Biological Database Design Week 3

Winter ’05

Melanie Nelson, Ph.D.

Question and Answer

- Discuss homework
- Q & A on last two weeks' material

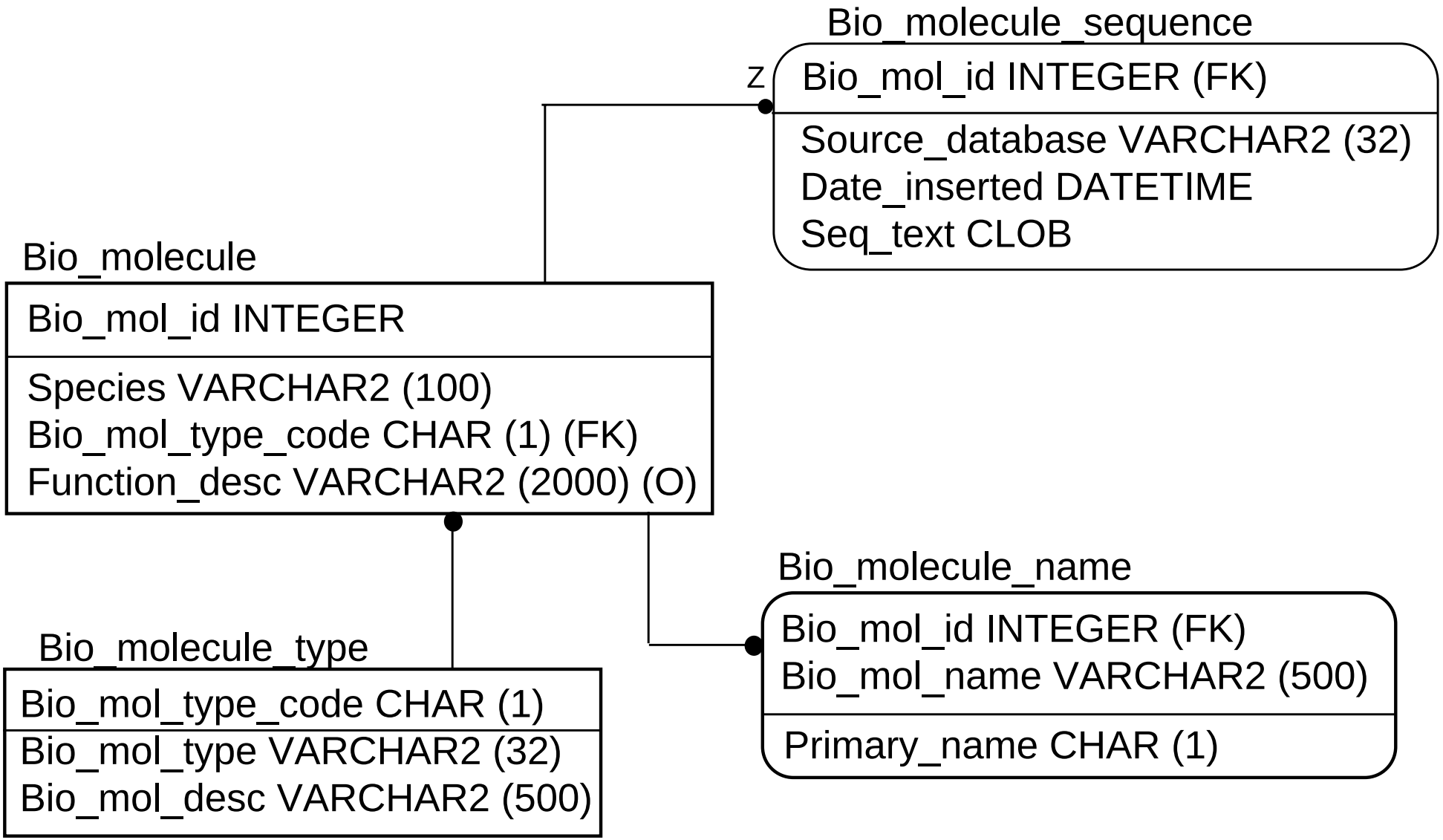
Introduction to SQL

- SQL = Structured Query Language
  - Except that the spec says SQL doesn't stand for anything
- Standard language for accessing data in relational databases
- A nonprocedural language
  - Say what you want, not how to get it
  - A RDBMS has a query optimizer that figures out how to get the data
- RDBMS purists point out that it is not fully compliant with relational database theory
  - Poor support of domains
  - Allows tables without keys

