Database History

A tale of two papers

Its Me!

Doug Turnbull

@softwaredoug

Search & Big Data Architect
OpenSource Connections
http://o19s.com
Charlottesville VA, USA

Outline

- *A Relational Model of Data for Large Shared Data Banks* -- Edgar F. Codd
- *Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services* -- Eric A. Brewer

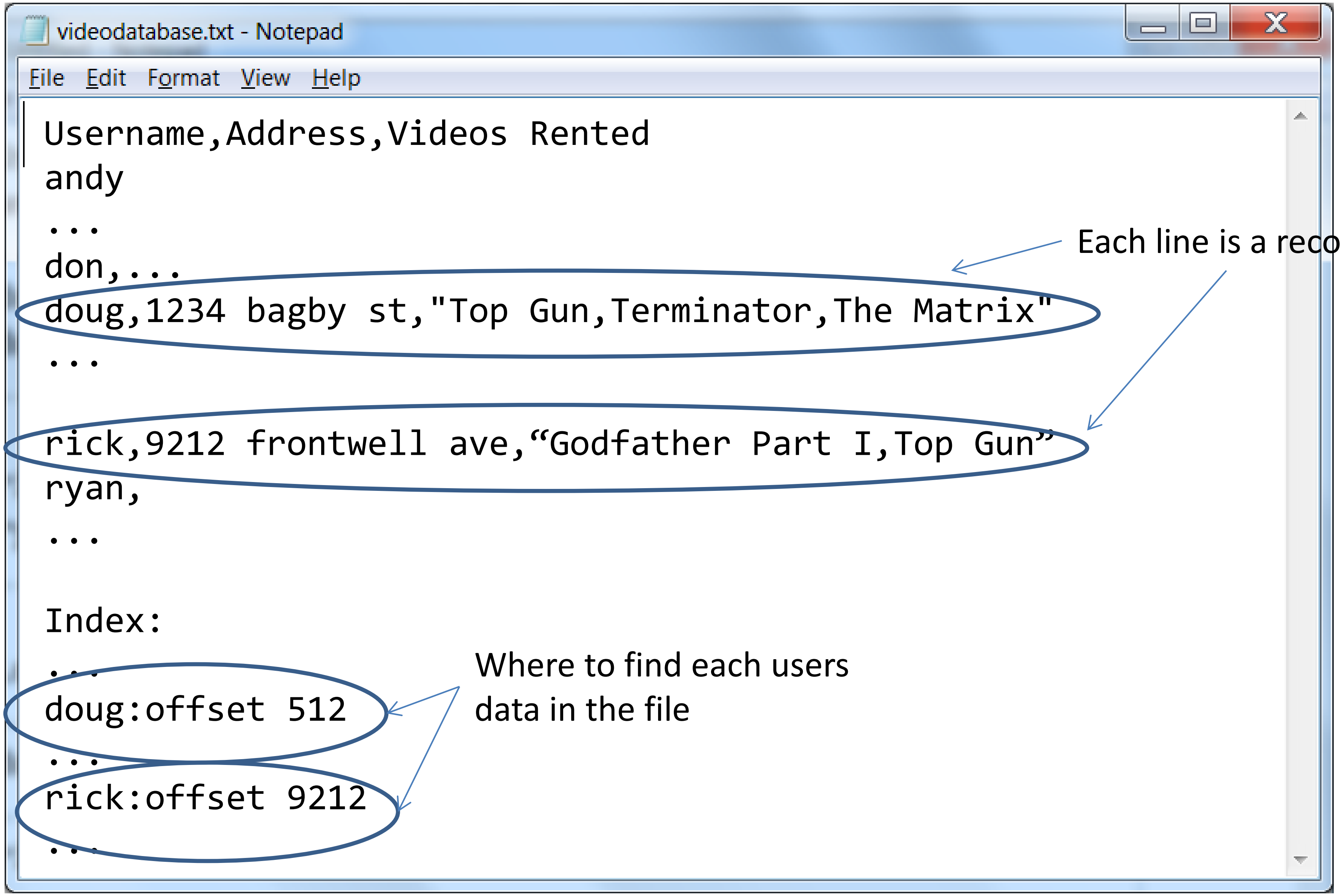
Why?

