

# Welcome!

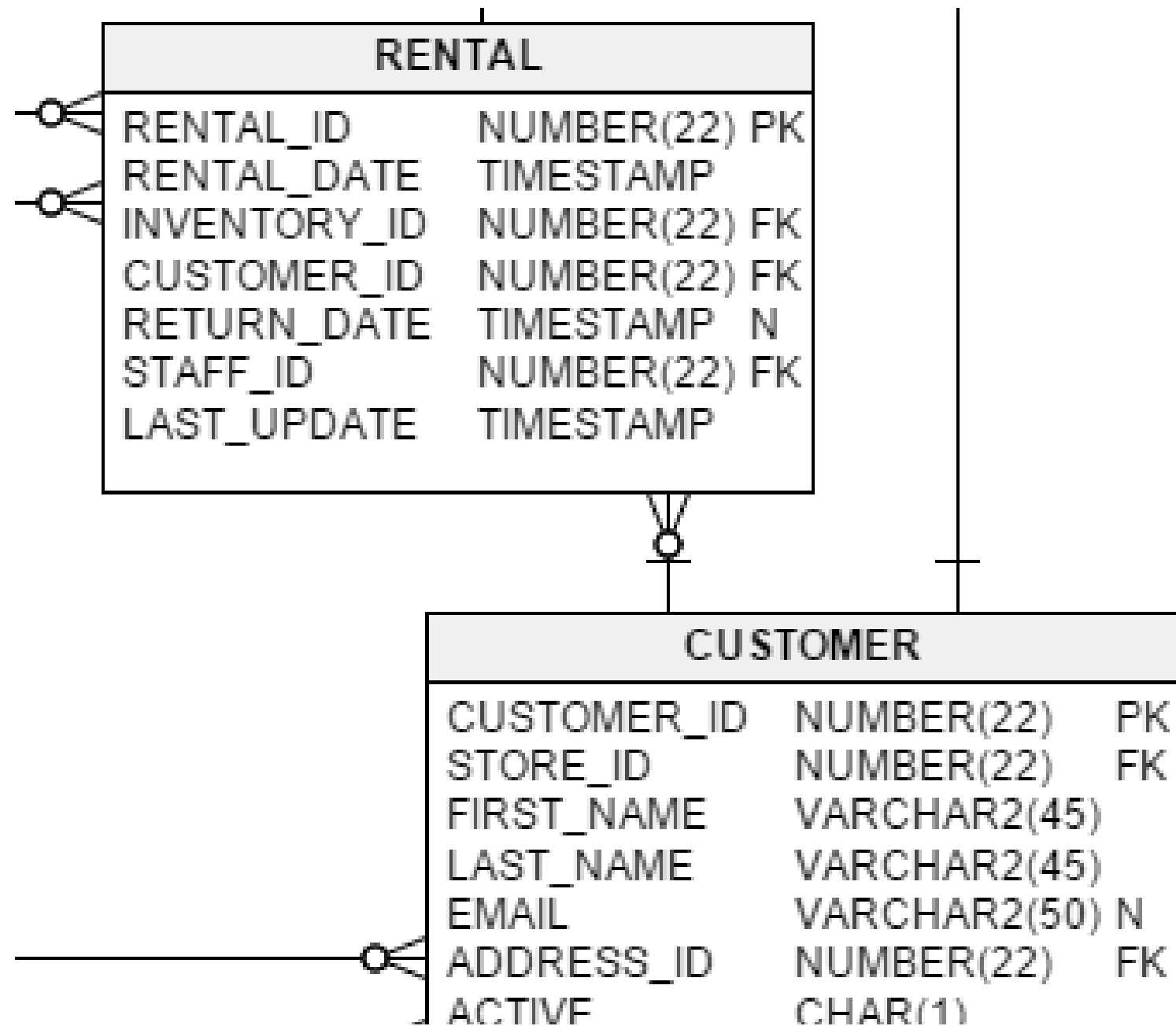
FUNCTIONS FOR MANIPULATING DATA IN POSTGRESQL



**Brian Piccolo**

Sr. Director, Digital Strategy

# The Sakila Database



- Highly normalized
- Representative data types
- Custom functions

# Topics

- Common data types in PostgreSQL
- Date and time functions and operators
- Parsing and manipulating text
- Full-text search and PostgreSQL Extensions

# Common data types

- Text data types
  - `CHAR` , `VARCHAR` and `TEXT`
- Numeric data types
  - `INT` and `DECIMAL`
- Date / time data types
  - `DATE` , `TIME` , `TIMESTAMP` , `INTERVAL`
- Arrays

# Text data types

```
SELECT title
FROM film
LIMIT 5
```

```
+-----+
| title          |
+-----+
| ACADEMY DINOSAUR |
| ACE GOLDFINGER  |
| ADAPTATION HOLES |
| AFFAIR PREJUDICE |
| AFRICAN EGG     |
+-----+
```

```
SELECT description
FROM film
LIMIT 2
```

```
+-----+
| description    |
+-----+
| A Epic Drama of a Feminist And a Mad |
| Scientist who must Battle a Teacher in |
| The Canadian Rockies.                  |
| A Astounding Epistle of a Database     |
| Administrator And a Explorer who      |
| must Find a Car in Ancient China       |
+-----+
```

# Numeric data types

```
SELECT
```

```
    payment_id
```

```
FROM payment
```

```
LIMIT 5
```

```
+-----+
| payment_id |
|-----|
| 1          |
| 2          |
| 3          |
| 4          |
| 5          |
+-----+
```

```
SELECT
```

```
    amount
```

```
FROM payment
```

```
LIMIT 5
```

```
+-----+
| amount |
|-----|
| 2.99   |
| 0.99   |
| 5.99   |
| 0.99   |
| 9.99   |
+-----+
```

