

# Physical Design: creating database with SQL Server

CIS 3730

Designing and Managing Data

J.G. Zheng  
Fall 2010



# Overview: 3 Level Database Design

Creating an Entity Relationship Diagram (ERD) and associated data dictionary to represent the reality and capture business data requirements

Conceptual Design

Transforming ERD to relational model: tables, keys (constraints), etc.

Logical Design

Creating the database and other supporting structures based on a specific DBMS

Physical Design

# Physical Design

- ◆ A physical database model adds more implementation level details to the design
  - It is specific to a DBMS product
- ◆ What to design?
  - Designing basic database logical model objects (DBMS specific)
    - ◆ Databases, tables, fields, views
    - ◆ Constraints: primary key, foreign key, uniqueness, referential integrity, default, range
  - Designing additional physical level objects
    - ◆ Files: storage files, partitions, etc.
    - ◆ Performance: indexes, views, etc.
    - ◆ Functional: triggers, scripts (stored procedures), functions, transactions, etc.
    - ◆ Security: users, roles, permissions, etc.

# Creating Databases with SQL Server 2008

- ◆ Using the graphical interface provided by SQL Server Management Studio to create databases interactively
  - See a separate tutorial for this
  
- ◆ Use SQL
  - SQL can define and modify data structures (metadata): database, tables, views, etc.

# SQL DDL

- ◆ DML - data manipulation language
  - Retrieving data: database queries
  - Manipulating data: insert, update, delete
- ◆ DDL - data definition language
  - Today → ■ Defining and modifying data structures (schema): databases, tables, views, etc.
- ◆ DCL - data control language
  - Control data access: permissions, etc.

# Defining Databases

## ◆ CREATE DATABASE

- <http://msdn.microsoft.com/en-us/library/>

## ◆ ALTER DATABASE

- <http://msdn.microsoft.com/en-us/library/ms174269.aspxms176061.aspx>

## ◆ DROP DATABASE

- <http://msdn.microsoft.com/en-us/library/ms178613.aspx>

# Defining Tables

- ◆ DDL table statements include
  - CREATE Table
  - ALTER Table
  - DROP Table

# Create Table

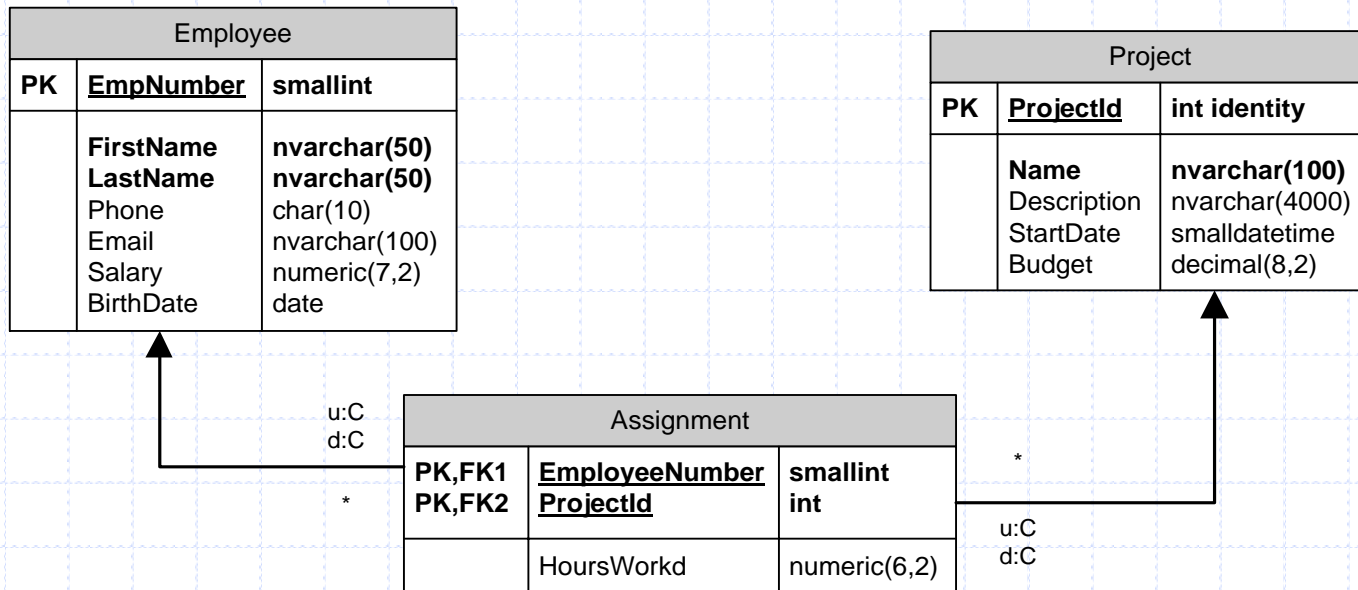
```
CREATE TABLE TableName  
(  
    Column Definitions,  
    Table Constraints  
)
```

## ◆ Reference

- <http://msdn.microsoft.com/en-us/library/ms174979.aspx>

# CREATE TABLE Example

## ◆ A logical model in Visio



## ◆ Other requirements

- Email is unique
- Budget < 500000

Please get the sample SQL script file "SQL-DDL-Example.sql" from the course website (in "Database Examples").

