Creating a database

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Course topics

- Commands for building databases
- PostgreSQL data types
- Database normalization
- Database access management

The PostgreSQL Database Management System

- object-relational database management system
- system components are objects
- database is top-level object

The CREATE DATABASE command

```
CREATE DATABASE db_name;
```

```
CREATE DATABASE my_db;
```

```
CREATE DATABASE _my_db;
```

```
CREATE DATABASE 321_db; -- Invalid
```

Scenarios for new database creation

```
CREATE DATABASE ncaa_bb;
```

```
CREATE DATABASE auto_depot;
```

```
CREATE DATABASE pod;
```

Let's practice!

CREATING POSTGRESQL DATABASES

Creating tables

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

The database table

- Variable number of rows
- Fixed number of columns (structure can be altered)
- Columns have specific data type
- Each row is a record

The CREATE TABLE command

```
CREATE TABLE table_name (
    column1_name column1_datatype [col1_constraints],
    column2_name column2_datatype [col2_constraints],
    ...
    columnN_name columnN_datatype [colN_constraints]
);
```

Name Restrictions

- maximum length of 31 characters
- must begin with letter or underscore ("_")

Example table 1

```
CREATE TABLE school (
    id serial PRIMARY KEY,
    name TEXT NOT NULL,
    mascot_name TEXT
);
```

Example table 2

```
CREATE TABLE topic (
    id SERIAL PRIMARY KEY,
    description TEXT NOT NULL
);
```

Table organization

- Which fields should I use?
- How many tables should I add?
- Which data types are best to use for the fields of my table?

Let's practice!

