Chapter 7
Integrity Constraints

Content

- **Introduction**
- Characteristics
- Categories
- Implementation

Introduction

- Integrity constraints
  - Derived from the rules or conditions in
    - The mini world that the database represents
    - The data model
      - Database modifications should not be made arbitrarily because they can lead the database into the “bad” state
  - Defined on a relation or many different relations
  - Are invariant conditions that all instances of relations have to satisfy at any time

Introduction

- Why do we need integrity constraints?
  - Guarantee the coherence of components that make up the database
  - Guarantee the consistency of data
  - Guarantee that the database always represents the context precisely

- Examples
  - The salary of an employee cannot be larger than the salary of the manager (C_1)
  - The supervisor of an employee must be an employee in the company (C_2)

Content

- Introduction
- **Characteristics**
  - Context
  - Content
  - Table of purview
- Categories
- Implementation

Context

- The context of an integrity constraint
  - Relations that are likely to be violated integrity constraints when we perform some modifications

- Example (C_1)
  - The salary of an employee cannot be larger than the salary of the manager
    - Modifications
      - Modify the salary for an employee
      - Assign one more employee to a department
      - Appoint an employee to manage a department
    - Context: EMPLOYEE, DEPARTMENT

Context

- Example (C_2)
  - The supervisor of an employee must be an employee in the company
    - Modifications
      - Change the supervisor of an employee
      - Add one new employee
    - Context: EMPLOYEE

Content

- Are stated by
  - Natural language
    - Easy to understand but lack of the coherence
    - Formal language
      - Condensed, coherent but sometimes difficult to understand
      - Represent via
        - Relational algebra
        - Relational calculus
        - Pseudo code

Content

- Example (C_1)
  - Natural language
    - The salary of an employee cannot be larger than the salary of the manager
  - Formal language

$$\forall t \in \text{EMPLOYEE} ($$
$$\exists u \in \text{DEPARTMENT} (\exists v \in \text{EMPLOYEE} ($$
$$u.\text{MGRSSN} = v.\text{SSN} \wedge$$
$$t.\text{DNO} = u.\text{DNUMBER} \wedge$$
$$t.\text{SALARY} \leq v.\text{SALARY})))$$

Content

- Example (C_2)
  - Natural language
    - The supervisor of an employee must be an employee in the company
  - Formal language

$$\forall t \in \text{EMPLOYEE} (t.\text{SUPERSSN} = \text{null} \vee \exists s \in \text{EMPLOYEE} (t.\text{SUPERSSN} = s.\text{SSN}))$$

Table of purview

- Is used to
  - Determine which modifications need to be checked the integrity constraint when they are performed on some context/relation

- 2 types
  - Table of purview for an integrity constraint
  - Synthesis table
    - Table of purview for many integrity constraints

Table of purview

- For one integrity constraint

Constraint _name	Insert	Delete	Update
Relation 1	+	–	+ (Attribute)
Relation 2	–	+	–
…			
Relation n	–	+	–

(+) Violate the constraint

(–) Not violate the constraint

Table of purview

- Synthesis table

	Constraint 1			Constraint 2						Constraint m		
	I	D	U	I	D	U	…	…	…	I	D	U
Relation 1	+	-	+	+	-	+				+	-	+
Relation 2	-	+	-									
Relation 3	-	-	+							-	+	-
…												
Relation n				-	+	-				-	-	+

