
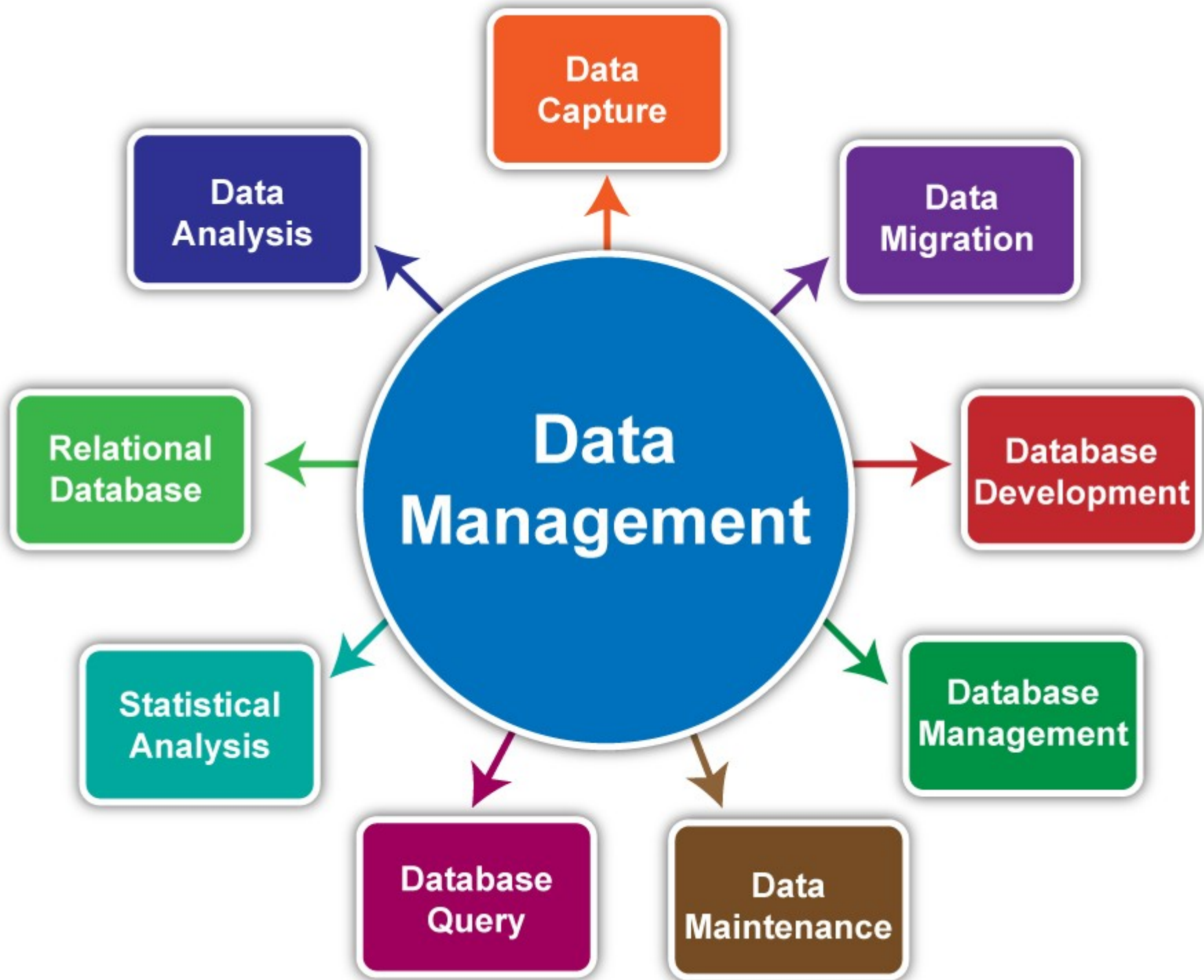


An abstract graphic in the top-left corner consisting of several overlapping squares and circles in various shades of blue and white, creating a layered, geometric effect.

