CISC 7610 Lecture 6
Midterm review

Topics:

Quick quick review

Example questions

Lec 1: Main question, Examples

Lec 2: Relational Databases

Lec 3: Multimedia data

Lec 4: Approaches to MMDBs

Lec 5: Distributed MMDBs

CISC 7610 Lecture 1
Introduction to multimedia databases

- Main question
  - How can we process and store multimedia data so that we can find what we are looking for in the future?
- Examples of multimedia databases
  - Production and distribution for audio and video
  - Distribution and discussion for audio, video, and images
  - Surveillance and intelligence for speech, images, video
- Issues related to multimedia datatypes
  - Storage of audio and video
  - Real-time streaming / processing / synchronization

CISC 7610 Lecture 2
Review of relational databases

- Relational database management systems
  - uses relational data structures / representation
  - has a declarative data manipulation language
- Structured query language (SQL)
- Example data modeling problem: music collection
- Entity-relationship diagrams
  - draw ER diagram for example problem
  - convert ER diagram to schema
  - create tables, insert data
  - query data

CISC 7610 Lecture 3
Multimedia data and data formats

- Redundant vs irrelevant information
- Perceptual limits of multimedia data: audio and video
- JPEG encoding of images
  - 8x8 blocks, YCbCr, DCT, Quantization, Encoding
- MPEG encoding of audio
  - quantization in audio, multiband quantization
  - MPEG audio compression hides quantization noise behind louder sounds
- MPEG and H.264 encoding of video
  - I, P, B frames
  - data streams: elementary stream, packetized ES, transport stream

CISC 7610 Lecture 4
Approaches to multimedia databases

- Metadata
- Loose vs tight coupling of data and metadata
- Storing multimedia data in an RDBMS
- Object (oriented) databases, e.g., VelocityDB
- Graph databases, e.g., Facebook TAO, Neo4j
  - converting relational to graph model
- Object-relational mapping

CISC 7610 Lecture 5
Distributed multimedia databases

- Scaling up vs out
- Replication: copy and coordinate, purposes, types
- Partitioning: horizontal vs vertical
  - horizontal advantages: scale forever
  - horizontal disadvantages: cross-shard joins difficult
- Scalability
- CAP Theorem: can't replicate to another node, go ahead or wait?
  - sacrificing availability
  - sacrificing consistency
- NoSQL: throw out ACID and SQL language
  - Key-value, key-document, column-family, graph
- NewSQL: keep SQL, some version of ACID
  - Google F1 / Spanner: mask high commit latency
  - VoltDB: restrict types of transactions allowed

Practice question: DB Comparison

	RDBMS	Obj	ORM	Graph	KV	Doc	Col
[Specifically]	MySQL	Velocity	Hibernate	Neo4j	Cassandra	CouchDB	BigTable
Schema flexibility							
Schema heterogeneity							
Hierarchical data							
Inheritance							
Query flexibility							
Media specific ops							
Tight coupling							
Impedance mismatch							
Administration ease							
Distributed ops							
Speed							
Consistency							
Availability							
ACID transactions							

Practice question: Sound effects DB

- You are building an audio production multimedia database for creating and using sound effects
- Sound effects can be composed of other sound effects
- We will compare the pros and cons of different systems and modeling decisions

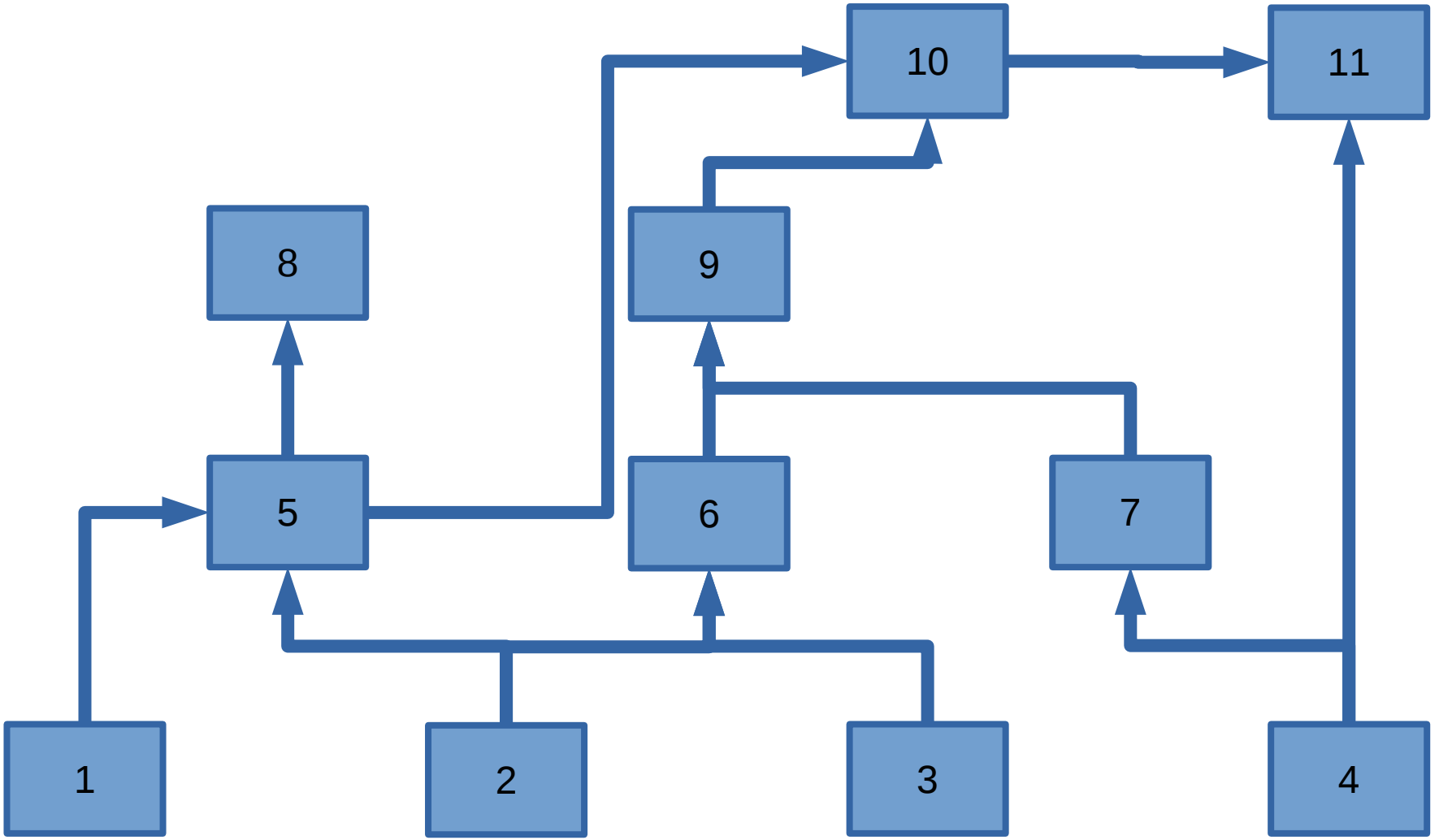
Sound effects DB: Create the ER diagram

- Sound effects can be either simple (wav file) or composite (other wav files)
  - all sound effects have a duration and a description
  - composite sound effects have constituent sound effects
    - with start times and gains for each of them
  - simple sound effects have a wav file
- Movies have many sound effects
  - each usage of a sound effect in a movie has a start time (offset) and a gain

Sound effects DB: Usage

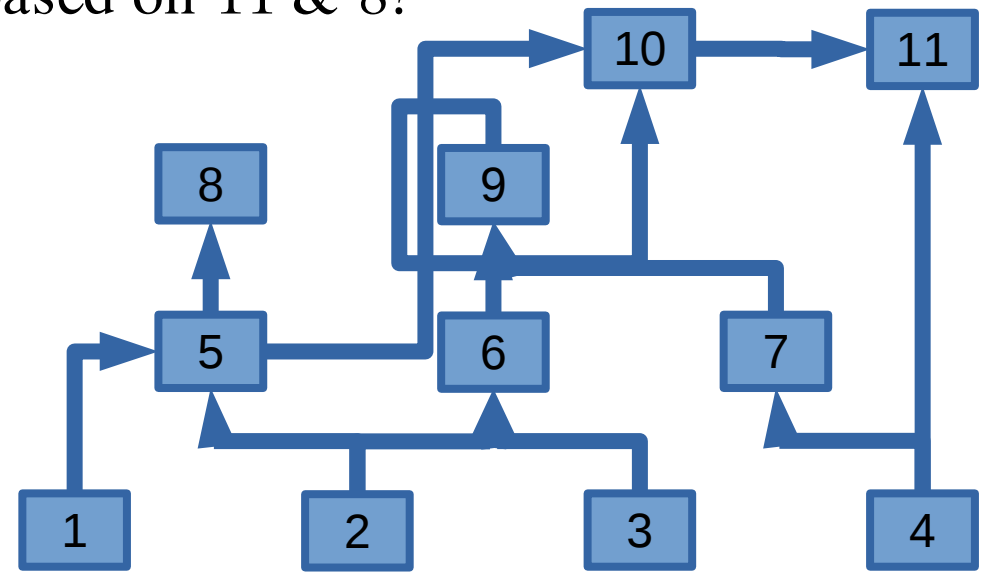
- The database will be used for building sound effects and including them in movies
- The following operations are necessary to support such use
  - Search for existing sound effects
  - Render sound effect
  - Create sound effect
  - Update sound effect

Sound effects DB:
Example sound effects



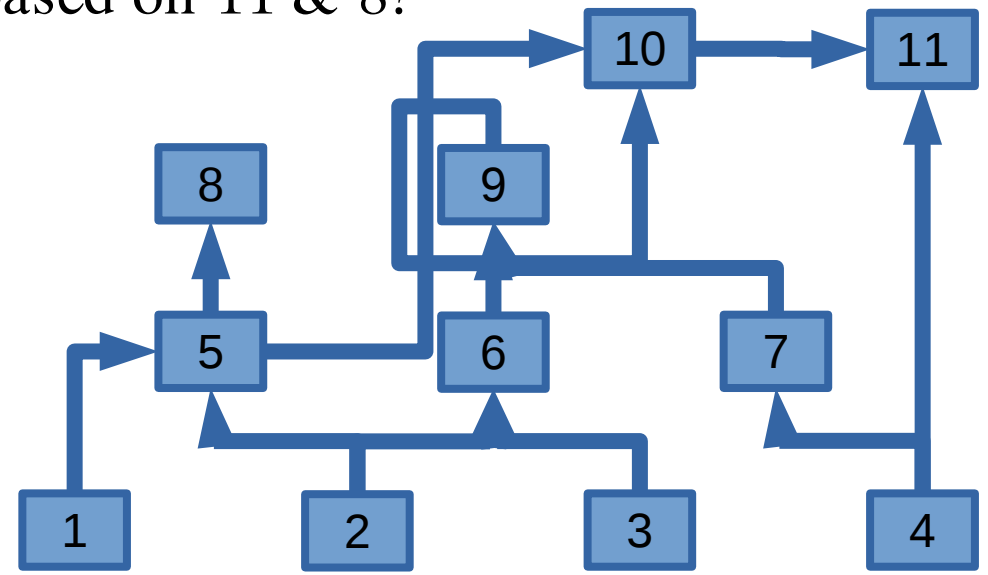
Sound effects DB: Using a relational database

- How would you search for sounds labeled "loud"?
- How would you render sound 11? How many queries would it take? What are they?
- How would you render sound 5? How many queries would it take? What are they?
- How would you render sound 4? How many queries would it take? What are they?
- How would you create a new sound based on 11 & 8?



