UNIT – V

Databases & Deployment

Functional dependency defines the relationship of two or more attributes, typically between the primary key and non-key attributes of another table. It is also defined by the relation of one attribute to another attribute in DBMS.

empId → { empName, skill, dependent, contract, project}, → Here, empId can determine or defines the values of fields empName, dependent, contract and employee project

Username Tables:

userName → dateCreate here if we can know the userName like we have email account if we know the email Id of user then there is possibility to find the date when account was created.

Multivalued Dependency:

Multivalency Dependency occurs in such a condition or time when two or more attributes in table are independent to each other but, both of them depend upon the third attributes.

Employee Table:

The attributes like empName, skill, dependent, contract, project all are independent of each other means not depends on one another but depends upon empId example empName can determine skill, or any other employee attribute because there can be or even more than one employee with same name or constraints.

empId->skill empId-> contract empId-> project empId-> dependent

These all of the columns is the multivalued dependency on the empId

Username Table:

We only have two attributes here, but there are no multiple attributes that are independent of each other but rely solely on the third variable.

userName → dateCreate here dateCreate is an attribute that depends or relate upon

the userName only dateCreate when there is not sufficient to find anything. b)

Minimal key is the minimum no of attributes which can find out other attributes of a table

i.e., a primary key or the candidate key. **In the Context of Employee Table:**

empId → { empName, skill, dependent, contract, project}

In the Context of Username Table:

userName → dateCreate

In the Context of Subject Table:

Consider Subject table which has sub_Code, subName sub_Code->subName

In the Context of Enrollment Table:

Considering the enrollment table which has the attributes like: enrollment Id, Name of employee, field in which employee enrolled and date

c)

We have the following Employee and Username Tables:

In Context of Employee Table:

Employee table is not in normalize or the normal forms. Because the Attributes in it like: Skill, project, contract and dependent attributes might have one or more values. According to the 1NF principle every field must contain the atomic values if they don't have the atomic value. There is need to decompose the table since the table should have the 1 value in each field.

In the Context of Username Table:

It is normalizing one Since it has two fields[UserName and dateCreate] in which both have atomic values or data , is fully functional dependent, no transition dependency etc.

d)

Normalization, Decomposition process will be done.

Normalization is a process or technique of organizing or collecting the data in database. It is mainly done for two purposes: Eliminating the redundancy or even the useless data

In 1st NF:

Every field must contain the single atomic value and the attribute like: skill, project, contract and dependent attribute has one or more than the decompose table so that the each and every field has atomic value which will increase the number of tuples in the table name "employee".

In the 2nd NF:

Each table should be at 1st NF.

- There should not be any functional dependency. So, in this case, after it is in 1st NF table is in 2nd NF Since the empId can find out all the attributes of the employees.

In 3rd NF:

- Table should be at 2nd NF Form.
- There should not be any transitive dependency in the table in which the non-primitive attribute can find another non-primitive attribute i.e., empName, skill, dependent, project is the non-primitive attribute and they cannot find the each other but the main prime attribute can or able to find all of them.

In BCNF:

- Table should be at 3rd NF.
- The LHS Side of attribute should have the candidate key or the super key. • So, In this case empId → { empName, skill, dependent, contract, project}, The attribute empId is a primary key and can find out all other attributes.

In the 4th NF:

Table should be at BCNF Form.

There should not be any multivalued Dependency.

So, in current Employee Table context, Employee might contain the multivalued dependency I.e.: skills, projects [0 or more], dependency [0 or more] and contract [1 or more]. So, there is lots of multivalued attributes or dependency on the empId which might increase the no of entries in the table which might increase the no of entries in the table after making it to 1stNF.

In the case of making the Employee table in 4th NF, it will decompose the employee table into following tables: empId → { empName, skill, dependent, contract, project}

EmployeeSkils

empId, empName,

skill

EmployeeDependency empId, empName, dependent **EmployeeContract** empId, empName,

contract

EmployeeProject empId, empName,

project

There is no need to change the Username Table since it is already on 4th NF.

Structured Query Language

SQL Tutorial

SQL tutorial provides basic and advanced concepts of SQL. Our SQL tutorial is designed for both beginners and professionals.

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.

SQL is not a database system, but it is a query language.

Suppose you want to perform the queries of SQL language on the stored data in the database. You are required to install any database management system in your systems, for example, Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2, etc.

