Name: Jinze Huang
Date: May 16, 2025
Section: CSCI-GA.3033-026

Course Project

Weekly Update Date: 03/03/2025
Team Members: Jinze Huang

Answers to Individual Questions: points / **100 points**

Professor's Comments:

1 Team Members and Links

- Jinze Huang @jh9108
- Github: https://github.com/RogerHuangPKX/cloudcomputing25

2 Introduction

The Archemy system, a legacy application, comprised several key components: an Oracle ADF application (referred to as 'App'), a Python/Django web application ('archemy-webapp'), a Java-based security component ('FortressSecurity'), and a MySQL database. The primary goal of this project was to modernize its deployment by migrating the database to Alibaba Cloud's RDS for MySQL (Serverless) and the 'archemy-webapp' to an Azure Virtual Machine (VM). A critical aspect of this migration was ensuring that all dependencies, including the 'FortressSecurity' layer reliant on an LDAP directory, were correctly configured and functional in the new environment. The whole project structure is shown in Appendix A.

The Structure can be divided into 3 parts: Frontend, Database, and Backend.

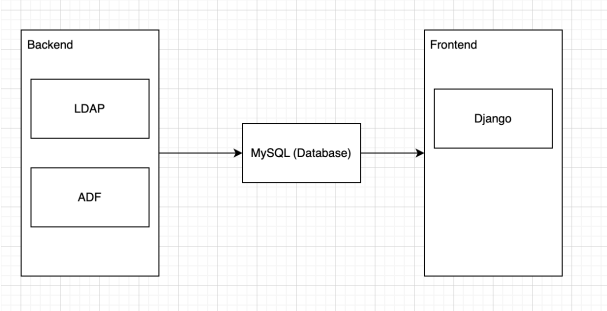

Figure 1: Structure of the Archemy system

3 Cloud Service Provider

The MySQL database is migrated to Alibaba Cloud RDS, and the Django application is deployed to Azure VM.

3.1 Why Alibaba Cloud RDS?

- Alibaba Cloud Serverless RDS is a cost-effective solution for small-scale applications. For example, we use MySQL 8.0, with high-available edition and multiple availability zones. The cost is **0.24-1.86 CNY per hour** with auto-scaling and pay-as-you-go. The price is shown in Figure 2.
- The RDS instance is deployed in the same region as the Azure VM, which is **US East**.
- Free Internet Access between Alibaba Cloud or the whole Internet.

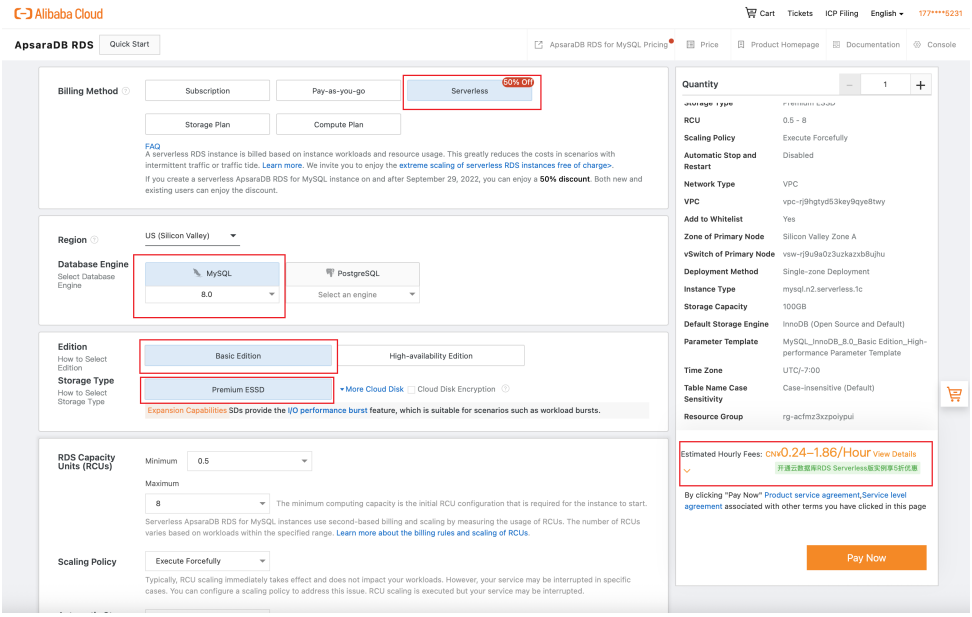

Figure 2: Price of Alibaba Cloud RDS

The public endpoint of the RDS instance is `rm-rj90131uzg61tgoqquo.rwlb.rds.aliyuncs.com`, and the port is `3306`. Figure 3 shows the endpoint of the RDS instance.

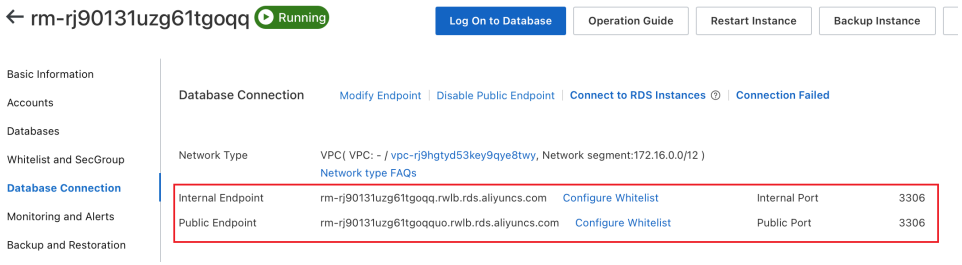

Figure 3: Endpoint of the RDS instance

3.2 Why Azure VM?

- Azure VM is a cost-effective solution for small-scale applications. For example, we use Ubuntu 20.04, with high-available edition and multiple availability zones. The cost is **134 USD per month**. The VM configuration is 4 cores, 8GB memory, 30GB SSD disk.
- The VM is deployed in the same region as the RDS instance, which is **US East**.
- Free Internet Access between Azure VM and the whole Internet.

4 Phase 1: Database Migration to Alibaba Cloud RDS

The initial phase focused on migrating the existing MySQL database to a managed MySQL 8.0 instance on Alibaba Cloud RDS.

4.1 Instance Setup and Connectivity

The target database was provisioned with the necessary credentials. We can set up the users and passwords and permissions via web console. I created a superuser called **jh9108** with all permissions.

