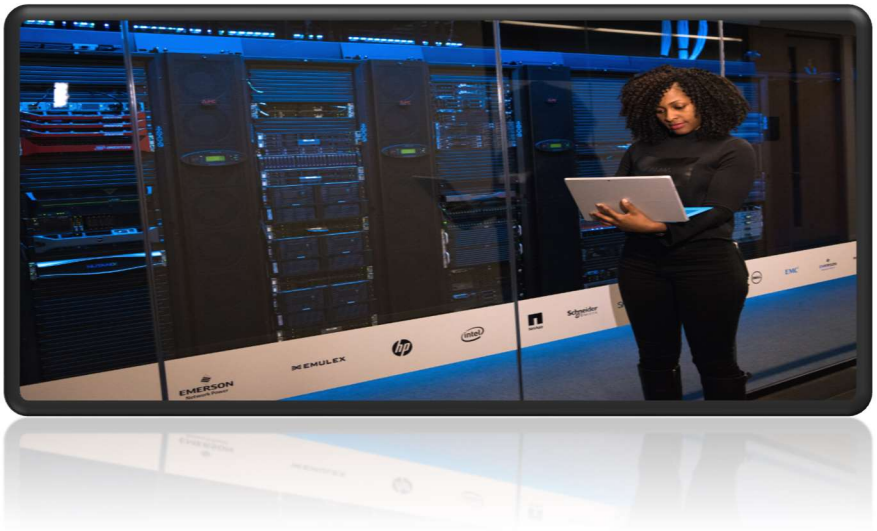
Web Development and Database Administration

Level-IV

Based on March 2022, Curriculum Version II



Module Title: Database Monitoring and Administration

Module code: EIS WDDBA4 M05 1123

Nominal duration: 40 Hour

Prepared by: Ministry of Labor and Skill

November, 2023

Addis Ababa, Ethiopia

Table of Content

Contents

Table of Content 2
Acknowledgment 4
Acronym 5
Unit one: Database startup 7
1.1. Principles of Database 8
1.2. System configuration for database startup 9
1.3. Monitoring irregularity for database startup and operation 11
Self-check 1 13
Unit Two: Database management 14
2.1. Data Dictionary Compilation and Structure Verification 15
2.2. Data Integrity Constraint Maintenance 18
2.3. Creation and design of indexes and multiple-field keys 19
2.4. Lock options for the database monitoring 23
2.5 Backup Verification and Retrieval 24
2.6. Monitoring and Resizing of Storage Resources 26
2.7. Data Update per Organizational Guidelines 28
Updating data 28
Self-check 2 31
Unit Three: Database access management 32
3.1. Access Privilege Management 33
3.2. Monitor network Server Logs 36
3.3. Manage system resources 37
Self-check 3 40
Operation sheet 3. 1 INSTALLING Active Directory 41
Operation sheet 3.2 Setting Up Domain Administrator Account 48

Operation sheet 3.3 Adding Nodes to Domain 50
LAP Test 51
References 52
Developer's Profile 53

Acknowledgment

Ministry of Labor and Skills wish to extend thanks and appreciation to the many representatives of TVET instructors and respective industry experts who donated their time and expertise to the development of this Teaching, Training and Learning Materials (TTLM).

Acronym

ACID----------Atomicity, Consistency, Isolation, Durability

ASC ---------- Ascending

CPU ---------- Central Processing Unit

DBA----------Database Administrator

DBMS----------Database Management System

DDL---------- Data Definition Language

DESC ---------- Descending

DNS ---------- Domain Name System

GUI ---------- Graphical User Interface

NAS ----------Network Attached Storage

RAID----------Redundant Array of Independent Disks

RTO ---------- Recovery Time Objective

SQL ---------- Structured Query Language

SSD ---------- Solid State Drive

SYSADMIN ---------- System Administrato

Introduction to the Module

These Module collectively provide a comprehensive coverage of fundamental aspects of database administration, ranging from the initial startup procedures to ongoing management and access control.

This module is designed to meet the industry requirement under web development and database administration occupational standard, particularly for the unit of competency: **Monitor and Administer Database**

This module covers the units:

- Database startup
- Database management
- Database access management

Learning Objective of the Module

At the end of the module the trainee will be able to:

- Understand principles of database
- Configure the system for effective database startup
- Implement data dictionary compilation and verify data structures
- Monitor network server logs for unauthorized access and security breaches
- Manage database access, including allocation or removal of access privileges
- Update data according to organizational guidelines
- Verify the storage and retrieval capability of recent database backups

Module Instruction

For effective use these modules trainees are expected to follow the following module instruction:

1. Read the information written in each unit
2. Accomplish the Self-checks at the end of each unit
3. Perform Operation Sheets which were provided at the end of units
4. Do the "LAP test" giver at the end of each unit and
5. Read the identified reference book for Examples and exercise

Unit one: Database startup

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- principles of databases
- System configuration for database startup
- Monitoring irregularity for database startup and operation

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Understand principles of database
- configure system settings necessary for database startup
- Understand and implement hardware and software requirements for the database
- Monitor Database Start-up and Operations

1.1. Principles of Database

Principles of databases encompass fundamental concepts and guidelines that govern the design, implementation, and management of databases. Here are some key principles:

- **Data Integrity:**
  - Entity Integrity: Each row in a table must have a unique identifier, usually expressed through a primary key, to ensure that each record is distinct.
  - Referential Integrity: Relationships between tables should be maintained, ensuring that foreign keys in one table correspond to primary keys in another.
- **Normalization:** Databases should be organized to reduce redundancy and dependency by breaking down tables into smaller, more manageable parts through a process called normalization.
- **Consistency:** Data in the database should be consistent and accurate. Any changes made to the database should maintain its overall integrity.
- **Atomicity, Consistency, Isolation, Durability (ACID):** ACID properties are essential for database transactions. Transactions should be Atomic (indivisible), Consistent (maintain database integrity), Isolated (independent of other transactions), and Durable (once committed, changes are permanent).
- **Data Models:** Choose an appropriate data model (e.g., relational, document-oriented, graph) based on the nature of the data and the requirements of the application.
- **Structured Query Language (SQL):** Use a standardized query language like SQL to interact with and manipulate the database. SQL provides a set of commands for data definition, data manipulation, and data control.
- **Security**: Implement security measures to protect sensitive data. This includes user authentication, authorization, and encryption.
- **Concurrency Control:** Manage multiple users accessing the database simultaneously to prevent conflicts and ensure data consistency.
- **Scalability**: Design databases to scale with the growth of data and user interactions. This involves considerations like indexing, partitioning, and clustering.

- **Backup and Recovery**: Regularly back up the database to prevent data loss in the event of system failures or other disasters. Establish procedures for database recovery.
  - **Data Independence**: Separate the logical and physical aspects of the database, allowing changes to one without affecting the other. This enhances flexibility and maintainability.

1.2. System configuration for database startup

It Involves setting up the environment and configuring various system parameters to ensure a smooth and optimized startup of the database. Here are key considerations for system configuration during the database startup process:

- **Hardware Configuration**

Evaluate and configure the hardware resources such as CPU, memory, and storage to meet the requirements of the database system. Ensure that the hardware specifications align with the anticipated workload and performance expectations.

- **Operating System Configuration**

Configure the operating system settings to support the requirements of the database. This may include adjusting parameters such as file system settings, network configurations, and system resource limits.

- **Database Software Installation:** Install the database software on the server, ensuring that the installation follows best practices and is compatible with the operating system. Configure the software with the necessary options and settings.
- **Memory Allocation:** Allocate and configure memory settings for the database system. This includes setting parameters such as buffer sizes, cache sizes, and other memory-related configurations to optimize database performance.
- **Storage Configuration**

Configure storage settings, including the location of database files, log files, and backup files. Ensure that there is adequate disk space, and consider factors like disk speed and RAID configurations for optimal performance.

- **Database Instance Configuration**

Configure the database instance with parameters specific to the database engine. These parameters may include settings related to transaction logs, data files, and temporary storage.

- **Network Configuration**

Set up network configurations to enable communication between the database server and clients. Configure network protocols, firewall settings, and ensure proper connectivity.

- **Authentication and Authorization**

Configure authentication methods and authorization settings to control access to the database. This involves setting up user accounts, roles, and permissions based on security requirements.

