

# Fundamentals of Database Systems

## [SQL – IV]

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit  
Indian Statistical Institute, Kolkata

September, 2019

- 1 Integrity Control
  
- 2 Basic Integrity Preservation
  - Fundamentals
  - Primary Key
  - Foreign Key
  - Nullity Check
  - General Check
  
- 3 Problems

# Basics

The term integrity in databases refers to the accuracy and consistency of data. The integrity control can set the following types of constraints on the data items:

- Basic integrity constraints
- Advanced integrity constraints

# Basic integrity constraints

The basic integrity constraints are of following four types:

- Defining the primary key constraint
  - specified as primary key  $(A_1, \dots, A_k)$
- Defining the foreign key constraint
  - specified as foreign key  $(A_p, \dots, A_q)$  references  $r$
- Defining the null constraint
  - specified as not null
- Defining the check constraint
  - specified as check <predicate>

# Consider a relational schema

- Branch =  $\langle \underline{\text{branch\_id}} : \text{integer}, \text{branch\_name} : \text{string}, \text{branch\_city} : \text{string}, \text{assets} : \text{real} \rangle$
- Customer =  $\langle \underline{\text{customer\_id}} : \text{integer}, \text{customer\_name} : \text{string}, \text{customer\_street} : \text{string}, \text{customer\_city} : \text{string}, \text{account\_number} : \text{integer} \rangle$
- Loan =  $\langle \text{loan\_number} : \text{integer}, \text{branch\_name} : \text{string}, \text{amount} : \text{real} \rangle$
- Borrower =  $\langle \text{customer\_name} : \text{string}, \text{loan\_number} : \text{integer} \rangle$
- Account =  $\langle \underline{\text{account\_number}} : \text{integer}, \text{branch\_name} : \text{string}, \text{balance} : \text{real} \rangle$
- Depositor =  $\langle \text{customer\_name} : \text{string}, \text{account\_number} : \text{integer} \rangle$

## Basic integrity constraints – primary key

The table Branch can be created with the following SQL query:

```
create table Branch(  
  branch_id int(10) not null,  
  branch_name varchar(30),  
  branch_city varchar(30),  
  assets float(20,2),  
  primary key (branch_id)  
);
```

## Basic integrity constraints – foreign key

The table `Customer` can be created with the following SQL query:

```
create table Customer(  
  customer_id int(20) not null,  
  customer_name varchar(30),  
  customer_street varchar(30),  
  customer_city varchar(30),  
  account_number int(20),  
  primary key (customer_id),  
  foreign key (account_number) references Account  
);
```

## Basic integrity constraints – null

The table Loan can be created with the following SQL query:

```
create table Loan(  
  loan_number int(10) not null,  
  branch_name varchar(30),  
  amount float(15,2)  
);
```



## Basic integrity constraints – check

It can be ensured that a predicate on the attributes (say the amount is non-negative) must be satisfied by every tuple in the table Loan by writing an SQL query as follows:

```
create table Loan(  
  loan_number int(10) not null,  
  branch_name varchar(30),  
  amount float(15,2),  
  check (amount >= 0)  
);
```

# Problems

- 1** Consider the following schema of an online code repository system like GitHub:
- Contributor =  $\langle \text{contributor\_name} : \text{string}, \text{contributor\_id} : \text{integer} \rangle$
  - Code-Group =  $\langle \underline{\text{contributor\_id}} : \text{integer}, \text{code\_group} : \text{string}, \text{count\_submissions} : \text{integer} \rangle$
- Set the basic integrity constraints on this schema.