
Biological Database Design

Week 1

Winter '04

Melanie Nelson, Ph.D.

Introductions

- About me
 - Ph.D. in biochemistry (1999)
 - Bioinformatics/IT in Biotech
 - Physiome Sciences (now Predix Pharmaceuticals)
 - GeneFormatics (now Cengent Therapeutics)
 - SAIC (Biomedical Information Solutions Division)
 - Experience with
 - Databases
 - XML
 - Perl
 - Protein Structure/Function Analysis
 - E-mail: m-nelson-1@alumni.uchicago.edu (1 day turnaround)

Course Overview

■ Goals

- Intro to DBs
- Overview of common types of biological data
- Introduction to biology-specific problems/issues

■ Grading

- 10% homework
- 40% midterm
- 50% final project

Course Overview

■ Books

- ❑ Handbook for Relational Database Design (Fleming and von Halle): old, but covers basics well
- ❑ An Introduction to Database Systems (Date): the classic: frequently updated but overkill for this course
- ❑ SQL books: pick one that suits your needs

■ Online resources

- My website:
www.32geeks.com/classes/biodb_design_2004

Course Overview

- Week 1
 - Introduction to databases
 - Fundamentals of the relational model
- Week 2
 - Database design process
 - ER diagrams
 - Normalization
- Week 3
 - Intro to SQL
 - Bio data 1: Gene and protein sequences and metadata
 - Midterm

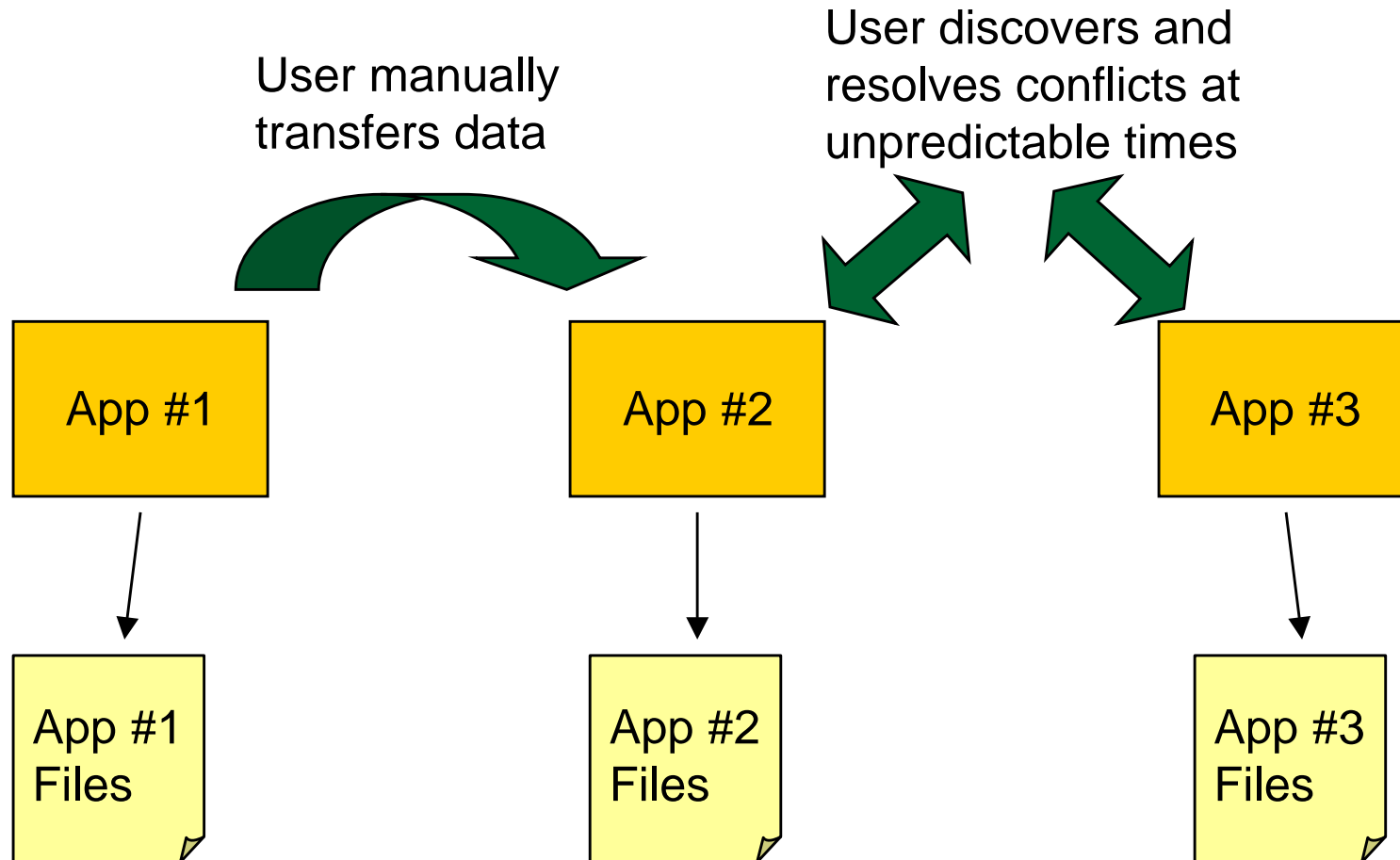
Course Overview

- Week 4
 - Bio data 2: gene expression
 - Bio data 3: LIMS
 - Project plans due
- Week 5
 - Biology-specific issues in database design
- Week 6
 - Biology-specific issues in database design and/or special topics
 - Final project presentations

