомы вывода

1. Рефлексивность $X \to X$

Аксиомы вывода

1. Рефлексивность $X \to X$
2. Пополнение $X \to Y \quad \Rightarrow \quad XZ \to Y$

Аксиомы вывода

1. Рефлексивность $X \to X$
2. Пополнение $X \to Y \quad \Rightarrow \quad XZ \to Y$
3. Аддитивность $X \to Y, X \to Z \quad \Rightarrow \quad X \to YZ$

Аксиомы вывода

1. Рефлексивность $X \to X$
2. Пополнение $X \to Y \Rightarrow XZ \to Y$
3. Аддитивность $X \to Y, X \to Z \Rightarrow X \to YZ$
4. Проективность $X \to YZ \Rightarrow X \to Y$

Аксиомы вывода

1. Рефлексивность	$X \to X$		
2. Пополнение	$X \to Y$	$\Rightarrow$	$XZ \to Y$
3. Аддитивность	$X \to Y, X \to Z$	$\Rightarrow$	$X \to YZ$
4. Проективность	$X \to YZ$	$\Rightarrow$	$X \to Y$
5. Транзитивность	$X \to Y, Y \to Z$	$\Rightarrow$	$X \to Z$

Аксиомы вывода

1. Рефлексивность	$X \to X$		
2. Пополнение	$X \to Y$	$\Rightarrow$	$XZ \to Y$
3. Аддитивность	$X \to Y, X \to Z$	$\Rightarrow$	$X \to YZ$
4. Проективность	$X \to YZ$	$\Rightarrow$	$X \to Y$
5. Транзитивность	$X \to Y, Y \to Z$	$\Rightarrow$	$X \to Z$
6. Псевдотранзитивность	$X \to Y, YZ \to W$	$\Rightarrow$	$XZ \to W$

Первая нормальная форма (1НФ)

Вторая нормальная форма (2НФ)

$$r(\underline{A}, \underline{B}, C, D, E)$$

$$AB \to C$$

$$AB \to D$$

$$AB \to E$$

$$B \to D$$

Вторая нормальная форма (2НФ)

$$r(\underline{A}, \underline{B}, C, D, E)$$

$$AB \to C$$
$$AB \to D$$
$$AB \to E$$
$$B \to D$$

$$\rho : \begin{cases} r_1(\underline{B}, D) \\ r_2(\underline{A}, \underline{B}, C, E) \end{cases}$$

Вторая нормальная форма

Запас(Товар, Поставщик, Адрес, Цена, Склад, Объем)

$\{\text{Товар}, \text{Поставщик}\} \to \text{Цена}$
$\{\text{Товар}, \text{Поставщик}\} \to \text{Склад}$
$\text{Склад} \to \text{Объем}$
$\text{Поставщик} \to \text{Адрес}$

Вторая нормальная форма

Запас(Товар, Поставщик, Адрес, Цена, Склад, Объем)

$\{\text{Товар}, \text{Поставщик}\} \to \text{Цена}$
$\{\text{Товар}, \text{Поставщик}\} \to \text{Склад}$
$\text{Склад} \to \text{Объем}$
$\text{Поставщик} \to \text{Адрес}$

Запас1(Поставщик, Адрес)
Запас2(Товар, Поставщик, Цена, Склад, Объем)

Третья нормальная форма

$$r(\underline{A}, B, C, D, E)$$

$$A \to B$$

$$A \to C$$

$$A \to D$$

$$A \to E$$

$$D \to B$$

Третья нормальная форма

$$r(\underline{A}, B, C, D, E)$$

$$A \to B$$
$$A \to C$$
$$A \to D$$
$$A \to E$$
$$D \to B$$

$$\rho : \begin{cases} r_1(\underline{D}, B) \\ r_2(\underline{A}, C, D, E) \end{cases}$$

Третья нормальная форма

Запас1(Поставщик, Адрес)
Запас2(Товар, Поставщик, Цена, Склад, Объем)

$\{\text{Товар}, \text{Поставщик}\} \to \text{Цена}$
$\{\text{Товар}, \text{Поставщик}\} \to \text{Склад}$
$\text{Склад} \to \text{Объем}$
$\text{Поставщик} \to \text{Адрес}$

Третья нормальная форма

Запас1(Поставщик, Адрес)
Запас2(Товар, Поставщик, Цена, Склад, Объем)

$\{\text{Товар}, \text{Поставщик}\} \to \text{Цена}$
$\{\text{Товар}, \text{Поставщик}\} \to \text{Склад}$
$\text{Склад} \to \text{Объем}$
$\text{Поставщик} \to \text{Адрес}$

Запас3(Склад, Объем)
Запас4(Товар, Поставщик, Цена, Склад)

Третья нормальная форма

Запас1(Поставщик, Адрес)
Запас3(Склад, Объем)
Запас4(Товар, Поставщик, Цена, Склад)

$\{\text{Товар}, \text{Поставщик}\} \rightarrow \text{Цена}$
$\{\text{Товар}, \text{Поставщик}\} \rightarrow \text{Склад}$
$\text{Склад} \rightarrow \text{Объем}$
$\text{Поставщик} \rightarrow \text{Адрес}$

Пример

Лекции

Студент	Предмет	Преподаватель
Иванов	Алгебра	проф. Белый
Иванов	Физика	проф. Яров
Петров	Алгебра	проф. Белый
Петров	Физика	проф. Серов

Пример

Лекции

Студент	Предмет	Преподаватель
Иванов	Алгебра	проф. Белый
Иванов	Физика	проф. Яров
Петров	Алгебра	проф. Белый
Петров	Физика	проф. Серов

$\{\text{Студент}, \text{Предмет}\} \rightarrow \text{Преподаватель}$
$\text{Преподаватель} \rightarrow \text{Предмет}$

Пример

Лекции

Студент	Предмет	Преподаватель
Иванов	Алгебра	проф. Белый
Иванов	Физика	проф. Яров
Петров	Алгебра	проф. Белый
Петров	Физика	проф. Серов

$\{\text{Студент}, \text{Предмет}\} \rightarrow \text{Преподаватель}$
$\text{Преподаватель} \rightarrow \text{Предмет}$

Ключ: $\{\text{Студент}, \text{Предмет}\}$

Нормальная форма Бойса-Кодда (НФБК)

$$r(\underline{A}, \underline{B}, C, D, E)$$

$$AB \to C$$

$$AB \to D$$

$$AB \to E$$

$$D \to B$$

Нормальная форма Бойса-Кодда (НФБК)

$$r(\underline{A}, \underline{B}, C, D, E)$$

$$AB \to C$$

$$AB \to D$$

$$AB \to E$$

$$D \to B$$

$$\rho : \begin{cases} r_1(\underline{D}, B) \\ r_2(\underline{A}, \underline{D}, C, E) \end{cases}$$

Нормальная форма Бойса-Кодда (НФБК)

Лекции

Студент	Предмет	Преподаватель
Иванов	Алгебра	проф. Белый
Иванов	Физика	проф. Яров
Петров	Алгебра	проф. Белый
Петров	Физика	проф. Серов

Лекции1

Студент	Преподаватель
Иванов	проф. Белый
Иванов	проф. Яров
Петров	проф. Белый
Петров	проф. Серов

Лекции2

Преподаватель	Предмет
проф. Белый	Алгебра
проф. Яров	Физика
проф. Серов	Физика

Пример

Преподаватель

ФИО	Имя	Предмет	Должность

Пример

Преподаватель

ФИО	Имя	Предмет	Должность
Яров	Ольга	Алгебра	Профессор
Яров	Иван	Геометрия	Профессор
Яров	Ольга	Геометрия	Профессор
Яров	Иван	Алгебра	Профессор

Пример

Преподаватель

ФИО	Имя	Предмет	Должность
Яров	Ольга	Алгебра	Профессор
Яров	Иван	Геометрия	Профессор
Яров	Ольга	Геометрия	Профессор
Яров	Иван	Алгебра	Профессор
Яров	Павел	Алгебра	Профессор
Яров	Павел	Геометрия	Профессор

Пример

Преподаватель

ФИО	Имя	Предмет	Должность
Яров	Ольга	Алгебра	Профессор
Яров	Иван	Геометрия	Профессор
Яров	Ольга	Геометрия	Профессор
Яров	Иван	Алгебра	Профессор
Яров	Павел	Алгебра	Профессор
Яров	Павел	Геометрия	Профессор
Яров	Ольга	Базы данных	Профессор
Яров	Иван	Базы данных	Профессор
Яров	Павел	Базы данных	Профессор

Многозначная зависимость

$$r(R), \quad X, Y, Z \in R$$

Многозначная зависимость

$$r(R), \quad X, Y, Z \in R$$

$$X \twoheadrightarrow Y$$

Многозначная зависимость

Преподаватель

ФИО	Имя	Предмет	Должность
Яров	Ольга	Алгебра	Профессор
Яров	Иван	Геометрия	Профессор
Яров	Ольга	Геометрия	Профессор
Яров	Иван	Алгебра	Профессор
Яров	Павел	Алгебра	Профессор
Яров	Павел	Геометрия	Профессор
Яров	Ольга	Базы данных	Профессор
Яров	Иван	Базы данных	Профессор
Яров	Павел	Базы данных	Профессор

Многозначная зависимость

Преподаватель

ФИО	Имя	Предмет	Должность
Яров	Ольга	Алгебра	Профессор
Яров	Иван	Геометрия	Профессор
Яров	Ольга	Геометрия	Профессор
Яров	Иван	Алгебра	Профессор
Яров	Павел	Алгебра	Профессор
Яров	Павел	Геометрия	Профессор
Яров	Ольга	Базы данных	Профессор
Яров	Иван	Базы данных	Профессор
Яров	Павел	Базы данных	Профессор

ФИО $\rightarrow$ Должность
ФИО $\twoheadrightarrow$ Имя
ФИО $\twoheadrightarrow$ Предмет

Четвертая нормальная форма

Преподаватель

ФИО	Имя	Предмет	Должность
Яров	Ольга	Алгебра	Профессор
Яров	Иван	Геометрия	Профессор
Яров	Ольга	Геометрия	Профессор
Яров	Иван	Алгебра	Профессор
Яров	Павел	Алгебра	Профессор
Яров	Павел	Геометрия	Профессор
Яров	Ольга	Базы данных	Профессор
Яров	Иван	Базы данных	Профессор
Яров	Павел	Базы данных	Профессор

Преподаватель1

ФИО	Должность
Яров	Профессор

Преподаватель2

ФИО	Имя
Яров	Ольга
Яров	Иван
Яров	Павел

Преподаватель3

ФИО	Предмет
Яров	Алгебра
Яров	Геометрия
Яров	Базы данных

Свойства декомпозиции

$$\rho : R \to R_1, R_2, \ldots, R_k$$
$$F \qquad F_1, F_2, \ldots, F_k$$

1. Соединение без потерь

$$\forall r(R): \quad r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \pi_{R_3}(r) \bowtie \cdots \bowtie \pi_{R_k}(r)$$

2. Сохранение функциональных зависимостей

$$\{F_1, F_2, F_3, \ldots, F_k\} \Longrightarrow F$$

Соединение без потерь

Студент

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Соединение без потерь

Студент

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Студент1

ФИО	Группа
Иванов	22305
Петров	22308
Булкин	22305

Студент2

ФИО	Город
Иванов	Кемь
Петров	Кемь
Булкин	Сегежа

Соединение без потерь

Студент

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Студент1

ФИО	Группа
Иванов	22305
Петров	22308
Булкин	22305

Студент2

ФИО	Город
Иванов	Кемь
Петров	Кемь
Булкин	Сегежа

Студент1 $\bowtie$ Студент2

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Соединение без потерь

Студент

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Соединение без потерь

Студент

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Студент3

ФИО	Группа
Иванов	22305
Петров	22308
Булкин	22305

Студент4

Группа	Город
22305	Кемь
22308	Кемь
22305	Сегежа

Соединение без потерь

Студент

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Студент3

ФИО	Группа
Иванов	22305
Петров	22308
Булкин	22305

Студент4

Группа	Город
22305	Кемь
22308	Кемь
22305	Сегежа

Студент3 $\bowtie$ Студент4

ФИО	Город	Группа
Иванов	Кемь	22305
Иванов	Сегежа	22305
Петров	Кемь	22308
Булкин	Кемь	22305
Булкин	Сегежа	22305

Соединение без потерь

Студент

ФИО	Город	Группа
Иванов	Кемь	22305
Петров	Кемь	22308
Булкин	Сегежа	22305

Студент3

ФИО	Группа
Иванов	22305
Петров	22308
Булкин	22305

Студент4

Группа	Город
22305	Кемь
22308	Кемь
22305	Сегежа

Студент3 $\bowtie$ Студент4

ФИО	Город	Группа
Иванов	Кемь	22305
Иванов	Сегежа	22305
Петров	Кемь	22308
Булкин	Кемь	22305
Булкин	Сегежа	22305

Сохранение функциональных зависимостей

Лекции

Студент	Предмет	Преподаватель
Иванов	Алгебра	проф. Белый
Иванов	Физика	проф. Яров
Петров	Алгебра	проф. Белый
Петров	Физика	проф. Серов

$$F = \left\{ \begin{array}{l} \{\text{Студент}, \text{Предмет}\} \rightarrow \text{Преподаватель} \\ \text{Преподаватель} \rightarrow \text{Предмет} \end{array} \right\}$$

Сохранение функциональных зависимостей

Лекции

Студент	Предмет	Преподаватель
Иванов	Алгебра	проф. Белый
Иванов	Физика	проф. Яров
Петров	Алгебра	проф. Белый
Петров	Физика	проф. Серов

$$F = \left\{ \begin{array}{l} \{\text{Студент}, \text{Предмет}\} \to \text{Преподаватель} \\ \text{Преподаватель} \to \text{Предмет} \end{array} \right\}$$

Лекции1

Студент	Преподаватель
Иванов	проф. Белый
Иванов	проф. Яров
Петров	проф. Белый
Петров	проф. Серов

$F_1 = \{\emptyset\}$

Лекции2

Преподаватель	Предмет
проф. Белый	Алгебра
проф. Яров	Физика
проф. Серов	Физика

$F_2 = \{\text{Преподаватель} \to \text{Предмет}\}$

Теоремы

Проектирование БД ”Гостиница”

Проектирование БД ”Гостиница”

tblKlient(intKlientId, txtFIO, txtNomerPasport, datPasport, txtKemPasport, txtInfo)

Проектирование БД ”Гостиница”

tblKlient(intKlientId, txtFIO, txtNomerPasport, datPasport, txtKemPasport, txtInfo)

tblKomnata(intNomerK, intMest, txtKategoria, fltStoimost)

Проектирование БД ”Гостиница”

tblKlient(intKlientId, txtFIO, txtNomerPasport, datPasport, txtKemPasport, txtInfo)

tblKomnata(intNomerK, intMest, txtKategoria, fltStoimost)

txtKategoria $\rightarrow$ fltStoimost
txtKategoria $\rightarrow$ intMest

Проектирование БД "Гостиница"

tblKlient(intKlientId, txtFIO, txtNomerPasport, datPasport, txtKemPasport, txtInfo)

tblKomnata(intNomerK, intMest, txtKategoria, fltStoimost)

txtKategoria $\rightarrow$ fltStoimost
txtKategoria $\rightarrow$ intMest

tblKomnata(intNomerK, intKategoriaId)
tblKategoria(intKategoriaId, intMest, txtKategoria, fltStoimost)

Проектирование

Проектирование реляционной базы данных

Проектирование базы данных

- Построение инфологической модели предметной области
- Построение реляционной модели данных
- Проектирование дополнительных объектов реляционной базы данных

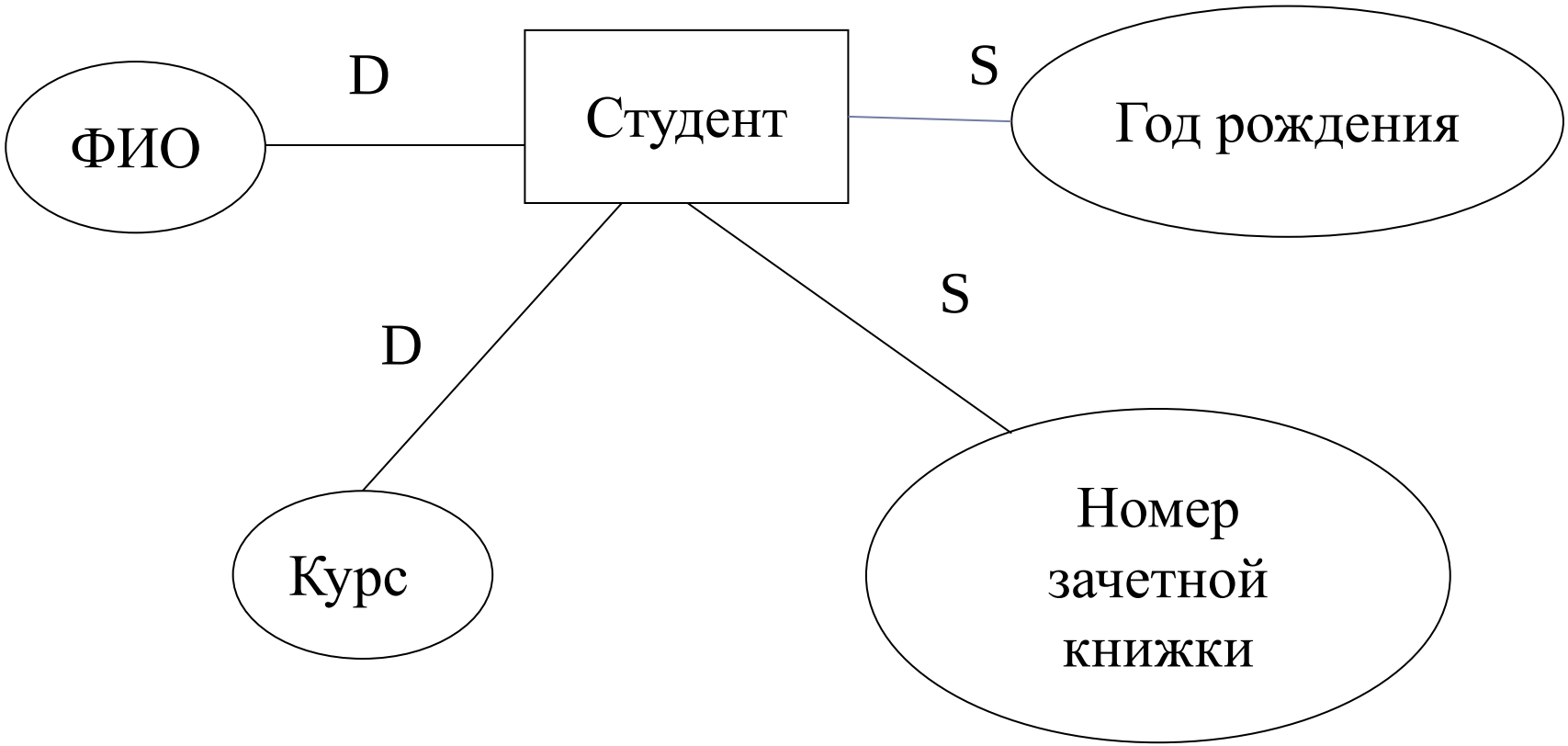
Инфологическая модель предметной области

- Инфологическая модель предметной области – это описание предметной области, выполненное без ориентации на используемые в дальнейшем программные и технические средства.
- Цель: представление семантики предметной области
- Для описания используется модель «сущность–связь» (Entity–Relationship) (ERD, П.Чен, 1976 г.)

Инфологическая модель предметной области

- Объект – это предмет или идея, который может быть четко идентифицирован
- Класс – объекты, объединенные по признакам, характеристикам
- Свойства класса – набор атрибутов, которыми характеризуются все объекты в классе

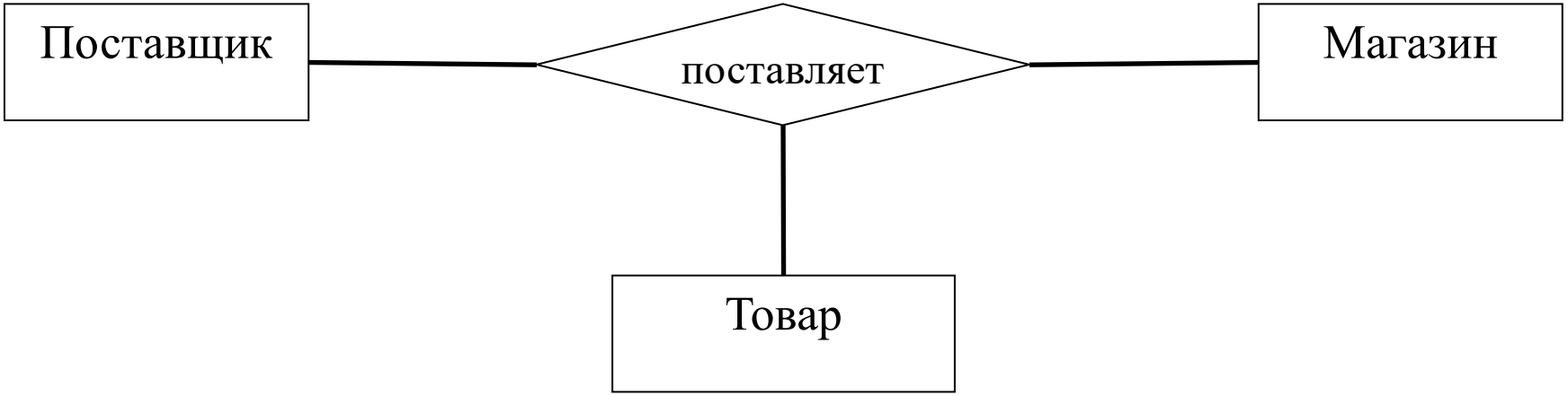
Класс «Студент»



Связи между классами

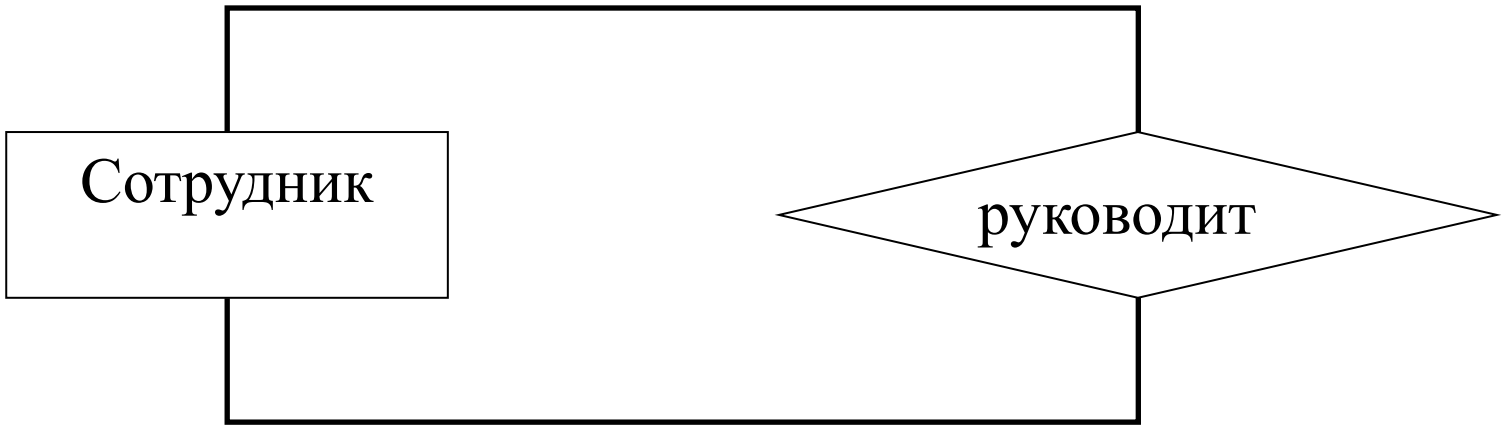
Бинарная связь – между двумя классами

Между тремя и более классами

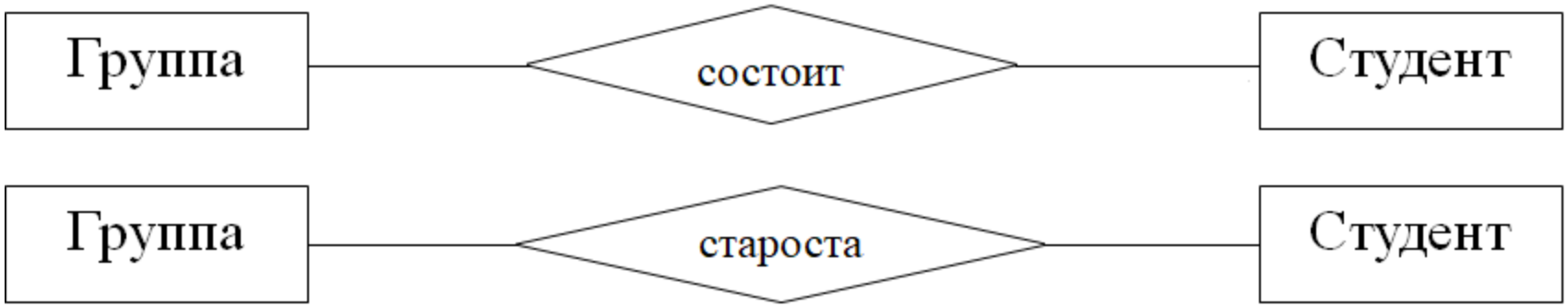


Связи между классами

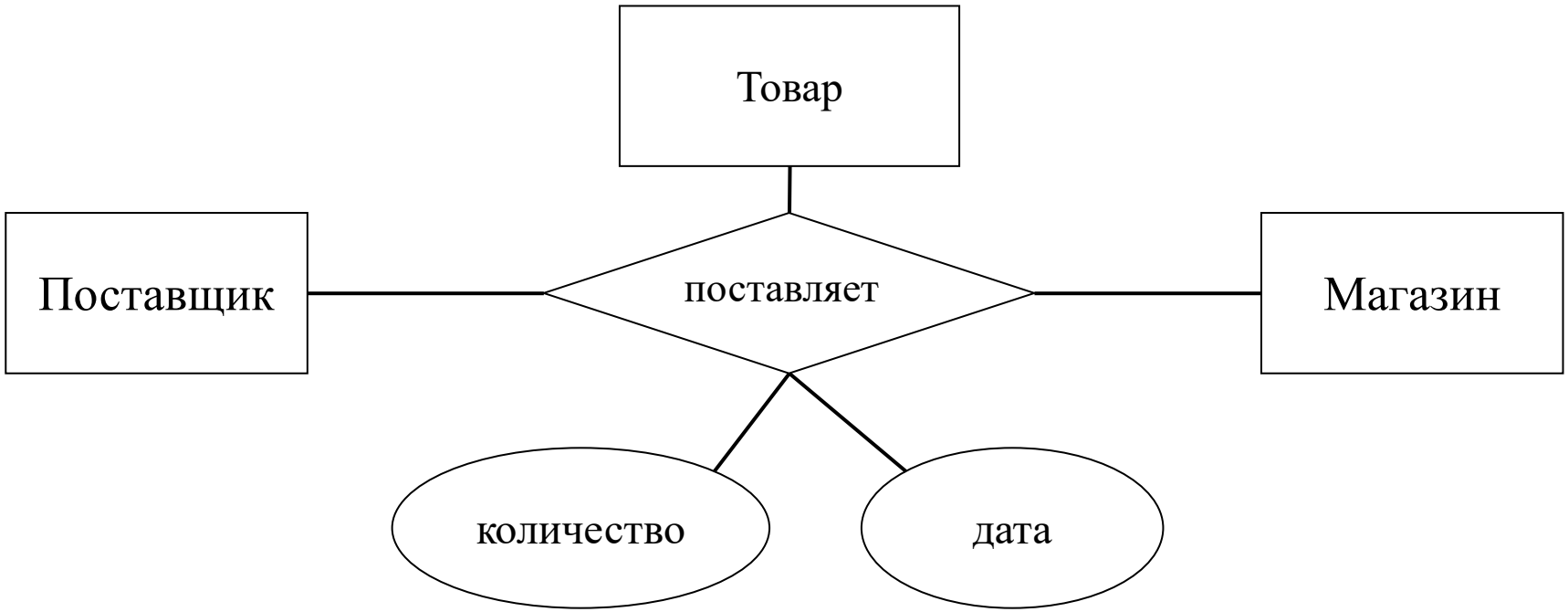
Между объектами одного и того же класса



Между классами можно определить любое количество связей



Связи могут иметь свойства



Типы бинарных связей

- По многозначности
- По обязательности

По многозначности

- «один-к-одному»
- одному объекту из первого класса соответствует ровно один объект из второго класса и, наоборот, одному объекту из второго класса соответствует ровно один объект из первого класса

По многозначности

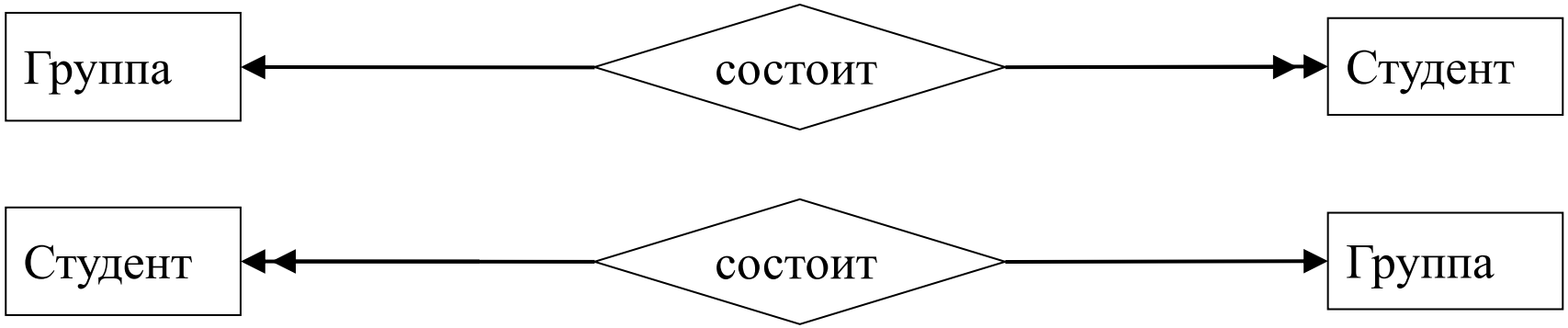
- «один-к-одному»
- одному объекту из первого класса соответствует ровно один объект из второго класса и, наоборот, одному объекту из второго класса соответствует ровно один объект из первого класса

