Database Management Systems - I, CS 157A

Introduction and Data Model

Agenda

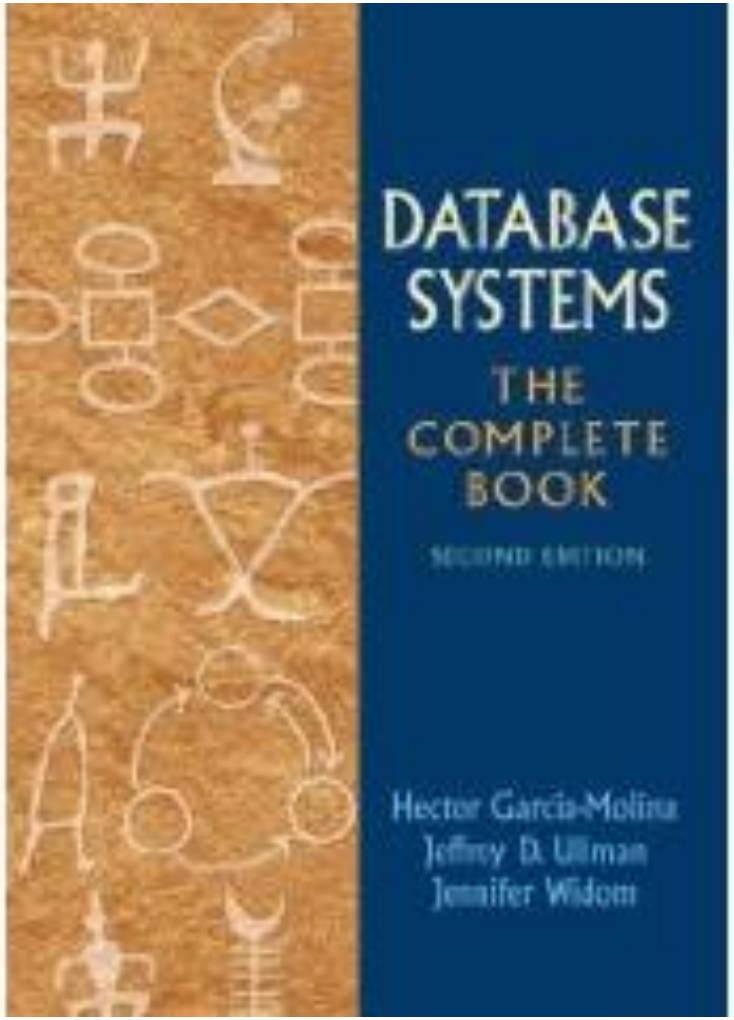
- Course Overview
- Intro to Database
- Data model

Course Overview

- Data model
- Relational algebra
- Database design
- E-R model
- Schema Refinement (Normalization)
- SQL Overview

Textbooks

- **Database Systems: Complete Book**
  by Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom
  *Prentice Hall **2 edition** (June 15, 2008)*
  - ***ISBN-10:*** *0131873253* ***ISBN-13:*** *978-0131873254*


DATABASE
SYSTEMS
THE
COMPLETE
BOOK
SECOND EDITION
Hector García-Molina
Jeffrey D. Ullman
Jennifer Widom

Grading

Homework	15%
Project	25%
Mid-term	30%
Final	30%

Can Not Attend?

- If you
  - cannot finish an assignment, or
  - cannot write any quiz or exam,
- you must send me an email **before** the event
  - explaining the reason and be accepted by myself
  - Late explanation will not be accepted

Emailing

- **Subject** line:
  - must start with **SJSU - CS157A FALL18**,
  - followed by your student ID#, then the actual subject line
- **Body**: always include your full name
- Remind me if you do not receive any reply within 24 hrs, and you think I might have missed it.

Sample Email

FROM: Joe Mohammed

TO: AhmedEzzat@aol.com

Subject: SJSU CS157A FALL18 - A0007 - About assignment #1

Body:

Dear Professor Ezzat,

I am Joe Mohammed (student ID # A0007) in class CS157A, Fall 2018. I have a question about …

Course Home Page

https://sjsu.instructure.com/courses/1270211

All class material are posted including Homework, Project and class PDF presentations

Introduction to Databases

About Databases

- It used to be about boring stuff: employee records, bank records, etc.
- Today, the field covers all the largest sources of data, with many new ideas.
  - Web application.
  - Data mining.
  - Scientific and medical databases.
  - Integrating information.

More about Databases

- You may not notice it, but databases are behind almost everything you do on the Web.
  - Google / Yahoo searches.
  - Queries at Amazon, eBay, etc.

