Database System Concepts

Qiang Yin

Spring 2024

Introduction

Instructor

- Qiang Yin (尹 强)
- Homepage: `https://cs.sjtu.edu.cn/~qyin`
- Email: `q.yin@sjtu.edu.cn`
- Office: Room 3-501, Dianxin Building
- Research interests: database theory and system, formal methods
- Short Bio: PhD. (STJU, 2016), Postdoc (BUAA, 2016 – 2018), Senior engineer (Damo Academy, 2018 – 2021), Assistant Professor (SJTU, 2021 – now)

Our TAs

Figure: **冯书瀚**

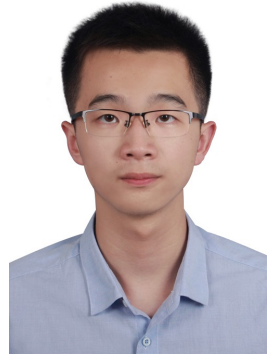


Figure: **把徐进**

Office hour

- 7:00 – 9:00 PM, every Tuesday, Room 3-325, Dianxin Building

References

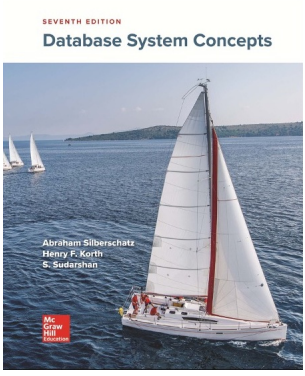


Figure: Database System Concepts 7th Ed. Silberschatz, Korth and Sudarshan

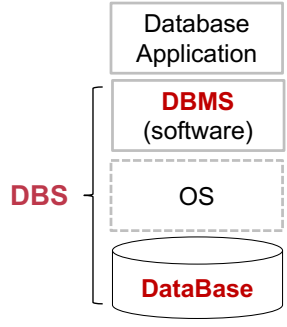
Figure: 数据库系统概论，王珊、萨师煊

Database applications

Figure: Database applications are everywhere

Database management system

- Database: an organized collection of inter-related data that models some aspect of the real-world.
- Database management system (DBMS): a software system that facilitates the creation, maintenance and uses of databases.
- Database system: DBMS + Database



Motivation: a music store application

Consider an application that models a digital music store to keep track of artists and albums.

Data we need to store includes:

- Information about Artists
- What Albums those Artists released

Flat file example (I)

Artist	Year	City
Mozart	1756	Salzburg
Beethoven	1770	Bonn
Chopin	1810	Warsaw

Figure: Artists.csv

Album	Artist	Year
The Marriage of Figaro	Mozart	1786
Requiem Mass In D minor	Mozart	1791
Für Elise	Beethoven	1867

Figure: Albumes.csv

- Store our database as comma-separated value files that we manage in our own code.
- Use a separate file per entity, e.g., Artists.csv and Albums.csv.
- The application has to parse the files each time they want to read/update records.

Flat file example (II)

Artist	Year	City
Mozart	1756	Salzburg
Beethoven	1770	Bonn
Chopin	1810	Warsaw

Figure: Artists.csv

Album	Artist	Year
The Marriage of Figaro	Mozart	1786
Requiem Mass In D minor	Mozart	1791
Für Elise	Beethoven	1867

Figure: Albums.csv

Example. To get the Albums composed by Beethoven:

```
for line in file:
    record = parse(line)
    if "Beethoven" == record[1]:
        print record[0]
```

Flat file example: data integrity

Artist	Year	City
Mozart	1756	Salzburg
Beethoven	1770	Bonn
Chopin	1810	Warsaw

Figure: Artists.csv

Album	Artist	Year
The Marriage of Figaro	Mozart	1786
Requiem Mass In D minor	Mozart	1791
Für Elise	Beethoven	1867

Figure: Albums.csv

- 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?
- How do we store that there are multiple artists on an album?

