Physical Design

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

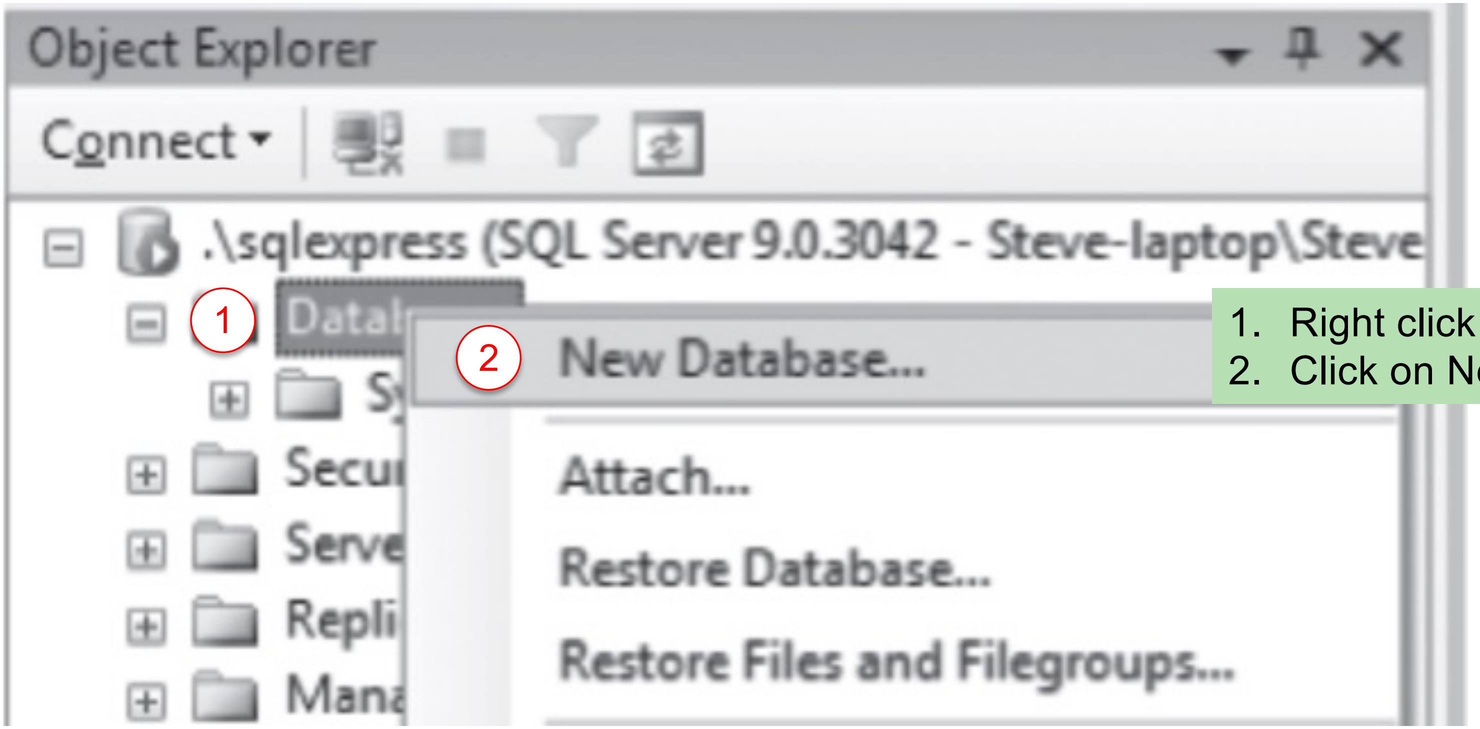
Chapter outcomes

- By the end of this chapter you will be able to:
  - Compare DBMSs and select one DBMS to use
  - Implement physical design based on logical ERD
  - Choose appropriate data types for columns
  - Enter sample data into tables

