香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

SCHOOL OF
DATA SCIENCE
數據科學學院

Ack: Prof. Jignesh Patel @ CMU

CSC3170
1: Rational Model

Chenhao Ma
School of Data Science
The Chinese University of Hong Kong, Shenzhen

What is a Database?

- Data
  - Some values referring to real-world facts.
  - May be in various formats, e.g., text, image, audio file, video file, etc.

- Database
  - A large collection of inter-related data.

- Database management system (DBMS)
  - DBMS = database(s) + a set of programs store and access data

Database Applications

- Many daily applications involve databases.
  - Banking
  - Universities
  - Digital music services
  - Social networks

What data are stored in the backend database of the digital music service?

Basic info
- Information about Artisits
- What Albums those Artists released
- …

Flat Files V.S. DBMS

- In early days, application programs were built on top of file systems.
- Store our database as comma-separated value (CSV) files that we manage ourselves in our application code.
  - Use a separate file per entity.
  - The application must parse the files each time they want to read/update records.

Artist(name, year, origin)

```
"The Chainsmokers", 2012, US
"Imagine Dragons", 2008, US
"Coldplay", 1997, UK
```

Album(name, artist, year)

```
"Memories...Do Not Open", "The
Chainsmokers", 2017
"Viva la Vida", ”Coldplay", 2008
"Evolve", "Imagine Dragons", 2017
```

Flat File Strawman

- Example: get the year that Imagine Dragons become active.

Artist(name, year, origin)

```
"The Chainsmokers", 2012, US
"Imagine Dragons", 2008, US
"Coldplay", 1997, UK
```

```
for line in file.readlines():
    record = parse(line)
    if record[0] == "Imagine Dragons":
        print(int(record[1]))
```

Flat Files: Data Integrity

- How do we ensure that the artist is the same for each album entry?
- What if somebody overwrites the album year with an invalid string?
- What if there are multiple artists on an album?
- What happens if we delete an artist that has albums?

Flat Files: Implementation

- How do you find a particular record?
- What if we now want to create a new application that uses the same database? What if that application is running on a different machine?
- What if two threads try to write to the same file at the same time?

Flat Files: Durability

- What if the machine crashes while our program is updating a record?

- What if we want to replicate the database on multiple machines for high availability?

DMBS

- A database management system (**DBMS**) is software that allows applications to store and analyze information in a database.

- A general-purpose DBMS supports the definition, creation, querying, update, and administration of databases in accordance with some **data model**.

Data Models

- A **data model** is a collection of concepts for describing the data in a database.

- A **schema** is a description of a particular collection of data, using a given data model.

Preview of the relational model

```
CREATE TABLE Artist(name VARCHAR, year DATE, country CHAR(60));
CREATE TABLE Album(Albumid INTEGER, name VARCHAR, year DATE);
...
```

Data Models

- Relational ← Most DBMSs
- Key/Value
- Graph
- Document / XML / Object
- Wide-Column / Column-family

  ← NoSQL
- Array / Matrix / Vectors ← Machine learning
- Hierarchical
- Network
- Multi-Value

  ← Obsolete/Legacy/Rare

Early DBMSs

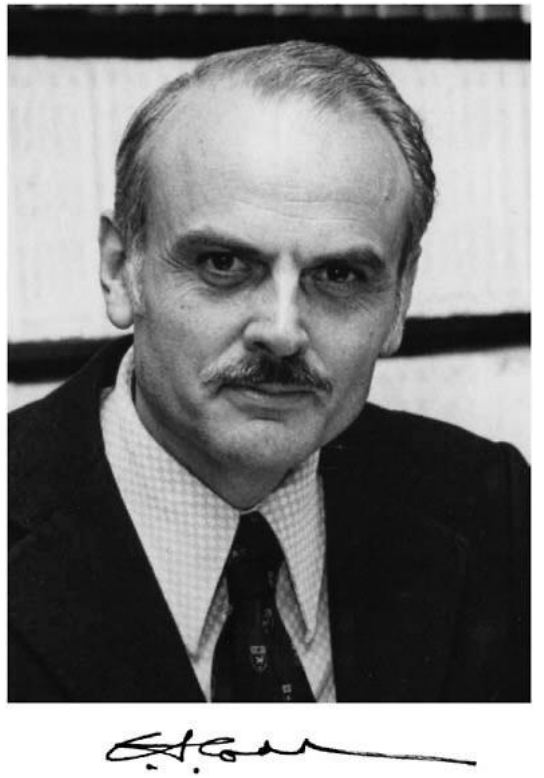
- Early database applications were difficult to build and maintain on available DBMSs in the 1960s.
  - Examples: CODASYL
  - Computers were expensive, humans were cheap.

