Базы Данных:
легче, чем кажется

Орлов Илья
Нижний Новгород
2019г.

Women in Big Data

Агенда

- Постановка задачи: БД для ОГО
- Свойства БД: порядок прежде всего
- Нарушение свойств: антипримеры
- Начинаем проектирование: вспоминаем грамматику
- Модели данных: выбор очевиден
- Реляционные СУБД: выбор не так уж и очевиден
- Нормализация отношений: долго, скучно, но необходимо
- Метод ER-диаграмм: нормальные отношения сразу
- Основы SQL: конфигурирование и CRUD
- Работа с БД в Python через DB-API
- Решение задачи: бета-версия и первые результаты
- SQL-инъекции и защита от них

Women
in Big Data

Постановка задачи: БД для ОГО

Список утверждений об - «ОГО» - одной грандиозной организации:

- организация состоит из сотрудников, у которых есть ФИО, должность и уникальный табельный номер;
- сотрудники работают в различных проектах: один сотрудник может участвовать в нескольких проектах; в проекте, разумеется, может быть несколько сотрудников; бывают проекты без сотрудников (которые еще только планируются) и сотрудники без проектов (проект закрылся, но его сотрудников пока не сократили);
- у каждого проекта есть название и уникальный идентификатор;
- в каждом проекте есть не более одного менеджера;
- все сотрудники получают зарплату (как ни странно), которая складывается из должностного оклада и премии.

Women
in Big Data

Постановка задачи

Задачи, которые должна решать База Данных и программа-клиент для работы с ней в 0-й версии:

- ~~хотя бы запускаться и не падать в течение первого часа работы;~~
- хранить данных о сотрудниках, проектах и связях между ними;
- предоставлять каждому сотруднику возможность посмотреть свою и только свою зарплату;
- предоставлять каждому менеджеру возможность смотреть и редактировать премии всех сотрудников его проекта (но свои, разумеется, редактировать нельзя), а также исключать сотрудников из этого проекта (опять же, кроме себя).

Women
in Big Data

Свойства баз данных: порядок прежде всего

База данных – это систематизированный набор данных, отображающий атрибуты и взаимосвязь объектов предметной области и предназначенный для удовлетворения информационных потребностей пользователей.

Набор данных

VS

Систематизированный набор данных

Women
in Big Data

Свойства баз данных

- Целостность (полнота, непротиворечивость, адекватность)
- Неизбыточность
- Безопасность

А что будет, если их нарушить?

Women in Big Data

Антипримеры (не повторять, опасно для кармы!)

ФИО	Должность	Оклад
Иванов И.И.	инженер	50000
Петров П.П.	старший инженер	51000
Сидоров С.С.	менеджер проекта	100000
Иванов И.И.	инженер	50000

Нарушение целостности (полноты)
информация неполная, т.к. записи не могут быть однозначно идентифицированы

ФИО	Должность	Оклад
Иванов И.И.	инженер	50000
Петров П.П.	старший инженер	51000
Сидоров С.С.	менеджер проекта	100000
Иванов И.И.	инженер	60000

Нарушение неизбыточности
информация избыточная, т.к. оклад должен однозначно определяться должностью

ФИО	Должность	Оклад
Иванов И.И.	инженер	150000
Петров П.П.	старший инженер	51000
Сидоров С.С.	менеджер проекта	10000
Иванов И.И.	инженер	50000

Нарушение безопасности
отсутствуют уровни доступа к информации, защита от изменения информации произвольным пользователем

Women in Big Data

Начинаем проектирование: вспоминаем грамматику

Формулируем утверждения о предметной области, строго выделяя подлежащие и дополнения (это будут сущности), и сказуемые (это будут связи)

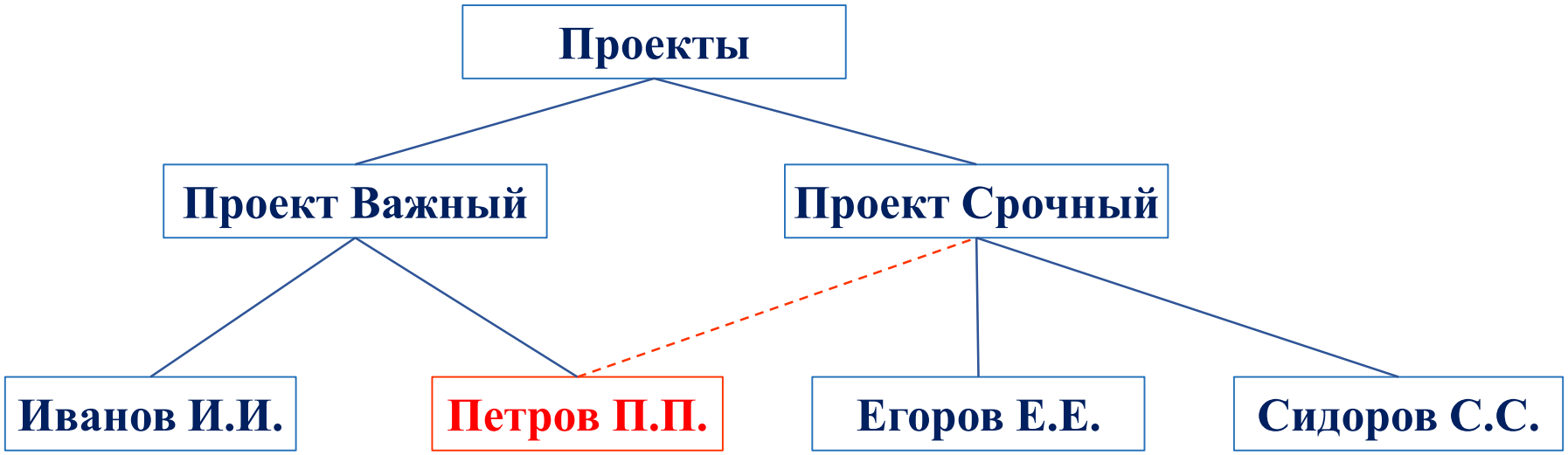
- **сотрудник** относится к **проекту**
- **сотрудник** имеет **оклад** и **премию** (из которых складывается зарплата, если кто не знает)
- **сотрудник** имеет **должность** (инженер, старший инженер, менеджер)
- **оклад сотрудника** однозначно определяется **должностью**

Модели данных: иерархическая – простая, как дерево

- Иерархическая (структура – дерево)

+ удобная, если сущностей мало и связи простые

- отсутствие гибкости (как отразить возможность нахождения сотрудника сразу в нескольких проектах?)