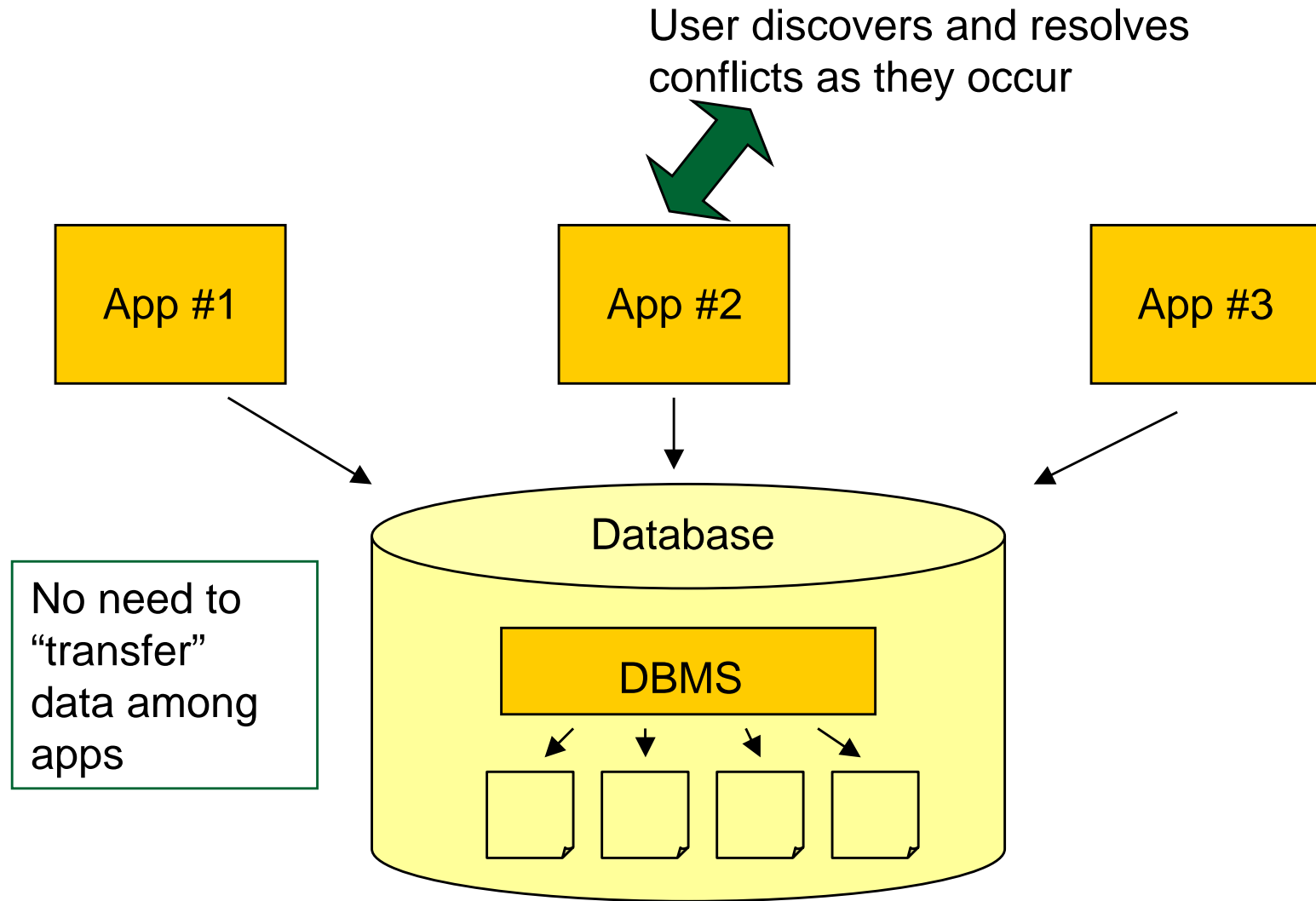
Introduction to Databases

- Information in a database is
 - Structured (searchable)
 - Capable of being shared with multiple applications (multiple uses)
 - Databases are supported by a database management system
 - Layer between applications using the data and the raw data
 - Handles requests for data
 - Manages concurrency
 - Protects data integrity
- } consistency