# Determining data types from existing tables

```
SELECT
    title,
    description,
    special_features
FROM FILM
LIMIT 5
```

```
+-----+-----+-----+
| title          | description      | special_features |
+-----+-----+-----+
| ACADEMY D...   | A Epic...       | {Deleted Scenes,Behi...} |
| ACE GOLD...    | A Astound..     | {Trailers,Deleted Scenes} |
| AFFAIR PR...   | A Fanciful,..   | {Commentaries,Behind the...} |
+-----+-----+-----+
```

# Determining data types from existing tables

```
SELECT
    column_name,
    data_type
FROM INFORMATION_SCHEMA.COLUMNS
WHERE column_name in ('title', 'description', 'special_features')
AND table_name = 'film';
```

```
+-----+-----+
| column_name | data_type |
+-----+-----+
| title       | character varying |
| description | text           |
| special_features | ARRAY       |
+-----+-----+
```



# Let's practice!

FUNCTIONS FOR MANIPULATING DATA IN POSTGRESQL

# Date and time data types

FUNCTIONS FOR MANIPULATING DATA IN POSTGRESQL



**Brian Piccolo**

Sr. Director, Digital Strategy

# TIMESTAMP data types

- ISO 8601 format: yyyy-mm-dd

```
+-----+
| timestamp                |
+-----+
| 2019-03-26 01:05:17.93027+00 |
+-----+
```

```
SELECT payment_date
FROM payment;
```

```
+-----+
| payment_date            |
+-----+
| 2005-05-25 11:30:37     |
+-----+
```

# DATE and TIME data types

```
+-----+-----+
| date       | time           |
|-----|-----|
| 2005-05-28 | 01:05:17.93027+00 |
+-----+-----+
```

```
SELECT create_date
FROM customer
```

```
+-----+
| create_date |
|-----|
| 2006-02-14  |
+-----+
```

# INTERVAL data types

```
+-----+  
| interval |  
|-----|  
| 4 days   |  
+-----+
```

```
SELECT rental_date + INTERVAL '3 days' as expected_return  
FROM rental;
```

```
+-----+  
| expected_return |  
|-----|  
| 2005-05-27 22:53:30 |  
+-----+
```

# Looking at date and time types

```
SELECT
    column_name,
    data_type
FROM INFORMATION_SCHEMA.COLUMNS
WHERE column_name in ('rental_date')
AND table_name = 'rental';
```

```
+-----+-----+
| column_name | data_type                |
|-----|-----|
| rental_date | timestamp without time zone |
+-----+-----+
```

# Let's practice!

FUNCTIONS FOR MANIPULATING DATA IN POSTGRESQL

# Working with ARRAYs

FUNCTIONS FOR MANIPULATING DATA IN POSTGRESQL



**Brian Piccolo**

Sr. Director, Digital Strategy



# Before we get started

## *CREATE TABLE example*

```
CREATE TABLE my_first_table (  
    first_column text,  
    second_column integer  
);
```

## *INSERT example*

```
INSERT INTO my_first_table  
    (first_column, second_column) VALUES ('text value', 12);
```

# ARRAY a special type

Let's create a simple table with two array columns.

```
CREATE TABLE grades (  
    student_id int,  
    email text[][],  
    test_scores int[]  
);
```

# INSERT statements with ARRAYS

Example INSERT statement:

```
INSERT INTO grades  
VALUES (1,  
       ' {"work", "work1@datacamp.com"}, {"other", "other1@datacamp.com"} }',  
       ' {92, 85, 96, 88} ' );
```

# Accessing ARRAYS

**SELECT**

```
email[1][1] AS type,  
email[1][2] AS address,  
test_scores[1],
```

**FROM** grades;

```
+-----+-----+-----+  
| type   | address           | test_scores |  
+-----+-----+-----+  
| work   | work1@datacamp.com | 92          |  
| work   | work2@datacamp.com | 76          |  
+-----+-----+-----+
```

*Note that PostgreSQL array indexes start with one and not zero.*

# Searching ARRAYSs

```
SELECT
```

```
    email[1][1] as type,  
    email[1][2] as address,  
    test_scores[1]
```

```
FROM grades
```

```
WHERE email[1][1] = 'work';
```

```
+-----+-----+-----+  
| type  | address          | test_scores |  
+-----+-----+-----+  
| work  | work1@datacamp.com | 92          |  
| work  | work2@datacamp.com | 76          |  
+-----+-----+-----+
```

# ARRAY functions and operators

**SELECT**

```
email[2][1] as type,  
email[2][2] as address,  
test_scores[1]
```

**FROM** grades

**WHERE** 'other' = **ANY** (email);

```
+-----+-----+-----+  
| type   | address           | test_scores |  
+-----+-----+-----+  
| other  | other1@datacamp.com | 92          |  
| null   | null              | 76          |  
+-----+-----+-----+
```

# ARRAY functions and operators

**SELECT**

```
email[2][1] as type,  
email[2][2] as address,  
test_scores[1]
```

**FROM** grades

**WHERE** email @> ARRAY['other'];

```
+-----+-----+-----+  
| type   | address           | test_scores |  
+-----+-----+-----+  
| other  | other1@datacamp.com | 92          |  
| null   | null              | 76          |  
+-----+-----+-----+
```

# Let's practice!

FUNCTIONS FOR MANIPULATING DATA IN POSTGRESQL