# DB Design



**Данные** - это информация, представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством.



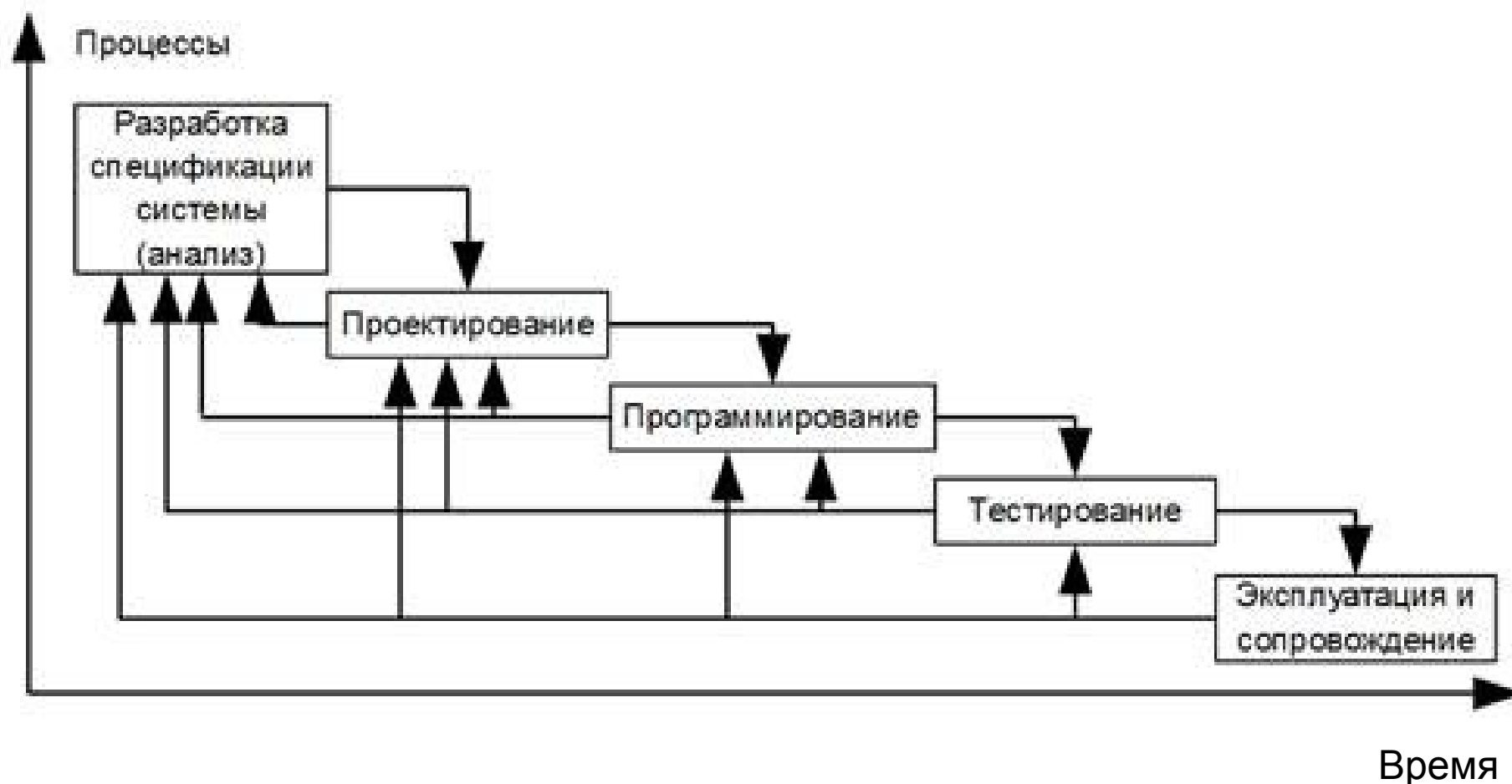


# Автоматизированная информационная система (АИС)

АИС - это совокупность программных и аппаратных средств, предназначенных для хранения и/или управления данными и информацией, а также для производства вычислений.