По многозначности

- «один-ко-многим»
- одному объекту из первого класса соответствует несколько объектов из второго класса и одному объекту из второго класса соответствует ровно один объект из первого класса

По многозначности

- «один-ко-многим»
- одному объекту из первого класса соответствует несколько объектов из второго класса и одному объекту из второго класса соответствует ровно один объект из первого класса



По многозначности

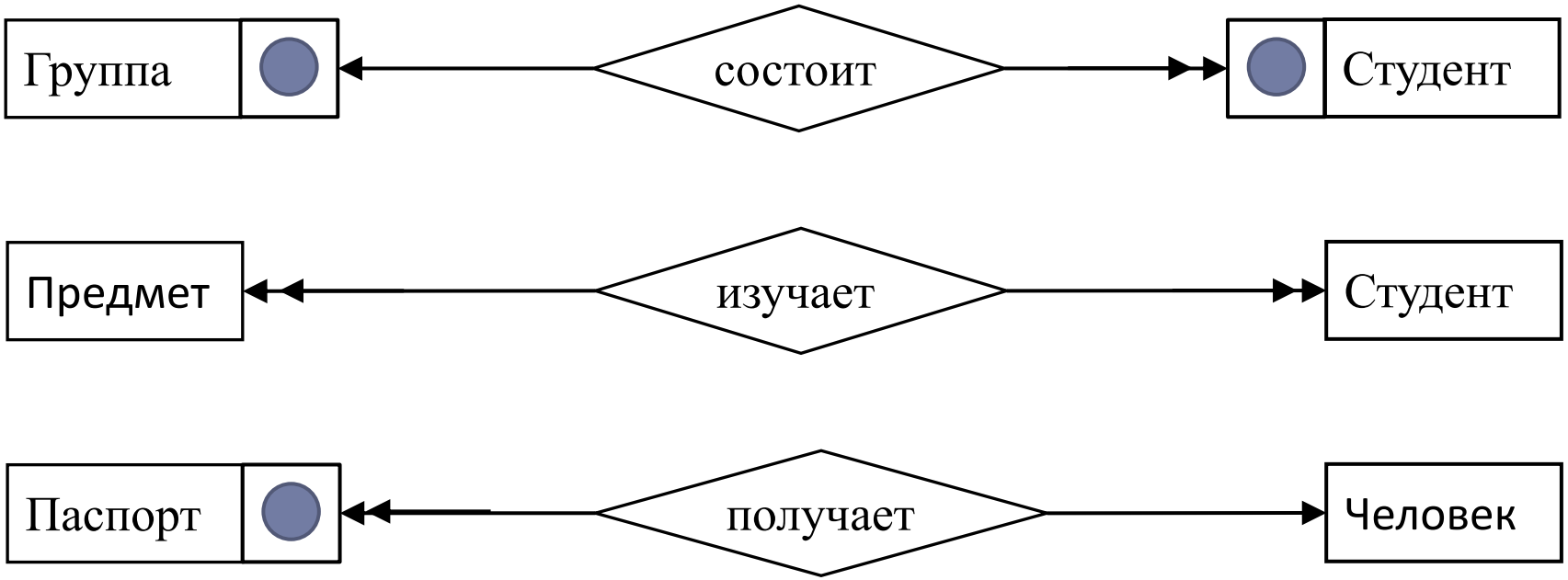
- «многие-ко-многим»
- одному объекту из первого класса соответствует несколько объектов из второго класса и, наоборот, одному объекту из второго класса соответствует несколько объектов из первого класса

По многозначности

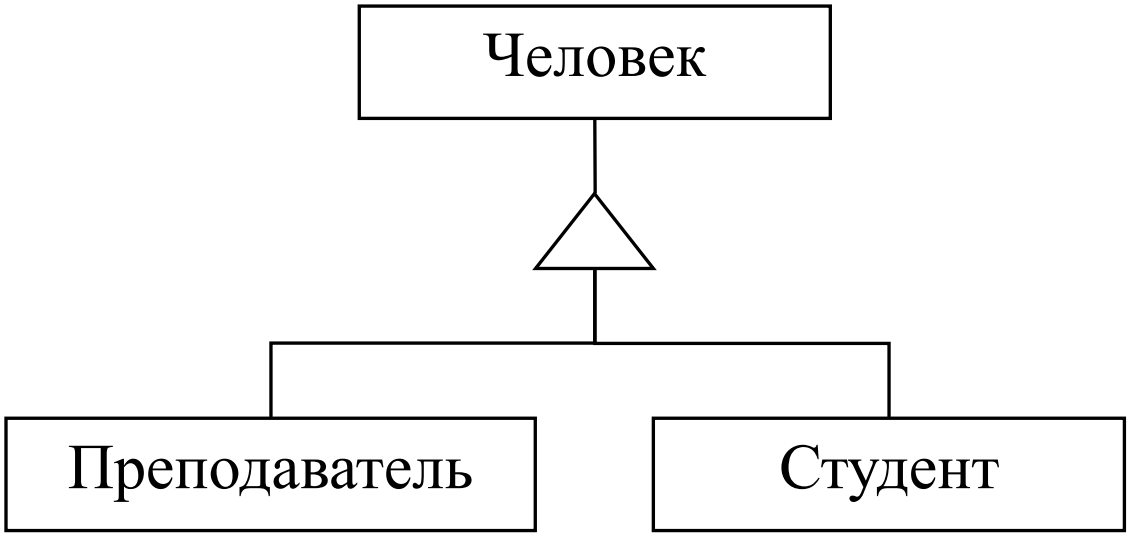
- «многие-ко-многим»
- одному объекту из первого класса соответствует несколько объектов из второго класса и, наоборот, одному объекту из второго класса соответствует несколько объектов из первого класса

По обязательности

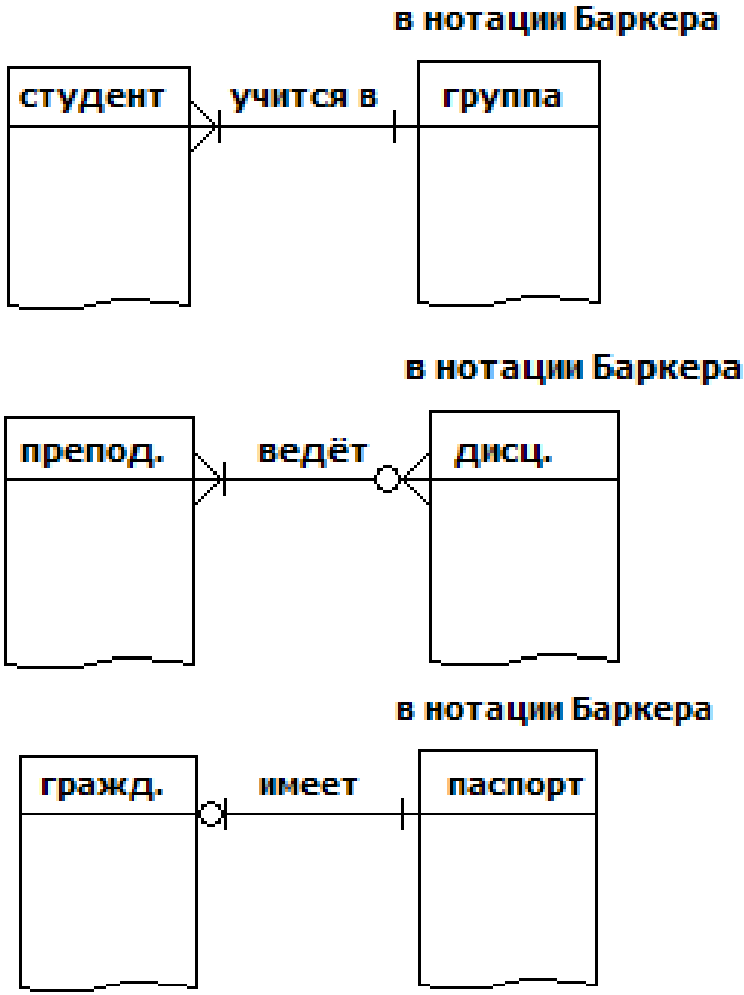
- Связь является обязательной по отношению к классу, если существование объектов класса зависит от наличия связи

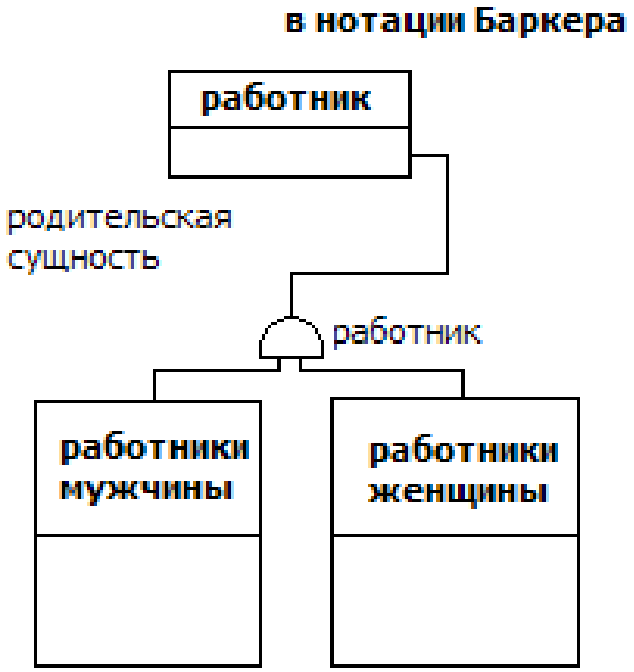


Дополнительные обозначения



В нотации Баркера

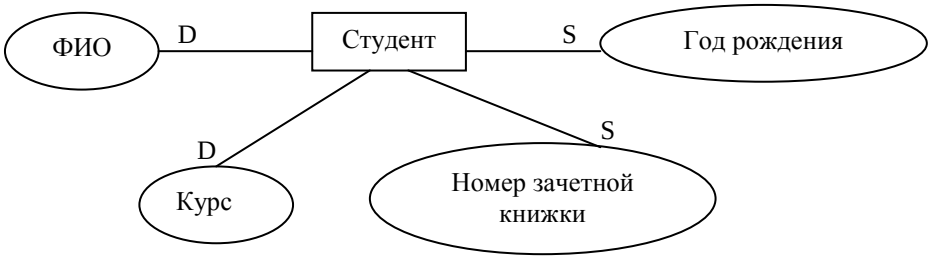




§ 1 Алгоритм построения реляционной модели данных на основе инфологической модели

От модели предметной области очень легко перейти к реляционной модели данных. Для этого нужно выполнить несколько последовательных действий:

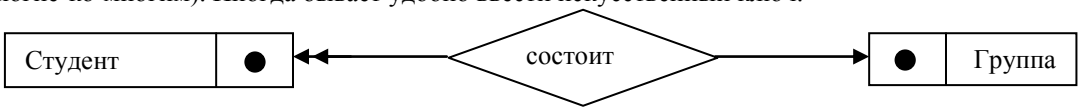
1) Построить отдельную схему отношений для каждого классов объектов:
   a) Для каждого класса объектов сформировать отдельное отношение, атрибутами которого будут свойства этого класса.



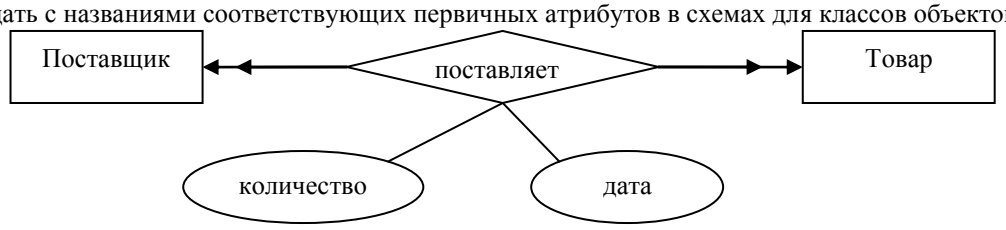
   Например, для объекта Студент создается таблица с именем *Студент* со столбцами: *ФИО, Курс, Год_рождения, Номер*).
   b) Для каждого атрибута определить домен и привести отношение к первой нормальной форме. Например атрибут *ФИО* можно заменить тремя атрибутами *Фамилия*, *Имя*, *Отчество*.
   c) Затем, нужно выделить в отношении первичный ключ. В данном случае ключом будет атрибут *Номер*, так как номера зачетных книжек не дублируются.

   Если ключ получается громоздким, то есть включает в себя много атрибутов или атрибут, имеющий в качестве домена длинную строку символов, то удобно ввести искусственный (суррогатный) ключ – новый атрибут с целочисленным доменом. Значения этого домена будут уникальными для каждого кортежа отношения.

2) Построить отдельную схему отношений для каждой связи.
   a) Для связи схема отношения строится следующим образом: для каждого класса, участвующего в связи, из соответствующей ему схемы отношения ключевые атрибуты копируются в новое отношение; если связь имеет свойства, то они также включаются в схему как атрибуты.
   b) Ключом такого отношения, скорее всего, будут либо ключ одного из классов, входящих в связь (это справедливо для связи «один-ко-многим», ключом будет ключ отношения, соответствующего классу, входящему в связь со стороны «многие»), либо оба ключа вместе (для бинарной связи «многие-ко-многим). Иногда бывает удобно ввести искусственный ключ.



   Например, для связи между классами Студент и Группа создается таблица с именем *Состоит* со столбцами: *Студент, Номер_группы*, где атрибут *Студент* будет содержать номер зачетной книжки студента, который является ключом таблицы **Студент**, а *Номер_группы* – номер группы, в которой состоит студент, который является ключом таблицы **Группа**. В этом отношении ключом будет атрибут *Студент*, так как, согласно предметной области, каждый студент состоит только в одной группе и в этом отношении не будет двух кортежей с одинаковыми номерами зачетных книжек. Подчеркнем здесь тот факт, что в схеме отношения для связи между классами объектов названия атрибутов, соответствующих ключам схем для классов объектов, необязательно должны совпадать с названиями соответствующих первичных атрибутов в схемах для классов объектов.



   Для связи между классами Товар и Поставщик создается таблица с именем *Поставляет* со столбцами: *Товар, Поставщик, Дата, Количество*.

   Это связь вида «многие-ко-многим», следовательно, в ключ должны входить и атрибут *Товар* и атрибут *Поставщик*. Но один и тот же поставщик может поставлять один и тот же товар несколько раз, следовательно, двух атрибутов для ключа недостаточно. Если в предметной области определено такое ограничение, что в один и тот же день поставщик не может поставить один и тот же товар, то тогда ключом будут три атрибута {*Товар, Поставщик, Дата*}. Если же этого ограничения нет, то в

этом случае даже всех атрибутов будет недостаточно для определения ключа и необходимо ввести искусственный ключ (например, *Номер_поставки* или т. п.).

c) Схемы отношений, соответствующие связям между классами, будут иметь внешние ключи – атрибуты, соответствующие ключам схем классов, входящих в связь. Например, в отношении **Поставки** атрибут *Товар* будет внешним ключом, ссылающимся на отношение **Товар**, а атрибут *Поставщик* будет внешним ключом, ссылающимся на отношение **Поставщик.**

d) Для атрибутов, соответствующих свойствам связи, необходимо определить домены.

3) Оптимизировать построенную структуру таблиц следующим образом. Таблицы, имеющие одинаковые первичные ключи, объединяются в одну таблицу.

4) Проверить, находятся ли все отношения в 3НФ. При правильном выполнении 1–3 шагов алгоритма построения реляционной модели отношения будут находиться в 3НФ, исключением могут быть только составные атрибуты, нарушающие условия 1НФ, но, исправив их, вы получите 3НФ.

Конспект лекций
по дисциплине «Базы данных»

Тема: Проектирование базы данных

составитель: Л. В. Щёголева

СОДЕРЖАНИЕ

1 Описание предметной области

Первый этап жизненного цикла информационной системы посвящен тщательному изучению предметной области в части ее информационных процессов, автоматизация которых является целью разработки и внедрения информационной системы. Информационные процессы сопровождают практически все бизнес-процессы предметной области. Основным носителем информации и данных в отсутствии информационной системы являются бумажные документы. Поэтому процесс изучения предметной области заключается в исследовании ее бизнес-процессов и документооборота.

В качестве предметной области была выбрана гостиница. Деятельность гостиницы охватывает много бизнес-процессов: учет кадров, бухгалтерский учет, работу с клиентами, финансовое планирование и т. д.

В рамках учебного курса охватить все эти процессы не представляется возможным, поэтому для примера был выбран только один процесс, связанный с работой дежурного администратора гостиницы в той части, которая относится к регистрации некоторых событий в работе с клиентами гостиницы.

Мы не будем подробно останавливаться на этапе анализа требований и формулировать полный перечень требований к информационной системе, так как этот процесс достоин отдельного многостраничного описания, а приведем здесь только те сведения о предметной области, которые необходимы для описываемых в настоящем пособии проектных работ. В первую очередь, это будет краткое описание самой предметной области и бизнес-процессов, выполняемых дежурным администратором, во вторую – описание документов, с которыми работает дежурный администратор при выполнении бизнес-процессов.

1.1 Описание деятельности дежурного администратора

Гостиница располагает номерами нескольких категорий: люкс, полулюкс, одноместные, двуместные, блок из двух двуместных номеров, трехместные, которые обозначаются соответственно Л, П, 1С, 2С, 2А, 3С. Цена номера определяется только его категорией. Клиент может заранее заказать один или несколько номеров в гостинице. Этот процесс называется бронированием. При бронировании дежурный администратор заполняет карту брони. Клиент может без предварительного бронирования прийти в гостиницу, чтобы снять номер. При поселении клиента на него заполняется карта визита. Если клиент представляет собой группу (семью), то карта визита заполняются на каждого

человека, карта брони – одна на всех (в карте брони указываются все бронируемые номера).

Проживающим в гостинице могут быть предоставлены дополнительные услуги (питание, бассейн, тренажерный зал, аренда конференц-зала, аренда места на автостоянке и др.).

Все оказанные услуги вносятся в карту визита клиента. В любой момент времени клиент может оплатить проживание и услуги. Для этого дежурный администратор выписывает наличный счет, который прикрепляется к карте визита. Для оплаты проживания всегда оформляется отдельный счет, в котором нет других услуг, этот счет включает стоимость номера и стоимость бронирования, если этот номер был забронирован клиентом. При выезде из гостиницы для всех неоплаченных ранее услуг формируется счет (или несколько по желанию клиента), а также формируется счет за проживание, если оно не было оплачено ранее, после чего карта визита закрывается.

Если клиент, забронировавший номер в гостинице, не приезжает в указанный срок, то его имя заносится в черный список и в следующий раз ему может быть отказано в бронировании.

Если клиент заранее отказывается от бронирования, его карта брони аннулируется и передается в архив.

Все карты брони, карты визита, счета сохраняются в течение пяти лет.

Ежедневно в конце смены каждый дежурный администратор готовит отчет. Отчет содержит информацию о выписанных дежурным администратором в течение смены картах визита и счетах.

После заполнения карты брони или карты визита, а также после отъезда клиента из гостиницы дежурный администратор вносит соответствующую информацию в Лист заселения, который отражает текущее состояние гостиницы.

Нумерация карт брони, карт визита и счетов начинается с единицы в начале каждого календарного года.

1.2 Описание документов

При выполнении бизнес-процессов дежурный администратор заполняет следующие документы: Карта визита (рис. 1.1), Карта брони (рис. 1.2), Счет (рис. 1.3), Лист заселения (рис. 1.4) и Отчет (рис. 1.5).

Карта визита № _____ Карта брони № _____

Фамилия _____

Имя _____

Отчество _____

Паспортные данные:

серия и номер паспорта _____

кем и когда выдан паспорт _____

Дата приезда _____

Дата отъезда _____

Комната _____ Стоимость _____ ☐ оплачено

Оказанные услуги

Наименование	Дата	Цена	Количество	Сумма	Оплачено

Выписанные счета

Номер счета	Дата	Сумма

Рисунок 1.1 – Карта визита

Карта брони №		Дата	
ФИО клиента			
Дата приезда			
Дата отъезда			
Бронируемые комнаты			
Контактная информация клиента			
Деж. адм-р			
	подпись		

Рисунок 1.2 – Карта брони

Счет №		Дата		
ФИО				
Дата приезда				
Дата отъезда				
Комната				
Наименование	Дата	Цена	Количество	Сумма
Итого				
Сумма прописью				
Деж. адм-р				
	подпись			

Рисунок 1.3 – Счет

Люкс	101 Иванов 2.10– 4.10	102	103 Лукин (бронь) 12.12– 18.12			
Одно- местные	201	202	203 Розова (бронь) 10.10– 14.10	204 Сидоров 30.09– 3.10	205	206
…	…	…	…	…	…	…

Рисунок 1.4 – Лист заселения по состоянию на 2 октября

Отчет дежурного администратора ФИО _____

Дата _____

Комната	ФИО	Дата отъезда

Номер счета	Комната	ФИО	Сумма

_____

подпись

Рисунок 1.5 – Отчет

2 Проектирование базы данных

2.1 Построение инфологической модели предметной области

В первую очередь построим инфологическую модель предметной области, описывающую предметную область без учета используемой в дальнейшем модели данных. Это означает, что по инфологической модели в дальнейшем может быть построена как реляционная, так и объектная база данных. Для описания инфологической модели воспользуемся моделью «сущность-связь» в нотации Баркера.

На первом шаге выделим классы (сущности) предметной области. Под классом понимается совокупность объектов предметной области, обладающих одинаковыми свойствами.

В рассматриваемой предметной области выделим следующие классы:

– Клиент – человек, который либо забронировал номер в гостинице, либо проживает в гостинице. Свойствами класса являются: ФИО, Паспортные данные, Контактная информация, Наличие в черном списке (рис. 2.1);

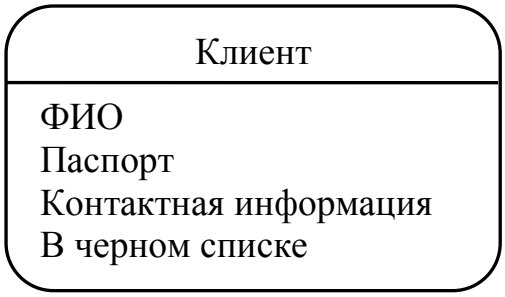


Рисунок 2.1 – Класс «Клиент»

– Комната – номер в гостинице. Так как слово номер может иметь много значений в рассматриваемой предметной области, например номер номера, то вместо словосочетания «Номер гостиницы» будем использовать для названия класса слово «Комната». Свойствами класса являются: Номер комнаты, этаж (рис. 2.2);

Рисунок 2.2 – Класс «Комната»

– Категория – категория номера. Свойствами класса являются: Название категории, Стоимость номера, Описание (рис. 2.3);

Рисунок 2.3 – Класс «Категория»

– Услуга – тип услуги, оказываемой в гостинице. Например, парковка, ужин, аренда конференц-зала и другие. Свойствами класса являются: Название типа услуги, Стоимость услуги (рис. 2.4);

Рисунок 2.4 – Класс «Услуга»

– Оказанная услуга – факт предоставления услуги клиенту, который заносится в карту визита. Свойствами класса являются: Дата оказания услуги, Количество услуг, Информация об оплате услуги (рис. 2.5);

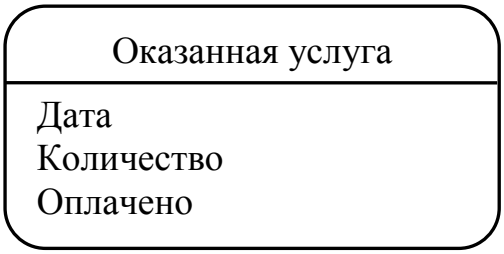


Рисунок 2.5 – Класс «Оказанная услуга»

– Визит – факт проживания одного клиента в гостинице в некоторый период времени, по факту проживания оформляется карта визита. Свойствами класса являются: Дата приезда, Дата отъезда, Номер карты визита, Информация об оплате проживания, Информация о том, проживает ли еще клиент, или карта визита уже закрыта (рис. 2.6);

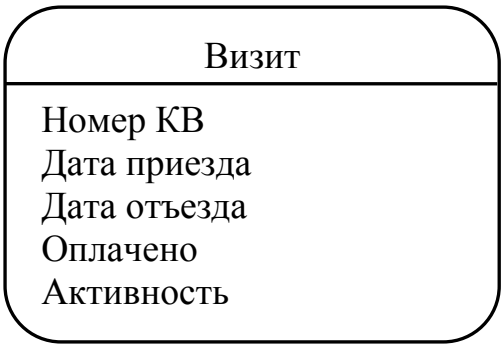


Рисунок 2.6 – Класс «Визит»

– Бронь – факт бронирования клиентом одного или нескольких номеров гостиницы на некоторый период времени, по факту бронирования оформляется карта брони. Свойствами класса являются: Дата приезда, Дата отъезда, Номер карты брони, Состояние (принимает значения: активна, реализована (т. е. клиент, оформивший бронь, приехал и оформил карту визита), аннулирована, просрочена) (рис. 2.7);

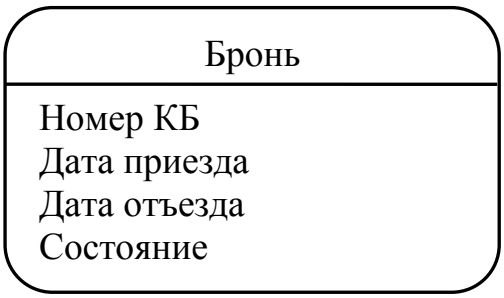


Рисунок 2.7 – Класс «Бронь»

– Счет – счет, выписываемый за проживание или оказанные услуги. Свойствами класса являются: Дата выписки счета, Номер счета, Итоговая сумма (рис. 2.8);

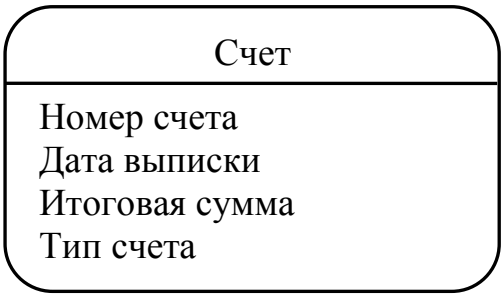


Рисунок 2.8 – Класс «Счет»

– Администратор – дежурный администратор гостиницы. Свойствами класса являются: ФИО (рис. 2.9);

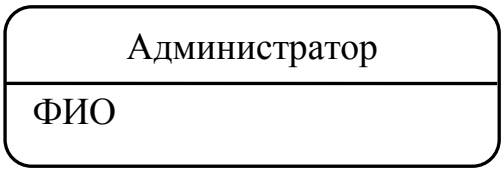


Рисунок 2.9 – Класс «Администратор»

– Отчет – послесменный отчет дежурного администратора. Свойствами класса являются: Дата формирования отчета (рис. 2.10).

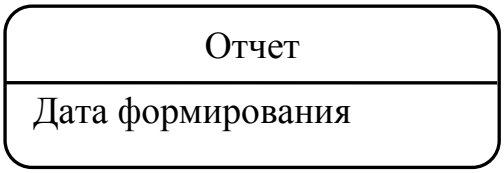


Рисунок 2.10 – Класс «Отчет»

Кому-то, может быть, захочется в качестве отдельной сущности выделить класс Гостиница. Этого делать не нужно. В рассматриваемом примере гостиница не является классом, гостиница – это предметная область, потому что база данных создается для хранения информации о деятельности только одной гостиницы. Если бы предполагалось хранить информацию о деятельности нескольких гостиниц, то тогда должен был бы появиться класс Гостиница, но предметная область в этом случае уже называлась бы «Сеть гостиниц».

Теперь построим связи между классами.

В рассматриваемой инфологической модели будут представлены следующие варианты связей (рис. 2.11):

- Класс А связан с Классом Б связью «один-ко-многим», по отношению к Классу А связь является необязательной, т. е. объект Класса А может быть не связан ни с одним объектом Класса Б, по отношению к Классу Б связь является обязательной, т. е. объект Класса Б должен быть связан с каким-либо объектом Класса А (рис. 2.11 (а));
- Класс А связан с Классом Б связью «один-ко-многим», по отношению к Классу А связь является необязательной, т. е. объект Класса А может быть не связан ни с одним объектом Класса Б, по отношению к Классу Б связь является необязательной, т. е. объект Класса Б может быть не связан ни с одним объектом Класса А (рис. 2.11 (б));
- Класс А связан с Классом Б связью «один-к-одному», по отношению к Классу А связь является необязательной, т. е. объект Класса А может быть не связан ни с одним объектом Класса Б, по отношению к Классу Б связь является необязательной, т. е. объект Класса Б может быть не связан ни с одним объектом Класса А (рис. 2.11 (в));
- Класс А связан с Классом Б связью «многие-ко-многим», по отношению к Классу А связь является необязательной, т. е. объект Класса А может быть не связан ни с одним объектом Класса Б, по отношению к Классу Б связь является обязательной, т. е. объект Класса Б должен быть связан с каким-либо объектом Класса А (рис. 2.11 (г)).

