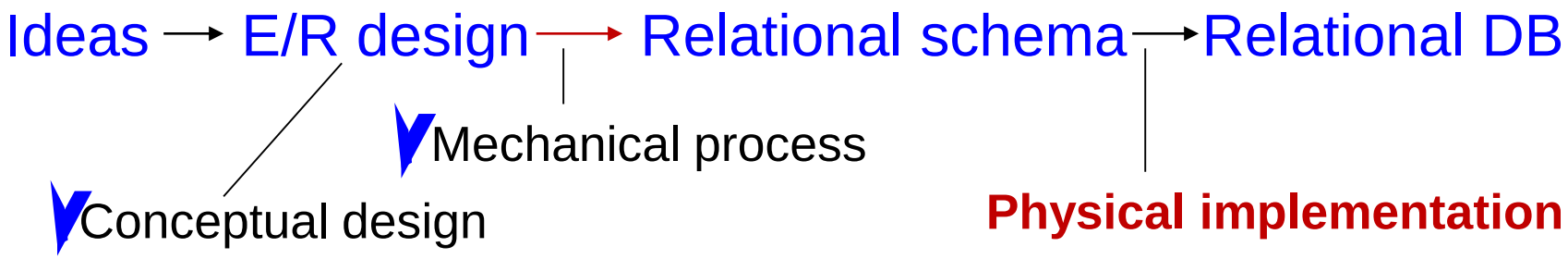
By *Marina Barsky*

Converting logical schema into physical tables

Lecture 3A



Using DBMS: PostgreSQL

- Powerful object-relational database management system (ORDBMS)
- Open source, originally developed at the University of California at Berkeley CS Department.
- Pioneered many concepts that only became available in some commercial database systems much later.
- Because of the liberal license, PostgreSQL can be used, modified, and distributed by anyone free of charge for any purpose, be it private, commercial, or academic.

2-tier client-server architecture

The DBMS software is running on Database server.

Your interaction with database consists of 2 processes:

- A server process: manages the database files, maintains connection pool, performs database actions on behalf of clients.
- The client (frontend) application: a text-oriented tool, a graphical application, a web server that accesses the database to display web pages, or a specialized database maintenance tool.

Note: The client and the server can be on different hosts. They communicate over a TCP/IP network connection. The files that can be accessed on a client machine might not be accessible on the database server machine.

Connecting to DB server

- Login to CDF
- ssh into dbsrv1.cdf.toronto.edu
- Now you are connected to the DB server as u_name.
- The databases are already created for each student, and they have name: csc343h-u_name.
- Each student has its own single database

Interactive shell client

- Connect to your specific database:

psql csc343h-u_name

- You see the following prompt:

csc343h-u_name=>

- You are now connected and you can enter sql commands

Schema in PostgreSQL

- A database contains one or more named schemas, which in turn contain tables.
- To create or access objects in a schema, write a *qualified name* consisting of the schema name and table name separated by a dot:

schema.table

- There is a default schema called *public*, for which you don't need to specify the qualified name, only the name of the table

Documentation: http://www.postgresql.org/docs/9.1/static/ddl-schemas.html

PostgreSQL – SQL standards

- PostgreSQL supports most of the major features of SQL:2003.
- Out of 164 mandatory features required for full Core conformance, PostgreSQL conforms to at least 150.
- In addition, there is a long list of supported optional features. (No current version of any database management system claims full conformance to Core SQL:2003).

SQL syntax is very similar to MySQL and Oracle

SQL tutorials: http://www.postgresql.org/docs/9.6/static/tutorial-sql.html

Data Definition Language (DDL): converitng Schema into physical tables

CREATE TABLE **table_name**
(
    column_name1 *data_type*,
    column_name2 *data_type*,
    column_name3 *data_type*,
    ....
)

Create Table

```
CREATE TABLE Movies (
   title VARCHAR(50),
   year INT,
   length INT,
   rating CHAR(2),
   studioname VARCHAR(20)
);

CREATE TABLE Studios(
   name VARCHAR(20),
   website VARCHAR(255)
);
CREATE TABLE Stars (
   name VARCHAR(20),
   gender CHAR(1),
   birthyear INT,
   birthplace VARCHAR(40)
);
```

Data types:

- **NUMERIC (precision, scale) :**
  - scale - count of decimal digits in the fractional part, to the right of the decimal point.
  - precision - the total count of significant digits in the whole number
- **CHAR(n)** allocates a fixed space, and if the string that we store is shorter than **n**, then it is padded with blanks.
- Differently, **VARCHAR(n)** denotes a string of up to n characters.
- CHAR has better performance. Use CHAR(n) for frequently used fields, and use VARCHAR(n) otherwise.
- Default date format: '1994-11-28'

Declaring primary keys

```
DROP TABLE IF EXISTS Movies;
DROP TABLE IF EXISTS Studios;

CREATE TABLE Studios (
   name VARCHAR(20) PRIMARY KEY,
   address VARCHAR(255)
);

CREATE TABLE Movies (
   title VARCHAR(20),
   year INT,
   length INT,
   rating CHAR(2),
   studioname VARCHAR(20),
   PRIMARY KEY (title, year)
);
```

Insert

```
INSERT INTO Movies
VALUES('Walk the Line', 2005, 136, 'PG', 'Fox');

INSERT INTO Movies
VALUES('Pretty Woman', 1990, 119, 'R', 'Disney');

INSERT INTO Movies
VALUES('Wayne''s World', 1991, 104, 'PG', 'Paramount');

INSERT INTO Movies
VALUES('Unfaithful', 2002, 124, 'R', 'Fox');

INSERT INTO Movies
VALUES('Runaway Bride', 1999, 116, 'PG', 'Paramount');

INSERT INTO Movies
VALUES('The Princess and the Frog', 2009, 97, 'G', 'Disney');
```

Altering, Dropping

```
ALTER TABLE Stars ADD [COLUMN] phone
   CHAR(16);


ALTER TABLE Stars ALTER COLUMN phone TYPE
   CHAR(26);


ALTER TABLE Stars DROP COLUMN phone;


DROP TABLE Stars;

DROP TABLE Movies;

DROP TABLE Studios;
```

Getting information about tables

- Describe all tables:

```
\dt
```

```
  List of relations
 Schema |    Name    | Type  |  Owner
--------+------------+-------+----------
 public | movie      | table | mgbarsky
 public | movie_exec | table | mgbarsky
 public | movie_star | table | mgbarsky
 public | starsin    | table | mgbarsky
 public | studio     | table | mgbarsky
(5 rows)
```

Describe columns of table movie

\d+ movie;

```
 Table "public.movie"
  Column     |         Type          | Modifiers | Storage
-------------+-----------------------+-----------+----------+-------------
 title       | character varying(30) | not null  | extended |
 year        | integer               | not null  | plain    |
 length      | integer               |           | plain    |
 incolor     | integer               |           | plain    |
 studioname  | character varying(20) |           | extended |
 producerc   | character varying(3)  |           | extended |
Indexes:
    "movie_pkey" PRIMARY KEY, btree (title, year)
```

Explain to each other the following terms:

- data model
- relational data model
- tuple
- component in a tuple
- data type of a component
- attribute
- relation
- schema
- relation instance

Identify any unclarities about the terms and discuss.