**Database Chapter Four Outline**

**Chapter 4: Advanced SQL**

**SQL Data Types and Schemas**

**EXACT(*field* from *d*)**

**CAST *e* AS *t***

**create type *Dollars* as numeric (12,2) final**

**create domain *person\_name* char(20) not null**

**Types and domains are similar. Domains can have constraints, such as not null, specified on them. Domains are not strongly typed.**

**Domain Constraints**

**Domain constraints are the most elementary form of integrity constraint. They test values inserted in the database, and test queries to ensure that the comparisons make sense.**

**Large-Object Types**

**Large objects (photos, videos, CAD files, etc.) are stored as a *large object*:**

**blob: binary large object -- object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system)**

**clob: character large object -- object is a large collection of character data**

**When a query returns a large object, a pointer is returned rather than the large object itself.**

**Constraints on a Single Relation**

**not null**

**primary key**

**unique**

**check (*P* )*,* where *P* is a predicate**

**Candidate keys are permitted to be null unless they have explicitly been declared to be not null (in contrast**

**to primary keys).**

**create table *depositor*(*customer\_name* char(20),  
*account\_number* char(10),  
primary key(*customer\_name, account\_number),*foreign key(*account\_number* ) references *account,*foreign key(*customer\_name* )references *customer* )**

**Assertions**

**create assertion <assertion-name> check <predicate>**

**Forms of authorization on parts of the database:**

**Read - allows reading, but not modification of data.**

**Insert - allows insertion of new data, but not modification of existing data.**

**Update - allows modification, but not deletion of data.**

**Delete - allows deletion of data.**

**Authorization Specification in SQL**

**The grant statement is used to confer authorization**

**grant <privilege list>**

**on <relation name or view name> to <user list>**

**grant select on *branch* to *U*1 with grant option**

**Revoking Authorization in SQL**

**revoke select on *branch* from *U1, U2, U3***

**Trigger Example in SQL:1999 ECA(Event-Condition-Action model)**

**Create trigger *overdraft-trigger* after update on *account*referencing new row as *nrow* for each row  
when *nrow.balance* < 0**

**referencing old row as : for deletes and updates**

**referencing new row as : for inserts and updates**

**SQL Functions**

**create function *account\_count* (*customer\_name* varchar(20))  
returns integer  
 begin  
 declare *a\_count* integer;  
 select count (*\** ) into *a\_count* from *depositor* where *depositor.customer\_name = customer\_name* return *a\_count;* end**

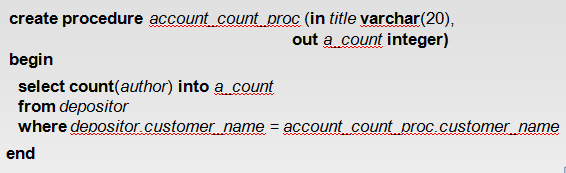
**create function *accounts\_of* (*customer\_name* char(20)**

**returns table ( *account\_number* char(10),**

***branch\_name* char(15),**

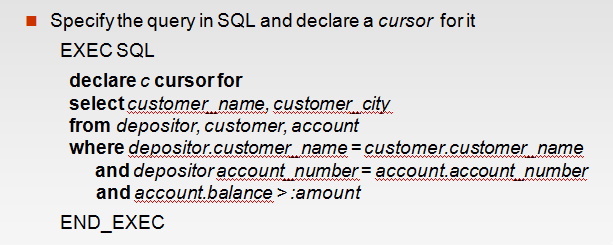
***balance* numeric(12,2))**

**SQL Procedures**

****

**declare *a\_count* integer;**

**call *account\_count\_proc*( ‘Smith’, *a\_count*);**

****

**ODBC**

**Open DataBase Connectivity(ODBC) standard**

**standard for application program to communicate with a database server.**

**application program interface (API) to**

* + - **open a connection with a database,**
    - **send queries and updates,**
    - **get back results.**

**Applications such as GUI, spreadsheets, etc. can use ODBC**

**JDBC**

**JDBC is a Java API for communicating with database systems supporting SQL**

**JDBC supports a variety of features for querying and updating data, and for retrieving query results**

**JDBC also supports metadata retrieval, such as querying about relations present in the database and the names and types of relation attributes**

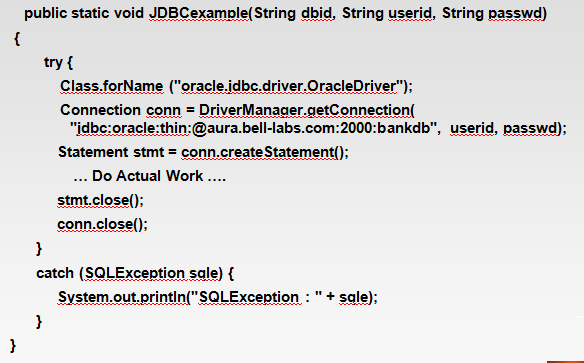
**Model for communicating with the database:**

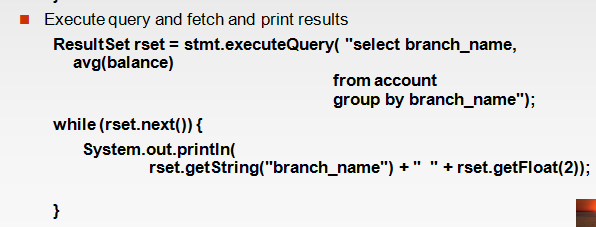
**Open a connection**

**Create a “statement” object**

**Execute queries using the Statement object to send queries and fetch results**

**Exception mechanism to handle errors**

****

****