What is SQL?

SQL is a short-form of the structured query language, and it is pronounced as S-Q-L or sometimes as See-Quell.

This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDSMS.

You can easily create and manipulate the database, access and modify the table rows and columns, etc. This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.

If you want to get a job in the field of data science, then it is the most important query language to learn. Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

Why SQL?

Nowadays, SQL is widely used in data science and analytics. Following are the reasons which explain why it is widely used:

- The basic use of SQL for data professionals and SQL users is to insert, update, and delete the data from the relational database.
- SQL allows the data professionals and users to retrieve the data from the relational database

management systems.
- It also helps them to describe the structured data.
- It allows SQL users to create, drop, and manipulate the database and its tables.
- It also helps in creating the view, stored procedure, and functions in the relational database.
- It allows you to define the data and modify that stored data in the relational database.
- It also allows SQL users to set the permissions or constraints on table columns, views, and stored procedures.

History of SQL

"A Relational Model of Data for Large Shared Data Banks" was a paper which was published by the great computer scientist "E.F. Codd" in 1970.

The IBM researchers Raymond Boyce and Donald Chamberlin originally developed the SEQUEL (Structured English Query Language) after learning from the paper given by E.F. Codd. They both developed the SQL at the San Jose Research laboratory of IBM Corporation in 1970.

At the end of the 1970s, relational software Inc. developed their own first SQL using the concepts of E.F. Codd, Raymond Boyce, and Donald Chamberlin. This SQL was totally based on RDBMS. Relational Software Inc., which is now known as Oracle Corporation, introduced the Oracle V2 in June 1979, which is the first implementation of SQL language. This Oracle V2 version operates on VAX computers.

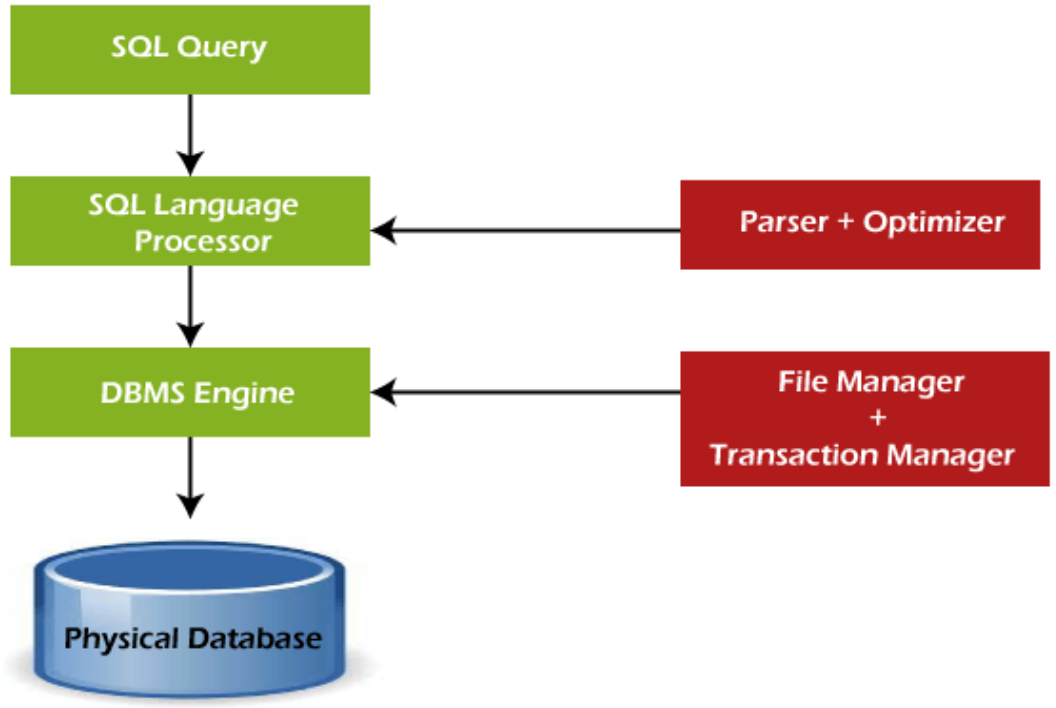
Process of SQL

When we are executing the command of SQL on any Relational database management system, then the system automatically finds the best routine to carry out our request, and the SQL engine determines how to interpret that particular command.

