Тема 2: Проектиране на база от данни

1. Какво означава „проектиране на база от данни“

Кратък извод: Проектирането е процесът на превръщане на реалния свят в логическа структура от таблици, връзки и правила, които гарантират коректност и ефективност.

2. Етапи на проектиране (подробно)

2.1. Анализ на изискванията

Цел: Да разберем какви данни трябва да се съхраняват и как ще се използват.

- Какви обекти съществуват?
- Какви свойства имат?
- Какви връзки има между тях?
- Какви операции ще се извършват?

Пример: Проектираме база за **училищна библиотека**. Нужни са данни за:

- книги;
- автори;
- ученици;
- заемания.

2.2. Концептуално моделиране (ER модел)

ER моделът описва обектите (Entity), техните свойства (Attributes) и връзките (Relationships).

Основни елементи:

- **Entity** – обект (Книга, Автор, Ученик).
- **Attribute** – свойство (заглавие, година, клас).
- **Relationship** – връзка (Автор → Книга; Ученик → Заемане).

Примерни обекти и връзки:

- Един **автор** може да има **много книги** → връзка 1:N.
- Един **ученик** може да заема **много книги** → връзка 1:N.
- Една **книга** може да бъде заемана много пъти → връзка 1:N.

2.3. Логически модел (таблици)

Преобразуваме ER модела в таблици.

Примерни таблици:

- **Authors**
  - id (PK)
  - name
  - country
- **Books**
  - id (PK)
  - title
  - year
  - author_id (FK → Authors.id)
- **Students**
  - id (PK)
  - name
  - class
- **Borrowings**
  - id (PK)
  - student_id (FK → Students.id)
  - book_id (FK → Books.id)
  - date_borrowed
  - date_returned

2.4. Ключове и ограничения

Ключовете осигуряват уникалност и връзки, а ограниченията гарантират коректност на данните.

- **Първичен ключ** (Primary Key, PK) - Уникален идентификатор на всеки запис. Не допуска NULL.
- **Външен ключ** (Foreign Key, FK) - Свързва таблици. Гарантира референтна цялостност.
- **Уникален ключ** (UNIQUE) - Гарантира уникалност, но допуска NULL.

- **Съставен ключ** - Състои се от повече от една колона.

2.5. Нормализация

Цел: да се избегнат излишни данни и зависимости.

1NF – Първа нормална форма

- Няма повтарящи се групи.
- Всички стойности са атомарни.

2NF – Втора нормална форма

- Няма частични зависимости от съставен ключ.

3NF – Трета нормална форма

- Няма транзитивни зависимости (атрибут зависи от друг атрибут, а не от ключа).

Пример: Ако в таблица Books има поле author_name, това нарушава 3NF → трябва да е в Authors.

2.6. Физически модел

Това е етапът, в който логическият модел на базата данни се превръща в реална, техническа структура, която може да бъде изпълнена от конкретна СУБД (MySQL, SQL Server, PostgreSQL). Тук вече говорим за таблици, типове данни, индекси, ключове, ограничения, файлове за съхранение и оптимизация.

3. Основни типове данни в базите от данни (SQL)

Типовете данни определят какъв вид информация може да се съхранява в дадена колона — числа, текст, дати, булеви стойности и др. Те са фундаментални за правилното проектиране на таблици и за гарантиране на точност и ефективност.

3.1. Числови типове данни

Тип	Описание	Пример
INT	Цели числа (обикновено от −2 147 483 648 до 2 147 483 647)	age INT
SMALLINT	По-малки цели числа	year SMALLINT
BIGINT	Много големи цели числа	id BIGINT
DECIMAL(p,s)	Десетични числа с фиксирана точност (p – брой цифри, s – брой след десетичната точка)	price DECIMAL(8,2)
FLOAT / REAL	Числа с плаваща запетая (приблизителна точност)	temperature FLOAT

3.2. Текстови типове данни

Тип	Описание	Пример
CHAR(n)	Текст с фиксирана дължина (винаги n символа)	code CHAR(5)
VARCHAR(n)	Текст с променлива дължина (до n символа)	name VARCHAR(100)
TEXT	Дълъг текст (няма ограничение в някои СУБД)	description TEXT

Забележка:

- **В MySQL и PostgreSQL VARCHAR и TEXT са най-често използвани.**
- **В SQL Server има и NVARCHAR за текст с Unicode символи.**

3.3. Дата и време

Тип	Описание	Пример
DATE	Само дата (година-месец-ден)	date_borrowed DATE
TIME	Само час	start_time TIME

Тип	Описание	Пример
DATETIME / TIMESTAMP	Дата и час заедно	created_at TIMESTAMP

Разлики:

- **В MySQL DATETIME и TIMESTAMP са сходни, но TIMESTAMP се обновява автоматично при промяна.**
- **В PostgreSQL има и INTERVAL за времеви разлики.**

3.4. Булеви типове

Тип	Описание	Пример
BOOLEAN	Истина/лъжа (TRUE/FALSE)	is_active BOOLEAN
BIT	В SQL Server – 0 или 1	is_returned BIT

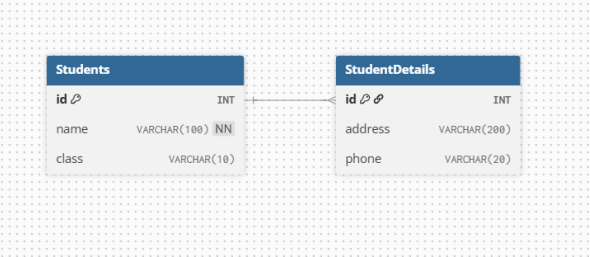
3.5. Други често използвани типове

Тип	Описание	Пример
ENUM	Списък от предварително дефинирани стойности (MySQL)	status ENUM('borrowed','returned')
BLOB	Двоични данни (изображения, файлове)	photo BLOB
JSON	Структурирани данни във формат JSON (PostgreSQL, MySQL)	metadata JSON