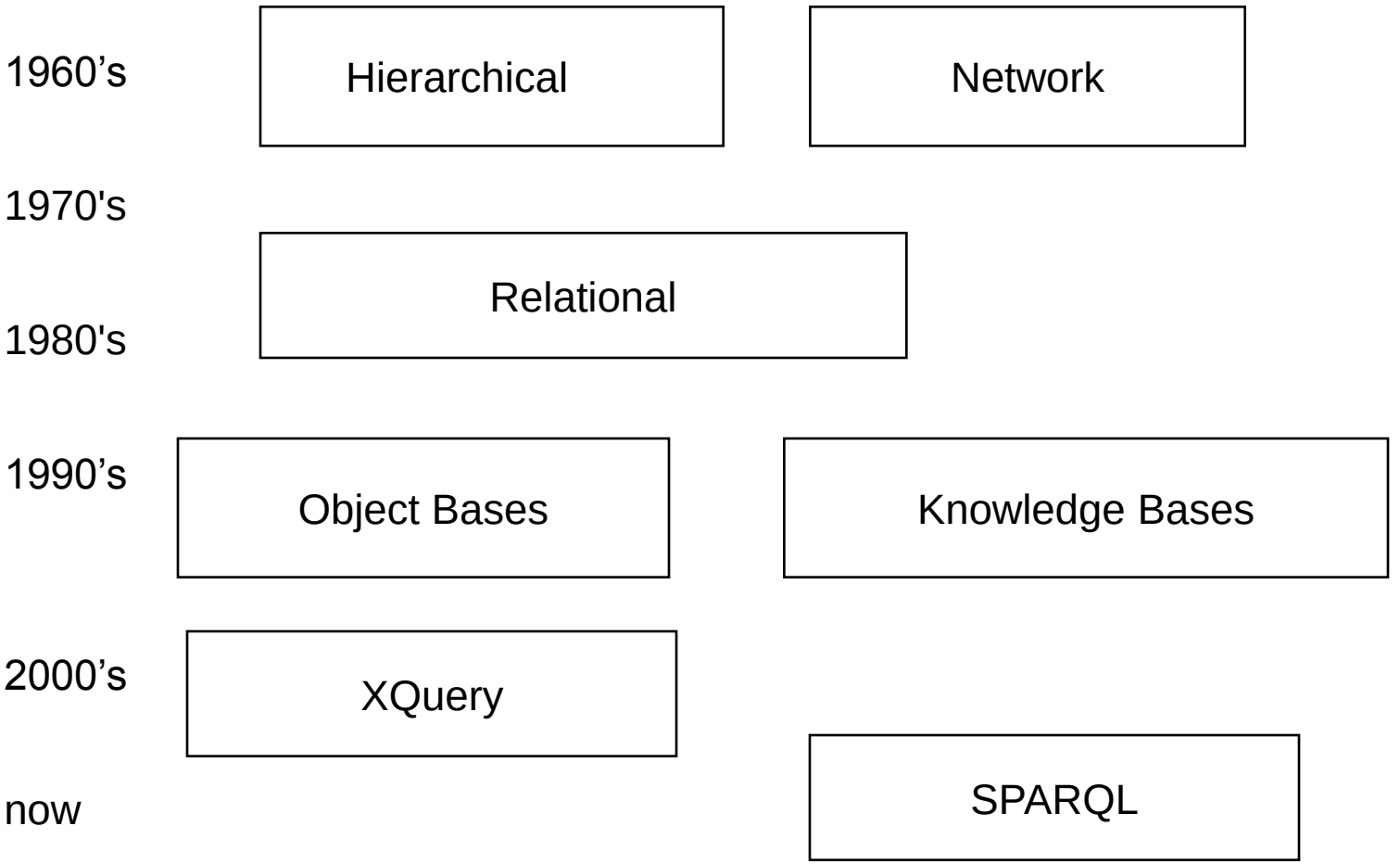
DBMS

- A Database Management System (DBMS) is a software package designed to store and manage databases
- Encompasses most areas of Computer Science
  - For example, OS, algorithms, languages, networking, parallel programming, etc

DBMS Advantages

- Data independence and efficient access
- Data integrity and security
- Concurrent access, high availability
- Transaction, Recovery from crash

Database Evolution



What is a RDBMS System?

- Manages very large amounts of data
- Supports efficient access to very large amounts of data
- Supports concurrent access to very large amounts of data
- Supports secure access to very large amount of data
- Supports atomic (ACID) access to very large amount of data

Interesting applications about RDBMS?

- It used to be about boring stuff like employee records, etc.

- Today, most interesting applications are based on RDBMS or RDBMS is behind it:
  - Web search
  - Data mining
  - Scientific and medical databases
  - Information integration
  - Google search
  - Queries at Amazon, eBay, etc.
  - And more…

High-level Overview of RDBMS (1)

- **Data Definition Language (DDL):** like “type defs” in C and typically is handled by the DBA. DDL manipulates the metadata to create/modify the schema
- **Data Manipulation Language (DML):** used to manipulate existing tables (Insert, Update, Delete - IUD)
- **SQL Processing**: user typically interact with RDBMS through either a query (answer a query) or a DML statements to manipulate the existing content of the database. A significant advantage of RDBMS is that the user specifies the “what” and the query processor decides the “how.”

