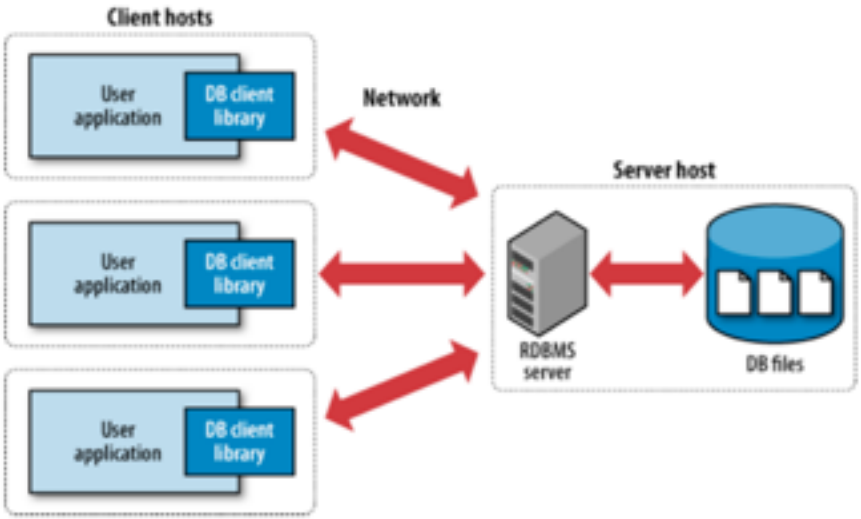
Relational Databases

Christopher Simpkins
chris.simpkins@gatech.edu

Relational Databases

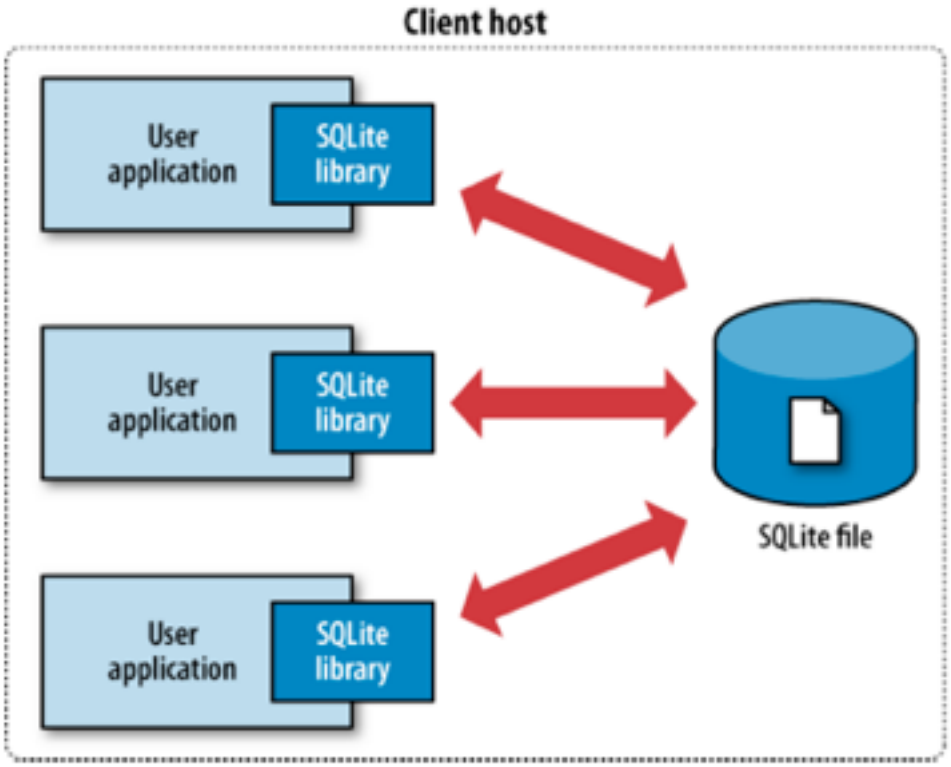
- A relational database is a collection of data stored in one or more tables
- A relational database management system (RDBMS) is software that stores and updates a relational database and provides a query and manipulation interface to the data
- In this lecture we'll cover
  - the two major RDBMS architectures
  - basic relational database design
  - basic SQL (structured query language)
  - SQLite, one of the RDBMS systems we'll use in this course (SQLite and MySQL)

Client-Server RDBMS



- Server software provides a middleware layer between the database files and client applications
- Clients typically connect through network protocols
- Lots of configuration overhead, but powerful security, scalability, availability features
- Examples: MySQL, PostgreSQL, Oracle, IBM DB2, MS SQL Server

Embedded RDBMS



- RDBMS engine is integrated with application program
- Data typically stored in a single file
- No configuration
- Much simpler, but less powerful
- SQLite is most popular example