Introduction to SQL

- Data Definition Language (DDL)
  - CREATE TABLE, DROP TABLE
  - CREATE INDEX
  - Constraints: UNIQUE, PRIMARY KEY, FOREIGN KEY, NOT NULL
- Data Manipulation Language (DML)
  - INSERT, UPDATE, DELETE
  - SELECT
  - UNION, INTERSECT, EXCEPT

Example Tables



CREATE TABLE

- Use to create a table
- **CREATE TABLE** table1
  (column1 datatype **PRIMARY KEY**,
  column2 datatype)
- Each table should have a primary key constraint on one or more columns
- Use **UNIQUE** to enforce alternate keys

CREATE TABLE

Create a table to store biological molecules

```
CREATE TABLE Bio_molecule (
    Bio_mol_id INTEGER PRIMARY KEY,
    Species VARCHAR2 (50) NOT NULL,
    Bio_mol_type_code CHAR (1) NOT NULL,
    Function_desc VARCHAR2 (2000)
)
```

PRIMARY KEY is equivalent to UNIQUE, NOT NULL

Other DDL Commands

- ALTER TABLE
  - Add/drop/modify a column of a table
  - Not all DBMS support drop and modify
- CREATE INDEX
  - Create an index on a column or combination of columns
  - Implementation detail: indexes are used by DBMS to enforce constraints and optimize lookup
  - UNIQUE constraints automatically create index
- DROP TABLE, DROP INDEX

INSERT

- Use INSERT to get data into a table
- **INSERT INTO** table1 (column list)
  VALUES (value list)
- Column list is optional, but should specify it if the statement is included in application code
  - Remember, the columns in a table are not in any particular order!

INSERT

Insert the name “PTP1B” for biological molecule #1456. It is a primary name.

```
INSERT INTO Bio_molecule_name
   (Bio_mol_id, Bio_mol_name, Primary_name)
VALUES (1456, ‘PTP1B’, ‘Y’)
```

Text is surrounded by single quotes.

UPDATE

- Use to alter data in a table
- **UPDATE** table1

  SET column1 = new value,

  column2 = new value

  WHERE column3 = condition
- WHERE clause is optional. Without it, the UPDATE will apply to all rows in the table

UPDATE

Change calmodulin to be the primary name.

```
UPDATE   Bio_molecule_name
SET      Primary_name = 'Y'
WHERE    Bio_mol_name = 'calmodulin'
AND      Bio_mol_id = 456
```

Bio_mol_id portion of where clause is probably unnecessary.

DELETE

- Removes row(s) from table
- **DELETE FROM** table1

  WHERE column1 = condition
- WHERE clause is optional. Without it, DELETE will remove all rows from the table.
  - Won't remove table
  - To do this, use DROP TABLE

DELETE

Delete all Incyte sequence data

```
DELETE FROM Bio_molecule_sequence
WHERE Source_database = 'INCYTE'
```

SELECT

- Use to get information out of tables
- **SELECT** column1, column2

  FROM table1

  WHERE column3 = condition
- WHERE clause is optional. Without it, the statement returns all rows in the table

SELECT

- List the primary name and bio_mol_id for all molecules:
  - SELECT Bio_mol_id, Bio_mol_name
    FROM Bio_molecule_name
    WHERE Primary_name = 'Y'

- List all biological molecules stored in the database:
  - SELECT *
    FROM Bio_molecule

