David M. Kroenke and David J. Auer
Database Processing:
Fundamentals, Design, and Implementation

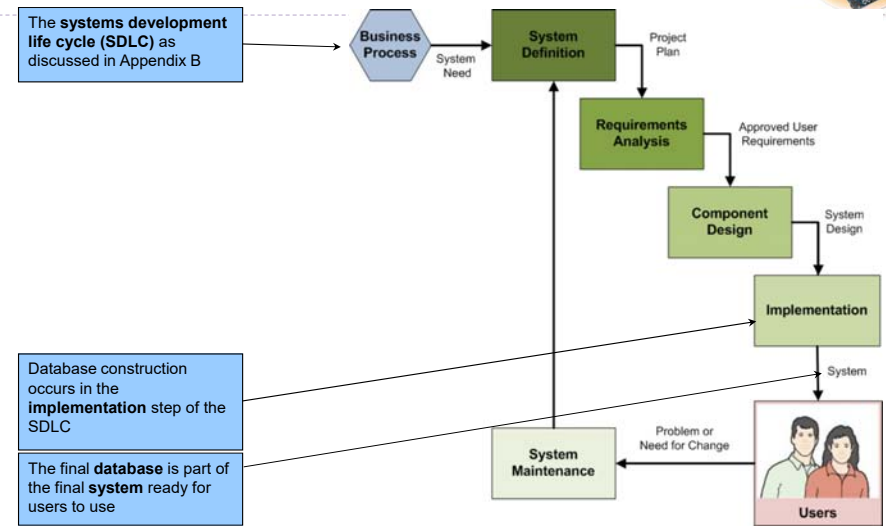
Chapter Seven:
SQL for Database Construction and Application Processing

Chapter Objectives

- To create and manage table structures using SQL statements
- To understand how referential integrity actions are implemented in SQL statements
- To create and use SQL constraints
- To understand several uses for SQL views
- To use SQL statements to create and use views
- To understand how SQL is used in an application programming
- To understand SQL/Persistent Stored Modules (SQL/PSM)
- To understand how to create and use functions
- To understand how to create and use triggers
- To understand how to create and use stored procedures

Data Modeling in the SDLC

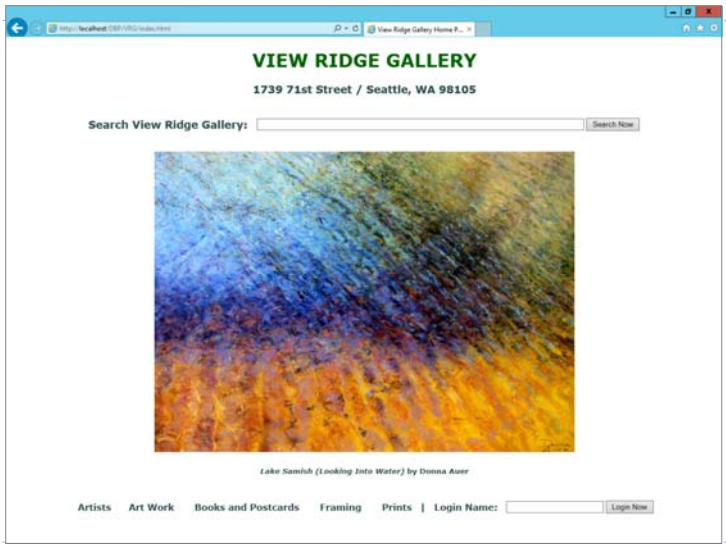
The **systems development life cycle (SDLC)** as discussed in Appendix B