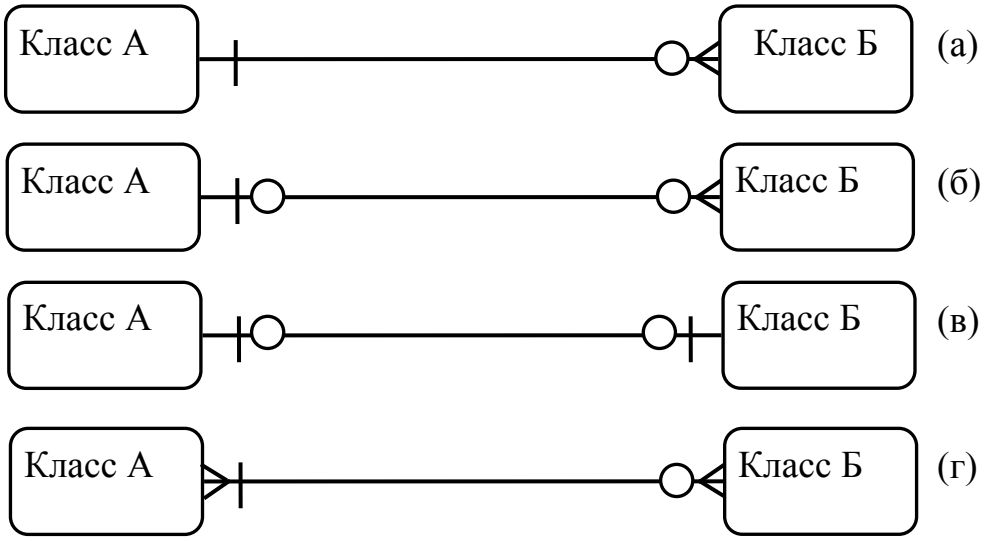


Рисунок 2.11 – Типы связей в инфологической модели

1. Бронь оформляется клиентом, следовательно, между классом Клиент и классом Бронь строим связь с названием **Оформляет бронь** (рис. 2.12). Клиент может и не бронировать заранее номер, а вот бронь (карта брони) не может быть оформлена без клиента, следовательно, связь является обязательной по отношению к классу Бронь и необязательной – по отношению к классу Клиент. Если клиенту понравилось в гостинице, то он может снова поселиться в гостинице, то есть снова забронировать номер, однако карта брони оформляется только на одного клиента, следовательно, связь Оформляет бронь является связью «один-ко-многим».

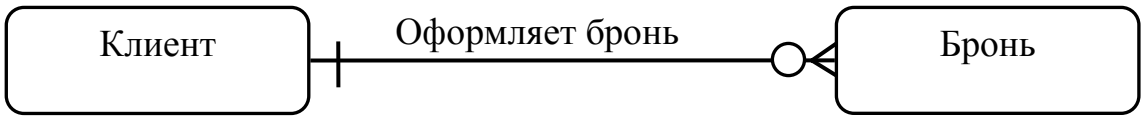


Рисунок 2.12 – Связь «Оформляет бронь»

2. Клиент бронирует один или несколько номеров гостиницы, следовательно, между классом Бронь и классом Комната строим связь с названием **Забронировано** (рис. 2.13). Комната может существовать и без бронирования, а вот бронь всегда оформляется на определенную комнату, следовательно, связь Бронирование является обязательной по отношению к классу Бронь и необязательной – по отношению к классу

Комната. В одной карте брони можно забронировать несколько номеров, и, конечно, на одну и ту же комнату в разные периоды времени могут быть оформлены разные карты брони, следовательно, связь Забронировано является связью «многие-ко-многим».

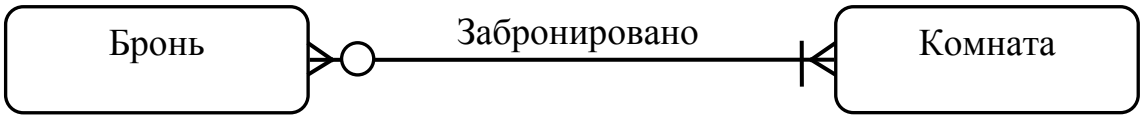


Рисунок 2.13 – Связь «Забронировано»

3. Если клиент приехал по брони, то в карте визита указывается номер карты брони, следовательно, класс Бронь связан с классом Визит. Назовем эту связь **Соответствует** (рис. 2.14). Связь является необязательной по отношению к обоим классам, так как бронирование может быть аннулировано, и значит, для брони не будет соответствующего визита, аналогично, поселиться в гостинице можно и без предварительного бронирования. Связь Соответствует является связью «один-ко-многим», так как можно забронировать несколько номеров, и в них будут проживать разные клиенты, следовательно, будут оформлены разные карты визита, но оформить карту брони в соответствии с несколькими картами брони нельзя.

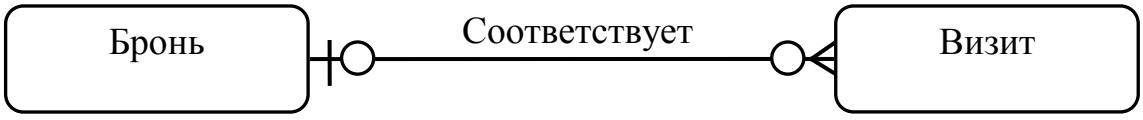


Рисунок 2.14 – Связь «Соответствует»

4. Бронь оформляется одним дежурным администратором, следовательно, между классами Бронь и Администратор определим связь с названием **Заполняет** (рис. 2.15). Связь является обязательной по отношению к классу Бронь и необязательной – по отношению к классу Администратор, так как дежурный администратор может существовать без оформления карт брони, а в карте брони обязательно должно быть указано, кто оформил карту. Связь является связью «многие-ко-одному», так как дежурный администратор за время работы может оформить любое количество карт брони, а каждую карту брони подписывает только один дежурный администратор.

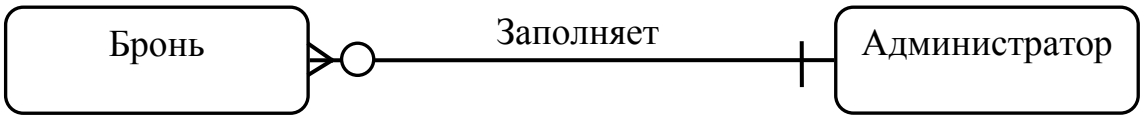


Рисунок 2.15 – Связь «Заполняет»

5. Класс Клиент связан с классом Визит связью с названием **Оформляет визит** (рис. 2.16). Эта связь является обязательной по отношению к классу Визит и необязательной – по отношению к классу Клиент. Связь является связью «один-ко-многим», так как клиент может приехать в гостиницу несколько раз, это будут несколько разных визитов.

Рисунок 2.16 – Связь «Оформляет визит»

6. Класс Визит связан с классом Комната связью с названием **Проживает** (рис. 2.17). Комната может существовать, даже если в ней никто не проживает, но если клиент оформил карту визита, то он должен проживать в каком-то номере гостиницы, поэтому связь является обязательной по отношению к классу Визит и необязательной – по отношению к классу Комната. Связь является связью «Многие-к-одному», так как клиент может проживать только в одном номере, а в разные периоды времени в номере могут проживать разные клиенты, что соответствует разным визитам.

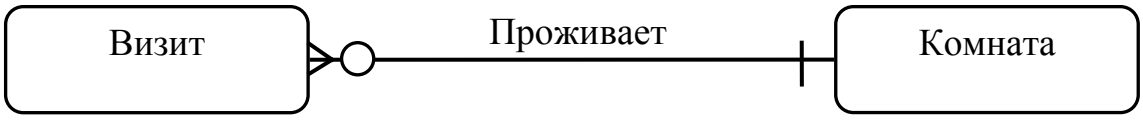


Рисунок 2.17 – Связь «Проживает»

7. Класс Визит связан с классом Счет связью с названием **Выписан** (рис. 2.18). Счет не может быть выписан для клиента, не проживающего в гостинице, но оплата за проживание и услуги может произведена при отъезде, следовательно, связь является обязательной по отношению к классу Счет и необязательной – по отношению к

классу Визит, так как некоторое время с визитом не связан ни один счет.

Рисунок 2.18 – Связь «Выписан»

8. Счет должен быть подписан дежурным администратором, следовательно, Класс Счет связан с классом Администратор связью с названием **Подписан** (рис. 2.19). Счет обязательно должен быть подписан, а дежурный администратор может и не выписать ни одного счета, поэтому связь Подписан является обязательной по отношению к классу Счет и необязательной – по отношению к классу Администратор. Каждый счет подписывается только одним дежурным администратором, а дежурный администратор может выписать любое количество счетов, поэтому связь является связью «многие-к-одному».

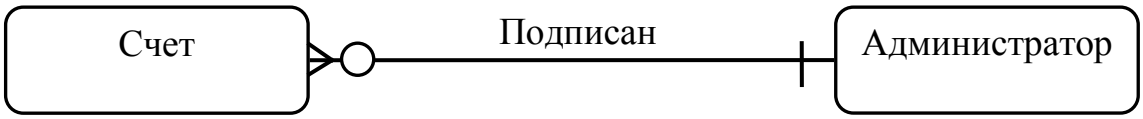


Рисунок 2.19 – Связь «Подписан»

9. Отчет оформляет дежурный администратор, следовательно, Класс Отчет связан с классом Администратор связью с названием **Смена** (рис. 4.20). Отчет обязательно составляется от имени одного администратора, а когда дежурный администратор принимается на работу, у него еще нет ни одного отчета, поэтому связь Смена является обязательной по отношению к классу Отчет и необязательной – по отношению к классу Администратор. Каждый отчет составляется одним дежурным администратором, а дежурный администратор составляет отчет каждую смену, поэтому связь является связью «многие-к-одному».

Рисунок 2.20 – Связь «Смена»

10. Каждая комната относится к одной из категорий, следовательно, класс Комната связан с классом Категория связью с названием **Принадлежит** (рис. 2.21). Связь является обязательной по отношению к классу комната и необязательной – по отношению к классу Категория. К каждой категории принадлежит несколько гостиничных номеров, а каждый номер – только к одной категории, следовательно, связь является связью «многие-к-одному».

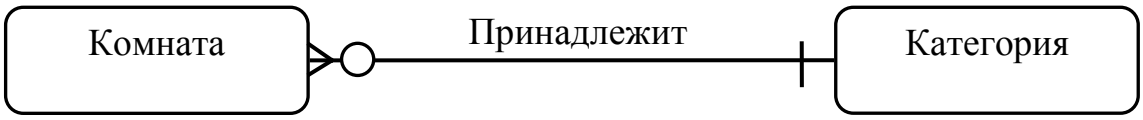


Рисунок 2.21 – Связь «Принадлежит»

11. Отчет включает в себя счета, следовательно, класс Счет связан с классом Отчет связью с названием **Включает счет** (рис. 2.22). Эта связь является необязательной к обоим классам, так как отчет может не содержать ни одного счет (содержит только карты визита), счет включается в отчет в конце смены, т. е. некоторое время существует без отчета. Связь является связью «многие-к-одному», так как отчет может включать несколько счетов, но каждый счет включается только в один отчет.

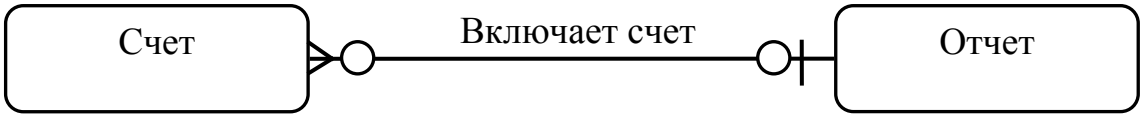


Рисунок 2.22 – Связь «Включает счет»

12. Отчет включает в себя информацию из карт визита, следовательно, класс Визит связан с классом Отчет связью с названием **Включает визит** (рис. 2.23). Эта связь является необязательной к обоим классам, так как отчет может не содержать ни одного визита (содержит только счета), визит включается в отчет в конце смены, т. е. некоторое время существует без отчета. Связь является связью «многие-к-одному», так как отчет может включать несколько визитов, но каждый визит включается только в один отчет.

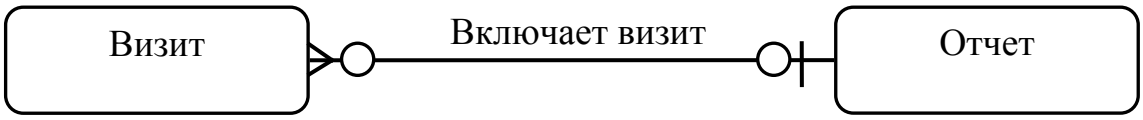


Рисунок 2.23 – Связь «Включает визит»

13. Формально карта визита не подписывается дежурным администратором, но для составления отчета дежурного администратора необходима информация о том, какие карты визита этот дежурный администратор подготовил, поэтому класс Визит связан с классом Администратор связью с названием **Готовит** (рис. 2.24). Связь является обязательной по отношению к классу Визит, так как карта визита всегда готовится дежурным администратором, а дежурный администратор может и не подготовить ни одну карту визита, поэтому по отношению к классу Администратор, связь является необязательной. Связь является связью «многие-к-одному», так как карту визита готовить только один дежурный администратор, а дежурный администратор может подготовить несколько карт визита.

Рисунок 2.24 – Связь «Готовит»

14. Оказанная клиенту услуга должна быть из списка возможных услуг, следовательно, класс Оказанная услуга связан с классом Услуга связью с названием **Тип услуги** (рис. 2.25). Оказанная услуга должна обязательно относиться к какому-либо типу услуг, но пока клиент не получил услугу, услуга не связана ни с какой оказанной услугой, поэтому связь является обязательной по отношению к классу Оказанная услуга и необязательной – по отношению к классу Услуга. Каждая оказанная услуга относится только к одному типу услуг, но по каждому типу услуг может быть оказано любое количество услуг, следовательно, связь является связью «многие-к-одному».

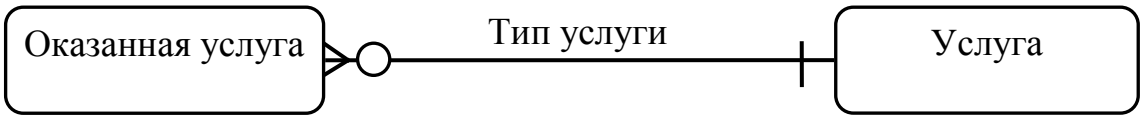


Рисунок 2.25 – Связь «Тип услуги»

15. Оказанная услуга записывается в карту визита, следовательно, класс Оказанная услуга связан с классом Визит связью с названием **Оказана** (рис. 2.26). Услуги оказываются только проживающим в гостинице клиентам, поэтому связь является обязательной по отношению к классу Оказанная услуга. Клиент может и не воспользоваться ни одной дополнительной услугой, следовательно, связь является необязательной по отношению к классу Визит. Связь является связью «многие-к-одному», так как каждая оказанная конкретному клиенту услуга записывается только в одну карту визита (в карту визита этого клиента), а дополнительных услуг клиент может получить сколько угодно.

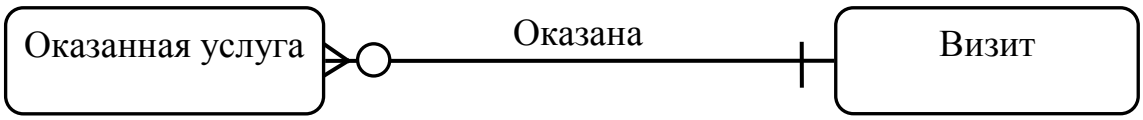


Рисунок 2.26 – Связь «Оказана»

16. Каждая оказанная услуга должна быть оплачена, следовательно, она должна войти в какой-то счет. Это означает, что между классами Оказанная услуга и Счет существует связь с названием **Состоит** (рис. 2.27). Связь является необязательной и по отношению к классу Оказанная услуга и по отношению к классу Счет, так как Оказанная услуга не обязательно должна быть оплачена сразу, следовательно, какое-то время может существовать без участия в связи, счет может содержать только оплату проживания без оплаты оказанных услуг. Связь является связью «многие-к-одному», так как услуга оплачивается только один раз, и, следовательно, состоит только в одном счете, а счет может содержать любое количество услуг.

Рисунок 2.27 – Связь «Состоит»

17. Проживание в номере должно быть оплачено, следовательно, стоимость номера должна попасть в какой-то счет. Это означает, что между классами Визит и Счет существует связь с названием **Оплата проживания** (рис. 2.28). Оплата проживания может быть произведена при отъезде клиента, а не каждый счет содержит оплату проживания, счет может содержать оплату только услуг, следовательно, связь является необязательной по отношению к обоим классам. Проживание оплачивается только одним счетом, следовательно, связь является связью «один-к-одному».

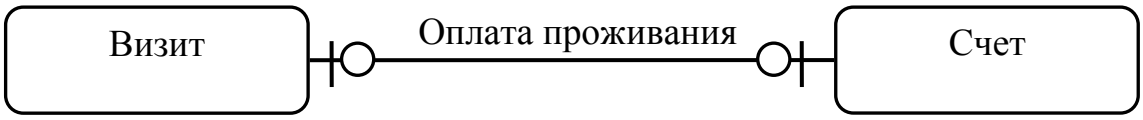


Рисунок 2.28 – Связь «Оплата проживания»

Все связи получились бинарными.

Второй вариант описания предметной области отличается от первого отсутствием класса Оказанная услуга. Вместо этого класса определяется небинарная связь с названием **Оказанная услуга** (рис. 2.29). В связь входят три класса: Визит, Услуга и Счет, что означает, что определенного типа услуга оказана конкретному клиенту, т. е. вписана в его карту визита и оплачена посредством конкретного счета. Связь имеет свойства: Дата оказания услуги, Количество оказанных услуг, Информацию об оплате услуги. В предыдущей версии инфологической модели эта связь описана с помощью класса Оказанная услуга, в который вошли три свойства и связей Тип услуги, Оказана, Состоит.

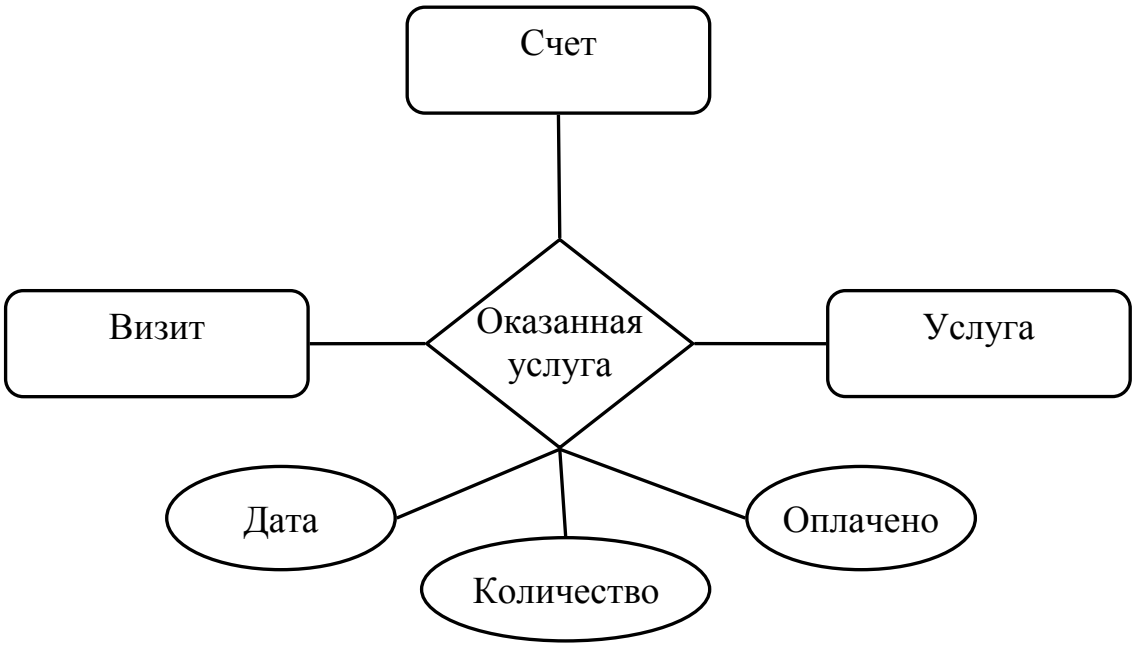


Рисунок 2.29 – Связь «Оказанная услуга»

Оба варианта правильно описывают предметную область и имеют одинаковые права на существование.

Общая схема, описывающая все классы и связи предметной области, называемая концептуальной моделью, представлена на рис. 2.30.

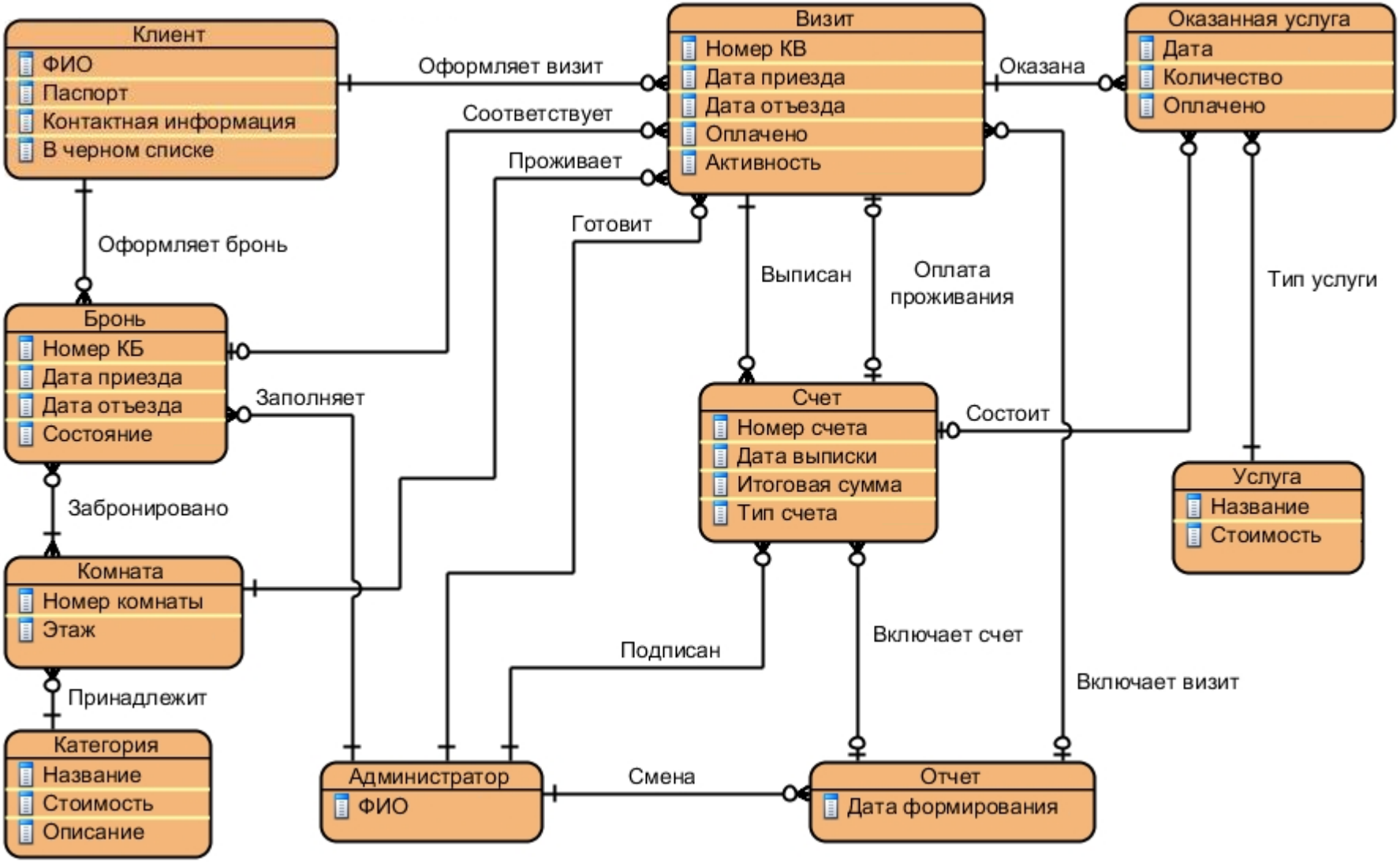


Рисунок 2.30 – Инфологическая модель предметной области

2.2 Построение реляционной модели базы данных

Построим реляционную модель данных для хранения и обработки информации из предметной области.

Для каждого класса построим отдельное отношение, атрибутами которого будут свойства класса. В качестве названия каждого отношения по мере возможности возьмем переведенное на английский язык название класса. К каждому названию добавим префикс «tbl», что будет означать, что спроектированный объект является таблицей базы данных (а не представлением, триггером, индексом и т. д.).

Описание каждого отношения представим в виде таблицы, включающей четыре столбца. Первый столбец содержит наименование атрибута реляционного отношения (столбца или поля реляционной таблицы), второй – описание домена, на котором определен атрибут (тип данных, которые могут быть записаны в столбец), включая ограничения, накладываемые на значения атрибута, третий – описание первичных и внешних ключей отношения (первичный ключ будем обозначать PK, внешний ключ – FK), четвертый – семантическое описание содержимого атрибута. Значение четвертого столбца, как правило, используется в качестве подписи соответствующего поля ввода или заголовка столбца таблицы на экранной форме пользовательского интерфейса. Второй столбец таблицы можно было бы разделить на два отдельных столбца: первый – встроенный в СУБД тип данных, второй – дополнительные ограничения, накладываемые значения атрибута. Такой подход был бы более практичным с точки зрения реализации схемы базы данных в СУБД, но с точки зрения реляционной модели эти два элемента, как раз и определяют домен атрибута, поэтому они были объединены в один столбец. Так как построение реляционной модели базы данных включает несколько шагов, в процессе которых структура реляционных отношений будет изменяться, то описание ограничений, накладываемых на значения атрибута, добавим на последнем шаге проектирования только с той целью, чтобы сэкономить бумагу печатного варианта пособия.

1. Составим реляционное отношение, соответствующее классу Клиент (рис. 2.31).

tblClient			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ClientId	integer	PK	Идентификатор клиента
LastName	varchar(40)		Фамилия клиента
FirstName	varchar(35)		Имя клиента
MiddleName	varchar(35)		Отчество клиента
PassportNumber	varchar(15)		Номер и серия паспорта
WhoPassportIssued	varchar(100)		Кем выдан паспорт
WhenPassportIssued	date		Когда выдан паспорт
ContactInfo	varchar(255)		Контактная информация
BlackList	bit		Наличие в черном списке. Принимает значения: 0 – не находится в черном списке 1 – находится в черном списке

Рисунок 2.31 – Описание отношения «Клиент» (tblClient)

Класс Клиент содержит свойство ФИО. В документе Карта визита необходимо заполнение отдельных граф Фамилия, Имя и Отчество, поэтому в реляционном отношении выделим три отдельных атрибута: LastName, FirstName, MiddleName. Аналогично поступим и со свойством Паспорт, для хранения паспортных данных выделим три отдельных атрибута, содержащих разнородную информацию о паспорте: PassportNumber – серия и номер паспорта, WhoPassportIssued – кем выдан паспорт и WhenPassportIssued – когда выдан паспорт. Атрибут PassportNumber имеет смысл определить на символьном домене, так как номера паспортов иностранных граждан могут содержать нецифровые символы, кроме того, для российских паспортов серия и номер паспорта составляют 10 цифр – это десятизначное число, не стоит хранить такие большие значения как числа, к тому же обычно серию и номер разделяют пробелами. Возьмем 15 символов, чтобы был запас на случай необычных номеров паспортов.

Ключом в отношении tblClient можно было бы сделать атрибут PassportNumber, так как он не должен повторяться. Но здесь есть два «но»: 1. В гостинице могут проживать иностранные граждане, и нет никакой гарантии, что у двух клиентов никогда не совпадут номера и серии паспортов, если они являются гражданами разных государств; 2. Атрибут PassportNumber определен на символьном домене. Желательно, чтобы первичные ключи принимали целочисленные значения, так как с целыми числами операции выполняются

быстрее и при хранении они занимают меньше места. Поэтому для отношения tblClient был определен искусственный (суррогатный) ключ, представляющий собой ничего не означающее с точки зрения предметной области числовое значение – идентификатор клиента.

2. Составим реляционное отношение, соответствующее классу Комната (рис. 2.32).

Двум свойствам класса поставим в соответствие два атрибута. Номер комнаты в общем случае может содержать не только цифры, но и буквы, например, для номеров, входящих в блок, поэтому этот атрибут определим на символьном домене, и не будем делать его первичным ключом. Для первичного ключа добавим в реляционное отношение еще один целочисленный атрибут RoomId – идентификатор комнаты.

tblRoom			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
RoomId	integer	PK	Идентификатор комнаты
RoomNumber	varchar(5)		Номер комнаты
Floor	integer		Этаж

Рисунок 2.32 – Описание отношения «Комната» (tblRoom)

3. Составим реляционное отношение, соответствующее классу Категория (рис. 2.33).

tblRoomCategory			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CategoryId	integer	PK	Идентификатор категории
CategoryName	varchar(2)		Название категории
CategoryInfo	varchar(255)		Описание категории
RoomPrice	numeric(7,0)		Стоимость номера

Рисунок 2.33 – Описание отношения «Категория» (tblRoomCategory)

Атрибут, соответствующий свойству Название категории, мог бы быть первичным ключом, но он принимает символьные значения, поэтому для удобства работы СУБД с первичным ключом введем искусственный ключ CategoryId – идентификатор категории.

