NarrVoca

Based on Vocora

Part A — Component 2: Database Design Document

Course:	CSCI 6333 — Database Systems
System Name:	NarrVoca (Based on Vocora)
Group Name:	NarrVoca Team
Group Captain:	To Be Decided
Members:	Ruben Aleman Silvia Osuna Andrea Garza
Due Date:	February 26, 2026

University of Texas Rio Grande Valley
Department of Computer Science

0 Contents

1 Problem Statement

NarrVoca (Based on Vocora) is a database-driven language learning system designed to support English-speaking learners studying Spanish and Mandarin Chinese through narrative-based comprehensible input. The system delivers short stories divided into interactive scenes where learners practice vocabulary and grammar in context. Unlike a traditional Learning Management System (LMS), NarrVoca organizes learning content around story moments (*story nodes*) and supports adaptive progression using branching logic based on learner performance or choice.

The original **Vocora** system (deployed at `vocora.vercel.app`) provides foundational functionality built with Next.js and TypeScript, backed by a Supabase (PostgreSQL) database. The base system stores AI-generated stories, vocabulary lists, user preferences, and cached word definitions. It supports audio generation, image generation, writing practice feedback, and a chat interface — all connected through the Supabase client (`lib/supabase.ts`) and a set of React hooks that manage real-time data fetching and state.

NarrVoca extends Vocora, an AI-powered language reinforcement platform originally developed at UTRGV in Spring 2025 (CSCI 4390), by introducing a structured relational database architecture, narrative node decomposition, grammar mapping, and adaptive branching — directly addressing the progress tracking and enhanced practice features identified as future work in the original system.

NarrVoca expands the base design into a structured narrative learning engine by introducing:

- Story decomposition (stories $\rightarrow$ nodes $\rightarrow$ multilingual text)
- Grammar mapping at the node level
- Node-level vocabulary targets
- User progress tracking per node and per story
- Interaction logging for performance scoring and adaptive branching
- Spaced repetition support via vocabulary mastery tracking

The database must ensure data integrity across story content, learning objectives, and learner progress while maintaining scalability for multilingual content and future extension (e.g., additional languages, tutoring modes, or analytics).

2 System Requirements and Scope

2.1 Target Users

- English-speaking learners studying Spanish (`es`) and Mandarin Chinese (`zh`).
- Users who learn best through short story-based input and repeated contextual practice.

2.2 Core System Functions

- Store story-based learning content in a structured, decomposed format.
- Link vocabulary and grammar points to specific story scenes (nodes).
- Track user progress through stories and scenes.
- Store learner interactions and performance scores.
- Support branching logic to determine the next scene based on performance or learner choice.
- Cache word definitions to minimize repeated AI generation calls.

2.3 Assumptions

- Users are identified by a unique `uid` (UUID) generated by Supabase Authentication.
- Spanish (`es`) and Chinese (`zh`) are the primary target languages; English (`en`) is used for translation support.
- The design follows normalization principles (3NF) to reduce redundancy and enforce data consistency.
- The system extends the existing Vocora schema without breaking existing functionality.

3 Entity-Relationship Design

3.1 Entities

3.1.1 Base (Vocora) Entities

- **USER_PREFERENCES**(uid, preferred_lang, practice_lang, email)
- **USER_STORIES**(id, uid, story, translated_story, image, language, created_at)
- **VOCAB_LISTS**(list_id, uid, language, name)
- **VOCAB_WORDS**(id, uid, word, language, list_id, created_at)
- **CACHED_DEFINITIONS**(word, story_hash, definition, part_of_speech, lang)

3.1.2 NarrVoca (New) Entities

- **STORIES**(story_id, title, target_language, difficulty_level, genre, created_at)
- **STORY_NODES**(node_id, story_id, sequence_order, context_description, is_checkpoint)
- **NODE_TEXT**(node_text_id, node_id, language_code, speaker, text_type, text_content, pinyin)

