**SQL environment**

Definition:

The context in which SQL-data exists and SQL-statements are executed

Architecture type:

C/S (Clients and Servers)

Three-Tier (Servers has three layer):

Web servers --- talk to the user

Application servers --- execute the business logic

Database servers --- get what the app servers need from the database

DB elements/descriptors defined in an SQL environment:

tables, views, triggers

Organization of DB Elements:

##########################################

Schemas (Basic unit, 在最底层，相当于文件) -->

Schema = {base tables, views, domains, UDTs, constraints, triggers, char sets, colations, …}

@Close to but less than what we call a “database”

Syntax:

Create-->

CREATE SCHEMA schema\_name

CREATE TABLE ...

CREATE VIEW ...

CREATE ASSERTION ...

Current-->

SET SCHEMA schema\_name;

##########################################

Catalogs（Basic unit for supporting unique, accessible terminology,1+ schemas组成，相当于文件夹）-->

Syntax:

**Create**-->

Wrong: Catalog = {INFORMATION\_SCHEMA, Schema1, Schema2, ...}， or Create Catalog…

@ add new elements in Catalog, you need to change to current .

-->SET CATALOG catalog\_name; # 将当前schema的目录设置为目的catelog,即添加schema到catalog, 虽然有些繁琐，但是SQL规定这么做。

@ 模式元素的完全名: Catelog name . Schema name. Elemets

@ 目录是支持唯一的可访问术语的基本单元

##########################################

Clusters（1+catalogs组成，相当于主文件夹）

Syntax:

Cluster = {catalogs}

@ Each user has an associated cluster of all catalogs accessible to the user.

(一个Cluster相当于一个分配给用户的主文件夹，与该用户有关的所有的数据库的信息（储存、执行）都包含于其中)

@A cluster is the maximum scope of DB operations

#############################################

#############################################

SQL:

SQL schema, catalog, cluster:

Information Schema:

1. Every catalog contains an INFORMATION\_SCHEMA that includes the descriptors of a number of schema objects

2. Mostly are view definitions --> define every descriptor in that catalog to be accessed

Each view in Information Schema is so specified that a given user can access only those rows of the view that represent descriptors on which that user has privileges.

3. Not changed, as though it was SQL-data.

@The mechanisms for creating and destroying catalogs|| schemas are implementation-defined (need execution of SQL-catalog|| SQL-schema statements )

SQL data:

SQL-data is data described by SQL-schemas

SQL-data consists entirely of table variables, called base tables.

**连接（Connection）、会话（Session）、模块（Module）**

SQL connections between C/S:

Syntax:

CONNECT TO server\_name(may be DEFAULT) AS conn\_name

AUTHORIZATION user and password

@ Default connection: simply executing SQL statements at a host with a SQL client.

States: active, dormant, terminated

active vs. dormant : many opened, one active --> SET CONNECTION conn\_name

terminated: --> DISCONNECT conn\_name

@ 一旦连接被终止，将不能再用Set connection 激活

SQL sessions:

Definition: SQL operations, performed while a connection is active

Components --> Each session has

1. a current catalog,

2. a current schema within that catalog

3. an authorized user

SQL modules:

Definition:

SQL term for an application program. (SQL agent: an execution of a module)