Data Management without Databases



Data Management with Databases



Some Advantages of Databases

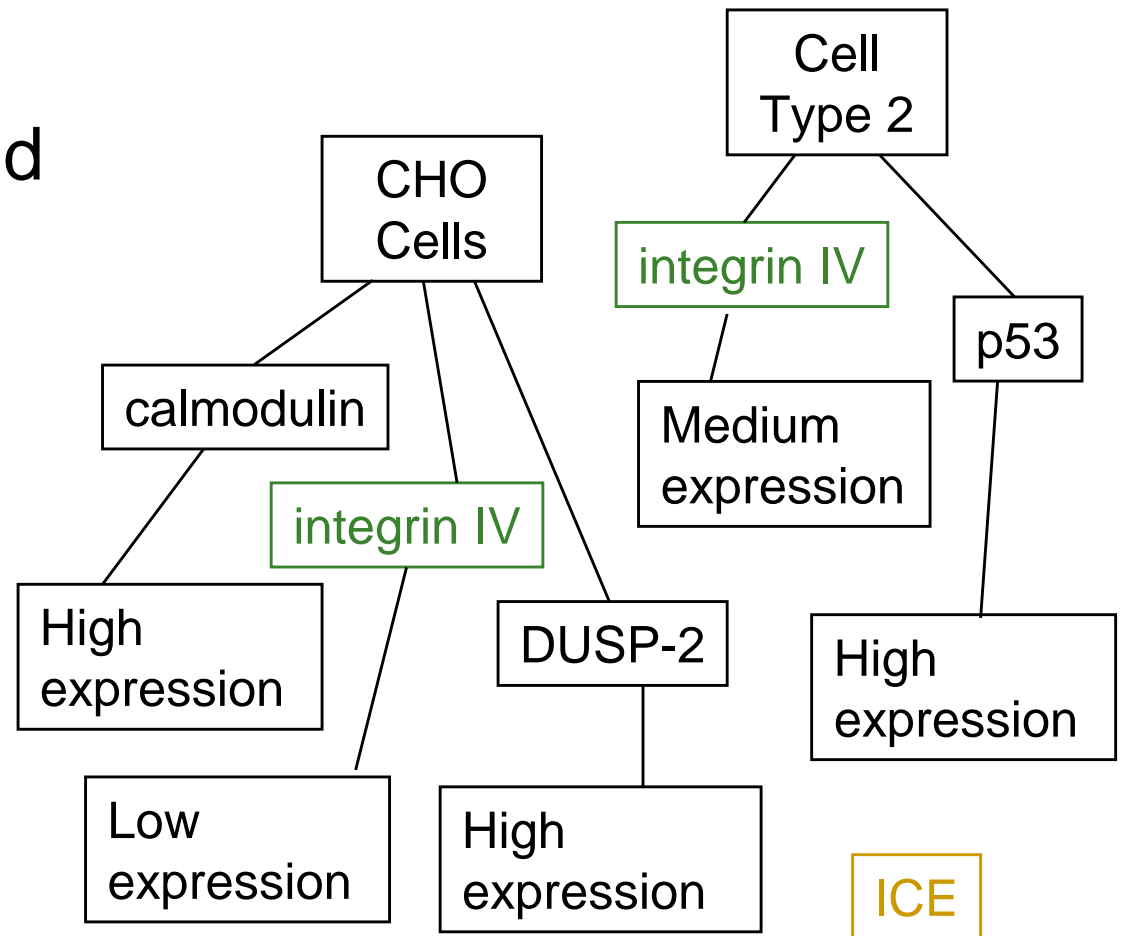
- **Improve interoperability:** app #1 has “sequence”, app #2 has “prot_seq”. Are they the same thing?
- **Reduce inconsistency:** app #1 says protein A binds drug Z, app #2 says it doesn't. Which is right?
- **Improve efficiency:** scientists/programmers don't have to gather data for each application/question

Types of Database Systems

- Four main types of databases:
 - Hierarchical
 - Network
 - Relational
 - Object-Oriented

Hierarchical Databases

- Information organized into tree, or parent-child relationships
- Data gets **uplicated** when one child has more than one parent
- Data gets **lost** when a child doesn't have a parent

