Your Name: Michael Bohen

This template contains placeholders for all sections you need to complete. Submit it as a Word or PDF document; avoid submitting in other formats such as Pages or Gdoc.

For text, make sure to highlight in red all your contributions. For diagrams, make sure to provide comments in red that explain what you contributed and what the AI contributed.

Recall that baseline mastery is worth the majority of the credit, and that real-world robustness and human-AI collaboration also contribute to your grade.

## Sections 1-12 were completed in prior iterations. In this iteration, you will track attribute history (Section 13) and provide data visualizations (Section 14). This will require updating your structural design and updating the SQL script (Section 9). You will then provide an AI Collaboration Summary in Section 15.

## **Section 1: Project Direction Overview**

Blue Sky Incorporated is a global professional services company specializing in providing world-class consulting and professional services for its customers throughout the world. The organization operates branch offices across major international cities including New York, Paris, Los Angeles, Boston, London, Mexico City, Tokyo, Sydney, Bangalore, and Cape Town. Blue Sky Incorporated has a global IT team that manages the network infrastructure in each location. Each office varies in size and has a structured, segmented network architecture comprising distinct VLANs and subnets for users, audio/visual systems, printers, physical security, and dedicated management networks. Dual Internet connections are deployed at every site to ensure redundancy, with static public IP allocations assigned by multiple ISPs.  
   
For many years, the company has relied on flat, manually updated Excel-based documentation to track the design and operations of its network. This documentation includes IP address assignments, subnets, and DHCP scopes across its offices. Over time, this approach has led to data duplication, inconsistency, and difficulty maintaining synchronization across regions. The global network team currently spends considerable time each month performing ad hoc and manual tasks to manage these issues. Additionally, these issues increase the effort required to deploy new services and to deploy new offices. The IT department has a strategic initiative to implement automation; however, these data issues create a critical barrier to being able to implement these technologies within a production setting. A centralized, authoritative database—a “source of truth”—is now essential to support the core department initiatives of global standardization and future network automation.  
   
The proposed database, \*\*BlueSky-IPAM\*\*, will serve as a relational IP Address Management (IPAM) system that stores information about the low-level design and utilization of network services. It will include hierarchical IP allocation data, IP address assignment information (including DHCP scope information and the static IP address reservations used for devices such as printers and meeting room equipment. This data will allow engineers and auditors to quickly determine which networks and IP addresses are in use, identify conflicts or deviations from standards, and provide reliable data for future automation systems to deploy new offices and to deploy and maintain services within existing offices.  
   
This database will be designed with normalization and auditability in mind, ensuring that all IP data adheres to consistent rules and remains queryable across time. Over time, it will also serve as the authoritative input for zero-touch provisioning workflows that use automation platforms to configure routers, switches, and DHCP services with minimal manual input.

## **Section 2: Artifacts and Analysis**

## This section describes the real-world artifacts that shaped the design of BlueSky-IPAM. Each artifact reflects how IP address management currently functions in Blue Sky’s environment or comparable enterprise networks. Entities, actions, and business rules were extracted from these artifacts and aligned to industry best practices.

## Artifact 1 – IP Subnet Tracking Spreadsheet

## It is a common practice throughout the IT industry to use spreadsheets and other similarly structured files to track the IP space allocation. The practices are sufficient for small environments that change infrequently. The following template is an example of such a file:

## A screenshot of a spreadsheet AI-generated content may be incorrect.

## Source: Public IP address spreadsheet template (ActiveDirectoryPro.com). <https://activedirectorypro.com/ip-address-spreadsheet-template/?utm_source=chatgpt.com>

This artifact reflects a spreadsheet for a single office. In a multi-office organization such as ours, it is common practice to create additional spreadsheets or tabs for additional networks. These spreadsheets also would then cover multiple levels of organizations identifying the IP address space assigned to particular offices and other network data specific to these offices. As the scale grows, however, these become increasingly difficult to keep consistent and up to date.

## **Description**:

This artifact mirrors the model used to document Blue Sky’s networks. It organizes data by site and network, IP addresses, subnet masks, DHCP ranges, and host assignments. Each spreadsheet tab typically represents a single site or region, with color-coded sections for user networks, AV systems, printers, and security devices. Although functional, this format lacks referential integrity and consistent naming conventions. This leads to errors, duplication of data, and data drift.

## **Entities**: From these, we can identify the following entities:

* **Site** – a physical office or regional location
* **LAN Supernet** – a /16 network allocation assigned to a specific site.
* **LAN Subnet** – a /24 network segment used by a specific floor or function.
* **DHCP Scope** – a defined address pool within a subnet (e.g., 10.1.1.50 – 10.1.1.200).
* **Host** – a unique device (such as a workstation, printer, camera, AV unit) assigned an IP.  
  (A host in turn possess attributes such as “Description,” “Subnet Mask,” and “DNS Server”).

**Actions:**

We can identify the following actions from this artifact:

* Create, edit, and export supernet and subnet data
* Add or remove subnet
* Add or remove DHCP scope
* Add and remove host assignment
* Verify uniqueness of IP address data
* Verify integrity of IP network data

## **Constraints and Business Rules:**

## We can identify several business rules from this data:

## Each site is allocated a /16 LAN supernet.

## Each supernet contains one or more subnets.

## All new supernets and subnets must be allocated from within the private IP address range of 10.0.0.0/8.

## A subnet is assigned for a specific function within a specific location.

## Each LAN subnet must be assigned with a prefix length of /24.

## Each IP address must be unique and associated with a single active host

## Each IP address can have only one active assignment at a time.

## Subnets and supernets may not overlap.

## All changes must be recorded with timestamps and responsible user IDs.

## **Influence:** This artifact is used to demonstrate the nature and structure of the networking entities that this database will work with including supernets, subnets, hosts, and DHCP scopes. It also helps establish the rules and constraints that we will use to design the database and highlights the operational need for relational integrity and constraint enforcement. Specifically, this drives the requirements for unique indexes on IP addresses and non-overlapping IP Address ranges within the database.

## Artifact 2 – Meraki MX DHCP Configuration Documentation

In networking, routers are used to forward traffic from one network to another. These routers in an enterprise have critical configuration that defines what networks exist within the organization. Additionally, most environments, including ours, use a service called Dynamic Host Configuration Protocol (DHCP) to automatically allocate IP addresses to devices from a pool of available addresses. This service is commonly configured on the network router as well. With a DHCP service, an administrator can define a static reservation for a device based on the device’s MAC address. With a static reservation, if a DHCP server receives a DHCP packet from a device with a static reservation defined, it will offer (and assign) that IP address to the device. In our example, we are using a Meraki MX router to define the networks and to host the DHCP service. As a result, we will use the DHCP configuration from Meraki routers as an artifact.

## <https://documentation.meraki.com/General_Administration/Cross-Platform_Content/Configuring_DHCP_Services_on_the_MX_and_MS>

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **Source**: Cisco Meraki documentation: “Configuring DHCP Services on the MX and MS” <https://documentation.meraki.com/General_Administration/Cross-Platform_Content/Configuring_DHCP_Services_on_the_MX_and_MS>

## **Description**: The Meraki MX appliance serves as both router and DHCP server in many Blue Sky locations. This artifact details how VLANs map to subnets and how DHCP services are defined, including scope ranges, lease durations, and static reservations. These provide additional parameters for network devices that will inform the database schema.

## **Entities**

From these we can identify the following entities:

* **Router** – the Meraki MX router providing routing and DHCP services.
* **LAN Interface** – a logical interface on the Device associated with a VLAN and subnet.
* **DHCP Scope** – the address pool configured for that subnet.
* **Host** – the endpoint associated with the reservation.

## **Actions**:

## We can identify the following actions from this artifact:

## Configure and modify DHCP scopes

## Add and remove static DHCP reservations

## Retrieve active IP address assignments from a DHCP scope

## Associate a LAN interface with a specific subnet and scope

## **Constraints and Business Rules:**

## We can identify several business rules from this artifact:

* A DHCP Scope must be fully contained within its parent Subnet
* Each Scope is tied to a single LAN Interface on a Device.
* Each static DHCP reservation must include a MAC address and hostname.
* Each dynamic DHCP lease has a finite duration.
* All static DHCP reservations and Dynamic DHCP releases must be allocated such that an IP address is only ever assigned to one device at a time.

## **Influence**: This artifact clarifies the relationships between the DHCP scope, subnet, and reservation entities. This ensures that the database properly aligns to the requirements of this functionality and that the database properly conforms to how these entities operate in production networks.

## Artifact 3 – Meraki Zero-Touch Provisioning and Scalable Deployment

The third artifact will describe our ultimate goal of being able to implement Zero Touch Deployments in our environment. This functionality is available on many platforms, but we will use our Meraki platform as an example technology. At present, engineers configure and deploy all network devices manually. In the future, we wish to use automation to prestage all required configuration. An engineer can plug in a network device, and the device will retrieve and apply all required configuration without human interaction. The document describes many of the settings in scope for such a deployment. These include such elements as Internet settings, templates, logging settings, switchport settings, and other elements. (This document is high level, and additional documentation and additional artifacts are needed to capture this detail in depth, however we will use this high-level document to identify entities we wish to include in our database).

## A screenshot of a computer AI-generated content may be incorrect.

## A screenshot of a computer AI-generated content may be incorrect. Source: Cisco Meraki documentation and best-practice guides (documentation.meraki.com). <https://documentation.meraki.com/General_Administration/Design_and_Configure/Architectures_and_Best_Practices/Cisco_Meraki_Best_Practice_Design/Building_a_Scalable_Meraki_Solution?utm_source=chatgpt.com>

## Description: This document describes, on a high level, the processes for building networks based on Cisco Meraki devices with a goal for facilitating scale and automation. It describes Meraki’s cloud-based deployment model, where devices are pre-provisioned in the Meraki Dashboard and automatically retrieve configurations when powered on. It outlines how organizations claim devices, assign them to sites, apply configuration templates, and define WAN/LAN.

## Entities:

We can identify the following entities from this artifact:

## **Organization** - the logical top-level container for all Sites and Devices.

## **Device** – A hardware appliance (i.e.: a router or switch)

## **Switch** – A hardware appliance to which a host is physically connected to

## **Switchport** – A physical interface on a switch the host connects to that contain a series of settings that define what subnet a host will connect to

## **Internet connection** – An entity containing the settings of a connection to the Internet (such as Internet carrier, circuit ID, IP address ranges)

## **WAN Interface –** a physical interface on a router providing connectivity for **an Internet connection**

## **LAN Interface –** a physical interface on a router providing connectivity for a network to other networks in the environment

## **Template** - the pre-defined configuration applied to Devices at deployment

## Actions:

## We can identify the following actions from this artifact:

## Claim devices

## Assign device to site

## Apply Template to device

## Configure switchport

## Assign host to switchport and remove host from switchport

## Create Internet connection

## Update Internet connection settings

## Associate Internet connect to WAN interface / dissociate Internet connection from WAN interface

## Associate VLAN to Subnet

## **Constraints and Business Rules**:

## We can identify several business rules from this artifact:

## Devices must exist in the inventory and be assigned to a site before they can be configured.

## Each Device has one or more Interfaces, each with a defined role

## A LAN Interface is associated with exactly one LAN Subnet at a time

## A WAN Interface is associated with exactly one Internet Connection.

## Templates can be applied to multiple Devices, but a Device may reference only one active Template.

## Pre-provisioned Interfaces and Templates must reference valid Subnets and Scopes to maintain referential integrity.

## All sites must have at least one valid and operational Internet connection.

## There is a one-to-one mapping of each VLAN to each subnet.

## A device is assigned to a port. A port is assigned to a VLAN which is associated with a subnet.

**Influence**: This document provides a high level of overview of the considerations and processes involved in scaling and automating the network technologies of Cisco Meraki products. (Other vendor platforms are similar). From this document we can infer the remaining entities including network device, site, VLAN, port configuration, and template. The templates will be the entity that provides a blueprint for the future automation integration where BlueSky-IPAM serves as the input source for provisioning scripts.  
  
This artifact introduces the entities and relationships required to sufficiently model the network configuration present within the BlueSky organization. It also creates the entities necessary to extend BlueSky-IPAM into automation workflows. It ensures that the model supports hierarchical relationships among Organization → Site → Device → Interface, while maintaining validation rules for Templates and pre-deployment configurations. These elements will directly inform the structural database rules and conceptual ERD developed going forward.

## **Section 3: Use Cases**

## The following use cases represent the core functions of the BlueSky-IPAM system. Each use case is designed around a specific operational objective that supports network engineering, configuration management, or compliance auditing activities.

## Use Case 0 – Create new Organization

**Actor**: Network Engineer

## **Preconditions**:

## The action is being performed by an authorized user.

## The requested organization does not exist.

## **Steps**:

## The engineer enters the metadata and attributes of the organization.

## The system creates the organization entries.

## **Postconditions**: A new organization is created.

## Use Case 1 – Create New Site

## **Actor**: Network Engineer

## **Preconditions**:

## The action is being performed by an authorized user.

## The requested site does not exist.

## There is a free /16 supernet available in the 10.0.0.0/8 IP Range.

## **Steps**:

## The engineer enters the metadata and attributes of the site.

## The system allocates a /16 LAN Supernet to the site.

## The system creates the Site and Supernet entries.

## **Postconditions**: A new site created with a unique supernet.

## Use Case 2 – Create New Subnet

## **Actor**: Network Engineer

## **Preconditions**:

## The site has a /16 supernet allocated.

## The requested subnet does not yet exist in the site.

## **Steps**:

## The Engineer requests a new subnet in a specified site.

## The Engineer (optionally) requests that the new subnet include a DHCP scope.

## The system assigns the next available /24 subnet.

## The system creates the subnet and DHCP scope entries.

## **Postconditions**: The subnet and DHCP scopes are created and recorded for site.

## Use Case 3 – Create New DHCP Reservation

## **Actor**: Network Engineer or Helpdesk Engineer

## **Preconditions**:

## The host exists and the MAC address of the host is known.

## An IP address is available within subnet.

## **Steps**:

## An engineer submits DHCP reservation details (IP, MAC, Hostname, Owner, Reason).

## The system validates the supplied information.

## If valid, the System creates the DHCP reservation.

## **Postconditions**: The host has an active DHCP reservation and will be assigned the specified IP address

## Use Case 4 – Get Site Data

## **Actor**: Network Engineer or Auditor

## **Steps**:

## The engineer or audit requests the details for a given site.

## The query returns site details of the subnets, DHCP scopes, and hosts present in the site.

## **Postconditions**: The requester receives a complete, authoritative snapshot of the site.

## Use Case 5 – Get Host Data

## **Actor**: Network Engineer or Helpdesk

## **Steps**:

## The actor performs a query for a particular IP address, MAC address, or Hostname

## The system returns current and historical records pertaining to his host.

## **Postconditions**: The actor receives valid data pertaining to the requested address or hostname.

## **Use Case 6 – Assign LAN Interface to Subnet (Future state with Network Automation)**

**Actor:** Network Engineer

**Preconditions:**  
The Device exists and contains one or more available LAN Interfaces. The target Subnet exists and is associated with the correct Site and VLAN.

**Steps:**

1. The engineer specifies the desired LAN interface
2. The engineer specifies the desired subnet and associated VLAN
3. The automation system validates the configuration and that the Interface is not already assigned to another Subnet.
4. The automation system applies the interface configuration to the router.
5. The automation system updates the associated attributes of the BlueSky-IPAM database

**Postconditions:** The LAN Interface is linked to a specific LAN Subnet and is ready for use in DHCP or routing operations.

## **Use Case 7– Configure WAN interface for Internet Connection (Future state with Network Automation)**

**Actor:** Network Engineer

**Preconditions:**  
The Device has at least one WAN Interface available, and an associated Internet Connection record (e.g., ISP Circuit) is available.

**Steps:**

1. The engineer specifies the desired WAN Interface.
2. The engineer specifies the desired Internet connection (containing settings such as Public IP address, Subnet Mask, and Default Gateway.
3. The system validates the configuration and that the WAN interface IP address falls within the ISP allocation and does not conflict with other records.
4. The automation system applies the interface configuration to the router.
5. The automation system updates the associated attributes of the BlueSky-IPAM database

**Postconditions:** The WAN interface configuration is applied to the router, and the BlueSky-IPAM database contains up to date records of the configuration of this interface.

