

# Better data quality with constraints

INTRODUCTION TO RELATIONAL DATABASES IN SQL

SQL

**Timo Grossenbacher**  
Data Journalist

# Integrity constraints

1. **Attribute constraints**, e.g. data types on columns (Chapter 2)
2. **Key constraints**, e.g. primary keys (Chapter 3)
3. **Referential integrity constraints**, enforced through foreign keys (Chapter 4)

# Why constraints?

- Constraints give the data structure
- Constraints help with consistency, and thus data quality
- Data quality is a business advantage / data science prerequisite
- Enforcing is difficult, but PostgreSQL helps

# Data types as attribute constraints

Name	Aliases	Description
<code>bigint</code>	<code>int8</code>	signed eight-byte integer
<code>bigserial</code>	<code>serial8</code>	autoincrementing eight-byte integer
<code>bit [ (n) ]</code>		fixed-length bit string
<code>bit varying [ (n) ]</code>	<code>varbit [ (n) ]</code>	variable-length bit string
<code>boolean</code>	<code>bool</code>	logical Boolean (true/false)
<code>box</code>		rectangular box on a plane
<code>bytea</code>		binary data ("byte array")
<code>character [ (n) ]</code>	<code>char [ (n) ]</code>	fixed-length character string
<code>character varying [ (n) ]</code>	<code>varchar [ (n) ]</code>	variable-length character string
<code>cidr</code>		IPv4 or IPv6 network address

From the [PostgreSQL documentation](#).

# Dealing with data types (casting)

```
CREATE TABLE weather (  
  temperature integer,  
  wind_speed text);  
SELECT temperature * wind_speed AS wind_chill  
FROM weather;
```

operator does not exist: integer \* text  
HINT: No operator matches the given name and argument type(s).  
You might need to add explicit type casts.

```
SELECT temperature * CAST(wind_speed AS integer) AS wind_chill  
FROM weather;
```

# Let's practice!

INTRODUCTION TO RELATIONAL DATABASES IN SQL

# Working with data types

INTRODUCTION TO RELATIONAL DATABASES IN SQL



**Timo Grossenbacher**  
Data Journalist

# Working with data types

- Enforced on columns (i.e. attributes)
- Define the so-called "domain" of a column
- Define what operations are possible
- Enforce consistent storage of values



# The most common types

- `text` : character strings of any length
- `varchar [ (x) ]` : a maximum of `x` characters
- `char [ (x) ]` : a fixed-length string of `x` characters
- `boolean` : can only take three states, e.g. `TRUE` , `FALSE` and `NULL` (unknown)

From the [PostgreSQL documentation](#).

# The most common types (cont'd.)

- `date` , `time` and `timestamp` : various formats for date and time calculations
- `numeric` : arbitrary precision numbers, e.g. `3.1457`
- `integer` : whole numbers in the range of `-2147483648` and `+2147483647`

From the [PostgreSQL documentation](#).

# Specifying types upon table creation

```
CREATE TABLE students (  
  ssn integer,  
  name varchar(64),  
  dob date,  
  average_grade numeric(3, 2), -- e.g. 5.54  
  tuition_paid boolean  
);
```

# Alter types after table creation

```
ALTER TABLE students  
ALTER COLUMN name  
TYPE varchar(128);
```

```
ALTER TABLE students  
ALTER COLUMN average_grade  
TYPE integer  
-- Turns 5.54 into 6, not 5, before type conversion  
USING ROUND(average_grade);
```

# Let's apply this!

INTRODUCTION TO RELATIONAL DATABASES IN SQL

# The not-null and unique constraints

INTRODUCTION TO RELATIONAL DATABASES IN SQL



**Timo Grossenbacher**  
Data Journalist

# The not-null constraint

- Disallow `NULL` values in a certain column
- Must hold true for the current state
- Must hold true for any future state

# What does NULL mean?

- unknown
- does not exist
- does not apply
- ...