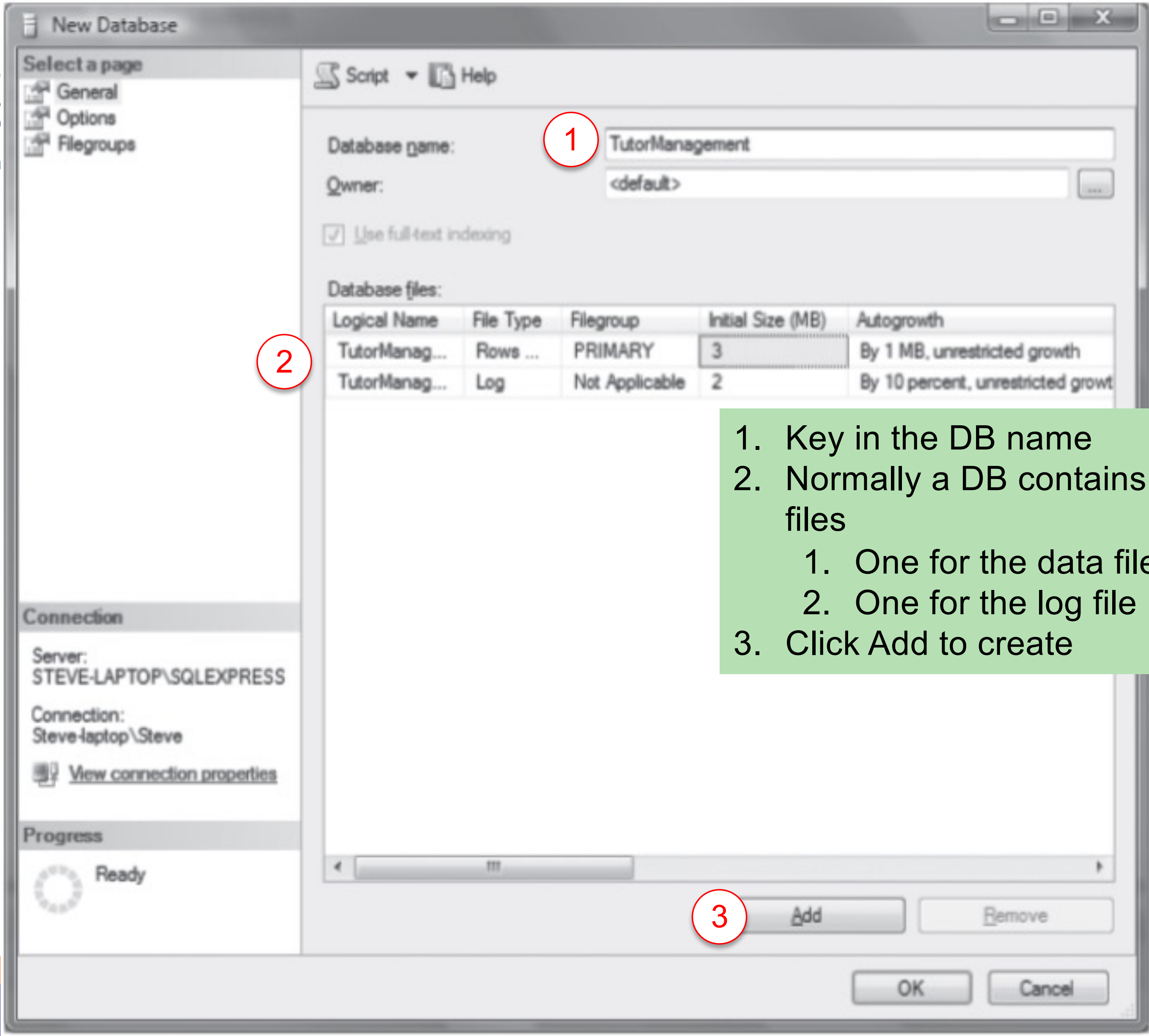
Choosing the DBMS

- There are several factors to consider
  - – Compatibility with your network and OS
  - – Hardware/software requirements for the DBMS
  - – Features of the DBMs in relation to your DB
  - – Familiarity and expertise in the DBMS for the DB developers and IT personnel
  - – Price and licensing requirements
  - – Product reliability and support

Creating The DB



1. Right click on Databases
2. Click on New Database…



1. Key in the DB name
2. Normally a DB contains two files
   1. One for the data file
   2. One for the log file
3. Click Add to create

Logical Design vs. Physical Design

- Logical design of a DB
  - It is the same no matter what DBMS you use
  - Entities, attributes, and relationships are purely in terms of logical structure of the data
- Physical design
  - Involves adapting logical design to the features and limitations of a particular DBMS product

Database Files

- Part of creating physical design is to understand how DBMS stores/manages files
  - Different DBMSs manages files in different ways.
- SQL Server databases have at least two files
  - A data file with the extension “.mdf” (can be more data files)
  - A log file with the extension “.ldf”

Data Types

- Second aspect of physical design involves data types
  - The column specification that determines what kind of data can be stored in that column
- ANSI provides set of basic data types,
  - Different DBMSs will make its own modification
  - We will look at typical data types in SQL Server

