



MANUAL TRANSMITTAL

Department of the Treasury
Internal Revenue Service

2.5.13

SEPTEMBER 27, 2022

EFFECTIVE DATE

(09-27-2022)

PURPOSE

- (1) This transmits revised IRM 2.5.13, Systems Development, Database Design Techniques and Deliverables.

BACKGROUND

- (1) This IRM 2.5.13 transmits the revised standards, guidelines, and other controls for documenting database systems. This IRM describes techniques for analyzing, designing, and modeling Internal Revenue Service (IRS) databases.

MATERIAL CHANGES

- (1) Effect on Other Documents, Changed the paragraph based on new requirement in IRM 1.11.2
- (2) 2.5.13.1, Moved the role for DB architects under “Roles and Responsibility”
- (3) 2.5.13.1.1 (1)(2), Removed and replaced better information because previous paragraphs did not reflect requirements of the Background subsection according to IRM 1.11.2.1.1
- (4) 2.5.13.2, Replaced period with a comma
- (5) 2.5.13.1.7, Added the following resources:
 - IRM 2.150.2 Configuration Management (CM) Process
 - San, Hirako. MongoDB Best Practices: Build Fault Tolerant Applications link
 - SQL Server Security Best Practices for taking on the SQL DBA Role as a developer link
 - PostgreSQL website link
 - E2E Cloud, Best Practices for PostgreSQL Database, May 19 2021 website link
- (6) 2.5.13.1.3 (13), Added a link for IRM 1.1.12 for Information Technology, Staffing Responsibilities
- (7) 2.5.13-3
- (8) 2.5.13.4, Removed extra space in front of Modifications/Redesign
- (9) 2.5.13.4(6), EDMO role was moved to the Roles and responsibilities subsection
- (10) 2.5.13.4.3.1 (2), Changed “interaction with,” to “interaction with” for better clarity.
- (11) 2.5.13.6.8.4.2 (1)(f)(g), Changed sentences to eliminate wordiness and more clarity
- (12) 2.5.13.7.2, Added Structured Query Language (SQL) Server Overview
- (13) 2.5.13.7.2.1, Added T-SQL Function Types
- (14) 2.5.13.7.2.2 (1), Changed “The are” to “The seven categories of data types are the following” for more clarity
- (15) 2.5.13.7.2.2 (1) (a-h), Added SQL Server Data Types
- (16) 2.5.13.7.2.3, Added SQL Server and Transact-SQL (T-SQL) Best Practices

- (17) 2.5.13.7.4 (1), Added MongoDB Overview
- (18) 2.5.13.7.4.1 (1), Added MongoDB Features and Benefits over RDBMS
- (19) 2.5.13.7.4.2 (1), Added Types of NoSQL Database Management Systems (DBMS)
- (20) 2.5.13.7.4.3 (1), Added MongoDB NoSQL Best Practices
- (21) 2.5.13.7.4.3.1 (1), Added MongoDB Security Best Practices
- (22) 2.5.13.7.4.3.1.1 (1), Added MongoDB Authentication and Authorization
- (23) 2.5.13.7.5, Relocated “Big Data Models and No Structured Query Language (NoSQL) Databases Overview” under MongoDB because it was in an incorrect subsection
- (24) 2.5.13.7.7 (1)(2)(3), Added more features as an overview for PostgreSQL
- (25) 2.5.13.7.7 (4), Added a table with the primary differences between PostgreSQL and MongoDB
- (26) 2.5.13.7.8, Added PostgreSQL Data Types
- (27) 2.5.13.7.9, Added PostgreSQL Best Practices
- (28) Figure 2.5.13-3, Added a caption
- (29) Figure 2.5.13-5, Added a caption
- (30) Figure 2.5.13-6, Added a caption
- (31) Figure 2.5.13-38, Changed “creates” to “create”
- (32) Exhibit 2.5.13-10 Acronyms and Terms, Added the following:
- BSON - Binary and JSON
 - CRUD - Create, Read, Update and Delete
 - CSV - Comma Separated Value
 - MIB - Management Information Base
 - PID - Process Identifier
 - PKI - Public Key Identifier
 - RBAC - Role Based Access Control
 - SAN - Subject Alternative Name
 - SDLC - System Development Life Cycle
 - SIEM - Security Information and Event Management
 - SSD - Solid State Disk
 - TLS - Transport Layer Security
 - TTL - Time To Live
- (33) Exhibit 2.5.13-11 Terms and Definitions, Added the following:

Shard	A single mongod instance or replica set that stores some portion of a sharded cluster's total data set. In production, all shards should be replica sets
Sharded cluster	The set of nodes comprising a sharded MongoDB deployment. A sharded cluster consists of config servers, shards, and one or more mongos routing processes.

Single-master replication	A replication topology where only a single database instance accepts writes. Single-master replication ensures consistency and is the replication topology employed by MongoDB.
Simulation	To represent the functioning of one system by another.
Simple Object Access Protocol	A messaging protocol for exchanging structured information via web services in a computer network.
Storage engine	The part of a database that is responsible for managing how data is stored and accessed, both in memory and on disk. Different storage engines perform better for specific workloads.
Subject Alternative Name	Subject Alternative Name (SAN) is an extension of the X.509 certificate which allows an array of values such as IP addresses and domain names that specify which resources a single security certificate may secure.

EFFECT ON OTHER DOCUMENTS

IRM 2.5.13, dated 12-23-2021, is superseded and supplements IRM 2.5.1 Information Technology, System Development.

AUDIENCE

The audience intended for this transmittal is personnel responsible for engineering, developing, or maintaining Agency software systems identified in the Enterprise Architecture. The mandates in this manual apply to all IRS production, development, and test database systems. This IRM for engineering, development, designing, and maintenance include work performed by government employees and contractors.

Nancy A. Sieger
Chief Information Officer

2.5.13

Database Design Techniques and Deliverables

Table of Contents

2.5.13.1 Program Scope and Objectives

2.5.13.1.1 Background

2.5.13.1.2 Authority

2.5.13.1.3 Roles and Responsibilities

2.5.13.1.4 Program Management and Review

2.5.13.1.5 Program Controls

2.5.13.1.6 Acronyms/Terms/Definitions

2.5.13.1.7 Related Resources

2.5.13.2 Internal Revenue Service (IRS) Relational Database Design Guidance

2.5.13.3 Data Modeling Overview

2.5.13.4 Database Design Overview

2.5.13.4.1 Types of Database Models

2.5.13.4.2 Business Analysis Best Practices

2.5.13.4.2.1 Database Design - Mission, Functions and Operations

2.5.13.4.2.2 Identify Tasks Performed and Data Usage

2.5.13.4.2.3 Identify Task/Data Relationships

2.5.13.4.2.4 Develop a List of Constraints

2.5.13.4.2.5 Develop a List of Potential Future Changes

2.5.13.4.3 Data Modeling Design

2.5.13.4.3.1 Identify Local Views of the Data

2.5.13.4.3.2 Formulate Entities/Entity Modeling

2.5.13.4.3.3 Specify Relationships

2.5.13.4.3.4 Add Descriptive Attributes

2.5.13.4.3.5 Consolidate Local Views and Design Perspectives

2.5.13.4.3.6 Present Data Model

2.5.13.4.3.7 Verify Data Model

2.5.13.5 Physical Database Design

2.5.13.5.1 Determine the User's Requirements

2.5.13.5.2 Determine the Processing Environment

2.5.13.5.3 Select DBMS Software

2.5.13.5.4 Design the Physical Placement of Data

2.5.13.5.5 Perform Sizing of Data

2.5.13.5.6 Consider Security and Recovery

2.5.13.6 Deliverables

2.5.13.6.1 Decision Analysis and Description Forms

- 2.5.13.6.2 Task Analysis and Description Forms
- 2.5.13.6.3 Task/Data Element Usage Matrix
- 2.5.13.6.4 Data Models
- 2.5.13.6.5 Entity-Attribute Lists
- 2.5.13.6.6 Data Definition Lists
- 2.5.13.6.7 Physical Database Specifications Document
 - 2.5.13.6.7.1 Physical Database Names
 - 2.5.13.6.7.2 Data Structure/Sizing
 - 2.5.13.6.7.3 Data Placement
- 2.5.13.6.8 Database Schema Refactoring Overview
 - 2.5.13.6.8.1 Evolutionary Database Techniques
 - 2.5.13.6.8.2 Database Refactoring Categories
 - 2.5.13.6.8.3 Common Indicators for Refactoring
 - 2.5.13.6.8.4 Database Refactoring Best Practices
 - 2.5.13.6.8.4.1 Database Refactoring Preparation Process
 - 2.5.13.6.8.4.2 Database Refactoring Strategies
- 2.5.13.7 Database Management System Software Supported by the IRS
 - 2.5.13.7.1 IBM Enterprise Database 2 Universal Database (DB2 UDB) Overview
 - 2.5.13.7.1.1 DB2 Physical Objects
 - 2.5.13.7.1.2 DB2 Performance Standards and Guidelines
 - 2.5.13.7.1.2.1 IRS DB2 Tables - Designing for Performance
 - 2.5.13.7.2 Structured Query Language (SQL) Server Overview
 - 2.5.13.7.2.1 T-SQL Function Types
 - 2.5.13.7.2.2 SQL Server Data Types
 - 2.5.13.7.2.3 SQL Server and Transact-SQL (T-SQL) Best Practices
 - 2.5.13.7.3 MySQL Overview
 - 2.5.13.7.4 MongoDB Overview
 - 2.5.13.7.4.1 MongoDB Features and Benefits over RDBMS
 - 2.5.13.7.4.2 Types of NoSQL Database Management Systems (DBMS)
 - 2.5.13.7.4.3 MongoDB NoSQL Best Practices
 - 2.5.13.7.4.3.1 MongoDB Security Best Practices
 - 2.5.13.7.4.3.1.1 MongoDB Authentication and Authorization
 - 2.5.13.7.5 Big Data Models and No Structured Query Language (NoSQL) Databases Overview
 - 2.5.13.7.6 Oracle Database Design Overview
 - 2.5.13.7.6.1 Oracle - Design for Performance Best Practices
 - 2.5.13.7.6.2 Relational Database Design Rules and SQL Coding Standards
 - 2.5.13.7.7 PostgreSQL Overview
 - 2.5.13.7.8 PostgreSQL Data Types
 - 2.5.13.7.9 PostgreSQL Best Practices

2.5.13.8 Database Security Design

2.5.13.8.1 Database Design Security Best Practices

2.5.13.9 IRS Extensible Markup Language (XML) Overview

2.5.13.9.1 IRS Extensible Markup Language (XML) Naming and Design Rules

Exhibits

2.5.13-1 Guidelines for Decision Analysis and Description Forms

2.5.13-2 Guidelines for Task Analysis and Description Forms

2.5.13-3 Sample Task/Data Element Matrix

2.5.13-4 Guidelines for Constructing Data Structure Diagrams (DSD)

2.5.13-5 Relationship between Two Entities-Classes

2.5.13-6 Explanation of Hierarchical Structure (One-to-Many) and (Many-to-Many) Relationships

2.5.13-7 Guidelines for Constructing Entity Relationship Diagrams (ERD)

2.5.13-8 Sample Data Definition List

2.5.13-9 Summary of Access Methods

2.5.13-10 Acronyms and Terms

2.5.13-11 Terms/Definitions

2.5.13-12 IRS Enterprise Life Cycle (ELC) Data Model Compliance Standards

2.5.13-13 Relational Database Design Constancy Rules - Types of Data Storing in Table

2.5.13-14 FIPS 127-2 Database Construct Sizing

2.5.13.1
(09-27-2022)
Program Scope and Objectives

- (1) **Scope:** This IRM establishes standards, guidelines, and other controls for documenting database systems developed for the Internal Revenue Service. This manual describes techniques for analyzing, designing, and modeling databases. This development includes that work performed by government employees as well as contractors. One of the core competencies IRS database engineers and database designers must have is the following:
 - a. **Operations** - This is the combined sum of all the skills, knowledge, and values of the Agency. Operations are the building blocks for designing, testing, building, and operating any system with scalability and dependability requirements.
- (2) **Objectives:** The main objectives of designing databases for the IRS is to produce logical and physical design models of the proposed new database system. The database design is the organization of data according to a database model and techniques pertaining to the following:
 - a. Analysis
 - b. Design
 - c. Description
 - d. Specification of data designed for automated business data processing using models to enhance communication between developers and customers
- (3) **Purpose:** To establish quality database designs with the outcome resulting in good consistency of data, elimination of data redundancy, efficient execution of queries, and produce robust, high performance applications.
- (4) **Audience:** The target audience for this manual is government employee and contractor data architects/analyst, database designer/administrators, developers, Information Technology engineers, and IT managers.
- (5) **Policy Owner:** The Application Development (AD) Associate, Chief Information Officer (ACIO), establishes all Information Technology internal controls for this IRM.
- (6) **Program Owner:** The AD, Technical Integration Office (TIO), Application Standards and Quality (ASQ) is the internal organization that is responsible for the administration, procedures, and updates related to this program. The TIO, Application Standards and Quality branch works closely with key stakeholders from Enterprise Operations (EOps), Data Management Services & Support (DMSS) when updating IRS policies, and industry standards related to all database initiatives.
- (7) **Primary Stakeholders:**
 - a. Application Development (AD)
 - b. Enterprise Operations (EOps), Data Management Services & Support (DMSS)
 - c. Information Technology, Cyber Security
- (8) **Program Goals:** Performing appropriate database design procedures are critical to meet the IRS databases' target-state for scalability, security, performance, and reliability requirements.

2.5.13.1.1
(09-27-2022)
Background

- (1) This IRM enables the IRS to meet federal requirements by providing approved National Institute of Standards and Technology (NIST) and IRS standards and procedures for developing and designing database systems which will allow greater control over the application development process.
- (2) The Office of Management and Budget (OMB) policies require that all agencies must comply with NIST guidance, unless they are national security programs and systems. NIST produces the guidelines to assist federal agencies with meeting the requirements of the Federal Information Security Management Act (FISMA) and the Federal Information Processing Standards (FIPS).
- (3) Some examples of standard practices for database design applies to establishing the following:
 - Database structure and segmentation
 - Access methods
 - Procedures to govern aspects of application design
 - Secondary indexing
 - Logical relationships
 - Different ways in which application programs use the database
 - Information Technology Security requirements and controls as specified in NIST Special Publication (SP) 800-53 Rev. 5 for Target Architecture

2.5.13.1.2
(09-27-2022)
Authority

- (1) The Clinger-Cohen Act of 1996
- (2) 21st Century Integrated Digital Experience Act (IDEA), December 2018
- (3) Federal Information Security Modernization Act of 2014, FISMA 2014
- (4) Government Accountability Office (GAO)
- (5) Government Performance Results Act (GPRA)
- (6) Administrator of the Government Service Administration (GSA)
- (7) Office of Management and Budget (OMB)
- (8) Presidential American Technology Council, 2017
- (9) Treasury Inspector General Tax Administration (TIGTA)
- (10) International Standard ISO -11179/(ISO/IEC 11179)

2.5.13.1.3
(09-27-2022)
Roles and Responsibilities

- (1) **Application Development's chain of command** include:
 - a. **Commissioner:** Oversees and provides overall strategic direction for the IRS. The Commissioner's and Deputy Commissioner's main focus is for the IRS's services programs, enforcement, operations support, and organizations. Additionally, the Commissioner's vision is to enhance services for the nation's taxpayers, balancing appropriate enforcement of the nation's tax laws while respecting taxpayers' rights.
 - b. **Deputy Commissioner, Operation Support (DCOS):** Oversees the operations of Agency-Wide Shared Services: Chief Financial Officer, Human Capital Office, Information Technology, Planning Programming and Audit Oversight and Privacy, and Governmental Liaison and Disclosure.

- c. **Chief Information Officer (CIO):** The CIO leads Information Technology, advises the Commissioner on Information Technology matters, manages all IRS IT resources, and is responsible for delivering and maintaining modernized information systems throughout the IRS. The Deputy Chief Information Officer for Operations assists the Chief Technology Officer (CTO).
 - d. **Application Development (AD), Associate Chief Information Officer (ACIO):** The AD ACIO reports directly to the CIO; oversees and ensures the quality of: building, unit testing, delivering, and maintaining integrated enterprise-wide applications systems to support modernized and legacy systems in the production environment to achieve the mission of the service.
 - e. **AD Deputy CIO (DCIO):** The AD Deputy ACIO reports directly to the AD ACIO and is responsible for:
 - Leading all strategic priorities to enable the AD Vision, IT Technology Roadmap and the IRS future state
 - Executive planning, and management of the development organization which ensures all filing season programs are developed, tested, and delivered on-time and within budget
- (2) **Application Development:** Responsible for building, testing, delivering, and maintaining integrated information applications systems, e.g., software solutions, to support modernized systems and production environment to achieve the mission and objectives of the service. Additionally, AD does the following:
- a. Works in partnership with customers to improve the quality of the IRS information systems, products, and services.
 - b. Maintains the effectiveness and enhance the integration of IRS installed base production systems and infrastructure while modernizing core business systems and infrastructure.
 - c. Establishes and maintains rigorous contract and fiscal management, oversight, quality assurance, and program risk management processes to ensure that strategic plans and priorities are being met.
 - d. Provides quality assessment/assurance of deliverables and processes.
 - e. Creates oversight support of enterprise modernization goals in coordination with Information Technology HR initiatives and policy.
 - f. Responsible for delivering filing season projects, and implementing Economic Stimulus changes.
 - g. AD has the following Domains:
 - Compliance
 - Corporate Data (CD)
 - Customer Service (CS)
 - Data Delivery Service (DDS)
 - Delivery Management; Quality Assurance (DMQA)
 - Identity & Access Management (IAM)
 - Internal Management (IA)
 - Submission Processing (SP)
 - Technical Integration Organization (TIO)
- (3) **Director, Compliance:** Provides executive direction for a wide suite of Compliance domain focused applications and oversee the IT Software Development organization to ensure the quality of production ready applications.

- a. Directs and oversees a unified cross-divisional approach to compliance strategies needing collaboration pertaining to the following:
 - Abusive tax avoidance transactions needing a coordinated response
 - Cross-divisional technical issues
 - Emerging issues
 - Service-wide operational procedures
- (4) **Director, AD Corporate Data:** Directs and oversees the provisioning of authoritative databases, refund identification, notice generation, and reporting.
- (5) **Director, Customer Service:** Directs and oversees Customer Service Support for service and communication with internal and external customers and providing taxpayers with self-service online capabilities.
 - a. Customer Service Domain's applications and systems provide:
 - Tax law assistance
 - Taxpayer education
 - Access to taxpayer account data
 - Maintenance of modernized information systems that meet the customer's needs for researching, updating, analyzing, and managing taxpayer accounts
 - b. Services to internal and external customers are provided through five primary means:
 - Centralized Contact Centers (for telephone, written, and electronic inquiries)
 - Self-service applications (via the telephone and Internet)
 - Field Assistance (for walk-in assistance)
 - Web Services
 - Management of Taxpayer Accounts
- (6) **Director, Data Delivery Services:** Oversees and ensures the quality of data with repeatable processes in a scalable environment. The Enterprise Data Strategy is to transform DDS into a data centric organization dedicated to deliver Data as a Service (DaaS) through:
 - Innovation - new methods, discoveries
 - Renovation - streamline or automate
 - Motivation - incent and enable individuals
- (7) **Director, Delivery Management & Quality Assurance (DMQA):**
 - Executes the mission of DMQA by ensuring AD has a coordinated, cross-domain, and cross-organizational approach to delivering AD systems and software applications
 - Reports to the AD ACIO and chairs the AD Risk Review Board.
 - Chairperson, Configuration Control Board, see IRM 2.5.1.2.2.2
 - Government Sponsor, Configuration Control Board, see IRM 2.5.1.2.2.2
- (8) **Director, Identity & Access Management (IAM) Organization:** Provides oversight and direction for continual secure online interaction by verifying and establishing an individual's identity before providing access to taxpayer information "identity proofing" while staying compliant within federal security requirements.
- (9) **Director, Internal Management:** Provides oversight for the builds, tests, deliveries, refund identification, notice generation, and reporting.

