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

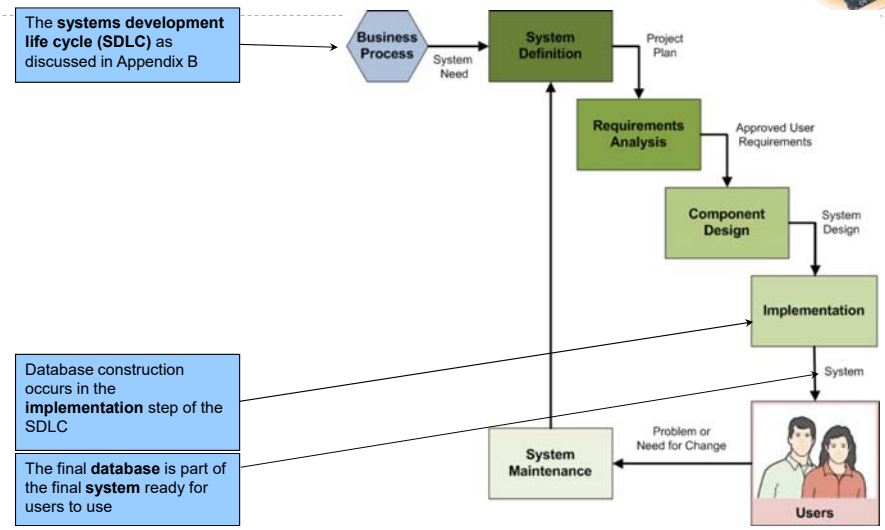
Chapter Seven:
SQL for Database Construction and Application Processing

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

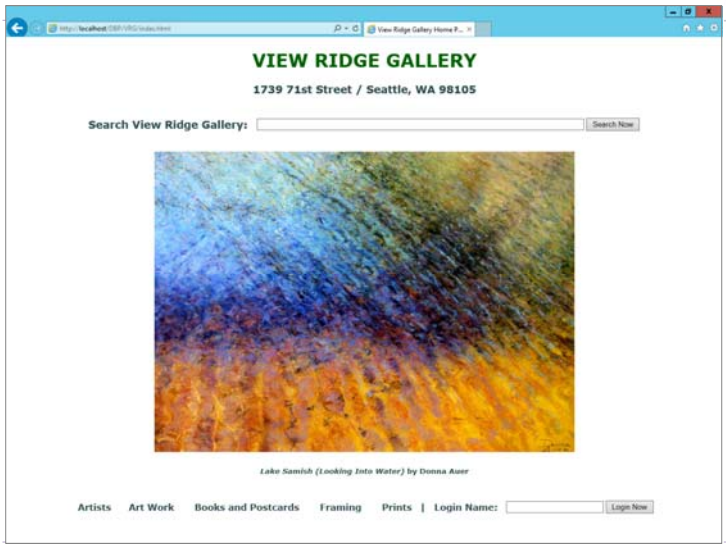
The **systems development life cycle (SDLC)** as discussed in Appendix B



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



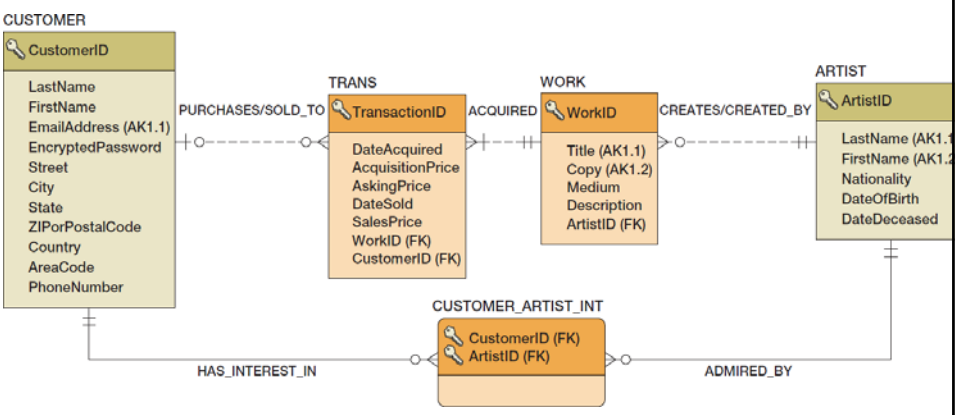
View Ridge Gallery

- 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 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

