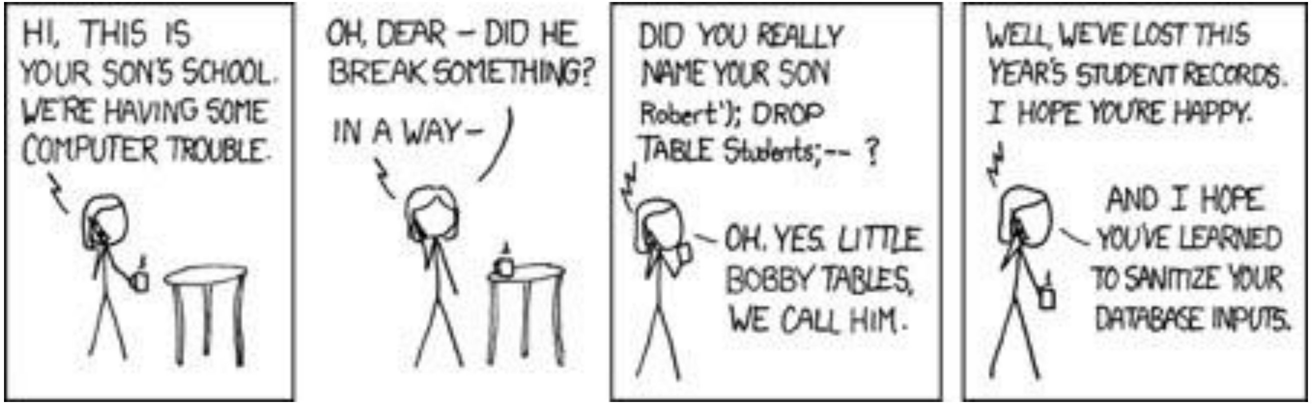


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Module 2-6

Database Design

Objectives

- Database Design Exercise
- Database Definition Language
- Database Control Language

Database Design Exercise

Gallery Customer History Form

Customer Name

Jackson, Elizabeth   Phone (206) 284-6783
123 – 4th Avenue
Fonthill, ON
L3J 4S4

Purchases Made

Artist	Title	Purchase Date	Sales Price
03 - Carol Channing	Laugh with Teeth	09/17/2000	7000.00
15 - Dennis Frings	South toward Emerald Sea	05/11/2000	1800.00
03 - Carol Channing	At the Movies	02/14/2002	5550.00
15 - Dennis Frings	South toward Emerald Sea	07/15/2003	2200.00

The Gill Art Gallery wishes to maintain data on their customers, artists and paintings. They may have several paintings by each artist in the gallery at one time. Paintings may be bought and sold several times. In other words, the gallery may sell a painting, then buy it back at a later date and sell it to another customer.

Normal Forms

Before a single CREATE statement is run, the tables and their relationships need to be well thought out.

Normalization is the process of organizing a database to reduce data redundancy and improve data integrity.

We normalize data to:

1. Avoid duplicate data
2. Fix anomalies
3. Simplify search queries.

Normal Forms: Before normalization

Gallery Customer History Form

Customer Name

Jackson, Elizabeth — Phone (206) 284-6783
123 – 4th Avenue
Fonthill, ON
L3J 4S4

Purchases Made

Artist	Title	Purchase Date	Sales Price
03 - Carol Channing	Laugh with Teeth	09/17/2000	7000.00
15 - Dennis Frings	South toward Emerald Sea	05/11/2000	1800.00
03 - Carol Channing	At the Movies	02/14/2002	5550.00
15 - Dennis Frings	South toward Emerald Sea	07/15/2003	2200.00

The Gill Art Gallery wishes to maintain data on their customers, artists and paintings. They may have several paintings by each artist in the gallery at one time. Paintings may be bought and sold several times. In other words, the gallery may sell a painting, then buy it back at a later date and sell it to another customer.