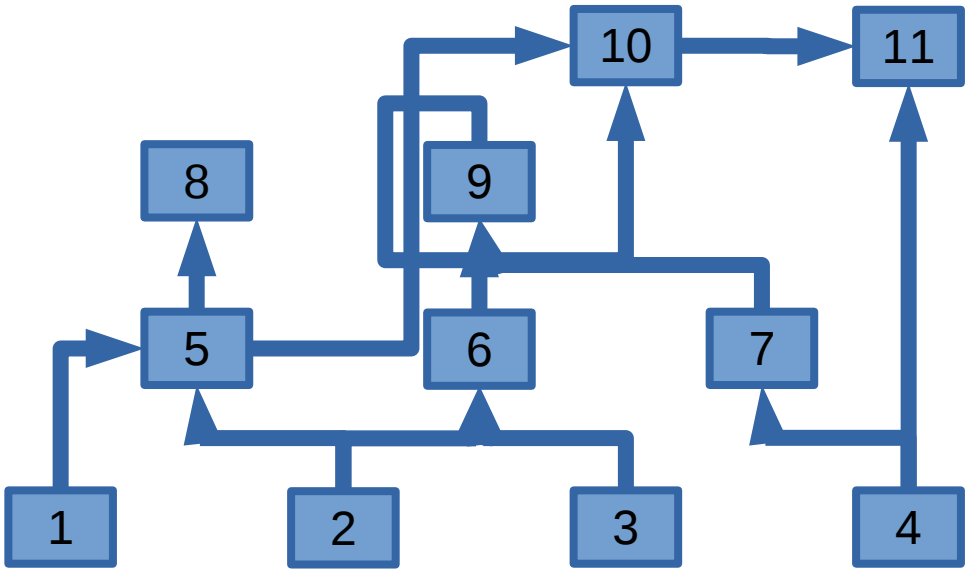
Sound effects DB: Using an object database

- How would you search for sounds labeled "loud"?
- How would you render sound 11? How many objects would you need to load? What are they?
- How would you render sound 5? How many objects would you need to load? What are they?
- How would you render sound 4? How many objects would you need to load? What are they?
- How would you create a new sound based on 11 & 8?



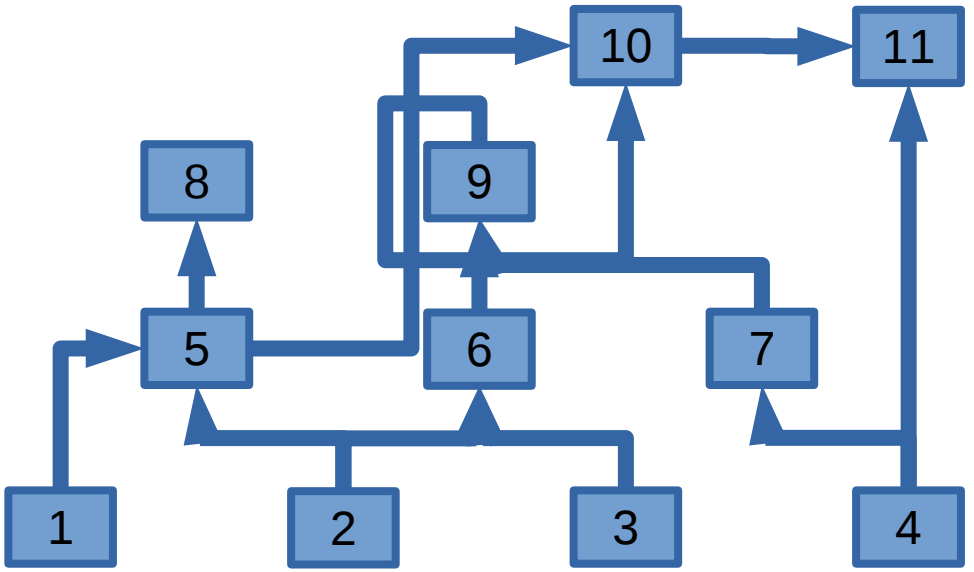
Sound effects DB: Other databases

- Is this a good candidate for a graph database?
- Is this a good candidate for an object relational mapper?
- How would you search, render, create, and update sound effects?



Sound effects DB: Distributed databases

- Is this a good candidate for single-master replication?
  - What if there were no update operations, just creation?
- Is this a good candidate for vertical partitioning?
- Is this a good candidate for horizontal partitioning?



Practice question: Image sharing site

- You are running an image-sharing site like Flickr
- Users mostly view images, but also upload and comment on them
- Say that you wanted to store different sized previews of the images
  - how much extra space would it take up to store one preview with half as many pixels in both width and height?
  - one half and one quarter?
  - one half, one quarter, and one eighth?
- How could you generate a lower quality version of a JPG file quickly without fully decoding it?
- How could you generate a 1/8th sized preview of a JPG file without fully decoding it?

Practice question: Image sharing site

- Your server is currently at 10% of its write capacity, but 80% of its read capacity
  - what would be the most cost-effective way to scale it?
- Your server is getting very popular in china, but images are loading too slowly
  - what would be the most cost-effective way to scale it?

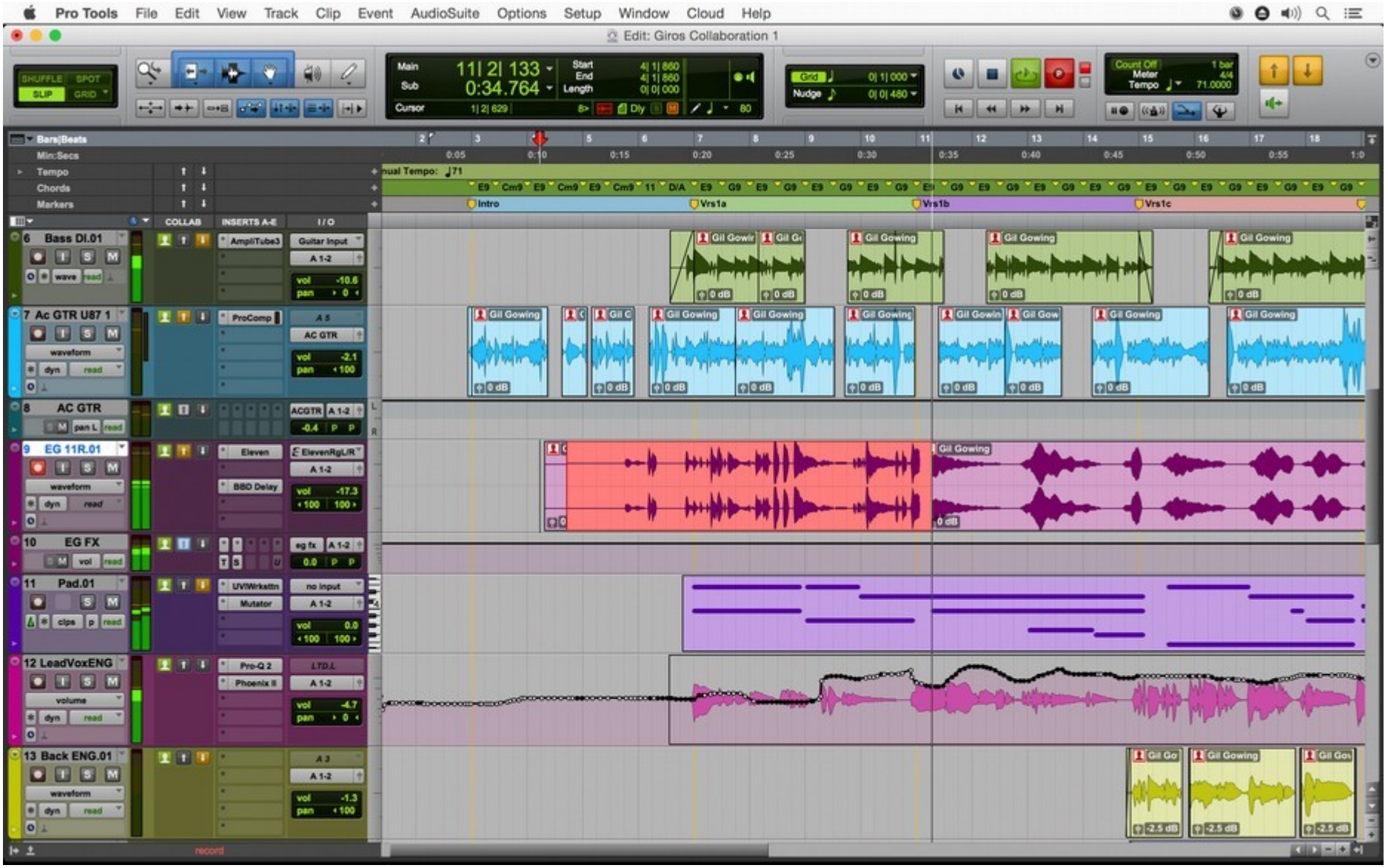
CISC 7610 Lecture 1
Introduction to multimedia databases

- Main question
  - How can we process and store multimedia data so that we can find what we are looking for in the future?
- Examples of multimedia databases
  - Production and distribution for audio and video
  - Distribution and discussion for audio, video, and images
  - Surveillance and intelligence for speech, images, video
- Issues related to multimedia datatypes
  - Storage of audio and video
  - Real-time streaming / processing / synchronization