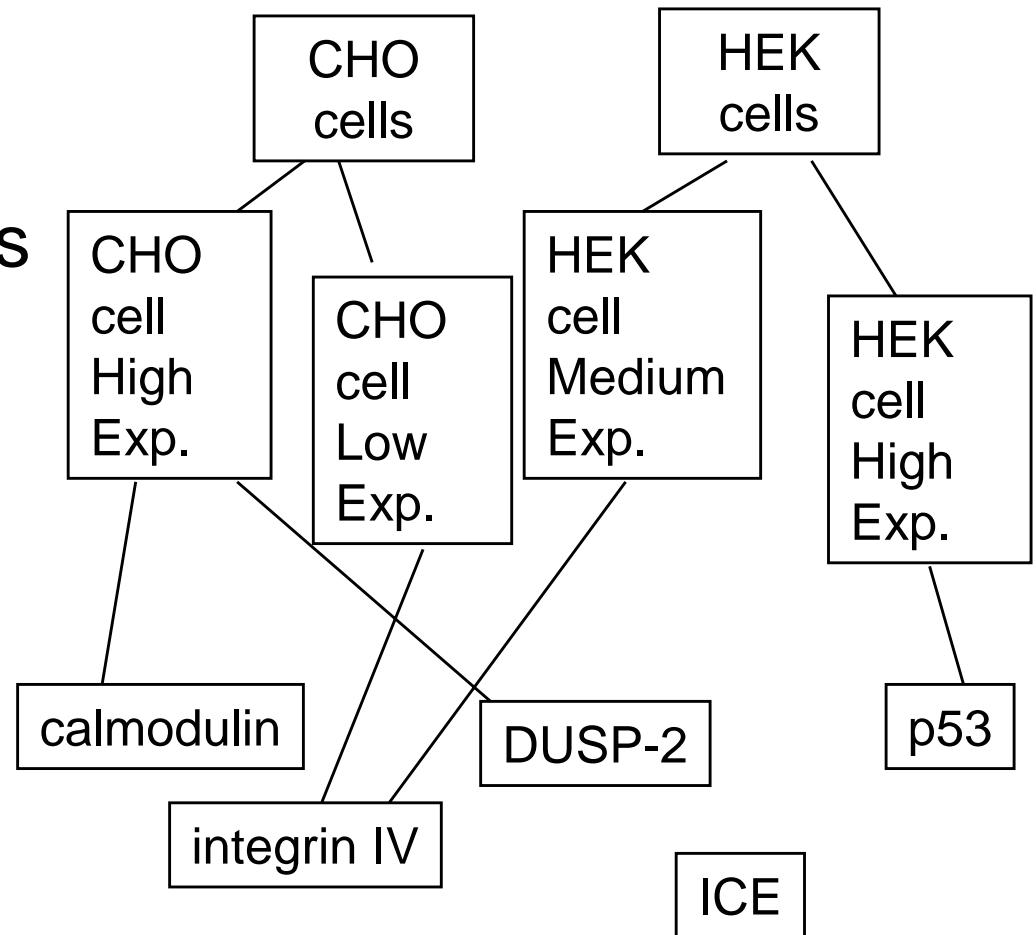


Hierarchical Databases

- Historically, the first type of database
 - IBM's Information Management System (IMS)
 - Introduced in 1968
- XML can be viewed as a hierarchical database
 - Information is organized into tree
 - Collections of XML files can be used as a database

Network Databases

- Extends hierarchical model to allow children to have multiple parents
- Model has:
 - Records
 - Links between records
- Careful design can avoid data duplication
- Complicated design and data access



Network Databases

- Emerged in the 70s
- Conference on Data Systems Languages (CODASYL) produced guidelines for databases
- XML with XLink can be viewed as a network database
 - XLink allows links across branches in the XML tree

Relational Databases

- Information is modeled as tables (relations) with links between tables
- Rigorous mathematical basis
 - Allows prevention of data duplication and other data integrity problems
 - Simplifies data access

Cell Line

Cell line ID	Cell line type
Cell line 1	CHO cells
Cell line 2	HEK cells

Protein Expression

Cell line ID	Protein ID	Expression level
Cell line 1	Protein 1	High
Cell line 1	Protein 2	Low
Cell line 1	Protein 3	High
Cell line 2	Protein 2	Medium
Cell line 2	Protein 4	High

