

COMP100 Introduction to Information Technology

Structured Query Language (SQL)

Structured Query Language

- **Structured Query Language (SQL)** was developed by the IBM Corporation in the late 1970's.
- SQL was endorsed as a United States national standard by the American National Standards Institute (ANSI) in 1992 [**SQL-92**].

2010/11/11

2

SQL as a Data Sublanguage

- SQL is not a full featured **programming language**.
 - C, C#, Java
- SQL is a **data sublanguage** for creating and processing database data and metadata.
- SQL is ubiquitous in enterprise-class DBMS products.
- SQL programming is a critical skill.

SQL DDL and DML

- SQL statements can be divided into two categories:
 - **Data definition language (DDL)** statements
 - Used for creating tables, relationships and other structures.
 - **Data manipulation language (DML)** statements.
 - Used for queries and data modification.

2010/11/11

3

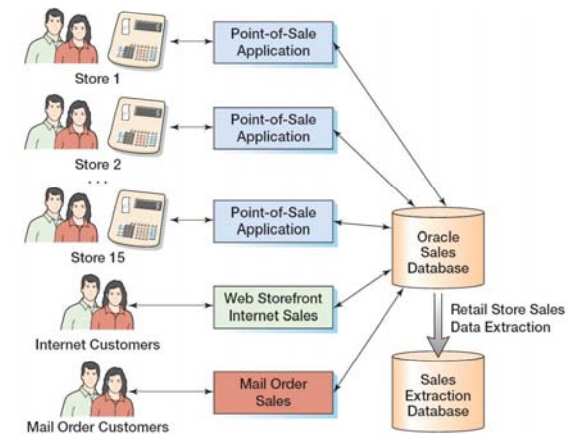
2010/11/11

4

Cape Codd Outdoor Sports

- Cape Codd Outdoor Sports is a fictitious company based on an actual outdoor retail equipment vendor.
- Cape Codd Outdoor Sports:
 - Has 15 retail stores in the US and Canada.
 - Has a on-line Internet store.
 - Has a (postal) mail order department.
- All retail sales recorded in an Oracle database.

Cape Codd Retail Sales Structure



Cape Codd Retail Sales Data Extraction

- The Cape Codd marketing department needs an analysis of in-store sales.
- The entire database is not needed for this, only an **extraction** of retail sales data.
- The data is extracted by the IS department from the **operational database** into a **separate, off-line database** for use by the marketing department.
- Three tables are used: **RETAIL_ORDER**, **ORDER_ITEM**, and **SKU_DATA** (SKU = Stock Keeping Unit).
- The extracted data is **converted** as necessary:
 - Into a different DBMS – MS SQL Server
 - Into different columns – OrderDate becomes OrderMonth and OrderYear.

Extracted Retail Sales Data Format

Table	Column	Date Type
RETAIL_ORDER	OrderNumber	Integer
	StoreNumber	Integer
	StoreZip	Character (9)
	OrderMonth	Character (12)
	OrderYear	Integer
	OrderTotal	Currency
ORDER_ITEM	OrderNumber	Integer
	SKU	Integer
	Quantity	Integer
	Price	Currency
	ExtendedPrice	Currency
SKU_DATA	SKU	Integer
	SKU_Description	Character (35)
	Department	Character (30)
	Buyer	Character (30)

Retail Sales Extract Tables [in MS SQL Server]

2010/11/11

9

The SQL SELECT Statement

- The fundamental framework for SQL query states is the **SQL SELECT statement**.
 - **SELECT** {ColumnName(s)}
 - **FROM** {TableName(s)}
 - **WHERE** {Conditions}
- All SQL statements end with a **semi-colon (;)**.

