Laboratory Exercise №2

Study relationships between entities in database

Introduction to the problem

In database management systems (DBMS), understanding relationships between tables is essential for efficient data modeling, storage, and retrieval. Even without the use of Entity-Relationship (E-R) Diagrams, you can grasp how relationships function by exploring their conceptual foundations, common types, and their significance in a DBMS environment.

The concept of relationships in databases originated from the need to represent and manage associations between real-world entities in a structured, logical format. In a database, relationships link data points across different tables, ensuring that information can be connected, queried, and used together effectively. **Relational databases**, pioneered by Edgar F. Codd in the 1970s, **rely on tables (or "relations") to store data, and relationships between these tables reflect the logical connections between real-world entities**.

A **table (also called a relation)** in a database typically stores records about a specific type of entity, such as customers, products, or orders. Each row in the table represents a single record, and each column holds an attribute of the entity.

In a relational database, relationships are mainly categorized into three types based on how records in one table relate to records in another. These relationships are established through keys—fields that uniquely identify records. We can use the following three types of relationships:

- **One-to-One (1:1) Relationship** - each record in Table A corresponds to exactly one record in Table B, and vice versa. One-to-one relationships are less common and typically used when logically splitting data into two tables for security or performance reasons.
- **One-to-Many (1:M) Relationship** - one record in Table A can be related to multiple records in Table B, but each record in Table B relates to only one record in Table A. This is the most common type of relationship. One-to-many relationships are essential for situations like managing orders for a customer, tracking students in a class, or assigning books to authors.
- **Many-to-Many (M:M) Relationship** - multiple records in Table A can relate to multiple records in Table B, and vice versa. Many-to-many relationships are common in scenarios where multiple associations exist, such as students and courses, authors and books, or products and suppliers.

To implement and enforce relationships between tables, keys are used. The Primary Key (PK) is a unique identifier for each record in a table. For example, if we want to store information about customers of our company we will create a Customers table, in which CustomerID property could serve as the primary key. It will guarantee that no two records will have the same value in the primary key field. The Foreign Key (FK) is a field in one table that refers to the primary key in another table. It establishes the link between tables. For instance, in the Orders table using to store information about transactions between our company and our clients, CustomerID would be a foreign key referencing the CustomerID in the Customers table. Foreign keys ensure referential integrity, meaning that relationships between tables remain valid (e.g., an order can't reference a non-existent customer).

Understanding relationships between tables in databases is crucial for effective data management. These relationships allow for organizing complex data sets into logical structures that prevent

redundancy, maintain data integrity, and facilitate efficient querying. The common relationship types—one-to-one, one-to-many, and many-to-many—represent different ways data can be interconnected, often based on real-world associations.

In a DBMS environment, these relationships are crucial for ensuring consistent data storage, enabling powerful queries, and scaling database systems as they grow. Even without the use of E-R Diagrams, recognizing how relationships are implemented via primary and foreign keys is essential for mastering database management.

You may find helpful the following online sources of information about SQL syntax:

- SQL Tutorial, https://www.w3schools.com/sql/default.asp
- SQL Tutorial, https://www.sqltutorial.org/

Requirements for documentation of the work done by students

Each student must prepare and send a text document (in DOCX or PDF format) with designed SQL queries for each question from tasks of the current laboratory exercise. Please read carefully the text of the questions, because **each query should return only the data necessary to answer the question! Prepared document must be sent on the e-mail of your teaching assistant no later than a day before your next laboratory exercise.**

Case Study „A Shipping Company"

Introduction. You are an employee of a shipping company that operates in the vicinity of the city of Boston and, as such, you oversee the delivery of packages, documents and mail across the city. As a part of the IT support department, you have an access to corporate database. For the most part, all packages sent are eventually delivered 😉. Except, every once in a while, someone calls you with a claim of a missing package ☹! For each customer that contacts to you by phone with a report of a missing package, your job is to determine using the corporate database following:

- The current address (or location!) of their missing package
- The type of address or location (e.g. residential, business, etc.)
- The contents of the package

All you know is what the customers themselves will tell you. To solve each problem with delivery, you'll need to use the mail delivery service's database, `packages.db`, which contains data on the transit of packages around the city. Using just the information in the database, your task is to help each customer find their missing package. Within `packages.db`, you'll find several tables that implement the relationships described in the ER diagram at the end of this document.

The `addresses` table contains the following columns:

- `id`, which is the ID of the address
- `address`, which is the street address itself (i.e., 7660 Sharon Street)
- `type`, which is the type of address (i.e., residential, commercial, etc.)

