Chapter 7: SQL

Modern Database Management 6<sup>th</sup> Edition Jeffrey A. Hoffer, Mary B. Prescott, Fred R. McFadden

© Prentice Hall, 2002

## SQL Is:

Structured Query Language

The standard for relational database management systems (RDBMS)

SQL-92 Standard -- Purpose:

- Specify syntax/semantics for data definition and manipulation
- Define data structures
- Enable portability
- Specify minimal (level 1) and complete (level 2) standards
- Allow for later growth/enhancement to standard

**Benefits of a Standardized Relational Language Reduced** training costs Productivity **Application** portability **Application** longevity Reduced dependence on a single vendor **Cross-system communication** 

# SQL Environment

### Catalog

– a set of schemas that constitute the description of a database

### Schema

 The structure that contains descriptions of objects created by a user (base tables, views, constraints)

### Data Definition Language (DDL):

- Commands that define a database, including creating, altering, and dropping tables and establishing constraints
- Data Manipulation Language (DML)
  - Commands that maintain and query a database
- Data Control Language (DCL)
  - Commands that control a database, including administering privileges and committing data

### Figure 7-1: A simplified schematic of a typical SQL environment, as described by the SQL-92 standard



# SQL Data types (from Oracle8)

### String types

- CHAR(n) fixed-length character data, n characters long Maximum length = 2000 bytes
- VARCHAR2(n) variable length character data, maximum 4000 bytes
- LONG variable-length character data, up to 4GB. Maximum 1 per table

### Numeric types

- NUMBER(p,q) general purpose numeric data type
- INTEGER(p) signed integer, p digits wide
- FLOAT(p) floating point in scientific notation with p binary digits precision

### Date/time type

DATE – fixed-length date/time in dd-mm-yy form

### Figure 7-4: DDL, DML, DCL, and the database development process



## **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 table and its columns
- CREATE VIEW defines a logical table from one or more views

Other CREATE statements: CHARACTER SET, COLLATION, TRANSLATION, ASSERTION, DOMAIN

## **Table Creation**

Figure 7-5: General syntax for CREATE TABLE

CREATE TABLE tablename ( {column definition [table constraint] } . . . . [ON COMMIT {DELETE | PRESERVE} ROWS] );

where column definition ::=
column\_name
{domain name | datatype [(size)] }
[column\_constraint\_clause . . .]
[default value]
[collate clause]

and table constraint ::= [CONSTRAINT constraint\_name] Constraint\_type [constraint\_attributes]

### **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 keyforeign key mates
- 5. Determine default values
- 6. Identify constraints on columns (domain specifications)
- 7. Create the table and associated indexes