- **BRANCHING_LOGIC**(branch_id, node_id, condition_type, condition_value, next_node_id)
- **VOCABULARY**(vocab_id, language_code, term, translation_en, pinyin, difficulty_score)
- **GRAMMAR_POINTS**(grammar_id, language_code, rule_name, explanation_en, example_target)
- **INTERACTION_LOG**(interaction_id, uid, node_id, user_input, llm_feedback, accuracy_score, created_at)
- **NODE_VOCABULARY**(node_id, vocab_id, is_target)
- **NODE_GRAMMAR**(node_id, grammar_id)
- **USER_NODE_PROGRESS**(uid, node_id, status, best_score, completed_at)
- **USER_VOCAB_MASTERY**(uid, vocab_id, mastery_score, next_review_at)

3.2 Key Relationships and Cardinalities

Entity A	Card.	Entity B	Relationship
STORIES	1 : M	STORY_NODES	A story contains many nodes
STORY_NODES	1 : M	NODE_TEXT	A node has text in multiple languages
STORY_NODES	1 : M	BRANCHING_LOGIC	A node may have many outgoing branches
STORY_NODES	M : M	VOCABULARY	Via NODE_VOCABULARY
STORY_NODES	M : M	GRAMMAR_POINTS	Via NODE_GRAMMAR
USER	1 : M	INTERACTION_LOG	A user produces many interaction records
USER	M : M	VOCABULARY	Via USER_VOCAB_MASTERY
USER	M : M	STORY_NODES	Via USER_NODE_PROGRESS
VOCAB_LISTS	1 : M	VOCAB_WORDS	A list contains many words

3.3 Business Rules

- Each story belongs to exactly one target language.
- Each story node belongs to exactly one story.
- Node text may exist in multiple languages (English support + target language).
- Branch transitions must reference valid nodes within the same story.
- A vocabulary item can be reused across many nodes and stories.

- A user's mastery level for vocabulary is tracked independently from other users.
- A user's progress through nodes is tracked per node and per story.
- `uid` values are managed by Supabase Auth and are never stored redundantly.

4 Relational Database Design

4.1 Base Schema — Vocora Foundation

4.1.1 Table: cached_definitions

PK: word *Caches AI-generated word definitions to avoid redundant API calls.*

Column	Type	Notes
word	text	PK, NOT NULL
story_hash	text	Hash of the story context
definition	text	AI-generated definition
part_of_speech	text	e.g., noun, verb
lang	text	Language code (es, zh)

4.1.2 Table: user_preferences

PK: uid *Stores per-user language preferences, linked to Supabase Auth.*

Column	Type	Notes
uid	uuid	PK, NOT NULL (from Supabase Auth)
preferred_lang	text	Display language (en, es, zh), NOT NULL
practice_lang	text	Language being studied
email	varchar	User email (nullable)

4.1.3 Table: user_stories

PK: id *Stores AI-generated stories per user (legacy archive from Vocora).*

Column	Type	Notes
id	smallint	PK, NOT NULL
uid	uuid	NOT NULL (references Auth user)
story	text	Generated story content, NOT NULL
translated_story	text	English translation (nullable)
image	text	URL to generated image
language	text	Target language of the story
created_at	timestamp	Creation timestamp

4.1.4 Table: vocab_lists

PK: list_id *Stores vocabulary collections per user.*

Column	Type	Notes
list_id	bigint	PK, NOT NULL
uid	uuid	NOT NULL
language	text	Language of the list
name	text	List name

4.1.5 Table: vocab_words

PK: id **FK:** list_id → vocab_lists(list_id)

Column	Type	Notes
id	bigint	PK, NOT NULL
uid	uuid	NOT NULL
word	text	Vocabulary word, NOT NULL
language	text	Language code
list_id	bigint	FK → vocab_lists(list_id)
created_at	timestamptz	NOT NULL

4.2 NarrVoca Extension Schema

4.2.1 Table: stories

PK: story_id *Master table for structured narrative stories.*

Column	Type	Notes
story_id	bigint	PK, NOT NULL, auto-increment
title	text	Story title, NOT NULL
target_language	text	es or zh, NOT NULL
difficulty_level	text	beginner / intermediate / advanced
genre	text	e.g., adventure, daily life
created_at	timestamptz	Default: now()

4.2.2 Table: story_nodes

PK: node_id **FK:** story_id → stories(story_id)

