Biological Database Design

Week 2

Spring ’06

Melanie Nelson, Ph.D.

Final Project: Reminder!

- Design a database to store biological data. The database must integrate at least two sources of data.
- Can work alone or in teams of up to three members
- Week 4: Hand in plan
  - What type of biological data will be stored
  - Scope statement (what aspects of the data are to be covered by your database)
- Week 6: Hand in and present design
  - Requirements document
  - ER or UML diagram
  - Short (1-2 page) report describing any difficult or unusual design decisions
  - Make 10-15 minute presentation about DB to class

Database Design Process

- Process = steps to follow
- Increases chances of project success
  - Encourages thinking about entire project before developing (less likely to get a patchwork data model)
  - Find problems early, when its easier (and cheaper!) to fix them
- Process doesn’t have to be onerous. Tailor to the needs of your team.

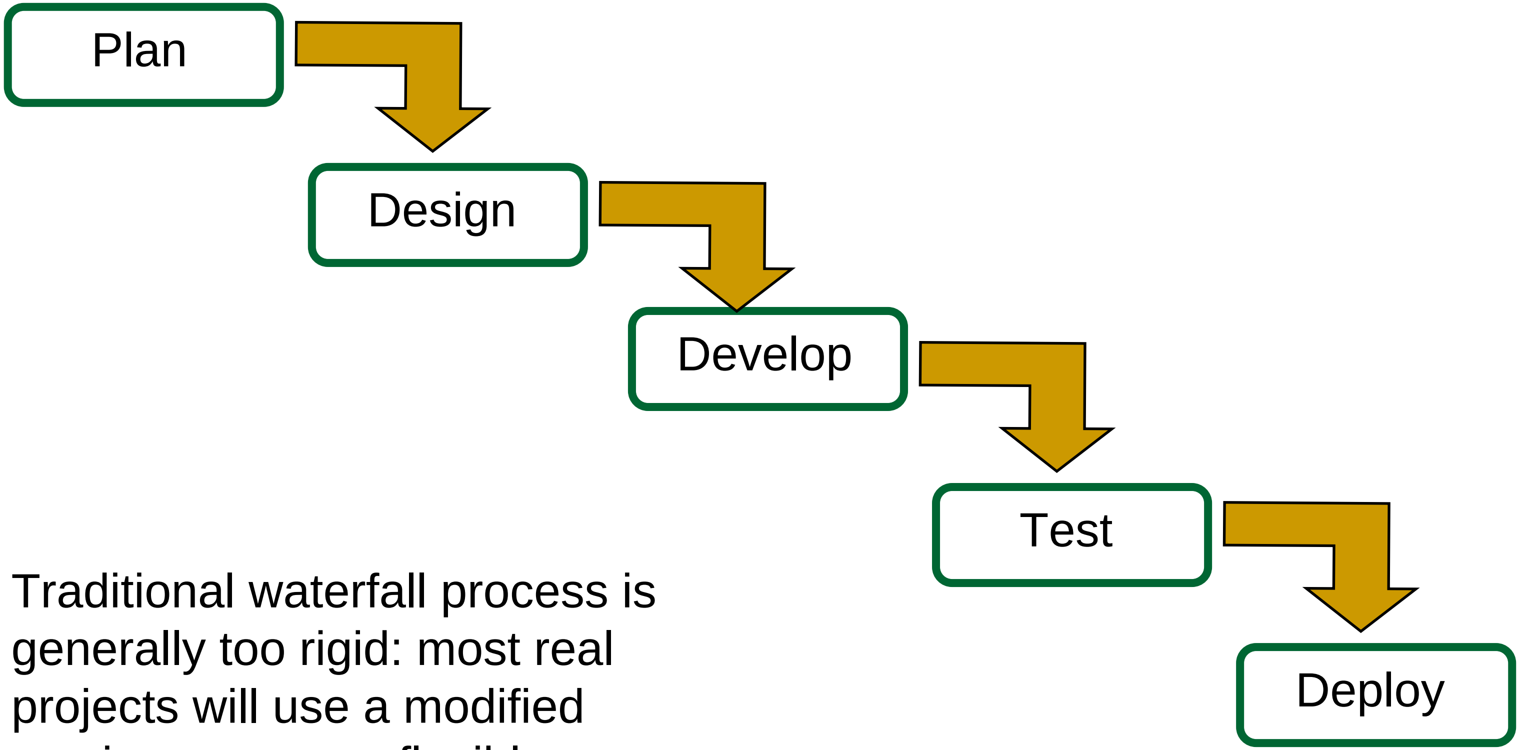
Attributes of a Good Process

- Involves all “stakeholders” (people who have a stake in outcome of project)
- Documents requirements
- Produces a well-documented database
- Tests that the requirements have been met

Parts of a Standard Process

- Plan
    - Gather and document requirements
    - Develop a project plan (how long will it take? Who will be involved?)
- Design
    - Design DB and applications to access it
- Develop
    - Create database, code applications
- Test
    - Does system meet all requirements?
- Deploy

“Waterfall” Process



Plan

Design

Develop

Test

Deploy

Traditional waterfall process is generally too rigid: most real projects will use a modified version or a more flexible process

Variations

- Prototyping
  - Often used to help explore requirements and design options
  - Shouldn't allow prototype to morph into final app
- Agile programming
  - "Extreme programming" is one version of this
  - Eliminates formal design: design is part of development
  - An attempt to mitigate risk of changing requirements
  - Difficult to do with databases
  - Biological databases may be particularly inappropriate for this technique

Step 1: Plan

- Gather database requirements
  - What is the scope of the database?
  - What data will be stored?
  - What relationships among the data must be captured?
  - What questions will need to be asked of the data?
  - How quickly do the answers need to be generated?
  - When must the final system be ready?