## Use Case 8 – Audit Compliance

## **Actor**: Compliance Auditor or Scheduled Process

## **Steps**:

## The system compares configuration data of the live network to the data recorded in the IPAM database.

## The system logs exceptions for overlaps, invalid ranges, or unapproved subnets.

**Postconditions**: Any Compliance deviations are identified and recorded for further review and remediation.

# Section 4: Structural Database Rules (revised)

This section defines the structural database rules that govern the relationships among entities in the schema that will be used by the BlueSky-IPAM database. Each rule describes how entities interact within the conceptual model and identifies the purpose behind the constraint. The rules are expressed in natural language and aligned with UML multiplicity notation (1, 0..1, 0..\*, 1..\*) on both ends of each association. Relationship constraints are defined at the level of general entities (e.g., Device, Interface) rather than subtype-specific entities (e.g., Switch).

## 4.1 Organizational Hierarchy Rules

Rule 1 — Organization to Site

Each **Organization** has zero or more **Sites**.   
Each **Site** belongs to exactly one **Organization**.

**Rationale**: This rule establishes the top-level hierarchy that aligns with Meraki’s organizational structure. This ensures that all sites are logically grouped under a single global organization for governance and reporting.

Rule 2 — Site to LAN Supernet

Each **Site** may have zero or one **LAN Supernet**.   
Each **LAN Supernet** is assigned to exactly one **Site**.

**Rationale**: This rule enforces Blue Sky’s addressing policy of assigning a dedicated /16 network per site to maintain a standardized and conflict-free IP address allocation scheme. It also allows for simplified routing between sites. A supernet always corresponds to a single site, but some sites may not yet be assigned a supernet (for example, during early planning or in very small deployments).

Rule 3 — LAN Supernet to LAN Subnet

Each **LAN Supernet** may contain zero or more **LAN Subnets**.   
Each **LAN Subnet** belongs to exactly one **LAN Supernet**.

**Rationale**: This hierarchical relationship provides clear segmentation of each /16 block into /24 subnets. This supports location and function dependent network design within a branch site while preserving traceability to the site. A supernet may temporarily have no child subnets. An example might be a brand-new site that has not yet been assigned a subnet.

## 4.2 LAN Subnet and DHCP Structure Rules

## Rule 4 — LAN Subnet to DHCP Scope

Each **LAN Subnet** may contain zero or more **DHCP Scopes**.   
Each **DHCP Scope** belongs to exactly one **LAN Subnet**.

**Rationale**: DHCP is an optional network function that is not required on every LAN subnet.

## Rule 5 — DHCP Scope to DHCP Reservation

Each **DHCP Scope** can contain zero or more **DHCP Reservations**.   
Each **DHCP Reservation** belongs to exactly **one DHCP Scope**.

**Rationale**: This structure supports both dynamic and static addressing while ensuring that reserved IP addresses remain within their designated scope.

Rule 6 — DHCP Reservation to Host Rules

Each **DHCP Reservation** is associated with exactly one **Host**.  
Each **Host** may have zero or more **DHCP Reservations**.

## **Rationale:** Reservations are assigned to a single host (for example, identified by MAC address). A host may accumulate multiple historical reservations across different scopes or sites, but at any given time it should have at most one active reservation per scope. The “one active per scope” rule is enforced as a business rule or unique index rather than by a separate entity. 4.3 Device and Interface Hierarchy Rules

Rule 7 — Site to (Network) Device

Each **Site** may contain zero or more **Devices**.   
Each **Device** belongs to exactly one **Site**.

Rationale: In our schema, a device specifically refers to a networking device such as a switch or a router. Linking devices to sites provides location-based organization for routers, switches, and other infrastructure, enabling accurate reporting and automation by geographic or functional region. When a site is first created, there will not be any devices assigned or deployed to it.

Rule 8 — Device to Interfaces

Each **Device** has one or more **Interfaces**.   
Each **Interface** belongs to exactly one **Device**

**Rationale**: This rule models the physical or logical connections available on each device and supports automated configuration through role-based templates.

## Rule 9 —Interface to LAN Subnet (LAN role only)

Each **Interface with a LAN role** must be associated with exactly one **LAN Subnet**.   
Each **LAN Subnet** may be associated with zero or more **Interfaces with a LAN role**.

**Rationale**: This relationship captures how switch or router interfaces connect to subnets and ensures that each interface corresponds to a single network. In a network, a single router can be the only device routing to a LAN subnet. A LAN subnet may also have multiple routers configured in a high availability posture where a single router is active and a backup router is in a passive state. A LAN subnet can be a transit network, whereby traffic may be routed through the LAN subnet to reach a destination on a different device.

Rule 10 —Interface to Internet Connection (WAN role only)

Each **Interfaces** **with** **a** **WAN role** connects to exactly one **Internet Connection**.   
Each **Internet Connection** may serve zero or more **interfaces** **with** **a** **WAN role**.

**Rationale**: This rule records ISP circuit details and enables high-availability pairs or dual-uplink scenarios to share the same provider information.

Rule 11 — Interface to Host (access connectivity)

Each **Host** must be connected to exactly one **Interface** at any given time.   
Each **Interface** may connect to zero or one **Host** at a given time.  
(Each SwitchPort may serve many hosts over its lifetime, but only one concurrently).

**Rationale**:   
This rule supports tracking which physical or logical port a host is currently using and maintains historical flexibility as hosts move between interfaces over time.

Rule 12 — Template to Device

Each **Template** may be applied to **zero or more** **Devices**.   
Each **Device** may reference **zero or one** active **Template**.

**Rationale**: Templates provide standardized configuration sets that can be reused across devices, supporting zero-touch provisioning and consistent deployment practices.

## 4.4 Specialization Rules

## Rule 13 — Device Specialization

A Device is a Router or a Switch, or none of these.

This specialization is **disjoint**. Devices may be only one subtype at a time. It is also **partially complete.** Some Devices may be neither Router nor Switch.

The specialization is implemented physically through two subtype tables.

• Router(deviceId PK/FK → Device.deviceId)

• Switch(deviceId PK/FK → Device.deviceId)

Each subtype includes attributes unique to that subtype.

**Device Subtype Discriminator.**  
To support this specialization between Router and Switch, the Device table includes an attribute **deviceType** constrained to the values ('Router', 'Switch', 'Other'). This discriminator identifies which subtype that each Device participates in. It also ensures that each specialized device is belongs to exactly one subtype. This discriminator also simplifies querying and reporting while reinforcing the disjoint nature of the specialization.

**Rationale:** This specialization recognizes that, while there are some attributes and properties that all devices share, some device types may serve fundamentally different networking functions. Routers and Switches require additional attributes that do not apply to all Devices (e.g., routing protocols, WAN failover modes, PoE support, stacking capabilities). Implementing this specialization with subtype tables will allow us to design the database to facilitate these differences while also allowing for the more universal attributes to be used by all device types.

## Rule 14 — Interface Role Specification

Conceptually, an Interface will have three roles.

* WAN\_Interface
* LAN\_Interface
* Management\_Interface

These roles will be implemented physically through the interfaceType attribute combined with role-specific constraints:

• If interfaceType = 'LAN', lanSubnetId must be NOT NULL and internetConnectionId must be NULL.

• If interfaceType = 'WAN', internetConnectionId must be NOT NULL and lanSubnetId must be NULL.

• If interfaceType = 'MGMT', both may be NULL unless future iterations define management-network rules.

No subtype tables will be created for Interface.

**Rationale**: The LAN, WAN, and Management interface roles represent conceptual distinctions in how interfaces are used within a network and their relationships with other entities. However, these roles do not have specific sets of attributes that justify separate physical subtype tables. Instead, the physical schema models this specialization through the interfaceType attribute and its associated conditional constraints, which determine which foreign keys must be present.

## Rule 15 — Host to Host IP Address History

Each **Host** may have zero or many IP address **records**.  
  
Each historical IP address **record** is associated with exactly one **Host**.

**Rationale:**  
Most hosts will receive their IP addresses dynamically via DHCP. As these leases expire or network changes occur, a host may be reassigned a new IP address. Tracking historical IP address values allows administrators to determine which Host held a particular IP address at any point in time. This supports critical operational tasks such as network troubleshooting, DHCP diagnostics, security investigations, and forensic analysis. A one-to-many relationship ensures that each Host can accumulate a sequence of recorded IP transitions throughout its lifecycle.

# Section 5: Conceptual ERD (revised)

The entities depicted in the ERD are as follows:

- **Organization** — Represents the corporate entity (e.g., BlueSky Incorporated) that owns and operates multiple sites.

- **Site** — Represents a physical location (headquarters, regional office, branch) where network infrastructure is deployed.

- **LAN\_Supernet** — Represents the high-level address block allocated to a site (for example, 10.1.0.0/16).

- **LAN\_Subnet** — Represents individual subnets (for example, 10.1.10.0/24) used for specific networks.

- **DHCP\_Scope** — Represents the dynamic address range configured for a subnet on a DHCP server.

- **DHCP\_Reservation** — Represents a reserved IP address that is assigned to a particular host.

- **Host** — Represents an endpoint or server using an IP address (ie: a user workstation, printer, or application server).

- **Device** — Represents a network device such as a router or switch deployed at a site.

- **Router** – Represents a network router that is configured to route traffic between networks

- **Switch** – Represents a network switch that is configured to switch traffic within a network (but not route traffic between networks)

- **Interface** — Represents a logical or physical interface on a Device that connects to subnets, circuits, or hosts.

- **Internet\_Connection** — Represents a provisioned Internet circuit (for example, an ISP link with specific bandwidth and circuit ID).

- **Template** — Represents a reusable configuration template that may be linked to multiple Devices.

- **HostIPAddressHistory** – Represents a historical record of which IP address was assigned to a host at a given point in time.

## 5.2 Specialization: Device → Router, Switch

**Device Specialization**

A Device may conceptually be specialized into one of the following subtypes:

* **Router**
* **Switch**

This specialization has the following characteristics:

* **Disjoint** - a Device may be only one subtype.
* **Partially** **complete** - a Device may be neither subtype.

The specialization is represented in the conceptual ERD as:

* A generalization triangle connected to Device at the top
* Two subtype lines leading to Router and Switch
* The UML constraints: {or} and {optional}

**Subtype Discriminator.**  
The Device entity includes an attribute deviceType: {Router, Switch, Other} that acts as the subtype discriminator in the conceptual model. Although Router and Switch have their own subtype entities, the discriminator also provides a single-variable representation of the disjoint specialization.

## 5.3 Conceptual Interface Specification

Conceptually, an Interface will have one of three roles:

* **WAN\_Interface** — Interfaces that terminate WAN circuits.
* **LAN\_Interface** — Interfaces that connect to LAN subnets.
* **Management\_Interface** — Interfaces used for out-of-band or management access.

These will be implemented via a type attribute. Since, this is not a true specialization (since there will not be subtypes with distinct attributes), and these will not be represented as a specialization relationship. This logic will nonetheless be important to establish the function of these interface types.

## 5.4 Associations and Multiplicities

The key associations are:

- **Organization**–**Site**: An Organization can operate multiple Sites. Each Site belongs to exactly one Organization.

- **Site**–**LAN\_Supernet**: A Site may have zero or one LAN\_Supernet associated with it. Each LAN\_Supernet is assigned to exactly one Site.

- **LAN\_Supernet**–**LAN\_Subnet**: A LAN\_Supernet may contain multiple LAN\_Subnets. Each LAN\_Subnet is contained within exactly one LAN\_Supernet.

- **LAN\_Subnet**–**DHCP\_Scope**: A LAN\_Subnet may have no DHCP\_Scope or multiple scopes. Each DHCP\_Scope is defined for exactly one LAN\_Subnet.

- **DHCP\_Scope–DHCP\_Reservation**: A DHCP\_Scope may include zero or more DHCP\_Reservations. Each DHCP\_Reservation is defined within exactly one DHCP\_Scope.

- **DHCP\_Reservation–Host**: A DHCP\_Reservation is associated with exactly one Host. A Host may have multiple reservations over time. Note: A host can have only one active reservation per scope at a given time, however these will be enforced by business rules, rather than the entity design.

- **Site–Device**: A Site may host zero or more Devices. Each Device is associated with exactly one Site.

- **Device–Interface**: A Device must have at least one Interface. Each Interface belongs to exactly one Device.

- **Interface (LAN role)–LAN\_Subnet**: An Interface with a LAN role must connect to exactly one LAN\_Subnet. Each LAN\_Subnet may be served by zero or more Interfaces with a LAN role.

- **Interface (WAN role)–Internet\_Connection**: An Interface with a WAN role must connect to exactly one Internet\_Connection. Each Internet\_Connection may be terminated by zero or more Interfaces with a WAN role.

- **Interface–Host**: Each Host must connect to exactly one Interface at a point in time. Each Interface may have zero or one Host connected concurrently.

- **Template–Device**: A Template may be linked to zero or more Devices. Each Device may reference zero or one Template.

- **Host–HostIPAddressHistory**: A host may be linked to zero or more historical IP addresses record. Each historical IP address record is linked to exactly one host.

## 5.5 ERD Diagram

The following is an entity-relationship diagram (ERD) for the entities we have identified:

A computer screen shot of a computer flow chart

AI-generated content may be incorrect.

## 5.6 Relationship Summary Table

The following table is a summary of these relationships.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Relationship | Cardinality (UML Notation) | Description |
| 1 | Organization ↔ Site | **Organization (1) —— (0..\*) Site** | An organization can operate many sites, and some organizations may temporarily have no active sites. Each site belongs to exactly one organization. |
| 2 | Site ↔ LAN Supernet | **Site (1) —— (0..1) LAN Supernet** | Each site belongs to 0 or 1 LAN Supernet, and each supernet belongs to one site. |
| 3 | LAN Supernet ↔ LAN Subnet | **LAN Supernet (1) —— (0..\*) LAN Subnet** | Each LAN supernet can contain zero or more LAN subnets, and each LAN subnet belongs to exactly one supernet. |
| 4 | LAN Subnet ↔ DHCP Scope | **LAN Subnet (1) —— (0..\*) DHCP Scope** | A LAN subnet may have no DHCP scope or multiple scopes, and each DHCP scope is defined for exactly one subnet. |
| 5 | DHCP Scope ↔ DHCP Reservation | **DHCP Scope (1) —— (0..\*) DHCP Reservation** | A scope can include multiple reservations, and each reservation belongs to one scope. |
| 6 | DHCP Reservation ↔ Host | **DHCP Reservation (0..\*) —— (1) Host** | Each reservation maps to one host, and each host may have zero or one reservation per scope. |
| 7 | Site ↔ Device | **Site (1) —— (0..\*) Device** | A site may host many devices, and each device resides at one site. A site may also contain no devices (if it is new with no devices deployed yet) |
| 8 | Device ↔ Interface | **Device (1) —— (1..\*) Interface** | A device may have multiple interfaces, and each interface belongs to one device. |
| 9 | Interface (LAN Role) ↔ LAN Subnet | **Interface (LAN Role) (0..\*) —— (1) LAN Subnet** | Each LAN-role interface connects to exactly one LAN subnet, and each subnet may have zero or more LAN-role interfaces. |
| 10 | Interface (WAN Role) ↔ Internet Connection | **Interface (WAN Role) (0..\*) —— (1) Internet\_Connection** | Each WAN-role interface connects to exactly one Internet connection, and each internet connection may be associated with zero or more WAN-role interfaces. |
| 11 | Interface ↔ Host | **Interface (1) —— (0..1) Host** | An interface may be connected to 0 or 1 host at a given time. Each host is connected to exactly one interface at a time. (If a host is not connected to an interface it is not on our network). |
| 12 | Template ↔ Device | **Template (0..1) —— (0..\*) Device** | A Template can apply to many Devices, and a Device may have zero or one active Template. |
| 13 | Host ↔ HostIpAddressHistory | **Host (1) —— (0..\*) HostIPAddressHistory** | A host can appear in the IP Address history zero or more times. Each IP Address History entry applies to one host |

# Section 6: Physical ERD (revised)

This section describes how the conceptual UML model for BlueSky-IPAM will be implemented in the physical database. Note: Device and Interface are each implemented as a single table with type-discriminating attributes rather than multiple subtype tables.