Protein

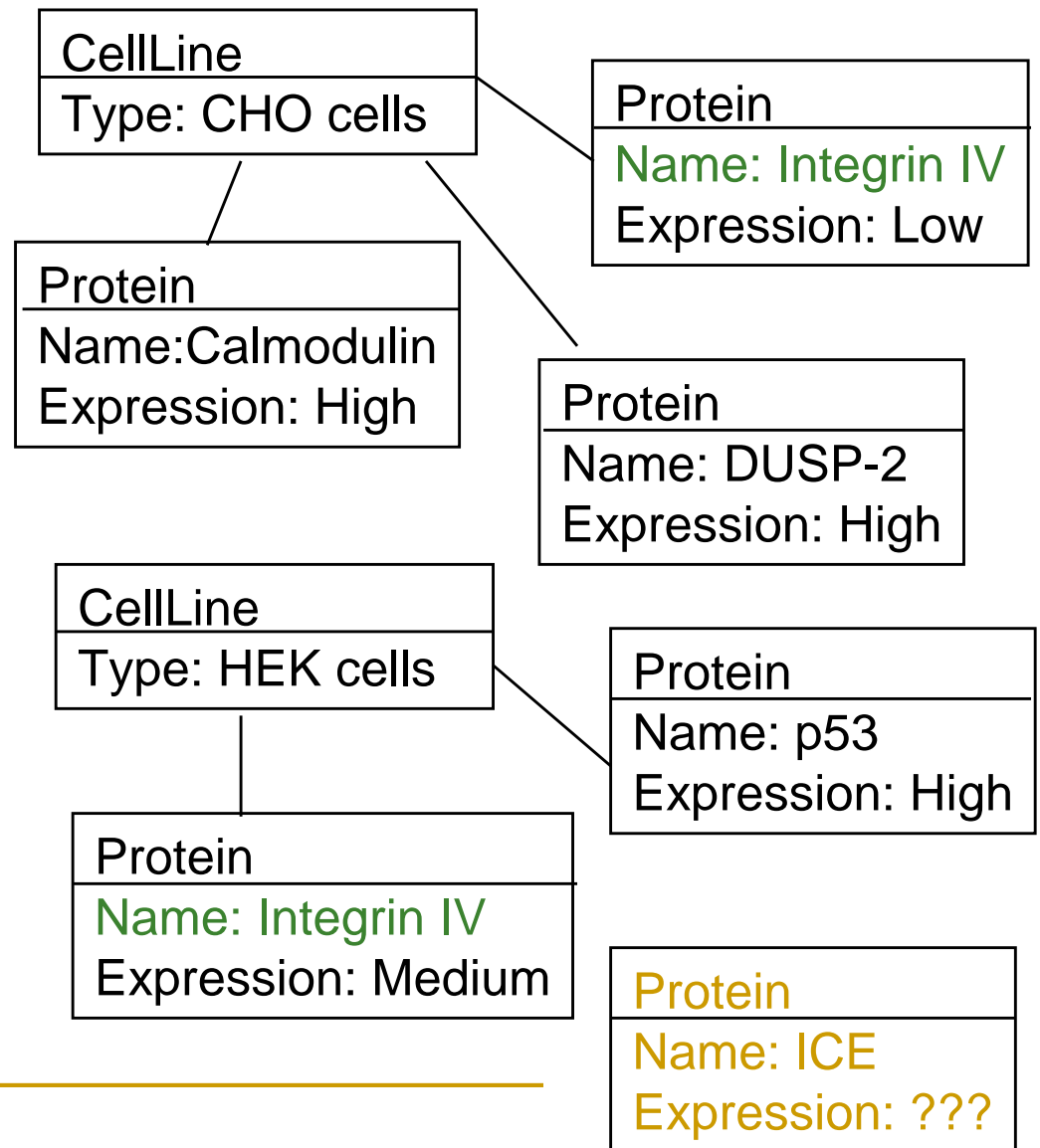
Protein ID	Protein Name
Protein 1	calmodulin
Protein 2	integrin IV
Protein 3	DUSP-2
Protein 4	ICE
Protein 5	p53

Relational Databases

- Developed in 70s by Dr. E.F. Codd at IBM
- Is the dominant model in use today
 - Oracle
 - IBM DB2
 - MS SQL Server
 - PostgreSQL
 - MySQL

Object-Oriented Databases

- OODBs store data in classes, with associations between classes
- Integrates data storage with data manipulation: methods are part of object
- Must be careful to avoid data duplication and “orphan” data



Object-Oriented Databases

- Introduced in 80s, in conjunction with rise in object-oriented programming techniques
 - There are difficulties integrating OO programming and relational DBs
 - Often have the same problems network DBs had
- Lack easy data access of relational DBs
- Major relational DBs have introduced “object extensions”
- Ongoing debate about how best to integrate DBs and OO programming

Relational vs the Other Models

- Relational model attempts to correctly represent data, without regard to how it will be used
- In other models, how the data will be used can greatly influence the design
 - If you design to a particular application, you will probably make it easy to answer the questions in that application...
 - But you may make it harder, or even impossible, to answer other types of questions!

Relational vs. Other Models

- Relational DBs were intended to free users from needing a programmer to write new code to answer each new question
- This is particularly useful in science: scientists will *always* think of a new question!
- SQL still too “programming-like” for many users
 - Flexible reporting apps attempt to address this

The Relational Model

- Direct quote from Date:
 - Data is perceived by users as tables (and nothing but tables)
 - The operators at the user's disposal...are operators that generate new tables from old, and those operators include at least **SELECT...**, **PROJECT**, and **JOIN**

The Relational Model

- The relational model speaks to:
 - Data structure
 - Data manipulation
 - Data integrity
- It does not speak to data storage
- Relational model refers to **logical** database design, not **physical** database design

The Relational Model

- Mathematically rigorous
- When correctly implemented, can guarantee accuracy of query results (assuming input was valid!)
- No current DBMS fully implements the relational model

Relational Terms

Relation = Table
Consists of

- heading (a fixed set of attributes)
- body (a set of tuples)

