BINF 8211/6211
Design and Implementation of Bioinformatics Databases
Lecture #8

Dr. D. Andrew Carr
Dept. Bioinformatics and Genomics UNCC
Spring 2016

Set Theory

- Relational algebra: an algebra of sets
  - Based on a subset of first-order logic that specifies legal operations over the sets.
  - Tuples are functions from a (finite) set U of attributes of a relation to a domain of values.
    - (Protein: "Green Fluorescent protein", mw: 55,000)
  - Operators provide the expressive power of the language
    - Primitive operators: selection, projection, Cartesian product (cross join), set union, set difference, rename.
    - Derived operators include: set intersection, division, natural join, etc.

- For more theory: this is an accessible and relatively short ebook on set theory and the relational calculus (relational algebra is restricted to finite tuples but otherwise has the same expressive abilities), for those who want to really understand the basis of this domain (see Moodle – Readings for this week).
  - http://www.academia.edu/1739363/DISCRETE_MATHEMATICS_VIA_RELATIONAL_DATABASES

- Set theory is a field with axioms, applications and areas of study – relations are not *exactly* sets so some care has to be taken in generalizing.
- Everything can be assigned to groups – what is the correspondence between groups, what constitutes a subset.

Symbol, Name	Use
σ Selection	Return rows of the input relation that are TRUE (satisfy the predicate)
Π Projection	Output columns from all rows of the input relation (remove duplicate tuples)
X Cartesian Product	Output all pairs of rows from TWO input relations
$\cup$ Union	Output the union of all tuples from two input relations.
- Difference	Allows you to find tuples in one relation but not another
ρ Rename	Given an expression E, ρ_x (E) returns the result of E under a name x.
|X| Join, with variations... e.g. ⟕ (Left Outer)	Return pairs of rows from TWO input relations where attribute values match (for attributes with the same name)

Relational Algebra with Examples

- A relational database handles finite, unordered sets (whereas in mathematics the sets are ordered and can be infinite).
- A relation is a set of tuples (e_1, e_2, ….e_n) where each element e_i is a member of E_i, a data domain.
  - Each element is an attribute value.
  - An attribute is a name paired with a domain.
  - attribute value pairs the attribute name with an element of the domain.
  - A tuple is a set of attribute values in which no two distinct elements have the same name.
    - You can describe the tuple as a function that maps the names to the values.

Relational Algebra with Examples

- A relationship is an association among attributes of 2 or more entities

  DNAase I *digests* DNA .

  enzyme entity relationship set *substrate* entity

- To generalize, using math notation, if $(e_1, e_2, \ldots e_n)$ is a relationship

  $\{(e_1, e_2, \ldots e_n) | e_1 \in E_1, e_2 \in E_2, \ldots e_n \in E_n\}$ And (DNAase I, DNA) $\in$ *digests*

A few more definitions and terms for relations

- A set of attributes in which no two distinct element have the same name is called a heading. A set of tuples having the same heading is called a body
- A relation is a heading paired with a body, the number of attributes in a heading is called the degree of the tuple.

Set theory has binary operations on sets that we can extend to relations, with a few restrictions.

- Union of sets A, B – the set of all objects that are a member of A or B or both.
- Intersection of sets A and B – the set of all objects that are in both set A and set B
- Set *difference* of sets A and B is the set of all members of B that are NOT members of A or, alternatively, A NOT B.
  - The complement of a set is those made of those things that are not in the set. The difference is the relative complement – comparing two finite sets.
  - In this case the order in which you consider the sets changes the outcome.

