

# Triggers

# Active Databases

- The database is monitoring the modification operations (I/U/D) and (re)acts
  - Enforce **CONSTRAINTS**
    - When: for every operation
    - How: limited operations imposed by the database
  - Execute **TRIGGERS** ([https://sqlite.org/lang\\_createtrigger.html](https://sqlite.org/lang_createtrigger.html))
    - *Event-Condition-Action (ECA)* rules
    - When: only when *Event* satisfies *Condition*
    - How: fully customizable programmer *Action*

# Example 1

CREATE TRIGGER insertPC\_no\_exists **BEFORE INSERT** ON PC

**FOR EACH ROW**

**EVENT:** I/U/D operation

**WHEN** (NOT EXISTS (

**BEFORE** or **AFTER**

select \*

**CONDITION:** anything that goes in WHERE

from Product p, PriceRange pr

**I/U/D operation is performed  
independent of the trigger execution**

where p.model = **NEW.model**

and p.maker = pr.maker

and p.type = pr.type))

**FOR EACH ROW**

**FOR EACH STATEMENT:** **not in SQLite**

**BEGIN**

insert into PriceRange      **ACTION:** I/U/D operation(s)

select maker, type, **NEW.price**, **NEW.price**

from Product

**NEW:** the tuple that gets inserted

where model = **NEW.model**;

**END**

# Example 2

```
CREATE TRIGGER deleteLaptop_all AFTER DELETE ON Laptop
FOR EACH ROW
WHEN (OLD.price = (select maxPrice from Product p, PriceRange pr
                     where p.model = OLD.model and p.maker = pr.maker and p.type = pr.type)
      AND OLD.price = (select minPrice from Product p, PriceRange pr
                     where p.model = OLD.model and p.maker = pr.maker and p.type = pr.type))
BEGIN
    delete from PriceRange
    where maker = (select maker
                   from Product p
                   where p.model = OLD.model)
          and type = (select type
                      from Product p
                      where p.model = OLD.model);
END
```

**OLD: the tuple that got deleted**

# Example 3

```
CREATE TRIGGER updatePrinter_min AFTER UPDATE ON Printer
FOR EACH ROW
WHEN (NEW.price < (select minPrice from Product p, PriceRange pr
                      where p.model = NEW.model and p.maker = pr.maker and p.type = pr.type))
BEGIN
    update PriceRange
    set minPrice = NEW.price
    where maker = (select maker
                    from Product p
                    where p.model = NEW.model)
        and type = (select type
                    from Product p
                    where p.model = NEW.model);
END
```

**OLD:** the old value of the tuple that got updated  
**NEW:** the new value of the tuple that got updated

# Example 4

CREATE TRIGGER insertPC\_Maker **INSTEAD OF INSERT** ON  
**PC\_Maker**

**FOR EACH ROW**

**BEGIN**

insert into Product(model, maker, type)

values(NEW.model, NEW.maker, 'pc');

insert into PC

values(NEW.model, NEW.speed, NEW.ram, NEW.hd, NEW.price);

**END**

- Trigger is executed instead of I/U/D operation
- Allows for I/U/D operations on views

# Triggers Summary

- Programmer has complete control
  - How constraints are enforced
  - How modification operations (I/U/D) are handled
- Unexpected interaction with I/U/D operations
- Transform views into base tables
- Implement materialized view maintenance