

# Database Ch. 2a (to page 82)

## Chapter 2 - Intro to SQL

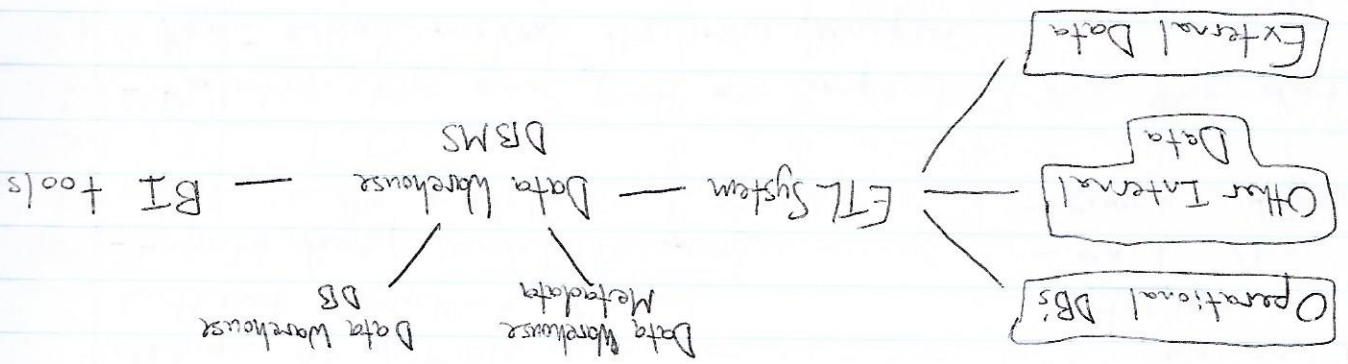
ad-hoc: created by the user as needed  
 Structured Query language: universal query language of relational DBMS products  
 Online Transaction Processing System: records sales transactions; backbone of operations for businesses

I.

## B.I. Systems & Data Warehouses

Data Warehouses: all systems that have data, programs, & personnel that specialize in preparation of data for B.I. systems (Figure 2-3)  
 B.I. System: used to support right decisions by producing info for assessment, analysis, planning, and control  
 ETL System: Extract, Transform, Load; reads, cleans, & prepares the data for BI processing

Figure 2-3



SKU: Stock Keeping Unit  
 Schema: a complete logical view of the db, containing all tables, their columns, PK's, FK's

## Data Extracts Are Common

- they happen all the time in industry, especially in B.I. Systems
- ad-hoc SQL queries are very practical

## II. SQL Background

- developed in late 1970's by IBM
- became a national standard by 1986/1987
- Since then, many versions have been released over the years w/ continual new features
  - \* SQL:2008 → support for Extensible Markup Lang (XML) file
- SQL is a data sublanguage

SQL Stmts are divided into 5 categories:

- 1) Data Definition Lang (DDL): create tables, relationships, etc.
- 2) Data Manipulation Lang (DML): query, insert, modify, & delete data
- 3) SQL/Persistent Stored Modules (SQL/PSM): SQL(+) procedural programming to provide programmability w/in SQL framework
- 4) Transaction Control Lang (TCL): marks transaction bounds & control transaction behavior
- 5) Data Control Lang (DCL): used to grant db permissions so users / groups can perform various ops on the data

\* 4 main actions for DML are: CRUD → Create, Read, Update, Delete

## III. The SQL Select / From / Where Framework

- Select specifies columns to be listed
- From specifies tables to be used
- Where specifies rows
- SQL Comment Syntax: /\* Comment here \*/
- \* wildcard = all

## IV. Submitting SQL Stmts to the DBMS

Microsoft Access 2013 runs ANSI-89 SQL ...  
to change versions:

File → Options → Object Designers → choose the ANSI 92 db & make it default



## Access 2013 Query


- can change view to SQL view to write the SQL
- CREATE → Query Design → select tables needed
- Click Run when ready
- Queries can be saved (Enterprise DBMS wouldn't let)

SQL Server 2014 - I've written queries in here before

- can be opened and ran as ~~sql~~ command or set of commands

- save it just like saving a word document

Oracle DBMS - query running is self-explanatory

MySQL Workbench - " " → execute button = 

- can save SQL scripts in either of the above DBMS
- to open, File → Open SQL script

## IV.

### SQL Enhancements for Querying a Single Table

- eliminate duplicate data: **DISTINCT** Keyword
- Ex. Select Distinct column from table(s);

