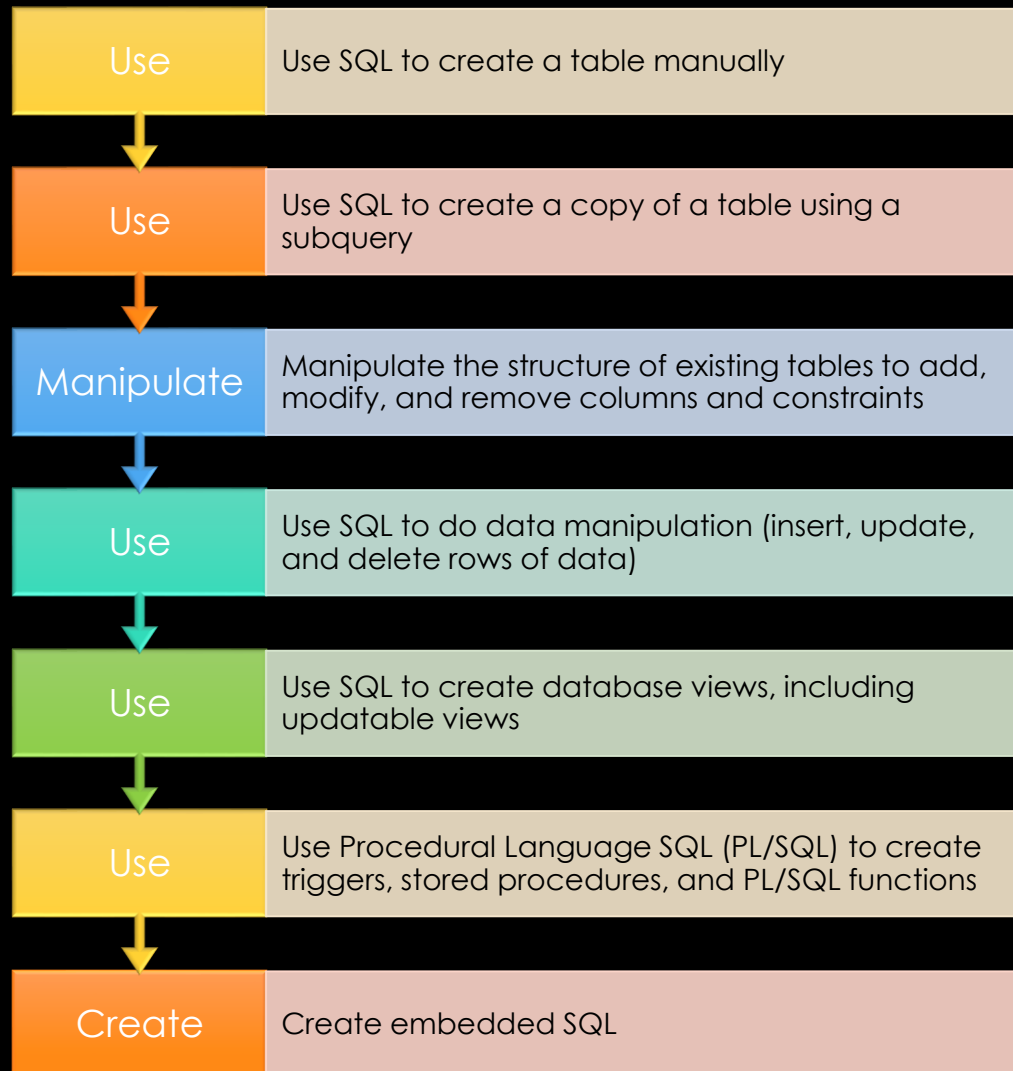


# 8. ADVANCED SQL

Database Design & Business Application Development

# LEARNING OBJECTIVES



# DATA DEFINITION COMMANDS (1 OF 3)

## Starting database model

- Refer to Figure 8.1

## Creating the database

- Before a new RDBMS can be used, the database structure and the tables that will hold the end-user data must be created