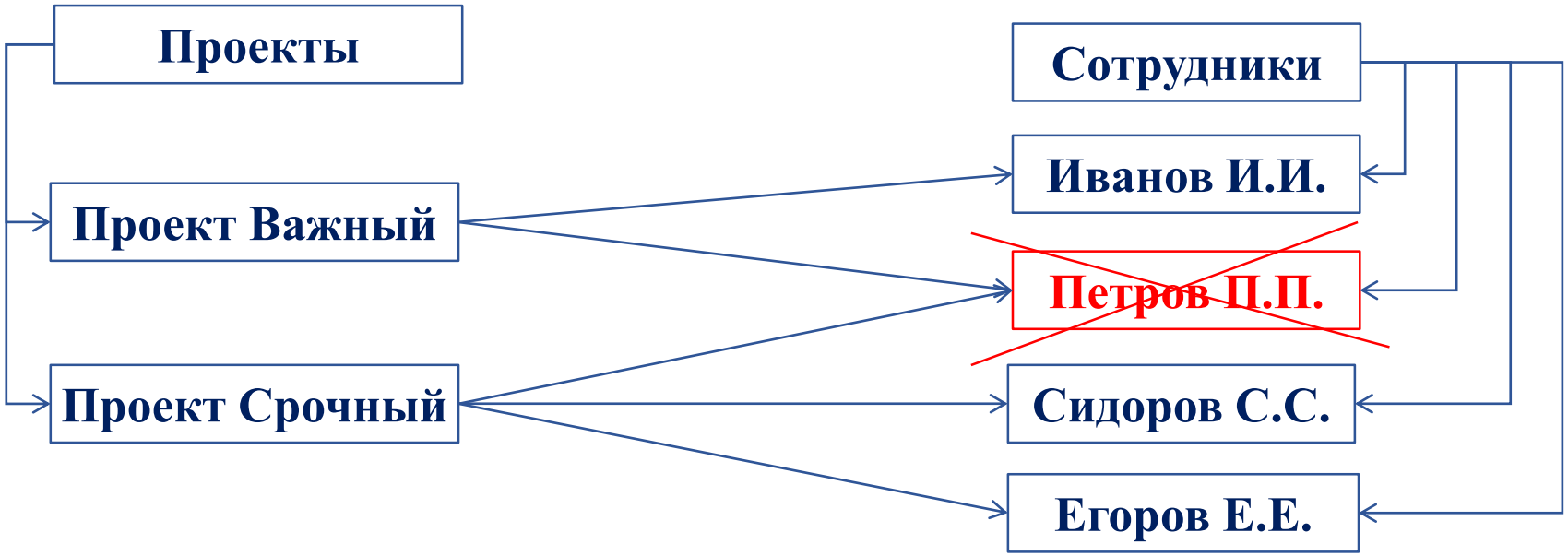
Women in Big Data

Модели данных: сетевая – легко запутаться

- Сетевая (структура – граф)

+ более гибкая, чем иерархическая

- сложно контролировать целостность и неизбыточность

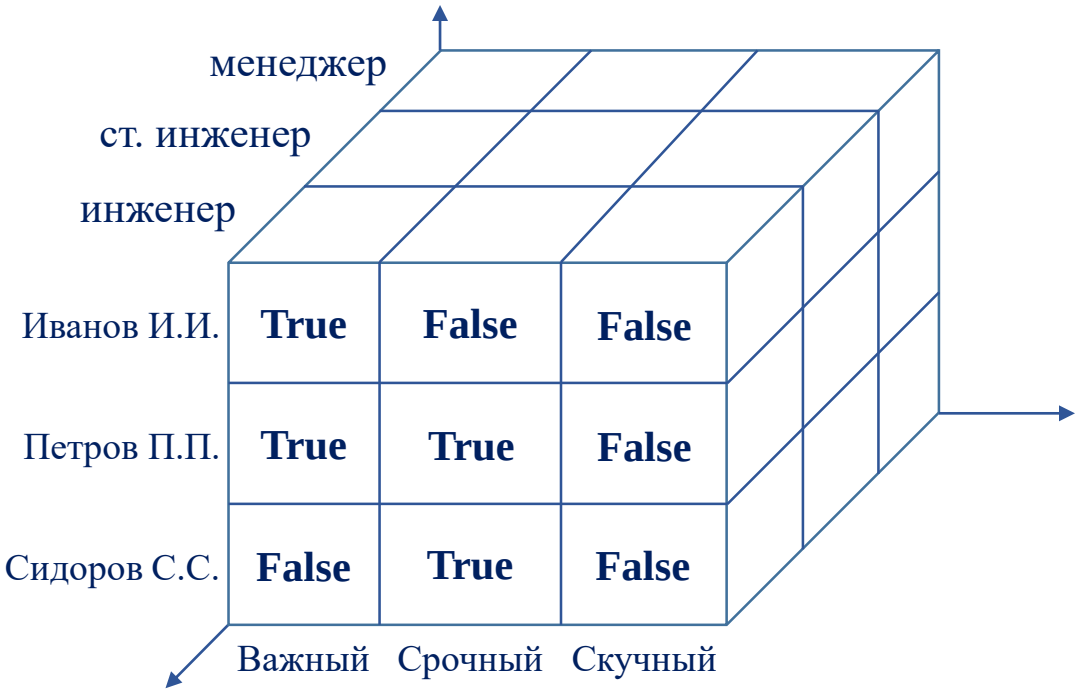


Модели данных: многомерная – для спецзадач

- Многомерная (структура – гиперкуб)

+ удобная для аналитической обработки больших объемов данных (особенно, привязанных ко времени)