- (10) **Director, Submission Processing:** Provides oversight to an organization of over 17,000 employees, comprised of: a headquarters staff responsible for developing program policies and procedures, five W&I processing centers, and seven commercially operated lockbox banks. Responsible for the processing of more than 202 million individual and business tax returns.
- (11) **Director, Technical Integration Office:** Provides strategic technical organization oversight ensuring applicable guidance, collaboration, and consolidation of technical integration issues and quality assurance for the Applications Development portfolio.
- (12) **Information Technology (IT), Cybersecurity:** Cybersecurity manages the IRS IT Security program in accordance with the Federal Information Security Modernization Act of 2014 (FISMA) with the goal of delivering effective and professional customer service to business units and support functions within the IRS. These procedures are done as the following:
 - a. Provide valid risk mitigated solutions to security inquiries.
 - b. Respond to incidents quickly and effectively in order to eliminate risks/threats.
 - c. Ensure all IT security policies and procedures are actively developed and updated.
 - d. Provide security advice to IRS constituents and monitor IRS robust security program for any required modifications or enhancements.
 - e. Created the Information Technology Security Program Plan which represents strategic alignment to the IRS Strategic Plan 2018 - 2022, the IT Modernization Plan, and the IRS Cybersecurity Five-year Strategic Plan. The purpose is for the enhancement of all IT investments.
- (13) **Information Technology, Enterprise Operations, Enterprise Data Management Organization (EDMO):** Establishes and disseminates standards for conceptual, logical, and physical data modeling.
- (14) **Database (DB) Architect:** The DB Architect is associated with the System Development Life Cycle (SDLC) process, and must define detailed database designs by determining tables, indexes views, constraints, triggers, stored procedures tablespaces or storage parameters, what data, and how the data elements interrelate. However, the Architect is not involved in the daily operations of the system once it is deployed.
- (15) For more information on the mission and responsibilities of Information Technology, Organization and Staffing see IRM 1.1.12.

2.5.13.1.4
(09-27-2022)
**Program Management
and Review**

- (1) **Program Reports:** The IRS Enterprise Architecture Office (EAO) work in alliance with all IRS organizations to provide guidance for Information Technology application design patterns for standard solutions. This EAO information assist IT managers to plan, and architect all IT solutions, and manage investments in business and technology programs. EA facilitates the EA Review cycle and ELC Compliance Review process to ensure all reports/artifacts relating to the Enterprise Life Cycle are in compliance.
- (2) **Program Effectiveness:** The IRS Enterprise Data Management Office (EDMO) implements, updates, and distributes data standards and guidelines to IRS executive leadership.

2.5.13.1.5
(09-27-2022)

Program Controls

- (1) The IRS Enterprise Data Management Office (EDMO) must review all conceptual, logical and physical data models using the standards-based checklists which mandates the level of detail necessary for all data model.

2.5.13.1.6
(09-27-2022)

**Acronyms/Terms/
Definitions**

- (1) For Acronyms and Terms, see Exhibit 2.5.13-10
- (2) For Terms and Definitions, see Exhibit 2.5.13-11

2.5.13.1.7
(09-27-2022)

Related Resources

- (1) The resources below provide information for system development documentation standards from National Institute of Standards and Technology (NIST), IRS IRMs, and the IRS organizations' SharePoint repository sites/Uniform Resource Locators (URLs).
 - IRM 2.5.1 System Development, is the overarching IRM for the System Development program within the IRS
 - IRM 10.8.21 IRS Information Technology (IT) Security Database Security Policy
 - IRM 10.5.1 Privacy Compliance and Assurance (PCA) Program
 - IRM 10.8.1 Information Technology, Policy and Guidance
 - IRM 10.8.6 Application Security and Development
 - IRM 2.150.2 Configuration Management (CM) Process
 - NIST SP 800-53 Rev. 5, Security and Privacy Controls for Federal Information Systems and Organization
 - NIST SP 800-122, Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)
 - NIST SP 800-64, Security Considerations for System Development Life Cycle: Information Security, Revision 2 October 2008
 - NIST SP 1500, Big Data Interoperability Framework: Volume 4, Security and Privacy
 - NIST Big Data Program: NIST Big Data Public Working Group, <https://bigdata.nist.gov/home.php>
 - IRS Enterprise Data Standards and Guidelines (EDSG), see IRM 2.5.13.2
 - San, Hirako. MongoDB Best Practices: Build Fault Tolerant Applications
 - Online Oracle Database Development Guide see, <https://docs.oracle.com/en/database/oracle/oracle-database/12.2/adfns/toc.htm>
 - DB2 see, https://www.tutorialspoint.com/db2/db2_introduction.htm
 - DB2 Design, see <https://www.ibm.com/developerworks/data/library/techarticle/dm-0408whitlark/index.html>
 - Khang, Alex. Relational Database Professional Handbook: Design Rules and SQL Coding Conventions Guidelines, ISBN-13 979-8612037262. February 10, 2020
 - SQL Server, see <https://www.sqlservertutorial.net/getting-started/what-is-sql-server/>
 - SQL Server Security Best Practices, *Best practices for taking on the SQL Server DBA role as a developer* (mssqltips.com)
 - PostgreSQL see, <https://www.postgresql.org/docs/9.5/tutorial-arch.html>
 - E2E Cloud, Best Practices for PostgreSQL Database, May 19, 2021
 - Scott W. Ambler and Pramod J. Sadalage. Refactoring Databases: Evolutionary Database Design, ISBN-0-321-29353-3. August 2007
 - Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, <https://github.com/ept/ddia-references>

2.5.13.2

(09-27-2022)

**Internal Revenue Service
(IRS) Relational
Database Design
Guidance**

- (1) Enterprise Data Standards and Guidelines (EDSG) provides standards and rules for the development and modification of the names, definitions, and metadata for classes, attributes, and data models, that pertain to the Modernized Environment (ME) data elements located in IRS' modernization system data dictionaries, data models, and Current Production Environment (CPE) data elements.

#

- (3) See Figure 2.5.13-1 for illustration of the Design Rules of Relational Database.

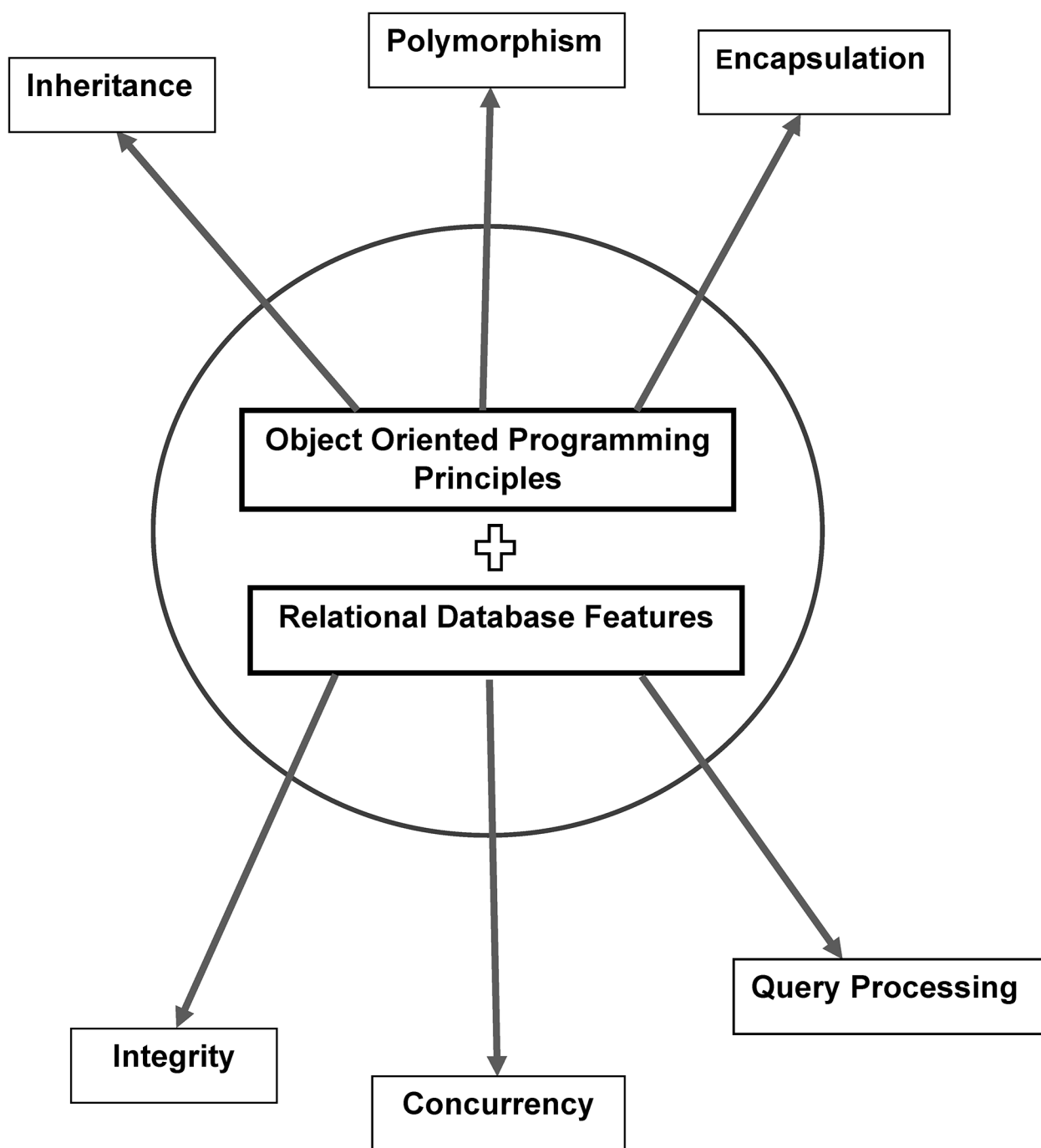


Figure 2.5.13-1

2.5.13.3
(09-27-2022)

Data Modeling Overview

- (1) Data modeling is the process of creating a data model for the data to be stored in a database. This data model emphasizes on what data is needed, and how it must be organized instead of what operations need to be completed on the data. It is also a conceptual representation of data objects, the associations between different data objects and the rules. The primary goal of using data models are the following:

- #

##

##

[illegible][illegible]

and sub-schemas are completed they are translated into their physical counterparts. Then the physical sub-schemas are supplied as part of the data specifications for program design.

- (8) The logical design encompasses a DBMS-independent view of data, and that physical design results in a specification for the database structure, as it is to be physically stored.
- (9) Implementation Design: Is the design step between the logical and physical design that produces a schema, and processed a DBMS.
- (10) Do not limit database development considerations to providing random access, or ad hoc query capabilities for the system.

2.5.13.4
(09-27-2022)
**Database Design
Overview**

- (1) A database design is the organization of data according to the database model. The designer determines what data must be stored, and how the data elements interrelate. The Database Life Cycle (DBLC) defines the five stages for creating a database as the following:
 - a. Requirements analysis
 - b. Logical Design
 - c. Physical Design
 - d. Implementation
 - e. Monitoring
 - f. Modifications/Redesign
 - g. Maintenance
- (2) Logical database design is the process of determining how to arrange the attributes of the entities in a business environment into database structures such as tables of a relational database.
- (3) To develop a logical database, analyze the business needs of the organization that the database would support, how the operations relate to each other, and what data is required in business operations. After this analysis, model the data.
- (4) Modeling involves studying data usage, and grouping data elements into logical units so that a task supported by one or more organizational units is independent of support provided for other tasks. Exhibit 2.5.13-9 provides the terms, and descriptions for logical database design.
- (5) Providing each task with its own data groups allow changes in data requirements of one task to be minimally impacted on data provided for another task. When data is managed as a synthesis, data redundancy is minimized, and data consistency among tasks and activities is improved. Figure 2.5.13-1 graphically expresses this point.

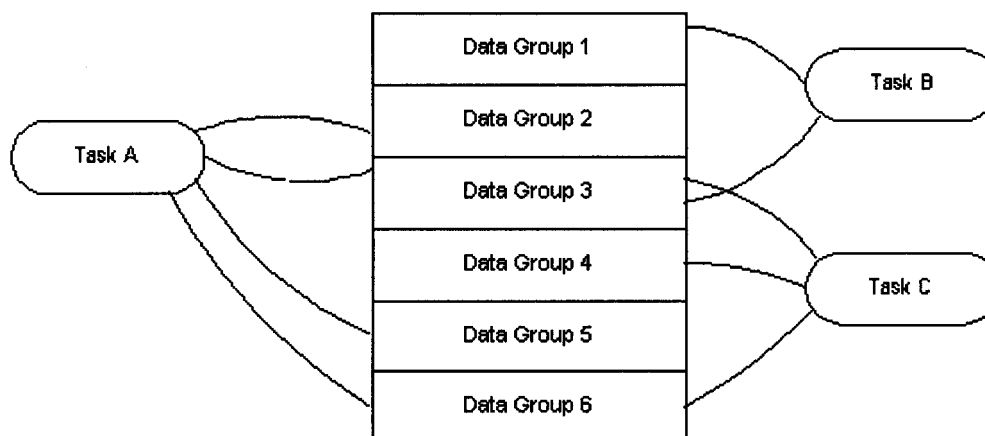


Figure 2.5.13-2

- (6) Logical database design comprises two methods to derive a logical database design. The first method is used to analyze the business performed by an organization. Following this analysis, the second method is used to model the data that supports the business. These methods are:
 - a. Business Analysis
 - b. Data Modeling
- (7) For IRS Enterprise Life Cycle (ELC) Data Model Compliance Standards, see Exhibit 2.5.13-12.

2.5.13.4.1 (09-27-2022)

Types of Database Models

- (1) There are five standard types of database models:
 - **Hierarchical:** Data is organized into a tree-line-structure, where the hierarchy starts from the Root data, and expands like a tree, adding parent to child nodes.
 - **Network:** This database model is used to map many -to-many data relationships, and becomes more connected as more relationships are created. This is an extension of the Hierarchical model, and was the most popular before the Relational Model was implemented
 - **Relational:** Data is organized into two-dimensional tables, and the connection is maintained through a common field. The structure of data in the relational model is tables. The tables are also known as relations in Relational Model.
 - **Entity-Relationship:** Entity-Relationships are created by dividing the object into entities, and its features into attributes. E-R Models are sketched to represent the relationships as image forms for easier comprehension, and are used during design.
 - **Object-Oriented Database (OODB):** A combination of an Object-Oriented database model and a Relational database model that supports objects, classes, inheritance etc. The goal of this model is to close the gap between relational databases, and the Object-Oriented practices used in many programming languages, e.g., C++, C#, and Java. Databases that represent data in the form of objects and classes. Object-Oriented databases have the same principles of Object-Oriented Programming (OOP) which is the combination model features of (concurrency, transaction, and recovery).

- **zObject-Oriented Database Model (OODBM):** Similar principles to an Object-Oriented programming language. An Object-Oriented database management system is a hybrid application that uses a mixture of Object-Oriented and Relational Database platform to process data. See Figure 2.5.13-3

Object-Oriented Database Model (OODBM)

Example of OODBM
Object-Oriented Programming + Relational Database Features = Object Oriented Database Model

Figure 2.5.13-3

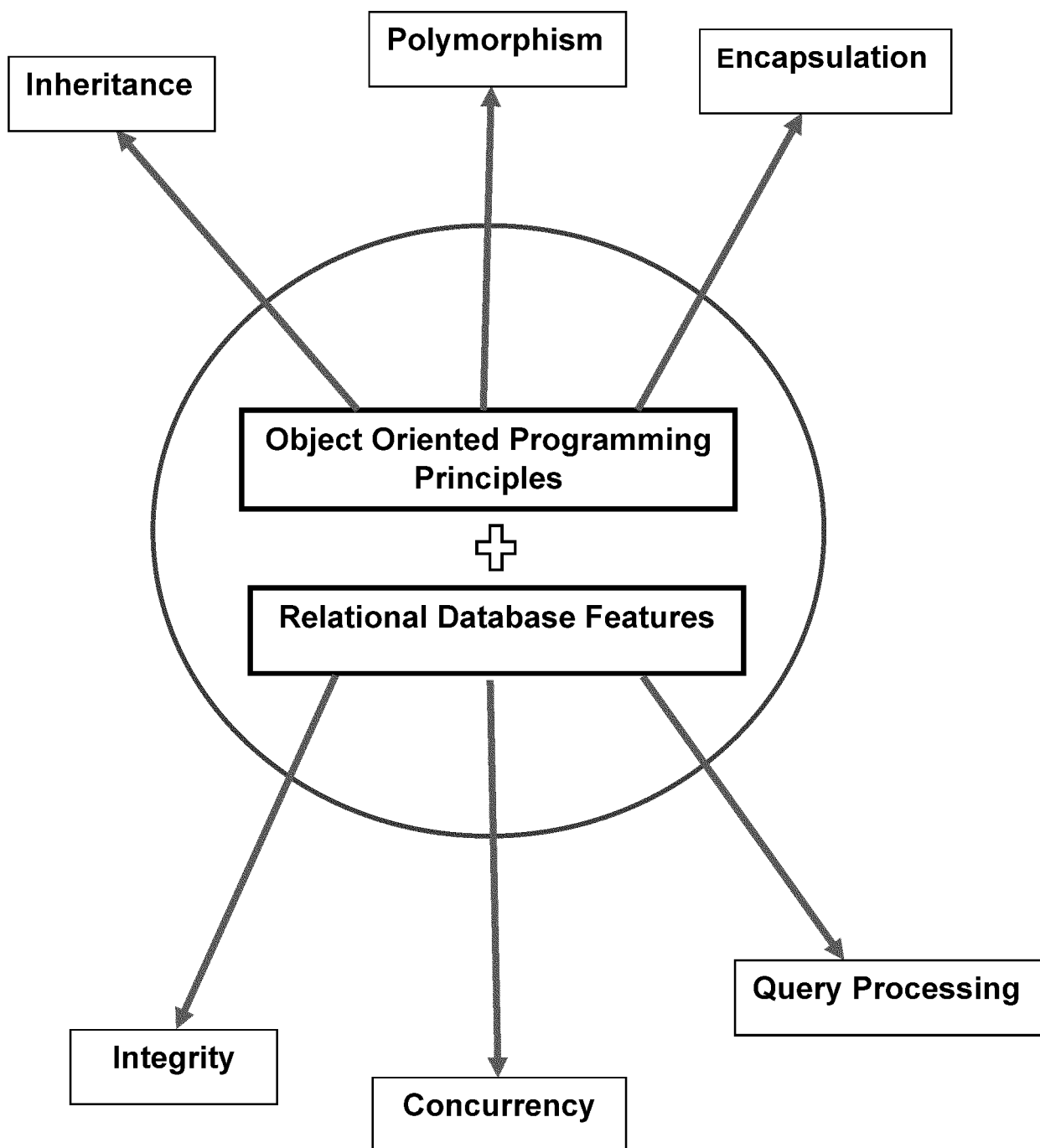


Figure 2.5.13-4 An example of Object-Oriented Database Model

2.5.13.4.2
(09-27-2022)

Business Analysis Best Practices

- (1) **Business Analysis** - Business analysis enables an enterprise to articulate the needs, justification for change, and to design and describe solutions that deliver value. Business Analysis is also a method for analyzing, and understanding a customer's enterprise business needs. The Business Analyst recommends relevant solutions by defining, documenting, and managing requirements while working with stakeholders to establish the requirements. In applying this method, the objectives are to:

- Must have obtain a clear understanding of an organization's objectives, and how it performs its mission
- Identify specific requirements that must be reflected in the database that involves decision analysis and task analysis
- Allow the analyst to focus on the information requirements, and how they are related after identifying each decision and task
- Gain an understanding of the requirements, not as a critique of the operations
- Identify stated data needs, and various indicators such as: organizational structure, environmental policies, interaction between functions, etc., which may indicate additional data requirements
- Define the scope of the database, and the environment that the database will support
- Identify any constraints for the database operation
- Define the Acceptance Criteria for projects and user stories
- Produce documentation that presents a valid picture of the organization's operation

(2) **Analyze Documentation** - Prior to applying this method, acquire and study the following documentation:

- A high-level data flow diagram (DFD) depicting the major applications to be supported, and the major data sources and outputs of these applications.
- Detailed DFDs depict the functions, tasks performed, and a list of the documents, files, and informal references (e.g., memos, verbal communications, etc.) used to perform each function.

(3) **Business Analysis eight (8) steps:**

- a. **Create a Business Case as follows:**
 - Identify and validate the mission, core values, functions, and operations
 - Identify the scope and primary business/project objectives
 - Use business analysis techniques such as Strength, Weakness, Opportunities, and Threats (SWOT) analysis
 - Identify tasks performed and data usage in the current and target environment
 - Identify all task/data relationships
 - Develop a thorough list of constraints
 - Create a cost-benefit analysis and/or Activity-Based Costing
- b. **Create a Stakeholder register as follows:**
 - Obtain valid information about the stakeholders
 - Identify who makes project decisions, and signs-off project documentation
 - Identify who are the product owners for the Enterprise Life Cycle phases.
 - Identify where the stakeholder works, and who they report to in the agency
 - Identify the stakeholder(s)' role in the project, and their level of influence
- c. **Create a Business Analysis Plan as follows:**
 - Define the resources and tasks associated with the requirements
 - Use business analysis tools such as a RACI matrix, this is a table that displays who are the responsible, accountable, consulted, and informed stakeholders

- d. **Elaborate on business objectives that were defined in the business case, and create the requirements as follows:**
 - Perform an assessment of the business objectives, and current business process creating the as-is (current state), and to-be (target state)
 - Focus information on the elements: infrastructure (technology), process, business related data, impacted business domain(s), goals, and measures
 - e. **Identify options that are the best solutions for project objectives:**
 - Assist the development team with providing more information about similar situations in the agency with other developers, and their practices, as documented in i.e., (lessons learned or playbooks)
 - Use business analysis tools, e.g., Work Breakdown Structure (WBS), Technology Capability assessment, and business or IT staff interviews
 - Use other tools such as Cause and Effect analysis
 - g. **Expand on product requirements:**
 - Once the high-level scope of the product has been defined, the person performing the Business Analyst functions must focus on the detailed product requirements before actual development begins
 - Use business analysis techniques such as use cases, story boards, prototypes, and/or wireframes
 - Document both functional and nonfunctional requirements
 - h. **Support the Project Manager and/or Scrum Master and development team:**
 - Support the programmers by constantly reviewing their deliverables to ensure they are in-line with objectives in the business case
 - Facilitate interviews with the Project Manager or Scrum Master, and based on their feedback and project team, facilitate and/or update requirements documentation
 - As needed, engage with quality control analysts (DMQA) to ensure that requirements are properly tested
- (4) **Gap Analysis** - Perform this analysis by comparing the identified current state with the desired outcomes to assess if there are any business process gaps that prevents the customer and stakeholders from achieving their business needs.