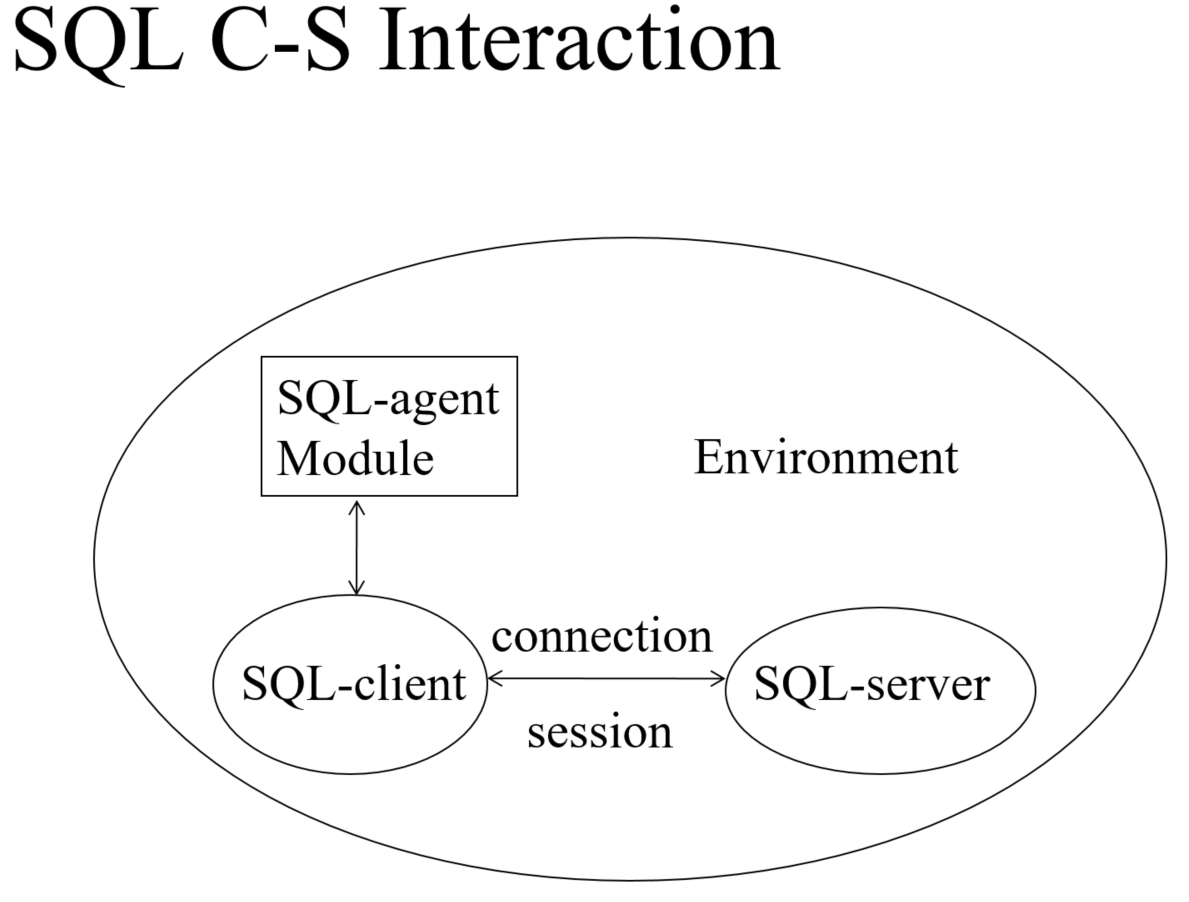
module vs. SQL agent like program vs. process

Types of modules:

Generic SQL interface: each query or other statement is a module by itself. （普通界面）

Embedded SQL: the compiled host-language program is a module.

True modules(PSM): a collection of stored procedures/functions.



@ SQL clients and servers:

1. An SQL-client, SQL-server is a processor, perceived by the SQL-agent as part of the SQL-implementation.

2. Client（可看成3-tier Architecture中 Application Server） establishes SQL-connections between itself and SQL-servers（可看成Database Server）, while Server manages SQL-data.

3. An SQL-agent is bound to an SQL-client

#####################################################

#####################################################

SQL with host language:

@SELECT … FROM cannot be used directly. --> It producees a set of tuples, cause impedance mismatch: HL doesn’t support set type directly. How to solve--> cursor

Without cursor selct --> Single-row SELECT

EXEC SQL SELECT \*

INTO :sno,:name,:age,:dept

FROM S

WHERE sno = ‘007’;

SQL cursor: a handle associated with a relation

Full syntax:

DECLARE cursor\_name

[SENSITIVE | INSENSITIVE | ASENSITIVE] A:Asynchronous -->default: ASENSITIVE. Protecting against concurrent updates

[SCROLL | NO SCROLL] CURSOR --> default: NO SCROLL

[WITH HOLD | WITHOUT HOLD] --> default: WITHOUT HOLD : whether auto close after transaction commit

[WITH RETURN | WITHOUT RETURN] FOR query --> default: WITHOUT RETURN

[ORDER BY …]

[FOR {READ ONLY | UPDATE [OF column]}]

@Read only cursors can run simultaneously with an insensitive cursor for the same relation

@query is executed when cursor is OPENed, not when it is DECLAREed

@Modifying DB by Cursor--> DELETE FROM …WHERE CURRENT OF cursor\_name || UPDATE … SET …WHERE CURRENT OF cursor\_name

e.g Delete the 3rd tuple of Student.

EXEC SQL DECLARE cursorS CURSOR FOR Student;

EXEC SQL OPEN cursorS;

EXEC SQL FETCH FROM cursorS INTO :v1,:v2,:v3,:v4;

EXEC SQL FETCH FROM cursorS INTO :v1,:v2,:v3,:v4;

EXEC SQL DELETE FROM S

WHERE CURRENT OF cursorS;

EXEC SQL CLOSE cursorS;

SQL shared variables:

EXEC SQL BEGIN DECLARE SECTION;

char snoInput[8], nameInput[10];

char SQLSTATE[6];

EXEC SQL END DECLARE SECTION

**SQL sytax example:**

1.Catelogs={Information schema, schema1, schema2,…}

2.Create schema my\_schema

Create table tablename(attributes)

Create view viewname as SQL\_query\_Definition

Create Constraints cons\_name Check (…) || Primary KEY … || references FK

3. Begin declare SECTION

Contents:

End declare SECTION

4. Declare cursor\_name CORSOR for …

Open cursor\_name

Fetch from cursor\_name into …

Delete from … where cursor\_name of …(Current)

Close cursor\_name