

Chapter 9 Database

created by HardyWen

Basics

1. Key definitions

- They might ask you about these basic definitions in the test...
- **Database**
 - an example of application software to store and manipulate data
 - a collection of data that is set up in a structured way
- **table**
 - a set of data about one type of object, e.g. student
- **field**
 - an individual piece of data (a **column** in the table)
 - e.g. date of birth
- **record**
 - all of the records in a table about one object (a **row** in the table)
 - e.g. all the personal data of one student
- **primary key**
 - a **unique identifier** for a record
 - it cannot appear twice in different records
 - e.g. student ID number
 - it purposes to index each record so that by searching the value of the primary key the whole record can be reached and found
- considering the data table named **EmpData** below

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7329	SMITH	CLERK	7692	17-DEC-03	800	100.00	20
7330	ALLEN	MANAGER	7839	19-APR-08	2975	400.00	30
7342	WARD	ANALYST	7788	23-MAY-05	3000	300.00	30
7521	JONES	CLERK	7566	19-OCT-07	1000	200.00	20
7623	LARRY	MANAGER	7692	16-DEC-09	2500	500.00	40

table: "EmpData"

records: "7329 SMITH CLERK 7692 17-DEC-03 800 100.00 20"

"7330 ALLEN MANAGER 7839 19-APR-08 2975 400.00 30" ...

fields: "EMPNO", "ENAME", "JOB", "MGR" ...

primary key: "EMPNO"

2. Data Types

- **Text & Alphanumeric**

- Text: just like **strings** in the pseudocode
- Alphanumeric: the combination of text and numbers
 - but it is treated as a string, so that arithmetic operations cannot be used on the text or alphanumeric data
- e.g., "Hello", "2JK8a9", "Hi!!!", "293"

- **Character**

- a single letter, symbol or number
- same as *text & alphanumeric*, the character data type is treated as a **char**, so arithmetic operations cannot be used on it
- e.g., "?", "a", "2"

- **Boolean**

- simply **true / false**
- such a data **does not** have speech mark around it
- e.g., True, False

- **Integer**

- integers in $[-\infty, +\infty]$
- e.g., -1, 299, 0

- **Real**

- like **float** in python
- **at least 1** decimal point
 - a number with at least one decimal place
- e.g., 10.0, 0.0, 29.292

- **Data & time**

- a date and/or a time
- e.g., 1/1/2001, 8:30, 9/1/2007 6:00

3. Defining a single table database

1. Identify the fields needed

2. Identify the data types for each field

3. Identify a primary key

SQL

- SQL stands for **Structured Query Language**
 - a standard language used to define and manipulate databases

1. SQL Scripts

- we are using **SQL Scripts** to carry out SQL statements
 - Scripts: a set of statements that are executed

Considering the data table **EmpData** for any scripts below

The diagram illustrates the structure of the **EmpData** table. The table has 8 columns: **EMPNO**, **ENAME**, **JOB**, **MGR**, **HIREDATE**, **SAL**, **COMM**, and **DEPTNO**. The **EMPNO** column is highlighted as the **Key Field** (primary key). The values in this column are 7329, 7330, 7342, 7521, and 7623. The **Key value** is indicated for the first row (7329). The **Field or Column** label points to the column headers. The **No duplicate or null value** annotation points to the **EMPNO** column. The **Record or Row** label points to the entire row of data.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7329	SMITH	CLERK	7692	17-DEC-03	800	100.00	20
7330	ALLEN	MANAGER	7839	19-APR-08	2975	400.00	30
7342	WARD	ANALYST	7788	23-MAY-05	3000	300.00	30
7521	JONES	CLERK	7566	19-OCT-07	1000	200.00	20
7623	LARRY	MANAGER	7692	16-DEC-09	2500	500.00	40

1. SELECT ... FROM

```
SELECT Field1, Field2, Field3...  
FROM TableName  
-- This statement fetches data from specific field(s) from the data table
```

- example

```
-- Example 1  
SELECT EMPNO FROM EmpData  
-- output will be  
7329  
7330  
7342  
7521  
7623  
  
-- Example 2  
SELECT EMPNO, ENAME, SAL FROM EmpData  
-- output will be  
7329 SMITH 800  
7330 ALLEN 2975  
7342 WARD 3000
```

```
7521 JONES 1000
7623 LARRY 2500
```

2. SELECT ... FROM ... WHERE ..

```
SELECT Field1, Field2 ...
FROM TableName
WHERE conditions -- SQL has the same operations with Pseudocode, using "="
and "<>"
```

- example

```
SELECT EMPNO, SAL
FROM EmpData
WHERE EMPNO > 7300 AND SAL <> 1000
-- output will be
7623 2500 -- this is the data for Larry
```

3. ORDER BY

```
SELECT Field1, Field2 ...
FROM TableName
WHERE conditions -- this is optional - you could ignore where it is is
unnecessary
ORDER BY Fieldname DESC/ASC -- ASC means ascending, DESC means descending
```

- example

```
SELECT ENAME, SAL
FROM EmpData
ORDER BY SAL DESC
-- output will be
WARD 3000
ALLEN 2975
LARRY 2550
JONES 1000
SMITH 900 -- this just arranges the names regarding the salary they earn
```

4. SUM

```
SELECT SUM(Fieldname)
FROM TableName
WHERE conditions -- again, WHERE is optional
```

- example

```
SELECT SUM(SAL)
FROM EmpData
WHERE SAL <= 1000
-- output will be
1800 -- 800 + 1000 = 1800
```

5. COUNT

- it counts how many records meet the criteria

```
SELECT COUNT(Fieldname)
FROM Tablename
WHERE conditions -- if you want to count the total number of records in a
table, ignore this WHERE
```

- example

```
SELECT COUNT(EMPNO)
FROM EmpData
WHERE DEPTNO = 20
-- output will be
2 -- because only Allen and Jones are in the 20th department
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7329	SMITH	CLERK	7692	17-DEC-03	800	100.00	20
7330	ALLEN	MANAGER	7839	19-APR-08	2975	400.00	30
7342	WARD	ANALYST	7788	23-MAY-05	3000	300.00	30
7521	JONES	CLERK	7566	19-OCT-07	1000	200.00	20
7623	LARRY	MANAGER	7692	16-DEC-09	2500	500.00	40