Attributes of Good Requirements

- Requirements should be testable
  - Need to be able to certify that the final system meets them
- Requirements come from the users and the data
  - Ask users what they need
  - Document what the data requires
- Developers must also agree to requirements
  - Agree only to what is feasible!
- Some requirements may contradict each other
  - Return to the users to get priorities

Requirements

- Ways to determine requirements:
  - User interviews
  - Prototyping
- Unspoken tool: developers' experience
  - Users do not always know what is possible, and they self-edit in interviews
  - Difficulty in determining complexity level
    - Scientists may over-simplify complex relationships when explaining to a non-scientist (teacher mode)
    - Some complexities may fall outside scope of database

Step 2: Design

- Developers determine how to meet the requirements
- Logical data model is developed
- Physical database design is developed
- Usually requires returning to the users for:
  - Clarification of requirements
  - Understanding data for the data model
- Strong temptation to short-change this step and rush to development

Logical Data Model vs. Physical Database Design

- Logical data model reflects structure of data
  - Accurately capture meaning of data
  - Accurately reflect relationships amongst the data
- Physical database design
  - How tables will be structured
  - Reflects any compromises necessary due to limitations of current database management systems

Design Tools

- Many tools to develop data model
  - ERWin
  - ER Studio
  - DBDesigner (free: fabforce.net)
- Most tools will automatically generate SQL to create database from data model
  - Physical database design may not completely mirror logical data model
  - Be sure to document logical data model as well as physical design

Step 3: Develop

- Database is created
- Initial data is imported
  - If there is no initial data, a test set should be used
- Initial version of the application is written
- Beware of “feature creep”
  - Tendency to add features after the design is complete
  - Rule of thumb: Only add if initial release will be useless without the feature. Otherwise, promise in a later release

Step 4: Test

- Test database with data that is:
  - Real
  - Representative
  - Attempts to cover “pathological” cases
- Test “incorrect” data, too
  - Database should reject
- Application is tested
  - Application code may enforce some business rules
  - Application is the “public face” of the database

Step 5: Deploy

- “Roll out” database and application
- Don’t forget training!
  - Since domain support is weak, users decide what values are actually valid
  - If users can’t make the application work, they will consider the project a failure

Data Modeling

- Model is a representation of our understanding of reality
- Data model reflects the database designer's understanding of the data to be stored
- Don't build a database without one!
  - Data model is always implicit in database design
  - Should be made explicit

Tools for Data Modeling

- Entity-relationship diagrams
  - Data is modeled as entities and relationships among entities
  - Most common in database design
- UML (Unified Modeling Language)
  - Data is modeled as classes and relationships among classes
  - More formal types of relationships
  - Common in object-oriented programming

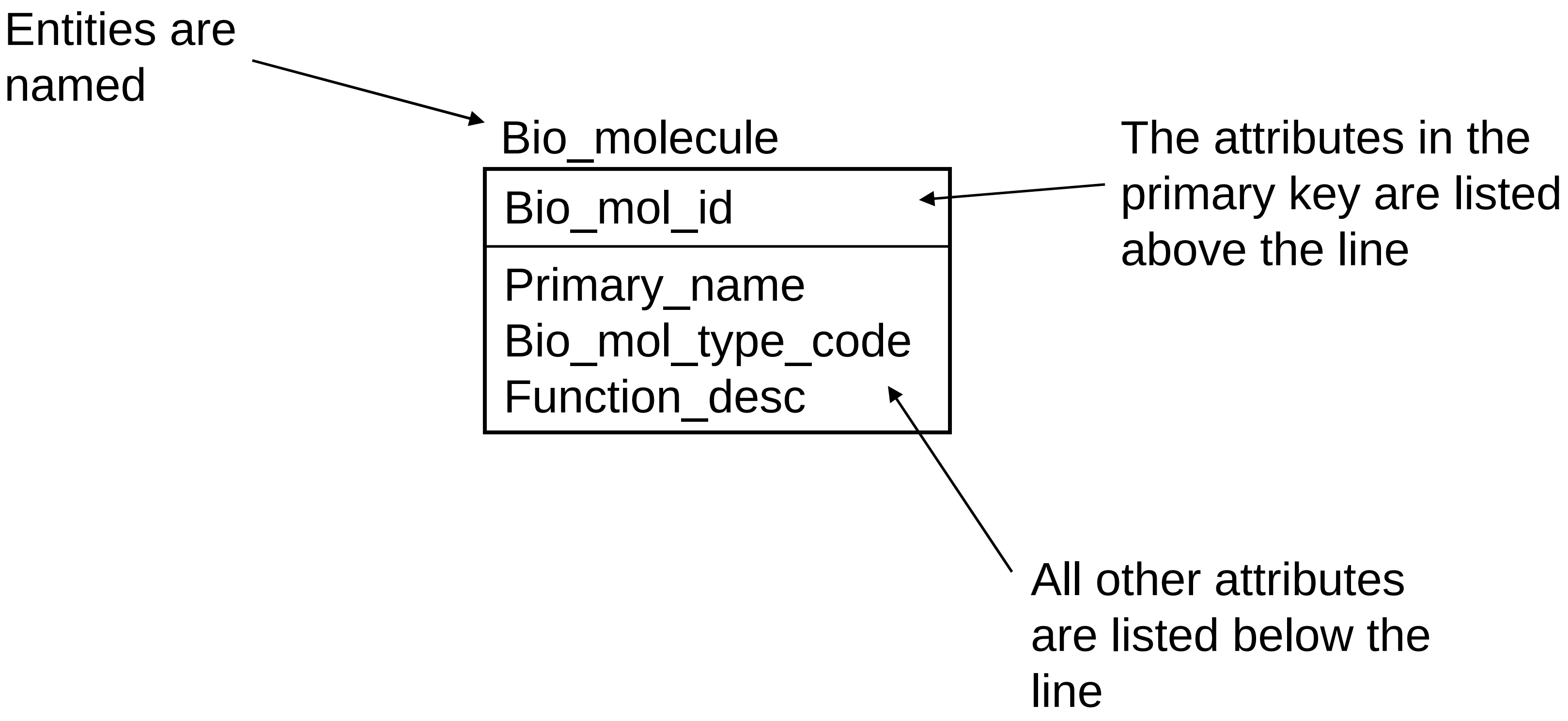
Entity-Relationship Diagrams

- There are many different types
- Differ primarily in syntax
- Pick one and be consistent
- For this class, I’ll use IDEF1X standard
- If you’re using a tool that doesn’t support IDEF1X, provide me with a mapping from IDEF1X to the syntax your tool uses

