Chapter7 SQL

—— **S**tructured **Q**uery **L**anguage

Kun Yue

May, 2009

Outline

- Introduction
- The role of SQL in a database architecture
- The SQL environment
- Defining a database in SQL
- Inserting, updating, and deleting data
- Internal schema definition in RDBMS
- Summary

Introduction (1)

- A query language for relational databases
- Standard language for querying and manipulating data
- Created in 1970s at IBM Research Labs, San Jose
- Has evolved, acquiring more and more features
- SQL 86—SQL 92—SQL 99—SQL 2003

  SQL 92 is widely supported at various levels
- DBMSs are SQL compliant
- SQL is supported by many products available running on all machine sizes, from small personal computer to large mainframes

Introduction (2)

Example 1:

```
SELECT   *
FROM     Company
WHERE    country='USA' AND stockPrice > 50
```

Example 2:

INSERT INTO R(A_1,…., A_n) VALUES (v_1,…., v_n)

Example 3:

```
CREATE  TABLE person(
        name                    varchar(30),
        social-security-number  int,
        age                     shortint,
 );
```

The role of SQL in a database architecture

- Introduction
- The role of SQL in a database architecture
- The SQL environment
- Defining a database in SQL
- Inserting, updating, and deleting data
- Internal schema definition in RDBMS
- Summary

The role of SQL in a database architecture (1)

- The information is retrieved using an SQL query
- SQL commands can be executed within the RDBMS

  - Data definition and manipulation
  - Data structure and operation definition
  - Standard specification
  - Handling referential integrity, managing transaction, user-defined functions, join operators

- Each vendor's version of SQL includes enhancements, features, and capabilities that extend their version beyond the baseline standards of SQL-92.

The role of SQL in a database architecture (2)

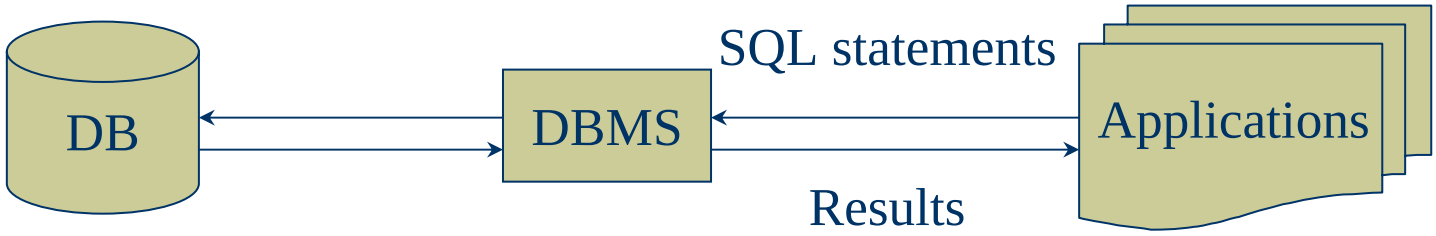
- **The benefits of SQL standard:**
- Reduced training cost
- Productivity
- Application portability
- Application longevity
- Reduced dependence on a single vendor
- Cross-system communication
- …

The SQL environment

- Introduction
- The role of SQL in a database architecture
- The SQL environment
- Defining a database in SQL
- Inserting, updating, and deleting data
- Internal schema definition in RDBMS
- Summary

The SQL environment (1)

- **DBMS as the interface**



- **Category — Database**

  - Each database is contained in a category
  - Category is a set of schemas
  - A schema is the structure which contains description of objects created by a user (E.g, tables, views, constraints, triggers, etc.)

