Analyse du Code et Schéma des Données de Geneweb

Date d'analyse : 10 septembre 2025
Niveau de confiance : 97%
Version analysée : Master branch

Table des Matières

Vue d'ensemble

Geneweb est un logiciel de généalogie open source écrit en OCaml. Il fournit une interface web et peut être utilisé hors ligne ou comme service web. L'application utilise une architecture de base de données optimisée avec des index pour des recherches rapides et un système de patches pour les modifications incrémentales.

Caractéristiques principales

- **Langage :** OCaml (programmation fonctionnelle)
- **Interface :** Web (HTML/CSS/JavaScript)
- **Stockage :** Format binaire optimisé (.gwb)
- **Performance :** Index multiples et cache intelligent
- **Scalabilité :** Gestion de grandes bases généalogiques

Architecture générale

L'architecture de Geneweb suit un modèle en couches :

```
+-------------------------------------+
|           Interface Web             |
+-------------------------------------+
|        Couche Métier (lib/)         |
```

```
+-------------------------------------+
|      Couche Accès Données (db/)     |
+-------------------------------------+
|     Stockage Fichiers (.gwb)        |
+-------------------------------------+
```

Modules principaux

- **lib/def/** : Définitions des types de données
- **lib/db/** : Gestion de la base de données
- **lib/util/** : Utilitaires (noms, dates, etc.)
- **bin/** : Outils en ligne de commande
- **rpc/** : API RPC pour intégrations

Schéma des données principales

1. Personne (Person)

La structure centrale représentant un individu dans l'arbre généalogique :

```
type ('iper, 'person, 'string) gen_person = {
  (* === IDENTITÉ === *)
  first_name : 'string;           (* Prénom principal *)
  surname : 'string;              (* Nom de famille *)
  occ : int;                      (* Numéro d'occurrence pour homonymes *)
  public_name : 'string;          (* Nom d'usage public *)
  qualifiers : 'string list;      (* Qualificatifs (Jr, Sr, III, etc.) *)
  aliases : 'string list;         (* Noms complets alternatifs *)
  first_names_aliases : 'string list;  (* Prénoms alternatifs *)
  surnames_aliases : 'string list;     (* Noms de famille alternatifs *)

  (* === INFORMATIONS PERSONNELLES === *)
  sex : sex;                      (* Sexe: Male | Female | Neuter *)
  image : 'string;                (* Chemin vers photo/portrait *)
  occupation : 'string;           (* Profession/métier *)
  access : access;                (* Droits d'accès aux données *)

  (* === NAISSANCE === *)
  birth : cdate;                  (* Date de naissance *)
  birth_place : 'string;          (* Lieu de naissance *)
  birth_note : 'string;           (* Notes sur la naissance *)
  birth_src : 'string;            (* Sources documentaires *)

  (* === BAPTÊME === *)
  baptism : cdate;                (* Date de baptême *)
```

```
  baptism_place : 'string;          (* Lieu de baptême *)
  baptism_note : 'string;           (* Notes sur le baptême *)
  baptism_src : 'string;            (* Sources baptismales *)

  (* === DÉCÈS === *)
  death : death;                    (* Statut et détails du décès *)
  death_place : 'string;            (* Lieu de décès *)
  death_note : 'string;             (* Circonstances du décès *)
  death_src : 'string;              (* Sources sur le décès *)

  (* === SÉPULTURE === *)
  burial : burial;                  (* Type: inhumation/crémation *)
  burial_place : 'string;           (* Lieu de sépulture *)
  burial_note : 'string;            (* Détails sépulture *)
  burial_src : 'string;             (* Sources sépulture *)

  (* === RELATIONS FAMILIALES === *)
  titles : 'string gen_title list;            (* Titres de noblesse *)
  rparents : ('person, 'string) gen_relation list;  (* Parents adoptifs/etc. *)
  related : 'person list;                     (* Personnes liées *)

  (* === ÉVÉNEMENTS ET DOCUMENTATION === *)
  pevents : ('person, 'string) gen_pers_event list;  (* Événements personnels *)
  notes : 'string;                  (* Notes biographiques *)
  psources : 'string;               (* Sources générales *)

  (* === TECHNIQUE === *)
  key_index : 'iper;                (* Identifiant unique *)
}
```