- громоздкая и неэффективная для оперативной обработки информации

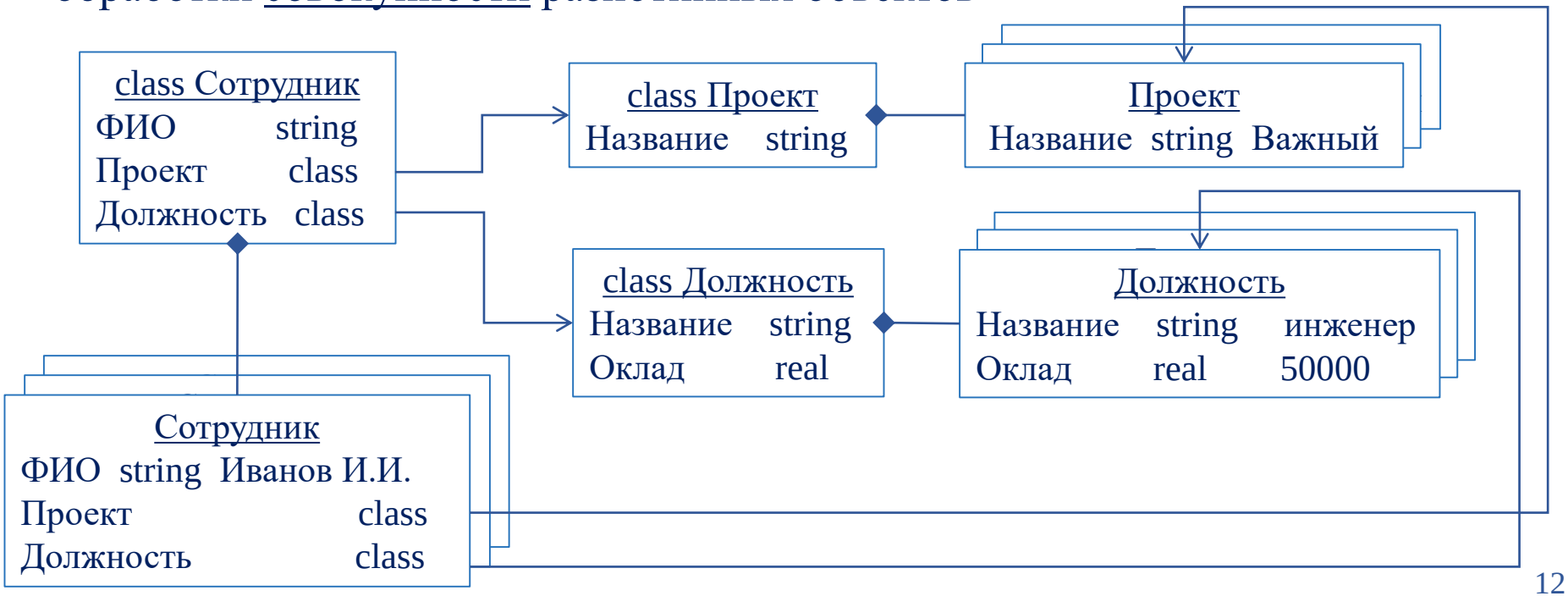


Модели данных: объектно-ориентированная – для ORM

- Объектно-ориентированная (структура – тоже дерево)

+ удобная для работы с отдельными объектами, полноценно представляющими соответствующие сущности со всеми их связями

- сложность алгоритмов и низкая скорость выполнения запросов для обработки совокупности разнотипных объектов



Women in Big Data

Модели данных: реляционная – то, что надо!

- Реляционная (структура – таблицы)

+ удобная для понимания, физической реализации и оперативной обработки данных

- в общем случае необходимо анализировать совокупность таблиц даже если модифицируются атрибуты отдельной сущности

Сотрудники		
ID	**ФИО**	**Должность**
1	Иванов И.И.	инженер
2	Петров П.П.	старший инженер
3	Сидоров С.С.	менеджер проекта
4	Егоров Е.Е.	инженер

Проекты	
ID	**Название**
1	Важный
2	Срочный
3	Скучный

Сотрудник-Проект	
ID сотрудника	**ID проекта**
1	1
2	1
2	2
3	2
4	2

Должность-Оклад	
Название	**Оклад**
инженер	50000
старший инженер	51000
менеджер проекта	100000

Women in Big Data

Модели данных: постреляционная – лучшее – враг хорошего

- Постреляционная (структура – таблицы с возможностью вложенности)

+ можно заменить совокупность связанных реляционных таблиц одной постреляционной

- сложно контролировать целостность и неизбыточность

Проекты		
ID	**Название**	**ID сотрудника**
1	Важный	1
		1
2	Срочный	2
		2
		2
3	Скучный	

