DS3000/DS5110: Foundations

Data Management with Tables: Pandas, Relational Databases, and Beyond

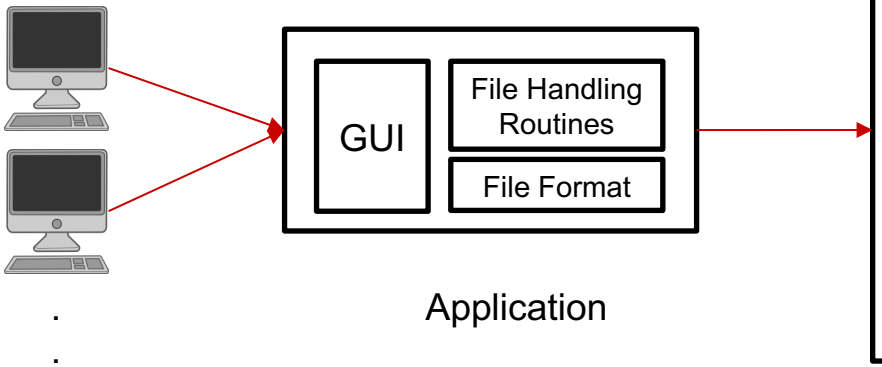
Dr. John Rachlin
Email: j.rachlin@northeastern.edu

Before Databases: File-Based Systems

Problem: You need to build a human-resources (HR) application to manage information about your employees.

User requirements:

- Search ("Show me information about Joe Smith")
- Add / Delete ("Enter data about Anne Jenkins, our new hire. Delete Saniya Lee, she retired")
- Update ("James Obatola was promoted, update his title.")



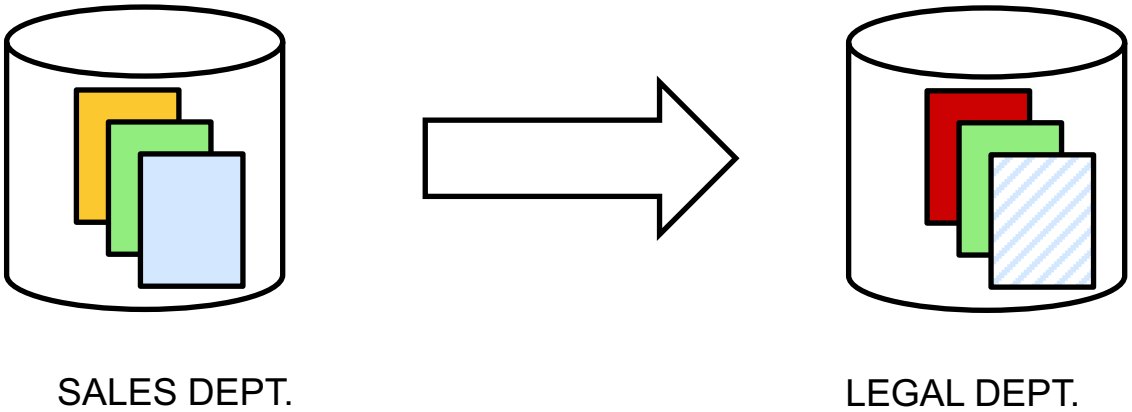

```
LAST,FIRST,DOB,TITLE
Smith,Joe,4/5/1992,Manager
Jenkins,Anne,10/08/2003,Programmer
Obtola,James,12/32/1982,HR
Mike,Crawford,2/29/2002,H.R.
Smith  ,Joe, 7/23/1966,CEO
.
.
```

Problems with File-Based Systems

Isolated data is more difficult to access and integrate.
More burden on application developer.

Problems with File-Based Systems

Decentralization leads to duplication.
Why is this bad?



Problems with File-Based Systems

Data dependence / Incompatible File Formats:
Applications are written with a particular file format in mind.

XML:

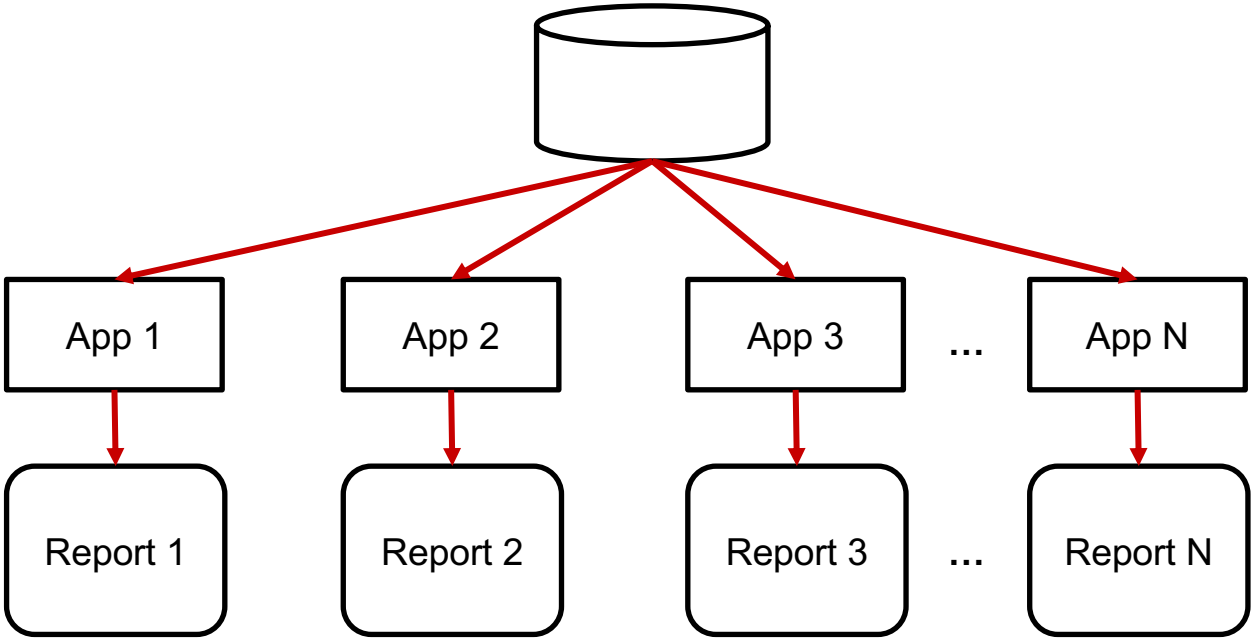
```
<?xml version="1.0" encoding="UTF-8"?>
<response uri="http://fake-url.com">
   <result>
      <Accounts>
         <row no="1">
            <FL val="id">1242160000000072037</FL>
            <FL val="phone"><![CDATA[null]]></FL>
            <FL
val="website"><![CDATA[www.joshuawyse.com]]></FL>
            <FL val="employees"><![CDATA[0]]></FL>
            <FL val="billingStreet"><![CDATA[null]]></FL>
            <FL val="shippingStreet"><![CDATA[null]]></FL>
            <FL val="billingCity"><![CDATA[null]]></FL>
            <FL val="shippingCity"><![CDATA[null]]></FL>
            <FL val="billingState"><![CDATA[null]]></FL>
            <FL val="shippingState"><![CDATA[null]]></FL>
            <FL val="billingCode"><![CDATA[null]]></FL>
            <FL val="shippingCode"><![CDATA[null]]></FL>
            <FL val="billingCountry"><![CDATA[null]]></FL>
            <FL val="shippingCountry"><![CDATA[null]]></FL>
            <FL val="description"><![CDATA[null]]></FL>
         </row>
      </Accounts>
   </result>
</response>
```

JSON:

```
{
    "id": "1242160000000072038",
    "description": "3",
    "website": "3",
    "numberOfEmployees": "3",
    "phone": "3",
    "name": "account3",
    "shippingAddress": {
        "country": "3",
        "stateOrProvidence": "3",
        "city": "3",
        "postalCode": "3",
        "street1": "3"
    },
    "billingAddress": {
        "country": "3",
        "stateOrProvidence": "3",
        "city": "3",
        "postalCode": "3",
        "street1": "3"
    }
}
```

Problems with File-Based Systems

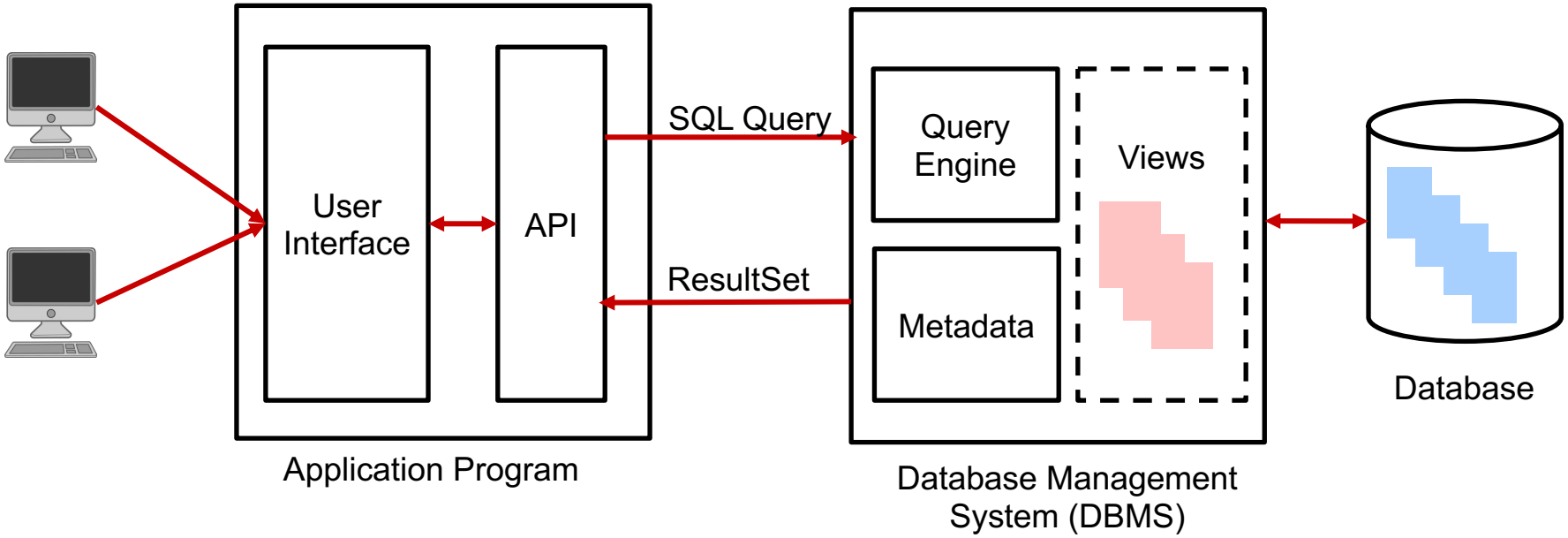
Fixed Queries (no *ad hoc* exploration) → App Proliferation



What is a Database?