Database construction occurs in the **implementation** step of the SDLC

The final **database** is part of the final **system** ready for users to use

View Ridge Gallery



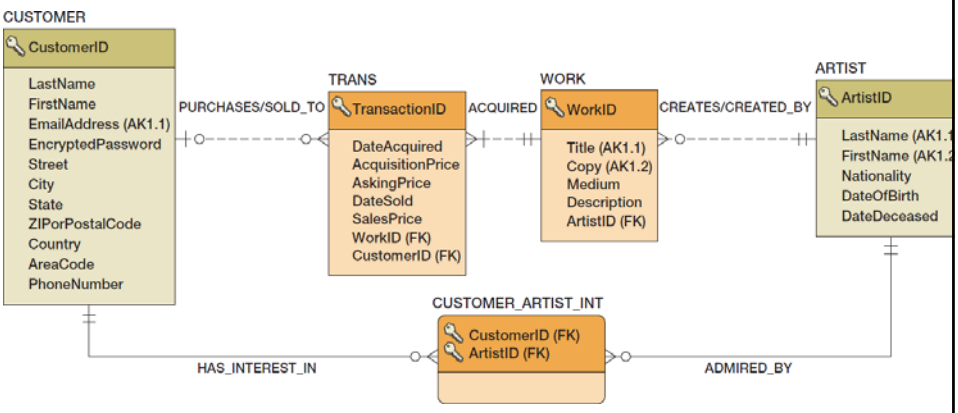
View Ridge Gallery

- View Ridge Gallery is a small art gallery that has been in business for 30 years.
- It sells contemporary European and North American fine art.
- View Ridge has one owner, three salespeople, and two workers.
- View Ridge owns all of the art that it sells; it holds no items on a consignment basis.

VRG Application Requirements

Summary of View Ridge Gallery Database Requirements
Track customers and their interest in specific artists
Record the gallery's purchases
Record customer's purchases
Report how fast an artist's works have sold and at what margin
Show the artists represented by the gallery on a Web page
Show current inventory on a Web page
Show all the works of art that have appeared in the gallery on Web pages

VRG Database Design



Minimum Cardinality Enforcement: VRG Database Relationships

Relationship		Cardinality		
Parent	**Child**	**Type**	**MAX**	**MIN**
ARTIST	WORK	Nonidentifying	1:N	M-O
WORK	TRANS	Nonidentifying	1:N	M-M
CUSTOMER	TRANS	Nonidentifying	1:N	O-O
CUSTOMER	CUSTOMER_ARTIST_INT	Identifying	1:N	M-O
ARTIST	CUSTOMER_ARTIST_INT	Identifying	1:N	M-O

VRG Database Available Online I

- Versions of the complete VRG database are available in the downloadable Student Files available at:

  http://www.pearsonhighered.com/kroenke/

- These include versions for:
  - Microsoft Access 2013
  - Microsoft SQL Server 2014
  - Oracle Database 12c and Oracle Database XE
  - MySQL 5.6
- **We recommend you actually run all material in a live database!**

VRG Database Available Online II

- To complete setting up the VRG database, set the referenced materials:
  - For Microsoft SQL Server 2014:
    - See Online Chapter 10A
  - For Oracle Database 12c and Oracle Database XE:
    - See Online Chapter 10B
  - For MySQL 5.6
    - See Online Chapter 10C
- Online chapters 10A, 10B, and 10C are available for download at:

  http://www.pearsonhighered.com/kroenke/

SQL Categories

- SQL statements can be divided into five categories:
  - **Data definition language (DDL)**
  - **Data manipulation language (DML)** statements
  - **SQL/Persistent Stored Modules (SQL/PSM)** statements
  - **Transaction control language (TCL)** statements
  - **Data control language (DCL)** statements

SQL DDL

- **Data definition language (DDL)** statements
  - Used for creating tables, relationships, and other structures
  - Covered in this chapter (Chapter 7)

SQL DML

- **Data manipulation language (DML)** statements
  - Used for:
    - Queries – SQL **SELECT** statement
    - Inserting data – SQL **INSERT** statement
    - Modifying data – SQL **UPDATE** statement
    - Deleting data – SQL **DELETE** statement
  - Previously covered in Chapter 2

SQL SQL/PSM

- **SQL/Persistent Stored Modules (SQL/PSM)** statements
  - Add procedural programming capabilities
    - Variables
    - Control-of-flow statements
  - Covered in Chapters:
    - This chapter (Chapter 7) [general introduction]
    - 10A (SQL Server 2014)
    - 10B (Oracle Database)
    - 10C (MySQL 5.6)

SQL TCL

- **Transaction control language (TCL)** statements
  - Used to mark transaction boundaries and control transaction behavior
  - Covered in Chapters:
    - 9 (general introduction)
    - 10A (SQL Server 2014)
    - 10B (Oracle Database)
    - 10C (MySQL 5.6)