Table 6-1 Numeric Data Types

Data Type	Description	Range/Examples
Bigint	8 bytes integer	–2^63 (–9,223,372,036,854,775,808) to 2^63–1 (9,223,372,036,854,775,807)
Int	4 bytes	–2^31 (–2,147,483,648) to 2^31–1 (2,147,483,647)
Smallint	2 bytes	–2^15 (–32,768) to 2^15–1 (32,767)
Tinyint	1 byte	0 to 255
Bit	1 bit	0, 1, or null
Decimal	User can set precision up to 10^38	decimal(10,2)
Money	8 bytes	–922,337,203,685,477.5808 to 922,337,203,685,477.5807
Smallmoney	4 bytes	–214,748.3648 to 214,748.3647
Numeric	User can set precision up to 10^38	Same as decimal
Float	Approximate numeric type, the number of bytes depends on the number	–1.79E + 308 to –2.23E–308, 0 and 2.23E–308 to 1.79E + 308
Real	Also approximate, 4 bytes	–3.40E + 38 to –1.18E – 38, 0 and 1.18E – 38 to 3.40E + 38

These are frequently used

Table 6-3 String and Character Types

Data Type	Description	Examples
Char	Fixed-length ASCII text.	"Jefferson"—max 255 characters
Text	Text stores large blocks of text data. The text and *n*text data types are deprecated; use *varchar*(MAX) or *nvarchar*(MAX).	2,147,483,647 bytes
Varchar	Variable-length ASCII.	"Los Angeles," maximum 255 characters unless MAX (MAX allows 2^31 – 1 bytes)
Nchar	Unicode fixed length	Uses Unicode UCS_2 character set
Ntext	Unicode large block. Deprecated.	
nvarchar	Unicode variable-length text.	

These are
frequently used

Table 6-2 Date Time Types

These are frequently used

Data Type	Description	Examples/Range
Date	New in 2008, stores date values.	January 1, 1 A.D. through December 31, 9999 A.D
datetime2	New. Stores date and time and allows user to set precision in fractions of seconds.	Same date range as given earlier. Time range = 00:00:00 through 23:59:59.9999999
datetimeoffset	Date and time but with time-zone awareness.	Same
smalldatetime	Smaller date and time type.	January 1, 1753, through December 31, 9999 00:00:00 through 23:59:59.997
Time	New. You can set the precision in fractions of a second.	00:00:00.0000000 through 23:59:59.9999999

Some extra data types

Table 6-4 Some Data Types

Data type	Description	Examples
Image	Variable-length binary data. The image data type is deprecated and will go away.	2^31–1 bytes
Binary	Fixed-length binary.	1 to 8000 bytes
varbinary	Variable-length binary.	1 to 8000 bytes unless you specify MAX, 2 ^31–1 bytes
uniqueidentifer	Generates a unique identifier.	6F9619FF-8B86-D011-B42D-00C04FC964FF
XML	Stores XML data as XML, can be validated against schema collections, and queried with xquery.	<employee> <name>Sue Larson </name></employee>

Things to think about

- Fixed-Length Character Data Types
  - The char and nchar are fixed length
  - E.g., char(50) will always write 50 characters, even if you input only 20 (30 spaces are padded)
- Variable-Length Character Data Types
  - The varchar and nvarchar are variable length
  - E.g., varchar(50) will only use 20 characters if you input only 20 characters (no padded spaces)