4. Составим реляционное отношение, соответствующее классу Услуга (рис. 2.34). Также как и в предыдущем отношении, введем искусственный первичный ключ ServiceId – идентификатор услуги.

tblService			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceId	integer	PK	Идентификатор услуги
ServiceName	varchar(50)		Название услуги
ServicePrice	numeric(7,2)		Стоимость услуги

Рисунок 2.34 – Описание отношения «Услуга» (tblService)

5. Составим реляционное отношение, соответствующее классу Оказанная услуга (рис. 2.35).

tblServiceFact			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceFactId	integer	PK	Идентификатор оказанной услуги
ServiceQuantity	numeric(5,2)		Количество
ServiceFactDate	date		Дата оказания услуги
ServiceFactPaid	bit		Оплата услуги. Принимает значения: 0 – услуга не оплачена 1 – услуга оплачена

Рисунок 2.35 – Описание отношения «Оказанная услуга» (tblClient)

В этом отношении ни один атрибут в отдельности и ни одна совокупность атрибутов не могут быть первичным ключом, так как в один и тот же день могут быть оказаны две услуги в одинаковом количестве с одинаковым значением оплаты. Для этого отношения обязательно надо ввести искусственный ключ ServiceFactId – идентификатор оказанной услуги. Свойство Оплачено может принимать только два значения: «да» или «нет», поэтому в качестве домена соответствующего этому свойству атрибута ServiceFactPaid возьмем логический тип данных. В описании этого атрибута отметим, что значение атрибута «ложь» (или 0) означает, что оказанная услуга не оплачена, а значение «истина» (или 1) означает, что оказанная услуга оплачена.

6. Составим реляционное отношение, соответствующее классу Визит (рис. 2.36).

tblVisit			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer	PK	Идентификатор визита
VisitCardNumber	integer		Номер карты визита
VisitDateBegin	date		Дата приезда
VisitDateEnd	date		Дата отъезда
VisitPaid	bit		Оплата проживания. Принимает значения: 0 – проживание не оплачено 1 – проживание оплачено
Activity	bit		Состояние карты визита. Принимает значения: 1 – активна (клиент проживает) 0 – закрыта (клиент выехал)

Рисунок 2.36 – Описание отношения «Визит» (tblClient)

Ключом отношения tblVisit, исходя из описания предметной области, нельзя сделать атрибут VisitCardNumber – номер карты визита, так как нумерация карт визита каждый календарный год начинается с начала, т. е. со значения 1. Поэтому для отношения tblVisit необходимо ввести искусственный ключ VisitId – идентификатор визита.

7. Составим реляционное отношение, соответствующее классу Бронь (рис. 2.37).

tblReserve			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReserveId	integer	PK	Идентификатор брони
ReserveCardNumber	integer		Номер карты брони
ReserveDateBegin	date		Дата приезда
ReserveDateEnd	date		Дата отъезда
ReserveSate	varchar(12)		Состояние. Принимает значения: – активна (клиент должен приехать) – реализована (клиент приехал) – аннулирована (клиент не приедет) – просрочена (клиент не приехал)

Рисунок 2.37 – Описание отношения «Бронь» (tblClient)

Аналогично номеру карты визита в реляционном отношении tblReserve атрибут номер карты брони – ReserveCardNumber не может быть ключом отношения, так как его значения повторяются каждый год. Введем искусственный первичный ключ ReserveId – идентификатор брони.

Рассмотрим свойство Состояние класса Бронь. Свойство может принимать четыре значения: активна, реализована, аннулирована, просрочена. В реляционной модели это свойство можно реализовать двумя способами. Первый способ заключается в следующем: каждому значению свойства Состояние приписывается целое число, например, значению «активна» ставим в соответствие число 1, значению «реализована» – число 2, значению «аннулирована» – число 3, значению «просрочена» – число 4. В отношении tblReserve определяем атрибут ReserveSate – состояние, доменом которого является множество целых чисел в диапазоне от 1 до 4. Второй способ заключается в том, что атрибут ReserveSate определяется на домене символьных строк длиной не более 12 символов, и значениями этого атрибута будут непосредственно названия состояний брони. В первом случае облегчается обработка на уровне СУБД, ведь с числами работать проще, надежнее и быстрее, чем со строками, но усложняется работа программиста, реализующего интерфейс пользователя, ведь для пользователя числа в качестве

значений состояния брони ничего не означают, и заставлять пользователя запоминать, что число 3 соответствует аннулированной карте брони, ни в коем случае нельзя. Значит, в тех местах пользовательского интерфейса, где нужно представить информацию о состоянии брони, необходимо перевести числа в названия состояний. И об этом надо помнить каждый раз, когда придется что-то исправлять или добавлять в пользовательский интерфейс. Во втором случае хранение значений состояний требует больше памяти, так как 12 символов занимают больше места, чем однозначное число. Но преимущество заключается в том, что при отображении значения в пользовательском интерфейсе не нужно делать никаких дополнительных действий, нужно просто отобразить хранимое значение.

При проектировании разработчик выбирает наиболее предпочтительный в конкретных условиях разработки вариант. В нашем примере большой разницы нет, поэтому для разнообразия примера выберем второй вариант.

8. Составим реляционное отношение, соответствующее классу Счет (рис. 2.38).

Нумерация счетов, как и номеров карты брони и номеров карты визита начинается с 1 каждый календарный год. В качестве первичного ключа реляционного отношения tblCheck введем атрибут CheckId – идентификатор счета.

Так как атрибут CheckType – тип счета принимает только два значения, то определим этот атрибут на логическом типе данных. Совершенно условно припишем значению «счет с услугами» значение «ложь» (или 0), а значению «счет на проживание» – значение «истина» (или 1). Такое соответствие является совершенно неочевидным в отличие от соответствия значений атрибута ServiceFactPaid в реляционном отношении tblServiceFact, поэтому описание значений атрибута CheckType являются очень важным и жизненно необходимым.

tblCheck			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CheckId	integer	PK	Идентификатор счета
CheckNumber	integer		Номер счета
CheckDate	date		Дата выписки счета
CheckTotal	numeric(10,2)		Итоговая сумма
CheckType	bit		Тип счета. Принимает значения: 0 – счет с услугами 1 – счет на проживание

Рисунок 2.38 – Описание отношения «Счет» (tblClient)

9. Составим реляционное отношение, соответствующее классу Администратор (рис. 2.39).

tblAdministrator			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
AdministratorId	integer	PK	Идентификатор администратора
LastName	varchar(40)		Фамилия администратора
FirstName	varchar(35)		Имя администратора
MiddleName	varchar(35)		Отчество администратора

Рисунок 2.39 – Описание отношения «Администратор» (tblClient)

Выделим в отдельные атрибуты фамилию, имя и отчество дежурного администратора: LastName, FirstName, MiddleName. В качестве первичного ключа введем атрибут AdministratorId – идентификатор администратора.

10. Составим реляционное отношение, соответствующее классу Отчет (рис. 2.40).

В качестве первичного ключа введем атрибут ReportId – идентификатор отчета.

tblReport			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReportId	integer	PK	Идентификатор отчета
ReportDate	date		Дата подготовки отчета

Рисунок 2.40 – Описание отношения «Отчет» (tblClient)

Теперь построим отдельное отношение для каждой связи. Атрибутами отношения будут первичные ключи отношений, соответствующих классам, входящим в связь, а также свойства связи.

11. Составим реляционное отношение, соответствующее связи Оформляет бронь (рис. 2.41).

tblReserveClient			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReserveId	integer	PK FK(tblReserve)	Бронь
ClientId	integer	FK(tblClient)	Клиент

Рисунок 2.41 – Описание отношения «Оформляет бронь» (tblClient)

Связь Оформляет бронь – это связь между классами Бронь и Клиент. Для класса Клинет построено реляционное отношение tblClient, первичным ключом которого является атрибут ClientId, следовательно, в отношение tblReserveClient включаем атрибут ClientId, который является внешним ключом, ссылающимся на отношение (таблицу) tblClient, что и записано в третьем столбце на рис. 2.41 как «FK(tblClient)».

Для класса Бронь построено реляционное отношение tblReserve, первичным ключом которого является атрибут ReserveId, следовательно, в отношение tblReserveClient включаем атрибут ReserveId, который является внешним ключом, ссылающимся на отношение (таблицу) tblReserve, что и записано в третьем столбце на рис. 2.41 как «FK(tblReserve)».

Свойств у этой связи нет, следовательно, отношение tblReserveClient будет состоять только из двух атрибутов.

Связь Оформляет бронь является связью «один-ко-многим». Клиент может оформить несколько броней, а каждая бронь оформляется только на одного клиента. Следовательно, в реляционном отношении tblReserveClient значения атрибута ReserveId повторяться не будут, а значения атрибута ClientId могут повторяться. Таким образом, в качестве первичного ключа отношения tblReserveClient можно взять атрибут ReserveId, значениями которого, во-первых, являются целые числа, а, во-вторых, как мы уже установили, значения этого атрибута не могут повторяться.

В результате анализа построенного реляционного отношения можно сформулировать следующее правило: в качестве первичного ключа

реляционного отношения, построенного для связи «один-ко-многим», можно выбирать атрибут, являющийся первичным ключом отношения, соответствующего классу, входящему в связь со стороны «многие».

Заметим, что в четвертом столбце таблицы на рис. 2.41 для атрибутов, являющихся внешними ключами, указываем не «идентификатор», а название того объекта, на который ссылается внешний ключ.

12. Составим реляционное отношение, соответствующее связи Забронировано (рис. 2.42).

tblReservedRoom			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
RoomId	integer	PK FK(tblRoom)	Комната
ReserveId	integer	PK FK(tblReserve)	Бронь

Рисунок 2.42 – Описание отношения «Забронировано» (tblClient)

Связь Забронировано – это связь между классами Комната и Бронь. Для класса Комната построено реляционное отношение tblRoom, первичным ключом которого является атрибут RoomId, следовательно, в отношение tblReservedRoom включаем атрибут RoomId, который является внешним ключом, ссылающимся на отношение (таблицу) tblRoom, что и записано в третьем столбце на рис. 2.42 как «FK(tblRoom)».

Для класса Бронь построено реляционное отношение tblReserve, первичным ключом которого является атрибут ReserveId, следовательно, в отношение tblReservedRoom включаем атрибут ReserveId, который является внешним ключом, ссылающимся на отношение (таблицу) tblReserve, что и записано в третьем столбце на рис. 2.42 как «FK(tblReserve)».

Свойств у этой связи нет.

Связь Забронировано является связью «многие-ко-многим». Следовательно, в реляционном отношении tblReservedRoom и значения атрибута ReserveId, и значения атрибута RoomId могут повторяться. Таким образом, в качестве первичного ключа отношения tblReservedRoom можно взять либо совокупность атрибутов {RoomId, ReserveId}, либо ввести искусственный ключ. Искусственный ключ вводить нецелесообразно, так как, с одной стороны, это трата лишнего объема памяти, а, с другой стороны, в базе данных не будет таблицы, атрибут которой будет ссылаться на первичный ключ

таблицы tblReservedRoom, следовательно, предпочтительнее в качестве первичного ключа отношения tblReservedRoom взять совокупность атрибутов {RoomId, ReserveId}, т. е. первичный ключ будет составным. В третьем столбце таблицы на рисунке 2.42 и для атрибута RoomId, и для атрибута ReserveId записываем «PK».

В результате анализа построенного реляционного отношения можно сформулировать следующее правило: в качестве первичного ключа реляционного отношения, построенного для связи «многие-ко-многим», можно выбирать совокупность атрибутов, являющихся первичными ключами отношений, соответствующих классам, входящим в связь.

Реляционные отношения для следующих связей строятся аналогично, поэтому подробное описание построения этих связей приводить не будем.

13. Реляционное отношение, соответствующее связи Соответствует, представлено на рис. 2.43.

tblVisitReserve			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer	PK FK(tblVisit)	Визит
ReserveId	integer	FK(tblReserve)	Бронь

Рисунок 2.43 – Описание отношения «Соответствует» (tblClient)

14. Реляционное отношение, соответствующее связи Заполняет, представлено на рис. 2.44.

tblReserveAdministrator			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReserveId	integer	PK FK(tblReserve)	Бронь
AdministratorId	integer	FK(tblAdministrator)	Администратор

Рисунок 2.44 – Описание отношения «Заполняет» (tblClient)

15. Реляционное отношение, соответствующее связи Оформляет визит, представлено на рис. 2.45.

tblVisitClient			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer	PK FK(tblVisit)	Визит
ClientId	integer	FK(tblClient)	Клиент

Рисунок 2.45 – Описание отношения «Оформляет визит» (tblClient)

16. Реляционное отношение, соответствующее связи Проживает, представлено на рис. 2.46.

tblVisitRoom			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer	PK FK(tblVisit)	Визит
RoomId	integer	FK(tblRoom)	Комната

Рисунок 2.46 – Описание отношения «Проживает» (tblClient)

17. Реляционное отношение, соответствующее связи Выписан, представлено на рис. 2.47.

tblCheckVisit			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CheckId	integer	PK FK(tblCheck)	Счет
VisitId	integer	FK(tblVisit)	Визит

Рисунок 2.47 – Описание отношения «Выписан» (tblClient)

18. Реляционное отношение, соответствующее связи Подписан, представлено на рис. 2.48.

tblCheckAdministrator			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CheckId	integer	PK FK(tblCheck)	Счет
AdministratorId	integer	FK(tblAdministrator)	Администратор

Рисунок 2.48 – Описание отношения «Подписан» (tblClient)

19. Реляционное отношение, соответствующее связи Смена, представлено на рис. 2.49.

tblReportAdministrator			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReportId	integer	PK FK(tblReport)	Отчет
AdministratorId	integer	FK(tblAdministrator)	Администратор

Рисунок 2.49 – Описание отношения «Смена» (tblClient)

20. Реляционное отношение, соответствующее связи Принадлежит, представлено на рис. 2.50.

tblRoomCategory			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
RoomId	Integer	PK FK(tblRoom)	Комната
CategoryId	Integer	FK(tblCategory)	Категория

Рисунок 2.50 – Описание отношения «Принадлежит» (tblClient)

21. Реляционное отношение, соответствующее связи Включает счет, представлено на рис. 2.51.

tblCheckReport			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CheckId	integer	PK FK(tblCheck)	Счет
ReportId	integer	FK(tblReport)	Отчет

Рисунок 2.51 – Описание отношения «Включает счет» (tblClient)

22. Реляционное отношение, соответствующее связи Включает визит, представлено на рис. 2.52.

tblVisitReport			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer	PK FK(tblVisit)	Визит
ReportId	integer	FK(tblReport)	Отчет

Рисунок 2.52 – Описание отношения «Включает визит» (tblClient)

23. Реляционное отношение, соответствующее связи Готовит, представлено на рис. 2.53.

tblVisitAdministrator			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	Integer	PK FK(tblVisit)	Визит
AdministratorId	Integer	FK(tblAdministrator)	Администратор

Рисунок 2.53 – Описание отношения «Готовит» (tblClient)

24. Реляционное отношение, соответствующее связи Тип услуги, представлено на рис. 2.54.

tblServiceFactService			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceFactId	Integer	PK FK(tblServiceFact)	Оказанная услуга
ServiceId	Integer	FK(tblService)	Услуга

Рисунок 2.54 – Описание отношения «Тип услуги» (tblClient)

25. Реляционное отношение, соответствующее связи Оказана, представлено на рис. 2.55.

tblServiceFactVisit			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceFactId	integer	PK FK(tblServiceFact)	Оказанная услуга
VisitId	integer	FK(tblVisit)	Визит

Рисунок 2.55 – Описание отношения «Оказана» (tblClient)

26. Реляционное отношение, соответствующее связи Состоит, представлено на рис. 2.56.

tblServiceFactCheck			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceFactId	integer	PK FK(tblServiceFact)	Оказанная услуга
CheckId	integer	FK(tblCheck)	Счет

Рисунок 2.56 – Описание отношения «Состоит» (tblClient)

27. Реляционное отношение, соответствующее связи Оплата проживания, представлено на рис. 2.57.

tblVisitPaid			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	Integer	PK FK(tblVisit)	Визит
CheckId	Integer	FK(tblCheck)	Счет

Рисунок 2.57 – Описание отношения «Оплата проживания» (tblClient)

Связь Оплата проживания является связью «один-к-одному». Каждый визит оплачивается только один счетом, каждый счет оформляется только на один визит. Следовательно, в реляционном отношении tblVisitPaid и значения атрибута VisitId, и значения атрибута CheckId повторяться не будут. Таким образом, в качестве первичного ключа отношения tblVisitPaid можно взять и атрибут VisitId, и атрибут CheckId. Исходя из смысла связи, первичным является класс Визит, так как каждый визит должен быть оплачен, поэтому в качестве первичного ключа выберем атрибут VisitId.

В результате анализа построенного реляционного отношения можно сформулировать следующее правило: в качестве первичного ключа реляционного отношения, построенного для связи «один-к-одному», можно выбирать любой из атрибутов, являющихся первичными ключами отношений, соответствующих классам, входящим в связь.

Итак, получили 27 реляционных отношений (в скобках указан первичный ключ отношения):

1. tblClient(ClientId)
2. tblRoom(RoomId)
3. tblRoomCategory(CatogoryId)
4. tblService(ServiceId)
5. tblServiceFact(ServiceFactId)
6. tblVisit(VisitId)
7. tblReserve(ReserveId)
8. tblCheck(CheckId)
9. tblAdministrator(AdministratorId)
10. tblReport(ReportId)
11. tblReserveClient(ReserveId)
12. tblReservedRoom(ReserveId, RoomId)
13. tblVisitReserve(VisitId)
14. tblReserveAdministrator(ReserveId)
15. tblVisitClient(VisitId)
16. tblVisitRoom(VisitId)
17. tblCheckVisit(CheckId)
18. tblCheckAdministrator(CheckId)
19. tblReportAdministrator(ReportId)
20. tblRoomCategory(RoomId)
21. tblCheckReport(CheckId)
22. tblVisitReport(VisitId)
23. tblVisitAdministrator (VisitId)
24. tblServiceFactService(ServiceFactId)
25. tblServiceFactVisit(ServiceFactId)
26. tblServiceFactCheck(ServiceFactId)
27. tblVisitPaid(VisitId)

Проведем оптимизацию полученных отношений.

Рассмотрим отношения:

6. tblVisit,
13. tblVisitReserve,
15. tblVisitClient,
16. tblVisitRoom,
22. tblVisitReport,

23. tblVisitAdministrator,
27. tblVisitPaid.

Перечисленные отношения имеют одинаковые первичные ключи. Каждый кортеж каждого из этих отношений описывает один визит: для таблицы tblVisit – это даты приезда, отъезда, для таблицы tblVisitReserve – по какой карте брони был зарезервирован номер, для таблицы tblVisitClient – данные клиента, на которого оформлена карта визита, для таблицы tblVisitRoom – комната, в которой проживает клиент, для таблицы tblVisitReport – в какой отчет включена информация о визите, для таблицы tblVisitAdministrator – кто подготовил карту визита, для таблицы tblVisitPaid – оплачено ли проживание. Поэтому эти отношения можно соединить в одно отношение с именем tblVisit (рис. 2.58).

Остались следующие отношения:

1. tblClient(ClientId)
2. tblRoom(RoomId)
3. tblRoomCategory(CatogoryId)
4. tblService(ServiceId)
5. tblServiceFact(ServiceFactId)
6. tblVisit(VisitId)
7. tblReserve(ReserveId)
8. tblCheck(CheckId)
9. tblAdministrator(AdministratorId)
10. tblReport(ReportId)
11. tblReserveClient(ReserveId)
12. tblReservedRoom(ReserveId, RoomId)
14. tblReserveAdministrator(ReserveId)
17. tblCheckVisit(CheckId)
18. tblCheckAdministrator(CheckId)
19. tblReportAdministrator(ReportId)
20. tblRoomCategory(RoomId)
21. tblCheckReport(CheckId)
24. tblServiceFactService(ServiceFactId)
25. tblServiceFactVisit(ServiceFactId)
26. tblServiceFactCheck(ServiceFactId)

tblVisit			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer	PK	Идентификатор визита
VisitCardNumber	integer		Номер карты визита
VisitDateBegin	date		Дата приезда
VisitDateEnd	date		Дата отъезда
VisitPaid	bit		Оплата проживания. Принимает значения: 0 – проживание не оплачено 1 – проживание оплачено
Activity	bit		Состояние карты визита. Принимает значения: 1 – активна (клиент проживает) 0 – закрыта (клиент выехал)
ReserveId	integer	FK(tblReserve)	Бронь
RoomId	integer	FK(tblRoom)	Комната
CheckId	integer	FK(tblCheck)	Счет на проживание
ClientId	integer	FK(tblClient)	Клиент
ReportId	integer	FK(tblReport)	Отчет
AdministratorId	integer	FK(tblAdministrator)	Администратор

Рисунок 2.58 – Описание отношения tblVisit

Отношения:

7. tblReserve(ReserveId)
11. tblReserveClient(ReserveId)
14. tblReserveAdministrator(ReserveId)

имеют одинаковые первичные ключи. Каждый кортеж каждого из этих отношений описывает одну бронь: для таблицы tblReserve – это даты приезда, отъезда, для таблицы tblReserveClient – данные клиента, на которого оформлена карта брони, для таблицы tblReserveAdministrator – ФИО администратора, оформившего карту брони. Поэтому эти отношения можно соединить в одно отношение с именем tblReserve (рис. 2.59).

tblReserve			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReserveId	integer	PK	Идентификатор брони
ReserveCardNumber	integer		Номер карты брони
ReserveDateBegin	date		Дата приезда
ReserveDateEnd	date		Дата отъезда
ReserveSate	varchar(12)		Принимает значения: – активна (клиент должен приехать) – реализована (клиент приехал) – аннулирована (клиент не приедет) – просрочена (клиент не приехал)
ClientId	integer	FK(tblClient)	Клиент
AdministratorId	integer	FK(tblAdministrator)	Администратор

Рисунок 2.59 – Описание отношения tblReserve

Остались следующие отношения:

1. tblClient(ClientId)
2. tblRoom(RoomId)
3. tblRoomCategory(CatogoryId)
4. tblService(ServiceId)
5. tblServiceFact(ServiceFactId)
6. tblVisit(VisitId)
7. tblReserve(ReserveId)
8. tblCheck(CheckId)
9. tblAdministrator(AdministratorId)
10. tblReport(ReportId)
12. tblReservedRoom(ReserveId, RoomId)
17. tblCheckVisit(CheckId)
18. tblCheckAdministrator(CheckId)
19. tblReportAdministrator(ReportId)
20. tblRoomCategory(RoomId)
21. tblCheckReport(CheckId)
24. tblServiceFactService(ServiceFactId)

25. tblServiceFactVisit(ServiceFactId)
26. tblServiceFactCheck(ServiceFactId)

Отношения:

8. tblCheck(CheckId)
17. tblCheckVisit (CheckId)
18. tblCheckAdministrator (CheckId)
21. tblCheckReport(CheckId)

имеют одинаковые первичные ключи. Каждый кортеж каждого из этих отношений описывает один счет: для таблицы tblCheck – это номер счета, дата выписки, для таблицы tblCheckVisit – на кого был выписан счет, для таблицы tblCheckAdministrator – ФИО администратора, оформившего счет. Поэтому эти отношения можно соединить в одно отношение с именем tblCheck (рис. 2.60).

tblCheck			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CheckId	integer	PK	Идентификатор счета
CheckNumber	integer		Номер счета
CheckDate	date		Дата выписки счета
CheckTotal	numeric(10,2)		Итоговая сумма
CheckType	bit		Тип счета. Принимает значения: 0 – счет с услугами 1 – счет на проживание
VisitId	integer	FK(tblVisit)	Визит
AdministratorId	integer	FK(tblAdministrator)	Администратор
ReportId	integer	FK(tblReport)	Отчет

Рисунок 2.60 – Описание отношения tblCheck

Остались следующие отношения:

1. tblClient(ClientId)
2. tblRoom(RoomId)
3. tblRoomCategory(CatogoryId)

4. tblService(ServiceId)
5. tblServiceFact(ServiceFactId)
6. tblVisit(VisitId)
7. tblReserve(ReserveId)
8. tblCheck(CheckId)
9. tblAdministrator(AdministratorId)
10. tblReport(ReportId)
12. tblReservedRoom(ReserveId, RoomId)
19. tblReportAdministrator(ReportId)
20. tblRoomCategory(RoomId)
24. tblServiceFactService(ServiceFactId)
25. tblServiceFactVisit(ServiceFactId)
26. tblServiceFactCheck(ServiceFactId)

Отношения:

5. tblServiceFact(ServiceFactId)
24. tblServiceFactService(ServiceFactId)
25. tblServiceFactVisit(ServiceFactId)
26 tblServiceFactCheck(ServiceFactId)

имеют одинаковые первичные ключи. Каждый кортеж каждого из этих отношений описывает одну оказанную услугу: для таблицы tblServiceFact – это дата оказания услуги, количество, для таблицы tblServiceFactService – к какому типу относится услуга, для таблицы tblServiceFactVisit – кому была оказана услуга, для таблицы tblServiceFactCheck –счет, в котором была оплачена услуга. Поэтому эти отношения можно соединить в одно отношение с именем tblServiceFact (рис. 2.61).

tblServiceFact			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceFactId	integer	PK	Идентификатор оказанной услуги
ServiceQuantity	numeric(5,2)		Количество
ServiceFactDate	date		Дата оказания услуги
ServiceFactPaid	bit		Оплата услуги. Принимает значения: 0 – услуга не оплачена 1 – услуга оплачена
ServiceId	integer	FK(tblService)	Тип услуги
VisitId	integer	FK(tblVisit)	Визит
CheckId	integer	FK(tblCheck)	Счет

Рисунок 2.61 – Описание отношения tblServiceFact

Остались следующие отношения:

1. tblClient(ClientId)
2. tblRoom(RoomId)
3. tblRoomCategory(CatogoryId)
4. tblService(ServiceId)
5. tblServiceFact(ServiceFactId)
6. tblVisit(VisitId)
7. tblReserve(ReserveId)
8. tblCheck(CheckId)
9. tblAdministrator(AdministratorId)
10. tblReport(ReportId)
12. tblReservedRoom(ReserveId, RoomId)
19. tblReportAdministrator(ReportId)
20. tblRoomCategory(RoomId)

Отношения:

2. tblRoom(RoomId)
20. tblRoomCategory(RoomId)

имеют одинаковые первичные ключи. Каждый кортеж каждого из этих отношений описывает одну комнату гостиницы: для таблицы tblRoom – это номер комнаты и этаж, на котором она расположена, для таблицы tblRoomCategory – к какой категории номеров относится комната. Поэтому эти отношения можно соединить в одно отношение с именем tblRoom (рис. 2.62).

tblRoom			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
RoomId	integer	PK	Идентификатор комнаты
RoomNumber	varchar(5)		Номер комнаты
Floor	integer		Этаж
CategoryId	integer	FK(tblCategory)	Категория

Рисунок 2.62 – Описание отношения tblRoom

Остались следующие отношения:

1. tblClient(ClientId)
2. tblRoom(RoomId)
3. tblRoomCategory(CatogoryId)
4. tblService(ServiceId)
5. tblServiceFact(ServiceFactId)
6. tblVisit(VisitId)
7. tblReserve(ReserveId)
8. tblCheck(CheckId)
9. tblAdministrator(AdministratorId)
10. tblReport(ReportId)
12. tblReservedRoom(ReserveId, RoomId)
19. tblReportAdministrator(ReportId)

Отношения:

2. tblReport(ReportId)
19. tblReportAdministrator(ReportId)

имеют одинаковые первичные ключи. Каждый кортеж каждого из этих отношений описывает один отчет: для таблицы tblReport – это номер дата формирования отчета, для таблицы tblReportAdministrator – ФИО администратора, сформировавшего отчет. Поэтому эти отношения можно объединить в одно отношение с именем tblReport (рис. 2.63).

tblReport			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReportId	integer	PK	Идентификатор отчета
ReportDate	date		Дата подготовки отчета
AdministratorId	integer	FK(tblAdministrator)	Администратор

Рисунок 2.63 – Описание отношения tblReport

Таким образом, база данных состоит из одиннадцати отношений:

1. tblClient
2. tblRoom
3. tblRoomCategory
4. tblService
5. tblServiceFact
6. tblVisit
7. tblReserve
8. tblCheck
9. tblAdministrator
10. tblReport
11. tblReservedRoom

2.3 Дальнейшее проектирование базы данных

2.3.1 Справочные и операционные таблицы

Рассмотрим таблицы

1. tblRoom
2. tblRoomCategory
3. tblService
4. tblAdministrator

Содержимое этих таблиц (т. е. кортежи) будут изменяться редко. Такие таблицы называются справочниками. Для выделения справочников изменим названия этих таблиц: префикс «tbl» заменим на префикс «spr».

Остальные таблицы называются операционными. Они содержат данные, которые постоянно в работе, постоянно обновляются, дополняются.

2.3.2 Дополнительные ограничения на значения данных одного атрибута

Добавим в таблицы, описывающие каждое отношение, ограничения, накладываемые на значения атрибутов реляционного отношения.

1. Для каждого первичного атрибута необходимо добавить ограничение «Запрет NULL», которое означает, что этот атрибут не может содержать пустого значения. По ограничениям реляционной теории, первичный ключ не может содержать пустых значений.
2. Ограничение «Запрет NULL» необходимо добавить и для некоторых других атрибутов, например, для атрибутов LastName и FirstName в отношении tblClient, так как Фамилия и Имя клиента обязательно должны быть заполнены. Например, для атрибута RoomPrice в отношении sprRoomCategory, ведь, у каждой категории номеров обязательно должна быть определена стоимость проживания.
3. Ограничение «Запрет NULL» необходимо добавить для атрибутов, являющихся внешними ключами, соответствующими обязательной связи. Например, связь Оформляет бронь между классами Бронь и Клиент является обязательной по отношению к классу Бронь, следовательно, в реляционном отношении tblReserve (Бронь) значения внешнего ключа ClientId должны быть всегда заполненными. Связь Заполняет является обязательной по отношению к классу Бронь, следовательно, в реляционном отношении tblReserve (Бронь) значения внешнего ключа AdministratorId должны быть всегда заполненными. Таким образом, ограничение «Запрет NULL» необходимо установить для следующих атрибутов:
   - ClientId в отношении tblReserve,
   - AdministratorId в отношении tblReserve,
   - ClientId в отношении tblVisit,
   - RoomId в отношении tblVisit,
   - VisitId в отнршении tblCheck,
   - AdministratorId в отношении tblCheck,
   - AdministratorId в отношении tblReport,
   - CategoryId в отношении sprRoom,
   - AdministratorId в отношении tblVisit,
   - ServiceId в отношении tblServiceFact,
   - VisitId в отношении tblServiceFact.

