

Chapter 6 Introduction to Structured Query Language (SQL)



DATABASE PROCESSING
Fundamentals, Design,
and Implementation, 9/e

Introduction

- Structured Query Language (SQL) is a data sublanguage that has constructs for defining and processing a database
- It can be
 - Used stand-alone within a DBMS command
 - Embedded in triggers and stored procedures
 - Used in scripting or programming languages

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/2

SQL-92

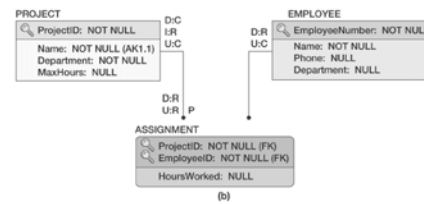
- SQL was developed by IBM in late 1970s
- SQL-92 was endorsed as a national standard by ANSI in 1992
- SQL3 incorporates some object-oriented concepts but has not gained acceptance in industry
- Data Definition Language (DDL) is used to define database structures
- Data Manipulation Language (DML) is used to query and update data
- SQL statement is terminated with a semicolon

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/3

Sample Database

Figure 6.1b Example Database — Database Design



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/4

Sample Data

Figure 6.2 Sample Data for PROJECT, EMPLOYEE, and ASSIGNMENT Relations

ProjectID	Name	Department	MaxHours
1000	Q3 Portfolio Analysis	Finance	75.0
1200	Q3 Tax Prep	Accounting	145.0
1400	Q4 Product Plan	Marketing	130.0
1500	Q4 Portfolio Analysis	Finance	110.0

EmployeeNumber	Name	Phone	Department
100	Mary Jacobs	285-8879	Accounting
200	Kenji Numoto	287-0098	Marketing
300	Heather Jones	287-9981	Finance
400	Rosalie Jackson	285-1273	Accounting
500	James Nestor		Info Systems
600	Richard Wu	287-0123	Info Systems
700	Kim Sung	287-3222	Marketing

ProjectID	EmployeeNumber	HoursWorked
1000	100	17.50
1000	300	13.50
1000	400	8.00
1000	500	20.25
1200	100	45.75
1200	400	70.50
1200	600	40.50
1400	200	75.00
1400	700	20.25
1400	600	20.25

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/5

Sample Data

Figure 6.2b Sample Data for PROJECT, EMPLOYEE, and ASSIGNMENT Relations

EMPLOYEE Relation

EmployeeNumber	Name	Phone	Department
100	Mary Jacobs	285-8879	Accounting
200	Kenji Numoto	287-0098	Marketing
300	Heather Jones	287-9981	Finance
400	Rosalie Jackson	285-1273	Accounting
500	James Nestor		Info Systems
600	Richard Wu	287-0123	Info Systems
700	Kim Sung	287-3222	Marketing

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/6

CREATE TABLE

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

- Example

```
CREATE TABLE PROJECT (
  ProjectID Integer Primary Key,
  Name Char(25) Unique Not Null,
  Department VarChar(100) Null,
  MaxHours Numeric(6,1) Default 100);
```

Data Types

- Standard data types
 - Char for fixed-length character
 - VarChar for variable-length character
 - It requires additional processing than Char data types
 - Integer for whole number
 - Numeric
- There are many more data types in the SQL-92 standard

Figure 6.4 SQL Data Types in DBMS Products
(a) Common Data Types in SQL Server
(b) Common Data Types in Oracle

Data Type	Description
Binary	Binary, length 0 to 8000 bytes.
Char	Character, length 0 to 8000 bytes.
Datetime	8-byte datetime. Range from January 1, 1753, through December 31, 9999, with an accuracy of three hundredths of a second.
Image	Variable-length binary data. Maximum length 2,147,483,647 bytes.
Integer	4-byte integer. Value range from -2,147,483,648 through 2,147,483,647.
Money	8-byte money. Range from -922,337,203,685,477,5808 through +922,337,203,685,477,5807, with accuracy to a ten-thousandth of a monetary unit.
Numeric	Decimal - can set precision and scale. Range -10 ³⁸ +1 through 10 ³⁸ -1.
Smalldatetime	4-byte datetime. Range from January 1, 1900, through June 6, 2079, with an accuracy of one minute.
Smallint	2-byte integer. Range from -32,768 through 32,767.
Smallmoney	4-byte money. Range from 214,748,3648 through +214,748,3647, with accuracy to a ten-thousandth of a monetary unit.
Text	Variable-length text, maximum length 2,147,483,647 characters.
Tinyint	1-byte integer. Range from 0 through 255.
VarChar	Variable-length character, length 0 to 8000 bytes.