- Tight coupling between logical and physical layers.

- Programmers had to (roughly) know what queries the application would execute before they could deploy the database.

Birth of Relational Model

- Ted Codd was a mathematician at IBM Research in the late 1960s.
- Codd saw IBM's developers rewriting database programs every time the database's schema or layout changed.
- Devised the relational model in 1969.



Debate

- The great debate: "The Differences and Similarities Between the Data Base Set and Relational Views of Data."

COBOL/CODASYL camp:

1. The relational model is too mathematical. No mere mortal programmer will be able to understand your newfangled languages.
2. Even if you can get programmers to learn your new languages, you won't be able to build an efficient implementation of them.
3. On-line transaction processing applications want to do record-oriented operations.

Relational camp:

1. Nothing as complicated as the DBTG proposal can possibly be the right way to do data management.
2. Any set-oriented query is too hard to program using the DBTG data manipulation language.
3. The CODASYL model has no formal underpinning with which to define the semantics of the complex operations in the model.

ACM SIGFIDET Workshop on Data Description, Access, and Control in Ann Arbor, Michigan, held 1-3 May 1974

Relational Model

- The relational model defines a database abstraction based on **relations** to avoid maintenance overhead.

Key tenets (rules):

- Store database in simple data structures (relations).
- Physical storage left up to the DBMS implementation.
- Access data through high-level language, DBMS figures out best execution strategy.

