Foundations of Databases

SSgt Clark, Athan L

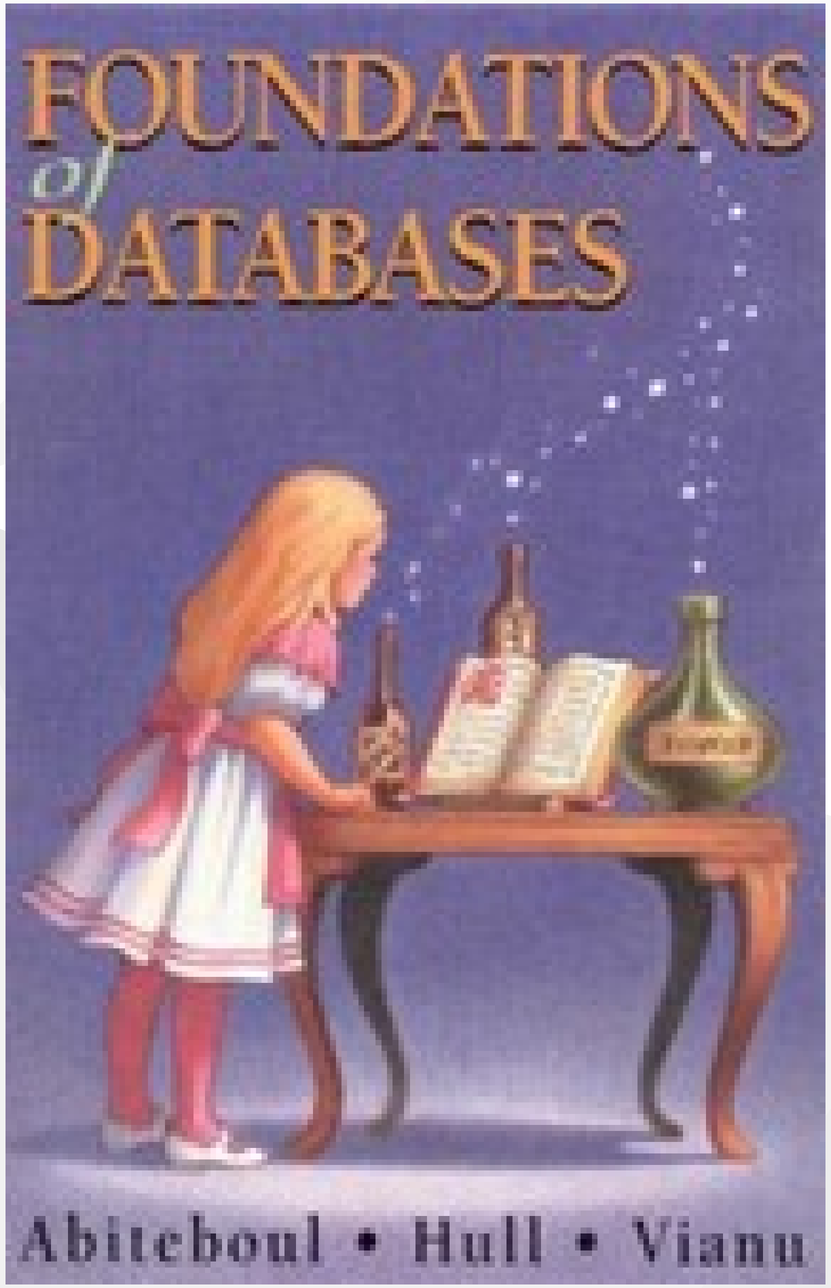
Presented on 20251204

SSgt Clark

Platoon Sergeant
3D Network Battalion, Detachment Hawaii

- Software Developer for ~15 Years
- Software Engineering & Computer Science Enthusiast
  - Working Toward Degree and PE Licensure
- CompTIA A+, Net+, Sec+, CySA+
- Google Data Analytics, Professional Scrum Master Level 1
- Lean Six Sigma Yellow Belt

Outline



Introduction

> “ In a nutshell, a database management system is a software system that enables the creation, maintenance, and use of large amounts of data. ”

Diverse Requirements Impose Competition:

Storage Paradigms, Language Models, Precision, Concurrency, Scalability

Definitions

- **Database** vs. Database Management System (DBMS)
- Query ~ *"Question"* or *"Do This"*
- Key / Index = Unique Identifier: EDIPI, Item Instance Number
- String, Integer / Floating Point Number, Boolean
    - `"Johnny B."`, `1775` / `5.56`, `FALSE`
- Tuple, Array, Dictionary
    - `("one", 2, 3.15)`, `["Dan Daly", "Smedly Butler"]`, `{ edipi: 12345678, name: "Schmukatelli" }`

Definitions

- Table, Column, Row - just like Microsoft Excel

EDIPI	Name
1553763807	`"Clark SSgt Athan L"`
5	`"Henderson BGen Archibald"`

Payment Method	Time	Amount
VISA1234	00:00:00	$5.00
VISA1234	12:11:34	$7.62

Definitions

- Table, Column, Row - alternative perspective

```
[
  (1553763807, "Clark SSgt Athan L"),
  (5, "Henderson BGen Archibald")
]
```

Are We Good?

Are We Good?

Cool.

Example Databases

Types of Databases

- Key / Value
- Relational
- Unstructured
- Graph
- Time-Domain

Example Databases - Key / Value

$$k \hookrightarrow v$$

$$k \hookrightarrow (v1, v2, v3)$$

“ A Value or Tuple of Values **Uniquely Identified** by Some Key ”

Features

Extremely Fast, Easy to Parallelize (Scalable)

Examples

Facebook Messanger, Session Cache

Implementations

Redis, Memcached, In-Memory (HashMap, BTree)

Example Databases - Relational

```
[Person]
*name
+birth_place_id

[`Birth Place`]
*`birth city`

Person *--1 `Birth Place`
```

“ Tables of Values & Keys **Referenced** By Other Tables ”

Features

Still Pretty Fast, Forces Data Consistency

Examples

Example Databases - Unstructured

```
[
  { serial: "1234", nomen: "TRC-209", color: "Green" },
  { serial: 2TKA1234, nomen: "Warfighting Laptop", weight: 1 },
]
```

> Collections of Data Blobs

Features

Scalable, Flexible for Growing Projects

Examples

Startups, When Final Requirements Aren't Certain

Implementations

MongoDB, Cassandra, DynamoDB

Example Databases - Graph

```
digraph D {
  rankdir=LR
  A -> B
  A -> C
  C -> B
  C -> D
  D -> A
  D -> B
  B -> B
}
```

```
graph D {
  rankdir=LR
  A -- B
  A -- C
  C -- B
  C -- D
  D -- A
  D -- B
}
```

“ Nodes and Edges ”

Features

Scalable, Flexible for Growing Projects, Queries that Follow Edges

Example Databases - Time-Domain

🕑 $\hookrightarrow (v1, v2, v3, v4)$

$$\mathbb{R} \hookrightarrow (v1, v2, v3, v4)$$

“ Optimized for Time-Based Queries ”

Features

Depends on Underlying Implementation

Examples

Stock Tickers, Log / Event Queues

Implementations

TimescaleDB, ElasticSearch

Example Databases - Honorable Mentions

- Full-Text Search
- Vector Databases
- Map/Reduce Databases

We O.K.?

We O.K.?

Cool.

Our Focus - Relational Databases

- Widely Used
- Flexible
- Definite
- Fast

Most Popular Query Language:
Structured **Q**uery **L**anguage (SQL)

Thinking about Data

- Creating Data
- Reading Data
- Updating Data
- Deleting Data

"CRUD"

Thinking about Data - In SQL

- Creating Data := `INSERT`
- Reading Data := `SELECT`
- Updating Data := `UPDATE`
- Deleting Data := `DELETE`

"CRUD"

Thinking about Data - In SQL

- Creating Data := `INSERT`
- Reading Data := `SELECT`
- Updating Data := `UPDATE`
- Deleting Data := `DELETE`

"CRUD"

> Only applies to an existing Table - SQL also permits creating tables, modifying tables, dropping them, etc.