- **Startup Parameters**

Define startup parameters for the database system. These parameters may include options related to recovery, logging, and other critical aspects of the database startup process.

- **Error Handling and Logging**

Configure error handling mechanisms and logging settings to capture information about the startup process. This is crucial for diagnosing issues and monitoring the health of the database system.

- **Backup Configuration**

Establish backup configurations to ensure that regular backups are scheduled and performed. This includes specifying backup locations, retention policies, and verification procedures.

- **Monitoring and Alerting**

Set up monitoring tools and configure alerting mechanisms to proactively monitor the database system. This allows for the early detection of issues and prompt resolution.

- **Documentation**

Document the entire system configuration process. This documentation serves as a reference for future maintenance, upgrades, and troubleshooting.

The system configuration for database startup is a critical step in ensuring the stability, performance, and security of the database environment. Proper configuration practices contribute to a well-tuned and efficiently running database system**.**

1.3. Monitoring irregularity for database startup and operation

Monitoring irregularities during database startup and operation is crucial for ensuring the stability, performance, and security of a database system. Here are key aspects to consider in this context:

- **Log Monitoring:** Regularly check database logs for any error messages, warnings, or abnormal events during startup and ongoing operation. Logs provide valuable information about the system's health.
- **Performance Monitoring:** Utilize performance monitoring tools to track database performance metrics. This includes monitoring CPU usage, memory utilization, disk I/O, and query response times. Deviations from normal patterns can indicate potential issues.
- **Alerts and Notifications:** Implement alerting mechanisms to notify administrators of any irregularities. Set up alerts for critical events such as system failures, performance bottlenecks, or security breaches.
- **Database Health Checks:** Conduct regular health checks to assess the overall state of the database. This involves reviewing system parameters, configurations, and resource utilization to identify any abnormalities.
- **Startup Procedures:** Establish and document clear procedures for starting up the database system. Regularly review the startup logs to ensure that the database initializes without errors.

Automated Monitoring Scripts: Develop and implement automated scripts to monitor key database parameters. These scripts can perform periodic checks and report any deviations from predefined thresholds.

- **Security Audits:** Conduct regular security audits to identify and address any vulnerabilities or unauthorized access attempts. Monitor login attempts, privilege changes, and data access patterns.

- **Backup Verification:** Regularly verify the integrity of database backups to ensure they can be successfully restored. This helps in preparing for potential disasters or data corruption issues.
- **Resource Utilization:** Monitor resource utilization such as CPU, memory, and disk space to ensure that the database has sufficient resources for normal operation. Plan for scalability if resource demands increase.
- **Query Performance:** Keep an eye on the performance of frequently executed queries. Identify and optimize poorly performing queries to prevent degradation of overall system performance.
- **User Activity Monitoring:** Track user activity, especially during peak usage periods. Unusual spikes in activity or unauthorized access attempts should be investigated promptly.
- **Data Consistency Checks**

  Implement routines to check the consistency of data stored in the database. Detect and resolve any discrepancies or anomalies that may arise during normal operation.

  By consistently monitoring these aspects, database administrators can proactively identify and address irregularities, minimizing the risk of downtime, data loss, or security breaches. Regular reviews and adjustments to monitoring strategies are essential to adapt to changing usage patterns and evolving system requirements.

Self-check 1

Part-I multiple choice

1. What does monitoring database logs during startup help identify?

   A. CPU utilization B. Security vulnerabilities

   C. Database growth rate D. Memory allocation

2. Why is tracking connection attempts important during database operation?

   A. To increase database size B. To identify potential security threats

   C. To reduce CPU utilization D. To improve query performance

3. Why is monitoring irregularities during database startup and operation important?

   A. To increase hardware costs B. To make the system more complicated

   C. To ensure the stability, performance, and security of the database D. To discourage users from accessing the database

Part-II Give short Answer

1. Explain the concept of "Entity Integrity" in the principles of databases.
2. Why is normalization considered an important principle in database design?
3. What are the ACID properties, and why are they essential for database transactions?
4. How does the choice of a data model impact the design of a database?

Unit Two: Database management

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- Data Dictionary Compilation and Structure Verification
- Data Integrity Constraint Maintenance
- Creation and design of indexes and multiple-field keys
- Lock options for the database monitoring
- Backup Verification and Retrieval
- Continuous Monitoring and Resizing of Data Storage
- Data Update per Organizational Guidelines

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- compile a comprehensive data dictionary
- Understand data relationships and dependencies
- Understand the importance of data integrity for overall system reliability
- create and manage multiple-field keys
- troubleshoot issues related to database locks
- retrieve and restore data from backups
- monitor data storage usage trends over time
- Adherence to organizational guidelines and policies during data updates

2.1. Data Dictionary Compilation and Structure Verification

Data Dictionary Compilation and Structure Verification involves creating and maintaining a data dictionary and ensuring that the structure of the database aligns with this dictionary.

2.1.1. Data Dictionary Compilation

A data dictionary is a centralized repository that provides metadata about the data within a database. It typically includes details such as data definitions, data types, relationships between tables, constraints, and other essential information.

A data dictionary is a collection of descriptions of the data objects or items in a data model for the benefit of programmers and others who need to refer to them.

It is a set of information describing the contents, format, and structure of a database and the relationship between its elements, used to control access to and manipulation of the database.

When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms.

Most DBMS keep the data dictionary hidden from users to prevent them from accidentally destroying its contents.

- A data dictionary may contains:
  - The definitions of all schema objects in the database.
  - How much space has been allocated for, and is currently used by the schema objects
  - Default values for columns.
  - Integrity constraint information (Constraints that apply to each field, if any)
  - Auditing information, such as who has accessed or updated various schema objects
  - Privileges and roles each user has been granted (Access Authorization)
  - Description of database users, their responsibilities and their access rights.

  Data dictionaries do not contain any actual values from the database, only bookkeeping information for managing it.
- Advantage of a Data Dictionary

When a new user is introduced to the system or a new administrator takes over the system, identifying table structures and types becomes simpler.

Example:

- In a data dictionary, you might document that the "Customers" table includes fields such as "CustomerID," "Name," "Email," and "Phone." For each field, you specify the data type, length, and any constraints.

2.1.2. Structure Verification

Verify that the actual structure of the database matches the information documented in the data dictionary. This ensures that the database schema aligns with the intended design and that any changes to the database structure are accurately reflected in the data dictionary.

Example:

If the data dictionary indicates that the "Orders" table should have a foreign key relationship with the "Customers" table, verify that this relationship exists in the database schema. Check that the data types, constraints, and relationships match the documented specifications.

- **Consistency Checks:** Conduct consistency checks to ensure that the data dictionary is consistent with other project documentation and requirements. This involves verifying that changes made to the database structure are appropriately updated in the data dictionary.

Example:

If there's a change in a table structure, such as adding a new field, ensure that the data dictionary is updated to reflect this change. Consistency checks prevent discrepancies between documentation and the actual database.

- **Data Type Verification:** Check that the data types assigned to each field in the database match the specifications in the data dictionary. This includes verifying numeric precision, string lengths, and other data type attributes.

Example

If the data dictionary specifies that the "Price" field should be a numeric data type with two decimal places, verify that this is accurately implemented in the database.

- **Constraint Verification:** Verify that constraints, such as primary keys, foreign keys, unique constraints, and check constraints, are implemented correctly in the database according to the data dictionary.

  Example

If the data dictionary specifies that the "ProductID" field is the primary key for the "Products" table, verify that this constraint is enforced in the database schema.

- **Documentation Updates:** If any discrepancies or changes are identified during the verification process, update the data dictionary and any related documentation accordingly. Keep the documentation synchronized with the actual database structure.

  Example

If a new index is created on a table for performance reasons, update the data dictionary to include information about this index.

- **Version Control:** Implement version control for the data dictionary to track changes over time. This ensures that you can trace modifications, additions, or deletions to the data dictionary and understand the evolution of the database structure.

  Example:

Use version control tools to track changes to the data dictionary, providing a history of alterations to the database structure.

By compiling a comprehensive data dictionary and regularly verifying the database structure against it, organizations can maintain consistency, accuracy, and documentation integrity, which is crucial for effective database management and development.