<u>RDMS</u>	<u>NoSQL</u>
Declarative	Procedural
Mathematical Precision	Computational Precision
Data Purity	Computational Transparency

See the wisdom in both paths

"A foolish consistency is the hobgoblin of little minds" – Emerson

Let's make a database!

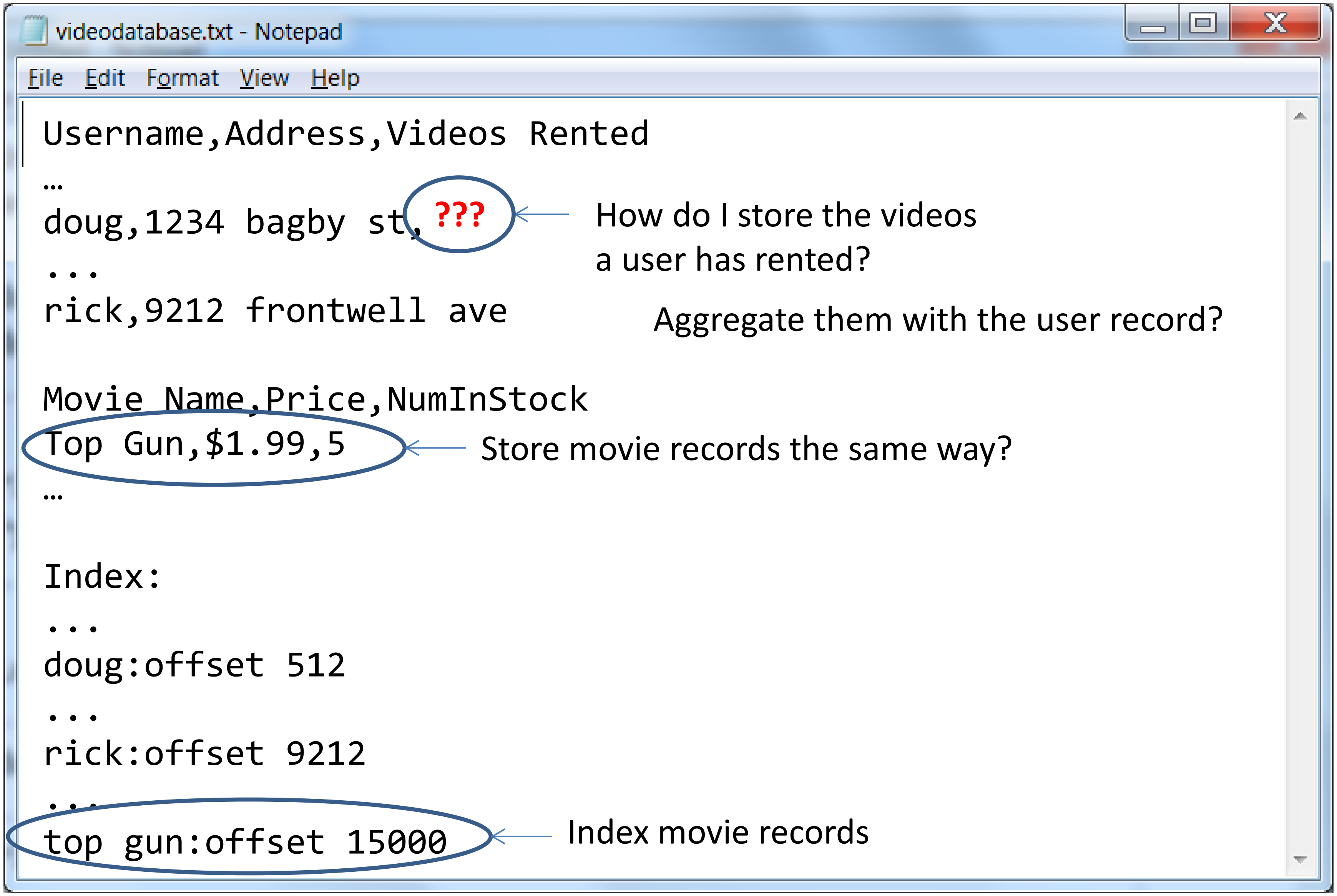


```
Username,Address,Videos Rented
andy
...
don,...
doug,1234 bagby st,"Top Gun,Terminator,The Matrix"
...

rick,9212 frontwell ave,"Godfather Part I,Top Gun"
ryan,
...

Index:
...
doug:offset 512
...
rick:offset 9212
...
```

