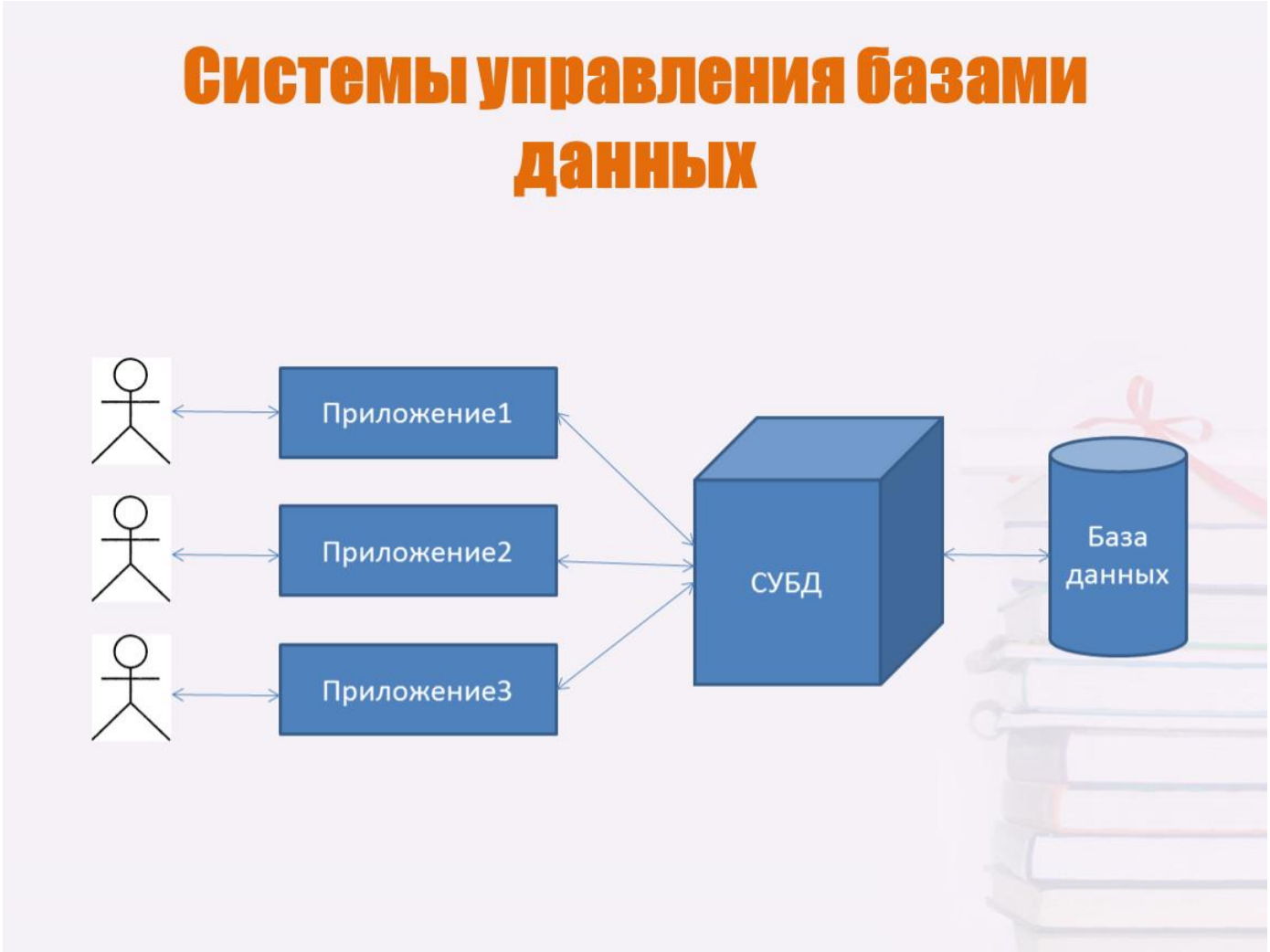
Базы данных

Основные понятия

- ***База данных (БД)*** — именованная совокупность данных, отражающая состояние объектов и их отношений в расс
- ***Система управления базами данных (СУБД)*** — совокупность языковых и программных средств, предназначенных для создания, наполнения, обновления и удаления баз данных
- ***Модель данных*** — это некоторая абстракция, которая, будучи приложима к конкретным данным, позволяет пользователям и разработчикам трактовать их уже как информацию, то есть сведения, содержащие не только данные, но и взаимосвязь между ними.
- ***Таблица –*** это структура хранения данных в БД.

Взаимодействия



Первичный ключ

- ***Первичный ключ.*** Первичным ключом называется поле или набор полей, однозначно идентифицирующих запись.

ПРОСТОЙ КЛЮЧ

Номер	Автор	Название	Год	Полка
001	Беляев А.Р.	Звезда КЭЦ	1990	3
002	Олеша Ю.К.	Избранное	1987	5
003	Беляев А.Р.	Избранное	1994	1

В базе данных «Домашняя библиотека» у разных книг могут совпадать значения полей, но инвентарный номер у каждой книги свой

СОСТАВНОЙ КЛЮЧ

Город	Номер школы	Директор	Телефон
Крюков	1	Иванов А.П.	12-35-60
Шадринск	1	Строев С.С.	4-33-11
Шадринск	2	Иванов А.П.	4-23-15

В этой таблице у разных записей не могут совпадать одновременно значения двух полей: «Город» и «Номер школы». Они образуют составной ключ таблицы

Вторичный ключ

- По полям, которые часто используются при поиске и сортировке данных устанавливаются ***вторичные ключи***: они помогут системе значительно быстрее найти нужные данные. В отличие от первичных ключей поля для индексов (вторичные ключи) могут содержать неуникальные значения.
- Первичные ключи используются для установления связей между таблицами в реляционной БД. В этом случае первичному ключу одной таблицы (родительской) соответствует ***внешний ключ*** другой таблицы (дочерней). Внешний ключ содержит значения связанного с ним поля, являющегося первичным ключом. Значения во внешнем ключе могут быть неуникальными, но не должны быть пустыми. Первичный и внешний ключи должны быть одинакового типа.

Связи между таблицами