CREATING POSTGRESQL DATABASES

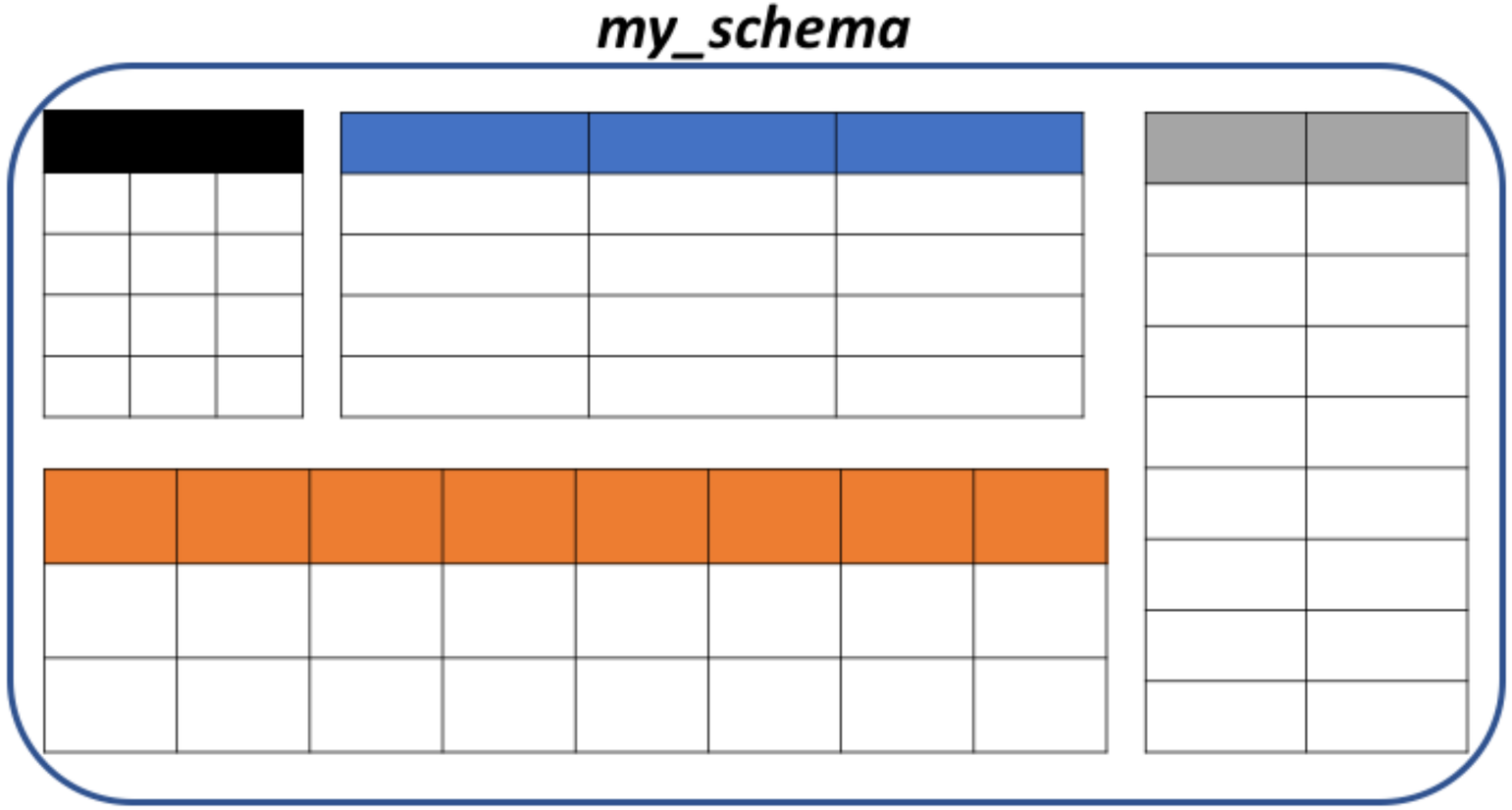
Creating schemas

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

PostgreSQL schemas

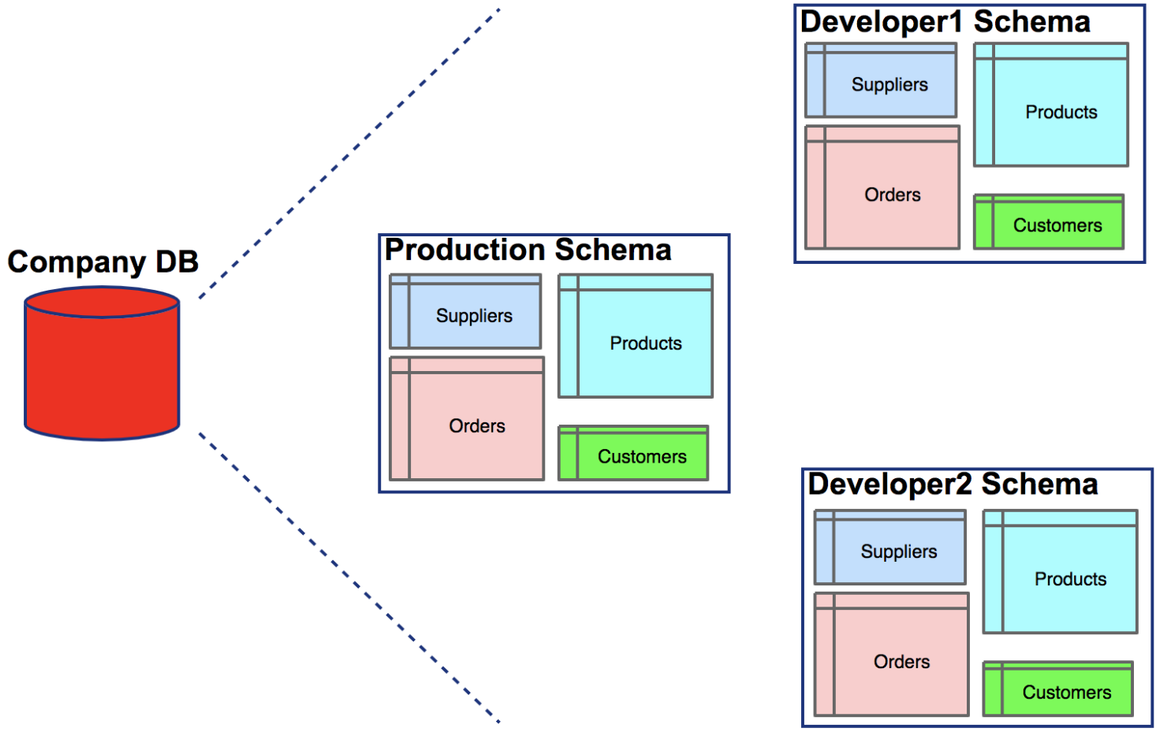
- A named container for tables



[1] https://www.postgresql.org/docs/9.1/ddl-schemas.html

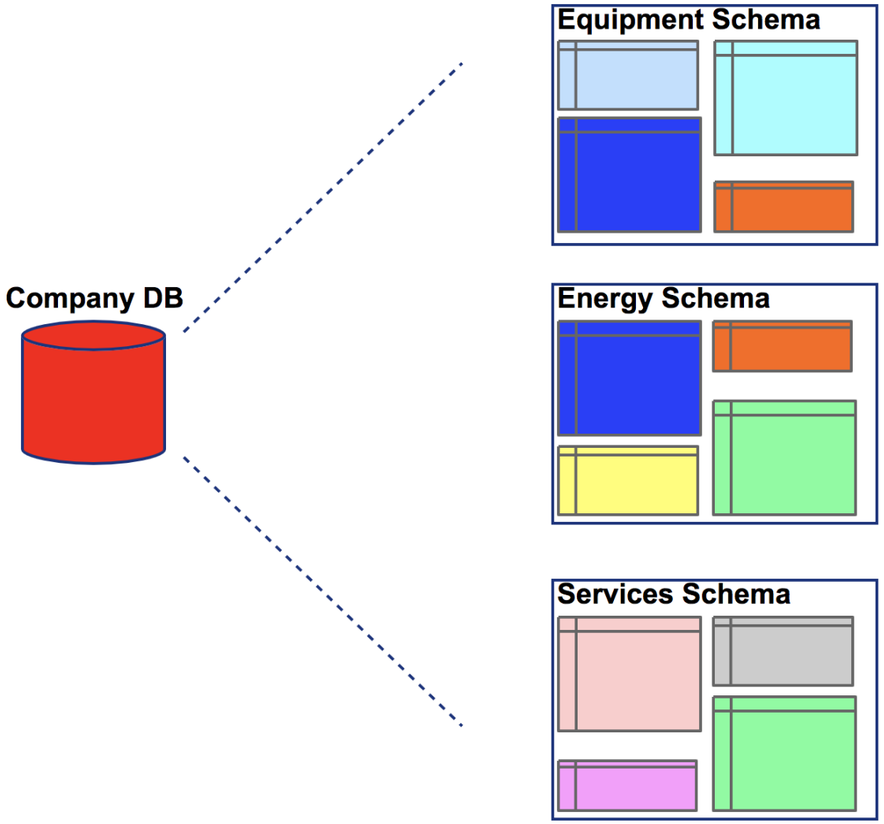
Schema uses

Providing database users with separate environments



Schemas uses

Organizing database objects into related groups



The default schema

- The `public` schema is the default schema in PosgreSQL

```
CREATE TABLE topic (
    id serial PRIMARY KEY,
    description TEXT NOT NULL
);
```

`public.topic`

The CREATE SCHEMA command

```
CREATE SCHEMA schema_name;
```

```
CREATE SCHEMA division1;
```

```
CREATE TABLE division1.school (
    id serial PRIMARY KEY,
    name TEXT NOT NULL,
    mascot_name TEXT,
    num_scholarships INTEGER DEFAULT 0
);
```

Schema naming restrictions

- Length of name less than 32
- Name begins with letter or underscore ("_")
- Schema name cannot begin with "pg_"

Let's practice!

CREATING POSTGRESQL DATABASES

Introduction to PostgreSQL data types

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Data categories in PostgreSQL

- Text
- Numeric
- Temporal
- Boolean
- Others: Geometric, Binary, Monetary

Example 1: representing birthdays

- Cathy: May 3rd, 2006
- Possible representations
  - "May 3, 2006" (text)
  - "5/3/2006" (text)
  - 2006-05-03 (date)

