Chapter 7 Constraints and Triggers

- Keys and foreign keys
- Constraints on attributes and tuples
- Modification of constraints
- Assertions
- **triggers**

Triggers: Motivation

- **Attribute- and tuple-based checks have limited capabilities.**
- **Assertions are sufficiently general for most constraint applications, but they are hard to implement efficiently.**
  - **The DBMS must have real intelligence to avoid checking assertions that couldn't possibly have been violated.**

Triggers: Solution

- **A trigger allows the user to specify when the check occurs.**
- **Like an assertion, a trigger has a general-purpose condition and also can perform any sequence of SQL database modifications.**

Triggers

Often called event-condition-action rules

- **Event= a class of changes in the DB, e.g.: insert, delete**
- **Condition= a test as in a where-clause for whether or not the trigger applies.**
- **Action=one or more SQL statements**

Triggers

Differ from checks, assertions:

- **Triggers are invoked by certain events specified by the database programmer.**
- **Once awakened, the trigger tests a condition.**
- **Only the condition is satisfied, the actions are performed. The action could be any sequence of database operations.**

Example: A Trigger

- **There are many details to learn about triggers.**
- **Here is an example to set the stage.**
- **Instead of using a foreign-key constraint and rejecting insertions into Sells(bar, beer, price) with unknown beers, a trigger can add that beer to Beers, with a NULL manufacturer.**

Example: Trigger Definition

```
CREATE TRIGGER BeerTrig
    AFTER INSERT ON Sells
    REFERENCING NEW ROW AS NewTuple
    FOR EACH ROW
    WHEN (NewTuple.beer NOT IN
        (SELECT name FROM Beers))
    INSERT INTO Beers(name)
        VALUES(NewTuple.beer);
```

The event

The condition

The action

Options: CREATE TRIGGER

- **CREATE TRIGGER <name>**
- **Option:**

CREATE OR REPLACE TRIGGER <name>

  - **Useful if there is a trigger with that name and you want to modify the trigger.**

Options: The Condition

- **AFTER can be BEFORE.**
  - **Also, INSTEAD OF, if the relation is a view.**
    - **A great way to execute view modifications: have triggers translate them to appropriate modifications on the base tables.**
- **INSERT can be DELETE or UPDATE.**
  - **And UPDATE can be UPDATE . . . ON a particular attribute.**

Options: FOR EACH ROW

- **Triggers are either *row-level* or *statement-level*.**
- **FOR EACH ROW indicates row-level; its absence indicates statement-level.**
- **Row level triggers are executed once for each modified tuple.**
- **Statement-level triggers execute once for an SQL statement, regardless of how many tuples are modified.**

Options: REFERENCING

- **INSERT statements imply a new tuple (for row-level) or new set of tuples (for statement-level).**
- **DELETE implies an old tuple or table.**
- **UPDATE implies both.**
- **Refer to these by**

[NEW OLD][ROW TABLE] AS <name>

Options: The Condition

- **Any boolean-valued condition is appropriate.**
- **It is evaluated before or after the triggering event, depending on whether BEFORE or AFTER is used in the event.**
- **Access the new/old tuple or set of tuples through the names declared in the REFERENCING clause.**

