SWUFE Database 2024-2025-2

1. Introduction

1.1. What is database

base: the main place where a person lives and works, or a place that a company does business from.

database: A database is an organized collection of data stored and accessed electronically from a computer

DBMS: A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data

two base goal for a DBMS: convenient and efficient

1.2. database application

- E-commerce
- Enterprise Organizations
- Standalone Application: SQLite

1.2.1. application categories

- online transaction processing
    - low delay
- data analysis
    - e.g. beers and diapers

1.3. Role of the database

In other words, why file-based database is not as good as expected?

- **File processing system's disadvantage**
    - **data redundancy and inconsistency**
        - Data redundancy means higher storage costs, and multiple copies of the same data can lead to data inconsistencies.
    - **data isolation**
        - The data is scattered in different files and may also use different formats. Therefore, it is difficult to write new programs to access the data.
    - **difficulty in accessing data**
    - **integrity problem**
        - Certain values may be subject to certain constraints. For example, salary > 0. Although it is possible to implement the constraint by adding code to the program, it is not

flexible enough. For example, new constraints may be added.

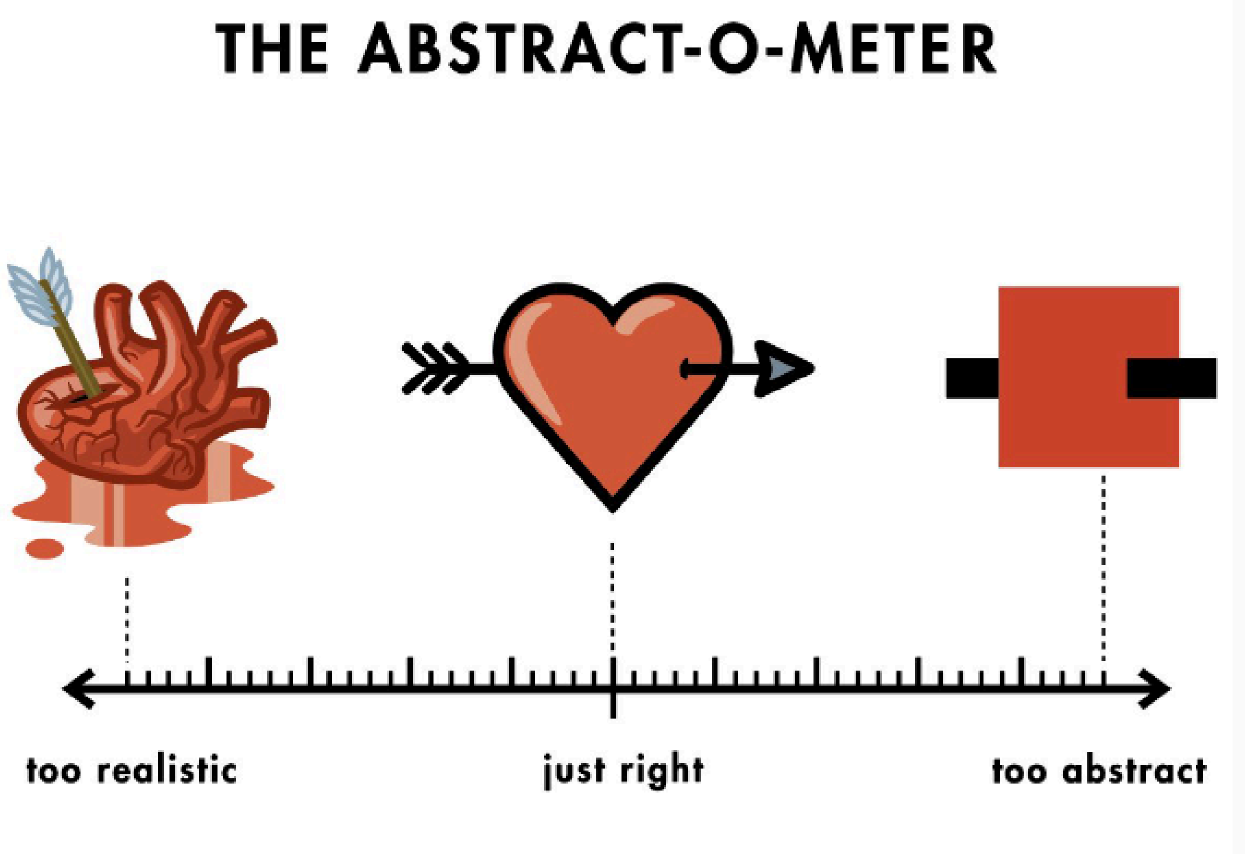
- **concurrent-access anomalies**
  - Assuming you have $100,000 in your account and two expenditures at the same moment ($10,000 and $20,000 respectively), the final result may be incorrect ($90,000 or $80,000).
- **atomicity problems**
  - You are in the process of making a payment and the system crashes, at this point it may appear that your money is deducted but the object is not received