## The database schema

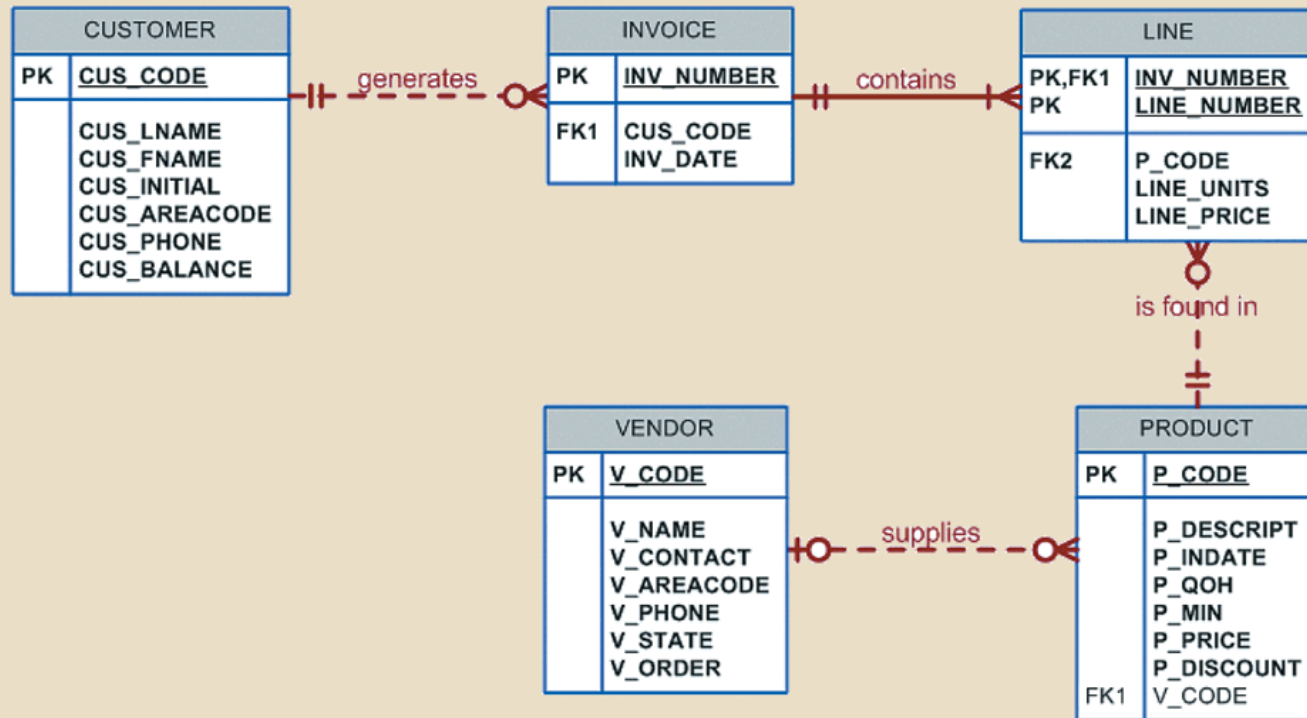
- Logical group of database objects—such as tables and indexes—that are related to each other

## Data types

- Character, numeric, and date

# DATA DEFINITION COMMANDS (2 OF 3)

FIGURE 8.1 DATABASE MODEL



# DATA DEFINITION COMMANDS (3 OF 3)

TABLE 8.1: Some Common SQL Data Types		
Data Type	Format	Comments
<b>Numeric</b>	NUMBER(L,D) or NUMERIC(L,D)	The declaration NUMBER(7,2) or NUMERIC(7,2) indicates that numbers will be stored with two decimal places and may be up to seven digits long, including the sign and the decimal place (for example, 12.32 or -134.99).
	INTEGER	May be abbreviated as INT. Integers are (whole) counting numbers, so they cannot be used if you want to store numbers that require decimal places.
	SMALLINT	Like INTEGER but limited to integer values up to six digits. If your integer values are relatively small, use SMALLINT instead of INT.
	DECIMAL(L,D)	Like the NUMBER specification, but the storage length is a minimum specification. That is, greater lengths are acceptable, but smaller ones are not. DECIMAL(9,2), DECIMAL(9), and DECIMAL are all acceptable.
<b>Character</b>	CHAR(L)	Fixed-length character data for up to 255 characters. If you store strings that are not as long as the CHAR parameter value, the remaining spaces are left unused. Therefore, if you specify CHAR(25), strings such as Smith and Katzenjammer are each stored as 25 characters. However, a U.S. area code is always three digits long, so CHAR(3) would be appropriate if you wanted to store such codes.
	VARCHAR(L) or VARCHAR2(L)	Variable-length character data. The designation VARCHAR2(25) or VARCHAR(25) will let you store characters up to 25 characters long. However, unlike CHAR, VARCHAR will not leave unused spaces. Oracle automatically converts VARCHAR to VARCHAR2.
<b>Date</b>	DATE	Stores dates in the Julian date format.