Each line is a record

Where to find each users data in the file

Make each movie a record?


videodatabase.txt - Notepad

File Edit Format View Help

```
Username,Address,Videos Rented
…
doug,1234 bagby st, ???
...
rick,9212 frontwell ave

Movie Name,Price,NumInStock
Top Gun,$1.99,5
…

Index:
...
doug:offset 512
...
rick:offset 9212
...
top gun:offset 15000
```

How do I store the videos a user has rented?

Aggregate them with the user record?

Store movie records the same way?

Index movie records

Network Databases

- Early databases (Codasyl/DBTG)
  - Record based
    - Either hierarchical or navigational
  - Navigational: Records own other records by means of a “set” construct

- How might this look in our example?

Codasyl/DBTG

- Early databases, weak abstraction over a file

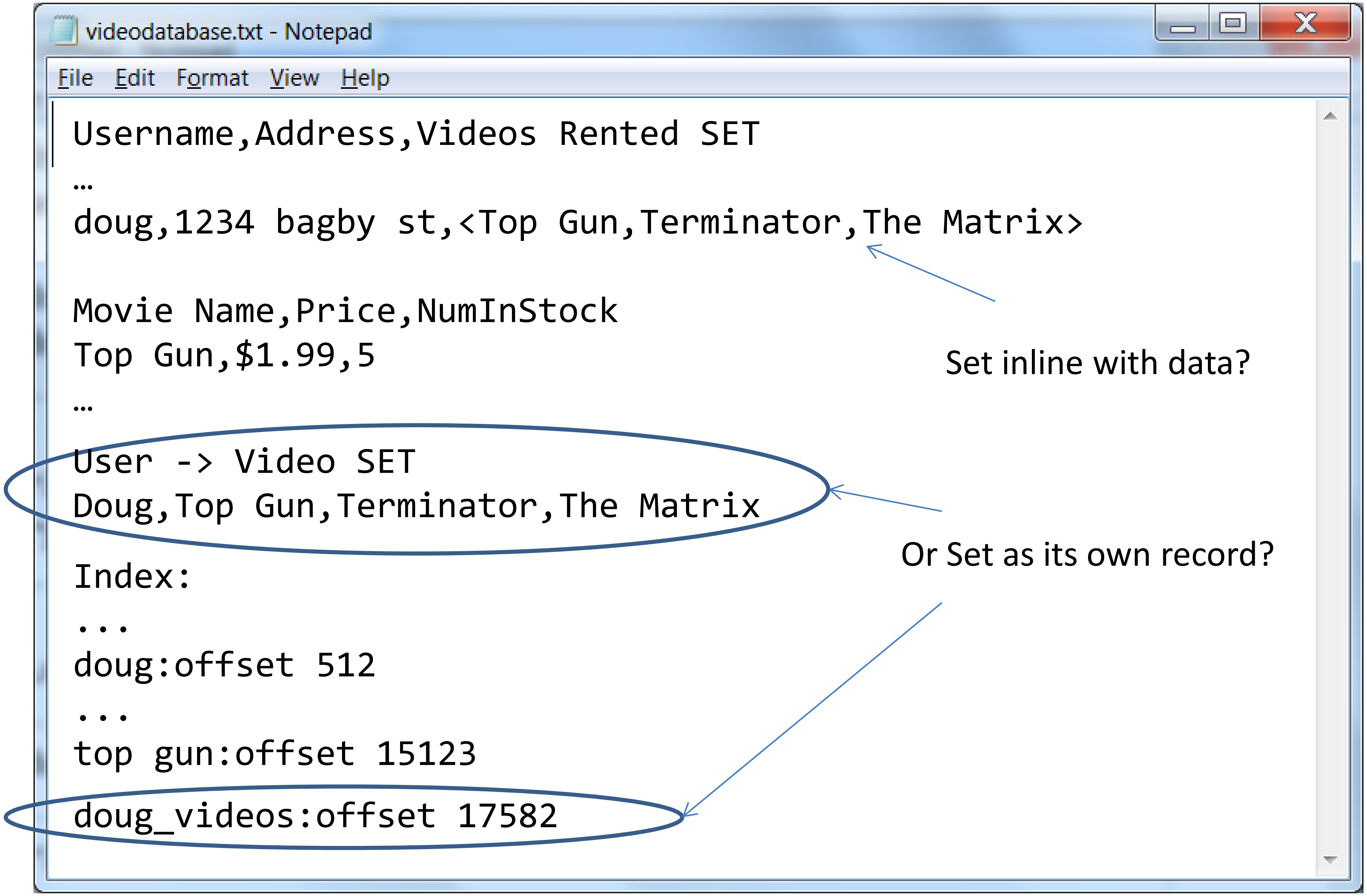
Basic Unit “Record”

```
Record Name is USER
  Location Mode is CALC Using username
  Duplicates are not allowed
  username Type character 25
  address Type character 50
  phonenumber Type character 10

Record Name is VIDEO
```