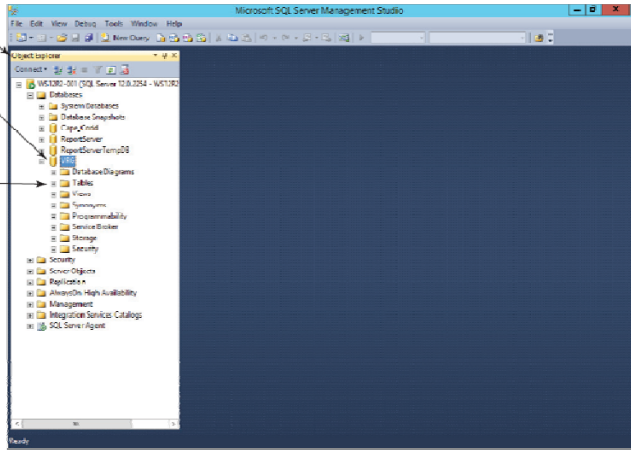
SQL Elements Discussed in Chapter 7
• SQL Data Definition Language (DDL)
– CREATE TABLE
– ALTER TABLE
– DROP TABLE
– TRUNCATE TABLE
• SQL Data Manipulation Language (DML)
– INSERT
– UPDATE
– DELETE
– MERGE
• SQL Views
– CREATE VIEW
– ALTER VIEW
– DROP VIEW
• SQL/Persistent Stored Modules (SQL/PSM)
–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

SQL for Other VRG Tables I

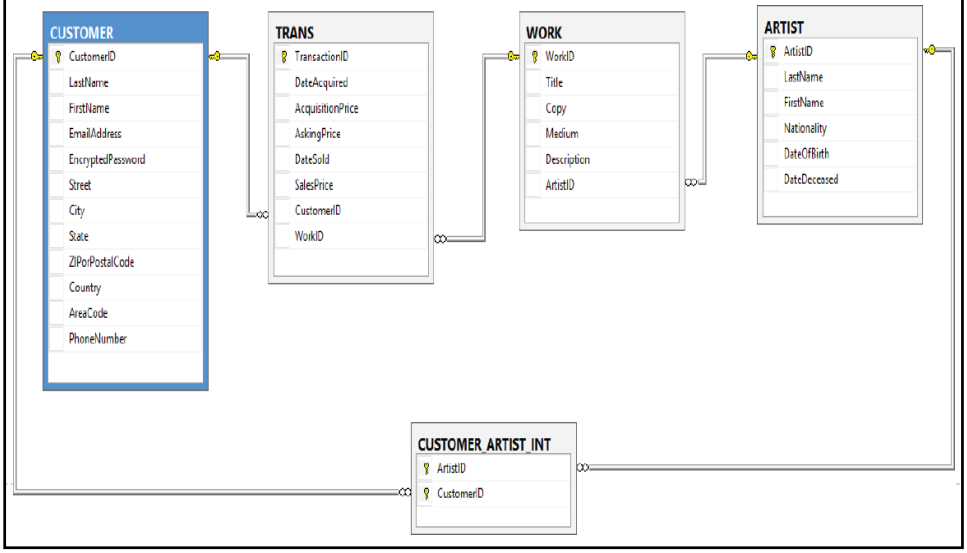
```
CREATE TABLE CUSTOMER (
       CustomerID           Int              NOT NULL IDENTITY(1000,1),
       LastName             Char(25)         NOT NULL,
       FirstName            Char(25)         NOT NULL,
       EmailAddress         Varchar(100)     NULL,
       EncryptedPassword    VarChar(50)      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

SQL DML—INSERT I

- The **SQL INSERT statement**:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-INSERT-CH07-01 *** */
INSERT INTO ARTIST
     (LastName, FirstName, Nationality, DateOfBirth, DateDeceased)
     VALUES ('Miro', 'Joan', 'Spanish', 1893, 1983);
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-INSERT-CH07-02 *** */
INSERT INTO ARTIST VALUES
     ('Miro', 'Joan', 'Spanish', 1893, 1983);
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-INSERT-CH07-04 *** */
INSERT INTO ARTIST
     (LastName, FirstName, Nationality)
     VALUES ('Miro', 'Joan', 'Spanish');
```

