Introduction to RDM and SQL

Live Online Training

12 Hours over 4 days

Prerequisites

- ✓ Familiar with IT terminology
- ✓ Some programming background

For beginners and experienced developers

Goals

- ✓ What the RDM, SQL, and RDBMS ***really*** are
- ✓ Understand SQL query processing
- ✓ Switch to a ‘set mindset’
- ✓ Declarative vs. Imperative Programming
- ✓ Write basic SQL queries
- ✓ Solid foundation, less focus on detail and syntax
- ✓ Passion for RDM and SQL

Day 1

What are SQL, RDM, and RDBMS?

Development Environment

SQL Sublanguages

- Data Definition Language
- Data Control Language
- Data Manipulation Language

Day 2

Query Logical Processing

The FROM Clause

The WHERE Clause

Day 3

GROUP BY

HAVING

SELECT

DISTINCT Set Quantifier

Sorting and Limiting Result Sets

Day 4

Subqueries

Set Operators

Conclusion

Administrative

Hours

Breaks

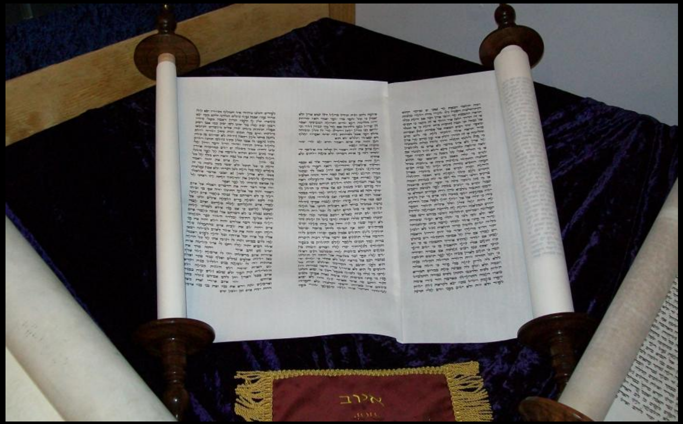
Exercises

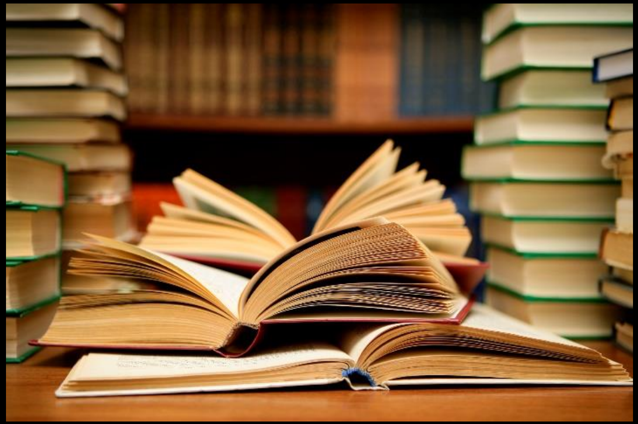
Course Evaluations

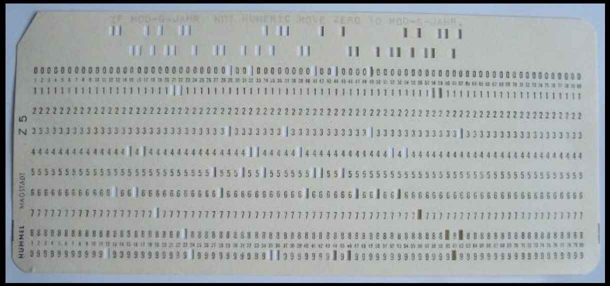
?

What is SQL?

