SQL or NoSQL?

That is the question....

Welcome!

Tonight I am going to try to:

- Explain different types of database structures
- Give some suggestions as to when you would use one over another
- Demystify some terminology
- Give you a crash course in different ways of modelling data
- Give you chance to have a go yourself!

Relational (SQL) vs NoSQL

Relational databases

Data stored in tables with strict schemas

Optimising for storage rather than compute power

Data storage is modelled around the data

Hard to scale

NoSQL databases

Data stored in tables / documents / graphs with NO strict schemas

Optimising for compute power rather than storage

Data storage is modelled around the questions

Easy to scale

Types of NoSQL database

Key - Value store

E.g. DynamoDB

Good for fast lookup where you have unique identifiers

Document database

E.g. MongoDB, Elasticsearch

Good for fuzzy search

Graph database

E.g. Neo4j, Neptune

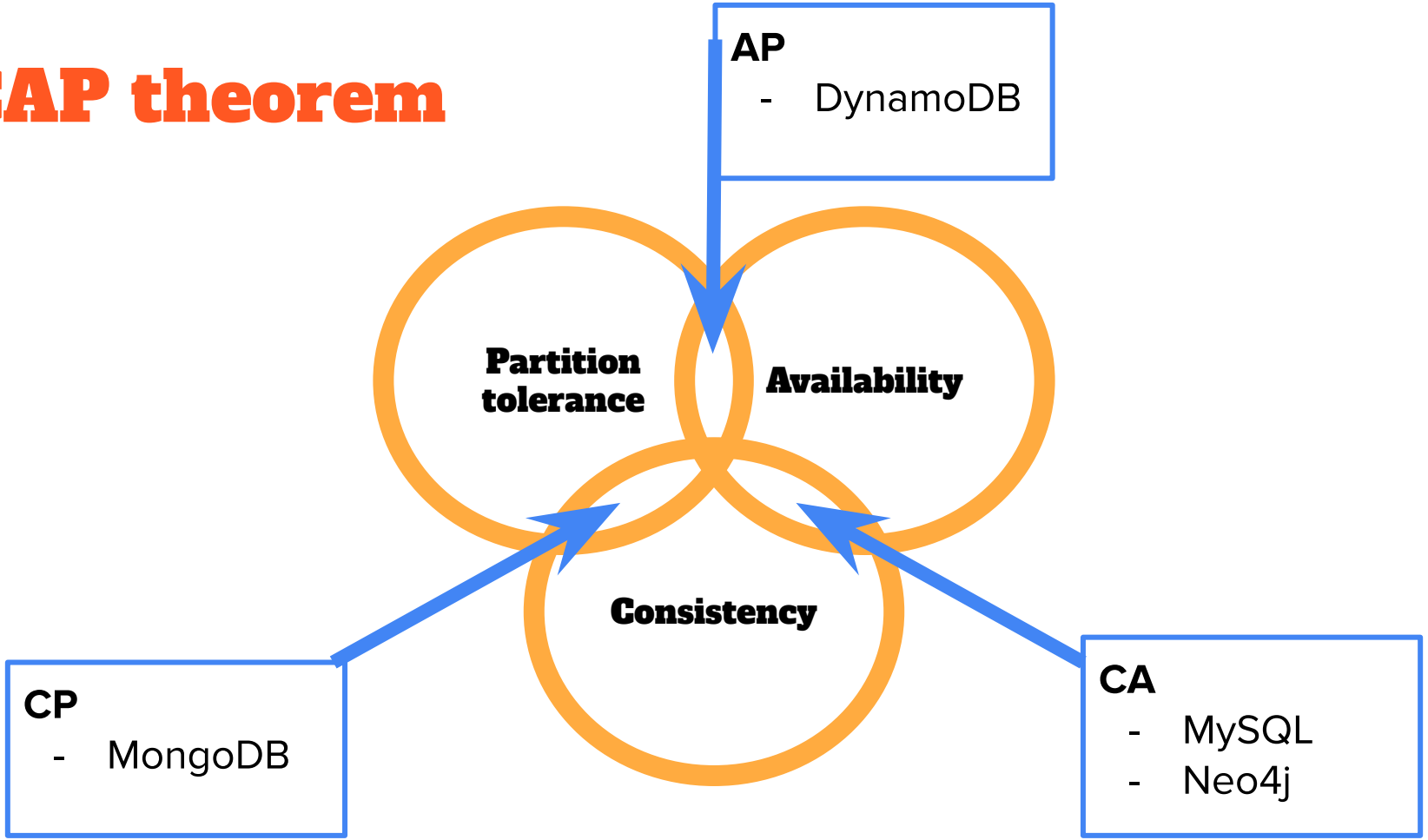
Good for complicated relationships

Columnar database

E.g. Athena querying parquet files on S3, Redshift

Good for grouping and aggregating

CAP theorem


AP
- DynamoDB
Partition tolerance
Availability
Consistency
CP
- MongoDB
CA
- MySQL
- Neo4j

ACID vs BASE

ACID

Atomic: All operations in a transaction succeed or every operation is rolled back.

Consistent: On the completion of a transaction, the database is structurally sound.

Isolated: Transactions do not contend with one another. Contentious access to data is moderated by the database so that transactions appear to run sequentially.

Durable: The results of applying a transaction are permanent, even in the presence of failures.