Flat file example: implementation

Artist	Year	City
Mozart	1756	Salzburg
Beethoven	1770	Bonn
Chopin	1810	Warsaw

Figure: Artists.csv

Album	Artist	Year
The Marriage of Figaro	Mozart	1786
Requiem Mass In D minor	Mozart	1791
Für Elise	Beethoven	1867

Figure: Albums.csv

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

Flat file example: durability

Artist	Year	City
Mozart	1756	Salzburg
Beethoven	1770	Bonn
Chopin	1810	Warsaw

Figure: Artists.csv

Album	Artist	Year
The Marriage of Figaro	Mozart	1786
Requiem Mass In D minor	Mozart	1791
Für Elise	Beethoven	1867

Figure: Albums.csv

- 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?

DBMS in early days

- Database applications were difficult to build and maintain.
- Tight coupling between logical and physical layers.
- You must (roughly) know what queries your application would execute before you deployed the database.

The relational revolution

Edgar F. Codd proposed the relational data model in 1970.

1. Store database in simple data structures
2. Access data through high-level language
3. Physical storage left up to implementation

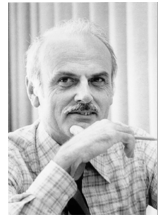


Figure: Edgar Frank Codd

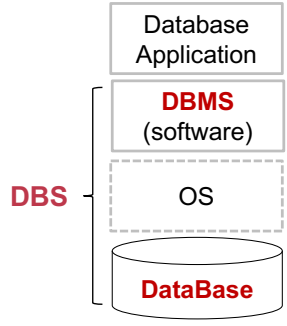
$\Rightarrow$ Provides physical data independence.

https://en.wikipedia.org/wiki/Edgar_F._Codd

DBMS features

A DBMS is software system that facilitates the creation, maintenance and uses of databases.

- Persistent storage of database
- Data abstraction: logical data model, declarative query language, integrity constraints.
- Efficient query processing.
- Concurrency control for high throughput transactions.
- Resilient to system failures.
- ...



Course overview

Relational databases

- Relational data model
- Relational algebra
- Structured query language (SQL)
- Relational database design theory

DBMS internals

- Database storage
- Indexing
- Query processing and optimization
- Concurrency control
- Crash recovery

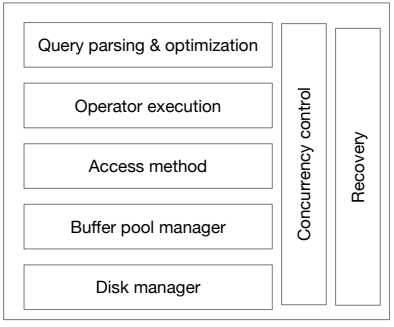


Figure: Classical DBMS architecture

Other topics (TBD): (i) graph database, (ii) parallel query processing

Course workloads & grading policy

- 5 assignments (30%): Relational model and SQL (hw1), database design theory (hw2), query processing (hw3), query optimization (hw4), transaction processing (hw5).
- Final exam (70%): closed-book, you are allowed to bring a cheat sheet of A4 size.

Other resources

Course projects

- If you want to get your hands dirty, see CMU 15-445.
- `https://15445.courses.cs.cmu.edu/fall2023/assignments.html`

Reading group

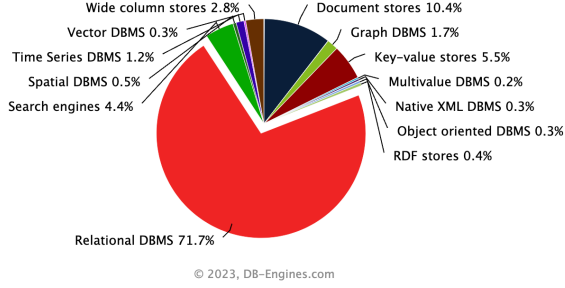
- 6:00 - 8:00 PM, every Wednesday, Room 3-320, Dianxin Building.
- Feel free to join us if you are interested.
- `https://cs.sjtu.edu.cn/~qyin/seminar`

Relational data model

Data models

A data model is a collection of concepts/tools for describing the data in a database.