(a)

Data Type	Description
BLOB	Binary large object. Up to 4 gigabytes in length.
CHAR(n)	Fixed length character field of length n. Maximum 2,000 characters.
DATE	7-byte field containing both date and time.
INTEGER	Whole number of length 38.
NUMBER(p,s)	Numeric field of length n, s places to the right of the decimal
VARCHAR(n)	Variable length character field up to n characters long. Maximum value of n = 4,000
VARCHAR2(n)	

(b)

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
- Five types of constraints:
 - PRIMARY KEY may not have null values
 - UNIQUE may have null values
 - NULL/NOT NULL
 - FOREIGN KEY
 - CHECK

Figure 6.6 Varieties of SQL CREATE TABLE Statements

```
CREATE TABLE PROJECT (
  ProjectID Integer Primary Key,
  Name Char(25) Unique Not Null,
  Department VarChar(100) Null,
  MaxHours Numeric(6,1) Default 100);

CREATE TABLE EMPLOYEE (
  EmployeeNumber Integer Not Null,
  Name Char(25) Not Null,
  Phone Char(8),
  Department VarChar(100),
  CONSTRAINT EmployeePK PRIMARY KEY (EmployeeNumber));

CREATE TABLE ASSIGNMENT (
  ProjectID Integer Not Null
  FOREIGN KEY REFERENCES PROJECT (ProjectID),
  EmployeeNum Integer Not Null,
  HoursWorked Numeric (5,2) Default 10,
  CONSTRAINT AssignmentPK PRIMARY KEY (ProjectID, EmployeeNum),
  FOREIGN KEY (EmployeeNum) REFERENCES EMPLOYEE (EmployeeNumber));
```

ALTER Statement

- ALTER statement changes table structure, properties, or constraints after it has been created
- Example


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

Figure 6.5 Adding Constraints with the ALTER Statement

```

CREATE TABLE PROJECT (
  ProjectID Integer Primary Key,
  Name Char(25) Unique Not Null,
  Department VarChar(100) Null,
  MaxHours Numeric(8,1) Default 100;

CREATE TABLE EMPLOYEE (
  EmployeeNumber Integer Not Null,
  Name Char(25) Not Null,
  Phone Char(9),
  Department VarChar(100);

ALTER TABLE EMPLOYEE
ADD CONSTRAINT EmployeePK PRIMARY KEY (EmployeeNumber);

CREATE TABLE ASSIGNMENT (
  ProjectID Integer Not Null,
  EmployeeNum Integer Not Null,
  HoursWorked Numeric (5,2) DEFAULT 10,
  CONSTRAINT AssignmentPK PRIMARY KEY (ProjectID, EmployeeNum);

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

ALTER TABLE ASSIGNMENT
ADD CONSTRAINT ProjectFK
FOREIGN KEY (ProjectID) REFERENCES PROJECT (ProjectID)
ON UPDATE CASCADE
ON DELETE CASCADE;

```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/13

Kroenke's Preferred Style

- David Kroenke uses a combination of the previous two styles to encourage a DDL that is both succinct (a relative term in SQL) and yet descriptive

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/14

Figure 6.7 Recommended Style of CREATE TABLE Statements

```

CREATE TABLE PROJECT (
  ProjectID Integer Not Null,
  Name Char(25) Unique Not Null,
  Department VarChar(100) Null,
  MaxHours Numeric(8,1) Default 100
  CONSTRAINT ProjectPK PRIMARY KEY (ProjectID));

CREATE TABLE EMPLOYEE (
  EmployeeNumber Integer Not Null,
  Name Char(25) Not Null,
  Phone Char(9),
  Department VarChar(100),
  CONSTRAINT EmployeePK PRIMARY KEY (EmployeeNumber));

CREATE TABLE ASSIGNMENT (
  ProjectID Integer Not Null,
  EmployeeNum Integer Not Null,
  HoursWorked Numeric (5,2) DEFAULT 10,
  CONSTRAINT AssignmentPK PRIMARY KEY (ProjectID, EmployeeNum),
  CONSTRAINT EmployeeFK FOREIGN KEY
  (EmployeeNum) REFERENCES EMPLOYEE (EmployeeNumber)
  ON UPDATE CASCADE
  ON DELETE NO ACTION,
  CONSTRAINT ProjectFK FOREIGN KEY
  (ProjectID) REFERENCES PROJECT (ProjectID)
  ON UPDATE CASCADE
  ON DELETE CASCADE);

```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/15

DROP Statements

- DROP TABLE statement removes tables and their data from the database
- A table cannot be dropped if it contains foreign key values needed by other tables
 - Use ALTER TABLE DROP CONSTRAINT to remove integrity constraints in the other table first
- Example:
 - DROP TABLE CUSTOMER;
 - ALTER TABLE ASSIGNMENT DROP CONSTRAINT ProjectFK;

