SQL: DATA DEFINITION LANGUAGE

Database Schemas in SQL

- SQL is primarily a query language, for getting information from a database.
  - **Data manipulation language (DML)**
- But SQL also includes a *data-definition* component for describing database schemas.
  - **Data definition language (DDL)**

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

24

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

SQL Values

25

- 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
  - Unless attribute has NOT NULL constraint
  - E.g., price REAL not null,

Dates and Times

26

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

  DATE 'yyyy-mm-dd'
  - Example: DATE '2007-09-30' for Sept. 30, 2007.

Times as Values

27

- 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

28

- 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.

Our Running Example

29

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).

Declaring Single-Attribute Keys

30

- 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

31

- 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

32

- 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

33

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.

Kinds of Constraints

34

- Keys
- Foreign-key, or referential-integrity.
- Domain constraints
  - Constrain values of a particular attribute.
- Tuple-based constraints
  - Relationship among components.
- Assertions: any SQL boolean expression

Foreign Keys

35

- Values appearing in attributes of one relation must appear together in certain attributes of another relation.
- Example: in Sells(bar, beer, price), we might expect that a beer value also appears in Beers.name

Expressing Foreign Keys

36

- Use keyword REFERENCES, either:
  1. After an attribute (for one-attribute keys).
  2. As an element of the schema:

  FOREIGN KEY (<list of attributes>)
  REFERENCES <relation> (<attributes>)
- Referenced attributes must be declared PRIMARY KEY or UNIQUE.

Example: With Attribute

37

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

CREATE TABLE Sells (
  bar      CHAR(20),
  beer     CHAR(20) REFERENCES Beers(name),
  price    REAL );
```

Example: As Schema Element

38

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

CREATE TABLE Sells (
  bar      CHAR(20),
  beer     CHAR(20),
  price    REAL,
  FOREIGN KEY(beer) REFERENCES
     Beers(name));
```

Enforcing Foreign-Key Constraints

39

- If there is a foreign-key constraint from relation *R* to relation *S*, two violations are possible:
  1. An insert or update to *R* introduces values not found in *S*.
  2. A deletion or update to *S* causes some tuples of *R* to "dangle."

Actions Taken --- (1)

40

- Example: suppose R = Sells, S = Beers.
- An insert or update to Sells that introduces a nonexistent beer must be rejected.
- A deletion or update to Beers that removes a beer value found in some tuples of Sells can be handled in three ways…

Actions Taken --- (2)

41

1. *Default* : Reject the modification.
2. *Cascade* : Make the same changes in Sells.
   - Deleted beer: delete Sells tuple.
   - Updated beer: change value in Sells.
3. *Set NULL* : Change the beer to NULL.

Example: Cascade

42

- Delete the Bud tuple from Beers:
  - Then delete all tuples from Sells that have beer = 'Bud'.
- Update the Bud tuple by changing 'Bud' to 'Budweiser':
  - Then change all Sells tuples with beer = 'Bud' to beer = 'Budweiser'.

Example: Set NULL

43

- Delete the Bud tuple from Beers:
  - Change all tuples of Sells that have beer = 'Bud' to have beer = NULL.
- Update the Bud tuple by changing 'Bud' to 'Budweiser':
  - Same change as for deletion.