Structured Query Language contains the following four components in its process:

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

A classic query engine allows data professionals and users to maintain non-SQL queries. The architecture of SQL is shown in the following diagram:



Some SQL Commands

The SQL commands help in creating and managing the database. The most common SQL commands which are highly used are mentioned below:

1. CREATE command
2. UPDATE command
3. DELETE command
4. SELECT command
5. DROP command
6. INSERT command

CREATE Command

This command helps in creating the new database, new table, table view, and other objects of the database.

UPDATE Command

This command helps in updating or changing the stored data in the database.

DELETE Command

This command helps in removing or erasing the saved records from the database tables. It erases single or multiple tuples from the tables of the database.

SELECT Command

This command helps in accessing the single or multiple rows from one or multiple tables of the database. We can also use this command with the WHERE clause.

DROP Command

This command helps in deleting the entire table, table view, and other objects from the database.

INSERT Command

This command helps in inserting the data or records into the database tables. We can easily insert the records in single as well as multiple rows of the table.

SQL vs No-SQL

The following table describes the differences between the SQL and NoSQL, which are necessary to understand:

SQL	No-SQL
1. SQL is a relational database management system.	1. While No-SQL is a non-relational or distributed database management system.
2. The query language used in this database system is a structured query language.	2. The query language used in the No-SQL database systems is a non-declarative query language.
3. The schema of SQL databases is predefined, fixed, and static.	3. The schema of No-SQL databases is a dynamic schema for unstructured data.
4. These databases are vertically scalable.	4. These databases are horizontally scalable.
5. The database type of SQL is in the form of tables, i.e., in the form of rows and columns.	5. The database type of No-SQL is in the form of documents, key-value, and graphs.
6. It follows the ACID model.	6. It follows the BASE model.

7. Complex queries are easily managed in the SQL database.	7. NoSQL databases cannot handle complex queries.
8. This database is not the best choice for storing hierarchical data.	8. While No-SQL database is a perfect option for storing hierarchical data.
9. All SQL databases require object-relational mapping.	9. Many No-SQL databases do not require object-relational mapping.
10. Gauges, CircleCI, Hootsuite, etc., are the top enterprises that are using this query language.	10. Airbnb, Uber, and Kickstarter are the top enterprises that are using this query language.
11. SQLite, Ms-SQL, Oracle, PostgreSQL, and MySQL are examples of SQL database systems.	11. Redis, MongoDB, Hbase, BigTable, CouchDB, and Cassandra are examples of NoSQL database systems.

Advantages of SQL

SQL provides various advantages which make it more popular in the field of data science. It is a perfect query language which allows data professionals and users to communicate with the database. Following are the best advantages or benefits of Structured Query Language:

1. No programming needed

SQL does not require a large number of coding lines for managing the database systems. We can easily access and maintain the database by using simple SQL syntactical rules. These simple rules make the SQL user-friendly.

2. High-Speed Query Processing

A large amount of data is accessed quickly and efficiently from the database by using SQL queries. Insertion, deletion, and updation operations on data are also performed in less time.

3. Standardized Language

SQL follows the long-established standards of ISO and ANSI, which offer a uniform platform across the globe to all its users.

4. Portability

The structured query language can be easily used in desktop computers, laptops, tablets, and even smartphones. It can also be used with other applications according to the user's requirements.

5. Interactive language

We can easily learn and understand the SQL language. We can also use this language for communicating with the database because it is a simple query language. This language is also

used for receiving the answers to complex queries in a few seconds.

6. More than one Data View

The SQL language also helps in making the multiple views of the database structure for the different database users.

Disadvantages of SQL

With the advantages of SQL, it also has some disadvantages, which are as follows:

1. Cost

The operation cost of some SQL versions is high. That's why some programmers cannot use the Structured Query Language.

2. Interface is Complex

Another big disadvantage is that the interface of Structured query language is difficult, which makes it difficult for SQL users to use and manage it.

3. Partial Database control

The business rules are hidden. So, the data professionals and users who are using this query language cannot have full database control.