Выбор реляционной СУБД: прагматичность превыше всего

PostgreSQL

ORACLE

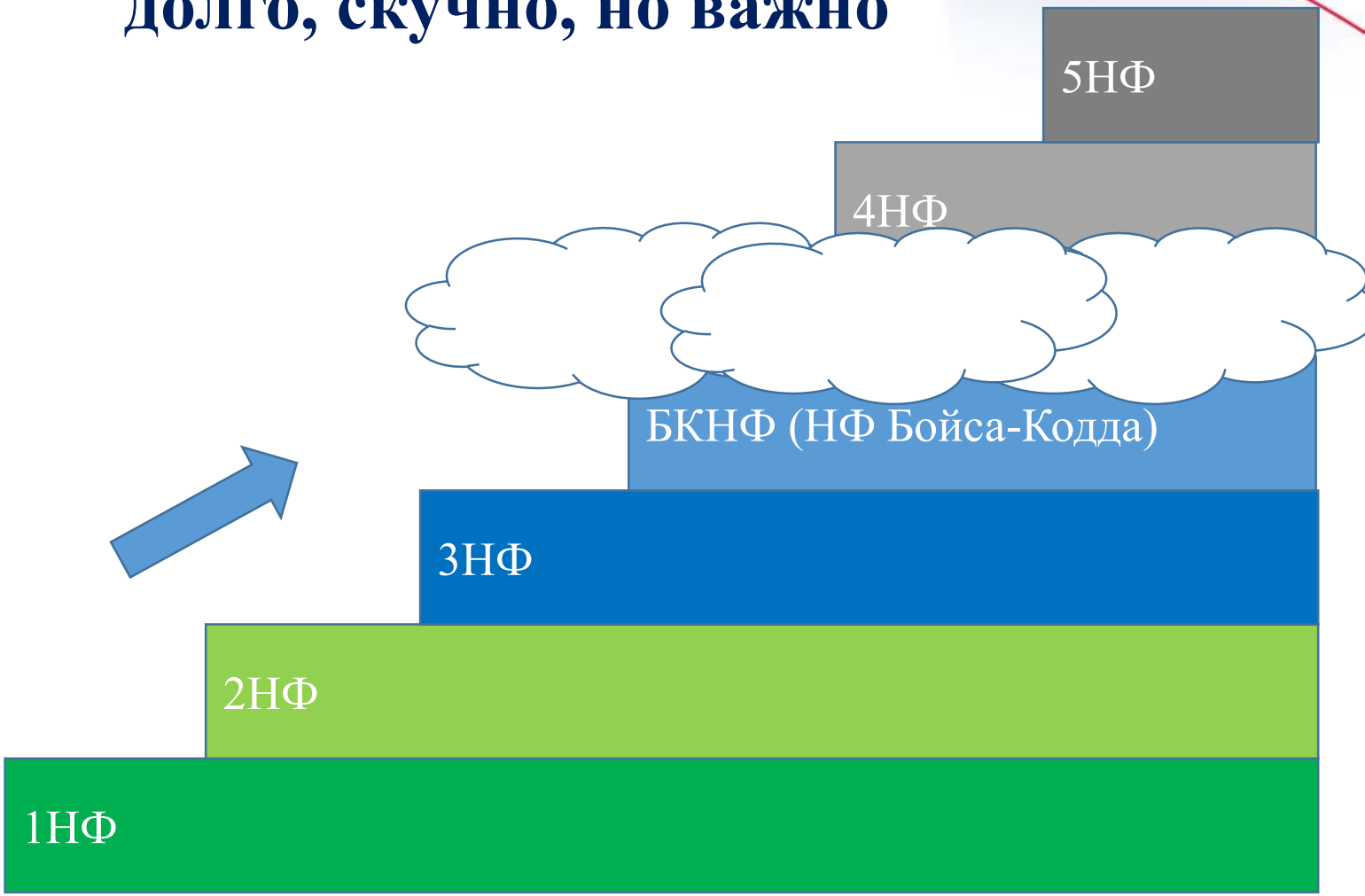
MySQL

Microsoft SQL Server

SQLite

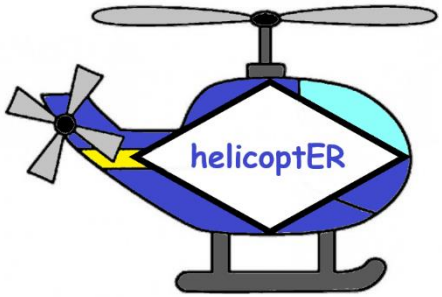
Women in Big Data

Нормализация отношений: долго, скучно, но важно



Women in Big Data

Метод ER-диаграмм: нормальные отношения сразу



5НФ

4НФ

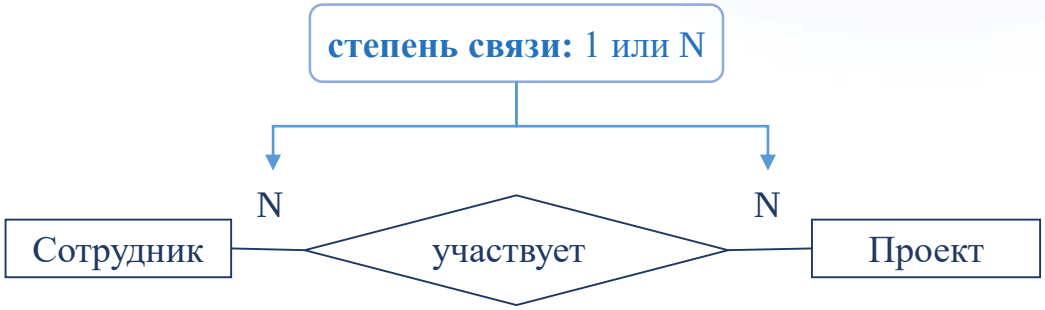
БКНФ (НФ Бойса-Кодда)

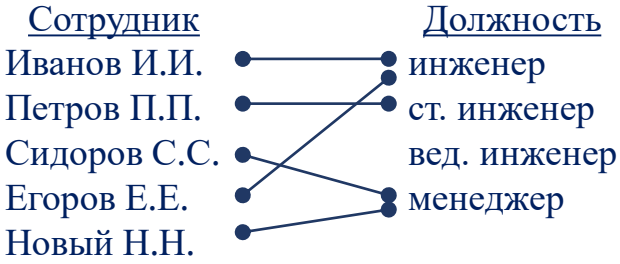
3НФ

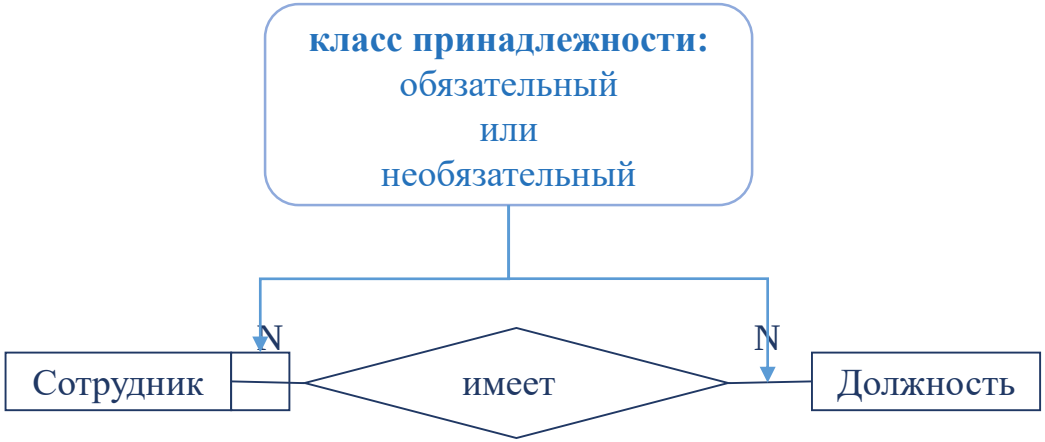
2НФ

1НФ

Метод ER-диаграмм







Women in Big Data

Метод ER-диаграмм

Должность	Оклад
инженер	50000
ст. инженер	51000
вед. инженер	70000
менеджер	100000

Сотрудник	Премия
Иванов И.И.	**30000**
Петров П.П.	50000
Сидоров С.С.	**30000**
Егоров Е.Е.	**20000**
Новый Н.Н.	**20000**

Сотрудник	Аккаунт
Иванов И.И.	ivanovi
Петров П.П.	petrovp
Сидоров С.С.	sidorovs
Егоров Е.Е.	egorove
Новый Н.Н.	novyin

Метод ER-диаграмм: 6 правил

- [1]:[1] = одно отношение
- 1:[1] = два отношения
- 1:[N] = два отношения
- 1:1 = три отношения
- 1:N = три отношения
- N:N = три отношения

Women in Big Data

Метод ER-диаграмм: результат проектирования

Сотрудники					
Таб. номер	**ФИО**	**Должность**	**Премия**	**Логин**	**Пароль**
1	Иванов И.И.	инженер	30000	ivanovi	ivanov123
2	Петров П.П.	старший инженер	50000	petrovp	p1e2t3
3	Сидоров С.С.	менеджер проекта	30000	sidorovs	zayka88
4	Егоров Е.Е.	инженер	20000	egorove	qwerty
5	Новый Н.Н.	инженер	20000	novyin	a1111

Проекты	
ID	**Название**
1	Важный
2	Срочный
3	Скучный

Должность-Оклад	
Название	**Оклад**
инженер	50000
старший инженер	51000
ведущий инженер	70000
менеджер проекта	100000