(+) Violate the constraint

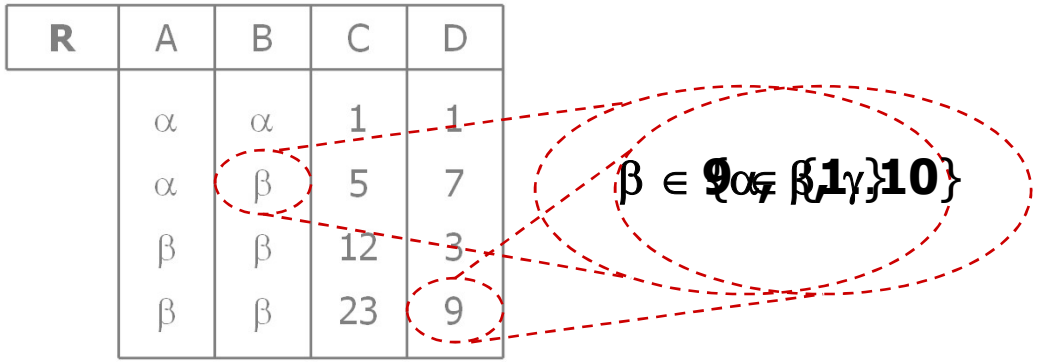
(–) Not violate the constraint

Content

- Introduction
- Characteristics
- **Categories**
  - One relation
    - Domain
    - Correlated tuples
    - Correlated attributes
  - Many relations
    - Reference
    - Correlated tuples
    - Correlated attributes
    - Aggregate attribute
    - Cycle
- Implementation

Domain constraint

- Within each tuple, the value of a certain attribute A must be an atomic value from the domain DOM(A)



- Domain
  - Continuous
  - Concreted

Example 3

- The working hour for one project is not over 60 hours

  - Context: WORKS_ON

  - Content:

$$\forall t \in \text{WORKS\_ON} \; (\; t.\text{HOURS} \leq 60 \;)$$

  - Table of purview:

C_3	Insert	Delete	Update
WORKS_ON	+	–	+ (HOURS)

Example 4

- The sex of an employee is either ‘Male’ or ‘Female’

  - Context: EMPLOYEE

  - Content: $\forall t \in \text{EMPLOYEE}\ (\ t.\text{SEX} \in \{\text{‘Male’}, \text{‘Female’}\}\ )$

    Or

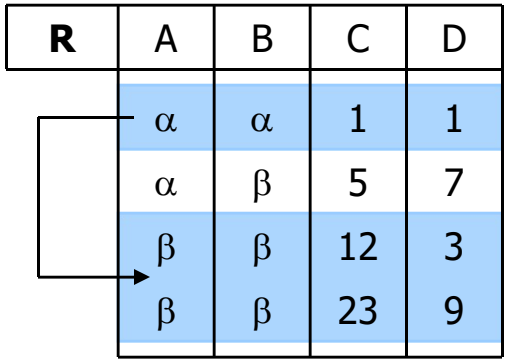
    $\text{DOM}(\text{SEX}) = \{\text{‘Male’}, \text{‘Female’}\}$

  - Table of purview:

C4	Insert	Delete	Update
EMPLOYEE	+	–	+ (SEX)

Correlated tuples

- The existence of one or more tuples depends on the existence of other tuples in the same relation



R	A	B	C	D
	α	α	1	1
	α	β	5	7
	β	β	12	3
	β	β	23	9

- Special cases
  - Primary key constraint
  - Unique constraint

Example 5

- The department name is unique

  - Context: DEPARTMENT

  - Content:

$$\forall t1, t2 \in \text{DEPARTMENT} ($$

$$t1 \neq t2 \Rightarrow t1.\text{DNAME} \neq t2.\text{DNAME})$$

  - Table of perview:

C_5	Insert	Delete	Update
DEPARTMENT	+	–	+ (DNAME)

Example 6

- The maximum number of projects in which an employee participate is 5

  - Context: WORKS_ON

  - Content:

$$\forall t \in \text{WORKS\_ON} \; (\; \text{card}(\{ s \in \text{WORKS\_ON} \mid s.\text{ESSN} = t.\text{ESSN}\}) \leq 5 \;)$$

  - Table of pur view:

C_6	Insert	Delete	Update
WORKS_ON	+	–	+ (ESSN)

Example 7

- COMPETITION(Date, Time, Team, Scores)
- Every match is the competition of two teams

  - Context: COMPETITION

  - Content:

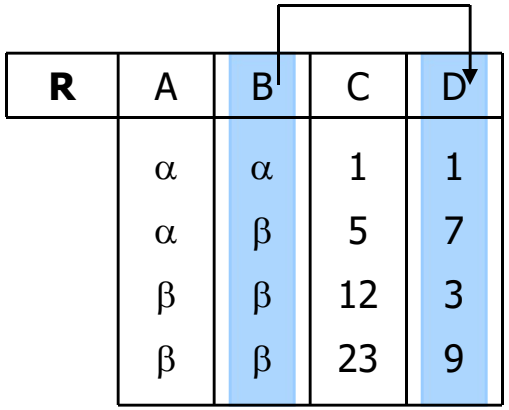
$$\forall t \in \text{COMPETITION} \; (\exists! \; s \in \text{COMPETITION} \; (t \neq s \wedge t.\text{DATE} = s.\text{DATE} \wedge t.\text{TIME} = s.\text{TIME}))$$

