

# **NET3000**

# **Database Concepts and SQL**

Professor: Kambiz Ghazinour

**Week 5**

**Database Design**

October 8, 2014

# Create DB

- In our last week Lab we created databases.
- What would be the next step?

# Create Tables

- The SQL CREATE TABLE Statement
- The CREATE TABLE statement is used to create a table in a database.
- Tables are organized into rows and columns; and each table must have a name.
- SQL CREATE TABLE Syntax

```
CREATE TABLE table_name
(
  column_name1 data_type(size),
  column_name2 data_type(size),
  column_name3 data_type(size),
  ....
);
```

# Create Tables

- The `column_name` parameters specify the names of the columns of the table.
- The `data_type` parameter specifies what type of data the column can hold (e.g. `varchar`, `integer`, `decimal`, `date`, etc.).
- The `size` parameter specifies the maximum length of the column of the table.
- **Tip:** Please check our lab 3 instruction where we learned about different data types.

# Create Tables

- Example
- Now we want to create a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City.

```
CREATE TABLE Persons  
(  
  PersonID int,  
  LastName varchar(255),  
  FirstName varchar(255),  
  Address varchar(255),  
  City varchar(255)  
);
```

# What is next?

- OK, now we have a database and a table inside that, what is next?
- Fill in the table?
- How?
- Remember the INSERT statement?

# Remember: Some components of RDBMS (cont.)

- Relation
  - A link between columns of two or multiple tables
- Primary key
  - A table typically has a column or combination of columns that contain values that uniquely identify each row in the table
- Foreign key
  - In a foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or columns in another table.

# Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column.

Typically used with GROUP BY and HAVING clause

Useful aggregate functions:

AVG() - Returns the average value

COUNT() - Returns the number of rows

FIRST() - Returns the first value

LAST() - Returns the last value



# Aggregate Functions

Useful aggregate functions (con't):

MAX() - Returns the largest value

MIN() - Returns the smallest value

SUM() - Returns the sum

Find examples here: [http://www.w3schools.com/sql/sql\\_func\\_avg.asp](http://www.w3schools.com/sql/sql_func_avg.asp)

# Views

- Application program's or individual user's picture of the database
- Less involved than full database
- Offers simplification
- Provides measure of security
  - Sensitive tables or columns omitted where not appropriate

# GROUP BY

The usage of SQL GROUP BY clause is, to divide the rows in a table into smaller groups.

The GROUP BY clause is used with the SELECT statement.

The grouping can happen after retrieves the rows from a table.

When some rows are retrieved from a grouped result against some condition, that is possible with HAVING clause.

# GROUP BY

```
SELECT <column_list> FROM < table name >  
WHERE <condition>  
GROUP BY <columns>  
[HAVING] <condition>;
```

# GROUP BY

The SQL AGGREGATE function can be used to get summary information for every group and these are applied to individual group.

The WHERE clause is used to retrieve rows based on a certain condition, but it can not be applied to grouped result.

In an SQL statement, suppose you are using GROUP BY, if required you can use HAVING instead of WHERE, after GROUP BY.

See the examples bellow to understand how group by works: <http://www.w3resource.com/sql/group-by.php>

# Views

- A view is a virtual table.
- SQL CREATE VIEW Statement
- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

# Views

- SQL CREATE VIEW Syntax

```
CREATE VIEW view_name AS  
SELECT column_name(s)  
FROM table_name  
WHERE condition
```

- **Note:** A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

# Views

- SQL CREATE VIEW Examples
- If you have the Northwind database you can see that it has several views installed by default.
- The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table. The view is created with the following SQL:

```
CREATE VIEW [Current Product List] AS  
SELECT ProductID,ProductName  
FROM Products  
WHERE Discontinued=No
```

- We can query the view above as follows:  
SELECT \* FROM [Current Product List]



# Views

- Another view in the Northwind sample database selects every product in the "Products" table with a unit price higher than the average unit price:

```
CREATE VIEW [Products Above Average Price] AS  
SELECT ProductName,UnitPrice  
FROM Products  
WHERE UnitPrice>(SELECT AVG(UnitPrice) FROM Products)
```

- We can query the view above as follows:  
SELECT \* FROM [Products Above Average Price]

# Views

- Another view in the Northwind database calculates the total sale for each category in 1997. Note that this view selects its data from another view called "Product Sales for 1997":

```
CREATE VIEW [Category Sales For 1997] AS  
SELECT DISTINCT CategoryName,Sum(ProductSales) AS  
CategorySales  
FROM [Product Sales for 1997]  
GROUP BY CategoryName
```

- We can query the view above as follows:  
SELECT \* FROM [Category Sales For 1997]
- We can also add a condition to the query. Now we want to see the total sale only for the category "Beverages":
- SELECT \* FROM [Category Sales For 1997]  
WHERE CategoryName='Beverages'

# Views

## SQL Updating a View

- You can update a view by using the following syntax:  

```
CREATE OR REPLACE VIEW view_name AS  
SELECT column_name(s)  
FROM table_name  
WHERE condition
```
- Now we want to add the "Category" column to the "Current Product List" view. We will update the view with the following SQL:  

```
CREATE VIEW [Current Product List] AS  
SELECT ProductID,ProductName,Category  
FROM Products  
WHERE Discontinued=No
```

