

Introduction to Databases

(INFR10080)

(Database Constraints)

Instructor: Yang Cao

(Fall 2025)



THE UNIVERSITY
of EDINBURGH

Changelog

v25.0 Initial version

Integrity constraints

Databases often required to satisfy some integrity constraints

Determine what tuples can be stored in the database

Instances that satisfy the constraints are called **legal**

Common constraints: **keys** and **foreign keys**

These are special cases of more general constraints

- **Functional dependencies**
- **Inclusion dependencies**

2/16

Functional dependencies (FDs)

Constraints of the form $X \rightarrow Y$, where X, Y are sets of attributes

Semantics (on sets)

A relation R satisfies $X \rightarrow Y$ if for every two tuples $t_1, t_2 \in R$

$$\pi_X(t_1) = \pi_X(t_2) \implies \pi_Y(t_1) = \pi_Y(t_2)$$

Intuition: The values for the X attributes
determine the values for the Y attributes

Trivial FDs: $X \rightarrow Y$ where $Y \subseteq X$

3/16

Examples of FDs

Employee	Department	Manager
John	Finance	Smith
Mary	HR	Taylor
Susan	HR	Taylor
John	Sales	Smith

Which of the following FDs would the above relation satisfy?

- Department $\rightarrow$ Manager **Yes**
- Employee $\rightarrow$ Department **No**
- Employee, Manager $\rightarrow$ Department **No**
- Manager $\rightarrow$ Department **No**

4/16

Keys

A set of attributes X is a **key** for relation R if for every $t_1, t_2 \in R$

$$\pi_X(t_1) = \pi_X(t_2) \implies t_1 = t_2$$

A **key** for a table is a set of attributes that **uniquely identify a row**

$\implies$ no two rows can have the same values for key attributes

Key constraints: special case of FDs $X \rightarrow Y$

where Y is the **whole set of attributes** of a relation

5/16

Inclusion dependencies (INDs)

Constraints of the form $R[X] \subseteq S[Y]$

where R, S are relations and X, Y are **sequences** of attributes

Semantics

R and S satisfy $R[X] \subseteq S[Y]$ if

for every $t_1 \in R$ there exists $t_2 \in S$ such that $\pi_X(t_1) = \pi_Y(t_2)$

Important: the projection must respect the attributes order

INDs are **referential constraints**: **link** the contents of one table
with the contents of another table

A **foreign key** constraint is the conjunction of two constraints:

- $R[X] \subseteq S[Y]$ (an IND)
- Y is key for S (a key constraint)

6/16

Examples of INDs

Employees		Departments	
Name	Dep	Name	Mgr
John	Finance	Finance	John
Mary	HR	HR	Mary
John	HR	Sales	Linda
Linda	Finance		
Susan	Sales		

Which of the following INDs would the above relations satisfy?

- $\text{Employees}[\text{Dep}] \subseteq \text{Departments}[\text{Name}]$ **Yes**
- $\text{Employees}[\text{Name}] \subseteq \text{Departments}[\text{Mgr}]$ **No**
- $\text{Departments}[\text{Mgr}] \subseteq \text{Employees}[\text{Name}]$ **Yes**
- $\text{Departments}[\text{Mgr}, \text{Name}] \subseteq \text{Employees}[\text{Name}, \text{Dep}]$ **No**

7/16

Basic SQL constraints

NOT NULL to disallow null values

UNIQUE to declare keys

PRIMARY KEY key + not null

FOREIGN KEY to reference attributes in other tables

NULL values are ignored when checking constraints except for **NOT NULL** and **PRIMARY KEY**

8/16

Not Null

Declaring an attribute as **NOT NULL** disallows **null values** for that attribute

```
CREATE TABLE Account (  
    accnum  VARCHAR(12) NOT NULL,  
    branch  VARCHAR(30),  
    custid  VARCHAR(10),  
    balance NUMERIC(14,2) DEFAULT 0  
);
```

