ICE2404P-01: 数据库系统概论

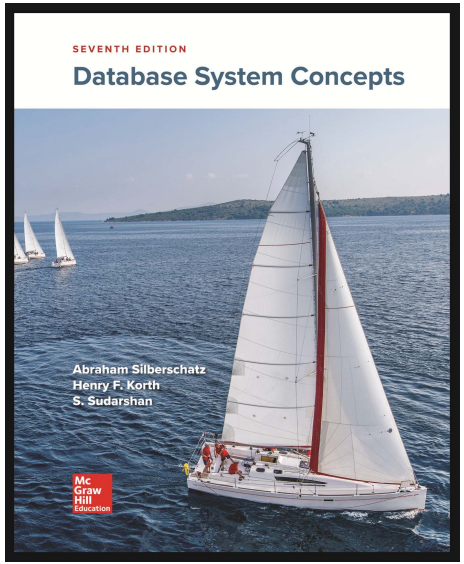
Course info

- ✓ Instructor: 尹强
  - Homepage: https://basics.sjtu.edu.cn/~yin
  - Email: q.yin@sjtu.edu.cn
- ✓ TAs
  - 王政栋：lnwzd2009@sjtu.edu.cn
  - 冯书瀚：count_von@sjtu.edu.cn
- ✓ Office hour
  - 7:00pm – 9:00pm, every Monday
  - 电院3号楼325

群聊: ICE3404P数据库系统概论

该二维码7天内(2月24日前)有效，重新进入将更新

References



Database System Concepts 7th Ed.
Silberschatz, Korth, & Sudarshan

数据库系统概念，第五版
王珊、萨师煊

Grading policy

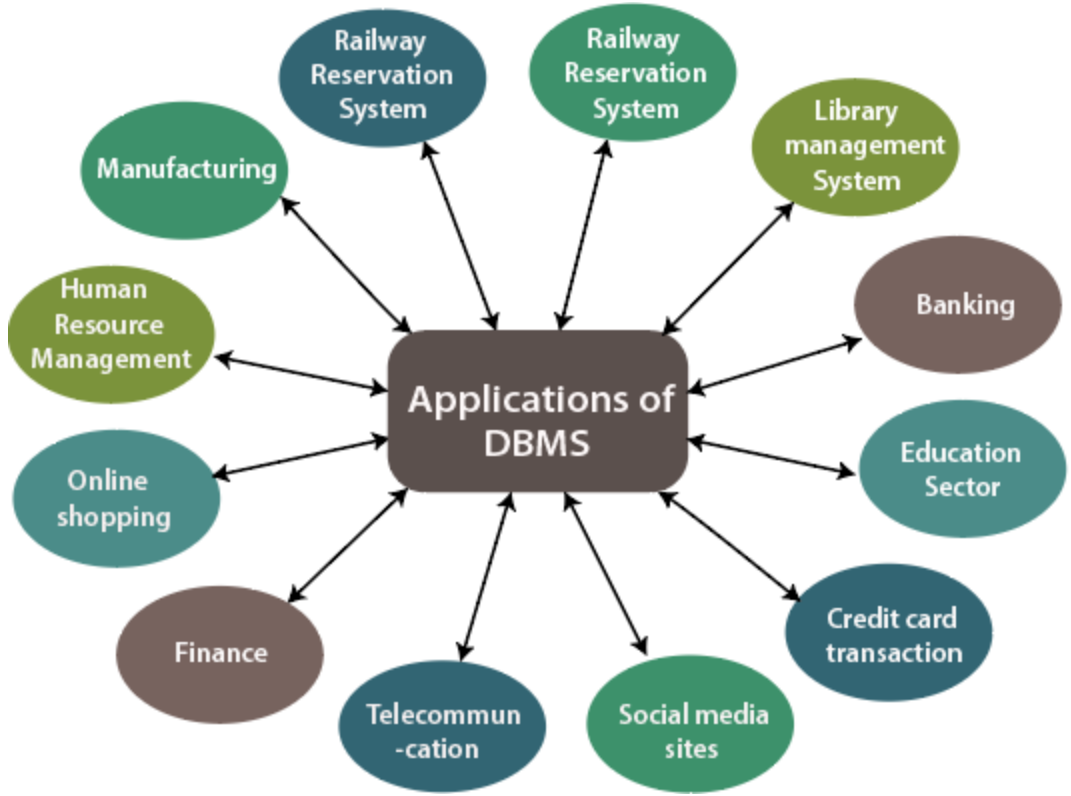
Homework	Quiz	Exam
30%	20%	50%

- ✓ All homework must be done individually.