# CREATING TABLE STRUCTURES (1 OF 2)

## CREATE TABLE command

- CREATE TABLE *tablename* (
  - *column1*            *data type*            [*constraint*] [,
  - *column2*            *data type*            [*constraint*] ] [,
  - PRIMARY KEY (*column1*            [, *column2*]) ] [,
  - FOREIGN KEY (*column1*            [, *column2*]) REFERENCES *tablename*] [,
  - CONSTRAINT *constraint* ] );

## SQL constraints

### FOREIGN KEY

### NOT NULL

### UNIQUE

### DEFAULT

### CHECK

# CREATING TABLE STRUCTURES (2 OF 2)



## Create a table with a SELECT statement

Rapidly creates a new table based on selected columns and rows of an existing table using a subquery

Automatically copies all of the data rows returned



## SQL indexes

CREATE INDEX improves the efficiency of searches and avoids duplicate column values

DROP Index deletes an index

# TABLE STRUCTURES (1 OF 3)

- All changes in the table structure are made by using the ALTER TABLE command followed by a keyword that produces the specific change you want to make
  - ADD, MODIFY, and DROP
- Changing a column's data type
  - ALTER
- Changing a column's data characteristics
  - If the column to be changed already contains data, you can make changes in the column's characteristics if those changes do not alter the data type
- Adding a column
  - You can alter an existing table by adding one or more columns
  - Be careful not to include the NOT NULL clause for the new column



# TABLE STRUCTURES (2 OF 3)

- Adding primary key, foreign key, and check constraints

- Primary key syntax:

```
ALTER TABLE PART
ADD
(PART_CODE);
PRIMARY KEY
```

- Foreign key syntax:

```
ALTER TABLE PART
ADD
(V_CODE) REFERENCES
VENDOR;
```

- Check constraint syntax:

```
ALTER TABLE PART
ADD
CHECK (PART_PRICE
>= 0);
```

# ALTERING TABLE STRUCTURES (3 OF 3)

## 01

### Dropping a column

- Syntax:
  - ALTER TABLE VENDOR
  - DROP COLUMN V\_ORDER;

## 02

### Deleting a table from the database

- Syntax:
  - DROP TABLE PART;

# DATA MANIPULATION COMMANDS (1 OF 2)

## Adding

- Adding table rows
- INSERT command syntax:
- Inserting rows with null attributes: use NULL entry
- Inserting rows with optional attributes: indicate attributes that have required values

## Inserting

- Inserting table rows with a SELECT subquery
- Add multiple rows to a table, using another table as the source, at the same time
- SELECT syntax:

## Saving

- Saving table changes
- COMMIT command syntax:

# DATA MANIPULATION COMMANDS (2 OF 2)

## Updating

Updating table rows

- UPDATE command is used to modify data in a table
- UPDATE syntax:

## Deleting

Deleting table rows

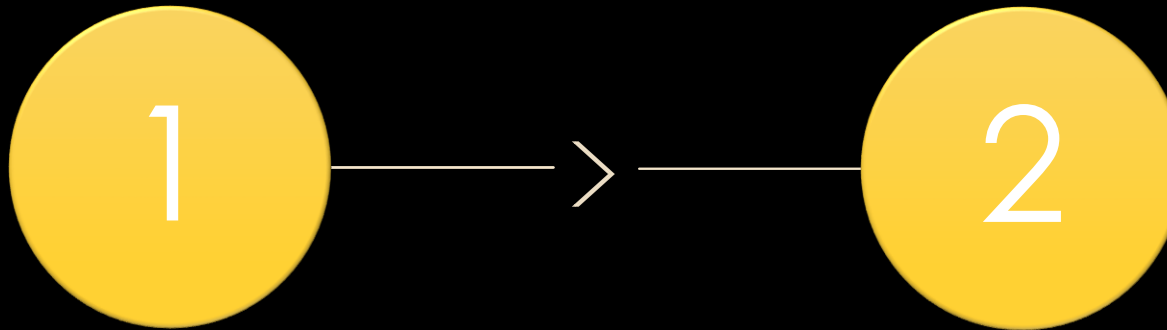
- DELETE statement syntax:

## Restoring

Restoring table contents

- ROLLBACK command is used restore the database to its previous condition

# VIRTUAL TABLES: CREATING A VIEW



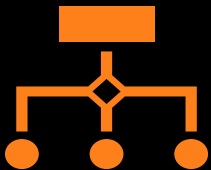
View: virtual table based on a SELECT query