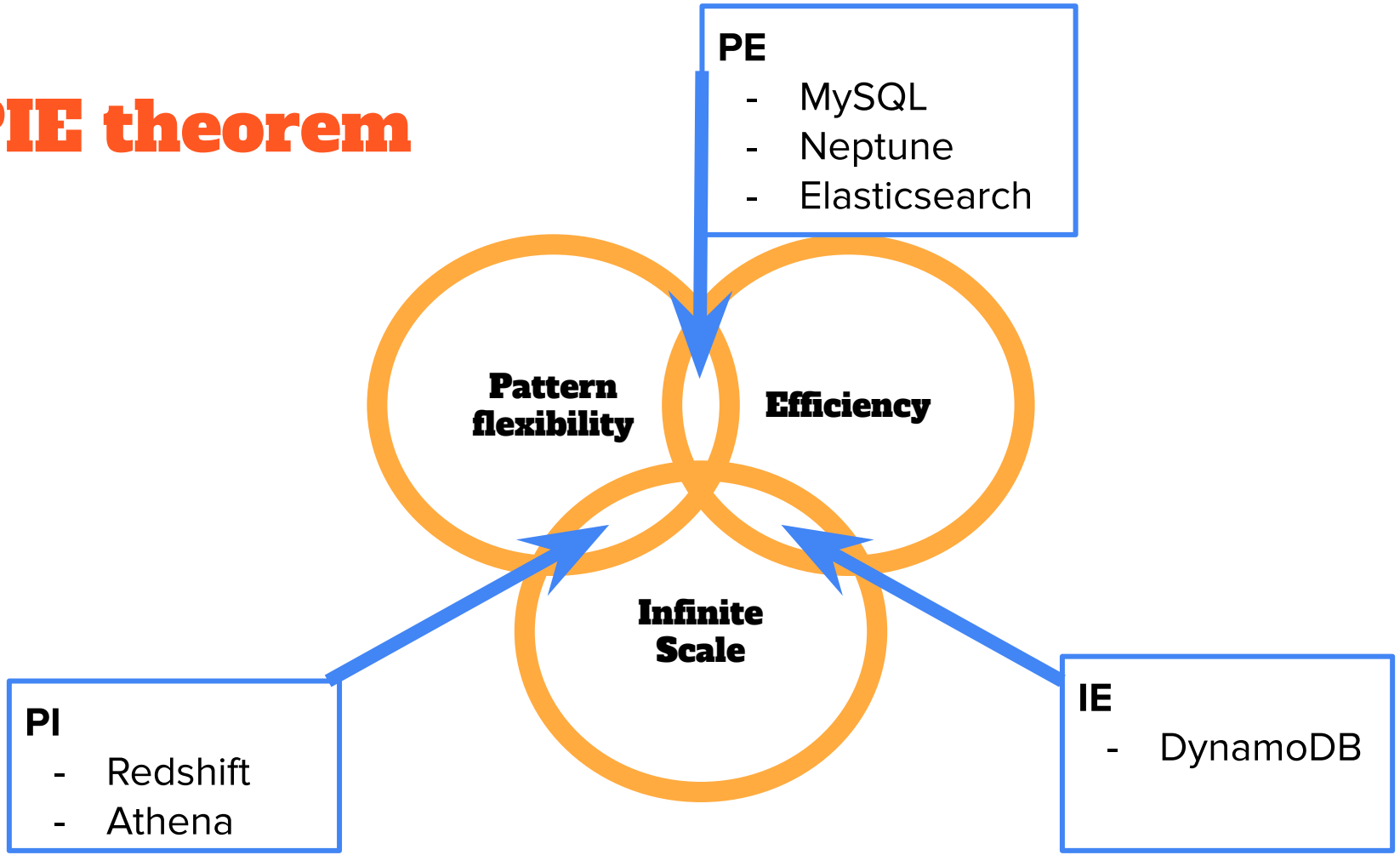
BASE

Basic Availability: The database appears to work most of the time.

Soft-state: Stores don't have to be write-consistent, nor do different replicas have to be mutually consistent all the time.

Eventual consistency: Stores exhibit consistency at some later point (e.g., lazily at read time).

PIE theorem


PE
- MySQL
- Neptune
- Elasticsearch
Pattern flexibility
Efficiency
