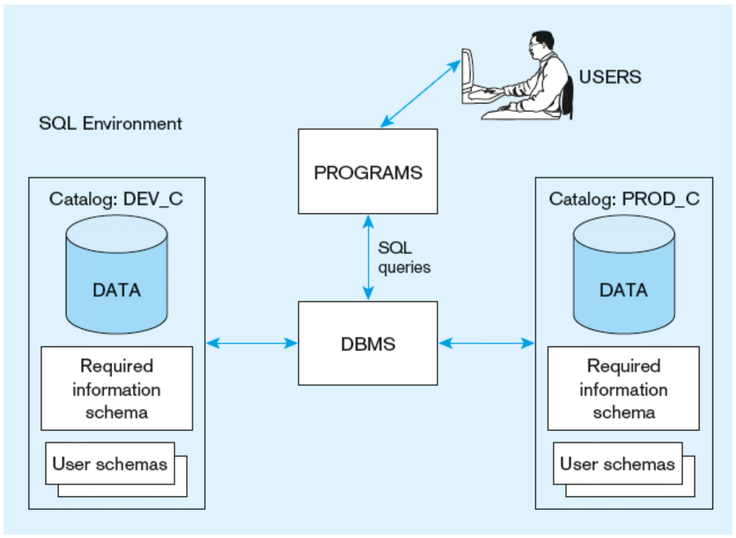
Chapter 6

Introduction to SQL: Structured Query Language

Objectives

- Define terms
- Define a database using SQL data definition language
- Write single table queries using SQL
- Establish referential integrity using SQL
- Discuss SQL:1999 and SQL:200n standards

Figure 6-1
A simplified schematic of a typical SQL environment, as described by the SQL: 200n standard



SQL Environment

- Data Definition Language (DDL)
  - Commands that define a database, including creating, altering, and dropping tables and establishing constraints
  - CREATE / DROP / ALTER, …
- Data Manipulation Language (DML)
  - Commands that maintain and query a database
  - INSERT, UPDATE, DELETE, SELECT, …
- Data Control Language (DCL)
  - Commands that control a database, including administering privileges and committing data
  - GRANT, ADD, REVOKE

SQL Database Definition

- Data Definition Language (DDL)
- Major CREATE statements:
  - CREATE SCHEMA–defines a portion of the database owned by a particular user
  - CREATE TABLE–defines a new table and its columns
  - CREATE VIEW–defines a logical table from one or more tables or views

DDL: Table Creation

General syntax for CREATE TABLE used in data definition language

```
CREATE TABLE table_name (
  field  type  constraints,
  field2 type2,
  CONSTRAINT name ...,
  ...
);
```

```
CREATE TABLE Book (
  ISBN  CHAR(9)     NOT NULL,
  Title VARCHAR(20) UNIQUE,
  Pages INTEGER,
  CONSTRAINT ISBN PRIMARY KEY
);
```

Steps in table creation:

1. Identify data types for attributes
2. Identify columns that can and cannot be null
3. Identify columns that must be unique (candidate keys)
4. Identify primary key
5. Determine default values
6. Identify constraints on columns (domain specifications)
7. Identify foreign keys

SQL Data Types

TABLE 6-2 Sample SQL Data Types

String	CHARACTER (CHAR)	Stores string values containing any characters in a character set. CHAR is defined to be a fixed length.
	CHARACTER VARYING (VARCHAR or VARCHAR2)	Stores string values containing any characters in a character set but of definable variable length.
	BINARY LARGE OBJECT (BLOB)	Stores binary string values in hexadecimal format. BLOB is defined to be a variable length. (Oracle also has CLOB and NCLOB, as well as BFILE for storing unstructured data outside the database.)
Number	NUMERIC	Stores exact numbers with a defined precision and scale.
	INTEGER (INT)	Stores exact numbers with a predefined precision and scale of zero.
Temporal	TIMESTAMP TIMESTAMP WITH LOCAL TIME ZONE	Stores a moment an event occurs, using a definable fraction-of-a-second precision.Value adjusted to the user's session time zone (available in Oracle and MySQL)
Boolean	BOOLEAN	Stores truth values: TRUE, FALSE, or UNKNOWN.

