Chapter 2
The relational Model of data

Relational model introduction

Contents

* What is a data model?
* Basics of the relational model

Next :

* How to define?
* How to query?
* Constraints on relations

What is a Data Model?

A data model is a notation for describing data or information. It consists of three parts:

- Structure of the data: mathematical representation of data
- Operations on data.
- Constraints.

Two important data models

* The relational model (and object-relational model):

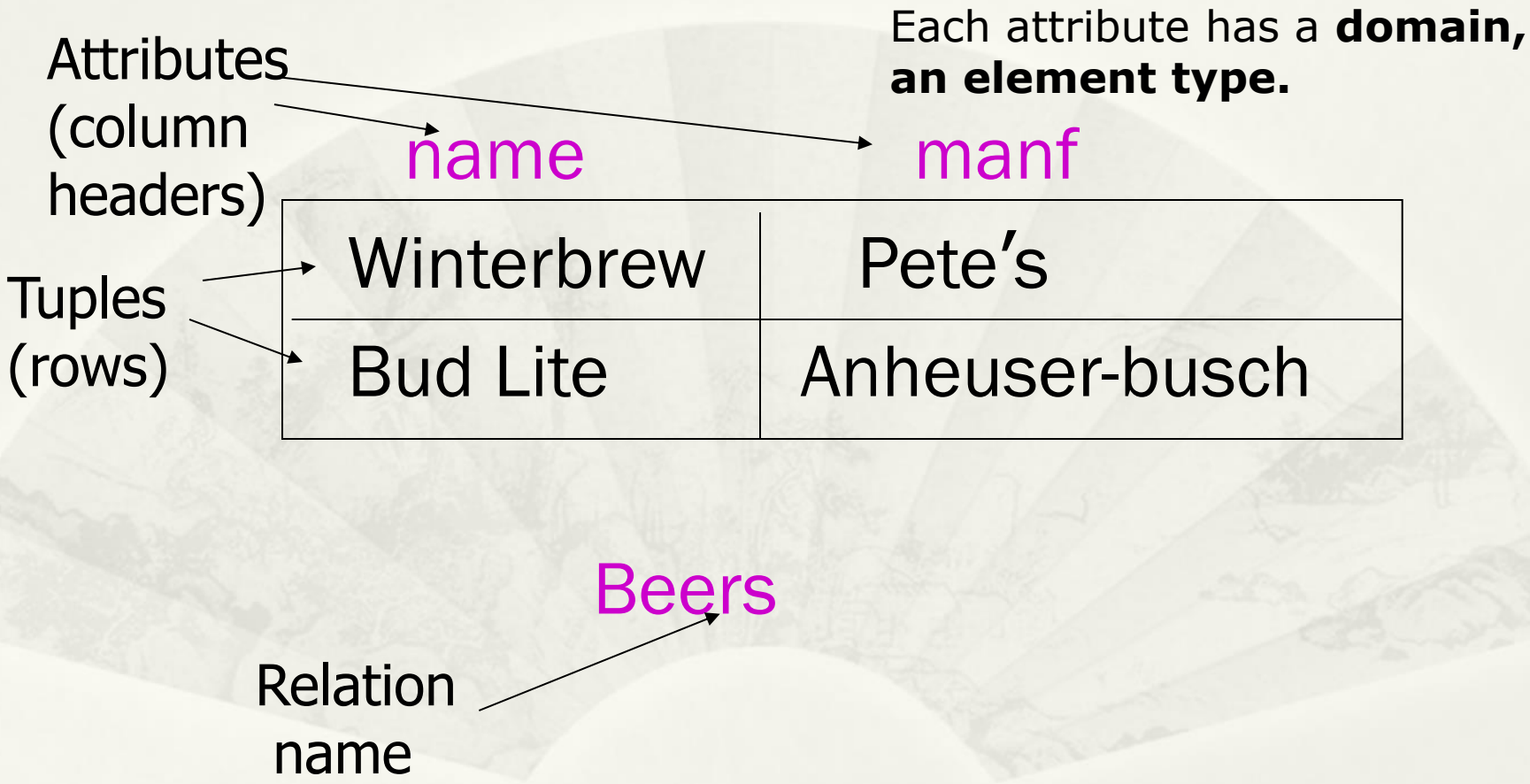
  relational model = tables

* The semistructured-data model

semistructured model = trees/graphs

XML and its related standards.

