1

Overview of Database

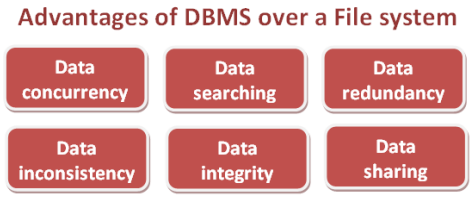
- **Database**: an organized **collection** of **information**.
  - Not really what we are specifically referring to in this module.
  - **Generic definition**. As long as the information is organized.
    - Many file types: Text files, simple records, etc., can all be database.
    - **Industry** often understands database in this way.
  - Any **size**. From a few KBs like text files to TBs, PBs like data warehouse.
  - **No rules**. Data access and edit implemented by different parties.
    - We may say most of the **functionalities** are **specialized**, not generalized.
    - **New functionalities**? Often needs **new programs**.
    - Data files changed? Programs shall be **changes** as well.
    - Not to mention the other aspects like **security**.
- Database management system (**DBMS**).
  - This is what we are mostly talking about in this module.
  - **Standardized**. Different companies follow the **same protocols and standards** for developing and using the DBMS.
  - Often called **database system** for short.

https://depositphotos.com/29748697/stock-photo-information-concept-data-collection-on.html

2

DBMS – Advantages

- Advantages of DBMS:
  - Data **independence**. (More details later)
  - **Efficient** data access.
  - Data **integrity** and **security**.
  - Data **administration**.
  - **Concurrent** access and crash **recovery**.
  - **Reduced** application **development time**.
- Scenarios:
  - # 1: Given a student ID, how do I find the student, and all the courses he/she is taking?
  - We need: **efficient** data access.
  - # 2: I have multiple people making changes to the data, how do I
    - Ensure that the data is in a consistent state?
    - Solve the problem of concurrent access (read/write)?
  - We need: data **integrity** and **concurrent** access.
  - # 3: How do we enforce different users to perform different operations on subset of data?
  - We need: **security** / access control.
- We use DBMS everywhere?



https://notesforgeeks.com/wp-content/uploads/2020/01/DBMS-advts.png

3

Motivations to Learn DBMS

- Reality (possibly): people won't praise us because we know but will trash us because we don't know.
- Can **win awards**!
  - Michael Stonebraker, Turning Award 2015. Fundamental contributions to the concepts and practices underlying modern database systems.
  - Charles W. Bachman, National Medal of Technology and Innovation. Integrated Data Store (IDS). First general-purpose DBMS. Foundations for network data model.
- Data is everywhere, **explosively increasing**.
  - Lazada: product description and photos; customer details and transactions.
  - Facebook: daily: 600TB. 2014.

https://www.javatpoint.com/types-of-databases

4

Stages of Database Design

- S1 • Requirement Analysis
- S2 • Conceptual Database Design
- S3 • Logical Database Design
- S4 • Refine the Schemas
- S5 • Physical Database Design
- S6 • Application and Security Design

5

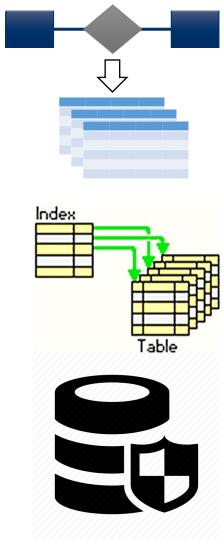
Stages of Database Design

- Stage 1: **requirement analysis**
  - Bidirectional:
    - Our **clients tell** what they want. Normally not, or even far from complete.
    - We **developers** also shall **ask and clarify**: what are the needs of our customers?
  - What **data** to be stored? Numerical or text?
  - What kind of **applications** to use the data? Online shopping or banking?
  - What **operation** needs to be performed? Read intensive? On many tables?
- Stage 2: **conceptual database design (for us)**
  - Given user requirements, what **tools** to model the requirements before creating the relevant tables in a DBMS?
  - Create one relation for all modules and students? Or create multiple?
  - Popular option: Entity Relationship (**ER**) **Diagram**.
    - Requirement analysis information -> develop a high-level **description** of the data.
    - Understand what are the **constraints** that need to be modeled.