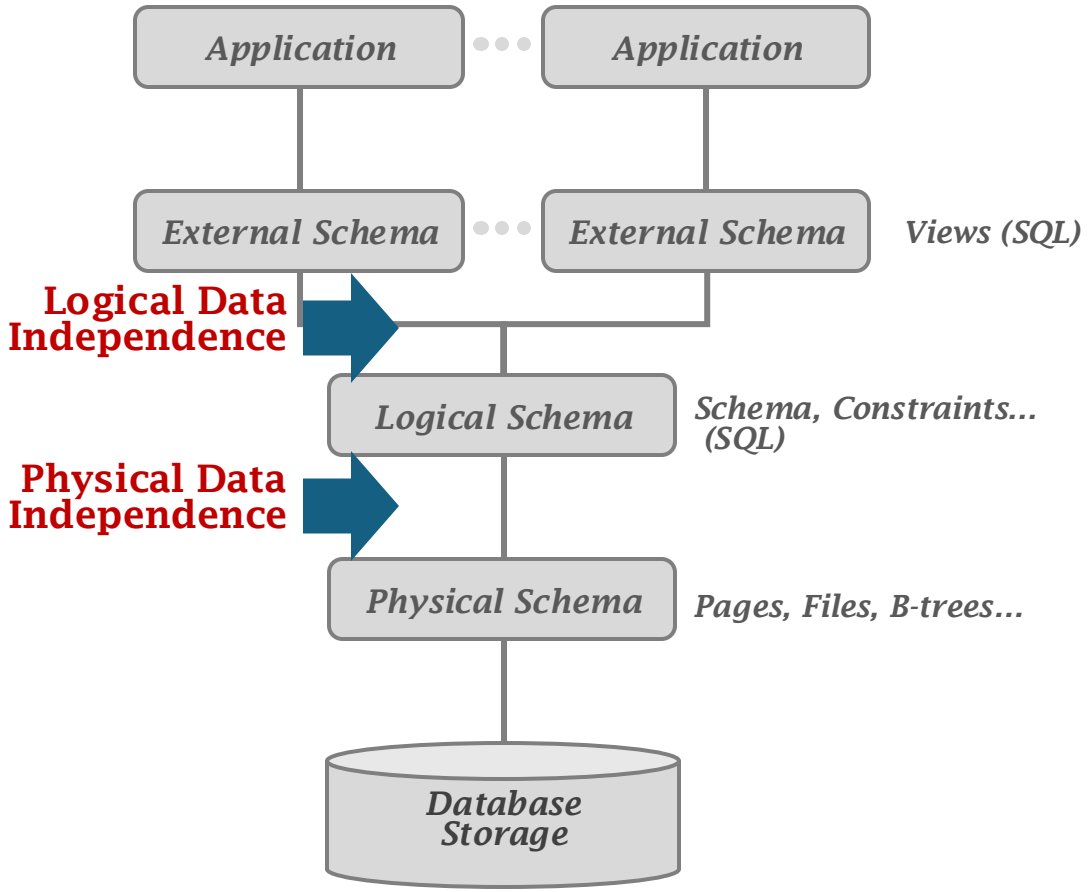
Decouple *what to do* and *how to do*

Relational Model

- **Structure:** The definition of the database's relations and their contents independent of their physical representation.
- **Integrity:** Ensure the database's contents satisfy constraints.
- **Manipulation:** Programming interface for accessing and modifying a database's contents.

Key Concept: Data Independence (DI)

- Isolate the user/application from low level data representation.
  - The user only worries about the application logic.
  - Database can optimize the layout (and re-optimize as the workload changes).



Relational Model

- A relation is an unordered set that contain the relationship of attributes that represent entities.

- A tuple is a set of attribute values (also known as its domain) in the relation.
  - Values are (normally) atomic/scalar.
  - The special value **NULL** is a member of every domain (if allowed).

Artist(name, year, origin)

name	year	origin
The Chainsmokers	2012	US
Imagine Dragon	2008	US
Coldplay	1997	UK

n-ary Relation
=
Table with n columns

Relational Model: Primary Keys

- A relation's primary key uniquely identifies a single tuple.
- Some DBMSs automatically create an internal primary key if a table does not define one.

- DBMS can auto-generation unique primary keys via an identity column:
  - `IDENTITY` (SQL Standard)
  - `SEQUENCE` (PostgreSQL / Oracle)
  - `AUTO_INCREMENT` (MySQL)

Artist(id, name, year, origin)

id	name	year	origin
1	The Chainsmokers	2012	US
2	Imagine Dragon	2008	US
3	Coldplay	1997	UK

Relational Model: Foreign Keys

- A foreign key specifies that an attribute from one relation maps to a tuple in another relation.

Relational Model: Foreign Keys

Artist(<u>id</u>, name, year, origin)

id	name	year	origin
1	The Chainsmokers	2012	US
2	Imagine Dragon	2008	US
3	Coldplay	1997	UK

Album(<u>id</u>, name, artist, year)

id	name	artist	year
11	Memories...Do Not Open	???	2017
22	Viva la Vida	3	2008
33	Evolve	2	2017

Relational Model: Foreign Keys

Artist(id, name, year, origin)

id	name	year	origin
1	The Chainsmokers	2012	US
2	Imagine Dragon	2008	US
3	Coldplay	1997	UK

ArtistAlbum(artist_id, album_id)

artist_id	album_id
1	11
3	11
3	22
2	33

Album(id, name, year)

id	name	year
11	Memories...Do Not Open	2017
22	Viva la Vida	2008
33	Evolve	2017

Relational Model: Constraints

- User-defined conditions that must hold for **any** instance of the database.
  - Can validate data within a single tuple or across entire relation(s).
  - DBMS prevents modifications that violate any constraint.

- Unique key and referential (fkey) constraints are the most common.