4.2 Schema and Data Migration

The database schema was originally defined in a MySQL Workbench file (archemy_schema.mwb). This was exported to an SQL script (Untitled.sql).

1. **Schema Creation**: The exported SQL script was executed against the new RDS instance to create the table structures. Figure 4 shows the schema of the RDS instance. And the exported SQL script is shown in Appendix A.

2. **Initial Data Import**: Data was imported using the provided ‘DatabaseImport.sql‘ script. This process encountered several issues requiring modifications to the script:

   - An error related to sql_mode specifically, `NO_AUTO_CREATE_USER`, was resolved by removing this option from the ‘sql_mode‘ settings within the script. In MySQL 8.0, this option is not supported, we just simply remove it.
   - Access denied errors due to ‘DEFINER‘ clauses in ‘CREATE PROCEDURE‘ and ‘CREATE VIEW‘ statements. These were resolved by removing the ‘DEFINER‘ clauses, allowing the procedures and views to be created by the current migration user.

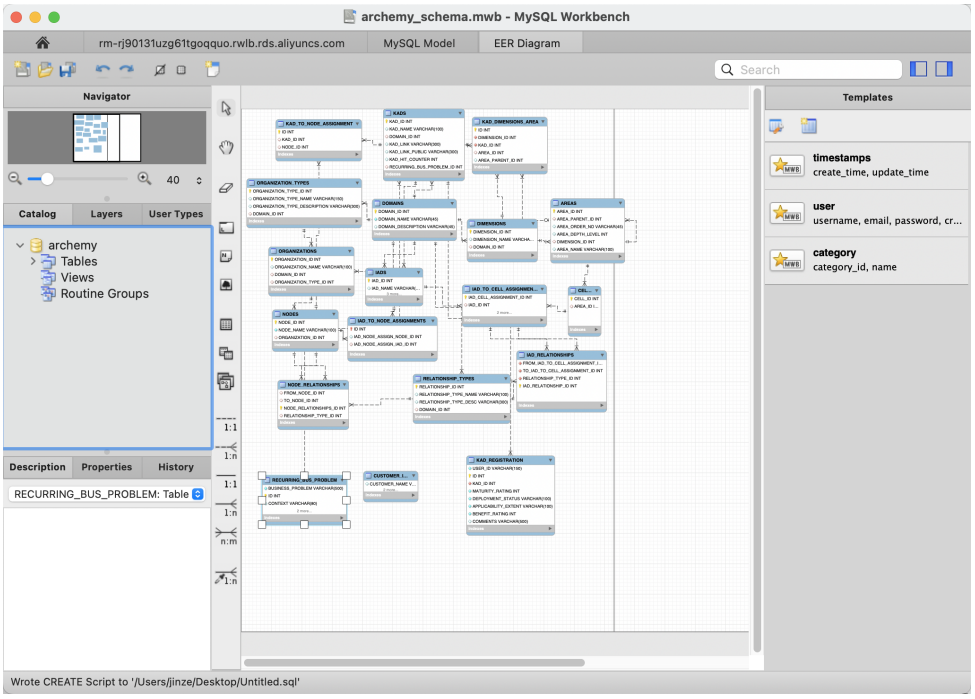

Figure 4: Schema of the RDS instance

Upon addressing these issues, the data import completed successfully.

5 Phase 2: Azure VM Setup and Django Application Deployment

This phase involved preparing the Azure VM and deploying the ‘archemy-webapp‘.

5.1 VM Preparation and Core Dependencies

The Azure VM, running an Ubuntu Server LTS, was prepared with essential software:

- Git (version 2.34.1) for version control.
- OpenJDK 8, chosen for compatibility with older Java components (specifically ‘FortressSecurity‘).
- Python 3, pip, and venv for the Django application. Initially, the system Python (3.10.12) was targeted.

5.2 Django Web Application (‘archemy-webapp‘) Deployment

Deploying the Django application presented significant challenges, primarily related to Python version compatibility and dependency resolution.

5.2.1 Python Environment Iterations

The application, built with Django 1.10.4, exhibited incompatibilities with newer Python versions:

- **Python 3.10**: Led to an ‘ImportError: cannot import name ’Iterator’ from ’collections’‘.
- **Python 3.9 and 3.8**: Both versions resulted in a ‘RuntimeError: __class__ not set defining ’AbstractBaseUser’‘ when attempting to run the Django server.
- **Python 3.7**: This version was ultimately chosen. System packages ‘python3.7‘, ‘python3.7-dev‘, and ‘python3.7-venv‘ were installed, and a dedicated virtual environment was created.

```
    note: This error originates from a subprocess, and is likely not a problem with pip.
error: metadata-generation-failed

 Encountered error while generating package metadata.
See above for output.
```

Listing 1: Python Environment Iterations

5.2.2 Dependency Installation (‘requirements.txt‘) on Python 3.7

The ‘requirements.txt‘ file required numerous adjustments:

- **MySQL Connector**: The original ‘MySQL-python==1.2.5‘ is not compatible with Python 3. After several iterations (including ‘mysqlclient==2.2.7‘ for Python 3.10 and ‘mysqlclient==2.1.1‘ for Python 3.7), ‘mysqlclient==1.4.6‘ was found to resolve a ‘KeyError: ¡class ’bytes’¿‘ issue encountered at runtime. This also necessitated installing system packages ‘libmysqlclient-dev‘ and ‘pkg-config‘.
- `unroll==0.1.0`: Installation failed due to ‘ez_setup‘ issues (‘ModuleNotFoundError‘ and subsequently an ‘HTTP Error 403: SSL is required‘ when ‘ez_setup‘ attempted to download ‘distribute‘ over HTTP). This was resolved by manually downloading the ‘unroll‘ source, commenting out ‘ez_setup‘ calls in its ‘setup.py‘, and installing from the modified local source.
- `Pillow==3.4.2`: Installation failed with ‘ValueError: jpeg is required‘. This was fixed by installing system dependencies ‘libjpeg-dev‘ and ‘zlib1g-dev‘.

- `numpy==1.11.2`: Failed to build due to a missing ‘xlocale.h‘ header. Removing the version constraint allowed a newer version to install successfully.
- `six==1.10.0`: Updated to ‘1.16.0‘ during troubleshooting, though the primary compatibility issues lay with the Python version itself.

5.2.3 Database Configuration and Migrations

The ‘archemy/settings.py‘ file was updated to connect to the new Alibaba Cloud RDS instance, using an environment variable (‘DB_PASSWORD‘) for the database password. After setting up dependencies, ‘python manage.py migrate‘ was run to apply Django’s database migrations, resolving an initial ‘Table ’adf.auth_user’ doesn’t exist‘ error.