Main question

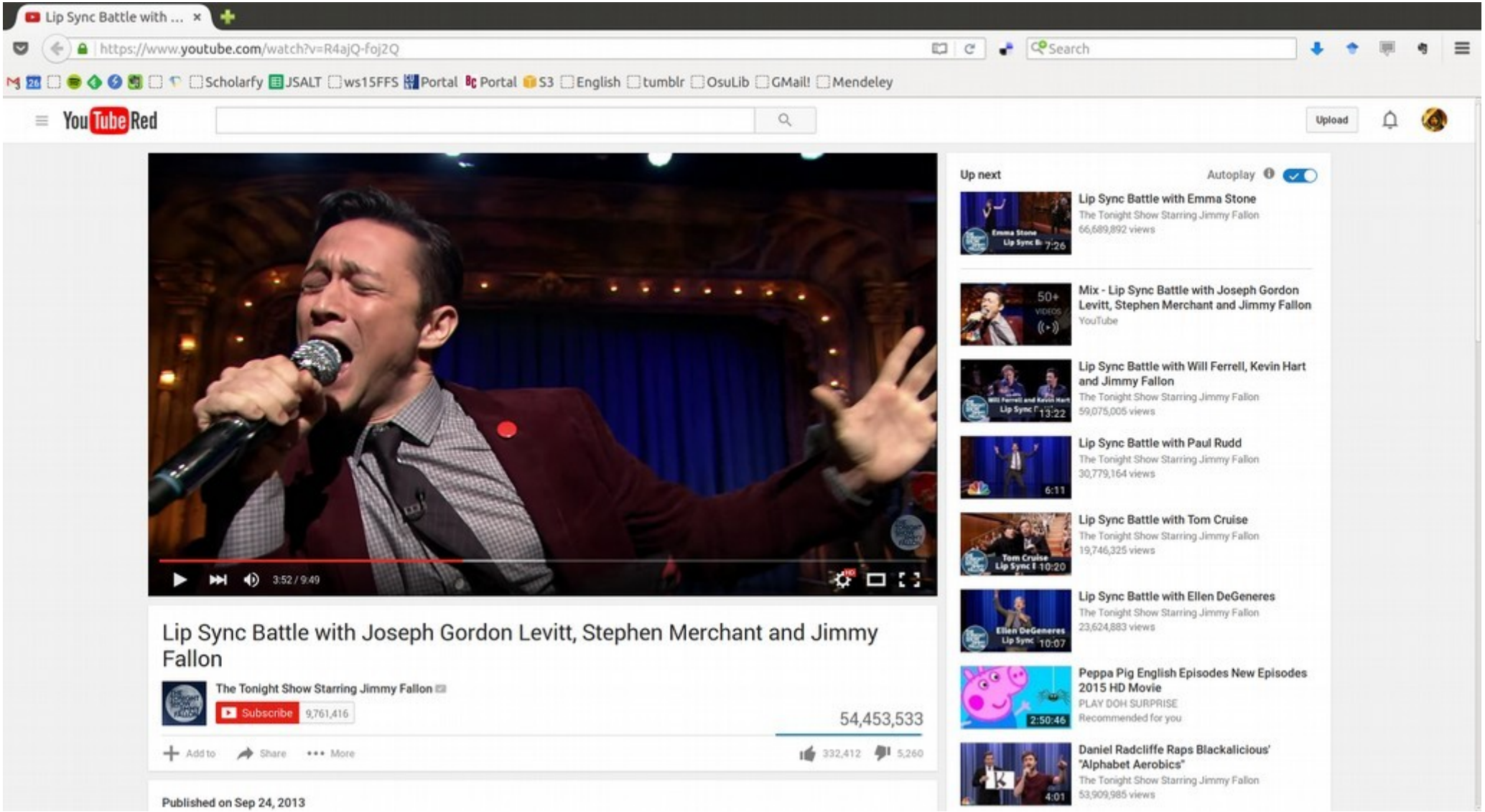
How can we process and store multimedia data so that we can find what we are looking for in the future?

Example: Production & distribution ProTools for audio



Example: Production & distribution
Media Composer for video

Example: Distribution & Discussion
YouTube for video



Example: Surveillance & Intelligence CCTV footage for events

Issues with multimedia datatypes: Querying

- How would you formulate a query?
  - Can't use SQL
  - At least need new user interface
- Additional queries you might want to run
  - Content-based similarity
  - Types of objects, their arrangement, composition
  - Specific people, places, days, times

Issues with multimedia datatypes: Storage

- Multimedia files are big
  - Audio
    - 44,100 samples per second
    - 2 channels
    - 16 bits per sample
  - Video
    - 1920 x 1080 pixels per frame
    - 3 colors per pixel
    - 30 frames per second

Issues with multimedia datatypes: Synchronization and real-time

- Audio and video are experienced in time
- Viewing, editing, and querying them must respect these temporal relationships (most of the time)
- For storage/transmission, data is broken into packets
  - Packets must be delivered by their deadline
  - Disks and networks introduce delivery delays
- Delivery quality can be characterized by throughput, latency, and jitter

CISC 7610 Lecture 2
Review of relational databases

- Relational database management systems
  - uses relational data structures / representation
  - has a declarative data manipulation language
- Structured query language (SQL)
- Example data modeling problem: music collection
- Entity-relationship diagrams
  - draw ER diagram for example problem
  - convert ER diagram to schema
  - create tables, insert data
  - query data

A relational database management system (RDBMS)

- Uses relational data structures
- Has a declarative data manipulation language at least as powerful as the relational algebra

Uses relational data structures

- Relation: table with rows and columns
- Attribute: column
- Tuple: row
- Key: combination of attributes that uniquely identifies each row
- Integrity rules: Constraints imposed upon the database

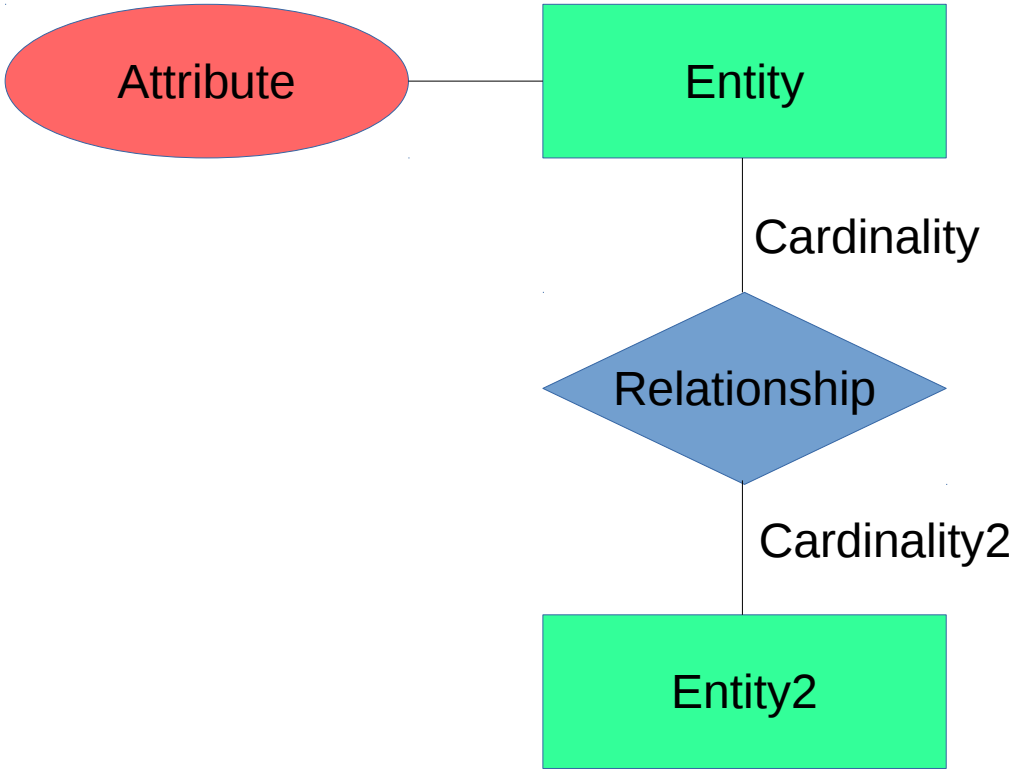
Has a declarative data manipulation language

- Declarative: says what, not how to manipulate data
- Relational algebra
  - Selection: extract a subset of tuples
  - Projection: extract a subset of attributes
  - Cartesian product: extract all combinations of pairs of tuples from two relations
  - Union: combine two sets of tuples
  - Set difference: remove one set of tuples from another

Structured query language (SQL)

- Data definition language
  - Define relational schemata (pl of schema)
  - Create/alter/delete tables and the attributes
- Data manipulation language
  - Insert/delete/modify tuples in relations
  - Query one or more tables
- Can implement relational algebra, but also takes some liberties with it

Entity-relationship diagrams



Data modeling example

- Artists: Name
- Albums: Name, Release date
- Tracks: Name, Duration, Number
- Each album has one artist
- Tracks can appear on multiple albums (compilations)

Translating ER diagrams to schema

- Entities become tables
- Attributes become their attributes
- Many-to-many relationships become join tables
  - Can have additional attributes
- Other relationships become foreign keys
  - One-to-one, many-to-one, one-to-many
  - Attributes added to table

Queries: find what we are looking for

- Search through the data
- Search through complex relationships
- Aggregate over the data for reporting
- And do all of this efficiently...

How do we make databases that are

- Effective (correct, durable, coherent, ...)
  - Transactions
- Efficient
  - Concurrency
  - Memory hierarchy
  - Indexing
  - Query optimization

Transactions

- A sequence of DB operations that represent a single real-world operation
- ACID properties – Guaranteed by RDBMSs
  - Atomicity: all operations happen or none
  - Consistency: transaction moves DB from one state that meets integrity constraints to another
  - Isolation: concurrent transactions have the same effect as serial
  - Durability: once committed, transactions effects are permanent
- Relaxed by NoSQL databases in various combinations

CISC 7610 Lecture 3
Multimedia data and data formats

- Redundant vs irrelevant information
- Perceptual limits of multimedia data: audio and video
- JPEG encoding of images
  - 8x8 blocks, YCbCr, DCT, Quantization, Encoding
- MPEG encoding of audio
  - quantization in audio, multiband quantization
  - MPEG audio compression hides quantization noise behind louder sounds
- MPEG and H.264 encoding of video
  - I, P, B frames
  - data streams: elementary stream, packetized ES, transport stream

Audio can be fully captured

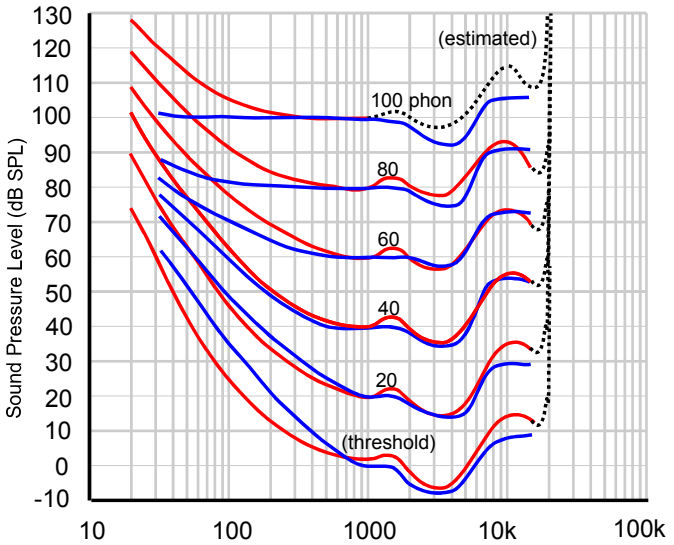
- Hearing
  - 20 – 20,000 Hz frequency range
  - 140 dB dynamic range of loudness (10,000,000 : 1)
- CD quality recording
  - Sampling rate: 44,100 Hz → max freq 22,050 Hz
  - Bit depth: 16 bits → dynamic range 96 dB
  - Bit rate: 44,100 samps/s x 16 bits x 2 chan. = 1.4 Mb/s
- Typical MP3 compressed recording
  - Bit rate: 128 kb/s (11x compression)