### Keywords

- 1) **Distinct**: eliminates duplicate records
- 2) **Top & numofrows**: specify how many records are in ~~set~~ the ResultSet
- 3) **Top & % Percent**: displays whatever percentage of the total query that was specified (\*only on SQL Server)

4) **Where**: gives opportunity to give keywords, like a Google search

5) **SQL Comparison Operators**: 14 operators; Figure 2-23 pg 63

- use single quotes 'x' if text or dates

6) **Order By**: after from clause; a column; specify DESC or ASC after column name, or will default to

ASC; can order by multiple columns

### WHERE clause options

- compound clauses, ranges, and wildcards

### Compound Clauses: AND, OR, NOT

- query requires these clauses are met
- NOT if you don't want a specific record
- Substitute for AND: NOT IN or IN operators
  - specify a set of values to be or not to be used in the query (Ex. WHERE Buyer NOT IN ('Tanner', 'Alex', 'Joel'));

### Note:

- a row qualifies for IN if column is = to any of the values in the parenthesis
- same concept for NOT IN

### Ranges of values: BETWEEN, NOT BETWEEN

- includes or excludes a range of numerical values Ex. price range, date range, etc.
- ~~could use <, >, <=, >=~~
- substitute for <, >, <=, >=
- use this to find a certain sequence of character strings as well: LIKE ~~and~~ " and NOT LIKE "

### Wildcards:

1.) underscore (\_): represents a single, unspecified character in a specific position in a char sequence

2.) percent (%): any sequence or contiguous, unspecified characters (including spaces)

Ex. LIKE 'Pete%' to find all ~~first names~~ ~~that~~ records w/ first name Pete

Ex. LIKE '%Tomac%' to find all ~~specific~~ records w/ 'Tomac' ~~word~~ in a set of text / description

Note: if a column type is CHAR(8) and an entry is less than 8 chars, the DBMS will fill the rest of the entry w/ spaces Fix: use RTRIM function (LIKE RTRIM('four'));



Null value: a missing data value

- if we want to include/exclude rows w/ NULL values, use IS NULL and IS NOT NULL
- IS NULL will only return ~~values~~ records w/ a NULL column

- IS keyword is equivalent to = comparison operator

## VI. Performing Calculations in SQL Queries

### Using Built-In SQL Aggregate Functions

- SUM, AVG, MIN, MAX, COUNT

Ex. Select SUM(Ordertotal) From Retail-order;

Result = 1 all answer w/ no column name  
Fix = AS Keyword

- gives answer call a column name

to go w/ the answer

Ex. Select SUM(ExtendedPrice) AS OrderItemSum  
- can name answer column whatever

COUNT: counts # of rows in a table

Ex. Select COUNT(\*) AS NumOfRows From Orders;

- only counts rows w/ NOT NULL values

Ex. Select COUNT(Riders) AS West250 From 250Riders;

- counts # of records in Riders column

Ex. no repeats! Select COUNT(DISTINCT Riders) ...

### Limitations to built-in functions:

1) cannot combine table column names (except for grouping)

Ex. Select Dept, COUNT(\*) From SUVs Data;

2) cannot use w/ WHERE clause

- WHERE operates on rows & aggregate functions operate on columns

## SQL Expressions in SQL Select Stmts

- can write math expressions

SQL Expression: a formula or set of values that determines the exact results of a query

Ex. select SKU, (Qty \* Price) AS EP From OrderItem;

- it's like an implied = sign ( $EP = Qty * Price$ )

- compare the math to the actual data:

Ex. Select SKU, (Qty \* Price) AS EP, Extended Price  
From OrderItem;

## Expressions in Where clause

- same scenario as above:

Ex. Select ~~Order~~ SKU From OrderItem WHERE  
(Qty \* Price) <> ExtendedPrice;

<> → not equal to

- if ~~all~~ the query returns an empty set,  
the data in ExtendedPrice is correct

+ → concatenation operator or CONCAT() in MySQL

- used to combine multiple columns into one

Ex. Select SKU, CONCAT(Buyer, 'in', Department)  
AS Sponsor From SKU-DATA OrderBy SKU;

NOTE: could also add RTRIM into the CONCAT() function to remove extra whitespace if needed

## Ch. 26 Notes

### Grouping Rows in SQL Select Stmts