


Alpha Release 1

Database Schema 1: Member Account & Profile

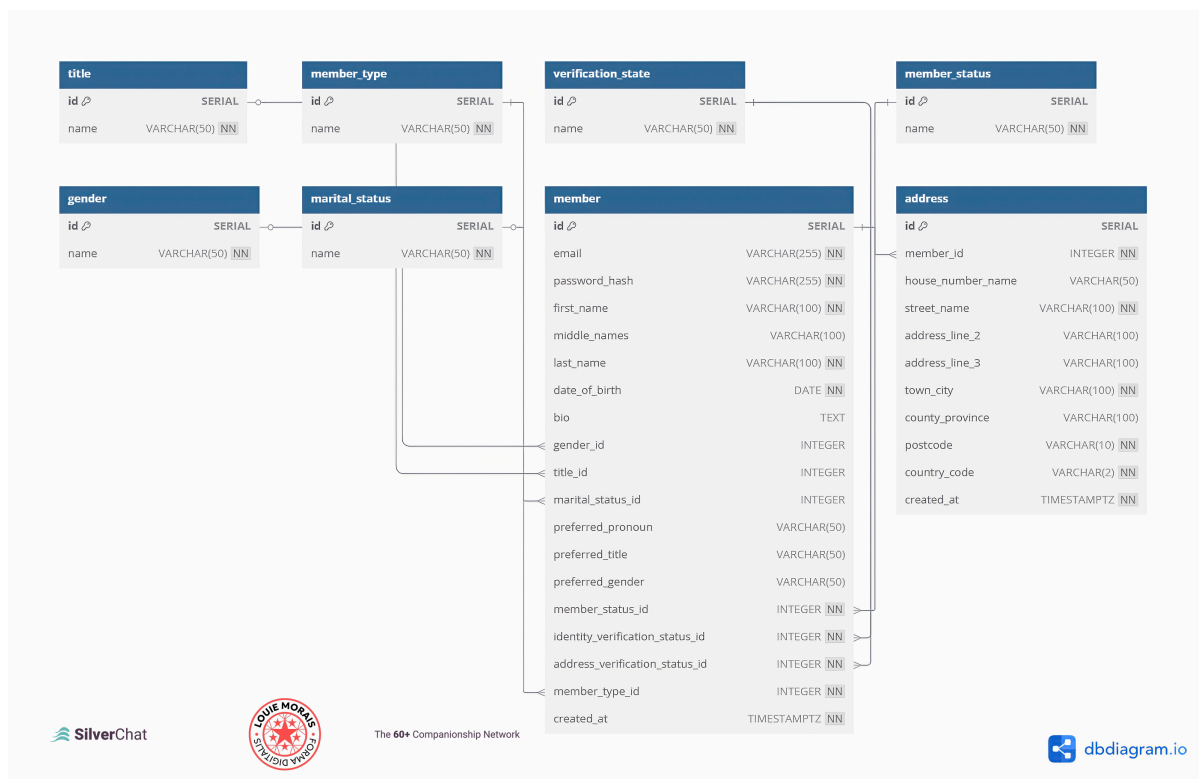


- Database Schema 1: Member Account & Profile
 - 1. Member Account & Profile Tables
 - 1.1. Lookup Tables
 - 1.2. Core Data Tables
 - 2. SQL Representation & Implementation Notes
 - 2.1. SQL Syntax
 - 2.2. DBML Syntax
 - 2.3. Knex.js Migration
 - 3. SilverChat Project Documentation
 - 4. External Sources
 - 4.1. Database Design for Social Networks (Schemas/Diagrams/ERMs/How Tos)
 - 4.2. Database GitHub Repos
 - 4.3. Database Infrastructure for Social Networks
 - 4.4. Database Design Software
 - 4.5. Facebook Reference
- 