Records own other records via sets

```
Set Name is USER-VIDEOS
  ORDER is NEXT
  RETENTION is MANDATORY
  Owner is USER
  Member is VIDEO
```

Users -> Videos



```
Username,Address,Videos Rented SET
…
doug,1234 bagby st,<Top Gun,Terminator,The Matrix>

Movie Name,Price,NumInStock
Top Gun,$1.99,5
…

User -> Video SET
Doug,Top Gun,Terminator,The Matrix

Index:
...
doug:offset 512
...
top gun:offset 15123
doug_videos:offset 17582
```

Set inline with data?

Or Set as its own record?

Querying for Videos

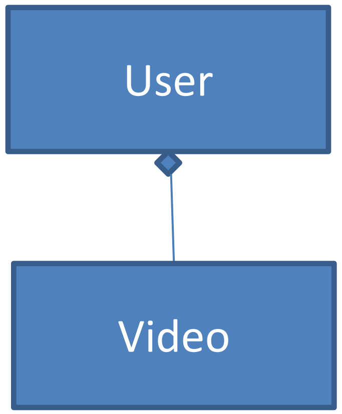
```
MOVE ‘Doug’ to USERNAME
FIND Any User USING USERNAME
FIND First VIDEO WITHIN User.Videos
DO WHILE (dbstatus=0)
        GET VIDEO
        PRINT (VIDEO)
        FIND NEXT VDEO WITHIN User.Videos
```

Summing Up

- Built from the bottom up
- Makes me think of:

```
public class User {
   private Video[] videos;

}
```

Is this ownership (aggregation)?
Or is this just an association with a video owned by another object?



Codd’s Criticisms

- Application is heavily dependent on storage constraints
  - Bottom Up
    - Access Path dependencies (which record do I access first? Users before videos? Who owns what?)
    - Order Dependencies (set order is defined at index time, iterations occur over that order)
    - Indexing Dependencies (indexes referenced by name)

- Changing these things breaks applications!

Codd

A tuple is a sufficient abstraction to represent a relation

(Doug, 1234 Bagby St, <Top Gun, 3.99, Terminator, 12.99>)

We can introduce "Normalization"

<u>Users</u>
(Doug, 1234 Bagby St)

<u>Rented Videos</u>
(Doug, Top Gun, 3.99)
(Doug, Terminator, 12.99)

