**TDS-I Project**

Database design for cryptocurrency portfolio and for

peer-to-peer transaction network

­­­­

Michal Ručka 29. 5. 2024

**DD S01 L02**

Explain the difference between the concept of data and information - an example in your project written in English

Data are raw facts and figures without context (for example 0.5, 30000, 2023-05-01) whereas information are processed and interpreted data that provides meaning. Example of information is this sentence: "0.5 units of Bitcoin traded for $30,000 on May 1, 2023".

**DD S02 L02**

Entities, instances, attributes and identifiers - describe in examples on your project

* Portfolio:
  + Portfolio\_ID (PK)
  + Title (VARCHAR)
  + Value (FLOAT)
  + Volume (FLOAT)
* Transactions
  + Transaction\_ID (PK)
  + Volume (FLOAT)
  + Value (FLOAT)
  + Date (DATE)
* Cryptocurrencies
  + Cryptocurrenncy\_ID (PK)
  + Name (VARCHAR)

This is example of 3 entities, that are part of mine project. Each entity can have it own‘s instance or instances. Exact instance of portfolio:

* Portfolio:
  + Portfolio\_ID: 6
  + Title: Mikyho shitcoin portf.
  + Value: 13,516.135Kc
  + Volume: 0.0016btc

**DD S03 L01**

Describe all relations in your database in English, including cardinality and membership obligation

**Users - Currencies:**

Each user is associated with exactly one currency (Users.Currency\_ID is mandatory).

One currency can be associated with multiple users (one-to-many relationship).

**Users - Documents:**

Each document is linked to exactly one user (Documents.User\_ID is mandatory).

One user can have multiple documents (one-to-many relationship).

**Admins - Users:**

Each admin is linked to exactly one user (Admins.User\_ID is mandatory).

One user can be associated with exactly one admin (one-to-one relationship).

**Users - Vexels:**

Each vexel is associated with exactly one user (Vexels.User\_ID is mandatory).

One user can be linked to one vexel (one-to-one relationship).

**Users - Portfolios:**

Each portfolio is associated with exactly one user (Portfolios.User\_ID is mandatory).

One user can have multiple portfolios (one-to-many relationship).

**Portfolios - Transactions:**

Each transaction is linked to exactly one portfolio (Transactions.Portfolio\_ID is mandatory).

One portfolio can have multiple transactions (one-to-many relationship).

**Cryptocurrencies - Transactions:**

Each transaction involves exactly one cryptocurrency (Transactions.Cryptocurrency\_ID is mandatory).

One cryptocurrency can be involved in multiple transactions (one-to-many relationship).

**Cryptocurrencies - P2P\_Offers:**

Each P2P offer involves one or more cryptocurrencies (P2P\_Offers.Cryptocurrency\_ID is mandatory).

One or more cryptocurrencies can be involved in multiple P2P offers (many-to-many relationship).

**P2P\_Offers - Vexels:**

Each P2P offer is linked to exactly one vexel (P2P\_Offers.Trader\_ID is mandatory).

One vexel can make multiple P2P offers (one-to-many relationship).