The django database migration command is `python manage.py migrate`.

```
< && export DB_PASSWORD='Panyikai0305' && python manage.py migrate
Operations to perform:
  Apply all migrations: admin, archemywebapp, auth, contenttypes, sessions, sites
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying archemywebapp.0001_initial... OK
  Applying archemywebapp.0002_auto_20161217_0559... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
```

Figure 5: Django Database Migration

We can use following command to check the deployment of Django application.

```
source venv/bin/activate && export DB_PASSWORD='Panyikai0305' && python manage.py
    runserver 0.0.0.0:8000
```

Listing 2: Check Django Deployment

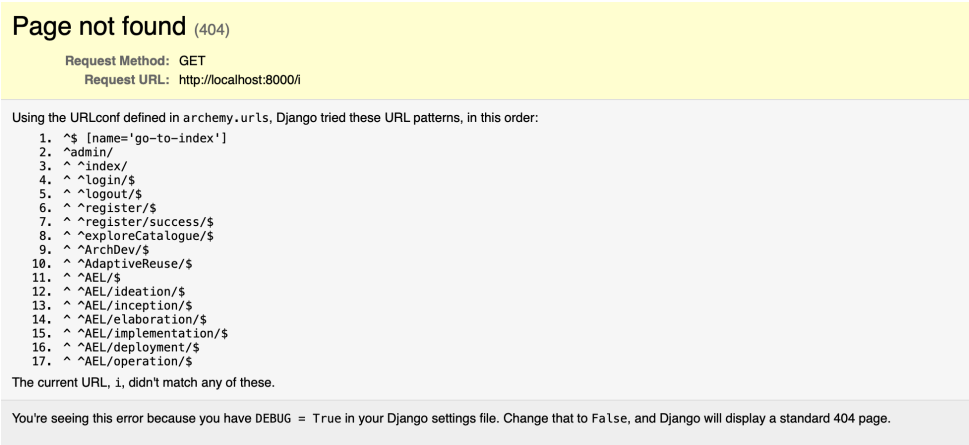

Figure 6: Django Deployment

The Django application is deployed successfully but we need the LDAP server to authenticate the users. When I try to login, I got the error

```
Internal Server Error: /login/
Traceback (most recent call last):
...
File "/home/jh9108/ADFEssentialsApp-SP25/archemy-webapp/archemywebapp/views.py", line
    32, in login
  return HttpResponseRedirect(settings.LOGIN_URL)
NameError: name 'settings' is not defined}.
```

Listing 3: LDAP Error

6 Phase 3: Identity Management: Fortress Security and OpenLDAP

The ‘archemy-webapp‘ relies on ‘FortressSecurity‘ for user authentication, which in turn uses an LDAP directory. This necessitated setting up an OpenLDAP server on the Azure VM. LDAP is a protocol for accessing and maintaining distributed directory information services. It is a standard way to store and retrieve information about users, groups, and other objects.

6.1 OpenLDAP Server Setup

1. ‘slapd‘ and ‘ldap-utils‘ were installed, using the command `sudo apt-get install slapd ldap-utils`.
2. The LDAP server was configured via ‘sudo dpkg-reconfigure slapd‘. Key settings included DNS domain ‘archemy.com‘, organization name ‘Archemy Org‘, and a new administrator password. The backend database chosen was MDB. This established the base naming context as ‘dc=archemy,dc=com‘. see Figure 7.
3. In the fortress.properties file, we can know the credentials of the admin user is ‘cn=Manager,ou=Users,dc=archemy,dc port is ‘389‘.
4. Initial attempts to add LDAP entries using ‘ldapadd‘ failed with ‘ldap_bind: Invalid credentials (49)‘. Troubleshooting revealed that the correct administrator DN for ‘dpkg-reconfigure slapd‘ was ‘cn=admin,dc=archemy,dc=com‘, not the ‘cn=Manager,...‘ often used as a default.
5. Using the correct admin DN, the necessary organizational units (OUs) and the Fortress-specific admin user were created using LDIF files:
   - ‘ou=Config,dc=archemy,dc=com‘
   - OUs: ‘People‘, ‘Groups‘, ‘Roles‘, ‘Permissions‘, ‘Users‘ under ‘dc=archemy,dc=com‘.
   - Fortress admin user: ‘cn=Manager,ou=Users,dc=archemy,dc=com‘ with the password ‘weartear‘ (as specified in the original ‘fortress.properties‘).