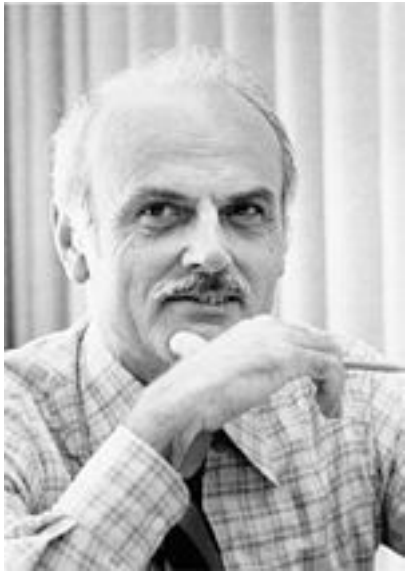
- Shared collection of **logically related** data (and a description of this data), designed to meet the information needs of an organization.
- System **catalog** (metadata) provides description of data to enable program–data independence.
- Logically related data comprises entities, attributes, and relationships of an organization's information.
- In a relational database, the data is represented as a **collection of tables**.

Database Application



(Usually running on a separate server)

Codd, 1970.



Edgar Frank Codd
1923-2003

Information Retrieval

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

History of Databases

1970 E.F. Codd (IBM) and the relational model

70's Early relational database implementations

1976 The Entity-Relationship Model: A tool for designing databases

1979 Oracle (#1) DB2 by IBM (#2)

1980 SQL Standards emerge (there are still many dialects)

1990's The growing importance of data-mining, machine learning, and the Internet

1995 MySQL

2000's Beyond the relational database, Big Data, NoSQL
2008 Sun Microsystems acquires MySQL
2010 Oracle acquires Sun Microsystems
2010's Non-Relational (NoSQL) databases proliferate, but relational DB's continue to dominate.

Advantages of DBMSs

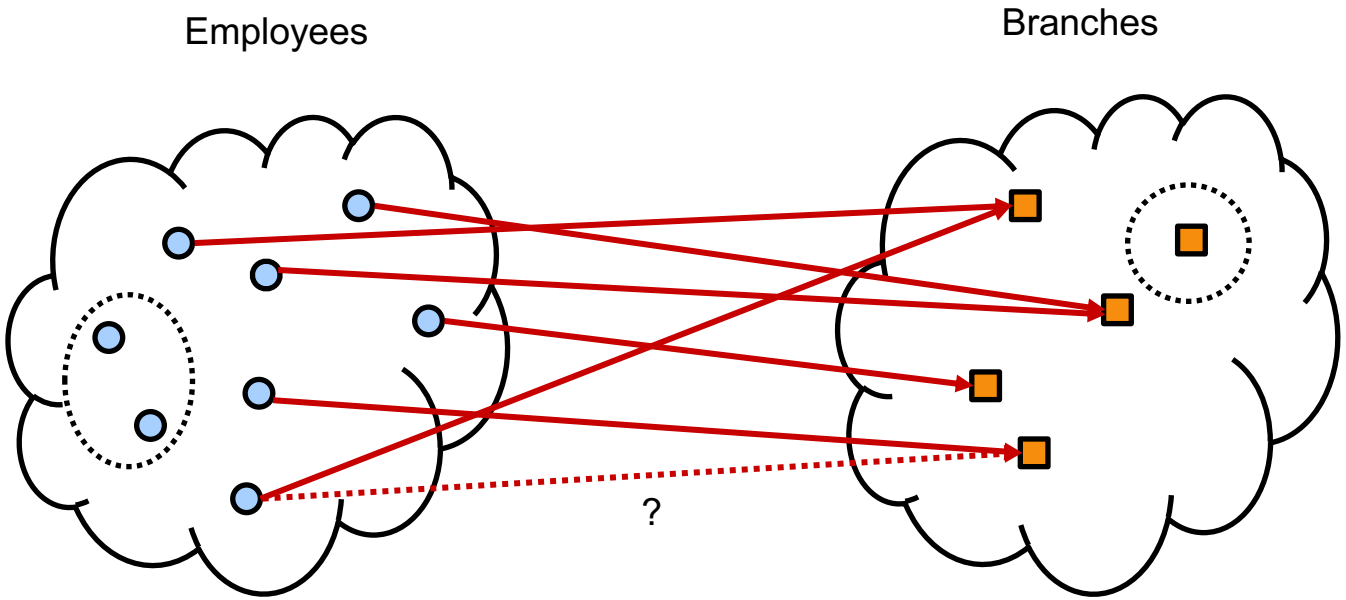
- **Control of data redundancy**
- **Data consistency**
- **Sharing** of data
- Improved **data integrity**
- Improved **security**
- Enforcement of standards
- Enabling **concurrent** users
- Backup and recovery
- *Ad hoc* queries (enable data exploration)

Disadvantages of DBMSs

- Complexity
- Size
- Cost of DBMS
- Additional hardware costs
- Cost of conversion / vendor lock-in
- Performance bottlenecks
- Higher impact of a failure

Many of these problems become magnified at “big data” scales.