Example 2: tracking payment status

- Did attending member pay?
- Possible representations:
  - "Yes"/"No" (text)
  - "Y"/"N" (text)
  - 'true'/'false' (boolean)
- Specific types provide restriction on values

Example 3: trip distances

- Mark flew 326 miles for client meeting
- Possible representations:
  - "326 miles" (text)
  - "326" (text)
  - 326 (numeric)

Let's practice!

CREATING POSTGRESQL DATABASES

Defining text columns

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Using text in PostgreSQL

```
CREATE TABLE book (
    isbn CHAR(13) NOT NULL,
    author_first_name VARCHAR(50) NOT NULL,
    author_last_name VARCHAR(50) NOT NULL,
    content TEXT NOT NULL
);
```

Text data types: `TEXT`, `VARCHAR(N)`, `CHAR(N)`

The TEXT data type

- Strings of variable length
- Strings of unlimited length
- Good for text-based values of unknown length

The VARCHAR data type

- Strings of variable length
- Strings of unlimited length
- Restriction can be imposed on column values
  - `VARCHAR(N)`
  - *N* - maximum number of characters stored
  - Column can store strings with less than *N* characters
  - Inserting string longer than *N* is error
- `VARCHAR` without *N* specified equivalent to `TEXT`

```
first_name VARCHAR(50) NOT NULL;
```

The CHAR data type

- `CHAR(N)` values consist of exactly *N* characters
- Strings are right-padded with spaces
- `CHAR` equivalent to `CHAR(1)`

```
isbn CHAR(13) NOT NULL;
```

Let's practice!

