



UPPSALA  
UNIVERSITET

# Lecture 4 – Relational Model

1DL301 Database Design I

Jan Kudlicka (jan.kudlicka@it.uu.se)

Fall 2019, Period 2





# INTENDED LEARNING OUTCOMES

To understand and to be able to describe the main concepts of the relational model:

- ▶ Relation (table), attribute (column), tuple (row)
- ▶ Null values
- ▶ Superkeys, candidate keys, primary keys
- ▶ Foreign keys

To be able to create corresponding structures in an RDBMS.

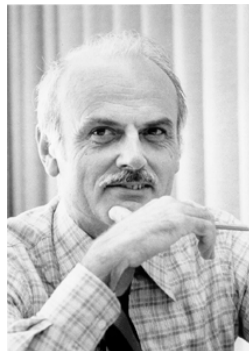


The relational model was proposed by Edgar F. Codd.

CODD, Edgar F. **A relational model of data for large shared data banks.**

Communications of the ACM, 1970, 13.6: 377-387.

[Link to the paper](#)



# RELATIONAL MODEL

Data in the relational model are structured **in the table form**.

A relational database is a set of named tables.

A **relational database schema**:

- ▶ **Names of the relations** (tables)
- ▶ **Names and the domains of the attributes** (columns) in each relation (table)
- ▶ **Integrity constraints** (IC) – conditions that data in the database must satisfy

# RELATIONAL MODEL

**Relation schema** – the structural description of a relation, denoted by  $R(A_1, A_2, \dots, A_n)$ , where  $R$  is the name of the relation with attributes  $A_1, A_2, \dots, A_n$ .

*E.g., Employee(Id, SSN, First\_Name, Last\_Name, Salary)*

**Instance / state of the relation** is a set of tuples (rows)

$t_j = \langle v_j^1, v_j^2, \dots, v_j^n \rangle$  stored in the relation:

$$r(R) = \{t_1, t_2, \dots, t_m\}$$

**Domain constraints:** Each  $v_j^i$  is an element of  $\text{dom}(A_i)$  (domain of  $A_i$ ).

*E.g.,  $r(R) = \{t_1, t_2, t_3\}$  where*

*$t_1 = \langle 1, 8302141234, \text{John}, \text{White}, 12500 \rangle$ ,*

*$t_2 = \langle 2, 8112272345, \text{Peter}, \text{Newman}, 11200 \rangle$ ,*

*$t_3 = \langle 3, 8208123456, \text{Anna}, \text{Johansson}, 12300 \rangle$*

# EXAMPLE

Relation Employee    Columns = Attributes

Diagram illustrating the structure of the Employee relation:

Schema →

Instance {

Id	SSN	First_Name	Last_Name	Salary
1	8302141234	John	White	12500
2	8112272345	Peter	Newman	11200
3	8208123456	Anna	Johansson	12300

Rows =  
= Tuples

```
1 CREATE TABLE Employee (  
2   Id int,  
3   SSN char(10),  
4   First_Name varchar(255),  
5   Last_Name varchar(255),  
6   Salary decimal(10, 2)  
7 )
```

# DATA TYPES

The most used data types:

- ▶ boolean, smallint, int, float, double

- ▶ decimal( $p$ ,  $s$ )

*Exact numeric value with  $p$  significant digits and  $s$  digits after the decimal point.*

- ▶ char( $size$ )

*Fixed-length string of length  $size$ . Max. value for  $size$  is usually 255.*

- ▶ varchar( $size$ )

*Variable-length string of max. length  $size$ . Max. value for  $size$  is usually 255.*

- ▶ enum( $s_1$ ,  $s_2$ , ...,  $s_n$ )

*Value from a set of allowed strings. (Note: No enum in SQLite).*

- ▶ text

- ▶ date, datetime

Note: Support for data types vary between different RDBMS. Check the documentation for the RDBMS you use:

MySQL: <https://dev.mysql.com/doc/en/data-types.html>

SQLite: <http://www.sqlite.org/datatype3.html>



# RELATIONAL MODEL, CONT'D

Instance of the relation is a **relation** in the mathematical sense (i.e., a subset of the Cartesian product):

$$r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \cdots \times \text{dom}(A_n)$$

Since  $r(R)$  is a set:

- ▶ order of tuples does not matter
- ▶ there are no duplicate tuples in a relation

**Relational database instance / state** – a set of the states of all relations.

The schema is prepared in advance while the state changes over time.