# CREATE TABLE SQL (1)

```
CREATE TABLE Employee
```

```
(
```

```
  EmpNumber
```

```
    SMALLINT
```

```
    PRIMARY KEY,
```

```
  FirstName
```

```
    NVARCHAR(50)
```

```
    NOT NULL,
```

```
  LastName
```

```
    NVARCHAR(50)
```

```
    NOT NULL,
```

```
  Phone
```

```
    CHAR (10)
```

```
    NULL,
```

```
  Email
```

```
    NVARCHAR(100)
```

```
    UNIQUE,
```

```
  Salary
```

```
    NUMERIC(7,2),
```

```
  BirthDate
```

```
    DATE,
```

```
);
```

Three part column definition, separated by ,

Primary key constraint

NOT NULL means required.

Unique constraint

NULL is the default value if not specified.

Use semicolon to end a statement

```
CREATE TABLE Project
```

```
(
```

```
  ProjectId
```

```
    INT
```

```
    PRIMARY KEY IDENTITY (1,1),
```

```
  Name
```

```
    NVARCHAR(100)
```

```
    NOT NULL,
```

```
  Description
```

```
    NVARCHAR(4000),
```

```
  StartDate
```

```
    SMALLDATETIME
```

```
    DEFAULT GETDATE(),
```

```
  Budget
```

```
    DECIMAL(8,2)
```

```
    NULL CHECK (Budget < 500000)
```

```
);
```

Defining a surrogate key: starting from 1 with an increment of 1.

Default value

Check constraint

# CREATE TABLE SQL (2)

```
CREATE TABLE Assignment
```

```
(
```

```
  EmployeeNumber
```

```
    SMALLINT
```

```
    NOT NULL,
```

```
  ProjectId
```

```
    INT
```

```
    NOT NULL,
```

```
  HoursWorked
```

```
    NUMERIC(6,2)
```

```
    NULL,
```

```
  CONSTRAINT Assignment_PK PRIMARY KEY(EmployeeNumber, ProjectId),
```

```
  CONSTRAINT Assignment_FK1 FOREIGN KEY(EmployeeNumber)
```

```
    REFERENCES Employee(EmpNumber) ON UPDATE CASCADE
```

```
);
```

```
ALTER TABLE Assignment
```

```
  ADD CONSTRAINT Assignment_FK2 FOREIGN KEY(ProjectId)
```

```
    REFERENCES Project(ProjectId) ON UPDATE CASCADE
```

```
    ON DELETE CASCADE;
```

Constraint name

Defining primary key constraint at the table level. A composite PK has to be defined at the table level.

Defining foreign key constraint at the table level, with table definitions.

We can also define/add a foreign key constraint after defining tables.

Referential actions

# Column Definition

- ◆ 3 part column definition
  - Column name (required)
  - Data type and length (required)
  - Column constraints (optional)
    - ◆ primary key (only for single column PK)
    - ◆ null/not null (if not specified, default is null)
    - ◆ default
    - ◆ unique
    - ◆ identity

# Data Types

## ◆ SQL Server 2008 data types reference

- <http://msdn.microsoft.com/en-us/library/ms187594.aspx>

nchar(x) char(x)	Fixed-length character data of $x$ characters. $x \leq 4000$ (8000 for char). "n" for national (Unicode).
nvarchar(x) varchar(x)	Variable-length character data. $x \leq 4000$ (nvarchar) or 8000 (char). If more than 4000 (or 8000 for char), use " <b>max</b> " to indicate that the maximum storage size is $2^{31}-1$ bytes.
datetime datetime2 date, time	<b>datetime2</b> can be considered as an extension of the existing <b>datetime</b> type that has a larger date range, a larger default fractional precision, and optional user-specified precision.
int	Range: $-2^{31}$ (-2,147,483,648) to $2^{31}-1$ (2,147,483,647), 4 Bytes
smallint	Range: $-2^{15}$ (-32,768) to $2^{15}-1$ (32,767), 2 Bytes
tinyint	Range: 0 to 255, 1 Byte
bit	0, 1 or Null
decimal(p,s) numeric(p,s) $0 \leq s \leq p$	<b>numeric</b> is functionally equivalent to <b>decimal</b> . p (precision): the maximum number of digits that can be stored s (scale): the maximum number of digits that can be stored to the right of the decimal point.