Infinite Scale
PI
- Redshift
- Athena
IE
- DynamoDB

Questions to ask yourself

How much data? How often does it change?

How messy?

Is query response speed important?

How important that everyone gets the same view of the data?

Do you know the questions you need to ask of the data in advance?

How many different ways do you need to query the data?

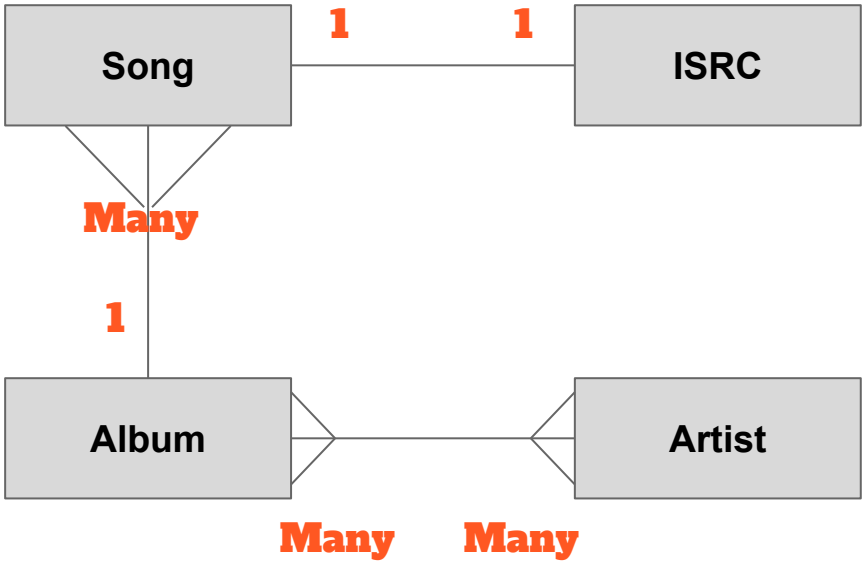
Crash course in data modelling!