High-level Overview of RDBMS (2)

A query processor consists of the following two main components:

- **SQL Compiler:** translates SQL statement into internal representation called a "query plan." The query plan is a sequence of actions that will be executed by the execution engine:
  - Query parser: builds a tree structure from the SQL text
  - Query preprocessor: performs semantic checking like relations accessed by the query actually exists, and transform the parse tree into tree of algebraic operators representing the query plan
  - Query optimizer: transform the query plan into available sequence of operations (SQL Operators) on the actual data
- **Execution engine:** executes the sequence of operations in the query plan.

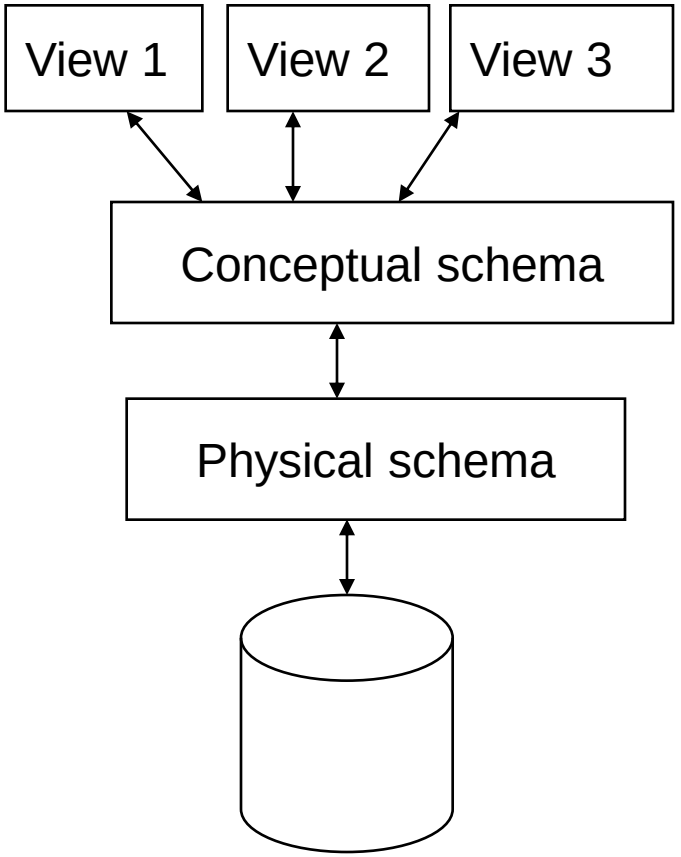
High-level Overview of RDBMS (3)

- **Transaction Processing**: queries and DML statements are grouped into transactions to provide ACID properties:
  - Concurrency control to assure atomicity and isolation of tx
  - Logging and recovery manager to ensure durability

- **Storage and Buffer Management**: Data in the RDBMS reside on secondary storage (disk). To do anything useful work, we need to bring the data into memory (buffer cache). Storage manager controls the placement of data on disk and movement between disk and main memory

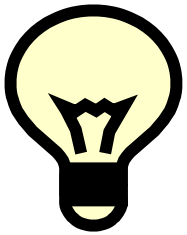
Levels of Abstraction

- **Many views**
  - Describes how users see the data
- **Single conceptual (logical) schema**
  - Define logical structure
- **Single physical schema**
  - Describes the files and indexes used



Data Independence

- Applications insulated from how data is structured and stored on disk
- ***Logical data independence***: Protection from changes in *logical* structure of data
- ***Physical data independence***: protection from changes in *physical* structure of data