АИС представляют совокупность функциональных подсистем сбора, ввода, обработки, хранения, поиска и распространения информации. Процессы сбора и ввода данных необязательны, поскольку вся необходимая и достаточная для функционирования АИС информация, может уже находиться в составе ее базы данных. Каждая АИС имеет дело с той или иной частью реального мира, которую принято называть предметной областью

# Жизненный цикл разработки ПО (АИС,ИС ...)





## Предметная область

Это сфера экономической деятельности, например, транспортная логистика, банковский сектор, биллинговые системы, медицина, электронная коммерция, игровое приложение.



## База данных (БД)

Именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, которая допускает использование данных оптимальным образом для одного или нескольких приложений.

В БД информация должна быть организована таким образом, чтобы обеспечить минимальную долю ее избыточности. Частичная избыточность информации необходима, но она должна быть минимизирована, т.к. чрезмерная избыточность данных влечет за собой ряд негативных последствий, главные из которых:

- увеличение объема информации
- появление ошибок при вводе дублирующей информации, нарушающих целостность базы данных и создающих противоречивые данные.



## Для организации БД используют СУБД

**Реляционные** - совокупность таблиц и связей между ними.  
Примеры: PostgreSQL, SQLite

**Документно-ориентированные** - хранит иерархические структуры данных (документы), как правило реализована с помощью подхода NoSQL. Примеры: MongoDB, CouchDB

**Объектно-ориентированные** - хранят информацию в виде объектов, как в объектно-ориентированных языках программирования. Примеры: db4o

**Графовые** - предназначены для хранения взаимосвязей и навигации в них, данные хранятся в виде структуры данных граф, где узлы хранят сущности, а ребра связи. Примеры: OrientDB, Amazon Neptune, Neo4j





# Система управления базами данных (СУБД)

Совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Возможности современных СУБД:

- язык определения данных(DDL), с помощью которого можно создать базу данных, описать ее структуру, типы данных и средства для задания ограничений целостности данных;
- язык управления данными(DML), которые позволяет вставлять, обновлять, удалять и извлекать информацию из БД;



## Возможности СУБД

- кроссплатформенность - независимость от используемой архитектуры или ОС;
- подсистема разграничения доступа к данным;
- подсистема поддержки целостности БД обеспечивающая непротиворечивое состояние хранимых данных;
- Подсистема управления параллельной работой приложений контролирующая процессы их совместного доступа к БД;
- подсистема восстановления, позволяющая вернуть БД к предыдущему состоянию, нарушенному из-за аппаратного или программного сбоя;



# Реляционные базы данных как один из типов БД.

**Реляционная база данных** – это совокупность отношений, содержащих всю информацию, которая должна храниться в БД. Однако пользователи могут воспринимать такую базу данных как совокупность таблиц.

**Моделирование данных** - это средство формального сбора данных, относящихся к бизнес-процессу данной организации. Моделирование является приемом анализа, на базе которого строятся реляционные БД.



# DB Design

Процесс моделирования данных включает три уровня моделей: от концептуальной к логической, а затем к физической.

**Концептуальная модель** - модель предметной области, состоящая из перечня взаимосвязанных объектов, используемых для описания этой области, вместе со свойствами и характеристиками. Концептуальная модель, как правило, представляет сущность бизнеса и ничего больше.

**Логическая модель** - представляется диаграммой «сущность-связь» или ER (Entity-Relationship) диаграммой.

**Физическая модель** - это схема базы данных для конкретной СУБД, где сущности предметной области превращаются в таблицы, атрибуты в ее колонки (часто называют полями) с использованием конкретного типа данных, связи в ограничения целостности.