Types de statut

```
(* Sexe *)
type sex = Male | Female | Neuter

(* Droits d'accès *)
type access = IfTitles | Public | SemiPublic | Private

(* Statut de décès *)
type death =
  | NotDead                         (* Vivant *)
  | Death of death_reason * cdate   (* Décédé avec cause et date *)
  | DeadYoung                       (* Mort jeune *)
  | DeadDontKnowWhen                (* Décédé, date inconnue *)
  | DontKnowIfDead                  (* Statut inconnu *)
  | OfCourseDead                    (* Forcément décédé (très âgé) *)
```

```
(* Cause de décès *)
type death_reason = Killed | Murdered | Executed | Disappeared | Unspecified

(* Type de sépulture *)
type burial = UnknownBurial | Buried of cdate | Cremated of cdate
```

2. Famille (Family)

Structure représentant l'union entre deux personnes et leurs enfants :

```
type ('person, 'ifam, 'string) gen_family = {
  (* === UNION === *)
  marriage : cdate;                (* Date de mariage/union *)
  marriage_place : 'string;        (* Lieu de cérémonie *)
  marriage_note : 'string;         (* Détails de la cérémonie *)
  marriage_src : 'string;          (* Sources documentaires *)

  (* === TYPE DE RELATION === *)
  relation : relation_kind;        (* Nature de l'union *)
  divorce : divorce;               (* Statut de séparation *)
  witnesses : 'person array;       (* Témoins de l'union *)

  (* === ÉVÉNEMENTS === *)
  fevents : ('person, 'string) gen_fam_event list;  (* Événements familiaux *)

  (* === DOCUMENTATION === *)
  comment : 'string;               (* Commentaires généraux *)
  origin_file : 'string;           (* Fichier source .gw *)
  fsources : 'string;              (* Sources familiales *)

  (* === TECHNIQUE === *)
  fam_index : 'ifam;               (* Identifiant unique *)
}
```

Types d'union

```
(* Nature de la relation *)
type relation_kind =
  | Married                        (* Mariage civil/religieux *)
  | NotMarried                     (* Union libre *)
  | Engaged                        (* Fiançailles *)
  | NoSexesCheckNotMarried         (* Union sans vérification sexes *)
  | NoMention                      (* Relation non précisée *)
  | NoSexesCheckMarried            (* Mariage sans vérification sexes *)
  | MarriageBann                   (* Publication des bans *)
  | MarriageContract               (* Contrat de mariage *)
```

```
  | MarriageLicense                (* Licence de mariage *)
  | Pacs                          (* PACS *)
  | Residence                     (* Cohabitation *)

(* Statut de divorce *)
type divorce =
  | NotDivorced                   (* Pas divorcés *)
  | Divorced of cdate             (* Divorcés avec date *)
  | Separated_old                 (* Séparés (ancien format) *)
  | NotSeparated                  (* Pas séparés *)
  | Separated of cdate            (* Séparés avec date *)
```

3. Relations Familiales

Structure d'ascendance

```
type 'family gen_ascend = {
  parents : 'family option;       (* Famille des parents *)
  consang : Adef.fix;            (* Taux de consanguinité *)
}
```

Structure de descendance

```
type 'person gen_descend = {
  children : 'person array;       (* Liste ordonnée des enfants *)
}
```

Structure d'unions

```
type 'family gen_union = {
  family : 'family array;        (* Familles où la personne est parent *)
}
```

Structure de couple

```
type 'person gen_couple = {
  father : 'person;               (* Père/parent 1 *)
  mother : 'person;               (* Mère/parent 2 *)
}
```