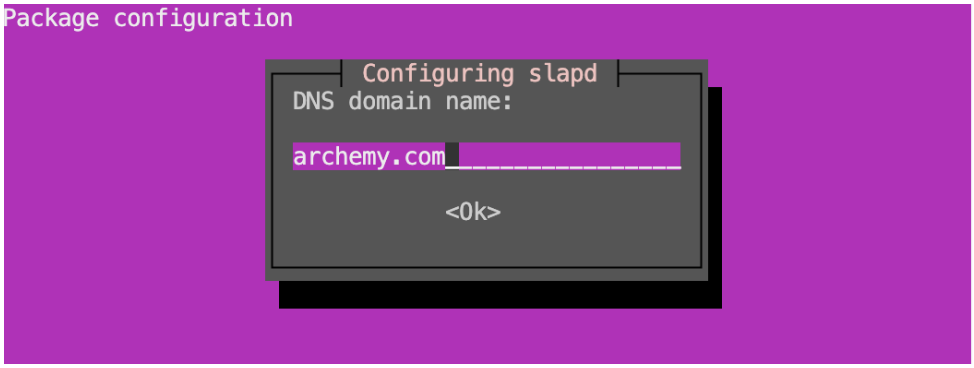

Figure 7: LDAP Configure

Verify the LDAP server is working by using the command `ldapsearch -x -b "" -s base "(objectclass=*)" namingContexts`.The output is shown in Figure 8.

```
jh9108@proj-1-linux:~$ ldapsearch -x -b "" -s base "(objectclass=*)" namingContexts
# extended LDIF
#
# LDAPv3
# base <> with scope baseObject
# filter: (objectclass=*)
# requesting: namingContexts
#

#
dn:
namingContexts: dc=archemy,dc=com

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Figure 8: LDAP Verify

6.2 Modification of ‘archemy-security‘ JAR

The ‘archemy-security-1.0-SNAPSHOT-jar-with-dependencies.jar‘ file contained a ‘fortress.properties‘ file that was hardcoded to connect to an external LDAP server (‘orastack.com:389‘) and used an incorrect DN structure for its admin user.

1. The JAR file was extracted into a temporary directory.
2. The ‘fortress.properties‘ file was modified:
   - ‘host‘: Changed from ‘orastack.com‘ to ‘localhost‘.
   - ‘admin.user‘: Changed from ‘cn=Manager,dc=archemy,dc=com‘ to the newly created ‘cn=Manager,ou=Users,d
3. The contents of the temporary directory were re-packaged into a new JAR file (e.g., ‘archemy-security-1.0-SNAPSHOT-jar-with-dependencies-MODIFIED.jar‘).
4. The original JAR in the Django application’s ‘lib‘ directory was backed up, and the modified JAR was put in its place.

7 Phase 4: Django Application Runtime Issues and Refinements

With the backend dependencies and security components configured, the focus shifted to runtime issues within the Django application.

7.1 CSRF (Cross-Site Request Forgery) Handling

Initial attempts to register or log in resulted in warnings related to CSRF context processors and, after removing ‘@csrf_exempt‘ decorators, ‘403 Forbidden (CSRF cookie not set.)‘ errors.

- The ‘archemy/settings.py‘ file was modified to set ‘CSRF_COOKIE_SECURE = False‘, as development was occurring over HTTP.
- In ‘archemywebapp/views.py‘, ‘@csrf_exempt‘ decorators were removed from the ‘login‘ and ‘register_page‘ views to enable CSRF protection. The ‘register_page‘ view was also updated to use the modern ‘render()‘ shortcut instead of ‘render_to_response‘.
- Template files (‘register_view.html‘, ‘login_view.html‘) were confirmed to include the ‘% csrf_token %‘ tag within their forms.

After these changes, and likely aided by clearing browser cache and cookies, CSRF-related errors were resolved, allowing form submissions.

We can visit the login page and register page to see the changes.

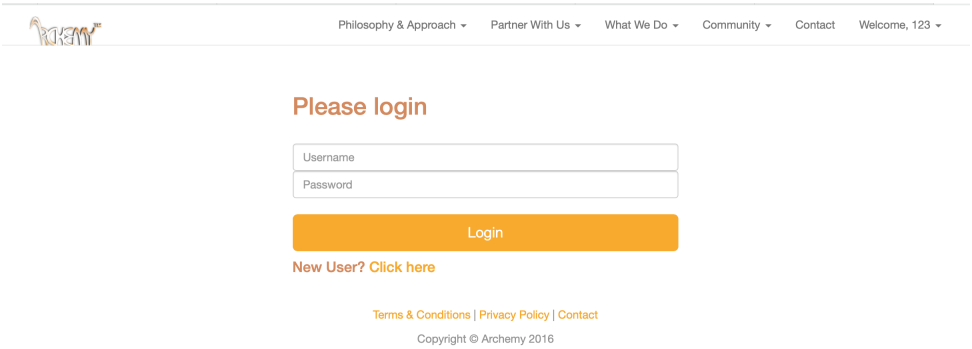

Figure 9: Login Page

The Django server output is shown in Figure 10.

```
[16/May/2025 03:02:10] "GET /login/?next=/index/ HTTP/1.1" 200 7926
[16/May/2025 03:02:30] "POST /login/?next=/index/ HTTP/1.1" 302 0
[16/May/2025 03:02:32] "GET /login/ HTTP/1.1" 200 7926
```

Figure 10: Django Server Output

7.2 View and Syntax Corrections

During troubleshooting, minor code corrections were made in ‘archemywebapp/views.py‘:

- An ‘ImportError‘ for ‘settings‘ was resolved by adding ‘from django.conf import settings‘.
- Python 3 ‘SyntaxError‘ for ‘print line‘ was corrected to ‘print(line)‘.

7.3 Root URL Redirection

To improve user experience, a redirect was configured for the root path (‘/‘) of the application.

- The main ‘archemy/urls.py‘ was modified to include a ‘RedirectView‘ that permanently redirects requests from ‘/‘ to ‘/index/‘.