SQL DML—INSERT II

- Bulk INSERT:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-INSERT-CH07-05 *** */
INSERT INTO ARTIST
     (LastName, FirstName, Nationality, DateOfBirth, DateDeceased)
     SELECT      LastName, FirstName, Nationality,
                 DateOfBirth, DateDeceased
     FROM        IMPORTED_ARTIST;
```


Populating the VRG Tables I

- The VRG database data contain **non-sequential surrogate key values.**

ArtistID	LastName	FirstName	Nationality	DateOfBirth	DateDeceased
1	Miro	Joan	Spanish	1893	1983
2	Kandinsky	Wassily	Russian	1866	1944
3	Klee	Paul	German	1879	1940
4	Matisse	Henri	French	1869	1954
5	Chagall	Marc	French	1887	1985
11	Sargent	John Singer	United States	1856	1925
17	Tobey	Mark	United States	1890	1976
18	Horiuchi	Paul	United States	1906	1999
19	Graves	Morris	United States	1920	2001

Populating the VRG Tables II

- **Cannot** just use SQL INSERT statement by itself.
- See discussions of how to handle this situation:
  - Chapter 10A - Microsoft SQL Server 2014
  - Chapter 10B - Oracle Database
  - Chapter 10C - MySQL 5.6

SQL DML—UPDATE I

- The **SQL UPDATE statement**:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-UPDATE-CH07-01 *** */
UPDATE        CUSTOMER
    SET         City = 'New York City'
    WHERE       CustomerID = 1000;
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-UPDATE-CH07-02 *** */
UPDATE        CUSTOMER
    SET         City = 'New York City', State = 'NY'
    WHERE       CustomerID = 1000;
```

SQL DML—UPDATE II

- Bulk UPDATE:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-UPDATE-CH07-03 *** */
UPDATE        CUSTOMER
    SET         City = 'New York City';
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-UPDATE-CH07-04 *** */
UPDATE        CUSTOMER
    SET         AreaCode = '303'
    WHERE       City = 'Denver';
```


SQL DML—UPDATE III

- Using values from other tables:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-UPDATE-CH07-06 *** */
UPDATE          PURCHASE_ORDER
    SET         TaxRate =
                (SELECT   Tax
                 From     TAX_TABLE
                 WHERE    TAX_TABLE.City = PURCHASE_ORDER.City)
    WHERE       PURCHASE_ORDER.Number = 1000;
```

SQL DML—MERGE

- The **SQL MERGE statement**:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-MERGE-CH07-01 *** */
MERGE INTO ARTIST AS A USING ARTIST_DATA_RESEARCH AS ADR
    ON   (A.LastName = ADR.LastName
          AND
          A.FirstName = ADR.FirstName)
WHEN MATCHED THEN
    UPDATE SET
         A.Nationality = ADR.Nationality,
         A.DateOfBirth = ADR.DateOfBirth,
         A.DateDeceased = ADR.DateDeceased
WHEN NOT MATCHED THEN
    INSERT (LastName, FirstName, Nationality,
            DateOfBirth, DateDeceased);
```


SQL DML—DELETE

- SQL DELETE statement:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-DELETE-CH07-01 *** */
DELETE       FROM CUSTOMER
WHERE        CustomerID = 1000;
```

- If you omit the WHERE clause, you will delete every row in the table.
- Does *not* reset surrogate key values.

