Лабораторная работ №2

Проектирование баз дынных

Процесс проектирования является важным этапом разработки баз данных.

Выделяют три этапа проектирования:

1. Концептуальное проектирование.
2. Логическое проектирование.
3. Физическое проектирование.

Концептуальная проектирование

Процесс концептуального проектирования предполагает разработку концептуальной модели (схемы), которая отражает рассматриваемую предметную область: сущности - объекты предметной области и связи между ними. Проектирование происходит без учёта какой-либо определённой СУБД (обобщённая модель).

При создании концептуальной модели используются графические нотации, например `ER-диаграммы`.

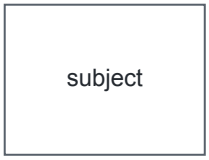
ER-диаграммы

`ER-диаграммы` (`Entity-Relationship` `Сущность-Связь`) - выразительное средство визуального представления концептуальных схем, которое позволяет описать сущности и связи между ними.

Сущности принято именовать в единственном числе:

Пользователи:

Учебные дисциплины:

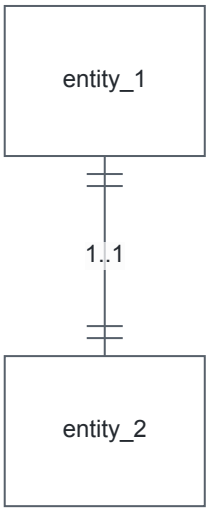


Свези между сущностями отображаются в виде специальных линий. Концы линии определяют тип связи между сущностями.

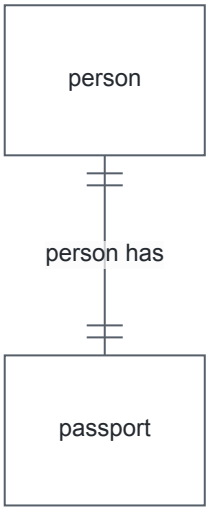
Типы связей

Сущности могут быть связаны друг с другом тремя типами связей:

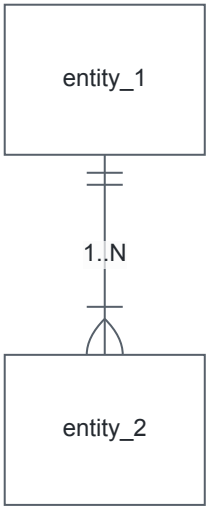
1. Один к одному `1..1`



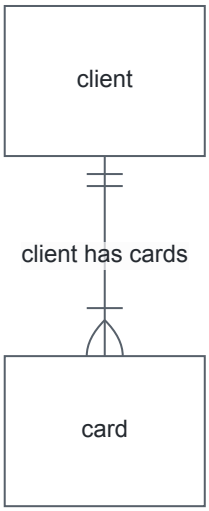
Пример: каждый гражданин должен иметь внутренний паспорт, гражданин может иметь только один действующий паспорт.



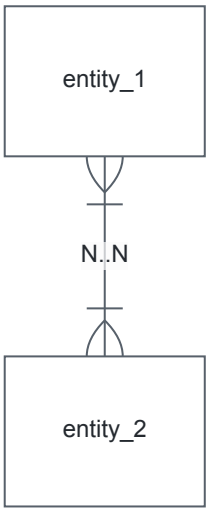
2. Один ко многим `1..N`



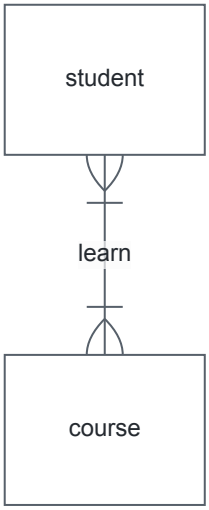
Пример: у клиента банка может быть `1` и `более` банковских карт.