Keep in mind the trade-off concept: a file-processing system and a database system (DBMS) are not completely opposed to each other; use the right system for the right situation.

- When to use DBMS
  - highly valuable
  - relatively Larger
  - accessed by multiple users and applications

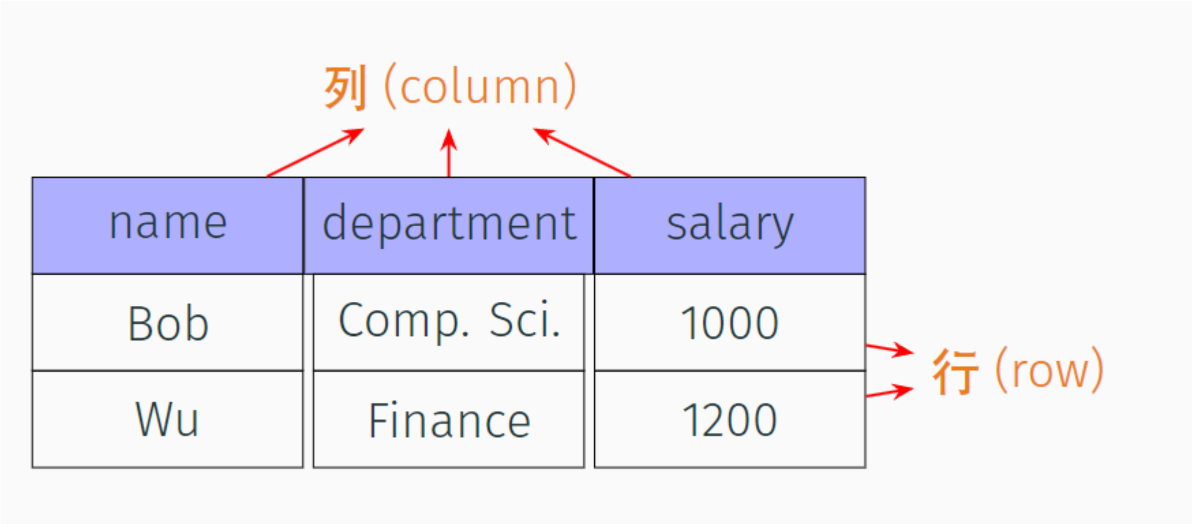
1.4. view of data

- Abstraction: help you to Ignore irrelevant details



- data model: A collection of conceptualization tools that describe data, data relationships, data semantics, and consistency constraints.

- relational model: table is relation ← Most commonly used models

- entity-relationship model: the (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects. ← Widely used in database design.
- semi-structured data model: individual data items of the same type may have different sets of attributes. ← Widely used in internet and big data scenarios
- object-based data model
- network model/ hierarchy model

- schema and instance
  - schema: The overall design of the database (translated as "綱要" in Taiwan) can be categorized into physical schema, logical schema, and sub-schema according to the different levels of data abstraction.
  - Instance: A collection of information stored in the database at a specific moment in time.