Physical design of tutor table

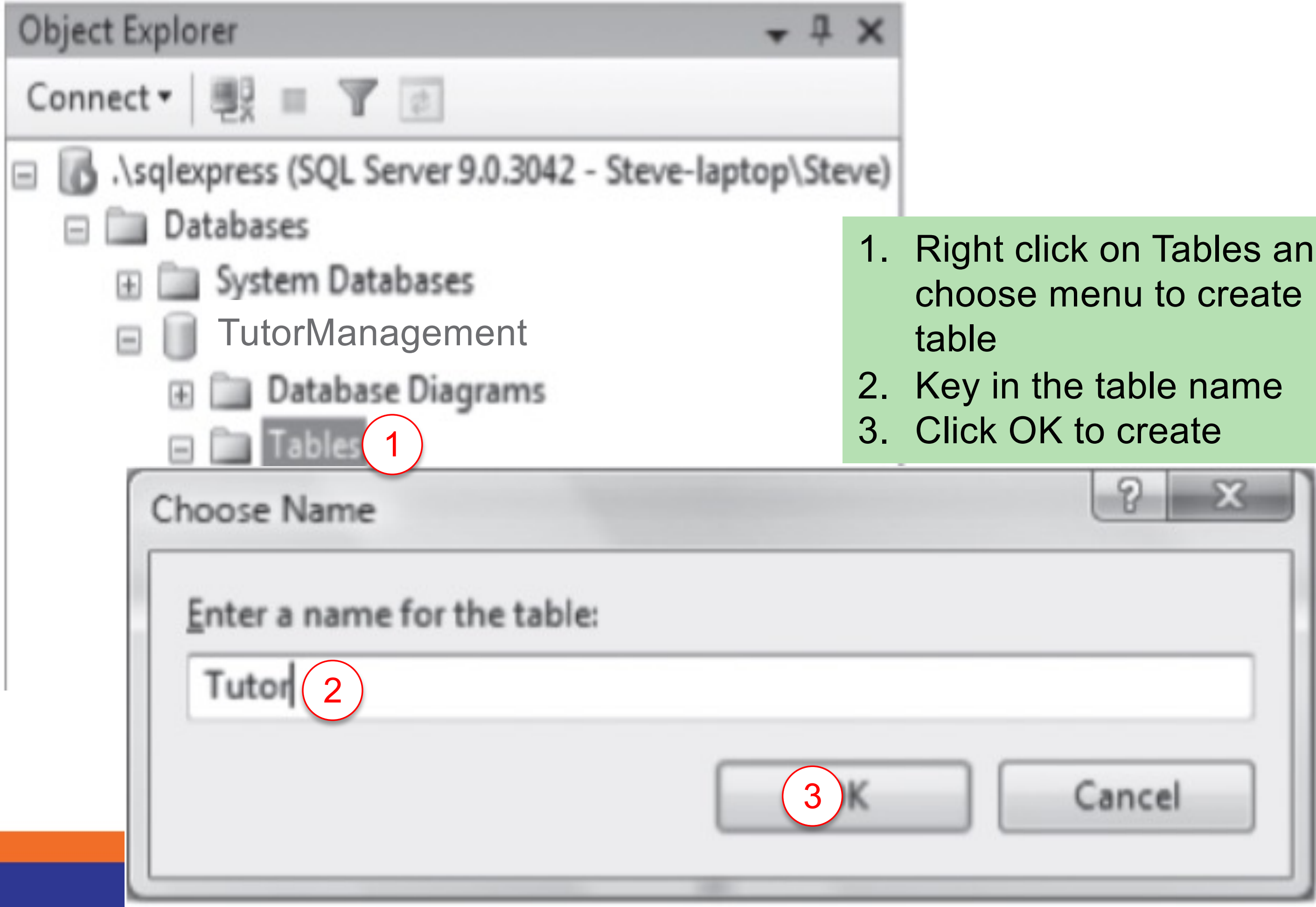
Logical design
(Entity)

Tutor	
PK	<u>TutorKey</u>
	TutorLastName TutorFirstName **TutorPhone** TutorEmail **TutorHireDate** **TutorStatus**

Physical design
(Table)

Column Name	Data Type	Allow Nulls
*TutorKey	nchar(10)	
TutorLastName	nvarchar(50)	
TutorFirstName	nvarchar(50)	X
TutorPhone	nchar(10)	
TutorEmail	nvarchar(50)	X
TutorHireDate	Date	
TutorStatus	nchar(10)	

Creating a table



1. Right click on Tables and choose menu to create new table
2. Key in the table name
3. Click OK to create

Nulls

- Nulls
  - A null represents the absence of a value.
  - A null value is unknown.
  - A null value could also be not applicable.
- Notice
  - Empty string is not null
  - Zero is not null

Activity: Create another table

Logical design (Entity)

Course	
PK	<u>CourseKey</u>
	CourseName CourseDescription

Physical design (Table)

Column Name	Data Type	Allow Nulls
*CourseKey	nchar(10)	
CourseName	nvarchar(50)	
CourseDescription	nvarchar(200)	X

Creating a Table in SQL

- You can use SQL (more in next lectures) to create tables
  - E.g., SQL code to create table Course

```
CREATE TABLE Course
      (
          CourseKey NCHAR(10) PRIMARY KEY,
          CourseName NVARCHAR(50) NOT NULL,
          CourseDescription NVARCHAR(200) NULL
      )
```

Activity: Create Table TutorCourse

Logical design
(Entity)