Сотрудник-Проект	
Таб. номер	**ID проекта**
1	1
2	1
2	2
3	2
4	2

Women in Big Data

Первичный ключ - уникальный идентификатор кортежа

Сотрудники					
Таб. номер	ФИО	Должность	Премия	Логин	Пароль
1	Иванов И.И.	инженер	30000	ivanovi	ivanov123
2	Петров П.П.	старший инженер	50000	petrovp	p1e2t3
3	Сидоров С.С.	менеджер проекта	30000	sidorovs	zayka88
4	Егоров Е.Е.	инженер	20000	egorove	qwerty
5	Новый Н.Н.	инженер	20000	novyin	a1111

Проекты	
ID	Название
1	Важный
2	Срочный
3	Скучный

Должность-Оклад	
Название	Оклад
инженер	50000
старший инженер	51000
ведущий инженер	70000
менеджер проекта	100000

Сотрудник-Проект	
Таб. номер	ID проекта
1	1
2	1
2	2
3	2
4	2

Women in Big Data

Внешний ключ: служит для связи двух отношений

Сотрудники					
Таб. номер	**ФИО**	**Должность**	**Премия**	**Логин**	**Пароль**
1	Иванов И.И.	инженер	30000	ivanovi	ivanov123
2	Петров П.П.	старший инженер	50000	petrovp	p1e2t3
3	Сидоров С.С.	менеджер проекта	30000	sidorovs	zayka88
4	Егоров Е.Е.	инженер	20000	egorove	qwerty
5	Новый Н.Н.	инженер	20000	novyin	a1111

Проекты	
ID	**Название**
1	Важный
2	Срочный
3	Скучный

Должность-Оклад	
Название	**Оклад**
инженер	50000
старший инженер	51000
ведущий инженер	70000
менеджер проекта	100000

Сотрудник-Проект	
Таб. номер	**ID проекта**
1	1
2	1
2	2
3	2
4	2

Основы SQL

SQL – Structured Query Language – язык структурированных запросов.

Условно подразделяется на:

- DDL – Data Definition Language – язык определения данных (операторы: CREATE, ALTER, DROP)
- DML – Data Manipulation Language – язык манипулирования данными (операторы: SELECT, INSERT, UPDATE, DELETE)
- DCL – Data Control Language – язык определения доступа к данным (операторы: GRANT, REVOKE, DENY)
- TCL – Transaction Control Language - язык управления транзакциями (операторы: COMMIT, ROLLBACK, SAVEPOINT)

Women in Big Data

Основы SQL: конфигурирование

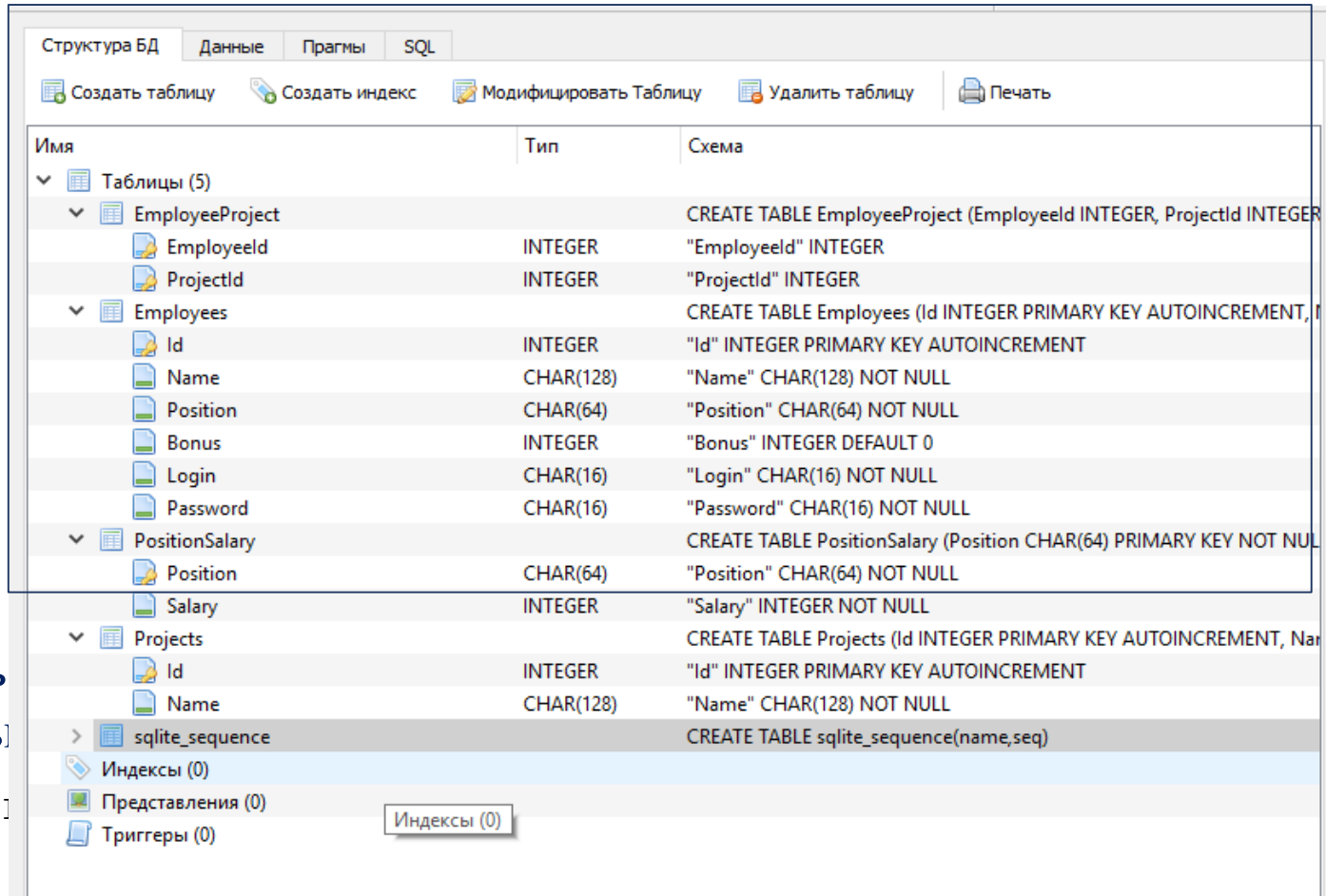
Конфигурируем Базу Данных: каждую таблицу создаем с помощью оператора CREATE.

```
CREATE TABLE Employees
(Id         INTEGER    PRIMARY KEY  AUTOINCREMENT,
Name       CHAR(128)  NOT NULL,
Position   CHAR(64)   NOT NULL,
Bonus      INTEGER    DEFAULT 0,
Login      CHAR(16)   NOT NULL,
Password   CHAR(16)   NOT NULL);

CREATE TABLE Projects
(Id         INTEGER    PRIMARY KEY  AUTOINCREMENT,
Name       CHAR(128)  NOT NULL);

CREATE TABLE PositionSalary
(Position   CHAR(64)   PRIMARY KEY  NOT NULL,
Salary     INTEGER    NOT NULL);

CREATE TABLE EmployeeProject
(EmployeeId  INTEGER,
ProjectId   INTEGER,
PRIMARY KEY (EmployeeId, ProjectId));
```

Women in Big Data

Основы SQL: конфигурирование

Результат можно увидеть в наглядной форме с помощью SQLite браузера.