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/16

SELECT Statement

- SELECT can be used to obtain values of specific columns, specific rows, or both
- Basic format:


```

SELECT (column names or *)
FROM (table name(s))
[WHERE (conditions)];

```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/17

Example: SELECT Statement

```

SELECT Name, Department, MaxHours
FROM PROJECT;

```

Figure 6.2 Sample Data for PROJECT, EMPLOYEE, and ASSIGNMENT Relations

PROJECT Relation

ProjectID	Name	Department	MaxHours
1000	Q3 Portfolio Analysis	Finance	75.0
1200	Q3 Tax Prep	Accounting	145.0
1400	Q4 Product Plan	Marketing	138.0
1500	Q4 Portfolio Analysis	Finance	110.0

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/18

Example: SELECT DISTINCT

```
SELECT DISTINCT Department
FROM PROJECT;
```

Figure 6.2 Sample Data for PROJECT, EMPLOYEE, and ASSIGNMENT Relations

PROJECT Relation

ProjectID	Name	Department	MaxHours
1000	Q3 Portfolio Analysis	Finance	75.0
1200	Q3 Tax Prep	Accounting	145.0
1400	Q4 Product Plan	Marketing	138.0
1500	Q4 Portfolio Analysis	Finance	110.0

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/19

WHERE Clause Conditions

- Require quotes around values for Char and VarChar columns, but no quotes for Integer and Numeric columns
- AND may be used for compound conditions
- IN and NOT IN indicate 'match any' and 'match all' sets of values, respectively
- Wildcards _ and % can be used with LIKE to specify a single or multiple unknown characters, respectively
- IS NULL can be used to test for null values

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/20

Example: SELECT DISTINCT

```
SELECT *
FROM PROJECT
WHERE Department = 'Finance' AND MaxHours > 100;
```

Figure 6.2 Sample Data for PROJECT, EMPLOYEE, and ASSIGNMENT Relations

PROJECT Relation

ProjectID	Name	Department	MaxHours
1000	Q3 Portfolio Analysis	Finance	75.0
1200	Q3 Tax Prep	Accounting	145.0
1400	Q4 Product Plan	Marketing	138.0
1500	Q4 Portfolio Analysis	Finance	110.0

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/21

Example: IN/NOT IN

```
SELECT Name, Phone, Department
FROM EMPLOYEE
WHERE Department IN ('Accounting',
'Finance',
'Marketing');
```

```
SELECT Name, Phone, Department
FROM EMPLOYEE
WHERE Department NOT IN
('Accounting',
'Finance',
'Marketing');
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/22

Employee Relation from DB

Figure 6.2b Sample Data for PROJECT, EMPLOYEE, and ASSIGNMENT Relations

EMPLOYEE Relation

EmployeeNumber	Name	Phone	Department
100	Mary Jacobs	285-8879	Accounting
200	Kenji Numoto	287-0098	Marketing
300	Heather Jones	287-9981	Finance
400	Rosalie Jackson	285-1273	Accounting
500	James Nestor		Info Systems
600	Richard Wu	287-0123	Info Systems
700	Kim Sung	287-3222	Marketing

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/23

Example: BETWEEN

```
SELECT Name, Department
FROM EMPLOYEE
WHERE EmployeeNumber BETWEEN 200
AND 500;
– Or WHERE EmployeeNumber >= 200 AND
EmployeeNumber <= 500;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/24

Example: LIKE

```
SELECT *  
FROM EMPLOYEE  
WHERE Phone LIKE '285-____';
```

```
SELECT *  
FROM EMPLOYEE  
WHERE Phone LIKE '285%';
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/25

Example: IS NULL

```
SELECT Name, Department  
FROM EMPLOYEE  
WHERE Phone IS NULL;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/26

Sorting the Results

- ORDER BY phrase can be used to sort rows from SELECT statement

```
SELECT Name, Department  
FROM EMPLOYEE  
ORDER BY Department;
```

- Two or more columns may be used for sorting purposes

```
SELECT Name, Department  
FROM EMPLOYEE  
ORDER BY Department DESC, Name ASC;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/27

Built-in Functions

- Five built-in functions for SELECT statement:
 - COUNT counts the number of rows in the result
 - SUM totals the values in a numeric column
 - AVG calculates an average value
 - MAX retrieves a maximum value
 - MIN retrieves a minimum value
- Result is a single number (relation with a single row and a single column)
- Column names *cannot* be mixed with built-in functions
- Built-in functions *cannot* be used in WHERE clauses

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/28

Example: Built-in Functions