- ***Связи между таблицами***. Записи в таблице могут зависеть от одной или нескольких записей другой таблицы. Такие отношения между таблицами называются *связями.* Связь определяется следующим образом: поле или несколько полей одной таблицы, называемое *внешним ключом,* ссылается на первичный ключ другой таблицы. Рассмотрим пример.
  Так как каждый заказ должен исходить от определенного клиента, каждая запись таблицы *Orders* (заказы) должна ссылаться на соответствующую запись таблицы *Customers* (клиенты). Это и есть связь между таблицами *Orders* и *Customers.* В таблице *Orders* должно быть поле, где хранятся ссылки на те или иные записи таблицы *Customers.*

Связи

- ***Один к одному*** — каждая запись родительской таблицы связана только с одной записью дочерней. Такая связь встречается на практике намного реже, чем отношение *один ко многим* и реализуется путем определения уникального внешнего ключа. Связь *один к одному* используют, если не хотят, чтобы таблица «распухала» от большого числа полей. Базы данных, в состав которых входят таблицы с такой связью не могут считаться полностью нормализованными.
- ***Один ко многим*** — каждая запись родительской таблицы связана с одной или несколькими записями дочерней. Например, один клиент может сделать несколько заказов, однако несколько клиентов не могут сделать один заказ. Связь о*дин ко многим* является самой распространенной для реляционных баз данных.
- ***Многие ко многим*** — несколько записей одной таблицы связаны с несколькими записями другой. Например, один автор может написать несколько книг и несколько авторов — одну книгу. В случае такой связи в общем случае невозможно определить, какая запись одной таблицы соответствует выбранной записи другой таблицы, что делает неосуществимой физическую (на уровне индексов и триггеров) реализацию такой связи между соответствующими таблицами. Поэтому перед переходом к физической модели все связи "многие ко многим" должны быть переопределены (некоторые CASE-средства, если таковые используются при проектировании данных, делают это автоматически).Подобная связь между двумя таблицами реализуется путем создания третьей таблицы и реализации связи типа «один ко многим» каждой из имеющихся таблиц с промежуточной таблицей.