Video streams can still be improved

- Vision
  - 30 frames per second
  - 300 dots per inch at one foot viewing distance
    - 226M highest-res pixels to cover field of view
  - 140 dB brightness dynamic range (10,000,000 : 1) over time
- HD broadcast quality video = 1.5 Gb/s
  - 1920 x 1080 pixels/frame x 30 frames/s x 24 bits/pixel
- Typical H.264 compressed recording = 30 Mb/s (>50x)
  - 25 GB Blu-ray disc holds 2 hours, including audio

Compression lets us store and process these data efficiently

- Remove data that are redundant or irrelevant
- Redundant: implicit in remaining data
  - Can be fully reconstructed (lossless compression)
- Irrelevant: unique but unnecessary
  - For example: imperceptible to humans (lossy compression)

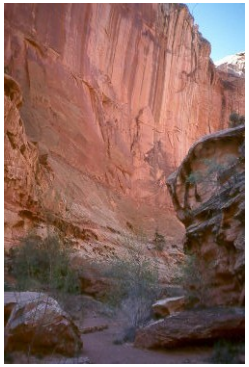


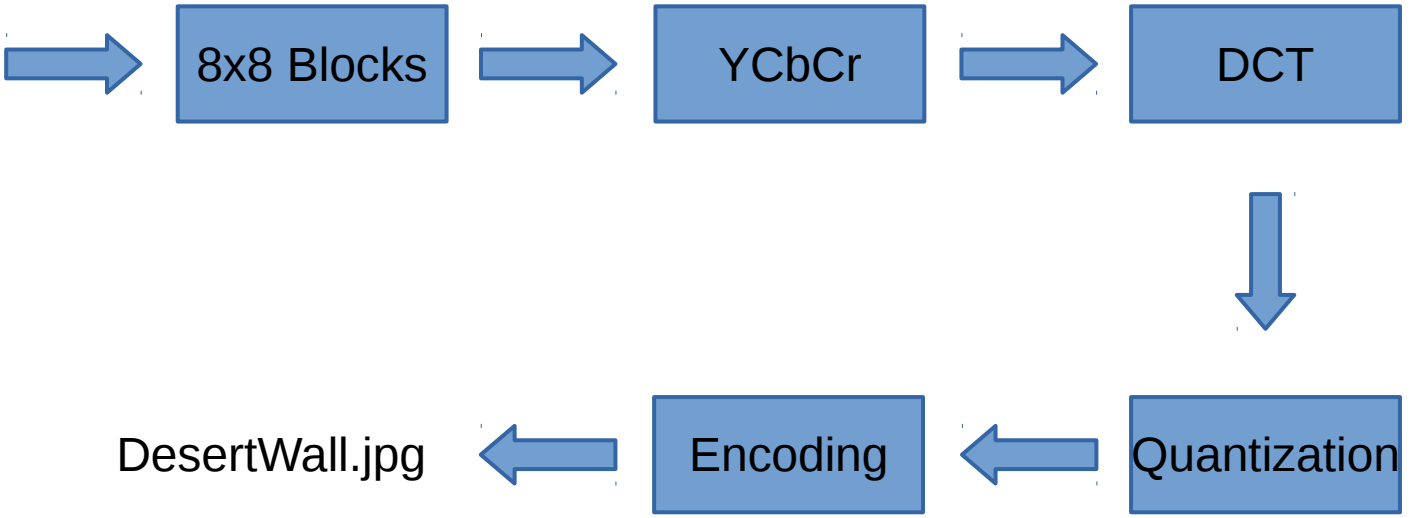
Equal-loudness contours (red) (from ISO 226:2003 revision)
Fletcher–Munson curves shown (blue) for comparison

Considerations for compression

- Latency: Amount of preceding signal that needs to be observed to compress a given sample
  - Important for real-time applications
- Locality: Amount of decoded signal that would be affected by changing one bit in encoded signal
  - Important for error robustness
- Generality: Variety of signals that can be encoded (efficiently) by a given encoder-decoder
- Decoder complexity vs encoder complexity

Image compression: JPEG encoding

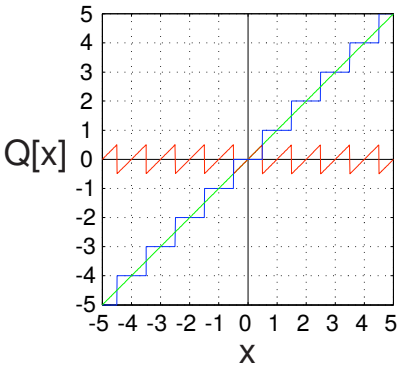


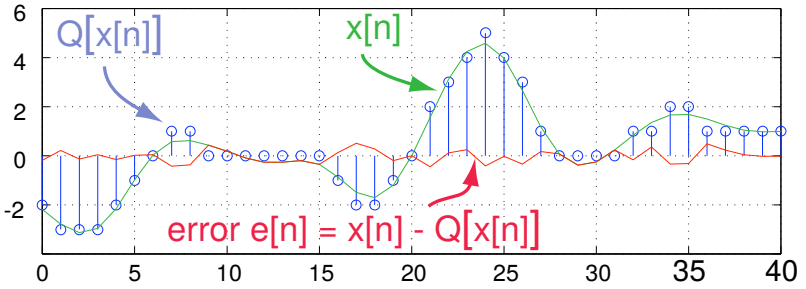


Images from: http://www.ams.org/samplings/feature-column/fcarc-image-compression

Quantization in audio

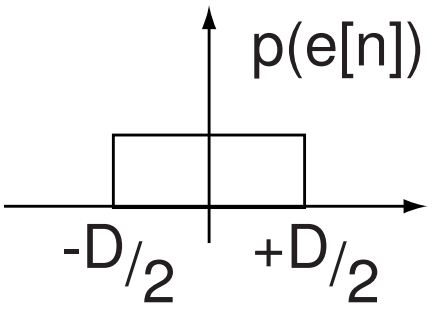
- Represent waveform with discrete levels





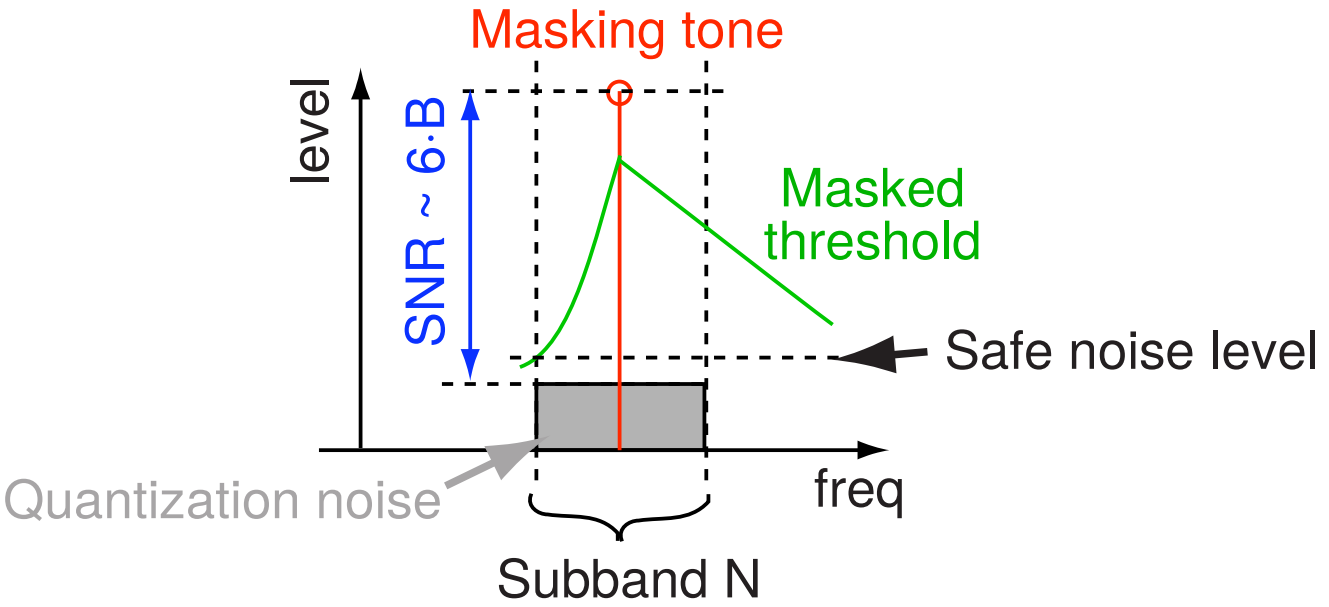
- Equivalent to adding an error of uniform noise