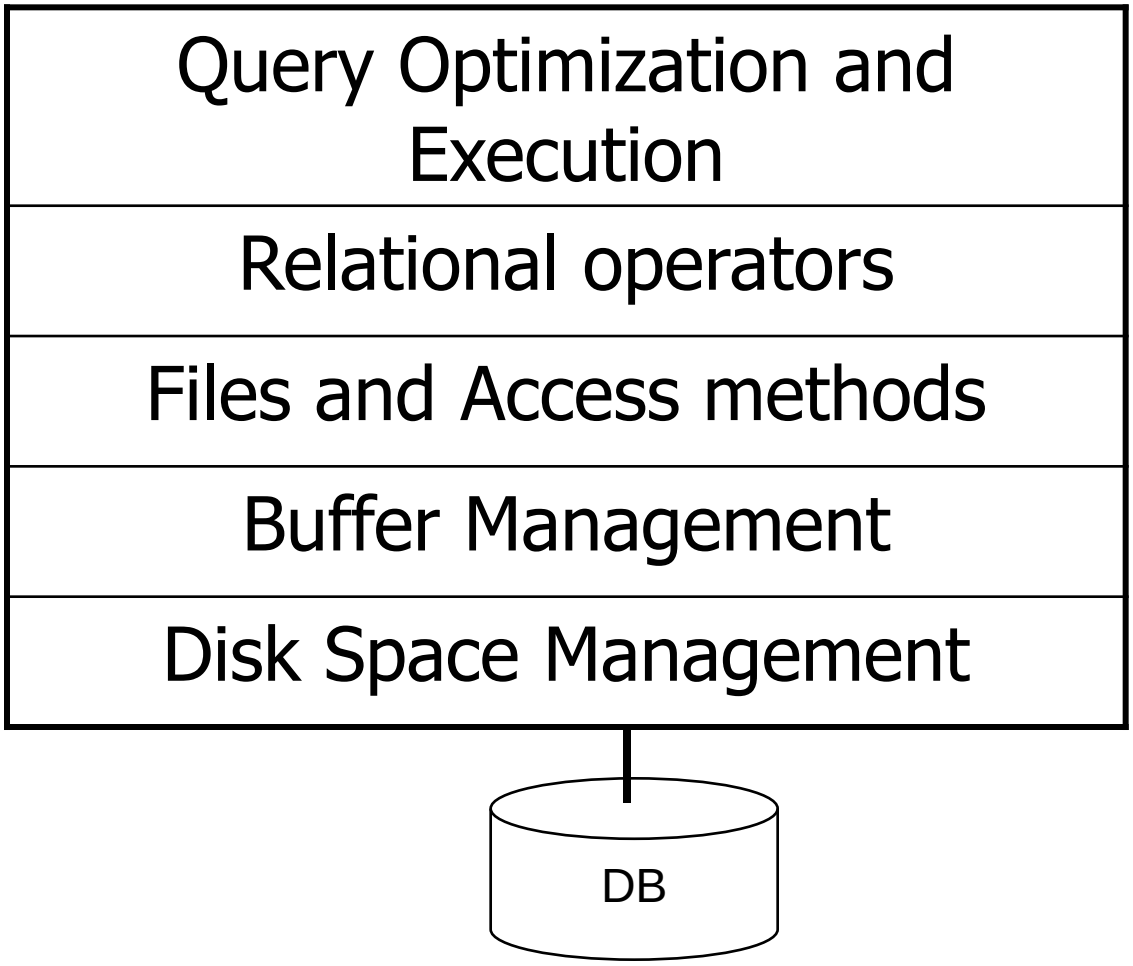
One of the most important benefits of using a DBMS

Databases Design/Administration

- Designs logical / physical schemas
- Handles security and authorization
- Data availability, crash recovery
- Database tuning as needs evolve

Must understand how a DBMS works!

