Czech University of Life Sciences

Faculty of Economics and Management

Database Systems project:

Database for Freelance Design Marketplace

Autor: Adina Sarsenbayeva

Introduction

This work designs a database for the Freelance design marketplace. Relationships between entities help to capture the basic information that needed to manage the marketplace. This database can manage information about projects, designers, clients, offers and payment. It's based on following assumptions:

- Each designer represents one designer who offers several of his services on the marketplace, each service represents design services.
- Each client represents one client that hires a designer for one or several projects.
- Each project represents design project posted by a client and a bid on by designer and the project can be a subject of payment (bought/sold).
- Each bid represents a bid made by one designer on the one project.
- Each design represents a design that is created as a part of one project.
- Each payment represents a payment made by a client to a designer for a one project

This project does not solve the problem of storing information about reviews, contracts, and portfolios. These are just simple examples of entities that could be included in a more complex freelance management system.

Outline

1. Possible use cases for the model

- Find the names of all designers who offered a service with a name Logo Design.
- Find name of all clients that posted a project with a budget greater than 2800.
- Find emails of all clients who have received a design for their projects.
- Find all projects from the bid where designer name is Mark.
- Find the names of all the clients, the name of the design and the paid price of the clients who paid for their projects.
- Find the titles and deadlines of all projects that have received a bid with an amount greater than 2600.

2. Entity relationship diagrams

Following section captures the proposed structure of database using entity relationship diagram. The diagrams were created in lucidchart.