Types de données auxiliaires

1. Système de dates

Geneweb gère plusieurs calendriers et formats de dates :

```
(* Date principale *)
type date =
  | Dgreg of dmy * calendar       (* Date dans un calendrier spécifique *)
  | Dtext of string               (* Date en format texte libre *)

(* Calendriers supportés *)
type calendar =
  | Dgregorian                    (* Calendrier grégorien *)
  | Djulian                       (* Calendrier julien *)
  | Dfrench                       (* Calendrier républicain français *)
  | Dhebrew                       (* Calendrier hébraïque *)

(* Date complète *)
type dmy = {
  day : int;                      (* Jour (1-31) *)
  month : int;                    (* Mois (1-12) *)
  year : int;                     (* Année *)
  prec : precision;               (* Précision de la date *)
  delta : int;                    (* Delta en jours *)
}

(* Précision de la date *)
type precision =
  | Sure                          (* Date certaine *)
  | About                         (* Date approximative *)
  | Maybe                         (* Date incertaine *)
  | Before                        (* Avant cette date *)
  | After                         (* Après cette date *)
  | OrYear of dmy2                (* Ou bien cette année *)
  | YearInt of dmy2               (* Intervalle d'années *)

(* Date compressée (optimisée) *)
type cdate =
  | Cgregorian of int             (* Date grégorienne compressée *)
  | Cjulian of int                (* Date julienne compressée *)
  | Cfrench of int                (* Date républicaine compressée *)
  | Chebrew of int                (* Date hébraïque compressée *)
  | Ctext of string               (* Date textuelle *)
  | Cdate of date                 (* Date complète *)
  | Cnone                         (* Pas de date *)
```

2. Événements

Événements personnels

```
type ('person, 'string) gen_pers_event = {
  epers_name : 'string gen_pers_event_name;      (* Type d'événement *)
  epers_date : cdate;                            (* Date de l'événement *)
  epers_place : 'string;                         (* Lieu *)
  epers_reason : 'string;                        (* Raison/contexte *)
  epers_note : 'string;                          (* Notes détaillées *)
  epers_src : 'string;                           (* Sources *)
  epers_witnesses : ('person * witness_kind) array;  (* Témoins *)
}

(* Types d'événements personnels *)
type 'string gen_pers_event_name =
  | Epers_Birth                    (* Naissance *)
  | Epers_Baptism                  (* Baptême *)
  | Epers_Death                    (* Décès *)
  | Epers_Burial                   (* Inhumation *)
  | Epers_Cremation                (* Crémation *)
  | Epers_Accomplishment           (* Accomplissement *)
  | Epers_Acquisition              (* Acquisition *)
  | Epers_Education                (* Éducation *)
  | Epers_Occupation               (* Profession *)
  | Epers_MilitaryService          (* Service militaire *)
  | Epers_Immigration              (* Immigration *)
  | Epers_Emigration               (* Émigration *)
  | Epers_Naturalisation           (* Naturalisation *)
  | Epers_Will                     (* Testament *)
  (* ... plus de 40 autres types *)
  | Epers_Name of 'string          (* Événement personnalisé *)
```

Événements familiaux

```
type ('person, 'string) gen_fam_event = {
  efam_name : 'string gen_fam_event_name;       (* Type d'événement *)
  efam_date : cdate;                            (* Date *)
  efam_place : 'string;                         (* Lieu *)
  efam_reason : 'string;                        (* Raison *)
  efam_note : 'string;                          (* Notes *)
  efam_src : 'string;                           (* Sources *)
  efam_witnesses : ('person * witness_kind) array;  (* Témoins *)
}

(* Types d'événements familiaux *)
type 'string gen_fam_event_name =
  | Efam_Marriage                  (* Mariage *)
  | Efam_NoMarriage                (* Pas de mariage *)
  | Efam_Engage                    (* Fiançailles *)
```

```
  | Efam_Divorce                  (* Divorce *)
  | Efam_Separated               (* Séparation *)
  | Efam_PACS                    (* PACS *)
  | Efam_Name of 'string         (* Événement personnalisé *)
```