Удалить
помощь

```
DROP TABL
```

Women in Big Data

Основы SQL: операторы CRUD

После создания необходимых таблиц, мы можем вносить в них данные. Для обозначения основных действий с данными существует специальная аббревиатура — CRUD (create, read, update, delete — «создать, прочесть, обновить, удалить») — акроним, обозначающий четыре базовые функции, используемые при работе с персистентными хранилищами данных.

В соответствии с CRUD в SQL имеются следующие операторы:

- INSERT — оператор языка SQL, который позволяет добавить строку со значениями в таблицу.
- SELECT — оператор запроса в языке SQL, возвращающий набор данных (выборку) из базы данных. Имеет множество опций.
- UPDATE — оператор языка SQL, позволяющий обновить значения в заданных столбцах таблицы.
- DELETE — в языках, подобных SQL, операция удаления записей из таблицы. Критерий отбора записей для удаления определяется выражением WHERE. В случае, если критерий отбора не определён, выполняется удаление всех записей.

Women in Big Data

Основы SQL: операторы CRUD

INSERT

```
INSERT INTO Employees (Name, Position, Bonus, Login, Password)
VALUES ('Иванов И.И.', 'инженер', 30000, 'ivanovi', 'ivanov123');
```

SELECT

```
SELECT * FROM Employees;
# или:
SELECT Id, Name, Bonus FROM Employees
 WHERE Bonus > 20000 ORDER BY Bonus DESC LIMIT 3;
```

UPDATE

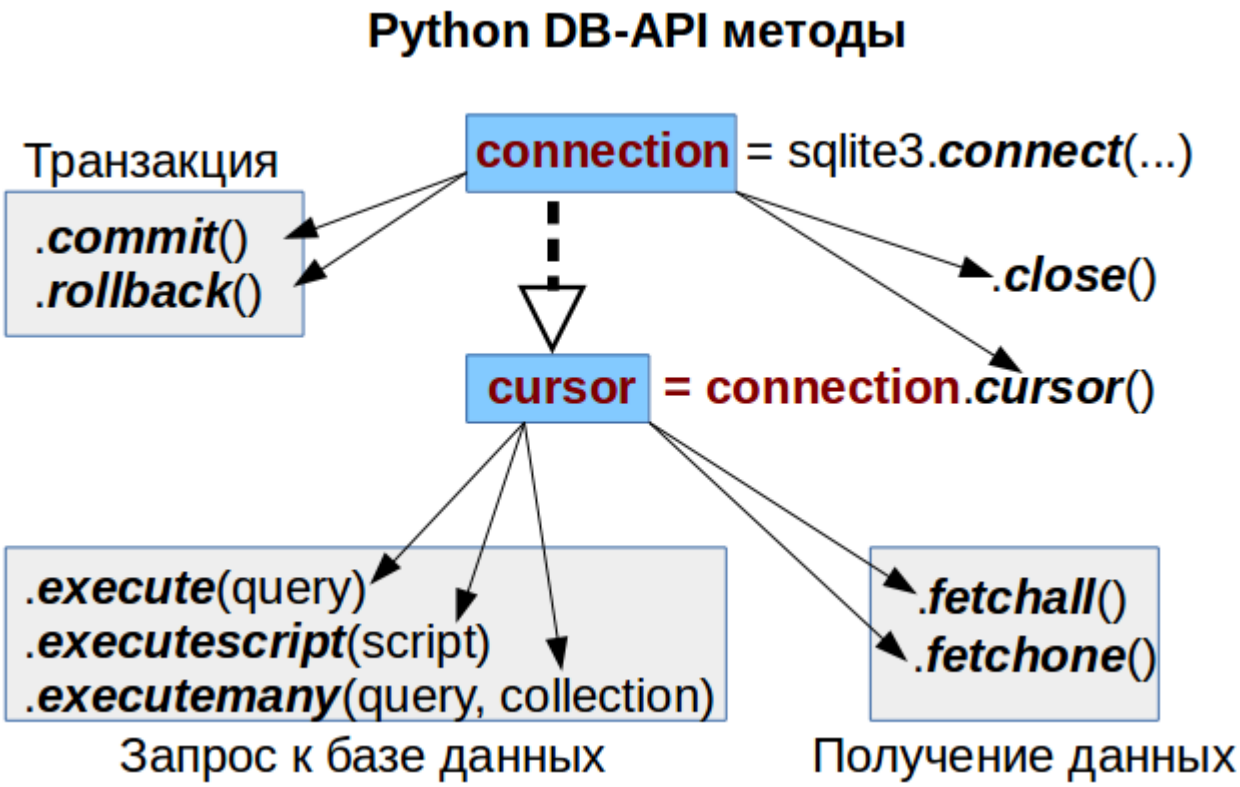
```
UPDATE Employees SET Bonus = 40000 WHERE Id = 1;
```

DELETE

```
DELETE FROM Employees;
# или:
DELETE FROM Employees WHERE Id = 5;
```

Работа с БД в Python через DB-API

PEP 249 определяет DP-API - набор методов и интерфейсов для работы с базами данных



Women in Big Data

БД ОГО через DB-API

Для работы с SQLite в Python используется библиотека sqlite3.

```
# Импортируем библиотеку, соответствующую типу нашей базы данных
import sqlite3


# Файл базы данных
# Если вместо файла указать :memory:, то база будет создана
# в оперативной памяти, а не в файле.
db_name = "ogo.db"

# Создаем соединение с нашей базой данных
# Если файл базы данных еще не создан, он создастся автоматически.
conn = sqlite3.connect(db_name)

# При необходимости меняем тип row_factory, чтоб в ответах
# базы данных отображались названия атрибутов.
conn.row_factory = sqlite3.Row

# РАБОТАЕМ С БАЗОЙ

# Не забываем закрыть соединение с базой данных после работы
conn.close()
```