Relational Database Concepts

- Entities are real-world objects to be modeled in the database
- Entities can be related to other entities by
  - one-to-one relationships
  - one-to-many relationships
  - many-to-many relationships
- A journal can have many articles, but an article can appear in only one journal
- An author can have many articles, and an article can have many authors

Tables

- Entities are stored in tables (a.k.a. relations)
- Each row of a table (a.k.a tuple) stores one record
- Each column of a table stores one attribute per record

publD	**title**
1	Recursive Functions of Symbolic Expressions and Their Computation by Machine
2	The Unix Time-sharing System

Keys

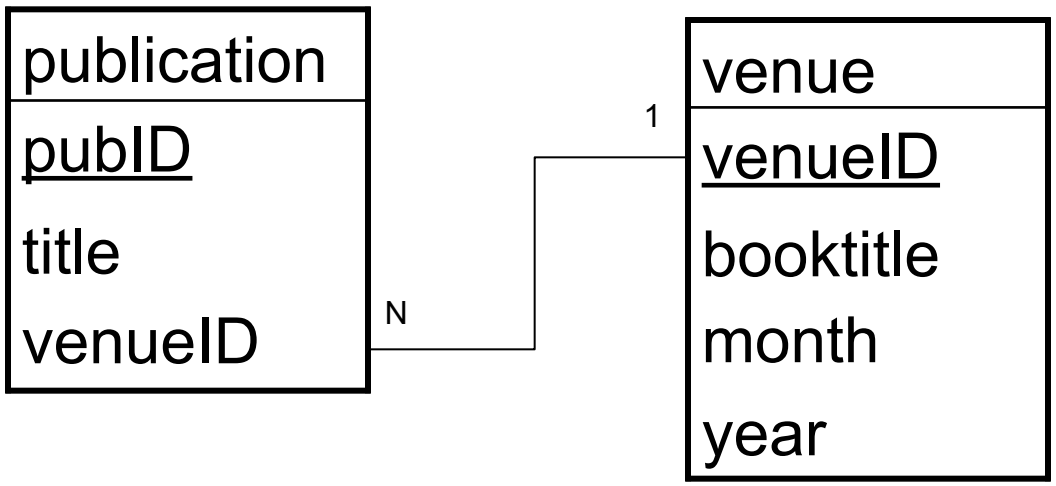
- The primary key of a relation is an attribute or (minimal) set of attributes (composite key) that uniquely identifies a record
  - There may be many candidate keys
- A foreign key links one table to another via the other table's primary key

pubID	title	venueID
1	Recursive Functions of Symbolic Expressions and Their Computation by Machine	1
2	The Unix Time-sharing System	2

venueID	booktitle	month	year
1	Communications of the ACM	April	1960
2	Communications of the ACM	July	1974

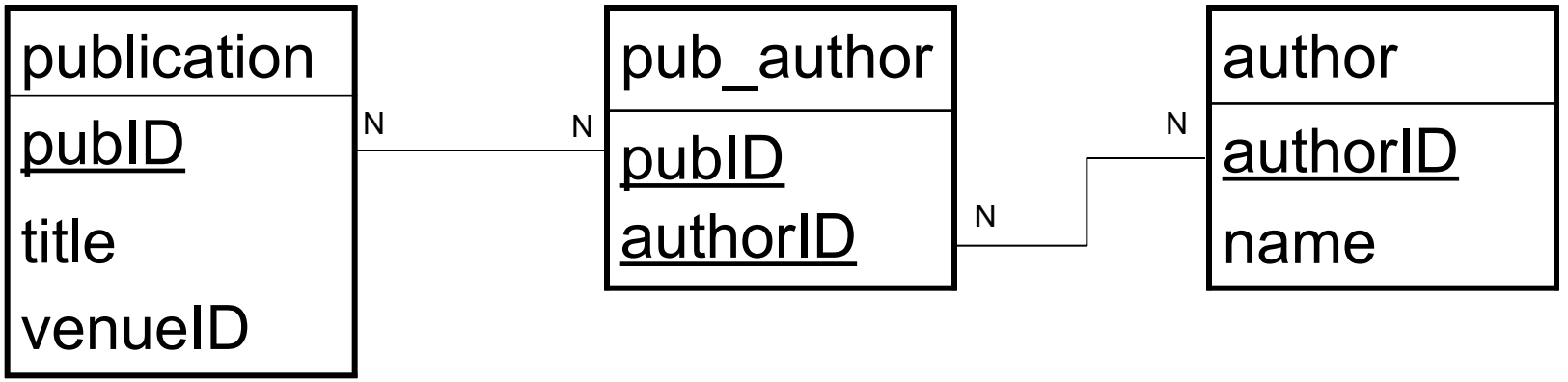
Entity-Relationship Diagrams

- ER diagrams can show attributes, keys, and relationships
- Primary key is underlined
- Here's our publication-venue schema depicted in an ER diagram:



Many-to-many Relationships

- Many-to-many relationships modeled with a link table
- Link table has only foreign keys
- For example, here's a link between publications and authors (each pub can have many authors, each author can have many pubs)



Structured Query Language

- ANSI standard language (SQL92) for interacting with a database
  - In practice every RDBMS has extensions
- Pronounced “ess-que-ell”, though many say “sequel”
- Declarative command language for creating, inserting data into, and getting data from a database
- Some RDBMSes define an additional imperative language for stored procedures
  - Stored procedures should be avoided for two reasons:
    - » Stored procedures are not even a little bit portable - they’re all RDBMS-specific
    - » Stored procedures encourage splitting application logic between database and main application code

SQL Data Languages

- SQL has four languages:
  - Data definition language (DDL) for creating, modifying and deleting database tables. Typically run when a database is first created, often contained in a script
  - Data manipulation language (DML) for inserting, updating, querying, and deleting data from tables
  - Transaction control language (TCL) for creating transactions (atomic sequences of commands)
  - Data control language (DCL) for controlling access to database resources and setting permissions
- SQLite does not support DCL. We'll deal with DDL and DML in this course

CREATE TABLE

General form

```
CREATE TABLE table_name (
  column_name column_type column_constraints...,
  [... ,]
  table_constraints,
  [...]
);
```

Example

```
CREATE TABLE parts (
  part_id INTEGER PRIMARY KEY,
  stock   INTEGER DEFAULT 0 NOT NULL,
  desc    TEXT    CHECK( desc != '') -- no empty strings
);
```

`DROP TABLE table_name` deletes a table.

Data Types

- Attributes have data types, a.k.a. domains
- SQLite is manifest-typed, meaning it can store any type of value in any column, but most RDBMSes are statically typed
- SQLite has typed storage classes and *type affinities* for columns (a suggestion to convert data to specified type)
- SQLite supports the following type affinities:
  - Text - NULL, text, or BLOB
  - Numeric - integers, floats, NULLs, and BLOBs
  - Integer - like numeric, but floats with no fractional part are converted to integers
  - Float - like numeric, but integers are converted to floats
  - None - no preference over storage class

INSERT

- Creates new rows in a table
  - `INSERT INTO table_name (column_name [, ...]) VALUES (new_value [, ...]);`
- Can leave off the column names to insert values positionally
- Example:

```
INSERT INTO parts (stock, desc) values (42, “ball bearings”)
```

By the way, we can leave off the primary key like this because the PK is an integer and therefore automatically an autoincrement field

UPDATE

- Modify an existing row in a table
  - `UPDATE table_name SET column_name=new_value [, ...] WHERE expression`
- Example:

```
UPDATE parts SET price = 4.25, stock = 75 WHERE part_id = 454;
```

DELETE

- Delete one or more rows from a single table
  - DELETE FROM table_name WHERE expression;
- Careful: if no WHERE clause, will delete all rows
- Example:

```
DELETE FROM parts WHERE part_id >= 43 AND part_id <= 246;
```

SELECT

- The big one. Used to extract data from a database
- The simple form:
  - `SELECT output_list FROM input_table WHERE row_filter;`
- The general form:

```
SELECT [DISTINCT] select_heading
  FROM source_tables
  WHERE filter_expression
  GROUP BY grouping_expressions
    HAVING filter_expression
  ORDER BY ordering_expressions
  LIMIT count
    OFFSET count
```

We'll go over some examples and keep it simple

Creating and Seeding a Database

- Can run a script with the `.read` command
- We'll use two scripts:
  - create-pubs.sql - will create the database schema
  - seed-pubs.sql - will insert data into the tables
- Tips:
  - .header on
  - .mode column

create-pubs.sql and seed-pubs.sql

```
create table if not exists author (
  author_id integer primary key,
  first_name text,
  last_name text
);

create table if not exists author_pub (
  author_id integer not null references author(author_id),
  pub_id integer not null references publication(pub_id),
  author_position integer not null, -- first author, second, etc?

  primary key (author_id, pub_id)
);

create table if not exists publication (
  pub_id integer primary key,
  title text,
  venue_id integer not null references venue(venue_id)
);

create table if not exists venue (
  venue_id integer primary key,
  booktitle text not null,
  month text,
  year integer not null
);
```

create-pubs.sql

```
insert into author values (1, "John", "McCarthy");
insert into author values (2, "Dennis", "Ritchie");
insert into author values (3, "Ken", "Thompson");

insert into publication values(1, "Recursive Functions of Symbolic ...",1);
insert into publication values(2, "The Unix Time-sharing System",2);

insert into author_pub values(1, 1, 1);
insert into author_pub values(2, 2, 1);
insert into author_pub values(3, 2, 2);

insert into venue values(1, "Communications of the ACM", "April", 1960);
insert into venue values(2, "Communications of the ACM", "July", 1974);
```

seed-pubs.sql