The SQL environment (2)

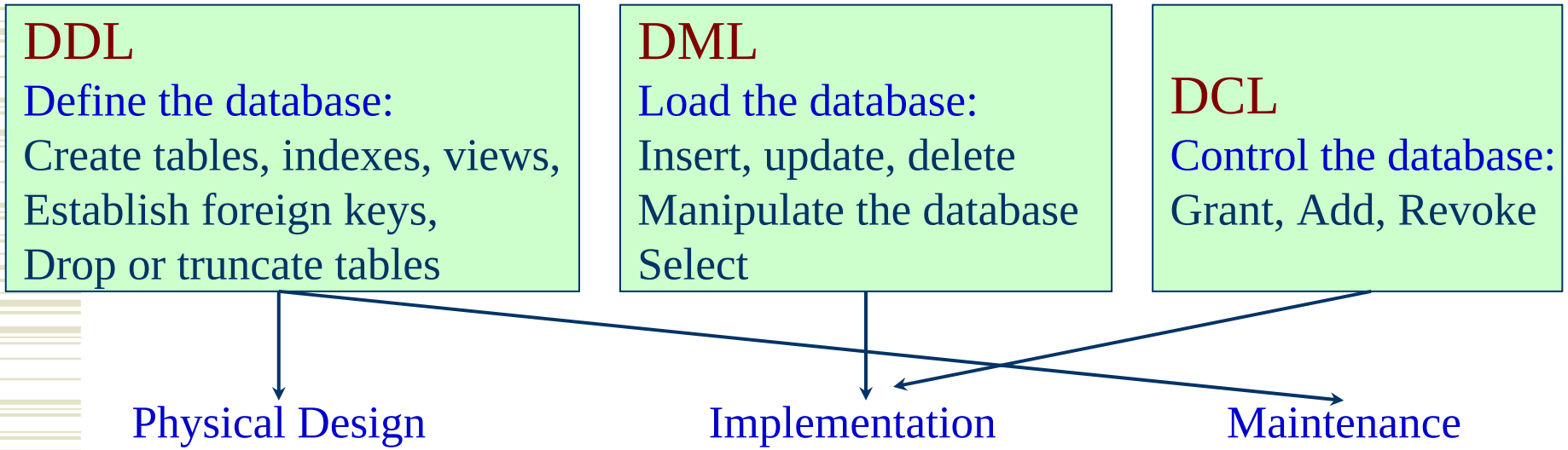
- **3 types**

  DDL: data definition language

  DML: data manipulation language

  DCL: data control language

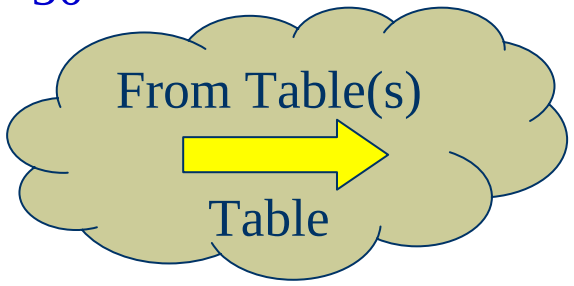
- **DDL, DML, DCL and the database development process**



Selection (1)

- **Projections and Ordering results**

```
SELECT  name, stock price
FROM   Company
WHERE   country='USA' AND stockPrice > 50
   ORDER BY  country, name
```



Note:

- Comparison operators: =, <>, <, >, <=, >=
- Arithmetic operations: *, /, +, —
- Pattern matching: *s* LIKE *p*
- Logic operations: AND, OR, NOT
- Special stuff for comparing dates and times
- …

Selection (2)

- **Join (Inner join)**

Tables:

Product (name, price, category, maker)
Purchase (***buyer***, seller, store, product)
Company (name, stock price, country)
Person (***name***, phone number, city)

buyer	seller	store	product
Mary	Com1	S1	gizmo
Mary	Com1	S1	cookie
Mary	Com2	S2	gizmo
John	Com1	S1	gizmo

name	Phone number	city
Mary	13000000000	KM
John	13500000000	KM
Rose	13900000000	SH

Query:

SELECT name, store
FROM Person, Purchase
WHERE Person.name=Purchase.buyer AND
city='KM' AND product= 'gizmo'

Selection (3)

- **Tuple variables (Alias)**

Table:

Product (name, price, category, maker)

Requirement:

Find pairs of companies making products in the same category

Query:

```
SELECT  product1.maker, product2.maker
FROM  Product AS product1, Product AS product2
WHERE  product1.category=product2.category
          AND  product1.maker <> product2.maker
```