#

2.5.13.4.2.1
(09-27-2022)

**Database Design -
Mission, Functions and
Operations**

- (1) Identify the mission, functions and operations of the organizational element that the database is to support. The purpose of this step is to define the scope of the potential database's current and future needs, and develop a reference point for further analysis. This step covers all relevant functional areas and be developed separately from any single application design effort.
- (2) In examining an organizational element, which may range in size from a branch to an entire organization, the following may provide sources of information:
 - **Business Impact Analysis (BIA):** This is the organization's BIA best source of information. This plan may vary widely in content, but must articulate the organization's current and future state, a discussion of each system's scope, and definitions of the dependencies between major systems (both automated and manual) and groups of data. With this information, it is possible to determine which functional areas must be included within the scope of the new design.
 - **Scope Responsibility:** If the BIA is not available, or does exist, but does not contain diagrams of systems and data dependencies, use it to determine the scope. In this case, persons within the relevant functional areas must be interviewed to determine how they relate to the rest of the organization.
 - **Interviews:** Additional interviews can be conducted for newly identified areas to ascertain the extent to which they share data with the application(s) under design.
 - **Requests for Information Services (RIS):** Potential sources of information for the , mission, functional statements, internal revenue manuals, and senior staff interviews.
 - **Future State:** Future changes to the organization must be considered when defining the scope of the design effort, i.e., any major changes in operating policy, regulations, etc. Each potential change must be identified, and further defined to determine whether it could change the definition, usage, or relationships of the data. Where a change could affect the database in the future, the design scope must be expanded to consider the effects of this change.
- (3) Construct a high-level DFD to graphically depict the boundaries after determining the scope of the database.

2.5.13.4.2.2
(09-27-2022)

**Identify Tasks Performed
and Data Usage**

- (1) Identify the tasks performed in each of the functions and operations. The purpose is to identify tasks performed in each function of the organizational element that the database would support and to identify the data usage or "data needs" of these tasks. The functions and their related tasks can be divided into two categories: operational and control/planning.
- (2) Decompose each function into the lowest levels of work that require, on a repetitive basis, unique sets of data. Work at this level is considered a "task", a unique unit of work consisting of a set of steps performed in sequence. All these steps are directed toward a common goal and use and/or create a common set of data.
- (3) Once a task has been defined, decompose it into subtasks. This decomposition must occur if one or more of the following conditions exist:

- **Separation-Of-Duty:** This is more than one person is needed to carry out the task, and each of them is required to have a different skill and/or carries out his/her part independently.
 - **Levels of Authorization:** The following are different levels of authorization:
 - Different people authorize different parts of the task
 - Different frequencies or durations apply to different parts of the task
 - Input documents are not used uniformly within the task
 - Totally different documents are used for different parts of the task
 - Many different operations are carried out within the task
 - Different primitive operations which each have separate input/output requirements
- (4) When a sub-task has been defined, ensure it is limited to that task, and does not span two or more tasks, or it cannot be considered a subtask.
- (5) Collect all information in a precise manner using interviews and documentation techniques. This approach is important when identifying operational functions because they provide the basic input to the database design process. These functions, and their associated tasks must be identified first. Therefore, begin by identifying the organizational areas within the scope of the design effort which perform the functions essential to conducting business. Once these functional areas have been determined, the persons to be interviewed can be specified. The recommended steps are as follows:
1. Determine the key individuals within these areas, and send out questionnaires requesting job titles of persons within their areas of responsibility; functions performed in each job; and a brief statement of the objective(s) of each job.
 2. Develop a document showing job title, functions performed, and the objectives of these functions after receiving the results of the questionnaire.
 3. Review and classify each job as either operational or control and planning.
 4. Contact the supervisor of each job which is identified as "operational" and ask to select one, preferably two, persons performing that job who can be interviewed.
 5. Conduct the operational interviews. Keep the following three objectives in mind: identify each operational function; identify the data associated with each of these functions; and identify the implicit and explicit rules determining when and how each function occurs.
- (6) When conducting operational interviews, accomplish the following steps during the interviews:
1. Begin by having each interviewee describe, in detail, the functions and tasks that are performed on a daily or potentially daily basis. Document these major actions, decisions, and interfaces on task analysis and decision analysis forms. See Exhibits 2.5.13-1 and 2.5.13-2. These actions, decisions, and interfaces must also be reflected on a detailed data flow diagram. This documentation can subsequently be used to verify that all operational functions and their sequence are correct. Repeat this same procedure for functions that occur weekly, monthly, quarterly, and annually.
 2. As the functions, tasks, and other activities are defined, determine the documents, files and informal references (memos, verbal communications, etc.) used to perform them and indicate these in a separate

- numbered list. A task/document usage matrix may also be used specifying a task's inputs and outputs in terms of documents.
3. Once the person interviewed agrees to the contents of the documentation, discuss more specifically each action, decision, and interface point to determine what specific documents or references are required. Then request a copy of each document that has been discussed.
 4. Finally, identify the data elements used or created on each document, and compile a list of these elements including their definitions and lengths. See Exhibit 2.5.13-6 any data elements that are not included in the dictionary must be entered.
- (7) The second type of information required for conceptual database development involves the organization's control and planning functions and their related data needs. An in-depth investigation of the organization's explicit and implicit operating policies is necessary. Such information can be obtained through interviews with management. Since the nature of the information collected will vary according to the organization and persons involved, there is no rigid format in which the interview must be documented. However, in order to minimize the possibility of losing or missing information, it is recommended that there be two interviewers who could alternate posing questions and taking notes.
- (8) Conduct interviews for control and planning functions with persons whose responsibilities include defining the goals and objectives of the organization, formulating strategies to achieve these goals, and managing plans to implement these strategies; and with those persons directly responsible for the performance of one or more operating areas. The objective of these interviews is to gain, where appropriate, an overall understanding of:
- The basic components of the organization, and how they interact with one another
 - The external environment that affects the organization directly or indirectly (i.e., Congressional directives, Treasury policies, etc.)
 - Explicit or implicit operating policies that determine how the mission is performed; some of these may be identified when discussing the internal and external environment
 - Information used currently, or required to plan organizational activities, and measure and control performance
 - Obtain examples if available of any action items
 - Changes that are forecast that may affect the organization
- (9) The following are steps for conducting control and planning interviews:
1. Present the designer's perception of functions and operations and seek confirmation and clarification, i.e., clarify which are main functions, support functions, and sub-functions or tasks.
 2. Ask what additional functions, if any, are performed.
 3. Ask what monitoring functions are performed and what critical indicators are used to trigger intervention.
 4. Ask what planning functions are performed and what data is used for planning purposes.
 5. Express appreciation by thanking the person interviewed for his/her time.
 6. If any new data elements are defined during the interviews, ensure they are incorporated in the Enterprise Data Dictionary (EDD) so that they may be properly cross-referenced.

2.5.13.4.2.3
(09-27-2022)
**Identify Task/Data
Relationships**

- (1) The process of defining task/data relationships begins with analyzing the documentation developed during the interviews. A task/data relationship is defined as the unique relationship created between data items when they are used to perform a specific task. The process includes the following:
 - a. Collect information about data usage, and identify task/data relationships.
 - b. Identify functions and tasks as either operational, control, and/or planning, and their data usage, for the task/data relationships. It is critical that these relationships be carefully and thoughtfully defined.
- (2) Identify a series of unique tasks, and the following rules must be applied:
 - a. A task must be performed within one functional area.
 - b. Each task must consist of a set of serially performed steps (or serially positioned symbols on a DFD).
 - c. If a decision point occurs, and one path of the decision involves a new action in effect, then the current task ends and a new one begins.
 - d. Each step within a single task must be performed within a period as stated during interviews with customers. If a significant amount of time can elapse between two steps, more than one task must be defined.
 - e. Each step within the task must use the same set of data. However, if new data is created in one step of the task and used in the next step, they may be considered as the same set of data.
 - f. After all the data flows, and other documentation have been analyzed and assigned to tasks, compare the tasks for each duplicate interview to determine if the same ones were defined - This is assuming that two persons with the same job title were interviewed in each relevant area. When conflicts are found, compare the two sets of documentation to determine if one is more detailed
 - g. If the DFDs, etc., appear to be the same and differ only on levels of detail, choose the one that best defines a complete unit of work.
 - h. If real functional differences are found, review the documents (and notes) associated with each. Sometimes people with similar titles perform different functions due to their seniority, or competence. When major differences are found, separate any unique tasks and add them to the list.
 - i. If differences are found and it is difficult to determine why they exist, request that the appropriate supervisor review the task definitions developed during the interviews. (However, do not include any portions of the interviews that are confidential).
 - j. Once any conflicting definitions have been resolved, task/data relationships specifically documented. Because it is likely that redundant tasks have been defined, arrange the documentation already produced by department or area. This method increases the likelihood that redundant tasks will be identified. It is suggested that the documentation of task/data element relationships begin with a table such as the one shown in Figure 2.5.13-2. The documentation must:
 - Numerically identify each task
 - Briefly define each task by a verb-object type command (e.g., fill out error report, request alternate items, etc.)
 - Classify tasks as operational or control/planning Identify the frequency and average volume for each task
 - Relate each task to a specific functional area
 - Construct a task/data element matrix to specify each task's inputs and outputs in terms of data elements

Task/Data Element Relationships

Task #	Task Definition	Type	Frequency	Average Volume	Department	Data Elements
1	Examine order request	Operational	Daily	500	Order Entry	410, 200, 201 - 225
...
...

Figure 2.5.13-5 This is a depiction of a Task/Data Relationships

2.5.13.4.2.4
(09-27-2022)

Develop a List of Constraints

- (1) Constraints are a very important feature in a relational model based on attributes or tables. Constraints are the rules associated with a database schema that can be implicit (implied), or explicit (fully understood), and is used for optimization purposes. The purpose of developing a list of all implicit and explicit constraints is to provide information for the physical database designer to use in determining operational considerations, e.g., access restrictions, interfaces to other packages, and recovery capabilities. Constraints provide a mechanism for ensuring the data conforms to guidelines specified by the Database Administrator (DBA), and force the Database Management System DBMS to check that data meets the semantics.
- (2) Develop a list of all implicit and explicit constraints such as:
 - a. **Security Constraints:** Rules where security levels are assigned to data. They can be used as integrity rules, derivation rules, or as schema rules (data dependencies).
 - b. **Enterprise Constraints (Semantic Constraint):** An enterprise constraint are additional rules specified by users or database administrators, and can be based on multiple tables. For example:
 - One class can have a maximum of 30 students
 - One instructor can guide a maximum of four classes per semester
 - An employee cannot take part in more than five projects
 - c. **Data Integrity:** Overall accuracy, completeness, and consistency of data which refers to data safety in regard to regulatory compliance and security. The major importance of data integrity is protection against data loss or a data leak. In order to keep data safe from outside forces with malicious intent, you must ensure that internal users are handling data correctly. By implementing the appropriate data validation and error checking sensitive data is never miscategorized or stored incorrectly.
 - d. **Domain Integrity:** A set of acceptable values that a column is allowed to contain. This can include constraints and other measures that limit the format, type, and amount of data entered for processes that ensure the accuracy of each piece of data.
 - e. **Referential Integrity:** Referential Integrity requires that a foreign key must have a matching primary key, or it must be null. This constraint is specified between two tables (parent and child), and maintains correspondence between rows in these tables. See examples of referential integrity constraint in the Organization/Employee table below:

Referential Integrity Constraint

Referential Integrity Constraint
Organization(OrgID , OrgName) Employee(EmpID , OrgID, EmpDate)

- f. Response
- g. Cyclic processing time requirements

(3) Document constraints using either a tabular, or a memo format. Examples of items to be considered are:

- Access and processing cycle time requirements
- Data security needs
- Special display or calculation requirements
- Special equipment utilization

2.5.13.4.2.5
(09-27-2022)
**Develop a List of
Potential Future
Changes**

- (1) Develop a list of potential future changes, and the way in which they may affect operations. The purpose of this step is to include in the database design considerations that may affect operations in the future. Consider future changes to include anything that may affect the scope of the organization, present operating policies, or the relationship of the organization to the external environment. When reviewing the interviews to identify changes, highlight anything that implies change, and the effect(s) of that change.
- (2) For more information, see steps for Business Analysis in subsection IRM 2.5.13.4.2.

2.5.13.4.3
(09-27-2022)
Data Modeling Design

- (1) Data modeling is a technique that involves the analysis of data usage and the modeling the relationships among entities. These relationships are modeled independent of any hardware or software system. The objective of logical design depicts user perspectives of data relationships and information needs.
- (2) All IRS modernizations systems' conceptual, logical, and physical data models must be represented using the object model notation of the Unified Modeling Language (UML).
- (3) The various approaches to logical database design involve two major design methodologies-entity analysis and attributes .
- (4) The primary tool used is the data relationship diagram. This type of diagram is used to facilitate agreement between the designer, and users on the specific data relationships and convey those relationships to the physical database designer. It is a graphic representation of data relationships. The format used must be either the data structure diagram or entity-relationship diagram. See Exhibits 2.5.13-4 and 2.5.13-5.
- (5) Simplify the modeling process by partitioning the model into the following four design perspectives:
 - **Organizational Perspective:** Reflects senior and middle management's view of the organization's information requirements, and how the organization operates.
 - **Application Perspective:** Represents the processing that must be performed to meet organizational goals, i.e., reports, updates, etc.

- **Information Perspective:** Depicts the generic information relationships necessary to support decision-making, and long-term information requirements. It is represented by user ad hoc queries, long-range information plans, and general management requirements.
 - **Event Perspective:** Pertains to time and scheduling requirements. It represents when things happen, e.g., frequency of reports.
- (6) There are two general rules that provide the foundation for design perspectives:
- The design perspectives are modeled by three types of constructs:
 - ✓ **Entities:** Number of tables needed for database
 - ✓ **Attributes:** Facts necessary to describe each table
 - ✓ **Relationships:** How are the tables linked together
 - In the design perspective, each component of information must be represented by only one of these constructs
- (7) An entity refers to an object about which information is collected, e.g., a person, place, thing, or event. A relationship is an association between the occurrences of two or more entities. An attribute is a property of an entity, that is, characteristic about the entity, e.g., size, color, name, age, etc.
- (8) Data usage is dynamic, and involves not only values, but relationships. Data must be divided into logical groups before being molded into whatever structures are appropriate, entity relationship diagrams, data structure diagrams, etc. After the completion of logical modeling the designer can determine if the database approach is practical, if not take an alternative path as soon as possible to save vital resources.
- (9) Data modeling involves the following steps:
1. Identify local views of the data
 2. Formulate entities/Entity Modeling
 3. Specify relationships
 4. Add descriptive attributes
 5. Consolidate local views and design perspectives
 6. Verify the data model

2.5.13.4.3.1
(09-27-2022)

Identify Local Views of the Data

- (1) Develop local views of data for the following:
- Organization
 - Application
 - Information
 - Event Design-Perspectives
- (2) For each of the functions, activities, and tasks identified, there is a "sub-perspective" or local view of the data. Normally there will be several local views of the data depending on the perspective. These views correspond to self-contained areas of data that are related to functional areas. The selection of a local view will depend on the perspective, and the size of the functional area. Factors which must be considered in formulating local views include a manageable scope, and minimum dependence, or interaction with, other views.
- (3) The primary vehicles for determining local views are: the task/data element matrices, and the task analysis and description forms constructed during logical database analysis. See Exhibits 2.5.13-2 and 2.5.13-3.

2.5.13.4.3.2
(09-27-2022)

Formulate Entities/Entity Modeling

- (1) Entity modeling is a technique used to describe data in terms of entities, attributes, and relationships. For each local view, formulate the entities that are required to capture the necessary information about that view.
- (2) At this point the designer is confronted with two major considerations as follows:
 - a. The existence of multiple entity instances must be addressed by using the concept of "type" or "role". For example, the population of the entity EMPLOYEE can be categorized into employees of "type": computer systems analyst, secretary, auditor, etc. The generalization of these types into the generic entity EMPLOYEE will be considered in the next stage of conceptual design where user views are consolidated.
 - b. Use of the entity construct - Information can be modeled as either an "entity", "attribute", or "relationship". See Figure 2.5.13-6 example of how the designer must be guided by rules

Design Process Rules

Design Process: the designer must be guided by rules	Rules Examples
1) Two employees are married can be modeled using the entity MARRIAGE	First - Use the construct that seems most natural
2) The relationship IS-MARRIED-TO, or the attribute CURRENT-SPOUSE	Second - Avoid redundancy in the use of modeling constructs; use only one construct to model a piece of information.

Figure 2.5.13-6 This a depiction of examples for Design Process Rules

- c. If this method later proves to be wrong, it must be factored out in subsequent design steps.
- (3) The second consideration deals with the use of the entity construct itself. See Figure 2.5.13-6 "Design Process Rules" as an example.
- (4) One rule of thumb, "magic number seven, plus or minus two", this technique has been successfully used to restrict the number of entities identified so that a local view can be properly represented. This states that the number of facts (information clusters) that a person can manage at one time is about seven plus, or minus two. When this is applied to the database design process, the maximum number of entities contained in a local view must not be more than nine, but closer to six or seven. If this restriction cannot be met, the scope of the local view is too large.
- (5) Selection and assignment of an "entity name": The entity name is important when views are consolidated because that next stage deals with homonyms and synonyms. If the name given to an entity does not clearly distinguish that entity, the integration and consolidation process will carry this distortion even further.
- (6) For more, guidance on naming data elements see, IRM 2.152.3 Data Engineering, Naming Data Elements(s)/Object(s).
- (7) Identify attributes for each entity: A collection of attributes may be used as the basis for formulating entities. The significant attribute is the identifier (or primary key) that uniquely distinguishes the individual entity instances (occur-

rences),e.g., employee number. This entity identifier is composed of one or more attributes whose value set is unique. This is also important later in the consolidation phase because the identifying attribute values are in a one-to-one correspondence with the entity instances. Therefore, two entities with the same identifiers could be redundant. However, this will depend on their descriptive attributes, and the degree of generalization.

2.5.13.4.3.3
(09-27-2022)

Specify Relationships

- (1) Identify relationships between the entities. In this step, additional information is added to the local view by forming associations among the entity instances. There are several types of relationships that can exist between entities. These include:
 - Optional relationships
 - Mandatory relationships
 - Exclusive relationships
 - Contingent relationships
 - Conditional relationships
- (2) In an optional relationship the existence of either entity in the relationship is not dependent on that relationship. For example, there are two entities, OFFICE and EMPLOYEE. Although an office may be occupied by an employee, they can exist independently.



Figure 2.5.13-7

- (3) In a mandatory relationship, the existence of both entities is dependent on that relationship.

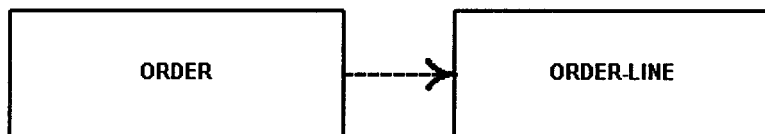


Figure 2.5.13-8

- (4) An exclusive relationship is a relationship of three entities where one is considered the prime entity that can be related to either one of the other entities, but not both.

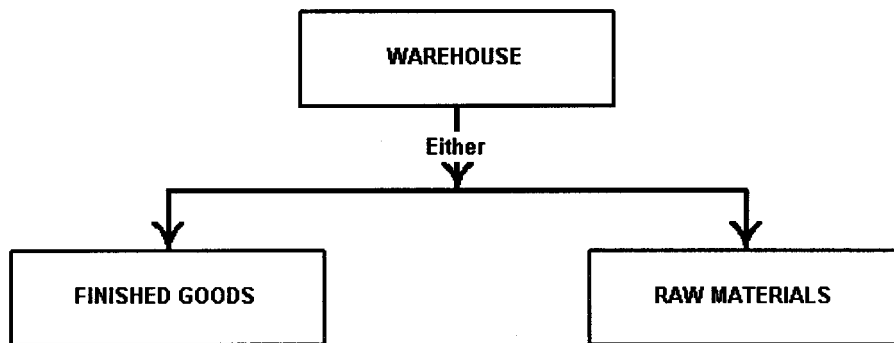


Figure 2.5.13-9

- (5) In a contingent relationship the existence of one of the entities in the relationship is dependent on that relationship. A VEHICLE is made from many PARTS.