Нормализация БД

- Классическая технология проектирования реляционных баз данных связана с *теорией нормализации*, основанной на анализе функциональных зависимостей между атрибутами отношений.
  Нормализация достигается путем проверки соответствия таблиц ряду условий, определенных в трех уровнях нормализации: первой, второй и третьей нормальных формах (существуют также и другие уровни).

Первая нормальная форма

- ***Первая нормальная форма*** **требует, чтобы каждое поле таблицы БД было неделимым и не содержало повторяющихся групп.**
- Неделимость поля означает, что содержащиеся в нем значения не должны делиться на более мелкие. Например, если в поле «Подразделение» содержится название факультета и кафедры, требование неделимости не соблюдается и необходимо выделить название факультета или кафедры в отдельное поле.
- Повторяющимися являются поля, содержащие одинаковые по смыслу значения. Например, если требуется получить статистику продаж четырех товаров по месяцам, можно создать поля для хранения данных о продаже по каждому товару. Однако что делать, если товаров не 4, а 104, и как быть, если количество товаров заранее не известно? Повторяющиеся группы следует устранить, сохранив в таблице единственное поле «Товар». В результате получим запись, содержащую информацию о статистике продаж по одному товару, но этот товар может быть любым.

Вторая нормальная форма

- ***Вторая нормальная форма*** требует, чтобы все поля таблицы зависели от первичного ключа, то есть, чтобы первичный ключ однозначно определял запись и не был избыточен. Если же в какой-либо таблице имеется зависимость каких-либо не ключевых полей от части первичного ключа, следует выделить их в отдельную таблицу, сделав первичным ключом новой таблицы ту часть первичного ключа, от которой зависят данные поля, и установить связь "один ко многим" от новой таблицы к старой.

Третья нормальная форма

- ***Третья нормальная форма*** требует, чтобы в таблицах не имелось транзитивных зависимостей между не ключевыми полями, то есть чтобы значение любого поля, не входящего в первичный ключ, не зависело от значения другого поля, также не входящего в первичный ключ.

Подключение к БД

```
import sqlite3
from sqlite3 import Error

def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")

    return connection
```

```
connection = create_connection("E:\\sm_app.sqlite")
```

Запросы к БД

```
def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("Query executed successfully")
    except Error as e:
        print(f"The error '{e}' occurred")
```

Создаем таблицу

```
create_users_table = """
CREATE TABLE IF NOT EXISTS users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  age INTEGER,
  gender TEXT,
  nationality TEXT
);
"""
```

Добавляем записи в таблицу

```
create_users = """
INSERT INTO
  users (name, age, gender, nationality)
VALUES
  ('James', 25, 'male', 'USA'),
  ('Leila', 32, 'female', 'France'),
  ('Brigitte', 35, 'female', 'England'),
  ('Mike', 40, 'male', 'Denmark'),
  ('Elizabeth', 21, 'female', 'Canada');
"""

execute_query(connection, create_users)
```

Функция для чтения данных

```
def execute_read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        return result
    except Error as e:
        print(f"The error '{e}' occurred")
```

Пример запроса пользователей

```
select_users = "SELECT * from users"
users = execute_read_query(connection, select_users)

for user in users:
    print(user)
```

Полезные ссылки

- Пример БД Как подружить Python и базы данных SQL. Подробное руководство (proglib.io)