1.5. database language

The database system provides:

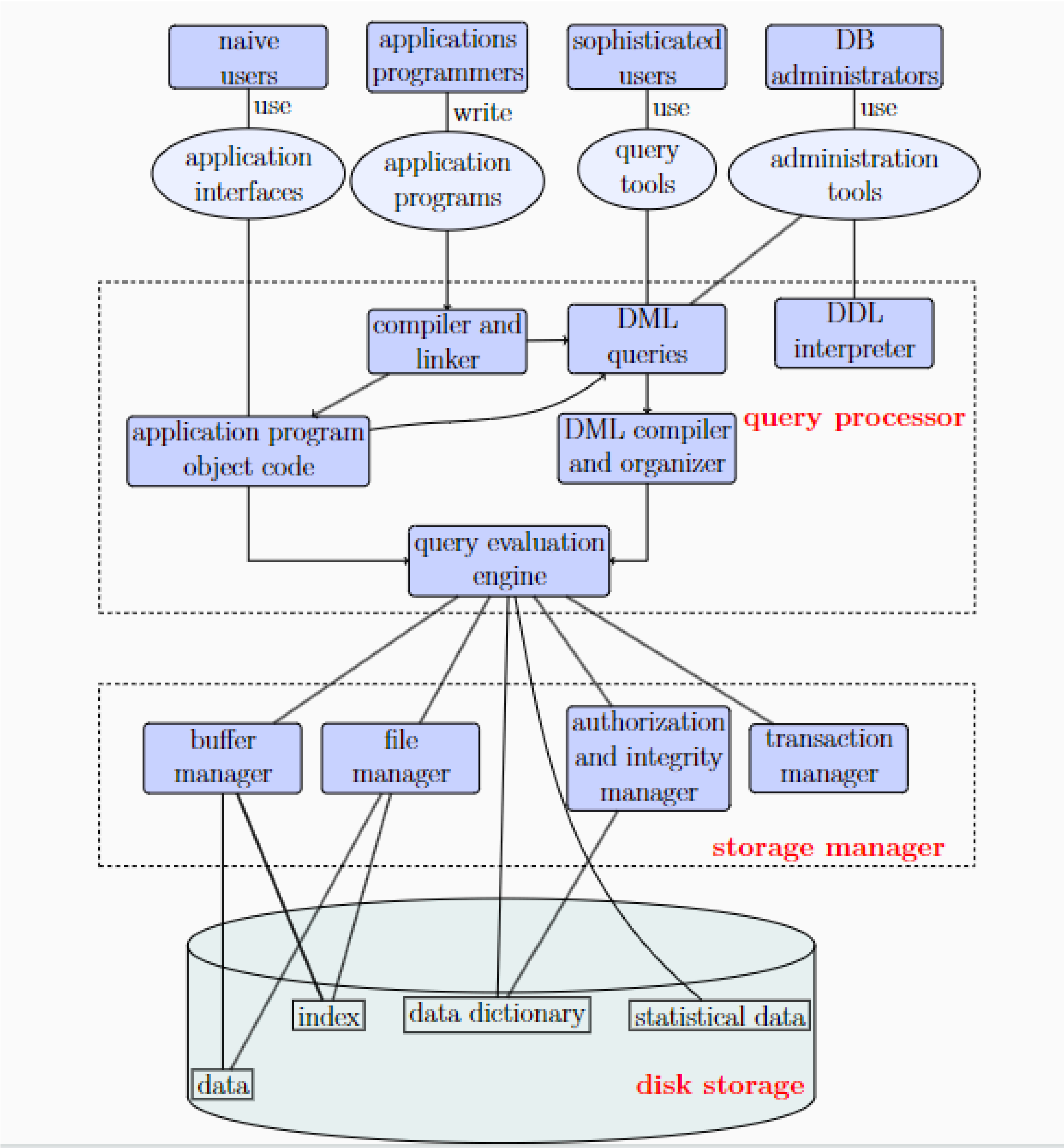
- data-definition language: defines the database schema
- data-manipulation language: expresses database queries and updates.