# Database Design Steps



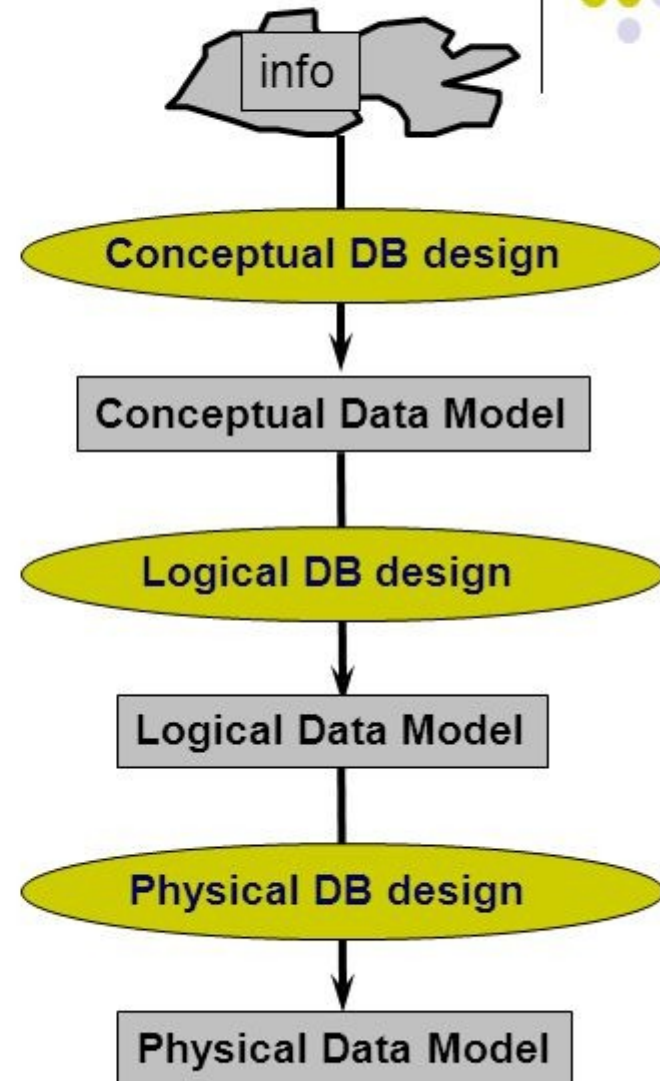
## Entity-relationship Model

Typically used for conceptual database design

## Three Levels of Modeling

## Relational Model

Typically used for logical database design





# Концептуальная модель данных

Концептуальная модель данных (КМД) – это общая информационная модель предметной области, охватывающая вопросы классификации, структуризации и семантической целостности (достоверности и согласованности данных).

Концептуальная модель данных разрабатывается независимо от ограничений, вытекающих из моделей данных, поддерживаемая той или иной СУБД.

Для описания концептуальной модели может быть использован естественный язык, но такое описание будет громоздким и неоднозначным, поэтому для описания КМ чаще всего используется формализованный язык.




# Логическое моделирование

На этапе логического моделирования выявляются сущности, их атрибуты и связи между ними.

**Сущность** - это объект реального мира, который может быть как материальным, так и нет. Пример материальных объектов: покупатель, продукт, автомобиль. Пример нематериальных объектов: заказ, формула, расписание.

С логической точки зрения сущность представляет собой совокупность однотипных объектов называемых экземплярами этой сущности. Экземпляры сущности должны быть уникальными, то есть полный набор значений их атрибутов не должен дублироваться.



**Атрибуты сущности** - это фактически свойства объекта. Например, у покупателя есть фамилия, имя и отчество - это его свойства или атрибуты.