1. Course introduction

Database history and course overview

Database applications is everywhere

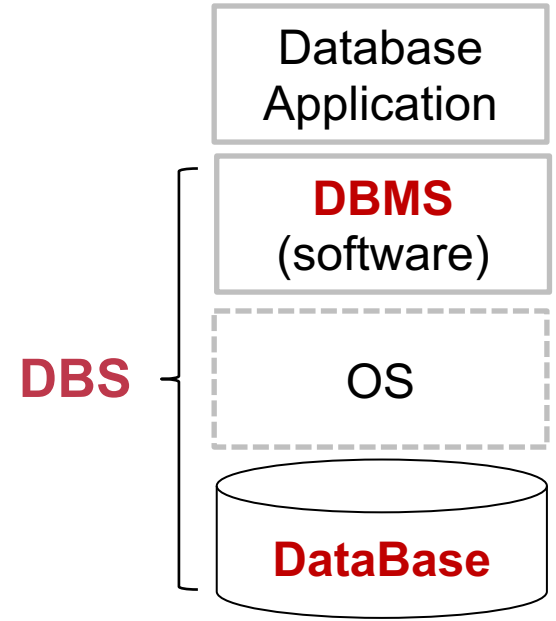


Ref. https://www.tutorialandexample.com/applications-of-dbms

Database systems

- Database: organized collection of inter-related data that models some aspect of the real-world.

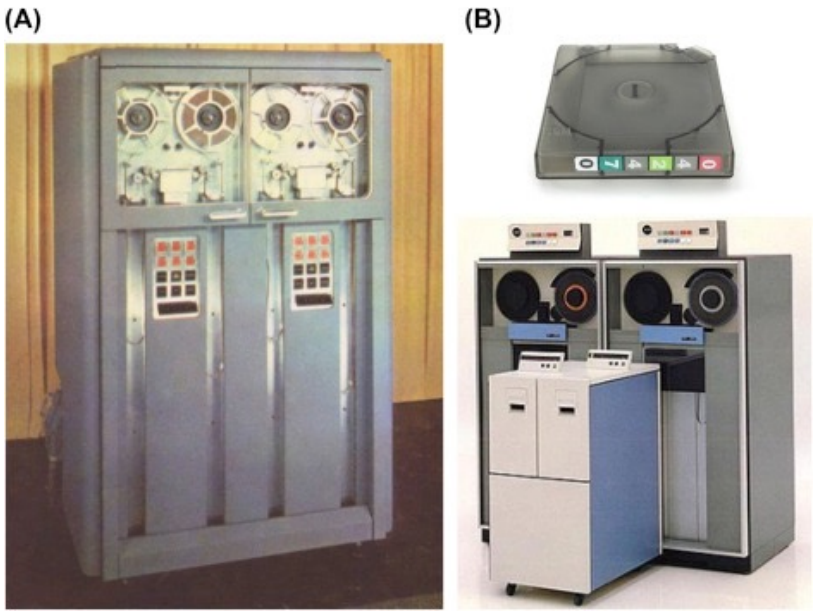
- DBMS: database management system
  - A software system that facilitates the creation and maintenance and uses of a database.



Database systems in the early days

- Data processing using magnetic tapes for storage
  - tapes support sequential access only