SELECT DISTINCT

- Use to get a list of distinct values
- **SELECT DISTINCT** (column1, column2)
  FROM table1
- Can have one or more columns in the select statement
- Multiple columns will provide distinct combinations of values of those columns

SELECT DISTINCT

Find out what types of biological molecules are represented in the Bio_molecule table:

```
SELECT DISTINCT Bio_mol_type_code
FROM Bio_molecule
```

JOIN

- Joins are used to combine information from multiple tables
- Two types of syntax
- **SELECT** table1.column1, table2.column2
  FROM table1, table2
  WHERE table1.column3 = table2.column3
- **SELECT** table1.column1, table2.column2
  FROM table1
  JOIN table 2 **ON** (table1.column3 = table2.column3)

JOIN

Show the biomolecule type, rather than the code, for all types represented in Bio_molecule:

```
SELECT DISTINCT Bio_mol_type
FROM  Bio_molecule bm,
          Bio_molecule_type bmt
WHERE bm.Bio_mol_type_code = bmt.Bio_mol_type_code


SELECT DISTINCT Bio_mol_type
FROM Bio_molecule bm
JOIN   Bio_molecule_type bmt
          ON bm.Bio_mole_type_code = bmt.Bio_mol_type_code
```

LIKE and Wildcards

- Wildcards are '%' and '_'
  - '%' = any number of characters
  - '_' = exactly one character
- Used with keyword LIKE
- Select information on all biomolecules with the word "kinase" in one of their names
  - ```
    SELECT   bm.Bio_mol_id, Bio_mol_name, Species
    FROM     Bio_molecule bm,
             Bio_molecule_name bmn
    WHERE    bm.Bio_mol_id = bmn.Bio_mol_id
    AND      Bio_mol_name LIKE '%kinase%'
    ```

Contents of strings are case-sensitive
    ```

ORDER BY

- ORDER BY returns rows in order
- List the names assigned to Biomolecule #478 in alphabetical order:
  - SELECT bio_mol_name
    FROM bio_molecule_name
    WHERE bio_mol_id = 478
    ORDER BY bio_mol_name ASC
- ASC or DESC

Aggregate Functions

- COUNT
  - Count number of sequences from RefSeq DB
  - SELECT COUNT (*)
    FROM Bio_molecule_sequence
    WHERE Source_database = 'RefSeq'
- GROUP BY
  - Count number of sequences from each DB
  - SELECT Source_database, COUNT (*)
    FROM Bio_molecule_sequence
    GROUP BY Source_database

Aggregate Functions

- MAX and MIN
  - SELECT MAX(Date_inserted)
    FROM Bio_molecule_sequence
  - Can be used on numeric and date fields
- SUM
- AVG

String Functions

- DBMS specific implementations
- Usually have at least:
  - Substrings
  - Length

Subqueries

- Can nest SQL statements:
  - Select all primary names for human proteins:

```
SELECT Bio_mol_name
FROM Bio_molecule_name
WHERE Bio_mol_id IN (
  SELECT Bio_mol_id
  FROM Bio_molecule
  WHERE Species = 'Homo sapiens'
  AND  Bio_mol_type_code = 'P'
)
```

Subqueries

- EXISTS
  - Another way to express subsets

```
SELECT Bio_mol_name
FROM Bio_molecule_name bmn
WHERE EXISTS (
   SELECT *
   FROM Bio_molecule bm
   WHERE Species = 'Homo sapiens'
   AND    Bio_mol_type_code = 'P'
   AND    bm.Bio_mol_id = bmn.Bio_mol_id
)
```

Subqueries

- Can also use NOT IN and NOT EXISTS
- Choice between using JOIN, IN, or EXISTS is a performance tuning issue
- Optimizer will usually “convert” for you, but sometimes it pays to optimize, or “tune” the query yourself
- For more details:
  - SQL Performance Tuning, by P. Gulutzan and T. Pelzer

Subqueries

- Can join back to the same table
- Show the primary name for all biomolecules for which there are no other names:

```
SELECT Bio_mol_name
FROM      Bio_molecule_name bmn1
WHERE    Primary = 'Y'
AND NOT EXISTS (
      SELECT *
      FROM     Bio_molecule_name bmn2
      WHERE  Primary <> 'Y'
      AND       bmn2.Bio_mol_id = bmn1.Bio_mol_id
)
```

CLOBs

- **CLOB** = **C**haracter **L**arge **Ob**ject
- Implementation is very DBMS specific
- Usually do not have access to many functions
  - No substring or length functions
  - Can't use in WHERE clause
  - Can even be difficult to load in and select out

Sequence Data

- Bioinformatics has traditionally focused on handling sequence data
- Many sequence databases are not relational
  - Particularly old ones: implemented prior to good DBMS support for CLOBs
  - GenBank and Swiss-Prot: originally flat file DBs, now have some relational storage
  - Lion's SRS (Sequence Retrieval System)
    - Popular way to handle sequences
    - Flat file based

Sources of Sequence Data

- Public
  - NCBI
    - GenBank = all sequences
    - RefSeq = curated sequences
  - ExPASY
    - SWISS-PROT = highly curated protein sequences
    - TrEMBL = uncurated protein sequences (translated EMBL)
- Private
  - Incyte (out of the genomics business)
  - Celera
- Proprietary
  - In house sequencing efforts

Sequence Data

- A typical sequence “entry” contains:
  - Sequence text
  - Metadata
- Metadata is not uniform across sources
  - Will almost always have the species
  - Curated data sources will usually have
    - Meaningful name (‘Mitogen-Activated Protein Kinase’)
    - Some indication of function
  - Uncurated data sources are often annotated by computer
    - Names often “similar to protein X” or “hypothetical protein”

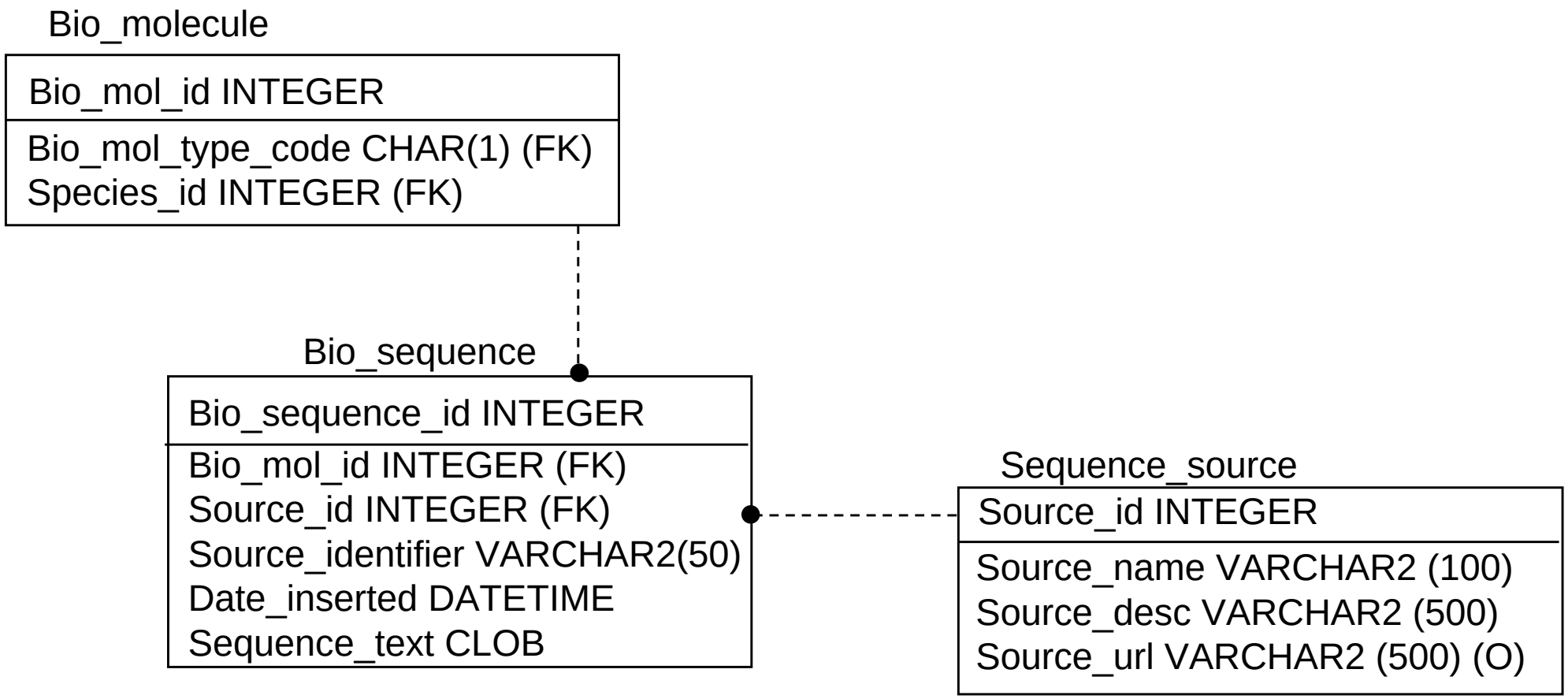
Molecule to Sequence Relationship

- The same “protein” or “gene” can be represented by multiple sequence entries
- Different databases often have slightly different sequences
  - Start codon selection
  - Initiator methionine included or not
  - SNPs (single nucleotide polymorphisms)
  - Sequencing errors
  - Splice variants (a headache in their own right)

Molecule to Sequence Relationship

- Difficult to ascertain when two sequences are the "same" molecule
- Requires scientists to set appropriate rules for your database
  - I've used 90 – 95% identity over at least 50 residues
  - Exact cutoffs depend on need for accuracy vs. need for inclusiveness
- Some databases bypass the issue and treat each sequence individually
  - Potential for lots of data duplication
  - Decision is ultimately made based on database scope

Relational Implementation



Sequence Text

- Protein and nucleotide
  - Nucleotides translate to proteins at 3 base pairs per amino acid
  - DNA sequences contain introns: unexpressed DNA “inserted” into gene
- Large range in size of sequence text
  - Common to study ESTs (~300 – 500 base pairs)