## 6.1 Core Tables and Keys (UML «table» Notation)

The following are the main tables that will be stored in the BlueSky-IPAM database. These specifications will follow the UML convention of using the «table» stereotype):

**«table» Organization**

- PK organizationId

- name

- description

**«table» Site**

- PK siteId

- FK1 organizationId → Organization.organizationId (NOT NULL)

- name

- region

- location

**«table» LanSupernet**

- PK lanSupernetId

- FK siteId → Site.siteId (NOT NULL, UNIQUE)

- cidr

- description

**«table» LanSubnet**

- PK lanSubnetId

- FK lanSupernetId → LanSupernet.lanSupernetId (NOT NULL)

- SubnetCidr

- gatewayAddress

- vlanId

**«table» DhcpScope**

- PK dhcpScopeId

- FK lanSubnetId → LanSubnet.lanSubnetId (NOT NULL)

- startAddress

- endAddress

- leaseTimeMinutes

**«table» Host**

- PK hostId

- hostName

- macAddress

- ipAddress

- assetTag

- FK interfaceId → Interface.interfaceId (NOT NULL, UNIQUE to ensure at most one active host per interface)

**«table» DhcpReservation**

- PK dhcpReservationId

- FK dhcpScopeId → DhcpScope.dhcpScopeId (NOT NULL)

- FK hostId → Host.hostId (NOT NULL)

- ipAddress

- reservationStatus

- validFrom

- validTo

**«table» Device**

- PK deviceId

- FK siteId → Site.siteId (NOT NULL)

- FK templateId 🡪 Template.templateID NULL

- name

- vendor

- model

- managementAddress

- serialNumber

- firmwareVersion

- installDate

**«table» Router**

- PK/FK deviceId → Device.deviceId (NOT NULL)

- wanFailoverType

- advancedSecurityEnabled

- maxThroughputMbps

- supportsBgp

- redundancyType

**«table» Switch**

- PK/FK deviceId → Device.deviceId (NOT NULL)

- poeSupported

- poeBudgetWatts

- stackable

- layer3Capable

**«table» Interface**

- PK interfaceId

- FK deviceId → Device.deviceId (NOT NULL)

- FK lanSubnetId → LanSubnet.lanSubnetId (nullable; NOT NULL when interfaceType = 'LAN')

- FK internetConnectionId → InternetConnection.internetConnectionId (nullable; NOT NULL when interfaceType = 'WAN')

- name

- interfaceType -- 'WAN', 'LAN', or 'MGMT' (enforced by CHECK constraint)

- adminState

**«table» InternetConnection**

- PK internetConnectionId

- provider

- circuitId

- bandwidthMbps  
- connectionCIDR  
- InstallDate

**«table» Template**

- PK templateId

- templateName

- version

- description

**<<table>> HostIPAddressHistory**

- PK hostIPAddressHistoryID

- FK hostId→ Host.hostID

- oldIpAddress

- newIpAddress

- changeDate

## 6.2 Key Physical Relationships and Constraints

The following is a summary of the primary and foreign key relationships between tables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Relationship | PK (Parent) | FK (Child → Parent) | Cardinality (Physical) | Notes |
| 1 | Organization ↔ Site | Organization.organizationId | Site.organizationId (NOT NULL) → Organization.organizationId | Organization (1) — (0..\*) Site | Each Site must belong to one Organization. |
| 2 | Site ↔ LanSupernet | Site.siteId | LanSupernet.siteId (NOT NULL, UNIQUE) → Site.siteId | Site (1) — (0..1) LanSupernet | UNIQUE ensures a Site may have at most one LanSupernet. |
| 3 | LanSupernet ↔ LanSubnet | LanSupernet.lanSupernetId | LanSubnet.lanSupernetId (NOT NULL) → LanSupernet.lanSupernetId | LanSupernet (1) — (0..\*) LanSubnet | Each Subnet belongs to exactly one Supernet. |
| 4 | LanSubnet ↔ DhcpScope | LanSubnet.lanSubnetId | DhcpScope.lanSubnetId (NOT NULL) → LanSubnet.lanSubnetId | LanSubnet (1) — (0..\*) DhcpScope | Multiple DHCP scopes per subnet allowed. |
| 5 | DhcpScope ↔ DhcpReservation | DhcpScope.dhcpScopeId | DhcpReservation.dhcpScopeId (NOT NULL) → DhcpScope.dhcpScopeId | DhcpScope (1) — (0..\*) DhcpReservation | Each Reservation is defined within one Scope. |
| 6 | DhcpReservation ↔ Host | Host.hostId | DhcpReservation.hostId (NOT NULL) → Host.hostId | Host (1) — (0..\*) DhcpReservation | A Host may have multiple reservations defined in different subnets |
| 7 | Site ↔ Device | Site.siteId | Device.siteId (NOT NULL) → Site.siteId | Site (1) — (0..\*) Device | A Site may contain zero or more Devices. |
| 8 | Device ↔ Interface | Device.deviceId | Interface.deviceId (NOT NULL) → Device.deviceId | Device (1) — (1..\*) Interface | Each Device must have at least one Interface. |
| 9 | Interface (LAN) ↔ LanSubnet | LanSubnet.lanSubnetId | Interface.lanSubnetId (NOT NULL when interfaceType = 'LAN') → LanSubnet.lanSubnetId | Interface (LAN) (0..\*) — (1) LanSubnet | lanSubnetId required only for LAN-role interfaces. |
| 10 | Interface (WAN) ↔ InternetConnection | InternetConnection.internetConnectionId | Interface.internetConnectionId (NOT NULL when interfaceType = 'WAN') → InternetConnection.internetConnectionId | Interface (WAN) (0..\*) — (1) InternetConnection | internetConnectionId required only for WAN-role interfaces. |
| 11 | Interface ↔ Host | Interface.interfaceId | Host.interfaceId (NOT NULL, UNIQUE) → Interface.interfaceId | Interface (1) — (0..1) Host | UNIQUE enforces one active Host per Interface. |
| 12 | Template ↔ Device | Template.templateId | Device.templateId (NULL) → Template.templateId | Device (0..\*) — (0..1) Template | A Template can apply to many Devices, and a Device may have zero or one active Template. |
| 13 | Host ↔ HostIpAddressHistory | Host.hostID | HostIPAddressHistory.hostID | Host (1) —— (0..\*) HostIPAddressHistory | A host can appear in the IP Address history zero or more times. Each IP Address History entry applies to one host |

“Note: DhcpReservation has its own primary key, dhcpReservationId, not shown in the relationship table because it does not participate as a parent in any relationship.”

## 6.3 Physical Specialization Mapping

The relationship between the device supertype and switch and router subtypes are of particular significance. These relationships are enforced using primary and foreign key relationships.

Device has Device.deviceID as a primary key. The switch and router subtypes, in turn, have deviceId defined as both a primary and foreign key. The foreign key constraint references the deviceId attribute in the Device table.

* Router.deviceId is both a primary key and a foreign key referencing Device.deviceId.
* Switch.deviceId is both a primary key and a foreign key referencing Device.deviceId.

## 6.4 Interface Role Implementation

We will also note the structure of the Interface entity.

interfaceType is a constrained attribute with values ('LAN', 'WAN', 'MGMT').

Conditional NOT NULL and NULL rules enforce the conceptual specialization:

|  |  |  |
| --- | --- | --- |
| interfaceType | lanSubnetID | internetConnectionID |
| LAN | NOT NULL | NULL |
| WAN | NULL | NOT NULL |
| MGMT | NULL | NULL |

## 6.5 Subtype Discriminator

To implement the between Device and its subtypes Router and Switch, the **Device** table includes a discriminator attribute **deviceType**:

This attribute provides several important functions:

1. **Enforces disjointness**  
   Each Device must be exactly one of the valid categories. This makes the specialization logically explicit prior to examining subtype tables.
2. **Supports partial completeness**  
   Devices that are neither routers nor switches (e.g., firewalls, wireless controllers) can be recorded using the value 'Other'. If additional device specific attributes are required in the future, additional subtypes can be defined.
3. **Simplifies query logic**  
   Administrators can directly reference this attribute to return or filter by a device type.

This specialization is consistent with standard relational database design practices.

## 6.6 HostIpAddressHistory Table

The physical ERD has been updated to include the **HostIpAddressHistory** table. This table records every change to the Host.ipAddress attribute.

The physical design includes the following attributes:

* **hostIpAddressHistoryId** (PK) — Surrogate primary key generated via sequence
* **hostId** (FK) — References Host(hostId)
* **oldIpAddress** — Previous IP address value
* **newIpAddress** — Updated IP address value
* **changeDate** — Date of the IP address change

The relationship between Host and HostIpAddressHistory is implemented through this foreign key relationship:

**HostIpAddressHistory.hostId** → **Host.hostId**

This table allows the database to track the changes in IP address assignments over time. This allows administrators to be able to query to determine which host was assigned which IP address at specific periods in the past.

## 6.7 Physical Diagram

The following is a physical diagram depicting this topology. (Note this diagram is limited to the primary and foreign keys and attributes that define the relationships between entities. This database will also include entity specific attributes not depicted in this diagram.

A computer screen shot of a diagram

AI-generated content may be incorrect.

# Section 7: Attribute Reasoning

This section provides a list of the attributes associated with eight key entities: Device, Router, Switch, Interface, LanSubnet, DhcpScope, Host, and InternetConnection. Each entry includes datatype, constraints, nullability, justification, and examples.

## 7.1 Device

**Entity Description:** Represents a physical network device deployed at a site. This entity also serves as the supertype for Router and Switch entities.

**Attribute:** deviceId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key  
**NULL Status:** NOT NULL  
**Reasoning:** Surrogate key uniquely identifying each Device and serves as a stable parent key for subtype relations.  
**Example:** 2001

**Attribute:** siteId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → Site.siteId  
**NULL Status:** NOT NULL  
**Reasoning:** Identifies the site where the device is installed. This supports an organizational hierarchy and allows for identifying where the device physically resides  
**Example:** 12

**Attribute:** templateID  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → Template.templateID  
**NULL Status:** NULL  
**Reasoning:** This represents a template being applied to a device.  
**Example:** 5001

**Attribute:** name  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Human-readable identifier used in operations and troubleshooting.  
**Example:** "NYC-RTR-01"

**Attribute:** vendor  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Specifies the equipment manufacturer for maintenance, lifecycle planning, and vendor support.  
**Example:** "Cisco Meraki"

**Attribute:** model  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Indicates hardware series and feature capabilities.  
**Example:** "MX250"

**Attribute:** managementAddress  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** IP address used to reach the device for monitoring and configuration.  
**Example:** "10.4.0.5"

**Attribute:** serialNumber  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Unique device identifier used for tracking, warranty, and licensing.  
**Example:** "Q2MN-A7D3-Z9J4"

**Attribute:** firmwareVersion  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Indicates currently installed firmware version; optional during staging.  
**Example:** "17.10.3"

**Attribute:** installDate  
**Data Type:** DATE  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Date the device was deployed; used for lifecycle and refresh planning.  
**Example:** "2024-08-15"

**Attribute:** deviceType  
**Data Type:** VARCHAR(16)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Identifies the functional role of the device (Router, Switch, or Other) and serves as the subtype discriminator for the Device supertype. This supports the disjoint specialization with Router and Switch. This also simplifies querying and reporting by allowing the subtype to be determined without joining to subtype tables.  
**Example:** "Router"

## 7.2 Router

**Entity Description:** Subtype of Device representing routing appliances that provide WAN connectivity and routing functions.

**Attribute:** deviceId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key and Foreign Key → Device.deviceId  
**NULL Status:** NOT NULL  
**Reasoning:** Establishes the PK to FK specialization link between Router and Device.  
**Example:** 2001

**Attribute:** wanFailoverType  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Indicates configured WAN failover behavior such as load balancing or active-passive redundancy.  
**Example:** "Active-Passive"

**Attribute:** advancedSecurityEnabled  
**Data Type:** BIT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Specifies whether advanced security features (such as IDS/IPS, or Layer 7 filtering) are active.  
**Example:** 1

**Attribute:** maxThroughputMbps  
**Data Type:** INT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Specifies the router’s maximum routing throughput. This is important for circuit sizing and performance planning.  
**Example:** 1000

**Attribute:** supportsBgp  
**Data Type:** BIT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Indicates whether the device can participate in BGP routing for complex WAN and Internet topologies.  
**Example:** 0

**Attribute:** redundancyType  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Identifies the router’s high-availability configuration such as VRRP or HSRP.  
**Example:** "VRRP"

## 7.3 Switch

**Entity Description:** Subtype of Device used for LAN access and switching functions.

**Attribute:** deviceId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key and Foreign Key → Device.deviceId  
**NULL Status:** NOT NULL  
**Reasoning:** Implements the specialization relationship linking Switch to Device.  
**Example:** 2005

**Attribute:** poeSupported  
**Data Type:** BIT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Indicates whether the switch provides Power over Ethernet to endpoints.  
**Example:** 1

**Attribute:** poeBudgetWatts  
**Data Type:** INT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Total wattage available for PoE-powered devices; may not apply to all switch models.  
**Example:** 370

**Attribute:** stackable  
**Data Type:** BIT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Indicates whether the switch supports stacking for redundancy or management consolidation.  
**Example:** 1

**Attribute:** layer3Capable  
**Data Type:** BIT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Identifies whether the switch offers Layer 3 routing capabilities.  
**Example:** 0

## 7.4 Interface

**Entity Description:** Represents an interface on a Device. The role semantics (LAN, WAN, MGMT) are implemented through interfaceType and conditional foreign keys.

**Attribute:** interfaceId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key  
**NULL Status:** NOT NULL  
**Reasoning:** Unique surrogate key for identifying individual interfaces.  
**Example:** 501

**Attribute:** deviceId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → Device.deviceId  
**NULL Status:** NOT NULL  
**Reasoning:** Associates interface with its parent device.  
**Example:** 2001

**Attribute:** lanSubnetId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → LanSubnet.lanSubnetId  
**NULL Status:** NULL allowed; required when interfaceType = 'LAN'  
**Reasoning:** Ensures LAN-role interfaces have a corresponding subnet association.  
**Example:** 301

**Attribute:** internetConnectionId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → InternetConnection.internetConnectionId  
**NULL Status:** NULL allowed; required when interfaceType = 'WAN'  
**Reasoning:** Ensures WAN-role interfaces reference an internet circuit.  
**Example:** 7002

**Attribute:** name  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Label identifying the interface (e.g., GE0/1, WAN1).  
**Example:** "GE0/1"

**Attribute:** interfaceType  
**Data Type:** VARCHAR(16)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Indicates conceptual role of the interface: LAN, WAN, or MGMT.  
**Example:** "LAN"

**Attribute:** adminState  
**Data Type:** VARCHAR(16)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Administrative status of the interface (UP/DOWN).  
**Example:** "UP"

## 7.5 LanSubnet

**Entity Description:** Represents an individual LAN subnet within a site’s LAN supernet. This represents a specific network assigned to a group of hosts for a specific function.