6

Stages of Database Design

- Stage 3: **logical database design (for computer)**
  - Conceptual design -> database **schema.**
    - Determine the **DBMS** to implement the database design.
    - ER -> relational database schemas.
    - e.g., what are the columns and what are the rows.
  - ER model is most relevant to the first three stages.
- Stage 4: **schema refinement**
  - Analyze the relations -> identify potential problem -> refine it.
  - **Normalization**: the process to **reduce** data **redundancy** and **improve** data **integrity.**
- Stage 5: **physical database design**
  - Consider typical workloads and further refine database design to ensure it meets performance criteria.
  - Build **indexes** on tables and cluster some tables.
- Stage 6: **application & security design**
  - Identify what is **accessible** or **inaccessible.**
  - Enforce access rules, e.g., guest and admin.

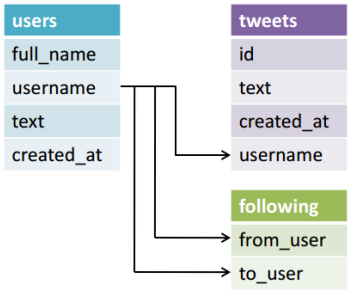


https://www.javatpoint.com/types-of-databases

7

Data Models

- Main components:
  - **Data**: key purpose is to store and manage data.
  - **Relationship**: how data is related, e.g., students and modules.
  - **Constraints**: which data shall be unique? e.g., student id.
- Most nowadays commercial products are relational-based.
- How different datasets are organized together?
  - Solution: **ER diagram.**
  - High-level design, and quite abstracted.
  - Useful and powerful.
  - Often not part of a DBMS.
  - Some design tools available.
  - Easy translation: ER diagram -> relational model.



https://www.javatpoint.com/types-of-databases 1005

8

Relational Databases

- Brief history:
  - 1960s: network data model by Charles W. Bachman.
  - 1960s: hierarchy data model by IBM.
  - 1970s: relational data model by Edgar F. ("Ted") Codd.
  - 1980s: structured query language (SQL) by IBM.
  - 1990s: many commercial products and even beyond relational database.
- Codd's 12 rules:
  - The establishment of the relational data model.
  - Fact: a set of 13 rules (0 - 12)
- Terminologies:
  - **Relation** – table; everything shown below.
  - **Relation instance** – sets of records.
  - **Relation schema** – attribute name and type.
  - **Tuple** – row or record.
  - **Attribute** – column.

https://cdn.lynda.com/course/604214/604214-637286219010719938-16x9.jpg

9

Relational Data Model

- Redefine:
  - **Database**, a collection of **≥ 1 relations**, each row represents a record.
  - **Relation**, a **table** with rows and columns.
- Advantages:
  - **Simple** data **representation**.
  - **Easy query**, expressing what you want, e.g., which column which row.
- Note: understand many of you know SQL but SQL ≠ relational data model.
- Each **relation** has a unique **name**: Students.
- Each relation has some (unique) **attributes** and **tuples**.

sid	name	NRIC	DOB	address
CSC2020001	Jon Snow	S9434567H	10/20/1994	Blk 123 Ang Mo Kio
CSC2020002	Arya Stark	S9445678J	7/8/1994	Blk 45 Sengkang
CSC2020003	Sansa Stark	S9456789B	3/20/1994	Blk 108 Bukit Batok

10

Relational Data Model

- Each row specifies the element relationship, checking headers.
  - **Fixed size**, with the same number of elements.
  - Can be **indexed**, e.g., t1 for the 1st row -> t1['name'] = t1[1] = 'Jon Snow'
  - Rows are **order-sensitive**?
- Able to specify some **constraints**.
  - **Unique** student identifier.
  - Addresses from Singapore.
- **Relation** Students:
  - Table "Students" has **3 rows**, or tuples, or records. **order-insensitive**.
  - Each row has **5 columns**, or fields, or attributes.

sid	name	NRIC	DOB	address
CSC2020001	Jon Snow	S9434567H	10/20/1994	Blk 123 Ang Mo Kio
CSC2020002	Arya Stark	S9445678J	7/8/1994	Blk 45 Sengkang
CSC2020003	Sansa Stark	S9456789B	3/20/1994	Blk 108 Bukit Batok

11

Relational Data Model

- Relation **schema**: a relation has a schema.
  - Specifies the headers of the schema, e.g., attribute name and domain.
  - **Order-sensitive**?
  - Different relations can share the same schema, e.g., year 1 year 2 students.
- Notation: r(R) - the schema of r is R.
  - Students(SID, Name, NRIC, DOB, Address)
    - r: Students.
    - R: (SID, Name, NRIC, DOB, Address)
  - Domains are not stated explicitly in this relation schema notation.
  - Some other representations include domains.