We can reason about data with <u>mathematical certainty</u>

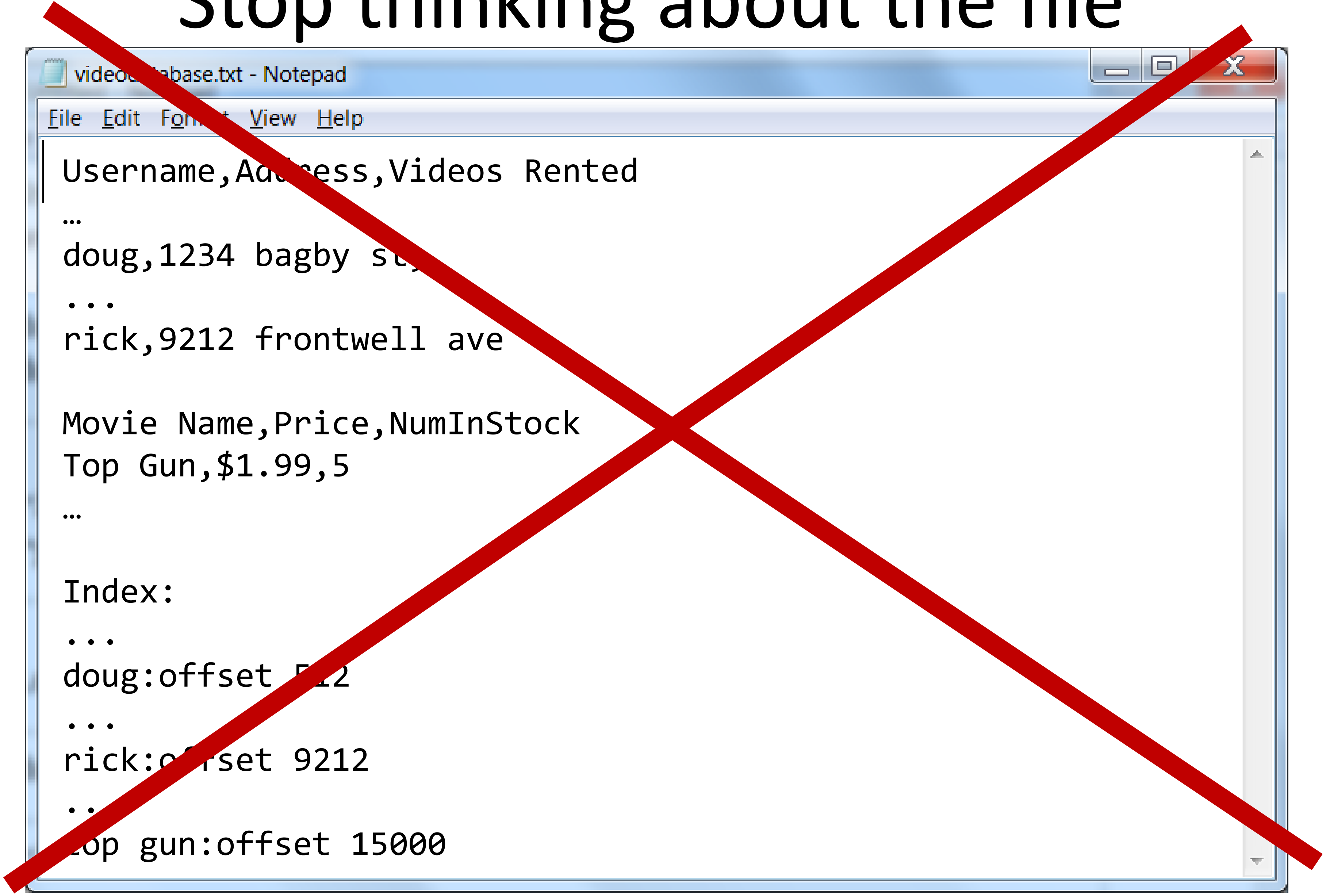
RDMS Features

- Codd defines a set of operations
- Most importantly the **JOIN**
  - Create any derived relation from a stored relation

Checking Codd’s Criticisms

- Access Dependencies – all data is normalized into a structure optimal for asking any question
- Order Dependencies – relations do not guarantee any order (though the query language can specify a sort)
- Indexing Dependencies – We don’t need to refer to the index when querying (its just a bonus)

Stop thinking about the file



```
Username,Address,Videos Rented
…
doug,1234 bagby st,
...
rick,9212 frontwell ave

Movie Name,Price,NumInStock
Top Gun,$1.99,5
…

Index:
...
doug:offset 512
...
rick:offset 9212
...
top gun:offset 15000
```

Start thinking about Normalized Relations!