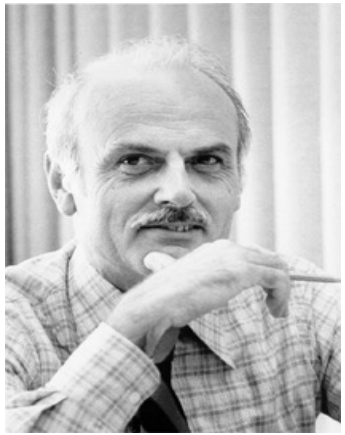
- Punched cards for input



Ref. Magnetic Tape - an overview | ScienceDirect Topics

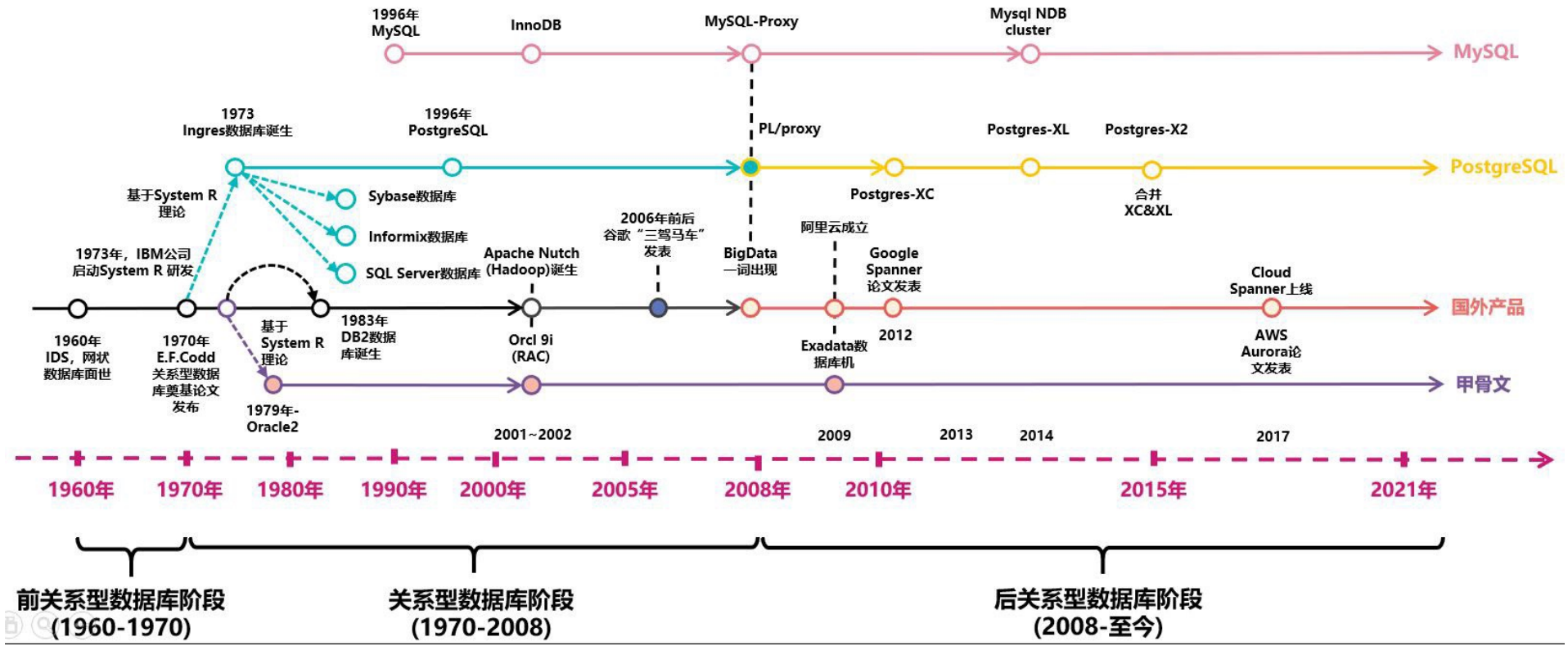
Late 1960s and 1970s

- Hard disks allowed direct access to data
- Network and hierarchical models in widespread use
- Edgar F. Codd defines the relational data model
  - Codd won the ACM Turing Award for this work
  - IBM Research begins System R prototype
  - UC Berkeley begins Ingres prototype
- High-performance (for the era) transaction processing



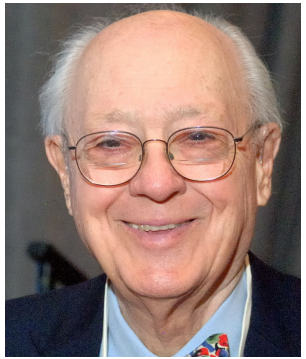
Edgar F. Codd

Brief history of database systems

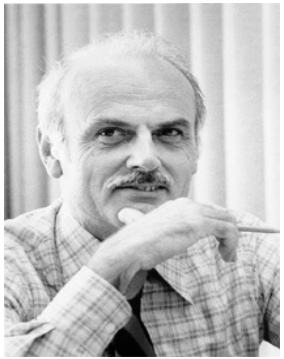


Ref. 数据库发展研究报告（2021年）.中国信通院

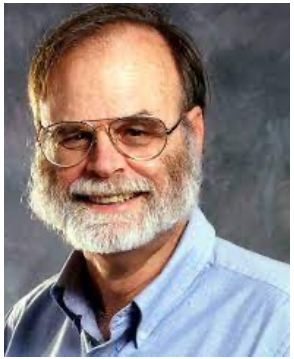
Database and Turing awards



C. Bachman
1973



E. Codd
1981



J. Gray
1998

M. Stonebraker
2014

Ref. https://amturing.acm.org/

Database in China

1978, 中国人民大学

萨师煊(1922-2010)