  - Smallest proteins are ~50-200 amino acids
  - Largest protein is titin, which has ~27,000 amino acids
  - Genomic DNA can be millions of base pairs long

Searches on Sequence Text

- Exact match
  - Not very useful, because small variations can occur in sequences that are scientifically “the same”
  - Used to remove (or flag) obvious redundancies
  - Some uses in intellectual property
- Global match (e.g., ClustalW)
  - Finds optimal alignment over entire length of two sequences
  - Allows insertions and substitutions
  - Not good at identifying matching regions within sequences that also have unmatched regions

Searches on Sequence Text

- Local match (e.g., BLAST)
  - Most common method of searching sequence DBs
  - Looks for regions of alignment within two sequences
  - Allows insertions and substitutions
- Motif or domain searches
  - Look for regions of sequence that match known patterns
  - Used to infer function
  - Search for characteristic motifs (BLOCKS, PRINTS, PROSITE)
  - Search for domains (Pfam, SMART)
  - Allow insertions and substitutions

Sequence Searching in RDBs

- Can't perform searches on CLOBs
- No easy way to implement the most useful types of searches in standard SQL
- Not all substitutions are equal
  - Some substitutions are more "conservative" than others
  - Preserve basic chemical properties of amino acid
  - Use a "substitution matrix such as BLOSSUM to specify "cost" of substitutions
  - Choice of substitution matrix may depend on personal preference, goals of project

Sequence Searching in RDBs

- Usually search on sequence text outside of relational database
- BLAST runs on a “database” of sequences in FASTA format
- Two options
  - Store sequences in database, but dump to FASTA for BLAST
  - Store sequences in FASTA flat files, reference these in database
  - Either way, DB and flat files can get out of sync
  - Storing sequences in database makes DB “gold standard”
- Oracle 10g implements BLAST searches in the database

Sequences as Non-Atomic Data

- In some databases, sequences are split into a table in which each amino acid or base pair is a row
- This is done when there is a need to store data about individual positions in the sequence
- Intermediate solutions: “break out” certain regions to store as individual residues
  - Functional motifs
  - Duplicates data

Sequence Metadata

- Metadata = data about data
  - Sequence is primary data
- Some metadata is a property of a particular sequence
  - Biophysical measurements: isoelectric point, extinction coefficents
- Some metadata is a property of the gene or protein that the sequence represents
  - Biological data: function, subcellular localization
- Species metadata can go either way
  - Depends on how you choose to handle orthologs in your database
