Abylkas Saginov Karaganda Technical University

Information Technologies and Security Department

**Laboratory work №3**

**Discipline**: Database management systems

**Topic**: Creating Database Relationships

**Prepared by:**

Student of SIB-22-5 group

Batyrkhanov Arman

**Approved by:**

Senior Lecturer

Klyuyeva Ye.G.

**Karaganda 2024**

1. **Build a diagram of your database**

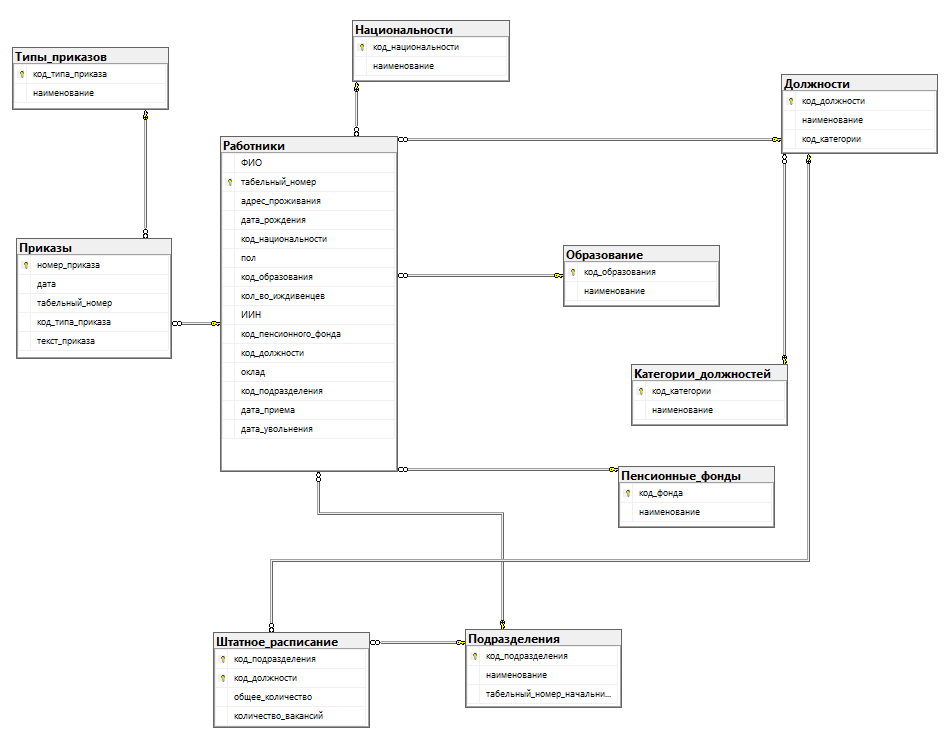


Fig. 1 – Database diagram

1. **Define foreign keys**

FOREIGN KEY (код\_категории) REFERENCES Категории\_должностей(код\_категории) ON UPDATE CASCADE

FOREIGN KEY (код\_национальности) REFERENCES Национальности(код\_национальности) ON UPDATE CASCADE,

FOREIGN KEY (код\_образования) REFERENCES Образование(код\_образования) ON UPDATE CASCADE,

FOREIGN KEY (код\_пенсионного\_фонда) REFERENCES Пенсионные\_фонды(код\_фонда) ON UPDATE CASCADE,

FOREIGN KEY (код\_должности) REFERENCES Должности(код\_должности) ON UPDATE CASCADE,

FOREIGN KEY (код\_подразделения) REFERENCES Подразделения(код\_подразделения) ON UPDATE CASCADE

FOREIGN KEY (табельный\_номер) REFERENCES Работники(табельный\_номер) ON UPDATE CASCADE,

FOREIGN KEY (код\_типа\_приказа) REFERENCES Типы\_приказов(код\_типа\_приказа) ON UPDATE CASCADE

FOREIGN KEY (код\_подразделения) REFERENCES Подразделения(код\_подразделения) ON UPDATE CASCADE,