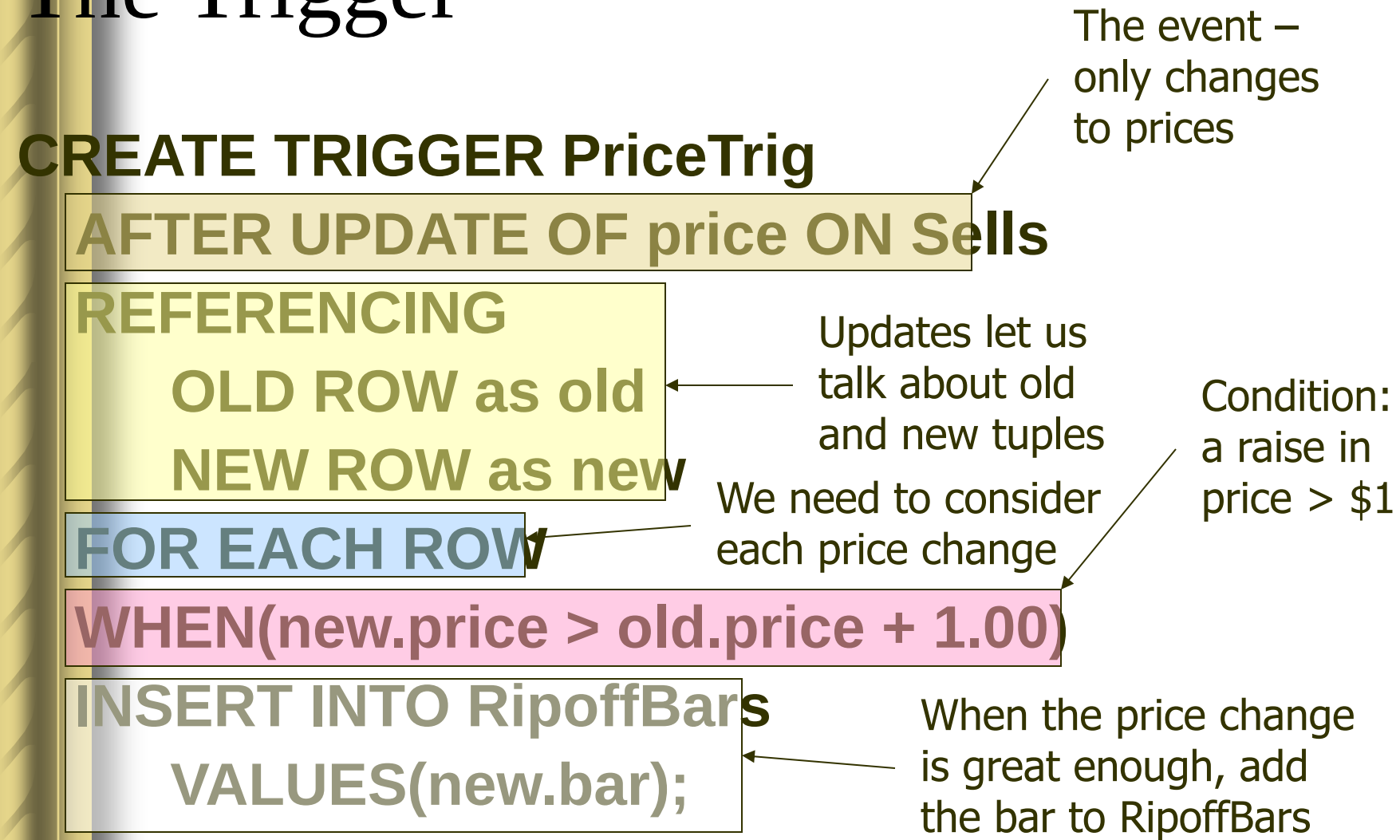
Options: The Action

- **There can be more than one SQL statement in the action.**
  - **Surround by BEGIN . . . END if there is more than one.**
- **But queries make no sense in an action, so we are really limited to modifications.**

Another Example

- **Using Sells(bar, beer, price) and a unary relation RipoffBars(bar) created for the purpose, maintain a list of bars that raise the price of any beer by more than $1.**

The Trigger


CREATE TRIGGER PriceTrig
AFTER UPDATE OF price ON Sells
REFERENCING
OLD ROW as old
NEW ROW as new
FOR EACH ROW
WHEN(new.price > old.price + 1.00)
INSERT INTO RipoffBars
VALUES(new.bar);
The event – only changes to prices
Updates let us talk about old and new tuples
We need to consider each price change
Condition: a raise in price > $1
When the price change is great enough, add the bar to RipoffBars

Summary

- **Key constraints**
- **Referential Integrity Constraints**
- **Value-based ,Tuple-based Check Constraints**
- **Assertions**
- **Triggers**
- **Invoking time**