# Constraints

## ◆ Five types of constraints:

- PRIMARY KEY
- UNIQUE
- NULL/NOT NULL
- FOREIGN KEY
- CHECK

## ◆ Defined at table level or column level

# Key Constraint and NOT NULL

- ◆ NULL and NOT NULL defines whether a value is required for the column
  - If not specified, the default is null (other DBMS may be different)
- ◆ PK cannot be NULL
- ◆ UNIQUE constraint defines a candidate key
  - Can be NULL, but there can't be only one null value for all the records
- ◆ Surrogate Key – automatically generated numbers for primary key
  - Use integer data types and set the “identity” property

# DEFAULT and CHECK Constraint

## ◆ Default

- Give the column a default value when a row is created.
- The default value is generally either a literal value, such as a specific number or string, or a reference to a function (popular for inserting the current date and time).
- The default value will be used when an insert occurs and the column name is excluded from the statement.

## ◆ CHECK

- Enforcing domain integrity, or setting valid value range for a column (or columns)

# ALTER Table

## ◆ Syntax

```
ALTER TABLE TableName  
    [Modification]
```

## ◆ Modification include

- Add, modify, and drop columns
- Add, modify, and drop table constraints

# Altering Columns

## ◆ Adding a new column

```
ALTER TABLE Employee  
  ADD (MidNAME NVARCHAR(20) NULL);
```

## ◆ Modifying a column

```
ALTER TABLE Employee  
  MODIFY (FirstName NVARCHAR(40));
```

The same style 3-part column definition



## ◆ Dropping a column

```
ALTER TABLE Employee  
  DROP COLUMN MidName;
```

# Altering Table Constraints

## ◆ Adding a foreign key

```
ALTER TABLE GroupAssignment
  ADD CONSTRAINT GroupAssignment_FK2
    FOREIGN KEY (GroupNumber)
      REFERENCES GROUPS (GroupId)
        ON UPDATE CASCADE;
```

## ◆ Dropping a foreign key

```
ALTER TABLE GroupAssignment
  DROP CONSTRAINT GroupAssignment_FK2
```

# DROP Table

- ◆ Warning! The DROP statement will permanently remove table structure and all data

```
DROP TABLE Project;
```

- ◆ Note!
  - The table cannot be dropped if another table is referencing the current table (a foreign key constraint) and referential integrity is enforced.

# Table Definition Sequence

- ◆ Be careful of the sequence of table creation, alteration, and drop
- ◆ Create tables in a reverse reference order.
  - Example: if table A has a foreign key referencing table B, then create B first and then A.

# Objects Naming Convention

- ◆ Avoid using blank space between words
  - Capitalize the first letter of each word and leave no space between words, or
  - Use “\_” to connect words
- ◆ Avoid using DBMS reserved words for names of tables, columns, constraints, etc.
  - SQL Server Reserved Keywords:  
<http://msdn.microsoft.com/en-us/library/ms190011.aspx>
- ◆ Or use delimited identifiers for object names
  - <http://msdn.microsoft.com/en-us/library/ms176027.aspx>

# SQL Views

- ◆ SQL view is a virtual table that is constructed from other tables or views
  - It has no data of its own, but obtains data from tables or other views

- Hide columns or rows
- Display results of computations
- Hide complicated SQL syntax
- Layer built-in functions
- Provide level of isolation between table data and users' view of data
- Assign different processing permissions to different views of the same table
- Assign different triggers to different views of the same table

# Create View

- ◆ SELECT statements are used to define views

- ◆ Example

```
CREATE VIEW CustomerNameView AS
  SELECT  LastName AS CustomerLastName,
          FirstName AS CustomerFirstName,
FROM    CUSTOMER;
```

- ◆ Query the view

```
SELECT * FROM CustomerNameView
ORDER BY CustomerLastName, CustomerFirstName;
```

# Summary

## ◆ Key concepts

- Physical design, physical data model
- SQL DDL
- Table constraints and column constraints
- Understand major data types in SQL Server 2008 database

## ◆ Key skills

- Use SQL DDL statements to define table structures and views
  - ◆ CREATE TABLE, ALTER TABLE, DROP TABLE
  - ◆ CREATE VIEW
- Use SQL Server Management Studio to
  - ◆ Create/modify/delete databases, tables, columns, constraints, and views
  - ◆ Insert, update, and delete records

# Summary: 3 Models/Schemas

	<i>Modeling Tool</i>	<i>Model Elements</i>
<b>Conceptual</b>	ERD	Entity, Attribute, Identifier, Relationship
<b>Logical</b>	Relational Model	Relation (table), Attribute (generic data type, size, uniqueness, required/not, domain, default), Primary Key, Foreign Key (uniqueness, required/not, referential integrity)
<b>Physical</b>	Specific DBMS Model	<ul style="list-style-type: none"><li>•Database files, storage;</li><li>•Table, column, data type (DBMS specific), constraint (PK, FK, unique, null, check, default) (DBMS specific);</li><li>•View, index, trigger, security, transaction, etc.</li></ul>