Module 3

NoSQL Databases

María del Mar Roldán – University of Málaga
Ismael Navas – University of Málaga

Learning Objectives

- Understand the Cassandra data model
- Introduce cqlsh
- Understand and use the DDL subset of CQL
- Introduce DevCenter
- Understand and use the DML subset of CQL
- Understand basics of data modeling

What is data modeling?

- Data modeling is a process that involves
  - Collection and analysis of data requirements in an information system
  - Identification of participating entities and relationships among them
  - Identification of data access patterns
  - A particular way of organizing and structuring data
  - Design and specification of a database schema
  - Schema optimization and data indexing techniques
- Data modeling = Science + Art

Steps for data modeling in Cassandra

1. Understand data and application queries
   - Data may or may not exist in some format (RDBMS, XML, CSV, …)
   - Queries can be organized into a query graph
2. Design column families
   - Design is based on access patterns or queries over data
3. Implement the design using CQL
   - Optimizations concerning data types, keys, partition sizes, ordering

Key steps for data modeling

The products of the data modeling steps are documented as

- Conceptual data model
  - Technology-independent, unified view of data
  - Entity-relationship model, dimensional model, etc.
- Logical data model
  - Unique for Cassandra
  - Column family diagrams
- Physical data model
  - Unique for Cassandra
  - CQL definitions

Apache Cassandra

Is relational database design similar to Cassandra database design?

DATASTAX

No!