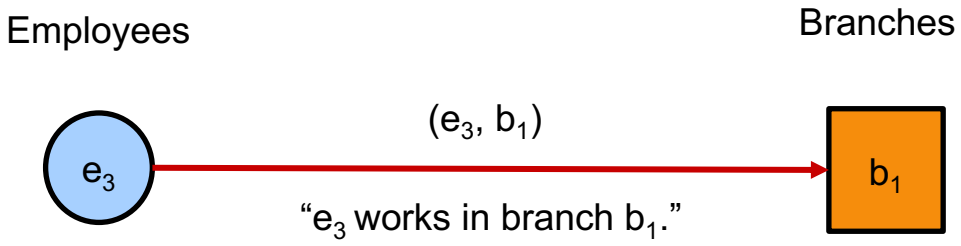
Relationships: Employees and Branches



E = $\{e_1, e_2, e_3, \ldots\}$

B= $\{b_1, b_2, b_3, \ldots\}$

An example: Employees and Branches



- In general, we can model the notion that an employee works in a particular branch with the tuple: (e,b) where e ∈ **E** and b ∈ **B**.

- Let **R** be the set of (e,b) tuples telling us which employees work at which branches. $R \subseteq E \times B$ where E X B is the *Cartesian Product* denoting all possible tuples (e_i, b_j)

- In set theory, **R** is called a **Relation**

Relational Databases (The dominant model)

Employee	Branch
e_2	b_5
e_3	b_5
e_7	b_2
e_{12}	b_9
e_1	b_9

. .
. .
. .

R = { (e_2,b_5), (e_3,b_5), (e_7,b_2), (e_{12},b_9), (e_1,b_9) … }

Key takeaways:

1. Relational databases store data in 2-dimensional tables
2. The tables establish connections between the different entities we want to model
3. The “Relational” in Relational Database (RDB) comes from the concept of a relation in set theory.

A Simple Relational Data Model

Name	Salary	Branch
John White	30000	22 Deer Rd, London
Ann Beech	12000	163 Main St, Glasgow
David Ford	18000	163 Main St, Glasgow
Mary Howe	9000	16 Argyll St, Aberdeen
Susan Brand	24000	163 Main St. Glasgow
Julie Lee	9000	22 Deer Rd, London

Discussion: What are the disadvantages of this design?

A Better Relational Data Model

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relational Data Model

Entities

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

A column or ***attribute*** or **field** providing more details about the particular entity.

- Name
- Datatype
- Other attributes

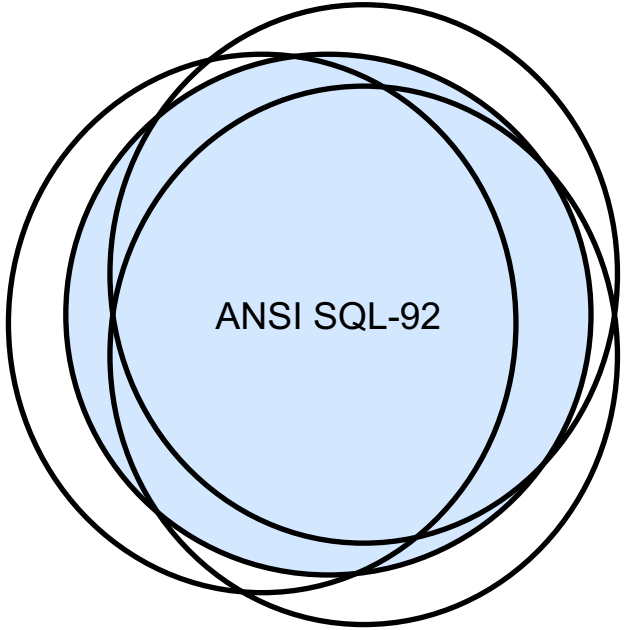
Staff

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

A row or ***record*** describing one instance of the entity

SQL: The language of databases

- SQL = Structured Query Language
- Pronounced *either* S.Q.L. or *See*-quel (both are perfectly acceptable.)
- The language was *standardized* over time (SQL-86, SQL-89, SQL-92) but most vendors do not implement the standard completely or they introduce vendor-specific features / dialects.



One standard, many dialects

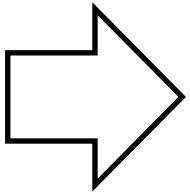
What is *structured* about SQL?

Simplified syntax for selecting data:

SELECT <Columns to retrieve>
FROM <Source tables>
WHERE <Conditions on individual rows>
ORDER BY <Sorting criteria>

An example of how SQL is a declarative language

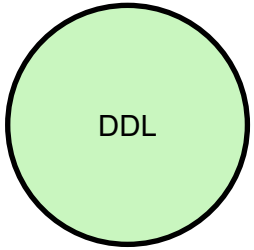
SELECT invoice_id, invoice_total
FROM invoices
WHERE invoice_total > 100.0
ORDER BY invoice_total