Using Aliases

- **Use of aliases:**

```
SELECT   C.Name, A.Name
FROM     CUSTOMER AS C
         JOIN
         CUSTOMER_ARTIST_INT AS CI
  ON     C.CustomerID = CI.CustomerID
         JOIN ARTIST AS A
               ON CI.ArtistID = A.ArtistID;
```

- DBMS products differ

  `CUSTOMER AS C` versus `CUSTOMER C`

VRG Database Data

CUSTOMER I

CustomerID	LastName	FirstName	EmailAddress	EncryptedPassword
1000	Janes	Jeffrey	Jeffrey.Janes@somewhere.com	ng76tG9E
1001	Smith	David	David.Smith@somewhere.com	ttr67i23
1015	Twilight	Tiffany	Tiffany.Twilight@somewhere.com	gr44t5uz
1033	Smathers	Fred	Fred.Smathers@somewhere.com	mnF3D00Q
1034	Frederickson	Mary Beth	MaryBeth.Frederickson@somewhere.com	Nd5qr4Tv
1036	Warning	Selma	Selma.Warning@somewhere.com	CAe3Gh98
1037	Wu	Susan	Susan.Wu@somewhere.com	Ues3thQ2
1040	Gray	Donald	Donald.Gray@somewhere.com	NULL
1041	Johnson	Lynda	NULL	NULL
1051	Wilkens	Chris	Chris.Wilkens@somewhere.com	45QZjx59

VRG Database Data

CUSTOMER II

CustomerID	LastName	FirstName	Street	City	State	ZIPorPostalCode
1000	Janes	Jeffrey	123 W. Elm St	Renton	WA	98055
1001	Smith	David	813 Tumbleweed Lane	Loveland	CO	81201
1015	Twilight	Tiffany	88 1st Avenue	Langley	WA	98260
1033	Smathers	Fred	10899 88th Ave	Bainbridge Island	WA	98110
1034	Frederickson	Mary Beth	25 South Lafayette	Denver	CO	80201
1036	Warning	Selma	205 Burnaby	Vancouver	BC	V6Z 1W2
1037	Wu	Susan	105 Locust Ave	Atlanta	GA	30322
1040	Gray	Donald	55 Bodega Ave	Bodega Bay	CA	94923
1041	Johnson	Lynda	117 C Street	Washington	DC	20003
1051	Wilkens	Chris	87 Highland Drive	Olympia	WA	98508

VRG Database Data

CUSTOMER III

CustomerID	LastName	FirstName	Country	AreaCode	PhoneNumber
1000	Janes	Jeffrey	USA	425	543-2345
1001	Smith	David	USA	970	654-9876
1015	Twilight	Tiffany	USA	360	765-5566
1033	Smathers	Fred	USA	206	876-9911
1034	Frederickson	Mary Beth	USA	303	513-8822
1036	Warning	Selma	Canada	604	988-0512
1037	Wu	Susan	USA	404	653-3465
1040	Gray	Donald	USA	707	568-4839
1041	Johnson	Lynda	USA	202	438-5498
1051	Wilkens	Chris	USA	360	876-8822

VRG Database Data

ARTIST

ArtistID	LastName	FirstName	Nationality	DateOfBirth	DateDeceased
1	Miro	Joan	Spanish	1893	1983
2	Kandinsky	Wassily	Russian	1866	1944
3	Klee	Paul	German	1879	1940
4	Matisse	Henri	French	1869	1954
5	Chagall	Marc	French	1887	1985
11	Sargent	John Singer	United States	1856	1925
17	Tobey	Mark	United States	1890	1976
18	Horiuchi	Paul	United States	1906	1999
19	Graves	Morris	United States	1920	2001