SQL (Structured Query Language) is the current mainstream. It is a declarative language, i.e., it focuses on What, not How.

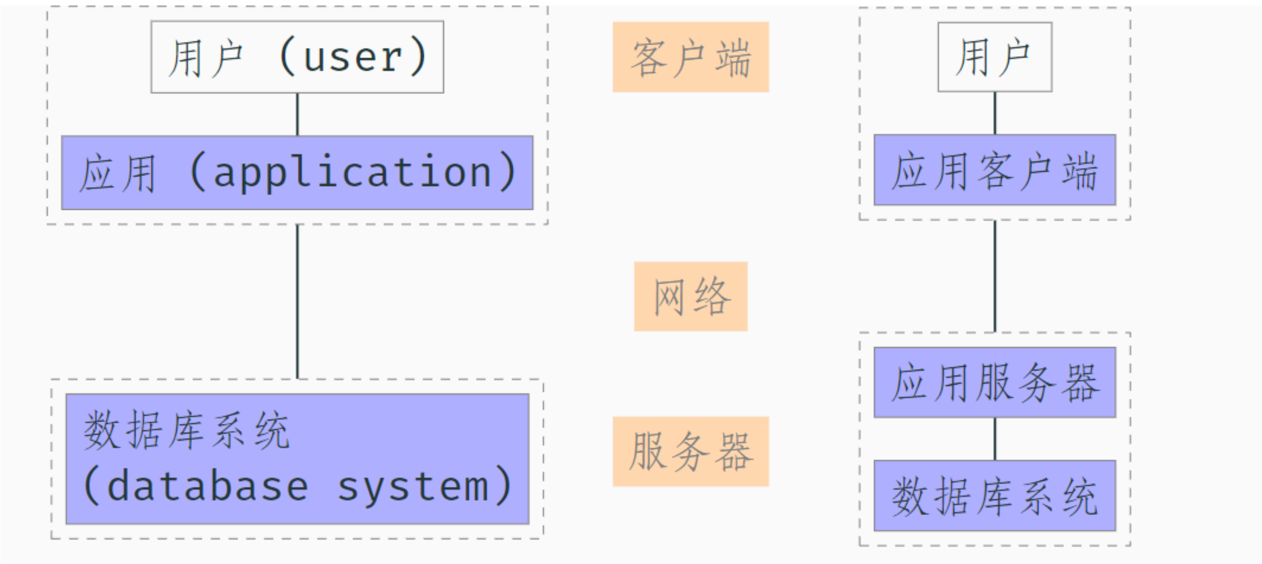
- DDL: Defines database structure (e.g., CREATE, ALTER, DROP).
- DML: Manipulates data within the database (e.g., SELECT, INSERT, UPDATE, DELETE).

1.6. Database System Architecture

too complex to understand, and meaningless as well.



- Tips：



- It's silly to connect a database system directly to an application like this on the left

2. Relational model

consists of a collection of tables

- Relation:
  - Given sets X and Y , the Cartesian product $X \times Y$ is defined as $\{(x, y)|x \in X, y \in Y\}$ and its elements called ordered pairs
  - A binary relation R over sets X and Y is a subset of $X \times Y$.

2.1. the structure of relational model

database	Excel
relation	table
tuple	row
attribute	column

A row in a table represents a relationship among a set of values.

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4

The *course* table

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101

Two courses are **related**.

The *prereq* table

The tuples here are not in order, but can be repeated

2.2. relation schema

- database schema: Logical design of the database
- database instance: A snapshot of the data in the database at a given moment

Similarly, there is a relation schema and a relation instance.

- relation schema: The name of a relation and the set of attributes for a relation
  - e.g. The relation schema of this table

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

is **instructor(ID, name, dept_name, salary)**

all schema during this course can check in reference/schema.pdf

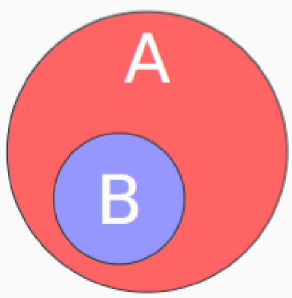
3. key

A way to **distinguish** between **different tuples** in a given relation.