The `drivers` table contains the following columns:

- `id`, which is the ID of the driver
- `name`, which is the first name of the driver

The `packages` table contains the following columns:

- `id`, which is the ID of the package
- `contents`, which contains the contents of the package
- `from_address_id`, which is the ID of the address from which the package was sent
- `to_address_id`, which is the ID of the address to which the package was sent. **It’s not necessarily where it ended up!**

The `scans` table contains the following columns:

- `id`, which is the ID of the scan
- `driver_id`, which is the ID of the driver who created the scan
- `package_id`, which is the ID of the package scanned
- `address_id`, which is the ID of the address where the package was scanned
- `action`, which indicates whether the package was picked up (“Pick”) or dropped off (“Drop”)
- `timestamp`, which is the day and time at which the package was scanned

Task 1. Your first call for a missing package comes from customer named Anneke. Anneke has told you the following:

“*Good afternoon! My name’s Anneke. I live over at 900 Somerville Avenue. Not long ago, I sent out a special letter. It’s meant for my friend Varsha. She’s starting a new chapter of her life at 2 Finnegan Street, uptown. (***Quick tip: It was a bit tricky to get that address right the first time.***) The letter is a congratulatory note—a cheery little paper hug from me to her, to celebrate this big move of hers. Can you check if it’s made its way to her yet*?”

For this problem, equally as important as finding the packages is the process that you use to do so. Please, keep a log of all SQL queries that you run on the database in a separate document. Above each query, label each with a comment: in SQL, comments are any lines that begin with two dashes “`--`”. Also each student must answer the next two questions:

```
At what type of address did the Lost Letter end up?
At what address did the Lost Letter end up?
```

Task 2. Your second call for a missing package comes from a mysterious fellow from out of town. He has told you the following:

“*Good day to you, Sir. Your colleagues might remember that not too long ago I made my way over from the town of Fiftyville. I gave a certain box into your reliable hands and asked you to keep things low. My associate has been expecting the package for a while now. But as it looks like, the package has been disappeared or it have grown wings and flown away. Ha! Any chance you could help clarify this mystery? Afraid there’s no “From” address. It’s the kind of parcel that would add a bit more… quack to someone’s bath times, if you catch my drift*.”

For this problem, equally as important as finding the packages is the process that you use to do so. Please, keep a log of all SQL queries that you run on the database in a separate document. Above each query, label each with a comment: in SQL, comments are any lines that begin with two dashes “`--`”. Also each student must answer the next two questions:

```
At what type of address did the Devious Delivery end up?
What were the contents of the Devious Delivery?
```

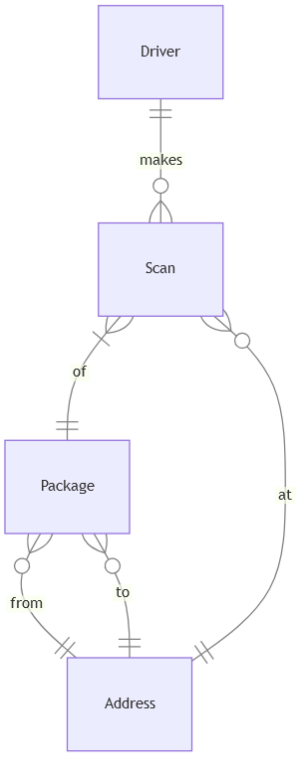
Task 3. Your third call for a missing package comes from a concerned grandparent. He has told you the following:

“*Oh, excuse me, lad. I had sent a mystery gift, to my wonderful granddaughter, off at 728 Maple Place. That was about two weeks ago. Now the delivery date has passed by seven whole days and I hear she still waits. I’m a bit worried wondering where my package has gone. Unfortunately, I don't remember what’s inside the package. Can we possibly track it down so it can fill her day with joy? I did send it from my home at 109 Tileston Street.*”

For this problem, equally as important as finding the packages is the process that you use to do so. Please, keep a log of all SQL queries that you run on the database in a separate document. Above each query, label each with a comment: in SQL, comments are any lines that begin with two dashes “`--`”. Also each student must answer the next two questions:

```
What are the contents of the Forgotten Gift?
Who has the Forgotten Gift?
```

The entity relationship (ER) diagram of `packages.db` database.



Case Study “Walking in Peter Brand’s Shoes”

Introduction. You will try to understand the specifics of the work involved in creating a successful selection of players for a baseball team. This was the task set for Peter Brand, an information technology specialist who took advantage of his database skills. The story was told in the 2011 film Moneyball but everything starts in year 2001. Peter Brand have been hired to help make the most of the Oakland Athletics baseball team’s dwindling player budget. Each year, teams hire new baseball players. Unfortunately, Oakland Athetics was low on star players - and on funds. Though, with a bit of SQL and some luck, who says you can’t still create a team that defies expectations and to repeat Peter Brand’s success?