Ref. 中国数据库的起步和开端

Database in China (cont.)

“成立研究所确实是当务之急。我们要集中人力，搞一些切实的研究课题，开发真正能与国外竞争的数据库系统、应用生成系统产品。这是对国家最大的贡献。”

王珊

Vendors & database systems

公司	产品
阿里巴巴/阿里云	PolarDB
蚂蚁金服	OceanBase
腾讯/腾讯云	TDSQL
华为	GaussDB
中兴通讯	GoldenDB
人大金仓	KingbaseES
武汉达梦	DM8
神州通用	神通数据库
东软集团	OpenBase
南大通用	Gbase
PingCAP	TiDB
巨杉数据库	SequoiaDB

排名不分先后

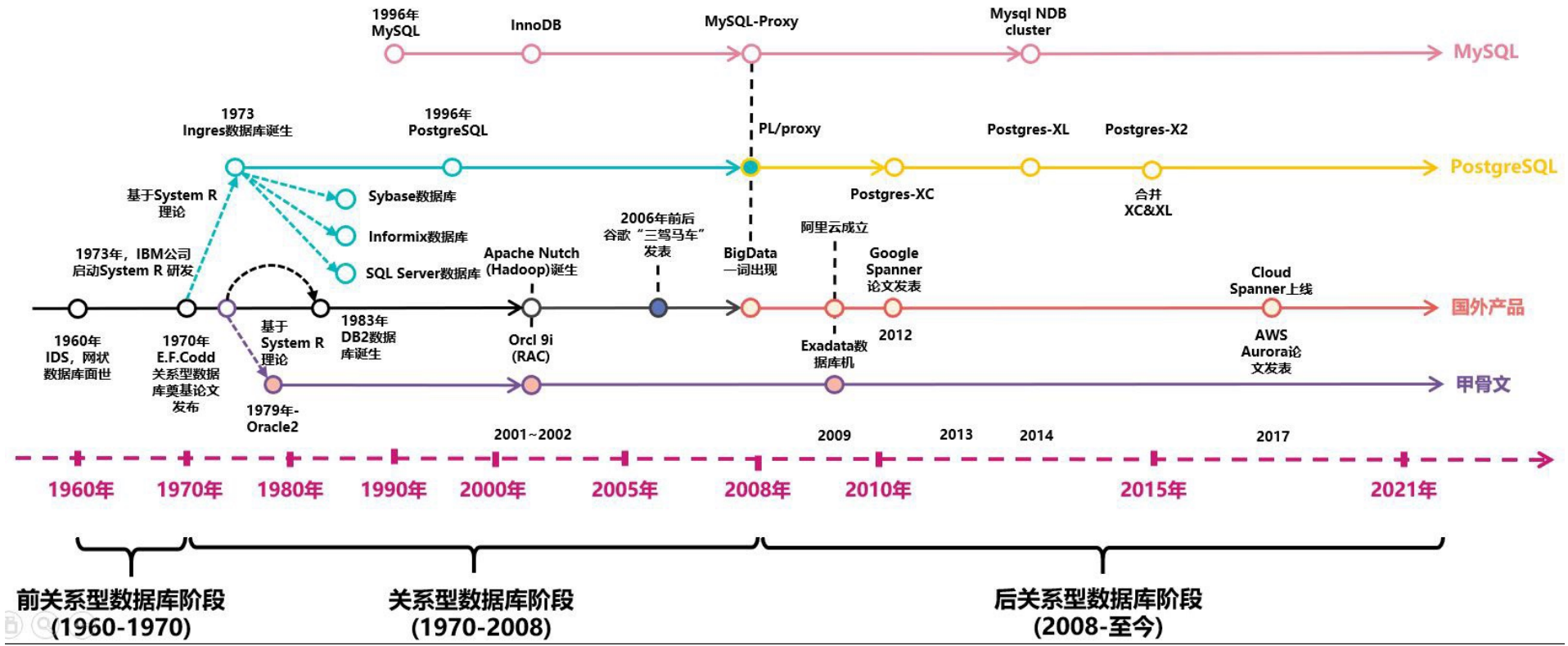
https://www.zhihu.com/question/349100846?spm=a2c6h.12873639.0.0.3e332bd80VTvl9

Vendors and systems (cont.)

