CSC 436 Assignment 3 Documentation

Programming Language Lookup Database

Section 1: Data Integrity Constraints

The database enforces data accuracy through the following constraints:

Primary Key Constraints

Every table has a primary key with AUTO_INCREMENT to automatically assign unique values. The primary keys of the strong entity tables are their own ID fields, and the child implementation tables' primary and foreign keys are implementation_id.

NOT NULL Constraints

NOT NULL constraints ensure that all essential fields are not null: language_name, function_name, structure_name, operator_name, and all foreign keys. This ensures that no partial data is inserted.

UNIQUE Constraints

The language_name field has a UNIQUE constraint to ensure that there are no duplicate programming language names.

CHECK Constraints

CHECK constraints are not entirely supported by the MySQL environment on cPanel, so validation of data is done during insertion.

Foreign Key Constraints

Foreign keys establish relationships between tables. The implementation table has a foreign key to the language table via language_id. The other three child tables have foreign keys to both the implementation table and their respective entity tables. All foreign keys have ON DELETE CASCADE and ON UPDATE CASCADE actions to ensure data integrity.

Validation During Data Insertion

Before inserting data, all mandatory fields were checked to ensure they contained values. Language names were checked for duplication. Parent records were inserted before child records. Foreign key values were checked to ensure they existed in parent tables.

Constraint Summary

Table	Primary Key	NOT NULL	Foreign Keys
language	language_id	language_name	None
function_table	function_id	function_name	None
data_structure	structure_id	structure_name	None
operator	operator_id	operator_name	None
implementation	implementation_id	language_id	language_id
function_implementation	implementation_id	function_id	implementation_id, function_id
structure_implementation	implementation_id	structure_id	implementation_id, structure_id
operator_implementation	implementation_id	operator_id	implementation_id, operator_id

Section 2: Table Screenshots

Insert screenshots from phpMyAdmin Browse tab for: language, function_table, data_structure, operator, implementation, function_implementation, structure_implementation, operator_implementation

Data_structure

	structure_id	structure_name	category	description
Edit Copy Delete	1	Array	Linear	Fixed-size sequential collection
Edit Copy Delete	2	List	Linear	Dynamic sequential collection
Edit Copy Delete	3	Dictionary	Associative	Key-value pair collection
Edit Copy Delete	4	Set	Collection	Unordered unique elements
Edit Copy Delete	5	Stack	Linear	LIFO data structure
Edit Copy Delete	6	Queue	Linear	FIFO data structure
Edit Copy Delete	7	Linked List	Linear	Pointer-based sequential collection
Edit Copy Delete	8	Tree	Hierarchical	Hierarchical node structure
Edit Copy Delete	9	Graph	Non-Linear	Nodes connected by edges
Edit Copy Delete	10	Hash Table	Associative	Hash function indexed structure

Function_implementation

←T→	implementation_id	function_id
Edit Copy Delete	1	1
Edit Copy Delete	2	1
Edit Copy Delete	3	1
Edit Copy Delete	4	1
Edit Copy Delete	5	2
Edit Copy Delete	6	2
Edit Copy Delete	7	2
Edit Copy Delete	8	3
Edit Copy Delete	9	3
Edit Copy Delete	10	3
Edit Copy Delete	11	4
Edit Copy Delete	12	4
Edit Copy Delete	13	4
Edit Copy Delete	14	5
Edit Copy Delete	15	5
Edit Copy Delete	16	6
Edit Copy Delete	17	6
Edit Copy Delete	18	9
Edit Copy Delete	19	9
Edit Copy Delete	20	10
Edit Copy Delete	21	10

Function_table

	function_id	function_name	category	description
Edit Copy Delete	1	print	I/O	Outputs text or values to standard output
Edit Copy Delete	2	input	I/O	Reads input from standard input
Edit Copy Delete	3	len	Utility	Returns the length of a collection or string
Edit Copy Delete	4	sort	Sorting	Sorts elements in a collection
Edit Copy Delete	5	map	Functional	Applies a function to each element
Edit Copy Delete	6	filter	Functional	Filters elements based on a condition
Edit Copy Delete	7	reduce	Functional	Reduces a collection to a single value
Edit Copy Delete	8	open	File I/O	Opens a file for reading or writing
Edit Copy Delete	9	split	String	Splits a string into an array/list
Edit Copy Delete	10	join	String	Joins elements into a string