2.2. Data Integrity Constraint Maintenance

Data integrity is a constraint which used to ensure accuracy and consistency of data in a database by validating the data before getting stored in the columns of the table.

Data integrity refers to the overall completeness, accuracy and consistency of data in according to business requirements.

Data integrity constraint maintenance is a critical aspect of managing a database. Data integrity constraints define the rules that must be adhered to when inserting, updating, or deleting data in a database. Here's an overview of key considerations for maintaining data integrity constraints:

- **Types of integrity constraints**
    1. Entity integrity
    2. Referential integrity
    3. Domain integrity
    4. User defined integrity

1. Entity integrity

This is concerned with the concept of primary keys. The rule states that every table must have its own primary key and that each has to be unique and not null.

2. Referential Integrity

This is the concept of foreign keys. The rule states that the foreign key value can be in two states. The first state is that the foreign key value would refer to a primary key value of another table, or it can be null. Being null could simply mean that there are no relationships, or that the relationship is unknown.

Referential integrity is a feature provided by relational DBMS that prevents users from entering inconsistent data.

3. Domain Integrity

This states that all columns in a relational database are in a defined domain.

The concept of data integrity ensures that all data in a database can be traced and connected to other data. This ensures that everything is recoverable and searchable. Having a single, well defined and well controlled data integrity system increases stability, performance, reusability and maintainability.

4. User Defined Integrity

User-defined integrity allows you to define specific business rules that do not fall into one of the other integrity categories. All of the integrity categories support user-defined integrity (all column- and table-level constraints in CREATE TABLE, stored procedures, and triggers).

Business rules may dictate/state that when a specific action occurs further actions should be triggered.

For example, deletion of a record automatically writes that record to an audit table.

2.3. Creation and design of indexes and multiple-field keys

The creation and design of indexes, as well as the use of multiple-field keys (composite keys), are crucial aspects of database optimization.

2.3.1. Creation and Design of Indexes:

- **What is index**

An index is a separate physical data structure that enables queries to access one or more data rows fast.

A database index is a separate physical data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and the use of more storage space to maintain the extra copy of data.

Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.

- **Why Use Indexes**?

Two primary reasons exist for creating indexes in SQL Server:

- To maintain uniqueness of the indexed column(s)
- To provide fast access to the data in tables.

- **Deciding which fields to be index**

The following list gives guidelines in choosing columns to index:

- You should create indexes on columns that are used frequently in WHERE clauses.
- You should create indexes on columns that are used frequently to join tables.
- You should create indexes on columns that are used frequently in ORDER BY clauses.
- You should create indexes on columns that have few of the same values or unique values in the table.
- You should not create indexes on small tables (tables that use only a few blocks) because a full table scan may be faster than an indexed query.
- If possible, choose a primary key that orders the rows in the most appropriate order.

- **Creating indexes**

Indexes can be created to order the values in a column in ascending or descending sequence.

- You can use the CREATE INDEX statement to create indexes.

The general form of CREATE INDEX statement is:

CREATE INDEX index_name ON table_name (column1 [ASC | DESC] ,...)

- **Delete an index**

Deleting an index means removing one or more relational indexes from the current database.

The DROP INDEX statement is used to delete an index in a table.

Syntax: DROP INDEX index_name ON table_name

To delete an index by using Object Explorer, you can follow the steps as shown below:

- In Object Explorer, expand the database that contains the table on which you want to delete an index.
- Expand the Tables folder.
- Expand the table that contains the index you want to delete.
- Expand the Indexes folder.

- Right-click the index you want to delete and select Delete.
- In the Delete Object dialog box, verify that the correct index is in the Object to be deleted grid and click OK.

To delete an index using Table Designer

- In Object Explorer, expand the database that contains the table on which you want to delete an index.
- Expand the Tables folder.
- Right-click the table that contains the index you want to delete and click Design.
- On the Table Designer menu, click Indexes/Keys.
- In the Indexes/Keys dialog box, select the index you want to delete.
- Click Delete.
- Click Close.
- On the File menu, select save table_name.

- **View and edit indexes**

To view all indexes in a database

- In Object Explorer, connect to an instance of the SQL Server Database Engine and then expand that instance.
- Expand Databases, expand the database that contains the table with the specified index, and then expand Tables.
- Expand the table in which the index belongs and then expand Indexes.

To modify an index using wizard

- In Object Explorer, connect to an instance of the SQL Server Database Engine and then expand that instance.

- Expand Databases, expand the database in which the table belongs, and then expand Tables.
- Expand the table in which the index belongs and then expand Indexes.
- Right-click the index that you want to modify and then click Properties.

In the Index Properties dialog box, make the desired changes. For example, you can add or remove a column from the index key, or change the setting of an index option.

2.3.2. Multiple-Field Keys (Composite Keys):

A multiple-field key, or composite key, involves using multiple columns as a unique identifier for a record.

- **Consideration for Composite Keys:** Use composite keys when a combination of multiple columns uniquely identifies a record, and this combination is more meaningful than any individual column.

Performance Implications: Be aware that using composite keys can impact query performance, and the choice of fields in the composite key should be based on the typical search patterns.

Data Integrity: Ensure that the combination of fields in a composite key maintains data integrity. This involves considering relationships between the fields and the overall business logic.

Primary Key and Composite Key: When choosing a primary key, evaluate whether a single-field primary key is sufficient or if a composite key is more appropriate based on the uniqueness constraints.

Indexing Composite Keys: Index composite keys to improve performance when querying based on the combination of fields.

Documentation: Document the rationale behind the choice of composite keys and regularly review their effectiveness in meeting database performance goals.

Effective index creation and the use of composite keys require a balance between optimizing read performance and considering the impact on write performance. Regular monitoring and adjustments based on evolving data patterns and usage scenarios are essential for maintaining an efficient database structure.

2.4. Lock options for the database monitoring

Lock options play a crucial role in managing concurrency and ensuring data consistency in a database. Database monitoring involves keeping track of these locks to identify potential issues and optimize performance.

Types of Locks:

- **Shared Locks:** Used for read operations, allowing multiple transactions to read a resource simultaneously but preventing any of them from writing to it.
- **Exclusive Locks:** Used for write operations, ensuring that only one transaction can modify a resource at a time.
- **Read Locks:** Similar to shared locks, allowing multiple transactions to read a resource simultaneously.
- **Write Locks:** Similar to exclusive locks, preventing other transactions from reading or writing to a resource.
- **Monitoring Locks:** Implement mechanisms to monitor the current state of locks in the database. This includes tracking which transactions hold locks and which are waiting for them.

Deadlock Detection:

- Set up deadlock detection mechanisms to identify situations where transactions are waiting indefinitely for each other to release locks. Deadlocks can lead to a standstill in processing and need to be resolved.

Lock Timeout: Configure lock timeouts to automatically release locks held by a transaction if they are not released within a specified time frame. This helps prevent long-running transactions from causing bottlenecks.

Isolation Levels: Understand and configure different isolation levels (e.g., READ COMMITTED, REPEATABLE READ, SERIALIZABLE) based on the application's requirements. Higher isolation levels generally involve more restrictive locking, which can impact performance.

Row-Level Locking: Consider using row-level locking instead of table-level locking when possible. Row-level locking allows for more granular control and reduces contention for resources.

Lock Escalation: Monitor and understand lock escalation mechanisms in the database system. Lock escalation occurs when a lower-level lock (e.g., row-level) is escalated to a higher-level lock (e.g., table-level) to manage resources more efficiently.

Lock Statistics: Regularly review lock statistics and performance metrics to identify patterns of contention. This information helps in making informed decisions about indexing, query optimization, and application design.

Database Management System (DBMS) Tools: Utilize built-in tools provided by the DBMS for monitoring locks and transactions. These tools often provide insights into lock wait times, deadlocks, and other relevant metrics.

Concurrency Control Mechanisms: Understand the concurrency control mechanisms implemented by the DBMS, such as optimistic or pessimistic concurrency control. Choose the appropriate mechanism based on the application's requirements.

Alerts and Notifications: Implement alerts and notifications for unusual lock-related activity. This includes notifications for prolonged lock wait times, frequent deadlocks, or other anomalies.

Documentation: Document lock configurations, monitoring strategies, and any tuning adjustments made to optimize database performance. This documentation helps in troubleshooting and future optimizations.