Artist(id, name, year, origin)

id	name	year	origin
1	The Chainsmokers	2012	US
2	Imagine Dragon	2008	US
3	Coldplay	1997	UK

```
CREATE TABLE Artist(
name VARCHAR NOT NULL,
year DATE,
country CHAR(60),
CHECK (date_trunc('year', year) = year));
```

Data Manipulation Language (DML)

- Methods to store and retrieve information from a database.

Procedural:
The query specifies the (high-level) strategy to find the desired result based on sets / bags.

← Relational Algebra

Non-Procedural (Declarative):
The query specifies only what data is wanted and not how to find it.

← Relational Calculus

Relational Algebra

- Fundamental operations to retrieve and manipulate tuples in a relation.
  - Based on set algebra (unordered lists with no duplicates).
- Each operator takes one or more relations as its inputs and outputs a new relation.
  - We can "chain" operators together to create more complex operations.

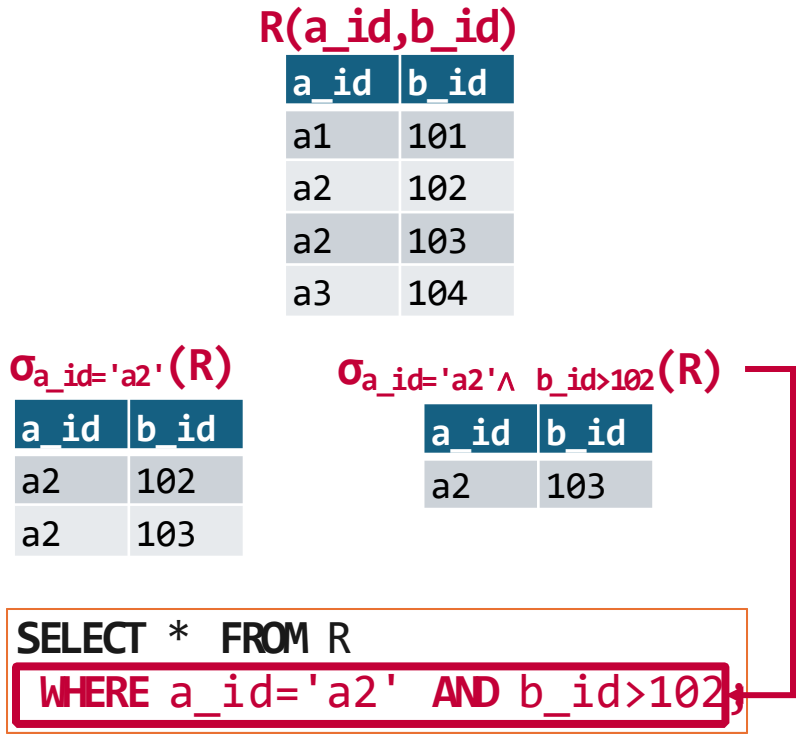
σ	Select
π	Projection
∪	Union
∩	Intersection
–	Difference
×	Product
⋈	Join

Relational Algebra: Select

Choose a subset of the tuples from a relation that satisfies a selection predicate.

- Predicate acts as a filter to retain only tuples that fulfill its qualifying requirement.
- Can combine multiple predicates using conjunctions / disjunctions.

Syntax: $\sigma_{predicate}(R)$



Relational Algebra: Projection

Generate a relation with tuples that contains only the specified attributes.

- Rearrange attributes' ordering.
- Remove unwanted attributes.
- Manipulate values to create derived attributes.

Syntax: $\Pi_{A1,A2,\ldots,An}(R)$

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a2	103
a3	104

$\Pi_{b\_id-100,a\_id}(\sigma_{a\_id='a2'}(R))$

b_id-100	a_id
2	a2
3	a2

```
SELECT b_id-100, a_id
  FROM R WHERE a_id = 'a2';
```

Relational Algebra: Union

Generate a relation that contains all tuples that appear in either only one or both input relations.