invoice_id	invoice_total
1	125.00
4	2199.99

The Language Hierarchy

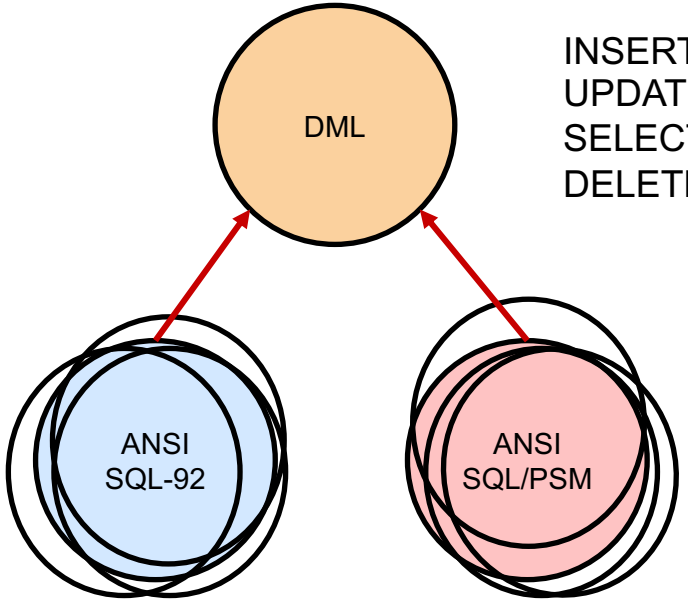
Data Definition Language



CREATE
DROP
ALTER

- Operations that create or modify the database Schema
- Metadata-focused
- Allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints.

Data Manipulation Language



INSERT
UPDATE
SELECT
DELETE

Declarative (4GL)

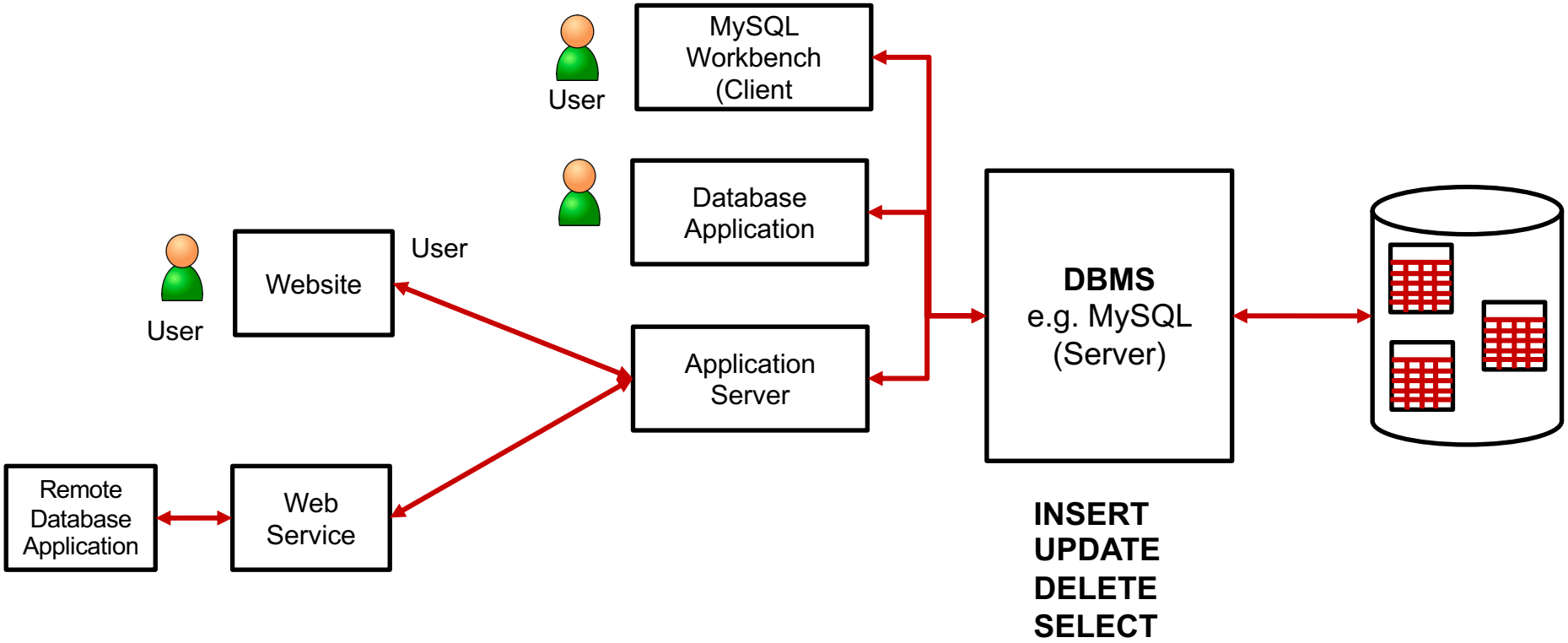
Procedural (3GL)

Procedural Languages for DB are numerous

Source	Common name	Full name
ANSI/ISO Standard	SQL/PSM	SQL/Persistent Stored Modules
Interbase / Firebird	PSQL	Procedural SQL
IBM DB2	SQL PL	SQL Procedural Language (implements SQL/PSM)
IBM Informix	SPL	Stored Procedural Language
IBM Netezza	NZPLSQL[18]	(based on Postgres PL/pgSQL)
Microsoft / Sybase	T-SQL	Transact-SQL
Mimer SQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MySQL	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
MonetDB	SQL/PSM	SQL/Persistent Stored Module (implements SQL/PSM)
NuoDB	SSP	Starkey Stored Procedures
Oracle	PL/SQL	Procedural Language/SQL (based on Ada)
PostgreSQL	PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language (implements SQL/PSM)
Sybase	Watcom-SQL	SQL Anywhere Watcom-SQL Dialect
Teradata	SPL	Stored Procedural Language
SAP	SAP HANA	SQL Script

Source: https://en.wikipedia.org/wiki/SQL

Data Storage, Retrieval, and Update



Access to metadata

- **Metadata** is data about data. It is pretty much everything but the data itself.
- Sometimes called the **data dictionary** or the **system catalog**
- Things we find in the system catalog include

  - Names of available databases
  - The names of each table in each database
  - The name and type of each column in each table and any column-specific constraints (Uniqueness, non-empty, etc.)
  - Referential constraints (The values in column X of table A must occur in column Y of table B)
  - Users and access privileges.
  - Usage statistics, open connections, etc.