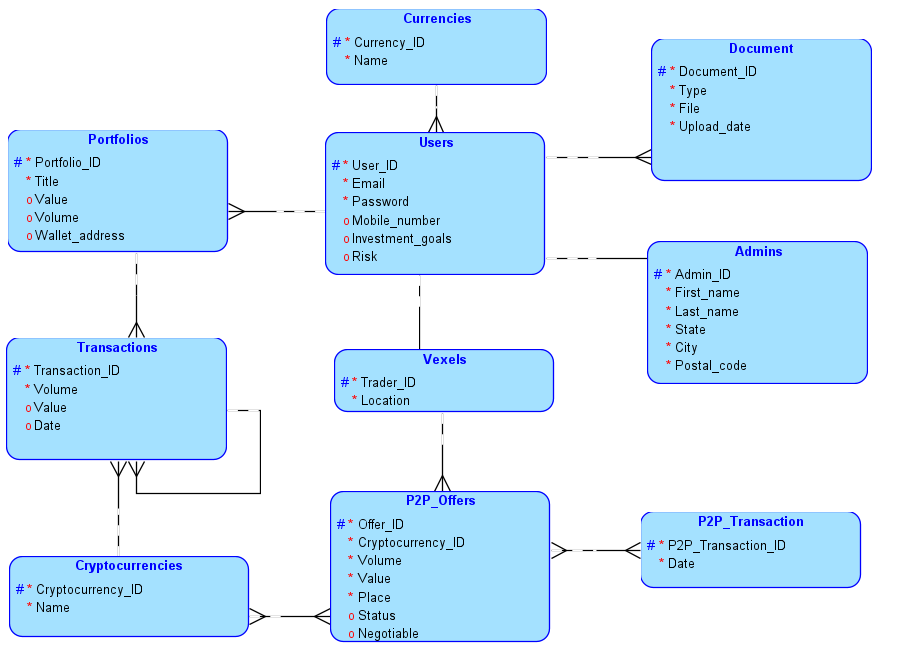
**P2P\_Offers - P2P\_Transactions:**

One or more P2P transactions are linked to one or more P2P offers (P2P\_Transactions.Offer\_ID is mandatory).

P2P offers can have multiple P2P transactions (many-to-many relationship).

**DD S03 L02**

Draw an ER diagram according to conventions



**DD S03 L04**

Matrix diagram with relationships, draw for your solution

1 – Users

2 – Currencies

3 - Documents

4 - Admins

5 - Vexels

6 – Portfolios

7 – Transactions

8 – Cryptocurrencies

9 – P2P\_Offers

10 – P2P\_Transactions

RT – Related to

CT – Connected to

BT – Belongs to

UB – Used by

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | X | RT | RT | CT | CT | RT |  |  |  |  |
| 2 | UB | X |  |  |  |  |  |  |  |  |
| 3 | BT |  | x |  |  |  |  |  |  |  |
| 4 | BT |  |  | x |  |  |  |  |  |  |
| 5 | BT |  |  |  | x |  |  |  | RT |  |
| 6 | BT |  |  |  |  | x | RT |  |  |  |
| 7 |  |  |  |  |  | RT | x | RT |  |  |
| 8 |  |  |  |  |  |  | UB | x | UB |  |
| 9 |  |  |  | BT |  |  |  | CT | x | RT |
| 10 |  |  |  |  |  |  |  |  | UB | x |

**DD S04 L01**

Supertypes and subtypes – define at least one instance of a supertype and a subtype in your project

Supertype: Users

The Users table can be considered a supertype.

Subtypes: Admins and Vexels

The Admins and Vexels tables can be considered subtypes of the Users supertype because they represent specific types of users with additional attributes unique to each type.

**DD S04 L02**

Description of business rules for your project

**User Authentication:**

Users must register and log in to access the system.

**Portfolio Management:**

Users can create multiple portfolios to manage their cryptocurrency holdings.

**Transaction Processing:**

Users can execute transactions to buy, sell, or transfer cryptocurrencies within their portfolios.

**Peer-to-Peer (P2P) Network:**

Users can create offers to buy or sell cryptocurrencies directly with other users.

**Document Management:**

Users may need to upload and store documents related to their transactions or account verification.

**Admin Privileges:**

Admins have special privileges to manage user accounts, resolve disputes, and oversee system operations.

**DD S05 L01**

Include at least one portable and one non-portable binding in your project

Portable: Cryptocurrencies – P2P\_Offers

Non-portable: User – Document

**DD S05 L03**

Have at least one M:N relationship without information and one M:N relationship with information in your project

With information: P2P\_Offers – P2P\_Transactions

Without information: Cryptocurrencies – P2P\_Offers

**DD S06 L01**

Incorporate at least one 1:N identifying relationship into your project, with the fact that the transferred foreign key will also be the key in the new table

User – Portfolio

**DD S06 L02-04**

Have your schema in:  
first normal form - no non-atomic attributes  
second normal form – no subkey bindings  
In third normal form - no links between secondary attributes

Done

**DD S07 L01**

Try to define ARC in your project (can be defined in ORACLE SQL Developer Data Modeler)

Attribute-Relation Chart

User can be basic user, vexel or admin. But cannot be admin and vexel at the same time due to security measures.

