Идентификатори, Бази данни

2. Увод

2.1 Идентификаторите гарантират уникалност на записите и коректни връзки между таблиците. Те са основа за референтна цялост и бързи заявки.

3. Основни понятия

- **3.1 Първичен ключ (PK)**: уникален и NOT NULL идентификатор на ред; стабилен във времето. Пример: `StudentId SERIAL PRIMARY KEY`.
- **3.2 Външен ключ (FK)**: колон(и), които сочат към PK/UNIQUE на друга таблица и гарантират валидна връзка. Пример: `FOREIGN KEY (StudentId) REFERENCES Students(StudentId)`.
- **3.3 Уникален ключ (UNIQUE)**: налага уникалност на колона/комбинация, различен от PK. Пример: `Email UNIQUE`.
- **3.4 Естествен ключ**: бизнес-значим атрибут, по който записът е уникален. Примери: `EGN`, `IBAN`, `ISBN`.
- **3.5 Заместителен ключ (surrogate)**: изкуствен идентификатор без бизнес смисъл (INT/UUID).
  - INT (Auto Increment / Serial)
  - UUID - универсално уникален идентификатор. Това е стандартен начин за създаване на уникални идентификатори, които могат да се използват в различни системи без риск от дублиране
- **3.6 Композитен ключ**: PK/UNIQUE, съставен от няколко колони. Пример: `PRIMARY KEY (StudentId, CourseId)`.
- **3.7 Референтна цялост, каскадни операции, индекси**: правила и механизми за валидни връзки (FK), действия при промяна/изтриване (`RESTRICT`, `CASCADE`, `SET NULL`), и структури за бързо търсене (индекси върху PK/FK/UNIQUE).

4. Цели

- **4.1** Разграничаване на естествени и заместителни ключове
- **4.2** Дефиниране и прилагане на PK, FK, UNIQUE
- **4.3** Избор на подходящи идентификатори в модел и SQL

5. Съдържание

5.1 Въведение

- Общ преглед на важността на идентификаторите в базите данни
- Без уникални идентификатори няма как да свържем данни между таблици
- Пример: без StudentId не можем да свържем студент с неговите записи

5.2 Основни понятия

- **PK (първичен ключ)**: уникален идентификатор на ред (StudentId)
- **FK (външен ключ)**: сочи към PK на друга таблица (Enrollment.StudentId → Student.StudentId)
- **UNIQUE**: налага уникалност без да е PK (Email UNIQUE)
- **Естествен ключ**: бизнес-значим (ЕГН, ISBN)
- **Заместителен ключ**: изкуствен (SERIAL, UUID)
- **Композитен ключ**: комбинация от колони (StudentId + CourseId)
- **Референтна цялост**: гарантира валидни връзки между таблици

5.3 Видове идентификатори

- **Естествен ключ**: предимства/недостатъци, примери (ЕГН, IBAN)
  - Пример: `CountryCode CHAR(2) PRIMARY KEY` (BG, US, DE)
  - Подходящ за кодове със стандарт (IBAN, ISO кодове на държави, EAN/UPC)
  - Когато промяна е изключение
- **Заместителен ключ**: INT/UUID, кога е уместен
  - Пример: `UserId BIGSERIAL PRIMARY KEY` (1, 2, 3, ...)
  - Когато бизнес атрибутите могат да се променят

    - За мащабируемост, разделяне на отговорности и анонимизация
- **Композитен ключ**: случаи на употреба и ефекти върху FK/индекси
    - Пример: `PRIMARY KEY (StudentId, CourseId)` за записвания
    - Подходящи при естествена уникалност от комбинация
    - Усложнява FK и индекси; може да влоши производителността при големи таблици

> **Примери:**
>
> **Естествен ключ като PK:**
>
> ```
> CREATE TABLE Countries (
>   IsoCode CHAR(2) PRIMARY KEY,
>   Name VARCHAR(100) NOT NULL UNIQUE
> );
> ```
>
> **Заместителен ключ и алтернативна уникалност:**
>
> ```
> CREATE TABLE Users (
>   UserId BIGSERIAL PRIMARY KEY,
>   Email CITEXT UNIQUE NOT NULL,
>   DisplayName VARCHAR(120) NOT NULL
> );
> ```

