



Relational Model

GRA 4142 Data Management and Python Programming

Jan Kudlicka (jan.kudlicka@bi.no)

Department of Data Science and Analytics

October 29, 2021

Domain: a set of valid values of an attribute.

The values must be atomic (indivisible), so we cannot store a list/array/tuple/set as an attribute value for a single tuple.

Common way how to specify domains is to specify a data type and optional restrictions.

E.g. the age of an employee: integer, must be greater than 15.

The most used data types:

▶ BOOLEAN, 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 varies between different RDBMSs. Check the documentation for the RDBMS you use.

Relational schema, denoted by $R(A_1, A_2, \dots, A_n)$, is a structural description of a relation:

- ▶ relation name R ,
- ▶ names and domains of the attributes A_i .

The number n of attributes is called the *degree* or *arity* of the relation.

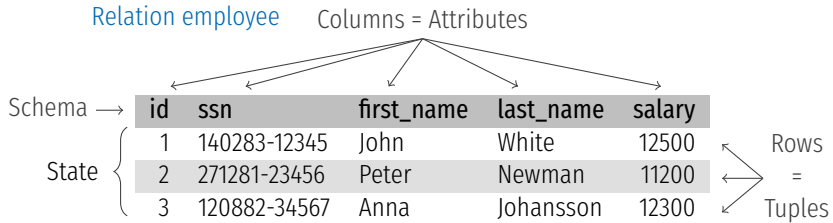
Relational database schema:

- ▶ relation schema for all relations in the database,
- ▶ integrity constraints (IC): conditions that the data in the database must satisfy.

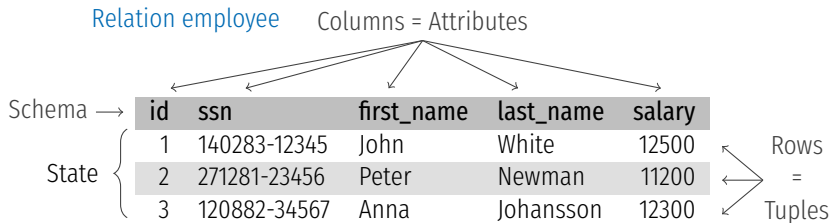
Relation state: a set of tuples stored in the relation.

Relational database state: a set of the states of all relations in the database.

The schema is prepared in advance and changes very rarely, the state changes often.



What data types would you use for each attribute in this relation?



```
1 CREATE TABLE employee (  
2     id INT,  
3     ssn CHAR(12),  
4     first_name VARCHAR(255),  
5     last_name VARCHAR(255),  
6     salary DECIMAL(10, 2)  
7 )
```

Relational model supports a special NULL value that is used when the attribute value is:

- ▶ Unknown or missing

Example: salary for a person is unknown.

- ▶ Not applicable

Example: Norwegian personal no. for a foreigner, forward position for defenceman or goaltender in ice hockey.

Three-valued logic – TRUE, FALSE, NULL.

Any comparison with NULL returns NULL.

For example, $1 = \text{NULL}$, $1 <> \text{NULL}$, $\text{NULL} = \text{NULL}$, $\text{NULL} <> \text{NULL}$ are all NULL.

Negation: (NOT 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

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

```
1 CREATE TABLE employee (  
2     id INT NOT NULL,  
3     ssn CHAR(12) 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 is a set of attributes that can uniquely identify a tuple in any possible state of the relation.

In different words: A set of attributes is a superkey if no two distinct tuples (in any possible state) can have the same value for all of these attributes.

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

employee

id	ssn	first_name	last_name	salary
1	140283-12345	John	White	12500
2	271281-23456	Peter	Newman	11200
3	120882-34567	Anna	Johansson	12300

employee

id	ssn	first_name	last_name	salary
1	140283-12345	John	White	12500
2	271281-23456	Peter	Newman	11200
3	120882-34567	Anna	Johansson	12300

- ▶ id, ssn, first_name, last_name, salary
- ▶ id
- ▶ id, first_name
- ▶ id, first_name, last_name
- ▶ ssn
- ▶ id, ssn
- ▶ ...

A superkey is **minimal** if by removing any of its attributes the remaining set is no longer a superkey.
A minimal superkey is called a **candidate key** (CK).

employee

id	ssn	first_name	last_name	salary
1	140283-12345	John	White	12500
2	271281-23456	Peter	Newman	11200
3	120882-34567	Anna	Johansson	12300

employee

id	ssn	first_name	last_name	salary
1	140283-12345	John	White	12500
2	271281-23456	Peter	Newman	11200
3	120882-34567	Anna	Johansson	12300

▶ id

▶ ssn

One relation 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 in the schema by underlining.

Example: In the employee relation either ssn or id can be chosen as the PK (but not both together!).

Entity integrity constraint: primary key value can not be NULL.

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