SQL DCL

- **Data control language (DCL)** statements
  - Used to grant (or revoke) database permissions to (from) users and groups
  - Covered in Chapters:
    - 9 (general introduction)
    - 10A (SQL Server 2014)
    - 10B (Oracle Database)
    - 10C (MySQL 5.6)

Chapter 7 SQL Elements

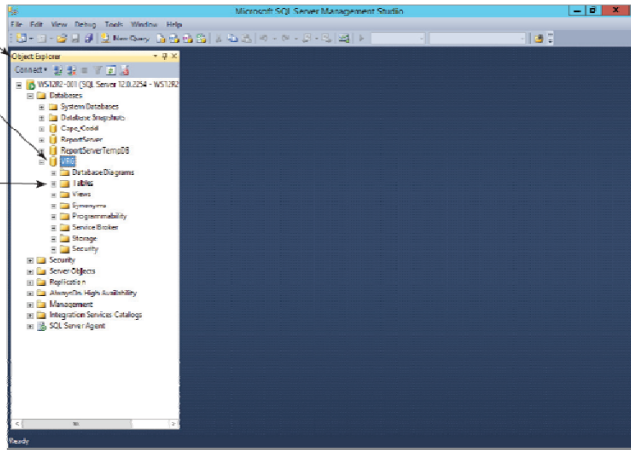
SQL Elements Discussed in Chapter 7
• SQL Data Definition Language (DDL)
– CREATE TABLE
– ALTER TABLE
– DROP TABLE
– TRUNCATE TABLE
• SQL Data Manipulation Language (DML)
– INSERT
– UPDATE
– DELETE
– MERGE
• SQL Views
– CREATE VIEW
– ALTER VIEW
– DROP VIEW
• SQL/Persistent Stored Modules (SQL/PSM)
– Functions
– Triggers
– Stored Procedures

Creating the VRG Database

The **Object Explorer**

The **VRG database** object

The VRG database object folders—when database objects such as tables are created, they will be visible in these folders



SQL CREATE TABLE Statement

- CREATE TABLE statement is used for creating relations.
- Each column is described with three parts: column name, data type, and optional constraints.
- Format:

```
CREATE TABLE NewTableName (
    ColumnName    DataType    OptionalConstraint,
    ColumnName    DataType    OptionalConstraint,
    ...
    Optional table constraint
    ...
    );
```


Column and Table Constraints

- Constraints can be defined within the CREATE TABLE statement, or they can be added to the table after it is created using the ALTER table statement.
- **Column and table constraints** include:
  - **PRIMARY KEY** – may not have NULL values
  - **FOREIGN KEY** – may not have NULL values
  - **NULL / NOT NULL**
  - **UNIQUE**
  - **CHECK**
- The **DEFAULT** keyword (not a constraint)

SQL CREATE TABLE Statement Example I

Column Characteristics:

ARTIST

Column Name	Type	Key	NULL Status	Remarks
ArtistID	Int	Primary Key	NOT NULL	Surrogate Key IDENTITY (1,1)
LastName	Char (25)	Alternate Key	NOT NULL	AK1.1
FirstName	Char (25)	Alternate Key	NOT NULL	AK1.2
Nationality	Char (30)	No	NULL	
DateOfBirth	Numeric (4,0)	No	NULL	
DateDeceased	Numeric (4,0)	No	NULL	

SQL CREATE TABLE Statement Example II

SQL CREATE TABLE statement:

```
CREATE TABLE ARTIST (
      ArtistID            Int              NOT NULL IDENTITY(1,1),
      LastName            Char(25)         NOT NULL,
      FirstName           Char(25)         NOT NULL,
      Nationality         Char(30)         NULL,
      DateOfBirth         Numeric(4,0)     NULL,
      DateDeceased        Numeric(4,0)     NULL,
      CONSTRAINT       ArtistPK         PRIMARY KEY(ArtistID),
      CONSTRAINT       ArtistAK1        UNIQUE(LastName, FirstName)
      );
```

Creating Relationships I

Relationship		Cardinality		
Parent	Child	Type	MAX	MIN
ARTIST	WORK	Nonidentifying	1:N	M-O

