CSC 436 - Semester DB Project

Phase I Revision Document

Programming Language Lookup Database

Part I: Review Feedback & Revisions

1. Feedback Summary

The following feedback was received during the Phase I peer review process:

1. **Consolidate Implementation Entities**

   "Why did you separate Implementation into three different entities instead of using one shared Implementation entity? If you later added another concept type (e.g., Control Flow Constructs), would you need to build a fourth Implementation table? Since the attributes are almost the same in all three cases, you might consider modeling a single Implementation entity."

2. **Bridge Table for Duplicates**

   "Look into Bridge table for duplicates"

3. **Data Entry Feasibility**

   "Copy and pasting everything from documentation for each language doesn't seem feasible in this timeframe"

2. Documented Revisions

Revision 1: Unified Implementation Entity

Original Design: Three separate implementation entities Function Implementation, Structure Implementation, Operator Implementation each containing all implementation details with nearly identical attributes.

Revised Design: A single unified Implementation entity that stores the shared attributes implementation_id, language_id, version, syntax, example, result, notes. This entity connects to Language by the Implemented In relationship.

Rationale:

- Eliminates redundant attribute definitions across three nearly identical tables
- Centralizes shared implementation data in one location
- Reduces code duplication in queries and application logic

Revision 2: Bridge Tables with Composite Keys

Change: Added three bridge tables Function Implementation, Data Structure Implementation, Operator Implementation that use composite primary keys to link the unified Implementation entity to each concept type.

Rationale:

- Avoids null foreign key values that would occur if all concept IDs were stored in a single Implementation table
- Composite key implementation_id + concept_id ensure unique mappings and prevent duplicate entries
- Maintains proper referential integrity without requiring application-level validation
- Each bridge table tracks its own date_added for when the mapping was created

Revision 3: Streamlined Data Collection Approach

Change: Adjusted data collection strategy to focus on curated, high-value content rather than comprehensive documentation copying.

Rationale:

- Addresses feasibility concerns about manual data entry within project timeframe
- Focus on commonly-used functions, structures, and operators across a select set of languages
- Prioritize quality examples and clear syntax documentation over quantity

Part II: Implementation of Revisions

3. Updated Entity Sets and Attributes

Primary keys are underlined. Foreign keys are marked with (FK). Composite keys are marked with (FK, PK).

Language (Strong Entity)

language_id, language_name

Function (Strong Entity)

function_id, function_name, category, description

Data Structure (Strong Entity)

structure_id, structure_name, category, description

Operator (Strong Entity)

operator_id, operator_name, symbol, category, description

Implementation (Weak Entity)

implementation_id, language_id (FK), version, syntax, example, result, notes

Function Implementation (Bridge Table)

implementation_id (FK, PK), function_id (FK, PK)

Data Structure Implementation (Bridge Table)

implementation_id (FK, PK), structure_id (FK, PK)

Operator Implementation (Bridge Table)

implementation_id (FK, PK), operator_id (FK, PK)

4. Updated Relationships and Constraints

Language - Implementation ("Implemented In")

- **Cardinality:** One-to-Many (1:N)
- **Participation:** Language has partial participation; Implementation has total participation
- **Descriptive Attribute:** version
- **Constraint:** Each Implementation must be associated with exactly one Language

Implementation - Function Implementation ("Maps To")

- **Cardinality:** One-to-Many (1:N)
- **Participation:** Implementation has partial participation; Function Implementation has total participation
- **Descriptive Attribute:** date_added
- **Constraint:** Links implementation details to specific functions

Function - Function Implementation ("Has")

- **Cardinality:** One-to-Many (1:N)
- **Participation:** Function has partial participation; Function Implementation has total participation
- **Constraint:** Each Function Implementation must reference exactly one Function

Implementation - Data Structure Implementation ("Maps To")

- **Cardinality:** One-to-Many (1:N)
- **Participation:** Implementation has partial participation; Data Structure Implementation has total participation
- **Descriptive Attribute:** date_added
- **Constraint:** Links implementation details to specific data structures

Data Structure - Data Structure Implementation ("Has")

- **Cardinality:** One-to-Many (1:N)
- **Participation:** Data Structure has partial participation; Data Structure Implementation has total participation
- **Constraint:** Each Data Structure Implementation must reference exactly one Data Structure

Implementation - Operator Implementation ("Maps To")

- **Cardinality:** One-to-Many (1:N)
- **Participation:** Implementation has partial participation; Operator Implementation has total participation
- **Descriptive Attribute:** date_added
- **Constraint:** Links implementation details to specific operators

Operator - Operator Implementation ("Has")

- **Cardinality:** One-to-Many (1:N)
- **Participation:** Operator has partial participation; Operator Implementation has total participation
- **Constraint:** Each Operator Implementation must reference exactly one Operator

5. Updated E-R Diagram

Please refer to the accompanying file: **er_diagram_revised.png**

6. Data Sources, Operations, and Queryable Questions

Data Sources

- **Official Documentation:** Python docs, MDN Web Docs (JavaScript), Java SE Documentation, cppreference.com (C/C++)
- **Community Resources:** Stack Overflow, GeeksforGeeks, W3Schools
- **Language Specifications:** ECMA-262 (JavaScript), Python Language Reference, Java Language Specification
- **Scope:** Focus on 4-6 popular languages (Python, JavaScript, Java, C++, C#, Go) with curated high-value content

Data Modification Operations

- **INSERT:** Add new languages, functions, data structures, operators, and their implementations
- **UPDATE:** Modify syntax examples, fix errors in descriptions, update version compatibility information
- **DELETE:** Remove deprecated implementations, delete entries for unsupported language versions

Types of Questions the Database Can Answer

- What is the syntax for a specific function in Python vs JavaScript?
- Which languages implement a particular data structure?
- What are all the arithmetic operators available in C++?
- Show me examples of string manipulation functions across all supported languages
- Compare the syntax for declaring arrays in Python, JavaScript, and Java
- What implementations were added to the database in the last 30 days?