- super key: A collection of **one or more attributes**, such that the combination of attributes allows us to **uniquely identify** a tuple in a relation.

people(name, age, origin, nationality, id)

- (id)
- (id, name)
- (id, age)
- (id, name, age)
- ...



- candidate key: Its true subset cannot form a super key of a super key (also called "minimal super key").
  - Also not unique
- primary key: Candidate key that is selected by the database designer.

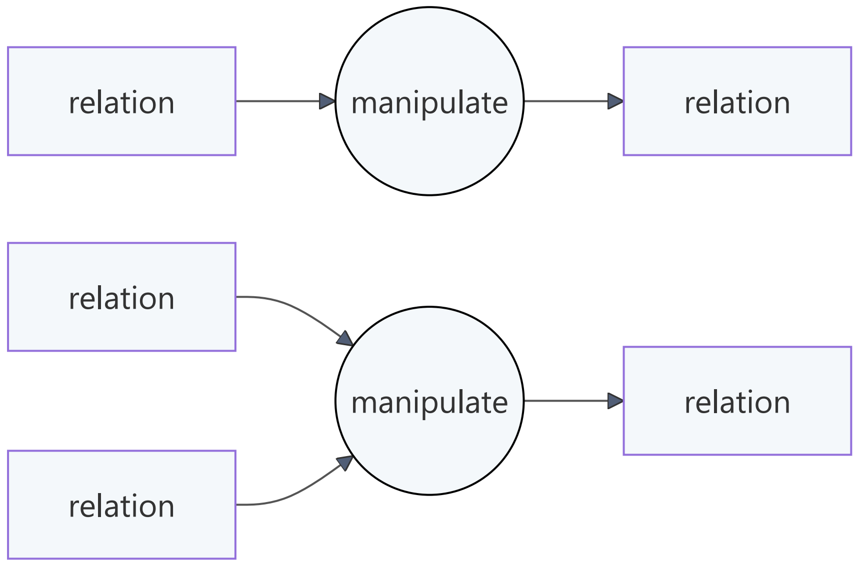
- use **underline** to identify
- ***atomicity***

- foreign key:
  - e.g. Attribute **dept_name** is a **foreign key** from instructor, **referencing** department. instructor(ID, name, dept_name, salary); department(dept_name, building, budget)
  - referencing relation and referenced relation
  - **Note: The foreign key does not necessarily have the same name as the primary key to which it is referentially related.**

4. relation algebra

> Relational algebra is the theoretical foundation of SQL

Relational algebra is defined over **relations, tuples and attributes.**



Operations are applied either to a single relation or to two relations.**The result is always a single relation.**

4.1. SELECT

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

e.g1: Select all teachers whose **dept_name** is **Physics**.

$$\sigma_{dept\_name=\text{"physics"}}(\text{instructor})$$

e.g2: Select all teachers with `salary greater than 90000'.

$$\sigma_{salary>90000}(\text{instructor})$$

Comparison operations on predicates (predicate) can use the symbols: $>, <, =, \geq, \leq, \neq$.In addition, multiple predicates can be combined by **logical connectives**:

- and: $\wedge$
- or: $\vee$
- not: $\neg$

e.g3: Select all teachers in the Physics Academy with a salary greater than 90000.

$$\sigma_{\text{physics="Physics"} \wedge \text{salary} > 90000}(\text{instructor})$$

4.2. PROJECT

e.g1: Returns the name, dept_name, and salary of all teachers.

$$\Pi_{\text{ID,dept\_name,salary}}(\text{instructor})$$

- select(σ) : Intercepts relationships horizontally, affecting [line].
- project(Π) : Intercepts relationships vertically, affects [columns].