Creating Relationships II

ARTIST Is Required Parent	Action on ARTIST (Parent)	Action on WORK (Child)
Insert	None	Get a parent
Modify key or Foreign key	Prohibit—ARTIST uses a surrogate key	Allow foreign key updates if parent primary key exists
Delete	Prohibit if WORK exists—data related to a transaction is never deleted (business rule) Allow if no WORK exists (business rule)	None

Creating Relationships III

```
CREATE TABLE ARTIST (
       ArtistID            Int              NOT NULL IDENTITY(1,1),
       LastName            Char(25)         NOT NULL,
       FirstName           Char(25)         NOT NULL,
       Nationality         Char(30)         NULL,
       DateOfBirth         Numeric(4,0)     NULL,
       DateDeceased        Numeric(4,0)     NULL,
       CONSTRAINT       ArtistPK          PRIMARY KEY(ArtistID),
       CONSTRAINT       ArtistAK1         UNIQUE(LastName, FirstName)
       );

CREATE TABLE WORK (
       WorkID              Int              NOT NULL IDENTITY(500,1),
       Title               Char(35)         NOT NULL,
       Copy                Char(12)         NOT NULL,
       Medium              Char(35)         NULL,
       [Description]       Varchar(1000)    NULL DEFAULT 'Unknown provenance',
       ArtistID            Int              NOT NULL,
       CONSTRAINT       WorkPK            PRIMARY KEY(WorkID),
       CONSTRAINT       WorkAK1           UNIQUE(Title, Copy),
       CONSTRAINT       ArtistFK          FOREIGN KEY(ArtistID)
                           REFERENCES ARTIST(ArtistID)
                              ON UPDATE NO ACTION
                              ON DELETE NO ACTION
       );
```

Implementing Cardinalities

Relationship Type	CREATE TABLE Constraints
1:N relationship, parent optional	Specify FOREIGN KEY constraint. Set foreign key NULL.
1:N relationship, parent required	Specify FOREIGN KEY constraint. Set foreign key NOT NULL.
1:1 relationship, parent optional	Specify FOREIGN KEY constraint. Specify foreign key UNIQUE constraint. Set foreign key NULL.
1:1 relationship, parent required	Specify FOREIGN KEY constraint. Specify foreign key UNIQUE constraint. Set foreign key NOT NULL.
Casual relationship	Create a foreign key column, but do not specify FOREIGN KEY constraint. If relationship is 1:1, specify foreign key UNIQUE.

Default Values and Data Constraints

Table	Column	Default Value	Constraint
WORK	Description	'Unknown provenance'	
ARTIST	Nationality		IN ('Candian', 'English', 'French', 'German', 'Mexican', 'Russian', 'Spainish', 'United States'.
ARTIST	DateOfBirth		Less than DateDeceased.
ARTIST	DateOfBirth		Four digits—1 or 2 is first digit, 0 to 9 for remaining three digits.
ARTIST	DateDeceased		Four digits—1 or 2 is first digit, 0 to 9 for remaining three digits.
TRANS	SalesPrice		Greater than 0 and less than or equal to 500,000.
TRANS	DateAcquired		Less than or equal to DateSold.

SQL for Constraints

```
CREATE TABLE ARTIST (
       ArtistID            Int              NOT NULL IDENTITY(1,1),
       LastName            Char(25)         NOT NULL,
       FirstName           Char(25)         NOT NULL,
       Nationality         Char(30)         NULL,
       DateOfBirth         Numeric(4,0)     NULL,
       DateDeceased        Numeric(4,0)     NULL,
       CONSTRAINT       ArtistPK         PRIMARY KEY(ArtistID),
       CONSTRAINT       ArtistAK1        UNIQUE(LastName, FirstName),
       CONSTRAINT       NationalityValues   CHECK
                           (Nationality IN ('Canadian', 'English', 'French',
                            'German', 'Mexican', 'Russian', 'Spanish',
                            'United States')),
       CONSTRAINT       BirthValuesCheck    CHECK (DateOfBirth < DateDeceased),
       CONSTRAINT       ValidBirthYear      CHECK
                           (DateOfBirth LIKE '[1-2][0-9][0-9][0-9]'),
       CONSTRAINT       ValidDeathYear      CHECK
                           (DateDeceased LIKE '[1-2][0-9][0-9][0-9]')
       );

CREATE TABLE WORK (
       WorkID              Int              NOT NULL IDENTITY(500,1),
       Title               Char(35)         NOT NULL,
       Copy                Char(12)         NOT NULL,
       Medium              Char(35)         NULL,
       [Description]       Varchar(1000)    NULL DEFAULT 'Unknown provenance',
       ArtistID            Int              NOT NULL,
       CONSTRAINT       WorkPK           PRIMARY KEY(WorkID)
```