The following slides create tables for this enterprise data model

(from Chapter 1, Figure 1-3)

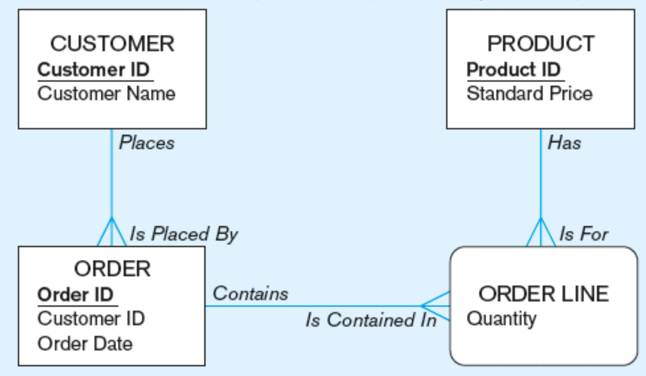
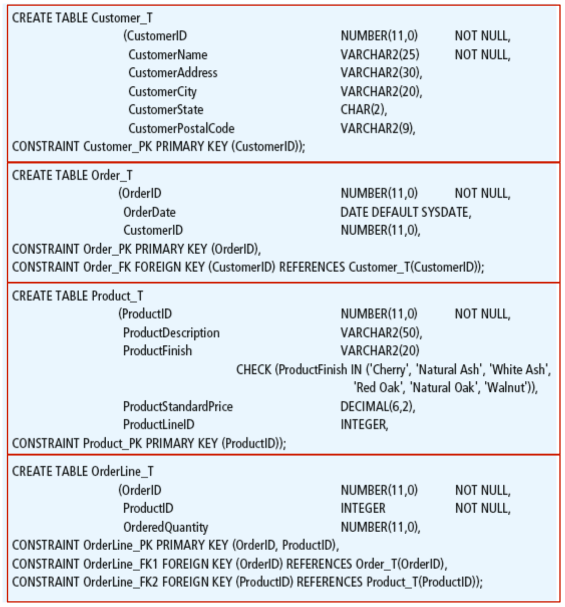
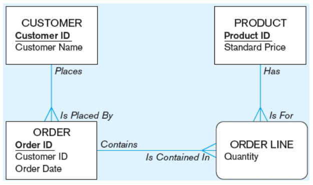


Figure 6-6 SQL database definition commands for Pine Valley Furniture Company (Oracle 11g)

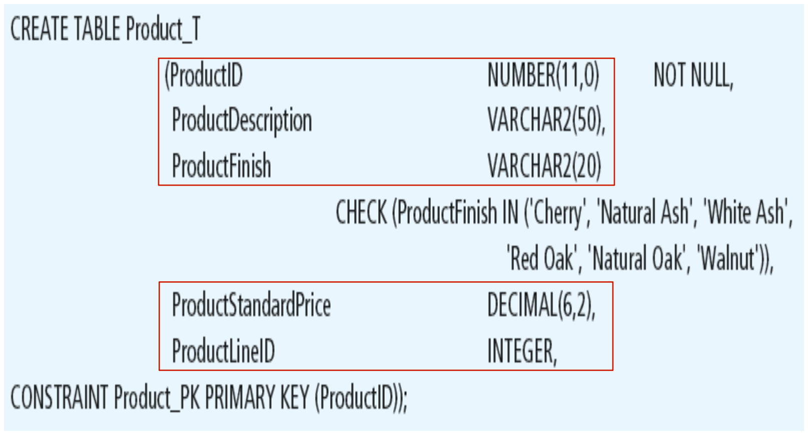


```
CREATE TABLE Customer_T
                (CustomerID                 NUMBER(11,0)    NOT NULL,
                 CustomerName               VARCHAR2(25)    NOT NULL,
                 CustomerAddress            VARCHAR2(30),
                 CustomerCity               VARCHAR2(20),
                 CustomerState              CHAR(2),
                 CustomerPostalCode         VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
                (OrderID                    NUMBER(11,0)    NOT NULL,
                 OrderDate                  DATE DEFAULT SYSDATE,
                 CustomerID                 NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));

CREATE TABLE Product_T
                (ProductID                  NUMBER(11,0)    NOT NULL,
                 ProductDescription         VARCHAR2(50),
                 ProductFinish              VARCHAR2(20)
                              CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                     'Red Oak', 'Natural Oak', 'Walnut')),
                 ProductStandardPrice       DECIMAL(6,2),
                 ProductLineID              INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
                (OrderID                    NUMBER(11,0)    NOT NULL,
                 ProductID                  INTEGER         NOT NULL,
                 OrderedQuantity            NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T(OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T(ProductID));
```