Effective monitoring of locks is essential for maintaining a balance between transaction concurrency and data consistency. Regular analysis of lock-related metrics enables administrators to identify and address issues proactively, ensuring optimal database performance.

2.5 Backup Verification and Retrieval

Backup verification and retrieval are critical components of a robust data management strategy. Ensuring that backups are created successfully, verifying their integrity, and having a reliable process for retrieval are essential for data protection and disaster recovery.

Backup Verification: Regularly verify the integrity of database backups to ensure that they are free from corruption and can be relied upon for recovery. Verification may involve checking

backup files, validating backup processes, and confirming that the backup captures the entire database.

Verifying that recent backups of a database have been stored successfully and can be retrieved as a full working copy is a critical aspect of database management. Here's a step-by-step guide on how you might confirm this:

Identify Backup Location: Determine where the recent backups are stored. This could be on local servers, network-attached storage (NAS), or cloud storage.

Check Backup Logs: Review backup logs to confirm that recent backup operations were successful. Examine any error messages or warnings that may indicate issues.

Verify Timestamps: Check the timestamps on the backup files to ensure they correspond to recent backup operations. This helps confirm that the backup files are up-to-date.

Perform a Trial Restoration: Conduct a trial restoration of the database from the backup to ensure that the process works as expected. This involves restoring the database to a test environment without affecting the production system.

Test the Restored Database: After the restoration, perform tests on the restored database to ensure that it is fully functional. Run sample queries, check data integrity, and validate that all necessary components are in place.

Verify File Integrity: Check the integrity of backup files to ensure they are not corrupted. You can use checksums or hash functions to verify the integrity of the backup files.

Check Backup Retention Policy: Confirm that the backup retention policy aligns with the organization's requirements. Ensure that backups are retained for an appropriate duration and that old backups are regularly pruned.

Ensure Accessibility: Verify that the individuals responsible for database recovery have access to the backup files and the necessary credentials to restore the database.

Test Recovery Scenarios: Simulate various disaster recovery scenarios (e.g., hardware failure, data corruption) and ensure that the backup and recovery processes can effectively address these situations.

Documentation Review: Refer to documentation related to backup and recovery procedures to confirm that the documented steps align with the actual processes followed.

Automation Verification: If backups are automated, ensure that the backup automation scripts or tools are running as scheduled and producing the expected results.

Coordinate with IT Operations: Communicate with IT operations or relevant teams to confirm that the backup storage infrastructure is operational and that there are no issues with storage devices or cloud services.

Notification Systems: Ensure that notification systems are in place to alert relevant personnel in case of backup failures or issues.

Regular Backup Schedule: Establish a regular schedule for backups based on the organization's data retention policies and recovery objectives. This could include daily, weekly, or incremental backups.

Automated Backup Verification: Implement automated processes to verify the success of backup operations. This involves checking that backups are completed without errors and are consistent t

2.6. Monitoring and Resizing of Storage Resources

Monitoring data storage space and resizing as needed is crucial for ensuring the ongoing viability and performance of a database system. Here are steps you can take to effectively monitor and manage data storage space:

Set up Monitoring Alerts: Implement monitoring alerts to notify administrators when storage space reaches predefined thresholds. This ensures timely intervention before critical issues arise.

Regularly Check Disk Space Usage: Periodically review the current disk space usage on both database and server levels. This can be done using system commands or database management tools.

Use Database Management Tools: Utilize the features provided by your database management system (DBMS) to monitor storage space. Many DBMSs offer built-in tools for space utilization analysis.

Monitor File Growth: Keep an eye on the growth rate of database files, including data files, log files, and any other file groups. Identify trends and anticipate future storage needs.

Scheduled Reports: Set up scheduled reports or scripts to automatically generate summaries of storage space usage. These reports can provide insights into historical trends and forecast future requirements.

Implement Auto-Growth: Configure auto-growth settings for database files to allow for automatic expansion when needed. However, monitor auto-growth events closely, as excessive auto-growth can impact performance.

Pre-allocate Space: Consider pre-allocating space for database files based on growth projections. This can help avoid frequent auto-growth events and potential performance bottlenecks.

Regularly Resize Database Files: Proactively resize database files when necessary. If you notice that a particular file group is consistently running out of space, resize the associated files accordingly.

Archive or Purge Data: Evaluate and implement data archiving or purging strategies to remove obsolete or historical data. This not only frees up space but also improves database performance.

Partitioning: If applicable, explore partitioning strategies to manage large datasets more efficiently. Partitioning can enhance both query performance and data management.

Evaluate Indexing: Review and optimize database indexes. Poorly designed or fragmented indexes can contribute to excessive storage usage.

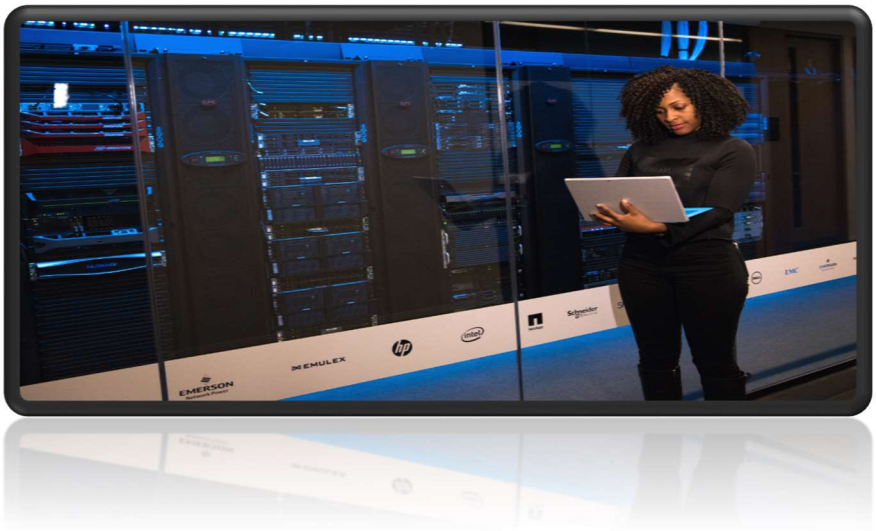


Fig 2.1 Monitoring Storage Resources

2.7. Data Update per Organizational Guidelines

Updating data

Change Management Process: Implement a formal change management process that outlines the steps and approvals required for making updates to the database. This process ensures that changes are well-documented and aligned with organizational goals.

Authorization and Access Controls: Define roles and permissions to control who has the authority to update data. Ensure that only authorized personnel have the necessary access rights, and implement a principle of least privilege to minimize the risk of unauthorized changes.

Data Validation and Verification: Establish procedures for validating and verifying data updates before they are applied. This involves checking the accuracy and completeness of the data to be modified.

Use of Transactions: Encourage the use of database transactions to ensure that updates are atomic, consistent, isolated, and durable (ACID). This minimizes the risk of incomplete or erroneous updates.

Version Control: Implement version control mechanisms to track changes to the database over time. This includes capturing information about who made the change, when it was made, and the nature of the change.

Audit Trails: Enable audit trails to log changes made to the database. Audit logs help in tracking modifications, identifying potential issues, and providing accountability.

Testing in a Staging Environment: Prior to making updates in the production environment, conduct testing in a staging environment to validate the impact of changes on data integrity and application functionality.

Scheduled Maintenance Windows: Coordinate data updates during scheduled maintenance windows to minimize disruption to users. Communicate maintenance schedules in advance to relevant stakeholders.

Communication Protocols: Establish communication protocols to inform relevant teams and stakeholders about planned data updates. This includes notifying users of potential downtime or changes to data that may impact their workflows.

Rollback Plans: Develop rollback plans in case issues arise during or after data updates. Having a plan for reverting to the previous state ensures quick recovery in the event of unexpected problems.

Documentation: Document all data update processes, including the rationale for the update, the SQL statements or procedures used, and any issues encountered during the process. This documentation serves as a reference for future updates and audits.

User Training and Awareness: Provide training to users and stakeholders on the data update processes and guidelines. Increasing awareness helps prevent unintentional or unauthorized updates and promotes a culture of responsible data management.

Collaboration with Application Teams: Collaborate closely with application development teams to ensure that data updates align with application requirements and do not negatively impact system functionality.