SID	Name	NRIC	DOB	Address
CSC2020001	Jon Snow	S9434567H	10/20/1994	Blk 123 Ang Mo Kio
CSC2020002	Arya Stark	S9445678J	7/8/1994	Blk 45 Sengkang
CSC2020003	Sansa Stark	S9456789B	3/20/1994	Blk 108 Bukit Batok

12

Relational Data Model

- Schema for **set operations**.
  - e.g., given r(R) and s(S), R = (sid, name, DOB) and S = (sid, DOB)
  - R ∩ S: a set of attributes that can be found both in R and S
  - R – S: attributes can be found in R but not in S
  - R' ⊆ R: a subset of the attributes in R

<null>

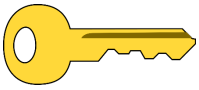
- NULL.
  - Definition: **NOT 0**; it indicates **unknown** or **unspecified** value.
  - Operations with NULLs are quite **troublesome** -> **avoid** using NULL if possible.
    - In DBMS, we can specify **default values**, e.g., if not specified, record 0.
    - Cannot find a page? Do not say NULL, say Error 404.
- **Tuples cannot** be the **same**.
  - Given two tuples in a relation.
  - May share the same values for some attributes, e.g., same age for some ones.
  - BUT, cannot be the same for every attributes -> duplication not allowed.
    - Not allowed in theory, but in practice may allow for a while. Why?

https://blog.sqlauthority.com/wp-content/uploads/2007/06/null-500x259.png

13

Avoid Duplications – Keys

- Objective: tell who is who or **distinguish** different tuples.
- Definition of **key**: a certain **minimal subset** of relation attributes **uniquely identifies** a tuple. Say,
  - **Unique**: two distinct tuples cannot have same values in all key attributes.
  - **Minimal**: this is not true for any subset of the key, or no smaller key.
- **Candidate key**:
  - Follow the definition of key.
  - **Multiple** candidate keys possible.
- **One** of the **candidate keys** to be chosen by DB admin as the **primary key**.
- The "unique" statement is true -> A **super-key**.
  - Add any other attribute in the super-key, we get another super-key.
  - Every key is super-key.



https://upload.wikimedia.org/wikipedia/commons/thumb/6/65/Crypto_key.svg/1280px-Crypto_key.svg.png

14

Keys – Example

- Super-keys is a **super-set** of candidate keys, a super-set of primary key.
- SID is a key for Students.
- SID is also a candidate key, as at least one attribute to index.
  - Candidate keys also include NRIC.
- SID is also a super-key.
- The set {SID, GPA} is a super-key.
- What about Name, or DOB?
- **Primary key** can be SID. How to indicate? -> **underline** it.
  - Students_schema = (SID, Name, NRIC, DOB, Address)

SID	Name	NRIC	DOB	Address
CSC2020001	Jon Snow	S9434567H	10/20/1994	Blk 123 Ang Mo Kio
CSC2020002	Arya Stark	S9445678J	7/8/1994	Blk 45 Sengkang
CSC2020003	Sansa Stark	S9456789B	3/20/1994	Blk 108 Bukit Batok

https://blog.sqlauthority.com/wp-content/uploads/2007/06/null-500x259.png

15

Foreign Keys

- **Foreign keys**: the primary keys of the **other relations**.
  - Set of fields in one relation used to refer to a tuple in another.
  - Referencing relation and referenced relation.
  - Must correspond to the primary key of the second relation.
  - Like a **logical pointer**.
- Example:
  - SID is a foreign key referring to Students:
  - Enrolled(SID: string, CID: string, Grade: string)

Enrolled: Referencing relation

SID	CID	Grade
53666	ICT1002	C
53666	ICT1009	B
53650	CSC2008	A
53666	CSC2008	B

Students (Referenced relation)

SID	Name	Login	Age	GPA
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

https://blog.sqlauthority.com/wp-content/uploads/2007/06/null-500x259.png

16