  - Table of purview

C7	Insert	Delete	Update
COMPETITION	+	+	+ (DATE, TIME, TEAM)

Correlated attributes

- Constraints among attributes in a relation



Example 8

- An employee does not supervise himself/herself

  - Context: EMPLOYEE

  - Content:

    $\forall t \in \text{EMPLOYEE}\ (\ t.\text{SuperSSN} \neq t.\text{SSN}\ )$

  - Table of purview:

C_8	Insert	Delete	Update
EMPLOYEE	–	–	+ (SUPERSSN)

At the time of adding one new employee,
SUPERSSN has the NULL value

Example 9

- COURSE(CNumber, CName, StartDate, EndDate)
- Each course lasts at least 3 months

  - Context: COURSE

  - Content:

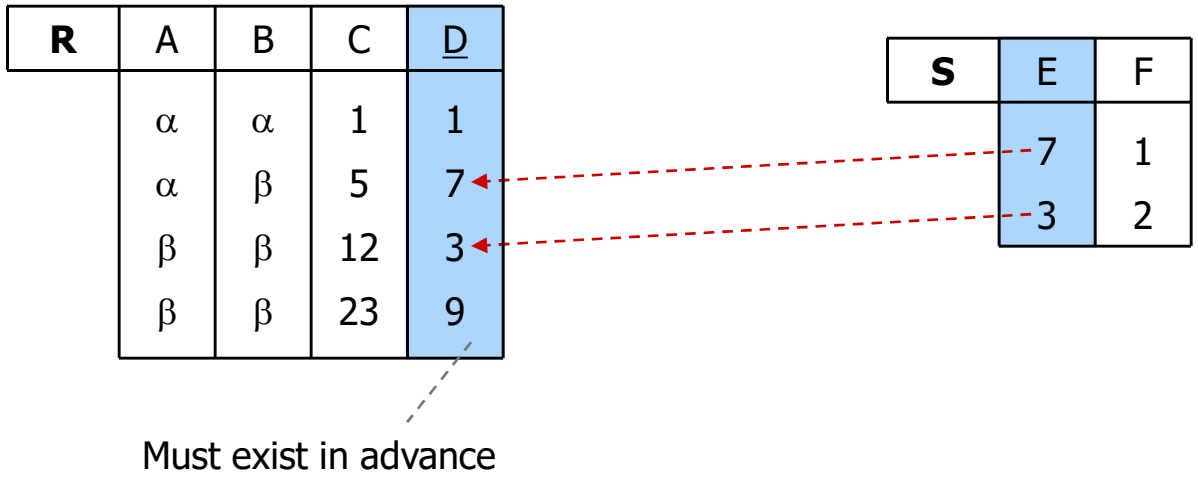
    $\forall t \in \text{COURSE}\ (\ t.\text{STARTDATE} - t.\text{ENDDATE} \geq 3\ )$

  - Table of purview:

C_9	Insert	Delete	Update
COURSE	+	–	+ (STARTDATE, ENDDATE)

Referential constraint

- The value of attributes in some relation must refer to the value of the primary key of other relations



- Special case
  - Foreign key constraint

Example 10

- Every dependent must has a relationship with an employee

  - Context: DEPENDENT, EMPLOYEE
  - Content:

$$\forall t \in \text{DEPENDENT} (\exists s \in \text{EMPLOYEE} (s.SSN = t.ESSN))$$

Or

$$\text{DEPENDENT.ESSN} \subseteq \text{EMPLOYEE.SSN}$$

  - Table of purview:

C_{10}	Insert	Delete	Update
EMPLOYEE	–	+	+ (SSN)
DEPENDENT	+	–	+ (ESSN)

Referential constraint

- Is also called existential dependences
- The context often consists of two relations
  - But in many cases it includes one relation
  - Example (C_2)
    - The supervisor of an employee must be an employee
    - Context: EMPLOYEE
    - Content:

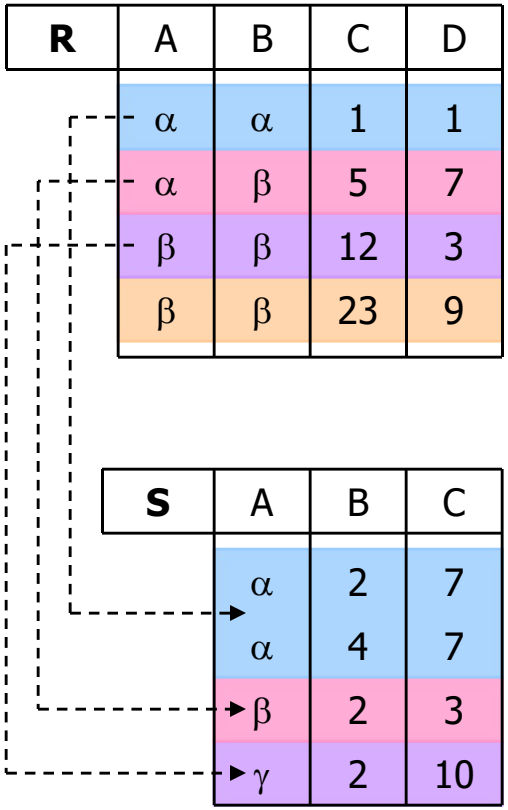
$$\forall t \in \text{EMPLOYEE} \; (\; t.\text{SUPERSSN} = \text{null} \vee \exists s \in \text{EMPLOYEE} \; (t.\text{SUPERSSN} = s.\text{SSN} \;))$$

    - Table of purview

C_2	Insert	Delete	Update
EMPLOYEE	+	+	+ (SSN, SUPERSSN)

Correlated tuples on many relations

- Constraints happen among tuples on many different relations



Example 11

- INVOICE(Number, CustID, Date)
- DETAIL(InvNumber, PNumber, Unit_Price, Quantity)
- Every invoice must have at least one detail

  - Context: INVOICE, DETAIL
  - Content:

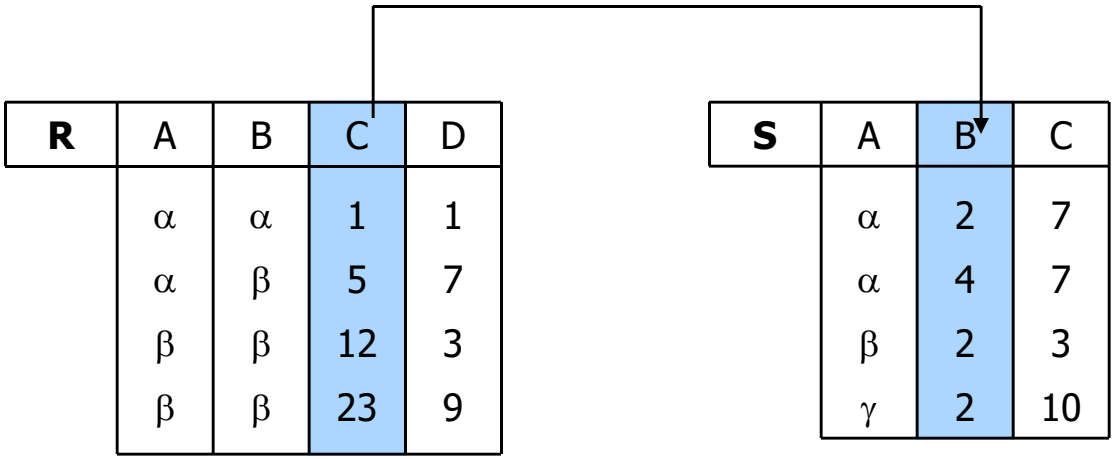
    $\forall t \in \text{INVOICE} (\exists s \in \text{DETAIL} (t.\text{NUMBER} = s.\text{InvNUMBER}))$

  - Table of purview:

C_{11}	Insert	Delete	Update
INVOICE	+	+	+ (NUMBER)
DETAIL	+	+	+ (InvNUMBER)