Users
(Doug, 1234 Bagby St, <Top Gun, 3.99, Terminator, 12.99>)

Rented Videos
(Doug, Top Gun)
(Doug, Terminator)

Videos
(Top Gun, 3.99)
(Terminator, 12.99)

Retrospective?

- How do NoSQL databases do with these issues? Access Dependencies, Indexing Dependencies, Order Dependencies?
  - Is it even a fair criticism?
  - Why is it ok in NoSQL but not in SQL (is it ok?)?
  - ???

Fast Forward to early 2000s

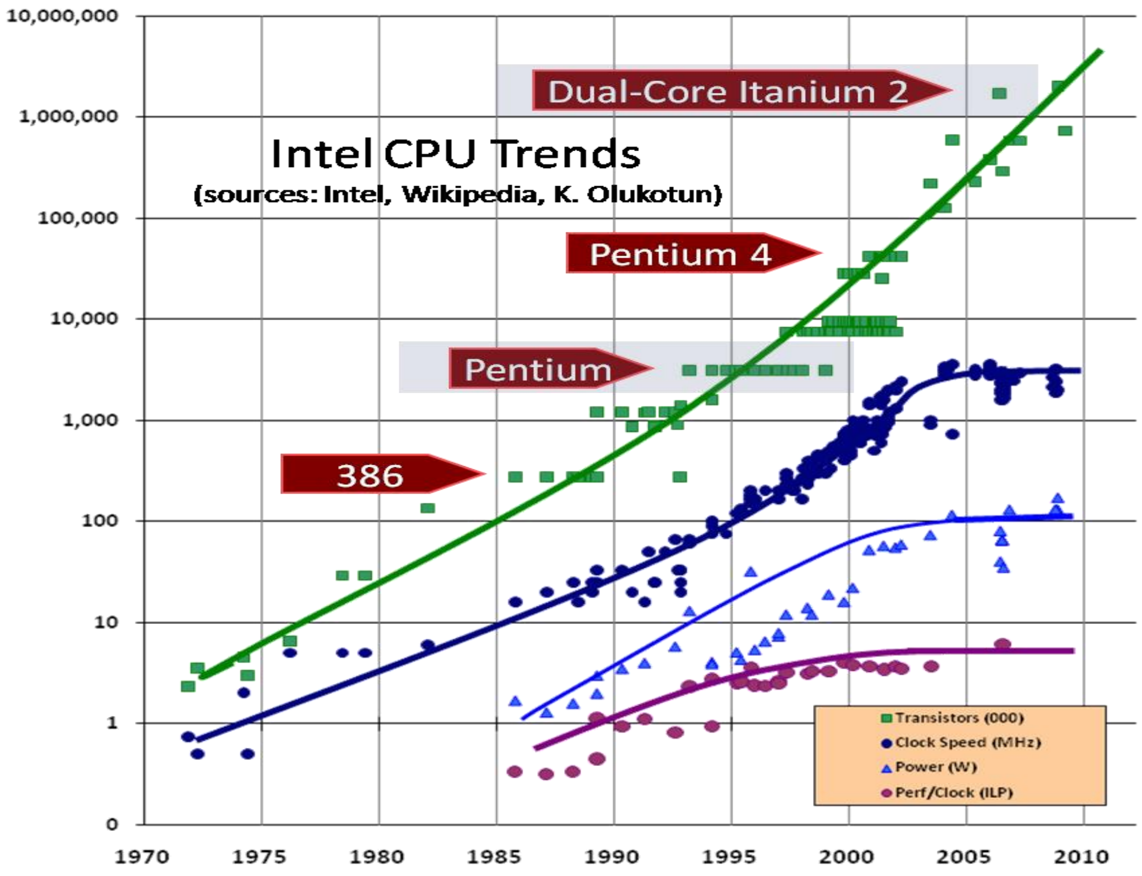
- SQL Databases have “won”; Codd’s vision thriving
- We can always scale with beefing up our hardware – “Vertical Scalability”
- Single system PoV

Trouble Ahead

"The Free Lunch is Over!" – Herb Sutter

The Free Lunch Is Over

A Fundamental Turn Toward Concurrency in Software



- Per HD size plateuing
- Hard Drive throughput plateauing

Trouble Ahead

- Instead of scaling vertically, we need to find ways to scale <u>horizontally</u>
  - “Elastic” scalability, add more systems to get more performance
  - Scaling horizontally (more less performant servers) than vertical horizontally

How do we design databases to take advantage of the scale, and grow

The Problem

- How do we design databases to take advantage of horizontal scalability?