**Attribute:** lanSubnetId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key  
**NULL Status:** NOT NULL  
**Reasoning:** Surrogate key uniquely identifying each LAN subnet for referencing by Interfaces and DHCP Scopes.  
**Example:** 301

**Attribute:** lanSupernetId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → LanSupernet.lanSupernetId  
**NULL Status:** NOT NULL  
**Reasoning:** Associates the LAN subnet with the parent LAN supernet assigned to a site.  
**Example:** 20

**Attribute:** subnetCidr  
**Data Type:** VARCHAR(32)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Stores the CIDR notation for the subnet, defining its IP address space.  
**Example:** "10.1.2.0/24"

**Attribute:** gatewayAddress  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Represents the default gateway assigned within the subnet, typically the router interface address.  
**Example:** "10.1.2.1"

**Attribute:** vlanId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Optional VLAN tag used when the subnet is associated with a specific Layer 2 VLAN  
**Example:** 20

**7.6 DhcpScope**

**Entity Description:** Defines the IP Address range of a DHCP scope within a LAN subnet.

**Attribute:** dhcpScopeId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key  
**NULL Status:** NOT NULL  
**Reasoning:** Unique surrogate identifier for DHCP scope records.  
**Example:** 410

**Attribute:** lanSubnetId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → LanSubnet.lanSubnetId  
**NULL Status:** NOT NULL  
**Reasoning:** Associates the DHCP scope with the subnet it resides in.  
**Example:** 301

**Attribute:** startAddress  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Defines the first IP address issued in the scope range.  
**Example:** "10.1.2.50"

**Attribute:** endAddress  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Defines the final IP address issued in the scope range.  
**Example:** "10.1.2.200"

**Attribute:** leaseTimeMinutes  
**Data Type:** INT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Sets the default DHCP lease duration for issued IP addresses.  
**Example:** 1440

**7.7 Host**

**Entity Description:** Represents an endpoint device connected to an Interface. May receive DHCP reservations.

**Attribute:** hostId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key  
**NULL Status:** NOT NULL  
**Reasoning:** Unique surrogate identifier for host records.  
**Example:** 9001

**Attribute:** interfaceId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Foreign Key → Interface.interfaceId  
**NULL Status:** NOT NULL  
**Reasoning:** Ensures each host is connected to exactly one interface, enforcing network topology.  
**Example:** 501

**Attribute:** hostName  
**Data Type:** VARCHAR(128)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Names the host for inventory, management, or DNS purposes.  
**Example:** "LAB-PC-01"

**Attribute:** macAddress  
**Data Type:** VARCHAR(32)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Captures the layer-2 identity of the host. These are used for DHCP reservations. These also uniquely identify an interface on a device.  
**Example:** "AA:BB:CC:DD:EE:FF"

**Attribute:** ipAddress  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Represents the host’s IP address if statically assigned or if retrieved from DHCP.  
**Example:** "10.1.2.57"

**Attribute:** assetTag  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Optional field used for corporate asset management and tracking.  
**Example:** "CH-WS-44322"

**7.8 InternetConnection**

Entity Description: Represents a WAN circuit (ISP-provided connection) used by one or more WAN-role Interfaces.

**Attribute:** internetConnectionId  
**Data Type:** INT  
**Primary / Foreign Key Status:** Primary Key  
**NULL Status:** NOT NULL  
**Reasoning:** Unique surrogate identifier for each internet connection record.  
**Example:** 7002

**Attribute:** provider  
**Data Type:** VARCHAR(128)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Identifies the ISP supplying the WAN circuit.  
**Example:** "NextHop Communications"

**Attribute:** bandwidthMbps  
**Data Type:** INT  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Specifies contracted download/upload bandwidth.  
**Example:** 500

**Attribute:** circuitId  
**Data Type:** VARCHAR(128)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** ISP-provided circuit identifier used for troubleshooting and support.  
**Example:** "NH-556344-A"

**Attribute:** connectionCIDR  
**Data Type:** VARCHAR(64)  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NOT NULL  
**Reasoning:** Represents the assigned public WAN address block in CIDR notation. Required for routing, NAT, and WAN configuration.  
**Example:** "203.0.113.0/30"

**Attribute:** installDate  
**Data Type:** DATE  
**Primary / Foreign Key Status:** Neither  
**NULL Status:** NULL allowed  
**Reasoning:** Date the WAN circuit was provisioned or activated.  
**Example:** "2023-07-10"

# Section 8: Normalization Reasoning

This section analyzes the BlueSky-IPAM schema using the principles of normalization. Complex entities are evaluated step-by-step through UNF, 1NF, 2NF, 3NF, and BCNF. To avoid repetition, remaining entities are assessed using high-level BCNF justification. The goal is to confirm that all relations meet Boyce–Codd Normal Form for consistency, integrity, and minimal redundancy.

## 8.1 Overview of Normalization Approach

Normalization ensures that each relation contains only attributes that depend on the key, the whole key, and nothing but the key. Within this database the following statements are true:

* Surrogate primary keys are used consistently
* Subtypes inherit keys from Device using PK=FK
* Attribute sets have been evaluated for partial and transitive dependencies
* Conditional relationships in Interface do not violate BCNF
* DHCP- and subnet-related entities have been reviewed for atomicity and dependency alignment

The remainder of this section presents normalization analyses for the key entities.

## 8.2 Detailed Normalization for Complex Entities

8.2.1 Device

The Device table contains many attributes and serves as a supertype. We will thus perform a full normalization analysis.

**UNF:**  
Attributes are atomic, but potential issues include multiple attributes describing device characteristics, and a possible dependency between vendor and model.

**1NF:**  
All attributes are atomic. There are no repeating groups or multivalued attributes.

**2NF:**  
We have added deviceId as a surrogate key, and no composite key exists. All non-key attributes depend on deviceId, and not on a subset of a composite key.  
Therefore Device satisfies 2NF.

**3NF:**  
A potential transitive dependency might exist if models functionally determine vendor. However, models vary by vendor. The vendor cannot be derived from model alone in this context. No attribute depends on another non-key attribute.

**BCNF:**  
All attributes depend solely on deviceId.  
**Conclusion:**  
Device is in BCNF.

8.2.2 Interface

Interface has multiple optional foreign keys and conditional integrity rules, making it a strong candidate for step-by-step normalization.

**UNF:**  
No repeating groups exist.

**1NF:**  
All attributes are atomic. Foreign key attributes (lanSubnetId, internetConnectionId) contain single values.

**2NF:**  
The primary key is interfaceId, and not a composite key. No partial dependencies are possible.

**3NF:**  
Potential transitive dependencies (e.g., lanSubnetId → subnetCidr → other attributes) do not exist within Interface itself.  
No attribute determines another non-key attribute within this entity.

**BCNF:**  
All attributes depend solely on interfaceId and not each other, and conditional rules do not introduce BCNF violations.

**Conclusion:**  
Interface is in BCNF.

8.2.3 Host

Host includes attributes that can suggest transitive or derived relationships (e.g., ipAddress assigned from DHCP). Host is a good candidate to evaluate explicitly.

**UNF:**  
All Host attributes are atomic. There are no repeating groups.

**1NF:**  
All values are single-valued fields only. There are no lists of IP addresses or MAC addresses.

**2NF:**  
The primary key is hostId, a surrogate key. No composite key exists, and so there are no partial dependencies.

**3NF:**  
A potential source of concern might be the ipAddress attribute depending on interfaceId. However, Host.ipAddress is not functionally determined by interfaceId in all cases (DHCP leases can change. It is also possible to statically assign IP addresses to a host). Therefore no transitive dependency exists.

**BCNF:**  
All attributes depend only on hostId.

**Conclusion:**  
Host is in BCNF.

8.2.4 DhcpScope

DhcpScope contains a range of IP addresses (from startAddress to endAddress) and references a subnet. These require careful evaluation for dependencies.

**UNF:**

All attributes are properly atomic.

**1NF:**

Both startAddress and endAddress are single values. There are no repeating structures.

**2NF:**

The Primary key is dhcpScopeId, and there is no composite key. There are no partial dependencies.

**3NF:**

No attribute determines another non-key attribute. startAddress does not determine endAddress.

lanSubnetId is a foreign key but this key does not determine any of the attributes of the scope.

**BCNF:**

All attributes depend only on dhcpScopeId.

**Conclusion:**

DhcpScope is in BCNF.

## 8.3 High-Level Normalization for Remaining Entities

The following entities naturally satisfy BCNF due to their structure, use of surrogate keys, or specialization pattern.

**Organization**

The Surrogate key (organizationId) ensures all attributes depend solely on the key. There are no derived attributes. Organization is in BCNF.

**Site**

This is a simple entity with attributes depending only on siteId. There are no dependency issues. Site is in BCNF.

**LanSupernet**

The attributes depend solely on lanSupernetId and do not derive from each other. LanSupernet is in BCNF.

**LanSubnet**

The SubnetCidr, gatewayAddress, and vlanId attributes depend only on lanSubnetId. LanSubnet is in BCNF. A gatewayAddress is not dependent on the Cidr since a gateway IP address of a subnet could be any valid IP address within that subnet. **DhcpReservation**

All attributes are fully functionally dependent on reservationId, and the reference to Host does not introduce any additional dependencies. DHCPReservation is in BCNF.

**Router (Subtype)**

deviceId is both PK and FK to Device. Subtype-only attributes depend entirely on deviceId. Specialization ensures Router is in BCNF.

**Switch (Subtype)**

deviceId is both PK and FK to Device. Subtype-only attributes depend entirely on deviceId. Specialization ensures Switch is in BCNF.

**InternetConnection**

All attributes depend on internetConnectionId. There are no transitive or partial dependencies. InternetConnection is in BCNF.

**Template**

templateId uniquely determines templateName and related settings. There are no further dependencies. Template is in BCNF.

## 8.4 Overall Schema Normal Form Assessment

Overall, the BlueSky-IPAM schema satisfies Boyce–Codd Normal Form for the following reasons:

* All relations use surrogate keys.
* Subtype entities use PK=FK specialization patterns, guaranteeing BCNF compliance.
* Conditional foreign key rules in Interface do not create functional dependencies that violate BCNF.
* No attributes in the schema are derivable from non-key attributes.
* All many-to-one and one-to-many relationships are implemented using foreign keys rather than embedded attributes.

The schema is therefore fully normalized to BCNF.

## 8.5 Normalization Summary Table

The following is a summary of this information.

|  |  |  |
| --- | --- | --- |
| Entity | Normal Form | Rationale (Summary) |
| Device | BCNF | Device is in BCNF because every attribute depends entirely on deviceId and no transitive dependencies exist. |
| Router | BCNF | Router is in BCNF because the PK=FK specialization ensures that all subtype attributes depend solely on deviceId. |
| Switch | BCNF | Switch is in BCNF because the PK=FK specialization guarantees that all subtype attributes depend only on deviceId. |
| Interface | BCNF | Interface is in BCNF because all attributes depend on interfaceId, and the conditional foreign key rules do not introduce any additional dependencies. |
| Host | BCNF | Host is in BCNF because all attributes depend on hostId and there are no partial or transitive dependencies. |
| DhcpScope | BCNF | DhcpScope is in BCNF because start and end address attributes depend only on dhcpScopeId and do not depend on each other. |
| LanSubnet | BCNF | LanSubnet is in BCNF because all subnet attributes depend exclusively on lanSubnetId. |
| InternetConnection | BCNF | InternetConnection is in BCNF because every attribute depends entirely on internetConnectionId. |
| Organization | BCNF | Organization is in BCNF because all attributes depend on organizationId and no derived attributes are present. |
| Site | BCNF | Site is in BCNF because all attributes depend solely on siteId and no additional dependencies exist. |
| Template | BCNF | Template is in BCNF because templateId uniquely determines all template attributes and there are no internal dependencies. |
| DhcpReservation | BCNF | DhcpReservation is in BCNF because all attributes depend solely on reservationId and the relationship to Host does not create additional dependencies. |

# Section 9: SQL Script (revised)

## Primary Screenshots

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Supplemental Screenshots

The following are supplemental screenshots that demonstrate that each sequence block executes successfully. They successfully cover the following stages of the script;

* The DROP TABLE statements to drop the existing tables
* The DROP SEQUENCE statements to drop the existing sequences
* The CREATE TABLE statements to create the required tables
* The CREATE SEQUENCE statements to create the required sequences
* The ALTER TABLE statements to apply the sequence to the primary key of each table

Additional validation commands can be found in Appendix 2.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Section 10: Transactions

## 10.1 Overview

In this section, we define three reusable stored procedures that implement the data-creation workflows specified in **Section 3 Use Cases**.

* Create New Organization
* Create New Site With LAN Supernet
* Create New Subnet With DHCP Scope

(Additionally, we added a new basic use case “Create New Organization” in section 3 to support this for completeness).

These stored procedures will be parameterized. All inserts will use sequence-generated surrogate keys, and no primary or foreign key values will be hardcoded. Transactions ensure that the database transitions between valid states without leaving behind partial or inconsistent data.

10.2 Transaction for Create New Organization

Use Case 0 describes the scenario in which a new business organization is introduced into the IPAM system. Organizations are the top-level entities in the BlueSky hierarchy and serve as the parent for all Sites. While this is an infrequent operation, creating an organization is a prerequisite for subsequent activities such as site provisioning and subnet assignment.

**Design Overview.**  
The stored procedure **AddOrganization** generates a new surrogate key using the Organization\_seq sequence and inserts a row into the Organization table using the supplied attributes. This procedure allows administrators to introduce new organizations without relying on hardcoded identifiers.

**Atomicity.**  
Although single-row inserts are naturally simple, they are still executed within an explicit transaction block. This ensures consistent behavior and keeps the organizational hierarchy synchronized with future operations that depend on its presence.

10.2.1 Stored Procedure Definition

**Stored Procedure: AddOrganization**

CREATE OR ALTER PROCEDURE AddOrganization

@organizationName VARCHAR(255),

@organizationDescription VARCHAR(500) = NULL

AS

BEGIN

SET NOCOUNT ON;

DECLARE @newOrgId INT;

SET @newOrgId = NEXT VALUE FOR Organization\_seq;

INSERT INTO Organization (

organizationId,

name,

description

)

VALUES (

@newOrgId,

@organizationName,

@organizationDescription

);

END;

GO

10.2.2 Execution Example

BEGIN TRANSACTION CreateOrganization;

EXEC AddOrganization

@organizationName = 'BlueSkyCorporate',

@organizationDescription = 'Primary global organization for BlueSky operations';

COMMIT TRANSACTION CreateOrganization;

10.2.3 Screenshots

The following are screenshots demonstrating the following:

* Creating the AddOrganization stored procedure
* Executing it with an example case
* Viewing the results.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

10.3 Transaction for Use Case 1 – Create New Site With LAN Supernet

**Use Case Summary (Section 3).**  
Use Case 1 describes provisioning and allocating a supernet to a new Site. When a Site is created, it must be associated with an existing Organization. Additionally, a /16 LAN Supernet must be allocated to define the address space for that Site. The Site and its associated Supernet must be inserted as a single atomic operation.

**Design Overview.**  
The stored procedure **AddSiteWithSupernet** inserts a new row into Site using a sequence-generated siteId. It also inserts a corresponding row into LanSupernet using a sequence-generated lanSupernetId. The caller provides the business attributes (organization name, site name, region, location, and supernet CIDR). The organizationId is obtained by lookup rather than hardcoding.

**Atomicity.**  
The Site and Supernet must be created together; therefore, the stored procedure call is wrapped in an explicit BEGIN TRANSACTION / COMMIT TRANSACTION block. If either insert fails, the entire transaction can be rolled back.

10.3.1 Stored Procedure Definition

**Stored Procedure: AddSiteWithSupernet**

CREATE OR ALTER PROCEDURE AddSiteWithSupernet

@organizationName VARCHAR(255),

@siteName VARCHAR(100),

@region VARCHAR(50) = NULL,

@location VARCHAR(255) = NULL,

@supernetCidr VARCHAR(50),

@supernetDesc VARCHAR(255) = NULL

AS

BEGIN

SET NOCOUNT ON;

DECLARE @organizationId INT;

DECLARE @newSiteId INT;

DECLARE @newLanSupernetId INT;

-- Lookup OrganizationId based on name

SELECT @organizationId = organizationId

FROM Organization

WHERE name = @organizationName;

IF @organizationId IS NULL

BEGIN

RAISERROR('Organization not found: %s', 16, 1, @organizationName);

RETURN;

END

-- Generate surrogate keys

SET @newSiteId = NEXT VALUE FOR Site\_seq;

SET @newLanSupernetId = NEXT VALUE FOR LanSupernet\_seq;

-- Insert the new Site

INSERT INTO Site (

siteId,

organizationId,

name,

region,

location

)

VALUES (

@newSiteId,

@organizationId,

@siteName,

@region,

@location

);

-- Insert the associated LAN Supernet

INSERT INTO LanSupernet (

lanSupernetId,

siteId,

cidr,

description

)

VALUES (

@newLanSupernetId,

@newSiteId,

@supernetCidr,

@supernetDesc

);

END;

GO

10.3.2 Execution Example

**Execution Example (Use Case 1)**

BEGIN TRANSACTION CreateNewSite;

EXEC AddSiteWithSupernet

@organizationName = 'BlueSkyCorporate',

@siteName = 'Sydney Office',

@region = 'APAC',

@location = 'Sydney, Australia',

@supernetCidr = '10.20.0.0/16',

@supernetDesc = 'Sydney primary LAN supernet';

COMMIT TRANSACTION CreateNewSite;

10.3.3 Screenshots

The following are screenshots demonstrating the following:

* Creating the AddSitewithSupernet stored procedure
* Executing it with an example case
* Viewing the results.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 10.4 Transaction for Use Case 2 – Create New Subnet With DHCP Scope

Use Case 2 creates a new LAN Subnet within an existing Supernet belonging to a site. Some subnets may use DHCP scopes and others be not have DHCP scopes. This transaction supports both behaviors:

1. Create a subnet only
2. Create a subnet together with a DHCP scope.

**Design Overview.**  
The stored procedure **AddSubnet** accepts the organization name and site name to ensure correct multi-organization behavior. It resolves the appropriate LAN Supernet, creates the LAN Subnet, and conditionally creates a DHCP Scope when DHCP parameters are provided. Surrogate keys are generated using LanSubnet\_seq and DhcpScope\_seq.

