University of Kragujevac

Artificial Intelligence

Project for an Advanced databases (VI-19)

Student:

Babaev Bogdan 0866511

Kragujevac, December 2025

Contents

The task

As part of the "Advanced Databases" course, the project is intended to demonstrate skills in the design and management of a relational database. The primary objectives include:

• Developing a relational database model using Entity-Relationship (ER) modeling;

• Applying normalization (up to Third Normal Form, 3NF) to eliminate data redundancy;

• Implementing SQL queries for table creation, data insertion, and analytical operations;

• Integrating external data sources (API) and storing the data in the database;

• Ensuring data integrity through the use of primary and foreign keys;

• Demonstrating practical skills in designing both the logical and physical models of a database.

The goal of the project is to build a functional database that addresses a real-world problem related to data storage and analysis, while also applying theoretical knowledge in relational algebra, normalization, and SQL.

1. Purpose and Description of the Database

Purpose

The objective of the project was to develop a relational database to store and analyze historical cryptocurrency price data (Bitcoin and Ethereum) retrieved from the CryptoCompare API. The database is designed to:

- Efficiently store information about cryptocurrencies, dates, and market metrics (price, trading volume, market capitalization).
- Ensure data integrity through normalization and foreign key constraints.
- Support analytical queries, such as calculating monthly average prices.
- Be scalable for adding new cryptocurrencies or time periods.

Database Description

The database, named crypto.db, is implemented using SQLite and consists of three tables:

1. **Cryptocurrency**: Stores information about cryptocurrencies.

- Fields: crypto_id (primary key), name (e.g., Bitcoin), ticker (unique, e.g., BTC), description (optional).

Example: (1, 'Bitcoin', 'BTC', NULL).

2. **Date**: Stores unique dates for aggregated data.

- Fields: date_id (primary key), year, month, day.

Example: (1, 2023, 1, 1).

3. **PriceData**: Stores market data, linking cryptocurrencies and dates.

- Fields: price_id (primary key), crypto_id (foreign key to Cryptocurrency), date_id (foreign key to Date), avg_price (average price), volume (trading volume), market_cap (market capitalization).
- Example: (1, 1, 1, 30000.0, 1000.0, 0.0).

E-R Model:

- Entities: Cryptocurrency, Date, PriceData.
- Relationships:
    - PriceData is linked to Cryptocurrency via crypto_id (many-to-one).
    - PriceData is linked to Date via date_id (many-to-one).
- Normalization to 3NF eliminates redundancy by separating cryptocurrency and date information into distinct tables.

Purpose: The database stores historical cryptocurrency data from January 1, 2023, to January 1, 2025, and supports analytics, such as calculating monthly average prices or analyzing trading volumes.

2. Implementation Details

The project was implemented in several stages, each aligning with the course requirements.

2.1. Database Design

- **E-R Modeling**: An Entity-Relationship model was developed with three entities (Cryptocurrency, Date, PriceData) and their attributes and relationships.
- **Normalization**:
  - **1NF**: All attributes are atomic (e.g., dates are split into year, month, day).
  - **2NF**: Partial dependencies are eliminated. For example, in PriceData, attributes avg_price, volume, and market_cap depend on the combination of crypto_id and date_id.
  - **3NF**: Transitive dependencies are removed. Cryptocurrency and date information are stored in separate tables to avoid duplication.
- **Implementation**: The tables were created using SQL queries in SQLite (Block 2 of the code):

```
CREATE TABLE IF NOT EXISTS Cryptocurrency (
    crypto_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    ticker TEXT NOT NULL UNIQUE,
    description TEXT
);
CREATE TABLE IF NOT EXISTS Date (
    date_id INTEGER PRIMARY KEY,
    year INTEGER NOT NULL,
    month INTEGER NOT NULL,
    day INTEGER NOT NULL,
    UNIQUE(year, month, day)
);
CREATE TABLE IF NOT EXISTS PriceData (
    price_id INTEGER PRIMARY KEY,
    crypto_id INTEGER,
    date_id INTEGER,
    avg_price REAL,
    volume REAL,
    market_cap REAL,
    FOREIGN KEY (crypto_id) REFERENCES Cryptocurrency(crypto_id),
    FOREIGN KEY (date_id) REFERENCES Date(date_id)
);
```

Storage: The tables are stored in the crypto.db file located at /content/drive/MyDrive/уник/cripto on Google Drive.