Entity-Relationship Diagrams

- Entity = a noun
  - A thing or concept about which information will be stored
- Entities have attributes
  - Information about the entity
  - Each particular instance of an entity only has one copy of each attribute
  - “capital city” is an attribute of “country”
  - “ally” is not an attribute: a country can have multiple allies

Entities and Attributes



Entities are named

Bio_molecule

Bio_mol_id

Primary_name
Bio_mol_type_code
Function_desc

The attributes in the primary key are listed above the line

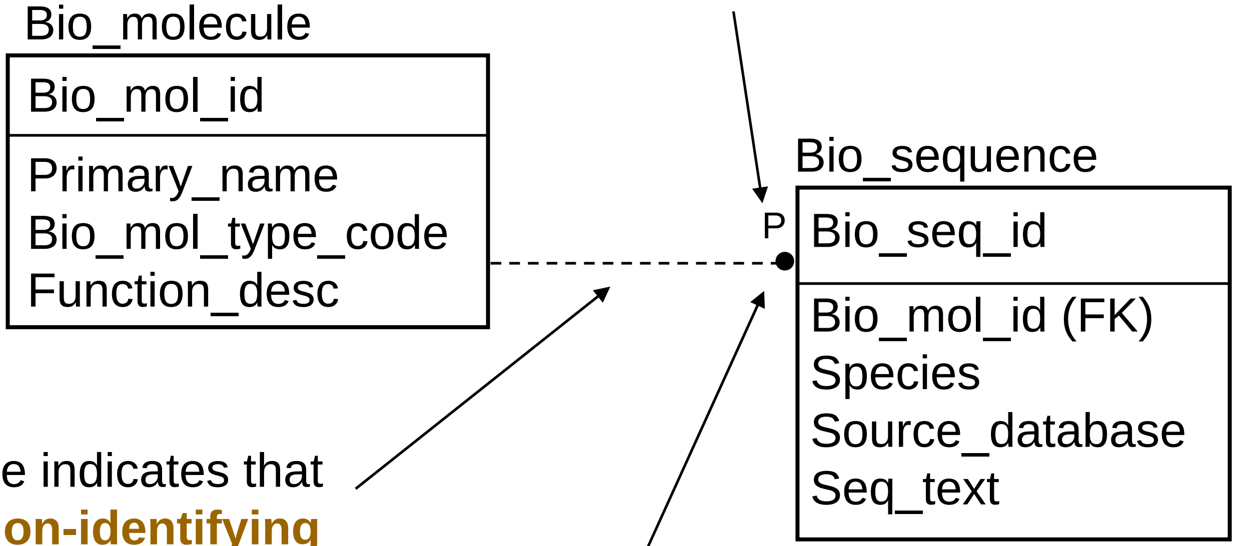
All other attributes are listed below the line

Entity-Relationship Diagrams

- “Multicopy” attributes are really other entities
- There are relationships among entities
    - Relationships are often named to indicate their meaning
    - Relationships have cardinality:
        - how many instances of one entity can reference the other entity
        - default is “zero, one, or many”

Basic Relationships

“P” indicates that there must be at least one bio_sequence per bio_molecule, but that there may be more.



Bio_molecule

Bio_mol_id

Primary_name
Bio_mol_type_code
Function_desc

Bio_sequence

P

Bio_seq_id

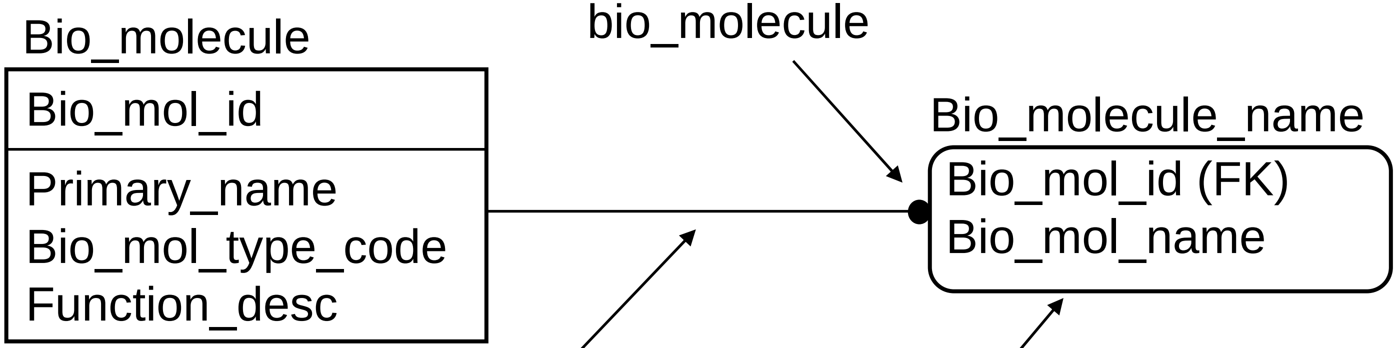
Bio_mol_id (FK)
Species
Source_database
Seq_text

Dotted line indicates that this is a **non-identifying** relationship: bio_mol_id is not part of the primary key of bio_sequence

Circle goes on the **child entity**: the entity that references the **parent entity**