### Figure 7-3: Sample Pine Valley Furniture data

icrosoft Ac	ccess - [CUSTOMER_t	: Table]					_ 8 ×					
<u>File E</u> dit <u>V</u>	jew Insert Format Re	cords <u>T</u> ools <u>W</u> inda	w Help				_ & ×					
- 🖬 🧉	5 D. 🕫 🕺 🖻 🖻	1 SF 10 🚷	21 X1 V Ta V	14 +* HX (	口酒・	2.						
Custom	er_ID Custom	er_Name	Customer_Address	City	State	Postal_Code						
3	Contemporary	Casuals	1355 S Hines Blvd	Gainesville	FL	32601-						
0	2 Value Furniture		15145 S.W. 17th St.	Plano	TX	75094-						
	3 Home Furnishir	ngs	1900 Allard Ave.	Albany	NY	12209-						
	4 Eastern Furnitu	ire	1925 Beltline Rd.	Carteret	NJ	07008-	2 Micro	oft Access - IO	RDER_t:Table1			
	5 Impressions		5585 Westcott Ct.	Sacramento	CA	94206-	E Sile	Edit View Incor	Format Bosonds Tools	Vindow Help		
	6 Furniture Galler	ry S	325 Flatiron Dr.	Boulder	CO	80514-		Edic Meve Inser	. Pormac Kecords Tools 1			
-	7 Period Furniture	B	394 Rainbow Dr.	Seattle	VVA	97954-	K •			5 2+ X+ 15 19 1	/ Ma ▶* M [D /2] •	2
	8 Calfornia Class	ICS C	516 Peach Rd.	Santa Clara	LA	96915-		Order_ld	Order_Date Custome	er_ID		
	10 Seminale Interi	- urniture	2			4620-	<b>)</b>	1001	10/21/2000	1		
	10 Seminole Inten	Lifectulos	<b>And</b>	0 100 0		7500	+	1002	10/21/2000	8		
	12 Battle Creek E	unituro 2	CUSU	UIIE	IS	2015	•	1003	10/22/2000	15		
	12 Battle Creek Pt	hinge (		· ·		7013	•	1004	10/22/2000	5		
	14 Kanaaha Home	nings c	112 Kiowai St	Kanaoha	н	96744	•	1005	10/24/2000	3	andana	
	15 Mountain Scen	0°	1132 Main Street	Orden	LIT	84403-	•	1006	10/24/2000	2	Jruers	
(AutoNu	mher)	60	102 Main Otreet	Ogden	01	04403	•	1007	10/27/2000	11		
(r idiorital	moony				1.0		+	1008	10/30/2000	12		
d: 14 🔍	1 + ++ +-	* of 15						1009	11/5/2000	4		
e number t	o identfy customer							1010	11/5/2000	1		
Micro	osoft Access - [Order_	line_t : Table]					Record:		1 + +1 +* of 10			
j 🛄 Eile	Edit View Insert For	mat <u>R</u> ecords <u>T</u> ool	s <u>W</u> indow <u>H</u> elp				Datashee	t View				
× -	₽ @ ₽ . ** %	h 🛍 🚿 👳		57 🖊 🕨	w E	· 2.						
0	Order_Id Produc	ct_ld Ordered	Quan									
		1	2									
	1001	2	2									
	1001	4	1									
	1002	2	2				🎤 Microsoft Acce:					_ <u>8</u> ×
	1003	6	2				III File Edit View	Insert Format	Records Tools Window He	lp		_ 리 ×
	1004	8	2				N.D.B.	A HES V FR			** 5 5 - 2	
	1005	4	-1						2+ A			
	1006	4			•		Product_II	Product_De	scription Product_Finis	h Standard_Price	Product_Line_Id	
	1006	5	- ord	ler I	In		•	1 End Table	Cherry	\$175.00	10001	
	1006	7					*	2 Coffee Table	Natural Ash	\$200.00	20001	
	1007	1	3				*	3 Computer De	sk Natural Ash	\$3/5.00	20001	
	1007	2	2					4 Entertainmen 5 Miniteria Deel	Chamu	\$650.00	10001	
	1008	3	3					5 writer's Desi	Cherry	\$325.00	20001	
	1008	8	3				· ·	7 Dining Table	Natural Ach	\$20.00	20001	
	1009	4	2				+	8 Computer Do	ek Walnut	\$000.00	20001	
	1009	7	3				* (AutoNumbe	n Computer De	on wannut	¢∠30.00 ¢∩.00		
	1010	8	10				(Autoridinite			\$0.00	nnodu	ota
*	0	0	0				Record: 14 4	1 🕨 🕅	▶* of 8		prouu	
Darre 1							Datasheet View					
Record:		or 18										
	tak Ularin											

CREATE TABLE CUSTOMER\_T (CUSTOMER\_ID NUMBER(11, 0) NOT NULL, CUSTOMER NAME VARCHAR2(25) NOT NULL. CUSTOMER\_ADDRESS VARCHAR2(30), VARCHAR2(20). CITY STATE VARCHAR2(2), POSTAL CODE VARCHAR2(9), CONSTRAINT CUSTOMER\_PK PRIMARY KEY (CUSTOMER\_ID)); CREATE TABLE ORDER\_T NUMBER(11, 0) NOT NULL, (ORDER\_ID ORDER DATE DEFAULT SYSDATE. DATE CUSTOMER ID NUMBER(11.0), CONSTRAINT ORDER PK PRIMARY KEY (ORDER ID), CONSTRAINT ORDER\_FK FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER\_T(CUSTOMER\_ID)); CREATE TABLE PRODUCT\_T INTEGER NOT NULL. (PRODUCT ID PRODUCT\_DESCRIPTION VARCHAR2(50). PRODUCT\_FINISH VARCHAR2(20) CHECK (PRODUCT\_FINISH IN ('Cherry', 'Natural Ash', 'White Ash', 'Red Oak', 'Natural Oak', 'Walnut')), STANDARD\_PRICE DECIMAL(6,2), PRODUCT\_LINE\_ID INTEGER. CONSTRAINT PRODUCT\_PK PRIMARY KEY (PRODUCT\_ID)); CREATE TABLE ORDER\_LINE\_T (ORDER\_ID NUMBER(11,0) NOT NULL, PRODUCT ID NUMBER(11.0) NOT NULL. ORDERED QUANTITY NUMBER(11,0), CONSTRAINT ORDER\_LINE\_PK PRIMARY KEY (ORDER\_ID, PRODUCT\_ID), CONSTRAINT ORDER\_LINE\_FK1 FOREIGN KEY(ORDER\_ID) REFERENCES ORDER\_T(ORDER\_ID), CONSTRAINT ORDER\_LINE\_FK2 FOREIGN KEY (PRODUCT\_ID) REFERENCES PRODUCT\_T(PRODUCT\_ID));