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Metadata

Data

Transaction Support

- A **transaction** is a set of updates that must either succeed together or fail together. If only some of the updates were to succeed, the database could be left in an inconsistent state.

- If transaction fails, the database is **rolled back** to its original state.

Examples:

Insert a new staff member into the staff table.

Remove a staff member and re-assign the properties she managed to other staff members.

Recovery Services

- Database hardware can fail: Servers, disk drives, and software can crash.
- Rollbacks need to occur in the event of these types of failure.
- Backups are your friend.

```
$ mysqldump --user root -p myprojectdb > myproject_01DEC017.sql
```

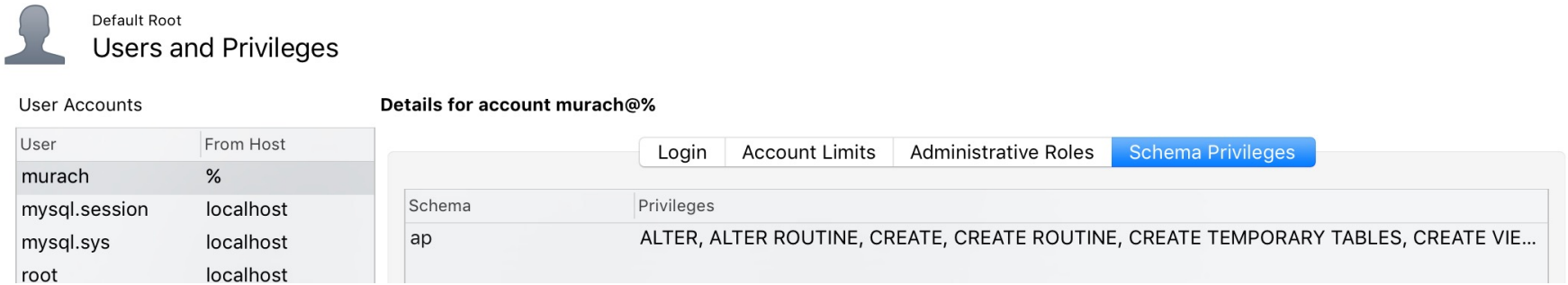
Concurrency Support

- Allow multiple users to perform changes to the database simultaneously
- The *lost update* problem: Multiple users sharing a single bank account doing deposits and withdrawals at different ATMs at the same time.

Time	Withdraw $50	Deposit $50	Balance
t_1		x = readBalance()	$100
t_2	x = readBalance()	x = x + 50	$100
t_3	x = x - 50	writeBalance(x)	$150
t_4	writeBalance(x)		$50
t_5			$50

Authorization / Access Control

- Only authorized users can access a database
- Specific kinds of privileges are **granted** or **revoked**

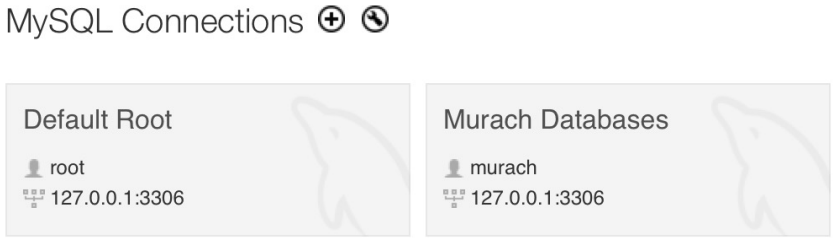


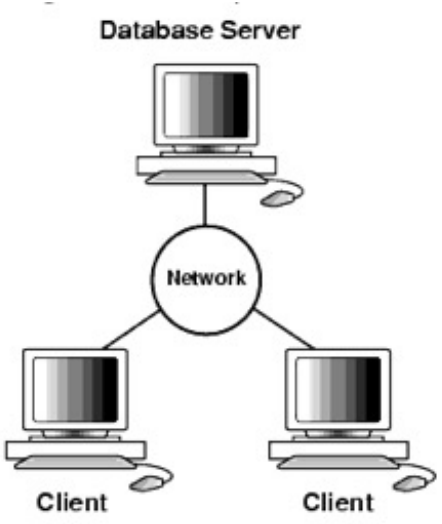

```
SELECT *
FROM mysql.user
```

Error Code: 1142. SELECT command denied to user 'murach'@'localhost' for table 'user'

Data Communication Support

- Users should be able to connect to a remote database over a network
- This is called *distributed processing* (upcoming lectures)





Database Integrity

- Stored data must be consistent and correct
- Operations on the data must adhere to specified constraints
- These constraints are often derived from an understanding of the **business rules** and are reflected in the database design.