Basic Relationships

Lack of number or letter on circle indicates that there can be zero, one or many bio_mol_names for each bio_molecule



Bio_molecule

Bio_mol_id

Primary_name
Bio_mol_type_code
Function_desc

Bio_molecule_name

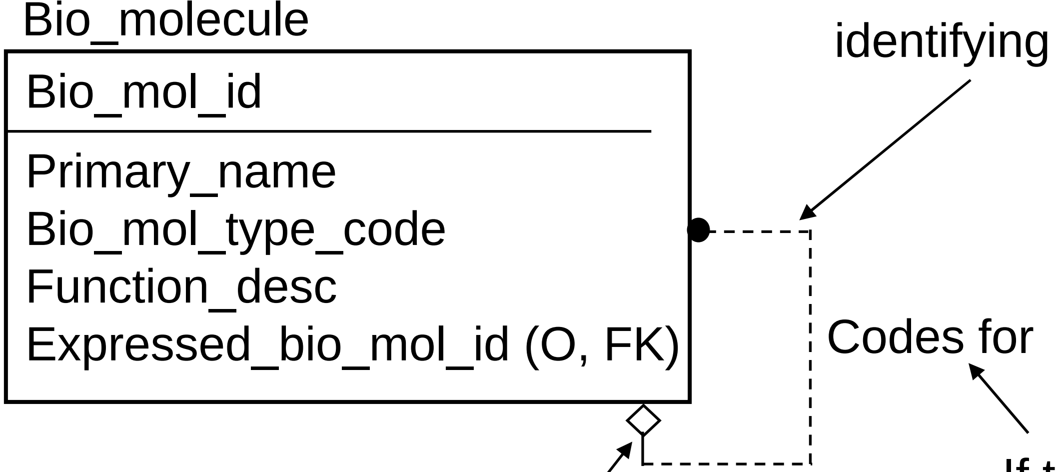
Bio_mol_id (FK)
Bio_mol_name

Solid line indicates that this is an **identifying** relationship: bio_mol_id is part of the primary key of bio_mol_name

Rounded corners on bio_mol_name indicates that it is **identifier-dependent**: it depends on bio_molecule for identification

Recursive and Optional Relationships

Relationships can be recursive. Recursive relationships must be non-identifying



Bio_molecule

Bio_mol_id

Primary_name
Bio_mol_type_code
Function_desc
Expressed_bio_mol_id (O, FK)

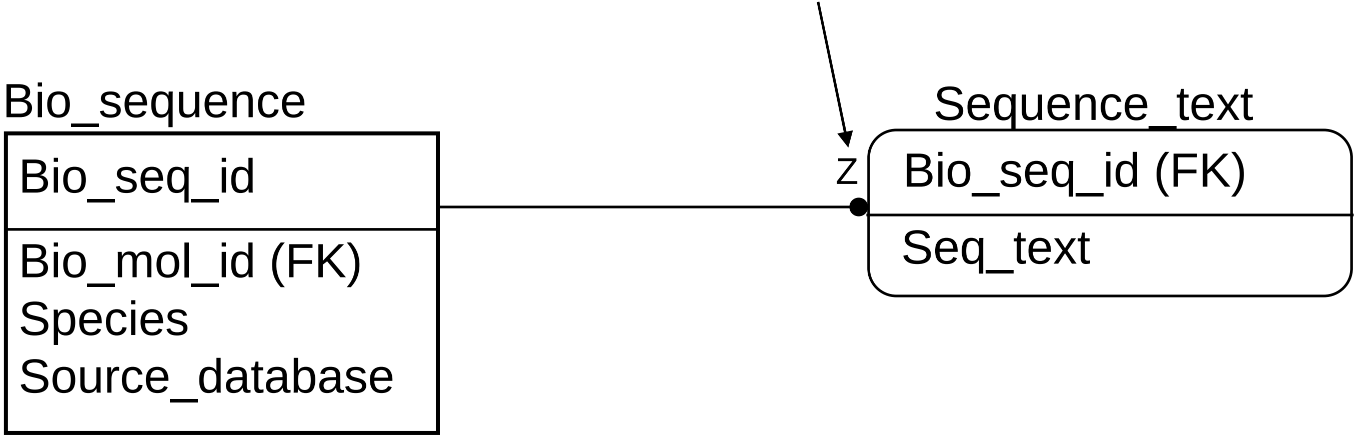
Codes for

Diamond indicates that not all child entries reference a parent entry

If the relationship can be descrbed with something more than “has”, the description is associated with the relationship line