We are going to look at three different types of data stores

- Relational database (e.g. MySQL)
- Key - value store (e.g. DynamoDB)
- Graph database (e.g. Neptune, Neo4j)

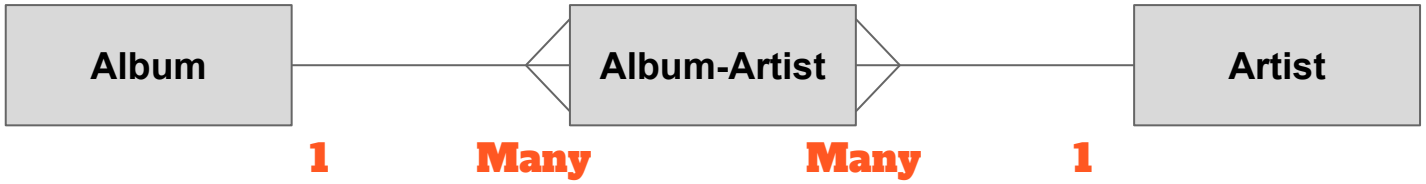
Relational data modelling

Entity - relationship model



Relational data modelling

Avoid many to many relationships - “normal form”



Album_ID
Album_Name
Release_Date

Album_ID
Artist_ID

Artist_ID
Artist_Name
Birth_Date

Key value data modelling

- “Give me all the Albums this Song is on, sorted by release date, given its ISRC”
- “Give me the title of this Album given its UPC”
- “Give me all the Songs on this Album given its UPC”

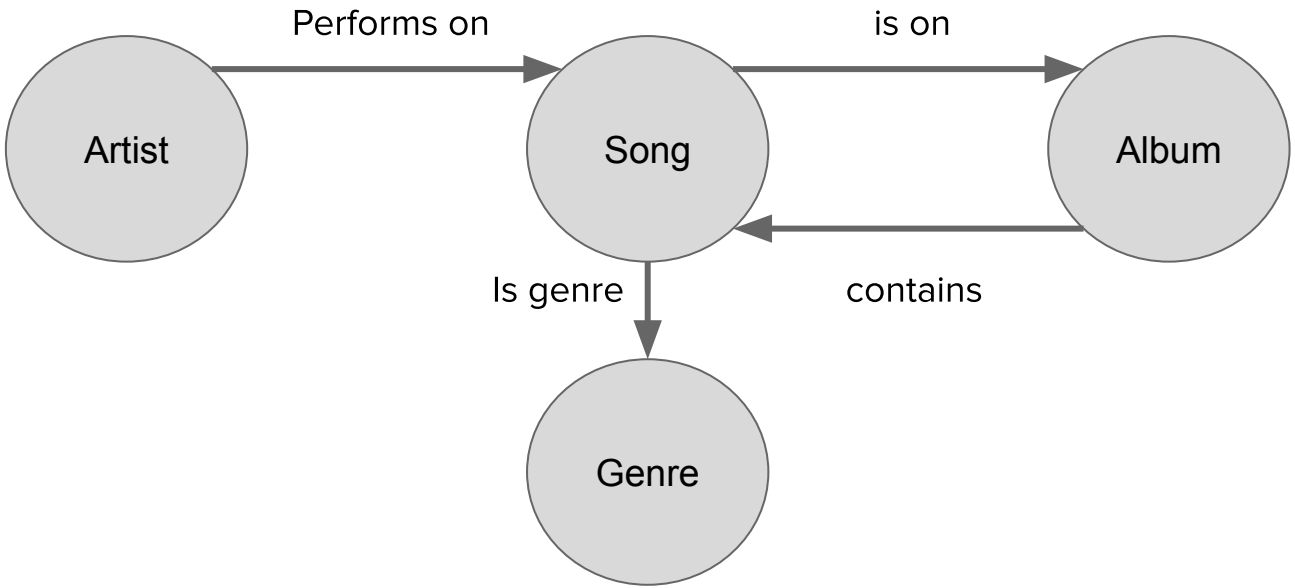
Partition Key	**Sort Key**			**Secondary index**
ISRC	**ReleaseDate-UPC**	**Song Name**	**Album Name**	**UPC**
GBTFD37373	20140401-123	A Great Song	Awesome Album	123
GBTFD37373	20180402-234	A Great Song	Greatest Hits	234