Data persistence using Spring

I'm used to using Spring Roo to generate my entities and having it handle injecting the entityManager as well as the persist and other methods via AspectJ classes. Now I'm trying to use Spring Boot to do something simple that will write things to the database ...

```
@Entity
@Table(name = "account")
publicclassAccount {

transientEntityManagerentityManager;

@Id
@GeneratedValue
privateLong id;

@Column(name = "username", nullable = false, unique = true)
private String username;

@Column(name = "password", nullable = false)
private String password;

  ... getters and setters

@Transactional
publicvoidpersist() {
```

```
if (this.entityManager == null) this.entityManager = entityManager();
this.entityManager.persist(this);
  }

@Transactional
public Account merge() {
if (this.entityManager == null) this.entityManager = entityManager();
Accountmerged=this.entityManager.merge(this);
this.entityManager.flush();
return merged;
  }
```

When I'm calling persist or merge, entityManager is obviously null.

I've also tried adding `implements CrudRepository<Account, Long>` to the `Account` class to see it'll give me that functionality via a Default Implementation, but what I'm getting is simply empty classes that needs to be filled in.

I've had a look at the Spring Boot docs, they cover it very briefly omitting just enough detail to so that it's not obvious what I'm missing.

I have an Application class that bootstraps the application:

```
@Configuration
@ComponentScan
@EnableAutoConfiguration
publicclassApplication {

publicstaticvoidmain(String[] args)throws Exception {
SpringApplication.run(Application.class, args);
  }

}
```

My properties file looks like this:

```
spring.application.name: Test Application

spring.datasource.url: jdbc:mysql://localhost/test
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driverClassName=com.mysql.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
```

This database is automatically being created thanks to the `ddl-auto=update` property

What is the correct way to persist entities in Spring Boot + JPA and if what I've done is correct so far, how do I "autowire" or auto-create the entityManager?

JDBC Agile development principles

What are the Agile Principles?

There are 12 agile principles outlined in The Agile Manifesto in addition to the 4 agile values. These 12 principles for agile software development help establish the tenets of the agile mindset. They are not a set of rules for practicing agile, but a handful of principles to help instill agile thinking.

Below we will review each of the 12 agile principles and describe how they may be practiced.

Agile Principle 1

"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

The best ways to ensure you make customers happy while continuously delivering valuable software are to ship early, iterate frequently, and listen to your market continually.

Unlike traditional approaches to product development, which have notoriously long development cycles, agile principles encourage minimizing the time between ideation and launch. The idea is to get a working product in the hands of customers as soon as possible. Doing this successfully means product managers are able to quickly get a minimum viable product (MVP) out and into the world and use it to get feedback from real customers. This feedback is then fed back into the product development process and used to inform future releases.

How it looks in practice:

- Product teams use minimum viable products and rapid experimentation to test hypothesis and validate ideas.
- Frequent releases help fuel a continuous feedback cycle between customer and product.
- Shipped and done are not the same thing. Instead of releasing a "finished" product, iterations continue to make incremental improvements to product based on customer and market feedback.

Agile Principle 2

"Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."

In the world around us, change is the only constant. Agile principles and values support responding to these changes rather than moving forward in spite of them. Previous approaches to product development were often change adverse; detailed, well-documented

plans were made before development began and were set in stone regardless of new findings. Agile principles support observing changing markets, customer needs, and competitive threats and changing course when necessary.

How it looks in practice:

- Product teams are guided by high-level strategic goals and perhaps even themes below those goals. The product department's success is measured against progress toward those strategic goals rather than by delivery of a predefined feature set.
- Product constantly has its ear to the ground monitoring the market, customer feedback, and other factors which could influence product direction. When actionable insight is uncovered, plans are adjusted to better serve customer and business needs.
- Product strategy and tactical plans are reviewed, adjusted, and shared on a regular cadence to reflect changes and new findings. As such, product needs to manage the expectations of executive stakeholders appropriately and ensure they understand the *why* behind changes.

Agile Principle 3

"Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."

Agile philosophy favors breaking a product's development into smaller components and "shipping" those components frequently. Using an agile approach, therefore—and building in more frequent mini-releases of your product—can speed the product's overall development.

This agile approach, with short-term development cycles of smaller portions of the product, results in less time spent drafting and poring over the large amounts of documentation that characterizes Waterfall product development. More importantly, this frequent-release approach creates more opportunities for you and your teams to validate your product ideas and strategies from the qualified constituencies who see each new release.