Implementation

	implementation_id	language_id	version	date_added	example	result	notes	syntax
Edit Copy Delete	1	1	3.x	2026-02-22	print("Score:", 95, sep=" ")	Score: 95	Supports multiple arguments with customizable sepa...	print(value, sep=" ", end="\n")
Edit Copy Delete	2	2	C++11	2026-02-22	std::cout << "Temperature: " << 72 << "F" << std::...	Temperature: 72F	Chain multiple values with << operator	std::cout << value << std::endl;
Edit Copy Delete	3	3	Java 8+	2026-02-22	System.out.println("Items in cart: " + 5);	Items in cart: 5	Use + for string concatenation	System.out.println(value);
Edit Copy Delete	4	4	ES6+	2026-02-22	console.log(`User ${name} logged in`);	User John logged in	Template literals use backticks and ${} for interp...	console.log(value);
Edit Copy Delete	5	1	3.x	2026-02-22	age = int(input("Enter your age: "))	User enters: 25 -> age = 25	Always returns string; cast to int/float as needed	variable = input(prompt)
Edit Copy Delete	6	2	C++11	2026-02-22	int age; std::cout << "Enter age: "; std::cin >> a...	User enters: 25 -> age = 25	Use getline(cin, str) for strings with spaces	std::cin >> variable;
Edit Copy Delete	7	3	Java 8+	2026-02-22	Scanner sc = new Scanner(System.in); String name =...	User enters: Alice -> name = "Alice"	Use nextInt(), nextDouble() for other types	Scanner sc = new Scanner(System.in); sc.nextLine()...
Edit Copy Delete	8	1	3.x	2026-02-22	words = ["apple", "banana", "cherry"] len(words)	3	Works on strings, lists, tuples, dicts, sets	len(sequence)
Edit Copy Delete	9	2	C++11	2026-02-22	std::string name = "Alice"; name.size();	5	Returns size_t type; use .length() for strings too	container.size()
Edit Copy Delete	10	3	Java 8+	2026-02-22	int[] nums = {10, 20, 30, 40}; nums.length;	4	Arrays use .length (no parens), Strings use .lengt...	array.length or string.length() or list.size()
Edit Copy Delete	11	1	3.x	2026-02-22	prices = [29.99, 9.99, 49.99, 19.99] prices.sort()	[9.99, 19.99, 29.99, 49.99]	.sort() modifies in place; sorted() returns new li...	list.sort() or sorted(iterable)
Edit Copy Delete	12	2	C++11	2026-02-22	std::vector<int> v = {5, 2, 8, 1}; std::sort(v.beg...	{1, 2, 5, 8}	Requires #include <algorithm>; average O(n log n)	std::sort(begin, end)
Edit Copy Delete	13	3	Java 8+	2026-02-22	List<String> names = Arrays.asList("Zoe", "Amy", "...	[Amy, Max, Zoe]	Use Comparator for custom sorting	Arrays.sort(array) or Collections.sort(list)
Edit Copy Delete	14	1	3.x	2026-02-22	nums = [1, 2, 3, 4] squared = list(map(lambda x: x...	[1, 4, 9, 16]	Returns iterator; wrap in list() to see results	map(function, iterable)
Edit Copy Delete	15	4	ES6+	2026-02-22	const prices = [10, 20, 30]; const withTax = price...	[10.8, 21.6, 32.4]	Arrow functions make this concise	array.map(callback)
Edit Copy Delete	16	1	3.x	2026-02-22	ages = [12, 18, 25, 8, 30] adults = list(filter(la...	[18, 25, 30]	Returns elements where function returns True	filter(function, iterable)
Edit Copy Delete	17	4	ES6+	2026-02-22	const scores = [85, 42, 93, 67, 55]; const passing...	[85, 93, 67]	Returns new array with elements that pass the test	array.filter(callback)
Edit Copy Delete	18	1	3.x	2026-02-22	csv_line = "John,25,Engineer" fields = csv_line.sp...	["John", "25", "Engineer"]	Default separator is whitespace	string.split(separator)

Console

Language

	language_id	language_name
Edit Copy Delete	5	C
Edit Copy Delete	2	C++
Edit Copy Delete	7	Go
Edit Copy Delete	3	Java
Edit Copy Delete	4	JavaScript
Edit Copy Delete	1	Python
Edit Copy Delete	6	Ruby
Edit Copy Delete	8	Rust

Operator

	operator_id	operator_name	category	description
Edit Copy Delete	1	Addition	Arithmetic	Adds two operands
Edit Copy Delete	2	Subtraction	Arithmetic	Subtracts operands
Edit Copy Delete	3	Multiplication	Arithmetic	Multiplies operands
Edit Copy Delete	4	Division	Arithmetic	Divides operands
Edit Copy Delete	5	Modulus	Arithmetic	Returns remainder
Edit Copy Delete	6	Equal	Comparison	Checks equality
Edit Copy Delete	7	Not Equal	Comparison	Checks inequality
Edit Copy Delete	8	Greater Than	Comparison	Checks if greater
Edit Copy Delete	9	Less Than	Comparison	Checks if less
Edit Copy Delete	10	Logical AND	Logical	Both must be true
Edit Copy Delete	11	Logical OR	Logical	One must be true
Edit Copy Delete	12	Logical NOT	Logical	Inverts boolean
Edit Copy Delete	13	Assignment	Assignment	Assigns value
Edit Copy Delete	14	Bitwise AND	Bitwise	Bitwise AND operation
Edit Copy Delete	15	Bitwise OR	Bitwise	Bitwise OR operation