4. Для первичных ключей, введенных искусственно, можно добавить ограничение на автоматическую генерацию нового значения. Так как значения этих атрибутов не имеют семантической нагрузки, и пользователя нельзя нагружать обязанностью вводить эти значения, то чтобы исключить возможность появления одинаковых значений ключа, можно доверить генерацию новых неповторяющихся значений СУБД.
5. Для атрибута ReserveState в отношении tblReserve необходимо установить ограничение на возможные принимаемые значения.
6. Для некоторых атрибутов можно определить значения по умолчанию, например, для атрибута VisitPaid в отношении tblVisit можно определить значение по умолчанию 0, так как при создании новой записи о визите визит еще не оплачен. Даже если клиент оплачивает проживание сразу во время оформления карты визита, с точки зрения базы данных сначала создается новая запись в отношении tblVisit, и только потом создается запись об оплате счета за проживание в отношении tblCheck. Это связано с тем, что при создании новой записи в отношении tblCheck необходимо обязательно заполнить атрибут VisitId, а его нельзя заполнить, если соответствующая запись не внесена в отношении tblVisit.

В общем случае могут быть определены и другие ограничения, накладываемые на каждое значение отдельно для одного атрибута.

2.3.3 Дополнительные ограничения, связывающие значения нескольких атрибутов

Также могут быть определены ограничения на значения нескольких атрибутов в рамках одного отношения или между атрибутами нескольких отношений. Например, можно определить ограничение на значения атрибутов VisitDateBegin и VisitDateEnd в отношении tblVisit следующего характера: дата отъезда должна быть позднее даты приезда, т. е. «VisitDateBegin < VisitDateEnd». Это будет ограничение для одного отношения. Аналогично можно наложить ограничение на атрибуты VisitDateBegin и VisitDateEnd в отношении tblVisit и атрибут CheckDate в отношении tblCheck следующего характера: дата выписки счета должна быть не ранее даты приезда и не позднее даты отъезда, т. е. «VisitDateBegin ≤ CheckDate ≤ VisitDateEnd для tblVisit.VisitId = tblCheck.VisitId». Это ограничение связывает данные из двух отношений.

Ограничения на данные в рамках одной таблицы могут быть реализованы средствами СУБД вместе с таблицей, а ограничения на данные, хранящиеся в разных таблицах, реализуются как отдельные объекты базы данных – триггеры.

2.3.4 Служебные таблицы

Несколько слов можно сказать про реляционное отношение tblAdministrator. В реальной базе данных записи этой таблицы, скорее всего, будут не только содержать информацию из документов предметной области, но и выполнять служебные для информационной системы функции. Ведь каждый дежурный администратор – это пользователь информационной системы. Прежде чем начать работу с информационной системой пользователь должен пройти процедуру авторизации. Это означает, что в базе данных должна храниться информация о логинах и паролях пользователей, которую можно было бы хранить вместе с ФИО в таблице tblAdministrator.

2.3.5 Правила для внешних ключей

Для всех внешних ключей должны быть определены следующие правила:

1. Запрет на удаление;
2. Каскадное обновление.

2.3.6 Дополнительные ограничения, реализованные в виде триггеров

Для построенной реляционной базы данных можно определить несколько триггеров.

1. Для таблицы tblVisit при обновлении поля CheckId: если поле получает значение Null, то поле VisitPaid получает значение False, иначе поле VisitPaid получает значение True.
2. Для таблицы tblServiceFact при обновлении поля CheckId: если поле получает значение Null, то поле ServiceFactPaid получает значение False, иначе поле ServiceFactPaid получает значение True.
3. Для таблицы tblCheck при добавлении новой записи или изменении записи проверяется условие: «tblVisit.VisitDateBegin ≤ tblCheck.CheckDate ≤ tblVisit.VisitDateEnd для tblVisit.VisitId = tblCheck.VisitId»

2.3.7 Проектирование индексов

Далее, необходимо определить, по каким полям, каких таблиц будут построены индексы. В первую очередь индексы необходимо построить для всех

первичных ключей. Обычно СУБД это делает автоматически. Во вторую очередь индексы необходимо построить для всех внешних ключей. И, наконец, в третью очередь индексы необходимо построить для полей, по которым часто будет выполняться поиск, например, для следующих атрибутов:

- tblClient.LastName;
- tblVisit.VisitCardNumber;
- tblVisit.VisitDateBegin;
- tblReserve.ReserveCardNumber;
- tblReserve.ReserveDateBegin;
- tblCheck.CheckNumber;
- tblCheck.CheckDate.

2.3.8 Оформление результата проектирования

Теперь, можно считать базу данных полностью спроектированной.

На рис. 2.64–2.74. представлен окончательный вариант отношений базы данных:

- ограничения на значения одного атрибута представлены во втором столбце таблиц и выделены курсивом;
- ограничения для внешних ключей представлены в третьем столбце и выделены курсивом;
- ограничения для отношения, индексы и триггеры представлены после описания структуры таблицы.

sprRoomCategory			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CategoryId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор категории
CategoryName	varchar(2) *Запрет NULL*		Название категории
CategoryInfo	varchar(255)		Описание категории
RoomPrice	numeric(7,0) *Запрет NULL*		Стоимость номера
Индексы: CategoryId			

Рисунок 2.64 – Описание отношения sprRoomCategory

sprRoom			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
RoomId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор комнаты
RoomNumber	varchar(5) *Запрет NULL*		Номер комнаты
Floor	integer *Запрет NULL* *Значение по умолчанию – 2*		Этаж
CategoryId	integer *Запрет NULL*	FK(tblCategory) *запрет на удаление; каскадное обновление*	Категория
Индексы: RoomId, CategoryId			

Рисунок 2.65– Описание отношения sprRoom

sprService			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор услуги
ServiceName	varchar(50) *Запрет NULL*		Название услуги
ServicePrice	numeric(7,2) *Запрет NULL*		Стоимость услуги
Индексы: ServiceId			

Рисунок 2.66 – Описание отношения sprService

sprAdministrator			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
AdministratorId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор администратора
LastName	varchar(40) *Запрет NULL*		Фамилия администратора
FirstName	varchar(35) *Запрет NULL*		Имя администратора
MiddleName	varchar(35)		Отчество администратора
Индексы: AdministratorId			

Рисунок 2.67 – Описание отношения sprAdministrator

tblClient			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ClientId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор клиента
LastName	varchar(40) *Запрет NULL*		Фамилия клиента
FirstName	varchar(35) *Запрет NULL*		Имя клиента
MiddleName	varchar(35)		Отчество клиента
PassportNumber	varchar(15) *Запрет NULL*		Номер и серия паспорта
WhoPassportIssued	varchar(100)		Кем выдан паспорт
WhenPassportIssued	date		Когда выдан паспорт
ContactInfo	varchar(255)		Контактная информация
BlackList	bit *Запрет NULL* *Значение по умолчанию – 0*		Наличие в черном списке. Принимает значения: 0 – не находится в черном списке 1 – находится в черном списке
Индексы: ClientId, LastName			

Рисунок 2.68 – Описание отношения tblClient

tblServiceFact			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceFactId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор оказанной услуги
ServiceQuantity	numeric(5,2) *Запрет NULL* *Значение по умолчанию – 1*		Количество
ServiceFactDate	date *Запрет NULL*		Дата оказания услуги
ServiceFactPaid	bit *Запрет NULL* *Значение по умолчанию – 0*		Оплата услуги. Принимает значения: 0 – услуга не оплачена 1 – услуга оплачена
ServiceId	integer *Запрет NULL*	FK(tblService) *запрет на удаление; каскадное обновление*	Тип услуги
VisitId	integer *Запрет NULL*	FK(tblVisit) *запрет на удаление; каскадное обновление*	Визит
CheckId	integer	FK(tblCheck) *запрет на удаление; каскадное обновление*	Счет
Индексы: ServiceFactId, ServiceId, VisitId, CheckId Триггеры: Update(CheckId)			

Рисунок 2.69 – Описание отношения tblServiceFact

tblVisit			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор визита
VisitCardNumber	integer *Запрет NULL*		Номер карты визита
VisitDateBegin	date *Запрет NULL*		Дата приезда
VisitDateEnd	date *Запрет NULL*		Дата отъезда
VisitPaid	bit *Запрет NULL* *Значение по умолчанию – 0*		Оплата проживания. Принимает значения: 0 – проживание не оплачено 1 – проживание оплачено
Activity	bit *Запрет NULL* *Значение по умолчанию – 1*		Состояние карты визита. Принимает значения: 1 – активна (клиент проживает) 0 – закрыта (клиент выехал)
ReserveId	integer	FK(tblReserve) *запрет на удаление; каскадное обновление*	Бронь
RoomId	integer *Запрет NULL*	FK(tblRoom) *запрет на удаление; каскадное обновление*	Комната
CheckId	integer	FK(tblCheck) *запрет на удаление; каскадное обновление*	Счет на проживание

ClientId	integer *Запрет NULL*	FK(tblClient) *запрет на удаление; каскадное обновление*	Клиент
ReportId	integer	FK(tblReport) *запрет на удаление; каскадное обновление*	Отчет
AdministratorId	integer *Запрет NULL*	FK(tblAdministrator) *запрет на удаление; каскадное обновление*	Администратор
Индексы: VisitId, ReserveId, RoomId, CheckId, ClientId, ReportId, AdministratorId, VisitCardNumber, VisitDateBegin Триггеры: Update(CheckId), Insert(работа с таблицей tblReport), Delete(работа с таблицей tblReport) Ограничения: «VisitDateBegin < VisitDateEnd»			

Рисунок 2.70 – Описание отношения tblVisit

tblReserve			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReserveId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор брони
ReserveCardNumber	integer *Запрет NULL*		Номер карты брони
ReserveDateBegin	date *Запрет NULL*		Дата приезда
ReserveDateEnd	date *Запрет NULL*		Дата отъезда
ReserveState	varchar(12) *Запрет NULL* *Возможные значения: «*активна*», «*реализована*», «*аннулирована*», «*просрочена*»* *Значение по умолчанию – «активна»*		Принимает значения: – активна (клиент должен приехать) – реализована (клиент приехал) – аннулирована (клиент не приедет) – просрочена (клиент не приехал)
ClientId	integer *Запрет NULL*	FK(tblClient) *запрет на удаление; каскадное обновление*	Клиент
AdministratorId	integer *Запрет NULL*	FK(tblAdministrator) *запрет на удаление; каскадное обновление*	Администратор
Индексы: ReserveId, ClientId, AdministratorId, ReserveCardNumber, ReserveDateBegin Ограничения: «ReserveDateBegin < ReserveDateEnd»			

Рисунок 2.71 – Описание отношения tblReserve

tblCheck			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
CheckId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор счета
CheckNumber	integer *Запрет NULL*		Номер счета
CheckDate	date *Запрет NULL*		Дата выписки счета
CheckTotal	numeric(10,2) *Запрет NULL*		Итоговая сумма
CheckType	bit *Запрет NULL* *Значение по умолчанию – 0*		Тип счета. Принимает значения: 0 – счет с услугами 1 – счет на проживание
VisitId	integer *Запрет NULL*	FK(tblVisit) *запрет на удаление; каскадное обновление*	Визит
AdministratorId	integer *Запрет NULL*	FK(tblAdministrator) *запрет на удаление; каскадное обновление*	Администратор
ReportId	integer	FK(tblReport) *запрет на удаление; каскадное обновление*	Отчет
Индексы: CheckId, VisitId, AdministratorId, ReportId, CheckNumber, CheckDate Триггеры: Insert(проверка ограничения «tblVisit.VisitDateBegin ≤ tblCheck.CheckDate ≤ tblVisit.VisitDateEnd» для условия «tblVisit.VisitId = tblCheck.VisitId»), Insert(работа с таблицей tblReport), Delete(работа с таблицей tblReport)			

Рисунок 2.72 – Описание отношения tblCheck

tblReport			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ReportId	integer *Запрет NULL* *Автоматическая генерация*	PK	Идентификатор отчета
ReportDate	date *Запрет NULL*		Дата подготовки отчета
AdministratorId	integer *Запрет NULL*	FK(tblAdministrator) *запрет на удаление; каскадное обновление*	Администратор
Индексы: ReportId, AdministratorId			

Рисунок 2.73 – Описание отношения tblReport

tblReservedRoom			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
RoomId	integer *Запрет NULL*	PK FK(tblRoom) *запрет на удаление; каскадное обновление*	Комната
ReserveId	integer *Запрет NULL*	PK FK(tblReserve) *запрет на удаление; каскадное обновление*	Бронь
Индексы: «RoomId, ReserveId», RoomId, ReserveId			

Рисунок 2.74 – Описание отношения tblReservedRoom

2.4 Изменение структуры базы данных

2.4.1 Денормализация схемы отношений

Практическое использование базы данных показало, что выделение отдельного класса Клиент является неудобным. Во-первых, бóльшая часть клиентов проживает в гостинице только один раз, во-вторых, информация о клиенте может поменяться, например, изменится фамилия и номер паспорта, в этом случае искать клиента и менять его паспорт и фамилию не имеет смысла, так как его прошлый визит проходил под другой фамилией, придется вводить его в базу данных как нового клиента, в-третьих, карта визита и карта брони не используются без имени клиента, поэтому каждый раз при обращении к таблицам tblVisit или tblReserve необходимо обратиться и к таблице tblClient, т. е. выполнить соединение таблиц, что требует больше времени, чем извлечение данных из одной таблицы. Поэтому целесообразно информацию о клиенте добавить и в таблицу tblVisit (рис. 2.75), и в таблицу tblReserve, а в таблице tblClient вести только черный список.

tblVisit			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
VisitId	integer	PK	Идентификатор визита
VisitCardNumber	integer		Номер карты визита
LastName	varchar(40)		Фамилия клиента
FirstName	varchar(35)		Имя клиента
MiddleName	varchar(35)		Отчество клиента
PassportNumber	varchar(15)		Номер и серия паспорта
WhoPassportIssued	varchar(100)		Кем выдан паспорт
WhenPassportIssued	date		Когда выдан паспорт
ContactInfo	varchar(255)		Контактная информация
VisitDateBegin	date		Дата приезда
VisitDateEnd	date		Дата отъезда
VisitPaid	bit		Оплата проживания. Принимает значения: 0 – проживание не оплачено 1 – проживание оплачено
Activity	bit		Состояние карты визита. Принимает значения: 1 – активна (клиент проживает) 0 – закрыта (клиент выехал)
ReserveId	integer	FK(tblReserve)	Бронь
RoomId	integer	FK(tblRoom)	Комната
CheckId	integer	FK(tblCheck)	Счет на проживание
ReportId	integer	FK(tblReport)	Отчет
AdministratorId	integer	FK(tblAdministrator)	Администратор

Рисунок 2.75 – Описание отношения tblVisit

Эти изменения означают нарушение третьей нормальной формы отношений, но для практических целей в некоторых случаях приходится жертвовать строгими правилами реляционной теории. Такие действия над структурой базы данных называются денормализацией. Денормализация обычно приводит к появлению дублирования данных в нескольких таблицах, это требует от разработчиков дополнительных усилий (т. е. разработки

дополнительных алгоритмов и тщательно проработанной проектной и технической документации) по обеспечению целостности данных, но дает выигрыш для пользователей информационной системы.

2.4.2 Изменение значений с течением времени

Второе замечание более существенное. Стоимость услуг и стоимость номера с течением времени меняются. Если после очередного изменения стоимости услуги попытаться посмотреть какой-либо счет с изменившейся услугой, выписанный до изменения ее стоимости, то сумма стоимости услуг, входящих в счет, не совпадет с итоговой суммой, сохраненной в таблице tblCheck.

Например, пусть таблицы sprService, tblCheck, tblServiceFact содержат информацию, представленную на рисунке 4.76. Она означает, что для визита с идентификатором, равным 2, выписан счет с номером 23 и идентификатором 9609.

sprService

ServiceId	ServiceName	ServicePrice
1	Парковка	100,00
2	Ужин	300,00
3	Обед	250,00

tblCheck

Check Id	Check Number	Check Date	Check Total	Check Type	Visit Id	Administrator Id	Report Id
9609	23	12.01.2012	850,00	0	2	1	53

tblServiceFact

Service FactId	Service Quantity	Service FactDate	ServiceFact Paid	Service Id	VisitId	CheckId
789583	1	10.01.2012	1	2	2	9609
789584	1	10.01.2012	1	3	2	9609
789585	1	11.01.2012	1	2	2	9609

Рисунок 2.76 – Содержимое отношений sprService, tblCheck, tblServiceFact

Счет включает три услуги: два ужина по 300 руб. и один обед за 250 руб. Это составляет 850 рублей. Печатная форма счета представлена на рисунке 2.77.

Счет № 23 Дата 12.01.2012

ФИО Иванов И.И.

Дата приезда 09.01.2012

Дата отъезда 12.01.2012

Комната 230

Наименование	Дата	Цена	Количество	Сумма
Ужин	10.01.2012	300	1	300
Обед	10.01.2012	250	1	250
Ужин	11.01.2012	300	1	300
Итого				850

Сумма прописью Восемьсот пятьдесят рублей

Деж. адм-р _____

подпись

Рисунок 2.77 – Печатная форма выписанного и оплаченного счета

Пусть после выписки и оплаты счета, через некоторое время стоимость ужина изменилась и стала составлять 350 рублей. В таблице sprService будет храниться новое значение стоимости услуги (рисунок 2.78).

sprService

ServiceId	ServiceName	ServicePrice
1	Парковка	100,00
2	Ужин	350,00
3	Обед	250,00

Рисунок 2.78 – Новое содержимое отношения sprService

В этом случае, если мы попытаемся просмотреть печатную форму счета (рисунок 2.79), то, во-первых, она будет отличаться от той, которая была распечатана и оплачена ранее, во-вторых, она будет содержать противоречивые данные: итоговая сумма не совпадает с суммой стоимостей услуг. Исправлять итоговую сумму счета нельзя, так как реально в кассу была внесена сумма 850 рублей.

Счет №	23	Дата	12.01.2012	
ФИО	Иванов И.И.			
Дата приезда	09.01.2012			
Дата отъезда	12.01.2012			
Комната	230			
Наименование	Дата	Цена	Количество	Сумма
Ужин	10.01.2012	300	1	350
Обед	10.01.2012	250	1	250
Ужин	11.01.2012	300	1	350
Итого				850
Сумма прописью	Восемьсот пятьдесят рублей			
Деж. адм-р	*подпись*			

Рисунок 2.79 – Печатная форма счета после изменения стоимости ужина

Таким образом, построенная структура базы данных не позволяет менять стоимости услуг.

Для того чтобы сделать возможным изменение и стоимости услуг, и стоимости номеров, необходимо изменить структуру базы данных.

Возможны два варианта.

Первый вариант. Хранить период времени, в течение которого действовала каждая стоимость услуги. Для этого потребуется определить два отношения sprService и sprServicePrice (рисунок 2.80). Отношение sprService будет хранить все характеристики услуги, кроме стоимости. Отношение sprServicePrice будет хранить стоимость услуги и дату начала действия этой стоимости. Каждый раз при изменении стоимости услуги в это отношение будет добавляться новый кортеж.

Для того чтобы сформировать счет, необходимо по дате оказания услуги определить в таблице sprServicePrice цену услуги и подставить ее в счет.

Аналогично можно поступить со стоимостью номера, добавив таблицу sprCategoryPrice.

sprService			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceId	integer	PK	Идентификатор услуги
ServiceName	varchar(50)		Название услуги

sprServicePrice			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServicePriceId	integer	PK	Идентификатор стоимости услуги
ServiceId	integer	FK(sprService)	Услуга
ServicePrice	numeric(7,2)		Стоимость услуги
ServicePriceBegin	date		Дата, с которой введена новая стоимость услуги

Рисунок 2.80 – Структура отношений sprService и sprServicePrice

Второй вариант заключается в следующем. В таблицу tblServiceFact необходимо добавить атрибут, который будет хранить стоимость услуги. Таблица sprService будет хранить стоимость услуги на момент ее оказания (рисунок 2.81).

tblServiceFact			
Название атрибута (поля)	Домен (Тип данных)	Ключи	Описание
ServiceFactId	integer	PK	Идентификатор оказанной услуги
ServiceQuantity	numeric(5,2)		Количество
ServiceFactDate	date		Дата оказания услуги
ServiceFactPaid	bit		Оплата услуги. Принимает значения: 0 – услуга не оплачена 1 – услуга оплачена
ServiceId	integer	FK(tblService)	Тип услуги
VisitId	integer	FK(tblVisit)	Визит
CheckId	integer	FK(tblCheck)	Счет
ServicePrice	numeric(7,2)		Стоимость услуги

Рисунок 2.81 – Структура отношения tblServiceFact

Тогда, счет будет формироваться не по справочнику sprService, а только из таблицы tblServiceFact. В этом случае имеет смысл продублировать в таблице tblServiceFact и название услуги.

Аналогично можно поступить со стоимостью номера, добавив в таблицу tblVisit атрибут со стоимостью номера.

Второй вариант приводит к денормализации структуры базы данных, но он значительно проще в реализации.

Выбор варианта зависит от условий реализации.

Redis

СУБД

Redis (REmote DIctionary Server)

сохраняет все данные в памяти

позволяет хранить данные в высокоуровневых структурах данных, таких как строки, хэши, списки, наборы

Redis-CLI (Common Line Interface) - интерфейс для взаимодействия с данными напрямую на сервере Redis

Команды

set <ключ> <значение> – создать запись

set <ключ> <значение> [EX <секунды> |PX <миллисекунды>] – установить время жизни

get <ключ> – получить <значение> по ключу

exists <ключ> – проверить наличие записи с ключом

flushall – удалить все данные сеанса

del <ключ> – удалить запись

append <ключ> <значение> – дополнить значение записи указанным значением

keys – выдать все ключи, имеющиеся в базе данных

incr <ключ> – увеличить значение по ключу на 1

decr <ключ> – уменьшить значение по ключу на 1

incrby <ключ> <приращение> – увеличить значение по ключу на приращение

decrby <ключ> <приращение> – уменьшить значение по ключу на приращение

rename <ключ1> <ключ2> – переименовать ключ1 на ключ2

База данных Redis

Ключ	Значение
A	AAAAA
B	10
Mykey	mom washed the frame
Mykey1	9589327582459
Student_Ivanov	{name: Ivan, code: 2345}
Student_Petrov	454hjfn89433iJ94EM;
22305_Ivanov	2, ABC, 3, 8, A1, A2
22306_Petrov	(F, 10), (A, 23), (D, 45)
22301-Ivanov-B2	13.361389 38.115556 "Palermo"
22301-Ivanov-C	Color red Name Arial

Ключ	Значение
	Строка
	Число
	Список
	Множество
	Упорядоченное множество
	Геоданные
	Записи с именованными полями
	…

Множества

sadd <ключ> <элемент> – добавить элемент в множество ключа

smembers <ключ> – получить все элементы множества

scard <ключ> – получить размер множества по ключу

sunion <ключ1> <ключ2> – объединение множеств двух ключей

sunionstore <ключ> <ключ1> <ключ2> – объединение множеств двух ключей и сохранение результата для нового ключа

sinter <ключ1> <ключ2> – пересечение множеств

sinterstore <ключ> <ключ1> <ключ2> – пересечение множеств двух ключей и сохранение результата для нового ключа

sdiff <ключ1> <ключ2> – вычитание из первого множества второго

sdiffstore <ключ> <ключ1> <ключ2> – разность множеств двух ключей и сохранение результата для нового ключа

sismember <ключ> <элемент> – проверить входит ли элемент в множество ключа

srem <ключ> <элемент> – удалить элемент из множества

Упорядоченные множества

zadd <ключ> <номер> <значение> - добавить значение под номером

zrange <ключ> <минимальный индекс> <максимальный индекс> [withscores] – получить упорядоченный по возрастанию номеров список значений (0 – первый элемент, -1 – последний элемент, withscores – вывод номеров)

zrevrange <ключ> <минимальный индекс> <максимальный индекс> [withscores] – получить упорядоченный по убыванию номеров список значений

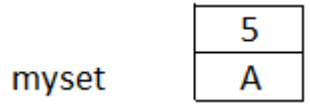
zincrby <ключ> <приращение> <значение> – увеличить номер на приращение для значения

zunion <количество ключей> <ключ> <ключ> … [aggregate sum|min|max] – агрегировать несколько множеств

zunionstore <ключ> <количество ключей> <ключ1> <ключ2> … [aggregate sum|min|max] – агрегировать несколько множеств

Упорядоченные множества

zadd myset 5 A



Упорядоченные множества

zadd myset 5 A

zadd myset 2 B

myset

2	5
B	A

Упорядоченные множества

zadd myset 5 A

zadd myset 2 B

zadd myset 9 C

myset

2	5	9
B	A	C

Упорядоченные множества

zadd myset 5 A

zadd myset 2 B

zadd myset 9 C

zincrby myset 5 A

myset

2	9	10
B	C	A

Упорядоченные множества

zadd myset 5 A

zadd myset 2 B

zadd myset 9 C

zincrby myset 5 A

zincrby myset 3 C

myset

2	10	12
B	A	C

Упорядоченные множества

zadd myset 5 A

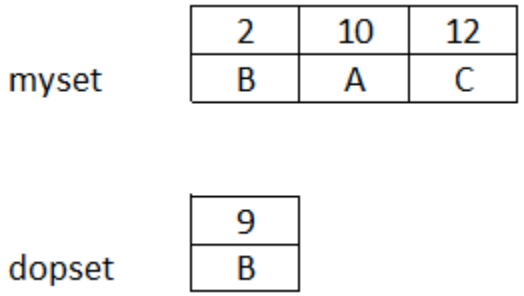
zadd myset 2 B

zadd myset 9 C

zincrby myset 5 A

zincrby myset 3 C

zadd dopset 9 B



Упорядоченные множества

zadd myset 5 A

zadd myset 2 B

zadd myset 9 C

zincrby myset 5 A

zincrby myset 3 C

zadd dopset 9 B

zunionstore newset 2 myset dopset aggregate sum

myset

2	10	12
B	A	C

dopset

9
B

newset

10	11	12
A	B	C

Упорядоченные множества

zadd myset 5 A

zadd myset 2 B

zadd myset 9 C

zincrby myset 5 A

zincrby myset 3 C

zadd dopset 9 B

zunionstore newset 2 myset dopset aggregate sum

zrange newset 1 2

myset

2	10	12
B	A	C

dopset

9
B

newset

10	11	12
A	B	C

newset

B
C

Списки

lpush <ключ> <значение> <значение> … – добавляет значение в начало списка

lpop <ключ> [<количество>] – получить и удалить первый элемент из списка

lrange <ключ> <минимальный индекс> <максимальный индекс> – получить список значений

rpush <ключ> <значение> <значение> … – добавляет значение в конец списка

rpop <ключ> [<количество>] – получить и удалить последний элемент из списка

Механизм подписок

subscribe <канал> – подписаться на канал

publish <канал> <сообщение> – отправить сообщение в канал

unsubscribe <канал> – отписаться

reset – выход из состояния ожидания

Python – создание и получение записи по ключу

```
import redis
client = redis.Redis(host=‘xxx.xxx.xxx.xxx', password=‘xxxxxx')
client.set('Language', 'Python', ex=5)
print(client.get('Language'))
client.close()
```

Python – получение сообщений

```
import redis
client = redis.Redis(host='xxx.xxx.xxx.xxx', password='xxxxxx')
pub = client.pubsub()
pub.subscribe(['firstchanel'])
count = 1
for item in pub.listen():
    print(count, ' -- ', item['data'])
    count += 1
    if count >6:
        pub.unsubscribe()
        break
```

Python – отправка сообщений

```
import redis
client = redis.Redis(host=‘xxx.xxx.xxx.xxx', password=‘xxxxxx')
client.publish('firstchanel', 'Hello!')
client.close()
```

MongoDB

Определения

База данных — это контейнер с коллекциями. У каждой базы данных есть свой собственный набор файлов в файловой системе

Коллекция — это группа документов MongoDB

Документ — это **набор** пар «ключ – значение». Документ имеет динамическую схему: не обязан иметь один одинаковый набор полей, а общие поля в коллекции могут иметь различные типы данных

_id — уникальный идентификатор документа MongoDB. Если новый документ не содержит поля _id, идентификатор создастся автоматически

Документ

```
{
  name : ‘Ivanov’,
  age : ‘21’,
  marks : [
                {title: ‘database’, level: ’10’},
                {title: ‘developer’, level: ’7’}
              ]
}
```

Документ

```
{
  name : ‘Petrov’,
  speciality : ‘designer’,
  projects : [
                {name: ‘White horse’, finish: ’02.12.20’}
              ]
}
```

Команды работы с базой данных

use <имя базы данных> – создание/переключение базы данных

show dbs – получить список баз данных

db.dropDatabase() – удалить базу данных

Команды работы с коллекцией

db.createCollection(<имя коллекции>) – создать коллекцию

show collections – получить список коллекций

db.ИМЯ_КОЛЛЕКЦИИ.drop() – удалить коллекцию

Команды работы с документами

db.ИМЯ_КОЛЛЕКЦИИ.insert(<документ>) – добавить документ в коллекцию

db. ИМЯ_КОЛЛЕКЦИИ.insertMany(<документ>, <документ>, …) – добавить несколько документов в коллекцию

db.ИМЯ_КОЛЛЕКЦИИ.remove(<критерий>) – удаление документа по критерию

db.ИМЯ_КОЛЛЕКЦИИ.update(<критерий>, {$set : <новые данные>}) – изменение документа по критерию