1. Member Account & Profile Tables

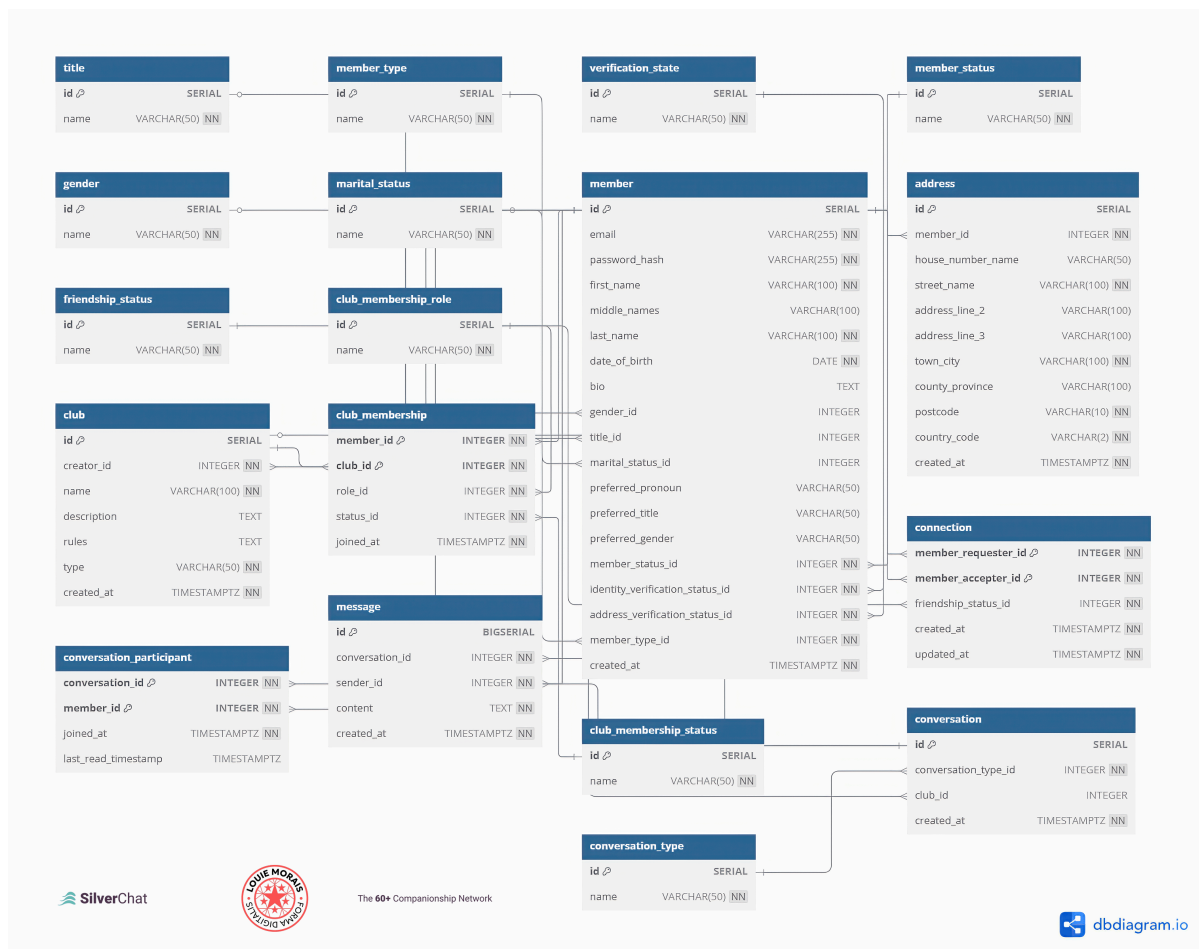
SilverChat - Member Account & Profile Database (Target: Alpha Release 1)

This schema defines the core tables required for member registration, authentication, basic profile information, status tracking, and address details for the SilverChat Alpha Release 1.



[View the live diagram on dbdiagram.io](#)

Final Database Design - Alpha Releases 1, 2 & 3:



[View the live diagram on dbdiagram.io](#)

1.1. Lookup Tables

These tables provide predefined options for various fields, ensuring consistency.