$$Q[x[n]] = \text{round}(x[n]/D)$$



MPEG Audio basic idea

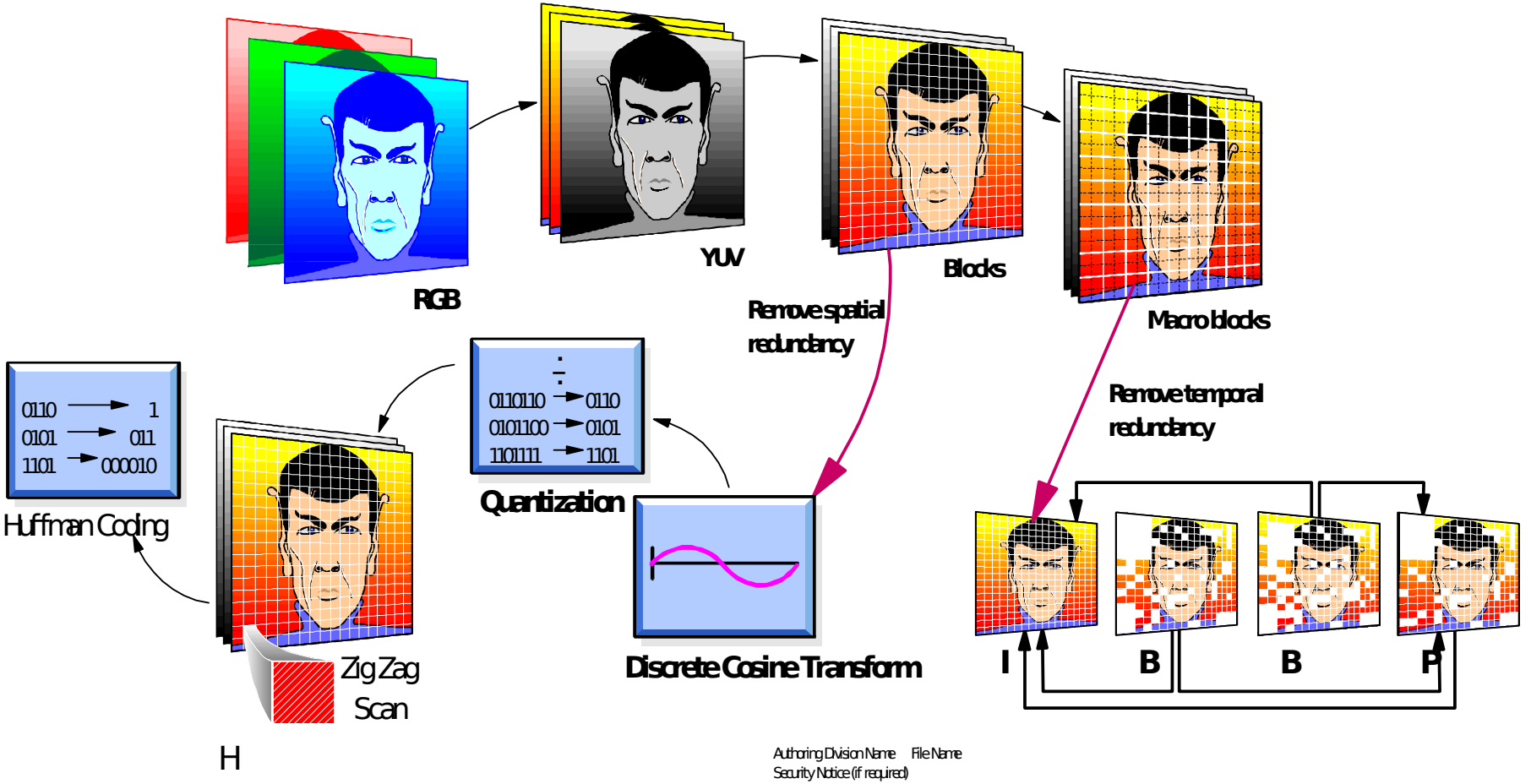
- Break audio into different frequencies
- Quantize each frequency as much as possible
  - While remaining imperceptible
- "Hide" quantization behind louder signals
  - Need psychoacoustic model of "hiding"



Video coding: MPEG2 and H.264

- Take advantage of redundancy in space and time
  - Predict pieces of frames from other frames
- Take advantage of irrelevance using quantization like JPEG
- Final lossless coding to squeeze out last bits
- Video codecs rely more heavily on “analysis-by-synthesis” than audio and image codecs

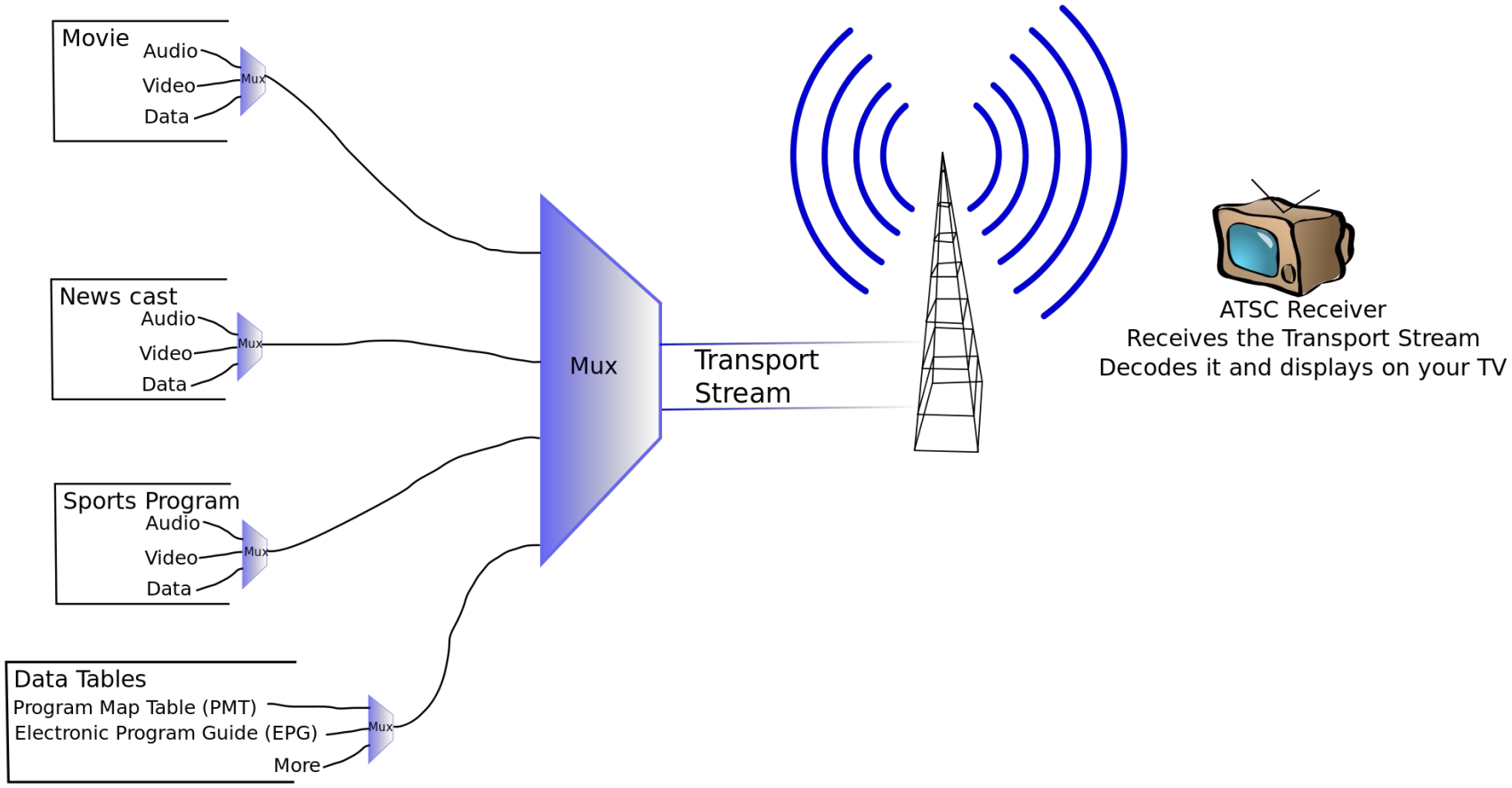
MPEG2 video encoding basics



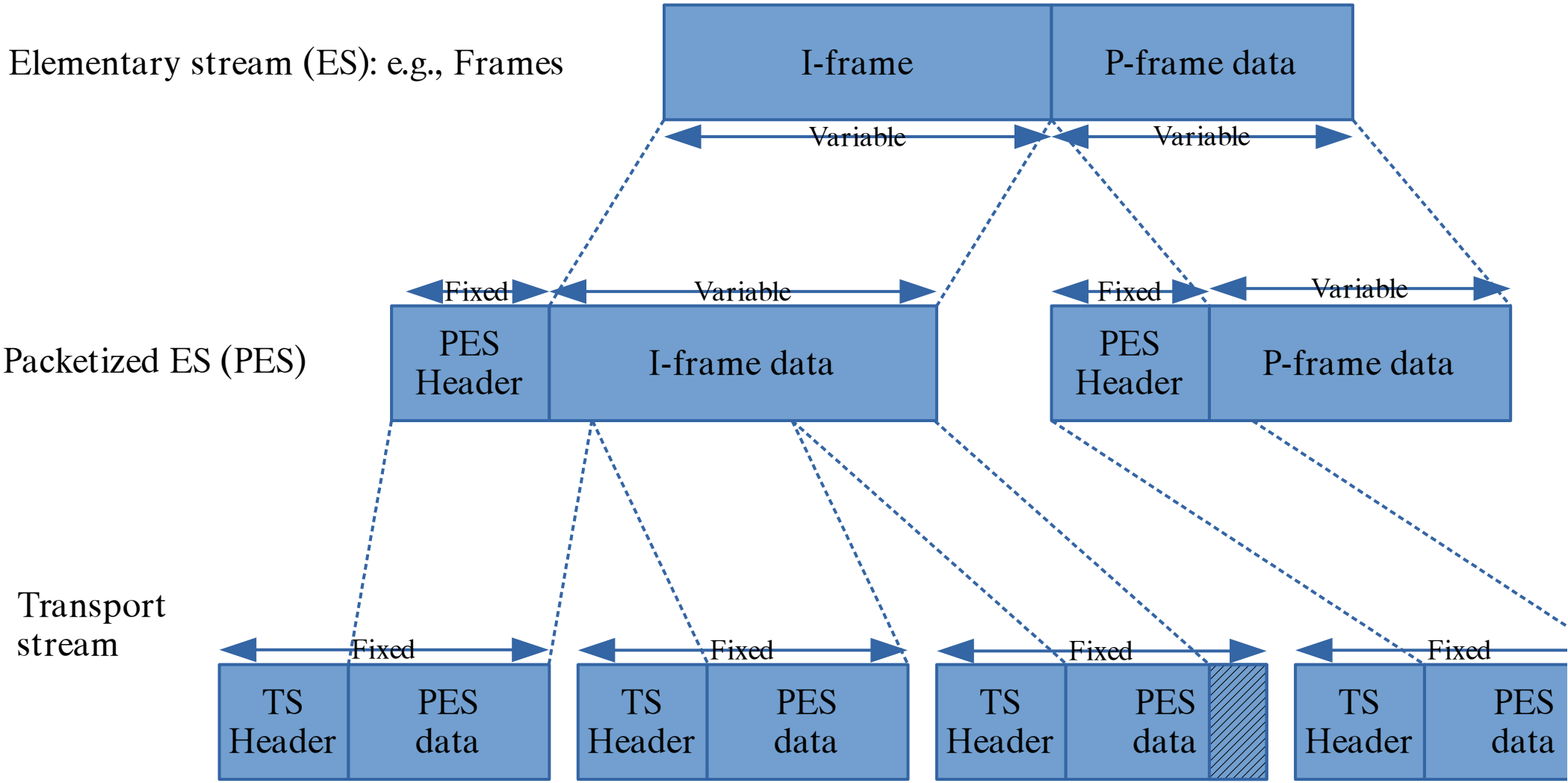
Used in DVD, digital (HD) TV

Image from: http://www.keysight.com/upload/cmc_upload/All/6C06MPEGTUTORIAL1.pdf

MPEG2 data is composed of streams



MPEG2 data is composed of streams



Summary

- Compression lets us store data efficiently
- Remove data that are redundant or irrelevant
- Redundant: implicit in remaining data
  - Can be fully reconstructed (lossless compression)
- Irrelevant: unique but unnecessary
  - For example: imperceptible to humans (lossy compression)