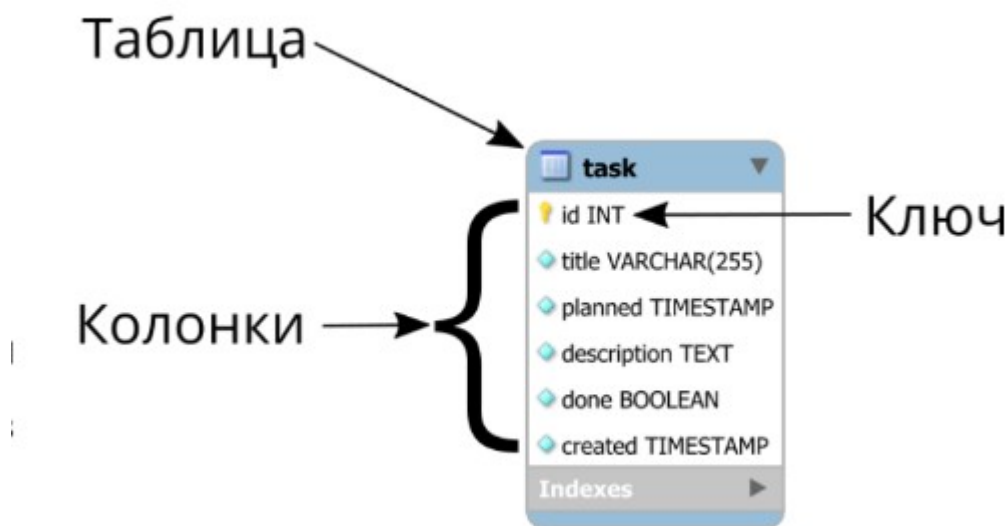
**Атрибуты** могут быть ключевыми и не ключевыми. Например, у покупателя может быть **уникальный идентификатор**, значение которого можно использовать в атрибуте “покупатель” сущности заказ.

На этапе логического проектирования для каждого атрибута обычно определяется примерный тип данных (строковый, числовой, логический, двоичные данные и др.).



# Физическая модель

Физическая модель - это схема базы данных для конкретной СУБД, где сущности предметной области превращаются в таблицы, атрибуты в ее колонки (часто называют столбцами) с использованием конкретного типа данных, связи и ограничения целостности. Строка таблицы - экземпляр одной сущности предметной области.





## Физическая модель

Специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных. Выбор методов управления дисковой памятью, разделение БД по файлам и устройствам (шардирование), методов доступа к данным, создание индексов, и т.д.



# Зачем нужен первичный ключ?

**Первичным ключом (Primary key)** - называется атрибут или набор атрибутов, который уникальным образом идентифицирует сущность. Если первичный ключ состоит более чем из одного атрибута, то его называют составным первичным ключом.

**Каждая сущность должна иметь первичный ключ**, в противном случае вы никогда не сможете однозначно ее идентифицировать. Требуются очень веские причины, почему ваша сущность не имеет первичного ключа, такое встречается, но крайне редко.