CREATE TABLE CUSTOMER_T (CUSTOMER_ID NUMBER(11, 0) NOT NULL, CUSTOMER_NAME VARCHAR2(25) NOT NULL, CUSTOMER_ADDRESS VARCHAR2(30), CITY VARCHAR2(20), STATE VARCHAR2(20), STATE VARCHAR2(2), POSTAL_CODE VARCHAR2(9), CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));	ng keys
CREATE TABLE ORDER_T (ORDER_ID NUMBER(11, 0) NOT NULL, ORDER_DATE DATE DEFAULT SYSDATE, <u>CUSTOMER_ID NUMBER(11, 0),</u> CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),	
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOM CREATE TABLE PRODUCT_T (PRODUCT_ID INTEGER NOT NULL, PRODUCT_DESCRIPTION VARCHAR2(50), PRODUCT_FINISH VARCHAR2(20) CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',	MER_ID));
'Red Oak', 'Natural Oak', 'Walnut')), STANDARD_PRICE DECIMAL(6,2), PRODUCT_LINE_ID INTEGER, CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID)); This is a c	omposito
CREATE TABLE ORDER_LINE_T (ORDER_ID NUMBER(11,0) NOT NULL, primary k PRODUCT_ID NUMBER(11,0) NOT NULL, ORDERED_QUANTITY NUMBER(11,0). CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID), CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY (ORDER_ID) REFERENCES ORDER_T(ORDER_ID) CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID)	ey

CREATE TABLE CUSTOMER_T (CUSTOMER_ID NUME CUSTOMER_NAME VARC CUSTOMER_ADDRESS VARC CITY VARC STATE VARC POSTAL_CODE VARC CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUST	ER(11, 0) NOT NULL, HAR2(25) NOT NULL, HAR2(30), HAR2(20), HAR2(2), HAR2(2), HAR2(2), HAR2(9), MER_ID));Identifying foreign keys and establishing relationships
CREATE TABLE ORDER_T (ORDER_ID NUME ORDER_DATE DATE CUSTOMER_ID NUME	ER(11, 0) NOT NULL, DEFAULT SYSDATE, ER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID	ID) REFERENCES CUSTOMER T/CUSTOMER ID)
CONSTRAINT ORDER_FR FOREIGN RET (CUSTOME	(_D) REFERENCES COSTOMER_I(COSTOMER_D)),
CREATE TABLE PRODUCT_T	
(PRODUCT_ID INTEC	ER NOT NULL,
PRODUCT_DESCRIPTION VARC	1AR2(50),
PRODUCT_FINISH VARC	IAR2(20)
CHECK (FRODUCI_	Oak' (Natural Oak' (Walaut'))
STANDARD PRICE DECI	MAL (6.2)
PRODUCT LINE ID INTEG	FR
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODU	(T_ID));
OPEATE TABLE OPDER LINE T	
ORDER ID NUME	ER/11 0) NOT NULL
PRODUCT ID NUME	ER(11.0) NOT NULL
ORDERED QUANTITY NUME	ER(11,0)
CONSTRAINT ORDER LINE PK PRIMARY KEY (ORDI	R ID. PRODUCT ID).
CONSTRAINT ORDER LINE FK1 FOREIGN KEY/ORD	ER ID) REFERENCES ORDER T(ORDER ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRO	DUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));