Types de témoins

```
type witness_kind =
  | Witness                      (* Témoin général *)
  | Witness_GodParent            (* Parrain/marraine *)
  | Witness_CivilOfficer         (* Officier d'état civil *)
  | Witness_ReligiousOfficer     (* Officier religieux *)
  | Witness_Informant            (* Informateur *)
  | Witness_Attending            (* Présent à la cérémonie *)
  | Witness_Mentioned            (* Mentionné *)
  | Witness_Other                (* Autre type *)
```

3. Titres de noblesse

```
type 'string gen_title = {
  t_name : 'string gen_title_name;     (* Nom du titre *)
  t_ident : 'string;                   (* Identifiant spécifique *)
  t_place : 'string;                   (* Territoire/lieu *)
  t_date_start : cdate;                (* Date d'obtention *)
  t_date_end : cdate;                  (* Date de fin *)
  t_nth : int;                         (* Numéro (1er, 2ème, etc.) *)
}

type 'string gen_title_name =
  | Tmain                        (* Titre principal *)
  | Tname of 'string             (* Titre nommé *)
  | Tnone                        (* Pas de titre *)
```

4. Relations non-biologiques

```
type ('person, 'string) gen_relation = {
  r_type : relation_type;        (* Type de relation *)
  r_fath : 'person option;       (* Père de relation *)
  r_moth : 'person option;       (* Mère de relation *)
  r_sources : 'string;           (* Sources *)
}

type relation_type =
  | Adoption                     (* Adoption *)
  | Recognition                  (* Reconnaissance *)
  | CandidateParent             (* Parent candidat *)
```

```
  | GodParent                    (* Parrain/marraine *)
  | FosterParent                 (* Famille d'accueil *)
```

5. Lieux

```
type place = {
  other : string;                (* Autres précisions *)
  town : string;                 (* Ville *)
  township : string;             (* Commune *)
  canton : string;               (* Canton *)
  district : string;             (* District *)
  county : string;               (* Département/Comté *)
  region : string;               (* Région *)
  country : string;              (* Pays *)
}
```

Architecture de stockage

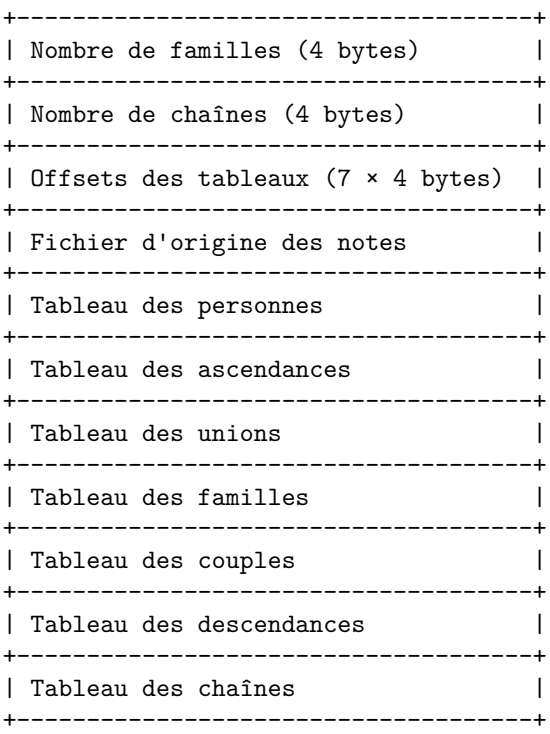
1. Structure des fichiers de base

Une base Geneweb (.gwb) est un dossier contenant plusieurs fichiers optimisés :

```
base.gwb/
   base                    # Fichier principal (données binaires)
   base.acc                # Accès directs aux tableaux
   strings.inx             # Index des chaînes de caractères
   names.inx               # Index des noms (mixte)
   names.acc               # Accès directs aux noms
   snames.inx              # Index des noms de famille
   snames.dat              # Données des noms de famille
   fnames.inx              # Index des prénoms
   fnames.dat              # Données des prénoms
   notes                   # Notes de la base
   notes_d/                # Répertoire des pages étendues
   particles.txt           # Particules de noms autorisées
   patches                 # Fichier des modifications
   nb_persons              # Nombre total de personnes
   synchro_patches         # Historique des modifications
```