```
from django.views.generic.base import RedirectView
# ... other imports

urlpatterns = [
    url(r'^$', RedirectView.as_view(url='/index/', permanent=True), name='go-to-
        index'),
    # ... other url patterns
]
```

Listing 4: Root URL redirection in archemy/urls.py

By now, the whole system is working and we successfully migrate the database to Alibaba Cloud RDS and deploy the Django application to Azure VM.

I register a user by following the steps in Figure 11.

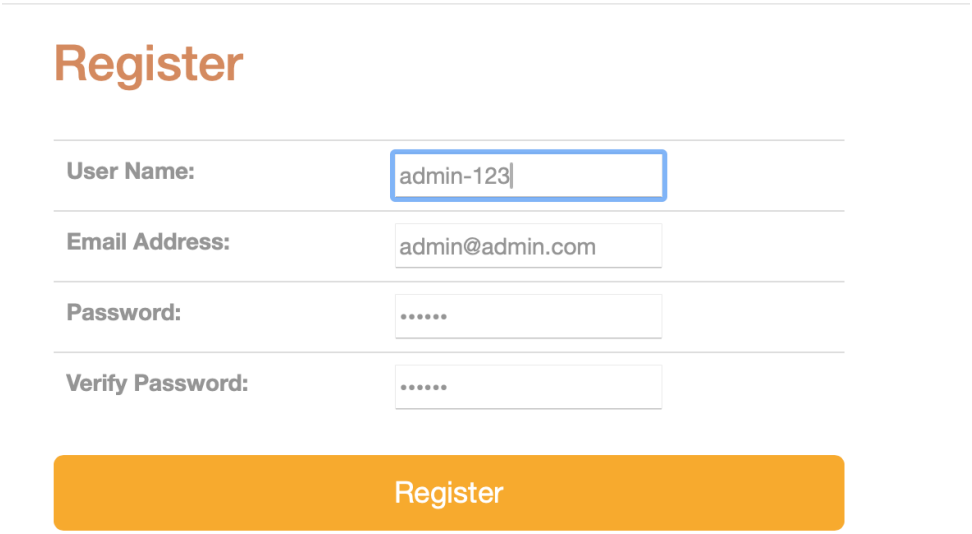

Figure 11: Register Page

We can visit the homepage with http://20.84.124.167:8000/login/?next=/index/.

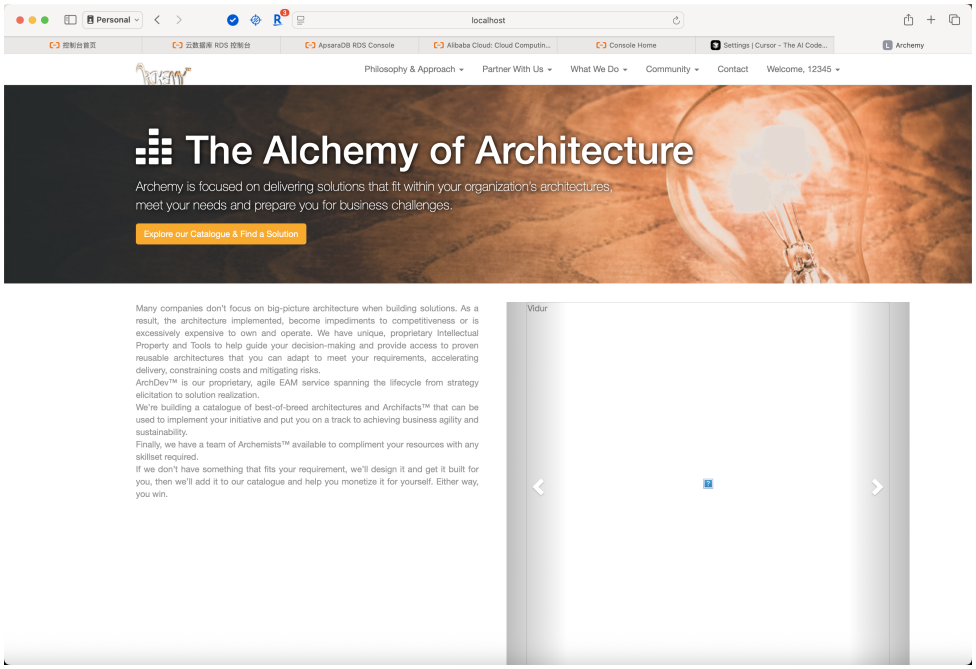

Figure 12: Homepage

8 Conclusion and Future Work

The migration of the ‘archemy-webapp‘ Django application to an Azure VM and its database to Alibaba Cloud RDS has been largely successful. Users can now register and log in, and several critical page rendering issues have been addressed. The process highlighted the complexities of migrating legacy systems, particularly in managing Python version compatibility, resolving obscure dependency conflicts, and reconfiguring integrated security components like OpenLDAP and Fortress Security.

Key learnings include:

- The necessity of iterative Python environment testing for older Django versions.
- The importance of detailed error message analysis for diagnosing both application-level and system-level issues.
- The requirement for careful adaptation of configuration files when moving between environments (e.g., ‘fortress.properties‘).

A Appendix

Appendix 1 - Project Structure

```
.
├── App
│   ├── adfutils.jar
│   ├── archemy-security-1.0-SNAPSHOT-jar-with-dependencies.jar
│   ├── Archemy.jws
│   ├── ArchemyAppModel
│   ├── ArchemyAppView
│   ├── GlassFish Deployment Archives
│   ├── mysql-connector-java-5.1.40-bin.jar
│   ├── mysql-connector-java-5.1.47-bin.jar
│   ├── mysql-connector-java-8.0.16.jar
│   ├── pom.xml
│   ├── resourcebundles
│   ├── src
│   └── WebLogic Deployment Archives-091217
├── archemy-webapp
│   ├── _gitignore
│   ├── archemy
│   ├── archemy-security-1.0-SNAPSHOT-jar-with-dependencies.jar
│   ├── archemywebapp
│   ├── db.sqlite3
│   ├── manage.py
│   ├── README.md
│   └── requirements.txt
├── ArchNavInstallationGuide.pdf
├── Data
│   └── DatabaseImport.sql
├── DB_MODEL
│   └── archemy_schema.mwb
├── FortressSecurity
│   ├── FortressSecurity.iml
│   ├── pom.xml
│   ├── src
│   └── target
└── LICENSE
```

Listing 5: Project Structure

Appendix 2 - Exported SQL Script

```
-- MySQL Script generated by MySQL Workbench
-- Thu May 15 12:41:27 2025
-- Model: New Model     Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
    NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema archemy
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema archemy
-- -----------------------------------------------------
```

```
CREATE SCHEMA IF NOT EXISTS `archemy` DEFAULT CHARACTER SET utf8 ;
USE `archemy` ;