CREATING POSTGRESQL DATABASES

Defining numeric data columns

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Numeric data with discrete values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL
}
```

Numeric data with discrete values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL,
    num_sales INTEGER
}
```

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767
`INTEGER`	typical choice for integer	-2147483648 to +2147483647

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767
`INTEGER`	typical choice for integer	-2147483648 to +2147483647
`BIGINT`	large-range integer	-9223372036854775808 to 9223372036854775807

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SMALLINT`	small-range integer	-32768 to +32767
`INTEGER`	typical choice for integer	-2147483648 to +2147483647
`BIGINT`	large-range integer	-9223372036854775808 to 9223372036854775807
`SERIAL`	autoincrementing integer	1 to 2147483647

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Integer types

Type	Description	Range
`SERIAL`	autoincrementing integer	1 to 2147483647
`BIGSERIAL`	large autoincrementing integer	1 to 9223372036854775807

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Numeric data with continuous values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL,
    num_sales INTEGER
}
```

Numeric data with continuous values

```
CREATE TABLE people.employee {
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(10) NOT NULL,
    last_name VARCHAR(10) NOT NULL,
    num_sales INTEGER,
    salary DECIMAL(8,2) NOT NULL
}
```

`DECIMAL (precision, scale)`

Floating-point types

Type	Description	Range
`DECIMAL` or `NUMERIC`	user-specified precision	131072 digits before the decimal point;16383 digits after the decimal point

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Floating-point types

Type	Description	Range
`DECIMAL` (`NUMERIC`)	user-specified precision	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
`REAL`	variable-precision	6 decimal digits precision

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Floating-point types

Type	Description	Range
`DECIMAL` (`NUMERIC`)	user-specified precision	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
`REAL`	variable-precision	6 decimal digits precision
`DOUBLE PRECISION`	variable precision	15 decimal digits precision

[1] https://www.postgresql.org/docs/9.1/datatype-numeric.html

Let's practice!

CREATING POSTGRESQL DATABASES

Defining boolean and temporal data columns

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Boolean and temporal data

```
CREATE TABLE book (
    isbn CHAR(13) NOT NULL,
    author_first_name VARCHAR(50) NOT NULL,
    author_last_name VARCHAR(50) NOT NULL,
    content TEXT NOT NULL
);
```

Boolean and temporal data

```
CREATE TABLE book (
    isbn CHAR(13) NOT NULL,
    author_first_name VARCHAR(50) NOT NULL,
    author_last_name VARCHAR(50) NOT NULL,
    content TEXT NOT NULL,
    originally_published DATE NOT NULL,
    out_of_print BOOLEAN DEFAULT FALSE
);
```

The BOOLEAN data type

- Three possible values
  - `true` state
  - `false` state
  - `NULL` (unknown state)
- Common for representing yes-or-no scenarios
- Can be defined with keyword `BOOL` or `BOOLEAN`

```
in_stock BOOL DEFAULT TRUE;
```

Temporal data types

Type	Descriptions	Format
`TIMESTAMP`	represents a date and time	2010-09-21 15:47:16
`DATE`	represents a date	1972-07-08
`TIME`	represents a time	05:30:00

Let's practice!

CREATING POSTGRESQL DATABASES

The importance of data normalization

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Example 1: redundant data

- Data redundancy can be problematic

```
CREATE TABLE loan (
    borrower_id INTEGER REFERENCES borrower(id),
    bank_name VARCHAR(50) DEFAULT NULL,
    ...
);
```

```
CREATE TABLE bank (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) DEFAULT NULL,
    ...
);
```

Example 1: redundant data

```
CREATE TABLE loan (
    borrower_id INTEGER REFERENCES borrower(id),
    bank_name VARCHAR(50) DEFAULT NULL,
    ...
);
```

```
CREATE TABLE bank (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) DEFAULT NULL,
    ...
);
```

- Problem 1: Different banks/same name
- Problem 2: Name changes

Example 1: redundant data

```
CREATE TABLE loan (
    borrower_id INTEGER REFERENCES borrower(id),
    bank_id INTEGER REFERENCES bank(id),
    ...
);
```

- Banks share name with distinct ids
- Updates to bank names will only affect bank table

Example 2: consolidating records

`applicant`

id	name
1	Jane Simmmons
2	Rick Demps
3	Pam Jones

`borrower`

id	name
1	Jack Smith
2	Sara Williams
3	Jennifer Valdez

Example 2: consolidating records

`applicant`

id	name
1	Jane Simmmons
2	Rick Demps
3	Pam Jones

`borrower`

id	name
1	Jack Smith
2	Sara Williams
3	Jennifer Valdez
4	Pam Jones

Example 2: consolidating records

`applicant`

id	name
1	Jane Simmmons
2	Rick Demps
3	Pam Jones

`borrower`

id	name
1	Jack Smith
2	Sara Williams
3	Jennifer Valdez

Example 2: consolidating records

`applicant`

id	name
1	Jane Simmmons
2	Rick Demps

`borrower`

id	name
1	Jack Smith
2	Sara Williams
3	Jennifer Valdez
4	Pam Jones

Example 2: consolidating records

```
CREATE TABLE borrower (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);
```

Example 2: consolidating records

