1.264 Lecture 15

SQL transactions, security, indexes

Download BeefData.csv and Lecture15Download.sql
Next class: Read “Beginning ASP.NET ” chapter 1.
Exercise due after class (5:00)

SQL Server diagrams

- Select “Database Diagrams” under your database
  - New diagram
  - Add tables
  - Make sure the diagram matches the data model

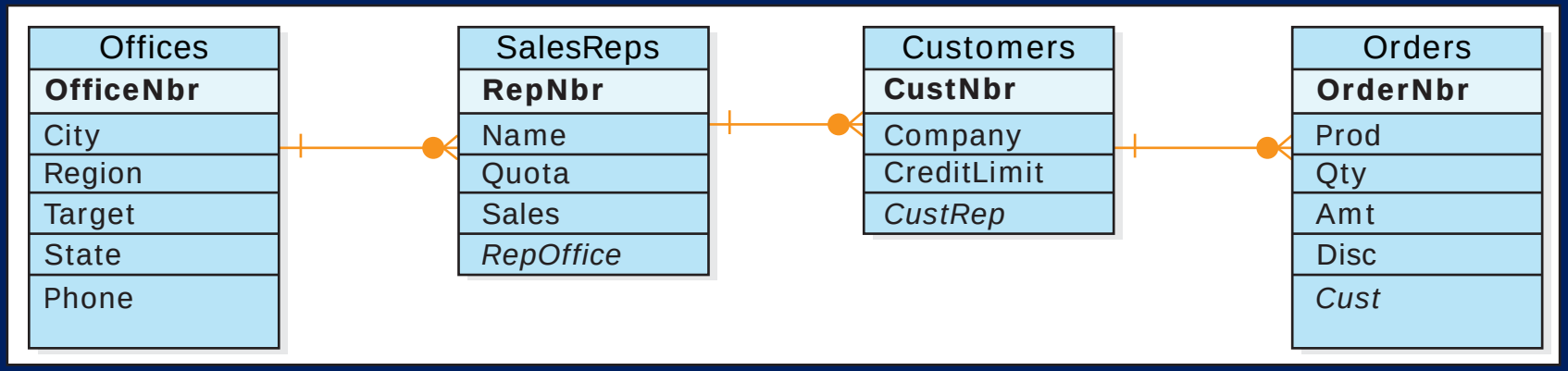

Image by MIT OpenCourseWare.

Don’t need to hand this in

SQL Server script creation

- After building a table, you can save its design
  - Right click on the table in the explorer
  - Select “Script table as”-> Create to (or other options)
  - This allows you to automate creating and working with the database. Write script to query window—hand in.

```
USE [MIT1264F2010]
GO

/****** Object:  Table [dbo].[Customers]    Script Date: 01/30/2011 20:30:51 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Customers](
    [CustNbr] [nchar](3) NOT NULL,
    [Company] [nvarchar](30) NOT NULL,
    [CustRep] [nchar](3) NOT NULL,
    [CreditLimit] [money] NOT NULL,
PRIMARY KEY CLUSTERED
(
    [CustNbr] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Customers]  WITH CHECK ADD FOREIGN KEY([CustRep])
REFERENCES [dbo].[SalesReps] ([RepNbr])
GO
```

SQL Server import/export

- To import data from a file, spreadsheet, etc:
  - Start->All Programs-> MS SQL Server 2012->Import/Export (64 bit)
  - Wizard starts
    - Data source: Flat file source (or other option listed)
    - Browse to file (usually csv—comma separated– or txt)
    - Select database (MIT1264 in this example)
    - New table is created with data from file
  - Exercise: Import Lecture 15 Beef Data.csv
    - No need to hand in

Data definition language (DDL)

- We've explored the data manipulation language (DML) so far: SELECT, INSERT, UPDATE, DELETE
- SQL also has a data definition language (DDL):
  - CREATE DATABASE
  - CREATE TABLE
  - CREATE INDEX (and other CREATE statements)
  - ALTER TABLE
  - ALTER VIEW (and other ALTER statements)
  - DROP DATABASE
  - DROP TABLE
  - DROP VIEW (and other DROP statements)
- You've seen some of these in the initial .sql file that created our database, and in the script example earlier

Transactions

- Group of operations often must be treated as atomic unit
  - Start transaction
    - Insert OrderHeader
    - While more OrderDetail (line items) exist:
      - Select Part
      - Update Part inventory
      - Insert OrderDetail row
  - Commit transaction if everything succeeds
  - Roll back transaction if any error occurs:
    - In Order Header
    - In OrderDetail
    - Server crashes
    - Disk crashes
    - Network dies
    - Etc.