- Relational (focus of this course)
- Key-value
- Graph
- Column-family
- Document
- Vector



Relational model is still the dominating technology today.

https://db-engines.com/en/ranking_categories

Relational data model

- A database is a collection of relations and each relation is an unordered set of tuples (or rows).
- Each relation has a set of attributes (or columns).
- Each attribute has a name and a domain and each tuple has a value for each attribute of the relation.
- Values are atomic/scalar.

ID	Artist	Year	City
1	Mozart	1756	Salzburg
2	Beethoven	1770	Bonn
3	Chopin	1810	Warsaw

Table: Artists(ID, Artist, Year, City)

Schema vs. instance

- Let A_1, ..., A_n be attributes.
- $R(A_1, A_2, \ldots, A_n)$ is a relational schema.
- A schema specifies the logical structure of data.
- Relational instance: concrete table content w.r.t. a given schema.
  – set of tuples (also called records) matching the scheme.
- A schema rarely changes after being defined, while an instance often changes rapidly.

Schema vs. instance

Example. A relational schema: Artists(ID, Artist, Year, City).

Below is an instance for the schema Artists(ID, Artist, Year, City).

ID	Artist	Year	City
1	Mozart	1756	Salzburg
2	Beethoven	1770	Bonn
3	Chopin	1810	Warsaw

Table: Artists(ID, Artist, Year, City)

Schema vs. database instance

Database scheme

- Artists (ID, Artist, Year, City)
- Albums (ID, Album, Artist_ID, Year)
- ArtistAlbum (Artist_ID, Album_ID)

Database instance

ID	Album	Artist_ID	Year
1	The Marriage of Figaro	1	1786
2	Requiem Mass In D minor	1	1791
3	Für Elise	2	1867

Table: Albums(ID, Album, Artist_ID, Year)

ID	Artist	Year	City
1	Mozart	1756	Salzburg
2	Beethoven	1770	Bonn
3	Chopin	1810	Warsaw

Table: Artists(ID, Artist, Year, City)

Artist_ID	Album_ID
1	1
1	2
2	3

Table: ArtistAlbum(Artist_ID, Album_ID)

Keys

$K \subseteq \{A_1, A_2, \ldots, A_n\}$ is a superkey of schema $R(A_1, \ldots, A_n)$ if values for K are sufficient to identify a unique tuple for each possible relation instance of $R(A_1, A_2, \ldots, A_n)$.

A superkey K is a candidate key if K is minimal.

ID	Artist	Year	City
1	Mozart	1756	Salzburg
2	Beethoven	1770	Bonn
3	Chopin	1810	Warsaw

Table: Artists(ID, Artist, Year, City)

Primary key

A primary key is a designated candidate key of a relation.

Some DBMSs automatically create an internal primary key if we don't define one.

ID	Artist	Year	City
1	Mozart	1756	Salzburg
2	Beethoven	1770	Bonn
3	Chopin	1810	Warsaw

Table: Artists(ID, Artist, Year, City)

```
create table Artists(
        ID, varchar(8),
        Artist, varchar(20) not null,
        Year, numeric(4,0),
        City, varchar(20),
        primary key (ID)
)
```

Foreign key

A foreign key specifies that a tuple from one relation must map to a tuple in another relation.

ID	Album	Artist_ID	Year
1	The Marriage of Figaro	1	1786
2	Requiem Mass In D minor	1	1791
3	Für Elise	2	1867

Table: Albums(ID, Album, Artist_ID, Year)

ID	Artist	Year	City
1	Mozart	1756	Salzburg
2	Beethoven	1770	Bonn
3	Chopin	1810	Warsaw

Table: Artists(ID, Artist, Year, City)

Artist_ID	Album_ID
1	1
1	2
2	3

Table: ArtistAlbum(Artist_ID, Album_ID)

Foreign key (cont'd)

Foreign key constraint

The referencing attribute(s) must be the primary key of the referenced relation.

```
create table ArtistAlbum(
        Artist_ID, varchar(8),
        Albumn_ID, varchar(8),
        primary key (Artist_ID, Albumn_ID),
        foreign key (Artist_ID) references Artists,
        foreign key (Albumn_ID) references Albumns,
)
```

- Referencing relation: ArtistAlbum
- Referencing attributes: Artist_ID, Album_ID
- Referenced relations: Artist, Album