```
CREATE TABLE borrower (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    approved BOOLEAN DEFAULT NULL
);
```

- `approved` is `NULL` => applicant
- `approved` is `true` => borrower
- `approved` is `false` => denied application

Why normalize data?

- Reduces data duplication
- Increases data consistency
- Improves data organization

Let's practice!

CREATING POSTGRESQL DATABASES

1st Normal Form

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Example: maintaining student records

```
CREATE TABLE student (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    courses VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

- Update errors
- Insertion errors
- Deletion errors

Example: duplicated data after update

id	name	courses	home_room
122	Susan Roth	Algebra I, Physics, Spanish II	101
413	Robert Cruz	History, Geometry, Biology	204
613	Thomas Wright	English III, Chemistry, Algebra II	102

Example: duplicated data after update

id	name	courses	home_room
122	Susan Roth	Algebra I, Chemistry, Spanish II	101
413	Robert Cruz	History, Geometry, Biology	204
613	Thomas Wright	English III, Chemistry, Algebra II	102

Example: duplicated data after update

id	name	courses	home_room
122	Susan Roth	Algebra I, Chemistry, Spanish II, Chemistry	101
413	Robert Cruz	History, Geometry, Biology	204
613	Thomas Wright	English III, Chemistry, Algebra II	102

Example: insertions with column restrictions

```
CREATE TABLE student (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    courses VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

id	name	courses	home_room
122	Susan Roth	Algebra I, Physics, Spanish II	101
413	Robert Cruz	History, Geometry, Biology	204
613	Thomas Wright	English III, Chemistry, Algebra II	102

Example: insertions with column restrictions

```
CREATE TABLE student (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    courses VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

id	name	courses	home_room
122	Susan Roth	Algebra I, Physics, Spanish II	101
413	Robert Cruz	History, Geometry, Biology, French Literature	204
613	Thomas Wright	English III, Chemistry, Algebra II	102

Example: data integrity impacted by deleting records

id	name	courses	home_room
122	Susan Roth	Algebra I, Physics, Spanish II	101
413	Robert Cruz	History, Geometry, Biology	204
613	Thomas Wright	English III, Chemistry, Algebra II	102

Example: data integrity impacted by deleting records

id	name	courses	home_room
122	Susan Roth	Algebra I, Physics, Spanish II	101
413	Robert Cruz	History, Geometry, Biology	204
613	Thomas Wright	???	102

Satisfying 1st Normal Form (1NF)

- 1NF Requirement:
  - Table values must be atomic

Example: student table satisfying 1NF

```
CREATE TABLE student (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    courses VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

Example: student table satisfying 1NF

```
CREATE TABLE student (
    id INTEGER,
    name VARCHAR(50) NOT NULL,
    courses VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

Example: student table satisfying 1NF

```
CREATE TABLE student (
    id INTEGER,
    name VARCHAR(50) NOT NULL,
    course VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

Example: student table satisfying 1NF

id	name	course	home_room
122	Susan Roth	Algebra I	101
122	Susan Roth	Physics	101
122	Susan Roth	Spanish II	101
413	Robert Cruz	History	204
413	Robert Cruz	Geometry	204
413	Robert Cruz	Biology	204

Example: student table satisfying 1NF

```
CREATE TABLE student (
    id INTEGER,
    name VARCHAR(50) NOT NULL,
    course VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

Example: student table satisfying 1NF

```
CREATE TABLE student (
    student_id INTEGER,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    course VARCHAR(50) NOT NULL,
    home_room SMALLINT NOT NULL
);
```

Example: student table satisfying 1NF

id	first_name	last_name	course	home_room
122	Susan	Roth	Algebra I	101
122	Susan	Roth	Physics	101
122	Susan	Roth	Spanish II	101
413	Robert	Cruz	History	204
413	Robert	Cruz	Geometry	204
413	Robert	Cruz	Biology	204

Let's practice!

CREATING POSTGRESQL DATABASES

2nd Normal Form

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Example: school textbooks

```
CREATE TABLE textbook (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    publisher_name VARCHAR(100) NOT NULL,
    publisher_site VARCHAR(50),
    quantity SMALLINT NOT NULL DEFAULT 0
);
```

Example: school textbooks

id	title	publisher_name	publisher_site	quantity
23	Introductory Algebra: 1st Edition	ABC Publishing	www.abc.com	32
74	Calculus Foundations	ABC Publishing	www.abc.com	27
112	Statistical Concepts	Martin House	www.mh.com	22

Example: inconsistency from updating url

id	title	publisher_name	publisher_site	quantity
23	Introductory Algebra: 1st Edition	ABC Publishing	www.abc.com	32
74	Calculus Foundations	ABC Publishing	www.abc.com	27
112	Statistical Concepts	Martin House	www.mh.com	22

