
Chapter# 5

An Introduction to SQL

Introduction

- SQL is the standard language for dealing with relational databases
- It is supported by just about every product in the market
- It was introduced by IBM in the early seventies
- System R
- The version of SQL/92 is also known as SQL2
- The official name is International Standard Database Language

-
- SQL3 is the new standard
 - No product in the market support SQL/92 in its entirety (It can't be supported!)
 - Products support a “superset of a subset of SQL/92”
 - SQL was originally intended as a data sub-language
 - SQL became computationally complete
 - It allows procedural statements
 - Call, return, Set, Case, If, loop, ...

-
- SQL3 is the new standard
 - No product in the market support SQL/92 in its entirety (It can't be supported!)
 - Products support a “superset of a subset of SQL/92”
 - SQL was originally intended as a data sub-language
 - SQL became computationally complete

-
- It allows procedural statements
 - Call, return, Set, Case, If, loop, ...
 - SQL doesn't use the terms relations/ relvar. It uses the term 'Table'
 - SQL is a huge language (The standard document is over 600 pages)
 - Far from being a perfect relational language

D(DEALER):

D#	DNAME	STATUS	LOCATION
D1	JOHN	40	SHELTON
D2	MIKEL	20	BRIDGEPORT
D3	THOMAS	40	SHELTON
D4	RON	30	STAMFORD
D5	JIM	50	WATERBURY
D4	SMITH	40	SHELTON

C(CARS):

C#	MODEL	YEAR	LOCATION
C1	VOLVO850	1990	SHELTON
C2	TAURUS	1996	WATERBURY
C3	FORD	1989	STAMFORD
C4	HONDA	1993	SHELTON
C5	BMW	1994	BRIDGEPORT
C6	SATURN	1999	SHELTON
C7	BMW	1995	WATERBURY
C8	FORD	1993	SHELTON

FIG:5.1

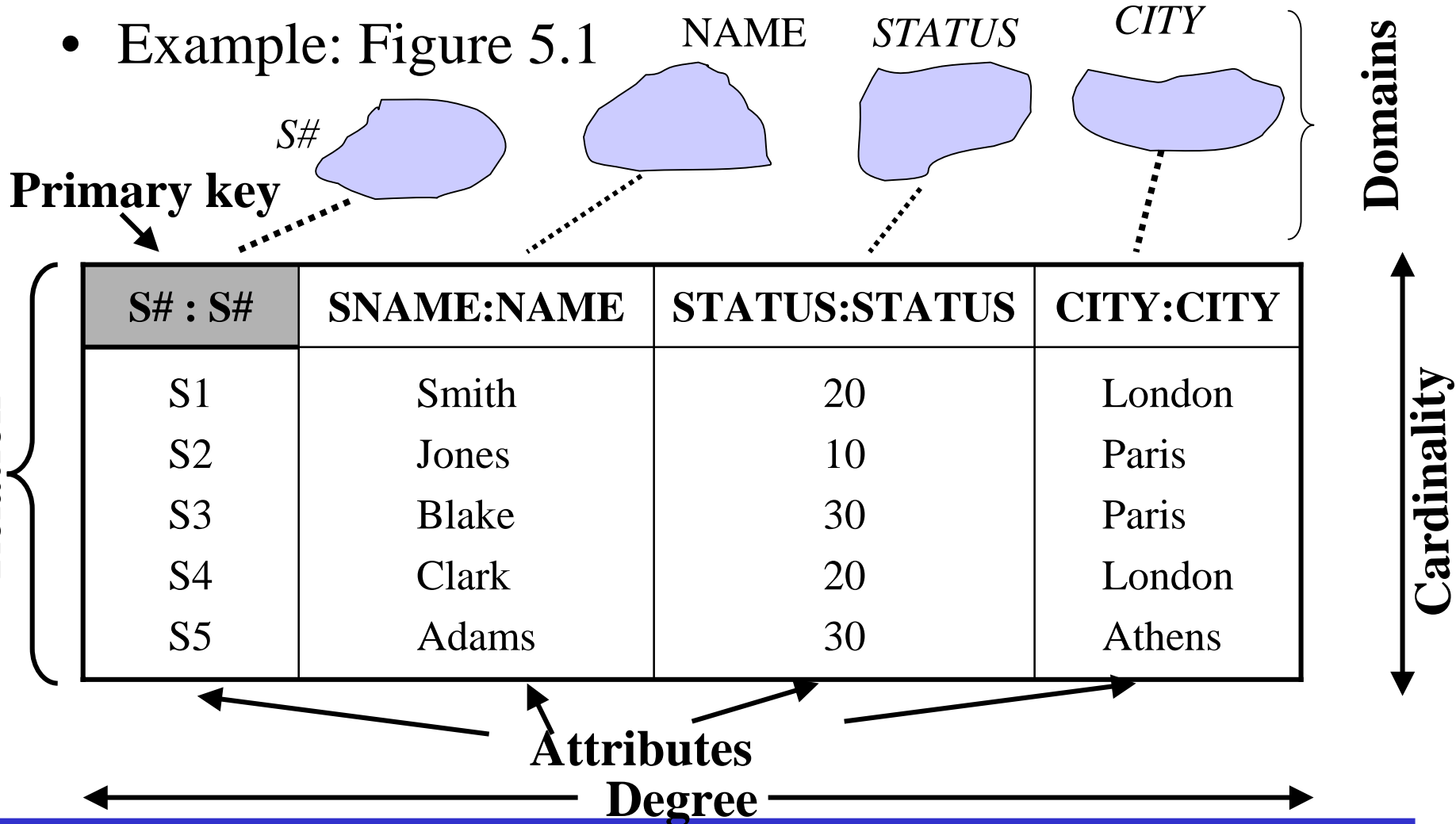
DC:

(DEALERS-CARS)

D#	C#	QUANTITY
D1	C1	30
D1	C2	20
D1	C3	40
D1	C4	20
D1	C5	10
D1	C6	10
D1	C7	25
D1	C8	15
D2	C1	30
D2	C2	40
D3	C2	20
D4	C2	20
D4	C4	30
D4	C5	40

Overview

- Example: Figure 5.1



-
- Syntactic points:
 - The ‘#’ character is not legal in SQL2
 - ‘;’ is used as a statement terminator. SQL2 uses such terminators depends on the context

Type definition

- SQL does not permit users to define their own types
- It permits domains
- Built-in types ‘system’ types
 - Character, Integer, Date, Bit, Smallint, Time, Numeric, Float, Timestamp, Decimal.

SQL Manipulative Operations

- Select, Insert, Update, Delete
- Select is used to perform the relational operations restrict, project, and join

RESTRICT:

SELECT C#,MODEL,YEAR FROM C WHERE
LOCATION = 'SHELTON';

RESULT:

C#	MODEL	YEAR
C1	VOLVO850	1990
C4	HONDA	1993
C6	SATURN	1999
C8	FORD	1992

PROJECT

```
SELECT D#,DNAME,LOCATION  
FROM D;
```

RESULT:

D#	DNAME	LOCATION
D1	JOHN	SHELTON
D2	MIKEL	BRIDGEPORT
D3	THOMAS	SHELTON
D4	RON	STAMFORD
D5	JIM	WATERBURY
D6	SMITH	SHELTON

JOIN:

RESULT:

```
SELECT  
D#,C.C#,MODEL,  
LOCATION  
FROM C,DC  
WHERE  
C.C#=DC.C#;
```

D#	C#	MODEL	LOCATION
D1	C1	VOLVO850	SHELTON
D2	C1	VOLVO850	SHELTON
D1	C2	TAURUS	WATERBURY
D2	C2	TAURUS	WATERBURY
D3	C2	TAURUS	WATERBURY
D4	C2	TAURUS	WATERBURY
D1	C3	FORD	STAMFORD
D1	C4	HONDA	SHELTON
D4	C4	HONDA	SHELTON
D1	C5	BMW	BRIDGEPORT
D4	C5	BMW	BRIDGEPORT
D1	C6	SATURN	SHELTON
D1	C7	BMW	WATERBURY
D1	C8	FORD	SHELTON

Update Operations

- Insert, Update, Delete are set-level operations

Examples:

1. Insert:

Insert into D

(D#,DNAME,STATUS,LOCATION)

values ('D7','ROJER',45,'NEW HEAVEN');

2. Update:

Update D set STATUS = STATUS + 10

where LOCATION = 'BRIDGEPORT';

3. Delete:

Delete from DC where c# = c5;

The Catalog

- Catalog: consists of the descriptors for an individual database
- Schema: consists of descriptors for that portion of that database that belongs to some individual user
- Each catalog as one schema called information schema (this schema has information regarding the catalog)

-
- Information schema consists of SQL tables
 - Domains
 - Tables
 - Views
 - Columns
 - Table_privileges
 - Column_privileges
 - Usage_privileges
 - Table_constraints

Views

➤ Example

Create View Good_Dealer

As Select D#, STATUS, LOCATION From D
Where STATUS > 30;

Result:

D#	DNAME	STATUS	LOCATION
D1	JOHN	40	SHELTON
D2	MIKEL	20	BRIDGEPORT
D3	THOMAS	40	SHELTON
D4	RON	30	STAMFORD
D5	JIM	50	WATERBURY
D6	SMITH	40	SHELTON

-
- SQL Query against this view

Select D#,status

From Good_Dealer

Where LOCATION = 'SHELTON'