CISC 7610 Lecture 4
Approaches to multimedia databases

- Metadata
- Loose vs tight coupling of data and metadata
- Storing multimedia data in an RDBMS
- Object (oriented) databases, e.g., VelocityDB
- Graph databases, e.g., Facebook TAO, Neo4j
  - converting relational to graph model
- Object-relational mapping

Metadata is data about data

- Information on creation, content, and interaction
  - Creation: recorded by device about hardware, file
  - Content: what a human would say it's "about"
  - Interaction: viewer statistics, comments
- Holistic or time-related
- Multimedia file types typically include facilities for embedding metadata
  - EXIF for camera data
  - ID3 for MP3 files

Holistic (Object-level) metadata

- Automatic metadata from creation: EXIF
  - Frame rate, resolution, quality, codecs, equipment
- Metadata about the content
  - Topics, tags, participants
  - Mostly human-generated
  - Will discuss machine-generated later in course
- Usage and interaction metadata
  - Comments, thumbs ups, skips, shares

Time-related metadata

- Automatic metadata from creation (theoretically)
  - Location, view direction, camera/lens parameters, scene changes
- Metadata about the content
  - Closed captions, events/actions, scene descriptions
- Usage and interaction metadata
  - Comments, thumbs ups, skips, shares

Loose vs tight coupling of DB and multimedia storage

- Loose coupling
  - DBMS for metadata, filesystem for media
  - Simpler to implement, maintain, optimize, improve
  - Difficult to maintain integrity and consistency
- Tight coupling
  - DMBS for metadata and media
  - Maintains integrity and consistency between metadata and media
- We will strive for **tightly coupled** systems

Loose coupling: Store data on the filesystem

```
-rw-r--r-- 1 mim mim   78237696 Sep 12 12:18 00145.MTS
-rw-r--r-- 1 mim mim    3428352 Sep 12 12:19 00146.MTS
-rw-r--r-- 1 mim mim  269383680 Sep 12 12:22 00147.MTS
-rw-r--r-- 1 mim mim   39493632 Sep 12 13:44 00148.MTS
-rw-r--r-- 1 mim mim  386801664 Sep 12 14:04 00149.MTS
-rw-r--r-- 1 mim mim  105848832 Sep 12 14:08 00150.MTS
-rw-r--r-- 1 mim mim   81948672 Sep 12 14:09 00151.MTS
-rw-r--r-- 1 mim mim   24281088 Sep 12 14:11 00152.MTS
-rw-r--r-- 1 mim mim   18026496 Sep 12 14:12 00153.MTS
-rw-r--r-- 1 mim mim   48340992 Sep 12 14:13 00154.MTS
-rw-r--r-- 1 mim mim  199428096 Sep 12 14:14 00155.MTS
-rw-r--r-- 1 mim mim  525441024 Sep 12 14:16 00156.MTS
-rw-r--r-- 1 mim mim   10819584 Sep 13 09:53 00157.MTS
[mim@mr-pc video]$
```

- Simple key-value store (filename → data)
- Stores some metadata (size, create/modify dates)
- Fast access, can be distributed
- But: hard to query, hard to add metadata

Tight coupling with RDBMS

- But media data is hard to store in one
  - Can store it in BLOBs (Binary Large Objects)
  - but can't do much with them
- No notion of constituent parts
  - Streams, tracks, frames
- Difficult to extend functionality to include media-specific operations
- The point of RDBMS is run-time querying
  - But BLOBs are not useful in this context

RDBMS strengths and weaknesses

- Strengths
  - Extremely flexible queries
  - Database can have arbitrary schema
  - Some overhead for accesses (esp with locking)
  - Can distribute reads across multiple servers
- Weaknesses
  - Difficult to build hierarchical models
  - Fixed schema
  - Can’t perform media-specific operations
  - Difficult to support distributed writes

Tight coupling with Object (oriented) databases

- Application using the database
  - Is procedural
  - Deals with one item of data at a time
  - Creates complex data structures
- The database, conversely
  - Is declarative
  - Deals with tuples in sets instead of individually
  - Holds data in flat tables
- Combining them exposes an “impedance mismatch”
- Can we create a database that stores data in a way more similar to the application?

Object databases

- Persistent store for objects
- Preserves object type and structure (nesting)
- Preserves references between objects
- Easy to combine data and metadata
- Little querying, mainly reference-following
  - Lookups performed in code, not in query language
- Example: VelocityDB for C#

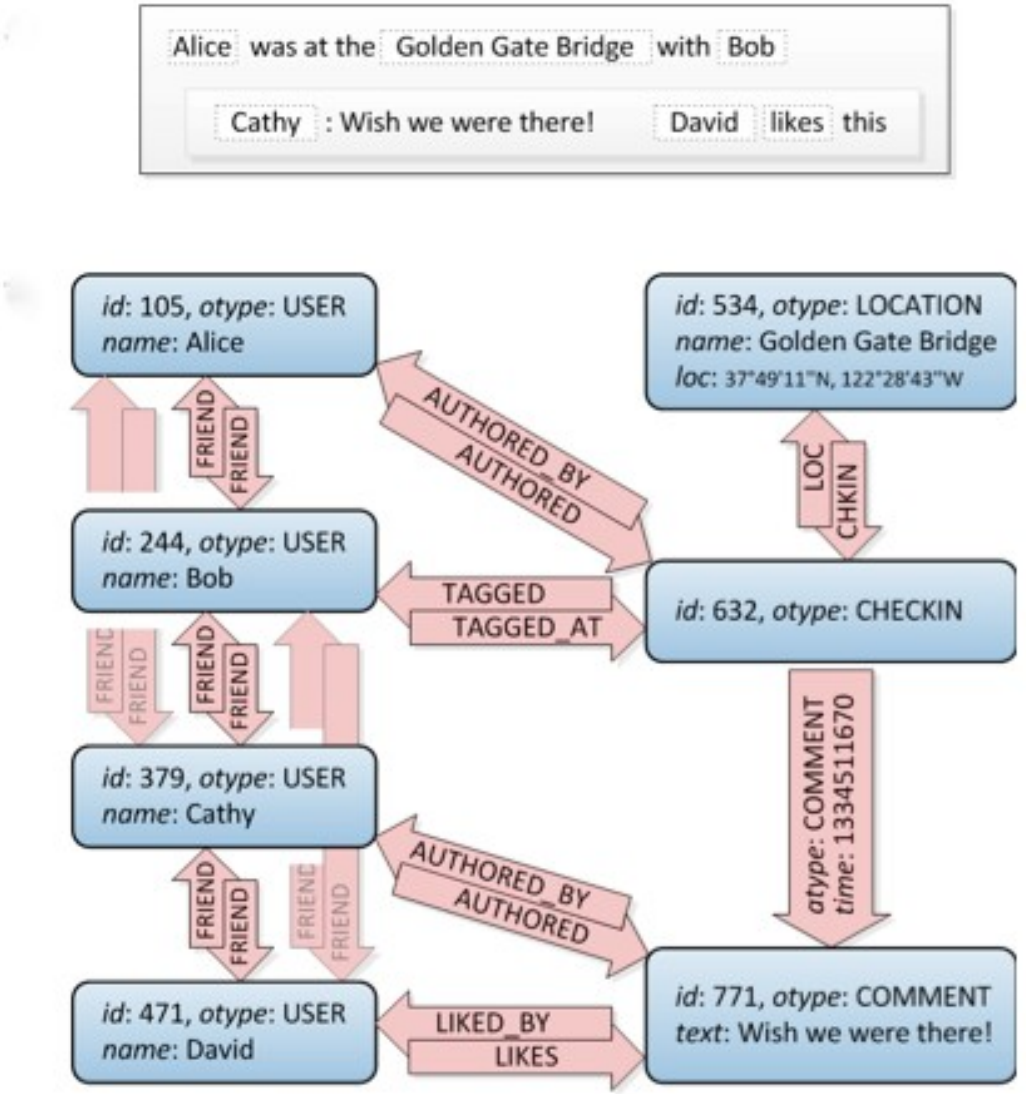
Object databases strengths and weaknesses

- Stengths
  - Stores objects directly
  - Easy to extend datatypes
  - Arbitrary and modify-able schema
  - Stores metadata and data together
  - Can define media-specific operations
- Weaknesses
  - Hard to distribute load across machines
  - Hard to query
  - Mixes code and schema
  - Difficult to administer

Tight coupling with Graph databases

- One speed advantage of object databases over RDBMS, comes from reference following
- Why compute joins that you know the answer to?
- Precompute joins and store as a graph of relationships: a graph database
- Examples: Facebook TAO, Neo4j

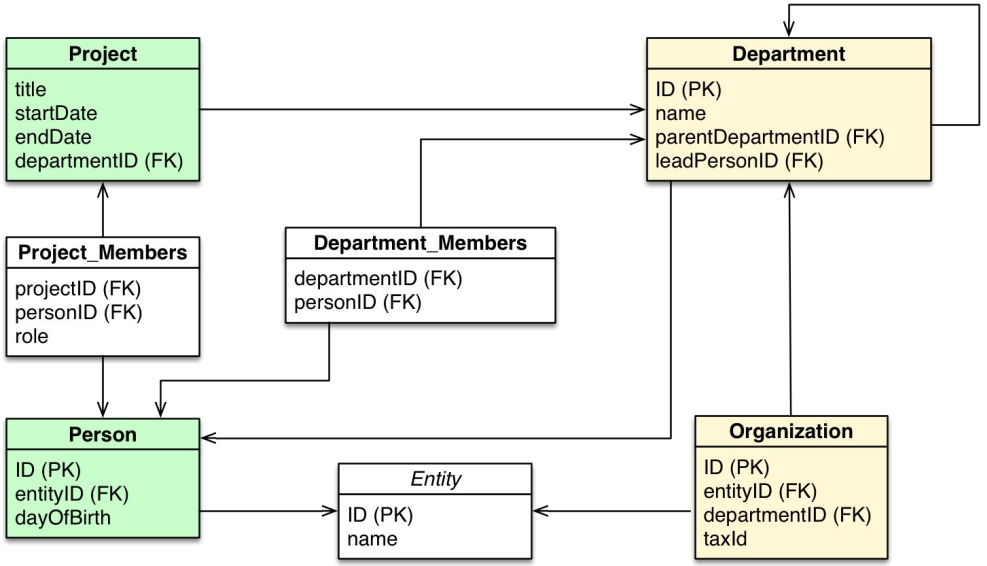
The Associations and Objects (TAO)