Structure of a DBMS

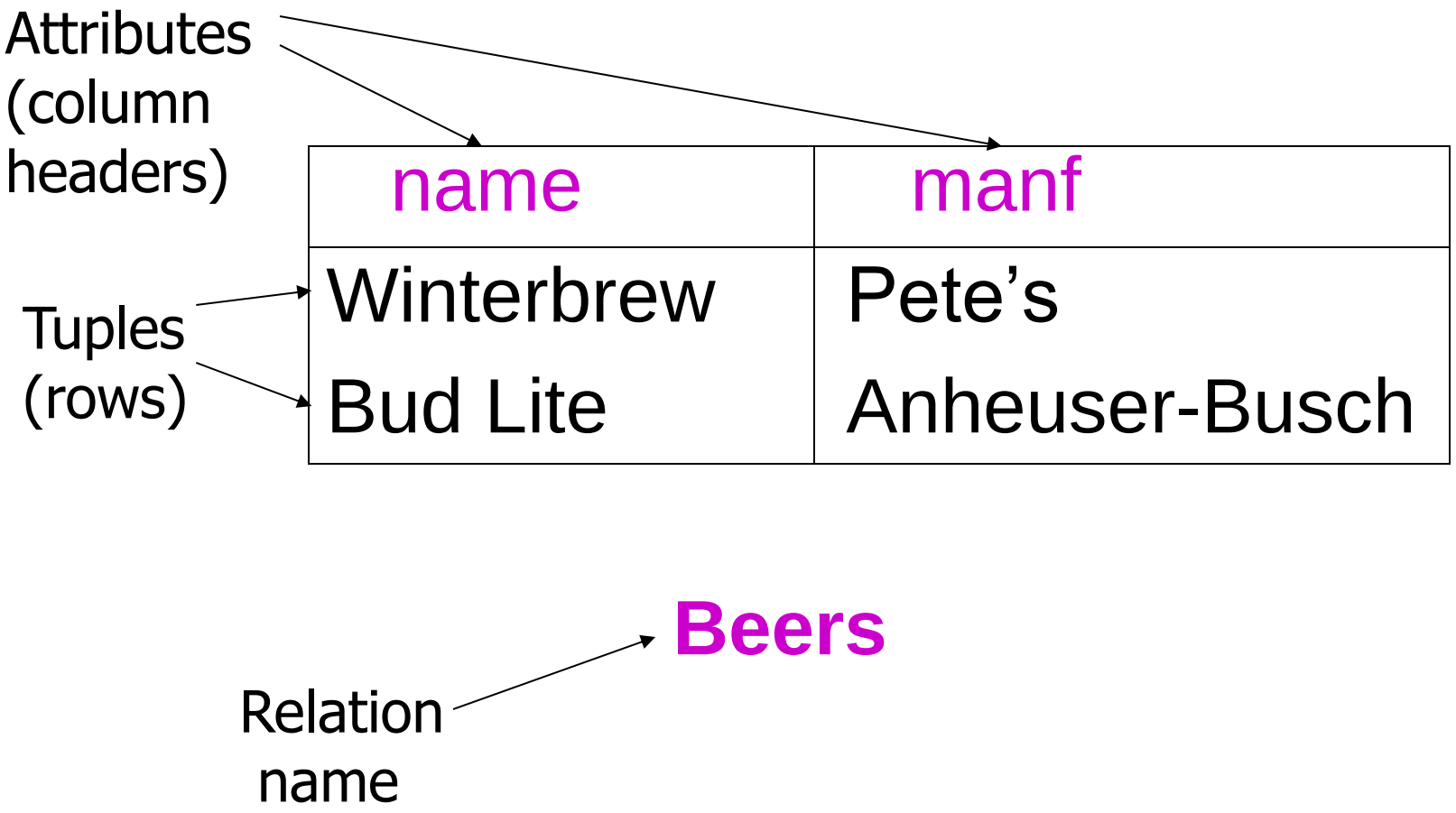


SQL Data Model

What is a Data Model?

1. **Mathematical representation of data**:
   - **Examples**: relational model = tables;
     semi-structured model = trees/graphs.
2. Operations on data.
3. Constraints.

A Relation is a Table



Schemas

- *Relation schema* = relation name and attribute list.
  - **Optionally**: types of attributes.
  - **Example**: Beers(name, manf) or Beers(name: string, manf: string)
- *Database* = collection of relations.
- *Database schema* = set of all relation schemas in the database.

Why Relations?

- Very simple model.
- ***Often*** matches how we think about data.
- Abstract model that underlies SQL, the most important database language today.

Our Running Example

Beers(name, manf)

Bars(name, addr, license)

Drinkers(name, addr, phone)

Likes(drinker, beer)

Sells(bar, beer, price)

Frequents(drinker, bar)

- Underline = *key* (tuples cannot have the same value in all key attributes – Primary key is unique).
  - Excellent example of a constraint.

Database Schemas in SQL

- SQL is primarily a query language, for getting information from a database.
- But SQL also includes a *data-definition* component for describing database schemas.

Creating (Declaring) a Relation

- Simplest form is:

  CREATE TABLE <name> (

  <list of elements>

  );

- To delete a relation:

  DROP TABLE <name>;

Elements of Table Declarations

- Most basic element: an attribute and its type.
- The most common types are:
  - **INT** or INTEGER (synonyms).
  - **REAL** or FLOAT (synonyms).
  - **CHAR(*n*)** = fixed-length string of *n* characters.
  - **VARCHAR(*n*)** = variable-length string of up to *n* characters.

Example: Create Table

- **Simple Table Declaration:**
  - Simplest form is:

```
CREATE TABLE <name> (
        title     CHAR(100),
        year      INT,
        attr3     VARCHAR(20),
        ….
);
```

  - **To delete a relation:**

```
DROP TABLE <name>;
```

Example: Create Table (Contd.)

- **Modifying Relation schemas:**
  - ALTER TABLE <name> ADD phone CHAR(16);
  - ALTER TABLE <name> DROP phone;

- **Default values:**
  - Title CHAR(100) DEFAULT “UNKNOWN” ← column declaration
  - ALTER TABLE <name> ADD phone CHAR(16) DEFAULT ‘unlisted’

- **Declaring keys:**
  - Key is an attribute or list of attributes
  - **Key types:** PRIMARY KEY or UNIQUE
  - The above keys says no 2 tuples of a relation will have same key

SQL Values

- Integers and reals are represented as you would expect.
- Strings are too, except they require single quotes.
  - Two single quotes = real quote, e.g., `'Joe''s Bar'`.
- Any value can be NULL.

Dates and Times

- DATE and TIME are types in SQL.
- The form of a date value is:

  DATE 'yyyy-mm-dd'

  - Example: **`DATE`** `'2011-09-10'` for Sept. 10, 2011.

Times as Values

- The form of a time value is:

  TIME ’hh:mm:ss’

with an optional decimal point and fractions of a second following.

  - Example: **`TIME`** `’15:30:02.5’` = two and a half seconds after 3:30PM.

Declaring Keys

- An attribute or list of attributes may be declared PRIMARY KEY or UNIQUE.
- Either says that no two tuples of the relation may agree in all the attribute(s) on the list.
- There are a few distinctions to be mentioned later.

Declaring Single-Attribute Keys

- Place PRIMARY KEY or UNIQUE after the type in the declaration of the attribute.
- **Example:**

```
CREATE TABLE Beers (
    name    CHAR(20) UNIQUE,
    manf    CHAR(20)
);
```

Declaring Multi-attribute Keys

- A key declaration can also be another element in the list of elements of a CREATE TABLE statement.
- This form is essential if the key consists of more than one attribute.
  - May be used even for one-attribute keys.