Overall table definitions



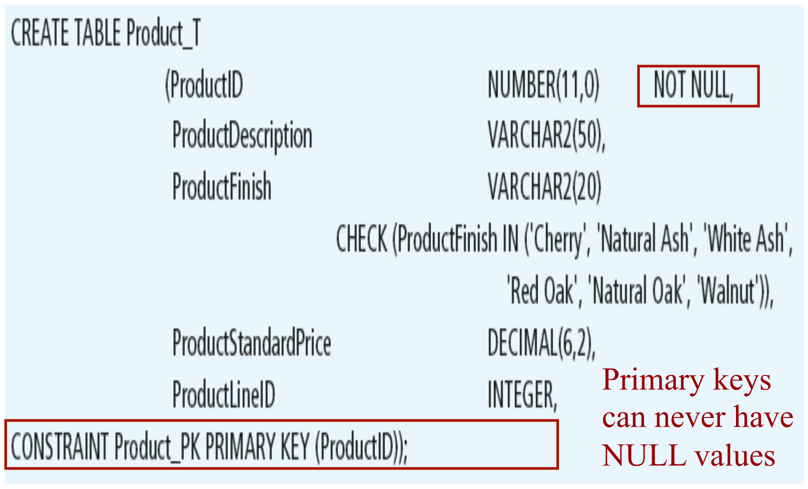
1. Defining attributes and their data types



```
CREATE TABLE Product_T
                (ProductID                  NUMBER(11,0)    NOT NULL,
                 ProductDescription         VARCHAR2(50),
                 ProductFinish              VARCHAR2(20)
                              CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                     'Red Oak', 'Natural Oak', 'Walnut')),
                 ProductStandardPrice       DECIMAL(6,2),
                 ProductLineID              INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

This is Oracle syntax.
In MySQL NUMBER should be replaced by NUMERIC
VARCHAR2 should be replaced by VARCHAR

2. Non-nullable specification

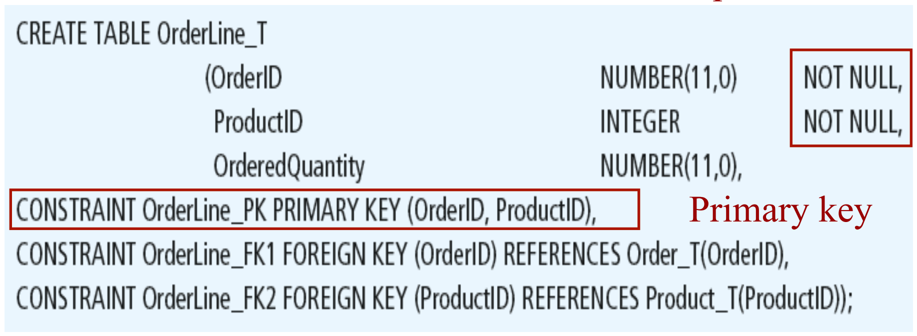


```
CREATE TABLE Product_T
            (ProductID                    NUMBER(11,0)    NOT NULL,
             ProductDescription           VARCHAR2(50),
             ProductFinish                VARCHAR2(20)
                          CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                        'Red Oak', 'Natural Oak', 'Walnut')),
             ProductStandardPrice         DECIMAL(6,2),
             ProductLineID                INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

Primary keys can never have NULL values

4. Identifying Primary Key

Non-nullable specifications



```
CREATE TABLE OrderLine_T
            (OrderID                          NUMBER(11,0)     NOT NULL,
             ProductID                        INTEGER          NOT NULL,
             OrderedQuantity                  NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T(OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T(ProductID));
```

Primary key

Some primary keys are composite

Controlling the Values in Attributes

```
CREATE TABLE Order_T
                    (OrderID                              NUMBER(11,0)      NOT NULL,
                     OrderDate                            DATE DEFAULT SYSDATE,
                     CustomerID                           NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));
```

5. Default value

```
CREATE TABLE Product_T
                    (ProductID                            NUMBER(11,0)      NOT NULL,
                     ProductDescription                   VARCHAR2(50),
                     ProductFinish                        VARCHAR2(20)
                                  CHECK (ProductFinish IN ('Cherry', 'Natural Ash', 'White Ash',
                                           'Red Oak', 'Natural Oak', 'Walnut')),
                     ProductStandardPrice                 DECIMAL(6,2),
                     ProductLineID                        INTEGER,
CONSTRAINT Product_PK PRIMARY KEY (ProductID));
```