The following insertion would fail:

```
INSERT INTO Account(branch , custid )  
VALUES ( 'London ' , 'cust1 ' );
```

9/16

Keys

```
CREATE TABLE Account (  
    accnum  VARCHAR(12) UNIQUE,  
    branch  VARCHAR(30),  
    custid  VARCHAR(10),  
    balance NUMERIC(14,2)  
);
```

The following insertion gives an error:

```
INSERT INTO Account VALUES  
(1, 'London',      'cust1', 100),  
(1, 'Edinburgh',  'cust3', 200);
```

The following insertion succeeds:

```
INSERT INTO Account VALUES  
(NULL, 'London',      'cust1', 100),  
(NULL, 'Edinburgh',  'cust3', 200);
```

10/16

Compound keys

Keys consisting of more than one attribute must be declared using a different syntax

```
CREATE TABLE Movies (  
    m_title      VARCHAR(30),  
    m_director   VARCHAR(30),  
    m_year       SMALLINT,  
    m_genre      VARCHAR(30),  
    UNIQUE (m_title,m_year)  
);
```

This declares the set {m_title,m_year} as a key for Movies

11/16

Primary Keys

Essentially **UNIQUE** + **NOT NULL**

```
CREATE TABLE Account (  
    accnum VARCHAR(12) PRIMARY KEY,  
    branch VARCHAR(30),  
    custid VARCHAR(10),  
    balance NUMERIC(14,2)  
);
```

same as

```
CREATE TABLE Account (  
    accnum VARCHAR(12) NOT NULL UNIQUE,  
    branch VARCHAR(30),  
    custid VARCHAR(10),  
    balance NUMERIC(14,2)  
);
```

12/16

Foreign keys in SQL (1)

```
CREATE TABLE Customer (  
    id VARCHAR(10) PRIMARY KEY  
    name VARCHAR(20),  
    city VARCHAR(30),  
    address VARCHAR(30)  
);
```

```
CREATE TABLE Account (  
    accnum VARCHAR(12),  
    branch VARCHAR(30),  
    custid VARCHAR(10) REFERENCES Customer(id),  
    balance NUMERIC(14,2)  
);
```

Every value for attribute custid in Account must appear among the values of the **key** id in Customer

13/16

Foreign keys in SQL (2)

General syntax (useful for declaring compound foreign keys)

```
CREATE TABLE <table1> (  
    <attr> <type>,  
    ...  
    <attr> <type>,  
    FOREIGN KEY (<list1>  
        REFERENCES <table2>(<list2>)  
);
```

where

- <list1> and <list2> are lists with the **same number** of attributes
- attributes in <list1> are from table <table1>
- attributes in <list2> are **unique** in <table2>

14/16

Referential integrity and database modifications (1)

Deletion can cause problems with foreign keys

<u>Customer</u>	<u>ID</u>	<u>Name</u>	<u>Account</u>	<u>Number</u>	<u>CustID</u>
	cust1	John		123456	cust1
	cust2	Mary		654321	cust2

where Account.CustID is a foreign key reference to Customer.ID

What happens if one deletes (cust1,John) from Customer?

Three approaches are supported in SQL:

1. Reject the deletion operation
2. Propagate it to Account by deleting also (123456,cust1)
3. "Don't know" approach: keep the tuple in Account, but set CustID value to **NULL**

15/16

Referential integrity and database modifications (2)

All three approaches are supported in SQL

```
CREATE TABLE <table1> (  
    <attr> <type>,  
    ...  
    FOREIGN KEY <list1> REFERENCES <table2> (<list2>)  
    <approach>  
)
```

where <approach> can be:

1. Empty: Reject deletions from <table2> causing the FK to be violated
(this is the default when <approach> is not specified)
2. **ON DELETE CASCADE**: Propagate the deletion to <name>
(tuples in <table1> that violate the FK will be deleted)
3. **ON DELETE SET NULL**: “Don’t know” approach
(the values of the attributes in <list1>, for tuples in <name>
that violate the FK, are set to **NULL**)