Selection (4)

- **Union**

```
(SELECT  name FROM  Person WHERE  City='Seattle')
    UNION
(SELECT  name FROM  Person, Purchase
 WHERE  Purchase.buyer=Person.name AND store='The Bon')
```

Note:

- The 2 components of UNION are the tables with the same attributes !
- UNION is not efficiently ☹

Selection (5)

- **Subqueries**

```
SELECT Purchase.product FROM  Purchase
WHERE  buyer =
          (SELECT  name
           FROM  Person
           WHERE  phone number = '13000000001')
```

Single value

```
SELECT Purchase.product FROM  Purchase
     WHERE  buyer IN
               (SELECT  name FROM  Person WHERE  city = 'KM')
```

Relations

You can also use:

- s > ALL R; - s > ANY R; - (not) **EXISTS** R

Selection (6)

- **Removing duplicates**

```
SELECT  DISTINCT buyer
WHERE product='gizmo'
```

- **Aggregation**

```
SELECT SUM(price)
FROM  Product
WHERE  maker='Toyota'

SELECT COUNT(category)
FROM Product
WHERE maker='Toyota'
```

Purchase

buyer	seller	store	product
Mary	Com1	S1	gizmo
Mary	Com1	S1	cookie
Mary	Com2	S2	gizmo
John	Com1	S1	gizmo

Aggregation operations returning an *integer*:

- SUM
- MIN
- MAX
- AVG
- COUNT

Applied to a single attribute

Selection (7)

- **Grouping and Aggregation**

Requirement: Find how much we sold of every product

Query:

```
SELECT    product, Sum(price)
FROM      Product,  Purchase
WHERE     Product.name = Purchase.product
GROUP BY  Product.name
```

Note:

1. Compute the relation (i.e., the FROM and WHERE).
2. Group by the attributes in the GROUP BY
3. Select one tuple for every group (and apply aggregation)

SELECT can have (1) grouped attributes or (2) aggregates.

Selection (8)

- **Having clause for aggregation**

Requirement:

Same query as the previous one, except that we consider only products that had at least 100 buyers.

Query:

```
SELECT  product, Sum(price)  FROM  Product,  Purchase
WHERE   Product.name = Purchase.product
GROUP BY  Product.name
HAVING  Count(buyer) > 100
```

Having:
conditions on aggregates

Defining a database in SQL

- The role of SQL in a database architecture
- The SQL environment
- Defining a database in SQL
- Inserting, updating, and deleting data
- Internal schema definition in RDBMS
- Summary

Defining a database in SQL (1)

- **Create tables**
  - columns and their data types
  - unique, if not null
  - primary and foreign keys (reference)
  - default value
  - check

Check:

Product_Finish varchar(20)
CHECK(Product_Finish IN ('Cherry', 'Natural Ash', 'White Ash, 'Red Oak', 'Natural Oak','Walnut'))

Example:

```
CREATE TABLE Order(
 Order_ID Number(11,0) Not Null
 Order_Date Date Default SysDate,
 Customer_ID Number(11,0),
 Constraint Order_PK Primary Key(Order_ID),
 Constraint Order_FK Foreign Key(Customer_ID));
```

Defining a database in SQL (2)

- **Using and defining views**

- Base table—physically store data

  View (Dynamic view)

  —***virtual*** table created dynamically upon request by a user

- View's definition: stored in the system category

  View's contents can be ***materialized*** as a result of an SQL query using the view

- **Motivation**

  ① View instead of the join operation on multiple tables — Simplify queries

  ② Help to establish security

  ③ Privacy and confidentiality of data

  ④ Greater programming productivity

- CREATE VIEW View-Name

  AS Selection statement

Defining a database in SQL (3)

- **Inherent characteristics and some problems of the view**

- A view is costly (time): Its contents must be calculated each time that they are requested.

- Trade-off strategy for the materialized view:

① Improve the performance in time by sacrificing storage space

② Copies and replications of data based on SQL

Queries created in the same manner as dynamic views

Problem of the materialized view:

How to synchronize it with its associated base tables?