阿里“去IOE”工程

https://developer.aliyun.com/article/722414
https://mp.weixin.qq.com/s/AndotCQnx8lmp483m7-INQ

Recap



Ref. 数据库发展研究报告（2021年）.中国信通院

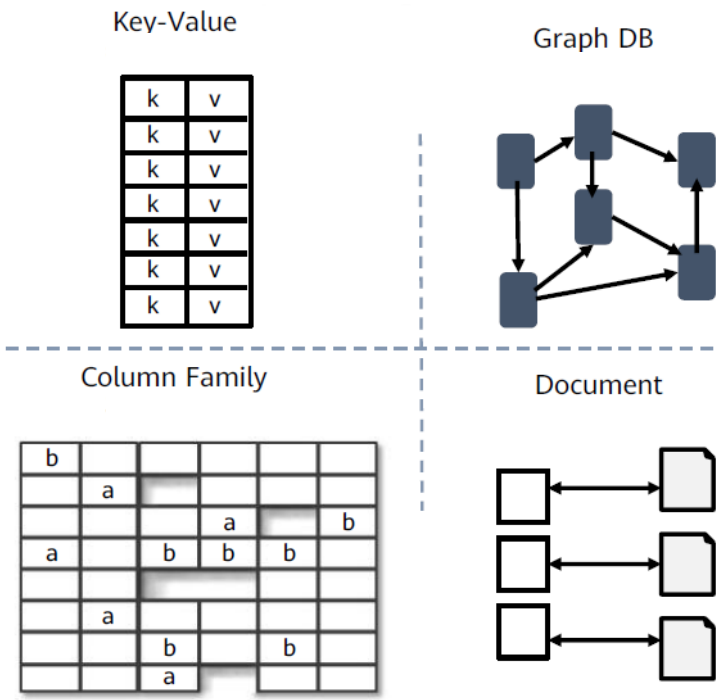
Course overview

- Relation data model
- Database query language
- Database design theory
- Database engine internals

Data models

- Relational model
- Key/Value
- Graph
- Column-family
- Document

NoSQL



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

Relational data model

Key

Attribute

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Tuple

(a) The *instructor* table

Levels of abstraction

Physical level

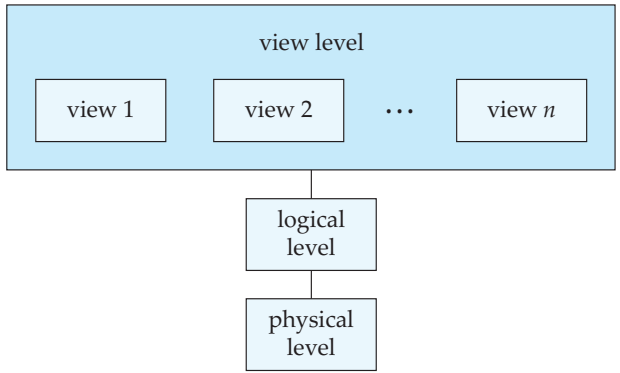
describes how a record (e.g., instructor) is stored.

Logical level

describes data stored in database, and the relationships among the data.

View level

application programs hide details of data types.



Data definition language (DDL)

Specification notation for defining the database schema.

Example:

```
create table instructor (
        ID          char(5),
        name        varchar(20),
        dept_name   varchar(20),
        salary      numeric(8,2))
```

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

- DDL compiler generates a set of table templates stored in a data dictionary.
- Data dictionary contains metadata (i.e., data about data)

Data manipulation language (DML)

- DML is also known as query language.

- Procedural DML -- require a user to specify what data are needed and how to get those data.

- Declarative DML -- require a user to specify what data are needed without specifying how to get those data.