Attribute = Column
Also called a field

Tuple = Row
Also called a record.
A set of attribute:value pairs

Proteins

Protein ID	Protein Name
Protein 1	calmodulin
Protein 2	integrin IV
Protein 3	DUSP-2
Protein 4	ICE
Protein 5	p53

Primary key = Unique identifier
Attribute or combination of attributes that uniquely identifies each tuple

Domain = Valid set of values
“A named set of scalar values”
Each attribute has a domain upon which it is defined

Properties of Relational Tables

- The following properties are a consequence of the definition of relations, attributes, and domains:
 - Each column has a unique name (The heading = a fixed set of attributes)
 - All entries in a given column are of the same kind (Attributes are defined on a domain)

Properties of Relational Tables

- There are no duplicate tuples
 - “Each row is unique”
 - The body of a relation is a mathematical set: sets do not have duplicate elements
 - Primary key ensures this rule is upheld
 - Do not circumvent!
 - Common to use system-assigned numerical value as primary key
 - Should have an “alternate key” that is inherent in the data

Properties of Relational Tables

- The sequence of tuples is unimportant
 - Sets are unordered
 - DBA may change way in which rows are partitioned in storage to improve performance of certain queries
 - Never write code that assumes a query will return results in a given order
 - If tuple order is meaningful, it should be specified by an attribute

Properties of Relational Tables

- The sequence of attributes is unimportant
 - The heading of a relation is also a set
 - DBA may change physical order of columns to improve performance of certain queries
 - Never assume the columns will be returned in a given order: specify the order in the query

Properties of Relational Tables

- Attribute values are atomic
 - “Entries in columns are single-valued”
 - First normal form

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

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

Types of Relations

- **Base relation** = an autonomous relation (i.e., not defined in terms of another relation)
 - What we typically mean when we talk about database tables
- **Derived relation** = a relation defined in terms of other relations
 - Query results, for instance
- **View** = a named derived relation
 - SQL to generate derived relation is stored in database
- **Materialized view** = a view in which data is actually copied
 - “snapshot”
 - Used to improve performance

Data Manipulation

- Closure: relational operators operate on relations and produce relations
 - Allows nested expressions
- Relational operators are not affected by changes to physical storage of data
- SQL is current standard

Relational Operators: Select

- Also called restrict
- Retrieve a subset of rows (tuples) from a relation
- Subset is determined by a selection criteria
- SELECT *

```
FROM Protein_Name  
WHERE Protein_ID = 1
```

List all the names of Protein 1

Protein_Name

Protein_ID	Protein_Name
Protein 1	Calmodulin
Protein 1	CaM
Protein 3	DUSP-2
Protein 3	Dual specificity phosphatase 2
Protein 3	PAC1



Protein_ID	Protein_Name
Protein 1	Calmodulin
Protein 1	CaM

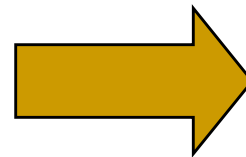
Relational Operators: Project

- Retrieve a subset of columns (attributes) from a relation
- `SELECT CellLineID, ProteinID
FROM ProteinExpression`

Get a list of proteins
expressed in each cell line

Protein Expression

Cell line ID	Protein ID	Expression level
Cell line 1	Protein 1	High
Cell line 1	Protein 2	Low
Cell line 1	Protein 3	High
Cell line 2	Protein 2	Medium
Cell line 2	Protein 4	High



Cell line ID	Protein ID
Cell line 1	Protein 1
Cell line 1	Protein 2
Cell line 1	Protein 3
Cell line 2	Protein 2
Cell line 2	Protein 4

Relational Operators: Product

- A cartesian product of two relations
- Each row in relation 1 is combined with each row in relation 2
- `SELECT *`
`FROM Cell_Line, Protein_Expresion`

Relational Operators: Product

Cell_Line

Cell_line_ID	Cell_line_type
Cell line 1	CHO cells
Cell line 2	HEK cells



Protein_Expression

Cell_line_ID	Protein_ID	Expr_level
Cell line 1	Protein 1	High
Cell line 1	Protein 2	Low
Cell line 1	Protein 3	High
Cell line 2	Protein 2	Medium
Cell line 2	Protein 4	High

cl.cell_line_id	cl.cell_line_type	pe.cell_line_id	pe.protein_id	pe.expr_level
Cell line 1	CHO cells	Cell line1	Protein 1	High
Cell line 1	CHO cells	Cell line1	Protein 2	Low
Cell line 1	CHO cells	Cell line1	Protein 3	High
Cell line 1	CHO cells	Cell line 2	Protein 2	Medium
Cell line 1	CHO cells	Cell line 2	Protein 4	High
Cell line 2	HEK cells	Cell line1	Protein 1	High
Cell line 2	HEK cells	Cell line1	Protein 2	Low
Cell line 2	HEK cells	Cell line1	Protein 3	High
Cell line 2	HEK cells	Cell line 2	Protein 2	Medium
Cell line 2	HEK cells	Cell line 2	Protein 4	High