5.4 Правила за първични ключове (PK)

- **Минималност**: използвай най-малко колони възможно
- **Стабилност**: не трябва да се променя
- **Индексиране**: автоматично се създава клъстерен индекс
- **Избор на тип**: INT за производителност, UUID за разпределени системи

5.5 Външни ключове (FK)

- **ON DELETE/UPDATE действия**:
    - RESTRICT: забранява изтриване (ако има зависими записи)
    - CASCADE: изтрива зависимите записи автоматично

- SET NULL: занулява FK (ако колоната позволява NULL)
- SET DEFAULT: задава стойност по подразбиране

- **Индексиране**: подобрява JOIN операции и каскадни действия

Примери:

Композитен ключ за много-към-много връзка:

Таблица Authors:

```
CREATE TABLE Authors (
  AuthorId BIGSERIAL PRIMARY KEY,
  FullName VARCHAR(120) NOT NULL
);
```

Таблица Books:

```
CREATE TABLE Books (
  BookId BIGSERIAL PRIMARY KEY,
  ISBN CHAR(13) UNIQUE,
  Title VARCHAR(200) NOT NULL
);
```

Junction таблица BookAuthors:

```
CREATE TABLE BookAuthors (
  BookId BIGINT NOT NULL,
  AuthorId BIGINT NOT NULL,
  Role VARCHAR(50) NOT NULL DEFAULT 'author',
  PRIMARY KEY (BookId, AuthorId),
  FOREIGN KEY (BookId) REFERENCES Books(BookId)
    ON DELETE CASCADE,
  FOREIGN KEY (AuthorId) REFERENCES Authors(AuthorId)
    ON DELETE CASCADE
);
```

Индекс за производителност:

```
CREATE INDEX idx_bookauthors_author
ON BookAuthors(AuthorId);
```

5.6 Уникалност и алтернативни ключове

- **UNIQUE върху колони**: `Email VARCHAR(100) UNIQUE`
- **UNIQUE върху комбинации**: `UNIQUE (FirstName, LastName, BirthDate)`
- **Бизнес правила**: един студент не може да се запише два пъти в същия курс
- **Частични индекси**: уникалност само за активни записи
    - Пример: `CREATE UNIQUE INDEX ... WHERE Active = TRUE`

> **Примери:**
>
> **Частичен уникален индекс за активни записи:**
>
> ***Създаване на таблицата:***

```
CREATE TABLE PromoCodes (
  PromoId BIGSERIAL PRIMARY KEY,
  Code TEXT NOT NULL,
  Active BOOLEAN NOT NULL DEFAULT TRUE
);
```

> ***Частичен индекс (само за активни записи):***

```
CREATE UNIQUE INDEX uniq_promocodes_active_code
ON PromoCodes(Code)
WHERE Active = TRUE;
```

5.7 Проверки и бизнес правила

- **CHECK ограничения**: валидиране на формати и стойности
    - Пример: `CHECK (Age >= 18 AND Age <= 100)`
    - Пример: `CHECK (EGN ~ '^[0-9]{10}$')` за български ЕГН
- **Валидация на дати**: `CHECK (BirthDate <= CURRENT_DATE)`

Примери:

CHECK ограничения за валидация:

```
CREATE TABLE StudentsEx (
  StudentId BIGSERIAL PRIMARY KEY,
  EGN CHAR(10) UNIQUE NOT NULL,
  BirthDate DATE NOT NULL,
  CHECK (EGN ~ '^[0-9]{10}$'),
  CHECK (BirthDate <= CURRENT_DATE)
);
```

5.8 Производителност и индексиране

- **Комбинирани индекси**: за чести заявки с WHERE и ORDER BY
  - Пример: `CREATE INDEX idx_student_grade ON Grades(StudentId, SubjectId, Date)`
- **Покриващи индекси**: включват всички нужни колони за заявката
- **Баланс**: повече индекси = по-бавни INSERT/UPDATE, по-бързи SELECT

Примери:

Покриващ индекс за JOIN и филтър:

```
CREATE INDEX idx_enrollments_student_course
ON Enrollments(StudentId, CourseId, EnrolledAt);
```