**DD S07 L02**

Try to define hierarchical and recursive relations in your project

Hierarchical relation:

Portfolios – Transactions

The portfolio consists of many transactions

Recursive relation:

Transactions – Transactions

Each transaction could be formed from many transactions

**DD S07 L03**

Describe how you record historical data in your system

I record historical data by including a timestamp attributes in relevant tables (Document, Transactions, P2P\_Transactions) to record the date and time when each record was created or last updated.

**DD S09 L01**

**DD S09 L02**

Try journaling in your project, i.e. saving past historical data (for example salary changes, workplace changes, etc.)

Change of admin’s name and last name:

CREATE TABLE Admins\_History (

History\_ID INTEGER PRIMARY KEY AUTOINCREMENT,

Admin\_ID SMALLINT,

First\_name VARCHAR2(30),

Last\_name VARCHAR2(30),

State VARCHAR2(20),

City VARCHAR2(30),

Postal\_code INTEGER,

User\_ID INTEGER,

Change\_Timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

Change\_Type VARCHAR2(20)

);

CREATE OR REPLACE TRIGGER Admins\_Update\_History

AFTER UPDATE ON Admins

FOR EACH ROW

BEGIN

INSERT INTO Admins\_History (

Admin\_ID, First\_name, Last\_name, State, City, Postal\_code, User\_ID, Change\_Type

) VALUES (

:OLD.Admin\_ID, :OLD.First\_name, :OLD.Last\_name, :OLD.State, :OLD.City, :OLD.Postal\_code, :OLD.User\_ID, 'UPDATE'

);

END;

UPDATE Admins

SET First\_name = 'NewFirstName', Last\_name = 'NewLastName'

WHERE Admin\_ID = 1;

**DD S10 L01**

Revise your design according to conventions for the readability of your schema

I've revised my database design to enhance readability by ensuring clear relationships and consistent naming conventions. I used straightforward table and column names, included primary and foreign key annotations, and organized the schema to visually represent the connections between entities.

**DD S10 L02**

Generic modeling – consider, possibly describe or use a generic model of data structures in your solution, how this approach is more advantageous compared to traditional data structure design methods

Using a generic model of data structures in my database design allows for greater flexibility and reusability, as it abstracts common features into a unified structure. This approach simplifies maintenance and scalability, reducing redundancy and making it easier to adapt to changing requirements compared to traditional, more rigid data structure designs.

**DD S11 L01**

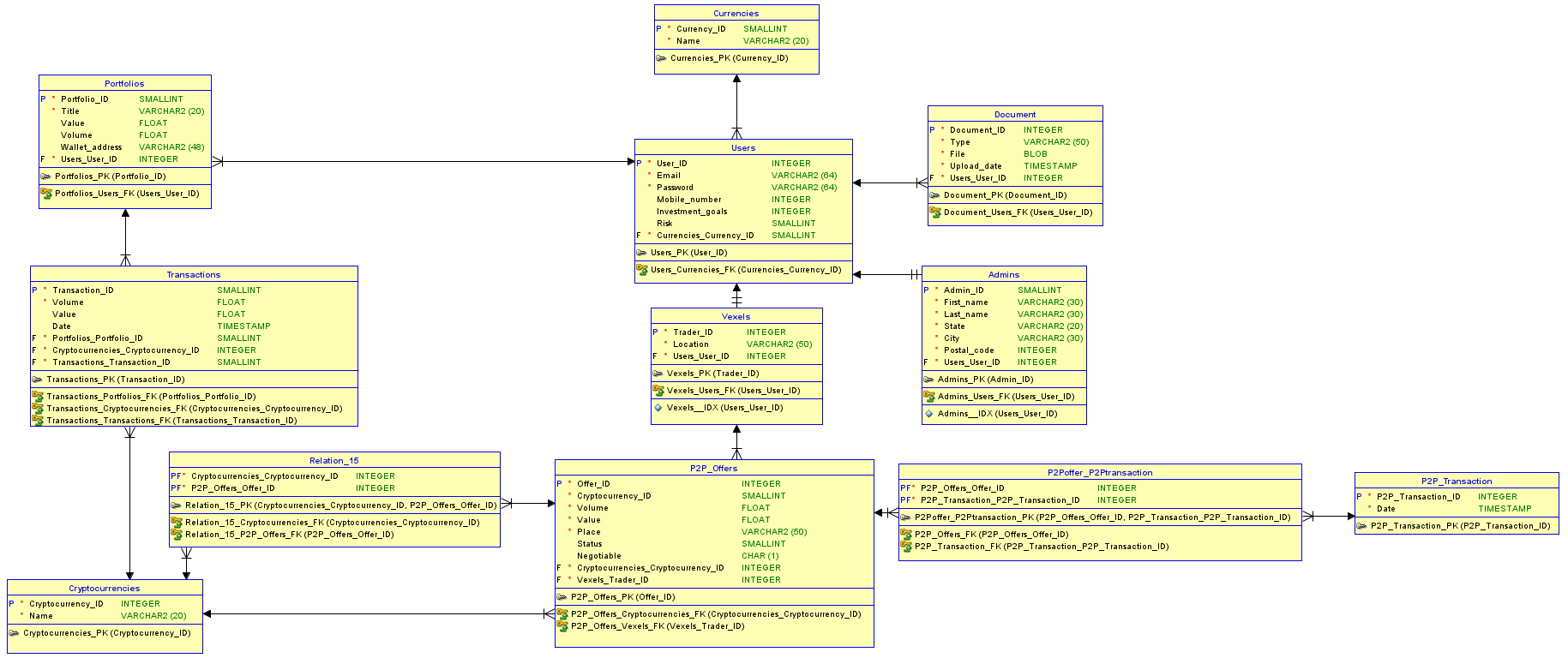
Describe examples of integrity constraints on your project for entities, bindings, attributes, and user-defined integrity