Relational Operators

■ Join

- ❑ Combination of product and select
- ❑ Combines row from relation 1 with row from relation 2 only when selection criteria are met
- ❑ Criteria specify when rows are to be combined
- ❑ `SELECT *`

`FROM CellLine, ProteinExpression`

`WHERE CellLine.CellLineID =
ProteinExpression.CellLineID`

Relational Operators: Join

Cell_Line

Cell_line_ID	Cell_line_type
Cell line 1	CHO cells
Cell line 2	HEK cells



Protein_Expression

Cell_line_ID	Protein_ID	Expr_level
Cell line 1	Protein 1	High
Cell line 1	Protein 2	Low
Cell line 1	Protein 3	High
Cell line 2	Protein 2	Medium
Cell line 2	Protein 4	High

Include cell line name in expression information

cl.cell_line_id	cl.cell_line_type	pe.cell_line_id	pe.protein_id	pe.expr_level
Cell line 1	CHO cells	Cell line1	Protein 1	High
Cell line 1	CHO cells	Cell line1	Protein 2	Low
Cell line 1	CHO cells	Cell line1	Protein 3	High
Cell line 2	HEK cells	Cell line 2	Protein 2	Medium
Cell line 2	HEK cells	Cell line 2	Protein 4	High

More meaningful than a product!
More likely to combine with a project and
exclude the cell line ID.

Relational Operators: Join

- Types of join
 - Equi-join
 - Join criterion is equality of attribute(s) in two tables
 - Natural join
 - Equi-join in which redundant columns are removed from the result set
 - Outer join
 - Returned relation includes rows that are missing from one of the original tables

Relational Operators: Union

- Merges two relations
- Result is a set that contains all rows in relation 1 and all rows in relation 2
- Useful for combining subsets
- `SELECT *`
`FROM Protein_Sequence`
`UNION`
`SELECT *`
`FROM Nucleotide_Sequence`

Relational Operators: Union


Protein_Sequence

Biopol_ID	Sequence
Protein 1	ALVCYFMIEGD....
Protein 2	KLMIKAGGKLV....

Nucleotide_Sequence

Biopol_ID	Sequence
DNA 1	ATTGCATTAGC....
DNA 2	GCGGTATGCC....

Get a list of all sequences



Biopol_ID	Sequence
Protein 1	ALVCYFMIEGD....
Protein 2	KLMIKAGGKLV....
DNA 1	ATTGCATTAGC....
DNA 2	GCGGTATGCC....

More likely to be used in combination with projection

Relational Operators: Intersection

- Returns rows common to both relations
- Used to identify overlapping subsets
- `SELECT *`
`FROM Protein_Stock`
`INTERSECT`
`SELECT *`
`FROM Plasmid_Stock`

Relational Operators: Intersection

Protein_Stock

Protein_ID	Stock_location
Protein 1	Box 2
Protein 2	Box 5

Find proteins for which lab has both plasmid and protein prep in stock

Protein_ID	Stock_location
Protein 1	Box 2



Plasmid_Stock

Protein_ID	Stock_location
Protein 1	Box 2
Protein 3	Box 3

Again, more likely to be used in combination with projection

Relational Operators: Difference

- Subtraction: returns rows found in relation 1 but not in relation 2
- Used to identify non-overlapping subsets
- `SELECT *`
`FROM Protein_Stock`
`EXCEPT`
`SELECT *`
`FROM Plasmid_Stock`

Relational Operators: Difference

Protein_Stock

Protein_ID	Stock_location
Protein 1	Box 2
Protein 2	Box 5

Plasmid_Stock

Protein_ID	Stock_location
Protein 1	Box 2
Protein 3	Box 3

Find proteins for which lab has plasmid but no protein prep in stock (time to make more!)



Protein_ID	Stock_location
Protein 3	Box 3

Again, more likely to be used in combination with projection

Relational Operators: Division

- Returns column values from one relation for which there are matching column values for every row in another relation
- A fancy sort of intersection:
 - Finds the subset of relation 1 that “meets criteria” established by relation 2
- No simple SQL implementation. See:
<http://www.developersdex.com/gurus/articles/113.asp>

Relational Operators: Division

Available_Protein

Protein_ID	Protein_Name	Species_Sci_Name	Species_Common_Name
Protein 1	calmodulin	Homo sapiens	human
Protein 2	integrin IV	Bos taurus	cow
Protein 1	calmodulin	Mus musculus	house mouse
Protein 3	ICE	Homo sapiens	human