Syntax: (R ∪ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

```
(SELECT * FROM R)
UNION
(SELECT * FROM S);
```

(R ∪ S)

a_id	b_id
a1	101
a2	102
a3	103
a4	104
a5	105

Relational Algebra: Intersection

Generate a relation that contains only the tuples that appear in both of the input relations.

Syntax: (R ∩ S)

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R ∩ S)

a_id	b_id
a3	103

```
(SELECT * FROM R)
INTERSECT
(SELECT * FROM S);
```

Relational Algebra: Difference

Generate a relation that contains only the tuples that appear in the first and not the second of the input relations.

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

Syntax: (R - S)

```
(SELECT * FROM R)
EXCEPT
(SELECT * FROM S);
```

(R – S)

a_id	b_id
a1	101
a2	102

Relational Algebra: Product

Generate a relation that contains all possible combinations of tuples from the input relations.

Syntax: (R × S)

```
SELECT * FROM R CROSS JOIN S;
```

```
SELECT * FROM R, S;
```

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id)

a_id	b_id
a3	103
a4	104
a5	105

(R × S)

R.a_id	R.b_id	S.a_id	S.b_id
a1	101	a3	103
a1	101	a4	104
a1	101	a5	105
a2	102	a3	103
a2	102	a4	104
a2	102	a5	105
a3	103	a3	103
a3	103	a4	104
a3	103	a5	105

Relational Algebra: Join

Generate a relation that contains all tuples that are a combination of two tuples (one from each input relation) with a common value(s) for one or more attributes.

R(a_id,b_id)

a_id	b_id
a1	101
a2	102
a3	103

S(a_id,b_id,val)

a_id	b_id	val
a3	103	XXX
a4	104	YYY
a5	105	ZZZ

(R ⋈ S)

R.a_id	R.b_id	S.a_id	S.b_id	S.val
a3	103	a3	103	XXX

a_id	b_id	val
a3	103	XXX

Syntax: (R ⋈ S)

```
SELECT * FROM R NATURAL JOIN S;
```

```
SELECT * FROM R JOIN S USING (a_id, b_id);
```

```
SELECT * FROM R JOIN S
    ON R.a_id = S.a_id AND R.b_id = S.b_id;
```

Relational Algebra: Extra Operators

- Rename (ρ)
- Assignment ($R \leftarrow S$)
- Duplicate Elimination (δ)
- Aggregation (γ)
- Sorting (τ)
- Division ($R \div S$)

Observation

- Relational algebra defines an ordering of the high-level steps of how to compute a query.
- Example: $\sigma_{b\_id=102}(R \bowtie S)$ vs. $(R \bowtie (\sigma_{b\_id=102}(S))$

- A better approach is to state the high-level answer that you want the DBMS to compute.
- Example: Retrieve the joined tuples from `R` and `S` where `b_id` equals `102`.

Relational Model: Queries

- The relational model is independent of any query language implementation.
- **SQL** is the *de facto* standard (many dialects).

```
for line in file.readlines():
  record = parse(line)
  if record[0] == "Imagine Dragons":
    print(int(record[1]))
```

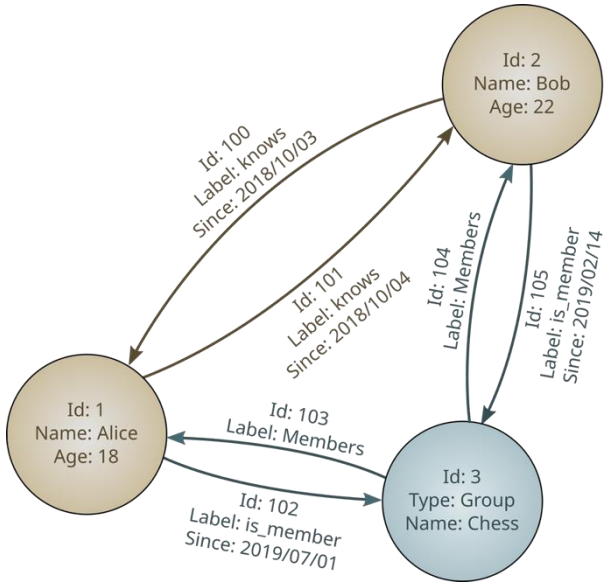
```
SELECT year FROM artists
 WHERE name = 'Imagine Dragons';
```