**Optional DHCP Logic.**  
A DHCP scope is created **only if both** @startAddress and @endAddress are provided.  
If DHCP parameters are omitted, the subnet is still created successfully.

**Atomicity.**  
The procedure supports multi-step creation while allowing callers to wrap the entire operation in an external transaction. All inserts belonging to a single operation are executed together, ensuring the database remains consistent.

10.4.1 Stored Procedure Definition

**Stored Procedure: AddSubnet**

CREATE OR ALTER PROCEDURE AddSubnet

@organizationName VARCHAR(255),

@siteName VARCHAR(255),

@subnetCidr VARCHAR(32),

@vlanId INT = NULL,

@gatewayAddress VARCHAR(64),

@startAddress VARCHAR(64) = NULL, -- Optional

@endAddress VARCHAR(64) = NULL, -- Optional

@leaseTimeMinutes INT = NULL -- Optional, required only if DHCP is created

AS

BEGIN

SET NOCOUNT ON;

DECLARE @organizationId INT;

DECLARE @lanSupernetId INT;

DECLARE @newLanSubnetId INT;

DECLARE @newDhcpScopeId INT;

-- Resolve Organization ID from name

SELECT @organizationId = organizationId

FROM Organization

WHERE name = @organizationName;

IF @organizationId IS NULL

BEGIN

RAISERROR('Organization not found: %s', 16, 1, @organizationName);

RETURN;

END

-- Resolve LAN Supernet associated with this organization's site

SELECT @lanSupernetId = ls.lanSupernetId

FROM LanSupernet AS ls

JOIN Site AS s ON s.siteId = ls.siteId

WHERE s.name = @siteName

AND s.organizationId = @organizationId;

IF @lanSupernetId IS NULL

BEGIN

RAISERROR('No LAN Supernet found for site %s in organization %s',

16, 1, @siteName, @organizationName);

RETURN;

END

-- Generate surrogate key for LAN Subnet

SET @newLanSubnetId = NEXT VALUE FOR LanSubnet\_seq;

-- Insert LAN Subnet

INSERT INTO LanSubnet (

lanSubnetId,

lanSupernetId,

subnetCidr,

vlanId,

gatewayAddress

)

VALUES (

@newLanSubnetId,

@lanSupernetId,

@subnetCidr,

@vlanId,

@gatewayAddress

);

-- Optional DHCP Scope creation

IF @startAddress IS NOT NULL AND @endAddress IS NOT NULL

BEGIN

IF @leaseTimeMinutes IS NULL

BEGIN

RAISERROR('leaseTimeMinutes must be provided when creating a DHCP scope.',

16, 1);

RETURN;

END

-- Generate surrogate key for DHCP Scope

SET @newDhcpScopeId = NEXT VALUE FOR DhcpScope\_seq;

INSERT INTO DhcpScope (

dhcpScopeId,

lanSubnetId,

startAddress,

endAddress,

leaseTimeMinutes

)

VALUES (

@newDhcpScopeId,

@newLanSubnetId,

@startAddress,

@endAddress,

@leaseTimeMinutes

);

END

END;

GO

10.4.2 Execution Example

BEGIN TRANSACTION CreateNewSubnet;

EXEC AddSubnet

@organizationName = 'BlueSkyCorporate',

@siteName = 'Sydney Office',

@subnetCidr = '10.20.10.0/24',

@vlanId = 10,

@gatewayAddress = '10.20.10.1',

@startAddress = '10.20.10.50',

@endAddress = '10.20.10.200',

@leaseTimeMinutes = 2880;

COMMIT TRANSACTION CreateNewSubnet;

10.4.3 Screenshots

The following are screenshots demonstrating the following:

* Creating the AddSubnet stored procedure
* Executing it with an example case
* Viewing the results.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 10.5 Summary

The stored procedures defined in this section implement the workflows defined in Use Case 0, Use Case 1, and Use Case 2. This supports the creation of Organizations, Sites, LAN Supernets, LAN Subnets, and their optional DHCP Scopes. Each procedure uses sequence-generated surrogate keys and performs internal lookups to avoid hardcoded identifiers. This ensures atomic, consistent updates to the database. These procedures provide a robust framework for populating and maintaining the hierarchical IP addressing structure used by the BlueSky IPAM database.

# Section 11: Questions and Queries

11.0 Synthetic Data Overview and Rationale

To support the analytical requirements in this section, a comprehensive synthetic dataset was created and is included in **Appendix 3**. We created and iteratively revised this dataset until it was appropriate to operate as a proof of concept. The dataset was designed to reflect realistic operational conditions for a multi-site enterprise network and to fully exercise the relational structure established during prior iterations. The sample data aligns with the following goals:

11.0.1 Realistic Topology Representation

The dataset models five organizational sites: **Boston HQ, New York Office, Dublin DC, São Paulo Office, and Sydney Office**. Each site was provisioned using the stored procedures developed in Section 10, ensuring that all inserts follow the same business rules used by real deployments.  
  
Boston and Sydney were selected as the primary sites for host-level analysis, and these contain both routers and access switches along with multiple LAN subnets.

11.0.2 Consistency with Physical and Logical Architecture

To maintain architectural integrity, hosts are connected to **switch interfaces** reflecting real-world design. Router interfaces are limited to WAN and LAN gateway functions, while switches were provisioned with individual user-facing ports (port1, port2, etc.) mapped to the appropriate LAN subnets. This aligns with the constraints that were defined and modeled earlier in this document.

**11.0.3 Business Rules Enforcement**

All data was inserted using the following validated stored procedures:

* **AddOrganization**
* **AddSiteWithSupernet**
* **AddSubnet**

These procedures enforce:

* Valid site-to-supernet relationships
* Proper VLAN and gateway assignments
* Optional DHCP scope creation
* Organization name lookup
* Appropriate referential integrity constraints

Where direct inserts were required (e.g., devices, specializations, interfaces, hosts), variables were used to capture primary key values. This improves the clarity of these queries and removes the need to multiple redundant queries, ensuring a clean and deterministic seed environment.

**11.0.4 Support for Analytical Diversity**

The data was designed so that analytical queries can demonstrate:

* Multi-site comparisons
* Host-to-interface mappings
* Subnet utilization patterns
* Device specialization visibility (router vs switch)
* Relationships between devices, interfaces, DHCP scopes, and hosts
* WAN connectivity at the site and device level
* Template inheritance

Each query in Section 11 will be validated against this dataset to ensure it produces non-trivial, meaningful results.

## 11.1 Overview of Analytical Query Requirements

The analytical queries in this section demonstrate the ability to retrieve and correlate information across the full functionality of the BlueSky IPAM database. The goal is not only to produce correct SQL statements, but also to show how the physical design supports efficient access to operational data.  
  
We will choose queries that:

1. Retrieve information from multiple related tables.
2. Use joins that reflect the logical relationships between the entities described by our database.
3. Provide meaningful results when executed against our synthetic dataset (provided in Appendix 3).
4. Demonstrate the completeness and correctness of the physical implementation.

These will be representative of real world use of the database.

## 11.2 Analytical Query 1 – Host and Interface Inventory Across Sites

11.2.1 Business Question and Context:

**“Which hosts are active in the network, which switch ports are they connected to, and which LAN subnet does each host belong to?”**

Purpose and Rationale

To answer this question, we will create a global inventory of hosts across sites. This query provides a list of each host, it’s physical connection points and its IP address. The query ties together **user identity (host), physical interface mapping (switch ports), and infrastructure topology (LAN subnets and sites).**

In a real-world setting this report could be used by IT administrators for:

* Troubleshooting
* Move/add/change operations
* Security investigations
* Asset tracking

Requirement review

For the purposes of the assignment, the requirement states that the query must join at least four tables via associative relationships.  
This query joins **six**:

1. Host
2. Interface
3. Device (switch)
4. LanSubnet
5. Site
6. Organization

11.2.2 SQL Query Body

SELECT

Organization.name AS organizationName,

Site.name AS siteName,

Host.hostName,

Host.macAddress,

Host.assetTag,

Host.ipAddress,

Device.name AS switchName,

Interface.name AS switchPort,

LanSubnet.subnetCidr AS lanSubnet

FROM Host

JOIN Interface

ON Host.interfaceId = Interface.interfaceId

JOIN Device

ON Interface.deviceId = Device.deviceId

JOIN LanSubnet

ON Interface.lanSubnetId = LanSubnet.lanSubnetId

JOIN Site

ON Device.siteId = Site.siteId

JOIN Organization

ON Site.organizationId = Organization.organizationId

ORDER BY

Site.name,

Host.hostName;

11.2.3 Screenshots

The following are examples of this query running.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 11.3 Analytical Query 2 – Device Specialization Summary

11.3.1 Business Question and Context:

**“What devices exist at each site, and what are their specialization-specific attributes (router vs. switch)?”**

Purpose and Rationale

To answer this question, we will create a global inventory of our devices. We will include attributes common to all devices as well as attributes that are specific to routers and switches.

In a real-world setting, a network administrator can use this report for tasks such as the following:

* Network capacity planning
* Security analysis
* Hardware lifecycle and licensing management
* WAN vs LAN role identification

Requirement Review

For the purposes of the assignment, the requirement states that the query must retrieve data from the supertype AND one or more subtypes.

This query joins:

* Device (supertype)
* Router (subtype)
* Switch (subtype)
* Site

## 11.3.2 SQL Query Body

SELECT

Device.deviceId,

Device.name,

Device.deviceType,

Site.name AS SiteName,

Router.wanFailoverType,

Router.advancedSecurityEnabled,

Router.maxThroughputMbps,

Router.supportsBgp,

Router.redundancyType,

Switch.poeSupported,

Switch.poeBudgetWatts,

Switch.stackable,

Switch.layer3Capable

FROM Device

JOIN Site

ON Device.siteId = Site.siteId

LEFT JOIN Router

ON Device.deviceId = Router.deviceId

AND Device.deviceType = 'Router'

LEFT JOIN Switch

ON Device.deviceId = Switch.deviceId

AND Device.deviceType = 'Switch'

ORDER BY Device.deviceType, Device.deviceId;

## 11.3.3 Screenshots

The following are examples of this query running.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 11.4 Analytical Query 3 – Subnet Utilization View and Query

11.4.1 Business Question and Context:

**“Which subnets have the highest host utilization across the organization?”**

Purpose and Rationale

To answer this question, we will create a view listing each subnet and its utilization. This query shows how full each LAN subnet is used by counting the number of connected hosts.

In a real-world setting, a network administrator can use this report for tasks such as the following:

* Capacity planning
* DHCP scope sizing
* Network expansion planning
* Subnet consolidation or split decisions

Additionally, this can be useful in determining other metrics such as the percent utilization of the DHCP scope or how many IP addresses are free within each scope. To provide these metrics an application could gather the prerequisite source data from this database, though the calculations themselves would need to be implemented within the application logic. There are modules, though that readily facilitate such analysis. As an example, python 3 natively includes a module that can be used to perform this analysis:  
<https://docs.python.org/3/library/ipaddress.html>

Requirement Review

For the purposes of the assignment, this query meets the requirements in the following ways.

This query uses a **view**, and includes constructs from both required groups:

**Group 1:**

* Join of multiple tables
* WHERE restriction optional
* ORDER BY included

**Group 2:**

* Aggregate function: COUNT(Host.hostId)
* HAVING clause used
* Could optionally use a LEFT JOIN to show empty subnets

## 11.4.2 SQL View Body

CREATE VIEW vSubnetHostCounts AS

SELECT

Site.name AS siteName,

LanSubnet.subnetCidr,

COUNT(Host.hostId) AS hostCount

FROM LanSubnet

JOIN LanSupernet

ON LanSubnet.lanSupernetId = LanSupernet.lanSupernetId

JOIN Site

ON LanSupernet.siteId = Site.siteId

LEFT JOIN Interface

ON Interface.lanSubnetId = LanSubnet.lanSubnetId

LEFT JOIN Host

ON Host.interfaceId = Interface.interfaceId

GROUP BY

Site.name,

LanSubnet.subnetCidr;

11.4.3 SQL Query Body Using the View

SELECT

vSubnetHostCounts.siteName,

vSubnetHostCounts.subnetCidr,

vSubnetHostCounts.hostCount

FROM vSubnetHostCounts

WHERE vSubnetHostCounts.hostCount > 0

ORDER BY vSubnetHostCounts.hostCount DESC;

## 11.3.4 Screenshots

The following are examples of this view and query running.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 11.5 Conclusion

The analytical queries in this section demonstrate that the BlueSky IPAM / Network Infrastructure Database is fully capable of supporting operational reporting, troubleshooting, and capacity planning tasks commonly performed by network and systems administrators. By designing a synthetic dataset that reflects realistic multi-site conditions, the queries are able to produce meaningful, non-trivial results that validate both the physical database structure, and the business rules the database is designed to support.

The first query illustrated creating an inventory of devices. This is achieved by combining information from hosts, switch interfaces, devices, LAN subnets, and sites. This report allows IT staff to identify where hosts and devices reside within the network.   
  
The second query creates an inventory of devices in the network. It shows the routers and switches present in the network and the attributes that are applicable to each device type.

The third query introduced a reusable view to show subnet utilization. This facilitates proactive analysis and management of the utilization of each subnet. This allows for ongoing capacity assessments and future network planning.

Together, these queries confirm that the schema, constraints, stored procedures, and stored data all function to produce reliable, actionable insights to the status and utilization of the organization’s global network.

# Section 12: Indexes

## 12.0 Indexing Strategy

This section identifies the columns in the BlueSky IPAM database that require or will benefit from the creation of indexes.

The following are the principles governing the best practice for index selection:

1. Columns that are defined as **primary keys** are automatically indexed.
2. Columns that serve as **foreign keys** ARE NOT automatically indexed, but it is a best practice to create an index including them.
3. Columns that are commonly used are also best practices to include

We will choose three columns to index that are selected based on thequeries defined in Section 11.

This ensures that the system is optimized for workloads that we anticipate the database to handle. Note: This is not an exhaustive list and we may choose to index additional columns in the future.

## 12.1 Primary Keys

Primary keys are indexed automatically by SQL Server and require no additional configuration.  
The following tables have primary keys that are already indexed:

* **Organization**(organizationId)
* **Site**(siteId)
* **LanSupernet**(lanSupernetId)
* **LanSubnet**(lanSubnetId)
* **DhcpScope**(dhcpScopeId)
* **Device**(deviceId)
* **Router**(deviceId) — PK and FK to Device
* **Switch**(deviceId) — PK and FK to Device
* **Interface**(interfaceId)
* **Host**(hostId)
* **InternetConnection**(internetConnectionId)
* **DhcpReservation**(dhcpReservationId)
* **Template**(templateId)

These indexes guarantee efficient row-level access and maintain entity uniqueness throughout the database.

## 12.2 Foreign Keys Requiring Indexes

It is a strongly recommended practice to index foreign keys. This ensures efficient joins and allows SQL Server to enforce referential integrity without full table scans.

The following foreign key columns must have indexes:

| **Table** | **Foreign Key Column** | **References** |
| --- | --- | --- |
| Site | organizationId | Organization |
| LanSupernet | siteId | Site |
| LanSubnet | lanSupernetId | LanSupernet |
| DhcpScope | lanSubnetId | LanSubnet |
| Device | siteId | Site |
| Device | templateId | Template |
| Router | deviceId | Device |
| Switch | deviceId | Device |
| Interface | deviceId | Device |
| Interface | lanSubnetId | LanSubnet |
| Interface | internetConnectionId | InternetConnection |
| Host | interfaceId | Interface |
| DhcpReservation | dhcpScopeId | DhcpScope |
| DhcpReservation | hostId | Host |

## 12.2.1 SQL Index Creation Statements

The following are SQL statements to generate these indexes.