TutorCourse	
PK,FK1 PK,FK2	**CourseKey** **TutorKey**

Physical design
(Table)

Column Name	Data Type	Allow Nulls
*TutorKey	nchar(10)	
*StudentKey	nchar(50)	

To create composite PK, select both attributes and right click and set them as key

Error: Change this to:
*CourseKey nchar(10),

Activity: Create Ethnicity Table

Logical design
(Entity)

Ethnicity	
PK	<u>**EthnicityKey**</u>
	EthnicityDescription

Physical design
(Table)

Column Name	Data Type	Allow Nulls
*EthnicityKey	nchar(10)	
EthnicityDescription	nvarchar(50)	X

Activity: Create Table Student

Logical design
(Entity)

Student	
PK	**StudentKey**
	StudentlLastName StudentFirstName **StudentPhone** StudentEmail StudentGender StudentAge StudentCitizen StudentWorkForceRetraining
FK1	EthnicityKey

Physical design
(Table)

Column Name	Data Type	Allow Nulls
*StudentKey	nchar(10)	
StudentLastName	nvarchar(50)	
StudentFirstName	nvarchar(50)	X
StudentEmail	nvarchar(100)	X
StudentPhone	nvarchar(10)	X
StudentGender	nchar(1)	X
StudentAge	int	X
StudentCitizen	bit	X
StudentWorkerRetraining	bit	X
EthnicityKey	nchar(10)	x

Activity: Create StudentCourse Table

Logical design
(Entity)

StudentCourse	
PK,FK1	<u>StudentKey</u>
PK,FK2	<u>CourseKey</u>
PK	<u>StudentCourseQuarter</u>

Physical design
(Table)

Column Name	Data Type	Allow Nulls
*StudentKey	nchar(10)	
*CourseKey	nchar(10)	
*StudentCourseQuarter	nchar(10)	

To create composite PK, select all attributes and right click and set them as key

Activity: Create Session Table

Logical design
(Entity)

Session	
PK	**SessionDate**
PK	**SessionTime**
PK,FK1	**TutorKey**
PK,FK1	**CourseKey**
FK2	StudentKey
	SessionStatus
	SessionMaterialCovered

Physical design
(Table)

Column Name	Data Type	Allow Nulls
*SessionDateKey	Date	
*SessionTimeKey	Time	
*TutorKey	nchar(10)	
*CourseKey	nchar(10)	
StudentKey	nchar(10)	X
SessionStatus	nchar(10)	X
SessionMaterialCovered	nvarchar(255)	X

To create composite PK, select all attributes and right click and set them as key

Activity: Create Request Table

Logical design
(Entity)

Request	
PK	<u>**RequestKey**</u>
FK1	**CourseKey**
	RequestDate
	RequestStatus
FK2	StudentKey

Physical design
(Table)

Column Name	Data Type	Allow Nulls
*RequestKey	nchar(10)	
CourseKey	nchar(10)	
RequestDate	Date	
RequestStatus	nchar(10)	
StudentKey	nchar(10)	

Activity: Create RequestNote Table

Logical design
(Entity)

RequestNote	
PK	**RequestNoteKey**
 FK1	RequestNoteText RequestKey

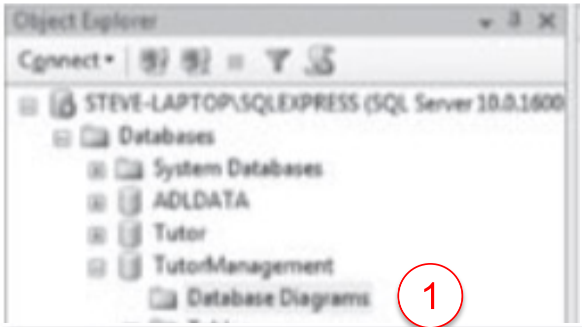
Physical design
(Table)

Column Name	Data Type	Allow Nulls
*RequestNoteKey	DateTime	
RequestNoteText	nvarchar(Max)	
RequestKey	nchar(10)	