# Views

## SQL Dropping a View

- You can delete a view with the DROP VIEW command.  
DROP VIEW view\_name

# Advantages of Views

- Provides data independence
- Same data viewed by different users in different ways
- Contains only information required by a given user

# Indexes

- Conceptually similar to book index
- Increases data retrieval efficiency
- Automatically assigns record numbers
- Used by DBMS, not by users
- Fields on which index built called Index Key

## Figure 4.10: Customer Table with Record Numbers

RecordNum	CustomerNum	CustomerName	...	Balance	CreditLimit	RepNum
1	148	Al's Appliance and Sport	...	\$6,550.00	\$7,500.00	20
2	282	Brookings Direct	...	\$431.50	\$10,000.00	35
3	356	Ferguson's	...	\$5,785.00	\$7,500.00	65
4	408	The Everything Shop	...	\$5,285.25	\$5,000.00	35
5	462	Bargains Galore	...	\$3,412.00	\$10,000.00	65
6	524	Kline's	...	\$12,762.00	\$15,000.00	20
7	608	Johnson's Department Store	...	\$2,106.00	\$10,000.00	65
8	687	Lee's Sport and Appliance	...	\$2,851.00	\$5,000.00	35
9	725	Deerfield's Four Seasons	...	\$248.00	\$7,500.00	35
10	842	All Season	...	\$8,221.00	\$7,500.00	20

**Figure 4.11: Customer Table Index on CustomerNum**

CustomerNum	RecordNum
148	1
282	2
356	3
408	4
462	5
524	6
608	7
687	8
725	9
842	10



# Pros/Cons of Indexes

- Can be added or dropped without loss of function
- Can make retrieval more efficient
- Occupies space that might be required for other functions
- DBMS must update index whenever corresponding data are updated

# SQL to Create Index

```
CREATE INDEX CustomerName  
ON Customer (CustomerName)  
;
```

# Creating Indexes

- Single-field index – an index whose key is a single field
- Multiple-field index
  - An index with more than one key field
  - List the most important key first
  - If data for either key appears in descending order, follow the field name with the letters DESC

# SQL to Delete Index

```
DROP INDEX RepBal
```

```
;
```

# Security

- Prevention of unauthorized access to database
- DCL statements provide control over access to objects in database
- Three SQL security mechanisms:

# Security (con't)

- GRANT to allow users access to an object  
[WITH GRANT OPTION]

```
GRANT SELECT ON Customer TO JONES;
```

- DENY to disallow user access to an object

```
DENY SELECT ON Customer TO JONES;
```

- REVOKE to remove a GRANT or a DENY.

```
REVOKE SELECT ON Customer FROM JONES;
```

# Integrity Rules

- Related to foreign keys and primary keys
- Defined by Dr. E.F. Codd
- Entity integrity
  - No field that is part of the primary key may accept null values

# Integrity Rules (con't)

- To specify primary key, enter a PRIMARY KEY clause in either an ALTER TABLE or a CREATE TABLE command
- Foreign key – a field (or collection of fields) in a table whose value is required to match the value of the primary key for a second table



# Referential integrity

- If Table A contains a foreign key matching the primary key of Table B, then values must match for some row in Table B or be null
- Usually a foreign key is in a different table from the primary key it is required to match
- The only restriction is that the foreign key must have a name that is different from the primary key because the fields are in the same table

# Cascade Delete and Update

- Cascade delete - ensures that the deletion of a master record deletes all records in sub tables related to it
- Cascade update – ensures that changes made to the primary key of the master table are also made in the related records

# Structure Changes

- Can change the database structure
  - By adding and removing tables and fields
  - By changing the characteristics of existing fields
  - By creating and dropping indexes
- The exact manner in which these changes are accomplished varies from one system to another
- Most systems allow all of these changes to be made quickly and easily
- Made using the SQL ALTER TABLE command

# Structure Changes – Add and Change

## **Adding new field**

```
ALTER TABLE Customer  
ADD CustType CHAR(1)  
;
```

## **Changing field properties**

```
ALTER TABLE Customer  
CHANGE COLUMN CustomerName TO CHAR(50)  
;
```

# Structure Changes - Delete

Deleting field

```
ALTER TABLE Part  
DELETE Warehouse  
;
```

Delete SQL Table

```
DROP TABLE SmallCust  
;
```

# System Catalog

- Information about database kept in system catalog
- Maintained by DBMS
- Example catalog has two tables
  - Systables – information about the tables known to SQL
  - Syscolumns – information about the columns or fields within these tables

# System Catalog (con't.)

- Other possible tables
  - Sysindexes – information about the indexes that are defined on these tables
  - Sysviews – information about the views that have been created

# Summary

- Views - used to give each user his or her own view of the data in a database
- View is defined in structured query language (SQL) by using a defining query
- Indexes are often used to facilitate data retrieval from the database
- Security is provided in SQL systems using the GRANT and REVOKE commands
- Entity integrity is the property that states that no field that is part of the primary key can accept null values



# Summary

- Referential integrity - property stating that the value in any foreign key field must either be null or match an actual value in the primary key field of another table
- The ALTER TABLE command allows you to add fields to a table, delete fields, or change the characteristics of fields

# Summary

- The DROP TABLE command lets you delete a table from a database
- The system catalog is a feature of many relational DBMSs that stores information about the structure of a database