2. Format du fichier principal (base)

```
Fichier "base" :
+-------------------------------------+
| Magic number (8 bytes)              |
+-------------------------------------+
| Nombre de personnes (4 bytes)       |
```



3. Types de stockage optimisés

```
(* Types utilisés sur disque (indices entiers) *)
type dsk_person = (int, int, int) Def.gen_person
type dsk_family = (int, int, int) Def.gen_family
type dsk_ascend = int Def.gen_ascend
type dsk_union = int Def.gen_union
type dsk_couple = int Def.gen_couple
type dsk_descend = int Def.gen_descend
type dsk_title = int Def.gen_title
```

4. Gestion des accès

```
type 'a record_access = {
  load_array : unit -> unit;          (* Charger en mémoire *)
  get : int -> 'a;                    (* Récupérer élément *)
  get_nopending : int -> 'a;          (* Sans patches en attente *)
  mutable len : int;                  (* Taille du tableau *)
  output_array : out_channel -> unit; (* Écrire sur disque *)
  clear_array : unit -> unit;         (* Vider le cache *)
}
```

```
type base_data = {
  persons : dsk_person record_access;      (* Accès aux personnes *)
  ascends : dsk_ascend record_access;      (* Accès aux ascendances *)
  unions : dsk_union record_access;        (* Accès aux unions *)
  families : dsk_family record_access;     (* Accès aux familles *)
  couples : dsk_couple record_access;      (* Accès aux couples *)
  descends : dsk_descend record_access;    (* Accès aux descendances *)
  strings : string record_access;         (* Accès aux chaînes *)

  (* Métadonnées *)
  particles_txt : string list;             (* Particules de noms *)
  particles : Re.re Lazy.t;                (* Expression régulière *)
  bnotes : Def.base_notes;                 (* Notes de base *)
  bdir : string;                           (* Répertoire de base *)
  perm : perm;                             (* Permissions *)
}
```

Système d'indexation

1. Index des noms (names.inx)

Le système d'indexation permet des recherches rapides :

```
Index des noms mixtes :
+-------------------------------------+
| Offset vers sindex                  |
+-------------------------------------+
| Offset vers findex                  |
+-------------------------------------+
| Index 1 : Hash -> Personnes         |
| (table_size entrées)                |
+-------------------------------------+
| Index 2 : Hash sous-chaînes noms    |
| -> Noms complets                    |
+-------------------------------------+
| Index 3 : Hash sous-chaînes prénoms |
| -> Prénoms complets                 |
+-------------------------------------+
```

2. Index des chaînes (strings.inx)

```
Index des chaînes :
+-------------------------------------+
| Taille du tableau d'offsets         |
+-------------------------------------+
```

```
| Table de hachage :                  |
|   - Tableau des offsets             |
|   - Tableau des listes chaînées     |
+-------------------------------------+
```

3. Index spécialisés

```
type string_person_index = {
  find : int -> int list;       (* Trouver personnes par nom *)
  cursor : string -> int;       (* Position dans l'index *)
  next : int -> int;           (* Élément suivant *)
}
```

4. Fonctions de recherche

```
type base_func = {
  (* Recherche par clé *)
  person_of_key : string -> string -> int -> int option;

  (* Recherche par nom *)
  persons_of_name : string -> int list;
  strings_of_sname : string -> int list;
  strings_of_fname : string -> int list;

  (* Index optimisés *)
  persons_of_surname : string_person_index;
  persons_of_first_name : string_person_index;

  (* Fonctions de modification *)
  patch_person : int -> dsk_person -> unit;
  patch_family : int -> dsk_family -> unit;
  insert_string : string -> int;
  commit_patches : unit -> unit;

  (* Utilitaires *)
  nb_of_real_persons : unit -> int;
  iper_exists : int -> bool;
  ifam_exists : int -> bool;
}
```