Запрос

db.ИМЯ_КОЛЛЕКЦИИ.find(<критерий>) – выбор данных, удовлетворяющих критерию

db.ИМЯ_КОЛЛЕКЦИИ.find().pretty()

Критерии

Операция	Синтаксис	Пример
Равно	{<ключ>: <значение>}	{name: ‘Ivanov’}
Меньше	{<ключ>: {$lt: <значение>}}	{age: {$lt: ‘22’}}
Меньше или равно	{<ключ>: {$lte: <значение>}}	{age: {$lte: ‘22’}}
Больше	{<ключ>: {$gt: <значение>}}	{age: {$gt: ‘22’}}
Больше или равно	{<ключ>: {$gte: <значение>}}	{age: {$gte: ‘22’}}
Не равно	{<ключ>: {$ne:<значение>}}	{name: {$ne: ‘Ivanov’}}
И	{$and: [{<ключ>:<значение>}, {<ключ>:<значение>}]}	{$and: [{age: {$gte: ‘18’}}, {age: {$lte: ‘25’}}]}
Или	{$or: [{<ключ>:<значение>}, {<ключ>:<значение>}]}	{$or: [{age: {$lt: ‘18’}}, {age: {$gt: ‘25’}}]}

Проекция

db.ИМЯ_КОЛЛЕКЦИИ.find({<критерий>}, {<атрибут>: 1, …}) – вывод значений перечисленных атрибутов

db.ИМЯ_КОЛЛЕКЦИИ.find({<критерий>}, {_id: 0, <атрибут>: 1, …})

Сортировка

db.ИМЯ_КОЛЛЕКЦИИ.find().sort({<атрибут>: 1, …}) – сортировка по возрастанию

db.ИМЯ_КОЛЛЕКЦИИ.find().sort({<атрибут>: -1, …}) – сортировка по убыванию

Агрегация

db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{$group : {<ключ>:<значение>, …, {<операция>: “$<ключ>”}}}])

Операция	Синтаксис
Сумма значений	$sum
Среднее значение	$avg
Минимальное значение	$min
Максимальное значение	$max
Первое значение	$first
Последнее значение	$last

Пример

```
use firstdb
db.createCollection(“student”)
db.student.insert({name : ‘Ivanov’, age : 19})
db.student.insert({name : ‘Petrov, age : 20, kurs: 2})
db.student.insert({name : ‘Sidorov’, spec: ’09.03.02’})
db.student.find().pretty()
db.student.update({name: ‘Petrov’}, {$set : {kurs : 3}})
db.student.find().pretty()
db.student.update({name: ‘Ivanov’}, {$set : {kurs : 3}})
db.student.find().pretty()
db.student.find({age: { $gte: 20 }},{_id:0, name:1, kurs:1}).pretty().sort({name:-1})
db.student.aggregate([{$group: {_id:”$kurs”, avg_age: {$avg: “$age”}}}])
```

MongoDB – запросы

Исходные данные

- {name : "Losev", kurs : 1, subjects : [{name : “Algebra”, mark : “4”}, {name : “Analiz”, mark : “4”}, {name : “Informatika”, mark : “5”}, {name : “English”, mark : “5”}], gruppa : 22105, spec : “ISIT”}
- {name : "Popov", kurs : 1, subjects : [{name : “Algebra”, mark : “5”}, {name : “Analiz”, mark : “5”}, {name : “Informatika”, mark : “5”}, {name : “English”, mark : “5”}], gruppa : 22105, spec : “ISIT”}
- {name : "Lobov", kurs : 2, subjects : [{name : “Algebra”, mark : “4”}, {name : “Analiz”, mark : “4”}, {name : “Informatika”, mark : “5”}, {name : “English”, mark : “5”}, {name : “Seti”, mark : “3”}, {name : “Istoriya”, mark : “4”}], gruppa : 22204, spec : “PMI”}
- {name : "Nemov", kurs : 2, subjects : [{name : “Algebra”, mark : “2”}, {name : “Analiz”, mark : “3”}, {name : “Informatika”, mark : “5”}, {name : “English”, mark : “4”}, {name : “Seti”, mark : “5”}, {name : “Istoriya”, mark : “4”}], gruppa : 22207, spec : “PI”}
- {name : "Bulkin", kurs : 2, subjects : [{name : “Algebra”, mark : “3”}, {name : “Analiz”, mark : “3”}, {name : “Informatika”, mark : “3”}, {name : “English”, mark : “4”}, {name : “Seti”, mark : “3”}, {name : “Istoriya”, mark : “3”}], gruppa : 22207, spec : “PI”, status : “publikaciya”}

```
from pymongo import MongoClient
client = MongoClient("mongodb://192.168.112.103")
db = client["student"]
db.create_collection("univer")
collection = db["univer"]
item = {"name" : "Losev", "kurs" : 1, "subjects" : [{"name" : "Algebra", "mark" : 4}, {"name" : "Analiz", "mark" : 4}], "gruppa" : "22105"}
collection.insert_one(item)
items = collection.find({"kurs" : {"$gte" : 2}},{"_id":0,"name": 1, "kurs": 1}).sort("name")
for item in items:
    print(item)
items.close()
client.close()
```

Получить информацию о студенте с именем "Lobov"

```
items = collection.find({"name" : "Lobov"})
for item in items:
    print(item)
```

{'_id': ObjectId('618e8fc5f9d0365ba240b2fa'), 'name': 'Lobov', 'kurs': 2, 'subjects': [{'name': 'Algebra', 'mark': 4}, {'name': 'Analiz', 'mark': 4}, {'name': 'Informatika', 'mark': 5}, {'name': 'English', 'mark': 5}, {'name': 'Seti', 'mark': 3}, {'name': 'Istoriya', 'mark': 4}], 'gruppa': '22204', 'spec': 'PMI'}

Получить информацию о студентах с оценкой по предмету "Istoriya"

collection.find({"subjects.name" : "Istoriya"})

{'_id': ObjectId('618e8fc5f9d0365ba240b2fa'), 'name': 'Lobov', 'kurs': 2, 'subjects': [{'name': 'Algebra', 'mark': 4}, {'name': 'Analiz', 'mark': 4}, {'name': 'Informatika', 'mark': 5}, {'name': 'English', 'mark': 5}, {'name': 'Seti', 'mark': 3}, {'name': 'Istoriya', 'mark': 4}], 'gruppa': '22204', 'spec': 'PMI'}

{'_id': ObjectId('618e8fc5f9d0365ba240b2fb'), 'name': 'Nemov', 'kurs': 2, 'subjects': [{'name': 'Algebra', 'mark': 2}, {'name': 'Analiz', 'mark': 3}, {'name': 'Informatika', 'mark': 5}, {'name': 'English', 'mark': 4}, {'name': 'Seti', 'mark': 5}, {'name': 'Istoriya', 'mark': 4}], 'gruppa': '22207', 'spec': 'PI'}

{'_id': ObjectId('618e8fc5f9d0365ba240b2fc'), 'name': 'Bulkin', 'kurs': 2, 'subjects': [{'name': 'Algebra', 'mark': 3}, {'name': 'Analiz', 'mark': 3}, {'name': 'Informatika', 'mark': 3}, {'name': 'English', 'mark': 4}, {'name': 'Seti', 'mark': 3}, {'name': 'Istoriya', 'mark': 3}], 'gruppa': '22207', 'spec': 'PI', 'status': 'publikaciya'}

Получить информацию о студентах с оценкой по предмету "Istoriya“ и состоят в группе 22207

collection.find({"subjects.name" : "Istoriya", "gruppa": "22207"})

{'_id': ObjectId('618e8fc5f9d0365ba240b2fb'), 'name': 'Nemov', 'kurs': 2, 'subjects': [{'name': 'Algebra', 'mark': 2}, {'name': 'Analiz', 'mark': 3}, {'name': 'Informatika', 'mark': 5}, {'name': 'English', 'mark': 4}, {'name': 'Seti', 'mark': 5}, {'name': 'Istoriya', 'mark': 4}], 'gruppa': '22207', 'spec': 'PI'}

{'_id': ObjectId('618e8fc5f9d0365ba240b2fc'), 'name': 'Bulkin', 'kurs': 2, 'subjects': [{'name': 'Algebra', 'mark': 3}, {'name': 'Analiz', 'mark': 3}, {'name': 'Informatika', 'mark': 3}, {'name': 'English', 'mark': 4}, {'name': 'Seti', 'mark': 3}, {'name': 'Istoriya', 'mark': 3}], 'gruppa': '22207', 'spec': 'PI', 'status': 'publikaciya'}

Найти номер группы для студента с именем "Lobov"

```
items = collection.find({"name" : "Lobov"},projection=({"_id":0, "gruppa":1}))
for item in items:
    print(item["gruppa"])
```

22204

items:
{'gruppa': '22204'}

Отсортировать всех студентов по фамилии

```
items = collection.find({},{"_id":0,"name": 1, "kurs": 1}).sort("name")
for item in items:
    print(item)
```

```
{'name': 'Bulkin', 'kurs': 2}
{'name': 'Lobov', 'kurs': 2}
{'name': 'Losev', 'kurs': 1}
{'name': 'Nemov', 'kurs': 2}
{'name': 'Popov', 'kurs': 1}
```

Получить отсортированный по имени список студентов 2-го курса и старше

```
items = collection.find({"kurs" : {"$gte" : 2}},{"_id":0,"name": 1, "kurs": 1}).sort("name")
for item in items:
    print(item)
```

```
{'name': 'Bulkin', 'kurs': 2}
{'name': 'Lobov', 'kurs': 2}
{'name': 'Nemov', 'kurs': 2}
```

Вывести оценки для студента с именем "Lobov"

```
items = collection.find({"name" : "Lobov"},projection=({"_id":0, "subjects":1}))
for item in items:
    sub = item["subjects"]
    for s in sub:
        print(s["name"], " - ", s["mark"])
```

```
Algebra  -  4
Analiz  -  4
Informatika  -  5
English  -  5
Seti  -  3
Istoriya  -  4
```

```
Items:
{'subjects': [{'name': 'Algebra', 'mark': 4},
{'name': 'Analiz', 'mark': 4},
{'name': 'Informatika', 'mark': 5},
{'name': 'English', 'mark': 5},
{'name': 'Seti', 'mark': 3},
{'name': 'Istoriya', 'mark': 4}]}
```

Вывести оценки по предмету "Seti" – обработка на уровне приложения

```
items = collection.find({"subjects.name": "Seti"})
for item in items:
    sub = item["subjects"]
    for s in sub:
        if s["name"] == "Seti":
            print(item["name"], " - ", s["mark"])
```

```
Lobov  -  3
Nemov  -  5
Bulkin  -  3
```

Вывести оценки по предмету "Seti" – обработка на уровне СУБД

```
items = collection.aggregate( [ { "$unwind" : "$subjects" },
{"$match": {"subjects.name" : "Seti"}} ,
{ "$replaceRoot": {"newRoot": {"name":"$name", "mark" :
"$subjects.mark" }}} ])
for item in items:
    print(item)
```

```
{'name': 'Lobov', 'mark': 3}
{'name': 'Nemov', 'mark': 5}
{'name': 'Bulkin', 'mark': 3}
```

unwind

```
items = collection.aggregate( [ { "$unwind" : "$subjects" }])
for item in items:
    print(item)
```

{'_id’: …, 'name': 'Lobov', 'kurs': 2, 'subjects': {'name': 'Algebra', 'mark': 4}, 'gruppa': '22204', 'spec': 'PMI'}
{'_id': …, 'name': 'Lobov', 'kurs': 2, 'subjects': {'name': 'Analiz', 'mark': 4}, 'gruppa': '22204', 'spec': 'PMI'}
{'_id': …, 'name': 'Lobov', 'kurs': 2, 'subjects': {'name': 'Informatika', 'mark': 5}, 'gruppa': '22204', 'spec': 'PMI'}
{'_id': …, 'name': 'Lobov', 'kurs': 2, 'subjects': {'name': 'English', 'mark': 5}, 'gruppa': '22204', 'spec': 'PMI'}
{'_id': …, 'name': 'Lobov', 'kurs': 2, 'subjects': {'name': 'Seti', 'mark': 3}, 'gruppa': '22204', 'spec': 'PMI'}
{'_id': …, 'name': 'Lobov', 'kurs': 2, 'subjects': {'name': 'Istoriya', 'mark': 4}, 'gruppa': '22204', 'spec': 'PMI'}
{'_id': …, 'name': 'Nemov', 'kurs': 2, 'subjects': {'name': 'Algebra', 'mark': 2}, 'gruppa': '22207', 'spec': 'PI'}

unwind, match

```
items = collection.aggregate( [ { "$unwind" : "$subjects" }, {"$match": {"subjects.name" : "Seti"}} ])
for item in items:
    print(item)
```

{'_id': ObjectId('618e8fc5f9d0365ba240b2fa'), 'name': 'Lobov', 'kurs': 2, 'subjects': {'name': 'Seti', 'mark': 3}, 'gruppa': '22204', 'spec': 'PMI'}

{'_id': ObjectId('618e8fc5f9d0365ba240b2fb'), 'name': 'Nemov', 'kurs': 2, 'subjects': {'name': 'Seti', 'mark': 5}, 'gruppa': '22207', 'spec': 'PI'}

{'_id': ObjectId('618e8fc5f9d0365ba240b2fc'), 'name': 'Bulkin', 'kurs': 2, 'subjects': {'name': 'Seti', 'mark': 3}, 'gruppa': '22207', 'spec': 'PI', 'status': 'publikaciya'}

unwind, match, replaceRoot

```
items = collection.aggregate( [ { "$unwind" : "$subjects" },
{"$match": {"subjects.name" : "Seti"}} ,
{ "$replaceRoot": {"newRoot": {"name":"$name", "mark" :
"$subjects.mark" }}} ])
for item in items:
    print(item)
```

```
{'name': 'Lobov', 'mark': 3}
{'name': 'Nemov', 'mark': 5}
{'name': 'Bulkin', 'mark': 3}
```

Получить количество студентов в каждой группе

```
items = collection.aggregate([{"$group": {"_id": "$gruppa", "count" : {"$sum": 1}}}])
for item in items:
    print(item)
```

```
{'_id': '22207', 'count': 2}
{'_id': '22204', 'count': 1}
{'_id': '22105', 'count': 2}
```

Количество студентов на 2-м курсе и старше

```
items = collection.aggregate([{"$match": {"kurs": {"$gte": 2}}},
{"$count": "students"}])

for item in items:
    print(item)
```

```
{'students': 3}
```

Получить для каждого студента его средний балл

```
items = collection.aggregate( [ { "$unwind" : "$subjects" }, {"$group": {"_id": "$name", "avg_mark" : {"$avg": "$subjects.mark"}}} ] )
for item in items:
    print(item)
```

```
{'_id': 'Bulkin', 'avg_mark': 3.166666666666665}
{'_id': 'Nemov', 'avg_mark': 3.833333333333335}
{'_id': 'Lobov', 'avg_mark': 4.166666666666667}
{'_id': 'Popov', 'avg_mark': 5.0}
{'_id': 'Losev', 'avg_mark': 4.5}
```

Дополнительные данные

```
item = {"year" : 2, "semester" : [3, 4]}
collection.insert_one(item)
```

Получить имена студентов 2-го курса и семестры обучения

```
items = collection.aggregate([
{"$match": {"kurs" : 2}},
{"$lookup" : {"from" : "univer", "localField" : "kurs", "foreignField" : "year",
"as" : "s"}},
{"$replaceRoot": {"newRoot" : {"name" : "$name", "sem" : "$s.semester"}}},
{"$project" : {"_id" : 0, "name" : 1, "sem" : 1}}    ])
```

```
{'name': 'Lobov', 'sem': [[3, 4]]}
{'name': 'Nemov', 'sem': [[3, 4]]}
{'name': 'Bulkin', 'sem': [[3, 4]]}
```

Дополнительные данные

{"box" : 1, "params" : {"size":12, "length": 20}}
{"box" : 2, "params" : {"size":10, "width": 30}}

Получить все названия параметров коробок (имена ключей вложенного документа)

items = collection.aggregate([
{"$match": {"box" : {"$exists": "true"}}}, выбираем только коробки
{"$project":{"arrayofkeyvalue":{"$objectToArray":"$$ROOT.params"}}}, создаем новый ключ arrayofkeyvalue – множество ключ (k) и значение (v) для каждого ключа вложенного документа params
{"$unwind":"$arrayofkeyvalue"}, разбиваем множество на отдельные документы
{"$replaceRoot": { "newRoot": "$arrayofkeyvalue" }}, переносим вложенный документ arrayofkeyvalue в корень
{"$group":{"_id":"k","allkeys":{"$addToSet":"$k"}}}, группируем по ключу k
{"$project": { "_id": 0, "allkeys": 1}}]) оставляем только названия параметров

Обновление данных

Изменить значение поля:
collection.update_one({"name" : "Lobov"}, {"$set":{"name" : "Lukov"}})

Добавить новое поле:
collection.update_one({"name" : "Lobov"}, {"$set":{"pass" : "ooo"}})

Удалить поле:
collection.update_one({"name" : "Lobov"}, {"$unset":{"pass" : ""}})

Удалить документ:
collection.delete_one({"year" : 5})

Neo4j

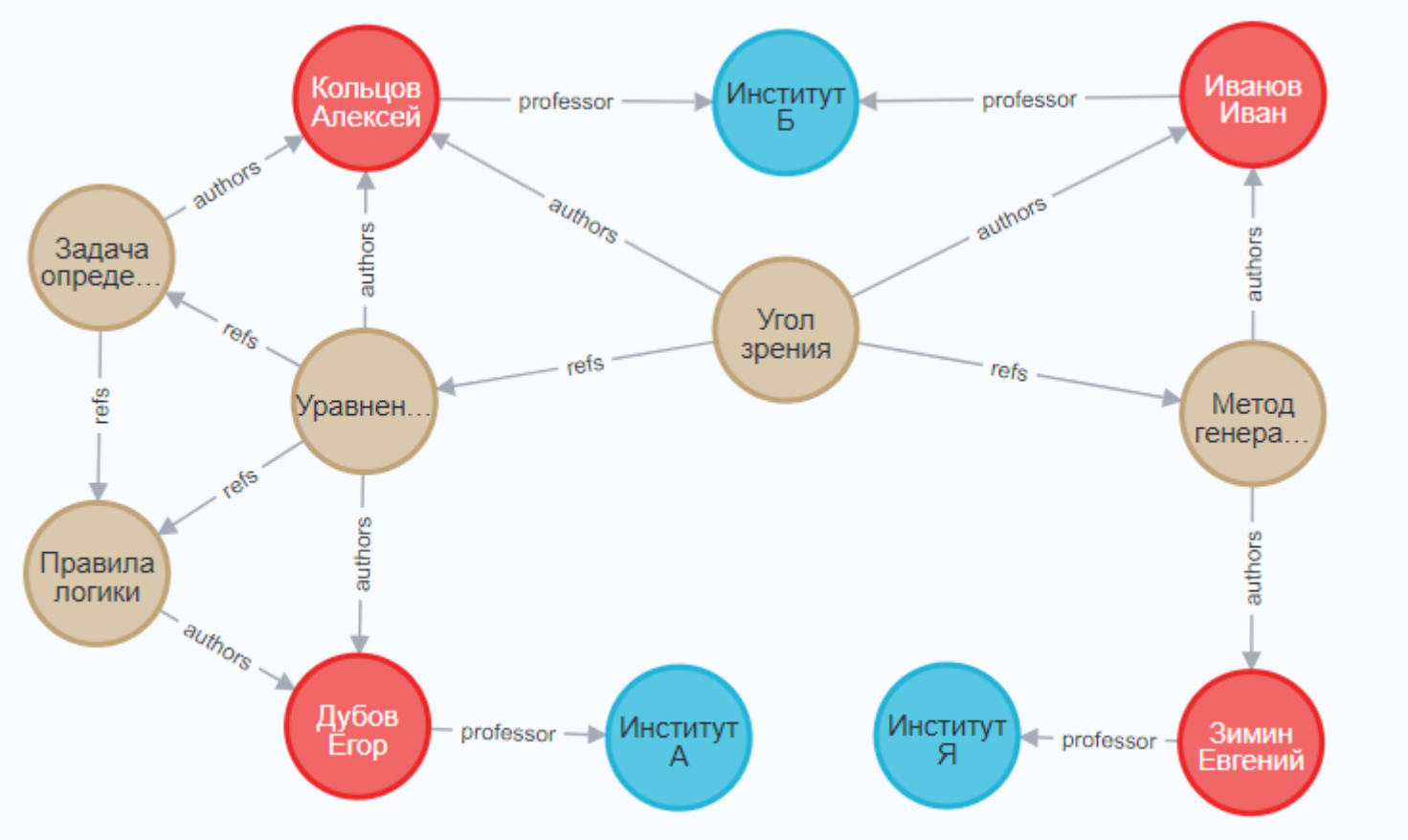
Описание

- Структура – граф:
  - Узел:
    - Метка (роль)
    - Свойства (ключ – значение)
  - Взаимосвязь:
    - Тип
    - Направление
    - Свойства (ключ – значение)
- Язык запросов – Cypher

Учебная база данных Science

Nodes	12
Relationships	16
Labels	3
Relationship Types	3
Property Keys	10

Учебная база данных Science

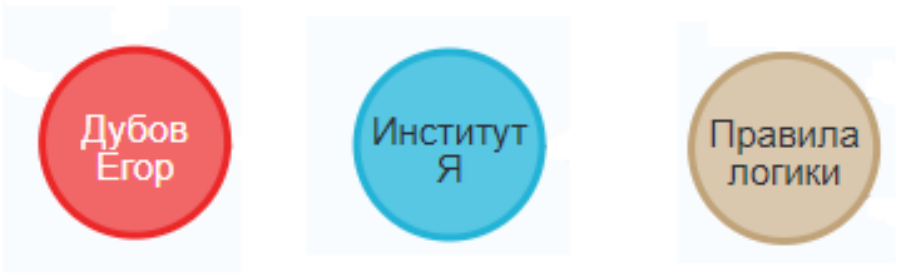


Создание узла

- create(:Author{name: 'Дубов Егор', if: 2})
- create(:Org{name: 'Институт Я', city: 'Тверь'})
- create(:Paper{title: 'Правила логики', year: 2021, journal: 'Математика', number: 3})

Результат

{ "identity": 3, "labels": [
"Author"], "properties": {
"name": "Дубов Егор", "if": 2 }
}
{ "identity": 9, "labels": ["Org"
], "properties": { "city":
"Тверь", "name": "Институт Я"
} }
{ "identity": 10, "labels": [
"Paper"], "properties": {
"number": 3, "journal":
"Математика", "year": 2021,
"title": "Правила логики" } }



Шаблоны для запросов

- () – узел
- (a) – узел
- (a: Paper) – узел с меткой Paper
- (a: Paper {title: “Title”}) – узел с меткой и значением ключа
- Типы значений: строка, число, логическое значение, коллекция

- (a)-->(b) – существует связь от узла a до узла b
- (a)--(b) – существует связь между узлами в любом направлении
- (a)-[r:refs]-(b) – существует связь типа refs между узлами
- (a)-[*]-(b) – существует связь между узлами любой глубины
- (a)-[*1..5]-(b) – существует связь между узлами длиной от 1 до 5

- p= (a)-[r]-(b) – путь между двумя узлами

Путь

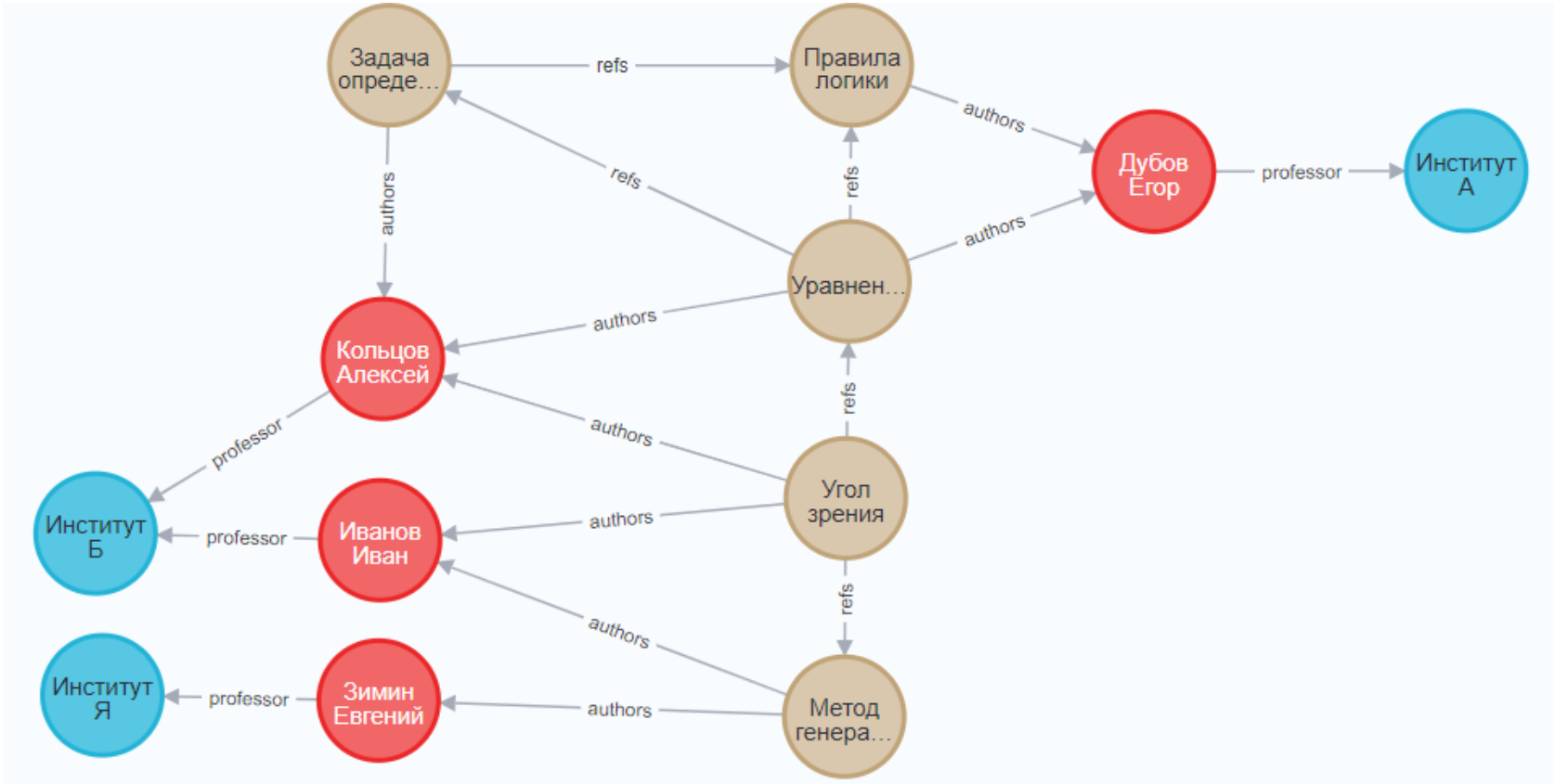
- nodes(p) – узлы в пути
- relationships(p) – дуги в пути
- shortestPath((a)-[]-(b))

Функции

- id(node)
- a.property
- has(a.property)
- id(relationship)
- type(relationship)
- startNode(relationship)
- endNode(relationship)

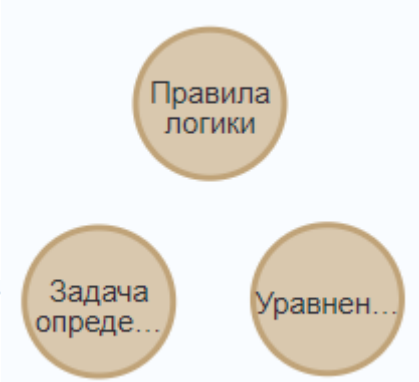
Запрос на выборку данных

match (a) return a



Запрос на выборку данных по значению ключа (вернуть узел)

match (a:Paper{year:2021}) return a




```
science$ match (a:Paper{year:2021}) return a
```