2010/11/11

10

Specific Columns on One Table

```
SELECT Department, Buyer
FROM SKU_DATA;
```

	Department	Buyer
1	Water Sports	Pete Hansen
2	Water Sports	Pete Hansen
3	Water Sports	Nancy Meyers
4	Water Sports	Nancy Meyers
5	Camping	Cindy Lo
6	Camping	Cindy Lo
7	Climbing	Jerry Martin
8	Climbing	Jerry Martin



Return duplicated records

2010/11/11

11

Specifying Column Order

```
SELECT Buyer, Department
FROM SKU_DATA;
```

	Buyer	Department
1	Pete Hansen	Water Sports
2	Pete Hansen	Water Sports
3	Nancy Meyers	Water Sports
4	Nancy Meyers	Water Sports
5	Cindy Lo	Camping
6	Cindy Lo	Camping
7	Jerry Martin	Climbing
8	Jerry Martin	Climbing

2010/11/11

12

The DISTINCT Keyword

```
SELECT DISTINCT Buyer, Department  
FROM SKU_DATA;
```

	Buyer	Department
1	Cindy Lo	Camping
2	Jery Martin	Climbing
3	Nancy Meyers	Water Sports
4	Pete Hansen	Water Sports

2010/11/11

13

Selecting All Columns: The Asterisk (*) Keyword

```
SELECT *  
FROM SKU_DATA;
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	201000	Half-dome Tent	Camping	Cindy Lo
6	202000	Half-dome Tent Footprint	Camping	Cindy Lo
7	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	302000	Locking carabiner, Oval	Climbing	Jerry Martin

2010/11/11

14

Specific Rows from One Table

```
SELECT *  
FROM SKU_DATA  
WHERE Department = 'Water Sports';
```

NOTE: SQL wants a plain ASCII single quote: 'xyz' !

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

2010/11/11

15

Specific Columns and Rows from One Table

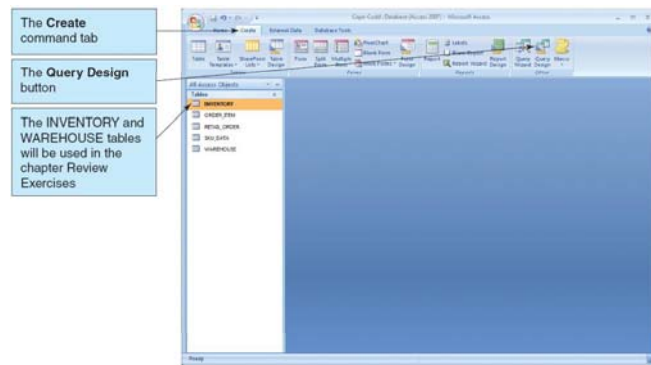
```
SELECT SKU_Description, Buyer  
FROM SKU_DATA  
WHERE Department = 'Climbing';
```

	SKU_Description	Buyer
1	Light Fly Climbing Harness	Jerry Martin
2	Locking carabiner, Oval	Jerry Martin

2010/11/11

16

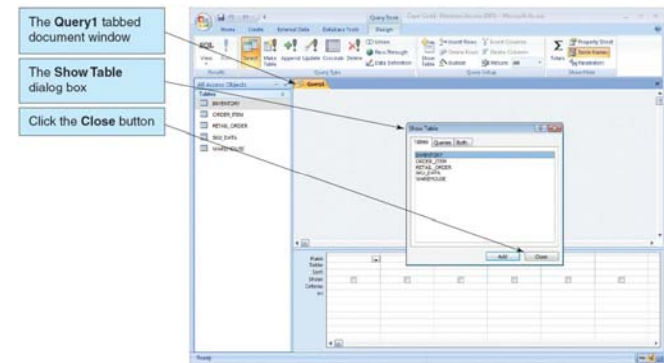
Using MS Access



2010/11/11

17

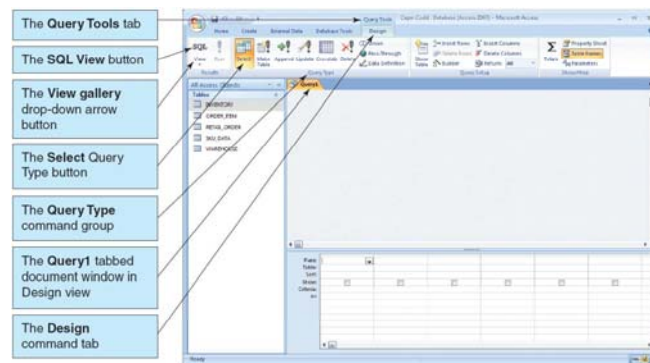
Using MS Access (Continued)



2010/11/11

18

Using MS Access (Continued)



2010/11/11

19

Using MS Access (Continued)



2010/11/11

20

Using MS Access (Continued)



Using MS Access - Results



Using MS Access Saving the Query



Sorting the Results – ORDER BY