History of Data Processing







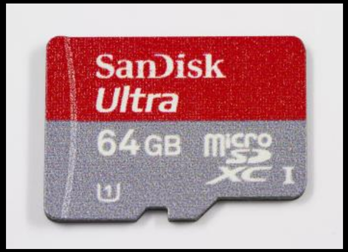


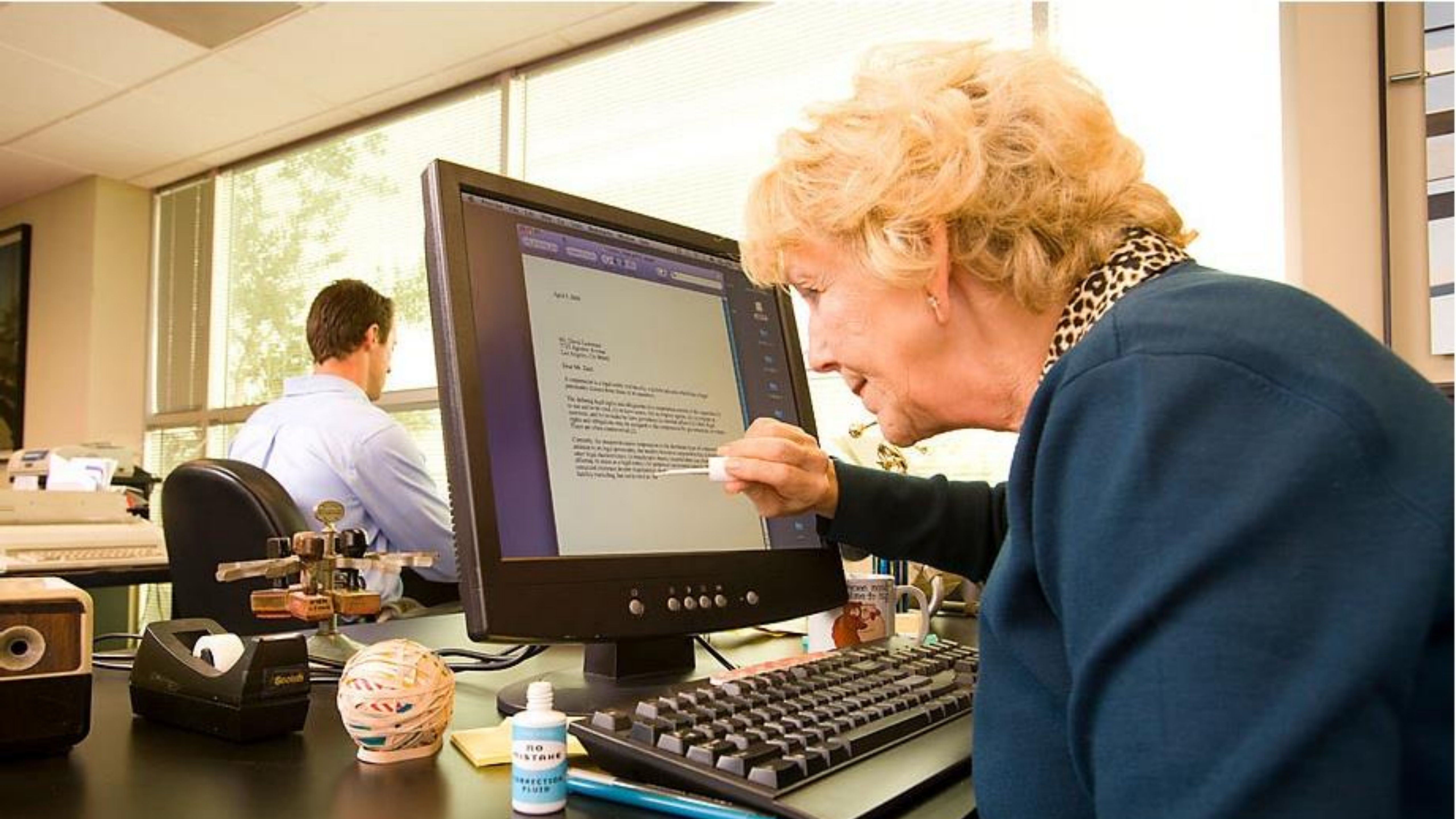
Photo attribution:
https://commons.wikimedia.org/w/index.php?curid=18997280
https://commons.wikimedia.org/w/index.php?curid=57651005
https://commons.wikimedia.org/w/index.php?curid=7799579

How Did We Access Data?