Example: inconsistency from updating url

id	title	publisher_name	publisher_site	quantity
23	Introductory Algebra: 1st Edition	ABC Publishing	www.newabc.com	32
74	Calculus Foundations	ABC Publishing	www.abc.com	27
112	Statistical Concepts	Martin House	www.mh.com	22

Example: adding publisher without textbook

id	title	publisher_name	publisher_site	quantity
23	Introductory Algebra: 1st Edition	ABC Publishing	www.abc.com	32
74	Calculus Foundations	ABC Publishing	www.abc.com	27
112	Statistical Concepts	Martin House	www.mh.com	22

Example: adding publisher without textbook

id	title	publisher_name	publisher_site	quantity
23	Introductory Algebra: 1st Edition	ABC Publishing	www.abc.com	32
74	Calculus Foundations	ABC Publishing	www.abc.com	27
112	Statistical Concepts	Martin House	www.mh.com	22
??	??	New Horizons	www.nhorizon.com	??

Example: removing a textbook

id	title	publisher_name	publisher_site	quantity
23	Introductory Algebra: 1st Edition	ABC Publishing	www.abc.com	32
74	Calculus Foundations	ABC Publishing	www.abc.com	27
112	Statistical Concepts	Martin House	www.mh.com	22

Example: removing a textbook

id	title	publisher_name	publisher_site	quantity
23	Introductory Algebra: 1st Edition	ABC Publishing	www.abc.com	32
74	Calculus Foundations	ABC Publishing	www.abc.com	27

- Publisher requires separate table
- Data anomalies from insertions and deletions

Satisfying 2nd Normal Form (2NF)

- 1NF is satisfied
- All non-key columns are dependent on the table's `PRIMARY KEY`

Example: textbooks and publishers in 2NF

```
CREATE TABLE textbook (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    publisher_name VARCHAR(100) NOT NULL,
    publisher_site VARCHAR(50),
    quantity SMALLINT NOT NULL DEFAULT 0
);
```

Example: textbooks and publishers in 2NF

```
CREATE TABLE textbook (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    quantity SMALLINT NOT NULL DEFAULT 0,
);
```

```
CREATE TABLE publisher (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    site VARCHAR(50)
);
```

Example: textbooks and publishers in 2NF

```
CREATE TABLE textbook (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    quantity SMALLINT NOT NULL DEFAULT 0,
    publisher_id INTEGER REFERENCES publisher(id)
);
```

```
CREATE TABLE publisher (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    site VARCHAR(50)
);
```

Let's practice!

CREATING POSTGRESQL DATABASES

3rd Normal Form

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Defining 3rd Normal Form

Requirements

- 2NF is satisfied
- No "transitive dependencies" exist
  - i.e., All non-key columns are only dependent on the `PRIMARY KEY`

Transitive dependencies

- Involve 3 columns in table
- Columns X, Y, Z
- column X -> column Y
- column Y -> column Z
- column X -> column Z

Example: course room assignments

id	name	teacher	num
157	Algebra	Maggie Winters	244
162	Physics	Maggie Winters	244
321	Spanish I	Jeremy Smith	309
497	History I	Sarah Williams	313
613	Spanish II	Jeremy Smith	309

- course name -> teacher
- teacher -> room number
- course name -> room number

Example: course room assignments

id	name	teacher	num
157	Algebra	Maggie Winters	244
162	Physics	Maggie Winters	244
321	Spanish I	Jeremy Smith	309
497	History I	Sarah Williams	313
613	Spanish II	Jeremy Smith	309

- course name -> teacher
- teacher -> room number
- course name -> room number
  (transitive dependency)

Example: course room assignments

id	name	teacher	num
157	Algebra	Maggie Winters	244
162	Physics	Maggie Winters	244
321	Spanish I	Jeremy Smith	309
497	History I	Sarah Williams	313
613	Spanish II	Jeremy Smith	309

1. Updating room number

Example: course room assignments

id	name	teacher	num
157	Algebra	Maggie Winters	244
162	Physics	Maggie Winters	244
321	Spanish I	Jeremy Smith	309
497	History I	Sarah Williams	313
613	Spanish II	Jeremy Smith	309

1. Updating room number
2. Adding new teachers

Example: course room assignments

id	name	teacher	num
157	Algebra	Maggie Winters	244
162	Physics	Maggie Winters	244
321	Spanish I	Jeremy Smith	309
497	History I	Sarah Williams	313
613	Spanish II	Jeremy Smith	309

1. Updating room number
2. Adding new teachers
3. Deleting all courses for a teacher

Example: course room assignments

How do we change the structure of our data in order to alleviate these potential problems?