Compliance with Regulations: Ensure that data updates comply with relevant regulations and industry standards. This is particularly important in industries with specific data governance and compliance requirements.

Regular reviews and updates to organizational guidelines contribute to a robust and reliable data management framework.

Self-check 2

Part-I multiple Choice

1. What is the primary purpose of a database index?
   A) To reduce storage space usage B) to enforce data integrity constraints
   C) To maintain uniqueness of the indexed column(s) D) To create a backup of the database

2. Which type of lock is used for read operations, allowing multiple transactions to read a resource simultaneously?

A) Exclusive Lock B) Write Lock C) Shared Lock D) Read Lock

3. What is a data dictionary in the context of database management?

A) A dictionary containing words and their meanings

B) A collection of descriptions of data objects in a database

C) A tool for spell-checking in databases

D) A data backup file

Part-II Give short answer

1. Why is it crucial to regularly verify the integrity of database backups?
2. Briefly explain the importance of backup verification and retrieval in data management.
3. What are some key steps in monitoring and resizing storage resources in a database system?

Unit Three: Database access management

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- Access Privilege Management
- Monitor network Server Logs
- Manage system resources

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Understand the process of granting and revoking access privileges
- allocate appropriate access levels based on user roles and responsibilities
- analyze server logs to identify and understand various types of network activities
- Identify and classify security threats and incidents

monitor and assess system resources, including CPU, memory, disk space, and network usage within a database environment

3.1. Access Privilege Management

3.1.1 Allocate or remove access privileges according to user

Status

Allocate or remove access privileges according to user status" involves managing user access based on changes in their status, such as new user onboarding, role changes, or when a user leaves the organization. Here are examples for this aspect of Access Privilege Management:

- **Allocate Access Privileges for New Users:**

Definition: Granting appropriate access privileges to new users based on their roles and responsibilities.

Example:

-- Granting basic read-only access to a new user

GRANT SELECT ON database.table TO 'new_user'@'localhost';

-- Granting additional privileges based on the user's role

GRANT INSERT, UPDATE, DELETE ON database.table2 TO 'new_user'@'localhost';

Remove Access Privileges for Departing Users:

Definition: Revoking access privileges for users who have left the organization or no longer require access.

Example:

-- Revoking all privileges for a departing user

REVOKE ALL PRIVILEGES ON database.* FROM 'departing_user'@'localhost';

-- Optionally, drop the user account

DROP USER 'departing_user'@'localhost';

Adjust Access Privileges for Role Changes:

Modifying access privileges when a user's role or responsibilities change within the organization.

Example:

```
-- Adjusting access privileges for a user with a new role

REVOKE SELECT ON database.table FROM 'user_with_previous_role'@'localhost';

GRANT UPDATE, INSERT ON database.table TO 'user_with_new_role'@'localhost';
```

Periodic Access Review and Adjustment:

Definition: Conducting regular reviews of user access privileges and adjusting them based on changes in job roles or responsibilities.

Example:

```
-- Identifying and revoking unnecessary privileges during a periodic review

REVOKE DELETE, UPDATE ON database.table3 FROM 'user_to_review'@'localhost';
```

Access Privileges Based on User Status:

Definition: Implementing conditional access privileges based on the user's status (e.g., active, inactive, probationary).

Example:

```
-- Granting conditional access based on user status

IF user_status = 'active' THEN

   GRANT SELECT ON database.table4 TO 'active_user'@'localhost';

ELSE

   REVOKE ALL PRIVILEGES ON database.table4 FROM 'inactive_user'@'localhost';

END IF;
```

User Access Termination: Ensuring that access privileges are promptly terminated when a user leaves the organization.

Example:

-- Terminating access for a user who has left the organization

REVOKE ALL PRIVILEGES ON *.* FROM 'former_user'@'localhost';

DROP USER 'former_user'@'localhost';

Access Privileges for Temporary Roles:

Definition: Granting temporary access privileges for users in specific roles or projects.

Example:

-- Granting temporary access for a specific project

GRANT SELECT, INSERT, UPDATE ON project_database.* TO 'temporary_user'@'localhost';

Access Based on User Approval:

Definition: Requiring approval for granting or modifying access privileges.

Example:

-- Granting access after approval

GRANT SELECT, INSERT, UPDATE ON database.table TO 'approved_user'@'localhost';

Access Audit and Logging:

Definition: Logging and auditing access changes to maintain a record of who has been granted or revoked access privileges.

Example:

-- Logging access changes

```
-- This could involve triggers or database audit features

INSERT INTO access_log (timestamp, user, action, database_object)

VALUES (CURRENT_TIMESTAMP, 'admin', 'GRANT', 'database.table5');
```

3.2. Monitor network Server Logs

Monitoring the network server log-in log file is crucial for identifying and responding to illegal log-in attempts and potential security breaches. This involves actively reviewing log files to detect patterns indicative of unauthorized access. Here's a procedure to implement this monitoring:

Access Log-in Log Files: Regularly access and review log-in log files on the network server. These files contain records of user log-ins, including successful and unsuccessful attempts.

Focus on Security-Relevant Logs: Concentrate on logs that are relevant to security, such as authentication logs and logs indicating user log-in activities.

Identify Suspicious Patterns: Look for suspicious patterns in log-in activities, including multiple failed log-in attempts, log-ins from unusual locations or IP addresses, or log-ins during non-business hours.

Automate Log Analysis: Implement automated log analysis tools to assist in the identification of potential security breaches. These tools can quickly analyze large volumes of log data and generate alerts for anomalies.

Set Thresholds for Alerts: Define thresholds for log-in activities that trigger alerts. For example, multiple failed log-in attempts within a short period or log-ins from geographically improbable locations.

Correlate with Other Logs: Correlate log-in log data with other logs, such as intrusion detection system (IDS) logs or firewall logs, to gain a comprehensive understanding of network security.

Real-time Monitoring: Implement real-time monitoring to receive immediate alerts for suspicious log-in activities. Real-time monitoring enhances the ability to respond promptly to security incidents.

Implement Geolocation Analysis: Utilize geolocation analysis to identify log-ins from locations inconsistent with normal user behavior. This helps detect potential unauthorized access.

Check for Unusual Log-in Times: Investigate log-ins that occur during unusual hours or outside of normal business hours. This can be an indicator of unauthorized access.

Review Failed Log-ins: Pay close attention to failed log-in attempts. Excessive failed attempts may suggest a brute-force attack or an attempt to gain unauthorized access.

Track User Accounts: Monitor log-ins for privileged user accounts closely. Unauthorized access to accounts with elevated privileges poses a significant security risk.

Regular Security Training: Conduct regular security training for users to raise awareness about the importance of secure log-in practices and to recognize and report suspicious activities.

Incident Response Plan: Have an incident response plan in place to guide actions in the event of a detected security breach. This plan should include steps for isolating affected systems, notifying relevant parties, and conducting forensic analysis.

Continuous Improvement: Continuously refine and improve log-in monitoring based on evolving security threats and the organization's specific requirements.

3.3. Manage system resources

Definition: Managing system resources is essential for ensuring optimal performance, reliability, and efficiency of a database system. It involves monitoring, allocating, and optimizing resources to meet the demands of the database and associated applications. Here's a procedure for managing system resources effectively:

Procedure:

Resource Monitoring: Utilize system monitoring tools to continuously track resource usage, including CPU utilization, memory consumption, disk I/O, and network activity.

Set Resource Thresholds: Define thresholds for resource usage that, when exceeded, trigger alerts. These thresholds can be customized based on the specific requirements and performance expectations.

Automated Alerts: Implement automated alerting systems to receive immediate notifications when resource thresholds are breached. Alerts facilitate proactive response to potential performance issues.

Capacity Planning: Conduct regular capacity planning assessments to predict future resource needs. This involves analyzing historical resource usage patterns and forecasting growth.

Scale Resources Appropriately: Based on capacity planning assessments, scale resources (CPU, memory, storage) as needed. This may involve upgrading hardware, adding more servers, or leveraging cloud-based scaling solutions.

Prioritize Critical Processes: Identify and prioritize critical database processes and allocate resources accordingly. Ensure that essential operations receive the necessary computing power and memory.

Database Indexing and Optimization: Optimize database indexes and queries to minimize resource-intensive operations. Well-optimized queries contribute to reduced resource consumption.