How it looks in practice:

- Agile development cycles, often called "sprints" or "iterations" break down product initiatives into smaller chunks that can be completed in a set timeframe. Often this timeframe is between 2 and 4 weeks which truly is a sprint if you consider the marathon-like development cycles waterfall teams often follow.
- Another popular alternative to agile sprints is continuous deployment. This method of shipping software frequently works less in terms of predetermined time boxes and more in terms of simply deciding what to do and doing it.

Agile Principle 4

"Business people and developers must work together daily throughout the project."

Communication is a critical component of any project or team's success, and agile principles essentially mandate that it's a daily event. It takes a village to raise a child they say, and that applies to product as well.

A successful product requires insight from the business and technical sides of an organization which can only happen if these two teams work together consistently. Regular communication between business people and developers helps improve alignment across the organization by building trust and transparency.

How it looks in practice:

- Cross-functional agile product development teams include product people. This means that product is represented on the development team and bridges the gap between technical and business aspects of the product.
- Daily update meetings, or standups, are one technique many agile shops use to put this principle in practice and keep everyone connected.

Agile Principle 5

"Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."

A key part of the agile philosophy is empowering individuals and teams through trust and autonomy. The agile team needs to be carefully built to include the right people and skill sets to get the job done, and responsibilities need to be clearly defined before the beginning of a project. Once the work has begun, however, there's no place in agile for micromanagement or hand holding.

How it looks in practice:

- Product must clearly ensure engineering understands strategy and requirements before development starts. This means not only sharing user stories with the cross-functional team but also the bigger picture outlined in the product roadmap.
- Product is not responsible for explaining "how" something should be built. They need to share what and why, but it's the delivery team's job to determine the how. Furthermore, during sprints product does not micromanage outcome, instead they make themselves available to answer questions and provide support as needed.

Agile Principle 6

"The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."

With so many distributed or remote development teams these days, this principle gets a bit of critique. But at the root of it, effective communication with developers means getting these conversations out of Slack and email and favoring more human interaction (even if done by video conference calls). The overall objective behind this principle is to encourage product people and developers to truly communicate in real time about the product, requirements, and the high-level strategy driving those things.

How it looks in practice:

- Daily standup meetings
- Collaborative backlog grooming sessions
- Sprint planning meetings
- Frequent demos
- Pair-programming

Agile Principle 7

"Working software is the primary measure of progress."

Proponents of the agile philosophy are quick to remind us that we're in the business of building software, and that's where our time should be spent. Perfect, detailed documentation is secondary to working software. This mentality pushes to get products to the market quickly rather than let documentation or an "it's not done until it's perfect" mentality become a bottleneck. The ultimate measure for success is a working product that customers love.

How it looks in practice:

- Designing and releasing "Minimum Viable Features" rather than fully-developed feature sets means thinking first and foremost about the smallest things we can ship to start getting customer feedback and validate as we continue to build software.
- A fail fast mentality means moving forward even in times of uncertainty and testing ideas rapidly.
- Ship software often: a useful product now is better than a perfect one later.

Agile Principle 8

"Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."

Keeping up with a demanding, rapid release schedule can be taxing on a team. Especially if expectations are set too high. Agile principles encourage us to be mindful of this and set realistic, clear expectations. The idea is to keep morale high and improve work-life balance to prevent burnout and turnover among members of cross functional teams.

How it looks in practice:

- Before every sprint, careful consideration of the amount of work that can be committed to is made. Development teams don't over promise on what they can and cannot deliver. Effort estimations are a common practice in setting output expectations for development teams.
- Everyone agrees on what will get done during a sprint. Once a sprint has begun, no additional tasks are to be added except in rare cases.
- Product managers should act as gatekeepers to reduce the noise from other stakeholders and to avoid squeezing in additional unplanned work during an ongoing sprint.
- Product people should do their part in promoting a sense of psychological safety across the cross-functional team that encourages open communication and freely flowing feedback.

Agile Principle 9

"Continuous attention to technical excellence and good design enhances agility."

While the agile philosophy encourages shorter cycles and frequent releases, it also puts emphasis on the importance of keeping things neat and tidy so they don't cause problems in the future. Product managers often forget about this aspect of development because they mostly don't spend their days wading through their products' codebases, but it is still of the utmost importance to them.

