Defteraiou Maria, Konstantinidis Konstantinos, Mertzani Styliani, Voulgari Despoina

Supplementary Material: Designing process

Brain Proteome Database Architecture

The database is designed to store and manage proteomic data from different brain regions of *Mus musculus* (mouse). It integrates protein identification, Gene Ontology (GO) classifications, and brain region-specific protein expression information.

Tables and Architecture

1. `main` (Primary Protein Reference Table)

This table contains the unique protein identifiers and their corresponding names. It serves as the central reference table, linking proteins to their expression data and GO annotations.

- **`uniprot_id` (TEXT, PRIMARY KEY)** – Unique UniProt identifier for each protein.
- **`protein_name` (TEXT)** – Descriptive name of the protein.

2. `brain_part` (Brain Region-Specific Protein Data)

This table stores quantitative proteomics data for different mouse brain regions. The brain regions are predefined as an ENUM type (`bp`).

- **`uniprot_id` (TEXT, FOREIGN KEY → `main.uniprot_id`)** – Unique UniProt identifier.
- **`brain_part` (bp, ENUM)** – The brain region where the protein is detected. The possible values are:
  - Cerebellum
  - Cerebral Cortex
  - Hippocampus
  - Hypothalamus
  - Mid Brain
  - Medulla
  - Olfactory Bulb

- **`score` (FLOAT)** – Identification score for the protein in the region.
- **`coverage` (FLOAT)** – Percentage of the protein sequence covered in the analysis.
- **`proteins` (INT)** – Number of protein groups detected.
- **`unique_peptides` (INT)** – Count of unique peptides identified.
- **`peptides` (INT)** – Total peptide count.

- **psms (INT)** – Peptide-spectrum matches.
- **area (FLOAT)** – Quantification area under the curve.
- **aas (INT)** – Number of amino acids in the protein sequence.
- **mw (FLOAT)** – Molecular weight (kDa).
- **pi (FLOAT)** – Isoelectric point.

Foreign Key Constraint:

- Deleting a protein entry from `main` will automatically remove its associated records in `brain_part` (`ON DELETE CASCADE`).

3. `go_terms_detailed` (Detailed Gene Ontology Annotations)

This table provides a detailed classification of proteins based on Gene Ontology (GO) terms. The GO annotations are stored as arrays.

- **uniprot_id (TEXT, FOREIGN KEY → main.uniprot_id)** – Unique UniProt identifier.
- **biological_process (TEXT[])** – List of GO terms related to the biological processes the protein is involved in.
- **molecular_function (TEXT[])** – List of GO terms describing the molecular functions of the protein.
- **cellular_component (TEXT[])** – List of GO terms specifying the subcellular localization of the protein.
- **ptm (TEXT[])** – Post-translational modifications (PTMs) associated with the protein.

Foreign Key Constraint:

- Deleting a protein entry from `main` removes its corresponding GO annotations.

4. `go_terms_general` (Generalized GO Annotations)

This table provides a summarized view of the protein's functional classifications by storing only the higher-level GO categories.

- **uniprot_id (TEXT, FOREIGN KEY → main.uniprot_id)** – Unique UniProt identifier.
- **biological_process (TEXT[])** – Generalized biological process GO terms.
- **molecular_function (TEXT[])** – Generalized molecular function GO terms.

Foreign Key Constraint:

- Deleting a protein entry from `main` removes its corresponding general GO annotations.

Key Features of the Database

- **Relational Structure:** The `main` table acts as the primary reference point, linking proteins to their brain region-specific data and functional annotations.
- **Brain Region-Specific Analysis:** The `brain_part` table enables comparisons of protein expression across different brain regions.
- **Gene Ontology Integration:** GO annotations are categorized into detailed (`go_terms_detailed`) and general (`go_terms_general`) levels for flexible querying.
- **Data Integrity:** Foreign key constraints ensure consistency—removing a protein from `main` automatically removes its associated entries across other tables.
- **Efficient Querying:** Storing GO terms as arrays allows for efficient searches using PostgreSQL’s array functions.