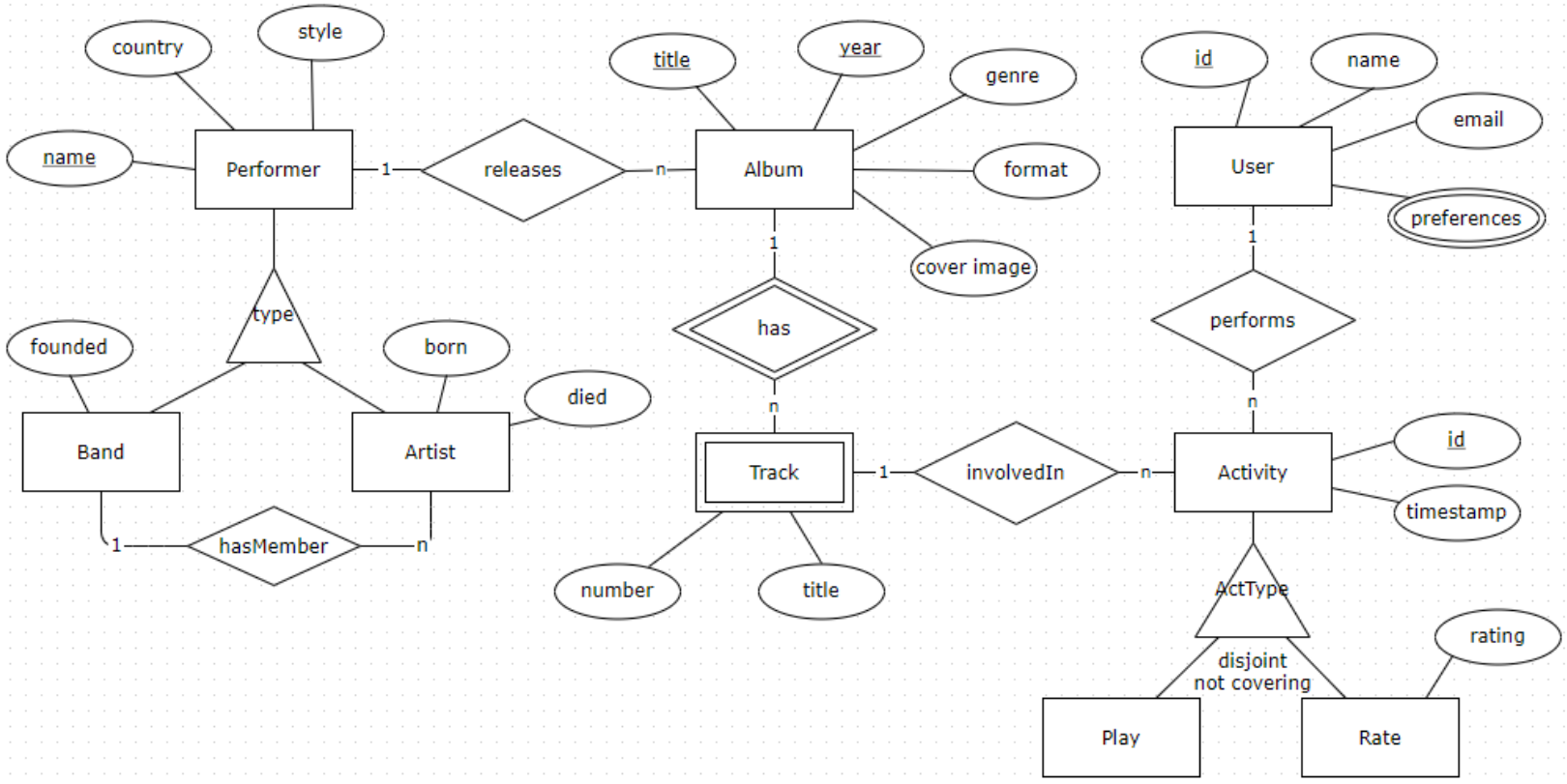
Cassandra	Relational

• Multi-dimensional column family
  • Equally good for simple and complex data
	• Two-dimensional relation
	  • Suited for simple data
	  • Complex data requires many relations and "star" schemas

• All data required to answer a query must be nested in a column family
  • Referential integrity is a non-issue
	• Data from many relations is combined to answer a query
	  • Referential integrity is important

• Data modeling methodology is driven by queries and data
  • Data duplication is considered normal (side effect of data nesting)
	• Data modeling is driven by data only
	  • Data duplication is considered a problem (normalization theory)

Example: Modeling musicdb Database



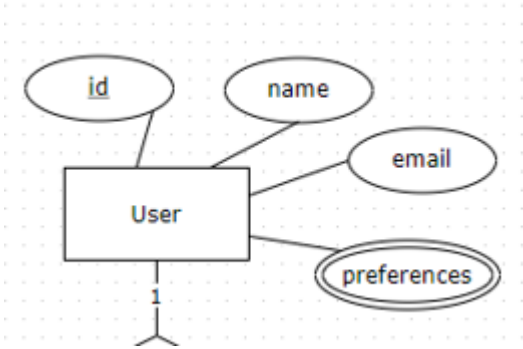
Example: musicdb Queries

- Q1: find information for a specified user.
- Q2: find performers for a specified style; order by performer (ASC).
- Q3: find information for a specified performer (artist or band)
- Q4: find information (performer, genre, tracks title) for a specified album (given its title and year)
- Q5: find albums (year, title and genre) for a specified performer; order by album release year (DESC) and title (ASC)
- Q6: find albums and performers for a specified genre; order by performer (ASC), year (DESC), and title (ASC).
- Q7: find activities for a specified user; order by activity time (DESC).
- Q8: find statistics for a specified track
- Q9: find user activities for a specified activity type

Example: musicdb Access Patterns

- Q1: find information for a specified user.

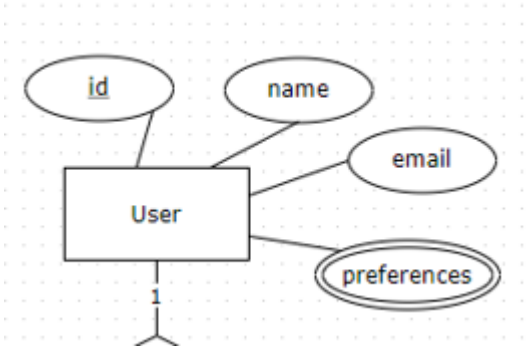
GIVEN	FIND
USER.ID	USER.NAME
	USER.EMAIL
	USER.PREFERENCES



Example: musicdb Logical Model

- Q1: find information for a specified user.

GIVEN	FIND
USER.ID	USER.NAME
	USER.EMAIL
	USER.PREFERENCES



USERS_BY_ID

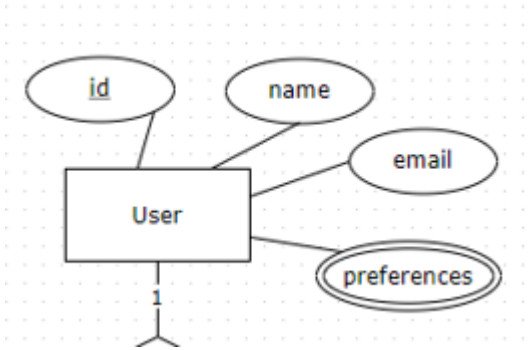
USER_ID NUMBER
USER_NAME TEXT
EMAIL TEXT
PREFERENCES SET<TEXT>

Example: musicdb Physical Model

- Q1: find information for a specified user.