Find proteins that are available in all species studied in the lab



Protein_ID	Protein_Name
Protein 1	calmodulin

Lab_Species

Species_Sci_Name	Species_Common_Name
Homo sapiens	human
Mus musculus	house mouse

Data Integrity

- Data in the database is meant to represent “reality”
- Certain combinations of values are not possible in the real world, so database should exclude them
- Rules apply to base relations
- Three types:
 - Entity Integrity
 - Referential Integrity
 - “Domain Integrity” (other rules)

Entity Integrity

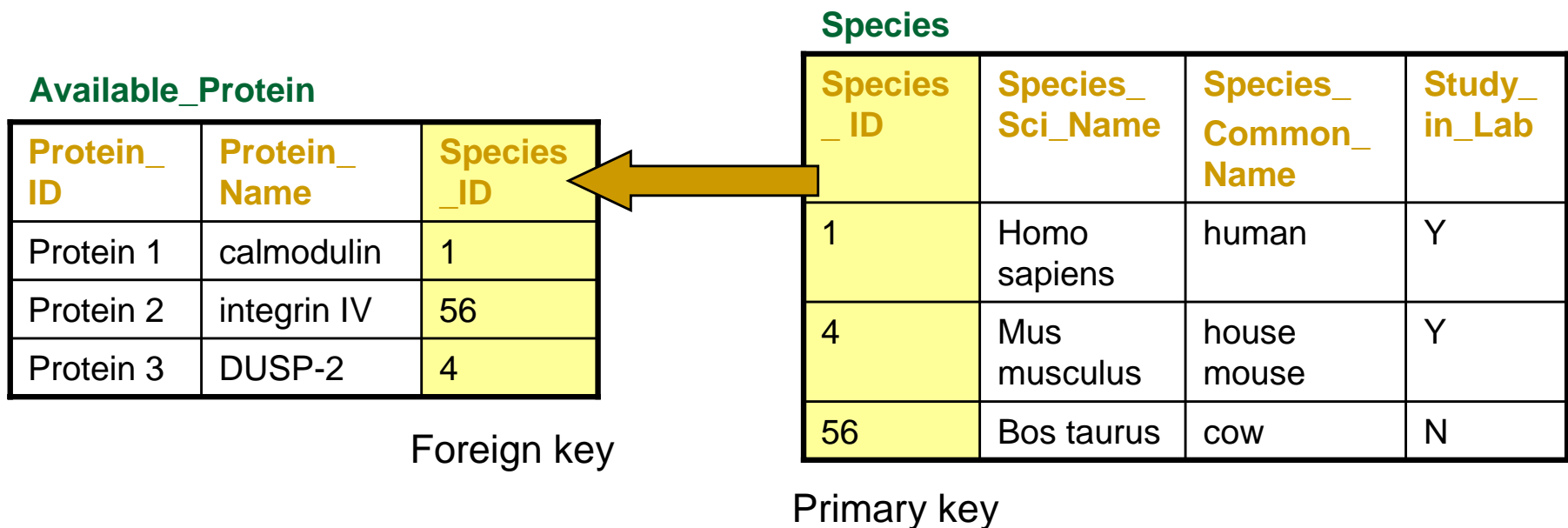
- No part of the primary key may be NULL
- NULL = absence of value
 - Value doesn't exist
 - Value isn't known
- Primary key uniquely identifies a row
 - If part is NULL, it means that we do not know the value
 - It could be a value that is already represented in the table
 - Therefore, we can't uniquely identify the row

Candidate Keys

- Primary key is a special type of candidate key
- Candidate keys
 - A candidate key can uniquely identify each row
 - A candidate key cannot be reduced: i.e., there is no subset of the attributes in the key that also uniquely identify each row
- Alternate keys = candidate keys not chosen to be primary key

Referential Integrity: Foreign Keys

- Links between two related tables are made via foreign keys
- Foreign key = the primary key of a related table



Referential Integrity

- A foreign key value must either
 - Match a primary key value in the referenced table
 - Be NULL

Available_Protein

Protein_ID	Protein_Name	Species_ID
Protein 1	calmodulin	1
Protein 2	integrin IV	56
Protein 3	DUSP-2	4
Protein 4	PTP4B	72

Species

Species_ID	Species_Sci_Name	Species_Common_Name	Study_in_Lab
1	Homo sapiens	human	Y
4	Mus musculus	house mouse	Y
56	Bos taurus	cow	N

Referential Integrity

- Prevents “orphan” rows in child table
 - Child data usually loses significant meaning without parent information
- In practice, allowing a foreign key to be NULL can create problems
- In practice, NULLs can create problems!
 - What does it mean? Value doesn't exist or value unknown?
 - Consider using defaults instead

Domain Integrity

- Attribute integrity
 - Values of an attribute are taken from the specified domain
 - Domain support in database management systems is weak
- Business rules
 - All the other rules the data must follow
 - Implemented in triggers, stored procedures, application logic

Reading and Homework

- Reading for this week's class: Chapters 1-4
- Homework handout

- Reading for next week's class: Chapter 8
- Optional reading: Chapters 5-7