Example: Multi-attribute Key

- The bar and beer together are the key for Sells:

```
CREATE TABLE Sells (
        bar         CHAR(20),
        beer        VARCHAR(20),
        price       REAL,
        PRIMARY KEY (bar, beer)
);
```

PRIMARY KEY vs. UNIQUE

1. There can be only one PRIMARY KEY for a relation, but several UNIQUE attributes.
2. No attribute of a PRIMARY KEY can ever be NULL in any tuple. But attributes declared UNIQUE may have NULL's, and there may be several tuples with NULL.

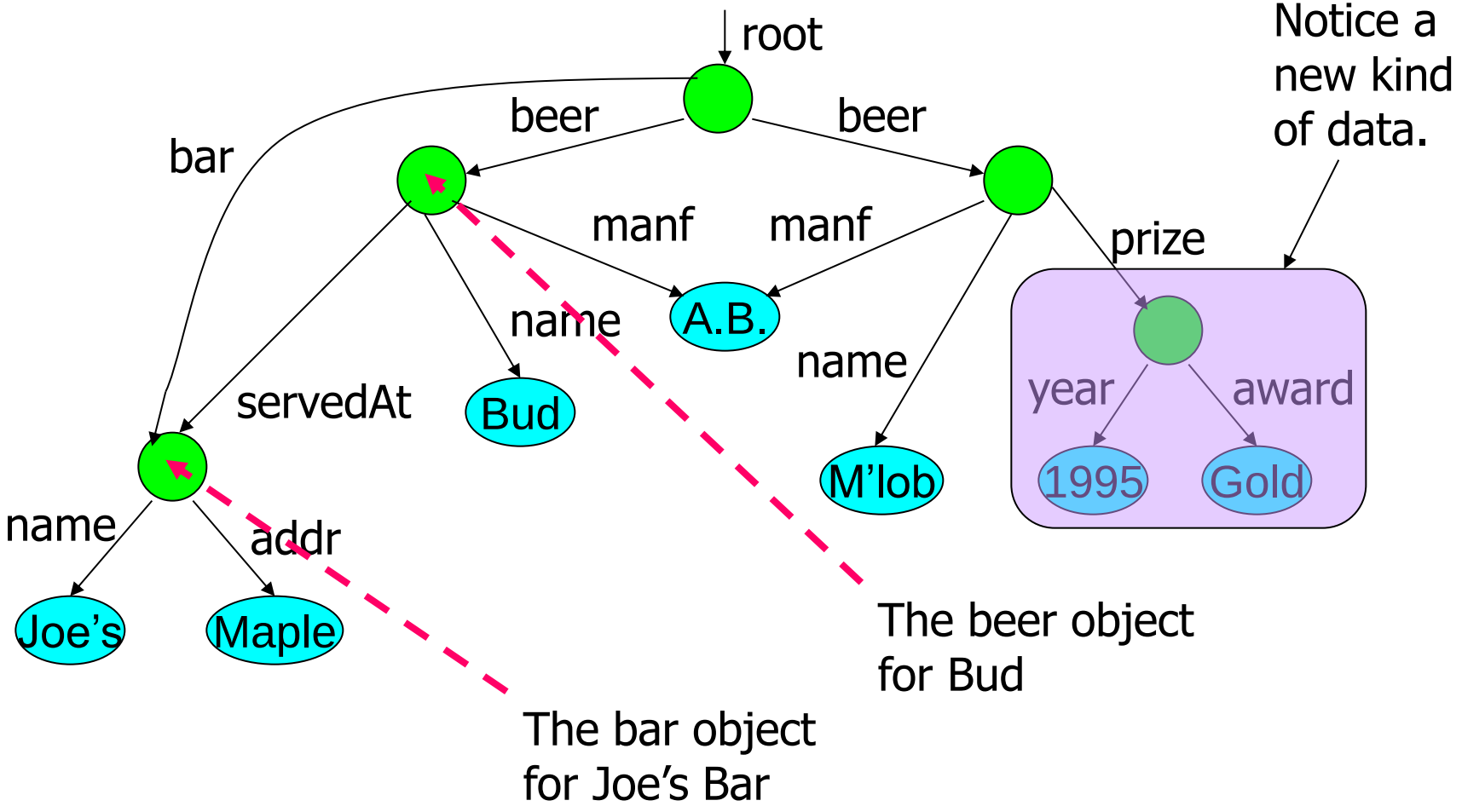
Semi-structured Data

- Another data model, based on trees.
- Motivation: flexible representation of data.
- Motivation: sharing of *documents* among systems and databases.

Graphs of Semi-structured Data

- Nodes = objects.
- Labels on arcs (like attribute names).
- Atomic values at leaf nodes (nodes with no arcs out).
- Flexibility - no restrictions on:
  - Labels out of a node.
  - Number of successors with a given label.

Example: Data Graph



XML

- XML = *Extensible Markup Language*.
- While HTML uses tags for formatting (e.g., “italic”), XML uses tags for semantics (e.g., “this is an address”).
- Key idea: create tag sets for a domain (e.g., genomics), and translate all data into properly tagged XML documents.