USERS_BY_ID

ID NUMBER
NAME TEXT
EMAIL TEXT
PREFERENCES SET<TEXT>



USERS_BY_ID

USER_ID UUID
USER_NAME VARCHAR
EMAIL VARCHAR
PREFERENCES SET<VARCHAR>
PRIMARY KEY(USER_ID)

SELECT * FROM USERS_BY_ID WHERE USER_ID = ?

Example: musicdb CQL script

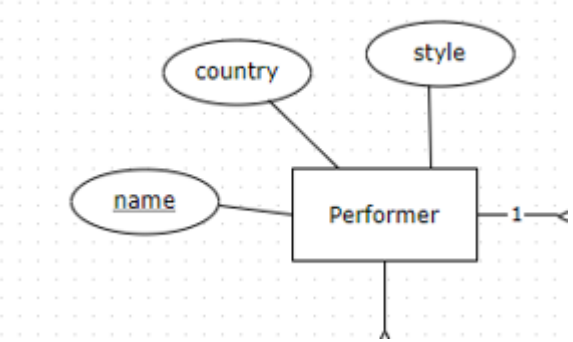
- Q1: find information for a specified user.

```
CREATE TABLE users_by_id (
        user_id UUID,
        user_name VARCHAR,
        email VARCHAR,
        preferences SET<VARCHAR>,
        PRIMARY KEY (user_id)
);
```

Example: musicdb Access Patterns

- Q2: find performers for a specified style; order by performer (ASC).

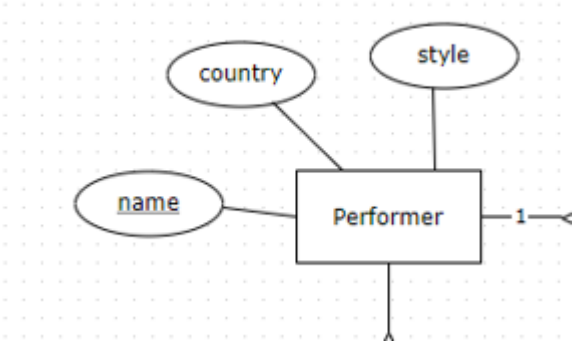
GIVEN	FIND
PERFORMER.STYLE	NAME (ASC)



Example: musicdb Logical Model

- Q2: find performers for a specified style; order by performer (ASC).

GIVEN	FIND
PERFORMER.STYLE	PERFORMER.NAME (ASC)



PERFORMER_BY_STYLE

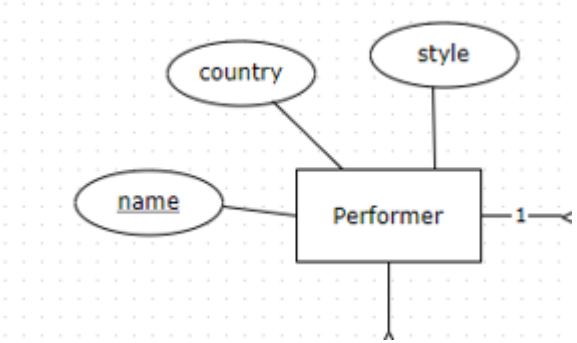
STYLE TEXT
NAME TEXT

Example: musicdb Physical Model

- Q2: find performers for a specified style; order by performer (ASC).

PERFORMER_BY_STYLE

STYLE TEXT NAME TEXT



PERFORMER_BY_STYLE

STYLE VARCHAR NAME VARCHAR PRIMARY KEY (STYLE, NAME)

Partition key

Clustering column

SELECT NAME FROM PERFORMER_BY_STYLE WHERE STYLE = ?

