**🏦 SQL Database Tables & Relationships Life Game**

**1. USERS**

sql

**CREATE** **TABLE** users (

id **SERIAL** **PRIMARY** **KEY**,

email **VARCHAR**(255) **UNIQUE** NOT NULL,

username **VARCHAR**(80) **UNIQUE** NOT NULL,

avatar **VARCHAR**(255),

join\_date **TIMESTAMP** **DEFAULT** NOW(),

password\_hash **VARCHAR**(255),

theme **VARCHAR**(40) **DEFAULT** 'default'

);

**2. CHARACTERS**

sql

**CREATE** **TABLE** characters (

id **SERIAL** **PRIMARY** **KEY**,

user\_id **INTEGER** **REFERENCES** users(id),

class **VARCHAR**(32),

bio **TEXT**,

goals **TEXT**,

**level** **INTEGER** **DEFAULT** 1,

xp **INTEGER** **DEFAULT** 0,

hp **INTEGER** **DEFAULT** 100,

coins **INTEGER** **DEFAULT** 100,

prestige\_level **INTEGER** **DEFAULT** 0,

title **VARCHAR**(120),

last\_login **TIMESTAMP**

);

**3. SKILLS**

sql

**CREATE** **TABLE** skills (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

name **VARCHAR**(64),

xp **INTEGER** **DEFAULT** 0,

**level** **INTEGER** **DEFAULT** 1,

unlocked **BOOLEAN** **DEFAULT** FALSE

);

**4. HABITS**

sql

**CREATE** **TABLE** habits (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

skill\_id **INTEGER** **REFERENCES** skills(id),

name **VARCHAR**(100),

**type** **VARCHAR**(10) **CHECK** (**type** IN ('good','bad')),

xp\_value **INTEGER** **DEFAULT** 0,

hp\_value **INTEGER** **DEFAULT** 0,

streak **INTEGER** **DEFAULT** 0,

last\_completed **DATE**,

template\_id **INTEGER** **REFERENCES** habit\_templates(id)

);

**5. HABIT\_TEMPLATES**

sql

**CREATE** **TABLE** habit\_templates (

id **SERIAL** **PRIMARY** **KEY**,

name **VARCHAR**(80),

skill\_name **VARCHAR**(64),

description **TEXT**

);

**6. EVENTS**

sql

**CREATE** **TABLE** events (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

event\_type **VARCHAR**(50),

value\_change **INTEGER**,

xp\_change **INTEGER**,

hp\_change **INTEGER**,

coins\_change **INTEGER**,

description **TEXT**,

event\_date **TIMESTAMP** **DEFAULT** NOW()

);

**7. PROJECTS (QUESTS)**

sql

**CREATE** **TABLE** projects (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

area\_id **INTEGER** **REFERENCES** areas(id),

title **VARCHAR**(120),

description **TEXT**,

total\_xp **INTEGER** **DEFAULT** 0,

coin\_reward **INTEGER** **DEFAULT** 0,

difficulty **VARCHAR**(32),

deadline **DATE**,

completed **BOOLEAN** **DEFAULT** FALSE

);

**8. AREAS (PARA Structure)**

sql

**CREATE** **TABLE** areas (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

name **VARCHAR**(50),

description **TEXT**

);

**9. TASKS (SUBQUESTS)**

sql

**CREATE** **TABLE** tasks (

id **SERIAL** **PRIMARY** **KEY**,

project\_id **INTEGER** **REFERENCES** projects(id),

title **VARCHAR**(120),

completed **BOOLEAN** **DEFAULT** FALSE,

xp **INTEGER** **DEFAULT** 0,

coins **INTEGER** **DEFAULT** 0,

difficulty **VARCHAR**(32)

);

**10. RESOURCES (Knowledge Notes)**

sql

**CREATE** **TABLE** resources (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

title **VARCHAR**(120),

**type** **VARCHAR**(32),

content **TEXT**,

added **TIMESTAMP** **DEFAULT** NOW()

);