Column	Type	Notes
node_id	bigint	PK, NOT NULL
story_id	bigint	FK → stories(story_id), NOT NULL
sequence_order	integer	Scene order within the story
context_description	text	Internal description for the scene
is_checkpoint	boolean	Whether node triggers assessment

4.2.3 Table: node_text

PK: node_text_id **FK:** node_id → story_nodes(node_id) *Supports multilingual text per node.*

Column	Type	Notes
node_text_id	bigint	PK
node_id	bigint	FK → story_nodes(node_id)
language_code	text	en, es, or zh
speaker	text	Character or narrator name
text_type	text	narration, dialogue, or prompt
text_content	text	Actual text content
pinyin	text	Romanization for Chinese (nullable)

4.2.4 Table: branching_logic

PK: branch_id **FK:** node_id, next_node_id → story_nodes

Column	Type	Notes
branch_id	bigint	PK
node_id	bigint	FK → story_nodes (source node)
condition_type	text	score_threshold, choice, or default
condition_value	text	e.g., “pass”, “fail”, “0.7”
next_node_id	bigint	FK → story_nodes (destination)

4.2.5 Table: vocabulary

PK: vocab_id *Master vocabulary dictionary shared across stories.*

Column	Type	Notes
vocab_id	bigint	PK
language_code	text	es or zh
term	text	Word/phrase in target language
translation_en	text	English translation
pinyin	text	Romanization (zh only, nullable)
difficulty_score	numeric	0.0–1.0 scale

4.2.6 Table: grammar_points

PK: grammar_id *Grammar rules used for tutoring and evaluation.*

Column	Type	Notes
grammar_id	bigint	PK
language_code	text	es or zh
rule_name	text	e.g., "ser vs. estar"
explanation_en	text	English explanation
example_target	text	Example in target language

4.2.7 Table: node_vocabulary (Associative)

PK: (node_id, vocab_id) *Resolves M:N between STORY_NODES and VOCABULARY.*

Column	Type	Notes
node_id	bigint	FK → story_nodes(node_id)
vocab_id	bigint	FK → vocabulary(vocab_id)
is_target	boolean	Whether this is a key learning word

4.2.8 Table: node_grammar (Associative)

PK: (node_id, grammar_id) *Resolves M:N between STORY_NODES and GRAMMAR_POINTS.*

Column	Type	Notes
node_id	bigint	FK → story_nodes(node_id)
grammar_id	bigint	FK → grammar_points(grammar_id)

4.2.9 Table: user_node_progress (Associative)

PK: (uid, node_id) *Tracks per-user progression through story nodes.*

Column	Type	Notes
uid	uuid	References Auth user
node_id	bigint	FK → story_nodes(node_id)
status	text	not_started, in_progress, completed
best_score	numeric	Highest accuracy score (0.0–1.0)
completed_at	timestamptz	Nullable until completed

4.2.10 Table: user_vocab_mastery (Associative)

PK: (uid, vocab_id) *Tracks mastery and supports spaced repetition scheduling.*

Column	Type	Notes
uid	uuid	References Auth user
vocab_id	bigint	FK → vocabulary(vocab_id)
mastery_score	numeric	0.0–1.0 mastery level
next_review_at	timestamptz	Spaced repetition schedule date

4.2.11 Table: interaction_log

PK: interaction_id *Logs all user responses for scoring and adaptive feedback.*

Column	Type	Notes
interaction_id	bigint	PK, auto-increment
uid	uuid	References Auth user
node_id	bigint	FK → story_nodes(node_id)
user_input	text	Learner's response
llm_feedback	text	AI-generated feedback
accuracy_score	numeric	Score 0.0–1.0
created_at	timestamptz	Default: now()

5 ER-to-Relational Mapping

5.1 Mapping Rule A — Strong Entity → Table

Each strong entity becomes a table with all its simple attributes. The entity's key becomes the primary key.

Examples:

- `stories(story_id PK, title, target_language, ...)`
- `vocabulary(vocab_id PK, language_code, term, ...)`
- `grammar_points(grammar_id PK, language_code, rule_name, ...)`

5.2 Mapping Rule B — 1:M Relationship → Foreign Key

For a 1:M relationship between A and B, the primary key of A is placed as a foreign key in B (the "many" side).