Table: `title`

Column Name	Data Type	Constraints	Description	Example Values
<code>id</code>	<code>SERIAL</code>	<code>PRIMARY KEY</code>	Auto-incrementing unique identifier for title.	1, 2, 3, 4, 5, 6, 7, 8, 9
<code>name</code>	<code>VARCHAR(50)</code>	<code>UNIQUE NOT NULL</code>	Name of the title option.	'Mr', 'Mrs', 'Ms', 'Miss', 'Mx', 'Dr', 'Prof', 'Prefer not to say', 'Other'

Table: `member_type` (Internal Use)

Column Name	Data Type	Constraints	Description	Example Values
<code>id</code>	<code>SERIAL</code>	<code>PRIMARY KEY</code>	Auto-incrementing unique identifier for type.	1, 2, 3, 4, 5
<code>name</code>	<code>VARCHAR(50)</code>	<code>UNIQUE NOT NULL</code>	Name of the member type (e.g., 'Standard').	'Standard', 'Former', 'Admin', 'Family', 'Supporter'

Table: `verification_state` (Used for identity and address checks)

Column Name	Data Type	Constraints	Description	Example Values
<code>id</code>	<code>SERIAL</code>	<code>PRIMARY KEY</code>	Auto-incrementing unique identifier for state.	1, 2, 3, 4, 5

Column Name	Data Type	Constraints	Description	Example Values
name	VARCHAR(50)	UNIQUE NOT NULL	Name of a verification step state.	'Not Started', 'Pending Review', 'Verified', 'Failed', 'Requires Resubmission'

Table: member_status (Internal Use - Overall account status)

Column Name	Data Type	Constraints	Description	Example Values
id	SERIAL	PRIMARY KEY	Auto-incrementing unique identifier for status.	1, 2, 3, 4, 5
name	VARCHAR(50)	UNIQUE NOT NULL	Name of the member account status (internal).	'Pending Verification', 'Active', 'Suspended', 'Under Investigation', 'Expelled'

Table: gender

Column Name	Data Type	Constraints	Description	Example Values
id	SERIAL	PRIMARY KEY	Auto-incrementing unique identifier for gender.	1, 2, 3, 4, 5
name	VARCHAR(50)	UNIQUE NOT NULL	Name of the gender option.	'Woman', 'Man', 'Non-binary', 'Prefer not to say', 'Other'

Table: marital_status

Column Name	Data Type	Constraints	Description	Example Values
id	SERIAL	PRIMARY KEY	Auto-incrementing unique identifier for marital status.	1, 2, 3, 4, 5, 6

Column Name	Data Type	Constraints	Description	Example Values
name	VARCHAR(50)	UNIQUE NOT NULL	Name of the marital status option.	'Single', 'Married', 'Civil Partnership', 'Divorced', 'Widowed', 'Prefer not to say'

1.2. Core Data Tables

These tables store the primary data about each member.

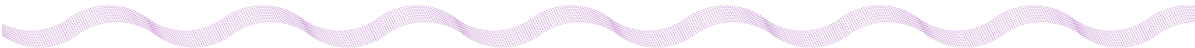
Table: member

Column Name	Data Type	Constraints	Description
id	SERIAL	PRIMARY KEY	Auto-incrementing unique identifier for the member.
email	VARCHAR(255)	UNIQUE NOT NULL	Member's unique email address (for login).
password_hash	VARCHAR(255)	NOT NULL	Securely hashed password.
first_name	VARCHAR(100)	NOT NULL	Member's first name(s).
middle_names	VARCHAR(100)		Member's middle name(s) (Optional).
last_name	VARCHAR(100)	NOT NULL	Member's last name(s).
date_of_birth	DATE	NOT NULL	Member's date of birth.
bio	TEXT		Short profile description (Optional).
gender_id	INTEGER		Foreign key to gender table (Optional/Nullable).
title_id	INTEGER		Foreign key to title table (Optional/Nullable).
marital_status_id	INTEGER		Foreign key to marital_status table (Optional/Nullable).
preferred_pronoun	VARCHAR(50)		Free-text preferred pronouns (Optional).
preferred_title	VARCHAR(50)		Free-text preferred title (Optional, e.g., if title is 'Other').
preferred_gender	VARCHAR(50)		Free-text preferred gender (Optional, e.g., if gender is 'Other').