Structured query language (SQL)

Example: to find all instructors in Comp. Sci. dept

```
select name
from instructor
where dept_name = 'Comp. Sci.'
```

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

- ✓ SQL query language is non-procedural.
- ✓ SQL is the de facto standard relational database language.

Database design

How to find a good collection of relation schemas?

Business decision : What attributes should be recorded in the DB?

Computer science decision

- What relation schemas should we have ?
- How attributes should be distributed among the various relation schemas?

Tools: Normalization Algorithms

Database engine

Storage Manager

Storing, retrieving, and updating data in the database.

Query Processor

Query compiler, query optimizer, execution engine

Transaction Manager

Concurrency control, logging, failure recovery

Storage manager

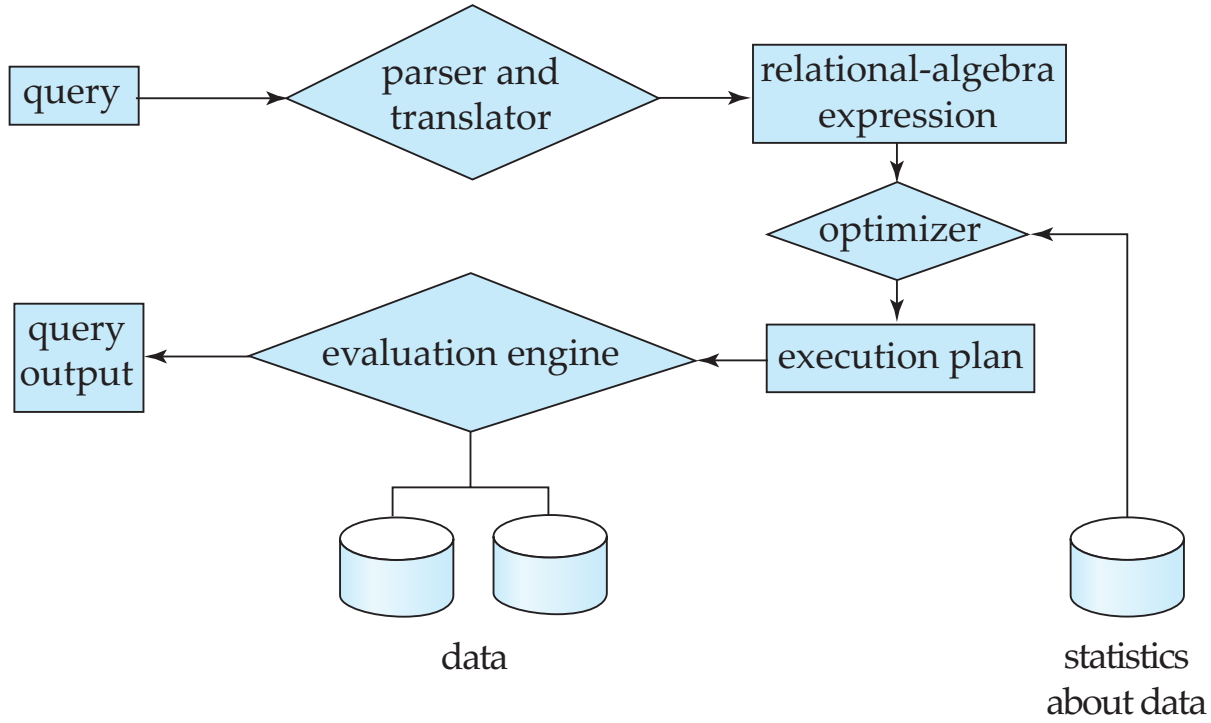
The storage manager implements several data structures as part of the physical system implementation:

Data files -- store the database itself

Data dictionary -- stores metadata about the structure of the database

Indices -- can provide fast access to data items.

Query processor



Transaction management

A transaction is a collection of operations that performs a single logical function in a database application.

- ✓ Atomicity
- ✓ Consistency
- ✓ Isolation
- ✓ Durability

Recap

- Relational data model and algebra
- Database query language
- Database design theory
- Database engine internals
  - Storage and indexing
  - Query processing and optimization
  - Concurrency control and recovery

2. Relational data model

The most successful database abstraction

Example: a music store application

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

Things we need to store:

- ✓ Information about Artists
- ✓ What Albums those Artists released

Flat file example (I)

- Store our database as comma-separated value (CSV) files that we manage in our own code.
- Use a separate file per entity.
- The application has to parse the files each time they want to read/update records

```
// Artists.csv
Mozart,1756,Salzburg
Beethoven,1770,Bonn
Chopin,1810,Warsaw

// Albumes.csv
The Marriage of Figaro,Mozart,1786
Requiem Mass In D minor,Mozart,1791
Für Elise,Beethoven,1867
```

Flat file example (II)

Example: Get the Albums composed by Beethoven.