3. Многие ко многим `N..N`

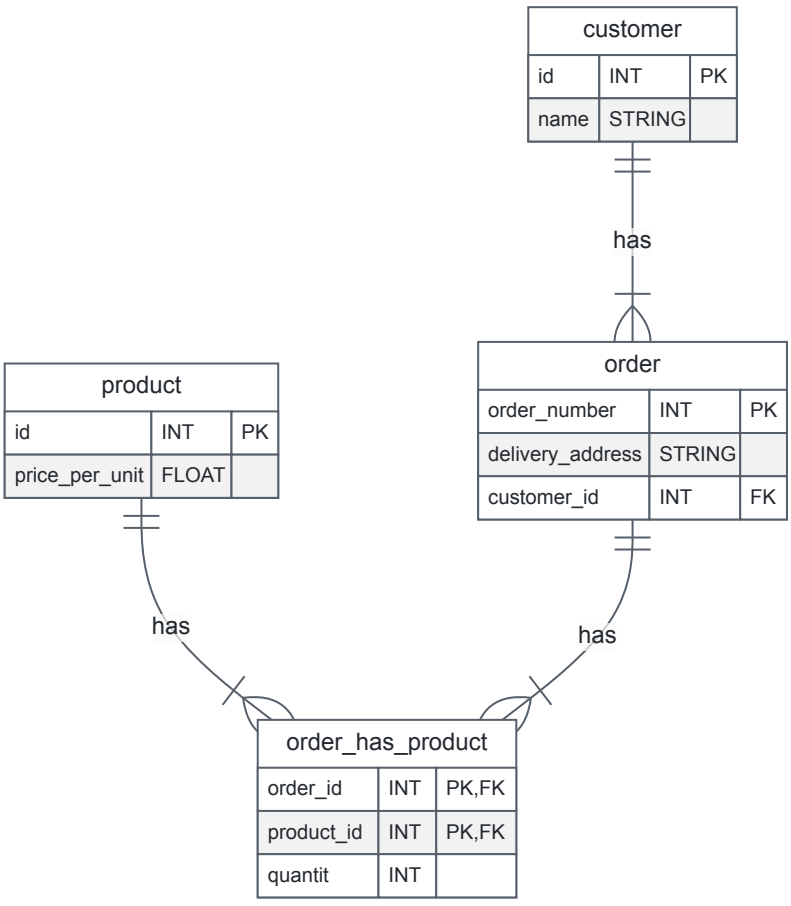


Пример: студенты могут проходить несколько курсов дополнительного образования одновременно.



Логическое проектирование

Логическая проектирование - разработка схемы базы данных, которая включает в себя описание атрибутов, первичных и внешних ключей.
Пример логической схемы, описывающей базу данных, которая содержит информацию о клиентах, товарах и заказах:



Физическое проектирование

Физическое проектирование - разработка схемы базы данных, в которой учитываются особенности конкретной базы данных: типы данных; ограничения, накладываемые на типы данных и именования объектов БД; способы физического хранения на диске и т.д. Результатом данного этапа
проектирования является набор сценариев на языке `SQL`.

Введение SQL

`SQL` (**S**tructured **Q**uery **L**anguage - "язык структурированных запросов") - декларативный язык программирования, используемый в основном для взаимодействия с реляционными базами данных.

`SQL` является реализацией следующий языков:

- `DDL` (Data Definition Language - "язык описания данных") - это специализированный язык, используемых для описания структуры данных: баз, таблиц, индексов и т.д. (`CREATE`, `ALTER`, `DROP`).
- `DML` (Data Manipulation Language - "язык манипуляции данных") - язык управления данными (запросы: `SELECT`, `INSERT`, `UPDATE`, `DELETE`)
- `DCL` (Data Control Language - "язык управления доступом к данным") - язык управления доступом к данным (запросы: `GRANT`, `REVOKE`).

Создание объектов системы управления базами данных `PostgreSQL`

Для создания объектов СУБД используется ключевое слово `CREATE`. Создавать объекты СУБД могут только те пользователи, которым предоставлены соответствующие права.

Некоторые объект СУБД:

- роли;
- базы данных;
- таблицы;
- индексы;
- и т.д.

Роли

Создание роли производится командой `CREATE ROLE`

```
CREATE ROLE ${NAME} WITH ${PARAMS};
```

Где `${NAME}` - имя роли, `${PARAMS}` - список параметров. Полный список параметров создаваемой роли можно получить из официальной документации.

Пример. Создание новой роли `abstract_admin` с параметрами:

- `LOGIN` - предоставляет возможность подключения к СУБД из клиентского приложения;
- `PASSWORD ${VALUE}` - определяет пароль пользователя (`${VALUE}` - значения пароля).
- `SUPERUSER` - определяет статус *суперпользователя* для создаваемой роли;

```
CREATE ROLE abstract_admin WITH
        SUPERUSER
        LOGIN
        PASSWORD '1234';
```

Базы данных

Создание базы данных производится командой `CREATE DATABASE`

```
CREATE DATABASE ${NAME} WITH ${PARAMS};
```

Где `${NAME}` - имя базы данных, `${PARAMS}` - список параметров. Полный список параметров создаваемой базы данных можно получить из официальной документации.