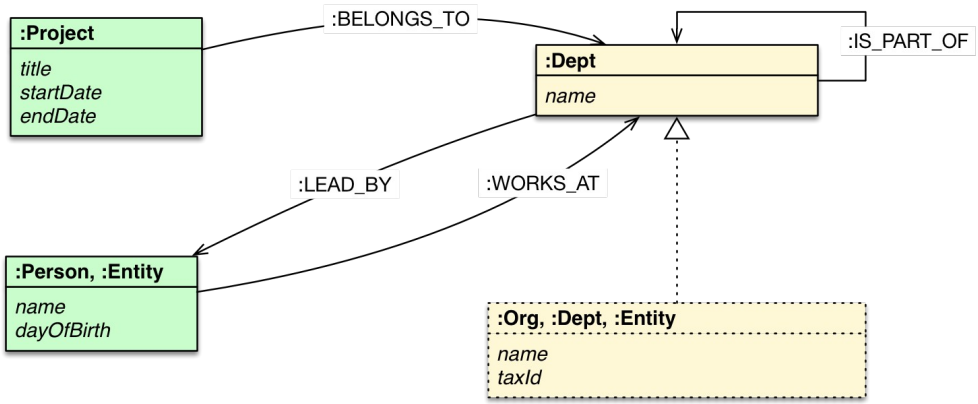


Neo4j Example: Converting relational to graph-based

- Each row in a entity table is a **node**
- Each entity table becomes a **label** on nodes
- Columns on those tables become node **properties**
- Foreign keys become **relationships** to the corresponding nodes in the other table
- Join tables become relationships, columns on those tables become **relationship properties**

Neo4j Example: Relational vs graph schema





Relational

Graph-based

Graph database
Strengths and weaknesses

- Strengths
  - Joins are precomputed
  - Easy to modify schema
  - Fast
  - Scalable
- Weaknesses
  - Harder to execute queries not embodied in relationships

Tight coupling with Object-relational mapping (ORM)

- Layer between RDBMS and OO program
- Translates certain OO calls into queries
- Translates results tuples into objects
- Almost all languages have an ORM

ORM Strengths and weaknesses

- Strengths
  - Easier to extend with new operations than RDBMS
  - Can be distributed across many computers (for reads)
  - Arbitrary schema
- Weaknesses
  - May require many queries
  - Still can’t store large files/objects
  - Fixed schema

Summary

- Awkward to fit multimedia data directly into relational databases
- Object-relational impedance mismatch makes it difficult to use RDBMS in programs anyway
- Object-relational mappings provide an imperfect bridge between the two
- Object databases remove the impedance mismatch
  - But make it awkward to query
- Graph databases extract associations from objects
  - Or pre-compute joins in a relational model

CISC 7610 Lecture 5
Distributed multimedia databases

- Scaling up vs out
- Replication: copy and coordinate, purposes, types
- Partitioning: horizontal vs vertical
  - horizontal advantages: scale forever
  - horizontal disadvantages: cross-shard joins difficult
- Scalability
- CAP Theorem: can't replicate to another node, go ahead or wait?
  - sacrificing availability
  - sacrificing consistency
- NoSQL: throw out ACID and SQL language
  - Key-value, key-document, column-family, graph
- NewSQL: keep SQL, some version of ACID
  - Google F1 / Spanner: mask high commit latency
  - VoltDB: restrict types of transactions allowed

Motivation

- YouTube receives 400 hours of video per minute
- That is 200M hours per year
- At 12 GB/hour (for H.264 HD quality)
- That is 2.6 Exabytes (2,600,000 TB) per year

- Which is more than one machine can handle
  - So how do we spread this across multiple machines?

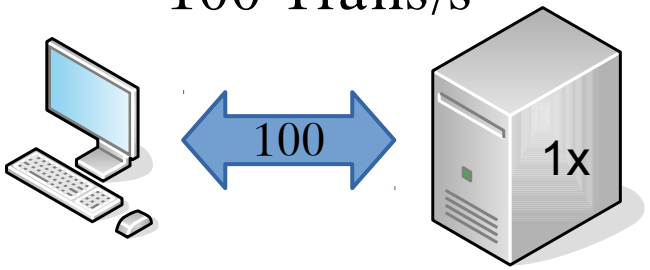
http://www.reelseo.com/vidcon-2015-strategic-insights-tactical-advice/

Scaling up vs scaling out

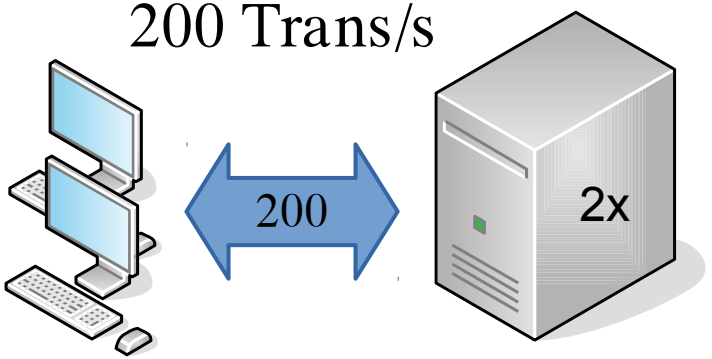
- Scaling up: buy a bigger server
  - Pro: Code stays pretty much the same
  - Con: Expensive, hard limits
- Scaling out: buy more servers
  - Pro: Much cheaper, soft limits
  - Con: Much more complicated code (distributed systems)

Scaling out: Replication vs partitioning

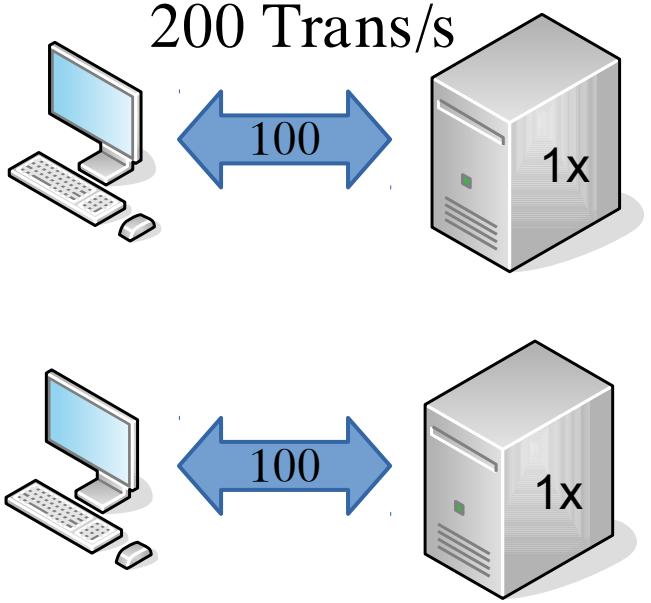
Initial system
100 Trans/s



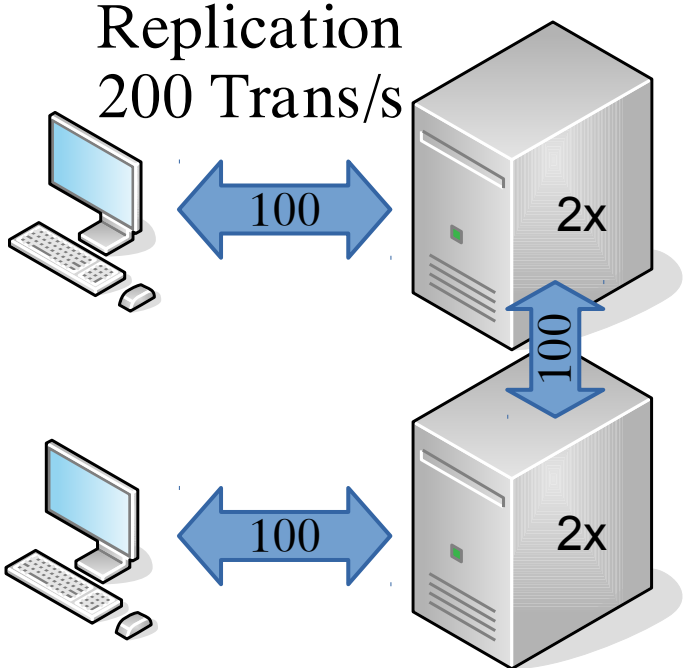
Scale up
200 Trans/s



Scale out
Partitioning
200 Trans/s

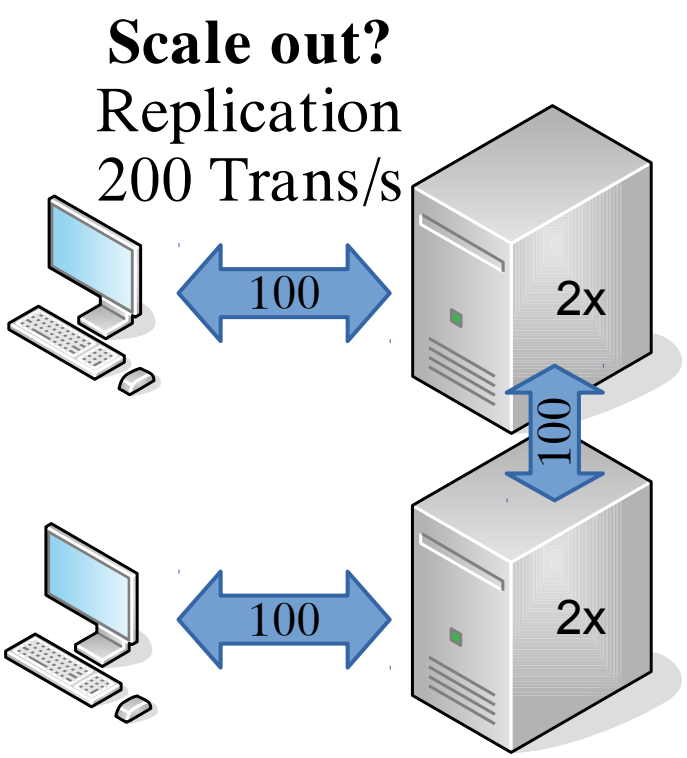


Scale out?
Replication
200 Trans/s



Replication: copy and coordinate

- Replication maintains identical copies of data on multiple machines
- Reads can come from any machine
- Writes must be applied to all machines



Purpose of replication

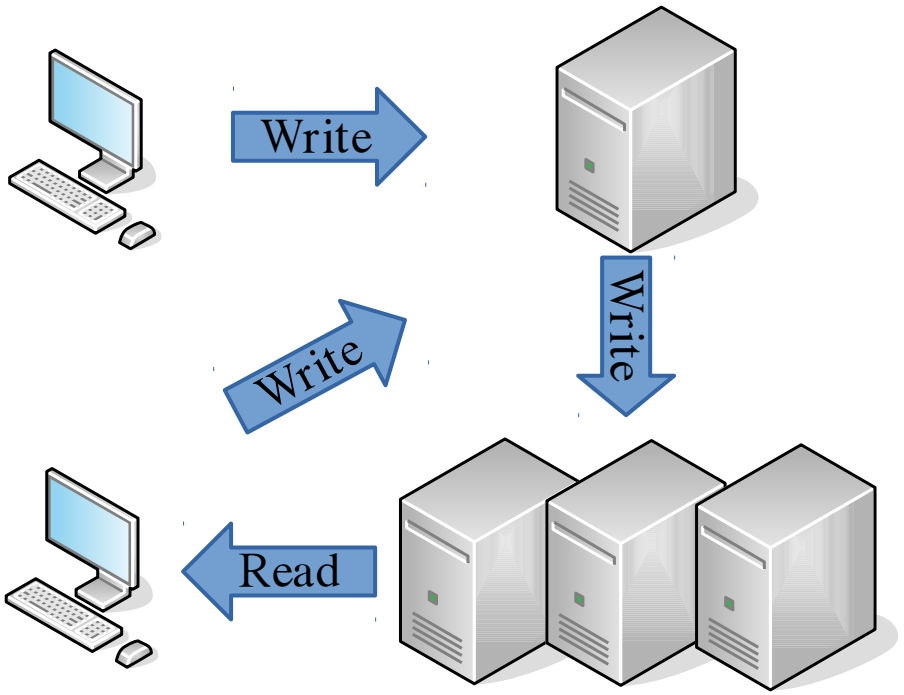
- Data distribution
  - Geographical diversity for lower latency and redundancy
- Load balancing
  - Spread requests among multiple servers
- Backups and recovery
  - Make and restore copies without downtime
- High availability
  - Hot spare for fast recovery