Moreover, the generic projection operation allows simple arithmetic on attributes:

$$\Pi_{\text{ID,dept\_name,salary}/12}(\text{instructor})$$

4.3. Combination of relational operations

e.g4: Find the **names** of **all Physics** faculty members.

$$\Pi_{name}\,(\sigma_{dept\_name} = \text{"Physics" (instructor)})$$

e.g5: Find information on all faculty members belonging to the **Physics' or** Chemistry' department.

$$\sigma_{dept\_name=\text{"Physics"}\vee \text{dept\_name="Chemistry"}}(\text{instructor})$$

e.g6: Find name and salary of the teacher with ID 10101

$$\Pi_{\text{name,salary}}(\sigma_{\text{ID}=10101})(instructor)$$

4.4. Cartesian product

$$A \times B = \{(a, b) : a \in A, b \in B\}$$

- instructor(ID, name, dept_name, salary)
- teaches(ID, course_id, sec_id, semester, year)
- The schema of r = instructor $\times$ teaches : r(instructor.ID, name, dept_name, salary, teaches.ID, course_id, sec_id, semester, year)

instructor.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2017
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2018
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2017
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2017
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2018

$r = instructor \times teaches$

Find information about all the teachers and their classes:

4.4.1. JOIN

JOIN is a combination of Cartesian product and selection:

$$\text{instructor} \bowtie_{\text{intructor.ID=teaches.ID}} \text{teaches}$$

- natural join

If the θ condition is that **attribute values of the same name are equal**, it can be omitted .In this case, it is called a **natural join** .

$$\text{instructor} \bowtie \text{teaches}$$

4.5. UNION, INTERSECT, DIFFERENCE

e.g. section(course_id, sec_id, semester, year, building, room_number, time_slot_id); Find all courses that are in both the Fall 2017 semester and Spring 2018 semester

$$\Pi_{\text{course\_id}}(\sigma_{\text{semester}=\text{"Fail"} \wedge \text{year}=2017}(\text{section})) \cap \Pi_{\text{course\_id}}(\sigma_{\text{semester}=\text{"Fail"} \wedge \text{year}=2018}(\text{section}))$$

The precondition for two relations to perform the parallelism operation is that they are compatible, i.e., the two relations have the same arity and each corresponding attribute is of the same type.

e.g. Find the names of all employees with a salary greater than $10,000

- employee(ID, person_name, street, city)
- works(ID, company_name, salary)
- company(company_name, city)

$$\Pi_{\text{person\_name}}(\sigma_{\text{salary}>10000}(\text{works} \bowtie \text{employee}))$$

5. Introduction to SQL

SQL: Structured Query Language

> SQL is a domain-specific language used in programming and designed for managing data held in a RDBMS.

5.1. SQL:DDL(Data definition language)

The DDL provides the Define/Modify Relationship Schema and Delete Relationship commands.

5.1.1. fundamental data type

form	typology	clarification
Numeric	`int`	Integer type (machine related, equivalent to `integer`)
	`smallint`	Small-range Integer (subset of int)
	`numeric(p,d)`	Fixed point numbers, (at most) with p digits and d digits to the right of the decimal point (equivalent to `decimal` in PG)
	`float(n)`	Floating point numbers with at least n (binary) bits of precision.
	`real/double precision`	Floating-point and double-precision floating-point numbers (8/16-bit significant figures)
String	`char(n)`	Fixed length string of length n (equivalent to `charater`)
	`varchar`	Variable-length strings up to n (equivalent to `character varying`).

form	typology	clarification
	text	Non-SQL standard, represents strings of arbitrary length.
	null	Each datatype may contain a special value called null, which indicates a missing value: may exist but unknown / may not exist
notes：		

- float(1) to float(24)： real
- float/float(25) to float(53)： double precision

Name	Storage Size	Range
smallint	2 bytes	-32768 to +32767
integer	4 bytes	-2147483648 to +2147483647
real	4 bytes	6 decimal digits precision
double precision	8 bytes	15 decimal digits precision

5.1.2. Basic Schema Defination

```SQL
create table department
(
dept_name varchar(20),
building varchar(15),
budget numeric(12, 2),
primary key (dept_name)
);
```