Column Name	Data Type	Constraints	Description
member_status_id	INTEGER	NOT NULL DEFAULT 1	Foreign key to member_status table (Default: 'Pending Verification').
identity_verification_status_id	INTEGER	NOT NULL DEFAULT 1	Foreign key to verification_state (Default: 'Not Started').
address_verification_status_id	INTEGER	NOT NULL DEFAULT 1	Foreign key to verification_state (Default: 'Not Started').
member_type_id	INTEGER	NOT NULL DEFAULT 1	Foreign key to member_type table (Default: 'Standard').
created_at	TIMESTAMPTZ	NOT NULL DEFAULT NOW()	Timestamp of record creation.

Table: address

Column Name	Data Type	Constraints	Description
id	SERIAL	PRIMARY KEY	Auto-incrementing unique identifier for the address.
member_id	INTEGER	UNIQUE NOT NULL	Foreign key to member table (One-to-one). Deletes if member does.
house_number_name	VARCHAR(50)		House number or name.
street_name	VARCHAR(100)	NOT NULL	Street name.
address_line_2	VARCHAR(100)		Optional address line 2.
address_line_3	VARCHAR(100)		Optional address line 3.
town_city	VARCHAR(100)	NOT NULL	Town or city name.
county_province	VARCHAR(100)		County or province (Optional).
postcode	VARCHAR(10)	NOT NULL	Postcode (e.g., UK format).
country_code	VARCHAR(2)	NOT NULL DEFAULT 'GB'	ISO 3166-1 alpha-2 country code.
created_at	TIMESTAMPTZ	NOT NULL DEFAULT NOW()	Timestamp of record creation.



2. SQL Representation & Implementation Notes

The following SQL `CREATE TABLE` statements represent the database structure defined above. These are provided for illustrative and cross-referencing purposes. Actual implementation will use Knex.js migrations.

2.1. SQL Syntax

```
-- Illustrative SQL for Schema 1 (Alpha Release 1) - Actual implementation via Knex Migrations
```

```
-- Lookup Tables
```

```
CREATE TABLE title (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(50) UNIQUE NOT NULL  
);
```

```
CREATE TABLE member_type (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(50) UNIQUE NOT NULL  
);
```

```
CREATE TABLE verification_state (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(50) UNIQUE NOT NULL  
);
```

```
CREATE TABLE member_status (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(50) UNIQUE NOT NULL  
);
```

```
CREATE TABLE gender (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(50) UNIQUE NOT NULL  
);
```

```
CREATE TABLE marital_status (  
  id SERIAL PRIMARY KEY,  
  name VARCHAR(50) UNIQUE NOT NULL  
);
```

```
-- Core Data Tables
```

```
CREATE TABLE member (  
  id SERIAL PRIMARY KEY,  
  email VARCHAR(255) UNIQUE NOT NULL,  
  password_hash VARCHAR(255) NOT NULL,  
  first_name VARCHAR(100) NOT NULL,  
  middle_names VARCHAR(100),  
  last_name VARCHAR(100) NOT NULL,
```

```

date_of_birth DATE NOT NULL,
bio TEXT,
gender_id INTEGER,
title_id INTEGER,
marital_status_id INTEGER,
preferred_pronoun VARCHAR(50),
preferred_title VARCHAR(50),
preferred_gender VARCHAR(50),
member_status_id INTEGER NOT NULL DEFAULT 1,
identity_verification_status_id INTEGER NOT NULL DEFAULT 1,
address_verification_status_id INTEGER NOT NULL DEFAULT 1,
member_type_id INTEGER NOT NULL DEFAULT 1,
created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

-- Foreign Key Constraints
FOREIGN KEY (gender_id) REFERENCES gender(id),
FOREIGN KEY (title_id) REFERENCES title(id),
FOREIGN KEY (marital_status_id) REFERENCES marital_status(id),
FOREIGN KEY (member_status_id) REFERENCES member_status(id),
FOREIGN KEY (identity_verification_status_id) REFERENCES verification_state(id),
FOREIGN KEY (address_verification_status_id) REFERENCES verification_state(id),
FOREIGN KEY (member_type_id) REFERENCES member_type(id)
);

CREATE TABLE address (
    id SERIAL PRIMARY KEY,
    member_id INTEGER UNIQUE NOT NULL,
    house_number_name VARCHAR(50),
    street_name VARCHAR(100) NOT NULL,
    address_line_2 VARCHAR(100),
    address_line_3 VARCHAR(100),
    town_city VARCHAR(100) NOT NULL,
    county_province VARCHAR(100),
    postcode VARCHAR(10) NOT NULL,
    country_code VARCHAR(2) NOT NULL DEFAULT 'GB',
    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

-- Foreign Key Constraint
FOREIGN KEY (member_id) REFERENCES member(id) ON DELETE CASCADE
);