6. Domain constraint

7. Identifying foreign keys and establishing relationships

```
CREATE TABLE Customer_T
                    (CustomerID                           NUMBER(11,0)      NOT NULL,
                     CustomerName                         VARCHAR2(25)      NOT NULL,
                     CustomerAddress                      VARCHAR2(30),
                     CustomerCity                         VARCHAR2(20),
                     CustomerState                        CHAR(2),
                     CustomerPostalCode                   VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
                    (OrderID                              NUMBER(11,0)      NOT NULL,
                     OrderDate                            DATE DEFAULT SYSDATE,
                     CustomerID                           NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T(CustomerID));
```

Primary key of parent table

Foreign key of dependent table

STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
…	

QUALIFIED (**FacultyID**, **CourseID**, DateQualified)

FacultyID	**CourseID**	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
…		

FACULTY (**FacultyID**, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
…	

SECTION (**SectionNo**, **Semester**, CourseID)

SectionNo	**Semester**	CourseID
2712	I-2008	ISM 3113
2713	I-2008	ISM 3113
2714	I-2008	ISM 4212
2715	I-2008	ISM 4930
…		

COURSE (**CourseID**, CourseName)

CourseID	CourseName
ISM 3113	Syst Analysis
ISM 3112	Syst Design
ISM 4212	Database
ISM 4930	Networking
…	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

Practice: Exercise #1

Write a database description for each of the relations shown, using SQL DDL. Assume the following attribute data types:

StudentID (integer, primary key)
StudentName (max 25 characters)
FacultyID (integer, primary key)
FacultyName (max 25 characters)
CourseID (8 characters, primary key)
CourseName (max 15 characters)
DateQualified (date)
SectionNo (integer, primary key)
Semester (max 7 characters)

Save your SQL code into a file **`StudentReg.sql`**

Using MySQL

- Available on csdb.csc.villanova.edu
- Invoke with

  mysql –u username –D database -p
- SHOW DATABASES;
- SHOW TABLES;
- DESCRIBE *name_T*; (or SHOW COLUMNS FROM *name_T*;)
- SOURCE *script.sql*
- \! *shell_command*

Conventions

-- comments until end of line

/* can also use C-style comments */

SQL is case insensitive (except for data)

But we usually type reserved words in ALL CAPS

Use single quotes for ‘character constants’

Changing Tables

ALTER TABLE statement allows you to change column specifications:

```
ALTER TABLE table_name alter_table_action;
```

Table Actions:

```
ADD [COLUMN] column_definition
ALTER [COLUMN] column_name SET DEFAULT default-value
ALTER [COLUMN] column_name DROP DEFAULT
DROP [COLUMN] column_name [RESTRICT] [CASCADE]
ADD table_constraint
```

Example (adding a new column with a default value):

Should be single quotes!

```
ALTER TABLE CUSTOMER_T
ADD COLUMN CustomerType VARCHAR2 (2) DEFAULT "Commercial";
```

Removing Tables

DROP TABLE statement allows you to remove tables from your schema:

```
DROP TABLE CUSTOMER_T
```

Practice: Exercise #4

Write SQL data definition commands for each of the following:

1. Add an attribute Class to the Student table, then drop it
2. Create a new Dummy table, then remove it
3. Change the FacultyName field from 25 characters to 40 characters

Insert

```
INSERT INTO table (fields)
VALUES (values)
```

Insert Statement

Adds one or more rows to a table

Inserting into a table

```
INSERT INTO Customer_T VALUES
(001, 'Contemporary Casuals', '1355 S. Himes Blvd.', 'Gainesville', 'FL', 32601);
```

Better practice is to list the fields that actually get data

```
INSERT INTO Product_T (ProductID,
ProductDescription, ProductFinish, ProductStandardPrice)
    VALUES (1, 'End Table', 'Cherry', 175, 8);
```

Inserting from another table

```
INSERT INTO CaCustomer_T
SELECT * FROM Customer_T
    WHERE CustomerState = 'CA';
```

STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
...	

QUALIFIED (**FacultyID**, **CourseID**, DateQualified)

FacultyID	**CourseID**	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
...		

Extend **StudentReg.sql** to populate your tables with this data.

FACULTY (**FacultyID**, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
...	

SECTION (**SectionNo**, **Semester**, CourseID)

SectionNo	**Semester**	CourseID
2712	I-2008	ISM 3113
2713	I-2008	ISM 3113
2714	I-2008	ISM 4212
2715	I-2008	ISM 4930
...		

