Database Implementation with SQL Server 2008

IT 4153 Advanced Database

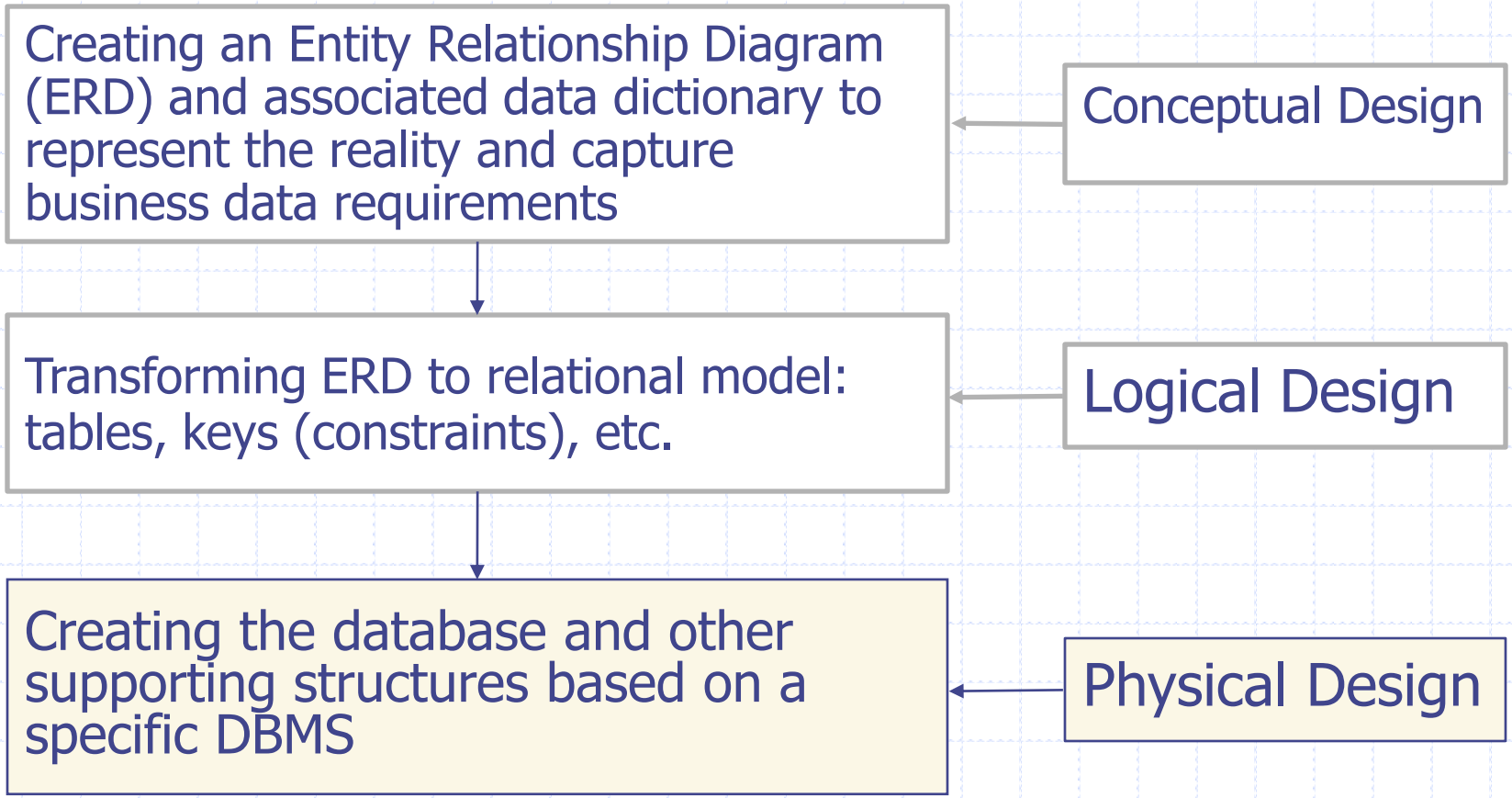
J.G. Zheng

Spring 2012

Overview

- Database physical design overview
- SQL Server 2008 physical level implementations and elements
- SQL DDL to create database objects
  - Table, constraints, view
  - ALTER, DROP
- Schema query
- SQL 2008 data modification
  - INSERT, UPDATE, DELETE, TRUNCATE