Result:

D#	status
D1	40
D3	40
D6	40

Substitution

```
Select Good_Dealer.D# , Good_Dealer.STATUS
From ( Select D#, STATUS, LOCATION
      From D
      Where STATUS > 30 ) As Good_Dealer
Where Good_Dealer.LOCATION = 'SHELTON'
```

-
- Simplification(the query executed is)

Select D#, STATUS

From D

Where STATUS > 30 and

LOCATION = 'SHELTON'

Delete

- Delete

Delete

From Good_Dealer

Where LOCATION = 'WATERBURY'

- Substitution & Simplification

Delete

From D

Where STATUS > 30 and LOCATION =
'WATERBURY'

Embedded SQL

- Direct execution: interactively from an on line terminal
- Embedded execution: SQL can be intermixed with programming language statements
- The application program can be written in a variety of languages

EXEC SQL BEGIN DECLARE SECTION;

DCL SQLSTATE CHAR (5);

DCL C# CHAR (5);

DCL MODEL CHAR (9);

EXEC SQL END DECLARE SECTION;

C# = 'C6'; */* TRYING TO RETRIEVE THE MODEL OF 'C6'*/*

EXEC SQL

SELECT C.MODEL INTO : MODEL

FROM C

WHERE C.C# = :C#;

IF SQLSTATE = '00000'

THEN; */* THE RETRIEVED VALUE IS STORED
IN MODEL */*

ELSE; */* EXCEPTION HANDLING
CONDITION */*

Oracle SQL

- Oracle SQL is a powerful implementation of SQL92
- Oracle SQL*Plus provides a convenient interactive environment with the Oracle database server
- Users can type the commands directly at SQL prompt
- SQL*Plus can execute commands residing in operating system files

Entering and Exiting SQL*Plus

- Enter

sqlplus <userid>/<password>

sqlplus <userid>

sqlplus

- Exit

The exit command is entered at the SQL prompt

Oracle Embedded SQL

- Embedded SQL allows SQL statements in a program written in a high-level programming language such as C or C++
- Oracle provides a tool, called Pro*C/C++ that allows applications to be developed in C or C++ language with embedded SQL statements

-
- The Pro*C/C++ preprocessor parses the embedded program and converts all SQL statements to system calls in C/C++ and produces a C/C++ program.
 - The C/C++ program can be compiled in the usual manner to produce an executable program

Oracle PL/SQL

- PL/SQL is Oracle's procedural extension to SQL.
- It supplements SQL with several high-level programming features
- PL/SQL is a superset of SQL (few exceptions)

Oracle JDBC

- JDBC consists of Java classes that make database access from a Java environment easy and powerful.
- JDBC is a trademark not an acronym (it is not Java Database Connectivity)
- JDBC is an application Programming Interface (API) that enables database access in Java

Oracle JDBC (cont)

- Advantages
 - Portability across database servers
 - Various database vendors (Oracle, Sybase, Informix) provide JDBC drivers, which are JDBC API for their database engines
 - Portability across hardware architecture
 - The JDBC drivers take care of the server dependencies.

Oracle JDBC (cont)

- Applications written in Java using JDBC are independent of the database server
- Portability across hardware platforms is a result of the Java language
- The combination of Java and JDBC to develop database applications is an ideal match, as it is possible for the applications to be written once and run anywhere

SQLJ

- A new standard that many vendors have already adopted
- SQLJ differs from JDBC in that SQL statements can be embedded directly in Java programs
- It is similar to Pro*C or Pro*C++
- SQLJ translator converts Java programs embedded with static SQL statements into pure Java code
- The translated code can be executed through JDBC driver against the database

Dual mode for Embedded SQL

- Any SQL statement that can be used interactively can be used in an application program
- Several embedded SQL statements can't be used in interactive mode

Example using PL/I

```
EXEC SQL BEGIN DECLARE SECTION;
DCL SQLSTATE CHAR (5);
DCL C# CHAR (5);
DCL MODEL CHAR (9);
EXEC SQL END DECLARE SECTION;

C# ='C6'; /* TRYING TO RETRIEVE THE MODEL OF 'C6'*/
EXEC SQL
SELECT C.MODEL INTO : MODEL
FROM C
WHERE C.C# = :C#;
IF SQLSTATE = '00000'
THEN .....; /* THE RETRIEVED VALUE IS STORED
IN MODEL */
ELSE .....; /* EXCEPTION HANDLING CONDITION */
```

Points regarding the embedded example

- Embedded SQL are prefixed by EXEC SQL
- The terminator symbol in PL/I is semicolon
- An executable SQL statement can appear wherever an executable host statement can appear
- Embedded SQL includes purely declarative statements

-
- SQL statements can include references to host variables (colon prefix)
 - INTO clause is used to specify target variables
 - All host variables referenced in SQL statements must be declared (DCL in PL/I) within an embedded SQL declare section.