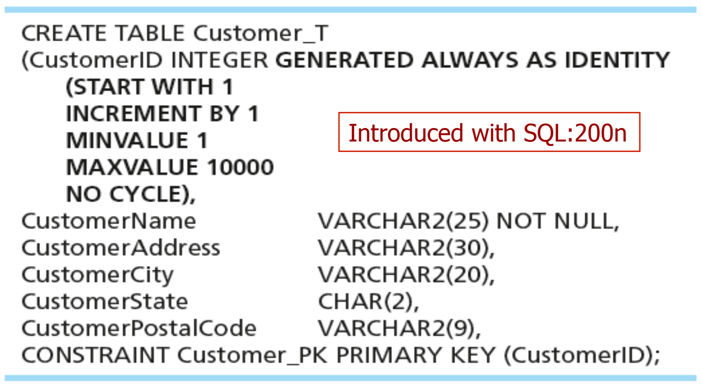
COURSE (**CourseID**, CourseName)

CourseID	CourseName
ISM 3113	Syst Analysis
ISM 3112	Syst Design
ISM 4212	Database
ISM 4930	Networking
...	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

Creating Tables with Identity Columns



```
CREATE TABLE Customer_T
(CustomerID INTEGER GENERATED ALWAYS AS IDENTITY
     (START WITH 1
     INCREMENT BY 1
     MINVALUE 1
     MAXVALUE 10000
     NO CYCLE),
CustomerName            VARCHAR2(25) NOT NULL,
CustomerAddress         VARCHAR2(30),
CustomerCity            VARCHAR2(20),
CustomerState           CHAR(2),
CustomerPostalCode      VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID);
```

Introduced with SQL:200n

Inserting into a table does not require explicit customer ID entry:

```
INSERT INTO CUSTOMER_T VALUES ( ‘Contemporary Casuals’,
‘1355 S. Himes Blvd.’, ‘Gainesville’, ‘FL’, 32601);
```

Note: In mysql only the primary key can be auto-incremented:

ID INT PRIMARY KEY NOT NULL AUTO_INCREMENT

Delete Statement

- Removes rows from a table:

```
DELETE FROM table
WHERE conditions;
```

- If no conditions, delete all data
- Does NOT delete the meta-data,
  use DROP TABLE for that

Delete Statement

Delete certain rows

```
DELETE FROM Customer_T WHERE
CustomerState = ‘CA’;
```

Delete all rows

```
DELETE FROM CUSTOMER_T;
```

Update Statement

Modifies data in existing rows:

```
UPDATE table
SET field = value
WHERE conditions
```

```
UPDATE Product_T
SET ProductStandardPrice = 775
     WHERE ProductID = 7;
```

Update Statement

- Can use the field to modify in an expression:

```
UPDATE Student
SET Age = Age+1
WHERE StudentID = 1
```

- Do this:
  - Add an Age field to the Student_T table, with a default value of 18
  - Increment the Age of the student with ID 54907

Practice: Exercise #5

Write SQL commands for the following:

1. Create two different forms of the INSERT command to add a student with a student ID of 65798 and last name Lopez to the Student table.

2. Now write a command that will remove Lopez from the Student table.

3. Create an SQL command that will modify the name of course ISM 4212 from Database to Introduction to Relational Databases.

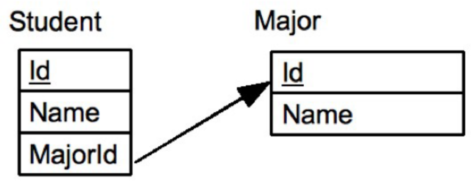
Data Integrity Controls

Referential integrity – constraint that ensures that foreign key values of a table must match primary key values of a related table in 1:M relationships

Restricting:

- Deletes of primary records
- Updates of primary records
- Inserts of dependent records

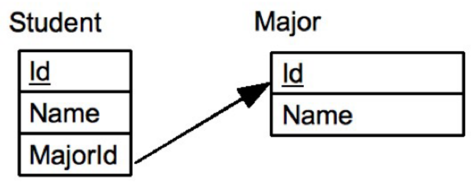
Data Integrity Controls



Write CREATE TABLE statements:

What if a major is deleted from the Major table?
What should happen to the rows pointing to that major?