Correlated attributes on many relations

- Constraints among attributes of many different relations



R	A	B	C	D
	α	α	1	1
	α	β	5	7
	β	β	12	3
	β	β	23	9

S	A	B	C
	α	2	7
	α	4	7
	β	2	3
	γ	2	10

Example 12

- The birth date of a manager is smaller than his/her starting date

  - Context: EMPLOYEE, DEPARTMENT
  - Content:

$$\forall t \in \text{DEPARTMENT} \; (\exists s \in \text{EMPLOYEE} \; (s.\text{SSN} = t.\text{MGRSSN} \wedge t.\text{STARTDATE} > s.\text{BIRTHDAY}))$$

  - Table of purview:

C_{12}	Insert	Delete	Update
EMPLOYEE	–	–	+ (BIRTHDATE, SSN)
DEPARTMENT	+	–	+ (STARTDATE, MGRSSN)

Aggregate attribute

- What is a aggregate attribute?
  - Its value is calculated from values of other attributes

- Aggregate attributes exist in the database
  - Constraints will guarantee the relationship between the aggregate attributes and source attributes

Example 13

- DEPARTMENT(DName, DNumber, MGRSSN, StartDate, Num_Emp)
- The value of Num_Emp of a department is equal to the total number of employees of that department from the relation employee
  - Context: EMPLOYEE, DEPARTMENT
  - Content:

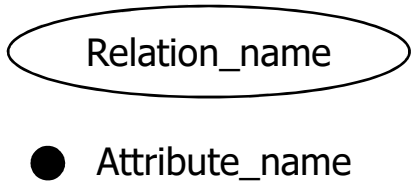
$$\forall t \in \text{DEPARTMENT} \; (\; t.\text{NUM\_EMP} = \text{card}(\{ \; s \in \text{EMPLOYEE} \mid s.\text{DNO} = t.\text{DNUMBER}\}) \;)$$

  - Table of purview:

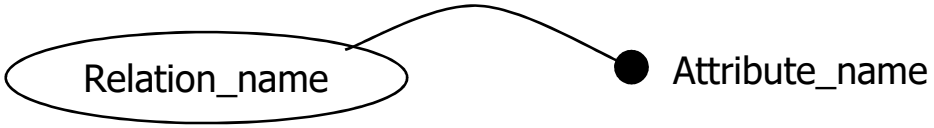
C_{13}	Insert	Delete	Update
EMPLOYEE	+	+	+ (DNO)
DEPARTMENT	–	–	+ (NUM_EMP, DNUMBER)

Cycle

- Database schemas are represented by graphs
  - Node
    - Relation
    - Attribute



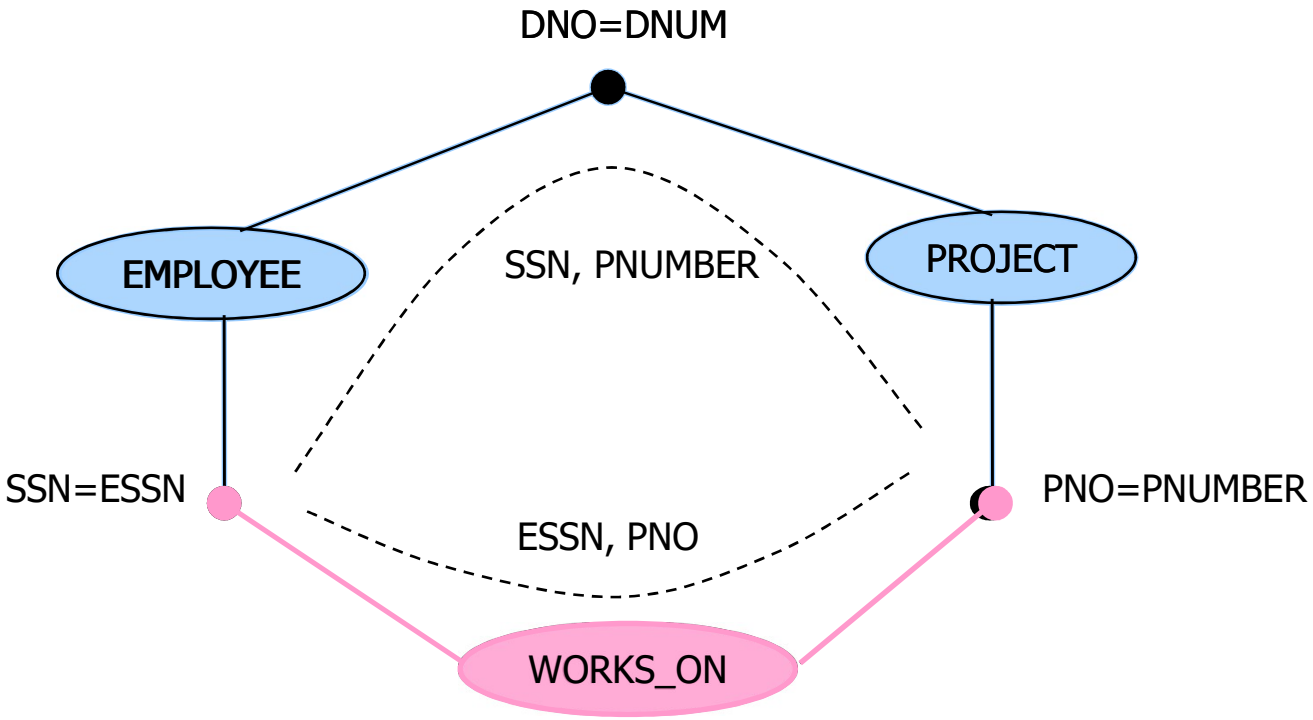
  - Edge
    - Connection between a node and an attribute