VRG Database Data

CUSTOMER_ARTIST_INT

ArtistID	CustomerID
1	1001
1	1034
2	1001
2	1034
4	1001
4	1034
5	1001
5	1034
5	1036
11	1001
11	1015
11	1036
17	1000
17	1015

ArtistID	CustomerID
17	1033
17	1040
17	1051
18	1000
18	1015
18	1033
18	1040
18	1051
19	1000
19	1015
19	1033
19	1036
19	1040
19	1051

VRG Database Data

WORK I

WorkID	Title	Medium	Description	Copy	ArtistID
500	Memories IV	Casein rice paper collage	31 × 24.8 in.	Unique	18
511	Surf and Bird	High Quality Limited Print	Northwest School Expressionist style	142/500	19
521	The Tilled Field	High Quality Limited Print	Early Surrealist style	788/1000	1
522	La Lecon de Ski	High Quality Limited Print	Surrealist style	353/500	1
523	On White II	High Quality Limited Print	Bauhaus style of Kandinsky	435/500	2
524	Woman with a Hat	High Quality Limited Print	A very colorful Impressionist piece	596/750	4
537	The Woven World	Color lithograph	Signed	17/750	17
548	Night Bird	Watercolor on Paper	50 × 72.5 cm.—Signed	Unique	19
551	Der Blaue Reiter	High Quality Limited Print	"The Blue Rider"—Early Pointilism influence	236/1000	2
552	Angelus Novus	High Quality Limited Print	Bauhaus style of Klee	659/750	3
553	The Dance	High Quality Limited Print	An Impressionist masterpiece	734/1000	4
554	I and the Village	High Quality Limited Print	Shows Belarusian folk-life themes and symbology	834/1000	5
555	Claude Monet Painting	High Quality Limited Print	Shows French Impressionist influence of Monet	684/1000	11
561	Sunflower	Watercolor and ink	33.3 × 16.1 cm.—Signed	Unique	19
562	The Fiddler	High Quality Limited Print	Shows Belarusian folk-life themes and symbology	251/1000	5
563	Spanish Dancer	High Quality Limited Print	American realist style—From work in Spain	583/750	11
564	Farmer's Market #2	High Quality Limited Print	Northwest School Abstract Expressionist style	267/500	17

VRG Database Data

WORK II

WorkID	Title	Medium	Description	Copy	ArtistID
565	Farmer's Market #2	High Quality Limited Print	Northwest School Abstract Expressionist style	268/500	17
566	Into Time	High Quality Limited Print	Northwest School Abstract Expressionist style	323/500	18
570	Untitled Number 1	Monotype with tempera	4.3 × 6.1 in.—Signed	Unique	17
571	Yellow covers blue	Oil and collage	71 × 78 in.—Signed	Unique	18
578	Mid Century Hibernation	High Quality Limited Print	Northwest School Expressionist style	362/500	19
580	Forms in Progress I	Color aquatint	19.3 × 24.4 in.—Signed	Unique	17
581	Forms in Progress II	Color aquatint	19.3 × 24.4 in.—Signed	Unique	17
585	The Fiddler	High Quality Limited Print	Shows Belarusian folk-life themes and symbology	252/1000	5
586	Spanish Dancer	High Quality Limited Print	American Realist style—From work in Spain	588/750	11
587	Broadway Boggie	High Quality Limited Print	Northwest School Abstract Expressionist style	433/500	17
588	Universal Field	High Quality Limited Print	Northwest School Abstract Expressionist style	114/500	17
589	Color Floating in Time	High Quality Limited Print	Northwest School Abstract Expressionist style	487/500	18
590	Blue Interior	Tempera on card	43.9 × 28 in.	Unique	17
593	Surf and Bird	Gouache	26.5 × 29.75 in.—Signed	Unique	19
594	Surf and Bird	High Quality Limited Print	Northwest School Expressionist style	366/500	19
595	Surf and Bird	High Quality Limited Print	Northwest School Expressionist style	366/500	19
596	Surf and Bird	High Quality Limited Print	Northwest School Expressionist style	366/500	19