- Base tables: tables on which the view is based

CREATE VIEW statement: data definition command that stores the subquery specification in the data dictionary

- CREATE VIEW command syntax:
- CREATE VIEW *viewname* AS SELECT *query*

# UPDATABLE VIEWS



## Used to update attributes

Batch update routine: pools multiple transactions into a single batch to update a master table field in a single operation



## Updatable view restrictions

GROUP BY expressions or aggregate functions cannot be used

Set operators cannot be used

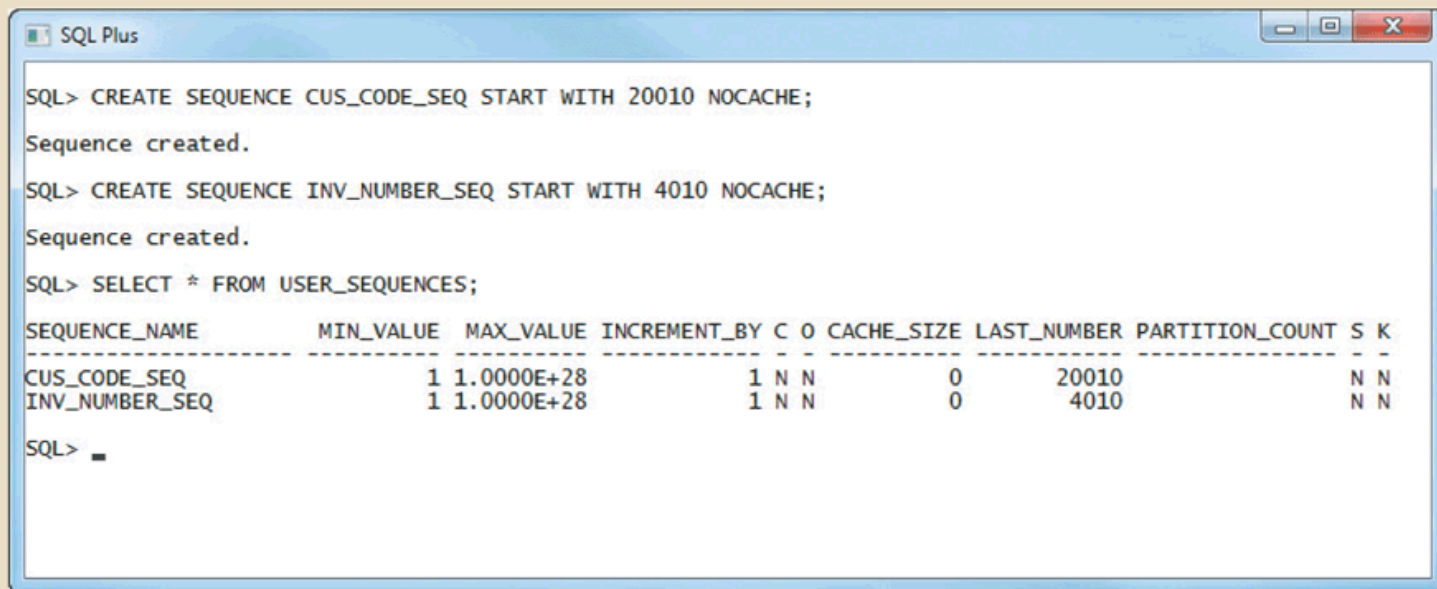
Most restrictions are based on the use of JOINS or group operators in views

# SEQUENCES (1 OF 2)

- Many similarities in the use of sequences across these DBMSs
  - Independent object in the database
  - Have a name and can be used anywhere a value expected
  - Not tied to a table or column
  - Generate a numeric value that can be assigned to any column in any table
  - Table attribute with an assigned value can be edited and modified

# SEQUENCES (2 OF 2)

FIGURE 8.10 ORACLE SEQUENCE



```
SQL> CREATE SEQUENCE CUS_CODE_SEQ START WITH 20010 NOCACHE;  
Sequence created.  
SQL> CREATE SEQUENCE INV_NUMBER_SEQ START WITH 4010 NOCACHE;  
Sequence created.  
SQL> SELECT * FROM USER_SEQUENCES;
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	C	O	CACHE_SIZE	LAST_NUMBER	PARTITION_COUNT	S	K
CUS_CODE_SEQ	1	1.0000E+28	1	N	N	0	20010		N	N
INV_NUMBER_SEQ	1	1.0000E+28	1	N	N	0	4010		N	N