-- -----------------------------------------------------
-- Table `archemy`.`DOMAINS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`DOMAINS` (
  `DOMAIN_ID` INT NOT NULL AUTO_INCREMENT,
  `DOMAIN_NAME` VARCHAR(45) NOT NULL COMMENT 'Contains Domain Description',
  `DOMAIN_DESCRIPTION` VARCHAR(45) NULL,
  PRIMARY KEY (`DOMAIN_ID`),
  UNIQUE INDEX `DOMAIN_NAME_UNIQUE` (`DOMAIN_NAME` ASC) VISIBLE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`DIMENSIONS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`DIMENSIONS` (
  `DIMENSION_ID` INT NOT NULL AUTO_INCREMENT,
  `DIMENSION_NAME` VARCHAR(45) NULL,
  `DOMAIN_ID` INT NULL,
  PRIMARY KEY (`DIMENSION_ID`),
  INDEX `DOMAIN_ID_FK_idx` (`DOMAIN_ID` ASC) VISIBLE,
  UNIQUE INDEX `DIMENSIONS_NAME_UQ_idx` (`DIMENSION_NAME` ASC, `DOMAIN_ID` ASC) VISIBLE
    ,
  CONSTRAINT `DIMENSION_DOMAIN_ID_FK`
    FOREIGN KEY (`DOMAIN_ID`)
    REFERENCES `archemy`.`DOMAINS` (`DOMAIN_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`AREAS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`AREAS` (
  `AREA_ID` INT NOT NULL AUTO_INCREMENT,
  `AREA_PARENT_ID` INT NULL,
  `AREA_ORDER_NO` VARCHAR(45) NULL,
  `AREA_DEPTH_LEVEL` INT NULL,
  `DIMENSION_ID` INT NULL,
  `AREA_NAME` VARCHAR(100) NULL,
  PRIMARY KEY (`AREA_ID`),
  INDEX `DIMENSION_FK_idx` (`DIMENSION_ID` ASC) VISIBLE,
  INDEX `AREA_ID_AREA_PARENT_FK_idx` (`AREA_PARENT_ID` ASC) VISIBLE,
  CONSTRAINT `AREA_DIMENSION_ID_FK`
    FOREIGN KEY (`DIMENSION_ID`)
    REFERENCES `archemy`.`DIMENSIONS` (`DIMENSION_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `AREA_ID_AREA_PARENT_FK`
    FOREIGN KEY (`AREA_PARENT_ID`)
    REFERENCES `archemy`.`AREAS` (`AREA_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`RECURRING_BUS_PROBLEM`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `archemy`.`RECURRING_BUS_PROBLEM` (
  `BUSINESS_PROBLEM` VARCHAR(500) NOT NULL,
  `ID` INT NOT NULL AUTO_INCREMENT,
  `CONTEXT` VARCHAR(80) NULL,
  `TYPE` VARCHAR(80) NULL,
  `DESCRIPTION` VARCHAR(500) NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`KADS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`KADS` (
  `KAD_ID` INT NOT NULL AUTO_INCREMENT,
  `KAD_NAME` VARCHAR(100) NULL,
  `DOMAIN_ID` INT NULL,
  `KAD_LINK` VARCHAR(300) NULL,
  `KAD_LINK_PUBLIC` VARCHAR(300) NULL,
  `KAD_HIT_COUNTER` INT NULL DEFAULT 0,
  `RECURRING_BUS_PROBLEM_ID` INT NULL,
  PRIMARY KEY (`KAD_ID`),
  INDEX `DOMAIN_ID_KAD_FK_idx` (`DOMAIN_ID` ASC) VISIBLE,
  UNIQUE INDEX `KAD_NAME_UNIQUE` (`KAD_NAME` ASC) VISIBLE,
  INDEX `KAD_RECURRING_BUS_FK_idx` (`RECURRING_BUS_PROBLEM_ID` ASC) VISIBLE,
  CONSTRAINT `KAD_DOMAIN_ID_FK`
    FOREIGN KEY (`DOMAIN_ID`)
    REFERENCES `archemy`.`DOMAINS` (`DOMAIN_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `KAD_RECURRING_BUS_FK`
    FOREIGN KEY (`RECURRING_BUS_PROBLEM_ID`)
    REFERENCES `archemy`.`RECURRING_BUS_PROBLEM` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`IADS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`IADS` (
  `IAD_ID` INT NOT NULL AUTO_INCREMENT,
  `IAD_NAME` VARCHAR(100) NULL,
  `IAD_TYPE` VARCHAR(100) NOT NULL,
  `DOMAIN_ID` INT NULL,
  `KAD_ID` INT NULL,
  PRIMARY KEY (`IAD_ID`),
  INDEX `KAD_DOMAIN_ID_FK_idx` (`DOMAIN_ID` ASC) VISIBLE,
  INDEX `IAD_KAD_ID_FK_idx` (`KAD_ID` ASC) VISIBLE,
  UNIQUE INDEX `IAD_NAME_UNIQUE` (`IAD_NAME` ASC, `DOMAIN_ID` ASC, `KAD_ID` ASC)
      VISIBLE,
  CONSTRAINT `IAD_DOMAIN_ID_FK`
    FOREIGN KEY (`DOMAIN_ID`)
    REFERENCES `archemy`.`DOMAINS` (`DOMAIN_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `IAD_KAD_ID_FK`
    FOREIGN KEY (`KAD_ID`)
    REFERENCES `archemy`.`KADS` (`KAD_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
141
143 -- -----------------------------------------------------
    -- Table `archemy`.`CELLS`
145 -- -----------------------------------------------------
    CREATE TABLE IF NOT EXISTS `archemy`.`CELLS` (
147   `CELL_ID` INT NOT NULL AUTO_INCREMENT,
      `AREA_ID` INT NULL,
149   PRIMARY KEY (`CELL_ID`),
      INDEX `CELL_AREA_ID_FK_idx` (`AREA_ID` ASC) VISIBLE,
151   CONSTRAINT `CELL_AREA_ID_FK`
        FOREIGN KEY (`AREA_ID`)
153     REFERENCES `archemy`.`AREAS` (`AREA_ID`)
        ON DELETE CASCADE
155     ON UPDATE CASCADE)
    ENGINE = InnoDB;
157

159 -- -----------------------------------------------------
    -- Table `archemy`.`IAD_TO_CELL_ASSIGNMENTS`
161 -- -----------------------------------------------------
    CREATE TABLE IF NOT EXISTS `archemy`.`IAD_TO_CELL_ASSIGNMENTS` (
163   `IAD_CELL_ASSIGNMENT_ID` INT NOT NULL AUTO_INCREMENT,
      `IAD_ID` INT NULL,
165   `CELL_ID` INT NULL,
      `AREA_ID` INT NULL,
167   PRIMARY KEY (`IAD_CELL_ASSIGNMENT_ID`),
      INDEX `CELL_ASSGN_IAD_ID_FK_idx` (`IAD_ID` ASC) VISIBLE,
169   INDEX `CELL_ASSGN_CELL_ID_FK_idx` (`CELL_ID` ASC) VISIBLE,
      CONSTRAINT `CELL_ASSGN_IAD_ID_FK`
171     FOREIGN KEY (`IAD_ID`)
        REFERENCES `archemy`.`IADS` (`IAD_ID`)
173     ON DELETE CASCADE
        ON UPDATE CASCADE,
175   CONSTRAINT `CELL_ASSGN_CELL_ID_FK`
        FOREIGN KEY (`CELL_ID`)
177     REFERENCES `archemy`.`CELLS` (`CELL_ID`)
        ON DELETE CASCADE
179     ON UPDATE CASCADE)
    ENGINE = InnoDB;
181