FOREIGN KEY (код\_должности) REFERENCES Должности(код\_должности) ON UPDATE CASCADE

1. **Create non-unique indexes on database table attributes that are foreign keys**

CREATE INDEX idx\_работники\_подразделение ON Работники(код\_подразделения);

CREATE INDEX idx\_работники\_должность ON Работники(код\_должности);

CREATE INDEX idx\_работники\_образование ON Работники(код\_образования);

CREATE INDEX idx\_работники\_дата\_приема ON Работники(дата\_приема);

CREATE INDEX idx\_приказы\_табельный ON Приказы(табельный\_номер);

CREATE INDEX idx\_приказы\_тип\_приказа ON Приказы(код\_типа\_приказа);

CREATE INDEX idx\_приказы\_дата ON Приказы(дата);

CREATE INDEX idx\_штатное\_подразделение ON Штатное\_расписание(код\_подразделения);

CREATE INDEX idx\_штатное\_должность ON Штатное\_расписание(код\_должности);

CREATE INDEX idx\_пенсионный\_фонд\_наименование ON Пенсионные\_фонды(наименование);

CREATE INDEX idx\_должности\_категория ON Должности(код\_категории);

CREATE INDEX idx\_категория\_наименование ON Категории\_должностей(наименование);

1. **Build relationships (connections) between database tables and set integrity constraint rules**

CREATE TABLE Национальности (

код\_национальности INT PRIMARY KEY,

наименование VARCHAR(100) NOT NULL

);

CREATE TABLE Образование (

код\_образования INT PRIMARY KEY,

наименование VARCHAR(100) NOT NULL

);

CREATE TABLE Пенсионные\_фонды (

код\_фонда INT PRIMARY KEY,

наименование VARCHAR(100) NOT NULL

);

CREATE TABLE Категории\_должностей (

код\_категории INT PRIMARY KEY,

наименование VARCHAR(100) NOT NULL

);

CREATE TABLE Должности (

код\_должности INT PRIMARY KEY,

наименование VARCHAR(100) NOT NULL,

код\_категории INT,

FOREIGN KEY (код\_категории) REFERENCES Категории\_должностей(код\_категории) ON UPDATE CASCADE

);

CREATE TABLE Подразделения (

код\_подразделения INT PRIMARY KEY,

наименование VARCHAR(100) NOT NULL,

табельный\_номер\_начальника ТабельныйНомер

);

CREATE TABLE Работники (

ФИО VARCHAR(100),

табельный\_номер ТабельныйНомер PRIMARY KEY IDENTITY,

адрес\_проживания VARCHAR(255) NOT NULL,

дата\_рождения DATE NOT NULL,

код\_национальности INT,

пол CHAR(1) CHECK (пол IN ('М', 'Ж')),

код\_образования INT,

кол\_во\_иждивенцев INT CHECK (кол\_во\_иждивенцев >= 0),

ИИН VARCHAR(12) UNIQUE NOT NULL,

код\_пенсионного\_фонда INT,

код\_должности INT,

оклад DECIMAL(10, 2) NOT NULL,

код\_подразделения INT,

дата\_приема DATE NOT NULL,

дата\_увольнения DATE,

FOREIGN KEY (код\_национальности) REFERENCES Национальности(код\_национальности) ON UPDATE CASCADE,

FOREIGN KEY (код\_образования) REFERENCES Образование(код\_образования) ON UPDATE CASCADE,

FOREIGN KEY (код\_пенсионного\_фонда) REFERENCES Пенсионные\_фонды(код\_фонда) ON UPDATE CASCADE,

FOREIGN KEY (код\_должности) REFERENCES Должности(код\_должности) ON UPDATE CASCADE,

FOREIGN KEY (код\_подразделения) REFERENCES Подразделения(код\_подразделения) ON UPDATE CASCADE

);