```
SELECT *
FROM ORDER_ITEM
ORDER BY OrderNumber, Price;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	202000	1	130.00	130.00
2	1000	201000	1	300.00	300.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	101200	1	50.00	50.00
6	3000	101100	2	50.00	100.00
7	3000	100200	1	300.00	300.00

Sort Order: Ascending and Descending

```
SELECT *
FROM ORDER_ITEM
ORDER BY Price DESC, OrderNumber ASC;
```

NOTE: The default sort order is ASC – does not have to be specified.
If you want to sort in descending order, use they keyword DESC

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	3000	100200	1	300.00	300.00
3	1000	202000	1	130.00	130.00
4	2000	101100	4	50.00	200.00
5	2000	101200	2	50.00	100.00
6	3000	101200	1	50.00	50.00
7	3000	101100	2	50.00	100.00

2010/11/11

25

WHERE Clause Options - AND

```
SELECT *
FROM SKU_DATA
WHERE Department = 'Water Sports'
AND Buyer = 'Nancy Meyers';
```

	SKU	SKU_Description	Department	Buyer
1	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

2010/11/11

26

WHERE Clause Options - OR

```
SELECT *
FROM SKU_DATA
WHERE Department = 'Camping'
OR Department = 'Climbing';
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Footprint	Camping	Cindy Lo
3	301000	Light Fly Climbing Harness	Climbing	Jery Martin
4	302000	Locking carabiner, Oval	Climbing	Jery Martin

2010/11/11

27

WHERE Clause Options - IN

```
SELECT *
FROM SKU_DATA
WHERE Buyer IN ('Nancy Meyers',
'Cindy Lo', 'Jerry Martin');
```

	SKU	SKU_Description	Department	Buyer
1	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
3	201000	Half-dome Tent	Camping	Cindy Lo
4	202000	Half-dome Tent Footprint	Camping	Cindy Lo
5	301000	Light Fly Climbing Harness	Climbing	Jery Martin
6	302000	Locking carabiner, Oval	Climbing	Jery Martin

2010/11/11

28

WHERE Clause Options – NOT IN

```
SELECT *
FROM SKU_DATA
WHERE Buyer NOT IN ('Nancy Meyers',
    'Cindy Lo', 'Jerry Martin');
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen

WHERE Clause Options – Ranges with BETWEEN

```
SELECT *
FROM ORDER_ITEM
WHERE ExtendedPrice
    BETWEEN 100 AND 200;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	2000	101100	4	50.00	200.00
2	3000	101100	2	50.00	100.00
3	2000	101200	2	50.00	100.00
4	1000	202000	1	130.00	130.00

WHERE Clause Options – Ranges with Math Symbols

```
SELECT *
FROM ORDER_ITEM
WHERE ExtendedPrice >= 100
    AND ExtendedPrice <= 200;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	2000	101100	4	50.00	200.00
2	3000	101100	2	50.00	100.00
3	2000	101200	2	50.00	100.00
4	1000	202000	1	130.00	130.00