In the past, data was accessed by some physical order

1. Books had page numbers
2. File cabinet drawers had index labels
3. Punched cards had a physical order
4. Magnetic tapes had physical address pointers
5. Even this bullet list has numerical order…

Unfortunately, many still see it that way…


NO
MISTAKE
CORRECTION
FLUID


Glacier
Wa

1, Dave, USA, 1/1/2018, 2 Pens $3, Pencil $0.75

2, John, USA, 1/2/2018, 3 Markers $9

3, Gerald, Canada, 1/3/2018, Marker $3, Pen $1.5

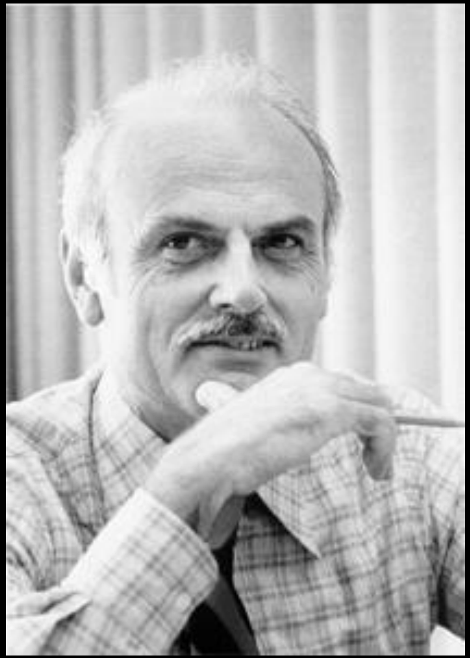
4, John, USA, 1/9/2018, 4 Pens $6, 2 Pencils $2.5, 2 Rulers $6

Sequential access

Data consistency

Modification anomalies

Dr. Edgar F. “Ted” Codd



- Born 1923 in England
- Studied Mathematics and chemistry at Oxford
- RAF Coastal Command pilot in WW2
- Moved to the USA in 1948
- 1953 - 1957, moved to Canada, angered by Sen. McCarthy
- In 1965 received doctorate in computer science
- 1967 Moved to IBM Research in San Jose, CA
- Received Turing award in 1981
- Died in 2003 from heart failure

Technology Survival

The RDM and SQL survived unscathed for
5 decades.

?

SQL, RDM, and RDBMS

A Relational Model of Data for Large Shared Data Banks

1969 internally at IBM

1970 published publicly

Codd's Alpha sub-language

System R used SEQUEL

Developed by Chamberlin and Boyce

SEQUEL was renamed to SQL

Competitors were early to adopt…

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.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

The Relational Model

Declarative method for specifying data and queries which is based on:

- Simple Set Theory
- First Order Predicate Logic
- Relational Algebra
- Tuple Relational Calculus

Relations and Tuples

- A tuple is a finite ordered list (sequence) of elements
- A relation is a set of tuples, where each set of corresponding elements (keys / attributes), are of a data domain

Order#	Customer	Country	Date	Items
1	Dave	USA	1/1/2018	2 Pens $3, 2 Pencils $0.75
2	John	USA	1/2/2018	3 Markers $9
3	Gerald	Canada	1/3/2018	Marker $3, Pen $1.5
4	John	USA	1/9/2018	4 Pens $6, 2 Pencils $2.5, 2 Rulers $6

Relations

A relation is a ‘thing’ in the real world. A concrete, real, identifiable ‘thing’.

It can be a material object, a ‘business concept’, or a relationship between concrete ‘things’.

Relation properties

- ✓ Unique tuples
- ✓ Uniquely referenceable attributes
- ✓ No tuple nor attribute order
- ✓ Atomic values
- ✓ “All at once” operations
- ✓ Independent of physical implementation (PDI)

SQL Approximates the RDM

SQL table (definition) → Predicate variable

A set of rows → Relation

Attributes → Columns

Each row → Tuple

Order#	Customer	Country	Date	Items
1	Dave	USA	1/1/2018	2 Pens $3, Pencil $0.75
2	John	USA	1/2/2018	3 Markers $9
3	Gerald	Canada	1/3/2018	Marker $3, Pen $1.5
4	John	USA	1/9/2018	4 Pens $6, 2 Pencils $2.5, 2 Rulers $6