  - Messiness of functional variation among orthologs means that a protein/gene is usually best associated with a single species

Sequence Species

- Species data is really a hierarchy
- For most applications, storing the full hierarchy is out of scope
  - Exceptions
    - Evolutionary biology
    - If need ability to perform deep searches on species (for “all mammals”, etc.)
- Usually need at least scientific name and one common name
  - Some people will also provide basic classifications: specifics depend on scope of DB
- Can link to/incorporate NCBI’s taxonomy DB
  - www.ncbi.nlm.nih.gov/Taxonomy

Sequence Function

- Two types of function (at least!)
  - Biochemical
    - The chemical process for which the protein/gene is responsible
    - Examples: kinase, calcium-binding
    - Enzymes: cross-reference EC (Enzyme commission) numbers (ENZYME: http://www.expasy.org/enzyme/)
    - Non-enzymes and enzymes: cross-reference molecular function Gene Ontology (http://www.geneontology.org)
  - Cellular/Process
    - The cellular pathway or process in which the protein/gene participates
    - Examples: DNA repair, long term potentiation
    - Cross-reference biological process Gene Ontology

Sequence Function

- Link to disease states may be considered a type of function, too
  - ICD codes (http://www.who.int/classifications/icd/en/)
- One gene or protein may be involved in multiple biochemical and cellular functions
  - Many enzymes have multiple binding sites
  - Many signal transduction proteins participate in multiple pathways
- There are always exceptions to standard ontologies
- If a scientist's favorite gene doesn't fit the standard ontology, and he can't explain why, he won't store the data!
  - Always provide a comment field

Additional Metadata

- Too numerous to list
  - Chromosome
  - Ligand binding sites
  - Intron locations
  - Active site residues
- Highly dependent on interests of group using database
- Often difficult to classify
- Constantly expanding list
- Some text, some numeric

Metadata Issues

- Due to incomplete nature of biological research, the features that are available vary widely by molecule
  - If you try to make a table with a column for each feature, you will have a lot of NULLs
  - Alternatively, making each feature its own table leads to an explosion of tables in your schema

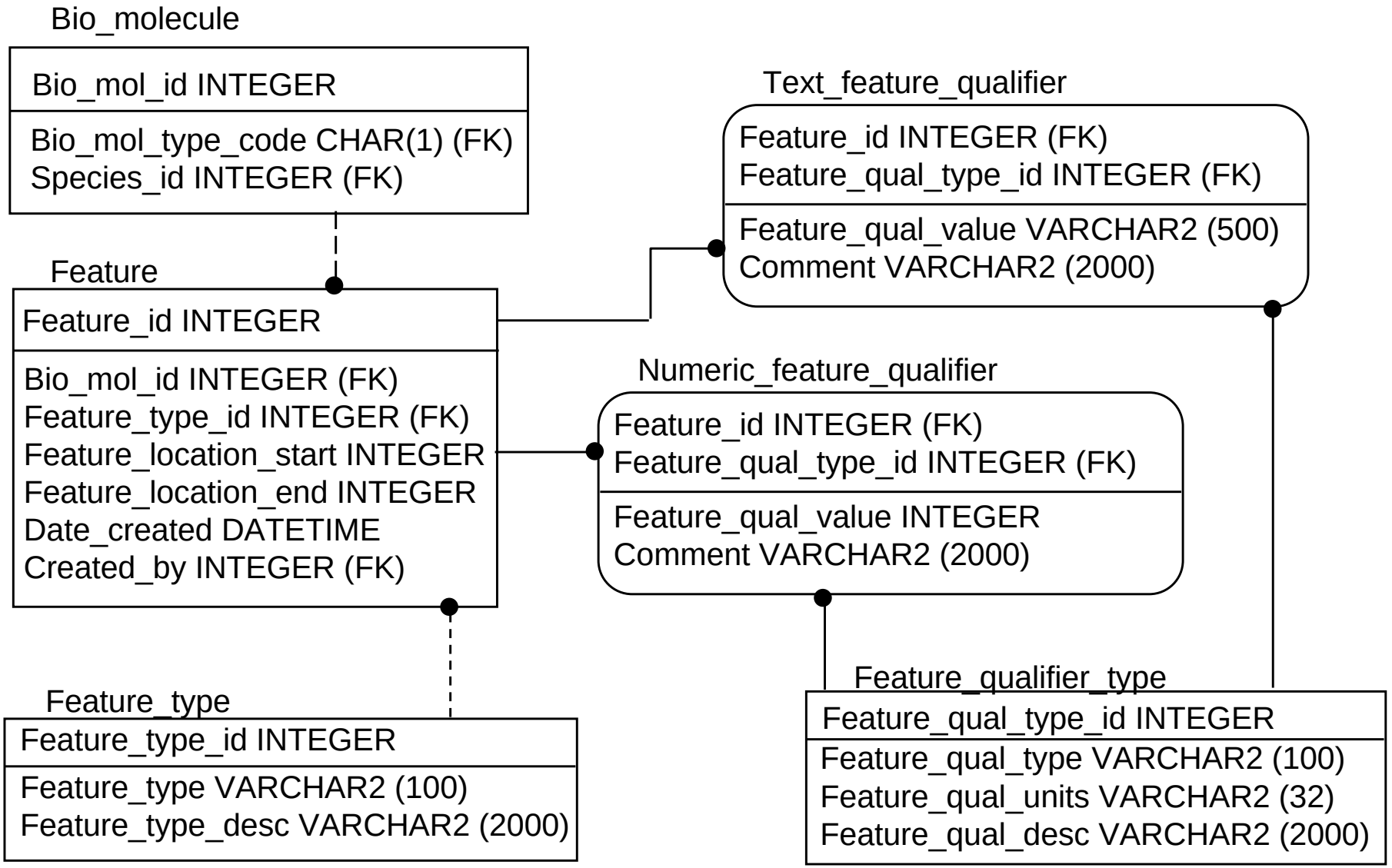
Additional Metadata

- Most public databases handle additional metadata as “feature table”
  - GenBank/EMBL feature table
    - Each feature has a location (optional: without location, feature is assumed to apply to entire sequence)
    - Features have “keys” (identifying names)
    - Features can have qualifiers (in GenBank spec, some are mandatory)
    - Example: primer-binding site feature
      - Key = primer_bind