Examples:

- STORIES (1) → (M) STORY_NODES:
  `story_nodes.story_id FK → stories.story_id`
- STORY_NODES (1) → (M) NODE_TEXT:
  `node_text.node_id FK → story_nodes.node_id`

- STORY_NODES (1) → (M) BRANCHING_LOGIC:
  `branching_logic.node_id FK → story_nodes.node_id`
- VOCAB_LISTS (1) → (M) VOCAB_WORDS:
  `vocab_words.list_id FK → vocab_lists.list_id`

5.3 Mapping Rule C — M:N Relationship → Associative Table

For a M:N relationship between A and B, a new associative table is created containing the PKs of both A and B as foreign keys. The composite (PK_A, PK_B) becomes the primary key.

Examples:

- STORY_NODES ↔ VOCABULARY ⇒
  `node_vocabulary(node_id FK, vocab_id FK, is_target)`
- STORY_NODES ↔ GRAMMAR_POINTS ⇒
  `node_grammar(node_id FK, grammar_id FK)`
- USER ↔ STORY_NODES ⇒
  `user_node_progress(uid, node_id FK, status, best_score)`
- USER ↔ VOCABULARY ⇒
  `user_vocab_mastery(uid, vocab_id FK, mastery_score, next_review_at)`

5.4 Mapping Rule D — Relationship with Own Identity → Entity Table

When a relationship requires its own attributes and may occur multiple times, it is modeled as its own entity/table.

Example: BRANCHING_LOGIC has `condition_type` and `condition_value` as its own attributes and can occur multiple times per node, so it is represented as a full table with its own PK (`branch_id`).

5.5 Mapping Rule E — Multivalued Attributes → Separate Table

If one entity has "many of a thing" (e.g., many text lines per node, per language), it is represented as a separate table.

Example: A node has many localized text rows ⇒ `node_text` table with `language_code` as a discriminator.

6 Normalization Summary

The NarrVoca schema is designed to conform to Third Normal Form (3NF):

- **1NF:** All tables have atomic values, unique column names, and a defined primary key. No repeating groups exist.

- **2NF:** All non-key attributes in every table are fully functionally dependent on the entire primary key. Associative tables (`node_vocabulary`, `node_grammar`, `user_node_progress`, `user_vocab_mastery`) contain only attributes that depend on the full composite key.
- **3NF:** No transitive dependencies exist. For example, `vocabulary` stores language-specific attributes independently from node-level or user-level data. Story metadata is not repeated in `story_nodes`.

The use of associative tables resolves all many-to-many relationships and maintains normalized structure. Story text is stored separately (`node_text`) to support multilingual display without duplicating story node records per language.

7 Technology Stack and Team Acknowledgment

NarrVoca is built upon the existing Vocora platform, which uses **Supabase** as its backend-as-a-service provider. Supabase exposes a fully managed **PostgreSQL** database, meaning all data storage, querying, and relational integrity for NarrVoca are handled through PostgreSQL — the same database engine underlying the original Vocora schema. The NarrVoca schema extensions (Section 4) are deployed directly within the existing Supabase/PostgreSQL project, maintaining backward compatibility with all original Vocora tables. No additional cloud providers (e.g., AWS, Azure) are used; all backend services are provided by Supabase.

All team members have reviewed the system architecture, confirmed access to the Supabase project, and are aligned on the tools and technologies in use for this project.

7.1 Database and Backend Platform

The table below summarizes the platform, SQL dialect, client library, and hosting environment used by NarrVoca.

Component	Details
Backend Platform	Supabase (`supabase.com`) — backend-as-a-service
Database Engine	PostgreSQL (fully managed by Supabase)
SQL Dialect	PostgreSQL SQL — standard SQL with PostgreSQL extensions (e.g., `UUID` types, `timestamptz`, `JSONB` support)
Client Library	Supabase JavaScript Client (`lib/supabase.ts`), used within the Next.js/TypeScript frontend
Hosting / Deployment	Supabase cloud project; original Vocora deployed at `vocora.vercel.app`; NarrVoca extends this same schema
Authentication	Supabase Authentication — provides UUID-based `uid` for all user-linked tables

7.2 Team Member Acknowledgment

Each team member listed below confirms they are aware of the Supabase/PostgreSQL stack, have access to the project, and are actively working within this technology environment.