```
1 CREATE TABLE employee (  
2     id INT NOT NULL PRIMARY KEY,  
3     ssn CHAR(12) NOT NULL UNIQUE, -- NOT NULL UNIQUE can be used for other CK  
4     first_name VARCHAR(255) NOT NULL,  
5     last_name VARCHAR(255) NOT NULL,  
6     salary DECIMAL(10, 2)  
7 )
```

If the primary key consists of several attributes:

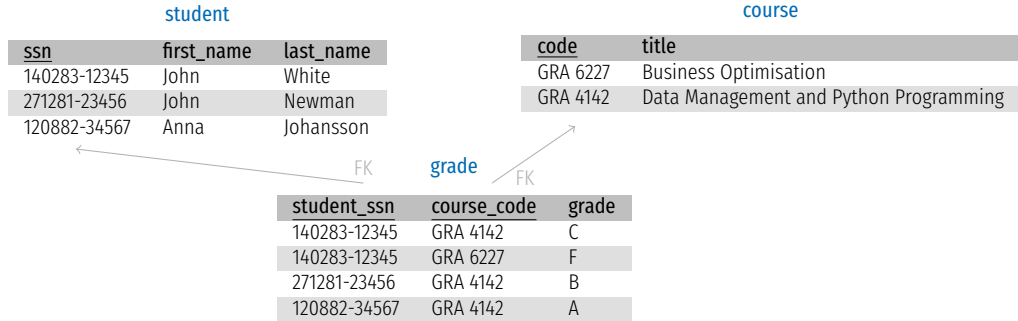
```
1 CREATE TABLE car (  
2     country VARCHAR(3) NOT NULL,  
3     license_plate VARCHAR(255) NOT NULL,  
4     PRIMARY KEY(country, license_plate)  
5 )
```

```
1 CREATE TABLE product (  
2     id INT NOT NULL PRIMARY KEY,  
3     name VARCHAR(255) NOT NULL,  
4     price DECIMAL(10, 2) CHECK (price > 0),  
5     discounted_price DECIMAL(10, 2) CHECK (discounted_price > 0),  
6     CHECK (discounted_price <= price)  
7 )
```

A set of attributes FK in R_1 is a **foreign key** (of R_1) **referencing / referring to** R_2 if it satisfies the following conditions:

1. The attributes in FK have the same domains as the primary key attributes of R_2 .
2. For each tuple in R_1 , the value of FK is either one of the values of the PK in R_2 or it is NULL.

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



student(ssn, first_name, last_name)

course(code, title)

grade(student_ssn, course_code, grade),

student_ssn ref. student(ssn), course_code ref. course(code)

```

1 CREATE TABLE student (
2     ssn CHAR(12) NOT NULL PRIMARY KEY,
3     first_name VARCHAR(255),
4     last_name VARCHAR(255)
5 );
6 CREATE TABLE course (
7     code VARCHAR(32)
8         NOT NULL PRIMARY KEY,
9     title VARCHAR(255)
10 );
11 CREATE TABLE grade (
12     student_ssn CHAR(12) NOT NULL,
13     course_code VARCHAR(32) NOT NULL,
14     grade CHAR(1),
15     PRIMARY KEY (student_ssn, course_code),
16     FOREIGN KEY (student_ssn) REFERENCES student(ssn),
17     FOREIGN KEY (course_code) REFERENCES course(code)
18 );

```

Notes:

- ▶ Note that you also need to include the names of the PK attribute(s) in the REFERENCES part.
- ▶ Example with a FK consisting of several attributes:
 FOREIGN KEY (car_state, car_license_plate) REFERENCES car(state, license_plate).

```
1 CREATE TABLE student (  
2     id INT NOT NULL PRIMARY KEY,  
3     ssn CHAR(12) 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     student_id INT NOT NULL,  
14     course_id INT NOT NULL,  
15     grade CHAR(1),  
16     PRIMARY KEY (student_id, course_id),  
17     FOREIGN KEY (student_id) REFERENCES student(id),  
18     FOREIGN KEY (course_id) REFERENCES course(id)  
19 );
```

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

Common reference options: NO ACTION, CASCADE, SET NULL.

- ▶ NO ACTION: Delete/update will fail if there exist tuples referring to the tuple being deleted/updated. This is the default option.
- ▶ CASCADE: Delete/update will also delete/update the referencing tuples.
- ▶ SET NULL: FK of the referencing tuples will be set to NULL (NULLs must be allowed).

Note: Some RDBMSs also support SET DEFAULT.

```
1 INSERT INTO student(id, ssn, first_name, last_name)
2 VALUES (1, "140283-12345", "John", "White")
```

Specifying attribute names is not needed if all attributes are inserted and in the same order as in the relation schema:

```
1 INSERT INTO student
2 VALUES (1, "140283-12345", "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, "GRA 4142")
```

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

Deleting a tuple from a relation:

```
1 DELETE FROM student WHERE id=1
```

Warning! `DELETE` without a `WHERE` clause will remove all tuples!

Modifying an existing tuple in a relation:

```
1 UPDATE course
2 SET title="Data Management and Python Programming"
3 WHERE id=2
```

Removing a relation:

```
1 DROP TABLE grade
```

ALTER TABLE can be used to modify the schema (e.g. adding, changing, removing the attributes or changing the data type) 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>