Example: musicdb CQL script

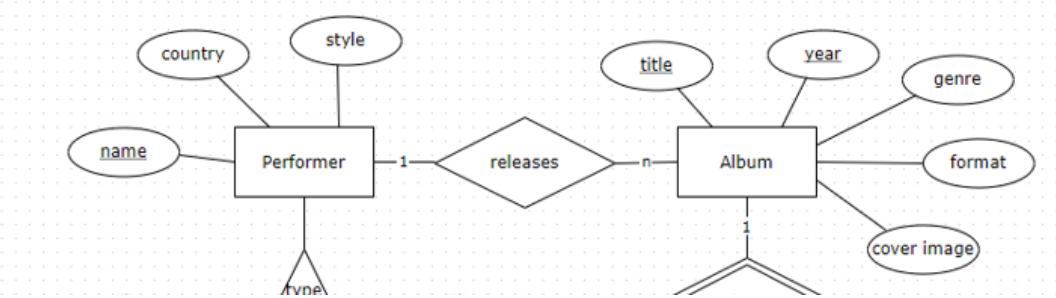
- Q2: find performers for a specified style; order by performer (ASC).

```
CREATE TABLE performer_by_style(
        style VARCHAR,
        name VARCHAR,
        PRIMARY KEY (style, name)
);
```

Example: musicdb Access Patterns

- Q6: find albums and performers for a specified genre; order by performer (ASC), year (DESC), and title (ASC).

GIVEN	FIND
ALBUM.GENRE	PERFORMER.NAME (ASC)
	ALBUM.YEAR (DESC)
	ALBUM.TITLE (ASC)



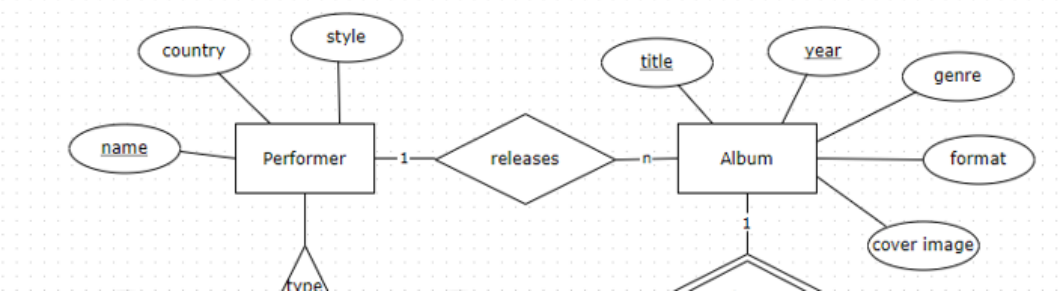
Example: musicdb Logical Model

- Q6: find albums and performers for a specified genre; order by performer (ASC), year (DESC), and title (ASC).

GIVEN	FIND
ALBUM.GENRE	PERFORMER.NAME (ASC)
	ALBUM.YEAR (DESC)
	ALBUM.TITLE (ASC)

ALBUMS_BY_GENRE

GENRE TEXT
PERFORMER TEXT
YEAR INT
TITLE TEXT

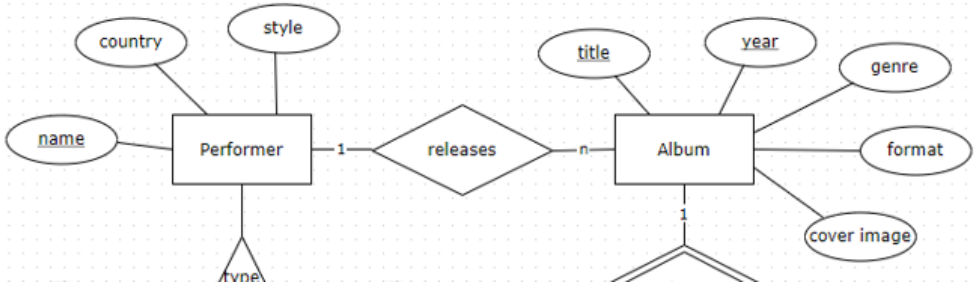


Example: musicdb Physical Model

- Q6: find albums and performers for a specified genre; order by performer (ASC), year (DESC), and title (ASC).

ALBUMS_BY_GENRE

GENRE TEXT
PERFORMER TEXT
YEAR INT
TITLE TEXT



ALBUMS_BY_GENRE

GENRE VARCHAR
PERFORMER VARCHAR
YEAR INT
TITLE VARCHAR
PRIMARY KEY(GENRE, PERFORMER, YEAR, TITLE)

Partition key

Clustering columns

SELECT PERFORMER, YEAR, TITLE FROM ALBUMS_BY_GENRE WHERE GENRE = ?