Data models

- Relational
- Key/Value
- Graph ← Hot in research communtiy
- Document / XML / Object ← Leading Alternative
- Wide-Column / Column-family
- Array / Matrix / Vectors ← Hot recently
- Hierarchical
- Network
- Multi-Value

Graph Data Model

- A graph database is a systematic collection of data that emphasizes the relationships between the different data entities.
- A graph contains a collection of nodes and edges.
- Each node has properties or attributes that describe it. In some cases, edges have properties as well. Graphs with properties are also called property graphs.



neo4j

ArangoDB

aws Amazon Neptune

OrientDB

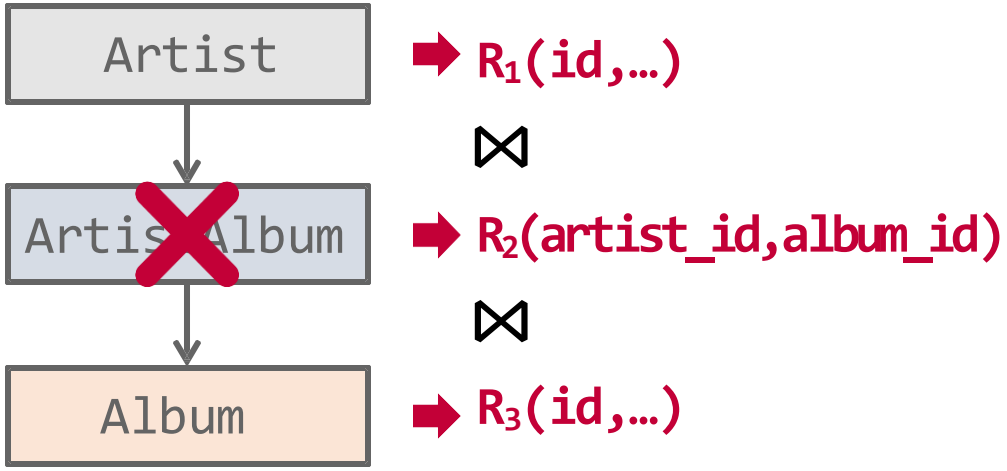
TigerGraph

LDBC

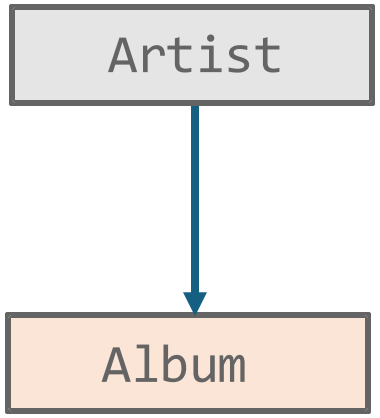
Document Data Model

- A collection of record documents containing a hierarchy of named field/value pairs.
  - A field's value can be either a scalar type, an array of values, or another document.
  - Modern implementations use JSON. Older systems use XML or custom object representations.
- Avoid "relational-object impedance mismatch" by tightly coupling objects and database.

Document Data Model



Document Data Model



Application Code

```
class Artist {
   int id;
   String
   name; int
   year;
   Album albums[];
}
class Album {
  int id;
  String
  name;
  int year;
}
```

```
{
  "name": "Imagine Dragons",
  "year": 2008,
  "albums": [
    {
      "name": "Evolve",
      "year": 2017
    },
    {
      "name": "Beneath the Surface",
      "year": 1999
    }
  ]
}
```