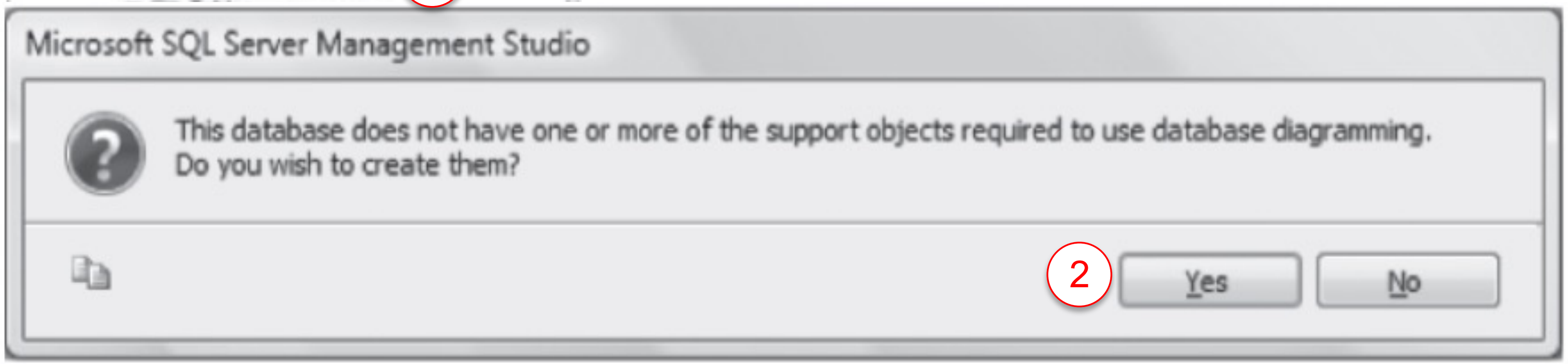
Coded Database Diagram

- After coding the database we can generate a Database Diagram for it
- A database Diagram will help us to visually see the tables (or views) that we have created