-- Site → Organization

CREATE NONCLUSTERED INDEX IX\_Site\_OrganizationId

ON Site (organizationId);

-- LanSupernet → Site

CREATE NONCLUSTERED INDEX IX\_LanSupernet\_SiteId

ON LanSupernet (siteId);

-- LanSubnet → LanSupernet

CREATE NONCLUSTERED INDEX IX\_LanSubnet\_LanSupernetId

ON LanSubnet (lanSupernetId);

-- DhcpScope → LanSubnet

CREATE NONCLUSTERED INDEX IX\_DhcpScope\_LanSubnetId

ON DhcpScope (lanSubnetId);

-- Device → Site

CREATE NONCLUSTERED INDEX IX\_Device\_SiteId

ON Device (siteId);

-- Device → Template

CREATE NONCLUSTERED INDEX IX\_Device\_TemplateId

ON Device (templateId);

-- Router → Device

CREATE NONCLUSTERED INDEX IX\_Router\_DeviceId

ON Router (deviceId);

-- Switch → Device

CREATE NONCLUSTERED INDEX IX\_Switch\_DeviceId

ON Switch (deviceId);

-- Interface → Device

CREATE NONCLUSTERED INDEX IX\_Interface\_DeviceId

ON Interface (deviceId);

-- Interface → LanSubnet

CREATE NONCLUSTERED INDEX IX\_Interface\_LanSubnetId

ON Interface (lanSubnetId);

-- Interface → InternetConnection

CREATE NONCLUSTERED INDEX IX\_Interface\_InternetConnectionId

ON Interface (internetConnectionId);

-- Host → Interface

CREATE NONCLUSTERED INDEX IX\_Host\_InterfaceId

ON Host (interfaceId);

-- DhcpReservation → DhcpScope

CREATE NONCLUSTERED INDEX IX\_DhcpReservation\_DhcpScopeId

ON DhcpReservation (dhcpScopeId);

-- DhcpReservation → Host

CREATE NONCLUSTERED INDEX IX\_DhcpReservation\_HostId

ON DhcpReservation (hostId);

## 12.3 Query-Driven Indexes

We will now choose three additional non-key columns to index. These are tied to workload patterns from Section 11.

## 12.3.1 Index 1 Supporting Query 1 (Host and Interface Inventory)

We will index this column Interface.Name.

**Justification:**

Interface names such as “port1”, “port2”, etc., are used to identify access ports and join hosts to devices. This index directly supports the join pattern used to map hosts to their switch ports. Although the query does not filter on this column, the interface table is accessed repeatedly during the Host → Interface → Device join sequence. Indexing Interface.name helps SQL Server retrieve the appropriate interface rows more efficiently, improving overall join performance. Additionally, sorting or filtering by these names will be common in operational reporting.

**SQL Index Creation Statement:**

CREATE NONCLUSTERED INDEX IX\_Interface\_Name

ON Interface (name);

12.3.2 Index 2 Supporting Query 2 (Device Specialization Report)

We will index this column Device.Name.

**Justification:**

Device names (such as BOS-MX100-1 and SYD-MS225-1) are frequently used for administrative lookups and appear in the ORDER BY clause of this query. Additionally, although the joins rely on primary and foreign keys, the device name is commonly used as a natural identifier in operational reports. Indexing this column improves responsiveness when retrieving device information or sorting results by device name, which would be common when comparing multiple devices across different sites. Because this column is not a primary key or foreign key but is frequently referenced in reporting, it is an appropriate choice for an index.

**SQL Index Creation Statement:**CREATE NONCLUSTERED INDEX IX\_Device\_Name

ON Device (name);

12.3.3 Index 3 Supporting Query 3 (Subnet Utilization View)

We will index this column LanSubnet.subnetCidr.

**Justification:**

For Query 3, the subnet utilization view groups results by the attribute LanSubnet.subnetCidr, making this column central to the aggregation and reporting logic. The subnet CIDR also acts as a natural record for identifying and comparing network segments. Filtering or displaying results by subnet is a common task for network administrators performing IP address management. Indexing subnetCidr improves performance for both grouping and lookup operations in the utilization query. This makes it a strong candidate for a query-driven index that is not tied to a primary or foreign key.

**SQL Index Creation Statement:**

CREATE NONCLUSTERED INDEX IX\_LanSubnet\_SubnetCidr

ON LanSubnet (subnetCidr);

## 12.4 Conclusion

The indexing strategy defined in this section focuses on the three areas required by the assignment: identifying primary keys, identifying foreign keys that require indexing, and selecting three additional query-driven indexes based on the queries defined in Section 11. Primary keys already provide the necessary clustered indexes. We created foreign key indexes. These are frequently used for joins, and creating indexes will maximize performance of these options. We also created three non-key indexes, on Interface.name, Device.name, and LanSubnet.subnetCidr. These were chosen because they directly support filtering, grouping, or presentation patterns demonstrated in the analytical queries. Together, these indexes provide targeted performance improvements without introducing unnecessary overhead, and they maximize the performance of the BlueSky IPAM database with most common anticipated patterns of use.

# Section 13: Attribute History

## 13.0 Summary

In network environments where DHCP is used to allocate IP addresses, a host may receive a different IP address over time. This can be due to changes such as the following:

* Expiration of DHCP leases
* A host being powered off or offsite
* A change to the configuration of a scope
* Hosts moving between networks.

Because a host’s IP address is usually not statically configured, it is important to maintain a historical record of these changes. Without such history, administrators cannot determine which Host was using a particular IP address at a specific point in the past. This information is vital for troubleshooting connectivity issues, correlating security events, and performing forensic analysis.

To support this requirement, the database has been extended to track historical values of the **Host.ipAddress** attribute. A new entity, **HostIpAddressHistory**, has been added to the database, along with a trigger that automatically records every transition in IP address. This ensures that historical values are preserved regardless of whether the change occurs through an application, a script, or manual administrative updates.

## 13.1 Structural Database Rule Summary

The structural database rule governing Host IP address history was introduced earlier in **Section 4**. In summary, the database model now supports tracking historical values of the Host.ipAddress attribute by defining a one-to-many relationship between **Host** and a new entity, **HostIpAddressHistory**. Each Host may accumulate multiple history records over time, while each history record corresponds to exactly one Host. This rule provides the foundation for recording DHCP-driven IP address reassignments.

## 13.2 Conceptual ERD Summary

As described in **Section 5**, the conceptual ERD has been updated to reflect the addition of the **HostIpAddressHistory** entity. This entity captures historical IP address transitions and is linked to the Host entity with a (1)-to-(0..\*) relationship. The conceptual model now shows that Hosts can generate a sequence of historical IP assignments as their network configuration changes.

## 13.3 Physical ERD Summary

Section 6 has been updated to incorporate the **HostIpAddressHistory** table. The table introduces a surrogate key, a foreign key to Host, and the attributes needed to store old and new IP address values along with the date of each change. The revised physical model ensures that historical Host IP transitions are maintained in a normalized, query-ready structure that follows the structures used elsewhere in the database.

## 13.4 SQL Implementation

The following are steps to implement and validate the creation and functionality of our HostIPAddressHistory table. This subsection implements the history-tracking design directly in SQL by creating the history table, defining the sequence used to generate its primary keys, and adding a trigger that automatically records every IP address change. All primary keys are assigned explicitly using NEXT VALUE FOR, rather than using default constraints.

13.4.1 History Table and Sequence

The **HostIpAddressHistory** table stores the old and new values of the Host’s IP address along with the date of each change. The table uses a surrogate key generated by a dedicated sequence.

CREATE TABLE HostIpAddressHistory (

hostIpAddressHistoryId INT NOT NULL PRIMARY KEY,

hostId INT NOT NULL,

oldIpAddress VARCHAR(64) NULL,

newIpAddress VARCHAR(64) NULL,

changeDate DATE NOT NULL,

CONSTRAINT FK\_HostIpAddressHistory\_Host

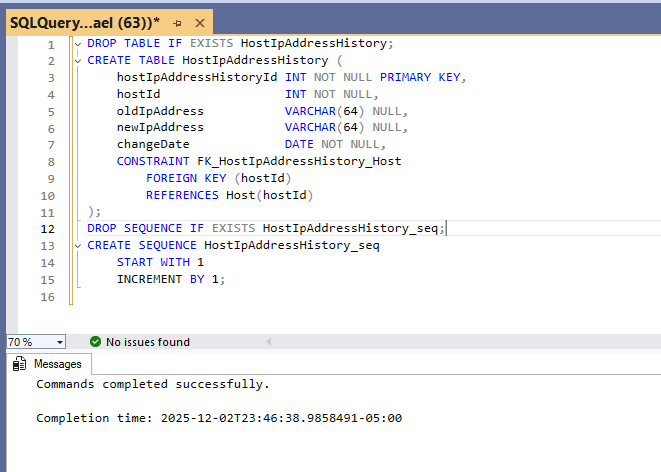
FOREIGN KEY (hostId)

REFERENCES Host(hostId)

);

CREATE SEQUENCE HostIpAddressHistory\_seq

START WITH 1

INCREMENT BY 1;  
  


13.4.2 Trigger Implementation

To ensure that historical IP address values are always recorded, two triggers are defined on the **Host** table.

|  |  |  |
| --- | --- | --- |
| Trigger Name | Condition | Notes |
| HostIpAddressHistoryInsertTrigger | AFTER INSERT | This will record the initial creation of a host when the host is created with an IP address specified. This has the “AFTER INSERT” condition. |
| HostIpAddressHistoryTrigger | AFTER UPDATE | This will record conditions where the IP address of a host changes (or changes between a NULL and a NON NULL value) |

**-- HostIpAddressHistoryInsertTrigger**

CREATE OR ALTER TRIGGER HostIpAddressHistoryInsertTrigger

ON Host

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

-- Capture initial IP assignment when a Host is created

INSERT INTO HostIpAddressHistory (

hostIpAddressHistoryId,

hostId,

oldIpAddress,

newIpAddress,

changeDate

)

SELECT

NEXT VALUE FOR HostIpAddressHistory\_seq,

inserted.hostId,

NULL AS oldIpAddress,

inserted.ipAddress AS newIpAddress,

GETDATE()

FROM inserted

WHERE inserted.ipAddress IS NOT NULL; -- Only record meaningful entries

END;

A screenshot of a computer

AI-generated content may be incorrect.

**--** HostIpAddressHistoryTrigger

CREATE OR ALTER TRIGGER HostIpAddressHistoryTrigger

ON Host

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

-- Insert a history row for every Host whose IP address changes

INSERT INTO HostIpAddressHistory (

hostIpAddressHistoryId,

hostId,

oldIpAddress,

newIpAddress,

changeDate

)

SELECT

NEXT VALUE FOR HostIpAddressHistory\_seq,

deleted.hostId,

deleted.ipAddress AS oldIpAddress,

inserted.ipAddress AS newIpAddress,

GETDATE()

FROM inserted

JOIN deleted

ON inserted.hostId = deleted.hostId

WHERE

(deleted.ipAddress IS NULL AND inserted.ipAddress IS NOT NULL)

OR (deleted.ipAddress IS NOT NULL AND inserted.ipAddress IS NULL)

OR (deleted.ipAddress IS NOT NULL AND inserted.ipAddress IS NOT NULL AND deleted.ipAddress <> inserted.ipAddress);

END;  
  
A screenshot of a computer

AI-generated content may be incorrect.

These triggers ensure that history is tracked automatically and consistently without any dependency on any external application logic.

## 13.5 Verification and Testing

To demonstrate that the trigger works as intended, we will create a new host and update its IP Address several times. Each update will produce a new history row.

## 13.5.0 Prepare an interface for the demonstration.

The following prepares a new interface to use for this demonstration.

DECLARE @SydneySwitchId INT = (

SELECT deviceId FROM Device WHERE name = 'SYD-MS225-1'

);

DECLARE @SydneySubnet10Id INT = (

SELECT lanSubnetId FROM LanSubnet WHERE subnetCidr = '10.20.10.0/24'

);

DECLARE @SydneySwitchPort5InterfaceId INT = NEXT VALUE FOR Interface\_seq;

INSERT INTO Interface (

interfaceId,

deviceId,

lanSubnetId,

internetConnectionId,

name,

interfaceType,

adminState

)

VALUES (

@SydneySwitchPort5InterfaceId,

@SydneySwitchId,

@SydneySubnet10Id,

NULL,

'port5',

'LAN',

'UP')

A screenshot of a computer

AI-generated content may be incorrect.

Note: There are some inconsistencies in the time stamps in these screenshots since I went through several iterations developing this logic. The final idempotent version is supplied in the iteration SQL script.

13.5.1 Create a Host for Demonstration

A new Host is created using the Host\_seq sequence to generate the primary key. This initial insert provides the baseline IP address for the history comparison.

-- Code to retrieve the required parameters

DECLARE @SydneySwitchId INT;

DECLARE @SydneySubnet10Id INT;

DECLARE @SydneySwitchPort5InterfaceId INT;

SELECT @SydneySwitchId = Device.deviceId

FROM Device WHERE Device.name = 'SYD-MS225-1';

SELECT @SydneySubnet10Id = LanSubnet.lanSubnetId

FROM LanSubnet WHERE LanSubnet.subnetCidr = '10.20.10.0/24';

SELECT @SydneySwitchPort5InterfaceId = Interface.interfaceId

FROM Interface

WHERE Interface.deviceId = @SydneySwitchId

AND Interface.name = 'port5';

-- Create new host

DECLARE @testHostId INT = NEXT VALUE FOR Host\_seq;

INSERT INTO Host (

hostId,

interfaceId,

hostName,

macAddress,

assetTag,

ipAddress

)

VALUES (

@testHostId,

@SydneySwitchPort5InterfaceId,

'history-test-host',

'AA-BB-CC-DD-EE-99',

'HIST-TEST-001',

'10.20.10.150'

);

A screenshot of a computer

AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.

13.5.2 Generate IP Address Changes

We then will perform a series of updates to simulate DHCP-driven reassignment of IP addresses. Each change is captured by the trigger and added to the **HostIpAddressHistory** table.

UPDATE Host SET ipAddress = '10.20.10.175' WHERE hostName = 'history-test-host';

UPDATE Host SET ipAddress = '10.20.10.200' WHERE hostName = 'history-test-host';

UPDATE Host SET ipAddress = NULL WHERE hostName = 'history-test-host';

UPDATE Host SET ipAddress = '10.20.10.225' WHERE hostName = 'history-test-host';

A screenshot of a computer

AI-generated content may be incorrect.

These updates cover the full set of conditions the trigger must handle, including changes to and from NULL values.

13.5.3 View Recorded History

We will next query the history table to confirm that each update was recorded accurately and that the Old IP Address, New IP Address, and Change Date attributes are present.

We will use the following query:

SELECT

hostIpAddressHistoryId,

hostId,

oldIpAddress,

newIpAddress,

changeDate

FROM HostIpAddressHistory

WHERE hostId = (

SELECT hostId FROM Host WHERE hostName = 'history-test-host'

)

ORDER BY hostIpAddressHistoryId;

We can see the transition of IP addresses accompanying the initial creation and each update to the host’s assigned IP address:

* NULL 🡪 10.20.10.150 – initial creation
* 10.20.10.150 → 10.20.10.175
* 10.20.10.175 → 10.20.10.200
* 10.20.10.200 → NULL
* NULL → 10.20.10.225

A screenshot of a computer

AI-generated content may be incorrect.

This verifies that the trigger correctly detects and records all changes, including those involving NULL values.

# Section 14: Data Visualizations

## 14.0 Data Visualizations

This section presents two data visualizations that display data about information stored in the BlueSky IP Address Management database. Each visualization includes (1) a SQL query used to obtain the underlying dataset, (2) an exported spreadsheet or chart created from that dataset, and (3) a narrative explaining the insight it provides. These demonstrate examples of how this database can be used for operational reporting, trend analysis, and decision support.

## 14.1 Synthetic Data in HostIpAddressHistory table

To prepare to provide a meaningful report in Visualization 1, we will add AI generated synthetic data to our IpAddressHistory table. We will add synthetic data for two of our synthetic hosts – bos-host-01 and syd-host-01 representing two different patterns. In Boston we will imagine a history that has “slow and steady churn.” In Sydney, we will imagine a history that has “higher, irregular churn.” The script is referenced in full in Appendix 2.