- Cycle
  - A graph has a closed walk ~ Database schema has a cycle

Example 14

- Employees are only assigned to projects that are controlled by their department



Example 14

- Employees are only assigned to projects that are controlled by their department
  - Context: EMPLOYEE, PROJECT, WORKS_ON
  - Content:

$$EMP\_PRO \leftarrow EMPLOYEE \bowtie_{DNO=DNUM} PROJECT$$

$$\forall t \in WORKS\_ON\ (\ \exists s \in EMP\_PRO\ (\ t.ESSN = s.SSN \wedge t.PNUM = s.PNUMBER\ ))$$

  - Table of purview:

C_{14}	Insert	Delete	Update
EMPLOYEE	–	–	+ (SSN, DNO)
PROJECT	–	–	+ (PNUMBER, DNUM)
WORKS_ON	+	–	+ (ESSN, PNUM)

Content

- Introduction
- Characteristics
- Categories
- **Implementation**
  - Assertion
  - Trigger
  - Transaction
  - Stored procedure

Implementation

- Constraints are implemented by
  - Primary key
  - Foreign key
  - Check constraint

  - Assertion
  - Trigger
  - Transaction

Assertion

- A boolean-valued SQL expression that must be TRUE at all times
  - Requires programmers to state what must be TRUE

- Syntax

CREATE ASSERTION <Assertion_name> **CHECK** (<Conditions>)

DROP ASSERTION <Assertion_name>

Example 12

- The birth date of a manager is smaller than his/her starting date

```
CREATE ASSERTION R12 CHECK (
        NOT EXISTS (
                SELECT  *
                FROM    EMPLOYEE, DEPARTMENT
                WHERE   SSN=MGRSSN
                AND BIRTHDATE > STARTDATE )
)
```

Example 15

- The salary of a manager must be larger than 50000

```
CREATE ASSERTION R15 CHECK (
        NOT EXISTS (
                SELECT  *
                FROM    EMPLOYEE, DEPARTMENT
                WHERE   SSN=MGRSSN
                AND SALARY < 50000 )
)
```

Example 15

- The salary of a manager must be larger than 50000

Check Constraint

```
ALTER TABLE DEPARTMENT (
        DNAME VARCHAR(20) UNIQUE,
        DNUBMER INT NOT NULL,
        MGRSSN CHAR(9),
        STARTDATE DATETIME,
        CONSTRAINT CHK_DEPT_SALARY_MGR CHECK (
                MGRSSN NOT IN (SELECT SSN FROM EMPLOYEE
                        WHERE SALARY < 50000 ))
)
```

Example 16

- The number of employees in a department is not larger than 20

```
CREATE ASSERTION R16 CHECK (
        20 >= ALL ( SELECT COUNT(ID)
                FROM EMPLOYEE
                GROUP BY DNO )
)
```