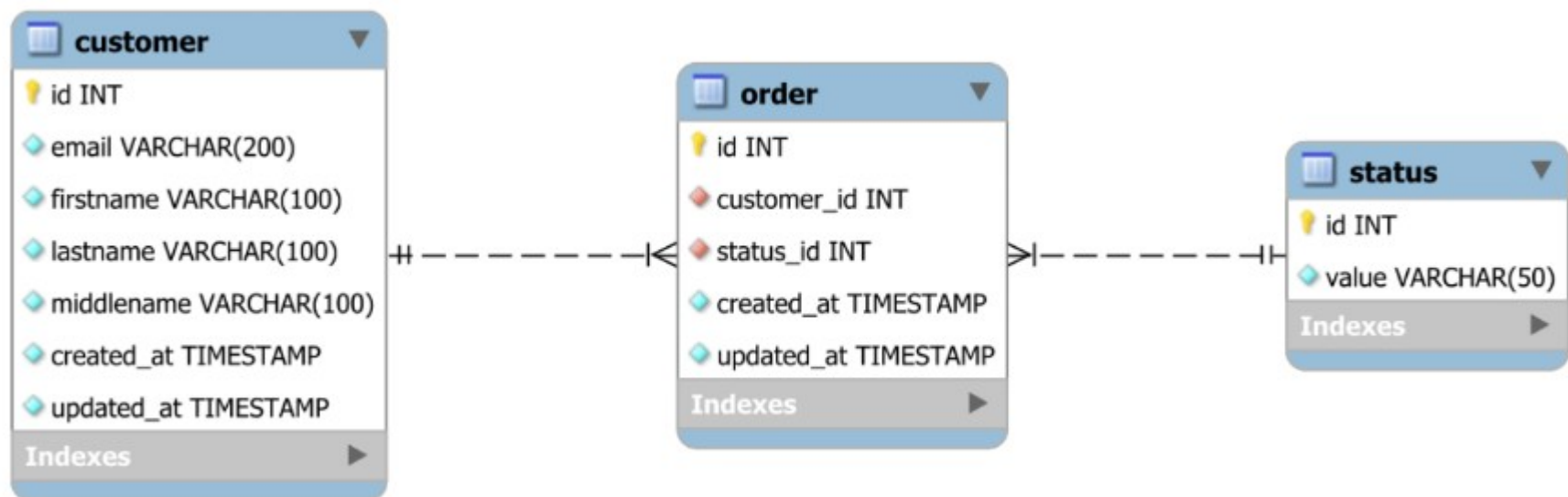


## Зачем нужен внешний ключ?

**Внешние ключи** (Foreign key) - используются для организации связей между таблицами базы данных (родительскими и дочерними) и для поддержания ограничений ссылочной целостности данных.

# Отношение один-ко-многим

Один экземпляр одной сущности может быть связан с множеством экземпляров другой сущности. Например, покупатель может совершить множество заказов в интернет магазине, но один конкретный заказ может принадлежать только одному покупателю. В таблицу order требуется добавить внешний ключ customer\_id. Значением внешнего ключа – будет значение первичного ключа из таблицы customer:





## Примеры

- Один статус заказа можно использовать во множестве заказов, но в один момент времени заказ может иметь только один статус;
- У товара может быть множество отзывов, но конкретный отзыв может быть оставлен только для одного товара;
- В одном отделе может работать множество сотрудников, но конкретный сотрудник работает только в одном отделе;