	A	B	C	D	E
1	**Name**	**Address**	**Phone**	**Purchases(ArtistId, ArtistName, ArtTitle, PurchaseDate, Price)**	
2	Jackson, Elizabeth	123 - 4th Avenue FonthillOl	(206) 284-6783	03, Carol Channing, Laugh with Teeth, 9/17/2000, $7000.00 15, Dennis Frings, South toward Emerald Sea, 5/11/2000, $1800.00 03, Carol Channing, At the Movies, 2/14/2002, $5550.00 15, Dennis Frings, South toward Emerald Sea, 7/15/2003, $2200.00	
3	Smith, John	123 Main StreetNew York, I	(123) 867-5309	02, Rick Steves, Travels through Europe, 9/01/2002, $1050.00	
4					
5					
6					
7					
8					

Normal Forms: 1NF

- No table should contain duplicative columns that one could use to get other types of information.
- Every table should be organized in rows with primary keys that uniquely identify it.

CUSTOMERS

CustomerID (PK)	Name	Address	Phone
1	Jackson, Elizabeth	123 - 4th Avenue FonthillON	(206) 284-6783
2	Smith, John	123 Main StreetNew York, NY 1	(123) 867-5309

CUSTOMER PURCHASES

CustomerID (PK, FK)	ArtCode (PK)	Purchase Date (PK)	ArtistId	ArtistName	Title	Price
1	LWT	9/17/2000	3	Carol Channing	Laugh with Teeth	$7,000.00
1	STES	5/11/2000	15	Dennis Frings	South toward Emerald Sea	$1,800.00
1	ATM	2/14/2002	3	Carol Channing	At the Movies	$5,550.00
1	STES	7/15/2003	15	Dennis Frings	South toward Emerald Sea	$2,200.00
2	TTE	9/1/2002	2	Rick Steves	Travels through Europe	$1,050.00

1NF

In 1st Normal Form, no table should contain duplicative columns (purchase 1, purchase 2, ... purchase n) that one could use to get other types of information.

Every table is organized in rows with primary keys that uniquely identify it.

Normal Forms: 2NF

- Must be in 1NF
- Any non-key attribute must be dependent on the primary key

A	B	C	D
CUSTOMERS			
CustomerID (PK)	**Name**	**Address**	**Phone**
1	Jackson, Elizabeth	123 - 4th Avenue FonthillON	(206) 284-6783
2	Smith, John	123 Main StreetNew York, NY 1	(123) 867-5309

CUSTOMER PURCHASES			
CustomerID (PK, FK)	**ArtCode (PK, FK)**	**Purchase Date (PK)**	**Price**
1	LWT	9/17/2000	$7,000.00
1	STES	5/11/2000	$1,800.00
1	ATM	2/14/2002	$5,550.00
1	STES	7/15/2003	$2,200.00
2	TTE	9/1/2002	$1,050.00

ART			
ArtCode (PK)	**Title**	**ArtistID**	**ArtistName**
LWT	Laughing with Teeth	3	Carol Channing
STES	South toward Emerald Sea	15	Dennis Frings
ATM	At the Movies	3	Carol Channing
TTE	Travels through Europe	2	Rick Steves

2NF
In 2nd Normal Form, the table must already be in 1NF.
Any Non-key Attribute must be dependent on the primary key.

Normal Forms: 3NF

- Must be in 2NF
- No transitive functional dependency – if column A is dependent on column B and column B is dependent on C, then column A is dependent on C

CUSTOMERS

CustomerID (PK)	Name	Address	Phone
1	Jackson, Elizabeth	123 - 4th Avenue FonthillON	(206) 284-6783

CUSTOMER PURCHASES

CustomerID (PK, FK)	ArtCode (PK, FK)	Purchase Date (PK)	Price
1	LWT	9/17/2000	$7,000.00
1	STES	5/11/2000	$1,800.00
1	ATM	2/14/2002	$5,550.00
1	STES	7/15/2003	$2,200.00
2	TTE	9/1/2002	$1,050.00

ART

ArtCode (PK)	Title	ArtistID (FK)
LWT	Laughing with Teeth	3
STES	South toward Emerald Sea	15
ATM	At the Movies	3
TTE	Travels through Europe	2

ARTISTS

ArtistID (PK)	First Name	Last Name
2	Rick	Steves
3	Carol	Channing
15	Dennis	Frings

3NF

In 3rd Normal Form, the table must already be in 2NF.

There cannot be any transitive dependency between columns (e.g. in 2NF Artist Name was dependent on Artist Id which was dependent on ArtCode)

Normal Forms: 3NF

There are several levels above 3NF of “normal form” compliance, but generally the third normal form is good enough for 99% of all situations.

An informal intuitive definition of 3NF is as follows:

There are no fields in a table that are not directly determined by the values of the primary key.