Possible strategies:

Refresh the materialized view on a predetermined time interval or triggered when the table needs to be accessed

Defining a database in SQL (4)

- **Inherent characteristics and some problems of the view (Cont.)**

- The maintenance overhead and benefit:

When remote copies as distributed data are stored locally as materialized view:

① Keep the local view synchronized with the remote base tables or data warehouse

② The performance of distributed queries is improved

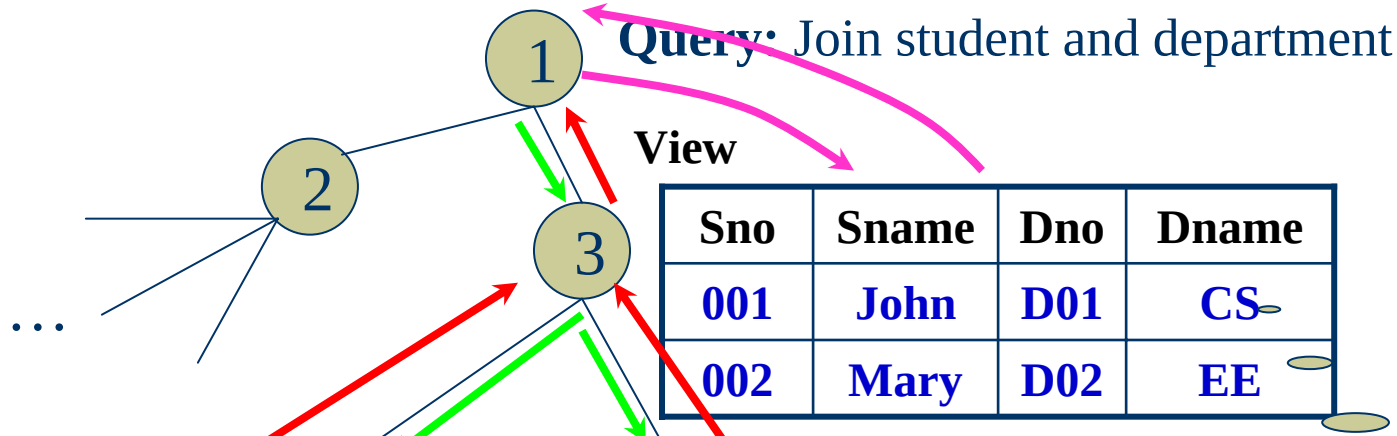
- Discussions:

① Which kinds of cases will improve the performance of materialized views to the maximum extent?

② How about the materialized views in P2P nodes?

Defining a database in SQL (5)

- **Materialized view in Web environments:**



Query: Join student and department

View

Sno	Sname	Dno	Dname
001	John	D01	CS
002	Mary	D02	EE

Department

Dno	Dname
D01	CSE
D02	EE

Student

Sno	Sname	Dep
001	John	D01
002	Mary	D02

How to do?

Defining a database in SQL (6)

- **Creating data integrity controls**

- restrict update:

E.g, Constraint Customer_PK Primary key (Customer_ID), On Update Restrict

- cascaded update:

E.g, On Update Cascade

- set null update or set default update:

E.g, On Update Set Null / On Update Set Default

- **Changing table definitions and removing tables**

- Add
- Drop
- Alter

Inserting, updating and deleting data

- The role of SQL in a database architecture
- The SQL environment
- Defining a database in SQL
- Inserting, updating, and deleting data
- Internal schema definition in RDBMS
- Summary

Inserting, updating and deleting data

- Insert into

(1) Insert Into Person (name, phone number, city)

Values (Zhang, '13900000000', 'ZZ')

(2) Insert Into KM_Person

Select * from Person Where city='KM'

- Delete from

Delete from Person Where city='KM'