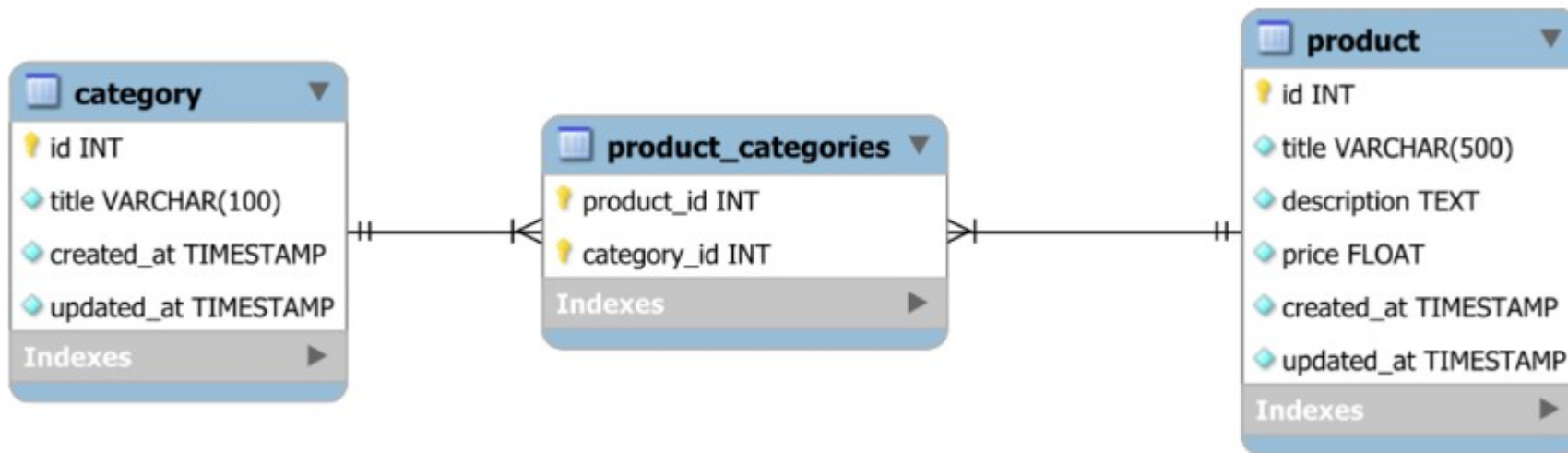
## МНОГИЕ-КО-МНОГИМ

Множество экземпляров одной сущности может быть связано с множеством экземпляров другой сущности. Например, в одну категорию можно добавить множество товаров, однако для удобства пользователя товару можно назначить множество категорий.

В реляционной БД связь многие-ко-многим можно разрешить только через вспомогательную. Таким образом связь многие-ко-многим превращается в две связи один-ко-многим.

## МНОГИЕ-КО-МНОГИМ

В таблицу ассоциации `product_categories` требуется добавить два внешних ключа `product_id` и `category_id`. Значением внешних ключей будут значения первичных ключей из таблиц `product` и `category` соответственно. Чтобы избежать случайного добавления товара в одну категорию дважды, либо присвоения категории одному товару дважды, внешние ключи вместе образуют первичный ключ:





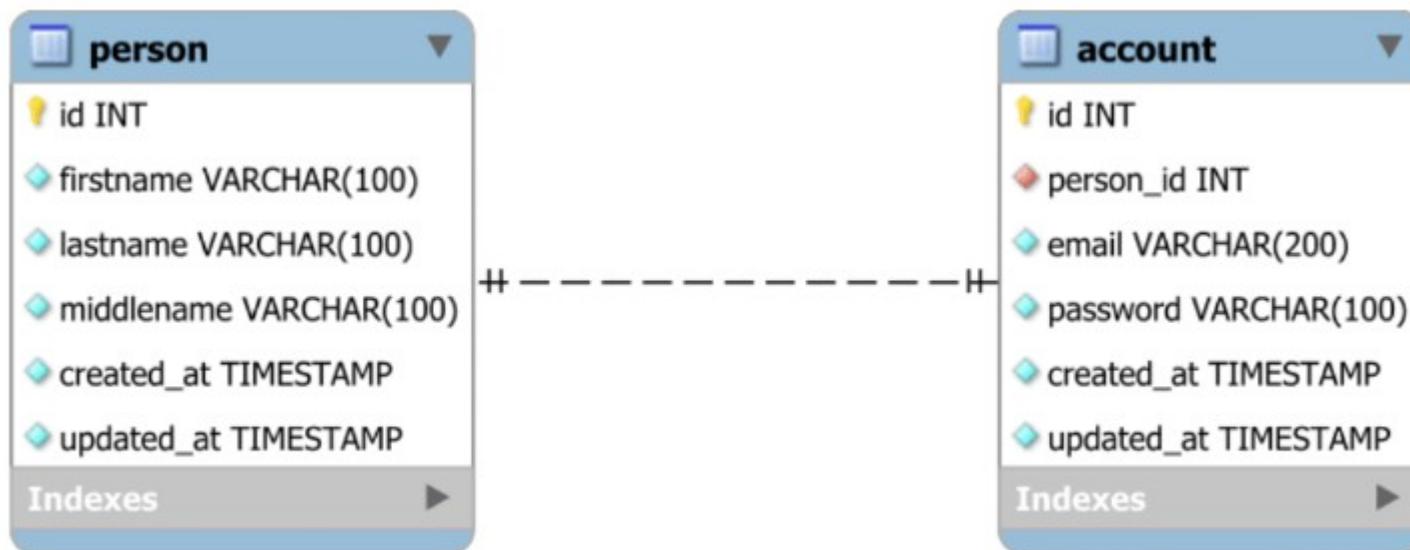


# Примеры многие-ко-многим

- Автор может написать множество книг, но одну книгу может написать множество авторов;
- На курсах дополнительного образования, студент может учиться на множестве курсов, но на одном курсе может учиться множество студентов;
- Посту в инстаграмме можно добавить множество хеш-тегов, но один хеш-тег можно использовать для множества постов;