Therefore, all fields in a table should be directly related to (determined by) the primary key of that table.

Normal Forms: 3NF Example

Suppose we have the following table:

InvoiceNumber (PK)	InvoiceDate	Inventory ID	Inventory Description
1000	10/1/2019	45	Hammer
1001	10/3/2019	28	Nails
1002	10/3/2019	17	Screwdriver
1003	10/4/2019	45	Hammer

Some questions to consider:

- Is an invoice date directly related to an invoiceNumber? → Yes
- Is an inventory description directly related to an invoiceNumber? → No

Normal Forms: 3NF Example

Suppose we need a Spanish version of this database, and we need to value to show *Martillo* instead of Hammer. This would entail an UPDATE statement that targets 2 rows.

InvoiceNumber (PK)	InvoiceDate	Inventory ID	Inventory Description
1000	**10/1/2019**	**45**	Martillo
1001	10/3/2019	28	Nails
1002	10/3/2019	17	Screwdriver
1003	**10/4/2019**	**45**	Martillo

Normal Forms: 3NF Example

In this situation, we could have split up the data into 2 tables, thus we end up with a less risky query, affecting only 1 row:

InvoiceNumber (PK)	InvoiceDate	Inventory ID
1000	**10/1/2019**	**45**
1001	10/3/2019	28
1002	10/3/2019	17
1003	**10/4/2019**	**45**

Inventory ID (pk)	Description
28	Nails
17	Screwdriver
45	Martillo

Many to Many relationships

Generally speaking, when there are 2 entities for which there is a “many to many” relationship, we will end up with 3 tables when considering 3NF as part of our design.

Many to Many relationships Example

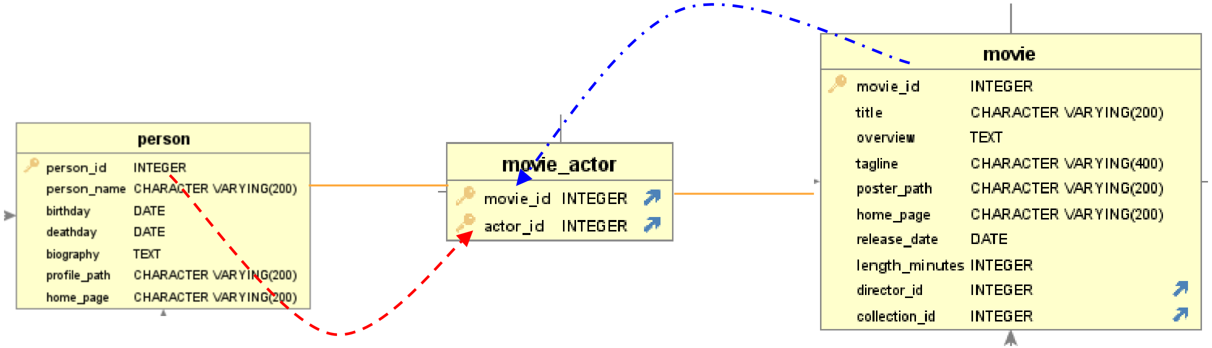
Consider the MovieDB example:

- An actor can be a cast member of several movies.
  - A movie can have several actors.

This is a “many to many” relationship.

Many to Many relationships Example

Consequently we end up with three tables to describe this relationship:



For this relationship to work we have defined two foreign keys in the movie_actor table, the primary keys of each of the other two tables.

DML vs DDL vs TCL

The SQL statements we have seen so far fall into a number of different categories:

- Data Manipulation Language (**DML**): SELECT, INSERT, UPDATE, DELETE
- Data Definition Language (**DDL**): CREATE, ALTER, DROP
- Transaction Control Language (TCL): BEGIN, TRANSACTION, COMMIT
- Data Control Language (DCL): GRANT, REVOKE

The focus of this lecture will be DDL statements with appropriate constraints.

Creating Tables Example

We are now ready to evaluate the syntax for table creation and alteration. This is the Create table syntax for all 3 of the previous tables:

```
CREATE TABLE person (
    person_id serial NOT NULL,
    person_name varchar(200) NOT NULL,
    birthday DATE NULL,
    CONSTRAINT pk_person PRIMARY KEY
(person_id)
);
```

In movie_actor are actor_id and movie_id foreign keys yet?

