The Database Language SQL (cont.)

1

Constraints

Commercial relational systems allow much more "fine-tuning" of constraints than do the modeling languages we learned earlier.

In essence: SQL programming is used to describe constraints.

Outline:

- Primary key declarations (already covered).
- Foreign-keys = referential integrity constraints.
- Attribute- and tuple-based checks = constraints within relations.
- SQL Assertions = global constraints

2

Foreign Keys

In relation *R* a clause that "attribute *A* **REFERENCES** *S*(*B*)" says that whatever values appear in the *A* column of *R* must also appear in the *B* column of relation *S*.

NOTE: *B* must be declared as the primary key for *S*.

Example:
Consider
MovieExec(*name*, *address*, *cert#*, *netWorth*)
Studio(*name, address, presC#*)

CREATE TABLE *Studio* (
name **CHAR**(30) **PRIMARY KEY**,
addr **CHAR**(50) **DEFAULT** '123 Sesame St',
presC# **REAL REFERENCES** *MovieExec(cert#)*)

3

Foreign Keys (cont.)

Alternative:

Add another element declaring the foreign key, as:

CREATE TABLE *Studio* (
name **CHAR**(30) **PRIMARY KEY**,
addr **CHAR**(50) **DEFAULT** '123 Sesame St',
presC# **REAL**.
FOREIGN KEY (presC#) **REFERENCES** *MovieExec(cert#)*)

Extra element is essential if the foreign key has more than one attribute…

4

Foreign Key Constraint Violation

Two ways to violate the constraint:

- Insert or update a *Studio* tuple so it refers to a non-existent movie executive
  - Always rejected.
- Delete or update a *MovieExec* tuple that has a *cert#* value some *Studio* tuples refer to.
  - Default: reject.
  - *Cascade*: Ripple changes to referring *Studio* tuples.
  - *Set* **Null**: Change referring tuples to have **NULL** in referring components.

5

Selecting a Policy

Add **ON** [**DELETE**, **UPDATE**] [**CASCADE**, **SET NULL**] to the declaration of foreign key.

Example:
CREATE TABLE *Studio* (
name **CHAR**(30) **PRIMARY KEY**,
addr **CHAR**(50) **DEFAULT** '123 Sesame St',
presC# **REAL**.
FOREIGN KEY (presC#) **REFERENCES** *MovieExec(cert#)*
ON DELETE SET NULL
ON UPDATE CASCADE)

Choosing a "correct" policy is a design decision…

6

Attribute-Based Checks

Follow an attribute by a condition that must hold for that attribute in each tuple of its relation.

- Syntax: **CHECK** (<condition>).
  - Condition may involve the checked attribute.
- Condition is checked only when the associated attribute changes (*i.e.*, an insert or update occurs).

7

Example

CREATE TABLE *Studio* (
name **CHAR**(30) **PRIMARY KEY**,
addr **CHAR**(50) **DEFAULT** '123 Sesame St',
presC# **REAL CHECK** (*presC#* >= 100000),
FOREIGN KEY (presC#) **REFERENCES** *MovieExec(cert#)*
ON DELETE SET NULL
ON UPDATE CASCADE)

- Check on *presC#* is like a foreign-key constraint, except:
  - The check occurs only when we add a tuple or change the *presC#* in an existing tuple, not when we delete a tuple from *Studio*

8

Tuple-Based Checks

Separate element of table declaration.

- Syntax: like attribute-based check.
- But condition can refer to any attribute of the relation.
  - Or to other relations/attributes in subqueries.
- Checked whenever a tuple is inserted or updated.

9

Example

If a star's gender is male, then his name must not begin with 'Ms.'

CREATE TABLE *MovieStar* (
name **CHAR**(30) **PRIMARY KEY**,
address **CHAR**(255),
gender **CHAR**(1),
birthdate **DATE**,
CHECK (*gender* = 'F' **OR** *name* **NOT LIKE** 'Ms. %'**)**

10

SQL Assertions

- Database-schema constraints
- Checked whenever a mentioned relation changes.
- Syntax:

CREATE ASSERTION < name>
CHECK(<condition>)

11

Example

Consider
MovieExec(*name*, *address*, *cert#*, *netWorth*)
Studio(*name, address, presC#*)

What if we wish to require that no one can become the president of a studio unless their net worth is at least $10,000,000

CREATE ASSERTION *RichPres* **CHECK**
(**NOT EXISTS**(**SELECT** *
FROM *Studio, MovieExec*
WHERE *presC#* = *cert#* **AND**
netWorth < 10000000))

Checked whenever *Studio* and *MovieExec* change

12

Triggers

Often called event-condition-action rules.

- *Event* = a class of changes in the DB, *e.g.*, "insertions into Beers."
- *Condition* = a test as in a where-clause for whether or not the trigger applies.
- *Action* = one or more SQL statements.
- Differ from checks or SQL assertions in that:
  - Triggers invoked by the event; the system doesn't have to figure out when a trigger could be violated.
  - Condition not available in checks.

13

Example

Consider

MovieExec(*name*, *address*, *cert#*, *netWorth*)

Whenever we update the netWorth of a movie executive we want the new value to be bigger than the old value.

CREATE TRIGGER *NetWorthTrigger*
AFTER UPDATE OF *netWorth* **ON** *MovieExec*
REFERENCING
OLD ROW AS *OldTuple,*
NEW ROW AS *NewTuple*
FOR EACH ROW
WHEN(*OldTuple.netWorth > NewTuple.netWorth)*
UPDATE *MovieExec*
SET *netWorth = OldTuple.netWorth*
WHERE *cert# = NewTuple.cert#*

14