Women in Big Data

Конфигурирование: CREATE

```
# Конфигурирование базы данных (если необходимо выполнить в скрипте)
def configure_db(conn):
    cur = conn.cursor()

    # Создаем таблицу Employees
    cur.execute("CREATE TABLE Employees"
                "    (Id         INTEGER    PRIMARY KEY  AUTOINCREMENT,"
                "     Name       CHAR(128)  NOT NULL,"
                "     Position   CHAR(64)   NOT NULL,"
                "     Bonus      INTEGER    DEFAULT 0,"
                "     Login      CHAR(16)   NOT NULL,"
                "     Password   CHAR(16)   NOT NULL)")

    # Создаем таблицу Projects
    cur.execute("CREATE TABLE Projects"
                "    (Id         INTEGER    PRIMARY KEY  AUTOINCREMENT,"
                "     Name       CHAR(128)  NOT NULL)")

    # Создаем таблицу PositionSalary
    cur.execute("CREATE TABLE PositionSalary"
                "    (Position   CHAR(64)   PRIMARY KEY  NOT NULL,"
                "     Salary     INTEGER    NOT NULL)")

    # Создаем таблицу EmployeeProject
    cur.execute("CREATE TABLE EmployeeProject"
                "    (EmployeeId  INTEGER,"
                "     ProjectId   INTEGER,"
                "     PRIMARY KEY (EmployeeId, ProjectId))")
```

Women in Big Data

Добавление записей: INSERT

```
# Добавление записей в таблицу Проекты
def insert_project(conn, name):
    # Создаем курсор - специальный объект,
    # который делает запросы и получает их результаты
    cur = conn.cursor()
    # Делаем INSERT запрос к базе данных, используя обычный SQL-синтаксис
    cur.execute("INSERT INTO Projects (Name)"
                " VALUES (:name)",
                {'name': name})
    # Если мы не просто читаем, но и вносим изменения в базу данных
    # - необходимо сохранить транзакцию
    conn.commit()
```

Women in Big Data

Добавление записей: INSERT

```
# Добавление записей в таблицу ДолжностьОклад
def insert_position(conn, position, salary):
    cur = conn.cursor()
    cur.execute("INSERT INTO PositionSalary (Position, Salary)"
                " VALUES (:position, :salary)",
                {'position': position, 'salary': salary})
    conn.commit()


# Добавление записей в таблицу Сотрудники
def insert_employee(conn, name, position, bonus, login, pwd):
    cur = conn.cursor()
    cur.execute("INSERT INTO Employees (Name, Position, Bonus, Login, Password)"
                " VALUES (:name, :position, :bonus, :login, :pwd)",
                {'name': name, 'position': position, 'bonus': bonus,
                 'login': login, 'pwd': pwd})
    conn.commit()


# Добавление записей в таблицу СотрудникиПроекты
def add_employee_to_project(conn, employee_id, project_id):
    cur = conn.cursor()
    cur.execute("INSERT INTO EmployeeProject (EmployeeId, ProjectId)"
                " VALUES (:employeeId, :projectId)",
                {'employeeId': employee_id, 'projectId': project_id})
    conn.commit()
```

Women in Big Data

Создаем БД ОГО

```
db_name = "ogo.db"
db_exists = os.path.exists(db_name)

conn = sqlite3.connect(db_name)
conn.row_factory = sqlite3.Row

if not db_exists:
    configure_db(conn)

    insert_project(conn, "Важный")
    insert_project(conn, "Срочный")

    insert_position(conn, "инженер", 50000)
    insert_position(conn, "старший инженер", 51000)
    insert_position(conn, "менеджер проекта", 100000)

    insert_employee(conn, "Иванов И.И.", "инженер", 30000,
                    "ivanovi", "ivanov123")
    insert_employee(conn, "Петров П.П.", "старший инженер", 50000,
                    "petrovp", "p1e2t3")
    insert_employee(conn, "Сидоров С.С.", "менеджер проекта", 30000,
                    "sidorovs", "zayka88")

    add_employee_to_project(conn, 1, 1)
    add_employee_to_project(conn, 2, 1)
    add_employee_to_project(conn, 2, 2)
    add_employee_to_project(conn, 3, 2)
```

Чтение данных: SELECT

```
# Проверка наличия пользователя в базе данных
# с указанным логином/паролем
def authentication(conn, login, pwd):
    cur = conn.cursor()
    # Делаем SELECT запрос к базе данных, используя обычный SQL-синтаксис
    cur.execute("SELECT E.Id, E.Name, E.Position, EP.ProjectId"
                " FROM Employees AS E, EmployeeProject AS EP"
                " WHERE E.Id = EP.EmployeeId"
                " AND E.Login = :login AND E.Password = :pwd",
                {'login': login, 'pwd': pwd})
    # Получаем результат сделанного запроса
    return cur.fetchone()


# Проверка наличия указанного сотрудника в указанном проекте
def is_employee_in_project(conn, employee_id, project_id):
    cur = conn.cursor()
    cur.execute("SELECT EP.ProjectId"
                " FROM EmployeeProject AS EP"
                " WHERE EP.EmployeeId = :employee_id"
                " AND EP.ProjectId = :project_id",
                {'employee_id': employee_id, 'project_id': project_id})
    return bool(cur.fetchone())
```

Women in Big Data

Чтение данных: SELECT

```
# Вывод информации для сотрудника
# Соединяем таблицы Employees, PositionSalary
def show_employee_info(conn, employee_id):
    cur = conn.cursor()
    cur.execute("SELECT E.Id, E.Name, P.Salary + E.Bonus As Pay"
                " FROM Employees AS E, PositionSalary AS P"
                " WHERE E.Position = P.Position"
                " AND E.Id = :employee_id",
                {'employee_id': employee_id})
    print("Информация для сотрудника:")
    for row in cur.fetchall():
        print(dict(row))
```

Employee

Id	Name	Position	Bonus	Login	Password
1	Иванов И.И.	инженер	30000	ivanovi	ivanov123
2	Петров П.П.	старший инженер	50000	petrovp	p1e2t3
3	Сидоров С.С.	менеджер проекта	30000	sidorovs	zayka88

PositionSalary

Position	Salary
инженер	50000
старший инженер	51000
менеджер проекта	100000

Id	Name	Position	Salary	Bonus
1	Иванов И.И.	инженер	50000	30000
2	Петров П.П.	старший инженер	51000	50000
3	Сидоров С.С.	менеджер проекта	100000	30000