183 -- -----------------------------------------------------
    -- Table `archemy`.`RELATIONSHIP_TYPES`
185 -- -----------------------------------------------------
    CREATE TABLE IF NOT EXISTS `archemy`.`RELATIONSHIP_TYPES` (
187   `RELATIONSHIP_ID` INT NOT NULL AUTO_INCREMENT,
      `RELATIONSHIP_TYPE_NAME` VARCHAR(100) NULL,
189   `RELATIONSHIP_TYPE_DESC` VARCHAR(300) NULL,
      `DOMAIN_ID` INT NULL,
191   PRIMARY KEY (`RELATIONSHIP_ID`),
      INDEX `RELATIONSHIP_TO_DOMAIN_ID_FK_idx` (`DOMAIN_ID` ASC) VISIBLE,
193   UNIQUE INDEX `RELATIONSHIP_TYPE_NAME_UNIQUE` (`RELATIONSHIP_TYPE_NAME` ASC, `
          DOMAIN_ID` ASC) VISIBLE,
      CONSTRAINT `RELATIONSHIP_TO_DOMAIN_ID_FK`
195     FOREIGN KEY (`DOMAIN_ID`)
        REFERENCES `archemy`.`DOMAINS` (`DOMAIN_ID`)
197     ON DELETE CASCADE
        ON UPDATE CASCADE)
199 ENGINE = InnoDB;

201

    -- -----------------------------------------------------
```

```
-- Table `archemy`.`IAD_RELATIONSHIPS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`IAD_RELATIONSHIPS` (
  `FROM_IAD_TO_CELL_ASSIGNMENT_ID` INT NOT NULL,
  `TO_IAD_TO_CELL_ASSIGNMENT_ID` INT NOT NULL,
  `RELATIONSHIP_TYPE_ID` INT NOT NULL,
  `IAD_RELATIONSHIP_ID` INT NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`IAD_RELATIONSHIP_ID`),
  INDEX `IAD_CELL_ASSGN_IAD_ID_idx` (`FROM_IAD_TO_CELL_ASSIGNMENT_ID` ASC) VISIBLE,
  INDEX `IAD_CELL_ASSGN_CELL_ID_idx` (`TO_IAD_TO_CELL_ASSIGNMENT_ID` ASC) VISIBLE,
  INDEX `IAD_RLTN_RELATION_ID_FK_idx` (`RELATIONSHIP_TYPE_ID` ASC) VISIBLE,
  CONSTRAINT `IAD_RLTN_IAD_CELL_IAD_ID_FK`
    FOREIGN KEY (`FROM_IAD_TO_CELL_ASSIGNMENT_ID`)
    REFERENCES `archemy`.`IAD_TO_CELL_ASSIGNMENTS` (`IAD_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `IAD_RLTN_IAD_CELL_CELL_ID_FK`
    FOREIGN KEY (`TO_IAD_TO_CELL_ASSIGNMENT_ID`)
    REFERENCES `archemy`.`IAD_TO_CELL_ASSIGNMENTS` (`CELL_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `IAD_RLTN_RELATION_ID_FK`
    FOREIGN KEY (`RELATIONSHIP_TYPE_ID`)
    REFERENCES `archemy`.`RELATIONSHIP_TYPES` (`RELATIONSHIP_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`ORGANIZATION_TYPES`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`ORGANIZATION_TYPES` (
  `ORGANIZATION_TYPE_ID` INT NOT NULL AUTO_INCREMENT,
  `ORGANIZATION_TYPE_NAME` VARCHAR(150) NULL,
  `ORGANIZATION_TYPE_DESCRIPTION` VARCHAR(300) NULL,
  `DOMAIN_ID` INT NULL,
  PRIMARY KEY (`ORGANIZATION_TYPE_ID`),
  INDEX `ORG_TYPE_TO_DOMAIN_ID_FK_idx` (`DOMAIN_ID` ASC) VISIBLE,
  UNIQUE INDEX `ORGANIZATION_TYPE_NAME_UNIQUE` (`ORGANIZATION_TYPE_NAME` ASC, `
      DOMAIN_ID` ASC) VISIBLE,
  CONSTRAINT `ORG_TYPE_TO_DOMAIN_ID_FK`
    FOREIGN KEY (`DOMAIN_ID`)
    REFERENCES `archemy`.`DOMAINS` (`DOMAIN_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`ORGANIZATIONS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`ORGANIZATIONS` (
  `ORGANIZATION_ID` INT NOT NULL AUTO_INCREMENT,
  `ORGANIZATION_NAME` VARCHAR(100) NULL,
  `DOMAIN_ID` INT NULL,
  `ORGANIZATION_TYPE_ID` INT NULL,
  PRIMARY KEY (`ORGANIZATION_ID`),
  INDEX `ORGANIZATION_TO_DOMAIN_ID_FK_idx` (`DOMAIN_ID` ASC) VISIBLE,
  INDEX `ORGANIZATIONS_TO_ORG_TYPE_idx` (`ORGANIZATION_TYPE_ID` ASC) VISIBLE,
  UNIQUE INDEX `ORGANIZATION_NAME_UNIQUE` (`ORGANIZATION_NAME` ASC, `DOMAIN_ID` ASC)
      VISIBLE,
  CONSTRAINT `ORGANIZATION_TO_DOMAIN_ID_FK`
```

```
    FOREIGN KEY (`DOMAIN_ID`)
    REFERENCES `archemy`.`DOMAINS` (`DOMAIN_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `ORGANIZATIONS_TO_ORG_TYPE`
    FOREIGN KEY (`ORGANIZATION_TYPE_ID`)
    REFERENCES `archemy`.`ORGANIZATION_TYPES` (`ORGANIZATION_TYPE_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`NODES`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`NODES` (
  `NODE_ID` INT NOT NULL AUTO_INCREMENT,
  `NODE_NAME` VARCHAR(100) NOT NULL,
  `ORGANIZATION_ID` INT NULL,
  PRIMARY KEY (`NODE_ID`),
  INDEX `NODE_TO_ORGANIZATION_ID_idx` (`ORGANIZATION_ID` ASC) VISIBLE,
  CONSTRAINT `NODE_TO_ORGANIZATION_ID`
    FOREIGN KEY (`ORGANIZATION_ID`)
    REFERENCES `archemy`.`ORGANIZATIONS` (`ORGANIZATION_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`NODE_RELATIONSHIPS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`NODE_RELATIONSHIPS` (
  `FROM_NODE_ID` INT NULL,
  `TO_NODE_ID` INT NULL,
  `NODE_RELATIONSHIPS_ID` INT NOT NULL AUTO_INCREMENT,
  `RELATIONSHIP_TYPE_ID` INT NULL,
  PRIMARY KEY (`NODE_RELATIONSHIPS_ID`),
  INDEX `NODE_RLTN_NODE_NODE_ID_FK_idx` (`FROM_NODE_ID` ASC) VISIBLE,
  INDEX `NODE_RLTN_NODE_ID_FK_2_idx` (`TO_NODE_ID` ASC) VISIBLE,
  INDEX `NODE_RLTN_RLTN_TYPE_ID_idx` (`RELATIONSHIP_TYPE_ID` ASC) VISIBLE,
  CONSTRAINT `NODE_RLTN_NODE_ID_FK`
    FOREIGN KEY (`FROM_NODE_ID`)
    REFERENCES `archemy`.`NODES` (`NODE_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `NODE_RLTN_NODE_ID_FK_2`
    FOREIGN KEY (`TO_NODE_ID`)
    REFERENCES `archemy`.`NODES` (`NODE_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `NODE_RLTN_RLTN_TYPE_ID`
    FOREIGN KEY (`RELATIONSHIP_TYPE_ID`)
    REFERENCES `archemy`.`RELATIONSHIP_TYPES` (`RELATIONSHIP_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`IAD_TO_NODE_ASSIGNMENTS`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`IAD_TO_NODE_ASSIGNMENTS` (
```