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000

```SQL
CREATE TABLE database_name
(
    A1 D1,
    A2 D2,
    ...,
    An Dn,
    [integerity-constraint 1]
    ...,
    [integerity-constraint k]
);
```

example on integrity constraint:

```SQL
1. PRIMARY KEY (...)
2. FOREIGN KEY  (...)  REFERENCES other_database_name
```

5.1.3. Delete

- Delete table

```SQL
drop table r;
```

- Delete table but remain itself

```SQL
delete from r;
```

Relationship names, attribute names, and keywords are not case sensitive, but keywords are generally capitalized.

e.g.

```sql
CREATE TABLE product
(
ID char(20)
price numeric(8,2),
name char(50),
sell_num int,
description varchar(100),
PRIMARY KEY (ID)
);
```

5.2. SQL:DML (Data Manipulation Language)

5.2.1. Basic Query Structure

The basic SQL query structure consists of 3 clauses: **SELECT, FROM, WHERE**.

```sql
SELECT A1, A2 , ... , An
FROM r1, r2, ... , rm
WHERE p;
```

- $\Pi_{\text{name}}(\text{instructor})$

```sql
SELECT name FROM instructor
```

- duplicate

```sql
SELECT DISTINCT dept_name FROM instructor
```

- $\Pi_{\text{name,salary}/12}(\text{instructor})$

```sql
SELECT name, salary/12 FROM instructor
```

- The **where** clause selects a tuple that satisfies a condition by means of a predicate; **and, or, not** can be used between predicates.

$$\Pi_{\text{name}}(\sigma_{\text{dept\_name}=\text{"Phtsics"} \wedge \text{salary}>70000}(\text{intructor}))$$

```SQL
SELECT name
FROM instructor
WHERE
    dept_name = 'Physics'
    and
    salary>70000;
```

- not equal： <> / !=

```SQL
SELECT name FROM instructor
WHERE dept_name !='Physics';

SELECT name FROM instructor
WHERE dept_name <>'Physics';
```

- between

```SQL
SELECT name FROM instructor
WHERE salary BETWEEN 90000 AND 100000;
```

- line constructor

```SQL
SELECT course_id FROM section
WHERE (semester,year) = ('Spring' , 2018 );
```

- Multi Table Query

```sql
SELECT name, instructor.dept_name, building
FROM insturctor, department
WHERE instructor.dept_name=department.dept_name
```

```sql
SELECT name, course_id
FROM instructor i , teaches t
WHERE
  i.ID = t.ID
  AND
  dept_name='Computer';
```

- Rename

$$\rho_X(E)$$

```sql
SELECT name AS teacher FROM instructor;
```

- `*` : asterisk

```sql
SELECT  * FROM instructor
```

- order (ascending by default)

```sql
SELECT *
FROM instructor
ORDER BY salary DESC , name ASC
```

5.2.2. string function

- some string function
  - lower() / upper()
  - trim(): **trim(' SWUFE ')='SWUFE'**
  - length()

- put together:
  - mysql:concat/ group_concat
  - sql server: +
  - pg: || / string_agg
- fuzzy query **LIKE**
  - % : Match any **string**
  - *: Match any* ***character*** *e.g. abc' LIKE 'abc' : True 'abc' LIKE 'a%' : True 'abc' LIKE '_b*' : True 'abc' LIKE 'c' : False
  - Escape characters: when you need to match **%, \, _** in a string: all preceded by **** escape.
- ELSE:
  - [1]https://www.postgresql.org/docs/14/functions-string.html
  - [2]https://www.postgresql.org/docs/14/functions-matching.html

5.2.3. set operation

- UNION
- INTERSECT
- EXCEPT

5.2.4. case

```sql
SELECT  somecolumns,
      CASE
         WHEN condition1 THEN result1
         WHEN condition2 THEN result2
         WHEN conditionn THEN resultn
         ELSE result
         END
FROM somewhere
```

5.2.5. NULL

NULL: Missing value

5.2.5.1. arithmetic operations

The result of an arithmetic expression (e.g., + - * /) is null if either input to the expression is null.