No!

```
CREATE TABLE movie_actor (
    actor_id integer NOT NULL,
    movie_id integer NOT NULL,
    CONSTRAINT pk_movie_actor PRIMARY KEY (actor_id, movie_id)
);
```

```
CREATE TABLE movie (
    movie_id int NOT NULL DEFAULT nextval('movie_serial'),
    title varchar(200) NOT NULL,
    overview text NULL,
    tagline varchar(400) NULL,
    poster_path varchar(200) NULL,
    home_page varchar(200) NULL,
    release_date date NULL,
    length_minutes int NOT NULL,
    director_id int NULL,
    collection_id int NULL,
            CONSTRAINT pk_movie PRIMARY KEY (movie_id)
);
```

Creating Tables Example

We finish by specifying that actor_id and film_id are actually foreign keys. The DBMS does not assume this just because it has the same name, we must use the ALTER command:

```
ALTER TABLE movie_actor
ADD FOREIGN KEY(movie_id)
REFERENCES movie(movie_id);

ALTER TABLE movie_actor
ADD FOREIGN KEY(actor_id)
REFERENCES
person(person_id);
```

CREATE/DROP syntax

```
CREATE DATABASE database_name;
DROP DATABASE database_name;
```

```
CREATE TABLE table_name
(
   column_name1 data_type(size),
   column_name2 data_type(size) NOT NULL,
   column_name3 data_type(size),
   CONSTRAINT pk_column_1 PRIMARY KEY (column_name1),
   CONSTRAINT fk_column_2 FOREIGN KEY (column_name2)
      REFERENCES table_name(column_1)
);
```

ALTER syntax

```
ALTER TABLE table_name
   ADD CONSTRAINT pk_constraint_name
   PRIMARY KEY (column_name(s));

ALTER TABLE table_name
   ADD CONSTRAINT fk_constraint_name
   FOREIGN KEY (column_name) REFERENCES
   table(column_name);

ALTER TABLE table_name
   ADD CONSTRAINT chk_constraint_name
   CHECK (column_name = 'value' OR
   column_name IN (values));
```

Sequences

Sequences are incrementing numbers that are commonly used as Surrogate Primary Keys. Start at 0, unless given a starting value. Never stops incrementing.

Creating a Sequence manually:

```
CREATE SEQUENCE custom_seq;
```

Getting the next number manually:

```
SELECT nextval('custom_seq');
```

Sequences are not affected by a *rollback* of a transaction.

Creating sequence when creating a table:

```
column_name serial
```

Getting the next number automatically:

```
INSERT… (serial_col) VALUES (DEFAULT);
```

Or don't include the column in the Insert, and it will create and populate it automatically.

DCL (Database Control Language)

Database Control Language (DCL) is used to administer the database, users, and permissions.

GRANT - gives access to a specific action for a resource to a user.

REVOKE - removes access to specific action for a resource from a user.

Data Control Language (DCL)

DCL commands deal with the permissions, rights and other controls for the database system.

- CREATE USER – Allows the creation of a user to the database
  - Users have permission to log in to the database by default
- CREATE ROLE – Allows the creation of a role to the database
  - Roles do not have access to log in to the database (but can be granted this)
- GRANT – allow a role or user access privileges to a database or table
- ALTER ROLE – allows a role to be modified
- REVOKE – remove access privileges to a database or table

Data Control Language (DCL)

DCL commands deal with the permissions, rights and other controls for the database system. Examples are GRANT and REVOKE

```
GRANT privileges
ON object
TO user;

GRANT ALL
  ON Movie_db
  TO margaret;
```

Provides user access privileges to the database

Data Control Language (DCL)

DCL commands deal with the permissions, rights and other controls fo the database system. Examples are GRANT and REVOKE

```
REVOKE privileges
ON object
FROM user;

REVOKE INSERT
  ON Movie_db
  FROM margaret;
```

Removes user access privileges to the database

Data Control Language (DCL)

To change attributes of a role, you use the following form of `ALTER ROLE` statement:

```
ALTER ROLE role_name [WITH] option;
```

The option can be:

- `SUPERUSER` | `NOSUPERUSER` – determine if the role is a `superuser` or not.
- `CREATEDB` | `NOCREATEDB` – allow the role to create new databases.
- `CREATEROLE` | `NOCREATEROLE` – allow the role to create or change roles.
- `INHERIT` | `NOINHERIT` – determine if the role to inherit privileges of roles of which it is a member.
- `LOGIN` | `NOLOGIN` – allow the role to log in.
- `REPLICATION` | `NOREPLICATION` – determine if the role is a replication roles.
- `BYPASSRLS` | `NOBYPASSRLS` – determine if the role to by pass a row-level security (RLS) policy.
- `CONNECTION LIMIT limit` – specify the number of concurrent connection a role can made, -1 means unlimited.
- `PASSWORD 'password'` | `PASSWORD NULL` – change the role's password.
- `VALID UNTIL 'timestamp'` – set the date and time after which the role's password is no long valid.

Objectives

https://en.wikipedia.org/wiki/Database_normalization

- Database Design

Normal forms [edit]

Codd introduced the concept of normalization and what is now known as the first normal form (1NF) in 1970.[4] Codd went on to define the second normal form (2NF) and third normal form (3NF) in 1971,[5] and Codd and Raymond F. Boyce defined the Boyce–Codd normal form (BCNF) in 1974.[6]

Informally, a relational database relation is often described as "normalized" if it meets third normal form.[7] Most 3NF relations are free of insertion, updation, and deletion anomalies.

The normal forms (from least normalized to most normalized) are:

- UNF: Unnormalized form
- 1NF: First normal form
- 2NF: Second normal form
- 3NF: Third normal form
- EKNF: Elementary key normal form
- BCNF: Boyce–Codd normal form
- 4NF: Fourth normal form
- ETNF: Essential tuple normal form
- 5NF: Fifth normal form
- DKNF: Domain-key normal form
- 6NF: Sixth normal form

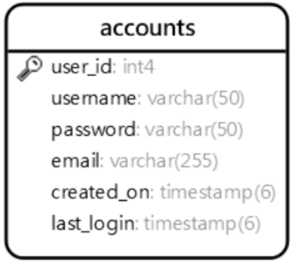
	UNF (1970)	1NF (1970)	2NF (1971)	3NF (1971)	EKNF (1982)	BCNF (1974)	4NF (1977)	ETNF (2012)	5NF (1979)	DKNF (1981)	6NF (2003)
Primary key (no duplicate tuples)[4]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Atomic columns (cells cannot have tables as values)[5]	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either does not begin with a proper subset of a candidate key or ends with a prime attribute (no partial functional dependencies of non-prime attributes on candidate keys)[5]	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with a prime attribute (no transitive functional dependencies of non-prime attributes on candidate keys)[5]	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
Every non-trivial functional dependency either begins with a superkey or ends with an elementary prime attribute	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	N/A
Every non-trivial functional dependency begins with a superkey	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	N/A
Every non-trivial multivalued dependency begins with a superkey	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	N/A
Every join dependency has a superkey component[8]	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	N/A
Every join dependency has only superkey components	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	N/A
Every constraint is a consequence of domain constraints and key constraints	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Every join dependency is trivial	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Objectives

- Database Design Exercise
- Database Definition Language

```
CREATE TABLE accounts (
        user_id serial PRIMARY KEY,
        username VARCHAR ( 50 ) UNIQUE NOT NULL,
        password VARCHAR ( 50 ) NOT NULL,
        email VARCHAR ( 255 ) UNIQUE NOT NULL,
        created_on TIMESTAMP NOT NULL,
        last_login TIMESTAMP
);
```

```
CREATE TABLE [IF NOT EXISTS] table_name (
   column1 datatype(length) column_contraint,
   column2 datatype(length) column_contraint,
   column3 datatype(length) column_contraint,
   table_constraints
);
```



Objectives

- Database Design Exercise
- Database Definition Language
- Database Control Language

Data Control (DCL) Commands – PostgreSQL Tutorial

This section consists of those commands which are used to control privileges in the database. The commands are:

- GRANT
- REVOKE

GRANT

The GRANT command is used to provide user access privileges or other privileges for the schema.

Syntax:

```
GRANT privileges ON object TO user;
```

Example:

```
GRANT INSERT ON TeachersInfo TO PUBLIC;
```

REVOKE

The REVOKE command is used to withdraw user's access privileges given by using the GRANT command.

Syntax:

```
REVOKE privileges ON object FROM user;
```

Example:

```
REVOKE INSERT ON TeachersInfo FROM PUBLIC;
```