3 Level Database Design



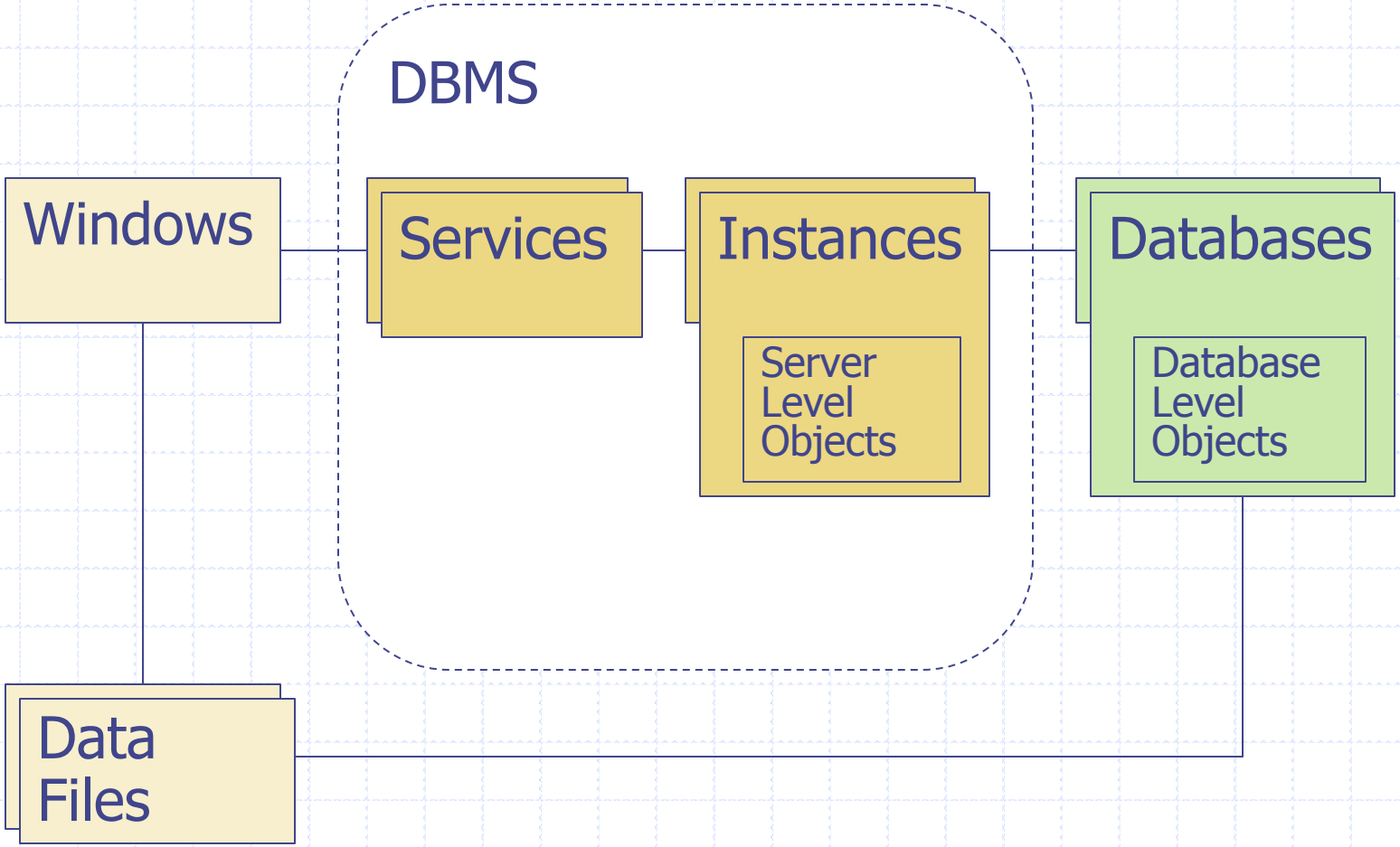
3 Models/Schemas

	Modeling Tool	*Model Elements*
Conceptual	ERD	Entity, Attribute, Identifier, Relationship
Logical	Relational Model	Relation (table), Attribute (generic data type, size, uniqueness, required/not, domain, default), Primary Key, Foreign Key (uniqueness, required/not, referential integrity)
Physical	Specific DBMS	Database files, storage; Table, column, data type (DBMS specific), constraint (PK, FK, unique, null, check, default) (DBMS specific); View, index, trigger, security, transaction, etc.