The following is an updated query and an updated screenshot of our HostIpAddressHistory table after we have added the synthetic data.

SELECT

hostIpAddressHistoryId,

Host.hostId,

Host.HostName,

oldIpAddress,

newIpAddress,

changeDate

FROM HostIpAddressHistory

JOIN Host on Host.hostId = HostIpAddressHistory.hostID

ORDER BY changeDate;

A screenshot of a computer

AI-generated content may be incorrect.

## 14.2 Visualization 1 – Daily Count of Host IP Address Changes

This visualization examines how Host IP address changes vary by site over time, using the historical data stored in the **HostIpAddressHistory** table.   
  
By plotting IP churn as a time series for each location, administrators can identify trends, spikes, and anomalies in DHCP behavior or network mobility.

A **line graph** is appropriate for this visualization because it clearly shows how change activity evolves day-by-day and allows easy comparison between sites. Differences in churn patterns can signal operational issues such as address exhaustion, unstable client networks, or variation in DHCP lease behavior across offices.

**SQL Query**

The following query aggregates the number of IP address changes per site for each date:

SELECT

Site.name AS SiteName,

CAST(HostIpAddressHistory.changeDate AS DATE) AS ChangeDay,

COUNT(\*) AS NumberOfChanges

FROM HostIpAddressHistory

JOIN Host

ON HostIpAddressHistory.hostId = Host.hostId

JOIN Interface

ON Host.interfaceId = Interface.interfaceId

JOIN Device

ON Interface.deviceId = Device.deviceId

JOIN Site

ON Device.siteId = Site.siteId

GROUP BY

Site.name,

CAST(HostIpAddressHistory.changeDate AS DATE) -- Collapse multiple rows occurring on the same day

ORDER BY

ChangeDay,

Site.name;  
  
The following is a screenshot of this query executing:  
  
A screenshot of a computer

AI-generated content may be incorrect.

14.2.1 Visualization Description

The query results were exported into a spreadsheet and plotted as a **multi-series line graph**, with the following configuration:

**X-axis:** Date of the recorded IP address change (across November and December 2025)

**Y-axis:** Number of changes occurring on that date

**Series:** One line per site (Boston, Sydney)

This produces a time-series chart with two distinct lines:

The **Boston** line shows relatively steady, low-volume churn.

The **Sydney** line shows higher and more irregular churn, with noticeable peaks on specific days.

The result is a clear visual comparison of daily churn patterns between the two locations.

**Narrative Interpretation**

The visualization shows that **Boston experiences low, consistent churn**, with small increments on six days in November. This reflects relatively stable DHCP behavior and low host mobility. Such a pattern is typical for an office with static workstation populations and long DHCP lease durations.

In contrast, **Sydney exhibits substantially higher churn**, including clusters of multiple changes on several days and a noticeable spike.

This pattern could indicate:

* Shorter DHCP lease durations
* Higher host mobility
* Network configuration changes
* Wireless roaming or subnet transitions
* Periods of increased provisioning or device turnover

The comparison highlights how different levels of network activity can manifest across organizational sites. The time-series line graph is particularly valuable because it conveys both **volume** and changes over time making deviations from expected behavior easy to spot.

## 14.3 Adding Synthetic Hosts to additional subnets

To prepare to provide a meaningful report in Visualization 2, we will add AI generated synthetic hosts to our additional subnets.

We will add the following:

* 1 host to the subnet 10.10.10/0/24 in the Boston HQ
* 3 hosts to the subnet 10.10.20.0/24 in Boston HQ
* 2 hosts to the subnet 10.20.20.0/24 in the Sydney Office
* 1 host to the subnet 10.10.30.0/24 in the Boston HQ

This synthetic data is visible in the script in Appendix 3.

## 14.4 Visualization 2 – Active Host Counts by Site

**Overview**

The second visualization focuses on understanding how hosts are distributed across the LAN subnets within the Boston HQ and Sydney Office sites. Whereas Visualization 1 analyzed time-based trends in IP address churn, Visualization 2 examines the quantity of hosts within each site’s subnet structure. This perspective helps identify utilization patterns, potential capacity constraints, and asymmetries in subnet usage.

**Query Used**

The following query summarizes the host counts for each subnet by combining information from the **LanSubnet**, **Interface**, and **Host** tables:

SELECT

Site.name AS SiteName,

LanSubnet.subnetCidr,

COUNT(Host.hostId) AS hostCount

FROM LanSubnet

JOIN LanSupernet

ON LanSubnet.lanSupernetId = LanSupernet.lanSupernetId

JOIN Site

ON LanSupernet.siteId = Site.siteId

LEFT JOIN Interface

ON Interface.lanSubnetId = LanSubnet.lanSubnetId

LEFT JOIN Host

ON Host.interfaceId = Interface.interfaceId

GROUP BY

Site.name,

LanSubnet.subnetCidr

ORDER BY

Site.name,

LanSubnet.subnetCidr;

This query aggregates host counts at the subnet level. It also ensures that subnets with no hosts are included for completeness.

14.4.1 Visualization and Interpretation

A bar chart was created to visualize the number of hosts present in each subnet.

The results reveal:

* Boston HQ exhibits a tiered distribution, with one heavily utilized subnet (10.10.10.0/24) and two progressively lighter used subnets.
* Sydney shows a more balanced distribution, with both subnets receiving moderate usage relative to their size and purpose.
* The differences between sites highlight variations in operational design, user/device density, and network segmentation strategy.

**Operational Insight**

This visualization provides insight for administrators responsible for capacity planning and subnet management. In a real-world setting, a site may have dozens of subnets and hundreds of hosts per subnet. Subnets approaching higher utilization levels may require future expansion or rebalancing, while unused or lightly used subnets may indicate legacy configuration or opportunities for cleanup and consolidation. The visualization reinforces the value of the structured IPAM design implemented throughout the project.

## 14.5 Summary

These two visualizations demonstrate how the BlueSky IP Address Management database can be used to visually analyze elements of the BlueSky networking environment. The first visualization highlights patterns of IP address churn over time by leveraging the new history-tracking logic added in this iteration, offering insight into host stability and operational change activity. The second visualization examines host distribution across subnets within the global network. This provides a clear view of how address space is utilized across the environment. Together, these visualizations show how the implemented schema can support both temporal and structural analysis, enabling administrators to understand changes over time while also assessing current network utilization. In a real world scenario such an environment may have thousands of hosts and hundreds of networks (or more). While the visualizations presented were produced with a small subset of representative data, these visualizations are examples of how this database can provide insight and assist with the operations of a large dynamic real-world environment.

# Section 15: AI Collaboration Summary for Iteration 6

## 15.1 AI Collaboration Overview

The following is an overview of the conversation that drove this week’s iteration.

**Stage 1 – Beginning a new context**

As in the previous weeks, I exported a context and decision log from the previous iteration conversation. I asked it to produce a bootstrap prompt. I then began a new conversation and provided the json context files, conversation transcripts, and iteration instructions for the previous iterations. I also supplied the final iteration 5 template files and the instructions and templates for iteration 6. I asked it to provide a rating from 1 to 10 on the level of understanding of the context, the stability over the conversation, and the amount of headroom available in the conversation. It gave me scores of 10, 9.9, and 9.5 respectively.

**Stage 2 – Beginning working on Section 13 – The History Table**

At the stage I began working on this section, the feedback for iteration 5 was not yet available. I started working on identifying the best attribute to track for the history table. After asking for the pros and cons of several options, I suggest using the IP Address history of a host as this has a lot real world value. (In Networking, needing to identify which host held which at a specific point in the past is a frequent challenge). AI gave me positive feedback, and we started working on a history table based on this attribute. AI provided verbiage to add to the structural database rules (section 4), conceptual ERD (section 5), and physical ERD (section 6). With some minor modifications, I added this content to those sections. Likewise, I generated and lightly revised the content added to section 13.

**Stage 3 – Incorporating Iteration 5 feedback into section 13**

The next day, the feedback for iteration 5 was now available. There were several elements to incorporate.

* Adding a subtype discriminator to the device table
* Updating the Table insertions to use the “NEXT VALUE FOR” function on the respective sequences.

We first incorporated that into the SQL code for section 14. I used AI to update code for testing the History Table several times. I focused on creating a good test use case to demonstrate the history table without excessive complexity (as there are many dependencies throughout the schema). I also worked to ensure the code was idempotent.

**Stage 4 – Incorporating Iteration 5 feedback into the previous sections**

I then turned to incorporating the feedback in the previous sections. I generated, revised, and added verbiage for the subtype discriminator into sections 4 (structural database rules), 5 (conceptual ERD), 6 (Physical ERD), 7 (Attribute reasoning), 9 (SQL Script), and 11 (Use Cases and Queries).

**Stage 5 – Section 14 (Data Visualizations)**

I started working on the content for Section 14. ChatGPT suggested 2 visualizations – A day by day count of the number of IP address changes and a list of active hosts by site. For the first visualization, I decided to create synthetic data to generate a months’ worth of DHCP changes so that the visualization presents compelling data. I updated it to also collapse the multiple entries generated by the script into a single day. I exported this data to Microsoft Excel. SQL generated a report with a separate row for each site for each day (which was desired). Despite a lot of effort, I was not able to get Excel to display a graph for the full range in this format. (It truncated the output after the first 18 days). I tried using AI to troubleshoot but none of the proposed solutions worked. As I was up against a deadline to study for the exam, I manually reformatted the table to be suitable for a graph and then produced the graph. (I believe the issue was that the output presented the data in “tall format” and I needed to restructure it to “wide format.” There were SQL queries and functions in Excel that could do this, but this seemed out of scope for the course, and I was up against a time constraint).

I then started working on the second visualization. I realized that, with the current data I had, the suggested query produced a spreadsheet with only 2 rows and 1 column. (There were 5 hosts residing in each of 2 sites). I decided this would not be interesting. I worked on creating synthetic data; however, I realized that this would be excessive code to review with the time constraints. (The schema required a number of entities per site). I asked for suggestions for alternate visualizations. I changed the second visualization to “hosts per subnet.” Since there were only 2 subnets with hosts I decided to create synthetic code to add additional hosts to the other 3 currently empty subnets. This was successful. I accepted the additional content (and made a few additional minor modifications).

**Stage 6 – Removing the sequence constraints**

I realized at this stage, that I had not yet addressed the feedback that all table insertions should use the “NEXT VALUE FOR” function rather than table constraints. I removed the table constraints to force the use of the sequence and then worked on updating the insertion statements in the code. This was difficult since the script is 1900 lines long. I tried to use AI to automate remediating this. I asked AI to update the script to remove these dependencies. It processed the file for 10 minutes, but then only returned a list of manual modifications required. The output was somewhat helpful, but it was difficult to find each instance since the output wasn’t able to give specific line references. I decided to debug the SQL code manually within the editor until this was completed.

**Stage 7 – Final review**

I then performed a final review. I saw that we were specifying that the discriminator was constrained to 3 values, but I was under whether we did indeed have an enforcement mechanism in the database. I prompted AI and it confirmed we had a check statement enforcing this value in the discriminator attribute but that this was not enforced in the subtype tables (a device in the switch table having the switch discriminator attribute and a device in the router tabling having the router discriminator attribute). It suggested we create an additional trigger that validates this and roll back non-compliant transactions. It also suggested code for this. I asked AI to review the requirements for the iteration, and it did not find a requirement stating that we needed to enforce subtype membership in this way. While this could be done in a real world setting, I decided this would not be needed in this project.

Finally, I asked for a final validation that the document and SQL script met the iteration requirements, were internally consistent, and did not have any major spelling or grammar errors. It identified a spelling error and a phrase that could be made clearer. I corrected those items. It otherwise gave positive feedback, and I completed both files.

## 15.2 Two Key Prompts

The following prompts were particularly helpful in driving to a successful outcome.

**Prompt 1**

My initial prompt helped create a useful context to begin the conversation.  
  
Begin Bootstrap Prompt

I am beginning Iteration 6 of my MET CS 669 term project (BlueSky IPAM / Network Infrastructure Database).

I am providing two authoritative context files:

iteration6-context.json

iteration6-decision-log.json

Please:

Load and internalize both files fully.

Confirm that all decisions from Iterations 1–5 are frozen and must not be altered unless Iteration 6 explicitly requires it.

Acknowledge that the authoritative schema, stored procedures, seed data, and SQL implementation come from Iteration 5.

Wait for me to upload the Iteration 6 Guided Walkthrough PDF before making any recommendations.

Do not rewrite earlier sections or regenerate SQL unless Iteration 6 requires changes.

End Bootstrap

Prompt Human Supplemental Prompt

This is iteration 6 for my term project project. We have previously worked on iterations 1 through 5 and this will be the final iteration. I will supply the following materials for this.

Iteration 4 Setup Materials

iteration-3-conversation.txt - This is a transcript of the conversation where we worked on the material for

iteration 3 iteration4-context.json - The context at this checkpoint - the beginning of iteration 4 iteration4-decision-log.json - The decision log at this checkpoint, the beginning of iteration 4

Iteration 5 setup materials

iteration-4-conversation-text.txt - This is a transcript of the conversation where we worked on the material for

iteration 4 iteration5-context.json - The context at this checkpoint - the beginning of iteration 5 iteration5-decision-log.json - The decision log at this checkpoint, the beginning of iteration 5

Iteration 6 setup materials

iteration-5-conversation-text.txt - This is a transcript of the conversation where we worked on the material for

iteration 5 iteration6-context.json - The context at this checkpoint - the beginning of iteration 6 iteration6-decision-log - The decision log at this checkpoint, the beginning of iteration 6

The previously submitted documents for iterations 5 are as follows: BohenMichael\_CS669\_Iteration5Template.docx - This is our current working documents BohenMichael\_CS669\_TermProjectIteration5Template.sql - This is our current working SQL template. Information from newer revisions supersedes context and decisions from previous revisions.

End Human Supplemental Prompt

Please provide a rating from 1 to 10 on the following: 1. Your level of understanding of the context. 2. The stability of this conversation 3. The headroom to add additional materials to this conversation

As in previous weeks, beginning this week allowed the conversation to have sufficient context to proceed efficiently with the final iteration of the term project. This week there was substantial additional iteration over the SQL script. It was likely again fortunate that we started from a clean state. By the end of the conversation, it appears that the conversation was starting to be truncated.

**Prompt 2**

The second prompt that stands out was the series of requests that I made to create a synthetic dataset to populate the history table.  
  
Can you provide synthetic data to allow us to have records of changes occurring over a 10 day period in 2 sites. You can use the hosts that we already have included in the Synthetic data

ChatGPT said:

Absolutely — here is clean, ready-to-run synthetic data that will generate 10 days of IP address changes in both Boston and Sydney, using the Hosts that already exist in your synthetic dataset.

[Redacted for length]

My Response:

Can you use bos-host-01 and syd-host-01. You also can hard code dates in November 2025 for the purposes of having synthetic data

ChatGPT said:

Absolutely — here is a clean, idempotent block of synthetic data that creates 10 days of IP address change history for two existing hosts:

bos-host-01

syd-host-01

using hard-coded dates in November 2025, suitable for screenshots and consistent visualization output.

This synthetic data:

✔ Does NOT modify or overwrite real host IPs

✔ Inserts rows directly into HostIpAddressHistory (because you want fixed dates)

✔ Is fully idempotent — old synthetic rows are deleted first

✔ Uses realistic change patterns

✔ Matches your November 2025 timeline

✔ Works perfectly with your Visualization 1 (changes by site)

⭐ FINAL SYNTHETIC HISTORY DATA BLOCK (for Section 14)

[Redacted for length]

Perfect for Visualization 1:

Your query…

SELECT Site.name, COUNT(\*) ...

…will now produce meaningful site-level bars showing:

Sydney has 10 events

Boston has 10 events

(but you can vary this if needed)

⭐ If you want stronger contrast for the visualization

I can adjust the numbers to:

Boston = 6 days of changes

Sydney = 14 days of changes

…so the bar chart reflects different operational patterns.

Would you like:

A) A version with unequal churn between sites?

B) A version with more hosts?

C) Visualizations 2 & 3 using these datasets?

D) Updates to the Section 14 narrative to match these hosts?

Just tell me!