SQL for Other VRG Tables I

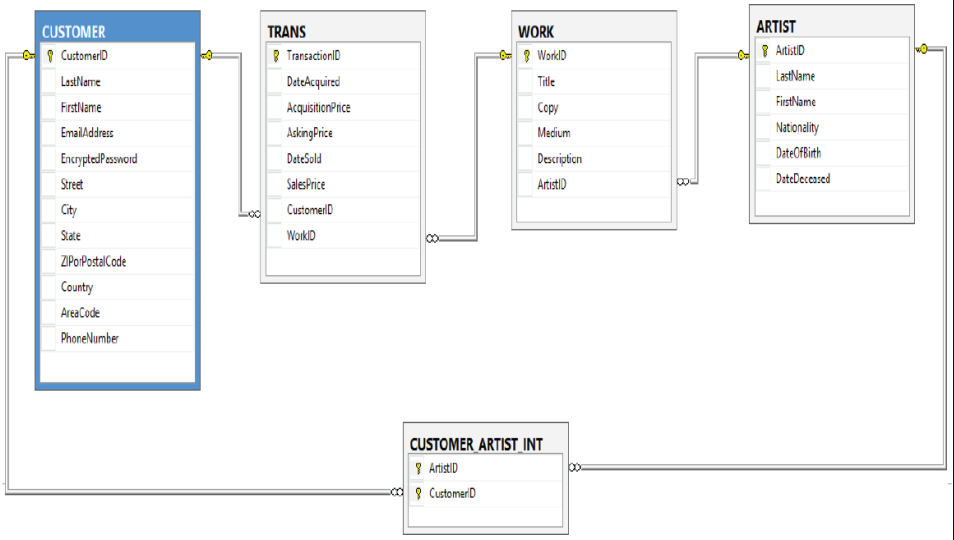
```
CREATE TABLE CUSTOMER (
       CustomerID              Int                 NOT NULL IDENTITY(1000,1),
       LastName                Char(25)            NOT NULL,
       FirstName               Char(25)            NOT NULL,
       EmailAddress            Varchar(100)        NULL,
       EncryptedPassword       VarChar(50)         NULL,
       Street                  Char(30)            NULL,
       City                    Char(35)            NULL,
       [State]                 Char(2)             NULL,
       ZIPorPostalCode         Char(9)             NULL,
       Country                 Char(50)            NULL,
       AreaCode                Char(3)             NULL,
       PhoneNumber             Char(8)             NULL,
       CONSTRAINT          CustomerPK          PRIMARY KEY(CustomerID),
       CONSTRAINT          EmailAK1            UNIQUE(EmailAddress)
       );
```


SQL for Other VRG Tables II

```
CREATE TABLE TRANS (
       TransactionID           Int                 NOT NULL IDENTITY(100,1),
       DateAcquired            Date                NOT NULL,
       AcquisitionPrice        Numeric(8,2)        NOT NULL,
       AskingPrice             Numeric(8,2)        NULL,
       DateSold                Date                NULL,
       SalesPrice              Numeric(8,2)        NULL,
       CustomerID              Int                 NULL,
       WorkID                  Int                 NOT NULL,
       CONSTRAINT          TransPK             PRIMARY KEY(TransactionID),
       CONSTRAINT          TransWorkFK         FOREIGN KEY(WorkID)
                              REFERENCES WORK(WorkID)
                                 ON UPDATE NO ACTION
                                 ON DELETE NO ACTION,
       CONSTRAINT          TransCustomerFK FOREIGN KEY(CustomerID)
                              REFERENCES CUSTOMER(CustomerID)
                                 ON UPDATE NO ACTION
                                 ON DELETE NO ACTION,
       CONSTRAINT          SalesPriceRange     CHECK
                              ((SalesPrice > 0) AND (SalesPrice <=500000)),
       CONSTRAINT          ValidTransDate      CHECK (DateAcquired <= DateSold)
       );

CREATE TABLE CUSTOMER_ARTIST_INT(
       ArtistID                Int                 NOT NULL,
       CustomerID              Int                 NOT NULL,
       CONSTRAINT          CAIntPK             PRIMARY KEY(ArtistID, CustomerID),
       CONSTRAINT          CAInt_ArtistFK      FOREIGN KEY(ArtistID)
                              REFERENCES ARTIST(ArtistID)
                                 ON UPDATE NO ACTION
```