A = {1,2,3} and B = {2,3,4}

A ∪ B	A ∩ B	A \ B or B \ A	(A ∪ B) \ (A ∩ B)	A x B
{1,2,3,4	{2,3}	{1} {4}	{1,4}	{1,2}{1,3}{1,4} {2,2}{2,3}{2,4} {3,2}{3,3}{3,4}

More operations on sets

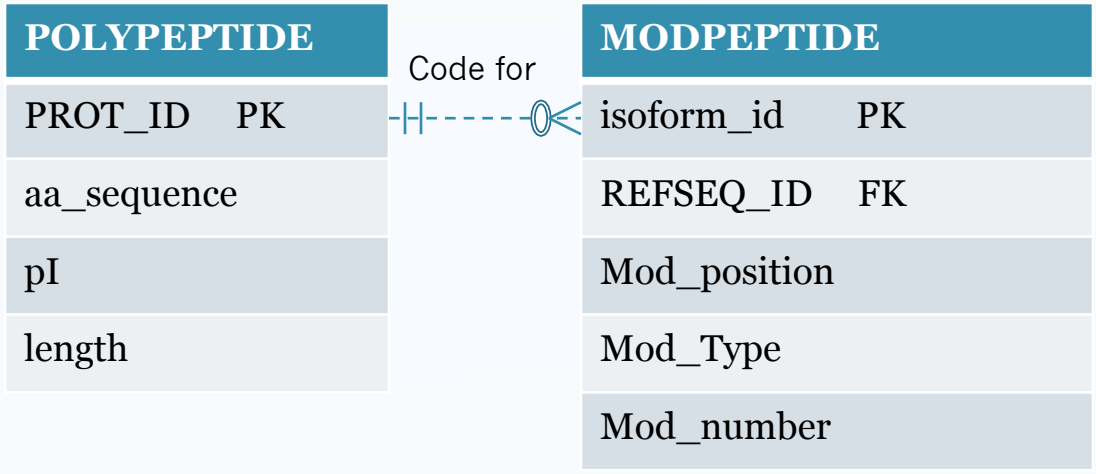
- *Symmetric difference* of sets A and B is the set of all objects that are a member of only A or only B but not in both – this is the same as the difference of the union and the intersection. {1,4}
- The Power Set of a set A is the set whose members are all possible subsets of A (the empty set is one).
- *Cartesian product* of A x B: the set whose members are all possible ordered pairs (a,b) where a is a member of A and b is a member of B.

A = {1,2,3} and B = {2,3,4}

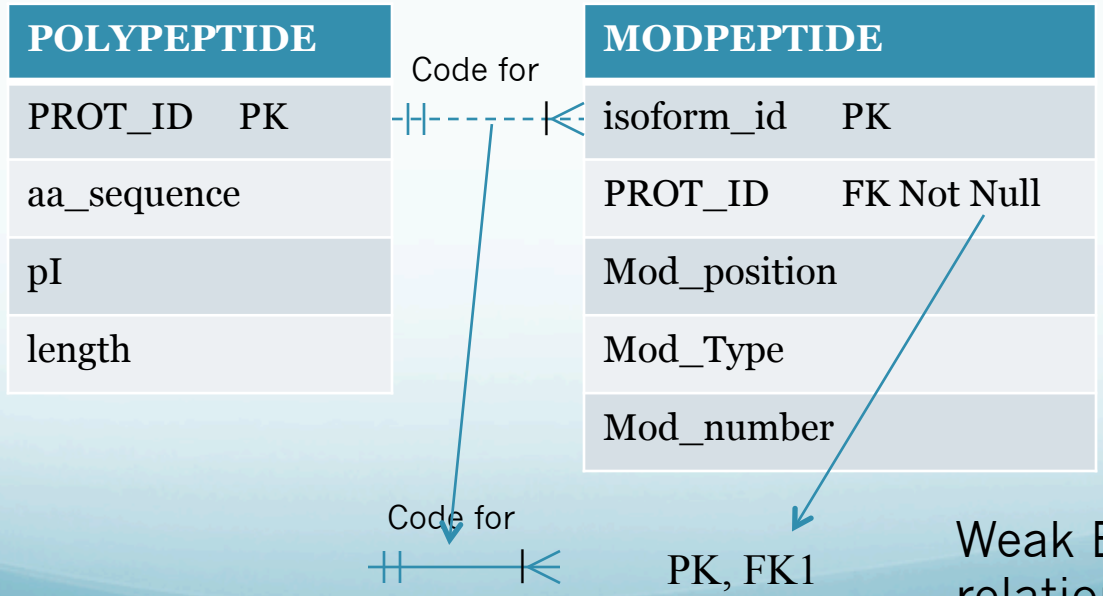
A ∪ B	A ∩ B	A \ B or B \ A	(A ∪ B) \ (A ∩ B)	A x B
{1,2,3,4}	{2,3}	{1} {4}	{1,4}	{1,2}{1,3}{1,4} {2,2}{2,3}{2,4} {3,2}{3,3}{3,4}