Specifying Cardinality

“Z” indicates that there can be zero or one sequence_text for each bio_sequence



Bio_sequence

Bio_seq_id

Bio_mol_id (FK)
Species
Source_database

Z

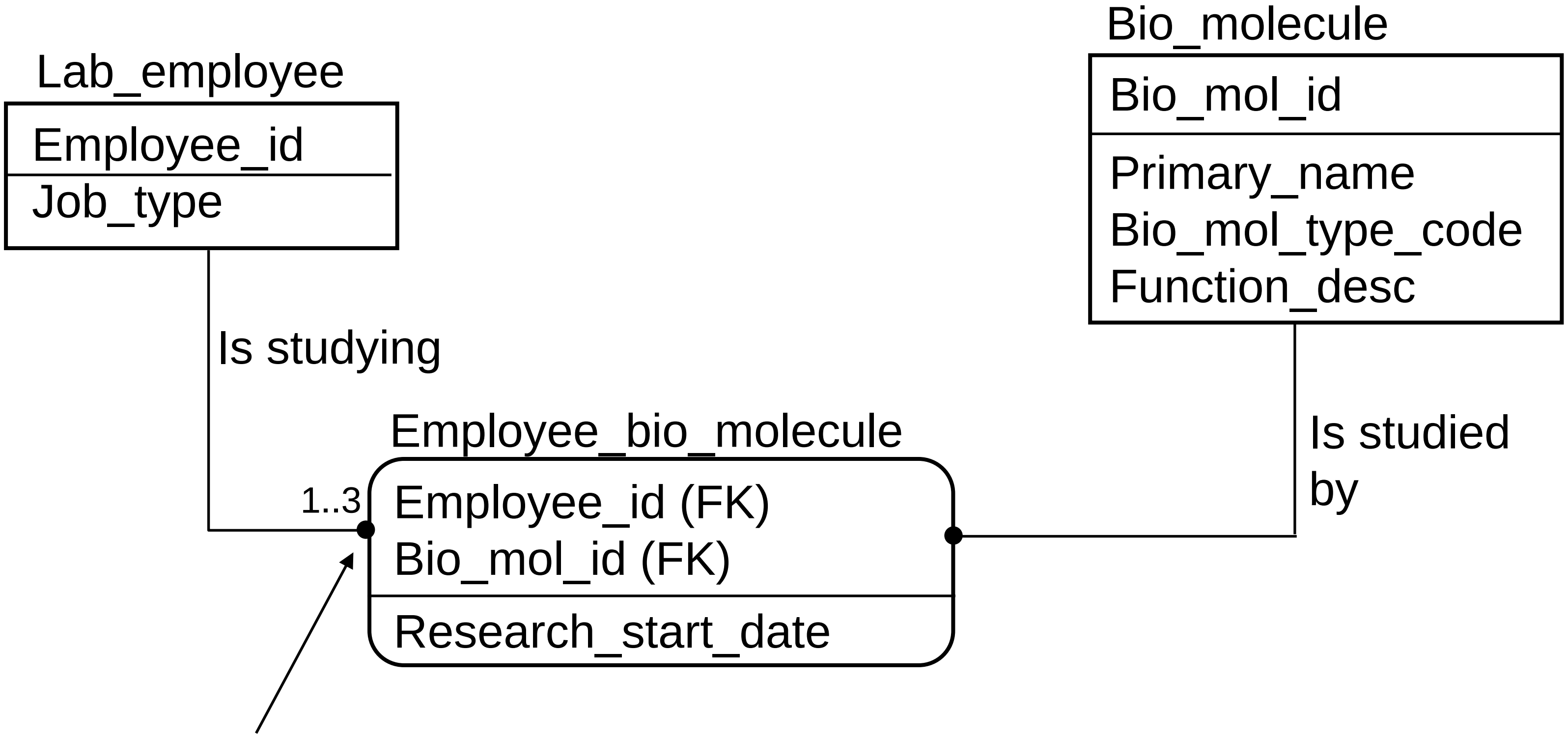
Sequence_text

Bio_seq_id (FK)

Seq_text

Used to avoid NULLs: perhaps in some cases we know that a sequence exists in a given species, but we don’t have the actual sequence text yet.

Specifying Cardinality



Lab_employee

Employee_id

Job_type

Is studying

Bio_molecule

Bio_mol_id

Primary_name
Bio_mol_type_code
Function_desc

Is studied by

Employee_bio_molecule

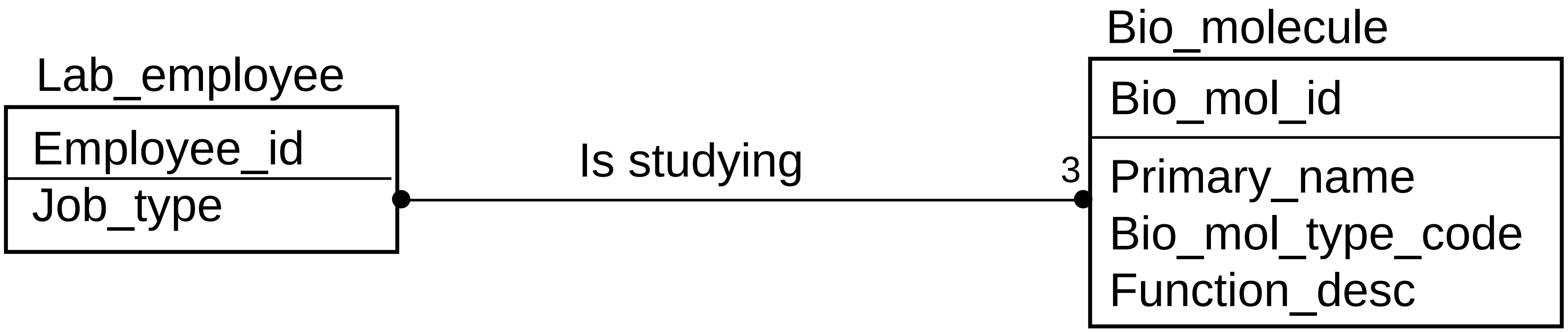
1..3

Employee_id (FK)
Bio_mol_id (FK)