2.1. Conceptual ERD

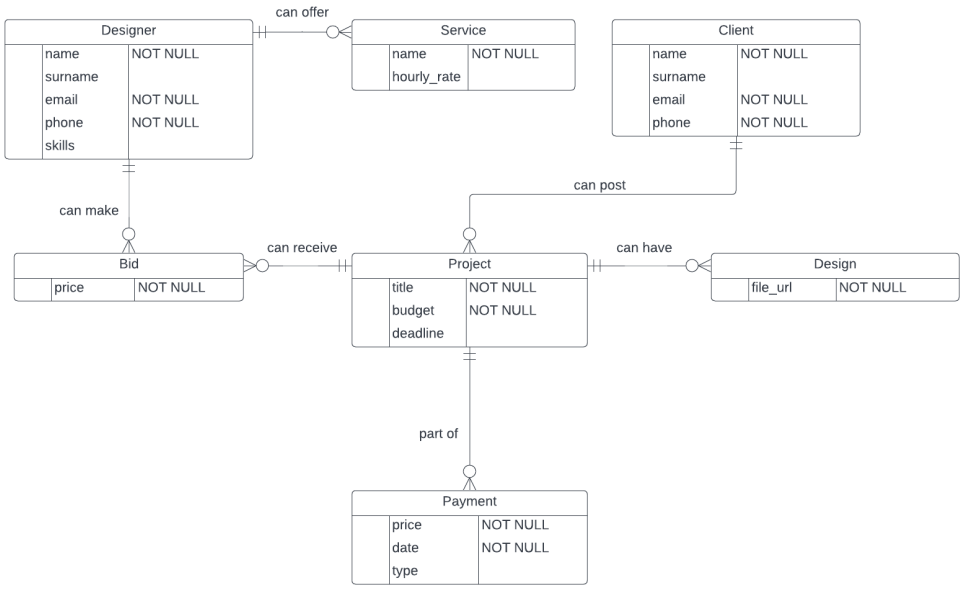

Figure 1: Conceptual ERD

2.2. Logical ERD

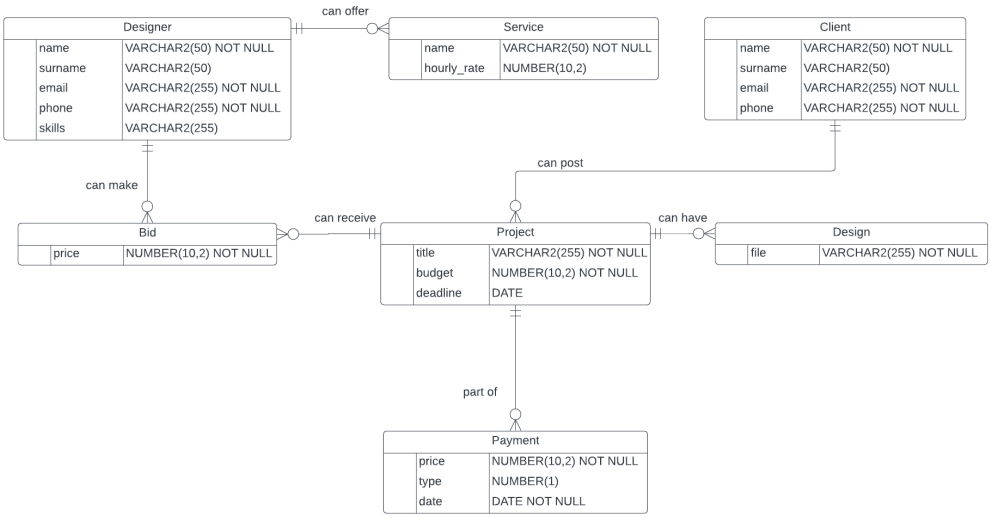

Figure 2: Logical ERD

2.3. Physical ERD

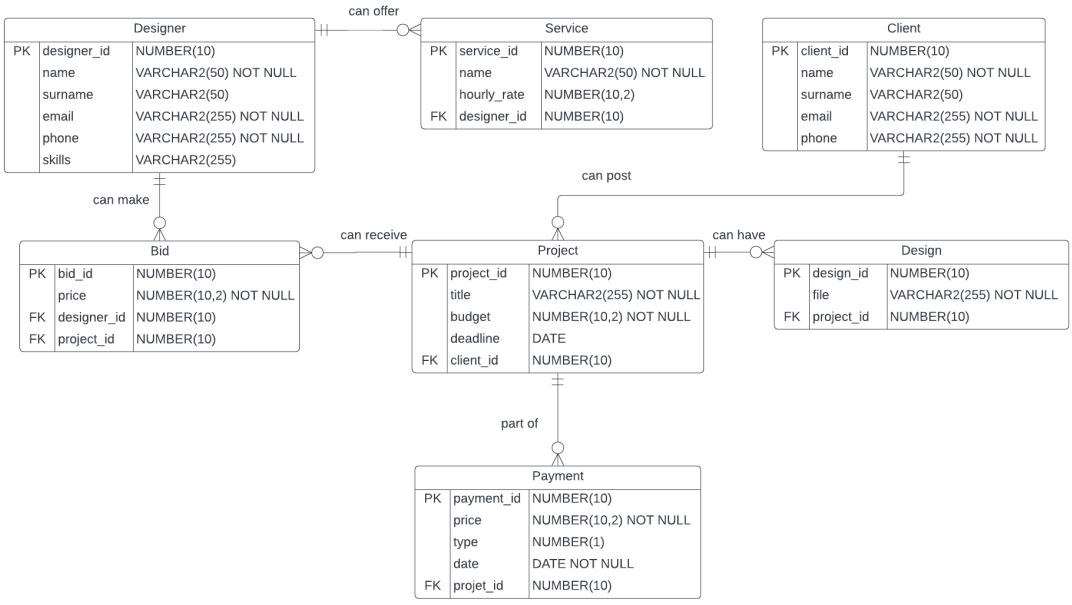

Figure 3: Physical ERD

3. SQL Implementation

The database was implemented in Oracle Application Express, which uses Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production as a DBMS.