Système de versioning

1. Gestion des modifications

Geneweb utilise un système de patches pour éviter la réécriture complète :

```
type ('iper, 'person, 'family, 'string) base_changed =
  (* Personnes *)
  | U_Add_person of ('iper, 'person, 'string) gen_person
  | U_Modify_person of (...) * (...)      (* Ancien * Nouveau *)
  | U_Delete_person of (...)
  | U_Merge_person of (...) * (...) * (...) (* P1 * P2 * Résultat *)

  (* Familles *)
  | U_Add_family of (...) * (...)
  | U_Modify_family of (...) * (...) * (...)
  | U_Delete_family of (...) * (...)
  | U_Merge_family of (...) * (...) * (...) * (...)

  (* Opérations spéciales *)
  | U_Send_image of (...)              (* Envoi d'image *)
  | U_Delete_image of (...)            (* Suppression d'image *)
  | U_Change_children_name of (...)    (* Changement noms enfants *)
  | U_Invert_family of (...) * (...)   (* Inversion famille *)
  | U_Kill_ancestors of (...)          (* Suppression ancêtres *)

  (* Modifications groupées *)
  | U_Multi of (...) * (...) * bool    (* Modifications multiples *)
  | U_Notes of int option * string     (* Modifications notes *)
```

2. Historique synchronisé

```
type synchro_patch = {
  mutable synch_list : (string * int list * int list) list;
}
(* timestamp * personnes_modifiées * familles_modifiées *)
```

3. Versions de base

```
type base_version =
  | GnWb0020
  | GnWb0021
  | GnWb0022
  | GnWb0023
  | GnWb0024    (* Version actuelle *)
```

Points clés de l'architecture

1. Optimisations de performance

- **Index multiples** : Recherche O(1) par hash sur noms/prénoms

- **Compression des dates** : Format binaire compact
- **Cache intelligent** : Chargement à la demande
- **Identifiants entiers** : Économie mémoire vs pointeurs

2. Scalabilité

- **Système de patches** : Modifications incrémentales
- **Accès direct** : Pas de chargement complet obligatoire

- **Index de sous-chaînes** : Recherche partielle efficace
- **Parallélisation** : Accès concurrent en lecture

3. Robustesse

- **Versioning de base** : Compatibilité ascendante
- **Validation des données** : Détection d'erreurs généalogiques
- **Sauvegarde incrémentale** : Historique des modifications
- **Gestion d'erreurs** : Recovery automatique

4. Flexibilité

- **Multiples calendriers** : Grégorien, julien, républicain, hébraïque
- **Événements extensibles** : Types personnalisés
- **Droits d'accès granulaires** : Par personne
- **Internationalisation** : Support UTF-8, particules

5. Standards généalogiques

- **GEDCOM compatible** : Import/export standard
- **Numérotation Sosa** : Navigation ancestrale
- **Relations complexes** : Adoption, parrainage, etc.
- **Sources documentaires** : Traçabilité complète

6. Architecture modulaire

```
geneweb/
  lib/def/          # Définitions de types
  lib/db/           # Couche base de données
  lib/util/         # Utilitaires (dates, noms)
  lib/core/         # Algorithmes centraux
  bin/              # Outils ligne de commande
  rpc/              # API RPC
  plugins/          # Extensions
```

Conclusion

L'architecture de Geneweb représente un excellent exemple d'optimisation pour les applications généalogiques. Elle combine :

- **Performance** : Index multiples et cache intelligent
- **Évolutivité** : Système de patches et versioning
- **Robustesse** : Validation et recovery
- **Standards** : Compatibilité GEDCOM et pratiques généalogiques

Cette conception permet de gérer efficacement des bases de données généalogiques importantes (>100k personnes) avec des temps de réponse optimaux pour les recherches et modifications.

Document généré le 10 septembre 2025
Basé sur l'analyse du code source Geneweb (Master branch)