```
SQL> _
```



# PROCEDURAL SQL (1 OF 3)

Performs a conditional or looping operation by isolating critical code and making all application programs call the shared code

- Better maintenance and logic control

Persistent stored module (PSM): block of code

- Contains standard SQL statements and procedural extensions that is stored and executed at the DBMS server

# PROCEDURAL SQL (2 OF 3)

## Procedural Language SQL (PL/SQL)

- Use and storage of procedural code and SQL statements within the database
- Merging of SQL and traditional programming constructs

Procedural code is executed as a unit by DBMS when invoked by end user

- Anonymous PL/SQL blocks
- Triggers
- Stored procedures
- PL/SQL functions

# PROCEDURAL SQL (3 OF 3)

<b>Table 8.4 PL/SQL BASIC DATA TYPES</b>	
<b>DATA TYPE</b>	<b>DESCRIPTION</b>
CHAR	Character values of a fixed length; for example: W_ZIP CHAR(5)
VARCHAR2	Variable-length character values; for example: W_FNAME VARCHAR2(15)
NUMBER	Numeric values; for example: W_PRICE NUMBER(6,2)
DATE	Date values; for example: W_EMP_DOB DATE
%TYPE	Inherits the data type from a variable that you declared previously or from an attribute of a database table; for example: W_PRICE PRODUCT.P_PRICE%TYPE Assigns W_PRICE the same data type as the P_PRICE column in the PRODUCT table

# TRIGGERS (1 OF 2)

Procedural SQL code automatically invoked by RDBMS when given data manipulation event occurs

## Parts of a trigger definition

- Triggering timing: indicates when trigger's PL/SQL code executes
- Triggering event: statement that causes the trigger to execute
  - Triggering level: statement- and row-level
  - Triggering action: PL/SQL code enclosed between the BEGIN and END keywords

# TRIGGERS (2 OF 2)

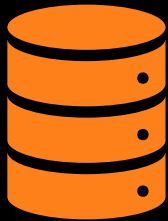
DROP TRIGGER  
trigger\_name  
command

- Deletes a trigger without deleting the table

Trigger action based  
on conditional DML  
predicates

- Actions depend on the type of DML statement that fires the trigger

# STORED PROCEDURES



## **Named collection of procedural and SQL statements**

Stored in the database  
Can be used to encapsulate and  
represent business transactions



## **Advantages**

Reduce network traffic and increase  
performance  
Decrease code duplication by means of  
code isolation and code sharing

# PL/SQL PROCESSING WITH CURSORS (1 OF 3)

Cursor: special construct used to hold data rows returned by a SQL query

- Implicit cursor: automatically created when SQL statement returns only one value
- Explicit cursor: holds the output of a SQL statement that may return two or more rows
- Syntax:
  - `CURSOR cursor_name IS select-query;`

Cursor-style processing involves retrieving data from the cursor one row at a time

- Current row is copied to PL/SQL variables

# PL/SQL PROCESSING WITH CURSORS (2 OF 3)

Table 8.5 Cursor Processing Commands	
Cursor Command	Explanation
OPEN	Opening the cursor executes the SQL command and populates the cursor with data, opening the cursor for processing. The cursor declaration command only reserves a named memory area for the cursor; it does not populate the cursor with the data. Before you can use a cursor, you need to open it. For example: <code>OPEN cursor_name</code>
FETCH	Once the cursor is opened, you can use the FETCH command to retrieve data from the cursor and copy it to the PL/SQL variables for processing. The syntax is: <code>FETCH cursor_name INTO variable1 [, variable2, ...]</code> The PL/SQL variables used to hold the data must be declared in the DECLARE section and must have data types compatible with the columns retrieved by the SQL command. If the cursor's SQL statement returns five columns, there must be five PL/SQL variables to receive the data from the cursor. This type of processing resembles the one-record-at-a-time processing used in previous database models. The first time you fetch a row from the cursor, the first row of data from the cursor is copied to the PL/SQL variables; the second time you fetch a row from the cursor, the second row of data is placed in the PL/SQL variables; and so on.
CLOSE	The CLOSE command closes the cursor for processing.



# PL/SQL PROCESSING WITH CURSORS (3 OF 3)

Table 8.6 Cursor Attributes	
Attribute	Description
%ROWCOUNT	Returns the number of rows fetched so far. If the cursor is not OPEN, it returns an error. If no FETCH has been done but the cursor is OPEN, it returns 0.
%FOUND	Returns TRUE if the last FETCH returned a row, and FALSE if not. If the cursor is not OPEN, it returns an error. If no FETCH has been done, it contains NULL.