Physical Design

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

SQL Server Implementation



SQL Server Database Storage

- SQL Server maps a database over a set of operating-system files
- Database files

File	Description
Primary	The primary data file contains the startup information for the database and points to the other files in the database. User data and objects can be stored in this file or in secondary data files. Every database has one primary data file. The recommended file name extension for primary data files is .mdf.
Secondary	Secondary data files are optional, are user-defined, and store user data. Secondary files can be used to spread data across multiple disks by putting each file on a different disk drive. Additionally, if a database exceeds the maximum size for a single Windows file, you can use secondary data files so the database can continue to grow. The recommended file name extension for secondary data files is .ndf.
Transaction Log	The transaction log files hold the log information that is used to recover the database. There must be at least one log file for each database. The recommended file name extension for transaction logs is .ldf.

SQL Server Data Types

nchar(x) char(x)	Fixed-length character data of *x* characters. x<=4000 (8000 for char). "n" for national (Unicode).
nvarchar(x) varchar(x)	Variable-length character data. x<=4000 (nchar) or 8000 (char). If more than 4000 (or 8000 for char), use "**max"** to indicate that the maximum storage size is 2^31-1 bytes.
Smaldatetime datetime datetime2 date, time	**datetime2** can be considered as an extension of the existing datetime 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 <= *s* <= *p*	**numeric** is functionally equivalent to **decimal**. 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.

SQL Server 2008 data types reference: http://msdn.microsoft.com/en-us/library/ms187594.aspx

Constraints

- Five types of constraints:
  - PRIMARY KEY
  - UNIQUE
  - NULL/NOT NULL
  - FOREIGN KEY
  - CHECK
- Constraints can be 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

Identity Property

- Identity property (of a table column) is the implementation of surrogate key
- Syntax (used in DDL)
  - IDENTITY [(seed , increment)]
  - Seed
    - Is the value that is used for the very first row loaded into the table.
  - Increment
    - Is the incremental value that is added to the identity value of the previous row that was loaded.
  - You must specify both the seed and increment or neither. If neither is specified, the default is (1,1).
- A table can have only one column defined with the IDENTITY property, and that column must be defined by using a decimal, int, numeric, smallint, bigint, ortinyint data type.

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)

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

Materialized View

- The query result is cached as a concrete table that may be updated from the original base tables from time to time.
  - Enables more efficient access, but at the cost of some data being potentially out-of-date.

- Implementation
  - Oracle: materialized view
  - DB2: materialized query table
  - SQL Server: indexed view
  - MySQL: not support natively

Index

- A database index is a data structure that improves the speed of data retrieval operations
  - at the cost of slower writes and increased storage space

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 (schema, or metadata): database, tables, views, etc.
  - DDL - data definition language

Defining Database

- SQL

```
CREATE DATABASE database_name
```

- database_name
  - Database names must be unique within an SQL Server instance.
  - A maximum of 128 characters.
  - If data file name is not specified, SQL Server uses database_name as both the logical_file_name and as the os_file_name. The default path is obtained from the registry.

- Reference
  - CREATE DATABASE
    - http://msdn.microsoft.com/en-us/library/ms176061.aspx
  - ALTER DATABASE
    - http://msdn.microsoft.com/en-us/library/ms174269.aspx
  - DROP DATABASE
    - http://msdn.microsoft.com/en-us/library/ms178613.aspx