LIVE EXAMPLE

Create a Simple Table, Add and Manipulate Some Data

SQL

Storage, Modification, Retreival via CRUD

Also Includes Organization, Data Relationships, Enforcement of Laws:

- Table Design
- Indexes / Foreign Keys
- Constraints
- Default Values

SQL - Table Design

Name	Rank	Favorite Color
Chesty Puller	MajGen	Green
Opha May Johnson	Sgt	
Carlos Hathcock	GySgt	Red

SQL - Table Design - Add Column

Name	Rank	Favorite Color	Gender
Chesty Puller	MajGen	Green	M
Opha May Johnson	Sgt		F
Carlos Hathcock	GySgt	Red	M

Constraints: Gender = ***Not Null***

SQL - Table Design - Modify Column

Name	Rank	Favorite Color	Gender
Chesty Puller	MajGen	Green	M
Opha May Johnson	Sgt	Blue	F
Carlos Hathcock	GySgt	Red	M

Favorite Color Default = ***Blue***

SQL - Table Design - Drop Column

Name	Rank	Gender
Chesty Puller	MajGen	M
Opha May Johnson	Sgt	F
Carlos Hathcock	GySgt	M

Live Example

Table Design

Check-In - Are We Okay?

Check-In - Are We Okay?

Cool.

SQL - Indexes

EDIPI	Name	Rank	Gender
1775	Chesty Puller	MajGen	M
1918	Opha May Johnson	Sgt	F
762	Carlos Hathcock	GySgt	M

Indexes *Must* be Unique

SQL - Foreign Keys

Marines:

EDIPI	Name	Rank	Gender
1775	Chesty Puller	MajGen	M
1918	Opha May Johnson	Sgt	F
762	Carlos Hathcock	GySgt	M

Awards:

EDIPI	Award
1775	Navy Cross
1775	Navy Cross
1775	Navy Cross
1775	Navy Cross
1775	Navy Cross

"Awards EDIPI" is a Foreign Key to "Marines EDIPI"

Live Example

Indexes and Foreign Keys

SQL - Default Values

Example Concepts:

- Default MCCU Size is "M/R"
- Default Submission Time for a Leave Request that was just created is "Now"
- Default EDIPI is "The Next One"

SQL - Constraints

- Uniqueness Constraints
  - No Marine should have a duplicate email address
- Boundary Constraints
  - All Marines' heights must be greater than 0
- General Purpose and Programmable
  - The IP Address for a device should be in its DHCP Zone

Live Example

Constraints & Defaults

How are we holding up?

How are we holding up?

Cool.

Advanced Concepts

1. Events and Triggers
2. Views and Joins
3. Sub-Queries
4. Stored Procedures
5. Transactions and Atomicity

No examples for these because I don't want to waste your time

Events and Triggers

> “ You can make a query run when something happens to a table ”

- Prevent a deletion of a row if some criteria is met
- Modify rows in another table if a row is created
- Destroy the entire database if my payroll hasn't been submitted in 1 month (logic bomb)

Views and Joins

> “ The ability to create partial view of a table or combine data from different tables ”

- Much higher performance than running multiple queries (important for large datasets like GCSS-MC)
- Useful for making code coherent

Sub-Queries

“ Use the results of another query without having to execute it first ”

"Select all of my Marines who have done their height and weight this semi-annual period"

"Select all of my Marines -> Select their hights and weights for this semi-annual period"

Stored Procedures

> SQL is not a *"Turing Complete"* programming language

SQL Can't create arbitrary programs that run forever on its own, but some people are crazy and want that ability in a database

PL/SQL (Oracle), T-SQL (Microsoft), PL/pgSQL (PostgreSQL)

Pros
- Do anything inside your database

Cons
- Hard to Debug, No Version-Control

Transactions and Atomicity

> “ Ability to run Multiple Queries at the same time that affect the same dataset without corruption ”

Multiple Queries running at the same time can cause a race condition

Atomic Transactions fix this, where conflicting datasets are locked if they're being modified

Did we make it?

Did we make it?

Cool.

Security Implications

Programmers are often dumb

- How passwords are stored
- Data Integrity - Permanently deleting or modifying data
- Session tokens are often stored in a session table - enables session hijacking

Conclusion

> “ Databases are designed to **retain**, **access**, and **manipulate** large amounts of data quickly and **preserve** them indefinitely. ”

SQL is a popular, decent solution to those problems. You'll likely see it again in your professional career.

Slides are available at github.com/athanclark/usmc-presentation-databases-20251204

Vote on Next Topic

1. Proxmox Virtualization System
   - great for home labs
2. Haskell Programming Language
   - it's like joining a cult
3. CPU Architecture
   - fun!
4. Abstract Algebra
   - don't be afraid

Questions / Comments