CREATE TABLE CUSTOMER_T (CUSTOMER_ID CUSTOMER_NAME CUSTOMER_ADDRESS CITY STATE POSTAL_CODE CONSTRAINT CUSTOMER_PK PRIMARY KEY	NUMBER(11, 0) NOT NULL, VARCHAR2(25) NOT NULL, VARCHAR2(30), VARCHAR2(20), VARCHAR2(2), VARCHAR2(2), VARCHAR2(9), (CUSTOMER ID));	Overall table definitions		
CREATE TABLE ORDER_T (ORDER_ID NUMBER(11, 0) NOT NULL, ORDER_DATE DATE DEFAULT SYSDATE, CUSTOMER_ID NUMBER(11, 0), CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID), CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));				
CREATE TABLE PRODUCT_T (PRODUCT_ID INTEGER NOT NULL, PRODUCT_DESCRIPTION VARCHAR2(50), PRODUCT_FINISH VARCHAR2(20) CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash', 'Red Oak', 'Natural Oak', 'Walnut')), STANDARD_PRICE DECIMAL(6,2), PRODUCT_LINE_ID INTEGER, CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));				
CREATE TABLE ORDER_LINE_T (ORDER_ID PRODUCT_ID ORDERED_QUANTITY CONSTRAINT ORDER_LINE_PK PRIMARY KE CONSTRAINT ORDER_LINE_FK1 FOREIGN K CONSTRAINT ORDER_LINE_FK2 FOREIGN K	NUMBER(11,0) NOT NULL, NUMBER(11,0) NOT NULL, NUMBER(11,0), Y (ORDER_ID, PRODUCT_ID), Y (ORDER_ID) REFERENCES OR Y (PRODUCT_ID) REFERENCES	DER_T(ORDER_ID), PRODUCT_T(PRODUCT_ID));		

## **Using and Defining Views**

Views provide users controlled access to tables

- Advantages of views:
  - Simplify query commands
  - Provide data security

Enhance programming productivity
 CREATE VIEW command

# View Terminology

Base Table

- A table containing the raw data
- Dynamic View
  - A "virtual table" created dynamically upon request by a user.
  - No data actually stored; instead data from base table made available to user
  - Based on SQL SELECT statement on base tables or other views

### Materialized View

- Copy or replication of data
- Data actually stored
- Must be refreshed periodically to match the corresponding base tables

## Sample CREATE VIEW

CREATE VIEW EXPENSIVE\_STUFF\_V AS SELECT PRODUCT\_ID, PRODUCT\_NAME, UNIT\_PRICE FROM PRODUCT\_T WHERE UNIT\_PRICE >300 WITH CHECK\_OPTION;

•View has a name

View is based on a SELECT statement
CHECK\_OPTION works only for updateable views and prevents updates that would create rows not included in the view



### Table 7-2: Pros and Cons of Using Dynamic Views

#### Table 7-2 Pros and Cons of Using Dynamic Views

Positive Aspects	Negative Aspects
Simplify query commands Help provide data security and confidentiality	Use processing time re-creating view each time it is referenced
Improve programmer productivity Contain most current base table data	May or may not be directly updateable
Use little storage space	
Provide a customized view for a user	
Establish physical data independence	

## **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

### Figure 7-7: Ensuring data integrity through updates



Cascaded Update: Changing a customer ID in the CUSTOMER table will result in that value changing in the ORDER table to match.

... ON UPDATE CASCADE);

Set Null Update: When a customer ID is changed, any customer ID in the ORDER table that matches the old customer ID is set to NULL.

... ON UPDATE SET NULL);

Set Default Update: When a customer ID is changed, any customer ID in the ORDER tables that matches the old customer ID is set to a predefined default value.

. . . ON UPDATE SET DEFAULT);

# Changing and Removing Tables

ALTER TABLE statement allows you to change column specifications:
ALTER TABLE CUSTOMER\_T ADD (TYPE VARCHAR(2))
DROP TABLE statement allows you to remove tables from your schema:
DROP TABLE CUSTOMER\_T

# **Schema Definition**

### Control processing/storage efficiency:

- Choice of indexes
- File organizations for base tables
- File organizations for indexes
- Data clustering
- Statistics maintenance
- Creating indexes
  - Speed up random/sequential access to base table data
  - Example
    - CREATE INDEX NAME\_IDX ON CUSTOMER\_T(CUSTOMER\_NAME)
    - This makes an index for the CUSTOMER\_NAME field of the CUSTOMER\_T table