Research_start_date

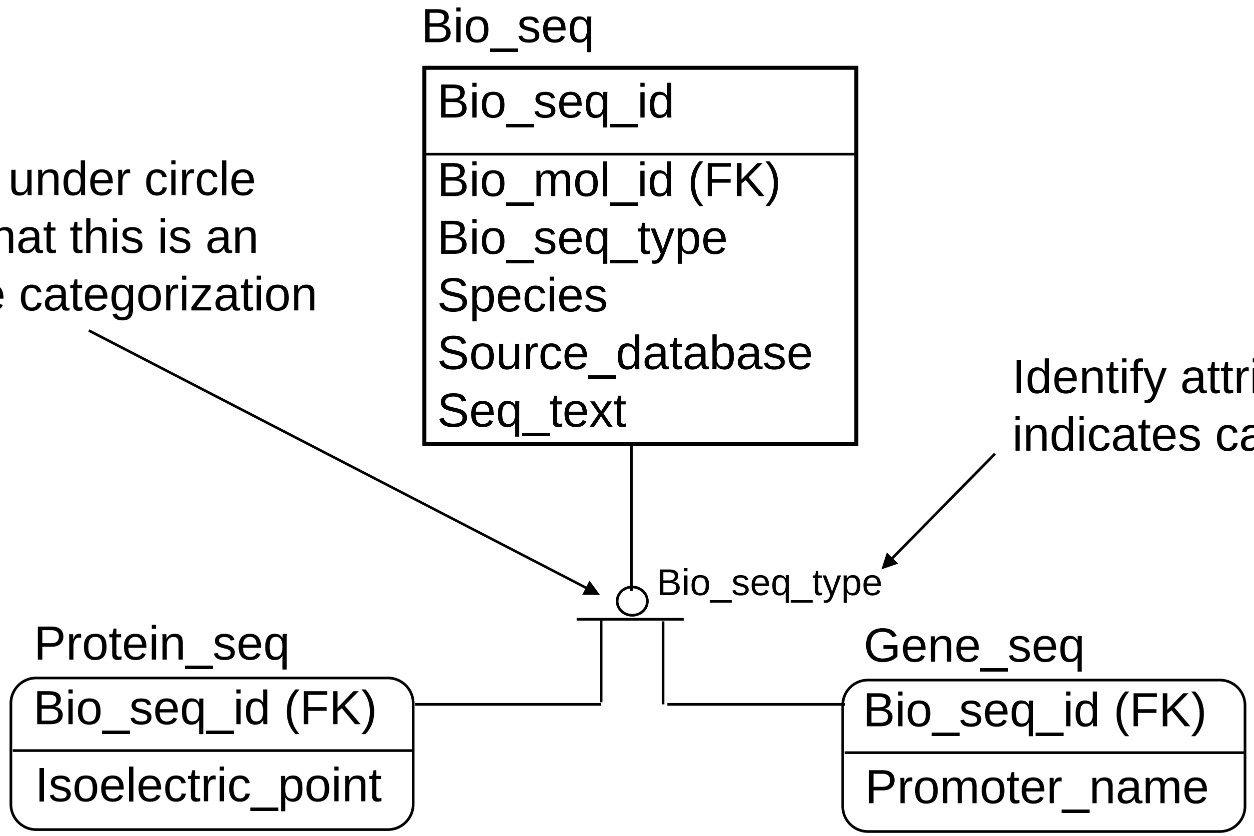
A lab employee must be studying at least one bio_molecule, but may not be studying more than three bio_molecules

Many-to-Many Relationships



Can represent the previous relationship as a many-to-many relationship

Categorization Relationships



Single line under circle indicates that this is an incomplete categorization

Identify attribute that indicates category

Bio_seq

Bio_seq_id

Bio_mol_id (FK)
Bio_seq_type
Species
Source_database
Seq_text

Bio_seq_type

Protein_seq

Bio_seq_id (FK)

Isoelectric_point

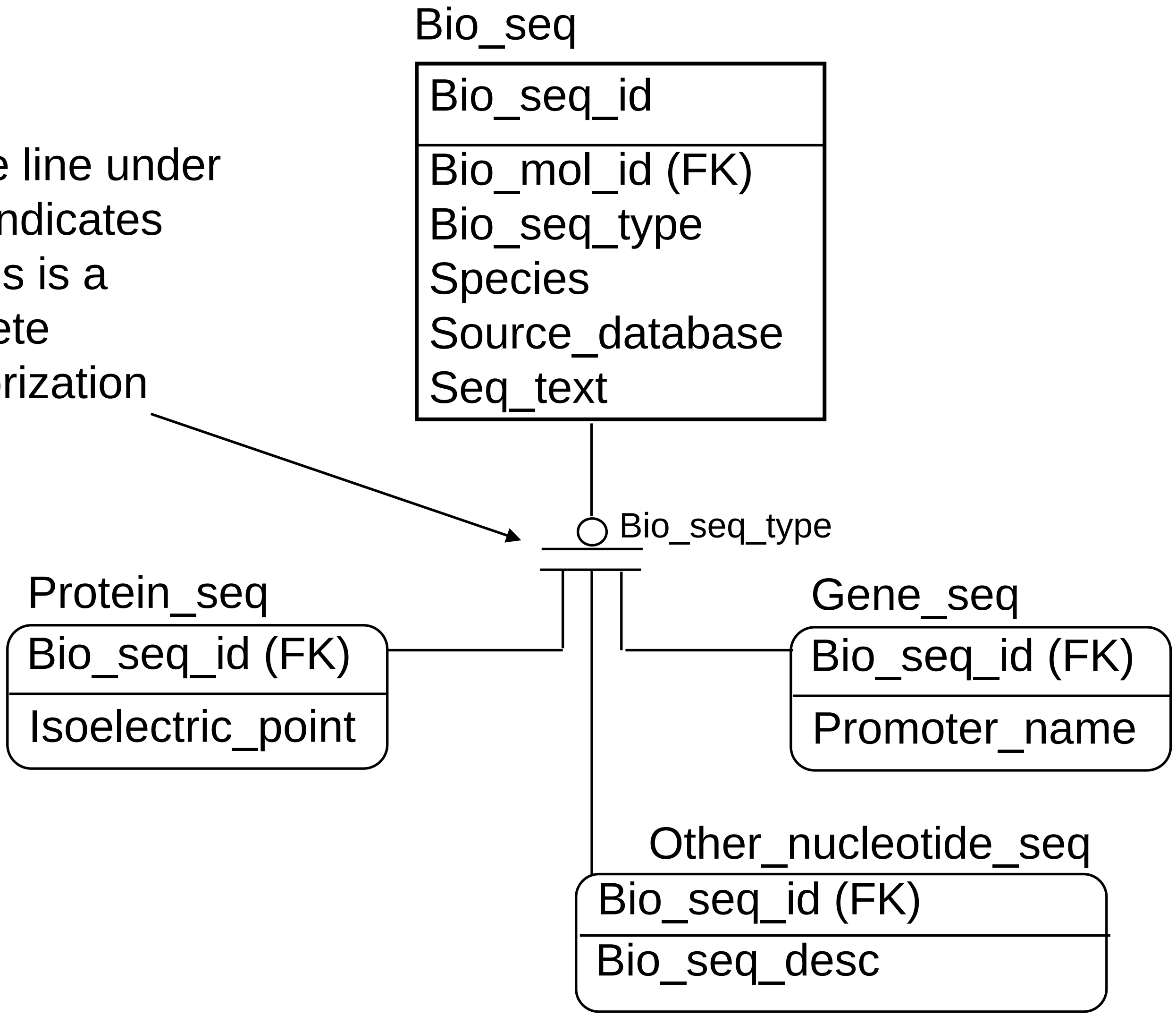
Gene_seq

Bio_seq_id (FK)