WHERE Clause Options – LIKE and Wildcards

- The SQL keyword LIKE can be combined with wildcard symbols:
 - SQL 92 Standard (SQL Server, Oracle, etc.):
 - `_` = Exactly one character
 - `%` = Any set of zero or more characters
 - MS Access (based on MS DOS)
 - `?` = Exactly one character
 - `*` = Any set of zero or more characters

WHERE Clause Options – LIKE and Wildcards

```
SELECT *  
FROM SKU_DATA  
WHERE Buyer LIKE 'Pete%';
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen

2010/11/11

33

WHERE Clause Options – LIKE and Wildcards

```
SELECT *  
FROM SKU_DATA  
WHERE SKU_Descripton LIKE '%Tent%';
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Footprint	Camping	Cindy Lo

2010/11/11

34

WHERE Clause Options – LIKE and Wildcards

```
SELECT *  
FROM SKU_DATA  
WHERE SKU LIKE '%2__';
```

	SKU	SKU_Description	Department	Buyer
1	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

2010/11/11

35

SQL Built-in Functions

- There are five SQL Built-in Functions:
 - COUNT
 - SUM
 - AVG
 - MIN
 - MAX

2010/11/11

36

SQL Built-in Functions

```
SELECT SUM (ExtendedPrice)
      AS Order3000Sum
FROM   ORDER_ITEM
WHERE  OrderNumber = 3000;
```

Order3000Sum	
1	450.00

2010/11/11

37

SQL Built-in Functions

```
SELECT SUM (ExtendedPrice) AS OrderItemSum,
      AVG (ExtendedPrice) AS OrderItemAvg,
      MIN (ExtendedPrice) AS OrderItemMin,
      MAX (ExtendedPrice) AS OrderItemMax
FROM   ORDER_ITEM;
```

	OrderItemSum	OrderItemAvg	OrderItemMin	OrderItemMax
1	1180.00	168.5714	50.00	300.00

2010/11/11

38

SQL Built-in Functions

```
SELECT COUNT (*) AS NumberOfRows
FROM   ORDER_ITEM;
```

NumberOfRows	
1	7

2010/11/11

39

SQL Built-in Functions

```
SELECT COUNT
      (DISTINCT Department)
      AS DeptCount
FROM   SKU_DATA;
```

DeptCount	
1	3

2010/11/11

40

Arithmetic in SELECT Statements

```
SELECT Quantity * Price AS EP,
       ExtendedPrice
FROM   ORDER_ITEM;
```

	EP	ExtendedPrice
1	300.00	300.00
2	200.00	200.00
3	100.00	100.00
4	100.00	100.00
5	50.00	50.00
6	300.00	300.00
7	130.00	130.00

2010/11/11

41

The SQL keyword GROUP BY

```
SELECT Department,
       Buyer, COUNT(*) AS
       Dept_Buyer_SKU_Count
FROM   SKU_DATA
GROUP BY
       Department, Buyer;
```

	Department	Buyer	Dept_Buyer_SKU_Count
1	Camping	Cindy Lo	2
2	Climbing	Jerry Martin	2
3	Water Sports	Nancy Meyers	2
4	Water Sports	Pete Hansen	2

2010/11/11

42

Original data

	Department	Buyer
1	Water Sports	Pete Hansen
2	Water Sports	Pete Hansen
3	Water Sports	Nancy Meyers
4	Water Sports	Nancy Meyers
5	Camping	Cindy Lo
6	Camping	Cindy Lo
7	Climbing	Jerry Martin
8	Climbing	Jerry Martin

4 groups

The SQL keyword GROUP BY

- In general, place WHERE before GROUP BY. Some DBMS products do not require that placement, but to be safe, always put WHERE before GROUP BY.
- The **HAVING** operator restricts the groups that are presented in the result.
- There is an ambiguity in statements that include both WHERE and HAVING clauses. The results can vary, so to eliminate this ambiguity SQL *always* applies WHERE before HAVING.