# FIRST NORMAL FORM (1NF)

**First normal form (1NF)** – each attribute value must be atomic and it must be a single value from the attribute's domain.

In simple words: *“Every cell has a single, indivisible value from the set of allowed values.”*

Example: Course(Code, Title)

⟨1DL301, "Database Design I"⟩ is ok.

⟨1DL301, {"Database Design I", "Databasteknik I"}⟩ is not.

Note: We will revisit this in the lectures on normal forms and normalization.



# NULL

Relational model supports a special NULL value (sometimes denoted by  $\perp$ ) that is used when the attribute value is:

- ▶ Unknown or missing

*E.g., salary for a person is unknown.*

- ▶ Not applicable

*E.g., Swedish personal no. for a foreigner, forward position for defenceman or goaltender in ice hockey*

NULL does not mean 0, nor an empty string, nor false, etc.



# THREE-VALUED LOGIC

Three-valued logic – TRUE, FALSE, NULL.

Any comparison involving NULL returns NULL.

*1 = 1 is TRUE and 1 = 2 is FALSE*

*1 = NULL, 1 <> NULL, NULL = NULL, NULL <> NULL are all NULL*

Negation:  $\neg$ NULL is also NULL.

Conjunction (“and”) and disjunction (“or”) involving NULL:

- ▶ NULL and FALSE is FALSE
- ▶ NULL and TRUE is NULL
- ▶ NULL or FALSE is NULL
- ▶ NULL or TRUE is TRUE

There are special operators in SQL to test if a value is or is not NULL:

- 1 IS NULL
- 2 IS NOT NULL

# NOT NULL CONSTRAINTS

We can forbid NULL values for attributes by adding **NOT NULL** in the attribute definitions in the **CREATE TABLE** command:

```
1 CREATE TABLE Employee (  
2     Id int NOT NULL,  
3     SSN char(10) NOT NULL,  
4     First_Name varchar(255) NOT NULL,  
5     Last_Name varchar(255) NOT NULL,  
6     Salary decimal(10, 2)  
7 )
```

Tip: the default is to allow NULL, so make sure to specify NOT NULL for non-nullable attributes.

# SUPERKEY

Given  $R(A_1, A_2, \dots, A_n)$ , a subset of attributes  $SK \subseteq \{A_1, A_2, \dots, A_n\}$  is a **superkey** if for any legal instance  $r(R)$ :

$$\forall t_1, t_2 \in r(R) : t_1 \neq t_2 \Rightarrow t_1[SK] \neq t_2[SK]$$

where  $t[SK]$  means a subtuple of values from  $t$  corresponding to the attributes in  $SK$ .

In simple words: A set of attributes is a superkey if no two distinct tuples can have the same value for all of these attributes.

Every relation has at least one trivial superkey – the set of all attributes.

# SUPERKEY, EXAMPLE

Employee

Id	SSN	First_Name	Last_Name	Salary
1	8302141234	John	White	12500
2	8112272345	Peter	Newman	11200
3	8208123456	Anna	Johansson	12300

- ▶ Id, SSN, First\_Name, Last\_Name, Salary
- ▶ Id
- ▶ Id, First\_Name
- ▶ Id, First\_Name, Last\_Name
- ▶ SSN
- ▶ Id, SSN
- ▶ ...

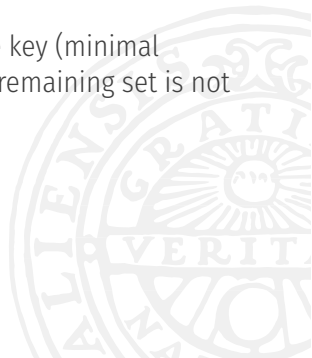


# MINIMAL SUPERKEY

A superkey  $K$  is **minimal** if there does not exist a proper subset  $K' \subset K$  that is a superkey.

A minimal superkey is called a **candidate key**.

In simple words: A set of attributes is a candidate key (minimal superkey) if by removing any of its attributes the remaining set is not a superkey anymore.





# CANDIDATE KEYS AND THE PRIMARY KEY

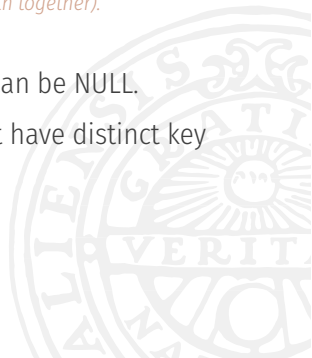
One table might have several candidate keys.