How it looks in practice:

- The team needs to be cognizant of technical debt and the technical debt implications of any new features or initiatives added to the backlog. Developers and product need to work together to understand if and when technical debt is acceptable.
- On a regular basis, product will need to allocate development resources to refactoring efforts. Refactoring cannot be an afterthought, it needs to be an ongoing consideration.

Agile Principle 10

"Simplicity—the art of maximizing the amount of work not done—is essential."

You've probably heard of the 80/20 rule—the concept that you can usually get 80% of your intended results with just 20% of the work. Agile principles encourage thinking this way; doing the things that can have the most impact. In a product management context this means having a laser sharp focus on organizational objectives and making some cutthroat prioritization decisions. Agile principles discourage building merely for the sake of building by emphasizing the importance of being strategic and building with purpose.

How it looks in practice:

- Product managers need to make very focused product decisions and closely align product strategy with organizational goals while being extremely picky about what user stories and features actually make the cut. Using prioritization techniques to prioritize initiatives by effort and predicted impact is one way product teams can apply this agile principle to product development.
- The short sprints that agile is characterized by present many opportunities for rapid testing and experimentation which can help reduce uncertainty around whether initiatives will truly have the predicted impact. Using experiments to validate ideas before building them up to spec is a great way to weed out bad ideas and identify good ones.

Agile Principle 11

"The best architectures, requirements, and designs emerge from self-organizing teams."

In traditional software development methodologies, you'll often see pyramid shaped teams where management makes key decisions for contributors. Agile principles suggest the use of self-organizing teams which work with a more "flat" management style where decisions are made as a group rather than by a singular manager or management team. The concept ties into agile's value of teams and interactions over processes and tools, and the intent behind the concept is to empower teams to work together as they need to.

How it looks in practice:

- Self-organizing teams are autonomous groups within the organization who take control and responsibility over their respective projects and have ownership of those areas. Different organizations practice this principle differently. Spotify, for example uses “product squads” to practice this.

Learn more about managing complex requirements in an agile world in the webinar below.

deploying applicationin Cloud

About Deploying Oracle Agile PLM on Cloud

If your organization wants to develop, deploy, and/or update parts of an Agile Product Lifecycle Management (PLM) application in a faster, more agile way, instead of investing in building on-premise implementations, then deploy Agile PLM on Oracle Cloud Infrastructure.

By using Agile PLM on Oracle Cloud, replication from on-premise to cloud and cloud-to-cloud platforms can easily be established and managed. You can also gain the benefits of faster infrastructure updates, easier scaling up (and down), lower capital expenditure, and fewer personnel dedicated to basic infrastructure maintenance.