Relationship Participation

- When you define a relationship between two entity sets, the model can support the *possibility* of linked instances, or you may *require* the presence of linked instances.
  - Optional participation means what it says – there can be an instance in either entity without a match the other.
    - Example – a polypeptide may have no post-translationally modified forms, so the existence of an instance (next slide) in the ProteinModified entity set for an instance in the Polypeptide set is Optional.
      - The FK in a child table is allowed to be NULL
      - The optional side shows a zero for range
    - Graphically, use the 0< notation instead of the |< notation, since the minimum cardinality is zero instead of 1.
  - Mandatory participation also means what it says – there must be corresponding instances of entities across the relationship between two entity sets.
    - The minimum cardinality is 1, and the graphical notation is | < for the mandatory entity (or ||) .
    - The FK in the child table is NOT NULL
    - The mandatory side has a ‘1 in the diagram



Optional participation



Mandatory participation

Weak Entity, strong relationship

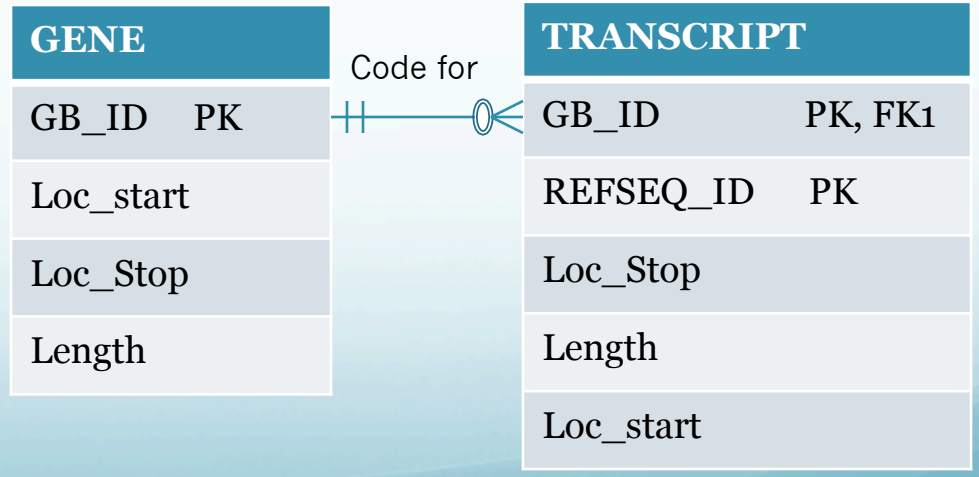
Relationship participation vs relationship strength

Relationship participation and strength are **not** the same thing: the relationship is not necessarily weak when the entities are in an optional relationship nor strong when it is mandatory.

Relationship strength (strong/weak) depends on how the PK is formulated, while *participation* (mandatory/optional) depends on how the *process* is defined (rules).

Entity strength depends on both participation and strength: a weak entity has mandatory participation and a strong relationship (PK of parent defines entity).