A Relational SQL Model

Customers

Customer	Country
Dave	USA
John	USA
Gerald	Canada
Jose	Peru
Tim	NULL

Orders

Order#	Customer	Order Date
1	Dave	1/1/1965
2	John	1/2/1965
3	Gerald	1/3/1965
4	John	2/3/1965

Order Items

Order#	Item	Quantity	Price
1	Pen	2	$1.5
1	Pencil	1	$0.75
2	Marker	3	$3
3	Marker	1	$3
3	Pen	1	$1.5
4	Pen	4	$1.5
4	Pencil	2	$1.25
4	Ruler	2	$3

Items

Item
Pencil
Pen
Marker
Notebook
Ruler

Constraints

Domains / Data types

Check

Keys

Declarative Referential Integrity (DRI)

ISO / IEC 9075 SQL Standard

First standard published in 1986 (SQL:86)

Latest (9^{th} generation) published in 2016 (SQL:2016)

SQL conformance levels - Entry, Intermediate, and Full

Every vendor adds proprietary extensions

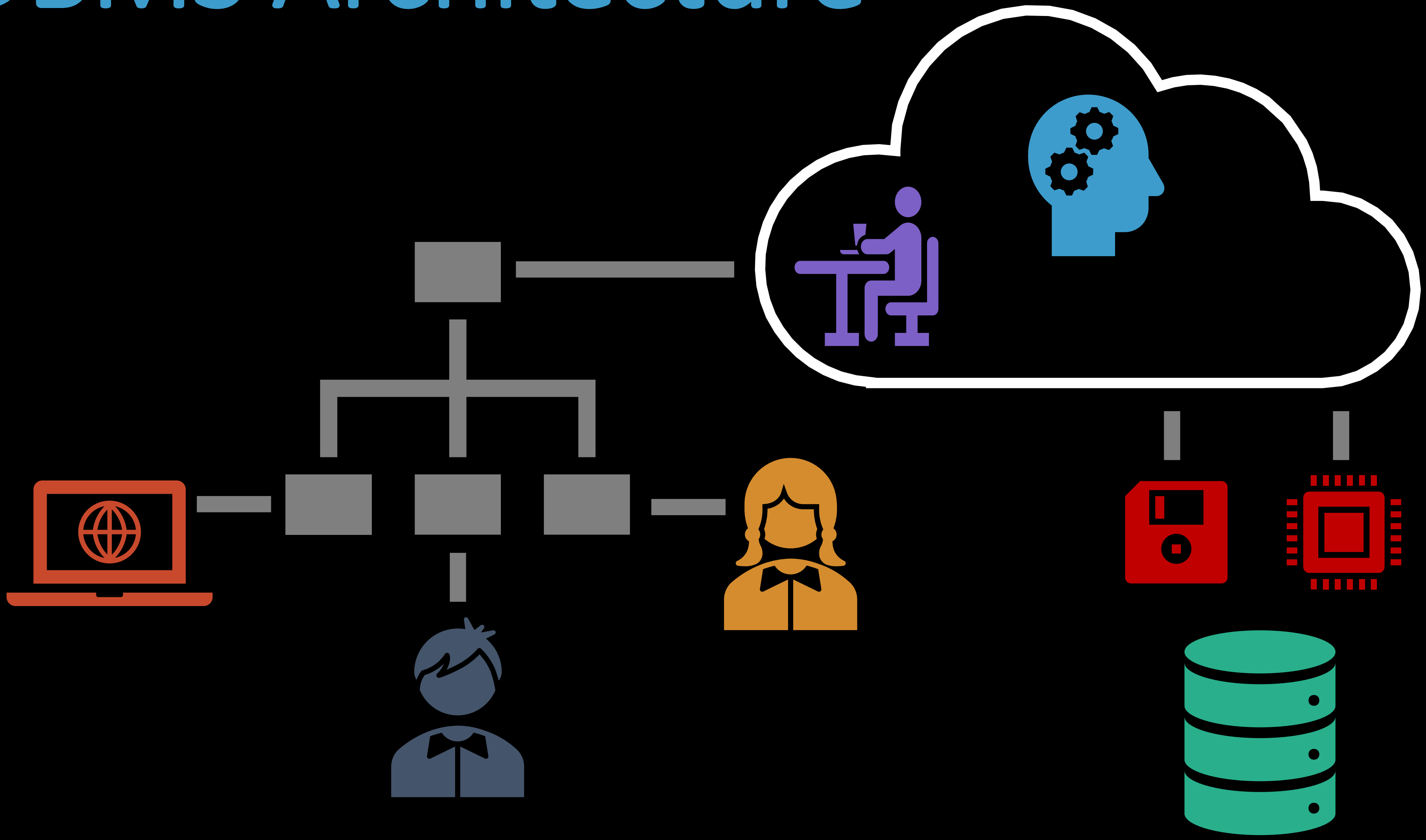
Relational Database Management Systems

- ✓ A relational data model is a model of reality
- ✓ Every relation represents one thing from the real world
- ✓ A tuple is an instance of that thing **and** is uniquely identifiable
- ✓ Every attribute describes an inherent property of that thing
- ✓ Identifying attributes distinguish one thing from another
- ✓ Every fact is represented in one place in the database, and always as an intersection of an attribute and a tuple
- ✓ A SQL database approximates the relational data model

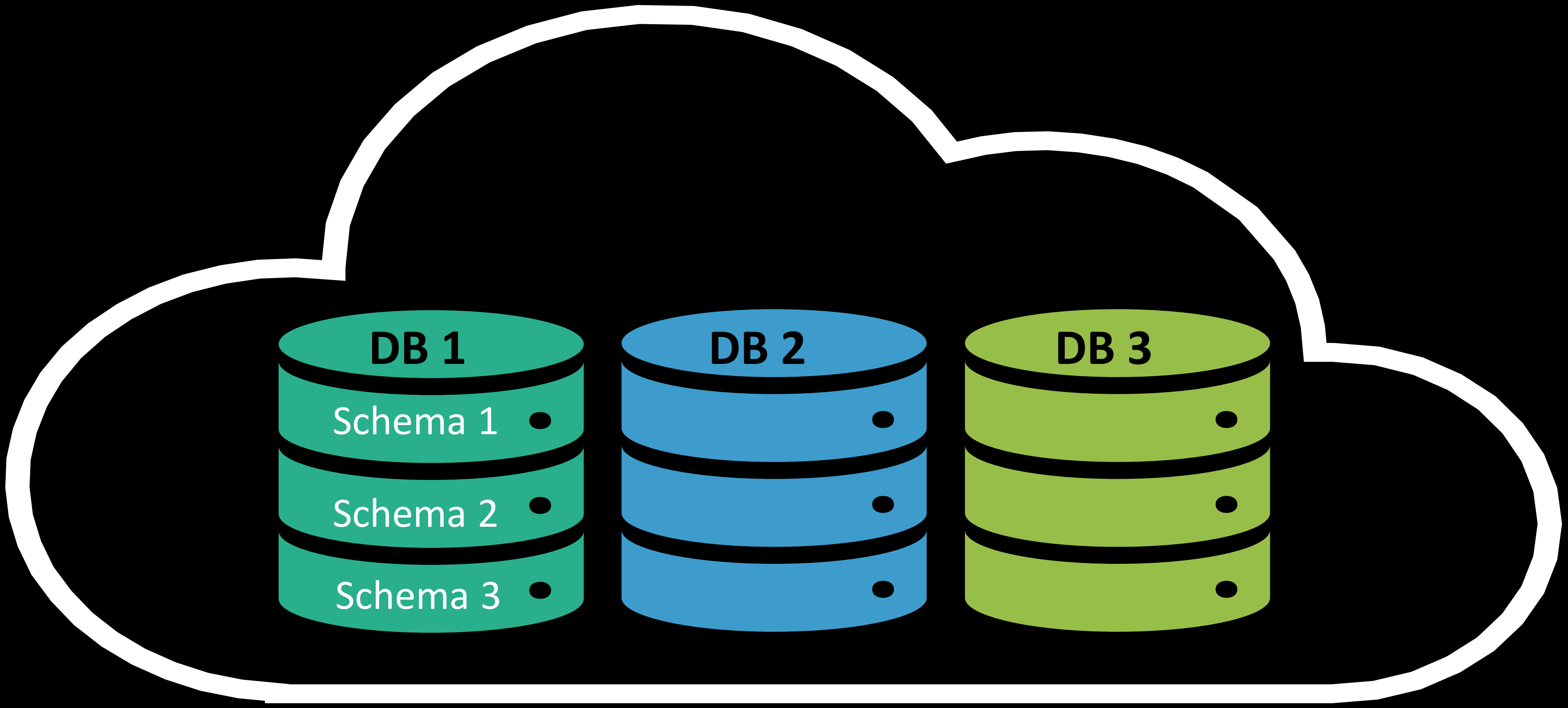
?