XML Documents

- Start the document with a *declaration*, surrounded by <?xml … ?> .
- Typical:

```
<?xml version = “1.0” encoding =
  “utf-8” ?>
```

- Balance of document is a *root tag* surrounding nested tags.

Tags

- Tags, as in HTML, are normally matched pairs, as <FOO> … </FOO>.
  - Optional single tag <FOO/>.
- Tags may be nested arbitrarily.
- XML tags are case sensitive.

Example: an XML Document

```
<?xml version = “1.0” encoding = “utf-8” ?>
<BARS>
   <BAR><NAME>Joe’s Bar</NAME>
        <BEER><NAME>Bud</NAME>
              <PRICE>2.50</PRICE></BEER>
        <BEER><NAME>Miller</NAME>
              <PRICE>3.00</PRICE></BEER>
   </BAR>
   <BAR> …
</BARS>
```

A NAME subobject

A BEER subobject

Attributes

- Like HTML, the opening tag in XML can have atttribute = value pairs.
- Attributes also allow linking among elements.

Bars, Using Attributes

```
<?xml version = “1.0” encoding = “utf-8” ?>
<BARS>
   <BAR name = “Joe’s Bar”>
      <BEER name = “Bud” price = 2.50 />
      <BEER name = “Miller” price = 3.00 />
   </BAR>
   <BAR> …
</BARS>
```

name and price are attributes

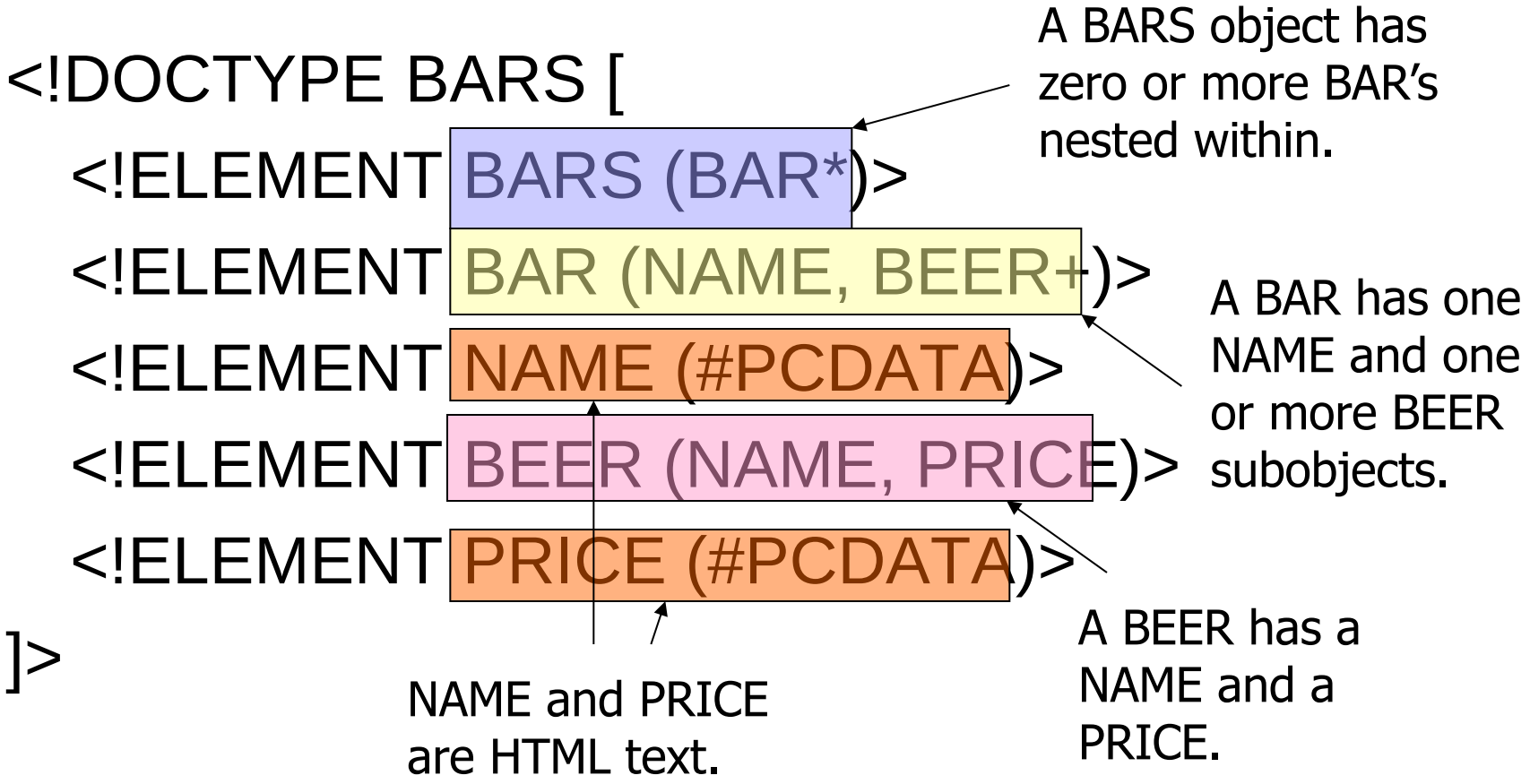
Notice Beer elements have only opening tags with attributes.