3.1. DDL : Defining the database objects

CREATE TABLE Design (design_id NUMBER(10), file_url VARCHAR2(255) NOT NULL, project_id NUMBER(10), PRIMARY KEY (design_id));

CREATE TABLE Bid (bid_id NUMBER(10), price NUMBER(10,2) NOT NULL, designer_id NUMBER(10), project_id NUMBER(10), PRIMARY KEY (bid_id));

CREATE TABLE Payment (payment_id NUMBER(10), price NUMBER(10,2) NOT NULL, type NUMBER(1), paymentDate DATE NOT NULL, project_id NUMBER(10), PRIMARY KEY (payment_id));

CREATE TABLE Service (service_id NUMBER(10), name VARCHAR2(50) NOT NULL, hourly_rate NUMBER(10,2), designer_id NUMBER(10), PRIMARY KEY (service_id));

CREATE TABLE Client (client_id NUMBER(10), name VARCHAR2(50) NOT NULL, surname VARCHAR2(50), email VARCHAR2(255) NOT NULL, phone VARCHAR2(255) NOT NULL, PRIMARY KEY (client_id));

CREATE TABLE Designer (designer_id NUMBER(10), name VARCHAR2(50) NOT NULL, surname VARCHAR2(50), email VARCHAR2(255) NOT NULL, phone VARCHAR2(255) NOT NULL, skills VARCHAR2(255), PRIMARY KEY (designer_id));

CREATE TABLE Project (project_id NUMBER(10), title VARCHAR2(255) NOT NULL, budget NUMBER(10,2) NOT NULL, deadline DATE, client_id NUMBER(10), PRIMARY KEY (project_id));

Constraints:

ALTER TABLE Bid ADD CONSTRAINT "can make" FOREIGN KEY (designer_id) REFERENCES Designer (designer_id);

ALTER TABLE Bid ADD CONSTRAINT "can receive" FOREIGN KEY (project_id) REFERENCES Project (project_id);

ALTER TABLE Project ADD CONSTRAINT "can post" FOREIGN KEY (client_id) REFERENCES Client (client_id);

ALTER TABLE Service ADD CONSTRAINT "can offer" FOREIGN KEY (designer_id) REFERENCES Designer (designer_id);

ALTER TABLE Design ADD CONSTRAINT "can have" FOREIGN KEY (project_id) REFERENCES Project (project_id);

ALTER TABLE Payment ADD CONSTRAINT "part of" FOREIGN KEY (project_id) REFERENCES Project (project_id);

3.2. DML: Inserting the data (examples)

INSERT INTO Designer (designer_id, name, surname, email, phone, skills) VALUES(1, 'Alex', 'Burton', 'alexb@gmail.com', '+420774306244', 'Logo Types, video montage');

INSERT INTO Designer (designer_id, name, surname, email, phone, skills) VALUES(2, 'Timothy', 'Tim', 'timmy@gmail.com', '+420345657234','Illustrations');

INSERT INTO Designer (designer_id, name, surname, email, phone, skills) VALUES(3, 'Margaret', 'Flower', 'flowers@gmail.com', '+77073452312', 'App design, Web design');

INSERT INTO Designer (designer_id, name, surname, email, phone, skills) VALUES(4, 'Mark', 'Sloan', 'MarkS@gmail.com', '+420456123789', 'Various types of animation');

INSERT INTO Client (client_id, name, surname, email, phone) VALUES(1, 'Korra', 'Naga', 'korra@gmail.com', '+420234456123');

INSERT INTO Client (client_id, name, email, phone) VALUES(2, 'Aang', 'aang@gmail.com', '+77082342434');

INSERT INTO Client (client_id, name, email, phone) VALUES(3, 'Martina', 'Martina@gmail.com', '+42045667834');

INSERT INTO Client (client_id, name, email, phone) VALUES(4, 'Jaro', 'kim@gmail.com', '+420456345135');

INSERT INTO Project (project_id, title, budget, deadline, client_id) VALUES(1,'New Logo for company', 2500, '12/13/2022',2);

INSERT INTO Project (project_id, title, budget, deadline, client_id) VALUES(2, 'Flower Illustrations', 3000, '01/05/2023', 3)

INSERT INTO Project (project_id, title, budget, deadline, client_id) VALUES(3, 'Logo Animation', 3500, '01/10/2023', 1);

INSERT INTO Project (project_id, title, budget, deadline, client_id) VALUES(4, 'Design for an e-shop application', 20000, '03/10/2023',1)

INSERT INTO Bid (bid_id, price, designer_id, project_id) VALUES(1, 2600, 1, 1);