Regular Performance Tuning: Conduct regular performance tuning activities, such as query optimization and index maintenance, to enhance database efficiency and reduce resource utilization.

Implement Caching Mechanisms: Introduce caching mechanisms to reduce the need for repetitive database queries, thereby decreasing the load on the database and improving response times.

Database Connection Management: Implement connection pooling and efficient connection management to avoid resource exhaustion caused by a large number of concurrent connections.

Disk Space Management: Monitor and manage disk space regularly. Implement practices such as archiving, purging, or compressing data to prevent unnecessary storage consumption.

Network Bandwidth Optimization: Optimize network bandwidth usage by minimizing unnecessary data transfers and ensuring efficient communication between database servers and clients.

Backup and Restore Optimization: Optimize backup and restore processes to minimize their impact on system resources. Consider incremental backups and ensure that backup schedules do not coincide with peak usage times.

Virtualization Management: If using virtualization, manage virtual machine (VM) resources effectively. Adjust VM configurations based on workload requirements and allocate resources appropriately.

Implement Load Balancing: If applicable, implement load balancing to distribute incoming traffic across multiple servers. This helps prevent resource bottlenecks on individual servers.

Regular System Updates: Keep the operating system, database software, and relevant components up to date with the latest patches and updates to benefit from performance.

Self-check 3

Part-I Multiple choice

1. What is the purpose of granting basic read-only access to a new user in the context of access privilege management?
   A. To allow the user to modify database records
   B. To provide full administrative access to the database
   C. To restrict the user from accessing the database
   D. To enable the user to view but not modify data
2. What is the purpose of adjusting access privileges for a user with a new role in access privilege management?

A. To increase the user's access privileges B. To simplify the access control process

C. To align privileges with the user's new responsibilities D. To grant access to all database tables

3. What is the primary purpose of access privilege management in a database system?
   A. To increase system complexity B. To grant unlimited access to all users
   C. To manage and control user access to database resources
   D. To ignore user access altogether
4. Why is monitoring network server logs crucial for database security?
   A. To increase internet speed
   B. To identify and respond to illegal log-in attempts and potential security breaches
   C. To track social media usage
   D. To measure the server's physical temperature

Part-II Give short answer

1. What is the purpose of access privilege management in a database system?
2. Explain the importance of monitoring network server logs for illegal log-in attempts or security breaches.

3. Explain the role of log files in SQL Server and why monitoring them is essential for system administrators.

OPERATION SHEET 3. 1 INSTALLING ACTIVE DIRECTORY

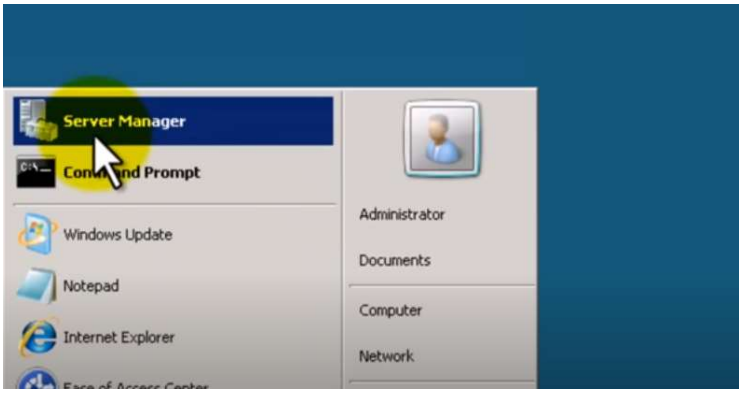
Operation title: Active directory installation

Tool and equipment: computer and windows server 2012

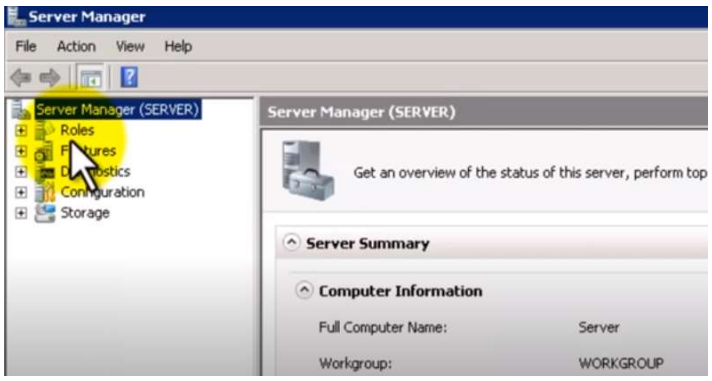
Steps

1. First assign an ip address to the computer

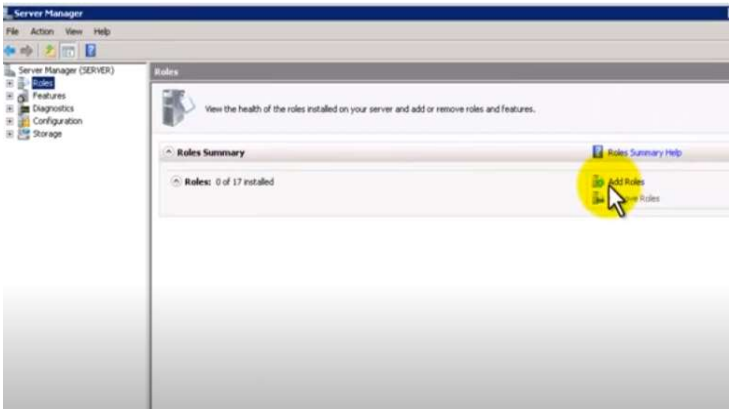
2. Start →server manager



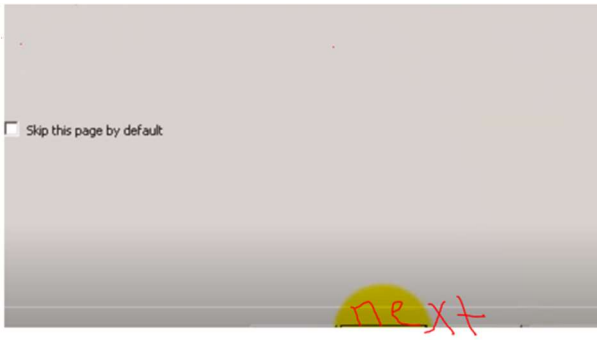
3. Click Role



4. Click Add Role



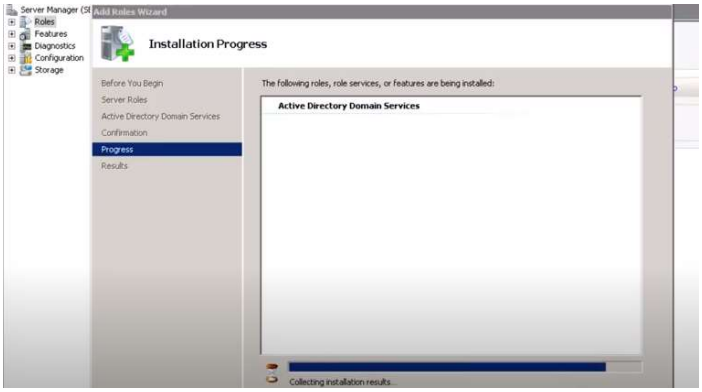
5. Click next



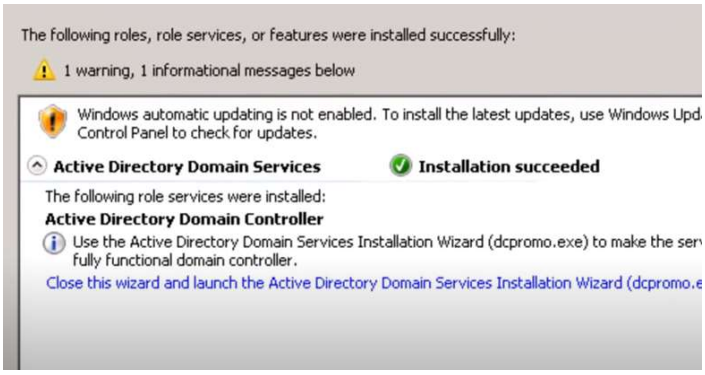
6. Click active directory Domain service



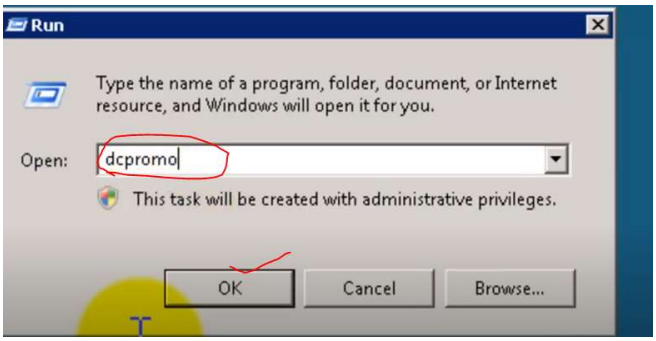
6. Click next and install



7.