- Are the traditional RDMS databases up to this task?

Enter Brewer’s CAP Theorem

- The CAP Theorem, introduced in *Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services*

CAP Theorem Explained

- In the presence of a partition system must chose between being *consistent* or *available*

<u>Consistent</u>

- Will not respond to request until consistency can be guaranteed.

<u>Available</u>

- Will respond to request, even if consistency cannot be guaranteed

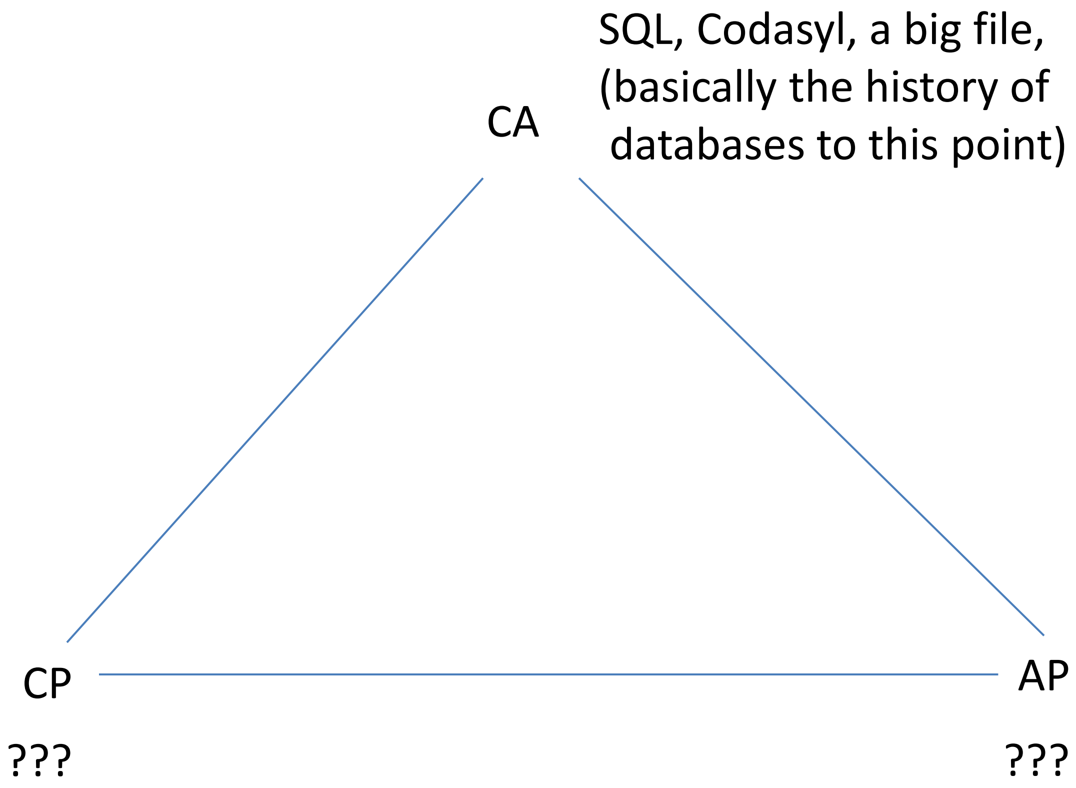
CAP Theorem

- In other words, in the case of horizontal scalability, (i.e., potential partitions) what do we do when servers can’t communicate?
  - Block? (wait till we can confirm consistency)
  - Respond? (we can figure this all out later)