Women in Big Data

Чтение данных: SELECT

```
# Вывод информации для менеджера проекта
# Соединяем таблицы Employees, PositionSalary, EmployeeProject
def show_manager_info(conn, project_id):
    cur = conn.cursor()
    cur.execute("SELECT E.Id, E.Name, P.Salary + E.Bonus As Pay"
                " FROM Employees AS E, PositionSalary AS P, "
                "      EmployeeProject AS EP"
                " WHERE E.Position = P.Position"
                " AND E.Id = EP.EmployeeId"
                " AND EP.ProjectId = :project_id",
                {'project_id': project_id})
    print("Информация для менеджера:")
    for row in cur.fetchall():
        print(dict(row))
```

Employee

Id	Name	Position	Bonus	Login	Password
1	Иванов И.И.	инженер	30000	ivanovi	ivanov123
2	Петров П.П.	старший инженер	50000	petrovp	p1e2t3
3	Сидоров С.С.	менеджер проекта	30000	sidorovs	zayka88

PositionSalary

Position	Salary
инженер	50000
старший инженер	51000
менеджер проекта	100000

Id	Name	Position	Salary	Bonus	ProjectID
1	Иванов И.И.	инженер	50000	30000	1
2	Петров П.П.	старший инженер	51000	50000	1
2	Петров П.П.	старший инженер	51000	50000	2
3	Сидоров С.С.	менеджер проекта	100000	30000	2

EmployeeProject

EmployeeID	ProjectID
1	1
2	1
2	2
3	2

Women in Big Data

Изменение данных: UPDATE и DELETE

```
# Изменение премии сотрудника
def update_employee_bonus(conn, employee_id, new_bonus):
    cur = conn.cursor()
    # Делаем UPDATE запрос к базе данных, используя обычный SQL-синтаксис
    cur.execute("UPDATE Employees"
                " SET Bonus = :new_bonus"
                " WHERE Id = :employee_id",
                {'employee_id': employee_id, 'new_bonus': new_bonus})
    conn.commit()


# Удаление сотрудника из проекта (но не из базы данных)
def delete_employee_from_project(conn, employee_id, project_id):
    cur = conn.cursor()
    # Делаем DELETE запрос к базе данных, используя обычный SQL-синтаксис
    cur.execute("DELETE FROM EmployeeProject"
                " WHERE EmployeeId = :employee_id"
                " AND ProjectId = :project_id",
                {'employee_id': employee_id, 'project_id': project_id})
    conn.commit()
```

Решение задачи: бета-версия

```
login = input("Логин: ")
pwd = input("Пароль: ")
res = authentication(conn, login, pwd)
if res:
    user = dict(res)
    print("Здравствуйте, {}".format(user['Name']))
    if user['Position'] == "менеджер проекта":
        show_manager_info(conn, user['ProjectId'])

        id_upd = int(input("Изменение премии. ID сотрудника (0 - отмена): "))
        if id_upd:
            if (id_upd != user['Id'] and
                    is_employee_in_project(conn, id_upd, user['ProjectId'])):
                new_bonus = input("Новая премия: ")
                update_employee_bonus(conn, id_upd, new_bonus)
            else:
                print("Невозможно изменить премию для данного сотрудника")

        id_del = int(input("Удаление сотрудника. ID сотрудника (0 - отмена): "))
        if id_del:
            if id_del != user['Id']:
                delete_employee_from_project(conn, id_del, user['ProjectId'])
            else:
                print("Невозможно удалить данного сотрудника из проекта")
    else:
        show_employee_info(conn, user['Id'])
else:
    print("Доступ запрещен")
```

Women in Big Data

Решение задачи: тестирование бета-версии

```
Логин: sidorovs
Пароль: zayka88
Здравствуйте, Сидоров С.С.
Информация для менеджера:
{'Id': 2, 'Name': 'Петров П.П.', 'Pay': 101000}
{'Id': 3, 'Name': 'Сидоров С.С.', 'Pay': 130000}
Изменение премии. ID сотрудника (0 - отмена): 2
Новая премия: 60000
Удаление сотрудника. ID сотрудника (0 - отмена): 0
```

```
Логин: sidorovs
Пароль: zayka88
Здравствуйте, Сидоров С.С.
Информация для менеджера:
{'Id': 2, 'Name': 'Петров П.П.', 'Pay': 111000}
{'Id': 3, 'Name': 'Сидоров С.С.', 'Pay': 130000}
Изменение премии. ID сотрудника (0 - отмена): 0
Удаление сотрудника. ID сотрудника (0 - отмена): 2
```

```
Логин: sidorovs
Пароль: 123
Доступ запрещен
```

Women in Big Data

SQL-инъекции: уязвимый код

```
def bad_authentication(conn, login, pwd):
    cur = conn.cursor()
    cur.execute("SELECT E.Id, E.Name, E.Position, EP.ProjectId"
                " FROM Employees AS E, EmployeeProject AS EP"
                " WHERE E.Id = EP.EmployeeId"
                " AND E.Login = '{login}' AND E.Password = '{pwd}'".
                format(login=login, pwd=pwd))
    return cur.fetchone()
```

```
Логин: ivanovi
Пароль: ivanov123
Здравствуйте, Иванов И.И.
Информация для сотрудника:
{'Id': 1, 'Name': 'Иванов И.И.', 'Pay': 80000}
```

```
Логин: ivanovi
Пароль: 123
Доступ запрещен
```

```
Логин: ivanovi
Пароль: 123' OR 'a'='a
Здравствуйте, Иванов И.И.
Информация для сотрудника:
{'Id': 1, 'Name': 'Иванов И.И.', 'Pay': 80000}
```

Women in Big Data

SQL-инъекции: защищенный код

```
def authentication(conn, login, pwd):
    cur = conn.cursor()
    cur.execute("SELECT E.Id, E.Name, E.Position, EP.ProjectId"
                " FROM Employees AS E, EmployeeProject AS EP"
                " WHERE E.Id = EP.EmployeeId"
                " AND E.Login = :login AND E.Password = :pwd",
                {'login': login, 'pwd': pwd})
    return cur.fetchone()


def authentication2(conn, login, pwd):
    cur = conn.cursor()
    cur.execute("SELECT E.Id, E.Name, E.Position, EP.ProjectId"
                " FROM Employees AS E, EmployeeProject AS EP"
                " WHERE E.Id = EP.EmployeeId"
                " AND E.Login = ? AND E.Password = ?",
                (login, pwd))
    return cur.fetchone()
```

```
Логин: ivanovi
Пароль: 123' OR 'a'='a
Доступ запрещен
```

далее ничего не придумали, импровизируй