# Отношения один-к-одному

Один экземпляр одной сущности может быть связан не более чем с одним экземпляром другой сущности. Используется, если необходимо отделить некоторый набор сведений, однозначно связанный с конкретным экземпляром. Связь один-к-одному в БД описывается также, как связь один-ко-многим. В таблицу `account` требуется добавить внешний ключ `person_id`. Значением внешнего ключа - будет значение первичного ключа из таблицы `person`. Если вы хотите избежать случайного добавления второго аккаунта, то значение внешнего ключа `person_id` можно сделать уникальным индексом:





# SQL - Structured Query Language

SQL - язык структурированных запросов, обычно используется в реляционных базах данных для выполнения запросов и состоит из двух подмножеств:

- DDL (Data Definition Language) - язык определения данных;
- DML (Data Manipulation Language) – язык манипулирования данными. Условно, можно сказать, что язык стандартизированный (одинаковый для любой СУБД), однако стандартизированный заканчивается в тот момент, когда в запросе используются специфичные для выбранной СУБД типы данных, функции или иной синтаксис.



# Python Database API

PEP-249 описывает программный интерфейс взаимодействия с БД. Таким образом любой модуль, который работает с БД, должен реализовывать (поддерживать) этот интерфейс. Пример наиболее часто используемых СУБД и модулей, поддерживающих DB API:

- MySQL (MariaDB) - PyMySQL, MySQLdb
- PostgreSQL - psycopg2
- SQLite3 - встроенный модуль sqlite3

Благодаря этому, на примере модуля psycopg2, вы сможете изучить и опробовать все методы. Если в будущем вам понадобится работать с другой СУБД, то достаточно установить сторонний модуль, все методы вам уже знакомы.

Единственное отличие при использовании этих модулей, это аргументы функции connect() - фактически это аргументы конструктора, а сама функция возвращает экземпляр объекта соединения.

# Алгоритм взаимодействия с БД

1. Установка соединения с сервером БД функцией `connect()`
2. Выполнение запроса:
  - 2.1. Получить объект курсора методом `cursor()`
  - 2.2. Выполнить запрос методом `execute()`
  - 2.3. Если запрос на изменение данных или структуры БД:
    - 2.3.1. Нужно зафиксировать изменения методом `commit()`
  - 2.4. Если запрос на получение данных (SELECT):
    - 2.4.1. Фактические данные нужно раз-fetch-ить:
      - `fetchall()` - получить все строки таблицы в список
      - `fetchone()` - получить одну строку из таблицы
      - `fetchmany(N)` - получить нужное кол-во строк (N) из таблицы
3. Закрывать соединение с сервером БД методом `close()`

```
# Note: the module name is psycopg, not psycopg3
import psycopg

# Connect to an existing database
with psycopg.connect("dbname=test user=postgres") as conn:

    # Open a cursor to perform database operations
    with conn.cursor() as cur:

        # Execute a command: this creates a new table
        cur.execute("""
            CREATE TABLE test (
                id serial PRIMARY KEY,
                num integer,
                data text)
            """)

        # Pass data to fill a query placeholders and let Psycopg perform
        # the correct conversion (no SQL injections!)
        cur.execute(
            "INSERT INTO test (num, data) VALUES (%s, %s)",
            (100, "abc'def"))

        # Query the database and obtain data as Python objects.
        cur.execute("SELECT * FROM test")
        cur.fetchone()
        # will return (1, 100, "abc'def")

        # You can use `cur.fetchmany()`, `cur.fetchall()` to return a list
        # of several records, or even iterate on the cursor
        for record in cur:
            print(record)

        # Make the changes to the database persistent
        conn.commit()
```



# Документация

<https://www.psycopg.org/psycopg3/docs/basic/usage.html>

ER – моделирование

<https://editor.ponyorm.com/>