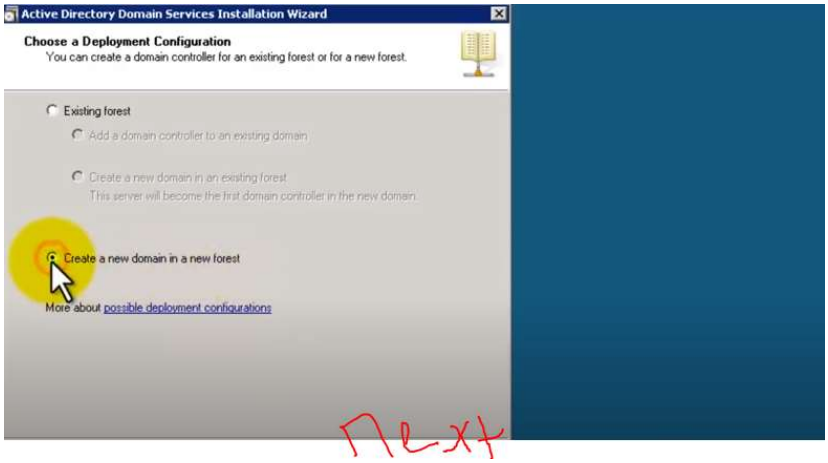


8. Click start→ RUN
9. Type DCPROMO

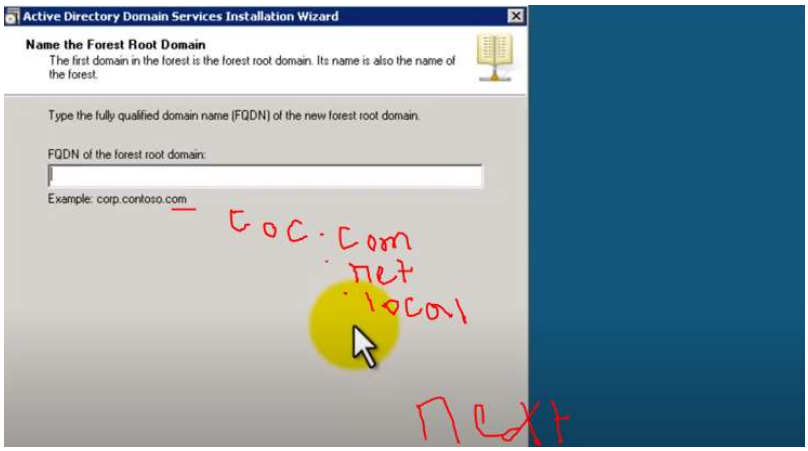


10. Click next

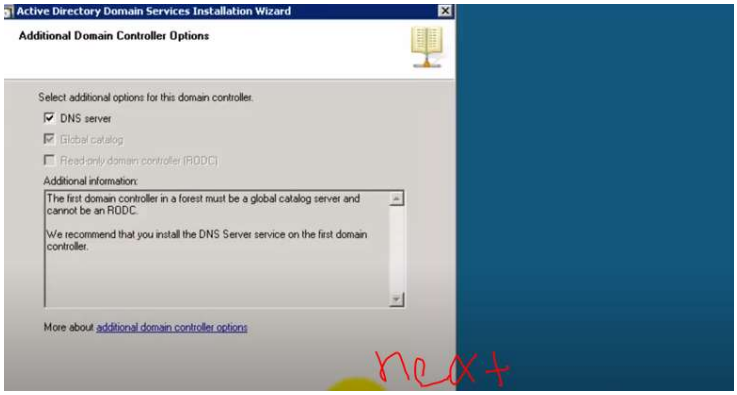
11. Click create an ew domain in a new forest then next