```
SELECT COUNT (DISTINCT Department)  
FROM PROJECT;
```

```
SELECT MIN(MaxHours), MAX(MaxHours),  
SUM(MaxHours)  
FROM PROJECT  
WHERE ProjectID < 1500;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/29

Built-in Functions and Grouping

- GROUP BY allows a column and a built-in function to be used together
- GROUP BY sorts the table by the named column and applies the built-in function to groups of rows having the same value of the named column
- WHERE condition must be applied before GROUP BY phrase
- Example

```
SELECT Department, Count(*)  
FROM EMPLOYEE  
WHERE EmployeeNumber < 600  
GROUP BY Department  
HAVING COUNT(*) > 1;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/30

Querying Multiple Tables

- Multiple tables can be queried by using either subqueries or joins
- If all of the result data comes from a single table, subqueries can be used
- If results come from two or more tables, joins must be used
- Joins cannot substitute for correlated subqueries nor for queries that involve EXISTS and NOT EXISTS

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/31

Subqueries

- Subqueries can be extended to include many levels
- Example

```
SELECT DISTINCT Name
FROM EMPLOYEE
WHERE EmployeeNumber IN
  (SELECT EmployeeNum
   FROM ASSIGNMENT
   WHERE HoursWorked > 40
   AND ProjectID IN
     (SELECT ProjectID
      FROM PROJECT
      WHERE Department = 'Accounting'));
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e
by David M. Kroenke

Chapter 6/32

Joins

- The basic idea of a join is to form a new relation by connecting the contents of two or more other relations
 - This joined table can be processed like any other table
 - Example
- ```
SELECT PROJECT.Name, HoursWorked,
 EMPLOYEE.Name
FROM PROJECT, ASSIGNMENT, EMPLOYEE
WHERE PROJECT.ProjectID = ASSIGNMENT.ProjectID
AND EMPLOYEE.EmployeeNumber =
 ASSIGNMENT.EmployeeNum;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 6/33

## Alternate Join Syntax

- SQL-92's alternative join syntax substitutes the words JOIN and ON for WHERE
- Using aliases for table names improves the readability of a join
- Example: alias E is assigned to the EMPLOYEE table

```
SELECT P.Name, HoursWorked, E.Name
FROM PROJECT P JOIN ASSIGNMENT A
ON P.ProjectID = A.ProjectID
JOIN EMPLOYEE E
ON A.EmployeeNum = E.EmployeeNumber;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 6/34

## Outer Joins

- Outer joins can be used to ensure that all rows from a table appear in the result
- Left (right) outer join: every row on the table on the left (right) hand side is included in the results even though the row may not have a match
- Outer joins can be nested

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke


Chapter 6/35

## Example: Outer Join

- Left outer join
- ```
SELECT Name, HoursWorked
FROM PROJECT LEFT JOIN ASSIGNMENT
ON PROJECT.ProjectID = ASSIGNMENT.ProjectID;
```
- Nested outer join
- ```
SELECT PROJECT.Name, HoursWorked,
 EMPLOYEE.Name
FROM ((PROJECT LEFT JOIN ASSIGNMENT
ON PROJECT.ProjectID = ASSIGNMENT.ProjectID)
LEFT JOIN EMPLOYEE
ON EMPLOYEE.EmployeeNumber =
Assignment.EmployeeNum);
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

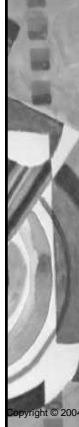
Chapter 6/36



## INSERT INTO Statement

- The order of the column names must match the order of the values
- Values for all NOT NULL columns must be provided
- No value needs to be provided for a surrogate primary key
- It is possible to use a select statement to provide the values for bulk inserts from a second table
- Examples:
  - INSERT INTO PROJECT VALUES (1600, 'Q4 Tax Prep', 'Accounting', 100);
  - INSERT INTO PROJECT (Name, ProjectID) VALUES ('Q1+ Tax Prep', 1700);

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/37




## UPDATE Statement

- UPDATE statement is used to modify values of existing data
- Example:

```
UPDATE EMPLOYEE
SET Phone = '287-1435'
WHERE Name = 'James Nestor';
```
- UPDATE can also be used to modify more than one column value at a time

```
UPDATE EMPLOYEE
SET Phone = '285-0091', Department = 'Production'
WHERE EmployeeNumber = 200;
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/38




## DELETE FROM Statement

- Delete statement eliminates rows from a table
- Example

```
DELETE FROM PROJECT
WHERE Department = 'Accounting';
```
- ON DELETE CASCADE removes any related referential integrity constraint of a deleted row

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 6/39



## Chapter 6

### Introduction to Structured Query Language (SQL)

---

**DATABASE PROCESSING**  
Fundamentals, Design,  
and Implementation, 9/e