Example: musicdb CQL script

- Q6: find albums and performers for a specified genre; order by performer (ASC), year (DESC), and title (ASC).

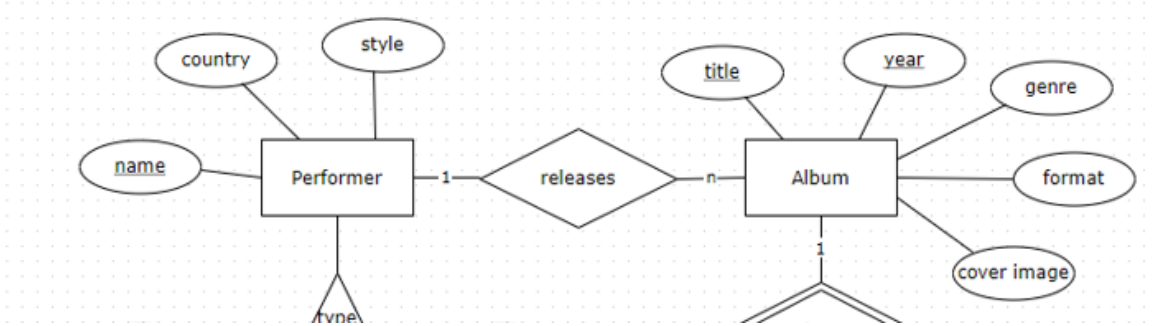
```
CREATE TABLE albums_by_genre(
        genre VARCHAR,
        performer VARCHAR,
        year INT,
        title VARCHAR,
        PRIMARY KEY (genre, performer, year, title)
)  WITH CLUSTERING ORDER BY (performer ASC, year DESC, title ASC);
```

Time for Exercises

Example: musicdb Access Patterns

- Q5: find albums (year, title and genre) for a specified performer; order by album release year (DESC) and title (ASC)

GIVEN	FIND
Performer.name	Album.genre
	Album.year
	Album.title



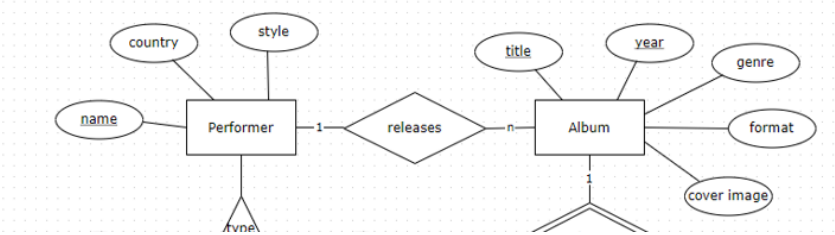
Example: musicdb Access Patterns

- Q5: find albums (year, title and genre) for a specified performer; order by album release year (DESC) and title (ASC)

GIVEN	FIND
Performer.name	Album.genre
	Album.year
	Album.title

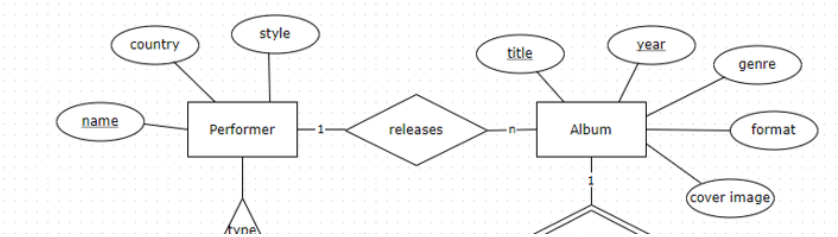
ALBUMS_BY_PERFORMER

Perfomer text
Genre text
Year number
Title text



Example: musicdb Access Patterns

- Q5: find albums (year, title and genre) for a specified performer; order by album release year (DESC) and title (ASC)



ALBUMS_BY_PERFORMER

Perfomer text
Genre text
Year number
Title text

ALBUMS_BY_PERFORMER

perfomer varchar
genre varchar
year int
title varchar
Primary key(performer, year, title)

Example: musicdb Access Patterns

- Q7: find activities for a specified user; order by activity time (DESC).