**11. ARCHIVE / LOGS**

sql

**CREATE** **TABLE** archive (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

project\_id **INTEGER** **REFERENCES** projects(id),

completed\_on **TIMESTAMP**,

xp\_earned **INTEGER**,

coins\_earned **INTEGER**

);

**12. ITEMS (Marketplace/Inventory)**

sql

**CREATE** **TABLE** items (

id **SERIAL** **PRIMARY** **KEY**,

name **VARCHAR**(64),

item\_type **VARCHAR**(32),

rarity **VARCHAR**(32),

description **TEXT**,

effect **TEXT**,

cost **INTEGER** **DEFAULT** 0

);

**13. INVENTORY**

sql

**CREATE** **TABLE** inventory (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

item\_id **INTEGER** **REFERENCES** items(id),

quantity **INTEGER** **DEFAULT** 1,

acquired **TIMESTAMP** **DEFAULT** NOW()

);

**14. TRANSACTIONS**

sql

**CREATE** **TABLE** **transactions** (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

**type** **VARCHAR**(32), *-- 'earn', 'spend', 'penalty', 'restore'*

amount **INTEGER**,

item\_id **INTEGER** **REFERENCES** items(id),

description **TEXT**,

trans\_date **TIMESTAMP** **DEFAULT** NOW()

);

**15. GUILDS**

sql

**CREATE** **TABLE** guilds (

id **SERIAL** **PRIMARY** **KEY**,

name **VARCHAR**(100) **UNIQUE**,

description **TEXT**,

leader\_id **INTEGER** **REFERENCES** users(id),

xp\_pool **INTEGER** **DEFAULT** 0,

created **TIMESTAMP** **DEFAULT** NOW()

);

**16. GUILD\_MEMBERS**

sql

**CREATE** **TABLE** guild\_members (

guild\_id **INTEGER** **REFERENCES** guilds(id),

user\_id **INTEGER** **REFERENCES** users(id),

joined **TIMESTAMP** **DEFAULT** NOW(),

is\_admin **BOOLEAN** **DEFAULT** FALSE,

**PRIMARY** **KEY**(guild\_id, user\_id)

);

**17. PARTY**

sql

**CREATE** **TABLE** party (

id **SERIAL** **PRIMARY** **KEY**,

project\_id **INTEGER** **REFERENCES** projects(id),

guild\_id **INTEGER** **REFERENCES** guilds(id),

**status** **VARCHAR**(32)

);

**18. FRIENDS**

sql

**CREATE** **TABLE** friends (

user\_id **INTEGER** **REFERENCES** users(id),

friend\_id **INTEGER** **REFERENCES** users(id),

**status** **VARCHAR**(32) **DEFAULT** 'pending',

**PRIMARY** **KEY**(user\_id, friend\_id)

);

**19. MESSAGES (Community, Guild Chat)**

sql

**CREATE** **TABLE** messages (

id **SERIAL** **PRIMARY** **KEY**,

sender\_id **INTEGER** **REFERENCES** users(id),

guild\_id **INTEGER** **REFERENCES** guilds(id),

content **TEXT**,

sent **TIMESTAMP** **DEFAULT** NOW()

);

**20. ACHIEVEMENTS**

sql

**CREATE** **TABLE** achievements (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

title **VARCHAR**(120),

description **TEXT**,

reward\_type **VARCHAR**(32),

bonus\_value **INTEGER**,

unlocked\_on **TIMESTAMP** **DEFAULT** NOW()

);

**21. SETTINGS**

sql

**CREATE** **TABLE** settings (

user\_id **INTEGER** **REFERENCES** users(id),

level\_xp\_formula **TEXT**,

overdraft\_rule **TEXT**,

notification\_times **TEXT**,

theme **VARCHAR**(32),

**PRIMARY** **KEY**(user\_id)

);

**22. JOURNAL (Reflection, AI)**

sql

**CREATE** **TABLE** journal (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

entry **TEXT**,

wisdom\_xp **INTEGER** **DEFAULT** 0,

entry\_date **TIMESTAMP** **DEFAULT** NOW()