Microsoft SQL Server 2014
VRG Database Diagram



SQL ALTER TABLE Statement

- The **SQL ALTER TABLE statement** changes table structure, properties, or constraints after it has been created.
- Example

```
ALTER TABLE ASSIGNMENT
  ADD CONSTRAINT EmployeeFK
   FOREIGN KEY (EmployeeNumber)
         REFERENCES EMPLOYEE (EmployeeNumber)
               ON UPDATE CASCADE
               ON DELETE NO ACTION;
```

Adding and Dropping Columns

- The following statement will add a column named MyColumn to the CUSTOMER table:
  - Note that the **SQL COLUMN keyword** is ***not*** used!

```
/* *** SQL-ALTER-TABLE-CH07-01 *** */
ALTER TABLE CUSTOMER
    ADD MyColumn Char(5) NULL;
```

- You can drop an existing column with the statement:

```
/* *** SQL-ALTER-TABLE-CH07-02 *** */
ALTER TABLE CUSTOMER
    DROP COLUMN MyColumn;
```


Adding and Dropping Constraints

- The **SQL ALTER TABLE statement** can be used to add a constraint:

```
/* *** SQL-ALTER-TABLE-CH07-03 *** */
ALTER TABLE CUSTOMER
    ADD CONSTRAINT MyConstraint CHECK
        (LastName NOT IN ('RobertsNoPay'));
```

- The **SQL ALTER TABLE statement** can be used to drop a constraint:

```
/* *** SQL-ALTER-TABLE-CH07-04 *** */
ALTER TABLE CUSTOMER
    DROP CONSTRAINT MyConstraint;
```

Removing Tables I

- The **SQL DROP TABLE statement**:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-DROP-TABLE-CH07-01 *** */
```

-]

```
DROP TABLE TRANS;
ALTER TABLE CUSTOMER_ARTIST_INT
    DROP CONSTRAINT
    Customer_Artist_Int_CustomerFK;
ALTER TABLE TRANS
    DROP CONSTRAINT  TransactionCustomerFK;
DROP TABLE CUSTOMER;
```


Removing Tables II

- If there are constraints:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-DROP-TABLE-CH07-02 *** */
DROP TABLE CUSTOMER_ARTIST_INT;
DROP TABLE TRANS;
DROP TABLE CUSTOMER;
```

- or

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-ALTER-TABLE-CH07-05 *** */
ALTER TABLE CUSTOMER_ARTIST_INT
    DROP CONSTRAINT Customer_Artist_Int_CustomerFK;
ALTER TABLE TRANS
    DROP CONSTRAINT TransactionCustomerFK;
/* *** SQL-DROP-TABLE-CH07-03 *** */
DROP TABLE CUSTOMER;
```

Removing Data Only

- The **SQL TRUNCATE TABLE statement**:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-TRUNCATE-TABLE-CH07-01 *** */
TRUNCATE TABLE CUSTOMER_ARTIST_INT;
```

- Cannot be used with a table that is referenced by a foreign key constraint.
- Resets surrogate key values to initial value.

SQL DDL—CREATE INDEX

- An index is a data structure used to improve database performance.
- The **SQL CREATE INDEX statement**
- The **SQL ALTER INDEX statement**
- The **SQL DROP INDEX statement**
- See:
  - Chapter 10A - Microsoft SQL Server 2014
  - Chapter 10B - Oracle Database
  - Chapter 10C - MySQL 5.6