Promoter_name

Categorization Relationships

Double line under circle indicates that this is a complete categorization



Bio_seq

Bio_seq_id

Bio_mol_id (FK)
Bio_seq_type
Species
Source_database
Seq_text

Protein_seq

Bio_seq_id (FK)

Isoelectric_point

Gene_seq

Bio_seq_id (FK)

Promoter_name

Other_nucleotide_seq

Bio_seq_id (FK)

Bio_seq_desc

Data Modeling Method

- Many processes recommend:
  - List entities
  - Define relationships
  - Fill in attributes
  - Fill in datatypes
- I gather all at once
  - Iteratively develop data model
  - Focus on “higher level”, i.e., entities and basic relationships in early iterations
  - Users don’t discriminate among entities, relationships, and attributes when describing their needs
  - If you make users repeat themselves, they will lose patience with the process

Data Modeling Method

- Define scope
- Identify users of data within scope
- Develop initial ER diagram
    - Serves as framework for user interviews
    - Must be prepared to discard much, if not all, of it
- Interview users
    - Don't interview too many at once
    - Don't interview users with very distinct usages of data together (at least not initially)

Data Modeling Method

- Build/refine ER diagram
  - Iterate!
  - Sleep on it
- Data model isn't complete until you specify datatypes for each attribute
  - Datatypes are closest to domain support in most DBMS
  - Text field lengths and exact names of datatypes are DBMS dependent
  - If you know your DBMS, fill them in. Otherwise, put general type
- Document constraints that will be enforced in DB
  - Business rules that are enforced by triggers
  - Put in the data dictionary or in an appendix to ER diagram

Data Modeling Method

- Once you have a “full” model, test it with some sample data
  - Can be done on paper, or by building DB and populating
  - Look for the “pathological” examples
- Will almost certainly need to iterate through the last three steps many times:
  - Interview users
  - Refine data model
  - Test data model

Rules for Better Design

- Use naming conventions
- Keep a data dictionary
- Don’t be afraid to discard parts (or all) of the model
    - Don’t iterate to “patchwork quilt” design
    - Catch and fix errors in design phase, when they are relatively cheap and easy to fix

Naming Conventions

- Naming conventions
  - Make the data model more readable
  - Simplify database queries
  - Allow automated maintenance
- A real world “thing” should always be represented by the same name or abbreviation
  - Sometimes, may use full word for table names and abbreviations in columns
  - Whatever you do, be consistent and document it!

Naming Conventions

- Standardize formatting
  - Separate words with underscores or hyphens: not both!
- Standardize suffixes
  - “code” for letter based code
  - “id” for numerical identifier

Data Dictionaries

- Data dictionaries
  - Define terms used in a database
  - Document what the data means
    - “related_protein”
    - Protein related by sequence identity, structural similarity, functional similarity, or any of the above?
  - Makes it easier for programmers and users to ensure correct data is entered into database

Data Dictionaries

- Data dictionary should include all entities and attributes
- It is best practice to document relationships
- I create data dictionary for logical model, and modify it as I move to physical design
  - Use data dictionary to document places where physical design does not match logical data model

Normalization

- Normalization is a process that ensures database follows rules that protect data integrity
  - **Remove redundancy!**
- As you get more experienced in data modeling, you’ll find you normalize “by default”, without thinking about rules
- Normalization is “loss-less” and reversible (via join)

Why Normalize

- Minimize risk of data inconsistencies
  - Two copies of the same data can get “out of sync”
- Minimize update and delete anomalies
  - If you update or delete one copy, what should happen to the other copy?
- Maximize database design stability
  - Associate attributes with entities based on the meaning of the data, not on application requirements
- Minimize storage requirements
  - Storing the same data multiple times wastes disk space
  - Not as important as it used to be

First Normal Form

- There are no repeating or multivalued attributes
- “Repeat down the rows, not across the columns”

Multivalued attributes!

Protein ID	Protein Name
Protein 1	Calmodulin, CaM
Protein 3	DUSP-2, dual specificity phosphatase 2, PAC1

Protein ID	Protein Name
Protein 1	Calmodulin
Protein 1	CaM
Protein 3	DUSP-2
Protein 3	Dual specificity phosphatase 2
Protein 3	PAC1

Repeating attributes!