);

**23. AI\_LOGS**

sql

**CREATE** **TABLE** ai\_logs (

id **SERIAL** **PRIMARY** **KEY**,

character\_id **INTEGER** **REFERENCES** characters(id),

message **TEXT**,

insight\_type **VARCHAR**(32),

**timestamp** **TIMESTAMP** **DEFAULT** NOW()

);

**24. RNG\_EVENTS (Random Cards, Daily Events)**

sql

**CREATE** **TABLE** rng\_events (

id **SERIAL** **PRIMARY** **KEY**,

description **TEXT**,

effect **TEXT**,

rarity **VARCHAR**(32),

available **BOOLEAN** **DEFAULT** TRUE,

last\_issued **DATE**

);

**25. MENTORSHIP**

sql

**CREATE** **TABLE** mentorship (

id **SERIAL** **PRIMARY** **KEY**,

mentor\_id **INTEGER** **REFERENCES** users(id),

mentee\_id **INTEGER** **REFERENCES** users(id),

started **TIMESTAMP** **DEFAULT** NOW(),

**status** **VARCHAR**(32)

);

**25. Systems Log**

sql

**CREATE** **TABLE** systems\_log (

id **SERIAL** **PRIMARY** **KEY**,

**timestamp** **TIMESTAMP** **DEFAULT** NOW(), *-- when the log was written*

actor\_type **VARCHAR**(40), *-- 'user', 'system', 'cron', 'admin', 'guild', etc.*

actor\_id **INTEGER**, *-- references users.id, admin id, automation id, etc.*

target\_type **VARCHAR**(40), *-- e.g. 'character', 'habit', 'project', 'guild', etc.*

target\_id **INTEGER**, *-- references target entity id*

**action** **VARCHAR**(100), *-- short action description ('habit\_marked', 'quest\_completed', 'coin\_spent', etc.)*

detail **TEXT**, *-- full description or JSONB detail for context*

outcome **VARCHAR**(64), *-- result or status ('success', 'failure', 'warning', etc.)*

severity **VARCHAR**(20) **DEFAULT** 'info', *-- info / warning / error / critical*

source **VARCHAR**(80), *-- where did it originate? ('web', 'mobile', 'n8n', 'system\_cron', etc.)*

ip\_address **VARCHAR**(64), *-- optional, for debugging/security tracing*

extra JSONB **DEFAULT** '{}' *-- extensible field for future metadata*

);

**🔗 ALL CONNECTIONS AND RELATIONS**

* Each user has one character (1:1)
* Each character has many skills, habits, achievements, inventory, journal, archive, projects, areas, resources, and events
* Each habit belongs to a skill
* Each project (quest) belongs to an area and has many tasks
* Guilds have many guild\_members; each user can join many guilds (M:N via guild\_members)
* Party links quests/projects and guilds together (for coop quests)
* Inventory tracks item ownership per character
* Friendship table links two users (M:N)
* Messages belong to a sender (user) and have an optional guild id (for chat rooms)
* Settings are per user
* AI logs and journals are per character for companion/reflective interactions
* Mentorship connects mentors and mentees by user id
* Events and rng\_events track all status changes, battles, bonuses, etc.

**🔗 Logging**

* Log every backend automation step: When an n8n workflow triggers (habit XP update, HP penalty, coin shop purchase), write an entry.
* Track user/session activity: When users complete quests/habits or change key settings.
* Monitor admin/system ops: When cron jobs or admin actions (e.g., data reset, mass notifications) run.
* Debugging and security: Records IP, timestamp, and relevant details so you can trace bugs, suspicious activity, or audit gameplay flows.
* Extensible "extra" field: Allows for future logging needs; e.g., storing diff snapshots, additional flags, etc.

**🔗 Relation Advice**

* The actor\_type and actor\_id allow you to link log activity to any other table/entity.
* Use indexes on timestamp, action, and outcome for fast querying/filtering, especially when scaling logs for analytics or audits.