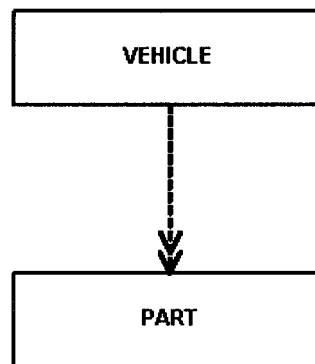


Figure 2.5.13-10

- (6) A conditional relationship is a special case of the contingent relationship. When it occurs, the arrow must be labeled with the condition of existence.

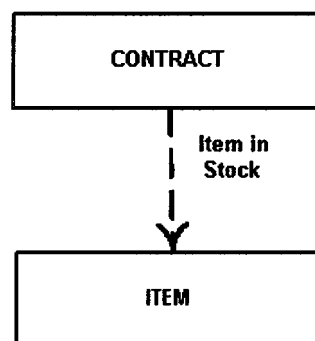


Figure 2.5.13-11

- (7) Relationships can exist in several forms. The associations can be one-to-one (1:1), one-to-many, or many-to-many (N:N), the descriptions are the following:
- One-to-One association:** Depicted as a single-headed arrow, and indicates that the relationship involves only one logical record, entity or entity class of each type.
 - One-to-Many association:** Depicted as a double-headed arrow, and documents the fact that a single entity, entity class or logical record of one type can be related to more than one of another type.
 - Many-to-Many association:** Depicted as a double-headed arrow in both directions.
- (8) When forming relationships do as follows:
- Use an informal procedure for identifying relationships by pairing each entity in the local view with all other entities contained in that view.
 - For each pair, ask if a meaningful question can be proposed involving both entities, or if both entities may be used in the same transaction. If the answer is “yes” to either question determine the type of relationship that is needed to form the association.
 - Determine which relationships are most significant and which are redundant. This can be done only with a detailed understanding of the design perspective under consideration.

2.5.13.4.3.4
(09-27-2022)
**Add Descriptive
Attributes**

- (1) Add descriptive attributes; attributes can be divided into two classes:
- Classes that identify entity instances:** Included when the entities were formulated.
 - Classes that provide the descriptive properties of entities:** Examples of descriptive attributes are: color, size, location, date, name, and amount.
- (2) In this step of local view modeling, the descriptive attributes are added to the previously defined entities. Only single-valued attributes are allowed for the description of an entity.

2.5.13.4.3.5
(09-27-2022)
**Consolidate Local Views
and Design Perspectives**

- (1) **Consolidate Local Views and Design perspectives:** Consolidation of the local views into a single information structure is the major effort in the logical database design. It is here that the separate views and applications are unified into a potential database. Three underlying concepts that form the basis for consolidating design perspectives; these concepts are:
- Identity:** Identity is a concept which refers to synonymous elements when two or more elements are identical, or have an identity relationship, if they are synonyms. Although the identity concept is simple, the determination of synonyms is not. Pertaining to inadequate data representation methods, the knowledge of data semantics is limited. Typically, an in-depth understanding of the user environments is required to determine if synonyms exist. Determining whether similar definitions may be resolved to identical definitions, or if one of the other element relationships really applies, requires a clear and detailed understanding of user functions and data needs.
 - Aggregation:** Aggregation is a concept in which a relation between elements is considered to become another higher-level element. For example, “EMPLOYEE” may be thought of as an aggregation of “NAME”,

“SSN”, and “ADDRESS”. Many aggregations are easy to identify since the major data models incorporate syntax that can represent aggregations.

- c. **Generalization:** Generalization is a concept in which a group of similar elements is thought of as a single generic element by suppressing the differences between them. For example, the entity “EMPLOYEE” may be thought of as a generalization of “FACTORY-WORKER”, “OFFICE-WORKER”, and “EXECUTIVE”. An instance of any of these three types is also an instance of the generalized “EMPLOYEE”. Care must be taken not to confuse it with aggregation. An analogy for aggregation is parts making up a “whole”, and generalization is the “whole”.
- (2) Since aggregation and generalization are similar in structure and application, one element may participate in both aggregation and generalization relationships.
 - (3) Inferences can be drawn about the aggregation dimension from the generalization dimension, and vice versa, e.g., it can be inferred that each instance of “EXECUTIVE” is also an aggregation of Name, SSN, and Address. See Figure 2.5.13-8.

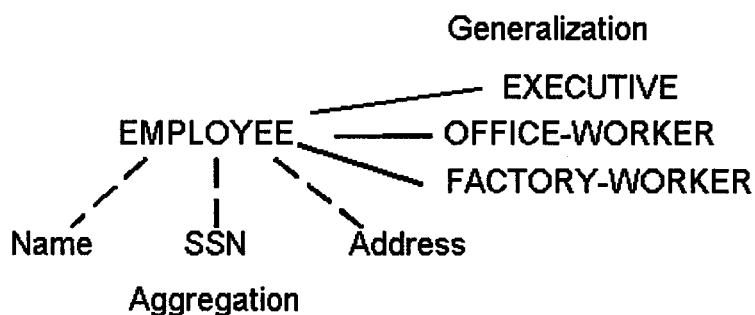


Figure 2.5.13-12 Aggregation Dimension from the Generalization Dimension

- (4) There are three consolidation types; these types may be combined in various ways to construct any type of relationship between objects (elements) in different user views. By combining consolidation types, powerful and complex relationships can be represented. Most semantic relationships are represented by some combination, and are listed as the following:
 - a. **Identity Consolidation:** Two objects may be semantically identical with the additional option of having identical names. Homonyms must be guarded against as well as similar, but not identical objects. Similarity is best expressed using aggregation and generalization as a check on the consistency of the consolidation, and user views. If an object from User view is found to be identical to an object from User second view, neither of these objects can participate further in any other identity consolidations between these two views. This is true because each object is assumed to be unique within the context of its own local user view.
 - b. **Aggregation Consolidation:** This may occur in two forms; the difference depends on whether one of the users has specified the aggregated “whole” object. An example of the simpler form is where User 1 has specified a number of objects without making any consolidation type rela-

tionships between them, e.g., an inventory view of “HANDLEBARS”, “WHEELS”, “SEATS”, and “FRAMES”. However, User 2 has specified an object, “BICYCLE”, which is an aggregation of User 1’s objects. The conceptually more difficult version of aggregation occurs when both users have specified some or all of the parts of an unmentioned “whole”. As an example, when separate inventory functions are maintained for basic, non-variable parts (FRAMES, WHEELS) and for parts that may be substituted by customer request (SEATS, HANDLEBARS). This type of aggregation is more difficult to recognize since neither user has defined a BICYCLE object.

- c. **Generalization Consolidation:** This may also occur in two forms; the difference lies in whether either of the users has specified the generalized or generic object.

(5) The consolidation process comprises four steps:

- a. **Select Perspectives:** First, confirm the sequence of consolidation by following the order of design perspectives. Since this order is general, check it against the objectives of the database being designed. For example, if you are designing a database for a process-oriented organization, you might consider the key perspectives to be the application and event perspectives and therefore begin the process with these.
- b. **Order Local Views within each Perspective:** Once the design perspectives have been ordered, focus the consolidation process on local views within each perspective. Several views comprise the perspective chosen, and this second step orders these views for the consolidation process. The order must correspond to each local view’s importance with respect to specific design objectives for the database.
- c. **Consolidate Local Views within each Perspective:** This step is the heart of the consolidation process. For simplicity and convenience, use binary consolidation, i.e., integrating only two user views at a time. This avoids the confusion of trying to consolidate too many views. The order of consolidation is determined by the previous step where the local views within a perspective have been placed in order. The process proceeds as follows:
 1. Take the top two views in the perspective being considered, and consolidate these using the basic consolidation principles.
 2. Using the binary approach, merge the next local view with the previously consolidated local views. Continue this process until the last view is merged.
 3. After the consolidation process is completed for the first design perspective.
 4. The next design perspective is introduced, and this process continues until all perspectives are integrated.

- (6) **Resolve Conflicts:** Conflicts can arise in the consolidation process primarily because of the number of people involved, and the lack of semantic power in our modeling constructs. They may also be caused by incomplete or erroneous specification of requirements. The majority of these conflicts are dealt with in the consolidation step using the rules previously discussed, any remaining conflicts that have to be dealt with by designer decisions are taken care of in this step. When a design decision is made, it is important to “backtrack” to the point in the consolidation process where these constructs were entered into

the design. At this point the implications of the design decision are considered, and their effects on the consolidation process.

2.5.13.4.3.6
(09-27-2022)
Present Data Model

- (1) The purpose of this step is to present the data model. Use data relationship diagrams to document local views and their consolidation. These must take the form of an entity-relationship diagram, or a data structure diagram. See Exhibits 2.5.13-4 and 2.5.13-5.
- (2) Use these rules when constructing either of these diagrams:
 - Each entity and relationship must be clearly labeled
 - Each entity must be related to at least one other entity
 - Show attributes only if they uniquely identify the entity, or are a common access path to the entity
 - Limit entities and relationships for a given activity to a single page
 - Identify the name of the activity supported at the top of the page
 - If a data relationship diagram for an activity needs to span more than one page, use the same off-page connectors as used in DFDs

2.5.13.4.3.7
(09-27-2022)
Verify Data Model

- (1) Verify the data model. The purpose of this step is to verify the accuracy of the data model and obtain user concurrence on the proposed database design.
- (2) The process of developing the information structure involves summarizing, and interpreting large amounts of data concerning how different parts of an organization create and/or use that data.
- (3) The design process is highly structured; therefore, take caution to not miss relationships and/or expressed them incorrectly.
- (4) The development of the information structure is the only mechanism that defines explicitly how different parts of an organization use, and manage data. Anticipate that management, with new knowledge shall require some changes. Because of this possibility, it is necessary to provide management with an understanding of the data relationships shown in the information structure, and how these relationships affect the way in which the organization performs, or can perform, its mission. Each relationship in the design and each relationship excluded from the design must be identified and expressed in very clear statements that can be reviewed and approved by management. Following management's review, the design will, if necessary, be adjusted to reflect its decisions.
- (5) The verification process is separated into two parts, self-analysis and user review. In self-analysis, the analyst must insure that:
 - All entities have been fully defined for each function and activity identified
 - All entities have at least one relation to another entity
 - All attributes have been associated with their respective entities
 - All data elements have been defined in the Enterprise Data Dictionary
 - All processes in the data flow diagram can be supported by the database when the respective data inputs and outputs are automated
 - All previously identified potential changes have been assessed for their impact on the database and necessary adjustments to the database have been determined

- (6) To obtain user concurrence on the design of the database, perform the following steps, in the form of a walk-through, to interpret the information structure for the user:
 - a. State what each entity is dependent upon (i.e., if an arrow points to it). Example: All "ORDERS" must be from "CUSTOMERS" with established accounts.
 - b. State what attributes are used to describe each entity.
 - c. Define the perceived access path for each entity.
 - d. Define the implications of each arrow (i.e., one-to-one, one-to-many etc.).
 - e. Define what information cannot exist if an occurrence of an entity is removed from the database.
- (7) Give the user an opportunity to comment on any perceived discrepancies in the actual operations or usage. If changes need to be made, then give the user the opportunity to review the full design at the completion of the changes.
- (8) After all changes have been updated for both the relationship diagram and the data definitions, obtain user concurrence on the design specifications.

2.5.13.5
(09-27-2022)
**Physical Database
Design**

- (1) The boundary between logical and physical database design is difficult to assess because of the lack of standard terminology. However, there seems to be general agreement that logical design encompasses a DBMS-independent view of data and that physical design results in a specification for the database structure, as it will be physically stored. The design step between these two that produces a schema that can be processed by a DBMS can be called implementation design. The DBMS-independent schema developed during logical design is one of the major inputs. Refinements to the database structure that occur during this design phase are developed from the viewpoint of satisfying DBMS-dependent constraints as well as the more general constraints specified in the user requirements.
- (2) The major objective of implementation design is to produce a schema that satisfies the full range of user requirements and that can be processed by a DBMS. These extend from integrity and consistency constraints to the ability to efficiently handle any projected growth in the size and/or complexity of the database. However, there must be considerable interaction with the application program design activities that are going on simultaneously with database design. Analyze high-level program specifications and program design guidance supplied to correspond to the proposed database structure.
- (3) The usefulness of these guidelines is directly related to the where one is in a development life cycle, and the level of expertise of a developer. This document assumes the reader is familiar with database concepts and terminology since designers will most likely be database administrators or senior computer specialists.
- (4) The criterion for physical design is determined by evaluating requirements e.g., operational efficiency, response time, system constraints and security concerns. This physical design layout must be routinely adjusted to improve the system operation, while maintaining the user's logical view of data. The physical structuring, or design will often be quite different from the user's perception of how the data is stored.

- (5) The following steps provide general guidance for physical database design. Since much of the effort will depend on the availability of data and resources, the sequence of these steps is flexible:
- Determine all user requirements.
 - Determine all relevant processing environments.
 - Select appropriate IRS database management system software.
 - Design the physical placement of data.
 - Perform sizing of data.
 - Use database security controls based on IRM 10.8.21 “Information Technology (IT) Security, Database Security Policy”.
 - Implement or use the recover procedures based on NIST SP 800-34 “Contingency Planning Guide for Federal Information Systems and policy” and IRM 10.8.60 “IT Security, IT Service Continuity Management (ITSCM) Policy”.
- (6) See Figure 2.5.13-13 for comparison of logical and physical design:

Logical Design Compared with Physical Design

Logical Design	Physical Design
Entity	Table
Relationship	Foreign Key
Attribute	Column
Unique Identifier	Primary Key

Figure 2.5.13-13#

#

2.5.13.5.1
(09-27-2022)
Determine the User’s Requirements

- (1) User(s)’ requirements are critical factors that add value to a product, service or environment. Fulfilling user requirements is a process of engaging users to understand their problems, process, goals and preferences. The following are user requirement examples:
- **Accessibility:** User requirement of 99.99% availability
 - **Accuracy:** Data initiatives must be correct
 - **Capacity:** Storage of Big data
 - **Compatibility:** Product or service must be compatible with other services and/or products
 - **Convenience:** Any element that saves the customer time and effort
 - **Comfort:** Physical design that is easier to understand, or better for vision, i.e., 508 compliance
 - **Durability:** A design must have stress testing to eliminate breakage
 - **Efficiency:** The processes’ desired outcome for resource constraints
 - **Features:** Preferred methods for serving needs
 - **Functions:** Functional requirements the user needs to perform work, e.g., a system may be required to calculate and print budgets
 - **Integrated Services:** Automatic availability of diverse multiple devices
 - **Performance:** At a minimum a set of performance requirements must be documented as follows:

- ✓ Maximum response time to be experienced for each user-computer interaction. Response time is measured from the time the user starts their action until feedback is received from the computer
 - ✓ Throughput required, and the time it will take place, e.g., the requirement for one program could be for it to run twice a day at a specific time
 - ✓ The size and time of maximum-throughput periods
 - ✓ Response time that is minimally acceptable the rest of the time. Response time degradations can cause users to think the system is down
 - **Readability:** Users prefer information, and visual data that is concise and easily understood
 - **Refinement (Schema):** Checking tables for redundancies and anomalies
 - **Reliability:** Product must perform consistently
 - **Risk:** Requirement to reduce risk' e.g., being able to roll-back the database, or obtain a backup
 - **Trainable:** User interfaces must be user-friendly
 - **Stability:** the system environment must be stable, and not susceptible to crashing
 - **Visual Appeal:** and Users strongly prefer products, services, and environments that are aesthetically appealing
- (2) Decision making, and the impact of approving user requirements without having all the facts could have negative consequence; therefore, performing an analysis is necessary before acting. Figure 2.5.13-14 displays each requirement, and an example of an associated result:

Example of User Requirements and Consequences

Requirements	Consequences
Retrieval time decreases with a simple database structure	To meet the logical design requirements, it may be necessary to implement a more complex multilevel structure.
Ease of recovery increases with a simple structure	Data relationships may require more complex mechanisms
Increased pointer or index requirements, hardware cost is increased if information is spread over many storage devices	Data clustering and compacting degrade performance
Privacy requirements may require stringent security, e.g., encryption, or data segmentation	These procedures decrease performance in terms of update, and retrieval time
Active files, that are accessed in real time, and need high-speed devices	The result is increased cost

Figure 2.5.13-14

2.5.13.5.2

(09-27-2022)

Determine the Processing Environment

- (1) To determine the primary type of processing, the designer must have a “framework of physical requirements”. Five environments are discussed; however, these are only guidelines because for some systems this information will not be applicable as a set of requirements. These considerations could conflict with user’s requirements or security needs; therefore, forcing the designer to make decisions regarding priority.
 - a. **Fast Response Time Environments:** This could have multiple run units actively sharing DBMS facilities, and in order to meet response time specifications, cost may increase due to the necessity for additional system resources. Recovery may be critical in such a volatile environment; whenever possible, use a simple structure.
 - b. **CODASYL (network) Structures:** Time specification would translate into reduced data levels, number of network relationships, number of sets and size of set occurrences.
 - c. **High-volume Processing Environment Requests:** Most frequently random in nature requiring small amounts of information transfer; thus affecting page and buffering considerations
 - d. **Low-volume Systems** - Generally process more data per request, indicating run units may remain in the system longer because of more sequential requests and reports, and response time is probably not the critical issue. Resources may be more limited in this environment, implying smaller buffers, and perhaps fewer peripherals. With the possibility of fewer resources, those resources may need to be more highly utilized. On-line recovery techniques may be unavailable since the resource requirements are costly. Although the number of transactions is low in this environment, the probability of multiple simultaneous run units accessing the same data may be high.
 - e. **Batch Environments:** When this is indicated, the designer is left with maximum flexibility since the requirement has reasonable turnaround time and effective use of resources. Because of job scheduling options, concurrency problems can be controlled. Recovery is less critical and is determined by factors such as file volatility, the time necessary to rerun update programs, and the availability of input data. For example, if the input data is readily available, the update programs short and processing 85% retrieval; the choice may be made to avoid the overhead of maintaining an on-line recovery file.

2.5.13.5.3

(09-27-2022)

Select DBMS Software

- (1) The DBMS must first physically support the logical design requirement that is, based on the logical data model, the package must support the required hierarchical, network or relational structure. Early stages of analysis must provide enough information to determine this basic structure. From a physical database design point of view, an analysis must then be made as to how effectively the DBMS can handle the organizational and environmental considerations. If the proposed package fails to provide adequate support of the requirements, the project manager, and supervisor(s) must be notified. The notification must include the specific point(s) of failure, anticipated impact(s), and any suggestions or alternatives for alleviating the failure(s).

- 2.5.13.5.4
(09-27-2022)
Design the Physical Placement of Data
- (1) Designing the placement of data involves selecting the physical storage, and access methods as well as secondary and multiple key implementation techniques. DBMS packages vary as to the options offered. The use of vendor documentation, providing specific software handling details, will be necessary to complete this process. Exhibit 2.3.13-8 provides a summary of access methods and uses.
- 2.5.13.5.5
(09-27-2022)
Perform Sizing of Data
- (1) Obtain the specifics of sizing from vendor documentation as each DBMS handles space requirements in a different manner. Consider sizing in conjunction with designing the placement of data. Once data records, files, and other DBMS specifics have been sized according to a proposed design, a decision may be made, because of the space allocation involved, to change the design. Data compaction techniques may be considered at this point. Flexibility to make changes and reevaluate trade-offs during this entire procedure is of critical importance.
- (2) For an example, see Exhibit 2.5.13-14
- 2.5.13.5.6
(09-27-2022)
Consider Security and Recovery
- (1) The DBMS selected must have the options necessary to implement security and recovery requirements from IRM 10.8.21 IT, Security, Database Security Policy, and IRM 10.8.32 IBM Mainframe System Security Requirements. Implementation of these considerations will often cause trade-offs in other design areas.
- 2.5.13.6
(09-27-2022)
Deliverables
- (1) The deliverables for database design are the following:
- a. Decision Analysis and Description Forms
 - b. Task Analysis and Description Forms
 - c. Task/Data Element Usage Matrix
 - d. Data Models
 - e. Entity-Attribute Lists
 - f. Data Definition Lists
 - g. Physical Data Base Specifications Document
- 2.5.13.6.1
(09-27-2022)
Decision Analysis and Description Forms
- (1) Decision Analysis and Description Forms must identify such items as type of decision, the decision maker, and the nature of the decision. Exhibit 2.5.13-1 provides guidelines and shows a sample form.
- 2.5.13.6.2
(09-27-2022)
Task Analysis and Description Forms
- (1) Task Analysis and Description Forms must include the name of the task, its description (overview), the people/departments involved, and subtasks and their relationships. Exhibit 2.5.13-2 shows a sample form and provides guidelines.
- 2.5.13.6.3
(09-27-2022)
Task/Data Element Usage Matrix
- (1) A task/data element usage matrix relates each data element to one or more tasks, see Exhibit 2.5.13-3 for a sample matrix.