Пример. Создание новой базы `new_database` с параметрами:

- `OWNER ${ROLE}` - задает владельца базы данных (`${ROLE}` - существующая роль);
- `LOCALE ${VALUE}` - задает параметры локали (`${VALUE}` - определённая локаль);
- `CONNECTION LIMIT ${NUMBER}` - верхний предел количества одновременных подключений, по умолчанию `-1` - неограниченно (`${NUMBER}` - число соединений).

```
CREATE DATABASE new_database WITH
        OWNER abstract_admin
        LOCALE 'ru_RU.UTF-8'
        CONNECTION LIMIT 100;
```

Таблицы

Таблица создается в определённой базе данных: необходимо иметь подключение к СУБД с указанием конкретной базы.

Создание таблицы производится командой `CREATE TABLE`:

```
CREATE TABLE IF NOT EXISTS ${TABLE_NAME} (
-- COLUMNS
);
```

В круглых скобках перечисляется список столбцов (атрибутов) таблицы. Каждый атрибут отделяется друг от друга запятыми.

Каждый столбец (атрибут) записывается следующим образом:

```
${NAME} ${TYPE} ${PARAMS}
```

где `${NAME}` - имя столбца (атрибута), `${TYPE}` - тип, `${PARAMS}` - список дополнительных параметров столбца.

Со списком поддерживаемых типов можно ознакомится на странице официальной документации.

К дополнительным параметрам можно отнести:

- `NOT NULL` - указывает, что записываемое значение не может быть `NULL`;
- `UNIQUE` - указывает, что все значения данного столбца являются уникальными;
- `DEFAULT ${VALUE}` - указание того, что при вставке записи в таблицу, будет использовано значение по умолчанию если в записываемой строке отсутствует токовое;
- `PRIMARY KEY` - указания того, что столбец является первичным ключом;

С полным перечнем вариантов создания таблиц можно ознакомится на странице официальной документации.

Пример. Создание таблицы описывающей автомобиль с атрибутами: модель, дата выпуска и цвет.

```
CREATE TABLE IF NOT EXISTS car (
        id SERIAL,
        model TEXT NOT NULL,
        created_on DATE NOT NULL DEFAULT NOW()::DATE,
        color TEXT NOT NULL DEFAULT 'unknown',
```

```
        PRIMARY KEY(id)
);
```

Необходимо обратить внимание на `SERIAL` - данный тип, который позволяет автоматически инкрементировать значение (увеличивать на единицу). Обычно такой тип данных используется для `первичных ключей`.

Индексы

Индексация используется для ускорения доступа к данным таблицы.

Индексы создаются для определенного столбца (атрибута) таблицы. Создание индекса производится командой `CREATE INDEX`

В самом простом случае индекс таблицы может быть создан следующим образом:

```
CREATE INDEX IF NOT EXISTS ${INDEX_NAME} on ${TABLE_NAME}(${COLUMN_NAME});
```

Где

- `${INDEX_NAME}` - название индекса;
- `${TABLE_NAME}` - имя базы;
- `${COLUMN_NAME}` - название столбца;

Детальное описание процесса создание индекс представлено в официальной документации.

Пример. Существует таблица `person` описывающая человека:

```
CREATE TABLE IF NOT EXISTS person (
        id SERIAL,
        name TEXT NOT NULL,
        surname TEXT NOT NULL,
        age SMALINT NOT NULL,
        PRIMARY KEY(name, surname, age)
);
```

Необходимо обеспечить ускорения доступа к данным по фамилии. Для этого создаётся индекс:

```
CREATE INDEX IF NOT EXISTS id_surname
        ON person(surname);
```

Удаление объектов системы управления базами данных PostgreSQL

Удаление объектов СУБД производится командой `DROP`:

- `DROP INDEX IF EXISTS ${INDEX_NAME}` - удаление индекса;
- `DROP TABLE IF EXISTS ${TABLE_NAME}` - удаление таблицы;
- `DROP ROLE IF EXISTS ${ROLE_NAME}` - удаление роли;
- `DROP DATABASE ${DATABASE_NAME}` - удаление базы.