CREATE TABLE Типы\_приказов (

код\_типа\_приказа INT PRIMARY KEY,

наименование NVARCHAR(100) NOT NULL

);

CREATE TABLE Приказы (

номер\_приказа INT PRIMARY KEY IDENTITY(1,1),

дата DATE NOT NULL,

табельный\_номер ТабельныйНомер,

код\_типа\_приказа INT,

текст\_приказа VARCHAR(MAX),

FOREIGN KEY (табельный\_номер) REFERENCES Работники(табельный\_номер) ON UPDATE CASCADE,

FOREIGN KEY (код\_типа\_приказа) REFERENCES Типы\_приказов(код\_типа\_приказа) ON UPDATE CASCADE

);

CREATE TABLE Штатное\_расписание (

код\_подразделения INT,

код\_должности INT,

общее\_количество INT CHECK(общее\_количество>=0),

количество\_вакансий INT CHECK(количество\_вакансий>=0),

PRIMARY KEY (код\_подразделения, код\_должности),

FOREIGN KEY (код\_подразделения) REFERENCES Подразделения(код\_подразделения) ON UPDATE CASCADE,

FOREIGN KEY (код\_должности) REFERENCES Должности(код\_должности) ON UPDATE CASCADE

);

**Control questions**

1. **Purpose of Relationship Mechanism and Types of Relationships**

The relationship mechanism in a database management system is crucial for establishing connections between different tables, which ensures data integrity and efficient data retrieval. There are three primary types of relationships. In a one-to-one relationship, each record in one table corresponds to exactly one record in another table. A one-to-many relationship allows a record in one table to be associated with multiple records in another table, while a many-to-many relationship involves records in one table being related to multiple records in another table, typically managed through a junction table.

1. **Ways to Create Relationships**

Relationships can be created using various methods in SQL Server Management Studio (SSMS). One common approach is to define a foreign key constraint on a column in one table that references the primary key of another table. Alternatively, SSMS provides diagramming tools that allow users to visually establish relationships between tables by dragging and dropping fields. Relationships can also be created programmatically using T-SQL commands, specifically the ALTER TABLE command combined with the ADD CONSTRAINT clause to define foreign keys.

1. **Referential Integrity of Data and Ways to Maintain It**

Referential integrity is a fundamental aspect of relational databases, ensuring that relationships between tables remain consistent. It can be maintained by enforcing foreign key constraints, which prevent orphaned records and ensure that any value in a foreign key column corresponds to a valid primary key in the related table. Additionally, using cascading actions such as cascading updates and deletes can help maintain referential integrity by automatically propagating changes through related records.

1. **Ways to Display Dependencies Between Tables**

Dependencies between tables in a database can be displayed through database diagrams in SSMS, which visually represent the relationships and constraints among tables. Furthermore, the "Object Explorer" feature in SSMS allows users to view dependencies by right-clicking on a table and selecting "View Dependencies," providing insight into how tables are related to one another.

1. **Methods for Building and Modifying Database Schemas**

Building and modifying a database schema can be accomplished in several ways. In SSMS, users can create new tables, modify existing ones, or delete them using the "Table Designer." Additionally, changes to the schema can be executed through T-SQL commands, allowing for precise control over table structures, data types, and constraints. Schema changes can also be managed using migrations, which are useful in development environments for tracking changes over time.

1. **Purpose and Classification of Indexes**

Indexes in a database serve the purpose of improving query performance by allowing faster data retrieval. They can be classified into several types, including clustered and non-clustered indexes. A clustered index determines the physical order of data in a table, while non-clustered indexes create a separate structure that references the data. Each type of index has its specific use cases and performance implications.

1. **Methods for Creating Indexes**

Indexes can be created in SSMS using the "Table Designer," where users can specify columns to index. Additionally, T-SQL commands such as CREATE INDEX can be used to define indexes programmatically. In some cases, SQL Server may also provide automatic indexing features that help optimize query performance based on workload patterns.