Transaction properties (ACID)

- Atomicity. Either all of transactions are executed or all are rolled back
  - Account transfer debit and credit both succeed or fail
- Consistency. Only legal states can exist
  - If order detail cannot be written, order header is rolled back
- Isolation. Results not seen by other transactions until the transaction is complete
  - Account transfer debit and credit either both seen or neither is seen
- Durability. Data is persistent even if hardware or software crashes: What is written on the disk is correct
  - Account balance is maintained

Transactions

- Multi-user databases have other transaction issues
- Two database actions conflict if one or both are write operations. Examples of problems:
  - Lost updates:
    - 7 parts in inventory
    - Transactions 1 and 2 simultaneously read 7 as the current quantity
    - Transaction 1 finishes first, adds 3 parts, writes 10 as quantity
    - Transaction 2 finishes second, subtracts 5 parts, writes 2 as quantity!
  - Uncommitted changes:
    - Transaction 1 adds 3 parts, writes 10 as quantity
    - Transaction 2 reads 10 as quantity
    - Transaction 1 aborts (rolls back), leaving transaction 2 with wrong data
  - Databases handle all these cases automatically

Transactions

- Databases use locks for concurrency. One simple scheme is pessimistic locking:
  - Writes obtain exclusive lock on a record, preventing reads or writes
  - Reads obtain nonexclusive locks, allowing other reads but preventing a writer from obtaining an exclusive lock
- Another, with higher performance, is optimistic locking:
  - No locks. Check if row exists and is same after operation
  - If not, issue error and program must retry
- Databases use logs for transactions, rollbacks, recovery.
  - Log file of all changes is written in addition to making the changes in the database. (This is a key bottleneck in software architecture.)
  - Transaction can't be committed until log is written to stable storage
    - Transactions usually committed before tables actually updated on disk
  - If a change is rolled back, the log is read to reverse the transactions.
  - If a system or disk crashes, the log is rerun from the last checkpoint to restore the database.

Transaction example and exercise

```
INSERT Customers VALUES (212, 'Smith Co', 89, 20000)        -- Independent INSERTs
INSERT Orders VALUES (212, 'Lathe', 3, 20000, 0.1)
INSERT Orders VALUES (212, 'Latte', 10, 2, 0.0)

-- INSERTs as a transaction (Usually done in program code, which is simpler)
BEGIN TRAN
INSERT Customers VALUES (213, 'Wang Co', 53, 100000)
IF @@ERROR = 0
  BEGIN
    INSERT Orders VALUES (213, 'Mill', 1, 50000, 0.2)
    IF @@ERROR = 0
      BEGIN
        INSERT Orders VALUES (213, 'Malt', 1, 2, 0.0)
        IF @@ERROR = 0
          COMMIT TRAN
        ELSE
          ROLLBACK TRAN
      END
    ELSE
      ROLLBACK TRAN
  END
ELSE
  ROLLBACK TRAN
```

Exercise: Modify the transaction:
It's in Lecture15Download.sql on Web
INSERT Customer 214
INSERT first order for 214 correctly
INSERT 2nd order incorrectly: leave out the last two fields
Then open Customers and Orders:
Are any of the INSERTs present?
Hand in your changed .sql file

Security (and short exercise)

- Security options
  - Use operating system logon/password (weak) to identify user
    - User gets access to all databases, all tables ("Windows authentication")
  - Use database logon/password (stronger)
    - Restrict access to databases, tables, but can still use all applications
    - "SQL Server authentication": we'll use this for the Web->db connection
  - Application level security (stronger still, but tough to administer)
    - Each application must look in its database to see if user authorized
  - Network level security (strongest), using certificates/tickets
    - Use Kerberos, MS Active Directory, others (covered under security)
- Classes of users: super-user (dba or sa), data owner, data user
- Assignment of database privileges (permissions)
  - GRANT and REVOKE: E.g.,
    - GRANT INSERT ON TableName TO PUBLIC WITH GRANT OPTION
    - REVOKE INSERT ON TableName FROM PUBLIC CASCADE
    - Order matters for GRANTs and REVOKEs. Last one governs.
    - Try these two statements; look at the table properties in client. Hand in.

Indexes

- Index is a separate data object in the database that lists table rows in order to allow rapid lookup
  - Each index for each table is a separate object
  - Primary keys and foreign keys automatically indexed
- Rapid access to indexed columns
  - Each index may be updated when a row is updated, so indexes slow updates, insertions and deletes
  - If a database is mostly read, use many indexes to speed performance
  - If database is mostly updates, use as few indexes as possible
  - Practical maximum of 3 or 4 indexes per table. If others are needed on occasion, add and drop them as needed
  - (Use indexes when working with transit fare, bus data)
- Clustered indexes
  - Physically rearrange rows by a single index to maximize disk access speed

Indexes (and short exercise)

- Customer database
  - Customer ID is primary key
  - We also want to search by:
    - Customer name (last, first)
    - City, state
    - Postal (zip) code
    - Address
  - Index the name, city/state, zip and address
    - Four indexes: slow insert, update, delete, but fast lookup
    - If customer database is fairly stable, this is fine
  - Similar logic for parts catalog, bill of materials, etc.
- Internet search engines use 'text retrieval engines'
  - Index every word in the entire database; count occurrences and rank matches. Recent advances (frequency of links, usage…) enhance this.
- Exercise (hand in):
  - CREATE INDEX IX_Orders ON Orders (Cust, OrderNbr)
  - Use the MIT1264 database

ODBC, ADO.NET, JDBC

- ODBC, etc. are a library of procedures (methods) to connect from an application (Web, Windows, Java) to a database, execute SQL statements and retrieve results
  - SQL syntax based on SQL-92 standard
  - Standard set of error codes
  - Standard way to connect and log on to database
  - Standard representation of data types
  - Standard methods for data type conversions
- These features overcome many nonstandard SQL issues noted in the first SQL lecture
  - We'll use ADO.NET when building Web sites and services
- We will look at database performance and costs in the system architecture lectures later

1.264J / ESD.264J Database, Internet, and Systems Integration Technologies
Fall 2013