Create Table

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

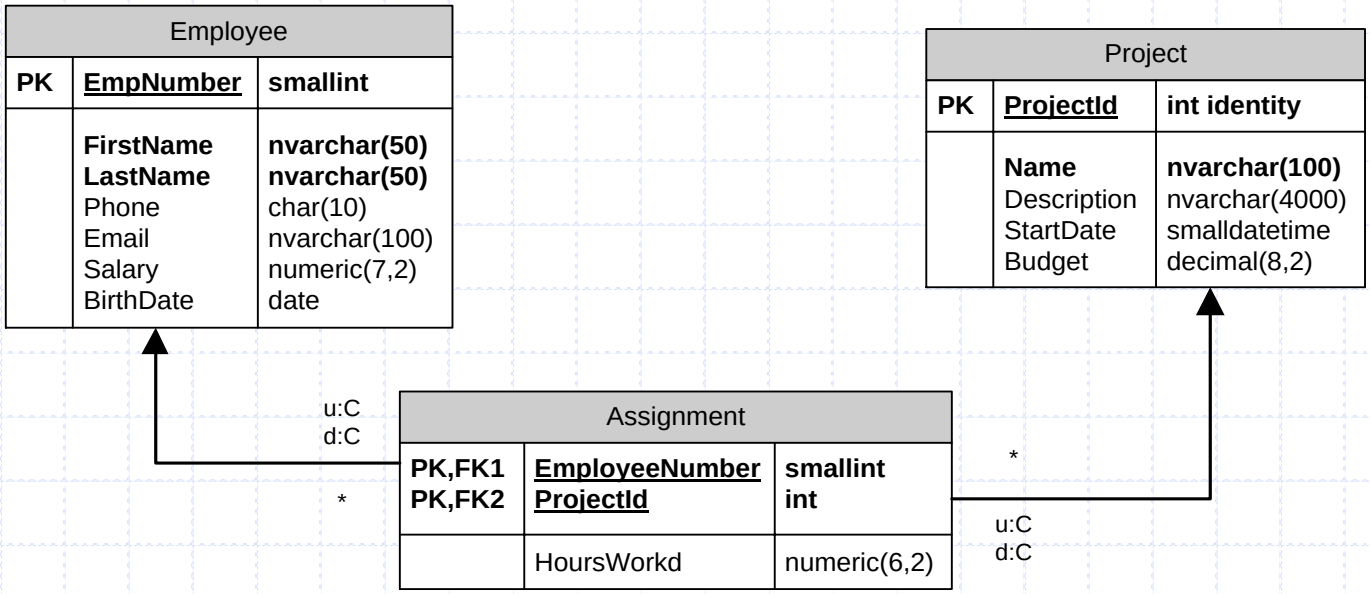
- Reference
  - http://msdn.microsoft.com/en-us/library/ms174979.aspx

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

CREATE TABLE Example

- A logical model in Visio



- Other requirements
  - Email is unique
  - Budget < 500000

Please get the sample SQL script file "DDL-Project.sql" from the course website.

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,
);

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)
);
```

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

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

Table Definition Sequence

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

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

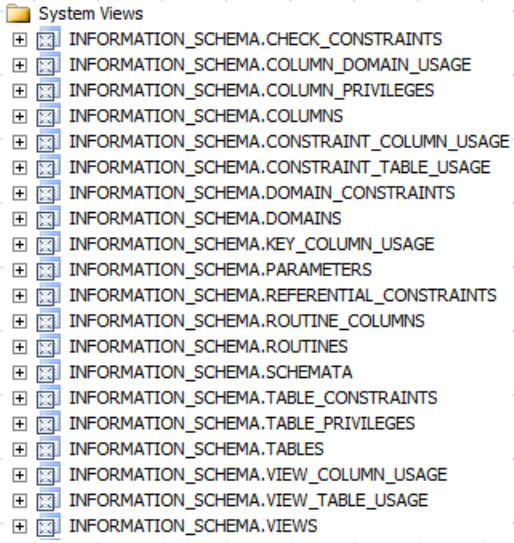
```
SELECT  * FROM CustomerNameView
ORDER BY CustomerLastName, CustomerFirstName;
```

Query Metadata

- Most modern relational databases store metadata in system tables
- In SQL Server 2008, we can query metadata through system views or stored procedures
- INFORMATION_SCHEMA
  - An information schema view is a common method for obtaining metadata.
  - This schema is contained in each database. Each information schema view contains metadata for all data objects stored in that particular database.

INFORMATION_SCHEMA

- Ways to obtain metadata
  - Standard SQL SELECT
  - System stored procedures
- Commonly used views
  - TABLES
  - COLUMNS
  - TABLE_CONSTRAINTS
  - VIEWS
  - KEY_COLUMN_USAGE
- Reference
  - http://msdn.microsoft.com/en-us/library/ms186778.aspx



Schema and Data Modification

- Schema
  - ALTER TABLE
  - DROP TABLE
- Data
  - INSERT
  - UPDATE
  - DELETE
  - TRUNCATE

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

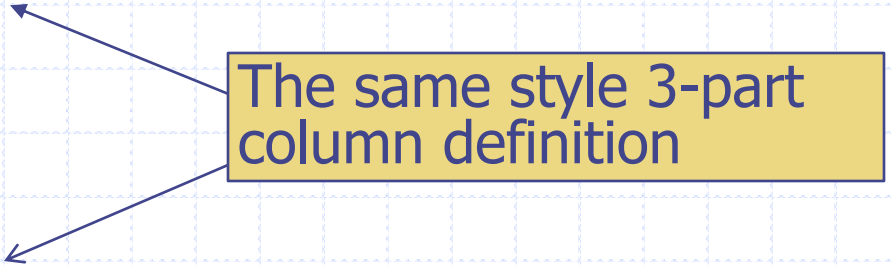
```
ALTER TABLE Employee
    ADD (MidNAME NVARCHAR(20) NULL);
```

- Modifying a column

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

- 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;
```