Обяснение: Този индекс покрива заявки като:

```
SELECT * FROM Enrollments
WHERE StudentId = 123
AND CourseId = 456
ORDER BY EnrolledAt;
```

5.9 PK типове: INT срещу UUID (v4, v7)

- **INT/BIGINT**: компактни (4-8 байта), бързи JOIN, разкриват броя записи
  - Пример: `StudentId SERIAL PRIMARY KEY` (1, 2, 3, ...)
- **UUID v4**: случайни, глобално уникални, по-бавни
  - Пример: `550e8400-e29b-41d4-a716-446655440000`
- **UUID v7**: времево подредени, по-добра производителност от v4

> **Примери:**
>
> **UUID v4 (PostgreSQL):**
>
> ***Активиране на UUID функцията:***
>
> ```
> CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
> ```
>
> ***Таблица с UUID PK:***
>
> ```
> CREATE TABLE Orders (
>   OrderId UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
>   CustomerId BIGINT NOT NULL,
>   Total NUMERIC(12,2) NOT NULL,
>   CreatedAt TIMESTAMP NOT NULL DEFAULT NOW()
> );
> ```

5.10 Генериране на идентификатори

- **SERIAL/IDENTITY**: автоматично нарастващи числа
  - PostgreSQL: `StudentId SERIAL PRIMARY KEY`
  - SQL Server: `StudentId INT IDENTITY(1,1) PRIMARY KEY`
- **UUID функции**: `uuid_generate_v4()`, `gen_random_uuid()`

5.11 Пример (SQL): студенти, курсове, записвания

- **Студенти**: `Students(StudentId, EGN, FullName)`
- **Курсове**: `Courses(CourseId, Title)`

- **Записвания**: `Enrollments(StudentId, CourseId, Date)` с композитен PK
- Практически пример с три свързани таблици и различни типове ключове

Практически пример:

Студенти, курсове и записвания:

Таблица Students:

```
CREATE TABLE Students (
  StudentId SERIAL PRIMARY KEY,
  EGN CHAR(10) UNIQUE NOT NULL,
  FullName VARCHAR(100) NOT NULL
);
```

Таблица Courses:

```
CREATE TABLE Courses (
  CourseId SERIAL PRIMARY KEY,
  Title VARCHAR(100) NOT NULL
);
```

Таблица Enrollments (композитен PK):

```
CREATE TABLE Enrollments (
  StudentId INT NOT NULL,
  CourseId INT NOT NULL,
  EnrolledAt TIMESTAMP NOT NULL DEFAULT NOW(),
  PRIMARY KEY (StudentId, CourseId),
  FOREIGN KEY (StudentId) REFERENCES Students(StudentId)
    ON DELETE CASCADE,
  FOREIGN KEY (CourseId) REFERENCES Courses(CourseId)
    ON DELETE CASCADE
);
```

5.12 Поведение на FK: демонстрации с различни действия

- **Тестване на RESTRICT**: опит за изтриване на студент с записи
- **Тестване на CASCADE**: автоматично изтриване на всички записи при изтриване на студент

- **Тестване на SET NULL**: зануляване на FK при изтриване на родителски запис

5.13 UNIQUE и частични индекси: примери

- **Частични индекси**: уникалност само за активни записи
  - Пример: `CREATE UNIQUE INDEX uniq_active_email ON Users(Email) WHERE Active = TRUE`
- **Практически случаи**: промо кодове (само активни да са уникални)

5.14 CHECK: примери

- **Валидация на ЕГН формат**: `CHECK (EGN ~ '^[0-9]{10}$')`
- **Валидация на дати**: `CHECK (BirthDate <= CURRENT_DATE)`
- **Валидация на стойности**: `CHECK (Grade >= 2 AND Grade <= 6)`
- **Валидация на проценти**: `CHECK (Discount >= 0 AND Discount <= 100)`

5.15 Миграция на ключове: стъпки и рискове

- **Стъпки**: добавяне на нова PK колона → пренапълване → премахване на старата
- **Рискове**: загуба на данни, нарушение на връзки, downtime
- **План**: backup, тестване, прозорец за поддръжка
  - Пример: `ALTER TABLE Students ADD COLUMN NewStudentId BIGSERIAL`