Data Integrity Controls



```
CREATE TABLE Student(
  Id INTEGER PRIMARY_KEY,
  Name VARCHAR(20) NOT NULL,
  MajorId CHAR(3) REFERENCES Major(Id) ON UPDATE RESTRICT
);
```

```
Options: ON [UPDATE | DELETE]
              RESTRICT       /* do not allow */
              CASCADE        /* propagate change */
              SET NULL       /* Set MajorId to NULL */
              SET DEFAULT    /* Set MajorId to its default value */
```

STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
...	

QUALIFIED (**FacultyID**, **CourseID**, DateQualified)

FacultyID	**CourseID**	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
...		

Data Integrity Controls?

FACULTY (**FacultyID**, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
...	

SECTION (**SectionNo**, **Semester**, CourseID)

SectionNo	**Semester**	CourseID
2712	I-2008	ISM 3113
2713	I-2008	ISM 3113
2714	I-2008	ISM 4212
2715	I-2008	ISM 4930
...		

COURSE (**CourseID**, CourseName)

CourseID	CourseName
ISM 3113	Syst Analysis
ISM 3112	Syst Design
ISM 4212	Database
ISM 4930	Networking
...	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

Basic SELECT

Basic SELECT

Used for queries on single or multiple tables.

```
SELECT [DISTINCT] attribute-list
FROM table-list
WHERE conditions
```

- **SELECT** : the columns (and expressions) to be returned from the query
- **FROM**: indicate the table(s) or view(s) from which data will be obtained
- **WHERE**: indicate the conditions under which a row will be included in the result

Basic SELECT

Used for queries on single or multiple tables.

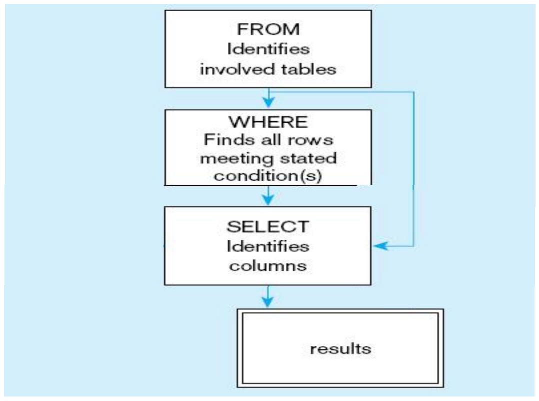
SELECT [DISTINCT] *attribute-list*
FROM *table-list*
WHERE *conditions*

- *Conditions*: comparisons, combined with AND, OR, NOT
- DISTINCT is an optional keyword indicating that the answer should not contain duplicates. The default is that duplicates are *not* eliminated!

TABLE 6-3 Comparison Operators in SQL

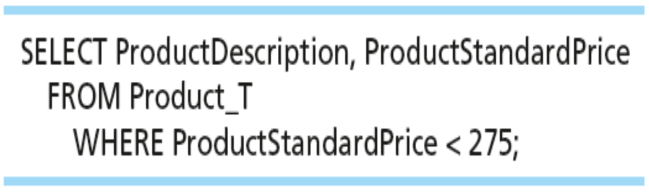
Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
!=	Not equal to

Fragment of Figure 6-10: SQL statement processing order


FROM
Identifies involved tables
WHERE
Finds all rows meeting stated condition(s)
SELECT
Identifies columns
results

SELECT Example

Find products with standard price less than $275



```
SELECT ProductDescription, ProductStandardPrice
  FROM Product_T
    WHERE ProductStandardPrice < 275;
```

Table 6-3: Comparison Operators in SQL

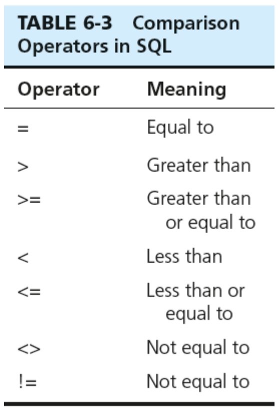


TABLE 6-3 Comparison Operators in SQL

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
!=	Not equal to

SELECT Example Using Alias

Alias is an alternative column or table name

```
SELECT Cust.CustomerName AS Name,
       Cust.CustomerAddress
FROM   Customer_T AS Cust
WHERE CustomerName = ‘Home Furnishings’;
```

Practice: Exercise #6

Write SQL queries to answer the following questions:

1. Which students have an ID number that is less than 50000?
2. What is the name of the faculty member whose ID is 4756?

Summary

- DDL
  - CREATE TABLE
  - DROP TABLE
  - ALTER TABLE
- DML
  - INSERT INTO
  - UPDATE
  - DELETE FROM
  - SELECT