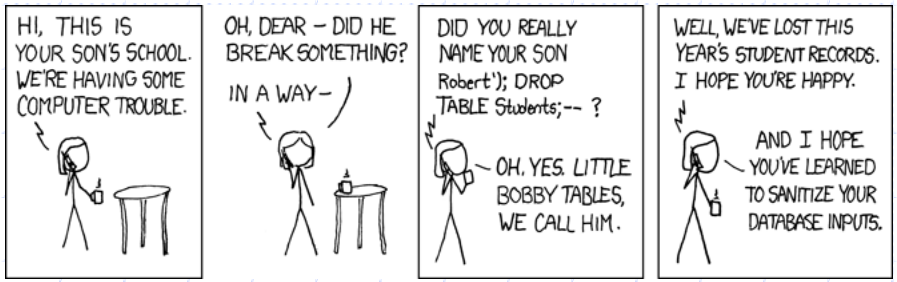
- Dropping a foreign key

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

DROP Table

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

DROP TABLE Students;



http://xkcd.com/327/

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

INSERT

- Basic syntax
  - INSERT INTO table_name [(column_list)] VALUES (value_list)
- If a column is not in *column_list*, the Database Engine must be able to provide a value based on the definition of the column; otherwise, the row cannot be loaded.
- The Database Engine automatically provides a value for the column if the column:
  - Has an IDENTITY property. The next incremental identity value is used.
  - Has a default. The default value for the column is used.
  - Has a timestamp data type. The current timestamp value is used.
  - Is nullable. A null value is used.
  - Is a computed column. The calculated value is used.
- Reference
  - http://msdn.microsoft.com/en-us/library/ms174335.aspx

Inserting Multiple Records

- Sub query

```
INSERT INTO Customers (CustID, CustName)
    SELECT CustID, CustName FROM LastYearSales
```

- 2008 new feature: table value constructor

```
INSERT INTO Sales.MySalesReason
VALUES ('Recommendation','Other'), ('Advertisement', DEFAULT),
(NULL, 'Promotion');
```

  - Mixed

```
INSERT INTO dbo.MyProducts (Name, ListPrice)
    VALUES ('Helmet', 25.50), ('Wheel', 30.00),
    (SELECT Name, ListPrice FROM Production.Product WHERE
    ProductID = 720);
```

UPDATE

- Examples
  - UPDATE Books SET ListPrice=29.99 WHERE ISBN='123456789';
    - Note: this is an update for a individual record
  - UPDATE Books SET ListPrice=29.99 WHERE ListPrice = 24.99
    - Note: batch update
- Note: without “WHERE” clause, update will modify all records in the table

DELETE

- Examples
  - DELETE FROM Books WHERE ISBN='123456789';
    - Note: this is a deletion of a single record
  - DELETE FROM Books WHERE YEAR < 2000
    - Note: batch deletion
- Note: without “WHERE” clause, delete will remove all records in the table

TRUNCATE

- TRUNCATE TABLE removes all rows from a table, but the table structure and its columns, constraints, indexes, and so on remain.
  - TRUNCATE TABLE table_name
- If the table contains an identity column, the counter for that column is reset to the seed value defined for the column.
- Compared to the DELETE statement, TRUNCATE TABLE is more efficient as less transaction log space is used.