> **Примери:**
>
> ```
> -- Скелет на миграция (PostgreSQL)
> ALTER TABLE Students ADD COLUMN NewStudentId BIGSERIAL;
> UPDATE Students s SET NewStudentId =
> nextval(pg_get_serial_sequence('Students','NewStudentId')) WHERE
> NewStudentId IS NULL;
> ALTER TABLE Students ADD CONSTRAINT Students_pk PRIMARY KEY
> ```

```
(NewStudentId);
-- Реконфигурирайте FK в зависимите таблици към NewStudentId
```

5.16 По-комплексен пример: библиотека (читатели, книги, заемания)

- **Читатели**: `Readers(ReaderId, Email UNIQUE, FullName)`
- **Книги**: `LibraryBooks(BookId, ISBN UNIQUE, Title, Copies CHECK >= 0)`
- **Заемания**: `Loans(LoanId, ReaderId FK, BookId FK, DueAt, ReturnedAt)`
- Пълна система с всички типове ключове, ограничения и индекси

Примери:

По-комплексен пример: библиотека

Таблица Readers (Читатели):

```
CREATE TABLE Readers (
  ReaderId BIGSERIAL PRIMARY KEY,
  Email CITEXT UNIQUE NOT NULL,
  FullName VARCHAR(120) NOT NULL
);
```

Таблица LibraryBooks (Книги):

```
CREATE TABLE LibraryBooks (
  BookId BIGSERIAL PRIMARY KEY,
  ISBN CHAR(13) UNIQUE,
  Title VARCHAR(200) NOT NULL,
  Copies INT NOT NULL CHECK (Copies >= 0)
);
```

Таблица Loans (Заемания):

```
CREATE TABLE Loans (
  LoanId BIGSERIAL PRIMARY KEY,
  ReaderId BIGINT NOT NULL,
  BookId BIGINT NOT NULL,
```

```
  LoanedAt TIMESTAMP NOT NULL DEFAULT NOW(),
  DueAt TIMESTAMP NOT NULL,
  ReturnedAt TIMESTAMP,
  UNIQUE (ReaderId, BookId, ReturnedAt),
  FOREIGN KEY (ReaderId) REFERENCES Readers(ReaderId)
    ON DELETE CASCADE,
  FOREIGN KEY (BookId) REFERENCES LibraryBooks(BookId)
    ON DELETE RESTRICT
);
```

Индекс за активни заемания:

```
CREATE INDEX idx_loans_reader_active
ON Loans(ReaderId)
WHERE ReturnedAt IS NULL;
```

5.17 Примери и задачи: практическа работа

- **Дизайн на таблици**: студентски групи, система за поръчки
- **Избор на ключове**: естествен vs заместителен за различни случаи
- **Тестване на FK**: различни ON DELETE поведения
- **CHECK ограничения**: валидиране на бизнес правила
- **Индексиране**: оптимизация на заявки

Практически задачи:

- Дизайнирай таблици за студентски групи: студенти, групи, членства; избери PK/FK и индекси
- Реализирай система за поръчки: клиенти, поръчки, редове на поръчка; обсъди INT срещу UUID
- Направи ON DELETE CASCADE за зависими записи и тествай ефекта при изтриване
- Добави CHECK ограничения за стойности (напр. процент 0-100) и покажи нарушаване
- Създай частичен уникален индекс за уникалност само на активни потребители

6. Пример (SQL)

```
CREATE TABLE Students (
  StudentId SERIAL PRIMARY KEY,
  EGN CHAR(10) UNIQUE NOT NULL,
  FullName VARCHAR(100) NOT NULL
);

CREATE TABLE Courses (
  CourseId SERIAL PRIMARY KEY,
  Title VARCHAR(100) NOT NULL
);

CREATE TABLE Enrollments (
  StudentId INT NOT NULL,
  CourseId INT NOT NULL,
  EnrolledAt TIMESTAMP NOT NULL DEFAULT NOW(),
  PRIMARY KEY (StudentId, CourseId),
  FOREIGN KEY (StudentId) REFERENCES Students(StudentId) ON DELETE
CASCADE,
  FOREIGN KEY (CourseId) REFERENCES Courses(CourseId) ON DELETE CASCADE
);
```

7. Заключение

7.1 Подборът на идентификатори влияе директно върху целостта, мащабируемостта и производителността на системата.