```

2.2. DBML Syntax

```

// SilverChat - Schema 1: Member Account & Profile (Alpha Release 1)
// DBML Schema Definition

// --- Lookup Tables ---

Table title {
    id SERIAL [pk]
    name VARCHAR(50) [unique, not null]
}

Table member_type {
    id SERIAL [pk]

```



```

    name VARCHAR(50) [unique, not null, note: 'Internal Use: Standard, Former, Admin,
Family, Supporter']
}

Table verification_state {
    id SERIAL [pk]
    name VARCHAR(50) [unique, not null, note: 'Used for identity and address checks: Not
Started, Pending Review, Verified, Failed, Requires Resubmission']
}

Table member_status {
    id SERIAL [pk]
    name VARCHAR(50) [unique, not null, note: 'Internal Use - Overall account status:
Pending Verification, Active, Suspended, Under Investigation, Expelled']
}

Table gender {
    id SERIAL [pk]
    name VARCHAR(50) [unique, not null, note: 'Woman, Man, Non-binary, Prefer not to say,
Other']
}

Table marital_status {
    id SERIAL [pk]
    name VARCHAR(50) [unique, not null, note: 'Single, Married, Civil Partnership,
Divorced, widowed, Prefer not to say']
}

// --- Core Data Tables ---

Table member {
    id SERIAL [pk]
    email VARCHAR(255) [unique, not null]
    password_hash VARCHAR(255) [not null]
    first_name VARCHAR(100) [not null]
    middle_names VARCHAR(100)
    last_name VARCHAR(100) [not null]
    date_of_birth DATE [not null]
    bio TEXT
    gender_id INTEGER [ref: > gender.id]
    title_id INTEGER [ref: > title.id]
    marital_status_id INTEGER [ref: > marital_status.id]
    preferred_pronoun VARCHAR(50)
    preferred_title VARCHAR(50) [note: 'Free-text if title is Other']
    preferred_gender VARCHAR(50) [note: 'Free-text if gender is Other']
    member_status_id INTEGER [not null, default: 1, ref: > member_status.id, note:
'Default: Pending Verification']
    identity_verification_status_id INTEGER [not null, default: 1, ref: >
verification_state.id, note: 'Default: Not Started']
    address_verification_status_id INTEGER [not null, default: 1, ref: >
verification_state.id, note: 'Default: Not Started']
    member_type_id INTEGER [not null, default: 1, ref: > member_type.id, note: 'Default:
Standard']
    created_at TIMESTAMPTZ [not null, default: `NOW()`]
}

Table address {
    id SERIAL [pk]
    member_id INTEGER [unique, not null] // FK defined via explicit Ref below
    house_number_name VARCHAR(50)
    street_name VARCHAR(100) [not null]
    address_line_2 VARCHAR(100)
    address_line_3 VARCHAR(100)