VRG Database Data

TRANS I

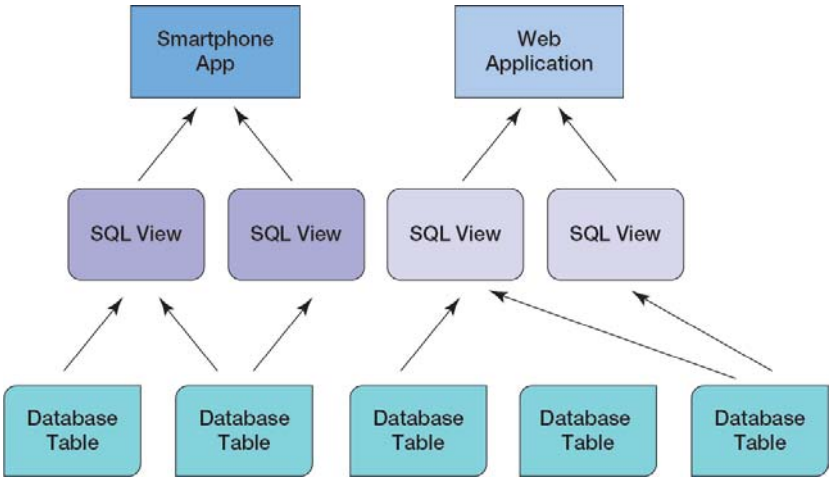
TransactionID	DateAcquired	AcquisitionPrice	AskingPrice	DateSoldID	SalesPrice	CustomerID	WorkID
100	11/4/2011	$30,000.00	$45,000.00	12/14/2011	$42,500.00	1000	500
101	11/7/2011	$250.00	$500.00	12/19/2011	$500.00	1015	511
102	11/17/2011	$125.00	$250.00	1/18/2012	$200.00	1001	521
103	11/17/2011	$250.00	$500.00	12/12/2012	$400.00	1034	522
104	11/17/2011	$250.00	$250.00	1/18/2012	$200.00	1001	523
105	11/17/2011	$200.00	$500.00	12/12/2012	$400.00	1034	524
115	3/3/2012	$1,500.00	$3,000.00	6/7/2012	$2,750.00	1033	537
121	9/21/2012	$15,000.00	$30,000.00	11/28/2012	$27,500.00	1015	548
125	11/21/2012	$125.00	$250.00	12/18/2012	$200.00	1001	551
126	11/21/2012	$200.00	$400.00	NULL	NULL	NULL	552
127	11/21/2012	$125.00	$500.00	12/22/2012	$400.00	1034	553
128	11/21/2012	$125.00	$250.00	3/16/2013	$225.00	1036	554
129	11/21/2012	$125.00	$250.00	3/16/2013	$225.00	1036	555
151	5/7/2013	$10,000.00	$20,000.00	6/28/2013	$17,500.00	1036	561
152	5/18/2013	$125.00	$250.00	8/15/2013	$225.00	1001	562
153	5/18/2013	$200.00	$400.00	8/15/2013	$350.00	1001	563
154	5/18/2013	$250.00	$500.00	9/28/2013	$400.00	1040	564
155	5/18/2013	$250.00	$500.00	NULL	NULL	NULL	565