Example: course room assignments

`teacher` table

id	name	room_num
1	Maggie Winters	244
2	Jeremy Smith	309
3	Sarah Williams	313

Example: course room assignments

`teacher` table

id	name	room_num
1	Maggie Winters	244
2	Jeremy Smith	309
3	Sarah Williams	313

`course_assignment` table

id	name	teacher_id
157	Algebra	1
162	Physics	1
321	Spanish I	2
497	History I	3
613	Spanish II	2

Let's practice!

CREATING POSTGRESQL DATABASES

Introduction to access control

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

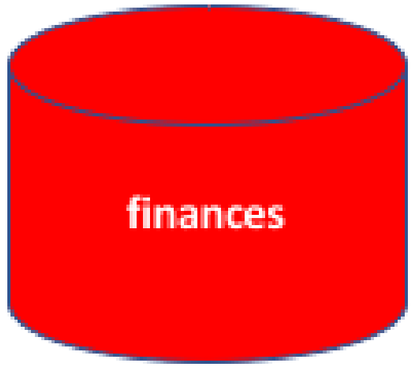
datacamp

The default superuser

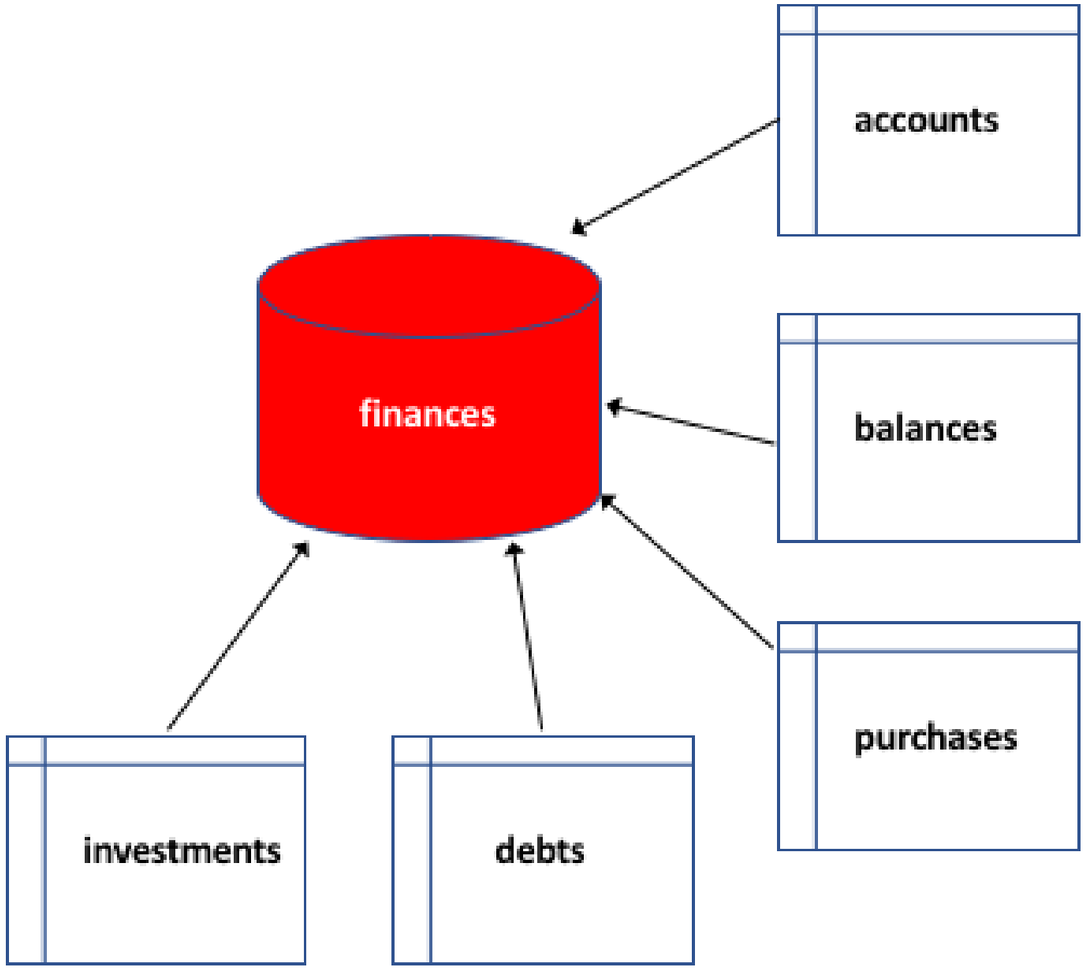
- `postgres` "superuser" role
- Administers database
- `postgres` privileges
  - Creating databases
  - Dropping databases
  - Inserting records
  - Deleting records
  - Dropping tables
- `postgres` user should be used with care

Example: a personal finance database

- Creation of `finances` database



Example: a personal finance database



Example: a personal finance database

- Database is personal and not publicly accessible
- User with restricted access should be created
- User abilities:
  - Adding records
  - Querying records
  - Editing records

Creating new users

- `CREATE USER`
  - Used to generate a new account

```
CREATE USER newuser;
```

- `newuser` can create tables in database
- No access to tables created by other users

Setting user password

- Passwords enhance security
- No passwords by default

```
CREATE USER newuser WITH PASSWORD 'secret';
```

```
ALTER USER newuser WITH PASSWORD 'new_password';
```

Let's practice!