- Update Table_name Set

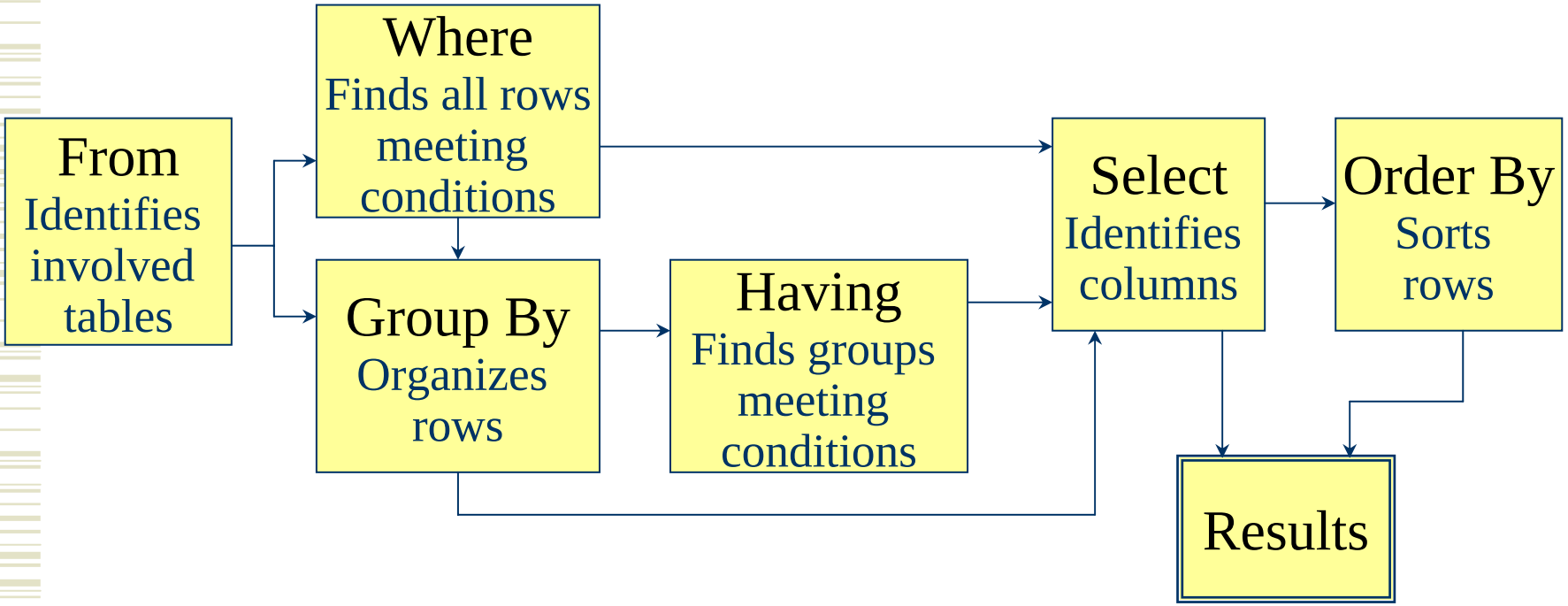
Update Person set phone number = '13600000000'
Where name='Wang'

Internal schema definition in RDBMS

- The role of SQL in a database architecture
- The SQL environment
- Defining a database in SQL
- Inserting, updating, and deleting data
- Internal schema definition in RDBMS
- Summary

Internal schema definition in RDBMS

- **SQL statement processing order:**



Summary

- The role of SQL in a database architecture
- The SQL environment
- Defining a database in SQL
- Inserting, updating, and deleting data
- Internal schema definition in RDBMS
- Summary

Summary

- Functionalities of SQL in RDBMS
- SQL environment: DDL, DML, DCL
- Defining a database in SQL
  - Tables, Views, Integrity control, Changing and removing tables
  - Materialized views
- Data Processing commands of SQL

Assignments

(1) Interpret and contract following terms:

① Base table;

② Dynamic view;

③ Materialized view

(2) Page 290: 1; 2; 6.(a), (b); 7; 9.(b), (c)

(3) Based on Figure 7-9 (*The answers are not unique*)**:**

① Delete the course which is taken by the faculty whose ID is '2143';

② Delete the courses that are taken by less than 10 faculties;

③ Update the faculty name as 'Updated Faculty' if he takes the course whose ID is 'ISM 3112'

The end

Thanks!