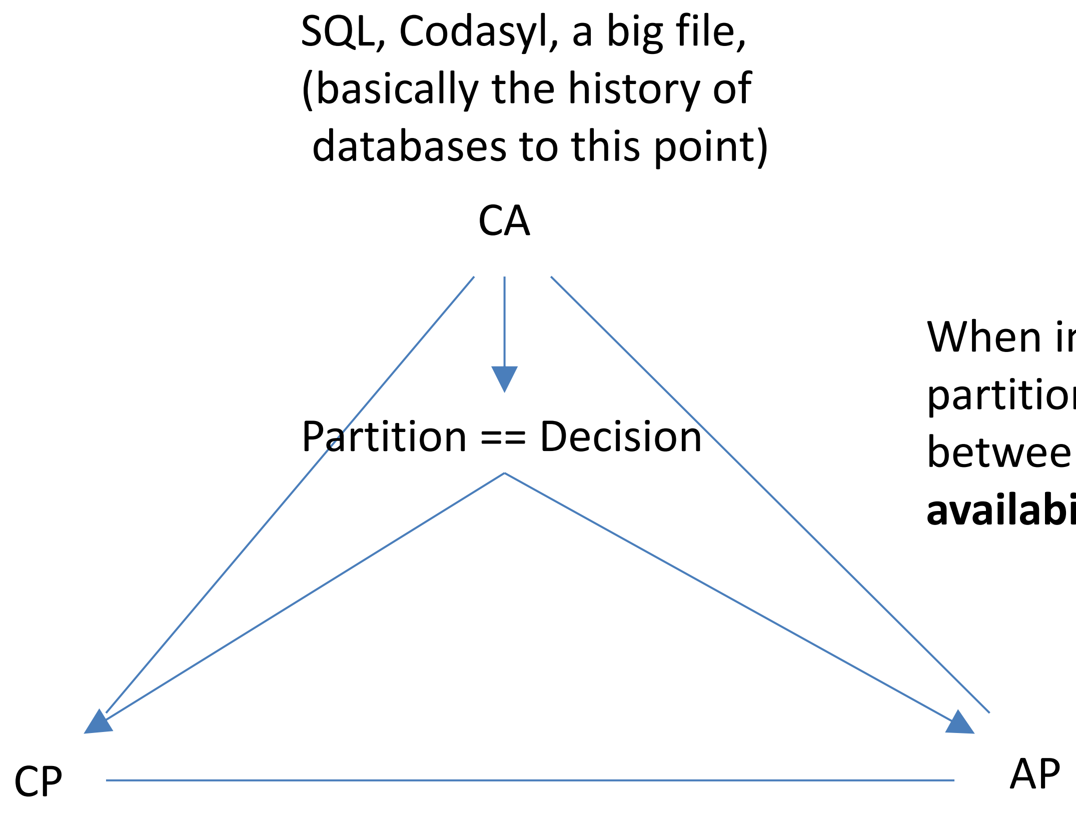
CAP Theorem in Human Organizations

- You receive an order from a customer over the phone do you:
  - Wait until the boss has signed off and reconciled with the rest of the orders?
    - Maybe blocking all your colleagues as your boss takes time to respond?
  - Or do you just respond saying “yes!” knowing maybe this customer is impatient (or maybe maintaining consistent inventory isn’t important)

What does this mean for databases?



What does this mean for databases?



When implementing a partitionable database, choose between **consistency** and **availability**

Call the boss before completing the order?

Respond quickly to guarantee the sale?

What else does this mean?

- Database designers must chose to focus on either consistent applications or available applications
- Thus… much of NoSQL is born
  - Big focus: options for more AP systems
    - Available and Partitioned
- Bottom line:
  - Choices choices choices, what corner of the triangle are you on?

What else does this mean?

- Many NoSQL databases end-up being designed bottom-up for horizontal scalability
  - Simpler, lower level APIS (set, get, put)
  - Hierarchical Schemas
  - Sometimes distributed based on order?

Controversial Question of the Day

- Have we come full circle?

- Or are we just responding to the technical challenges of the CAP theorem?

- Answers? (questions ok too☺)