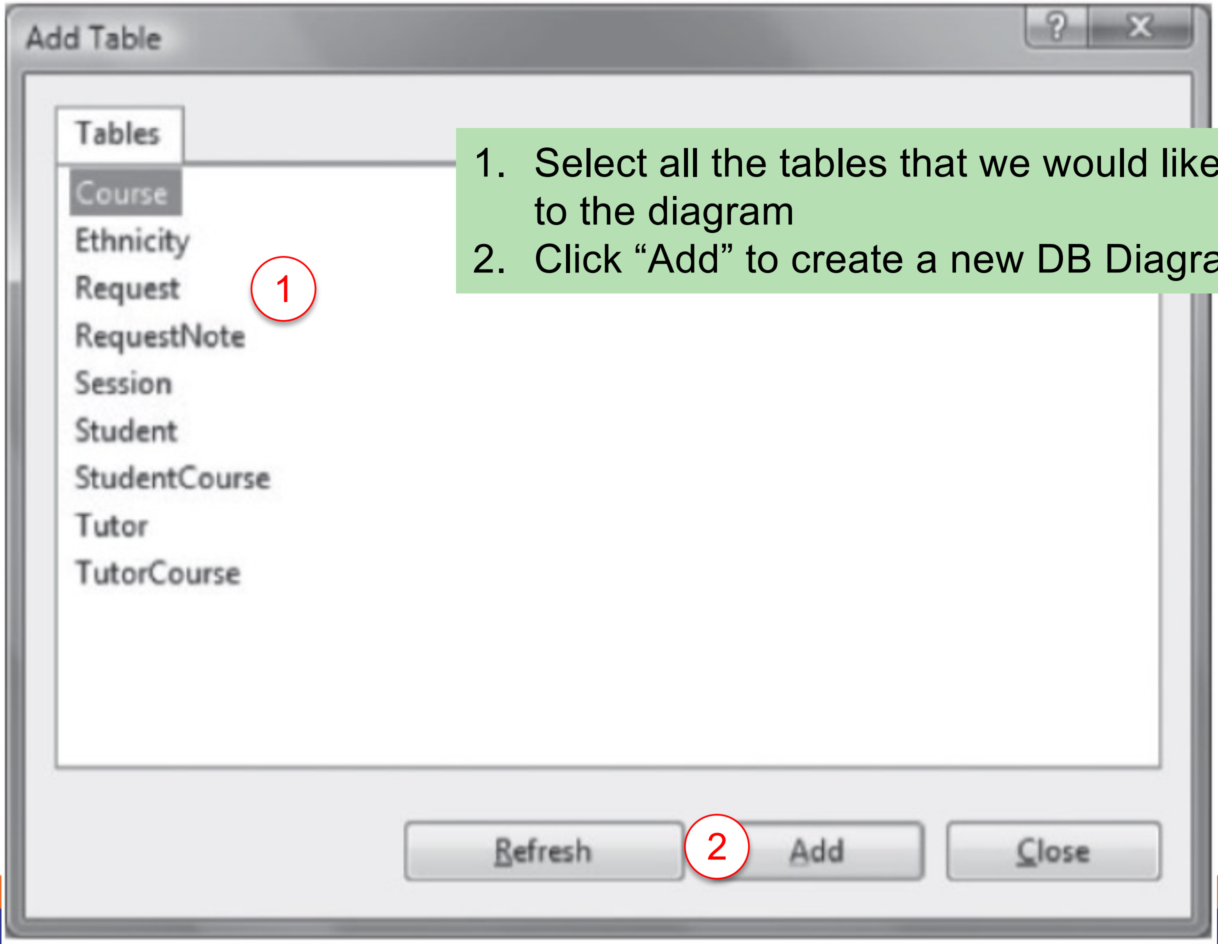
Create the Database Diagram



1. Right click on “Database Diagrams” and select the menu to create a new one
2. Click “Yes” to create a new DB Diagram

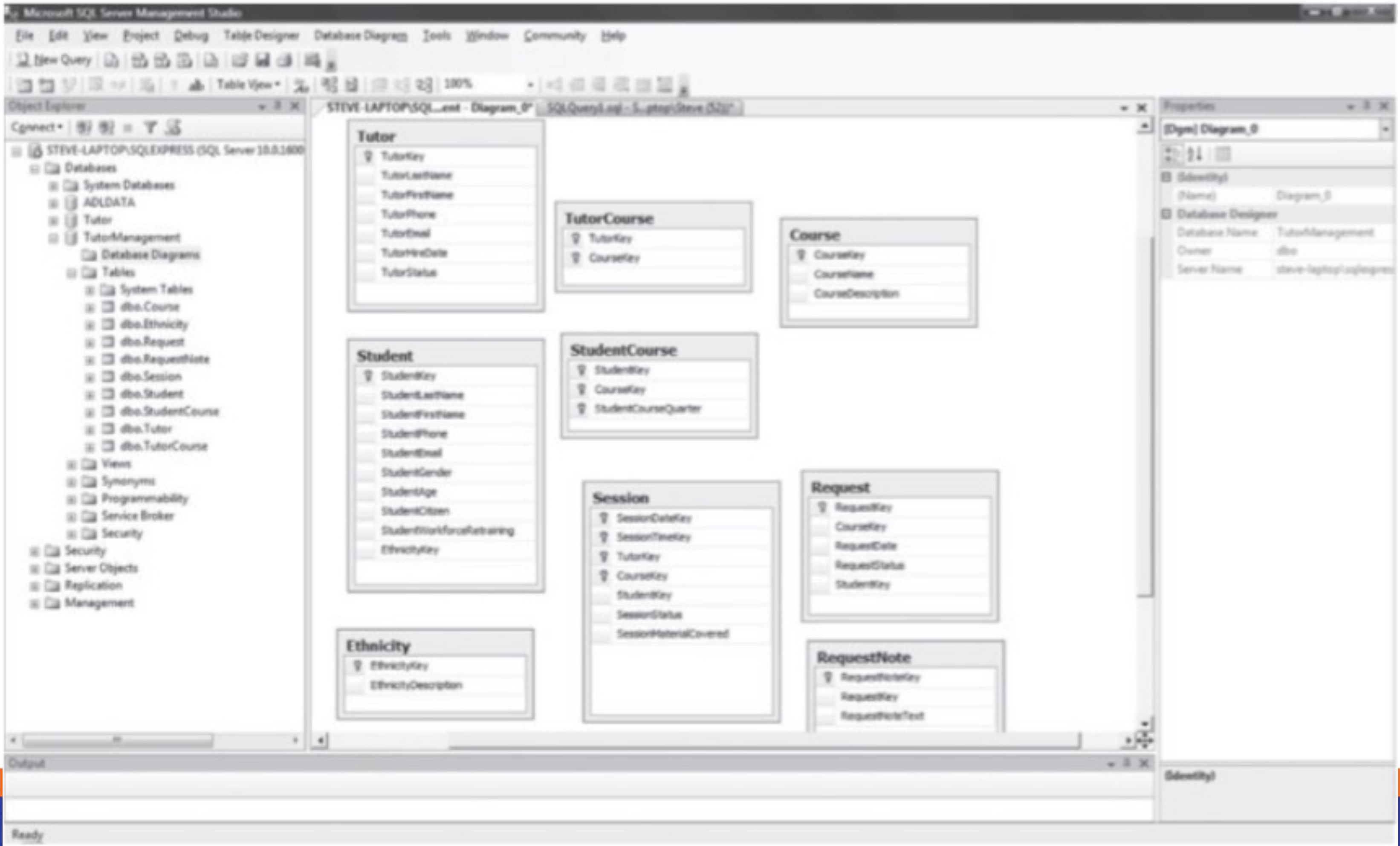