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

```
// Artists.csv
Mozart,1756,Salzburg
Beethoven,1770,Bonn
Chopin,1810,Warsaw

// Albumes.csv
The Marriage of Figaro,Mozart,1786
Requiem Mass In D minor,Mozart,1791
Für Elise,Beethoven,1867
```

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

```
// Artists.csv
Mozart,1756,Salzburg
Beethoven,1770,Bonn
Chopin,1810,Warsaw

// Albumes.csv
The Marriage of Figaro,Mozart,1786
Requiem Mass In D minor,Mozart,1791
Für Elise,Beethoven,1867
```

Flat files: implementation

- 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?

```
// Artists.csv
Mozart,1756,Salzburg
Beethoven,1770,Bonn
Chopin,1810,Warsaw

// Albumes.csv
The Marriage of Figaro,Mozart,1786
Requiem Mass In D minor,Mozart,1791
Für Elise,Beethoven,1867
```

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?

```
// Artists.csv
Mozart,1756,Salzburg
Beethoven,1770,Bonn
Chopin,1810,Warsaw

// Albumes.csv
The Marriage of Figaro,Mozart,1786
Requiem Mass In D minor,Mozart,1791
Für Elise,Beethoven,1867
```

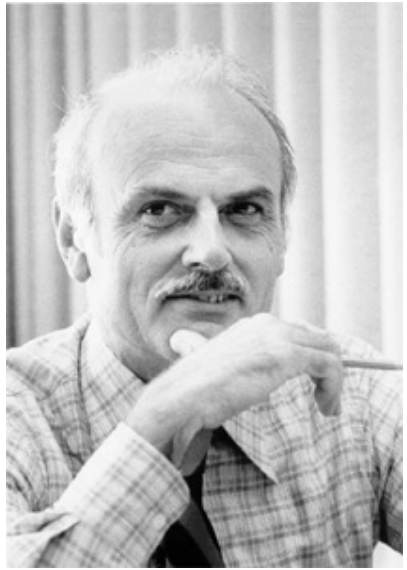
Early DBMS

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

Relational data model

- Proposed in 1970 by Edgar F. Codd.

- The most successful database abstraction
  - Store database in simple data structures.
  - Access data through high-level language.
  - Physical storage left up to implementation.



Edgar Frank Codd
Turing Award 1981

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

Relation

Artists (Artist, Year, City)

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

- A relation is an unordered set of tuples. Each tuple represents an entity.
- A tuple is a set of attribute values (also known as its domain) in the relation.
- Values are atomic/scalar.

Schema vs. instance

- Let $A_1, A_2, \ldots, A_n$ be attributes.
- $R(A_1, A_2, \ldots, A_n)$ is a relation schema.
- Relation instance: concrete table content
  – set of tuples (also called records) matching the schema

A relation schema:
Artists (Artist, Year, City)

A relation instance

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

Database schema vs. instance

- Database schema

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

- Database instance

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

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

ArtistAlbum

Artist_ID	Album_ID
1	1
2	1
2	2

Keys

- $K \subseteq \{A_1, A_2, \ldots, A_n\}$ is a superkey of schema R if values for K are sufficient to identify a unique tuple of each possible relation.
- A superkey K is a candidate key if K is minimal.

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

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

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.

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

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

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

Foreign key

Artists (ID, Artist, Year, City)

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

ArtistAlbum (Artist_ID, Album_ID)

ArtistAlbum

Artist_ID	Album_ID
1	1
2	1
2	2

Albums (ID, Album, Artist_ID, Year)

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

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

Foreign key

Artists (ID, Artist, Year, City)

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

ArtistAlbum (Artist_ID, Album_ID)

ArtistAlbum	Artist_ID	Album_ID
	1	1
	2	1
	2	2

Albums (ID, Album, Artist_ID, Year)

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

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

- Foreign key constraint: the referenced attribute(s) must be the primary key of the referenced relation.

Thanks!