2.5.13.6.4
(09-27-2022)
Data Models

- (1) Data relationship diagrams depict the relationships between entities. These are tools that provide one way of logically showing how data within an organization is related. They must be models using conventions for either data structure diagrams, or entity relationship diagrams. Exhibits 2.5.13-4 and 2.5.13-5 provide samples of these diagrams, and more detailed guidelines. The term "entity" refers to an object representing a person, place, thing event about which information is collected. When constructing either of these diagrams it is recommended that the entities be limited to those of fundamental importance to the organization.

2.5.13.6.5
(09-27-2022)
Entity-Attribute Lists

- (1) Entity-attribute relation lists may be derived from the Enterprise Engineering Data Dictionary listings.

2.5.13.6.6
(09-27-2022)
Data Definition Lists

- (1) Data definition lists may be derived from Enterprise Engineering Data Dictionary listings. Exhibit 2.5.13-6 provides a sample.

2.5.13.6.7
(09-27-2022)
Physical Database Specifications Document

- (1) The objective of this subsection is to state the required content for Physical Database Specifications. Duplication of effort can be eliminated if there are existing documents available containing physical specifications. Use the following resources for developing documentation:
 - **DBMS Documentation:** For example, a listing of the scheme from a CODASYL DBMS will provide details as data names, sizing, placement, and access methods.
 - **Data Dictionary Listing:** Provides certain physical specifications, e.g., data format and length.
 - **Project Documentation:** All documentation submitted as Physical Database Specifications must be organized in a macro to micro manner, or global to specific. Hence, begin at the schema level, moving to subschema, indices, data elements, etc. The objective is to organize documentation in a manner that is clear, and easy for the user to read.
 - Adhere to the standards and guidelines

2.5.13.6.7.1
(09-27-2022)
Physical Database Names

- (1) Where appropriate in the documentation, identify names of physical database items. Specifically, the items will be all those defined to the DBMS software, such as:
 - Attribute Name
 - Subject Area Name
 - Super Class or Class
 - Schema
 - Subschema
 - Set
 - Record
 - Field
 - Key
 - Index Names
 - Relationship Name
- (2) Be atomic, and represent only one concept.
- (3) Be unique, avoid adding a name like that of another.

- (4) Be concise, using minimal words as possible.
- (5) Table and column names must be sized to fit the requirements of the target RDBMS. Different database products permit different lengths.

#

- (7) Where data naming standards are applicable, these standards shall be met to the extent possible with the DBMS software. For example, if the DBMS does not permit hyphens in naming, an exception would be made to the standard "all words in a name must be separated with a hyphen".
- (8) For more guidance on data naming see, IRM 2.152.3 IT Data Engineering, Naming Data Elements/Object.
- (9) For definition of all terms listed, see Exhibit 2.5.13-11

2.5.13.6.7.2
(09-27-2022)**Data Structure/Sizing**

- (1) Identification of data elements, associations within and between record types as well as sizing requirements are identified and documented during the logical database design process. The physical representation of data structures will vary from the logical, however, since the physically stored data must adhere to specific DBMS characteristics. As applicable to the DBMS, the following structures must be documented:
 - Records
 - Blocks/Pages
 - Primary Key (PK) attributes
 - Files
- (2) Describe the physical record layout. In this description, include the data fields, embedded pointers, spare data fields, and database management system overhead (flags, codes, etc.). Besides data record types, document any other record types such as index records. If records are handled as physical blocks or pages, provide the following:
 - Calculations determining block/page size
 - Total number of blocks/pages allocated
- (3) Describe the strategy for determining block/page sizes.
- (4) Specify the amount of space allocated for each database file. This must be consistent with the total record, and block/page sizing documentation described above.

2.5.13.6.7.3
(09-27-2022)**Data Placement**

- (1) For each record type:
 - State the storage and access method used
 - Describe the storage and assess method
 - Where applicable, identify the physical data location (track, cylinder)
- (2) When using an algorithm access method:
 - Provide the primary record key to be used by the algorithm

- Describe the algorithm used, including the number, and size of randomized address spaces available to the algorithm
- Give the packing density, and the strategy for its determination

(3) When using an index sequential access method:

- Provide the primary record key
- State indexing strategy/levels
- State initial load strategy

(4) When using chains access method:

- Provide the access path to the record type (i.e., Is this primary access of detail record though the master record, or is this a chain of secondary keys?)
- List the pointer options used (i.e., forward, backward, owner, etc.)
- Indicate whether the chain is scattered or stored contiguously with the master

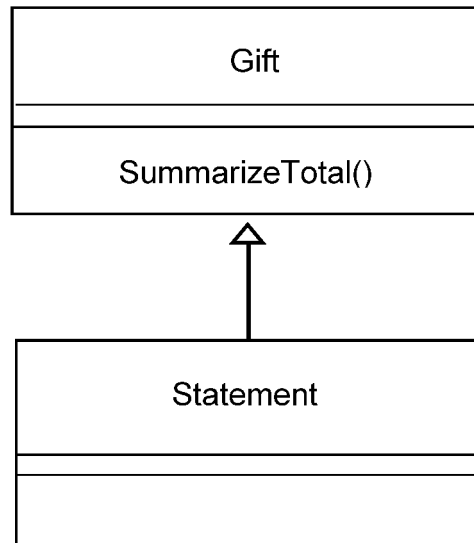
(5) When an index access method is used:

- Identify keys used to index a record type

2.5.13.6.8
(09-27-2022)
Database Schema
Refactoring Overview

- (1) Refactoring is a way to restructure code in small steps, and enables the programmer to evolve the code to slowly over time. This is an iterative and incremental approach to programming.
- (2) Another very important aspect of refactoring is that it retains the behavioral semantics, and improves the design of your code. Functionality is not added when refactoring, nor is it taken away. An example of refactoring is to rename the getBoxes() operation to getPallets().
- (3) See another example in Figure 2.5.13-15 when applying a Push Down Method refactoring to move the “SummarizeTotal()” operation from “Gift” into its subclass “Statement” as seen you will need to change the code that invokes this operation to with “Statement ”objects rather than “Gift” objects. After these changes are done, your code is refactored.

Push Down Method Example (Before)



Push Down Method Example (After)

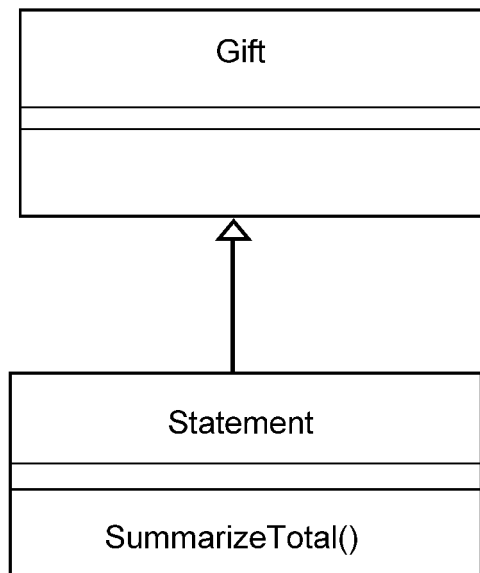


Figure 2.5.13-15

2.5.13.6.8.1
(09-27-2022)

Evolutionary Database Techniques

- (1) Evolutionary database techniques represent database processes that must be developed both iteratively and incrementally. Examples of these processes are:
- Agile Unified Process (AUP)
 - Dynamic System Development Method (DSDM)
 - Extreme Programming (XP)
 - Enterprise Unified Process (EUP)
 - Rapid Application Development (RAD)
 - Scrum

- Safe
- (2) Database development iterative work is considered performing small stages of repeatable processes for modeling, testing, coding, and organized into them series of releases. Database developers need to adopt the evolutionary techniques similar to those of developers (Thinking Agile). This is done by accomplishing the following:
- a. Developers must be proficient at data techniques, and data professionals must learn modern development technologies and skills for the purpose of having a better understanding of each other requirements.
 - b. One challenge of agile adoption for databases is unlike applications it has a current state that must always be managed to maintain the integrity of the data. However, despite this challenge development teams need to implement the proper tools and processes for automating database development and release tasks. If the entire software stack including the database is evolving incrementally, flexibly, and rapidly the organization could receive a better return on investments and shorter response time for customer requirements.

2.5.13.6.8.2
(09-27-2022)

- (1) Database categories for refactoring is comprised as seen in Figure 2.5.13-16

Database Refactoring Categories

Refactoring Categories

	Database Refactoring Category	Description	Examples
1	Structural	A modification to the definition of one or more tables or views.	Moving a column from one table to another or splitting a multipurpose column into several separate columns, one for each purpose.
2	Data Quality	A modification that improves the quality of the information contained within a database	Making a column non-nullable to ensure that it always contains a value or applying a common format to a column to ensure consistency.
3	Referential Integrity	A modification that ensures that a referenced row exist within another table and/or ensures that a row that is no longer needed is removed appropriately.	Adding a trigger to enable a cascading delete between two entities, code that was formerly implemented outside of the database.
4	Architectural	A modification that improves the overall manner in which external programs interact with a database.	Replacing an existing Java operation in a shared code library with a stored procedure in the database. As a stored procedure it is available to non-Java applications.

	Database Refactoring Category	Description	Examples
5	Method	A modification to a method (a stored procedure, function, or trigger) that improves its quality. Many code refactorings are applicable to database methods.	Renaming a stored procedure to make it easier to understand.
6	Non-Refactoring Transformation	A modification to your database schema that changes its semantics	Adding a new column to an existing table.

Figure 2.5.13-16

2.5.13.6.8.3
(09-27-2022)

Common Indicators for Refactoring

- (1) Fowler (1997) presented the concept of a “code smell,” which is a common category of problems in your code that signals the need to refactor it. Common code smells include switch statements, long methods, duplicated code, and feature envy.
- (2) Common database problems that indicate the need to refactor it are as follows:
 - a. **Multipurpose Column** - If a column is being used for several purposes, i.e. used for someone’s birth date as a customer or start date if that person is a employee.
 - b. **Redundant Data** - Redundant data is a serious problem because the same data is stored in numerous places, and causes inconsistencies with the outcome of erroneous information.
 - c. **Tables with Too Many Columns** - When a table has many columns, it indicates that the table lacks cohesion— it is trying to store data from numerous entities. For example the Employee table containing columns to store three different addresses or several phone numbers could require the structure to be normalized by adding “Address” and “Phone” tables.
 - d. **Table with Too Many Rows** - Large tables are suggestive of performance problems because it becomes time- consuming to search a table with thousands of rows. The solution is to split the table vertically by moving some columns into another table.
 - e. **Smart Columns** - A smart column is when columns different positions within the data represent different concepts. For example if the first four digits of the personnel ID indicate the personnel’s home branch, then personnel ID is smart column because you can parse it to discover more granular information.
 - f. **Fear of Change** - “Fear of Change” can stop developers from taking action, lead to “Resistance of Change” and could result in unmanageable systems. For example, the fear of unwanted side effects when cleaning up functioning components.

2.5.13.6.8.4
(09-27-2022)

Database Refactoring Best Practices

- (1) This section provides database refactoring best practices and strategies as the following:
 - a. Data professionals must be knowledgeable with techniques that enable them to work in an evolutionary way. Evolutionary database development techniques are as follows:

- **Database Refactoring** - Develop an existing database schema a small portion at a time to improve the quality of its design without changing the semantics
 - **Evolutionary data modeling** - Model the data aspects of a system iteratively and incrementally as with other aspects of a system, to ensure that the database schema evolves in step with the application code
 - **Database Regression Testing** - Ensure the database schema works, if it does not roll back the changes
 - **Configuration Management of Database Artifacts** - Database models, database tests, test data, etc. must be managed like the Enterprise Life Cycle documentation
 - **Developer Sandboxes** - When possible, provide separate sandboxes for developers; they need their own working environment where they can modify the portion of the system they are building, and get it working
- b. Make smaller changes to your database because It is easier and less risky to apply, and if something is broken the change can be identified and rolled back faster.
 - c. Identify individual refactorings distinctly because refactorings normally build upon each other and you need to ensure they are applied in the correct order and add any dependencies between them.
 - d. When in a multi-application environment in which multiple project teams may be applying refactoring to the same database schema, identify which team produced a refactoring i.e., team 1 could have refactorings identification ((ID) 01 - 07). See Version Identification Strategies in Figure 2.5.13-17

Version Identification Strategies

Approach	Description	Advantages	Disadvantages
Build Number	The application build number which is normally an integer number that is assigned by your build tool whenever your application compiles, and all your unit tests run successfully after a change	<ul style="list-style-type: none"> • Easy strategy • Refactorings can be treated as First In, First Out (FIFO) queue to be applied in order by the build number • Database version can be directly linked to the application version 	<ul style="list-style-type: none"> • Many builds do not involve database changes; therefore, the version identifiers are not contiguous for the database • The build is difficult to manage when you have multiple applications being developed against the same database because each team will have the same build numbers

Approach	Description	Advantages	Disadvantages
Date/Timestamp	The current date/time is assigned to the refactoring	<ul style="list-style-type: none"> • Easy strategy • Refactorings managed by FIFO queue 	<ul style="list-style-type: none"> • A script-based approach to implementing refactorings using a date/timestamp for a filename can be inconvenient • A strategy is needed to associate the refactorings with the appropriate application build
Unique Identifier	A unique identifier, such as a Globally Unique Identifier (GUID) or an incremental value, is assigned to the refactoring	May use existing strategies for generating unique values, e.g. Globally Unique Identifier (GUID) generator	<ul style="list-style-type: none"> • GUIDs are controversial names • A strategy is needed to associate the refactorings with the appropriate build

Figure 2.5.13-17

- e. Implement a large change by creating many small increments. For example, when applying a consistent naming strategy throughout your database consider splitting an existing table into two.
- f. Create a database configuration table for identifying the current schema version of the database to enable you to update the schema. For example, if you identify the refactorings using a date/timestamp strategy you must identify the current schema version the same way.

2.5.13.6.8.4.1
(09-27-2022)

Database Refactoring Preparation Process

- (1) An analysis of the database table structure must be completed to ensure refactoring is appropriate for your project. The database refactoring process is as follows:
 - a. Verify that a database refactoring is relevant.
 - b. Deprecate the original database schema: support both the original and new schemas in parallel to provide time for refactoring.
 - c. Test before, during, and after refactoring. Write test for:
 - Testing the database schema by writing database oriented test with: stored procedures and triggers, Referential Integrity (RI) rules, View definitions, default values, and data invariants (forms of constraints)
 - Testing the way your application uses the database schema
 - Validating your data migration
 - Testing your external program code
 - d. Modify the database schema.
 - e. Modify external access programs(s) when necessary.
 - f. Run regression tests.
 - g. Version control your modified work.

2.5.13.6.8.4.2
(09-27-2022)

Database Refactoring Strategies

- (1) This section includes various strategies for implementing database refactoring. The strategies are as follows:
- a. Apply smaller changes, implementing smaller steps will decrease project risk, and less likely to have defects.
 - b. Ensure individual refactorings are uniquely identified so they are applied in the right order.
 - c. Ensure you have a build number strategy process to identify teams when you are involved in a multi-application environment where multiple project teams are applying refactorings to the same database schema
 - d. Implement a large change by breaking by up into increments. For example, instead of splitting a table into two refactoring is done multiple times for introducing new tables, moving columns, implementing a primary key, etc.
 - e. Ensure a database configuration table is implemented. This is for identifying the current schema version of the database. See example database configuration table:

Implementation of a Database Configuration Table

Refactoring Number 21 to Schema

```
CREATE TABLE DatabaseConfiguration
(SchemaVersion NUMBER NOT NULL);
```

```
INSERT INTO DatabaseConfiguration
(SchemaVersion ) VALUES (0);
```

```
UPDATE DatabaseConfiguration
SET SchemaVersion = 21;
```

Figure 2.5.13-18

- f. Use triggers over views or Batch synchronization when possible. For benefits see Figure 2.5.13-19

Schema Synchronization

Strategy	Description	Advantages	Disadvantages
Trigger	One or more triggers are implemented that make the proper update to the other version of the schema	Real-time update	<ul style="list-style-type: none"> • Possible performance bottlenecks • Possibility of trigger cycles • Possible deadlocks • Commonly introduces duplicate data (data is stored in both the original and new schema)
Views	Representing the original table (s) are introduced	<ul style="list-style-type: none"> • Real-time update • No need to move the physical data between tables/ columns 	<ul style="list-style-type: none"> • Updatable views are not supported by some database, or the database does not support joins within an updateable view • Complexity of introducing and removing views

Strategy	Description	Advantages	Disadvantages
Batch Updates	A batch job that processed and updates the data accordingly is run on a regular basis	Performance impact from data synchronization is absorbed during non-peak loads	<ul style="list-style-type: none"> • Great potential for referential integrity problems • Must keep track of previous versions of data to determine which changes were made to the record • Often creates duplicate data

Figure 2.5.13-19

2.5.13.7

(09-27-2022)

**Database Management
System Software
Supported by the IRS**

- (1) The IRS supports various database management software which is installed on both client-server application systems, and legacy application systems with features for:

- a. Data storage
- b. Data backup and recovery
- c. Data presentation and reporting
- d. Data security management
- e. Database communication
- f. Multi-user access Control

#

- (3) Database software is classified into six sub-types:

- a. **Analytical Databases:** Allows users to pull data from a variety of databases, and examine them for the purpose of quantifying, or assessing performance of the business environment.
- b. **Data Warehouse Databases:** Allows users to pull key data from a variety of databases, and store it in a central location for reporting, or other purposes.

- c. **Distributed Databases:** Pertains to centralized database management systems that controls information stored in a variety of locations including cloud, or network servers.
- d. **End-user Databases:** Stores information that is used primarily by one person, e.g., spreadsheets.
- e. **External Databases:** Compiles information that must be accessed by a variety of users via the Internet.
- f. **Operational Databases:** Allows the user to modify data in real-time.

#

2.5.13.7.1
(09-27-2022)
**IBM Enterprise Database
2 Universal Database
(DB2 UDB) Overview**

- (1) DB2 is a Relational database Management System product of IBM which was release during 1996 for distributed platforms, and designed to store, analyze and retrieve data. The Universal Database (UDB) DB2 Server can run on any operating systems such as Linux, UNIX, and Windows. During 2016, IBM released DB2 11.1 with enhanced analytics, increased availability, reliability, and included more security for business applications.

2.5.13.7.1.1
(09-27-2022)
DB2 Physical Objects

- (1) The Enterprise Data Standards and Guidelines Office (EDSG) developed twelve (12) DB2 unique DBMS objects standards and guidelines. A data element entered is not correct, or complete unless there are entries in all mandatory fields. See Figure # # 2.5.13-20 # below:

#

				# #
				# # # # # # #
				# # # # # #
				# # #
				# # # # #
				# # # # # # #
				# # # # # # # # #
				# # # # # # # # #
				# # # # # # # # #

[illegible]

2.5.13.7.1.2
(09-27-2022)
**DB2 Performance
Standards and
Guidelines**

- (1) During the initial DB2 environment design process of creating a strategy for quality performance, the focus must be on DB2 database, applications, and/or transactions that have the most importance workload. If the performance of the applications accessing the database is addressed after development, it will be more difficult, and time-consuming to make the modifications to obtain adequate response-times.
- (2) When designing for performance, objectives must be realistic when determining what applications have the most vital workloads, and be in-line with the organization's expectations based on the following characteristics:
 - a. Data processing needs and priorities
 - b. Largest percentage of the total business workload
 - c. Critical response time requirement
 - d. Data access requirements and/or complex logic
 - e. Using vast amounts of resources (CPU, memory, Input/Output)

- 2.5.13.7.1.2.1
(09-27-2022)
**IRS DB2 Tables -
Designing for
Performance**

- (1) This subsection specifically addresses the best organization of database tables, columns, and optimum index definitions for enterprise DB2 performance established by the EDMO Database Administration team. The goal is to ensure high quality, and efficient DBMS throughout the agency. See Figure # # 2.5.13-21 # for standards:

##

#

#

#

#

##

				# # # # #
				# # # # #
				# # # # # #
				# # # # #
				# # #
				# # #
				# # #
				# # # # # #
				# # # # #
				# # # #
				# # # # #
				# # # #
				# # # # #
				# # # # #
				# # # #

				# #
				# #
				# #
				# # # #
				# # # #
				# # # #
				# # # #
				# # #
				# # #
				# # #
				# # #

Note: Important: Using the wrong data type for a column can mislead the DB2 optimizer. Cardinality could escalate, and the DB2 optimizer might not make the correct decision when determining the path to the data.

2.5.13.7.2
(09-27-2022)
**Structured Query
Language (SQL) Server
Overview**

- (1) Structured Query Language (SQL) Server is a relational database management system, or RDBMS, developed by Microsoft. SQL Server is built on top of SQL, a standard programming language for interacting with the relational databases. Transact-SQL (T-SQL) is the main procedural language used by Microsoft in SQL Server. It contains various extensions of standard SQL, i.e., local variables, control statements, transaction control, built-in functions, bulk insert, and options on the DELETE and UPDATE statements. T-SQL is proprietary while SQL is open format.
- (2) As of 2016, SQL Server 2017 became available, and runs on both Windows and Linux environments.