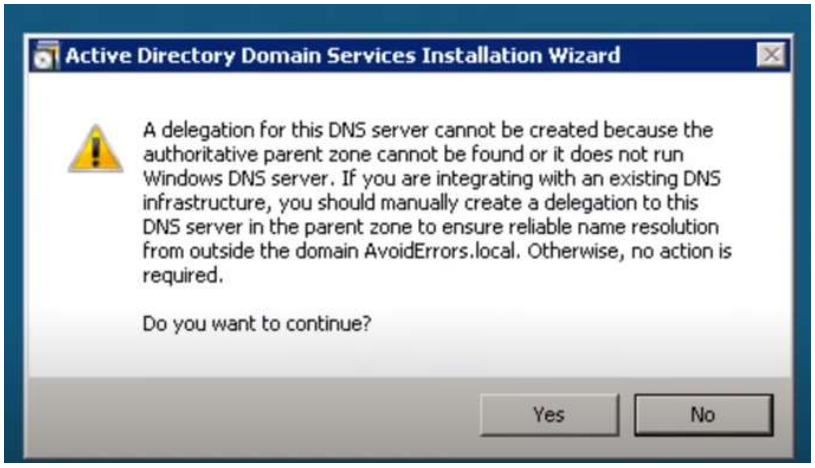
12. Type the domain name then next



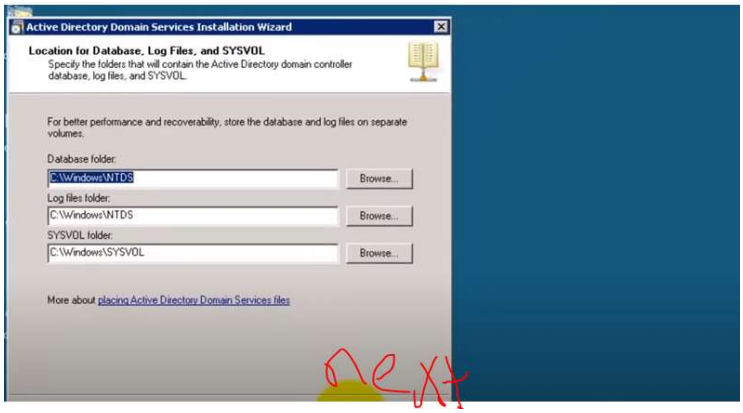
13. next→next



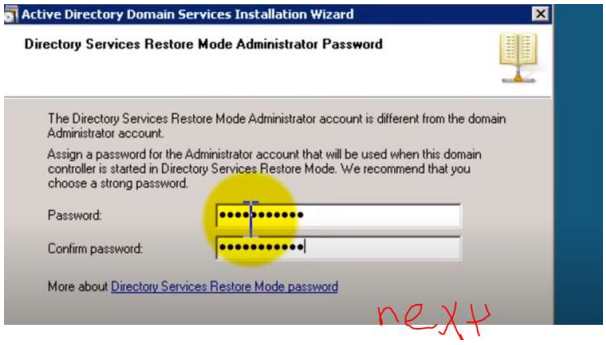
14. Click yes



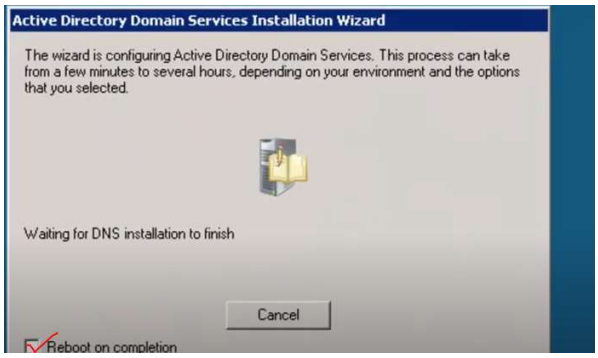
15. Click next



16. Enter password the next



17. Reboot the after finish installation



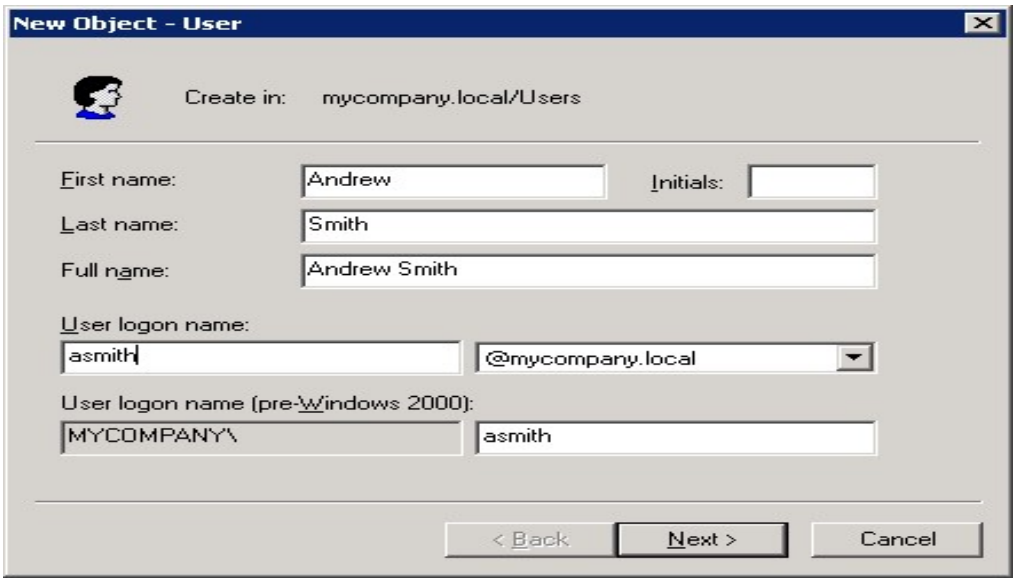
OPERATION SHEET 3.2 SETTING UP DOMAIN ADMINISTRATOR ACCOUNT

Purpose: Setting up Domain Administrator Account

Tools and equipment: Computer Window Server 2012

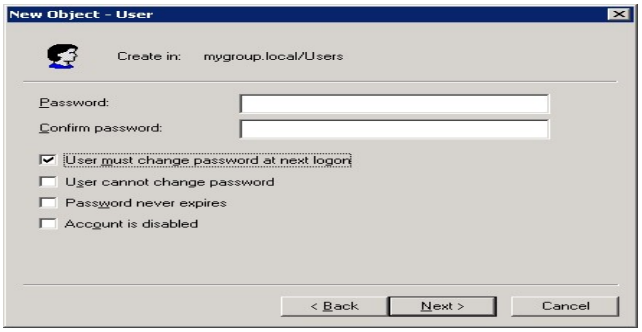
Steps

1. Log in to the domain controller.
2. Click **Start**, point to **Administrative Tools**, and click **Active Directory Users and Computers**.
3. In the left pane of the **Active Directory Users and Computers** window, expand the contents of the newly created Active Directory domain.
4. Right-click the **Users** folder, point to **New**, and select **User**.
5. In the **New Object - User** window, do the following:
    - Type your first and last names in the **First name** and **last name** fields, respectively.
    - In the **User logon name** field, type a name that will be used to log on to the Active Directory domain. For example:



6. After providing the necessary information, click **Next**.

7. Specify an arbitrary password for the domain administrator account and click **Next**



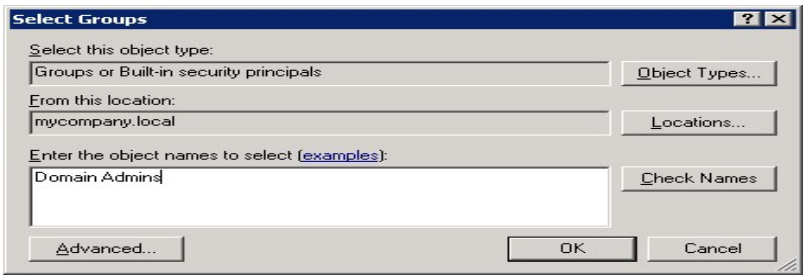
8. The last window allows you to review the parameters provided by on the previous steps. If you wish to modify any parameters, click **Back**; otherwise, click **Finish** to create the domain administrator account.

9. Now you should include the newly created account in the Domain Admins group, which will allow this account to perform administrative tasks in the domain context. To this effect, do the following:

10. In the **Active Directory Users and Computers** window (**Start** -->**Administrative Tools** -->**Active Directory Users and Computers**), right-click the created user account and select **Properties**.

11. Select the **Member Of** tab and click **add**.

12. In the **Select Groups** dialog box, type Domain Admins and click **OK**.



13. Click **OK**.

Quality Criteria: create Domain administrator account

Operation sheet 3.3 Adding Nodes to Domain

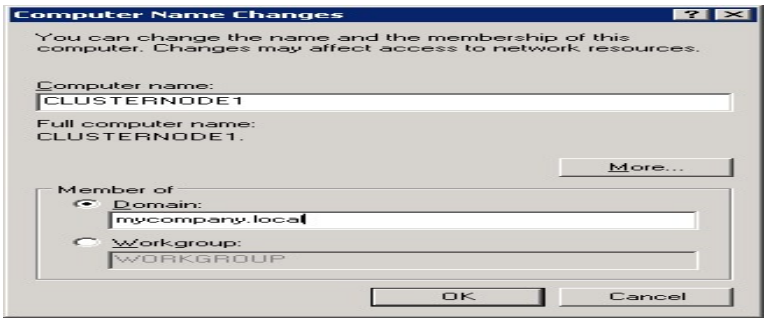
Purpose: Adding Nodes to Domain

Tool and equipment: computer and windows server 2012

Steps

1. Log in to the first node you wish to add to the domain, right-click the **My Computer** icon, and click **Properties**.
2. Select the **Computer Name** tab and click **Change**.
3. In the **Computer Name Changes** window, do the following:
   - In the **Computer name** field, specify a server hostname. This name will be used to uniquely identify the given node among other nodes in the cluster. By default, you are
   - Select the **Domain** radio button and type the domain DNS name (you specified this name during the Active Directory domain). In our example the domain DNS name should be set to mycompany.local.

After providing the necessary information, your window may look like the following:



When you are ready, click **OK**.

4. In the **Computer Name Changes** window, type the username and password of the domain administrator account and click **OK**.
5. Click **OK** to close the displayed message welcoming you to the domain and then click **OK** once more to close the **Computer Name Changes** window.
6. Restart the node.

LAP Test

Instructions: Given necessary templates, tools and materials you are required to perform the

1. Setting Up Domain Administrator Account

References

Books

Ten Steps to Results-Based

Monitoring and and

Evaluation System System by Jody Zall Kusek Ray C. Rist

SQL Server ® 2008 Administration

URL

https://www.alibabacloud.com/blog/why-is-a-sql-log-file-huge-and-how-should-i-deal-with-it_598491#:~:text=What%20Is%20a%20SQL%20Server,operations%20performed%20on%20a%20database.

file:///C:/Users/John_Mobile/Downloads/sql_admin_2%20(1).pdf

https://www.oreilly.com/library/view/database-administration-the/0201741296/

https://www.geeksforgeeks.org/10-best-books-for-database-administrators-and-developers/

https://www.motadata.com/database-monitoring/

Developer's Profile

No	Name	Qualification	Field of Study	Organization/ Institution	Mobile number	E-mail
1	Frew Atkilt	M-Tech	Network & Information Security	Bishoftu Polytechnic College	0911787374	frew.frikii@gmail.com
2	Gari Lencha	MSc	ICT Managment	Gimbi Polytechnic	0917819599	Garilencha12@gmail.com
3	Kalkidan Daniel	BSc	Computer Science	Entoto Polytechnic	0978336988	kalkidaniel08@gmail.com
4	Solomon Melese	M-Tech	Computer Engineering	M/G /M/Polytechnic College	0918578631	solomonmelese6@gmail.com
5	Tewodros Girma	MSc	Information system	Sheno Polytechnic College	0912068479	girmatewodiros @gmail.com
6	Yohannes Gebeyehu	BSc	Computer Science	Entoto Polytechnic College	0923221273	yohannesgebeyehu73@gmail.com