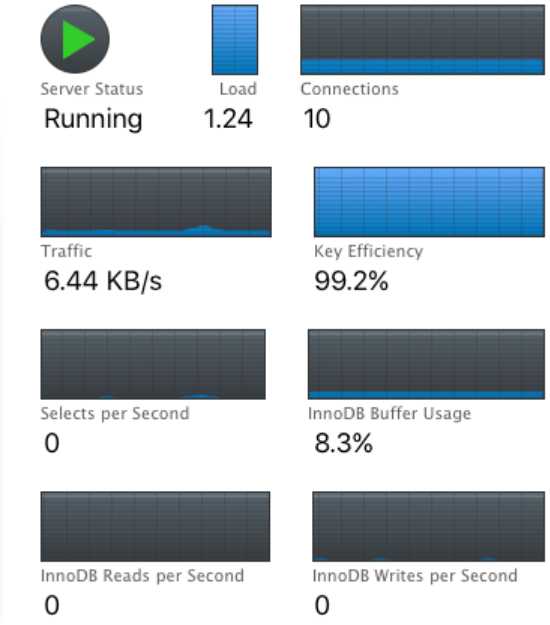
Some examples of constraints supported by MySQL:

NON NULL: A value must be provided for a particular field
ENUM/SET: The value must be one of the specified values
Datatype: The stored value must adhere to the datatype of the column
Uniqueness: Values in the column must be unique (no duplicates!)
Primary Key: Primary keys must be unique and non-null
Foreign Key: The entity referenced in another table must exist

Utility Services*

- May be third-party tools or provided by the vendor

Service	Supported by MySQL Workbench	Alternative MySQL Utility
Import / Export from flat files (.csv for example) and general backup	YES	mysqlimport mysqldump Mysqlpumpdf (5.7.8+)
Usage monitoring	YES	-
Statistical analysis	YES	-
Validation / Repair	YES	mysqlcheck
Garbage collection	YES	mysqlcheck



* Supplemented by Connolly and Begg

MySQL Workbench Dashboard

Tables for Data Management

Mechanism	Language	Typical Sizes (# records)	Advantages	Disadvantages
.CSV file	Text	10^3 to 10^6	Simplicity	• Duplication • No enforcement of consistency • Static
Pandas DataFrame	Python	10^3 to 10^6	Ease of use Flexibility Interactive Python eco-system	No cross-table consistency guarantees In-memory model Single threaded
Database Tables in a Relational Database	SQL	10^3 to 10^7	Standard and mature conceptual model	Vendor-specific features Limits of SQL
Apache Spark and SparkSQL	Scala Java Python SQL	10^3 to 10^9 “Big data”	A distributed table abstraction with multi-core processing	Mostly for batch processing rather than interactive analysis

SQLite (sqlite.org)

SQLite

Small. Fast. Reliable.
Choose any three.

Home | About | Documentation | Download | License | Support | Purchase | Search

What Is SQLite?

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day. More Information...

The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050. SQLite database files are commonly used as containers to transfer rich content between systems [1] [2] [3] and as a long-term archival format for data [4]. There are over 1 trillion (1e12) SQLite databases in active use [5].

SQLite source code is in the public-domain and is free to everyone to use for any purpose.

Latest Release

Version 3.42.0 (2023-05-16). Download Prior Releases

Common Links

- Features
- When to use SQLite
- Getting Started
- Try it live!
- Prior Releases
- SQL Syntax
  - Pragmas
  - SQL functions
  - Date & time functions
  - Aggregate functions
  - Window functions
  - Math functions
  - JSON functions
- C/C++ Interface Spec
  - Introduction
  - List of C-language APIs
- The TCL Interface Spec
- Quirks and Gotchas
- Frequently Asked Questions
- Commit History
- Bugs
- News

Pre-compiled executables

Precompiled Binaries for Linux

sqlite-tools-linux-x86-3420000.zip (2.18 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff program, and the sqlite3_analyzer program. (SHA3-256: a9dc5804cb61efc73a66aa61b034589216de9f7e53aa34a64dd2268d8e85bfdb)

Precompiled Binaries for Mac OS X (x86)

sqlite-tools-osx-x86-3420000.zip (1.56 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff program, and the sqlite3_analyzer program. (SHA3-256: 87a4dd36c3781a13181657cee85bf0e335554de3d89e7dbbe306c4a9b292f5d1)

Precompiled Binaries for Windows

sqlite-dll-win32-x86-3420000.zip (570.83 KiB)	32-bit DLL (x86) for SQLite version 3.42.0. (SHA3-256: 5edbd7244c91cae59dedfea6cdea6f9683116b034a0d18495e2aeb4a9592c87a)
sqlite-dll-win64-x64-3420000.zip (1.16 MiB)	64-bit DLL (x64) for SQLite version 3.42.0. (SHA3-256: 2425efa95556793a20761dfdab0d3b56a52e61716e8bb65e6a0a3590d41c97c0)
sqlite-tools-win32-x86-3420000.zip (1.93 MiB)	A bundle of command-line tools for managing SQLite database files, including the command-line shell program, the sqldiff.exe program, and the sqlite3_analyzer.exe program. (SHA3-256: 93d10287cd1a20dce57bb671bcd620cc827c3a316660326338cff0478514e6ee)

About SQLite

About SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen()

When to use SQLite

- Embedded devices / mobile applications / IoT
- Application file format for data storage and data transfer
- Low-volume websites
- Data analysis and exploration via SQL
- Education and training (no administrative overhead)

When not to use SQLite

High-volume websites
Large distributed datasets where a single file is insufficient (e.g., Big Data)
High concurrency applications. SQLite supports one write at a time.