	#
	#
	#
	#
	#

	#
	#
	#
	#
	#
	#
	#
	#
	#
	#
	#
	#
	#
	#
	#
	#
	#

- 2.5.13.7.2.1
(09-27-2022)
T-SQL Function Types
- (1) T-SQL provides different support functions for string processing and data processing. There are four types of functions:

a. **Aggregate:** Operates on a collection of values, but returns a single value.
b. **Ranking:** Returns a ranking value for each partitioning row.
c. **Rowset:** Returns an object that can be used in a place of table reference in the SQL statement.
d. **Scalar:** Operates on a single value and returns a single value.

- 2.5.13.7.2.2
(09-27-2022)
SQL Server Data Types
- (1) SQL Server data type is a feature that specifies types of data of any object. Each column, variable, and expression has related data type in SQL Server, and can be used when creating tables. The seven categories of data types are the following:

a. Exact Numeric Types:

Exact Numeric		
Type	From	To
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10 ³⁸ +1	10 ³⁸ –1
numeric	-10 ³⁸ +1	10 ³⁸ –1

Type	From	To
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

Figure 2.5.13-24 This table depicts the Exact Numeric Type

- b. **Approximate Numeric:** Uses Real and float.
- c. **Date and time:** Uses DateTime, Date, Datetime2, DateTimeOffset, Smalldatetime, Time.
- d. **Character strings:** Uses Char, Varchar, Text.
- e. **Unicode character strings:** Uses Nchar, Nvarchar, Ntext.
- f. **Binary strings:** Uses Binary, image and varbinary.
- g. **Other data types:** Pertain to Cursor, Hierarchyid, sql_variant, Table, Rowversion, Uniqueidentifier, XML, Spatial and Geography.
- h. For additional information see, <https://www.sqlshack.com/an-overview-of-sql-server-data-types/>.

2.5.13.7.2.3
(09-27-2022)
**SQL Server and
Transact-SQL (T-SQL)
Best Practices**

- (1) SQL is not only for writing queries, you also need to ensure your queries have good performance, fast and readable. This subsection describes best practices that database designers and database development team must follow to create coding standards and conventions in SQL Server database and/or T-SQL.
- (2) The following rule is when creating a query in a relational database:
 - a. Use only T-SQL syntax for supporting SQL Server.
 - b. Use built-in and user-defined functions in SQL Server instead of third party functions for the reason of portability, and ease of use.
 - c. Use End comment text which states a change in content with a period.
 - d. Add a new line if the length of comment text is greater than 100 characters.
- **T-SQL Select Command:** Do not use “*” in your SELECT queries instead of specifying name of columns

#

2.5.13.7.3
(09-27-2022)
MySQL Overview

- (1) MySQL was purchased by Sun Microsystems in 2008, and is open source with premium add-ons. MySQL is used for website, and as a back-end database solution for enterprise applications.
- (2) During 2019 MySQL 8, included features such as NoSQL document store for Big Data. This includes better sorting, support for partial updated crash-safe DDL sentences, and JSON extended syntax.

2.5.13.7.4
(09-27-2022)
MongoDB Overview

- (1) MongoDB is an open-source document oriented database built on a horizontal scale-out architecture that uses a flexible schema for storing data. MongoDB is written in C++, and was founded in 2007. MongoDB has a worldwide following in the developer community.
- (2) MongoDB is cross-platform database which means it can run on diverse operating systems (Windows, Linux, and Mac) and computer architectures..

- (3) MongoDB allows you to immediately start building your application without spending time configuring a database. MongoDB was built on a scale-out architecture, a structure that allows many small machines to work together to create fast systems and handle huge amounts of data.
- (4) Instead of storing data in tables of rows or columns like SQL databases, each record in a MongoDB database is a document depicted in a Binary JSON (BSON) and contains extensions that allow representation of data types that are not part of JSON. Applications can retrieve this information in a JavaScript Object Notation (JSON) format. This means the MongoDB document is a dictionary of key-value pairs, where the value may be one of the number of types:
 - a. Primitive JSON types (e.g., number, string, Boolean)
 - b. Primitive BSON types (e.g., datetime, ObjectId, UUID, regex)
 - c. Arrays of values
 - d. Objects composed of key-value pairs
 - e. Null
- (5) MongoDB has the following key capabilities:
 - a. **Ad-hoc queries:** Allows developers to update ad-hoc queries in real time.
 - b. **Indexing:** Indexes allow for better query executions with quicker search speed and better performance. Without this feather, a database will be forced to scan each document to match query statements.
 - c. **Replication:** Mitigates vulnerabilities (e.g., multiple point-of-failures, such as a server crash or server interruptions) by deploying multiple servers for disaster recovery and backup.
 - d. **Sharding:** The process of splitting larger datasets across multiple distributed collections/shards assist with the database distribute and better execute problematic and cumbersome queries.
 - e. **Load balancing:** Large-scale database management for growing enterprise applications through horizontal scaling features like replication; therefore, there's no need to add an external load balancer.
 - f. **Consistency, Availability, Partition tolerance (CAP) theorem:** This enables greater flexibility in building a transactional data model that can horizontally scale in a distributed environment and has no impact on performance for multi-document transactions.
 - g. **Best Portability:** Runs the same everywhere, whether a local server or cloud and is available globally as a service with cloud platforms..
- (6) See example of information in a documents that display why normalization is not required as follows:

Example of MongoDB Document Format

Example of Document for Phone Contact

```
{
  "id_": 10,
  "name": "Tammy",
  "zip_code": "19923",
  "phone": [ "453-234-1999", "453-234-1966" ]
}
```


- (7) Cloud computing is the perfect choice for MongoDB because cloud-based storage needs to distribute data across multiple servers easily.

2.5.13.7.4.1
(09-27-2022)

MongoDB Features and Benefits over RDBMS

- (1) MongoDB works on the concept of collection and document, and the database is the physical container for collections. Figure 2.5.13-25 display the relationship between Relational Database Management System (RDBMS) and MongoDB:

MongoDB to RDBMS Comparison

MongoDB (NoSQL database)	SQL database RDBMS
Database	Database
Collection based and key value pair- a group of MongoDB documents	Table based
Document based - a set of key-value pairs	Tuple/Row based
Field	Column based: each column represents an attribute
Embedded Documents	Table Join
Contains a predefined schema	Contains a dynamic schema
Primary Key (Default key_id provided by MongoDB itself)	Primary Key or Key-value
No support for foreign key	Support for foreign key
No support for triggers	Support for triggers
Schemaless	Schema - predefines model for database and data structures
Best fit for hierarchical data storage	Not fit for hierarchical data storage
Horizontally scalable - adds more servers	Vertically scalable - increases RAM
Emphasizes the CAP theorem (Consistency, Availability, and Partition tolerance) potential for being ACID compliant	Emphasizes ACID properties (Atomicity, Consistency, Isolation, and Durability)
Database Server and Client	
mongod	mysqld/Oracle
mongo	mysql/sqlplus

Figure 2.5.13-25 RDBMS Comparison to MongoDB

2.5.13.7.4.2
(09-27-2022)

Types of NoSQL Database Management Systems (DBMS)

- (1) MongoDB has four types of NoSQL DBMS as follows:
- Key-value paired databases:** Data is stored as key/value pair hash table where each key is unique, and is designed with the capability of handling heavy data loads. Key value stores help the developer to store schema-less data. See example figure below:

Key-value Database Example

Key	Value
Agency	Department of Education
State	California
County	Orange Grove
Occupation	Educator
Age	36

Figure 2.5.13-26 Key Value Pair Based Database Example

- b. **Column-Oriented Databases:** Allows you to store the columns of data instead of rows, and are normally used to manage data warehouses, business intelligence, and CRM.

Column	Family		
Row	Column Name		
Key	Key	Key	Key
	Value	Value	Value
	Column	Name	
	Key	Key	Key
	Value	Value	Value

Figure 2.5.13-27 Column-oriented NoSQL Database Example

- c. **Document-Oriented databases:** see Table IRM 2.5.13.7.4
- d. **Graph-Oriented Database:** A type of database used to represent data in the form of a graph with three components as the following:
- **Nodes (or vertices):** This is defined as the nouns in the database, they store data about the people, place and things.
 - **Edges (or relationships):** Defined as the verbs in the database, they stored data about the actions that are done between nodes.
 - **Properties:** Used to model the data
 - **Labels:** Used to tag a group of related notes

2.5.13.7.4.3
(09-27-2022)
MongoDB NoSQL Best Practices

- (1) Since MongoDB is a high performance NoSQL database it offers developers working on high-performance applications horizontal scaling and load balancing which includes a great balance of custom-tailoring and scalability.
- (2) When designing your MongoDB schema ensure it is designed to work well with your application. Two different applications using the same data could have dissimilar schemas. Unlike RDBMS schema design MongoDB does not have a formal process, algorithms or rules.
- (3) To achieve high-performance with a NoSQL database consider:

- a. Storage of data
 - b. **Good query performance:** Create indexes to improve query performance; however, be cautious of too many indexes that are not being used because it could result in slower performance.
 - Best Strategy for Designing Indexes: Profile a variety of index configurations with data sets similar to the ones you'll be running in production to see which configurations perform best. For more indexing information see <https://www.mongodb.com/docs/manual/applications/indexes>.
 - c. Use a reasonable amount of RAM : As applicable, select a platform that has more RAM than your working data set size. The working set is the set of data and indexes accessed during normal operations. Being able to keep the working set in memory is very important for cluster performance.
 - d. Consider using Sharding to increase the amount of available RAM in a cluster.
- (4) **Know when to scale up:** If your instance shows a load over 60%, consider scaling up by sharding. Your load should be consistently below this threshold during normal operations. If above 60% this could impact recover and vertical scaling scenarios.
 - (5) Ensure you use compression.
 - (6) Ensure you store all data for a record in a single document.
 - (7) Eliminate all unnecessary indexes.
 - (8) Avoid large documents.
 - (9) Avoid negation in queries.
 - (10) Avoid unnecessarily long field names.
 - (11) Run a single MongoDB per server.
 - (12) Run explain() for every complex query.
 - (13) Use Solid State Drives (SSDs) for write heavy applications.
 - (14) Use bulk inserts when needed.
 - (15) **Always use Replica Sets:** Replica Sets provides an automatic failover. Replica Sets in MongoDB is a group of mongod processes that maintain the same data set. Replication provides high availability of your data if the primary node fails in the cluster the secondary node will remain functional.
 - (16) Update only modified fields.
 - (17) Have at least one secondary and one arbiter.
 - (18) Set write concern to "2" when the data is critical.

2.5.13.7.4.3.1
(09-27-2022)

MongoDB Security Best Practices

- (1) MongoDB NoSQL's security architecture must include the following:
 - a. **User access management:** Restrict access to sensitive data such as Personal Identifiable Information (PII).
 - b. **Auditing:** Provide an analysis of all database actions and events.

- c. **Data Protection:** Ensure data is encrypted in-motion over the network, in-use in the database, and at-rest in persistent storage.
 - d. **Environmental Protection:** The host operating system, network, and database infrastructure must be secure.
 - (2) Database security is a major concern because of the increase in database hacking incidents. Failure of developers to follow standard security practices could result with a negative impact to the organization. See practices as follows:
 - a. **Encrypt the data where it is stored (Data at Rest):** This feature is available with MongoDB Enterprise and MongoDB Atlas.
 - b. **Ensure all moving data: (Data in Transit):** is encrypted appropriately.
 - c. **Ensure audit trails are enabled:** The purpose is to track any changes to the configuration of the database.
 - d. **Avoid using default ports:** (e.g., 27017 and 27018) because hackers will attempt to access the database through these default ports first.
 - e. **Avoid improper user credential storage:** Do not store passwords in plaintext, see IRM 10.8.21 Security, Privacy and Assurance, Information Technology Security, Database Security Policy and IRM 10.8.11 , Privacy and Information Protection, Information Technology (IT) Security, Application Security Policy.
 - (3) Encrypt all traffic with **Transport Layer Security (TLS):** A secure connection is very crucial for ensuring that only trusted connections make it to the database, and for communication between mongod, mongos, applications, and MongoDB.
 - (4) Create separate security Credentials: Each entity must have unique credentials to access the database.
 - (5) **Role-Based Access Control:** Recommend using predefined roles (i.e., out of the box) such as dbAdmin, dbOwner, clusterAdmin and customize them to meet the functional needs aligned with organizational policies.
 - a. Recommend creating separate login credentials for each application or service that will access the database for the purpose of recorded in audit trails
 - (6) **Limit connections to the database:** Allow connections only from a specific IP address (i.e., whitelisting). This will mitigate any intruders gaining access to the database and reduce risks.
 - (7) For more information on recommended database security see subsection IRM 2.5.13.8.1.
-
- (1) Use MongoDB Atlas for robust encryption features. Encryption is the process that transforms plaintext data into output known as ciphertext.
 - (2) MongoDB authentication is the process of validating the identity of a client.(e.g., administrators, users, applications connecting to the database, or MongoDB utilities).
 - (3) When authentication is enabled the database allows permissions for the clients and servers to connect, restrict user actions based on roles, and the MongoDB instance enables tracking and auditing of systems events.

2.5.13.7.4.3.1.1
(09-27-2022)

MongoDB Authentication and Authorization

- (4) Use Salted Challenge Response Authentication Mechanism (SCRAM) which is the default authentication mechanism for MongoDB when a user authenticates themselves. SCRAM verifies the user credentials against the user's name, password and authentication database.
- (5) Allow only trusted clients to access the network interfaces and ports where the MongoDB instances reside.
- (6) Disable direct Secure Shell (SSH) root access.
- (7) Ensure you enable authentication and authorization as the following:

Enabling MongoDB authentication and authorization

- | |
|---|
| <ol style="list-style-type: none"> 1. Create an administration account 2. Start MongoDB without authentication initially 3. Connect to server using the mongo shell from the server itself and write command as shown below: |
| <pre style="margin: 0;">mongo mongodb://localhost:27017</pre> <p>Note: Change the default port if you are using another port.</p> |
| <ol style="list-style-type: none"> 4. Create an administrator in the admin database as a database |

Figure 2.5.13-28 Description of how to enable MongoDB Authentication and Authorization

- (8) Create login credentials for each entity because it is easier to define, manage, track system access. If your credential are compromised it is easier to revoke without interrupting access by other entities (e.g., human users, service accounts and internal communication).
- (9) Encrypt and protect all data using file-system permissions.
- (10) For more information MongoDB security see, <https://www.mongodb.com/docs/manual/core/security-client-side-encryption>.

2.5.13.7.5
(09-27-2022)

**Big Data Models and No
Structured Query
Language (NoSQL)
Databases Overview**

- (1) Relational databases that store data in a fixed, tabular format is only enough for small to intermediate-scale database applications.
- (2) Big data normally refers to online transaction processing of high volumes of data or digital information such as: Terabytes (TB) = 1,000 GB, Petabyte (PB) = 1,000 TB, Exabyte (EB) = 1,000 PB, Zettabyte (ZB) = 1,000 EB, Yottabyte (YB) = 1,000 ZB, and of low-density unstructured data. Since Big Data implies enormous storage it would be costly, and impractical when using relational databases. Hence, NoSQL (non-relational databases) like "MongoDB" which is document-oriented is currently used by IRS Information Technology systems for big-data analytics and data warehousing.
- (3) NoSQL databases does not use tables, nor the links between tables in order to store and organize information. This database is critical for large volumes of quickly changing data applications and real-time web applications because of its ability to manage agility challenges, and provide highly resilient scalability on the fly. There are four major types of NoSQL databases:
 - a. **Column Database:** Data is stored in a columnar format.

- b. **Graph Database:** This database is arranged in the form of a graph with the elements connected using the relations between them.
- c. **Key-value database:** This database is organized as key value pairs that only appear once, e.g., Couchbase and ArangoDB.
- d. **Document Database:** Document databases store data in documents comparable to JavaScript Object Notation (JSON) objects, but can also use XML, text, or Binary Large Object (BLOB).

(4) The benefits of using NoSQL databases are:

- a. Flexible for developers to manipulate, and map to objects in their code
- b. The schema dynamically adapts to change, and is self-designing — it does not need to be pre-defined in the database
- c. Document databases like MongoDB use JSON format so rules pertaining to document structure can be imposed to store data

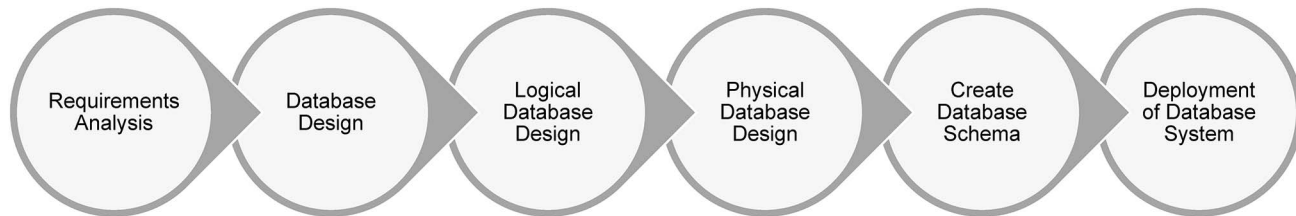


Figure 2.5.13-29

2.5.13.7.6
(09-27-2022)

Oracle Database Design Overview

- (1) Oracle database (Oracle DB) is a relational database management system (RDBMS) from the Oracle Corporation which was originally developed during 1977. This system has a relation database framework where data objects may be directly accessed by users, or application front end via SQL. Oracle is a fully scalable relational database architecture, and is used by global enterprises.
- (2) The key to database and application performance is design, not tuning.
- (3) Oracle SOAP (Simple Object Access Protocol) interfaces support (Atomicity, Consistency, Isolation, and Durability) ACID compliant database transactions which guarantee transactions are processed reliably.

#

2.5.13.7.6.1
(09-27-2022)

Oracle - Design for Performance Best Practices

- (1) **Design for Performance:** When the design is poor the data model is not efficient. A great start to an efficient design is well-defined performance goals and metrics, and a good benchmarking strategy; therefore, design goals for developers must be the following steps:
 - a. **Design quality Data Models for Optimal Performance:** This is very important — perform frequent queries by doing the following:
 - Analyze the data requirements of the application
 - Implement the database design for the application
 - Maintain the database and database application by monitoring the system

- Schedule the maintenance periods, and inform users when it will begin
- Fix any bugs, apply patches and release upgrades
- b. **Tools for Performance:** Use performance tools for reporting runtime information about the application as the following:
 - **DBMS_APPLICATION_INFO Package:** Use this package with the SQL Trace facility and Oracle Trace, and to record the names of executing module, or transactions in the database.
 - **SQL Trace Facility (SQL_TRACE):** For best results, use this tool with “TKPROF” formats the trace file contents in a readable file. The EXPLAIN PLAN statement show the execution plans chosen by the optimizer, and the execution plan for the specified SQL statement if it were executed in the current state.
 - **EXPLAIN PLAN Statement:** This statement stores the execution plan for the statement in a plan table after a SQL statement is run, and the optimizer generates several execution plans. The plan table contains optimization information such as the cost, and cardinality of each operation, accessed partitions, and distribution method of join inputs.
- c. **Testing for Performance:** Use the following guidelines for testing an application performance:
 - **Automatic Database Diagnostic Monitor (ADDM):** ADDM is used for design validation, and determines where database performance problems might exist, and recommends corrections.
 - **SQL Tuning Advisor:** SQL Tuning Advisor which is also used for design validation, e.g., if ADDM finds high-load SQL statements SQL Tuning Advisor can analyze the statements, and provide tuning recommendations.
 - **Test a Single User Performance First:** Start testing in a single user mode first; therefore, if an acceptable performance is not achieved with low loads, then multiple user cannot obtain acceptable performance.
- d. **Design for Scalability:** Use appropriate application development techniques, i.e., bind variables, and Oracle Database architecture features like shared server connections, clustering, partitioning, and parallel operations.