Изменение объектов системы управления базами данных PostgreSQL

Изменения объектов СУБД производится командой `ALTER`, примеры работы будут рассматриваться в следующей лабораторной работе.

Управление записями таблиц

Под управлением записями (строками) понимается четыре основные операции:

- `INSERT` - вставка записи;
- `SELECT` - выборка (получение) записи из таблицы.
- `UPDATE` - обновление записи;
- `DELETE` - удаление записи;

Вставка записи

Вставка записи в таблицу осуществляется командой:

```
INSERT INTO ${TABLE_NAME}
        [(${COL_1}, ${COL_2}, ...)]
        VALUES(${VAL_1}, ${VAL_2}, ...);
```

`${TABLE_NAME}` - имя таблицы. После имени таблицы может указываться список столбцов в порядке отличном от того, что указан в таблице. Используя список столбцов можно осуществить частичную вставку: записывается только часть параметров, остальные берутся из значений по умолчанию.

Подробная информация изложена на странице официальной документации

Пример. Вставка записей в таблицу `car`:

```
CREATE TABLE IF NOT EXISTS car (
        id SERIAL,
        model TEXT NOT NULL,
        created_on DATE NOT NULL DEFAULT NOW()::DATE,
        color TEXT NOT NULL DEFAULT 'unknown',
```

```
        PRIMARY KEY(id)
);
```

```
INSERT INTO car(model) VALUES('type-m');

INSERT INTO car(model, created_on, color) VALUES('type-s', '2021-10-
21'::DATE, 'red');

INSERT INTO car(model, created_on, color) VALUES('type-s', '2022-12-
10'::DATE, 'grey');

INSERT INTO car(model, created_on, color) VALUES('type-m', '2022-12-
10'::DATE, 'red');
```

Выборка записей

Выборка записей (строк) производится командой:

```
SELECT * | ${ATT_1}, ${ATTR_2} as ${ALIAS ATTR_2} FROM ${TABLE_NAME} WHERE
${CONDITION}
```

В секции `WHERE` указываются критерии фильтрации - набор логических выражений объединённых логическими операторами:

- AND - И;
- OR - ИЛИ;
- NOT - НЕ.

Пример. Выборка записей из таблицы, созданной на предыдущем шаге, записи соответствуют условию `model = 'type_s'`:

```
SELECT * FROM car WHERE model = 'type_s';
```

Обновление записей

Обновление записи производится командой:

```
UPDATE ${TABLE_NAME} SET ${COLUMN_1} = ${VALUE_1}, ${COLUMN_2} =
${VALUE_2}, ... WHERE ${CONDITION};
```

Обновлять можно как один, так и более атрибутов. Если отсутствует секция `WHERE`, произойдет обновление во всех строках (записях) таблицы

Пример. Обновление строк таблицы, созданной на предыдущем шаге. Критерии фильтрации `model = 'type-s' AND color = 'unknown'`:

```
UPDATE car SET color = 'red' WHERE model = 'type-s' AND color = 'unknown';
```

Удаление записи

Удаление записей из таблицы производится с помощью команды:

```
DELETE FROM ${TABLE_NAME} WHERE ${CONDITION};
```

Пример. Удаление всех записей, соответствующих условию `color = 'grey'`:

```
DELETE FROM car WHERE color = 'grey';
```

Если отсутствует секция `WHERE`, то будут удалены все строки таблицы. Такой же результат можно получить с помощью выполнения команды:

```
TRUNCATE ${TABLE_NAME};
```

Задание

Необходимо спроектировать базу данных `university`, которая могла бы быть использована для контроля успеваемости студентов в университете.

База данных должна учитывать:

- группы (минимум 2);
- студентов (минимум по 8 человек на группу);
- дисциплины (минимум 8 предметов);
- преподавателей (минимум 2 человека на каждую дисциплину);
- успеваемость студентов по дисциплинам (студент должен иметь минимум по 2 оценки за каждую дисциплину).

План работ

1. Необходимо разработать концептуальную схему.
2. Необходимо разработать логическую схему.
3. Разработать программную реализацию на языке `SQL` для СУБД `PostgreSQL`.
4. Программная реализация должна содержать:
   4.1. Роль для подключения к базе.
   4.2. Базу данных.
   4.3. Необходимые таблицы и индексы для них.

5. Таблицы должны быть заполнены данными в соответствии с заданием.