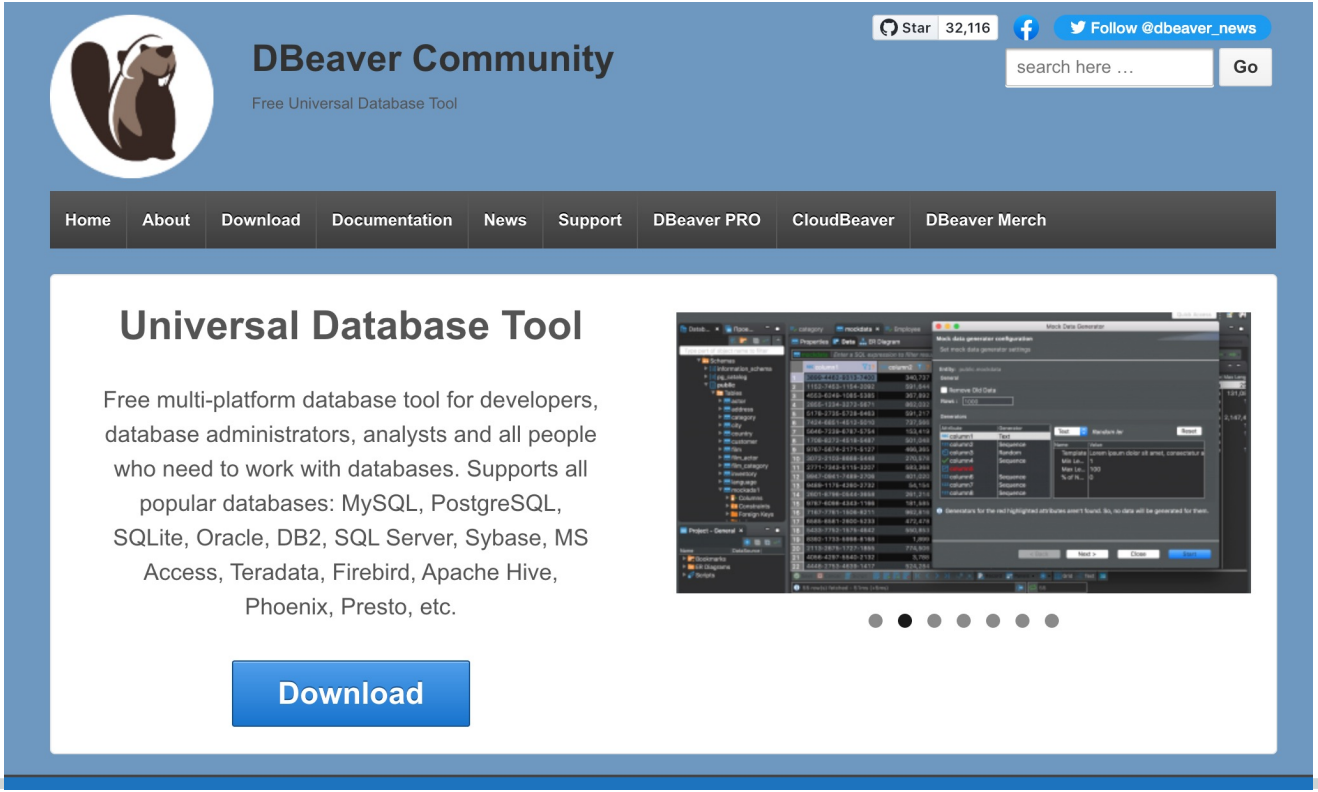
Create an SQLite database (demo.db)

```
Commands to run to build a database:

  sqlite3
  .help
  .mode csv
  .import iris.csv iris
  .import loan.csv loan
  .import vsx.csv vsx
  .save demo.db
  .quit
```

These commands will load three .csv files into a database so that we can explore the data using SQL queries.

dbeaver.io: A database client



Setting up your connection

Select your database

Create new database connection. Find your database driver in the list below.

Type part of database/driver name to filter

Sort by: Title Score

- All
- **Popular**
- SQL
- NoSQL
- Analytical
- Timeseries
- Embedded
- Hadoop / BigData
- Full-text search
- Graph databases

MySQL | SQLite | Db2 for LUW | DuckDB | MariaDB

Oracle | PostgreSQL | SQL Server | TiDB | Apache Calcite Avatica

Apache Drill | Apache Hive

Connecting to a .db File

Connect to a database

Generic JDBC Connection Settings

SQLite connection settings

Main | Driver properties

General

Connect by: Host URL

JDBC URL: jdbc:sqlite:/Users/rachlin/Documents/teach/ds/ds3000/lec/09_database/demo.db

Path: /Users/rachlin/Documents/teach/ds/ds3000/lec/09_database/demo.db

Open ... Create ...

You can use variables in connection parameters.

Connection details (name, type, ...)

Driver name: SQLite

Driver Settings Driver license

Test Connection ... < Back Next > Cancel Finish

Writing Queries…..

DBeaver 23.0.5 - <demo.db> Script-5

```
SELECT
    age,
    job,
    marital,
    education,
    balance
FROM loan
WHERE job = 'management'
```

SELECT age, job, marital education, balance FROM

	age	job	education	balance
1	58	management	married	2,143
2	51	management	married	10,635
3	51	management	married	6,530
4	54	management	married	4,080
5	49	management	married	3,237
6	45	management	divorced	24,598
7	40	management	married	8,486
8	40	management	single	3,877
9	39	management	married	2,877
10	57	management	married	2,142
11	31	management	married	2,019
12	23	management	single	2,605

200 row(s) fetched - 1ms (1ms fetch), on 2023-05-24 at 11:59:26