My response:

Can you make the churn vary a bit to provide a good visual demonstration

This allowed me to create a representative dataset that was usable for visualization 1. The synthetic data allowed this report to be representative of the patterns that might be seen in the real world. It also allowed me to demonstrate that the visualization that can be scaled to greater numbers of hosts and additional sites.

## 15.3 Two Key Iterations

The following are two key iterations.

**Iteration 1**

The first key iteration was to change the visualization 2. Originally we had this as listing the number of hosts per site. To make this display interesting data, I was originally going to add synthetic data to the other 3 sites. I issued the following prompt  
  
 I am working on Visualization 2 now. Can you add some more synthetic data to give us a 1-3 hosts in the other 3 sites. It is OK to simply add a subnet and a switch for each of these.

I added these and realized that this added several hundreds of lines to my script and would be challenging to debug in the time available for this iteration. (Unfortunately, at this stage, much of the AI conversation regarding this interaction at this stage appears to be missing or truncated. This might be due to the number of iterations of the SQL script throughout the conversation). After realizing that my first idea for synthetic data might be too much for the scope of this project, I asked for several alternative ideas for visualizations and decided upon number of hosts per subnet. (There were far fewer inserts required for this metric). I currently had 2 subnets that currently each had 5 hosts in it. I asked for synthetic data to add additional hosts so that each of the 5 existing subnets had a unique quantity of hosts. This then provided data that allowed us to create an interesting and representative visualization.

**Iteration 2**

The second key iteration was addressing the feedback related to sequences. Originally, we had table constraints that added a default sequence number as the primary key on each table insert. This did not conform to the iteration requirement, which requested that we explicitly reference the entity’s sequence on each table insert. I removed these constraints to force errors in each location where there was an insert without a reference to the sequence. The script then did produce many such errors. I tried multiple strategies with AI to evaluate the SQL file to remediate these errors. These did allow me to identify a few of them, but the output was not sufficient to quickly identify and remediate each line of code throughout a 1900 line script. ChatGPT offered to evaluate entire script and produce a corrected version. I accepted this suggestion, and AI processed the file for 10 minutes, however it did not produce a corrected version, but rather provided a similar list of remediations. (Interestingly, I clicked on the “show thinking” button and it was clear that, internally, ChatGPT invoking a couple dozen possible algorithms to do this and then validating whether they were successful, but the validation kept failing. Perhaps, this task was just a bit too far beyond what this model was capable of at the moment). I decided to pivot and directly debug this script manually in SQL Management Studio itself. After a bit of time, I had worked through and corrected each of the insert statements.

## 15.4 AI Transcripts (Separate Documents or Links)

Iteration 5 – <https://chatgpt.com/share/692a5d70-02a8-8013-8d1b-d03af07b28d6>

Note: the bottom of this transcript has the commands I used to generate the context for iteration 6.

Iteration 6 - <https://chatgpt.com/share/6933db35-756c-8013-9e0a-d07741fd7966>

# Appendix 1 – JSON history

The following are the contents of the file iteration6-context.json supplied at the beginning of this week’s prompt.

{

"project": "BlueSky IPAM / Network Infrastructure Database",

"course": "MET CS 669",

"iteration": 6,

"status": "Starting Iteration 6 with all Iteration 1–5 decisions locked.",

"authoritativeSources": {

"physicalERD": "Finalized in Iteration 4; unchanged in Iteration 5",

"sqlImplementation": "Iteration5\_Final\_SQL",

"priorIterations": ["Iteration1", "Iteration2", "Iteration3", "Iteration4", "Iteration5"],

"schemaState": "Stable; no structural changes requested for Iteration 6 unless explicitly specified."

},

"primaryDesignDecisions": {

"surrogateKeys": "All tables use integer surrogate primary keys, generated via SQL Server SEQUENCE objects.",

"specializationModel": {

"supertype": "Device",

"subtypes": ["Router", "Switch"],

"mapping": "Shared primary key with foreign key reference to Device; subtype rows created only when needed.",

"discriminatorHandling": "No explicit discriminator column; subtype existence determines type."

},

"referentialIntegrity": {

"org→site": "1-to-many",

"site→supernet": "1-to-1 or 0-to-1",

"supernet→subnet": "1-to-many",

"subnet→dhcpscope": "0-to-1",

"device→interface": "1-to-many",

"interface→host": "1-to-0/1 (unique constraint implemented)"

},

"storedProcedures": {

"AddOrganization": "Creates an organization by name; prevents duplicates.",

"AddSiteWithSupernet": "Creates a site and its associated LAN supernet.",

"AddSubnet": "Creates a subnet and optional DHCP scope; looks up organization and site by name."

},

"seedData": {

"sites": ["Boston HQ", "New York Office", "Dublin DC", "São Paulo Office", "Sydney Office"],

"primaryAnalysisSites": ["Boston HQ", "Sydney Office"],

"hostCounts": {

"boston": 5,

"sydney": 4

},

"deviceAssignments": "Routers assigned WAN and LAN-gateway interfaces; switches assigned multiple access ports."

},

"analyticalQueries": {

"query1": "Host–Interface–Subnet inventory across multiple tables.",

"query2": "Device specialization report using Device, Router, and Switch tables.",

"query3": "Subnet utilization via vSubnetHostCounts view with grouping and aggregation."

},

"indexingStrategy": {

"foreignKeyIndexes": "All FK columns indexed for join performance.",

"queryDrivenIndexes": [

"Interface.name",

"Device.name",

"LanSubnet.subnetCidr"

]

}

},

"iteration6StartingPoint": {

"noSchemaChanges": "Schema is frozen unless Iteration 6 explicitly requires modifications.",

"noSPChanges": "Stored procedures locked unless required for Iteration 6.",

"seedDataFrozen": "Synthetic dataset finalized; changes require explicit request.",

"queriesCompleted": "Analytical queries finalized in Iteration 5; not to be regenerated unless required."

}

}

The following are the contents of the file iteration6-decision-log.json supplied at the beginning of this week’s prompt.

{

"iteration": 6,

"previousDecisions": [

{

"iteration": 1,

"summary": "Established conceptual model, early entities, and high-level IPAM structure."

},

{

"iteration": 2,

"summary": "Developed structural rules and refined conceptual relationships."

},

{

"iteration": 3,

"summary": "Introduced specialization (Device → Router/Switch) and finalized conceptual integrity constraints."

},

{

"iteration": 4,

"summary": "Built physical ERD, added sequences, defined attributes, and implemented final table structures."

},

{

"iteration": 5,

"summary": "Implemented stored procedures, synthetic dataset, analytical queries, view definition, and indexing strategy."

}

],

"lockedDecisions": [

"All ERD entities and relationships from Iterations 1–4 are frozen.",

"The physical schema from Iteration 4 is the authoritative implementation.",

"Stored procedures from Iteration 5 (AddOrganization, AddSiteWithSupernet, AddSubnet) are frozen.",

"The host counts (Boston=5, Sydney=4) and device/interface assignments are frozen.",

"Analytical queries (Queries 1–3) and vSubnetHostCounts view are finalized and cannot be changed unless Iteration 6 explicitly requires it.",

"Index selections (PK, FK, and 3 query-driven indexes) are frozen."

],

"pendingDecisionsForIteration6": [

"New requirements will be handled only if described in the Iteration 6 Walkthrough.",

"No changes made yet; awaiting Iteration 6 instructions."

],

"notes": "This file should be loaded alongside iteration6-context.json at the beginning of the new thread."

}

# Appendix 2 – Synthetic Data Script for Visualization 1

The following is the synthetic data we are adding to the HostIpAddressHistory Table to prepare for Visualization 1:

-- 14.1

-------------------------------------------------------------

-- SYNTHETIC HISTORY FOR VISUALIZATION (Boston vs Sydney)

-- Boston: Light churn (6 days)

-- Sydney: Heavy, irregular churn (14 days)

-- All dates in November 2025

-------------------------------------------------------------

PRINT '--- Generating enhanced synthetic history data for Visualization 14.1 ---';

-------------------------------------------------------------

-- 1. Clear older synthetic history for these two hosts

-------------------------------------------------------------

DELETE FROM HostIpAddressHistory

WHERE hostId IN (

SELECT hostId FROM Host

WHERE hostName IN ('bos-host-01','syd-host-01')

);

-------------------------------------------------------------

-- 2. Retrieve host IDs

-------------------------------------------------------------

DECLARE @bosHostId INT;

DECLARE @sydHostId INT;

SELECT @bosHostId = hostId FROM Host WHERE hostName = 'bos-host-01';

SELECT @sydHostId = hostId FROM Host WHERE hostName = 'syd-host-01';

-------------------------------------------------------------

-- 3. Boston: Low, steady churn across November 2025

-------------------------------------------------------------

INSERT INTO HostIpAddressHistory VALUES

(NEXT VALUE FOR HostIpAddressHistory\_seq, @bosHostId, '10.10.10.101', '10.10.10.111', '2025-11-02'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @bosHostId, '10.10.10.111', '10.10.10.112', '2025-11-05'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @bosHostId, '10.10.10.112', '10.10.10.113', '2025-11-10'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @bosHostId, '10.10.10.113', '10.10.10.114', '2025-11-15'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @bosHostId, '10.10.10.114', '10.10.10.115', '2025-11-22'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @bosHostId, '10.10.10.115', '10.10.10.116', '2025-11-28');

-------------------------------------------------------------

-- 4. Sydney: Higher, irregular churn across the same month

-------------------------------------------------------------

-- Early November burst

INSERT INTO HostIpAddressHistory VALUES

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.101', '10.20.10.201', '2025-11-01'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.201', '10.20.10.202', '2025-11-01'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.202', '10.20.10.203', '2025-11-02'),

-- Mid-November spike (visual peak)

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.203', '10.20.10.204', '2025-11-10'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.204', '10.20.10.205', '2025-11-10'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.205', '10.20.10.206', '2025-11-10'),

-- Irregular changes throughout month

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.206', '10.20.10.207', '2025-11-14'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.207', '10.20.10.208', '2025-11-17'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.208', '10.20.10.209', '2025-11-18'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.209', '10.20.10.210', '2025-11-22'),

-- End-of-month flurry

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.210', '10.20.10.211', '2025-11-28'),

(NEXT VALUE FOR HostIpAddressHistory\_seq, @sydHostId, '10.20.10.211', '10.20.10.212', '2025-11-30');

PRINT '--- Enhanced synthetic churn creation complete ---';

# Appendix 3 – Synthetic Data Script for Visualization 2

The following is the synthetic data that we added representing additional hosts to prepare for Visualization 2:

---------------------------------------------------------------

-- SYNTHETIC HOST POPULATION FOR VISUALIZATION 2

-- Ensures each subnet has a unique host count:

-- Boston HQ 10.10.10.0/24 → now 6 hosts

-- Boston HQ 10.10.20.0/24 → now 3 hosts

-- Boston HQ 10.10.30.0/24 → now 1 host

-- Sydney 10.20.10.0/24 → already 5 hosts

-- Sydney 10.20.20.0/24 → now 2 hosts

---------------------------------------------------------------

---------------------------------------------------------------

-- REFRESH REQUIRED VARIABLES

---------------------------------------------------------------

GO

DECLARE @BostonSwitchId INT = (

SELECT deviceId FROM Device WHERE name = 'BOS-MS225-1'

);

DECLARE @SydneySwitchId INT = (

SELECT deviceId FROM Device WHERE name = 'SYD-MS225-1'

);

DECLARE @BostonSubnet10Id INT = (

SELECT lanSubnetId FROM LanSubnet WHERE subnetCidr = '10.10.10.0/24'

);

DECLARE @BostonSubnet20Id INT = (

SELECT lanSubnetId FROM LanSubnet WHERE subnetCidr = '10.10.20.0/24'

);

DECLARE @BostonSubnet30Id INT = (

SELECT lanSubnetId FROM LanSubnet WHERE subnetCidr = '10.10.30.0/24'

);

DECLARE @SydneySubnet20Id INT = (

SELECT lanSubnetId FROM LanSubnet WHERE subnetCidr = '10.20.20.0/24'

);

---------------------------------------------------------------

-- BOSTON SUBNET 10.10.10.0/24 — ADD 1 HOST (NOW 6 HOSTS)

---------------------------------------------------------------

DECLARE @Bos10\_Port6 INT = NEXT VALUE FOR Interface\_seq;

INSERT INTO Interface (

interfaceId, deviceId, lanSubnetId, internetConnectionId,

name, interfaceType, adminState

)

VALUES (

@Bos10\_Port6, @BostonSwitchId, @BostonSubnet10Id,

NULL, 'port10', 'LAN', 'UP'

);

INSERT INTO Host (

interfaceId, hostName, macAddress, assetTag, ipAddress

)

VALUES (

@Bos10\_Port6,

'bos10-host-06',

'AA-BB-CC-DD-10-06',

'BOS10-HOST-06',

'10.10.10.106'

);

---------------------------------------------------------------

-- BOSTON SUBNET 10.10.20.0/24 — ADD 3 HOSTS

---------------------------------------------------------------

DECLARE @Bos20\_Port1 INT = NEXT VALUE FOR Interface\_seq;

DECLARE @Bos20\_Port2 INT = NEXT VALUE FOR Interface\_seq;

DECLARE @Bos20\_Port3 INT = NEXT VALUE FOR Interface\_seq;

INSERT INTO Interface (

interfaceId, deviceId, lanSubnetId, internetConnectionId,

name, interfaceType, adminState

)

VALUES

(@Bos20\_Port1, @BostonSwitchId, @BostonSubnet20Id, NULL, 'port6', 'LAN', 'UP'),

(@Bos20\_Port2, @BostonSwitchId, @BostonSubnet20Id, NULL, 'port7', 'LAN', 'UP'),

(@Bos20\_Port3, @BostonSwitchId, @BostonSubnet20Id, NULL, 'port8', 'LAN', 'UP');

INSERT INTO Host (interfaceId, hostName, macAddress, assetTag, ipAddress)

VALUES

(@Bos20\_Port1, 'bos20-host-01', 'AA-BB-CC-DD-20-01', 'BOS20-HOST-01', '10.10.20.101'),

(@Bos20\_Port2, 'bos20-host-02', 'AA-BB-CC-DD-20-02', 'BOS20-HOST-02', '10.10.20.102'),

(@Bos20\_Port3, 'bos20-host-03', 'AA-BB-CC-DD-20-03', 'BOS20-HOST-03', '10.10.20.103');

---------------------------------------------------------------

-- BOSTON SUBNET 10.10.30.0/24 — ADD 1 HOST

---------------------------------------------------------------

DECLARE @Bos30\_Port1 INT = NEXT VALUE FOR Interface\_seq;

INSERT INTO Interface (

interfaceId, deviceId, lanSubnetId, internetConnectionId,

name, interfaceType, adminState

)

VALUES (

@Bos30\_Port1, @BostonSwitchId, @BostonSubnet30Id,

NULL, 'port9', 'LAN', 'UP'

);

INSERT INTO Host (

interfaceId, hostName, macAddress, assetTag, ipAddress

)

VALUES (

@Bos30\_Port1,

'bos30-host-01',

'AA-BB-CC-DD-30-01',

'BOS30-HOST-01',

'10.10.30.101'

);

---------------------------------------------------------------

-- SYDNEY SUBNET 10.20.20.0/24 — ADD 2 HOSTS

---------------------------------------------------------------

DECLARE @Syd20\_Port1 INT = NEXT VALUE FOR Interface\_seq;

DECLARE @Syd20\_Port2 INT = NEXT VALUE FOR Interface\_seq;

INSERT INTO Interface (

interfaceId, deviceId, lanSubnetId, internetConnectionId,

name, interfaceType, adminState

)

VALUES

(@Syd20\_Port1, @SydneySwitchId, @SydneySubnet20Id, NULL, 'port5', 'LAN', 'UP'),

(@Syd20\_Port2, @SydneySwitchId, @SydneySubnet20Id, NULL, 'port6', 'LAN', 'UP');

INSERT INTO Host (interfaceId, hostName, macAddress, assetTag, ipAddress)

VALUES

(@Syd20\_Port1, 'syd20-host-01', 'AA-BB-CC-DD-20-11', 'SYD20-HOST-01', '10.20.20.101'),

(@Syd20\_Port2, 'syd20-host-02', 'AA-BB-CC-DD-20-12', 'SYD20-HOST-02', '10.20.20.102');