Key Workload Requirements

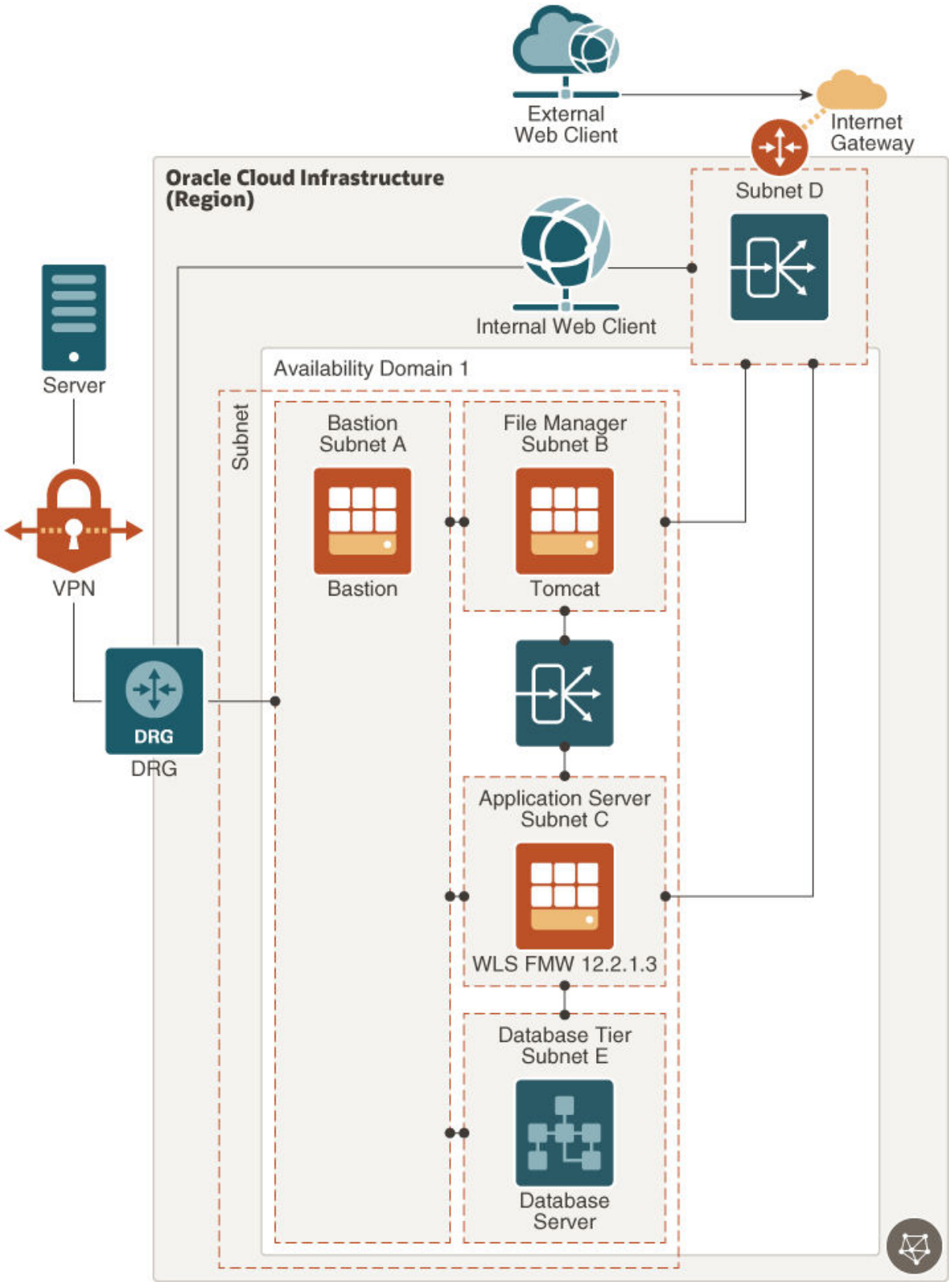
The architectures that Oracle provides help you address these requirements:

- Designing for high availability and disaster recovery
- Deploying a secure architecture.
- Matching your high-performance and highly isolated network model.
- Deploying your application and database environments into the cloud.
- Maintaining visibility over costs and usage.
- Monitoring infrastructure health and performance.

Architecture for Deploying Agile PLM on Cloud

You can deploy Agile PLM in a single availability domain while ensuring high availability. Use this architecture when you want to ensure that your application is available even when an application instance goes down. The other available application instances in the availability domain continue to process the requests.

Oracle Agile PLM can be deployed on cloud in a multi-tiered architecture. The architecture consists of a virtual cloud network (VCN) with the bastion host, load balancer tier, application tier, and database tier. The tiers are placed in separate subnets of the VCN in a single availability domain.



Description of the illustration agile_plm_reference_architecture_high_availability.png

The Agile PLM application server can be set up in a standalone or clustered configuration. In the image shown, a standalone server is considered, which has only one Oracle WebLogic Server instance. All client servers and users connect to the application server either directly

or indirectly. To permit traffic to the web server from the internet, you can create load balancers in the public subnet. You can access Oracle Cloud instances in the private subnet from your data centers by connecting through the dynamic routing gateway (DRG). The DRG is the gateway that connects your on premise network to your cloud network and you can enable communication between the two using VPN. You'll also have to update the route table to enable traffic to and from the DRG.

The load balancer receives requests from users, and then routes these requests to the application tier. You can allow for redundancy (and scalability) by configuring multiple instances of the WebLogic server for the core application, Tomcat for File Manager, and RAC for database. You can augment redundancy through the use of fault domains so that you can continue accessing the application even if an instance goes down. All instances are active and receive traffic from the load balancer.