VRG Database Data

TRANS II

TransactionID	DateAcquired	AcquisitionPrice	AskingPrice	DateSoldID	SalesPrice	CustomerID	WorkID
156	5/18/2013	$250.00	$500.00	9/27/2013	$400.00	1040	566
161	6/28/2013	$7,500.00	$15,000.00	9/29/2013	$13,750.00	1033	570
171	8/23/2013	$35,000.00	$60,000.00	9/29/2013	$55,000.00	1000	571
175	9/29/2013	$40,000.00	$75,000.00	12/18/2013	$72,500.00	1036	500
181	10/11/2013	$250.00	$500.00	NULL	NULL	NULL	578
201	2/28/2014	$2,000.00	$3,500.00	4/26/2014	$3,250.00	1040	580
202	2/28/2014	$2,000.00	$3,500.00	4/26/2014	$3,250.00	1040	581
225	6/8/2014	$125.00	$250.00	9/27/2014	$225.00	1051	585
226	6/8/2014	$200.00	$400.00	NULL	NULL	NULL	586
227	6/8/2014	$250.00	$500.00	9/27/2014	$475.00	1051	587
228	6/8/2014	$250.00	$500.00	NULL	NULL	NULL	588
229	6/8/2014	$250.00	$500.00	NULL	NULL	NULL	589
241	8/29/2014	$2,500.00	$5,000.00	9/27/2014	$4,750.00	1015	590
251	10/25/2014	$25,000.00	$50,000.00	NULL	NULL	NULL	593
252	10/27/2014	$250.00	$500.00	NULL	NULL	NULL	594
253	10/27/2014	$250.00	$500.00	NULL	NULL	NULL	595
254	10/27/2014	$250.00	$500.00	NULL	NULL	NULL	596

SQL Views I



SQL Views II

- An SQL view is a virtual table that is constructed from other tables or views.
- It has no data of its own, but obtains data from tables or other views.
- SELECT statements are used to define views:
  - A view definition may not include an ORDER BY clause.
- SQL views are a subset of the external views:
  - They can be used only for external views that involve one multivalued path through the schema.

SQL Views

Uses of SQL Views
Hide columns or rows.
Display results of computations.
Hide complicated SQL syntax.
Layer built-in functions.
Provide level of isolation between table data and users' view of data.
Assign different processing permissions to different views of the same table.
Assign different triggers to different views of the same table.

SQL CREATE VIEW Statement I

- The **SQL CREATE VIEW statement**:

```
/* *** SQL-CREATE-VIEW-CH07-01 *** */
CREATE VIEW CustomerNameView AS
    SELECT        LastName AS CustomerLastName,
                  FirstName AS CustomerFirstName
    FROM          CUSTOMER;
```

- In the SQL standard, views do *not* support the **SQL ORDER BY clause**.
  - Individual DBMS products may support the SQL ORDER BY clause – see documentation.

SQL CREATE VIEW Statement II

- To see the results, use an **SQL SELECT statement** with the **view name** as the table name in the FROM clause:

```
/* *** SQL-Query-View-CH07-01 *** */

SELECT          *

FROM            CustomerNameView

ORDER BY        CustomerLastName, Customer
```

	CustomerLastName	CustomerFirstName
1	Frederickson	Mary Beth
2	Gray	Donald
3	Janes	Jeffrey
4	Johnson	Lynda
5	Smathers	Fred
6	Smith	David
7	Twilight	Tiffany
8	Waming	Selma
9	Wilkens	Chris
10	Wu	Susan

SQL ALTER VIEW Statement I

- The **SQL ALTER VIEW statement**:

```
/* *** EXAMPLE CODE - DO NOT RUN *** */
/* *** SQL-ALTER-VIEW-CH07-01 *** */
ALTER VIEW CustomerNameView AS
    SELECT        FirstName AS CustomerFirstName,
                  LastName AS CustomerLastName,
    FROM          CUSTOMER;
```

- In the Oracle Database or MySQL 5.6, use the **SQL CREATE OR REPLACE VIEW statement**.
  - This allows creation and modification of SQL VIEW code.

Updateable Views

Updatable Views
View based on a single table with no computed columns and all non-null columns present in the view.
View based on any number of tables, with or without computed columns, and INSTEAD OF trigger defined for the view.
Possibly Updatable Views
Based on a single table, primary key in view, some required columns missing from view, update and delete may be allowed. Insert is not allowed.
Based on multiple tables, updates may be allowed on the most subordinate table in the view if rows of that table can be uniquely identified.