```

```

town_city VARCHAR(100) [not null]
county_province VARCHAR(100)
postcode VARCHAR(10) [not null]
country_code VARCHAR(2) [not null, default: 'GB']
created_at TIMESTAMPTZ [not null, default: `NOW()`]

indexes {
  member_id [unique]
}
}

// --- Relationships ---
Ref address_member: address.member_id > member.id [delete: cascade]

```

2.3. Knex.js Migration

This code represents the content of the first migration file (e.g., `YYYYMMDDHHMMSS_create_alpha1_schema.js`) generated via `npx knex migrate:make create_alpha1_schema`. It uses the Knex.js schema builder API to create the tables defined in this schema document.

```

// migrations/YYYYMMDDHHMMSS_create_alpha1_schema.js

exports.up = function(knex) {
  return knex.schema
    // 1. Lookup Tables
    .createTable('title', function(table) {
      table.increments('id').primary(); // SERIAL PRIMARY KEY
      table.string('name', 50).unique().notNulltable();
    })
    .createTable('member_type', function(table) {
      table.increments('id').primary();
      table.string('name', 50).unique().notNulltable();
    })
    .createTable('verification_state', function(table) {
      table.increments('id').primary();
      table.string('name', 50).unique().notNulltable();
    })
    .createTable('member_status', function(table) {
      table.increments('id').primary();
      table.string('name', 50).unique().notNulltable();
    })
    .createTable('gender', function(table) {
      table.increments('id').primary();
      table.string('name', 50).unique().notNulltable();
    })
    .createTable('marital_status', function(table) {
      table.increments('id').primary();
      table.string('name', 50).unique().notNulltable();
    })
    // 2. Core Data Tables
    .createTable('member', function(table) {

```

```

    table.increments('id').primary(); // SERIAL PRIMARY KEY
    table.string('email', 255).unique().notNullable();
    table.string('password_hash', 255).notNullable();
    table.string('first_name', 100).notNullable();
    table.string('middle_names', 100);
    table.string('last_name', 100).notNullable();
    table.date('date_of_birth').notNullable();
    table.text('bio');

    table.integer('gender_id').unsigned().references('id').inTable('gender').onDelete('SET
    NULL').onUpdate('CASCADE'); // Optional FK

    table.integer('title_id').unsigned().references('id').inTable('title').onDelete('SET
    NULL').onUpdate('CASCADE'); // Optional FK

    table.integer('marital_status_id').unsigned().references('id').inTable('marital_status'
    ).onDelete('SET NULL').onUpdate('CASCADE'); // Optional FK
    table.string('preferred_pronoun', 50);
    table.string('preferred_title', 50);
    table.string('preferred_gender', 50);

    table.integer('member_status_id').unsigned().notNullable().defaultTo(1).references('id'
    ).inTable('member_status').onDelete('RESTRICT').onUpdate('CASCADE'); // Default: Pending
    Verification

    table.integer('identity_verification_status_id').unsigned().notNullable().defaultTo(1).
    references('id').inTable('verification_state').onDelete('RESTRICT').onUpdate('CASCADE');
    // Default: Not Started

    table.integer('address_verification_status_id').unsigned().notNullable().defaultTo(1).r
    eferences('id').inTable('verification_state').onDelete('RESTRICT').onUpdate('CASCADE');
    // Default: Not Started

    table.integer('member_type_id').unsigned().notNullable().defaultTo(1).references('id').
    inTable('member_type').onDelete('RESTRICT').onUpdate('CASCADE'); // Default: Standard
    table.timestamp('created_at', { useTz: true
    }).notNullable().defaultTo(knex.fn.now()); // TIMESTAMPTZ DEFAULT NOW()

    // Add indexes for frequently queried columns if needed later (e.g., email)
    table.index('email');
  })
  .createTable('address', function(table) {
    table.increments('id').primary();

    table.integer('member_id').unsigned().unique().notNullable().references('id').inTable('
    member').onDelete('CASCADE').onUpdate('CASCADE'); // One-to-one with CASCADE delete
    table.string('house_number_name', 50);
    table.string('street_name', 100).notNullable();
    table.string('address_line_2', 100);
    table.string('address_line_3', 100);
    table.string('town_city', 100).notNullable();
    table.string('county_province', 100);
    table.string('postcode', 10).notNullable();
    table.string('country_code', 2).notNullable().defaultTo('GB');
    table.timestamp('created_at', { useTz: true
    }).notNullable().defaultTo(knex.fn.now());

    // Index on member_id is created automatically by .unique()
    table.index('postcode'); // Index postcode for potential searches
  });
};

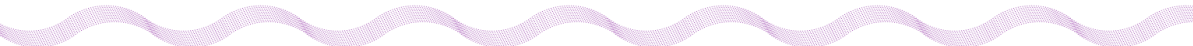
exports.down = function(knex) {