Example 16

- The number of employees in a department is not larger than 20

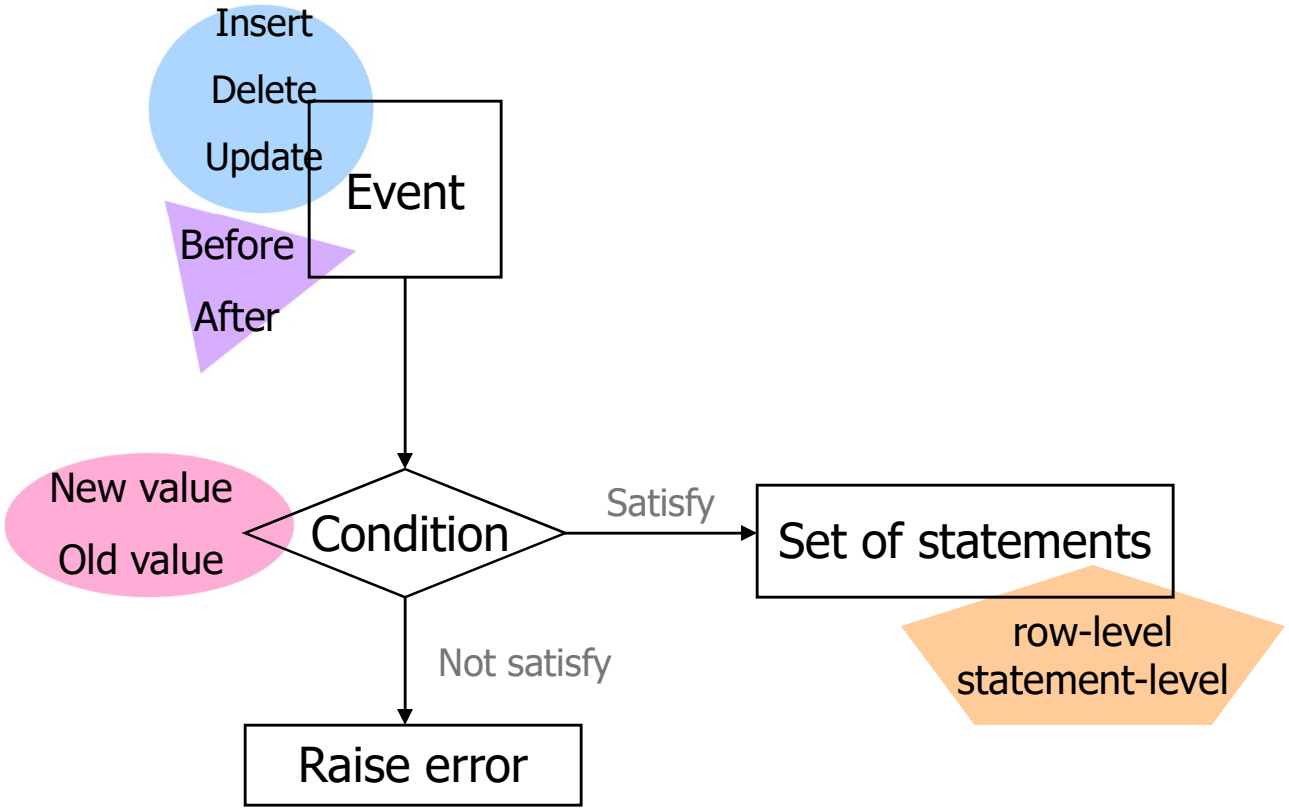
Check Constraint

```
ALTER TABLE EMPLOYEE ADD
CONSTRAINT CHK_EMP_NUM CHECK (
        20 >= ALL ( SELECT COUNT(ID)
                    FROM EMPLOYEE
                    GROUP BY DNO ))
```

Assertion or Check Constraint ?