CREATING POSTGRESQL DATABASES

PostgreSQL access privileges

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

PostgreSQL roles and privileges

- Users are a type of role
- Group roles can also be defined
- Database object access given to roles

The GRANT command

- Privileges are "granted" to roles by owner
- The `GRANT` command bestows privileges
- Many privileges can be granted including:
  - `SELECT`
  - `DELETE`
  - `UPDATE`

```
GRANT p ON obj TO grantee;
```

Example: personal finance database

```
CREATE TABLE account (
    id SERIAL PRIMARY KEY,
    short_name VARCHAR(25),
    provider_id INTEGER REFERENCES provider(id),
    balance DECIMAL
);
```

```
CREATE USER fin WITH PASSWORD '38\5)uk1+3&}*Y';
```

Example: personal finance database

- `fin` user needs access to `account` table
- `fin` access
  - Add new accounts
  - Update accounts
  - Query accounts
- Superuser grants privileges

```
GRANT INSERT ON account TO fin;
```

```
GRANT UPDATE ON account TO fin;
```

```
GRANT SELECT ON account TO fin;
```

Table modification privileges

- Some privileges cannot be granted
- Modifying table requires ownership

```
ALTER TABLE account ADD COLUMN date_opened DATE;
```

```
ALTER TABLE account RENAME COLUMN short_name
TO nickname;
```

```
ALTER TABLE account OWNER TO fin;
```

Let's practice!

CREATING POSTGRESQL DATABASES

Hierarchical access control

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

datacamp

Access control with schemas

- Schema - named container for db objects
- Schemas can be used for access control

Example: schema use in finances database

- Spouse access to `finances` database
- `public` schema used by default
- Two new schemas: `me` and `spouse`

```
CREATE SCHEMA me;
```

```
CREATE SCHEMA spouse;
```

```
CREATE TABLE me.account (...);
```

```
CREATE TABLE spouse.account (...);
```

Granting schema privileges

```
CREATE USER better_half WITH PASSWORD 'changeme';
```

```
GRANT USAGE ON SCHEMA spouse TO better_half;
```

```
GRANT USAGE ON SCHEMA public TO better_half;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA spouse;
TO better_half;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public
TO better_half;
```

Schema-based access control implemented

Using groups

- Group - a type of role that identifies one or more users
- Access control can be applied at group level

```
CREATE GROUP family;
```

```
GRANT USAGE ON SCHEMA public TO family;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA
public TO family;
```

```
ALTER GROUP family ADD USER fin
ALTER GROUP family ADD USER better_half;
```

Shared and individual data access

- Shared schema access enabled to `public` schema
- Individual schemas control data access

Let's practice!

CREATING POSTGRESQL DATABASES

Removing access

CREATING POSTGRESQL DATABASES

Darryl Reeves
Assistant Professor, Long Island University - Brooklyn

datacamp

Example: rolling back privileges

- Cousin interested in databases
- Superuser access mistakenly provided
- Good backup strategy saves the day
- New user account added

```
CREATE USER cousin;
```

```
ALTER GROUP family ADD USER cousin;
```

```
GRANT ALL PRIVILEGES ON finances.* TO cousin;
```

`finances` data deleted again

Example: rolling back privileges

- Privileges removed using `REVOKE` command
- `REVOKE` follows similar format to `GRANT`

```
REVOKE DELETE, TRUNCATE ON finances.* FROM cousin;
```

Example: rolling back privileges

- Privileges can be reset

```
REVOKE ALL PRIVILEGES ON finances.* FROM cousin;
```

```
GRANT SELECT ON finances.* FROM cousin;
```

- `REVOKE` can remove users from groups

```
REVOKE family FROM cousin;
```

Let's practice!

CREATING POSTGRESQL DATABASES

Course wrap-up

CREATING POSTGRESQL DATABASES

Darryl Reeves
Industry Assistant Professor, New York University

Course content

Chapter 1: Structure of PostgreSQL Databases

Chapter 2: PostgreSQL Data Types

Chapter 3: Database Normalization

Chapter 4: Access Control in PostgreSQL

Next steps

- Database objects (e.g. views and functions)
- Data types (e.g. geometric and array-based)
- Normalization (e.g 4NF)
- Access control

Congratulations!

CREATING POSTGRESQL DATABASES