Types of replication

- Eager / synchronous
  - Transaction waits for all replicas to be updated
  - Maintains consistency, but can cause delays
- Lazy / eventual / asynchronous
  - Transaction waits for one replica to be updated
  - Changes propagated to other replicas eventually
  - Faster, but can lead to conflicts between replicas modified in different ways

Master/slave replication

- Only “master” replica accepts writes
  - Avoid consistency issues
  - Sends updates to others
  - Single point of failure
- All replicas serve reads
- If master goes down, another replica can be promoted



Partitioning

- Sometimes, data can be divided into uncoupled or loosely coupled partitions
  - Then scaling to more machines just requires dividing into more partitions
- Horizontal partitioning: divide relations by ID
  - Also known as sharding
- Vertical partitioning: divide relations by attributes
  - Compare to database normalization

Partitioning example

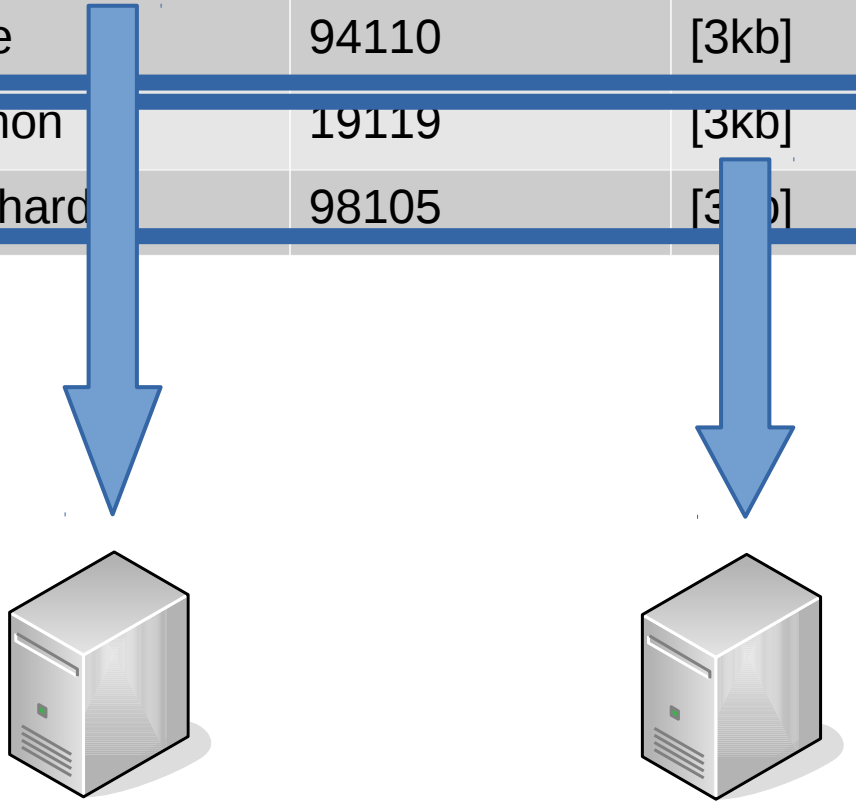
ID	Name	Zipcode	Thumbnail	Photo
1	David	02138	[3kb]	[2MB]
2	Jared	43201	[3kb]	[2MB]
3	Sue	94110	[3kb]	[2MB]
4	Simon	19119	[3kb]	[2MB]
5	Richard	98105	[3kb]	[2MB]

Vertical Partitioning example

ID	Name	Zipcode	Thumbnail	Photo
1	David	02138	[3kb]	[2MB]
2	Jared	43201	[3kb]	[2MB]
3	Sue	94110	[3kb]	[2MB]
4	Simon	19119	[3kb]	[2MB]
5	Richard	98105	[3kb]	[2MB]

Horizontal Partitioning example

ID	Name	Zipcode	Thumbnail	Photo
1	David	02138	[3kb]	[2MB]
2	Jared	43201	[3kb]	[2MB]
3	Sue	94110	[3kb]	[2MB]
4	Simon	19119	[3kb]	[2MB]
5	Richard	98105	[3kb]	[2MB]



Horizontal Partitioning advantages

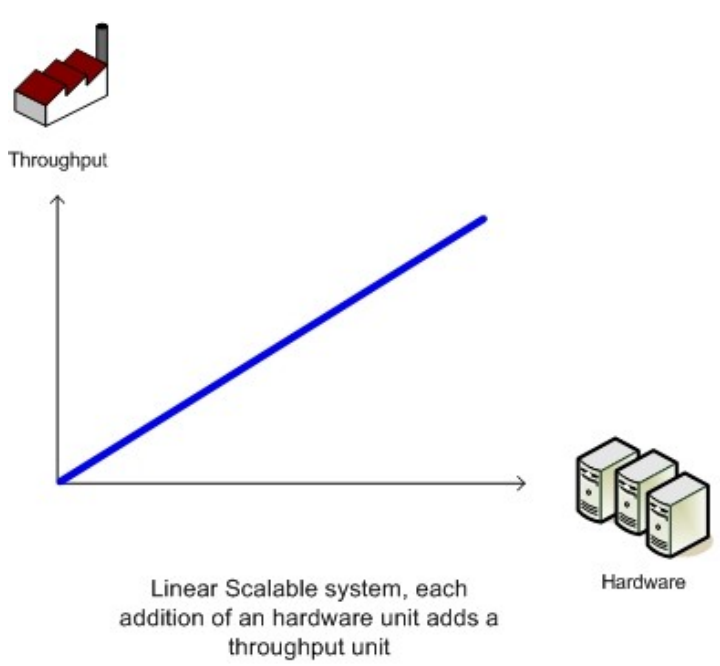
- Higher write bandwidth than replication
  - All shards accept writes
  - So N times higher capacity
- Higher scalability than replication
  - Continue to sub-divide shards to scale up without limit

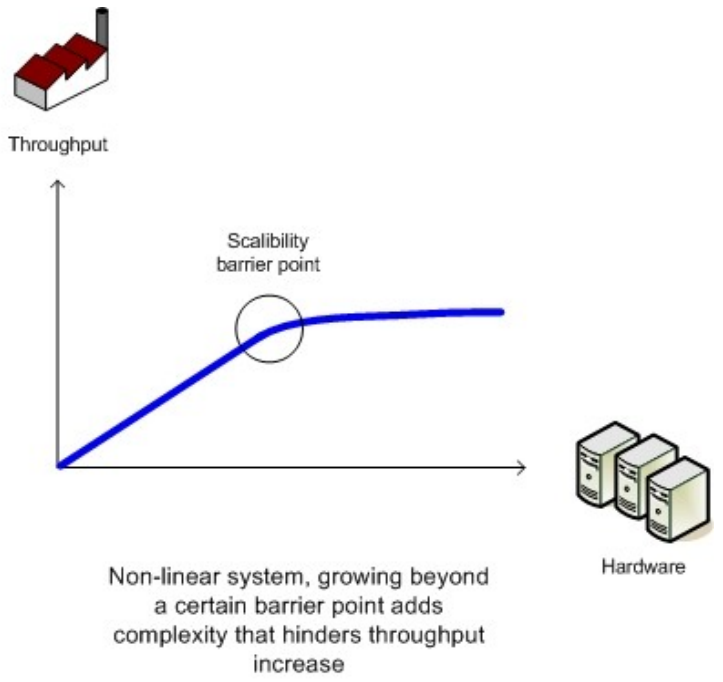
Horizontal Partitioning disadvantages

- Re-balancing is necessary and costly
  - When one shard grows too large
  - When adding new machines
- Cross-shard joins are slow or impossible
  - Additional reliance on network
  - Shards could be in separate data centers (US vs Europe)

Scalability

- How hard is it to double your current capacity?
  - Capacity for traffic, storage, transactions
  - Capacity for read or write operations
- Independent of current processing speed





http://docs.gigaspaces.com/sbp/first-xap-app-step-4.html

Brewer's CAP theorem

- These three properties cannot be achieved by a distributed system simultaneously
  - Strong Consistency: All clients see the same data at the same time
  - Availability: All requests receive a response as to their success or failure
  - Partition tolerance: The system continues to function in the event of network failures

Brewer's real CAP theorem

- A node fails to communicate with another node when attempting to replicate its data (P)
- It can decide to
  - Go ahead anyway, sacrificing consistency (C)
  - Wait for the other node, sacrificing availability (A)
- Business considerations: availability > consistency
  - Take the customer's order and sort it out later if necessary

Sacrificing availability

- Wait until all nodes can synchronize
- “Amazon claim that just an extra one tenth of a second on their response times will cost them 1% in sales. Google said they noticed that just a half a second increase in latency caused traffic to drop by a fifth.”
- Examples: Multi-master DBs, Neo4j, Google BigTable

http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

Sacrificing consistency

- Go ahead with update (Eventual consistency)
- Problems
  - Pushes complexity from database into application
  - When is “eventually”?
- Examples
  - Domain name service
  - Facebook’s Cassandra and Voldemort
  - CouchDB

NoSQL vs NewSQL

- NoSQL: restrictive interpretation of CAP theorem
  - Throw out ACID transactions with SQL
  - In order to increase scalability
- NewSQL: more nuanced interpretation of CAP
  - Regain ACID transactions and SQL
  - Maintain scalability by optimizing for quick, localized transactions
- SQL is just the query language, independent of CAP
  - And still need a way to query

Types of NoSQL Databases

- Key-value store
- Key-document store
- Column-family stores
- Graph databases (again)

Summary

- Scaling up is easier, but scaling out is more sustainable in the long term
- Replication copies data, allowing parallel reads
- Partitioning divides up data to allow parallel writes
- CAP theorem says that you must choose between availability and consistency
- NoSQL sacrifices ACID transactions for scalability
- NewSQL maintains ACID transactions with scalability