PROG8850 Assignment 4 – Question 1 PDF Submission

Question 1: Analysis and Integration of Database Migration Tools

1.1 Tool Overview and Comparison

◈ Tool 1: Flyway

Overview:
Flyway is an open-source database migration tool focused on simplicity and ease of use. It manages database changes using versioned SQL scripts, making it easy to implement and understand. It integrates well with CI/CD pipelines and supports numerous relational databases.

Key Features:

- SQL-based version-controlled migrations
- Supports Java-based migrations (optional)
- CLI, Maven, Gradle, and Docker support
- Integrates with GitHub Actions, Jenkins, GitLab CI/CD
- Supports MySQL, PostgreSQL, Oracle, SQL Server, etc.

◈ Tool 2: Liquibase

Overview:
Liquibase is an advanced open-source database schema change management tool. It supports changes defined in SQL as well as XML, JSON, or YAML formats. It offers powerful tracking, rollback, and documentation capabilities.

Key Features:

- Supports declarative migrations (SQL, XML, JSON, YAML)
- Rollback and changelog tracking
- Compatible with CI tools (Jenkins, GitHub Actions, Bitbucket)
- Broad database support including MySQL, Oracle, PostgreSQL, DB2, MongoDB

- **Comparison Table**

Feature	Flyway	Liquibase
Ease of Use	Simple, script-based approach using raw SQL	Moderate complexity due to multiple supported formats (XML/JSON/YAML/SQL)
CI/CD Integration	Seamless with CLI, Docker, GitHub Actions, Jenkins, GitLab	Supports Maven, Ant, Jenkins, GitHub Actions with XML/YAML changelogs
Supported Databases	MySQL, PostgreSQL, Oracle, SQL Server, H2, etc.	Extensive – MySQL, PostgreSQL, Oracle, SQL Server, DB2, MongoDB, etc.
Rollback Capability	Limited rollback (only via manual down scripts)	Strong rollback support via rollback command and rollbackCount
Scripting Language	SQL or Java	SQL, XML, JSON, YAML

1.2 Integration Strategy for CI/CD

To integrate both Flyway and Liquibase in a software project CI/CD pipeline:

🔧 **Flyway Integration Strategy:**

- Store migration scripts in versioned folders (e.g., migrations/init, migrations/update)
- Use Docker image or CLI in the GitHub Actions pipeline to apply migrations
- Run migrations in a job before application deployment
- Example Command:

bash

```
docker run --rm -v ${{ github.workspace }}/migrations:/flyway/sql flyway/flyway -url=jdbc:mysql://db:3306/mydb -user=root -password=root migrate
```

🔧 **Liquibase Integration Strategy:**

- Maintain changelogs in XML/YAML under db/changelogs
- Add a build step in CI/CD pipeline (GitHub Actions, Jenkins) to run Liquibase CLI
- Use parameters for database credentials and schema tracking
- Example Command:

bash

liquibase --changeLogFile=db/changelogs/db.changelog.xml update

---

Conclusion:
Flyway is excellent for teams preferring SQL-first migrations with simple pipelines, while Liquibase is ideal for complex enterprise-grade scenarios requiring rollback and format flexibility. Depending on the use case, either tool can be effectively integrated into modern DevOps workflows.