DTD's (Document Type Definitions)

- A grammatical notation for describing allowed use of tags.
- Definition form:

```
<!DOCTYPE <root tag> [
   <!ELEMENT  <name> (<components>) >
   . . . more elements . . .
]>
```

Example: DTD


<!DOCTYPE BARS [
<!ELEMENT BARS (BAR*)>
<!ELEMENT BAR (NAME, BEER+)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT BEER (NAME, PRICE)>
<!ELEMENT PRICE (#PCDATA)>
]>
A BARS object has zero or more BAR's nested within.
A BAR has one NAME and one or more BEER subobjects.
A BEER has a NAME and a PRICE.
NAME and PRICE are HTML text.

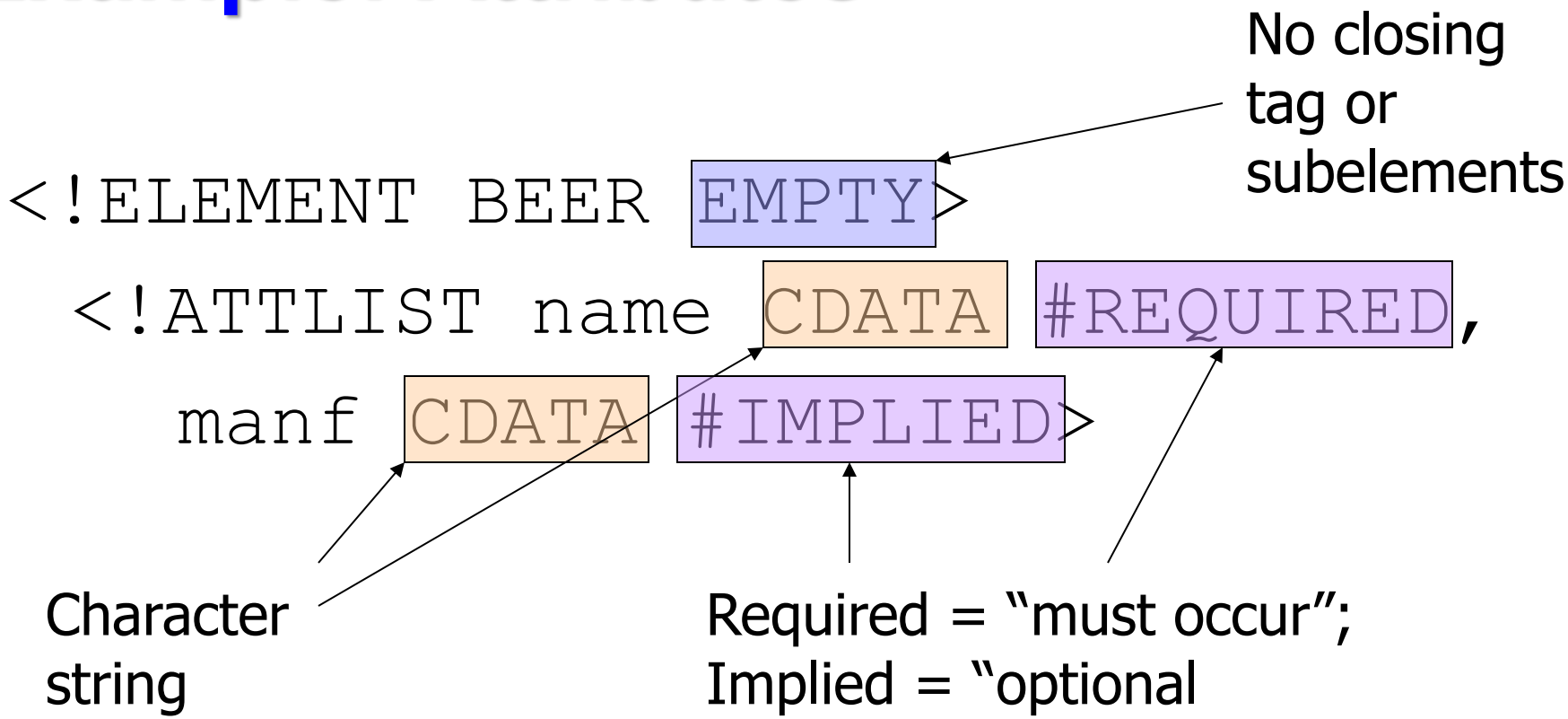
Attributes

- Opening tags in XML can have *attributes*.
- In a DTD,

`<!ATTLIST` *E* `. . . >`

declares an attribute for element *E*, along with its datatype.

Example: Attributes



Example use:

<BEER name="Bud" />

END