First integrity constraint is risk’s range. It has to be between numbers 0-100. The second one is P2P\_Offer’s status. It has 3 states:

* Available
* Reserved
* Completed

**DD S11 L02-04**

Generate a relational schema from your conceptual model and note the changes that have occurred in the schema and why



**DD S15 L01**

Write query for concatenate strings by pipes || , and CONCAT() o SELECT DISTINCT

SELECT 'FirstName' || 'LastName' AS FullName

FROM Users;

SELECT CONCAT('FirstName', ' ', 'LastName') AS FullName

FROM Users;

SELECT DISTINCT Currency\_ID

FROM Users;

**DD S16 L02**

WHERE condition for selecting rows

SELECT \*

FROM Users

WHERE Risk = 5;

Functions LOWER, UPPER, INITCAP

SELECT LOWER(Email) AS LowercaseEmail

FROM Users;

SELECT UPPER(Email) AS UppercaseEmail

FROM Users;

SELECT INITCAP(Email) AS CapitalizedEmail

FROM Users;

**DD S16 L03**

BETWEEN … AND

SELECT \*

FROM Transactions

WHERE Value BETWEEN 1000 AND 5000;

LIKE (%, \_)

SELECT \*

FROM Users

WHERE Email LIKE '%@gmail.com';

IN()

SELECT \*

FROM Users

WHERE Currency\_ID IN (1, 2, 3);

IS NULL, IS NOT NULL

SELECT \*

FROM Users

WHERE Mobile\_number IS NULL;

SELECT \*

FROM Users

WHERE Mobile\_number IS NOT NULL;

**DD S17 L01**

AND, OR, NOT

SELECT \*

FROM Users

WHERE Risk = 5 AND Currency\_ID = 1;

SELECT \*

FROM Users

WHERE Risk = 5 OR Currency\_ID = 1;

SELECT \*

FROM Users

WHERE NOT (Risk = 5);

Evaluation priority ()

SELECT \*

FROM Users

WHERE (Risk = 5 OR Currency\_ID = 1) AND Email LIKE '%@example.com';

**DD S17 L02**

ORDER BY atr [ASC/DESC]

SELECT \*

FROM Users

ORDER BY Email ASC;

SELECT \*

FROM Users

ORDER BY Email DESC;

Sorting by using one or more attributes

SELECT \*

FROM Users

ORDER BY Currency\_ID ASC, Risk DESC;

**DD S17 L03**

Single row functions

SELECT UPPER(Email) AS UppercaseEmail

FROM Users;

Column functions MIN, MAX, AVG, SUM, COUNT

SELECT MIN(Value) AS MinValue, MAX(Value) AS MaxValue, AVG(Value) AS AvgValue, SUM(Value) AS SumValue, COUNT(\*) AS TotalTransactions

FROM Transactions;