Declaring Single-Attribute Keys

1

- 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)
);
```

1

Declaring Multiattribute Keys

2

- 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.

2

Example: Multiattribute Key

3

- 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)
);
```

3

PRIMARY KEY vs. UNIQUE

4

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.

4

Kinds of Constraints

5

- 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

5

Foreign Keys

6

- 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

6

Expressing Foreign Keys

7

- 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.

7

Example: With Attribute

8

```
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 );
```

8

Example: As Schema Element

9

```
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));
```

9

Enforcing Foreign-Key Constraints

10

- 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."

10

Actions Taken --- (1)

11

- 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…

11

Actions Taken --- (2)

12

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.

12

Example: Cascade

13

- 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'.

13

Example: Set NULL

14

- 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.

14

Choosing a Policy

15

- 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.

15

Example: Setting Policy

16

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

16

Attribute-Based Checks

17

- 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 name must be in a subquery.

17

Example: Attribute-Based Check

18

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

18

Timing of Checks

19

- 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).

19

Tuple-Based Checks

20

- 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 subquery.
- Checked on insert or update only.

20

Example: Tuple-Based Check

21

- 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)
);
```

21