Protein ID	Protein Name 1	Protein Name 2	Protein Name 3
Protein 1	Calmodulin	CaM	
Protein 2	DUSP-2	Dual specificity phosphatase 2	PAC1

Second Normal Form

- Attributes depend on the entire primary key

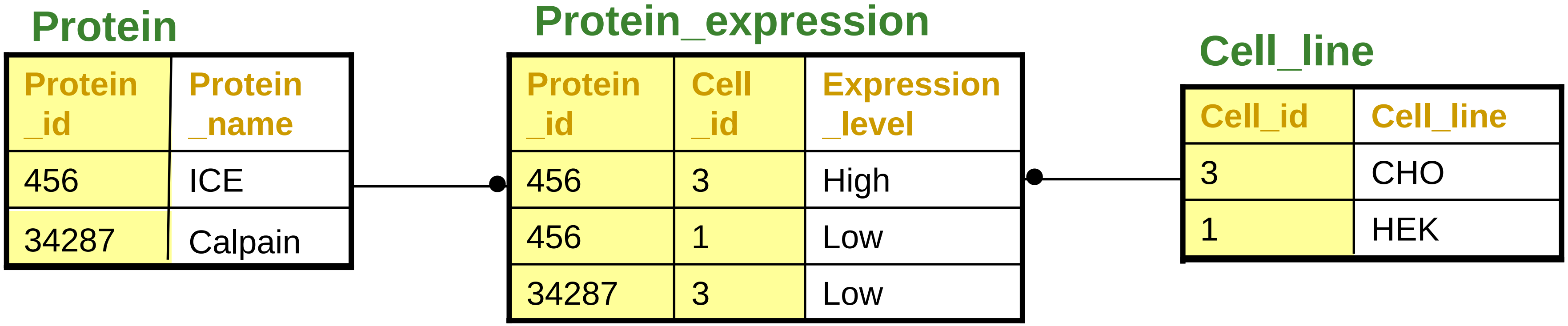
Protein_expression

Protein_id	Cell_id	~~Protein_name~~	~~Cell_line~~	Expression_level
456	3	~~ICE~~	~~CHO~~	High
456	1	~~ICE~~	~~HEK~~	Low
34287	3	~~Calpain~~	~~CHO~~	Low

•Protein_name depends on Protein_id, but not on Cell_id

•Cell_line depends on Cell_id, but not on Protein_id

•Only Expression_level depends on both parts of the key



Protein

Protein_id	Protein_name
456	ICE
34287	Calpain

Protein_expression

Protein_id	Cell_id	Expression_level
456	3	High
456	1	Low
34287	3	Low

Cell_line

Cell_id	Cell_line
3	CHO
1	HEK

Third Normal Form

- Attributes depend only on the primary key
- Except: they can depend on candidate keys, too

Protein_seq_id	Source_db	Source_db_url	Seq_text
456	Swiss-Prot	us.expasy.org/sprot	MLVEGF….
32	Swiss-Prot	us.expasy.org/sprot	MGGKGL….
142	RefSeq	www.ncbi.nlm.nih.gov/RefSeq	MAGKKG….

Source_db_url depends on source_db, not protein_seq_id

Protein_seq_id	Source_db	Seq_text
456	Swiss-Prot	MLVEGF….
32	Swiss-Prot	MGGKGL….
142	RefSeq	MAGKKG….

Source_db	Source_db_url
Swiss-Prot	us.expasy.org/sprot
RefSeq	www.ncbi.nlm.nih.gov/RefSeq

Boyce-Codd Normal Form

- All attributes depend on each full candidate key, and not on a subset of any candidate key
- Particularly important to consider if using automatic numeric IDs

Seq_ id	Protein _id	Source_ db	Source_db_url	Seq_text
1	456	Swiss-Prot	us.expasy.org/sprot	MLVEGF….
2	32	Swiss-Prot	us.expasy.org/sprot	MGGKGL….
3	456	RefSeq	www.ncbi.nlm.nih.gov/ RefSeq	LVEGF….

•Protein_id and Source_db together identify each row

•Source_db_url still depends on Source_db, but not on Protein_id

Seq_ id	Protein_ id	Source_db	Seq_text
1	456	Swiss-Prot	MLVEGF….
2	32	Swiss-Prot	MGGKGL….
3	456	RefSeq	LVEGF….

Source_db	Source_db_url
Swiss-Prot	us.expasy.org/sprot
RefSeq	www.ncbi.nlm.nih.gov/ RefSeq

Fourth Normal Form

- A composite primary key should not contain independently multivalued components

~~**Protein_name**~~	~~**Protein_function**~~	~~**Gene**~~
~~Calmodulin~~	~~Binds calcium~~	~~CALM1~~
~~Calmodulin~~	~~Activates CaMKII~~	~~CALM1~~
~~Calmodulin~~	~~Binds calcium~~	~~CALM2~~
~~Calmodulin~~	~~Activates CaMKII~~	~~CALM2~~

Protein_name and Protein_function vary independently of Protein_name and Gene

Protein_name	**Protein_function**
Calmodulin	Binds calcium
Calmodulin	Activates CaMKII

Protein_name	**Gene**
Calmodulin	CALM1
Calmodulin	CALM2

Fifth Normal Form

- Remove pairwise cyclic dependencies from composite primary keys with three or more components

Normalization Rules

- Normalize to at least Boyce-Codd Normal Form
  - 3NF is acceptable if you aren’t using system-generated primary keys
- Goal is remove redundancy
  - Redundancy can lead to inconsistency
- Always keep business rules in mind while normalizing
  - Make sure you understand dependencies among attributes before moving to 2NF and beyond

Homework

- Reading for this week's class:
  - Suggested: Fleming and von Halle Chapter 8 (Chapters 4-7 optional, covering design process)
  - The Trip-Packing Dilemma article (on website)
  - Optional: Writing Quality Requirements article (on website)
- Homework: Develop ER diagram for project described on next slide, and include a list of questions you'd ask to refine your diagram.
  - Grading focuses on normalization and diagram syntax
  - We'll discuss your questions and the design at the start of next week's class

- Reading for next week's class
  - GenBank portion of the NCBI handbook, UniProt user manual (on website)
- Review for the midterm!
  - Time for questions at the beginning of next week's class

Homework

- Scientists want a database that stores information about proteins
- Each protein can have multiple names. All proteins have at least one name.
- Each protein can have multiple sequences, each one comes from an external database, and has an identifier assigned by that database. Not all proteins will have a sequence associated with them.
  - You can assume that sequences from different databases are different.
- Each protein can have multiple functions associated with it, but not all proteins will have an associated function.