Key value data modelling

Tips

- Spend time working out what questions are needed
- You can use the same table for lots of different types of data
- You can put secondary indexes on to other columns to allow more questions on the same data
- Don’t worry about data duplication, but make sure records don’t get out of sync if data changes
- Keys must be unique or the data will be overwritten

Graph data modelling



Graph data modelling

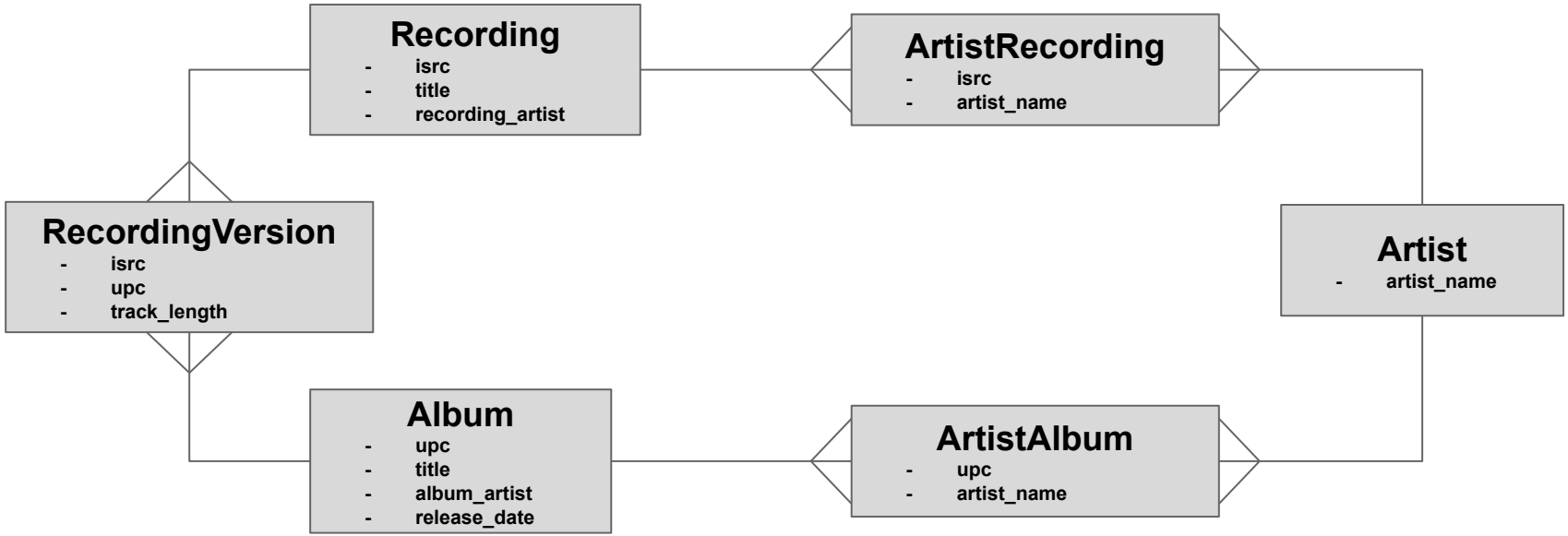
Tips

- A bit like Entity Relationship modelling, but you don’t need to worry about normalisation
- Spend time working out what questions are needed
- If a piece of data (e.g. genre name, ISRC) is used often in queries, make it a node, if you’re only reading it then keep it as metadata for a node or edge

Data modelling exercise

- In groups look at the problem and use one of the previous data structures to model it.
- THERE IS NO SINGLE RIGHT ANSWER - understand the trade offs you are making
- If you need more information, just ask, or make an assumption and write it down

Relational database



Key Value database

Table 1

Partition key	**Sort key**	**Secondary index**				
isrc	release_date-upc	upc	recording_title	recording_artist	album_title	album_artist

Table 2

Partition key	**Sort key**						
artist_name	release_date-upc-isrc	isrc	upc	recording_title	recording_artist	album_title	album_artist

Graph database