2010/11/11

43

The SQL keyword GROUP BY

```
SELECT Department,
       COUNT(*) AS
       Dept_SKU_Count
FROM   SKU_DATA
WHERE  SKU <> 302000
GROUP BY
       Department
ORDER BY
       Dept_SKU_Count;
```

	Department	Dept_SKU_Count
1	Climbing	1
2	Camping	2
3	Water Sports	4

2010/11/11

44

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	201000	Half-dome Tent	Camping	Cindy Lo
6	202000	Half-dome Tent Footprint	Camping	Cindy Lo
7	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	302000	Locking carabiner, Oval	Climbing	Jerry Martin

The SQL keyword GROUP BY

```
SELECT Department, COUNT(*) AS
      Dept_SKU_Count
FROM   SKU_DATA
WHERE  SKU <> 302000
GROUP BY Department
HAVING COUNT (*) > 1
ORDER BY Dept_SKU_Count;
```

	Department	Dept_SKU_Count
1	Camping	2
2	Water Sports	4

2010/11/11

45

Querying Multiple Tables: Subqueries

```
SELECT SUM (ExtendedPrice) AS Revenue
FROM   ORDER_ITEM
WHERE  SKU IN
      (SELECT SKU
       FROM   SKU_DATA
       WHERE  Department = 'Water Sports');
```

Note: The second SELECT statement is a **subquery**.

	Revenue
1	750.00

2010/11/11

46

Querying Multiple Tables: Subqueries

```
SELECT Buyer
FROM   SKU_DATA
WHERE  SKU IN
      (SELECT SKU
       FROM   ORDER_ITEM
       WHERE  OrderNumber IN
            (SELECT OrderNumber
             FROM   RETAIL_ORDER
             WHERE  OrderMonth = 'January'
                  AND OrderYear = 2004));
```

	Buyer
1	Pete Hansen
2	Nancy Meyers
3	Nancy Meyers

2010/11/11

47

Querying Multiple Tables: Joins

```
SELECT Buyer, ExtendedPrice
FROM   SKU_DATA, ORDER_ITEM
WHERE  SKU_DATA.SKU = ORDER_ITEM.SKU;
```

	Buyer	ExtendedPrice
1	Pete Hansen	300.00
2	Nancy Meyers	200.00
3	Nancy Meyers	100.00
4	Nancy Meyers	100.00
5	Nancy Meyers	50.00
6	Cindy Lo	300.00
7	Cindy Lo	130.00

2010/11/11

48

Querying Multiple Tables: Joins

```
SELECT Buyer, SUM(ExtendedPrice)
      AS BuyerRevenue
FROM   SKU_DATA, ORDER_ITEM
WHERE  SKU_DATA.SKU = ORDER_ITEM.SKU
GROUP BY Buyer
ORDER BY BuyerRevenue DESC;
```

	Buyer	BuyerRevenue
1	Nancy Meyers	450.00
2	Cindy Lo	430.00
3	Pete Hansen	300.00

2010/11/11

49

Querying Multiple Tables: Joins

```
SELECT Buyer, ExtendedPrice, OrderMonth
FROM   SKU_DATA, ORDER_ITEM, RETAIL_ORDER
WHERE  SKU_DATA.SKU = ORDER_ITEM.SKU
      AND ORDER_ITEM.OrderNumber =
      RETAIL_ORDER.OrderNumber;
```

	Buyer	ExtendedPrice	OrderMonth
1	Pete Hansen	300.00	January
2	Nancy Meyers	200.00	December
3	Nancy Meyers	100.00	January
4	Nancy Meyers	100.00	December
5	Nancy Meyers	50.00	January
6	Cindy Lo	300.00	December
7	Cindy Lo	130.00	December

2010/11/11

50

Reference

- Chapter 2, Database Processing -
Fundamentals, Design, and Implementation
(11 Ed.)

2010/11/11

51