# What does NULL mean? An example

```
CREATE TABLE students (  
  ssn integer not null,  
  lastname varchar(64) not null,  
  home_phone integer,  
  office_phone integer  
);
```

```
NULL != NULL
```

# How to add or remove a not-null constraint

When creating a table...

```
CREATE TABLE students (  
  ssn integer not null,  
  lastname varchar(64) not null,  
  home_phone integer,  
  office_phone integer  
);
```

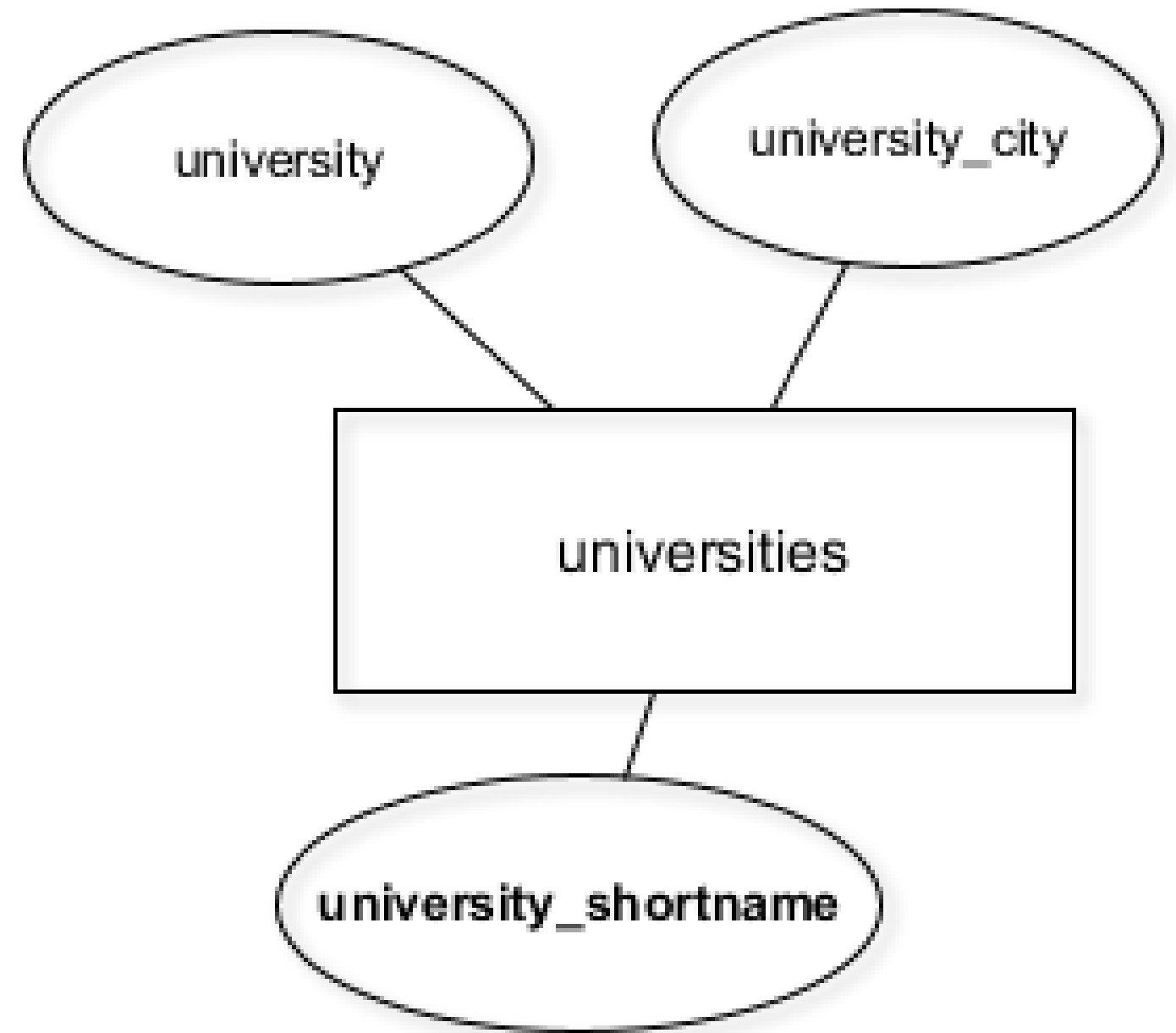
After the table has been created...

```
ALTER TABLE students  
ALTER COLUMN home_phone  
SET NOT NULL;
```

```
ALTER TABLE students  
ALTER COLUMN ssn  
DROP NOT NULL;
```

# The unique constraint

- Disallow duplicate values in a column
- Must hold true for the current state
- Must hold true for any future state



# Adding unique constraints

```
CREATE TABLE table_name (  
  column_name UNIQUE  
);
```

```
ALTER TABLE table_name  
ADD CONSTRAINT some_name UNIQUE(column_name);
```

# Let's apply this to the database!

INTRODUCTION TO RELATIONAL DATABASES IN SQL