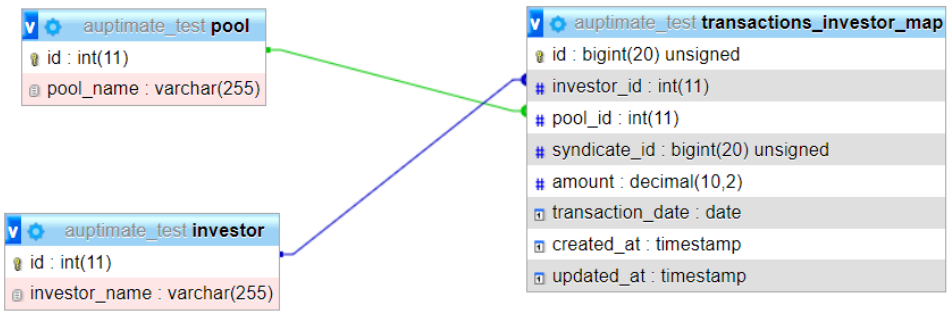
Problem 3:

A high-level architectural diagram illustrating the major components and their interactions:



A high-level architectural diagram

The main elements and how they work together to introduce a new feature that lets fund managers build and oversee investment pools are shown in the diagram above.

Components:

- **Web UI:** The system's user interface is called the web UI. Fund managers may examine real-time updates on investment amounts, payouts, and other associated activities, as well as build and manage investment pools and add and remove participants.
- **API:** The API serves as an interface between the web user interface and the system's other parts. It offers the features required for the web user interface to work.
- **Investment Pool Manager:** The system's fundamental component is the investment pool manager. It is in charge of establishing and overseeing investment pools, monitoring the contributions and withdrawals made by investors, and computing the most recent changes to investment amounts, payouts, and associated activities.
- **Database:** All of the system's data, including information on investors, investment pools, and transaction data, is kept in the database.

Relationships:

1. A fund manager opens a new investment pool by logging into the online user interface.
2. To create the new investment pool, the web user interface sends a request to the API.
3. To establish a new investment pool, the API makes a request to the investment pool management.
4. After establishing the new investment pool in the database, the investment pool manager sends a confirmation to the API.
5. The web UI receives the confirmation from the API.
6. The fund manager receives notification via the online user interface that the new investment pool has been established.

Similar to this, other activities like adding and deleting investors from pools and monitoring real-time updates on investment amounts, dividends, and other relevant transactions are carried out using the online user interface in conjunction with the API.

Using a combination of webhooks and streaming data, the system manages real-time changes of investment amounts, payouts, and other associated transactions. The system may be informed of occurrences in other systems, like a brokerage account, by using webhooks. The system receives real-time updates on market data, including stock prices, through streaming data.

An overview of the key elements and how they work together is given by this high-level architecture diagram, which is part of a new tool that lets fund managers construct and manage investment pools. The particulars of the organization's requirements will determine how the system is implemented.

A list of technologies and tools I would use, and a brief justification for each:

1. **Programming language:** Python is a popular, flexible language that is ideal for creating online applications. It has a sizable and vibrant community, and creating financial apps can be facilitated by the abundance of tools and frameworks available.
2. **Web framework:** Renowned for its security and scalability, Django is a well-liked Python web framework. It has several characteristics, such an integrated authentication system and an MVC design, that might be helpful in creating an investment pool administration system.

3. **Database:** PostgreSQL is an open-source, robust relational database that works well for holding financial information. It provides several features including views, foreign keys, and transactions that might be helpful for an investment pool administration system.

4. **Framework for APIs:** The widely used Django REST framework offers a RESTful API for Django applications. The investment pool management system would utilize it to give an API.
5. **Platform for streaming data:** Real-time data may be gathered and processed using Kafka, a distributed streaming platform. The purpose of it would be to gather and handle current market data.
6. **Webhooks:** A way for applications to communicate with each other in real time is through webhooks. They would be employed to alert the investment pool management system of happenings in other systems.

The selection of these technologies and tools was based on their suitability for creating an investment pool management system that is safe, scalable, and effective.

A discussion of any potential bottlenecks and my strategies for overcoming them:

Bottleneck: A large number of transactions.

- Asynchronous transaction processing can be achieved by utilizing a queuing mechanism. As a result, during times of heavy activity, the system won't get overwhelmed.
- **Approach:** Store data on several servers by utilizing a distributed database. This is going to enhance the system's scalability.

Cautionary calculations are the bottleneck.

- **Plan:** To save the output of computations that are done often, implement a caching mechanism. Because of this, fewer requests will need to be processed in total.
- **Technique:** To split up requests among several servers, use a load balancer. The system's operation under load will be enhanced as a result.

Bottleneck: **Updates to data in real time**

- **Strategy:** Gather and handle real-time data using a streaming data platform. This will guarantee that the data in the system is as current as possible.
- **Approach:** Use webhooks to tell the system when anything happens in other systems. By doing this, you can make sure the system is constantly up to date on any modifications.

Congested area: Security

- **Approach**: Make use of a safe web platform with integrated security measures.
- **Plan:** Put in place a thorough security policy that addresses every facet of the system.
- **Plan:** Perform security audits on a regular basis to find and fix any possible vulnerabilities.

You can make sure that your investment pool management system can manage complicated computations, large transaction volumes, real-time data changes, and security issues by putting these techniques into practice.

An outline of the steps and considerations for implementing and deploying this feature in a remote-first environment:

Procedures for putting a new feature into practice and rolling it out in a remote-first setting:

1. Planning and design

    - Clearly state the needs and extent of the feature.
    - Determine whether there are any possible reliances on other services or systems.
    - Create a strategy for quality control and testing.
    - Develop a communication strategy to update stakeholders on developments.

2. Development

    - To keep track of code modifications, use a version control system.
    - Review code frequently to make sure it's of high quality.
    - Create unit tests that address every facet of the code.
    - Connect the newly added functionality to the current codebase.
    - Use a version control system to manage code changes.

3. Testing

    - Run unit tests to make sure the new functionality functions as planned.
    - To make sure the new feature functions well with other systems, do integration tests.
    - Carry out user acceptability testing to make sure the new feature satisfies user demands.

4. Deployment

    - Develop a deployment plan that outlines the steps involved in deploying the new feature to production.
    - Create a rollback plan in case of any problems with the deployment.
    - Deploy the new feature to a staging environment for testing.
    - Deploy the new feature to production.

5. Monitoring and support

    - Monitor the new feature for any problems.
    - Provide support to users who have questions or problems with the new feature.

When introducing and rolling out a new feature in a remote-first setting, keep the following in mind:

- **Communication and collaboration**
    1. Make use of resources and procedures that help team members collaborate and communicate effectively.
    2. Keep in mind that different time zones affect how meetings and deadlines are scheduled.
    3. Ensure that the information required by each team member is available to them.
- **Documentation**
    1. Write and keep accurate, comprehensive documentation for the new functionality.
    2. Ensure that every team member has access to the documentation.
    3. Track modifications to documentation using version control.
- **Testing**
    1. Use a test automation framework to reduce the time and effort required to test the new feature.
    2. Use a continuous integration/continuous delivery (CI/CD) pipeline to automate the testing and deployment process.
- **Training**
    1. Provide training to users on how to use the new feature.
    2. Make sure that training materials are accessible to all users.
    3. Provide support to users who have questions or problems with the new feature.

You may improve your chances of success while implementing and launching a new feature in a remote-first environment by paying attention to these points and taking these actions into account.