-
- **SQLSTATE** host variable
 - code 00000: statement executed successfully
 - code 02000: statement executed but no data found
 - In principle each SQL statement should be followed by a test on **SQLSTATE**
 - Host variables must have an appropriate data type that match the corresponding SQL target variable
 - Host variables and SQL can have the same name or different names

WHENEVER

- Exec SQL Whenever <condition> <action>
 - <condition> ::= Sqlerror | Not Found
 - <action> ::= Continue | Go to Label
 - Not found *means* no data was found
 - Sqlerror *means* an error occurred
- New WHENEVER statements override previous statements

Problem

- Retrieval operations are set-operations while host languages variables are not equipped to handle more than one row at a time.
- Cursors are provided to bridge this gap
- Cursor is a pointer that is used to run through a collection of rows

Operations not involving cursors

- Singleton Select
- Insert
- Update (except current)
- Delete (except current)

Singleton Select

- Get STATUS and LOCATION for the DEALER whose D# number is given by the host variable Given#

Exec SQL

```
Select STATUS, LOCATION  
Into :Status , :Location  
From D  
Where D# = :Given# ;
```

- Result

- One row: `SQLSTATE = 00000`
- No rows: `SQLSTATE = 02000`
- More than one row: `SQLSTATE` will be set to an error code

Insert

- Insert a new DEALER into table D whose values of D#,DNAME and LOCATION are given by the host variables D#,DNAME and LOCATION and STATUS is unknown.
- Exec SQL

Insert

Into D (D#, DNAME, LOCATION)

Values :D#, :DNAME,:LOCATION);

Update

- Increase the STATUS of all DEALERS in SHELTON by the value given by the host variable Raise

Exec SQL Update D

Set STATUS = STATUS + :Raise

Where LOCATION = 'SHELTON';

Delete

- Delete all Dealers from DC whose D# is given by the host variable D#.
- Exec SQL

Delete

From DC

Where :D# =

(Select D#

From DC

Where D# = :D#);

Operations Involving Cursors

- Declare X Cursor
 - defines a cursor with an associated table expression
 - The table is not evaluated at the declaration point
 - The table is evaluated when the cursor is opened (Open X)
 - 'Fetch X Into' is used to retrieve rows one at a time

Declare Cursor statement

- Exec SQL Declare <cursor name> Cursor
For <table expression> [<ordering>];
 - <ordering> ::= Order By <order item
commadlist>
 - <order item> ::= column name { ASC | DESC }
 - The table expression can include host variables
 - The program can contain more than one cursor
operating on different tables

Executable statements for cursors

- Open
 - Exec SQL Open <cursor name>
 - Activates the specified cursor
 - The table expression is executed
 - A set of rows becomes the current active set
 - A position before the first row is identified
 - The active set is ordered by the 'Order by'

-
- Fetch
 - Exec SQL Fetch <cursor name>
Into <host variable(s)>
 - Each time the Fetch statement is executed advances the cursor to the next row
 - If there is no next row, **SQLSTATE** is set to 02000

-
- Close
 - Exec SQL Close <cursor name>
 - Deactivates the specified cursor
 - The cursor will not have an active set
 - It can be opened later. The active set may be similar or different from the previous set

Example:

- **EXEC SQL DECLARE X CURSOR** */*define the cursor*/*
SELECT D#,DNAME,STATUS
FROM D
WHERE LOCATION = :loc */*:loc is a hostvariable*/*
ORDER BY D# DESC;

EXEC SQL OPEN X; */*execute the query*/*

DO FOR ALL D ROWS ACCESSIBLE VIA X;
EXEC SQL FETCH X INTO :D#,:DNAME,:STATUS;
.....
END;

EXEC SQL CLOSE X; */*deactivate the cursor*/*

- Current

- Used with update and delete to manipulate the current row

- Exec SQL Update D

Set STATUS = STATUS + :Raise
Where Current of X;

Dynamic SQL

- Dynamic SQL consists of a set of embedded SQL facilities that are intended to support the construction of generalized, online, and possibly interactive applications

-
- Steps of an online application
 - Accept a command from the terminal
 - Analyze the command
 - Execute appropriate SQL statements on the database
 - Return a message and/or results to the terminal
 - Hardwired SQL statements in the program
 - Construct SQL statements dynamically and then compile and execute the constructed statements dynamically
 - Prepare and Execute instructions

Example

- DCL Sqlsource Char Varying (65000);

```
Sqlsource = 'Delete From DC Where D# = 'D3';  
Exec SQL Prepare Sqlprepped From :Sqlsource;  
Exec SQL Execute Sqlprepped ;
```