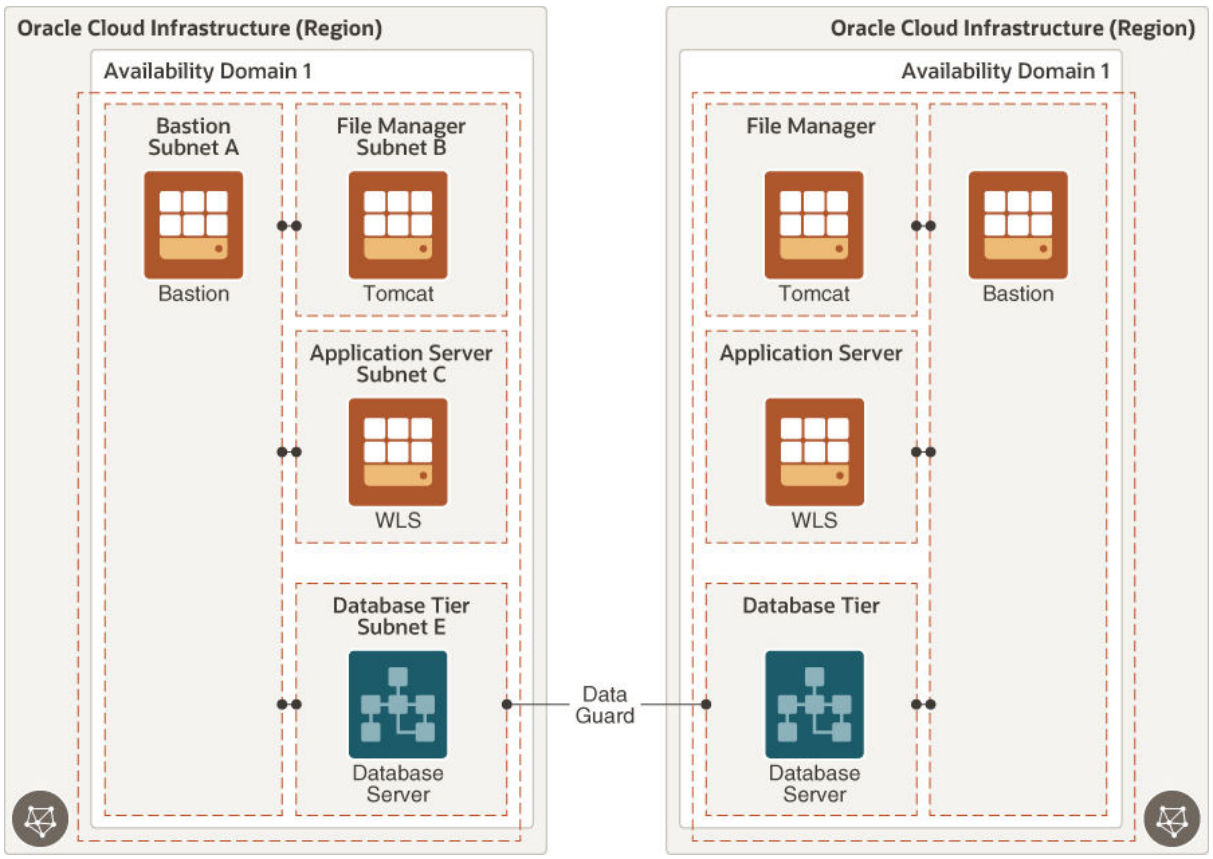
There's a private Load Balancer between File Manager and Application Server to distribute traffic to your application instances within a VCN. This service provides a primary and a standby instance of the load balancer to ensure that if the primary load balancer becomes unavailable, the standby load balancer forwards the requests. The load balancer ensures that requests are routed to the healthy application instances. If there's a problem with an application instance, then the load balancer removes that instance and starts routing requests to the remaining healthy application instances.

The database server stores all product content and system settings and is placed in the private subnet. This database is accessed only by the application server. For performance and high availability requirements, Oracle recommends that you use two-node Oracle Real Application Clusters (Oracle RAC) database systems in Oracle Cloud Infrastructure.

Architecture of Agile PLM Disaster Recovery

Oracle Cloud provides Agile PLM implementations that ensure you can build disaster recovery (DR) into your deployment in unforeseen events that would require you to failover and still keep Agile PLM up and running.

The following image illustrates the reference architecture for deploying Agile PLM in multiple regions with high availability and disaster recovery.



Description of the illustration agile_plm_reference_architecture_high_availability_and_dr.png

Oracle Data Guard protects your database tier by replicating data across availability domains.