Graph

Table

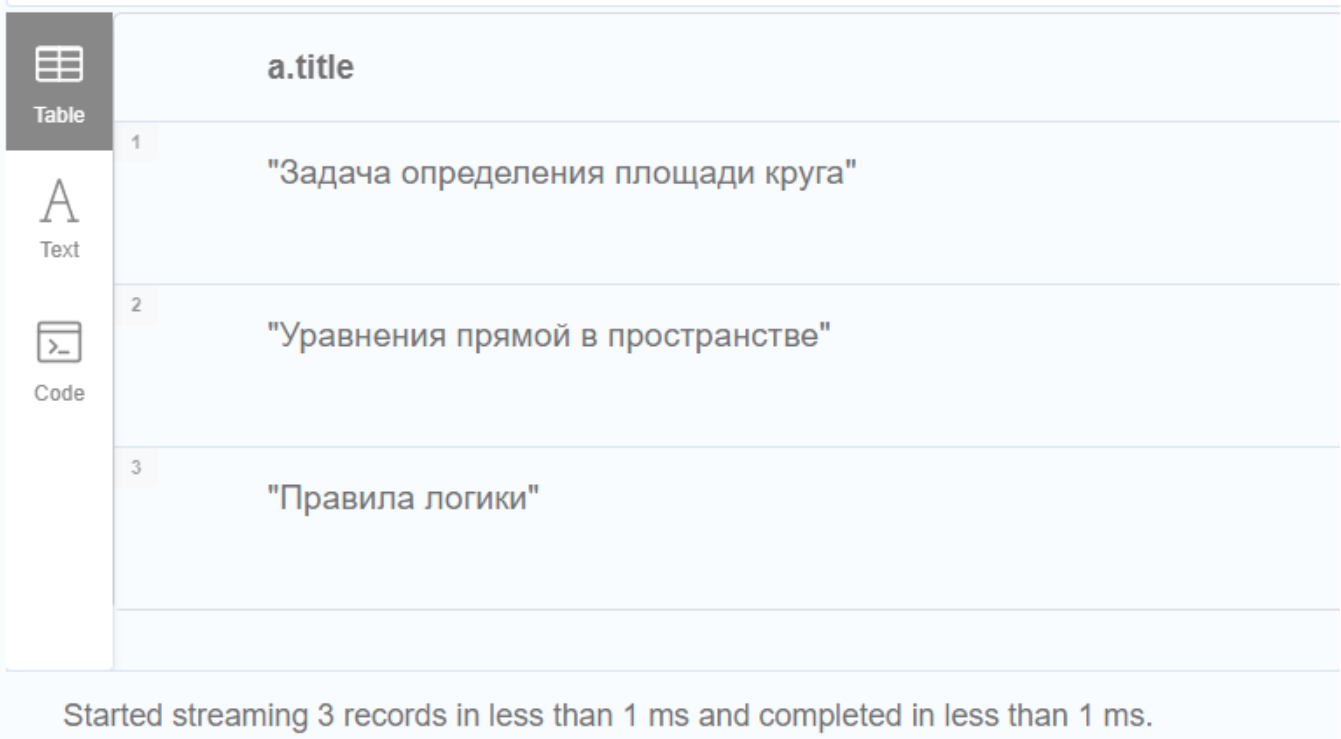
Text

Code

```
╒═══════════════════════════════════════════════════════════════════════════════╕
│"a"                                                                            │
╞═══════════════════════════════════════════════════════════════════════════════╡
│{"number":1,"journal":"Математика","year":2021,"title":"Задача определ│
│ения площади круга"}                                                           │
├───────────────────────────────────────────────────────────────────────────────┤
│{"number":2,"journal":"Математика","year":2021,"title":"Уравнения прям│
│ой в пространстве"}                                                            │
├───────────────────────────────────────────────────────────────────────────────┤
│{"number":3,"journal":"Математика","year":2021,"title":"Правила логики│
│"}                                                                             │
└───────────────────────────────────────────────────────────────────────────────┘
```

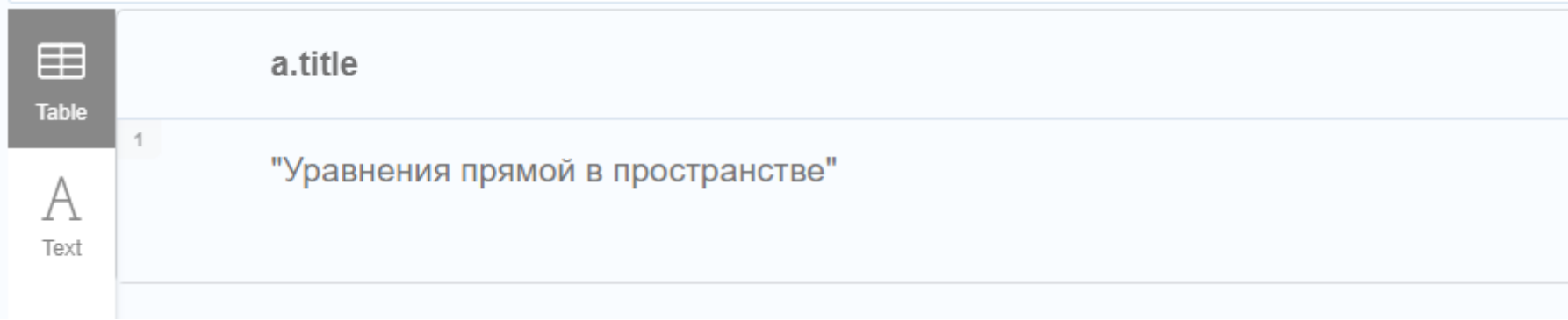
Запрос на выборку значений ключа (вернуть свойство узла)

```
match (a:Paper{year:2021}) return a.title
```



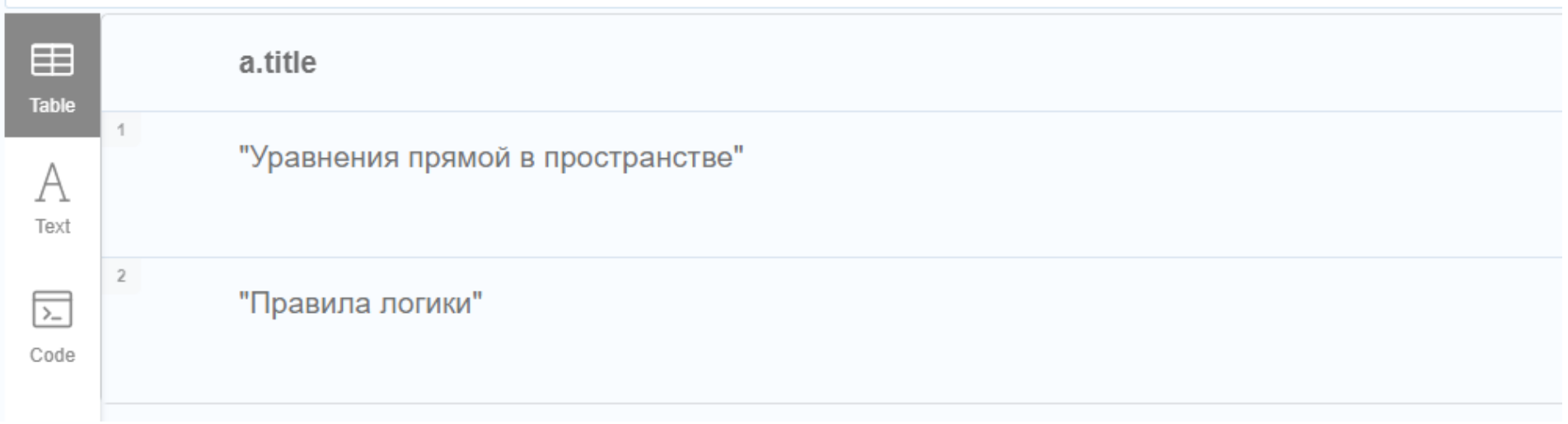
Запрос на выборку значений ключа (два условия)

```
match (a:Paper{year:2021, number:2})
return a.title
```



Запрос на выборку значений ключа (условие where)

```
match (a:Paper{year:2021})
where a.number > 1
return a.title
```



Создание связи

```
match (a:Paper {title: 'Правила логики'}),
(b:Author {name :  'Дубов Егор' })
create (a)-[r:authors{contribution: 1}]->(b)
```

```
match (a:Paper), (b:Author)
where a.title='Правила логики' AND b.name='Дубов Егор'
create (a)-[r:authors{contribution: 1}]->(b)
```

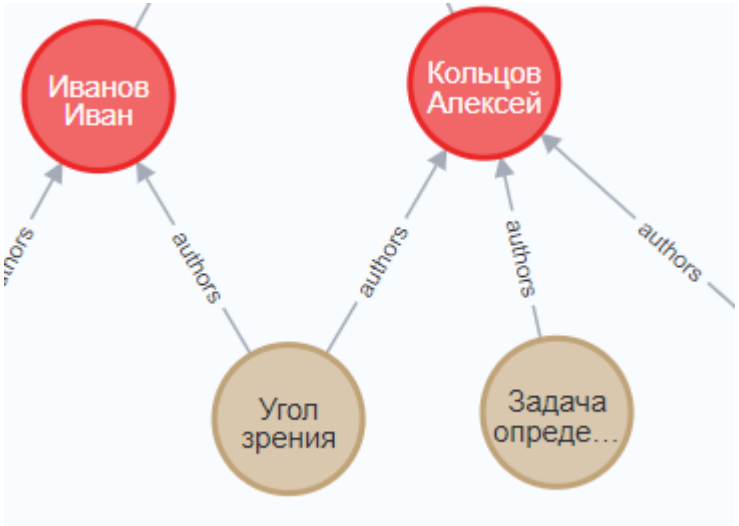
```
match (a:Paper {title : 'Угол зрения'}), (b:Paper {title : ‘Уравнения прямой в пространстве’})
create (a)-[r:refs{number: 1}]->(b)
```

```
match (a:Org {name: ‘Институт Б’}), (b:Author{name : 'Кольцов Алексей'})
create (a)<-[r:professor{contribution: 0.5}]-(b)
```

Результат

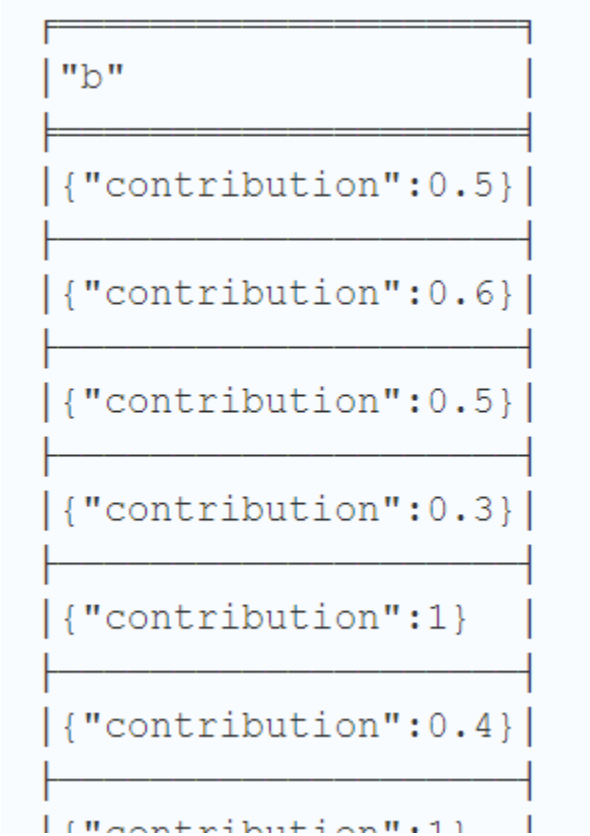
{ "identity": 11, "start": 10, "end": 3, "type": "authors", "properties": { "contribution": 1 } }

{ "identity": 10, "start": 11, "end": 1, "type": "authors", "properties": { "contribution": 0.5 } }



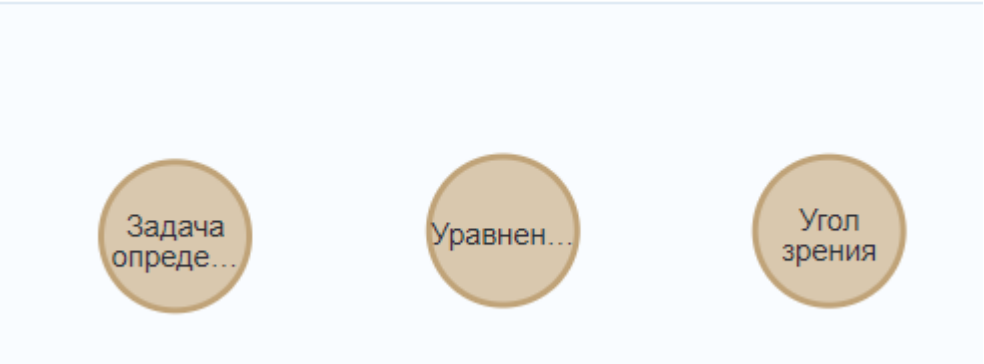
Запрос на выборку связей (вернуть связь)

match (a)-[b : authors]-(c) return b



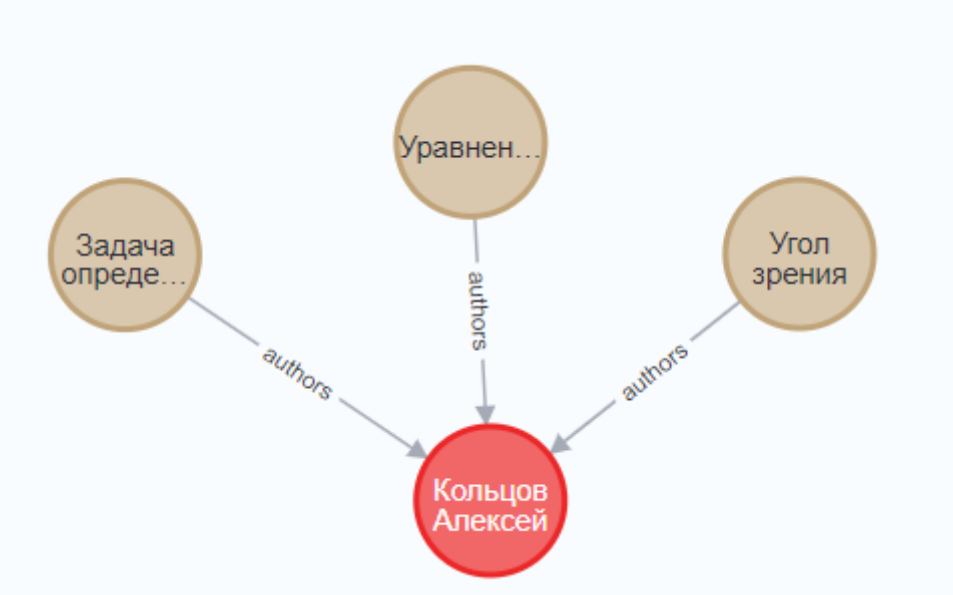
Получить статьи автора «Кольцов Алексей»

```
match (a:Author {name: "Кольцов Алексей"})-[b:authors]-(c) return c
```



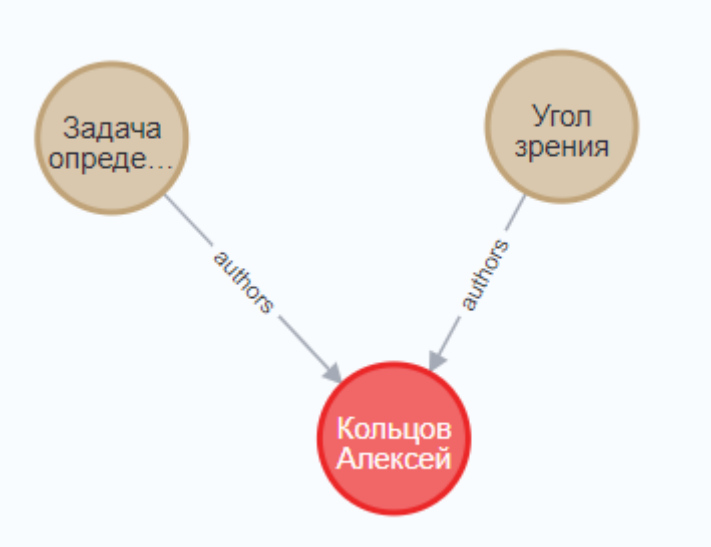
Получить статьи автора «Кольцов Алексей» со связями

match (a:Author {name: "Кольцов Алексей"})-[b:authors]-(c) return a, c



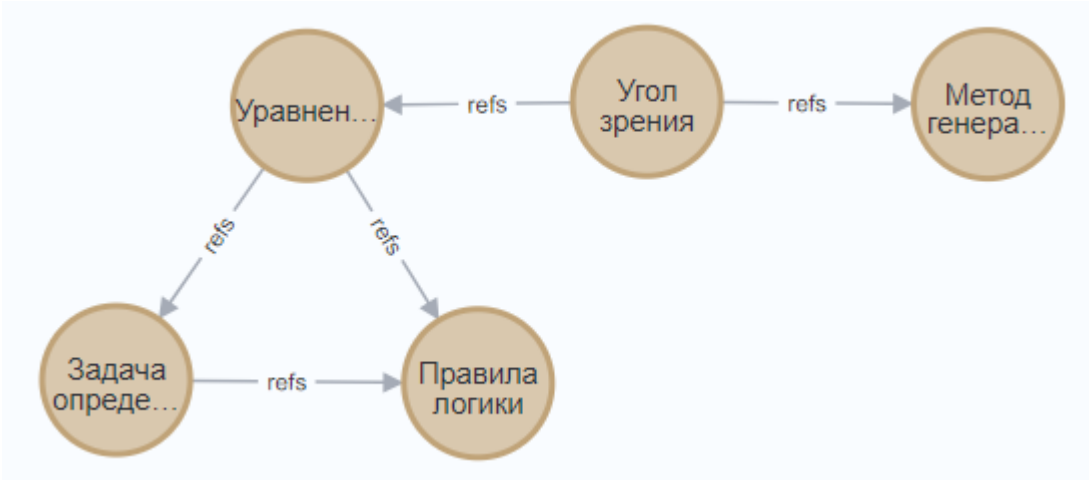
Получить статьи автора «Кольцов Алексей» с вкладом более 0,3 со связями

```
match (a:Author {name: "Кольцов Алексей"})-
[b:authors]-(c:Paper)
where b.contribution > 0.3
return a, c
```



Получить статьи, на которые ссылается статья «Угол зрения» со всеми следующими ссылками

```
match (a:Paper {title: 'Угол зрения'})-[r:refs*]->(b:Paper)
return a, b
```



Запрос на все связи

```
match (a{title: 'Угол зрения'})-[r*]-(b)
return a, b
```

Результат – вся база данных, если граф связный

Алиас

match (a:Paper)-[:authors]->(b:Author)-[:professor]->(c:Org) return a.title as Paper

"Paper"
"Правила логики"
"Уравнения прямой в пространстве"
"Угол зрения"
"Метод генерации столбцов"
"Угол зрения"
"Уравнения прямой в пространстве"
"Задача определения площади круга"
"Метод генерации столбцов"

Уникальные записи

match (a:Paper)-[:authors]->(b:Author)-[:professor]->(c:Org) return distinct a.title as Paper

"Paper"
"Правила логики"
"Уравнения прямой в пространстве"
"Угол зрения"
"Метод генерации столбцов"
"Задача определения площади круга"

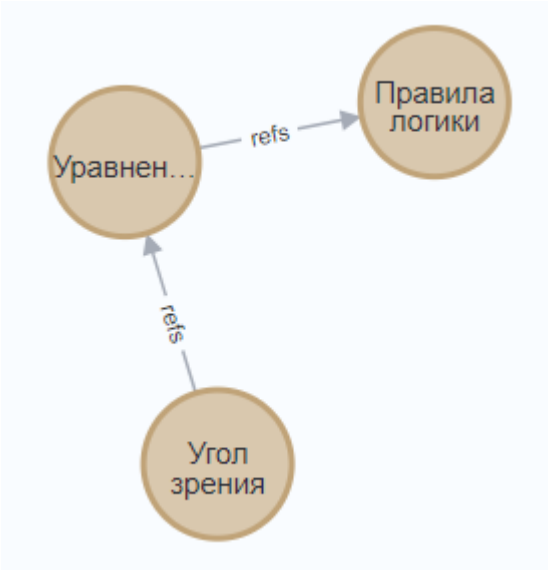
Сортировка

match (a:Paper)-[:authors]->(b:Author)-[:professor]->(c:Org) return distinct a.title as Paper order by Paper

"Paper"
"Задача определения площади круга"
"Метод генерации столбцов"
"Правила логики"
"Угол зрения"
"Уравнения прямой в пространстве"

Получить самую короткую последовательность между двумя статьями

```
match (a:Paper{title:'Угол зрения'}),
(b:Paper{title:'Правила логики'}),
s=shortestPath((a)-[*..5]-(b))
return s
```



Получить последовательность ссылок между двумя статьями длиной 2

match p=(a:Paper{title:'Угол зрения'})-[r:refs*2]-(b:Paper{title: 'Правила логики’}) return p

Путь:

[{"number":7, "journal":"Право", "year":2020, "title":"Угол зрения"},
{"num":2},
{"number":2, "journal":"Математика", "year":2021, "title":"Уравнения прямой в пространстве"},
{"number":2, "journal":"Математика", "year":2021, "title":"Уравнения прямой в пространстве"},
{"num":2},
{"number":3, "journal":"Математика", "year":2021, "title":"Правила логики"}]

Получить последовательность ссылок между двумя статьями длиной 2

```
match p=(a:Paper{title:'Угол зрения'})-
[r:refs*2]-(b:Paper{title:'Правила логики'})
return relationships(p)
```

Результат:

[{"num":2},{"num":2}]

Получить номера ссылок, их сумму

```
match p=(a:Paper{title:'Угол зрения'})-[*2]-
(b:Paper{title:'Правила логики’})
with relationships(p) as r
unwind r as rx
return rx.num as Number
```

Number
2
2

```
match p=(a:Paper{title:'Угол зрения'})-[*2]-
(b:Paper{title:'Правила логики'}) with relationships(p)
as r unwind r as rx return sum(rx.num) as SumNumbers
```

Результат: 4

Получить последовательность ссылок между двумя статьями длиной 2

```
match p=(a:Paper{title:'Угол зрения'})-[*2]-
(b:Paper{title:'Правила логики’})
return nodes(p)
```


[{"number":7, "journal":"Право", "year":2020,
"title":"Угол зрения"},

{"number":2, "journal":"Математика",
"year":2021, "title":"Уравнения прямой в
пространстве"},

{"number":3, "journal":"Математика",
"year":2021, "title":"Правила логики"}]

Получить последовательность ссылок между двумя статьями длиной 2

```
match p=(a:Paper{title:'Угол зрения'})-[*2]-
(b:Paper{title:'Правила логики’})
with nodes(p) as r
unwind r as rx
return rx
```

rx
{"number":7,"journal":"Право","year":2020,"title":"Угол зрения"}
{"number":2,"journal":"Математика","year":2021,"title":"Уравнения прямой в пространстве"}
{"number":3,"journal":"Математика","year":2021,"title":"Правила логики"}

Получить статьи, имеющие более одной ссылки

```
match (a:Paper)-[]->(b:Paper)
with a, count(b) as cb
where cb>1
return a.title, cb
```

a.title	cb
"Угол зрения"	2
"Уравнения прямой в пространстве"	2

Получить ФИО авторов статей, имеющих две ссылки

```
match (au:Author)<--(a:Paper)-->(b:Paper)
with a, au, count(b) as cn
where cn>1
return distinct au.name
```

au.name
"Кольцов Алексей"
"Иванов Иван"
"Дубов Егор"

Получить ФИО авторов статей, имеющих две ссылки

```
match (au:Author)<--(a:Paper)-[]->(b:Paper)
with a, au, count(b) as cn
where cn>1
return au.name, a.title
```

au.name	a.title
"Кольцов Алексей"	"Угол зрения"
"Иванов Иван"	"Угол зрения"
"Дубов Егор"	"Уравнения прямой в пространстве"
"Кольцов Алексей"	"Уравнения прямой в пространстве"

Получить для каждой статьи количество ссылок в ней

match (a:Paper)-[]->(b:Paper) with a, count(b) as bcount return a.title, bcount

"a.title"	"bcount"
"Угол зрения"	2
"Уравнения прямой в пространстве"	2
"Задача определения площади круга"	1

Получить для каждой статьи количество ссылок на нее

```
match (a:Paper)-[]->(b:Paper) with b, count(a) as acount return b.title, acount
```

"b.title"	"acount"
"Метод генерации столбцов"	1
"Задача определения площади круга"	1
"Уравнения прямой в пространстве"	1
"Правила логики"	2

Получить для каждой статьи самую длинную последовательность ссылок

```
match p=(a:Paper)-[:refs*]->(b:Paper)
with a, max(length(p)) as lp
call{with a, lp
match pp = (a)-[:refs*]->(:Paper)
where length(pp) = lp
return a as a1, nodes(pp) as np}
with a1, np
call{with a1, np
unwind np as x
return a1 as a2, collect(x.title) as ptitle}
return a2.title, ptitle
```

Результат

a2.title	ptitle
Задача определения площади круга	Задача определения площади круга, Правила логики
Уравнения прямой в пространстве	Уравнения прямой в пространстве, Задача определения площади круга, Правила логики
Угол зрения	Угол зрения, Уравнения прямой в пространстве, Задача определения площади круга, Правила логики

Получить суммарный вклад для каждого автора

```
match (a:Paper)-[r:authors]->(b:Author)
with b, sum(r.contribution) as bsum
return b.name, bsum
```

"b.name"	"bsum"
"Иванов Иван"	1.1
"Кольцов Алексей"	1.8
"Зимин Евгений"	0.4
"Дубов Егор"	1.7

Получить количество статей, написанных авторами института

```
match (a:Paper)-[r:authors]->(b:Author)-
[s:professor]->(c:Org)
with distinct c, a
return c.name, count(c)
```

"c.name"	"count(c)"
"Институт А"	2
"Институт Б"	4
"Институт Я"	1

Получить количество ссылок на каждого автора

```
match (a:Paper)-[:refs]->(:Paper)-[r:authors]->(b:Author)
with distinct b, a
return b.name, count(a)
```

"b.name"	"count(a)"
"Зимин Евгений"	1
"Иванов Иван"	1
"Кольцов Алексей"	2
"Дубов Егор"	3

Удаление связей и узлов

- Сначала удалить все связи с узлом
- Потом удалить узел

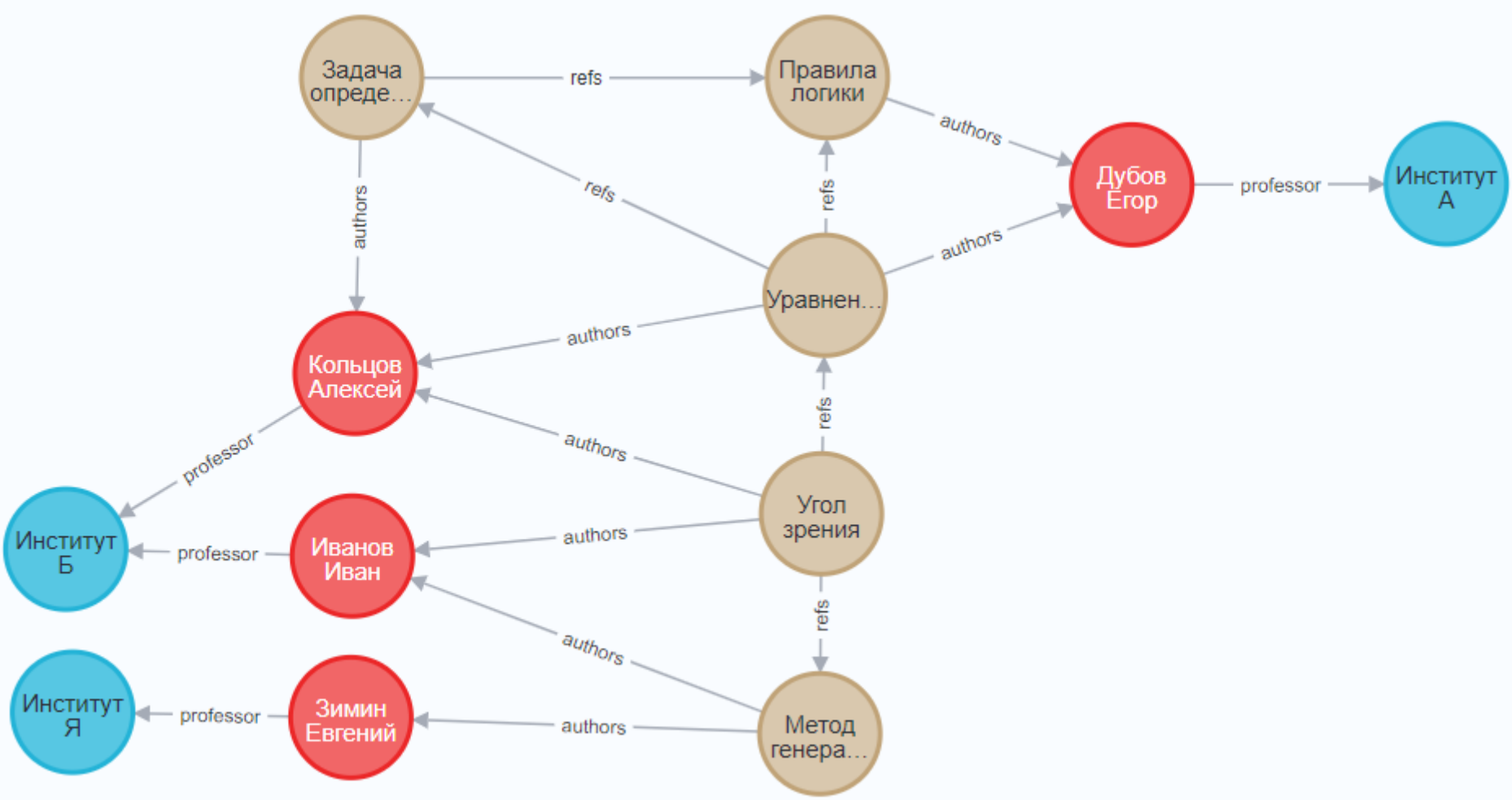
```
match (a:Paper{title: 'У'})-[r]->(b:Author{name:
'Кольцов Алексей'})
 delete r, a
```

```
match (a:Paper) where a.title='У'
delete a
```

Обновление узлов и связей

```
match (a:Paper{title: 'Угол зрения'})
set a.number = 8, a.downloads = 30
```

```
match (a:Paper{title: 'Угол зрения'})
-[r:authors]->
(b:Author{name: 'Кольцов Алексей'})
set r.contribution = 0.8
```


Задача опреде…
refs
Правила логики
authors
Дубов Егор
professor
Институт А
authors
refs
refs
authors
Уравнен…
authors
Кольцов Алексей
professor
authors
refs
Институт Б
professor
Иванов Иван
authors
Угол зрения
refs
authors
Институт Я
professor
Зимин Евгений
authors
Метод генера…