2.5.13.7.6.2
(09-27-2022)
**Relational Database
Design Rules and SQL
Coding Standards**

- (1) This section is for enterprise database design, regulation of linked databases, e.g., COMMON databases, MASTER databases, and Transaction databases. See Exhibit 2.5.13-13

2.5.13.7.7
(09-27-2022)
PostgreSQL Overview

- (1) PostgreSQL is authorized for IRS use, and is an open source enterprise - class relational database based on POSTGRES, version 4.2. It was created at the University of California, at Berkeley Computer Science Department. PostgreSQL is the descendant of this Berkeley code, and offers features as follows:
- Complex queries
 - Foreign keys
 - Triggers
 - Updatable views
 - Transactional integrity
 - Multi-version concurrency control
 - Streaming Replication as of version 9.0

- Hot Standby as of version 9.0
- (2) PostgreSQL is cross-platform it runs on Windows, Linux, FreeBSD, and UNIX Solaris operating systems, and includes features like Oracle and IBM DB2. PostgreSQL supports high concurrent traffic loads and full ACID compliance for transactions. PostgreSQL 11.1 was released in November 2018. It is a highly scalable database that supports both SQL and JSON querying.
 - (3) PostgreSQL includes many features to help developers write stored procedures and functions in various programming interfaces (e.g., Procedural Language (PL)/Perl, PL/Python, PL/Ruby, and PL/R, Tcl and Open Database Connectivity (ODBC)).
 - (4) The primary differences between PostgreSQL and MongoDB are as follows:

PostgreSQL and MongoDB Comparison

Comparison between MongoDB	PostgreSQL and
PostgreSQL	MongoDB
Best used for transactional applications - normalized type, joins, data constraints and transactional support.	Best use is with Big data with high volume and velocity wherever data consistency and integrity are not required.
For business logic PostgreSQL is centralized with triggers procedures.	For business logic MongoDB is distributed across applications.
In MongoDB there are various drivers available for using such as C, C++, Python, Java	PostgreSQL supports languages like C, C++ Python and Java.
In MongoDB, you can insert documents with varying schema. All documents should not have to adhere to a fixed schema.	In PostgreSQL, you can not insert records that have a varying schema.
MongoDB does not support joins	PostgreSQL support joins
It is the best-suited database for IoT and real-time analytics	Use PostgreSQL if you need a transactional and ACID-compliant database.
Data Types: Boolean, Character, Numeric, Temporal, UUID, Array, JSON, key-value pairs, and special types such as network address and geospatial data. BSON maximum document size - 1.6 TB	Data Types: String, Numeric, Boolean, Min/Max keys, Arrays, Timestamps, Object, Null, Symbol, Date, Object ID, Binary, Code, and Regular Expression. BSON maximum document size - 16 MB
Scalability: Scaling is built-in, you can have many nodes as required in a sharded cluster	Scalability: An extension is required to add the capability. No limit on database size
Consistency and Availability: Similar setup as MongoDB with a single master, and passive nodes can be configured for reading.	Consistency and Availability: Has a single master in a replica set that can accept reads and writes, and the secondaries can be configured for reading.

Figure 2.5.13-30 Depiction of Comparison Table for PostgreSQL and MongoDB

2.5.13.7.8

(09-27-2022)

PostgreSQL Data Types

(1) When creating tables a data type is necessary to store in table fields. Proper use of the data types provides efficient storage of data. Users can also create their own custom data types by using the SQL command “CREATE TYPE SQL”. The different PostgreSQL categories of data types are as follows:

- a. **Numeric types:** Consist of two-byte, four-byte, and eight-byte integers, four-byte and eight-byte floating-point numbers, and specifiable-precision decimals as seen in Figure 2.5.13-31 .

Numeric Types

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	9223372036854775808 to 9223372036854775807
decimal	variable	user-specified precision, exact	Up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	Up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Figure 2.5.13-31 Depiction of Table for PostgreSQL Numeric Data Type

- b. **Monetary Types:** Do not use Floating point numbers due to the possibility of rounding errors. This type stores a currency amount with a fixed fractional accuracy. Values of the **numeric**, **int**, and **bigint** data types can be cast to **money** as seen in Figure 2.5.13-32.

PostgreSQL Money Data Type

Name	Storage Size	Description	Range
money	8 bytes	currency amount	92233720368547758.08 to +92233720368547758.07

Figure 2.5.13-32 Depiction of Table for PostgreSQL Money Data Type

- c. **Character Types:** The general-purpose character types available in PostgreSQL as seen in the table below.

PostgreSQL Character Data Types

No.	Name & Description
1	character varying(n), varchar(n) variable-length with limit
2	character(n), char(n) fixed-length, blank padded
3	text

Figure 2.5.13-33 Depiction of Table for PostgreSQL Character Data Type

- d. **Binary Data Type:** The bytea data type allows storage of binary strings as in the table given below.

PostgreSQL Binary Data Types

Name	Storage Size	Description
bytea	1 or 4 bytes plus the actual binary string	variable-length binary string

Figure 2.5.13-34 Depiction of Table for PostgreSQL Binary Data Types

- e. PostgreSQL supports all SQL date and time types. Dates are counted according to the Gregorian calendar. All the types have resolution of 1 microsecond / 14 digits except date type, whose resolution is day as seen in the table below.

PostgreSQL Date/Time Types

Name	Storage Size	Description	Low Value	High Value
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD
TIMESTAMPTZ	8 bytes	both date and time, with time zone	4713 BC	294276 AD
date	4 bytes	date (no time of day)	4713 BC	5874897 AD
time [(p)] [without time zone]	8 bytes	time of day (no date)	00:00:00	24:00:00
time [(p)] [with time zone]	12 bytes	times of day only, with time zone	00:00:00+1459	24:00:00-1459
interval [fields] [(p)]	12 bytes	time interval	-178000000 years	178000000 years

Figure 2.5.13-35 Depiction of Table for PostgreSQL Date/Time Data Types

- f. **Boolean Type:** PostgreSQL provides the standard SQL type Boolean. The Boolean data type can have the states **true**, **false**, and a third state, **unknown**, which is represented by the SQL null value.

- g. **Enumerated Type:** Enumerated (enum) types are data types that consist of a static, ordered set of values and are equivalent to the enum types supported in a number of programming languages. Unlike other types, Enumerated Types must be created using the “CREATE TYPE” command. This type is used to store a static, ordered set of values. After Enumerated is created, it can be used like any other types. (e.g., days of the week) as shown below.

PostgreSQL Enumerated Type

Example of Enumerated Type
CREATE TYPE week AS ENUM ('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun');

- h. **Geometric Data Types:** Represents two-dimensional spatial objects. The most profound type, the point, forms the basis for all of the other types.

PostgreSQL Geometric Type

Name	Storage Size	Representation	Description
point	16 bytes	Point on a plane	(x,y)
point	32 bytes	Infinite line (not fully implemented)	((x1,y1),(x2,y2))
lseg	32 bytes	Finite line segment	((x1,y1),(x2,y2))
box	32 bytes	Rectangular box	((x1,y1),(x2,y2))
path	16+16n bytes	Closed path (similar to polygon)	((x1,y1),...)
path	16+16n bytes	Open Path	[(x1,y1),...]
polygon	40+16n	Polygon (similar to closed path)	((x1,y1),...)
circle	24 bytes	Circle	<(x,y),r> (center point and radius)

Figure 2.5.13-36 Depiction of Table for PostgreSQL Geometric Data Type

- (2) For more information on PostgreSQL see a PostgreSQL tutorial link <https://www.postgresqltutorial.com/>.

2.5.13.7.9
(09-27-2022)
PostgreSQL Best Practices

- (1) For the best database performance developers must practice good configuration and maintenance of PostgreSQL as follows:
- Use environment variables to expose connection strings (e.g., DATABASE_URL).
 - Do not store database credentials in the codebase, the code is considered.
 - End all queries with a semi-colon (;).

Example of PostgreSQL Query Semi-colon Use

Example of Using Semi-colon with Queries
SELECT employee column FROM q_table;

- d. Avoid using "SELECT * " whenever possible. Always stipulate only the columns needed because this will reduce the query execution time.
- e. Do not add changes directly to your Production database. Only apply changes to the Production environment after all testing has been completed satisfactorily.
- f. Do not use CamelCase for object names (schema, table, column, etc.). PostgreSQL will convert all entries to lowercase by default unless quoted.
- g. Use "CONSTRAINT" although the following form is correct "column TYPE PRIMARY KEY," "CONSTRAINT" will eliminate PostgreSQL from assigning a default name. See example

Example of Using PostgreSQL CONSTRAINT

Using CONSTRAINT
CONSTRAINT accounts_pk PRIMARY KEY (account_id)

Figure 2.5.13-37 Example of Using PostgreSQL CONSTRAINT

- h. Do not use "id " as the primary key name in every table. Create something meaningful, see example below:

Example of Using PostgreSQL Primary key (PK) Names

For employment table, use employment_id.
For transportation table, use transportation_id

Figure 2.5.13-38 Creation of Meaningful PostgreSQL Primary Key Names

- i. Do not connect to the PostgreSQL databases directly. Recommend developers use a (connection pooler) such as PGBouncer to configure clients. This action should help reduce the memory and CPU footprint of open connections.
- j. **Maintain Data Compatibility:** Always implement foreign keys, checks, and normalization practices to store data in the database.
- k. **Avoid Public Schema:** Recommend creating a separate schemas for each entity. Reason: If you create a new database and create a table without a specific schema, PostgreSQL will automatically creates a public schema and grant access to the role named "public".
- l. **Create Audit Triggers:** Audits will help track changes made to a table, who made the changes and include the timestamp. This is a great benefit when there is a need to track activities for a specific event.
- m. **Maintain Database Schema Version Control:** All code written must be included in a version control system. Use an IRS approved process for creating versions for the database schema along with the rest of the projects. See IRM 2.150.2 Configuration Management (CM) Process for version control guidance.

2.5.13.8
(09-27-2022)
**Database Security
Design**

- (1) All organizations must work on a continual basis protecting their databases that have sensitive data by identifying and remedying security vulnerabilities and exploits. In addition to doing monitoring and security assessments, ensure results are analyzed and properly audited for demonstrating compliance with federal security regulations: IRM 10.8.21 Information Technology (IT) Security, Database Security Policy, OWASP, and NIST 800-53A industry standards.

2.5.13.8.1

(09-27-2022)

**Database Design
Security Best Practices**

- (1) This subsection is guidance for ensuring adequate security controls for all databases storing sensitive or protected data.
- (2) For databases the following Federal Information Processing Standard (FIPS) government standards must be followed:
 - a. FIPS 127-2: Identifies and describes the recommended construct sizes, see Exhibit 2.5.13-14.
 - b. FIPS 140-2: Identifies the cryptographic security requirements, and is designed to protect data at rest, and in transit over the network. FIPS have several levels ranging from (1 - lowest) to (4 - highest).

2.5.13.9

(09-27-2022)

**IRS Extensible Markup
Language (XML)
Overview**

- (1) As of January 2003, the IRS has approved XML as the recommended syntax specification in more than 20+ applications. The Enterprise Data Management Organization (EDMO) is the program lead, and collaborates daily with organizations using projects associated with this language by establishing policy and standards. The IRS XML Community of Practice (xmlCoP) was launched to facilitate, and review the necessary changes XML policies, standards and practices.
- (2) Extensible Markup Language (XML) is used to define documents with a standard format that can be read by an XML compatible application. XML can be used with HTML pages, but is not a markup language. It is a “metalan-
guage - information used to describe language” that can be used to create markup languages for particular applications.
- (3) Technical experts can use XML to create a database of information without having a real database. XML is commonly used in web applications, and other programs, e.g., Arbortext which is a XML compatible application used to create/update IRS Internal Revenue Manuals (IRMs).
- (4) The XML Industry Standards are as follows:
 - a. The World Wide Web Consortium (W3C) is family of XML standards.
 - b. Electronic Business is using eXtensible Markup Language (ebXML).
 - c. Universal Business Language (UBL): UBL is an implementation of ebXML, ebXML originally an Oasis standard is now an ISO standard (ISO 15000-5), and focuses on the design of reusable components.

2.5.13.9.1

(09-27-2022)

**IRS Extensible Markup
Language (XML) Naming
and Design Rules**

- (1) EDMO is responsible for the IRS XML naming and design rules, and the relevant standardized XML components or products after use. The naming and design rules for constructing and naming XML components or products pertain to:
 - a. Attributes
 - b. Data Types
 - c. Elements
 - d. Namespaces
 - e. Schema
- (2) For all definitions, see Exhibit 2.5.13-11
- (3) The main objective of the XML naming, and design rules is to maintain uniformity, and interoperability across the IRS, and its data exchange partners. This

naming and design rules established by EDMO apply to all XML data exchange between services, reusable system components, and system to system interfaces.

#

Exhibit 2.5.13-1 (09-27-2022)

Guidelines for Decision Analysis and Description Forms

The following is an example of a Decision Analysis and Description Form. This type of documentation should be used to describe having a significant impact on the functions and tasks being analyzed.

Sample Decision Analysis and Description Form
Decisions:
Person(s) Responsible:
Nature:
Information Requirements:
Source(s) of Data/Information:
Frequency of Input:
Relationship to Other Projects:
Identify the decision, which needs to be made, and the person(s) responsible for making it. Then record the nature of the decision, i.e., routine or ad hoc, and its perceived importance and impact. Next, identify the source(s) of the data/information needed to make this particular decision. It is also advisable to document how often this data/information should be provided. Finally, determine what relationship this decision has to other decisions that must be made relative to the functions and task being studied. For example, does it have prerequisites? Is it a prerequisite to other decisions?
Note: Enter "N/A" in those cases where an entry does not apply.

Exhibit 2.5.13-2 (09-27-2022)**Guidelines for Task Analysis and Description Forms**

The following example of a Task Analysis and Description Form is presented merely as a guide. This type of documentation should be used to describe any task, or major subtask, having a significant impact on the environment being analyzed.

Sample Task Analysis and Description Form
<ol style="list-style-type: none"> 1. Task/Subtask Number: 2. Task/Subtask Name: 3. Authorizations: 4. Purpose: 5. Performed By: 6. Description: 7. Frequency: 8. Duration: 9. Input: 10. Output: 11. Usage of Documents: 12. Operations Performed: 13. Subtasks: 14. Error Conditions: <p>When assigning numbers to a task and its major subtasks, it is recommended that their relationship be indicated with the assigned numbers. For example, a task "T-1" could have subtasks "T-1-1" and "T-1-2" or "T-1a" and "T-1b." The names assigned to tasks and subtasks should also be meaningful.</p> <p>It is recommended that the authorization(s) for a task be specified in its major subtasks if there are any, because different people may authorize different parts of a task.</p> <p>Be concise when indicating the purpose of each task/subtask. Indicate the person and/or title or area responsible for performing the task/subtask, e.g., John Smith, Distribution Coordinator.</p> <p>Clearly and concisely describe the task/subtask.</p> <p>Frequencies and durations may differ between subtasks within the same task.</p>

Input and output documents should be listed and assigned numbers. Then, when the entries are made for "Usage of documents" and "Operations Performed," these documents can be referred to by their numbers only. For example, under "Usage of Documents," sample entries would be: "All columns of D-3 and D-11; Part I of D-4 and Part II of D-12." Under "Operations Performed," a sample entry would be "New project information on D-4 is entered on D-1; D-3 is used to update D-7 and generate D-9 daily." If there are cases where there is no hard copy input and/or output, reference appropriate video screens, etc.

Briefly, describe the operations performed to accomplish the task/subtask.

List any subtasks by name and number and cite any error conditions or anomalies.

Note: Enter "N/A" in those cases where an entry does not apply.

Exhibit 2.5.13-3 (09-27-2022)
Sample Task/Data Element Matrix

This document specifies a task's inputs and outputs in terms of data elements. It is recommended that both name and number refer to all tasks.

Tasks	T-10 Distribution			
Data Elements				
Proj-Name	I			
Proj-Main-Loc-No	I			
Proj-Main-Loc-Addr	I			
Ord-No	I/O			
Date-Posted	I/O			
Proj-No	I/O			
Item-Line-No	O			

Exhibit 2.5.13-4 (09-27-2022)

Guidelines for Constructing Data Structure Diagrams (DSD)

#

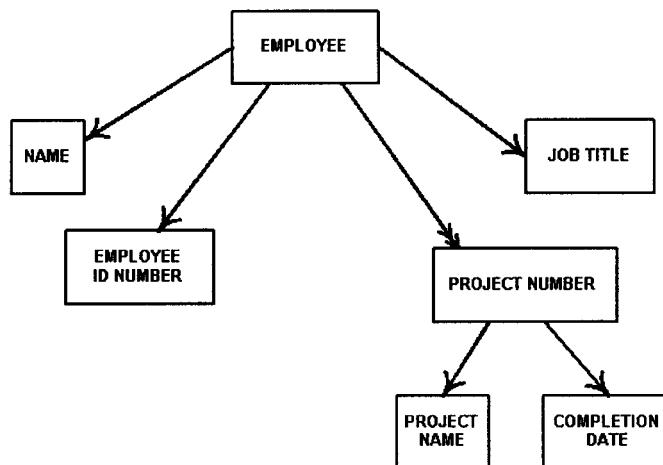
DEPARTMENT

EMPLOYEE

Hierarchical Network Structure

Hierarchical Network Diagram

Data structure diagrams can represent hierarchical structure, network structure, or a combination of these structures. The following diagram depicts a hierarchical structure.

Exhibit 2.5.13-4 (Cont. 1) (09-27-2022)**Guidelines for Constructing Data Structure Diagrams (DSD)*****Explanation of Hierarchical Structure Above***

	Explanation of Hierarchical Structure 1
1.	This depicts that an employee works several projects, and that each project must have only one name and one completion date.
2.	Hierarchical structures are distinguished from other relationships because every node must have only one parent node, except the root, which has no parent.
3.	The descendants of a node are called children

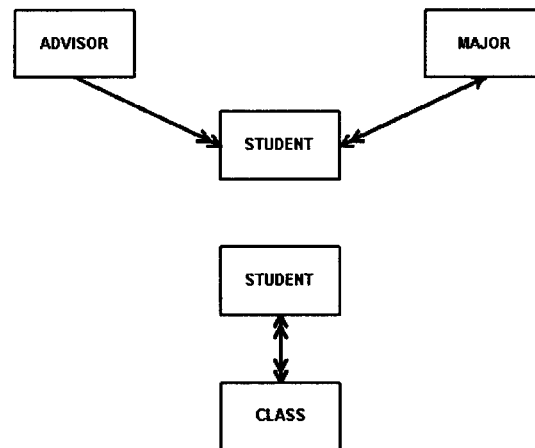
Exhibit 2.5.13-5 (09-27-2022)**Relationship between Two Entities-Classes**

	Two Entity Classes
1.	These symbols do not indicate how many individual entities constitute each entity class. Relationships depict cardinality, e.g., (One to One), (One to Many), (Many to Many), etc. The following depicts a relationship between entity classes.
2.	This relationship indicates that each Department comprises (One to Many) Employee



Exhibit 2.5.13-6 (09-27-2022)**Explanation of Hierarchical Structure (One-to-Many) and (Many-to-Many) Relationships**

Explanation of Hierarchical Structure 2	
1.	This network structure allows a logical record, entity, or entity class to have more than one parent.
2.	<ul style="list-style-type: none">• The following examples show a one-to-many and a many-to-many relationship:<ul style="list-style-type: none">✓ The first example shows that a student has only one advisor and one major, but an advisor counsels many students and many students choose a particular academic major.✓ The second example shows that a student has many classes, and that a class comprises many students.