The database designer chooses one of the candidate keys to be the **primary key** (PK). We will denote the PK by underlining.

*E.g., for Student: either SSN or Id can be chosen as a PK (but not both together).*

**Entity integrity constraint:** no primary key value can be NULL.

**Key / uniqueness constraint:** distinct tuples must have distinct key values.



# KEY CONSTRAINTS IN SQL

```
1 CREATE TABLE Employee (  
2     Id int NOT NULL PRIMARY KEY,  
3     SSN char(10) NOT NULL UNIQUE,  
4     First_Name varchar(255) NOT NULL,  
5     Last_Name varchar(255) NOT NULL,  
6     Salary decimal(10, 2)  
7 )
```

NOT NULL UNIQUE can be used for candidate keys.

If a key contains several attributes:

```
1 CREATE TABLE WorksAt (  
2     EmployeeId int NOT NULL,  
3     CompanyId int NOT NULL,  
4     PRIMARY KEY(EmployeeId, CompanyId)  
5 )
```



# EXERCISE 1

In the relational model, if a set of attributes  $K$  is a superkey of a relation  $R$  then:

1.  $R$  contains at least two different tuples  $t_1$  and  $t_2$  with  $t_1[K] = t_2[K]$ .
2.  $R$  cannot contain two different tuples  $t_1$  and  $t_2$  with  $t_1[K] = t_2[K]$ .
3.  $R$  contains exactly two different tuples  $t_1$  and  $t_2$  with  $t_1[K] = t_2[K]$ .
4.  $R$  contains at least two different tuples  $t_1$  and  $t_2$  with  $t_1[K] \neq t_2[K]$ .

## EXERCISE 2

A1	A2	A3	A4
a	1	alpha	0
a	2	beta	0
b	3	gamma	0
b	4	beta	0
b	5	beta	0
c	1	alpha	0
d	2	gamma	0

1. A2 is a candidate key
2. A4 is the primary key
3. (A1, A2, A3, A4) is a superkey
4. (A2, A3) is a superkey
5. None of the previous answers

## EXERCISE 3

In the relational model, if a set of attributes,  $K$ , is a candidate key of a relation  $R$  and  $X$  is an attribute of  $R$  not in  $K$ , then

1.  $K \cup \{X\}$  is also a candidate key.
2.  $K \setminus \{X\}$  is also a candidate key.
3.  $K$  is also a primary key of  $R$ .
4. None of the previous answers.

$A \setminus B$  means the set of elements of  $A$  which are not in  $B$  (set difference).



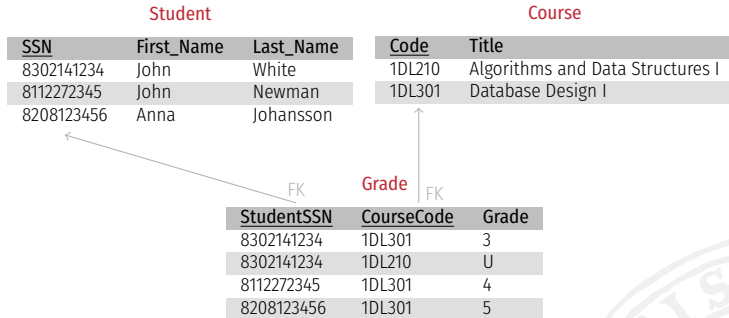
# FOREIGN KEYS AND REFERENTIAL INTEGRITY

A set of attributes  $FK$  in  $R_1$  is a **foreign key** (of  $R_1$ ) that **references** / **refers to** relation  $R_2$  if it satisfies the following conditions:

1. The attributes in  $FK$  have the same domain(s) as the primary key attributes  $PK$  of  $R_2$ .
2. For  $\forall t_1 \in r_1(R_1)$ 
  - ▶ either  $\exists t_2 \in r_2(R_2)$  such that  $t_1[FK] = t_2[PK]$  ( $t_1$  refers to  $t_2$ ),
  - ▶ or  $t_1[FK]$  consists of NULLs ( $t_1$  does not refer to any tuple in  $r_2(R_2)$ ).