Пример БД

tblPerson

intPassport	txtName
1	Иванов Василий
2	Петров Сергей
3	Иванов Антон
4	Иванова Екатерина
5	Петрова Ольга
6	Серова Ирина

tblPhone

intPassport	intPhoneNumber
1	123
1	234
2	345
2	456
3	567
4	678

tblChild

intPassport	intChildPassport
1	3
1	4
2	5
4	6

ЗАПРОС: Получить номера телефонов внуков Василия Иванова

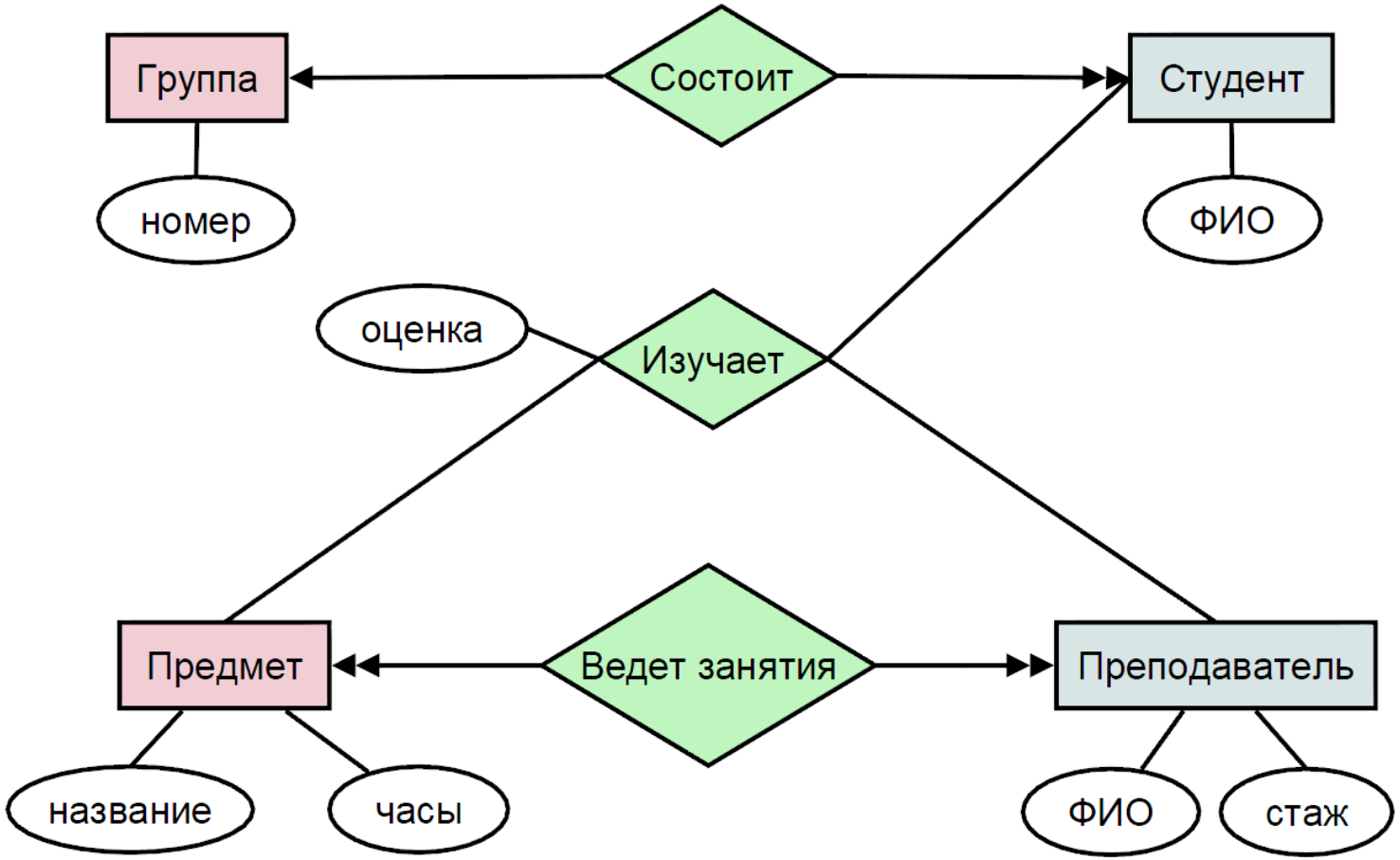
SQL

Select tblPhone.intPhoneNumber

From tblPhone, tblPerson, tblChild as A, tblChild as B

Where (tblPerson.intPassport = A.intPassport) and (A.intChaildPassort = B.intPassport) and (B.intChildPassport = tblPhone.intPassport) and (tblPerson.txtName = ’Иванов Василий’)

Инфологическая модель



Связь «один-ко-многим» – вариант 1

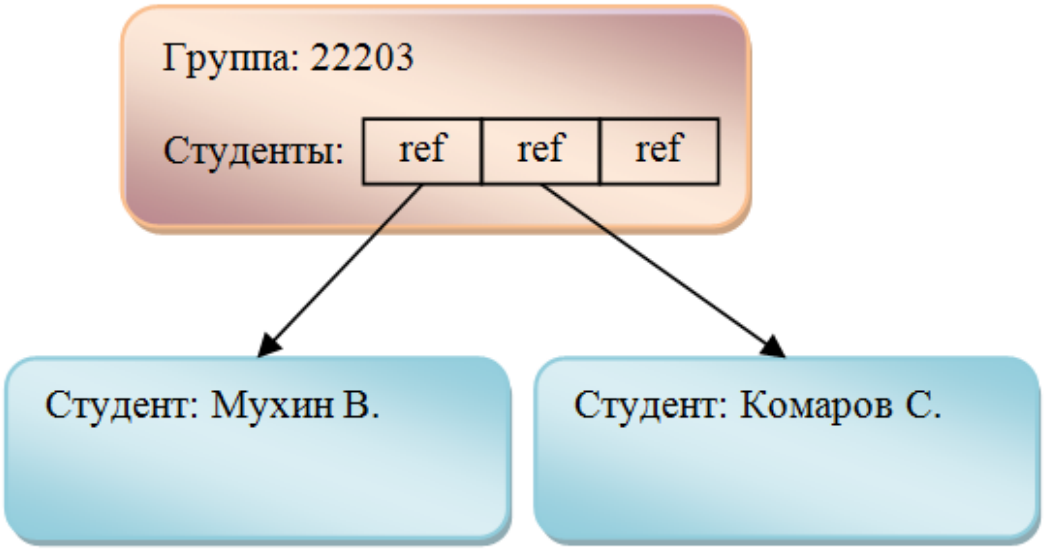
{_id: <ObjectId1>,
GroupName: 22203,
Students: [

{FirstName: Влад, LastName: Мухин},

{FirstName: Сергей, LastName: Комаров}

]}

Связь «один-ко-многим» – вариант 2

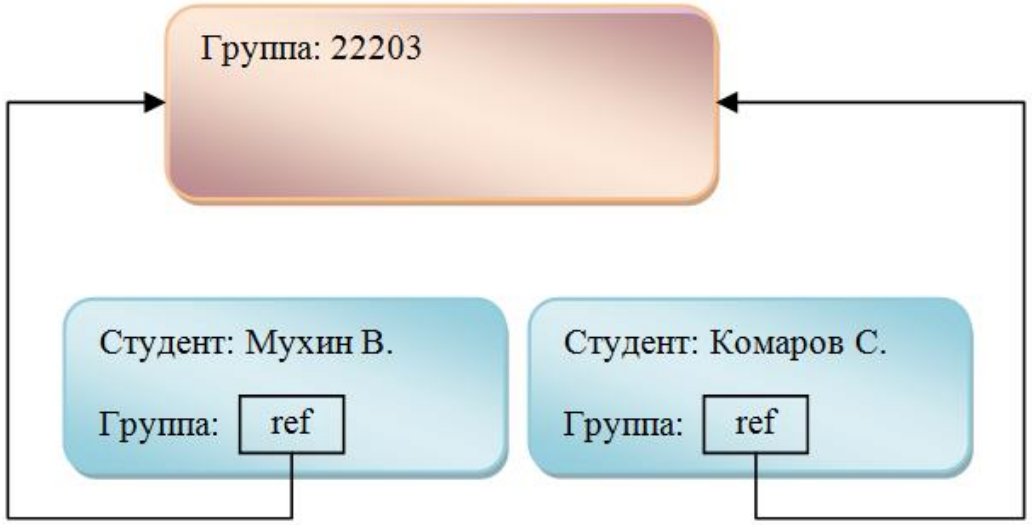


{_id: <ObjectId1>,
GroupName: 22203,
Students: [<ObjectId2>, <ObjectId3>]}

{_id: <ObjectId2>,
FirstName: Влад,
LastName: Мухин}

{_id: <ObjectId3>,
FirstName: Сергей,
LastName: Комаров}

Связь «один-ко-многим» – вариант 3

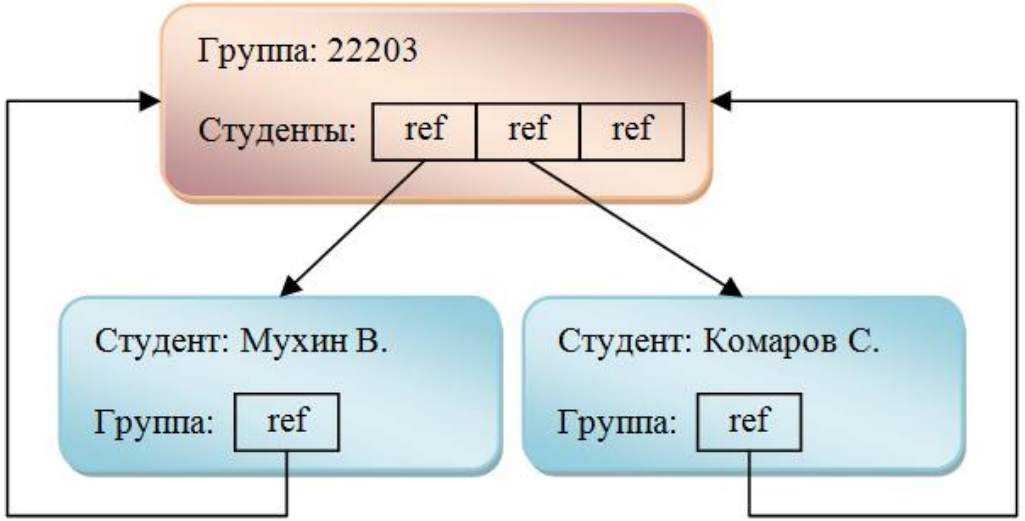


{_id: <ObjectId1>,
GroupName: 22203}

{_id: <ObjectId2>,
FirstName: Влад,
LastName: Мухин,
Group: <ObjectId1>}

{_id: <ObjectId3>,
FirstName: Сергей,
LastName: Комаров,
Group: <ObjectId1>}

Связь «один-ко-многим» – вариант 4

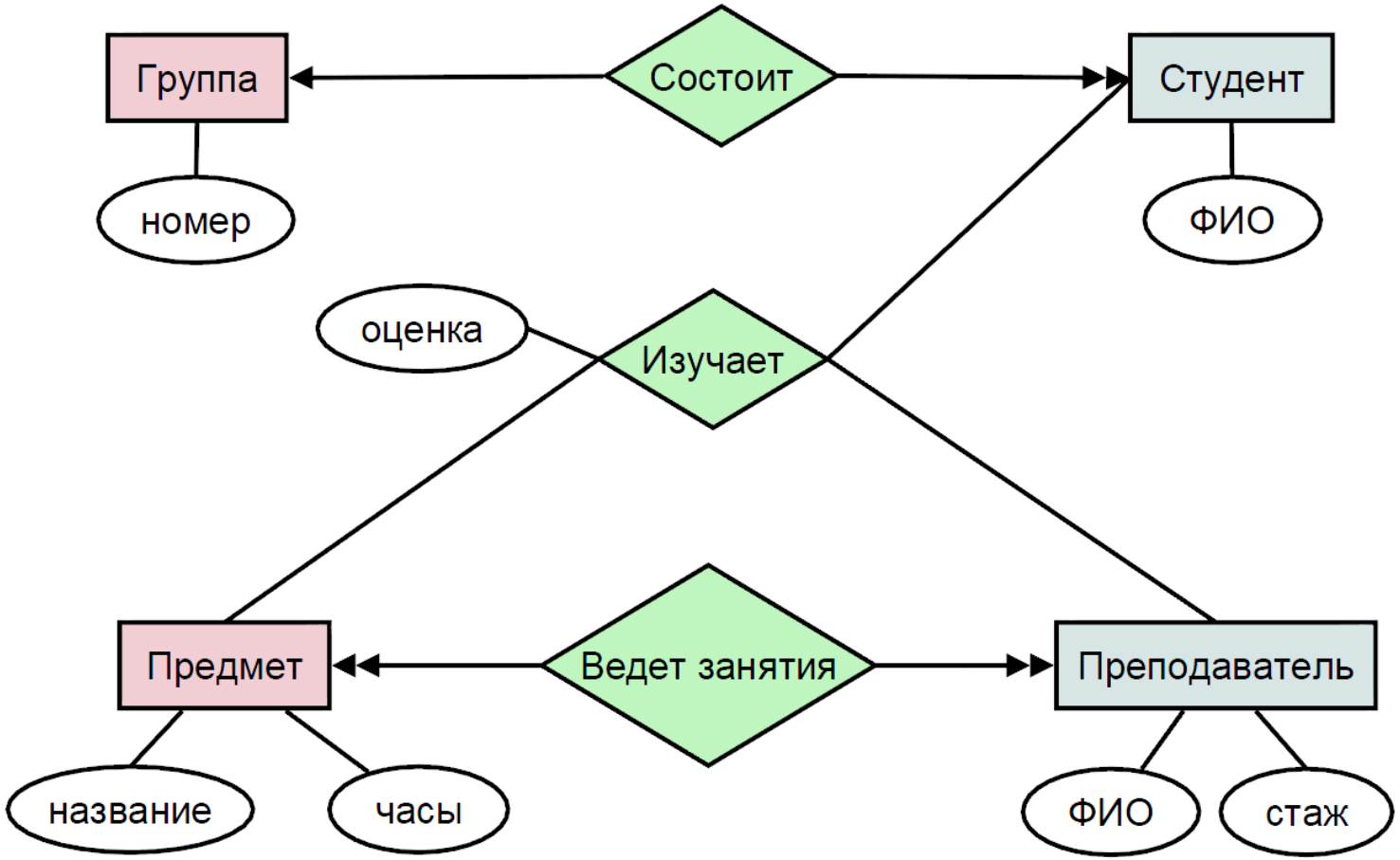


{_id: <ObjectId1>,
GroupName: 22203,
Students: [<ObjectId2>, <ObjectId3>]}

{_id: <ObjectId2>,
FirstName: Влад,
LastName: Мухин,
Group: <ObjectId1>}

{_id: <ObjectId3>,
FirstName: Сергей,
LastName: Комаров,
Group: <ObjectId1>}

Инфологическая модель



Связь «многие-ко-многим» – вариант 1

{_id: <ObjectId10>,
CourseName: Базы данных,
Hours: 44
Teachers: [

{FirstName: Сергей Иванович,
LastName: Булкин},

{FirstName: Андрей Павлович,
LastName: Перов}

]}

{_id: <ObjectId20>,
CourseName: Проектирование ИС,
Hours: 44
Teachers: [

{FirstName: Сергей Иванович,
LastName: Булкин},

{FirstName: Ирина Ивановна,
LastName: Веремей}

]}

Связь «многие-ко-многим» – вариант 2

{_id: <ObjectId11>,
FirstName: Сергей Иванович,
LastName: Булкин,
Courses: [
{CourseName: Базы данных,
Hours: 44},
{CourseName: Проектирование ИС,
Hours: 44}]
}

{_id: <ObjectId15>,
FirstName: Ирина Ивановна,
LastName: Веремей}
Courses: [
{CourseName: Проектирование ИС,
Hours: 44}
]}

{_id: <ObjectId12>,
FirstName: Андрей Павлович,
LastName: Перов
Courses: [
{CourseName: Базы данных,
Hours: 44}
]}

Связь «многие-ко-многим» – вариант 3

{_id: <ObjectId10>,
CourseName: Базы данных,
Hours: 44
Teachers: [<ObjectId11>, <ObjectId12>]}

{_id: <ObjectId20>,
CourseName: Проектирование ИС,
Hours: 44
Teachers: [<ObjectId11>, <ObjectId15>]}

{_id: <ObjectId11>,
FirstName: Сергей Иванович,
LastName: Булкин,
Courses: [<ObjectId10>, <ObjectId20>]}

{_id: <ObjectId12>,
FirstName: Андрей Павлович,
LastName: Перов
Courses: [<ObjectId10>]}

{_id: <ObjectId15>,
FirstName: Ирина Ивановна,
LastName: Веремей}
Courses: [<ObjectId20>]}

Связь «многие-ко-многим» – вариант 4

{_id: <ObjectId10>,
CourseName: Базы данных,
Hours: 44
Appointment: [<ObjectId31>, <ObjectId32>]}

{_id: <ObjectId20>,
CourseName: Проектирование ИС,
Hours: 44
Appointment: [<ObjectId33>, <ObjectId34>]}

{_id: <ObjectId11>,
FirstName: Сергей Иванович,
LastName: Булкин,
Appointment: [<ObjectId31>, <ObjectId33>]}

{_id: <ObjectId12>,
FirstName: Андрей Павлович,
LastName: Перов
Appointment: [<ObjectId32>]}

{_id: <ObjectId15>,
FirstName: Ирина Ивановна,
LastName: Веремей}
Appointment: [<ObjectId34>]}

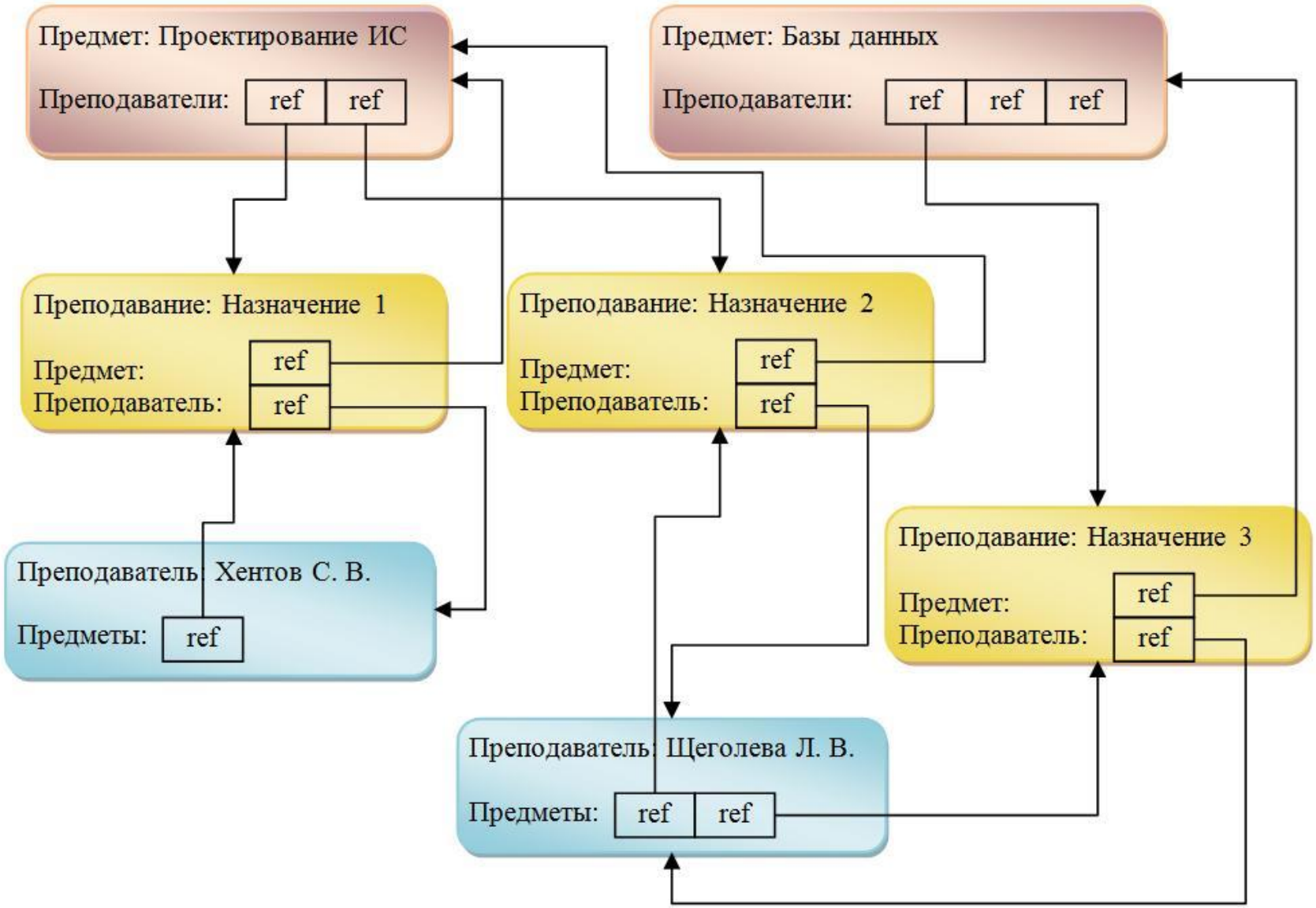
{_id: <ObjectId31>,
Course: <ObjectId10>,
Teachers : <ObjectId11>}

{_id: <ObjectId32>,
Course: <ObjectId10>,
Teachers : <ObjectId12>}

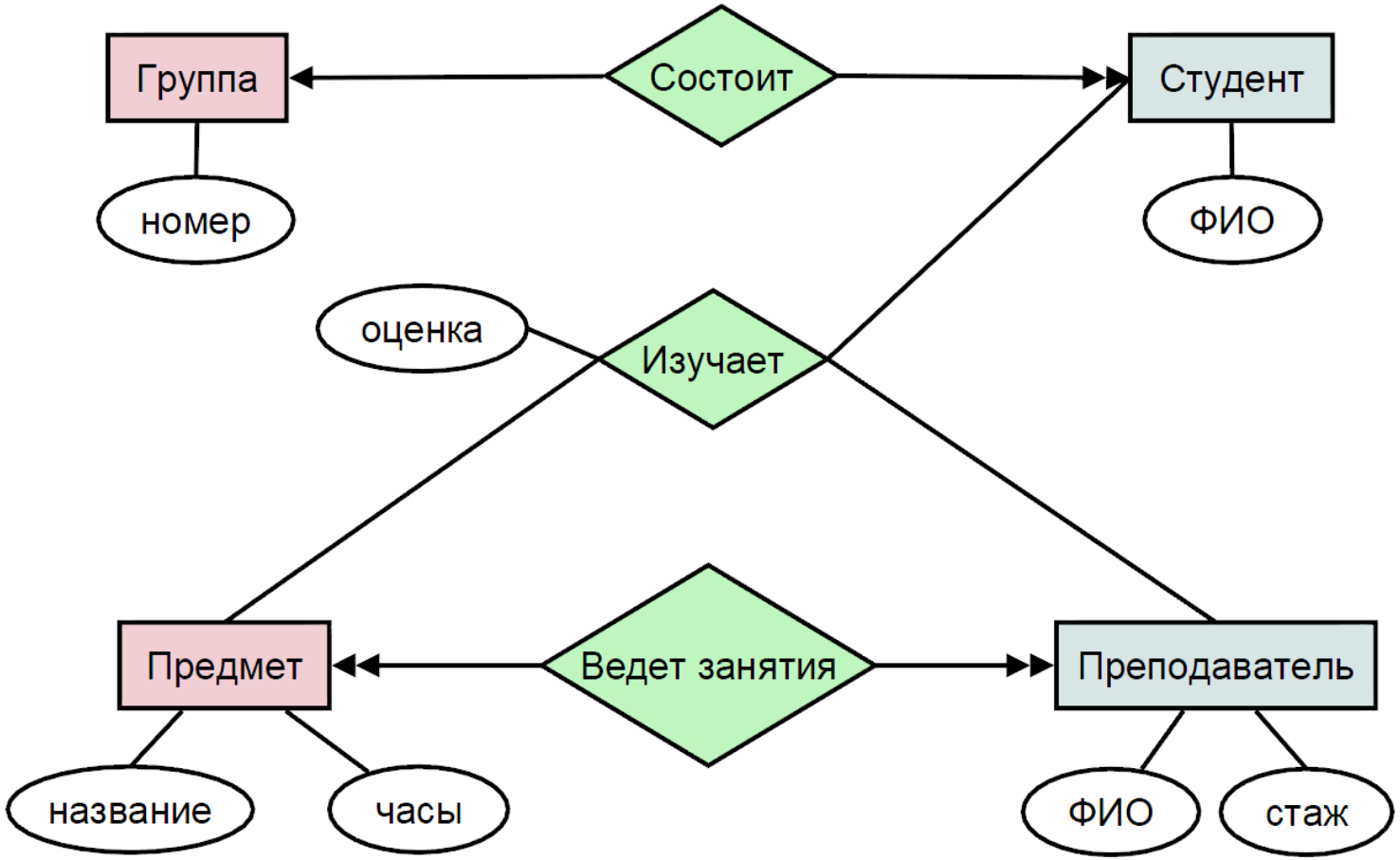
{_id: <ObjectId33>,
Course: <ObjectId20>,
Teachers : <ObjectId11>}

{_id: <ObjectId34>,
Course: <ObjectId20>,
Teachers : <ObjectId15>}

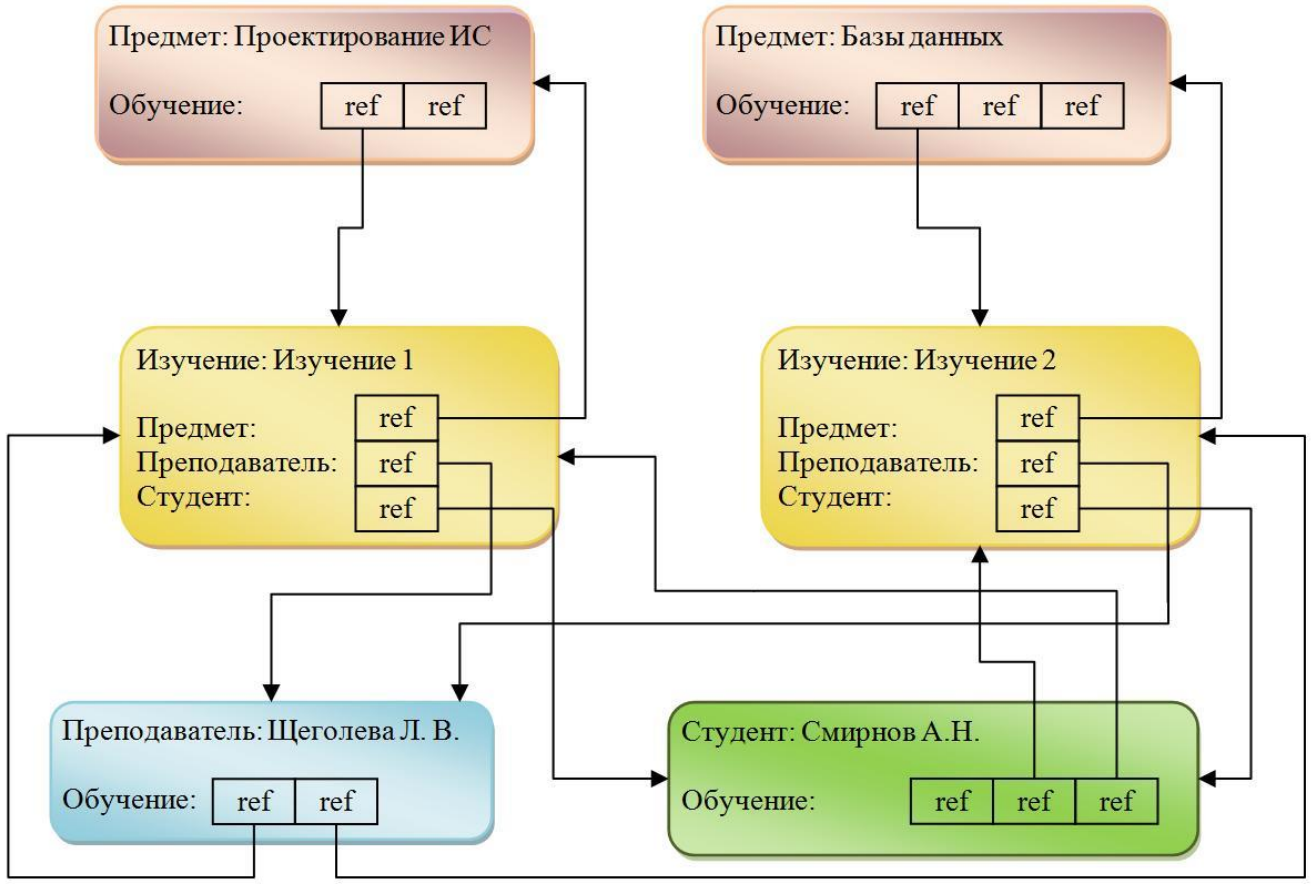
Связь «многие-ко-многим» – вариант 4



Инфологическая модель



Связь между тремя классами



Связь между тремя классами

{_id: <ObjectId10>,
CourseName: Базы данных,
Hours: 44
Appointment: [<ObjectId31>, <ObjectId32>]}

{_id: <ObjectId20>,
CourseName: Проектирование ИС,
Hours: 44
Appointment: [<ObjectId33>, <ObjectId34>]}

{_id: <ObjectId11>,
FirstName: Сергей Иванович,
LastName: Булкин,
Appointment: [<ObjectId31>, <ObjectId33>]}

{_id: <ObjectId12>,
FirstName: Андрей Павлович,
LastName: Перов
Appointment: [<ObjectId32>]}

{_id: <ObjectId3>,
FirstName: Сергей,
LastName: Комаров,
Group: <ObjectId1>}

{_id: <ObjectId41>,
Course: <ObjectId10>,
Teachers : <ObjectId11>,
Student: <ObjectId3>,
Mark: хорошо}

{_id: <ObjectId42>,
Course: <ObjectId20>,
Teachers : <ObjectId15>,
Student: <ObjectId3>,
Mark: отлично}