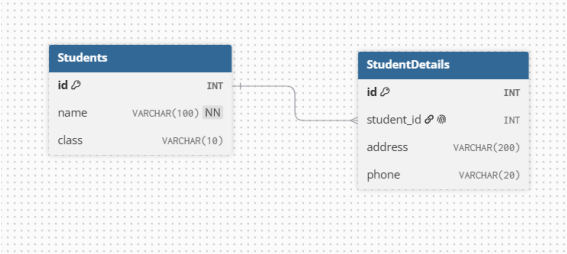
4. Практически пример:

4.1. Проектиране на база данни чрез използване на връзки 1:1

Students ↔ StudentDetails



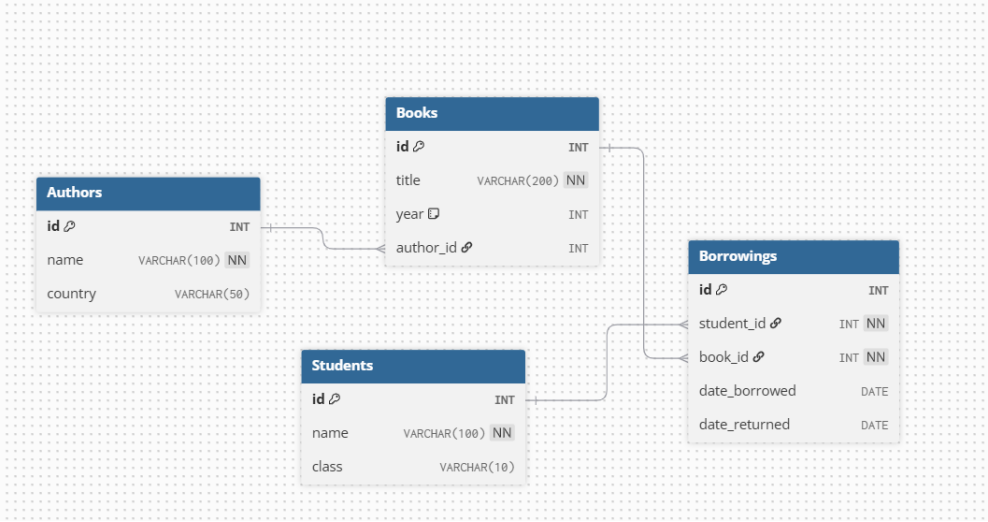
1:1 със споделен PK



1:1 чрез UNIQUE FK

4.2. Проектиране на база данни чрез използване на връзки 1:N и N:M

ER диаграма – Вариант 1



Връзки:

- Author 1:N Book

Един автор може да има много книги, но всяка книга може да има точно един автор.

- Student 1:N Borrowing

Един ученик може да има много заемания, като всяко заемане принадлежи на точно един ученик.

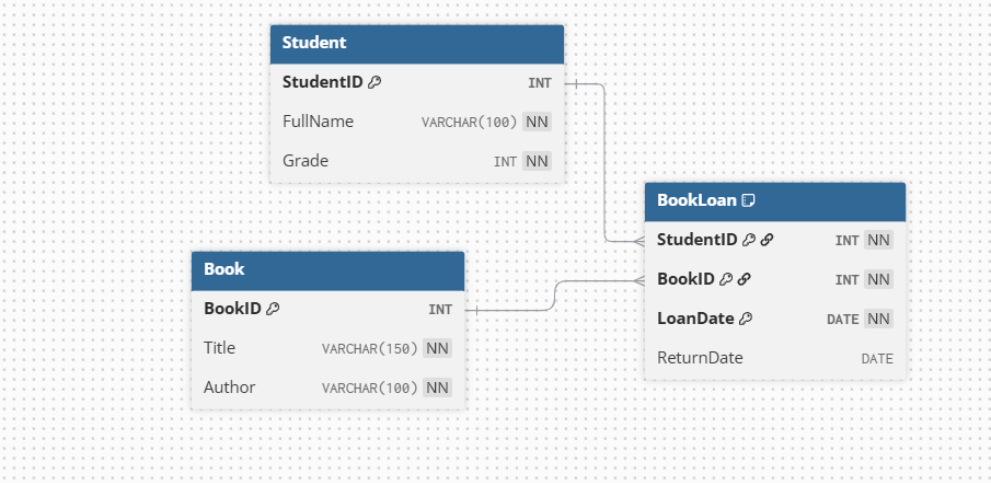
- Book 1:N Borrowing

Една книга може да бъде заемана много пъти, като всяко заемане се отнася към точно една книга.

- Students ↔ Books чрез Borrowings (много към много)

Един ученик може да заема много книги. Една книга може да бъде заемана от много ученици. Таблицата Borrowings е междинна таблица, която реализира тази връзка.

ER диаграма – Вариант 2 (чрез съставен ключ)



Връзки:

- Author 1:N BookLoan
- Book 1:N BookLoan
- N:M чрез LoanDate

Какво означава връзката

- Един ученик може да заема много книги.
- Една книга може да бъде заемана от много ученици.
- Тази връзка се реализира чрез **междинната таблица BookLoan**.

Ролята на съставния ключ

Съставният ключ **(StudentID, BookID, LoanDate)** позволява:

- един ученик да заема **една и съща книга многократно**, но в различни дати;
- да няма дублиращи записи за заемане в една и съща дата;
- да се съхранява пълна история на заеманията.

Таблицата вече има естествен уникален идентификатор → комбинацията от трите полета. Няма нужда от поле id.