## **Insert Statement**

Adds data to a table

Inserting into a table

 INSERT INTO CUSTOMER\_T VALUES (001, 'CONTEMPORARY Casuals', 1355 S. Himes Blvd.', 'Gainesville', 'FL', 32601);

Inserting a record that has some null attributes requires identifying the fields that actually get data

 INSERT INTO PRODUCT\_T (PRODUCT\_ID, PRODUCT\_DESCRIPTION, PRODUCT\_FINISH, STANDARD\_PRICE, PRODUCT\_ON\_HAND) VALUES (1, 'End Table', 'Cherry', 175, 8);

Inserting from another table

 INSERT INTO CA\_CUSTOMER\_T SELECT \* FROM CUSTOMER\_T WHERE STATE = 'CA';



## **Delete Statement**

Removes rows from a table
Delete certain rows
- DELETE FROM CUSTOMER\_T WHERE
STATE = 'HI';
Delete all rows
- DELETE FROM CUSTOMER\_T;

## Update Statement

Modifies data in existing rows

### UPDATE PRODUCT\_T SET UNIT\_PRICE = 775 WHERE PRODUCT\_ID = 7;



# The SELECT Statement

Used for queries on single or multiple tables Clauses of the SELECT statement:

- SELECT
  - List the columns (and expressions) that should 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
- GROUP BY
  - Indicate categorization of results
- HAVING
  - Indicate the conditions under which a category (group) will be included
- ORDER BY
  - Sorts the result according to specified criteria

Figure 7-8: SQL statement processing order (adapted from van der Lans, p.100)



## **SELECT** Example

Find products with standard price less than \$275

### SELECT PRODUCT\_NAME, STANDARD\_PRICE FROM PRODUCT\_V WHERE STANDARD\_PRICE < 275

Table 7-3: 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 with ALIAS**

Alias is an alternative column or table name

SELECT CUST.CUSTOMER AS NAME, CUST.CUSTOMER\_ADDRESS FROM CUSTOMER\_V CUST WHERE NAME = 'Home Furnishings'; SELECT Example Using a Function Using the COUNT aggregate function to find totals

### SELECT COUNT(\*) FROM ORDER\_LINE\_V WHERE ORDER\_ID = 1004;

Note: with aggregate functions you can't have singlevalued columns included in the SELECT clause

## **SELECT Example – Boolean Operators**

AND, OR, and NOT Operators for customizing conditions in WHERE clause

SELECT PRODUCT\_DESCRIPTION, PRODUCT\_FINISH, STANDARD\_PRICE FROM PRODUCT\_V WHERE (PRODUCT\_DESCRIPTION LIKE '% Desk' OR PRODUCT\_DESCRIPTION LIKE '% Table') AND UNIT\_PRICE > 300;

Note: the LIKE operator allows you to compare strings using wildcards. For example, the % wildcard in '% Desk' indicates that all strings that have any number of characters preceding the word "Desk" will be allowed Chapter 7 © Prentice Hall, 2002 SELECT Example – Sorting Results with the ORDER BY Clause Sort the results first by STATE, and within a state by CUSTOMER\_NAME

SELECT CUSTOMER\_NAME, CITY, STATE FROM CUSTOMER\_V WHERE STATE IN ('FL', 'TX', 'CA', 'HI') ORDER BY STATE, CUSTOMER\_NAME;

Note: the IN operator in this example allows you to include rows whose STATE value is either FL, TX, CA, or HI. It is more efficient than separate OR conditions

### SELECT Example – Categorizing Results Using the GROUP BY Clause

For use with aggregate functions

- Scalar aggregate: single value returned from SQL query with aggregate function
- Vector aggregate: multiple values returned from SQL query with aggregate function (via GROUP BY)

SELECT STATE, COUNT(STATE) FROM CUSTOMER\_V GROUP BY STATE;

Note: you can use single-value fields with aggregate functions if they are included in the GROUP BY clause

SELECT Example – Qualifying Results by Categories Using the HAVING Clause For use with GROUP BY

SELECT STATE, COUNT(STATE) FROM CUSTOMER\_V GROUP BY STATE HAVING COUNT(STATE) > 1;

Like a WHERE clause, but it operates on groups (categories), not on individual rows. Here, only those groups with total numbers greater than 1 will be included in final result