Development Environment

RDBMS Architecture



Databases and Schemas



Installing vs. Hosting

LOCAL INSTALLATION

HOSTING IN PUBLIC CLOUD

PaaS

ONLINE QUERY SERVICES

Free and Open Source

Platform	Tools	O/S
SQLite	DB Browser	
MySQL Community	MySQL Workbench	
PostgreSQL	PgAdmin	
SQL Server Express	Azure Data Studio Management Studio	
Oracle Express	Oracle Developer	
Online resources	Browser	

Install and Configure Development Environment

SQL Sublanguages

SQL Sublanguages

DDL - Data Definition Language

DCL - Data Control Language

DML - Data Manipulation Language

DQL – Data Query Language

Vendor specific extensions including imperative constructs

Data Definition Language

- CREATE
- ALTER
- DROP

```sql
CREATE TABLE Customers
(
Customer    VARCHAR(20)
            NOT NULL PRIMARY KEY,
Country     VARCHAR(20) NULL
);
```

```sql
ALTER TABLE Customers
ADD StateCode CHAR(2) NULL;

ALTER TABLE Users
ADD CONSTRAINT CheckEmailFormat
CHECK (Email LIKE '%@%’); *

ALTER TABLE Customers DROP StateCode; *
```

** Not supported in SQLite*

```sql
DROP TABLE Users;

DROP TABLE Users, Orders;
```

Data Control Language

- GRANT
- REVOKE
- *DENY (SQL Server)*

```sql
GRANT SELECT ON Users TO Ami;

GRANT DELETE ON Orders TO Guest;

GRANT ALL ON Customers TO Admins;
```

```sql
REVOKE SELECT ON Users FROM Ami;

REVOKE DELETE ON Orders FROM Guest;

REVOKE ALL ON CreditCards FROM Admins;
```

Data Manipulation Language

- INSERT INTO
- UPDATE
- DELETE FROM
- *MERGE / UPSERT*

Data Query Language

- SELECT
- *OUTPUT / RETURNING*

Vendor Extensions

Additional SQL functionality

Additional Languages

Data formats

Controlling session and default behavior

Flow control, variables, performance objects, server administration

?

Data Query & Manipulation Language

```sql
SELECT 'Hello World';

SELECT * FROM Customers;

SELECT Customer, Country FROM Customers;
```

```sql
INSERT INTO Customers (Customer, Country)
VALUES ('Dave', 'USA'), ('John', 'USA');

INSERT INTO CustomersBackup
SELECT * FROM Customers;
```

```
UPDATE Customers
SET    Country = ‘Costa Rica’
WHERE  Customer = ‘Dave’;
```

```sql
DELETE FROM Customers
WHERE Customer = ‘Dave’;
```

?

INSERT, UPDATE, DELETE

Review

- What are SQL, RDM, and RDBMS?
- Development Environment
- SQL Sublanguages
    - Data Definition Language
    - Data Control Language
    - Data Manipulation Language

?

Next week:
Query Logical Processing

```
            6
5  SELECT [DISTINCT] Customer,
                     COUNT(*) AS Count
1  FROM   Orders
2  WHERE  OrderDate > ‘19650101’
3  GROUP BY   Customer
4  HAVING COUNT(*) > 1
7  ORDER BY NumOrders DESC
8  OFFSET 0 FETCH NEXT 10 ROWS ONLY;
```