**Referential integrity constraint** in simple words: All references must refer to *existing* tuples.

# EXAMPLE



Student(SSN, First\_Name, Last\_Name)

Course(Code, Title)

Grade(StudentSSN, CourseCode, Grade)

StudentSSN ref. Student(SSN)

CourseCode ref. Course(Code)

# REFERENTIAL INTEGRITY CONSTRAINTS IN SQL

```
1 CREATE TABLE Student (  
2     SSN char(10) NOT NULL PRIMARY KEY,  
3     First_Name varchar(255),  
4     Last_Name varchar(255)  
5 );  
6 CREATE TABLE Course (  
7     Code varchar(32) NOT NULL PRIMARY KEY,  
8     Title varchar(255)  
9 );  
10 CREATE TABLE Grade (  
11     StudentSSN char(10) NOT NULL,  
12     CourseCode varchar(32) NOT NULL,  
13     Grade char(1),  
14     PRIMARY KEY (StudentSSN, CourseCode),  
15     FOREIGN KEY (StudentSSN) REFERENCES Student(SSN),  
16     FOREIGN KEY (CourseCode) REFERENCES Course(Code)  
17 );
```





## TIP: USE INTEGER PRIMARY KEYS

```
1 CREATE TABLE Student (  
2     Id int NOT NULL PRIMARY KEY,  
3     SSN char(10) NOT NULL UNIQUE,  
4     First_Name varchar(255),  
5     Last_Name varchar(255)  
6 );  
7 CREATE TABLE Course (  
8     Id int NOT NULL PRIMARY KEY,  
9     Code varchar(32) NOT NULL UNIQUE,  
10    Title varchar(255)  
11 );  
12 CREATE TABLE Grade (  
13     StudentId int NOT NULL,  
14     CourseId int NOT NULL,  
15     Grade char(1),  
16     PRIMARY KEY (StudentId, CourseId),  
17     FOREIGN KEY (StudentId) REFERENCES Student(Id),  
18     FOREIGN KEY (CourseId) REFERENCES Course(Id)  
19 );
```



# REFERENCE OPTIONS

```
1 FOREIGN KEY (...)  
2 REFERENCES ...(...)  
3 [ON DELETE reference_option]  
4 [ON UPDATE reference_option]
```

Reference options: CASCADE, SET NULL, NO ACTION

- ▶ CASCADE = Deletes/update matching rows in the child tables too
- ▶ NO ACTION = Delete/update will fail if there are matching rows in the child tables
- ▶ SET NULL = FK for matching rows in the child tables will be changed to NULL (NULLs must be allowed)

(Child tables = tables with a FK referencing the table on which we run DELETE or UPDATE.)

Note: Some RDBMS also support SET DEFAULT.

# INSERTING NEW DATA

```
1 INSERT INTO Student(Id, SSN, First_Name, Last_Name)
2 VALUES (1, "8302141234", "John", "White")
```

Specifying attributes is not necessary if the order of values is the same as in the relation schema:

```
1 INSERT INTO Student
2 VALUES (1, "8302141234", "John", "White")
```

Tip: Always include the attribute names.

Note: If some of the values being inserted are NULL (and NULLs are allowed):

```
1 INSERT INTO Course(Id, Code)
2 VALUES (2, "1DL301")
```

It is also possible to specify default values for attributes. Omitting values for such attributes will lead to using the defaults.

# DELETING AND MODIFYING DATA

Deleting a tuple from a relation:

```
1 DELETE FROM Student WHERE Id=1
```

Modifying a tuple in a relation:

```
1 UPDATE Course  
2 SET Title="Database Design I"  
3 WHERE Id=2
```

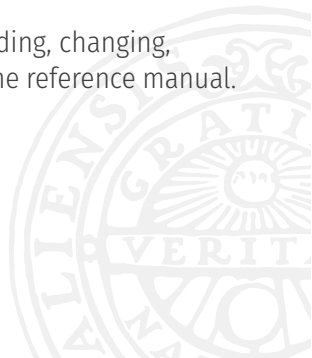


# REMOVING AND MODIFYING RELATIONS

Removing a relation:

1 `DROP TABLE` Student

`ALTER TABLE` can be used to modify the schema (adding, changing, removing the attributes) and constraints. Check the reference manual.



Syntax and support for different features might vary for different RDBMS.

Use the reference manual for your RDBMS actively:

- ▶ MySQL: <https://dev.mysql.com/doc/>
- ▶ SQLite: <http://www.sqlite.org/docs.html>