A relation is a Table



Schemas (模式)

- *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.

Relation Instances (关系实例)

- is current set of rows for a relation schema.
- Example: beer relation

Name	Manf.
Winterblue	Peters
Budlit	A.B.

Dynamic changing

Key of Relations

* There are many constraints on relations
* Key constraints is one of them

For example:

Beer(name, manf)

If name is a key, do not allow two tuples to have the same name.

* Each object should be distinguished in the world

Why Relations?

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

a 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).

Database Schemas in SQL

- SQL is primarily a query language, for getting information from a database.
- 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>;
```

Creating (Declaring) a Relation (cont.)

- To modify schemas

ALTER TABLE <name> ADD <new attribute>

ALTER TABLE <name> DROP <attribute>

Three kinds of table

* Stored relations: tables, a relation that exists in the database, can be modified or queried. real, stored.
* Views: relations defined by a computation. virutal, not really exists.
* Temporary tables: constructed by the SQL processor when it performs. thrown away, not stored.

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

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

SQL Values

- Integers
- reals
- Strings requires *single quotes*.
  - Two single quotes = real quote, e.g., `'Joe''s Bar'`.
- Bit strings of fixed or varying length, BIT(n) means bit string of length n
- Any value can be NULL.
- Boolean: true, false, unknown

Dates and Times in SQL

- The form of a date value is:

  DATE 'yyyy-mm-dd'

  Example: `DATE '2007-09-30'` for Sept. 30, 2007.

- The form of a time value is:

  TIME 'hh:mm:ss'

  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.
- Meaning: no two tuples of the relation may agree in all the attribute(s) on the list.
- PRIMARY KEY or UNIQUE attributes can be declared when creating a table.

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 Multiattribute 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: Multiattribute 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

In **a table** declaration:

1. PRIMARY KEY : only one PRIMARY KEY , No attribute of a PRIMARY KEY can ever be NULL in any tuple.
2. UNIQUE: several UNIQUE attributes, may have NULL's values.

Other Attributes Properties

- NOT NULL = every tuple must have a real value for this attribute. i.e. the value for this attribute may never be NULL.
- DEFAULT value = says that if there is no specific value known for this attribute's component in some tuple, use the stated <value>.

Example: Default Values

```
CREATE TABLE Drinkers (
   name CHAR(30) PRIMARY KEY,
   addr CHAR(50)
        DEFAULT '123 Sesame St.',
   phone CHAR(16)
);
```

Effect of Defaults

- insert the fact that Sally is a drinker, but we know neither her address nor her phone.

