SQL – DATA DEFINITION LANGUAGE

SFWR ENG 3DB3 – FALL 2016

MICHAEL LIUT (LIUTM@MCMASTER.CA)
DEPARTMENT OF COMPUTING AND SOFTWARE
MCMASTER UNIVERSITY

DATABASE SCHEMAS IN SQL

- SQL is primarily a query language, for getting information from a database.
  - DML: Data Manipulation Language
- But SQL also includes a *data-definition* component for describing database schemas.
  - DDL: Data Definition Language

2

CREATING (DECLARING) A RELATION

- Simply:
  - CREATE TABLE <table_name> (
    <list of elements>
    );
- Deleting a relation:
  - DROP TABLE <table_name>;

3

ELEMENTS OF TABLE DECLARATIONS

- Most basic element: an attribute and its type.
- Common Types:
  - 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.

4

EXAMPLE: CREATE TABLE

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

5

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
  - Unless attribute has NOT NULL constraint

    e.g. price REAL not null,

6

DATES AND TIMES

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

  DATE 'yyyy-mm-dd'

  e.g. DATE '2007-09-30' is September 30th, 2007.

7

TIMES AND VALUES

- The form of a time value is:

  TIME 'hh:mm:ss' → with an optional decimal point for a fraction of a second.

  e.g. TIME '15:30:02.5' = two and a half seconds after 3:30pm.

8

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.

9

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)

10

DECLARING SINGLE-ATTRIBUTE KEYS

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

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

11

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.
  - Could be used for one-attribute keys.

12

EXAMPLE

- 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)
);
```

13

PRIMARY KEY VS. UNIQUE

1. There can be only one PRIMARY KEY for a relation, but several UNIQUE attributes.

1. 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.

14

TYPES OF CONSTRAINTS

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

15

FOREIGN KEYS

- 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

16

EXPRESSING FOREIGN KEYS

- 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.

17

EXAMPLE: WITH ATTRIBUTE

```
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
);
```

18

EXAMPLE: AS SCHEMA ELEMENT

```
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)
);
```

19

ENFORCING FOREIGN-KEY CONSTRAINTS

- 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."

20

ACTIONS TAKEN

- 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.

21

ACTIONS TAKEN

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

22

EXAMPLE: CASCADE

- 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'

23

EXAMPLE: SET NULL

- 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.

24

CHOOSING A POLICY

- When we declare a foreign key, we may choose policies SET NULL or CASCADE independently for deletions and updates.
- Follow the foreign-key declaration by: ON [UPDATE, DELETE][SET NULL CASCADE]
- Two such clauses may be used.
- Otherwise, the default (reject) is used.

25

EXAMPLE: SETTING POLICY

```
CREATE TABLE Sells (
        bar  CHAR(20),
        beer  CHAR(20),
        price  REAL,
        FOREIGN KEY(beer)
                REFERENCES Beers(name)
                ON DELETE SET NULL
                ON UPDATE CASCADE
);
```

26

ATTRIBUTE-BASED CHECKS

- Constraints on the value of a particular attribute.
- Add CHECK(<condition>) to the declaration for the attribute.
- The condition may use the name of the attribute, but any other relation or attribute must be in a sub-query.

27

EXAMPLE: ATTRIBUTE-BASED CHECK

```
CREATE TABLE Sells (
        bar  CHAR(20),
        beer  CHAR(20)  CHECK (
                beer IN (SELECT name FROM Beers)),
        price  REAL  CHECK (price <= 5.00)
);
```

28

TIMING OF CHECKS

- Attribute-based checks are performed only when a value for that attribute is inserted or updated.
  - **Example**: CHECK (price <= 5.00) checks every new price and rejects the modification (for that tuple) if the price is more than $5.
  - **Example**: CHECK (beer IN (SELECT name FROM Beers)) not checked if a beer is deleted from Beers (unlike foreign-keys).

29

TUPLE-BASED CHECKS

- CHECK (<condition>) may be added as a relation-schema element.
- The condition may refer to any attribute of the relation.
  - But other attributes or relations require a sub-query.
- Checked on insert or update only.

30

EXAMPLE: TUPLE-BASED CHECK

- Only Joe's Bar can sell beer for more than $5.

```
CREATE TABLE Sells (
        bar     CHAR(20),
        beer    CHAR(20),
        price       REAL,
        CHECK (bar = 'Joe''s Bar' OR price <= 5.00)
);
```

31

CITATIONS, IMAGES AND RESOURCES

- Based on the slides of Dr. Fei Chiang - http://www.cas.mcmaster.ca/~fchiang/
- Database Management Systems (3rd Ed.), Ramakrishnan & Gehrke
- http://csharpcorner.mindcrackerinc.netdna-cdn.com/UploadFile/BlogImages/0611201603191 0AM/sql.png

32