      - Optional qualifiers: allele, citation, db_xref, evidence, gene, label, locus_tag, map, note, standard_name, PCR_conditions
  - Swiss-Prot has similar feature design
    - Comments apply to entire sequence
      - Examples: function, tissue specificity
    - Features are assigned a location
      - Examples: domain, binding site, post-translationally modified residue

Entity-Attribute-Value Design

- Standard design pattern used in many fields
- Values in table specifiy the feature, feature qualifier, and feature value
- If database needs to store features that apply only to regions of the sequence, add a “location” column
  - Requires separate tables for feature and qualifier, to avoid duplicating location
- Consider making feature type and feature qualifier lookup tables
  - Prevents duplicate names for same feature
- Store text and numeric features separately
  - Preserve ability to use numeric aggregate functions
  - Store units of numeric features

Relational Implementation



Difficulty Classifying Biological Data

- Biology is often a very “fuzzy” science
- Data is incomplete: scientists are constantly forming and discarding hypotheses
- Nature has a seemingly infinite way of combining features
- Dilemma
  - “Fuzziness” is real and important
  - Need “hard” classifications to support truly deep queries
  - Compromise
    - Make classification system user-extensible
    - Provide comment fields into which all of the real ambiguity can be entered

Tracking the Source of Data

- It is often desirable to track the source of features
  - Particularly if features may be entered by users (rather than downloaded from source databases only)
  - Also desirable because different source databases may provide contradictory metadata
- Lack of "feature source" tracking has created a problem with function annotations in public databases
  - Sequence A is annotated as a kinase because of sequence similarity with Sequence B
    - Sequence B turns out not to be a kinase
    - More likely: Sequence A has same basic structure as Sequence B, but lacks kinase function
  - Sequence C is annotated as a kinase because of similarity to Sequence A
  - If none of the "function transfers" are traceable, the function annotations cannot be trusted

Tracking the Source of Data

- In science, it is important to be able to lookup and evaluate source reference
- Science is incomplete
  - Your research contradicts the data in the database
  - Which is in error? Are both right, and we don't see the full picture yet?
  - Scientist needs to return to original source and evaluate the experiment

Tracking the Source of Data

- Gold standard is publication in peer reviewed journal
- Usually, but not always, indexed in PubMed (www.ncbi.nlm.nih.gov/PubMed)
- Other sources
  - Chemistry journals
  - Dissertations (rarely read, let alone cited…)
  - Webpages