GIVEN	FIND
User.id	Track.number
	Track. title
	Album.title
	Album.year
	Rate.rating
	Activity.timestamp

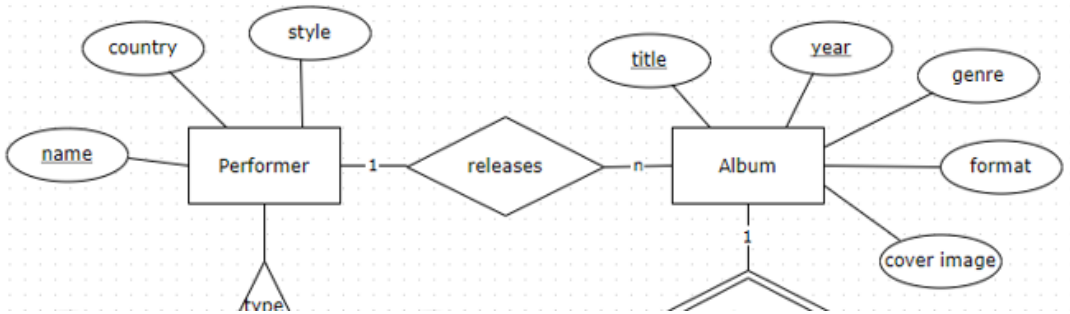
ACTIVITIES_BY_USER

User UUID
Track_number int
Album_title varchar
album_year int
Rating int
Activity TIMEUUID
PRIMARY KEY(user, activity)

Example: musicdb Access Patterns

- Q5: find albums (year, title and genre) by each year; order by album title (ASC) and performer (ASC)

GIVEN	FIND

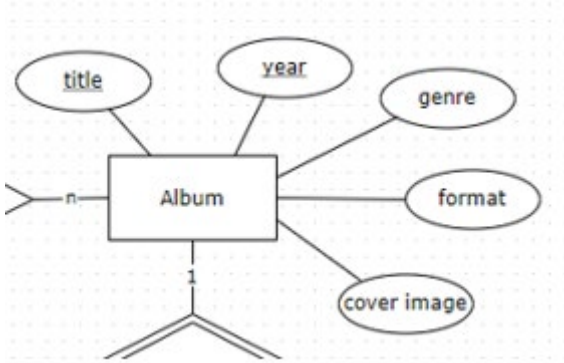


Example: musicdb Access Patterns

- Q5: find albums (year, title and genre) by each year; order by album title (ASC) and performer (ASC)

GIVEN	FIND
	Album.title
	Album.year
	Album.genre

EMPTY GIVEN!!!!



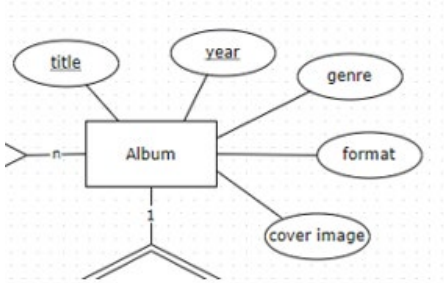
Example: musicdb Access Patterns

- Q5: find albums (year, title and genre) by each year; order by album title (ASC) and performer (ASC)

GIVEN	FIND
	Album.year
	Album.title
	Album.genre

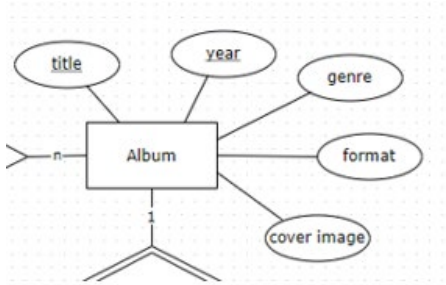
ALBUMS_PER_YEAR

year number title text genre text



Example: musicdb Access Patterns

- Q5: find albums (year, title and genre) by each year; order by album title (ASC) and performer (ASC)



ALBUMS_PER_YEAR

year int title varchar genre varchar Primary key(year, title, performer)

ALBUMS_PER_YEAR

year number title text genre text

SELECT * FROM ALBUMS_PER_YEAR;

SELECT * FROM ALBUMS_PER_YEAR where title < … allow filtering;

SELECT * FROM ALBUMS_PER_YEAR where title = … and genre = … allow filtering;