```
    `ID` INT NOT NULL AUTO_INCREMENT,
    `IAD_NODE_ASSIGN_NODE_ID` INT NULL,
    `IAD_NODE_ASSIGN_IAD_ID` INT NULL,
    PRIMARY KEY (`ID`),
    INDEX `NODE_ASSGN_TO_IAD_ID_idx` (`IAD_NODE_ASSIGN_IAD_ID` ASC) VISIBLE,
    CONSTRAINT `NODE_ASSGN_TO_IAD_ID`
      FOREIGN KEY (`IAD_NODE_ASSIGN_IAD_ID`)
      REFERENCES `archemy`.`IADS` (`IAD_ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT `NODE_ASSGN_TO_NODE_ID`
      FOREIGN KEY (`ID`)
      REFERENCES `archemy`.`NODES` (`NODE_ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`KAD_DIMENSIONS_AREA`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`KAD_DIMENSIONS_AREA` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `DIMENSION_ID` INT NOT NULL,
    `KAD_ID` INT NOT NULL,
    `AREA_ID` INT NULL,
    `AREA_PARENT_ID` INT NULL,
    PRIMARY KEY (`ID`),
    INDEX `KAD_DIMENSION_ID_FK_idx` (`DIMENSION_ID` ASC) VISIBLE,
    INDEX `KAD_DIMENSIONS_KAD_ID_FK_idx` (`KAD_ID` ASC) VISIBLE,
    INDEX `KAD_DIM_AREA_idx` (`AREA_ID` ASC) VISIBLE,
    UNIQUE INDEX `KAD_UNQ_IDX` (`DIMENSION_ID` ASC, `KAD_ID` ASC, `AREA_ID` ASC, `
        AREA_PARENT_ID` ASC) VISIBLE,
    CONSTRAINT `KAD_DIM_ID_FK`
      FOREIGN KEY (`DIMENSION_ID`)
      REFERENCES `archemy`.`DIMENSIONS` (`DIMENSION_ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT `KAD_DIM_KAD_ID_FK`
      FOREIGN KEY (`KAD_ID`)
      REFERENCES `archemy`.`KADS` (`KAD_ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    CONSTRAINT `KAD_DIM_AREA`
      FOREIGN KEY (`AREA_ID`)
      REFERENCES `archemy`.`AREAS` (`AREA_ID`)
      ON DELETE CASCADE
      ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `archemy`.`KAD_TO_NODE_ASSIGNMENT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `archemy`.`KAD_TO_NODE_ASSIGNMENT` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `KAD_ID` INT NULL,
    `NODE_ID` INT NULL,
    PRIMARY KEY (`ID`),
    INDEX `KAD_NODE_ASGN_KAD_FK_idx` (`KAD_ID` ASC) VISIBLE,
    INDEX `KAD_NODE_ASGN_NODE_ID_FK_idx` (`NODE_ID` ASC) VISIBLE,
    CONSTRAINT `KAD_NODE_ASGN_KAD_FK`
      FOREIGN KEY (`KAD_ID`)
```

```
389       REFERENCES `archemy`.`KADS` (`KAD_ID`)
          ON DELETE CASCADE
391       ON UPDATE CASCADE,
        CONSTRAINT `KAD_NODE_ASGN_NODE_ID_FK`
393       FOREIGN KEY (`NODE_ID`)
          REFERENCES `archemy`.`NODES` (`NODE_ID`)
395       ON DELETE CASCADE
          ON UPDATE CASCADE)
397   ENGINE = InnoDB;


399
    -- -----------------------------------------------------
401 -- Table `archemy`.`CUSTOMER_INFO`
    -- -----------------------------------------------------
403 CREATE TABLE IF NOT EXISTS `archemy`.`CUSTOMER_INFO` (
      `CUSTOMER_NAME` VARCHAR(150) NULL,
405   `INDUSTRY` VARCHAR(150) NULL,
      `USER_ID` VARCHAR(150) NOT NULL,
407   PRIMARY KEY (`USER_ID`))
    ENGINE = InnoDB;
409


411 -- -----------------------------------------------------
    -- Table `archemy`.`KAD_REGISTRATION`
413 -- -----------------------------------------------------
    CREATE TABLE IF NOT EXISTS `archemy`.`KAD_REGISTRATION` (
415   `USER_ID` VARCHAR(150) NOT NULL,
      `ID` INT NOT NULL AUTO_INCREMENT,
417   `KAD_ID` INT NOT NULL,
      `MATURITY_RATING` INT NOT NULL,
419   `DEPLOYMENT_STATUS` VARCHAR(100) NOT NULL,
      `APPLICABILITY_EXTENT` VARCHAR(100) NOT NULL,
421   `BENEFIT_RATING` INT NOT NULL,
      `COMMENTS` VARCHAR(500) NULL,
423   PRIMARY KEY (`ID`),
      INDEX `KAD_REGISTER_TO_KAD_FK_idx` (`KAD_ID` ASC) VISIBLE,
425   UNIQUE INDEX `KAD_REGISTER_UK_IDX` (`USER_ID` ASC, `KAD_ID` ASC) VISIBLE,
      CONSTRAINT `KAD_REGISTER_TO_KAD_FK`
427     FOREIGN KEY (`KAD_ID`)
        REFERENCES `archemy`.`KADS` (`KAD_ID`)
429     ON DELETE CASCADE
        ON UPDATE CASCADE)
431 ENGINE = InnoDB;


433
    SET SQL_MODE=@OLD_SQL_MODE;
435 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
    SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Listing 6: Exported SQL Script