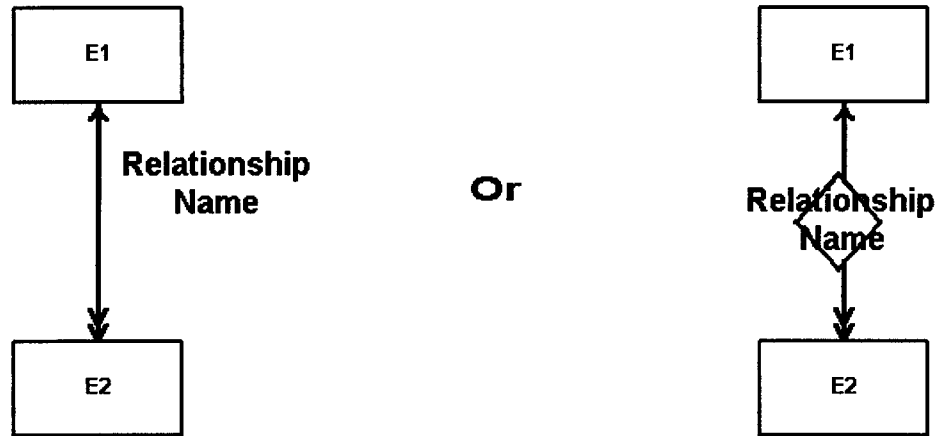


Guidelines for Constructing Entity Relationship Diagrams (ERD)

[illegible][illegible]

Exhibit 2.5.13-7 (Cont. 1) (09-27-2022)

Guidelines for Constructing Entity Relationship Diagrams (ERD)



#

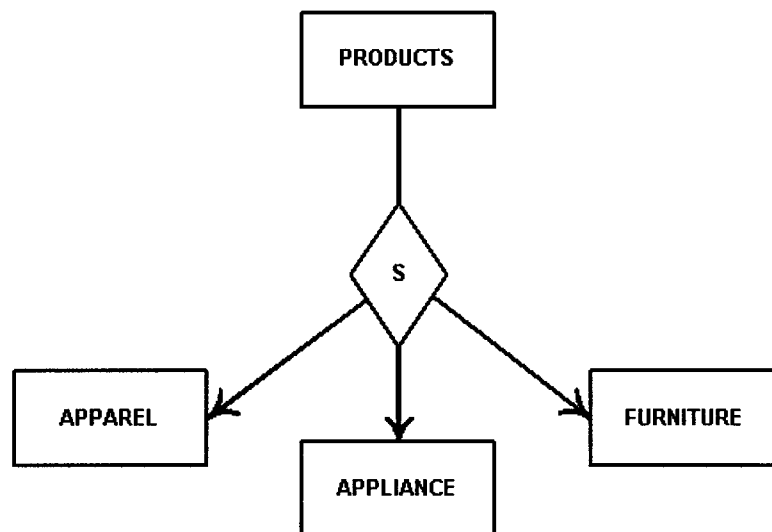
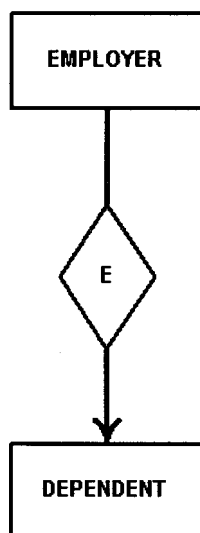


Exhibit 2.5.13-7 (Cont. 2) (09-27-2022)

Guidelines for Constructing Entity Relationship Diagrams (ERD)

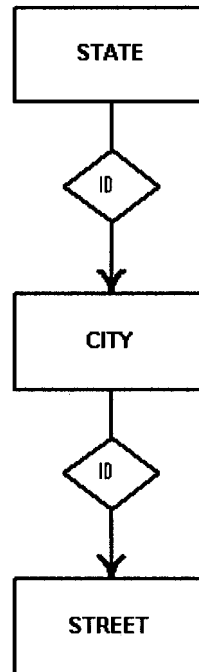
#



#

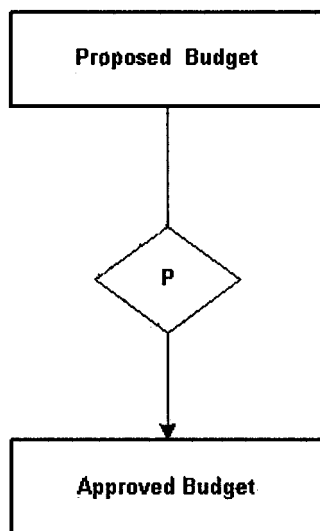
Exhibit 2.5.13-7 (Cont. 3) (09-27-2022)

Guidelines for Constructing Entity Relationship Diagrams (ERD)



#

Guidelines for Constructing Entity Relationship Diagrams (ERD)

[illegible]

~~# #~~
~~# #~~

~~# #~~

#

Exhibit 2.5.13-7 (Cont. 5) (09-27-2022)

Guidelines for Constructing Entity Relationship Diagrams (ERD)

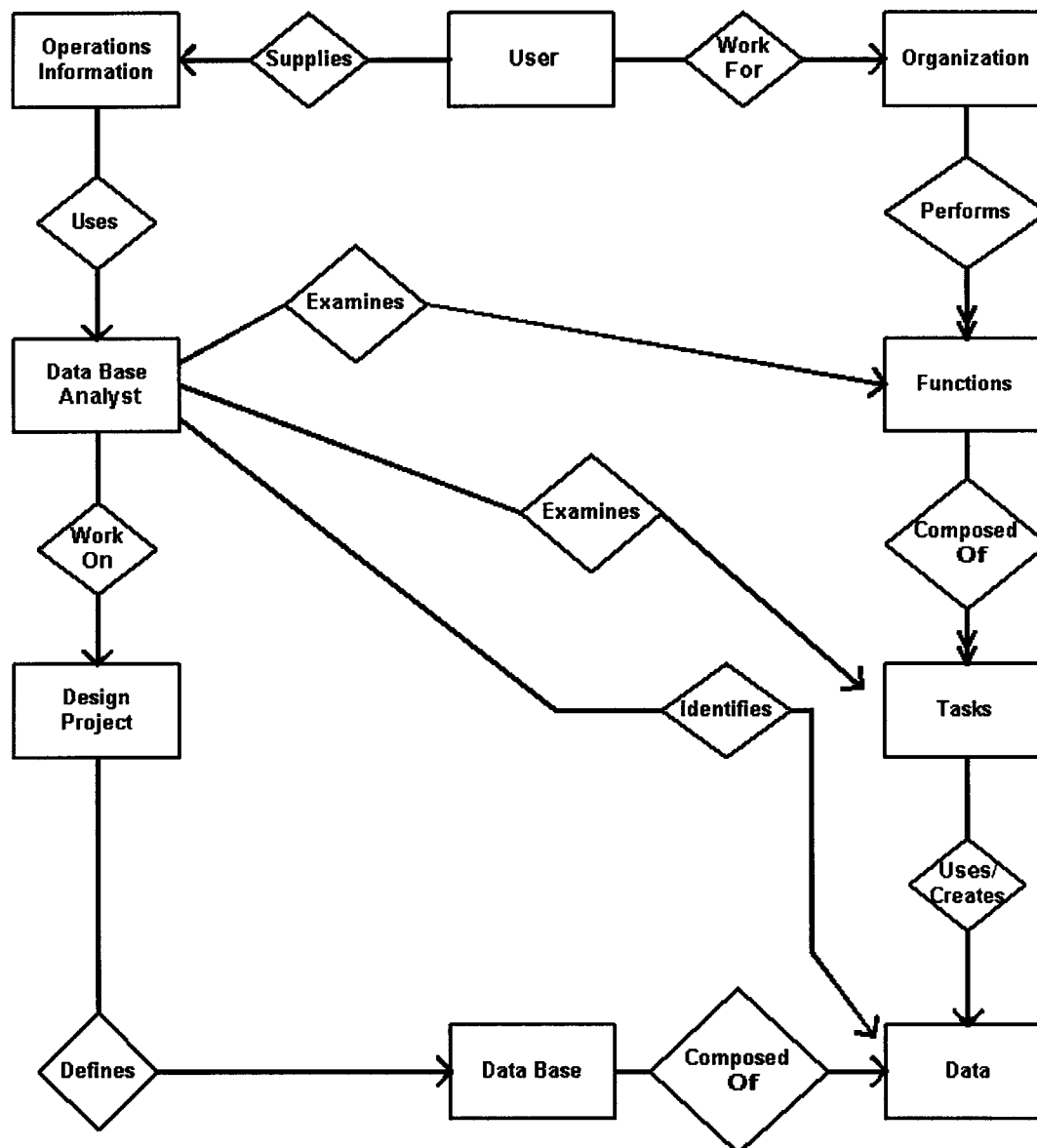


Exhibit 2.5.13-8 (09-27-2022)**Sample Data Definition List**

	Sample Data Definition List	
Name	Definition	Length
(Name of data element)	(Definition of data element)	(Length of data element)
...
...

Exhibit 2.5.13-9 (09-27-2022)
Summary of Access Methods

Access Method	Direct	Algorithm (assume even distribution)	Index Sequential	Index Non- Sequential	Chains
Usage					
Volatile Records		x		x	
Fast Access	x	x	x	x	If records contiguous
Random Processing	x	x	x	x	
Sequential Processing	Pre- Determined		x		x
Good Space Utilization	x		Depends on index size	Depends on index size	Depends on pointer sizing
Overall Efficiency	x	x	If little overflow	Small to medium files	If records contiguous

Exhibit 2.5.13-10 (09-27-2022)
Acronyms and Terms

Acronyms	Terms
ACID	Atomicity, Consistency, Isolation, Durability
BLOB	Binary Large Object
BSON	Binary and JSON
CASE	Computer Aided Software Engineering
CODASYL	Conference on Data Systems Languages
CLOB	Character Large Object
CRUD	Create, Read, Update and Delete
CSV	Comma Separated Value
CSS	Cascading Style Sheet
DBA	Database Administrator
DBMS	Database Management System
DBRM	Database Request Module
DDL	Data Description (Definition) Language
DSD	Data Structure Diagram
DSN	Data Set Name
EDD	Engineering Data Dictionary
EDSG	Enterprise Data Standards and Guidelines
ERD	Entity Relationship Diagram
FIFO	First In First Out
GUI	Globally Unique Identifier
HTML	Hypertext Markup Language
IoT	Internet of Things
JSON	Java Script Object Notation
LOB	Large Object
MIB	Management Information Base
ORDBMS	Object Relational Database Management System
PID	Process Identifier
PPDM	Project Physical Data Model
PKI	Public Key Infrastructure
RACI	Responsible Accountable Consulted and Informed

Exhibit 2.5.13-10 (Cont. 1) (09-27-2022)**Acronyms and Terms**

Acronyms	Terms
RBAC	Role Based Access Control
SAN	Subject Alternative Name
SDLC	System Development Life Cycle
SIEM	Security Information and Event Management
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSD	Solid State Disk
TLS	Transport Layer Security
TTL	Time To Live
UML	Unified Modeling Language
XML	Extensible Markup Language
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WSDL (NDR)	WSDL Naming and Design Rules

Exhibit 2.5.13-11 (09-27-2022)
Terms/Definitions

Terms	Definitions
ACID	A set of properties for database transactions that guarantee validity in the event of errors, power failures, or other adverse system issues.
Algorithm	A process or set of rules to be followed in calculations or other problem-solving operations.
Attribute	A data item that comprises a single piece of information about an entity. It usually cannot be broken down into parts that have meanings of their own. Attributes can be divided into two classes--those that identify entity instances (occurrences) and those that provide the descriptive properties of an entity.
Bachman Diagram	Another name for a data structure diagram. It is named for Charles Bachman who invented it.
Bucket	An area of storage containing one or more records referred to by the same address.
Buffer	Storage used to temporarily hold data being transferred from one device to another.
Block	Usually a combination of two or more records which are read and written jointly by one machine instruction.
Branches	The relationship between the records in a tree (hierarchical) data structure.
Business Rule	Obtained from users when gathering requirement, and are used to determine cardinality
Byte	Computers use a combinations of eight bits, called bytes, to represent one character of data or instructions. For example, the word latch has five characters, and would be represented by five bytes.
Cardinality	Expresses the minimum and maximum number of entity occurrences associated with one occurrence of a related entity.
Cascading Style Sheet	CSS is normally the standard way to define the presentation of HTML pages. CSS controls the layout of the page features such as the color, font, size, and the complete layout and is considered more efficient than HTML.
Chain	A list of data items strung together by means of a pointer.
Entity Integrity	Requires that every table have a primary key; neither the primary key, nor any part of it, can contain null values.
CODASYL	A network data model developed by the Conference on Data System Languages, Database Task Group.
Compaction	Reduces the number of bits In data without effecting the informational content.

Exhibit 2.5.13-11 (Cont. 1) (09-27-2022)**Terms/Definitions**

Terms	Definitions
Conceptual Design	Description as part of the design process of how a new product will work and meet its performance requirements.
Conceptual Model	The overall logical structure of a database (often referred to as conceptual schema) which is independent of any software or data storage structure.
Concurrency	In data base systems, refers to the number of run units actively sharing the DBMS facilities.
Data Item	The smallest unit of data that has meaning in describing information, the smallest unit of named data.
Database Administrator (DBA)	One or more individuals, possibly aided by a staff, who manage an organization's database resource.
Database	A collection of interrelated data stored together with controlled redundancy to serve one or more applications; the data are stored so that they are independent of the programs which use them.
Database Design	The process of developing a database structure from user requirements.
Data Description (Definition) Language (DDL)	A language for describing data. In some cases, software uses DDL only for logical data, or only for physical data, or both.
Database Design Methodology	A collection of techniques and tools employed within an organizational framework that can be applied consistently to successive database development projects.
Database Management System (DBMS)	The collection of software required for using a database, which presents multiple views of the data to the users and programmers.
Data Independence	The property of being able to change the overall logical or physical structure of the data without changing an application program view of that data.
Data Model	A logical representation of the data structure that forms a database. It should represent the inherent properties of the data independent of software, hardware or machine performance implications. These representations are independent of the class of software that will be used for implementation.
Database link	A database link is a pointer that defines a one-way communication path from an Oracle database server to another database server. A database link connection allows local users to access data on a remote database.
Deliverable	A deliverable is a tangible outcome that is produced during the execution of a project.

Exhibit 2.5.13-11 (Cont. 2) (09-27-2022)**Terms/Definitions**

Terms	Definitions
Dimension	A dimension is a schema object that defines hierarchical relationships between pairs of columns or column sets. A hierarchical relationship is a functional dependency from one level of a hierarchy to the next level in the hierarchy. A dimension is a container of logical relationships between columns and does not have any data storage assigned to it.
Encryption	The process of encoding/decoding when transferring data to and from the data base.
Entity	A person, place, thing or concept that interests an organization. An entity is something about which data is stored. An entity has various attributes which can be recorded, e.g., COLOR, SIZE, etc.
Gigabyte (GB)	This is 1024 megabytes also named a Gig.
Hierarchical	A tree structure where some records are subordinate to others.
Identifying Relationship	The primary key contains the foreign key; indicated in an ERD by the solid line.
Implementation Design	A database design activity that involves transforming and refining a conceptual schema into a schema that can be implemented through a database management system.
Interface	The interconnections that allow a device, a program, or a person to interact. Hardware interfaces are the cables that connect the device to its power source and to other devices. Software interfaces allow the program to communicate with other programs (such as the operating system), and user interfaces allow the user to communicate with the program (e.g., via mouse, menu commands, icons, voice commands, etc.).
Key	The data item which is used to address or identify a record.
Namespace	The canonical name for a collection or index in MongoDB. The namespace is a combination of the database name and the name of the collection or index, like so: [database-name].[collection-or-index-name]. All documents belong to a namespace
Logical Database Design	A description of the structure of logical relationships among the data elements of the system being designed.
Network	A structure in which a detail record can have more than one master record.
Orphan Record	A record whose foreign key value is not found in the corresponding entity (the entity where the primary key is located)
Overflow	An area of placement assigned to a record which for some reason cannot be stored in its home address (i.e. logically assigned address).

Exhibit 2.5.13-11 (Cont. 3) (09-27-2022)

Terms/Definitions

Terms	Definitions
Materialized view	A materialized view is a database object that provides indirect access to table data by storing the results of a query in a separate schema object. A materialized view contains the rows resulting from a query against one or more base tables, views, and/or other materialized views.
Package	Arrangement or collection of procedures and functions into logical groupings.
Packing Density	The number of records stored in a bucket compared to the number that could be stored.
Physical Database	A database in the form in which it is stored on the storage media, including pointers or other means of interconnecting it. Multiple logical databases may be derived from one or more physical databases.
Procedure	A particular course or mode of action, or a subfunction.
Relational	Pertaining to a database in normalized, two-dimensional flat form. The DBMS recombines data elements giving different relations or greater flexibility.
Real Time	Application or processing in which response to input is fast enough to affect subsequent Input (i.e. terminal dialogues on interactive systems).
Response Time	Total time between an instruction being given to access particular data, and that data being available (seek time + read or write time).
Responsible, Accountable, Consulted and Informed	The RACI matrix defines the roles and responsibilities for any activity or group of activities and assigns ownership for their tasks and decisions.
Schema	A map of the overall logical structure of the database covering all data item and record types. The global logical view of the data.
Servicewide Data Dictionary	The authoritative source for the Service's standard data names, definitions and codes.
Sets	In CODASYL, refers to a collection of record types e.g., an owner type if defined with one of its member types.
Sequence	Sequence numbers are Oracle integers of up to 38 digits defined in the database. The sequence generator provides a unique sequential series of numbers.
Shard	A single mongod instance or replica set that stores some portion of a sharded cluster's total data set. In production, all shards should be replica sets.
Sharded cluster	The set of nodes comprising a sharded MongoDB deployment. A sharded cluster consists of config servers, shards, and one or more mongos routing processes.

Exhibit 2.5.13-11 (Cont. 4) (09-27-2022)**Terms/Definitions**

Terms	Definitions
Single-master replication	A replication topology where only a single database instance accepts writes. Single-master replication ensures consistency and is the replication topology employed by MongoDB.
Simulation	To represent the functioning of one system by another.
Simple Object Access Protocol	A messaging protocol for exchanging structured information via web services in a computer network.
Storage engine	The part of a database that is responsible for managing how data is stored and accessed, both in memory and on disk. Different storage engines perform better for specific workloads.
Subject Alternative Name	Subject Alternative Name (SAN) is an extension of the X.509 certificate which allows an array of values such as IP addresses and domain names that specify which resources a single security certificate may secure.
Sub-schema	A map of a programmer's view of data used. It is derived from the schema.
Task	The lowest level of work that requires, on a repetitive basis, a unique set of data. A unique unit of work consisting of a set of sequential steps all directed toward a common goal and all using and/or creating a common set of data.
Transfer Time	Time taken to move data between a direct access device and the central processor.
Volatile File	A file with high rate of additions and deletions.
Tree Structure	A hierarchy of groups of data such as: 1) The highest level in the hierarchy has only one group called a "root". 2) All groups except the root are related to one, and only one group on a higher level than themselves.
World Wide Web Consortium	The primary international standards organization for the World Wide Web which was founded in 1994.

Exhibit 2.5.13-12 (09-27-2022)**IRS Enterprise Life Cycle (ELC) Data Model Compliance Standards**

					# # #
					# # #
					# # # # # #
					# # # # # #
					# # # # # #
					# # # # # #
					# # # # # #
					# # # # # #
					# # # # # #

IRS Enterprise Life Cycle (ELC) Data Model Compliance Standards

[illegible]

Exhibit 2.5.13-12 (Cont. 2) (09-27-2022)

IRS Enterprise Life Cycle (ELC) Data Model Compliance Standards

					# # #
					# # # # # #
					# # # # # # #

Relational Database Design Constancy Rules - Types of Data Storing in Table

[illegible][illegible]

Exhibit 2.5.13-14 (09-27-2022)

FIPS 127-2 Database Construct Sizing

	Sizing for Database Constructs	FIPS	Example of Oracle Compliance	Notes
1.	Length of an identifier (in bytes)	18	30	Note: 1. - The numbers of SET clauses in an UPDATE.
2.	Length of CHARACTER datatype (in bytes)	240	2000	Note: 2. - The FIPS PUB defines the length of a collection of columns to be the sum of: twice the number of columns, the length of each character column in bytes, decimal precision plus 1 of each exact numeric column, binary precision divided by 4 plus 1 of each approximate numeric column.
3.	Decimal precision of NUMERIC datatype	15	38	Note: 3 - The Oracle limit for the maximum row length is based on the maximum length of a row containing a LONG value of length 2 gigabytes and 999 VARCHAR2 values, each of length 4000 bytes: $2(254) + 231 + (999(4000))$.
4.	Decimal precision of DECIMAL datatype	15	38	Note: 4 - The Oracle limit for a UNIQUE key is half the size of an Oracle data block (specified by the initialization parameter DB_BLOCK_SIZE) minus some overhead.

Exhibit 2.5.13-14 (Cont. 1) (09-27-2022)
FIPS 127-2 Database Construct Sizing

	Sizing for Database Constructs	FIPS	Example of Oracle Compliance	Notes
5.	Decimal precision of SMALLINT datatype	4	38	Note: 5 - Oracle places no limit on the number of columns in a GROUP BY clause or the number of sort specifications in an ORDER BY clause. However, the sum of the sizes of all the expressions in either a GROUP BY clause, or an ORDER BY clause is limited to the size of an Oracle data block (specified by the initialization parameter DB_BLOCK_SIZE) minus some overhead.
6.	Binary precision of FLOAT datatype	20	126	Note: 6 - The Oracle limit for the number of cursors simultaneously opened is specified by the initialization parameter OPEN_CURSORS. The maximum value of this parameter depends on the memory available on your operating system, and exceeds 100 in all cases.
7.	Binary precision of REAL datatype	20	63	Left blank intentionally
8.	Binary precision of DOUBLE PRECISION datatype	30	126	Left blank intentionally
9.	Columns in a table	100	1000	Left blank intentionally
10.	Values in an INSERT statement	100	1000	Left blank intentionally
11.	SET clauses in an UPDATE statement (see Note 1)	20	1000	Left blank intentionally
13.	Length of a row (see Note 2 and Note 3)	2,000	2,000,000	Left blank intentionally
14.	Columns in a UNIQUE constraint	6	32	Left blank intentionally

Exhibit 2.5.13-14 (Cont. 2) (09-27-2022)
FIPS 127-2 Database Construct Sizing

	Sizing for Database Constructs	FIPS	Example of Oracle Compliance	Notes
15	Length of a UNIQUE constraint (see Note 2)	120	(see Note 4)	Left blank intentionally
16	Length of a foreign key column list (see Note 2)	120	(see Note 4)	Left blank intentionally
17.	Columns in a GROUP BY clause	6	255 (see Note 5)	Left blank intentionally
18.	Length of GROUP BY column list	120	(see Note 5)	Left blank intentionally
19.	Columns in a referential integrity constraint	6	32	Left blank intentionally
20.	Tables referenced in a SQL statement	15	No limit	Left blank intentionally
21.	Cursors simultaneously open	10	(see Note 6)	Left blank intentionally
22.	Items in a select list	100	1000	Left blank intentionally