For this case study you will use database called `moneyball.db` - one that contains information on players, their performances, and their salaries. The information stored in database represents all of Major League Baseball’s (MLB) players, teams, salaries, and performances up until 2001. In particular, moneyball.db represents the following entities:

- A player, which includes anyone who’s played Major League Baseball for any amount of time
- A team, which includes all teams, past and present, in Major League Baseball
- A performance, which describes the types of hits a player made for their team in a given year
- A salary, which is the amount of money a team paid one of their players in a given year

You can also look at the E-R diagram at the end of the case study, if you want to understand existing relationships between entities stored in database.

The `players` table contains the following columns:

- `id`, which is the ID of the player
- `first_name`, which is the first name of the player
- `last_name`, which is the last name of the player
- `bats`, which is the side (“R” for right or “L” for left) the player bats on
- `throws`, which is the hand (“R” for right or “L” for left) the player throws with
- `weight`, which is the player’s weight in pounds
- `height`, which is the player’s height in inches
- `debut`, which is the date (expressed as YYYY-MM-DD) the player began their career in the MLB
- `final_game`, which is the date (expressed as YYYY-MM-DD) the player played their last game in the MLB
- `birth_year`, which is the year the player was born
- `birth_month`, which is the month (expressed as an integer) the player was born
- `birth_day`, which is the day the player was born
- `birth_city`, which is the city in which the player was born
- `birth_state`, which is the state in which the player was born
- `birth_country`, which is the country in which the player was born

The `teams` table contains the following columns:

- `id`, which is the ID of each team
- `year`, which is the year the team was founded
- `name`, which is the name of the team

- `park`, which is name of the park at which the team plays (or played)

The `performances` table contains the following columns:

- `id`, which is the ID of the performance
- `player_id`, which is the ID of the player who generated the performance
- `team_id`, which is the ID of the team for which the player generated the performance
- `year`, which is the year in which the player generated the performance
- `G`, which is the number of games played by the player, for the given team, in the given year
- `AB`, which is the player’s number of “at bats” (i.e., times they went up to bat), for the given team, in the given year
- `H`, which is the player’s number of hits, for the given team, in the given year
- `2B`, which is the player’s number of doubles (two-base hits), for the given team, in the given year
- `3B`, which is the player’s number of triples (three-base hits), for the given team, in the given year
- `HR`, which is the player’s number of home runs, for the given team, in the given year
- `RBI`, which is the player’s number of “runs batted in” (i.e., runs scored), for the given team, in the given year
- `SB`, which is the player’s number of stolen bases, for the given team, in the given year

The `salaries` table contains the following columns:

- `id`, which is the ID of the salary
- `player_id`, which is the ID of the player earning the salary
- `team_id`, which is the ID of the team paying the salary
- `year`, which is the year during which the salary was paid
- `salary`, which is the salary itself in US dollars (not adjusted for inflation)

Task 1. Start your survey by getting a sense for how average player salaries have changed over time. Write a SQL query to find the average player salary by year.

- Sort by year in descending order.
- Round the salary to two decimal places and call the column “average salary”.
- Your query should return a table with two columns, one for year and one for average salary.

Task 2. The general manager (i.e., the person who makes decisions about player contracts) asks you whether the team should trade a current player for Cal Ripken Jr., a star player who’s likely nearing his retirement. Write a SQL query to find Cal Ripken Jr.’s salary history.

- Sort by year in descending order.
- Your query should return a table with two columns, one for year and one for salary.

Task 3. The team is going to need a great home run hitter. Ken Griffey Jr., a long-time Silver Slugger and Gold Glove award winner, might be a good prospect. Write a SQL query to find Ken Griffey Jr.’s home run history.

- Sort by year in descending order.
- Note that there may be two players with the name “Ken Griffey.” This Ken Griffey was born in 1969.

- Your query should return a table with two columns, one for year and one for home runs.

Task 4. Try to make a recommendation about which players the team should consider hiring. With the team’s dwindling budget, the general manager wants to know which players were paid the lowest salaries in 2001. Write a SQL query to find the 50 players paid the least in 2001.

- Sort players by salary, lowest to highest.
- If two players have the same salary, sort alphabetically by first name and then by last name.
- If two players have the same first and last name, sort by player ID.
- Your query should return three columns, one for players’ first names, one for their last names, and one for their salaries.

Task 5. You are doing very well with today’s exercise and feel empowered with database analysis tools and your growing skills 😊. Though Satchel Paige no longer plays, you are trying to write a SQL query to find all teams that Satchel Paige played for.