12	98345	Kim	Elec. Eng.	80000.00
13	8888	Zhongpu	Comp. Sci.	<null>

5.2.5.2. Comparison of null values

Consider Boolean: **true, false, unknown** (this is the logical state, but will show up as NULL)

Neither false nor unknown will appear in the results.

- AND
  - true and unknown is unknown
  - false and unknown is false
  - unknown and unknown is unknown
- OR
  - true or unknown is true
  - false or unknown is unknown
  - unknown or unknown is unknown
- NOT
  - not unknown is unknown

In other words, It's unselectable in almost every cases

5.2.5.3. test null

IS NULL

5.2.6. Aggregate Functions

e.g.:

- max
- min
- avg
- count Cannot have **count(distinct *)** , and will count NULL fields
- sum

Aggregate functions are not allowed in WHERE clauses, except with subqueries

5.2.6.1. Aggregate by groups

use **GROUP BY**

5.2.6.2. Having()

If you need to qualify a grouping, such as "average salary exceeds $42,000", you cannot use the where clause, but need to use the having clause.

Must be used in conjunction with `GROUP BY`, otherwise not legal

5.2.7. Order of execution

- First compute a relation based on the `from` clause.
- If there is a `where`, apply the `where` predicate to the relationship.
- If there is a `group by`, form a group based on the above result. If not, the entire set of tuples is treated as a group by.
- If `having` is present, it will be applied to each group; groups that do not satisfy the predicate `having` will be discarded.
- `select` uses the remaining groupings to produce the tuples in the query result

5.2.8. Sub-Queries

`select-from-where` is nested in another query

5.2.8.1. set membership

To test whether a tuple is in an enumerated set, use the conjunction `IN`, usually used in a set generated by `SELECT

5.2.8.2. scalar sub-query

Returns only **a single element** with a single attribute that can appear anywhere

5.2.9. Comparison of sets

- **at least bigger than one** :with `>some`

```sql
SELECT name
FROM instructor
WHERE
    salary > SOME (
        SELECT salary
        FROM instructor
        WHERE
            dept_name = 'History'
    );
-- Can also be expressed as "greater than minimum"
SELECT name
FROM instructor
WHERE
    salary > (
        SELECT min(salary)
        FROM instructor
        WHERE
            dept_name = 'History'
    );
```

You can replace `some` with `any`

ALL PARA	ALL PARA
$<$some	$\geq$ some
$\leq$ some	=some $\Leftrightarrow$ in
$>$ some	<>some

- **Bigger than all** with `> all`.

5.2.10. Empty Relationship Test

Use `select` to test if the relationship returned by a subquery is empty

5.2.11. WITH()

The `with` statement defines a temporary relationship to be used for the current query. e.g.

```sql
WITH max_buedget(value) AS (SELECT MAX(budget) FROM department)
SELECT budget
FROM department,
    max_buedget
WHERE department.budget = max_buedget.value;
```

5.3. SQL: CHANGE

5.3.1. DELETE

```sql
DELETE FROM r
WHERE p
```

Notice that **delete** only deletes the content reservation table, while **drop** deletes the entire table

5.3.2. INSERT

```sql
-- Order by attribute list (not recommended)
INSERT INTO course
VALUES ('CS-205', 'Database Systems', 'Comp. Sci.', 4); -- Specify attributes (not in DDL attribute order).
-- Specify attributes (not in DDL attribute order)
INSERT INTO course(course_id, title, dept_name, credits)
VALUES ('CS-205', 'Database Systems', 'Comp. Sci.', 4); -- Specify attributes (not in DDL attribute order).
```

It is also possible to insert a collection of tuples, or partial attributes (remaining null by default), the It is also possible to insert the result of a query or quickly import them via COPY, etc.

5.3.3. UPDATE

```sql
UPDATE table
SET attribute = p   -- p can also be CASE(...)
```