```
INSERT INTO Drinkers(name)
VALUES('Sally');
```

Effect of Defaults (cont.)

- What tuple appears in Drinkers?

name	addr	phone
'Sally'	'123 Sesame St'	NULL

- If we had declared phone NOT NULL, this insertion would have been rejected.

Semistructured Data

Based on **trees.**

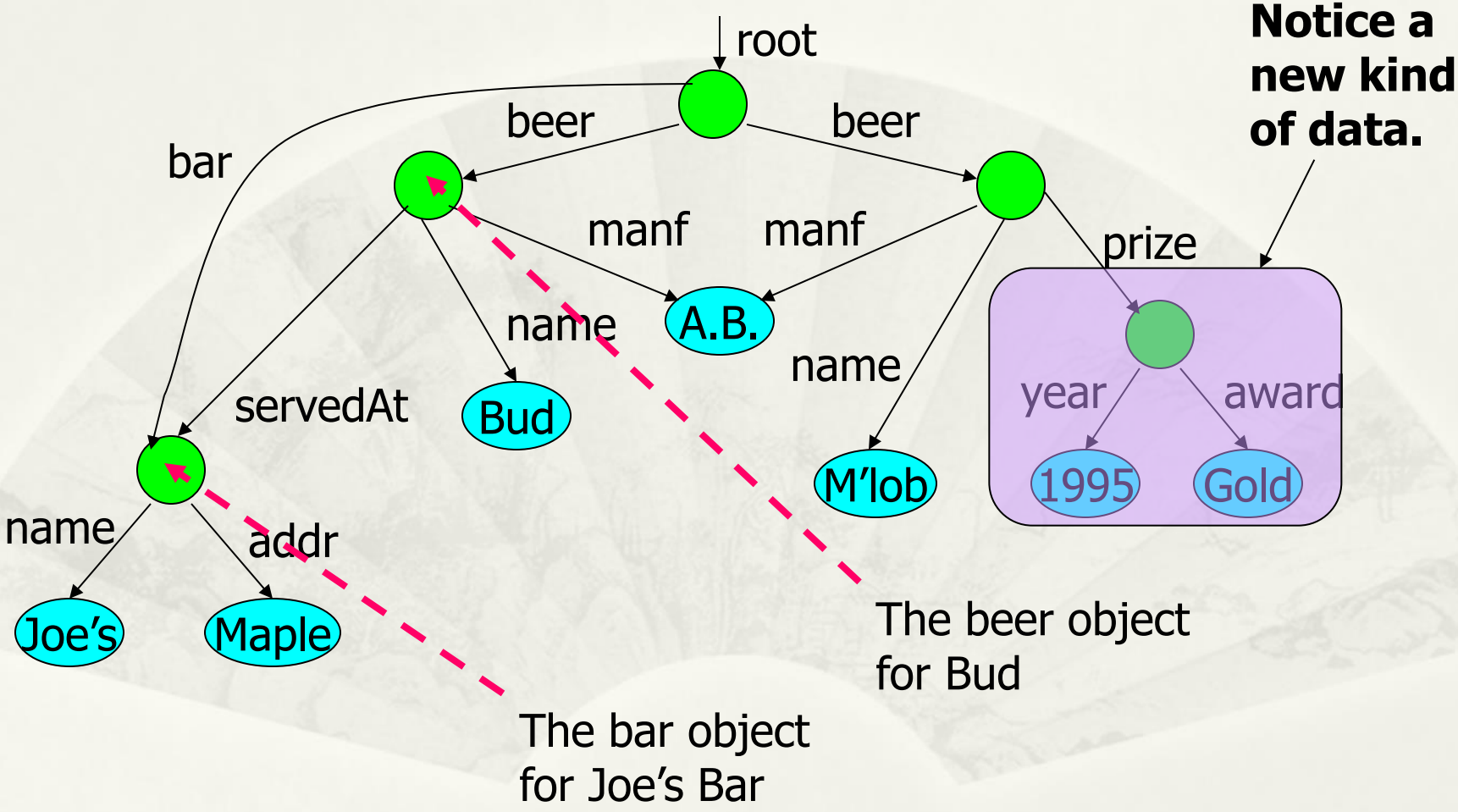
Motivation:

* flexible representation of data.
* sharing of *documents* among systems and databases.

Graphs of Semistructured Data

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

Example: Data Graph



JavaScript Object Notation (JSON)

- Standard for “serializing” data objects
- Human-readable, useful for data interchange
- Useful for representing and storing semistructured data

JSON example

```
{“Beers”:
  [ {“name”: “Bud”,
      “manf”: “A.B.”,
      “price”: 13},
    {“name”: “Mobel”,
      “manf”: “A.B.”,
      “Prize”: {“year”: 1995,
                “award”:”gold”}
  ] }
```

Basic constructs (recursive)

- Base values

number, string, boolean, …

- Objects { }

sets of label-value pairs

- Arrays [ ]

lists of values

Relational Model versus JSON

	Relational	JSON
Structure	Tables	Nested sets, array
schema	Fixed in advance	Flexible, self descripting
Queries	Simple expressive language	Not widely used
Ordering	none	arrays
Implementation	Native system	NOSQL system

XML versus JSON

	XML	JSON
Verbosity	More	Less
Complexity	More	Less
Validity	DTD, XSD, widely used	JSON scheme, not widely used
Prog. Interface	mismatch	More direct
Querying	Xpath,Xquery	Json Path, Json Query

Summarization

Relational model, XML model, JSON notations

A data model consists of three parts:

* Data structure√
* Operations on the data ?
* Constraints ?

Next:

* Relational algebra: operations & constraints.
* Relational algebra: the core of the SQL.