%NOT FOUND	Returns TRUE if the last FETCH did not return any row, and FALSE if it did. If the cursor is not OPEN, it returns an error. If no FETCH has been done, it contains NULL.
%ISOPEN	Returns TRUE if the cursor is open (ready for processing) or FALSE if the cursor is closed. Remember, before you can use a cursor, you must open it.

# PL/SQL STORED FUNCTIONS

- Stored function: named group of procedural and SQL statements that returns a value
  - Indicated by a RETURN statement in its program code
- Can be invoked only from within stored procedures or triggers
  - Cannot be invoked from SQL statements unless the function follows some very specific compliance rules

# EMBEDDED SQL (1 OF 4)

- SQL statements contained within an application programming language
  - Host language: any language that contains embedded SQL statements
- Differences between SQL and procedural languages
  - Run-time mismatch
    - SQL is executed one instruction at a time
    - Host language runs at client side in its own memory space
  - Processing mismatch
    - Conventional programming languages process one data element at a time
    - Newer programming environments manipulate data sets in a cohesive manner
  - Data type mismatch
    - Data types provided by SQL might not match data types used in different host languages

# EMBEDDED SQL (2 OF 4)

- Embedded SQL framework defines:
  - Standard syntax to identify embedded SQL code within the host language
  - Standard syntax to identify host variables
  - Communication area used to exchange status and error information between SQL and host language

# EMBEDDED SQL (3 OF 4)

Table 8.7 SQL Status and Error Reporting Variables		
Variable Name	Value	Explanation
SQLCODE		Old-style error reporting supported for backward compatibility only; returns an integer value (positive or negative)
	0	Successful completion of command
	100	No data; the SQL statement did not return any rows and did not select, update, or delete any rows
	-999	Any negative value indicates that an error occurred
SQLSTATE		Added by SQL-92 standard to provide predefined error codes; defined as a character string (5 characters long)
	00000	Successful completion of command
		Multiple values in the format XXYYY where: XX-> represents the class code YYY-> represents the subclass code

# EMBEDDED SQL (4 OF 4)

- Static SQL: programmer uses predefined SQL statements and parameters
  - SQL statements will not change while application is running
- Dynamic SQL: SQL statement is generated at run time
  - Attribute list and condition are not known until end user specifies them
  - Slower than static SQL
  - Requires more computer resources
  - Inconsistent levels of support and incompatibilities among DBMS vendors

# SUMMARY (1 OF 2)

- The ANSI standard data types are supported by all RDBMS vendors in different ways
- The basic data definition commands allow you to create tables and indexes
- Data manipulation commands allow you to add, modify, and delete rows from tables
- Views can be created to expose subsets of data to end users primarily for security and privacy reasons
- In Oracle and SQL Server, sequences may be used to generate values to be assigned to a record
- Procedural Language SQL (PL/SQL) can be used to create triggers, stored procedures, and PL/SQL functions

# SUMMARY (2 OF 2)

- A stored procedure is a named collection of SQL statements
- When SQL statements are designed to return more than one value inside the PL/SQL code, a cursor is needed
- Embedded SQL refers to the use of SQL statements within an application programming language such as Visual Basic .NET, C#, COBOL, or Java



# REFERENCES

- Concepts of Database Management, 10th Edition | Lisa Friedrichsen, Lisa Ruffolo, Ellen Monk, Joy Starks, Philip Pratt, Mary Last | ISBN: 978- 0357422083 © 2020 | Publisher Course Technology – Cengage Learning
- Database Concepts, 9th Edition | David M. Kroenke, David Auer, David J. Auer, Scott L. Vandenberg, Robert C. Yoder | ISBN: 978- 0135188392 © 2020 | Publisher Pearson Education
- Database System Concepts 7E, Abraham Silberschatz, Henry F. Korth, S. Sudarshan ©2020 McGraw Hill
- DATABASE SYSTEMS Design, Implementation, and Management 13E, Carlos Coronel | Steven Morris, SBN-13: 978-1337627900 © 2018 Cengage Learning, Inc.
- Microsoft SQL documentation - <https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15>
- Educational SQL resources - <https://docs.microsoft.com/en-us/sql/sql-server/educational-sql-resources?view=sql-server-ver15>
- SQL Server Technical Documentation - <https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15>
- SQL Developer Documentation - [https://docs.oracle.com/cd/E12151\\_01/index.htm](https://docs.oracle.com/cd/E12151_01/index.htm)