- Your query should return a table with a single column, one for the name of the teams.

Task 6. Which teams might be the biggest competition for the Oakland Athletics this year? Write a SQL query to return the top 5 teams, sorted by the total number of hits by players in 2001.

- Call the column representing total hits by players in 2001 “total hits”.
- Sort by total hits, highest to lowest.
- Your query should return two columns, one for the teams’ names and one for their total hits in 2001.

Task 7. The manager asks you to make a recommendation about which player (or players) to avoid recruiting. Write a SQL query to find the name of the player who’s been paid the highest salary, of all time, in Major League Baseball.

- Your query should return a table with two columns, one for the player’s first name and one for their last name.

Task 8. How much would the Oakland Athletics need to pay to get the best home run hitter this past season? Write a SQL query to find the 2001 salary of the player who hit the most home runs in 2001.

- Your query should return a table with one column, the salary of the player.

Task 9. What salaries are other teams paying? Write a SQL query to find the 5 lowest paying teams (by average salary) in 2001.

- Round the average salary column to two decimal places and call it “average salary”.
- Sort the teams by average salary, least to greatest.
- Your query should return a table with two columns, one for the teams’ names and one for their average salary.

Task 10. The general manager has asked you for a report which details each player’s name, their salary for each year they’ve been playing, and their number of home runs for each year they’ve been playing. To be precise, the table should include:

- All player's first names
- All player's last names
- All player's salaries
- All player's home runs
- The year in which the player was paid that salary and hit those home runs

To perform successfully this task, you should write a query to return just such a table.

- Your query should return a table with five columns, per the above.
- Order the results, first and foremost, by player's IDs (least to greatest).
- Order rows about the same player by year, in descending order.
- Consider a corner case: suppose a player has multiple salaries or performances for a given year. Order them first by number of home runs, in descending order, followed by salary, in descending order.
- Be careful to ensure that, for a single row, the salary's year and the performance's year match.

Quick Tip. To help you visualize what the general manager would like, they gave you an example table:

```
+------------+-----------+--------+------+----+
| first_name | last_name | salary | year | HR |
+------------+-----------+--------+------+----+
| Don        | Aase      | 400000 | 1989 | 0  |
| Don        | Aase      | 675000 | 1988 | 0  |
| Don        | Aase      | 625000 | 1987 | 0  |
| Don        | Aase      | 600000 | 1986 | 0  |
| Jeff       | Abbott    | 300000 | 2001 | 0  |
| Jeff       | Abbott    | 255000 | 2000 | 3  |
| Jeff       | Abbott    | 255000 | 1999 | 2  |
+------------+-----------+--------+------+----+
```

If all goes well, you might also see two rows like this in your final table:

```
+-------------+---------------+----------+----+------+
| first_name  | last_name     | salary   | HR | year |
+-------------+---------------+----------+----+------+
| Todd        | Zeile         | 3700000  | 9  | 1995 |
| Todd        | Zeile         | 3700000  | 5  | 1995 |
+-------------+---------------+----------+----+------+
```

As an aside, based on the schema of the database, why do you think Todd Zeile appears to have two different salaries (and two different HR counts) for the same year?

Task 11. You need a player that can get hits. Who might be the most underrated? Write a SQL query to find the 10 least expensive players per hit in 2001.

- Your query should return a table with three columns, one for the players' first names, one of their last names, and one called "dollars per hit".
- You can calculate the "dollars per hit" column by dividing a player's 2001 salary by the number of hits they made in 2001. Recall you can use AS to rename a column.
- Dividing a salary by 0 hits will result in a NULL value. Avoid the issue by filtering out players with 0 hits.
- Sort the table by the "dollars per hit" column, least to most expensive. If two players have the same "dollars per hit", order by first name, followed by last name, in alphabetical order.

- As in query from Task 10, ensure that the salary’s year and the performance’s year match.
- You may assume, for simplicity, that a player will only have one salary and one performance in 2001.

Task 12. Hits are great, but so are RBIs (“runs batted in” (i.e., runs scored))! Write a SQL query to find the players among the 10 least expensive players per hit and among the 10 least expensive players per RBI in 2001.

- Your query should return a table with two columns, one for the players’ first names and one of their last names.
- You can calculate a player’s salary per RBI by dividing their 2001 salary by their number of RBIs in 2001.
- You may assume, for simplicity, that a player will only have one salary and one performance in 2001.
- Order your results by player ID, least to greatest (or alphabetically by last name, as both are the same in this case!).
- Keep in mind the lessons you’ve learned in Task 10 and Task 11!

E-R Diagram related to `moneyball.db`