Operator_implementation

←T→	implementation_id	operator_id	symbol
Edit Copy Delete	35	1	+
Edit Copy Delete	36	1	+
Edit Copy Delete	37	2	-
Edit Copy Delete	38	3	*
Edit Copy Delete	39	3	*
Edit Copy Delete	40	4	/ or //
Edit Copy Delete	41	4	/
Edit Copy Delete	42	5	%
Edit Copy Delete	43	5	%
Edit Copy Delete	44	6	==
Edit Copy Delete	45	6	=== or ==
Edit Copy Delete	46	7	!=
Edit Copy Delete	47	8	>
Edit Copy Delete	48	9	<
Edit Copy Delete	49	10	and
Edit Copy Delete	50	10	&&
Edit Copy Delete	51	11	or
Edit Copy Delete	52	11	||
Edit Copy Delete	53	12	not
Edit Copy Delete	54	12	!
Edit Copy Delete	55	13	=

Structure_implementation

	implementation_id	structure_id
Edit Copy Delete	22	2
Edit Copy Delete	23	2
Edit Copy Delete	24	2
Edit Copy Delete	25	2
Edit Copy Delete	26	3
Edit Copy Delete	27	3
Edit Copy Delete	28	3
Edit Copy Delete	29	4
Edit Copy Delete	30	4
Edit Copy Delete	31	5
Edit Copy Delete	32	5
Edit Copy Delete	33	6
Edit Copy Delete	34	6

Section 3/4: SQL Queries

Query 1: SELECT with WHERE and Logical Operators

Purpose: Find all functions in Functional or String categories

```
SELECT function_id, function_name, category
FROM function_table
WHERE category = 'Functional' OR category = 'String';
```

	function_id	function_name	category
Edit Copy Delete	5	map	Functional
Edit Copy Delete	6	filter	Functional
Edit Copy Delete	7	reduce	Functional
Edit Copy Delete	9	split	String
Edit Copy Delete	10	join	String

Query 2: JOIN Expression

Purpose: Compare sort function across languages using multiple table joins

```
SELECT l.language_name, f.function_name, i.syntax, i.example
FROM implementation i
JOIN function_implementation fi ON i.implementation_id = fi.implementation_id
JOIN language l ON i.language_id = l.language_id
JOIN function_table f ON fi.function_id = f.function_id
```

WHERE f.function_name = 'sort';

language_name	function_name	syntax	example
Python	sort	list.sort() or sorted(iterable)	prices = [29.99, 9.99, 49.99, 19.99] prices.sort()
C++	sort	std::sort(begin, end)	std::vector<int> v = {5, 2, 8, 1}; std::sort(v.beg...
Java	sort	Arrays.sort(array) or Collections.sort(list)	List<String> names = Arrays.asList("Zoe", "Amy", "...

Query 3: GROUP BY with Aggregate Functions

Purpose: Count operators per category using GROUP BY and COUNT

```
SELECT l.language_name, COUNT(*) AS implementation_count
FROM implementation i
JOIN language l ON i.language_id = l.language_id
GROUP BY l.language_name
ORDER BY implementation_count DESC;
```

category	operator_count ▾ 1
Arithmetic	5
Comparison	4
Logical	3
Bitwise	2
Assignment	1

Query 4: View Query

Purpose: Use the pre built view to find all Python function implementations

```
SELECT language_name, function_name, syntax, example
FROM vw_function_lookup
WHERE language_name = 'Python';
```

language_name	function_name	syntax	example
Python	filter	filter(function, iterable)	ages = [12, 18, 25, 8, 30] adults = list(filter(la...
Python	input	variable = input(prompt)	age = int(input("Enter your age: "))
Python	join	separator.join(iterable)	words = ["Hello", "World", "2024"] " ".join(words)
Python	len	len(sequence)	words = ["apple", "banana", "cherry"] len(words)
Python	map	map(function, iterable)	nums = [1, 2, 3, 4] squared = list(map(lambda x: x...
Python	print	print(value, sep=" ", end="\n")	print("Score:", 95, sep=" ")
Python	sort	list.sort() or sorted(iterable)	prices = [29.99, 9.99, 49.99, 19.99] prices.sort()
Python	split	string.split(separator)	csv_line = "John,25,Engineer" fields = csv_line.sp...

Query 5: COUNT all available functions

Purpose: show the number of functions that could be "translated" or that are stored in the database.

SELECT COUNT(*) AS total_functions

FROM function_table;

total_functions
10