```

```
return knex.schema
  // Drop tables in reverse order of creation due to dependencies
  .dropTableIfExists('address')
  .dropTableIfExists('member')
  // Drop lookup tables last
  .dropTableIfExists('marital_status')
  .dropTableIfExists('gender')
  .dropTableIfExists('member_status')
  .dropTableIfExists('verification_state')
  .dropTableIfExists('member_type')
  .dropTableIfExists('title');
};
```



3. SilverChat Project Documentation

- [SilverChat Project Scope \(README\) | GitHub](#)
 - [SilverChat Technical Architecture | GitHub](#)
 - **THIS DOCUMENT:** *SilverChat Database Schema 1: Member Account and Profile*
 - [SilverChat Database Schema 2: Member Networking | GitHub](#)
 - [SilverChat Database Schema 3: Member Messaging | GitHub](#)
- 

4. External Sources

4.1. Database Design for Social Networks (Schemas/Diagrams/ERMs/How Tos)

- [Facebook database schema | Reverse engineering by Anatoly Lu... | Flickr](#)
- [Database schema for Social Networking Platform - Surfside Media](#)
- [Building a Social Network: Part I | by Kenneth Reilly | ITNEXT](#)
- [Social network schema design in DynamoDB - Amazon DynamoDB](#)
- [Building a social Media Platform: How should the database schema be designed to efficiently store user data, content, and interactions? | by Brecht Corbeel | Medium](#)
- [SQLAlchemy: Designing a Social Network Database Schema - Sling Academy](#)
- [Databases, SQL Server, and Data Models Examples](#)
- [mysql - Implementing Comments and Likes in database - Stack Overflow](#)
- [How to Design Database for Social Media Platform | GeeksforGeeks](#)

- [How to Design Database for Followers-Following Systems in Social Media Apps? | GeeksforGeeks](#)
 - [How to Design ER Diagrams for Social Media Networks | GeeksforGeeks](#)
 - [Resources: Database Design for Social Network - Code Doodle](#)
 - [Design Database For Social Network System In MySQL | Tutorials24x7](#)
-

4.2. Database GitHub Repos

- [Messenger Database Design Concept](#)
 - [The Social Network System Database Design in MySQL to manage the Users, Friends, Follower, Messages, and Groups.](#)
-

4.3. Database Infrastructure for Social Networks

- [A thorough insight into the databases used @Facebook - Scaleyourapp](#)
 - [Which database is best for creating a social networking application? - Quora](#)
 - [MySQL vs. MongoDB: The Pros and Cons When Building a Social Network](#)
-

4.4. Database Design Software

- [Top 10 Free Database Diagram Design Tools in 2025](#)
-

4.5. Facebook Reference