Strong (identifying) relationship

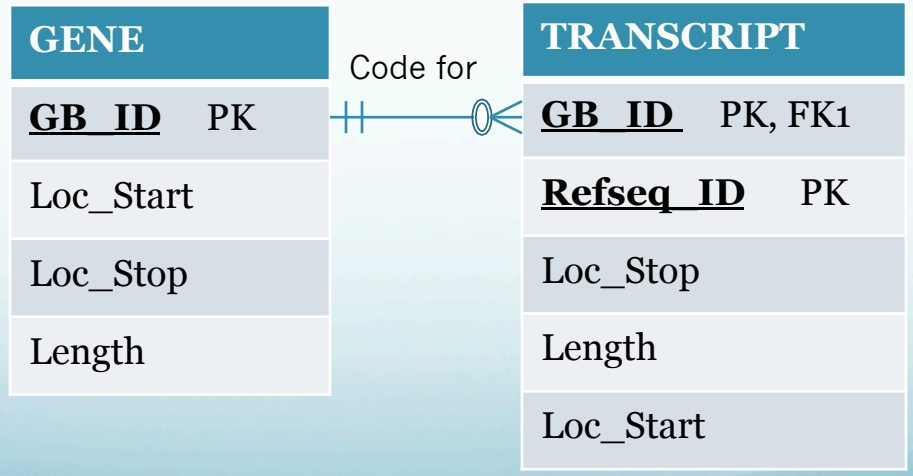


Graphical Notation Summary and also a check-list for figuring out the categories.

Relationship	FK - child side	Domain - child side	IE connector
Optional	NULL	(0…N)	-0<
Mandatory	NOT NULL	(1…N)	-|<
Weak	Single PK		- - - - - - -
Strong	Composite PK		----------
Entity			
Weak	Composite PK	(1…N)	- - - |<

A shorthand notation for text communication

- Graphical notation is a good shorthand way to communicate a lot of information
  - It is slow to encode and requires special software
- Text notation convention for entities:
  - GENE(**GB_ID,**Loc_Start, Loc_Stop, Length**),** TRANSCRIPT(**GB_ID**,**Refseq_ID**, Loc_Stop, Length, Loc_Start**)**
- Why can I have the same attribute name in both entities?



Eight fundamental operators of relational algebra pt 1.

- SELECT
  - Restrict
  - Horizontal subset of the data
- PROJECT
  - All values in a given attribute
  - Vertical subset of the data
- INTERSECT
  - From set theory, overlap between tables
- DIFFERENCE
  - The non-overlapping entries between tables
- UNION
  - Combination of tables with shared attribute spaces

Eight fundamental operators of relational algebra pt 2.

- PRODUCT
  - Cartesian product from mathematics
    - All combine with all
- DIVIDE
  - Complicated and requires 2d to 1d relationship
  - Returns the intersection between specific attribute, entity pairs
- JOIN
  - This is the fundamental tool of the RDBMS system
  - Natural Join
    - Rows and columns that have common attributes
    - PRODUCT → SELECT → PROJECT
  - LEFT OUTER, RIGHT OUTER
    - Keeps all the values from 1 table and merges with the other table where it can

Lecture 8

- What is an identifying relationship and how do you indicate its presence in a diagram? How about the non-identifying relationship?
- Mandatory and optional relationship participation is defined how?
- A weak entity is defined how?

INDEXING Review

- What is indexing in a DBMS?
  - How do systems index?
- Why do you index?

Indexing: in large tables, data retrieval can be quite slow. Indexes are a kind of Table of Contents to tables in a database, they can be generated automatically by the DBMS, or the administrator can direct their creation.

GENE	
Refseq_id	symbol
NG_0089723.1	HCN2
NG_0000523.2	SLC34A2
NG_0899664.1	FOX4A

How can I find all of the alternative transcripts known for the gene HCN2?

Method 1: Scan TRANSCRIPT for NG_089723.1

TRANSCRIPT		
Transcript_id	Transcript_symbol	Refseq_id
AC_4366875	HCN2.3	NG_0089723.1 *
AC_8824311	FOX4A.2	NG_0899664.1
AC_3455769	HCN2.5	NG_0089723.1 *
AC_7778112	HCN2.1	NG_0089723.1 *

Method 2: pre-scan the table and organize the records

TRANSCRIPT		
Transcript_id	Transcript_symbol	Refseq_id
AC_4366875 *	HCN2.3	NG_0089723.1 *
AC_8824311	FOX4A.2	NG_0899664.1
AC_3455769 *	HCN2.5	NG_0089723.1 *
AC_7778112 *	HCN2.1	NG_0089723.1 *

NG_0089723.1 maps to rows 1, 3 and 4 so I can use AC_4366875, AC_3455769 and AC_7778112 as *the index keys* and store them with pointers to rows 1, 3 and 4.

It is allowed to index on one *or more* attributes in a given table.

Most DBMS ***automatically index the PKs*** of each table, as a unique index key with a single –row pointer.