INSERT INTO Bid (bid_id, price, designer_id, project_id) VALUES(2, 3000, 4, 3);

INSERT INTO Bid (bid_id, price, designer_id, project_id) VALUES(3, 30000, 3, 4);

INSERT INTO Bid (bid_id, price, designer_id, project_id) VALUES(4, 2000,2, 2)

INSERT INTO Bid (bid_id, price, designer_id, project_id) VALUES(5, 2800, 2, 1);

INSERT INTO Design(design_id, file_url, project_id) VALUES(1,
'https://www.example.com/design/logo.png', 1);

INSERT INTO Design (design_id, file_url, project_id) VALUES(2,
'https://www.example.com/designs/flowers_illustration.png', 2);

INSERT INTO Design (design_id, file_url, project_id) VALUES(3,
'https://www.example.com/designs/logo_animation.png', 3);

INSERT INTO Design (design_id, file_url, project_id) VALUES(4,
'https://www.example.com/designs/design.png', 4);

INSERT INTO Design (design_id, file_url, project_id) VALUES(5,
'https://www.example.com/designs/logo2.png', 1);

INSERT INTO Payment(payment_id, price, type, paymentDate, project_id) VALUES(1, 2600, 1, '12/25/2022',1);

INSERT INTO Payment(payment_id, price, type, paymentDate, project_id) VALUES(2, 30000, 2, '12/27/2022', 2);

INSERT INTO Payment(payment_id, price, type, paymentDate, project_id) VALUES(3, 3000, 1, '12/25/2022',3);

INSERT INTO Payment(payment_id, price, type, paymentDate, project_id) VALUES(4, 25000, 1, '12/07/2022',4);

INSERT INTO Service(service_id, name, hourly_rate, designer_id) VALUES(1, 'Logo Design', 500, 1);

INSERT INTO Service(service_id, name, hourly_rate, designer_id) VALUES(2, 'Montage', 700, 1);

INSERT INTO Service(service_id, name, hourly_rate, designer_id) VALUES(3, 'Illustrations', 650, 2);

INSERT INTO Service(service_id, name, hourly_rate, designer_id) VALUES(4, 'Design', 400, 3);

INSERT INTO Service(service_id, name, hourly_rate, designer_id) VALUES(5, 'Animation', 700, 4);

INSERT INTO Service(service_id, name, hourly_rate, designer_id) VALUES(6, 'Animation', 700, 1);

3.3. SQL Queries

- Find the names of all designers who offered a service with a name Logo Design.

  SELECT d.name FROM Designer d INNER JOIN Service s ON s.designer_id = d.designer_id WHERE s.name = 'Animation'

- Find name of all clients that posted a project with a budget greater than 2800.

  SELECT c.name, p.budget FROM Client c INNER JOIN Project p ON p.client_id = c.client_id WHERE budget > 2800

- Find emails of all clients who have received a design for their projects.

  SELECT c.email FROM Client c INNER JOIN Project p ON p.client_id = c.client_id INNER JOIN Design d On d.project_id = p.project_id

- Find all projects from the bid where designer name is Mark.

  SELECT * FROM Project WHERE project_id IN (SELECT project_id FROM Bid INNER JOIN Designer D ON Bid.designer_ID = D.designer_id WHERE D.Name = 'Mark')

- Find the names of all the clients, the name of the design and the paid price of the clients who paid for their projects.

  SELECT c.name, p.title, py.price FROM Client c INNER JOIN Project p ON p.client_id = c.client_id INNER JOIN Payment py ON py.project_id = p.project_id

- Find the titles and deadlines of all projects that have received a bid with an amount greater than 2600.

  Select p.title, p.deadline FROM Project p INNER JOIN Bid b ON b.project_id = p.project_id WHERE b.price > 2600;

4. Conclusion

This database is well-suited for managing the basic information and relationships needed to support a freelance design marketplace. It contains definitions of essential database objects, and examples of possible use cases realized in the form of SQL queries. We may also consider whether other entities or relationships may be needed to support more complex or specialized scenarios. Therefore, this project is an essential part of the possible implementation of full-scale systems.