2.2. Data Collection and Integration

- Historical price data for Bitcoin (BTC) and Ethereum (ETH) were collected via the CryptoCompare API for the period from January 1, 2023, to January 1, 2025 (Block 3).
- Data were processed using Pandas to calculate the average price (avg_price = (high + low) / 2) and trading volume, then saved to a CSV file for further loading into the database.
    - Example code for data collection:

```python
def get_crypto_data(symbol, start_date, end_date, api_key=None):
    base_url = "https://min-api.cryptocompare.com/data/v2/histoday"
    days = (end_date - start_date).days
    params = {
        'fsym': symbol,
        'tsym': 'USD',
        'limit': min(days, 2000),
        'toTs': int(end_date.timestamp())
    }
    if api_key:
        params['api_key'] = api_key

    response = requests.get(base_url, params=params)
    data = response.json()
    if data['Response'] != 'Success':
        raise ValueError(f"Ошибка API: {data.get('Message', 'Неизвестная ошибка')}")
```

Fig1: data collection

2.3. Data Aggregation

- Data were aggregated monthly using Apache Spark to calculate average prices, trading volumes, and market capitalization (Block 4).
- Example aggregation code:

```python
monthly_data = df_spark.groupBy(
    'ticker', year('date').alias('year'), month('date').alias('month')
).agg(
    avg('avg_price').alias('monthly_avg_price'),
    avg('volume').alias('monthly_avg_volume'),
    avg('market_cap').alias('monthly_avg_market_cap')
).filter(
    (col('year').between(2023, 2025)) & (col('month').between(1, 12))
).orderBy('ticker', 'year', 'month')
```

Fig2: data collection

2.4. SQL Operations

- **Data Insertion**: Data were bulk-inserted into the tables using executemany for efficiency (Block 5). Example:

```
cursor.executemany(
    "INSERT OR IGNORE INTO Cryptocurrency (ticker, name) VALUES (?, ?)",
    crypto_data
)
```

Fig3: SQL

2.5. Additional Features

- The database is integrated with Google Drive for data storage (/content/drive/MyDrive/уник/cripto).
- Data encryption using AES was implemented to secure the database (Block 6), demonstrating an understanding of data protection.

3. Technologies Used

- **SQLite**: For storing and managing the relational database.
- **Python**: For data processing, API interaction, and database management.
- **Pandas**: For data preparation and preprocessing.
- **Apache Spark**: For aggregating large datasets.
- **CryptoCompare API**: For retrieving historical cryptocurrency data.
- **Google Colab and Google Drive**: For code execution and data storage.
- **PyCryptoDome**: For database encryption.

4. Results

- A relational database (crypto.db) was created, containing the Cryptocurrency, Date, and PriceData tables, normalized to 3NF.
- SQL queries were implemented for table creation, data insertion, and analytics, including complex JOIN operations.
- Cryptocurrency price data (BTC, ETH) were collected, aggregated, and stored in the database.
- The project demonstrates proficiency in E-R modeling, normalization, SQL operations, and data integration, fully meeting the requirements of the "Advanced Databases" course.
- Data encryption was implemented as an additional feature, showcasing an understanding of data security.

Conclusion

The project successfully addresses the task of storing and analyzing cryptocurrency data, applying key relational database concepts. The proposed enhancements (indexing and views) provide a pathway for further optimization and scalability. I am prepared to provide the source code, the database file (crypto.db), and additional clarifications on any aspect of the project.