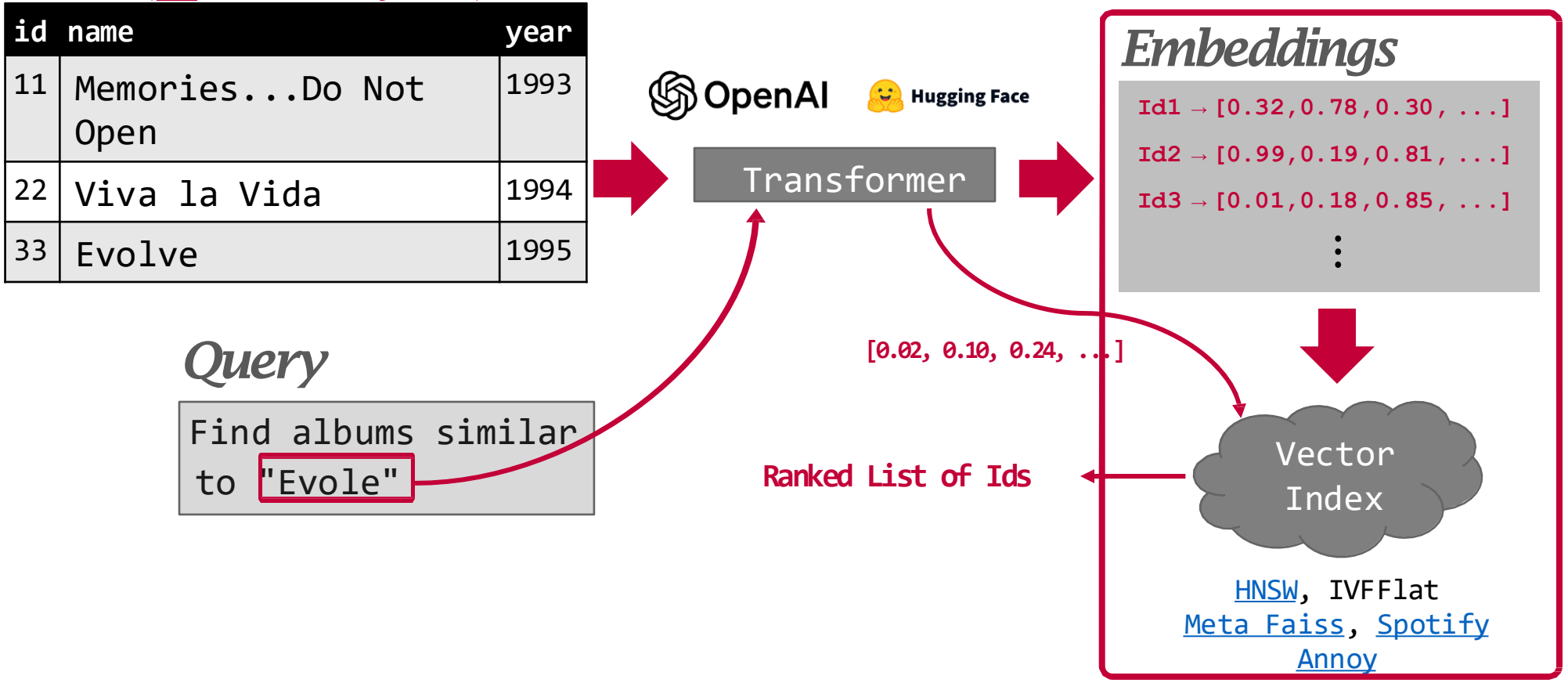
Vector Data Model

- One-dimensional arrays used for nearest-neighbor search (exact or approximate).
  - Used for semantic search on embeddings generated by ML-trained transformer models (think ChatGPT).
  - Native integration with modern ML tools and APIs (e.g., LangChain, OpenAI).
- At their core, these systems use specialized indexes to perform NN searches quickly.

Vector Data Model

Album(id, name, year)

id	name	year
11	Memories...Do Not Open	1993
22	Viva la Vida	1994
33	Evolve	1995



Conclusion

- Databases are ubiquitous.
- Relational algebra defines the primitives for processing queries on a relational database.
- We will see relational algebra again when we talk about query optimization + execution.