Trigger

- A series of actions associated with certain events and performed whenever these events arise



Trigger

- Syntax

```
CREATE TRIGGER <Trigger_name>
AFTER|BEFORE   INSERT|UPDATE|DELETE   ON   <Table_name>
REFERENCING
        NEW   ROW|TABLE   AS   <Variable_name_1>
        OLD   ROW|TABLE   AS   <Variable_name_2>
FOR EACH ROW | FOR EACH STATEMENT
WHEN (<Condition>)
        <SQL_statements>
```

```
DROP TRIGGER <Trigger_name>
```

Example 15

- The salary of a manager must be larger than 50000

```
CREATE TRIGGER TR_DEPT_UPD
AFTER UPDATE OF MGRSSN ON DEPARTMENT
REFERENCING
        NEW ROW AS NewTuple
FOR EACH ROW
WHEN (50000 >= (SELECT SALARY
                FROM EMPLOYEE
                WHERE SSN=NewTuple.MGRSSN))
        Raise_error_to_user
```

Example 15

- The salary of a manager must be larger than 50000

```
CREATE TRIGGER TR_DEPT_UPD
AFTER UPDATE OF MGRSSN ON DEPARTMENT
REFERENCING
        NEW ROW AS NewTuple
        OLD ROW AS OldTuple
FOR EACH ROW
WHEN (50000 >= (SELECT SALARY FROM EMPLOYEE
                WHERE SSN=NewTuple.MGRSSN))
        UPDATE DEPARTMENT
        SET MGRSSN=OldTuple.MGRSSN
        WHERE MGRSSN=NewTuple.MGRSSN
```

Example 15

- The salary of a manager must be larger than 50000

```
CREATE TRIGGER TR_EMP_UPD
AFTER UPDATE OF SALARY ON EMPLOYEE
REFERENCING
        NEW ROW AS NewTuple
        OLD ROW AS OldTuple
FOR EACH ROW
WHEN (NewTuple.SALARY <= 50000 AND NewTuple.SSN IN (
                SELECT MGRSSN FROM DEPARTMENT ))
        UPDATE EMPLOYEE
        SET SALARY=OldTuple.SALARY
        WHERE SALARY=NewTuple.SALARY
```

Transaction

- A set of statements performing a certain process in a database, such that
  - Either all statements are performed successfully
  - Or nothing happens

- Eg. Transfer money in banks

```
Transaction Transfer_money
        Decrease balance of sending account
        Increase balance of receiving account
        If all is successful then commit transaction
        Else rollback transaction
End transaction
```

Transaction

- Must guarantee
  - Atomicity
  - Consistency
    - All constraints are not violated
      - During a transaction is performed
      - Before and after performing a transaction

Example 7

- COMPETITION(Date, Time, Team, Scores)
- Every match is the competition of two teams

```
Transaction Insert_Competition(t, s)
        Insert t into COMPETITION
        Insert s into COMPETITION
        If one statement fails then
                Rollback transaction
        Else
                Commit transaction
        End if
End transaction
```

Example 7

Transaction Delete_Competition(date, time)

For s ∈ COMPETITION (s.DATE = date ∧ s.TIME = time)

Delete s from COMPETITION

End for

If one statement fails then

Rollback transaction

Else

Commit transaction

End if

End transaction

Example 11

- Every invoice must have at least one detail

```
Transaction Insert_Invoice
        Insert into INVOICE
        Insert detail_1 into DETAIL
        Insert detail_2 into DETAIL
        …
        If one statement fails then
                Rollback transaction
        Else
                Commit transaction
        End if
End transaction
```

Stored procedure

- Commercial DBMSs provide the way to store functions or procedures
  - Stored in the database schema
  - Used in SQL statements

- Syntax

```
CREATE PROCEDURE <Procedure_name>
<List_of_parameters>
AS
        local variable declaration
        Body of the program
GO
EXEC <Procedure_name> <List_of_parameters>
```

Example 7

- Every match is a competition of two teams

```
CREATE PROCEDURE Insert_Competition
t COMPETITION , s COMPETITION
AS
        begin tran
                Insert t into COMPETITION
                If @@error<>0 rollback tran

                Insert s into COMPETITION
                If @@error<>0   rollback tran
        commit tran

GO


EXEC Insert_Competition  x, y
```

Discussion

- DBMS will check integrity constraints
  - After a modification was done in database
  - At the end of a transaction

- Where should integrity constraints are implemented ???
  - DBMS
  - Application

  - Too many triggers $\rightarrow$ the system will be slow
  - Stored procedure $\rightarrow$ more efficiently