Team Member	Backend Platform	Database / SQL
Ruben Aleman	Supabase	PostgreSQL
Silvia Osuna	Supabase	PostgreSQL
Andrea Garza	Supabase	PostgreSQL

Note: The NarrVoca project does not utilize AWS or any external SQL server beyond Supabase's managed PostgreSQL instance. All schema design, queries, and data operations in this document are written in standard PostgreSQL SQL.

8 Diagrams

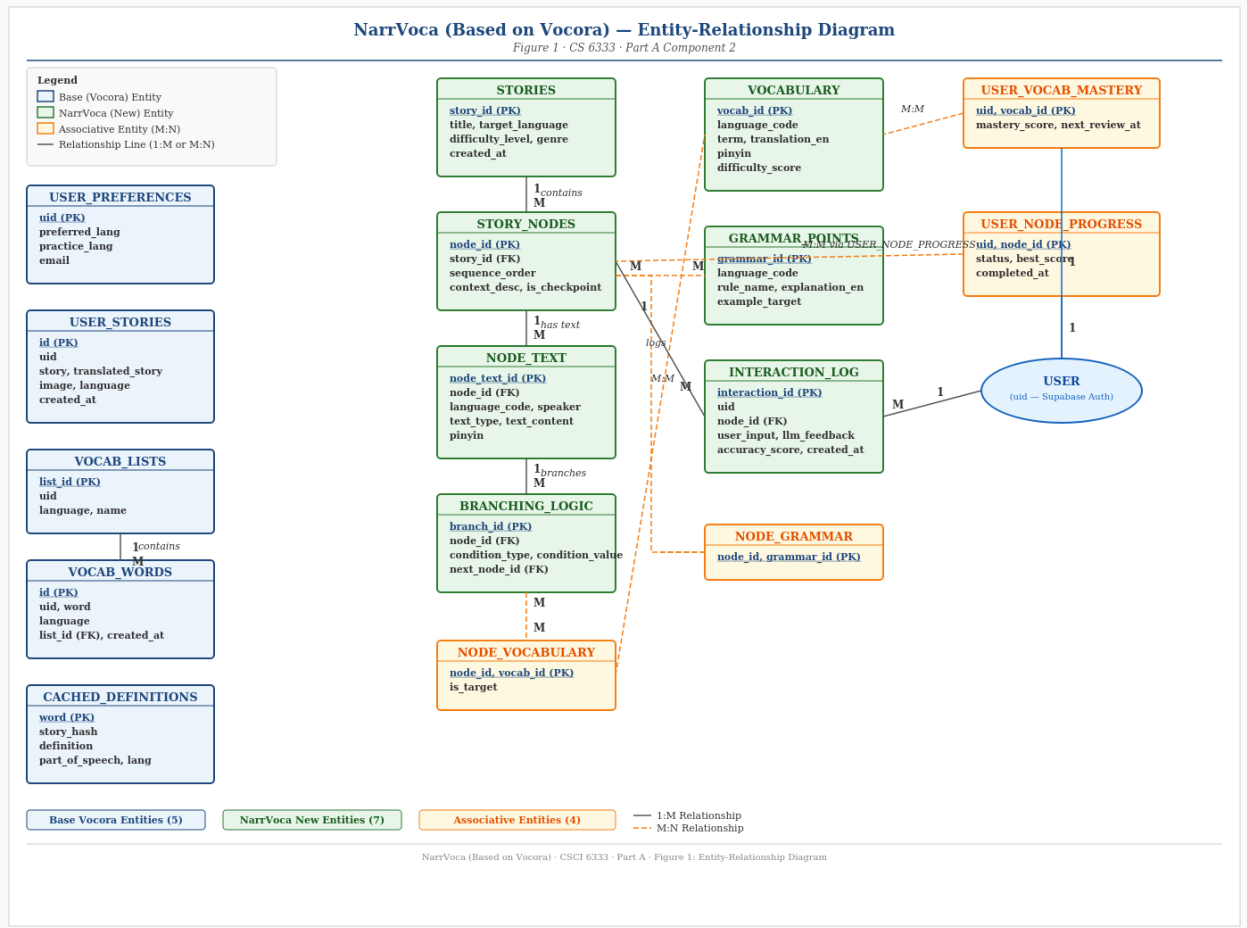

Figure 1: NarrVoca Entity-Relationship Diagram

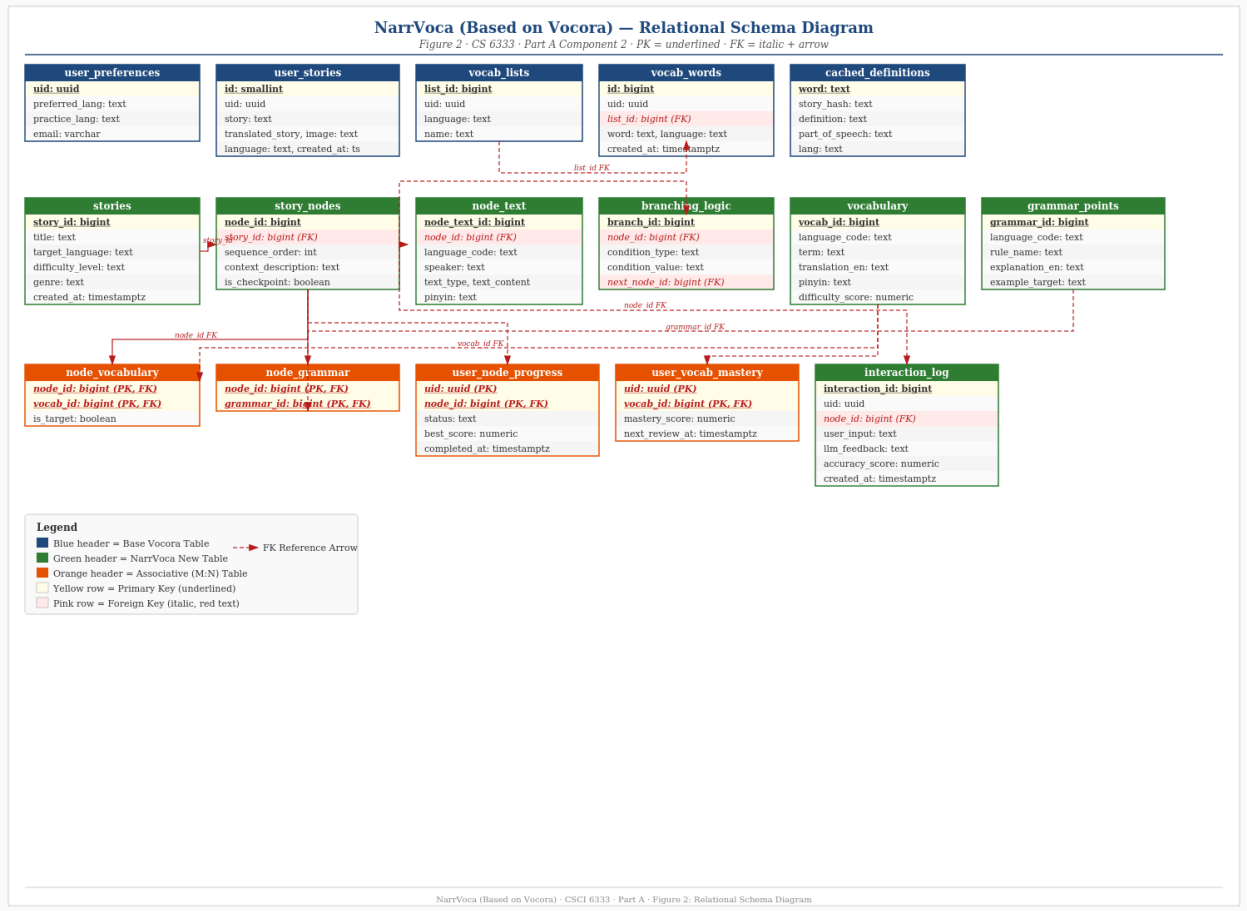

Figure 2: NarrVoca Relational Schema Diagram