  - Internal company reports

Tracking the Source of Data

- Reference data is actually quite complex
- In many applications, it is enough to link to PubMed
  - I usually provide ability to create internal, non-structured reference object for things not indexed in PubMed
- If need to allow queries into references, must store the reference itself
  - Find all features supported by papers on which Joe Q. Scientist is an author
- NCBI allows downloading of an XML version of reference, which is easy to parse into your database
- Object Management Group Bibliographic Query Service (OMG-BQS) model
  - http://industry.ebi.ac.uk/openBQS/
  - class diagram is in the specification section

Sequence Versioning

- Some public databases now version their sequences
  - Example: RefSeq
  - Sequence is identified by an accession number and a version
    - NM_005842.2
  - In general, only latest version of sequence is available
- Must decide how to handle versioning in your database
  - Keep all versions or latest version only?
  - If you keep all versions, do you associate different versions of the same sequence with each other?
  - What happens to any metadata added to the sequence when a new version comes out?

Questions to Ask

- Is your primary interest the sequences or the proteins/genes they represent? (Or both?)
  - Tells you whether you can simplify one or the other
- Do you need to search over “aggregate” species designations?
  - Tells you how much of the species hierarchy you need to store
- Do you need to search on details of supporting data, or just link to it?
  - Tells you whether you need to store all reference data, or just a link to it
- Do you need to associate data with a particular version of a sequence?
  - Tells you whether you need to track versions

Additional Data Models

- ENSEMBL data model
  - Relational database for ENSEMBL
  - http://www.ensembl.org/Docs/schema_description.html
- bioSQL
  - http://obda.open-bio.org
  - From the Open Bioinformatics Foundation (open-bio.org)
- aMAZE
  - Interesting data model for representing function
  - http://www.amaze.ulb.ac.be
  - Representing and analysing molecular and cellular function using the computer. J. van Helden, et. al. (2000) Biol. Chem. 381:921-935.

Homework

- Reading for this week's class
  - GenBank portion of the NCBI handbook, UniProt user manual (on website)
- Homework: Project plans are due next week

- Reading for next week's class
  - Paper discussing GeneLogic's approach to managing gene expression data
  - Implementing LIMS: A "How To" Guide
- Optional reading for next week's class
  - Nature Genetics paper on MIAME (strongly recommended, but will require a trip to the library)
  - A computer scientist's explanation of microarrays (strongly recommended for those not familiar with the technique)
  - MAGE-ML paper