Create the Database Diagram

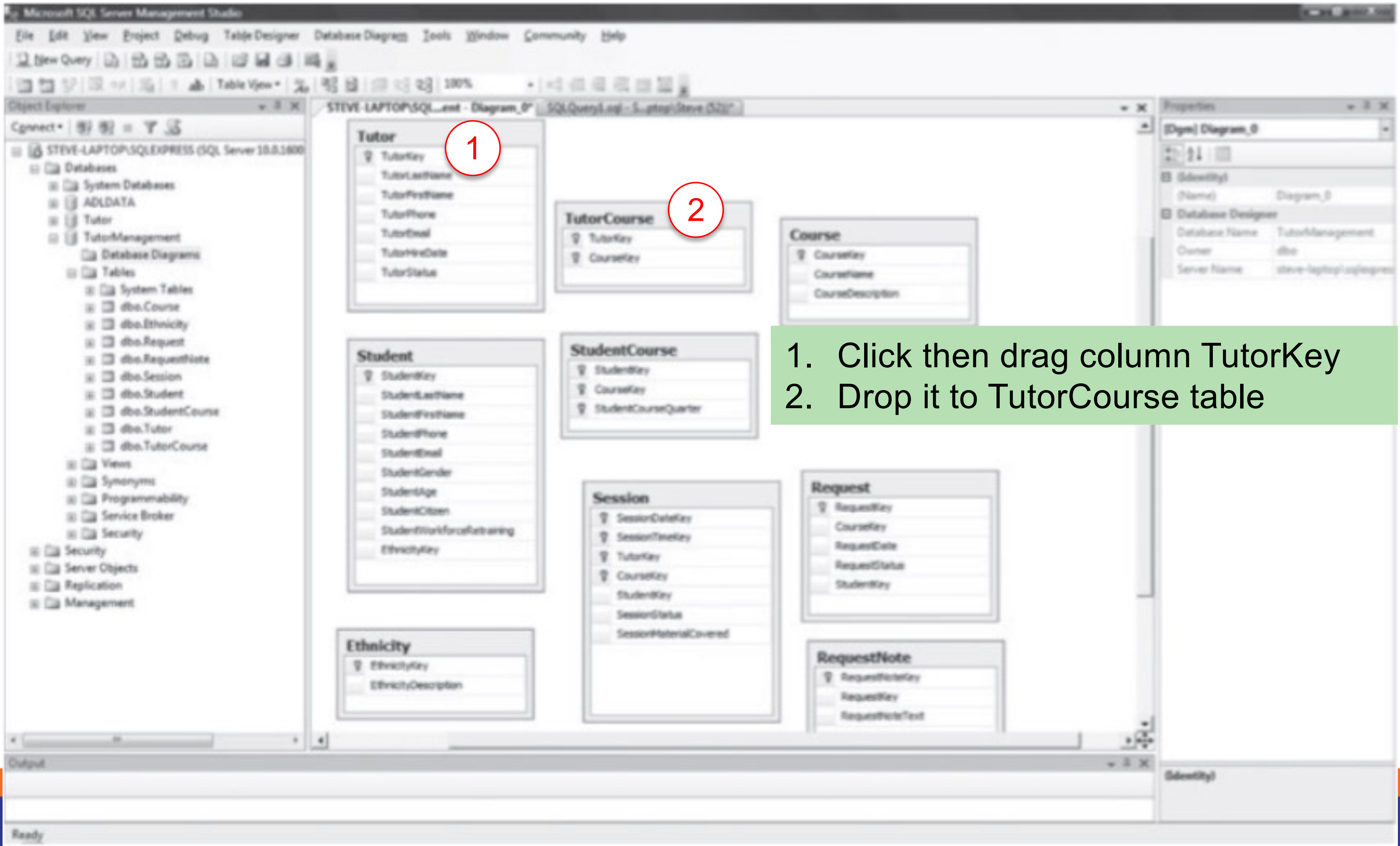


1. Select all the tables that we would like to add to the diagram
2. Click “Add” to create a new DB Diagram

The Created DB Diagram

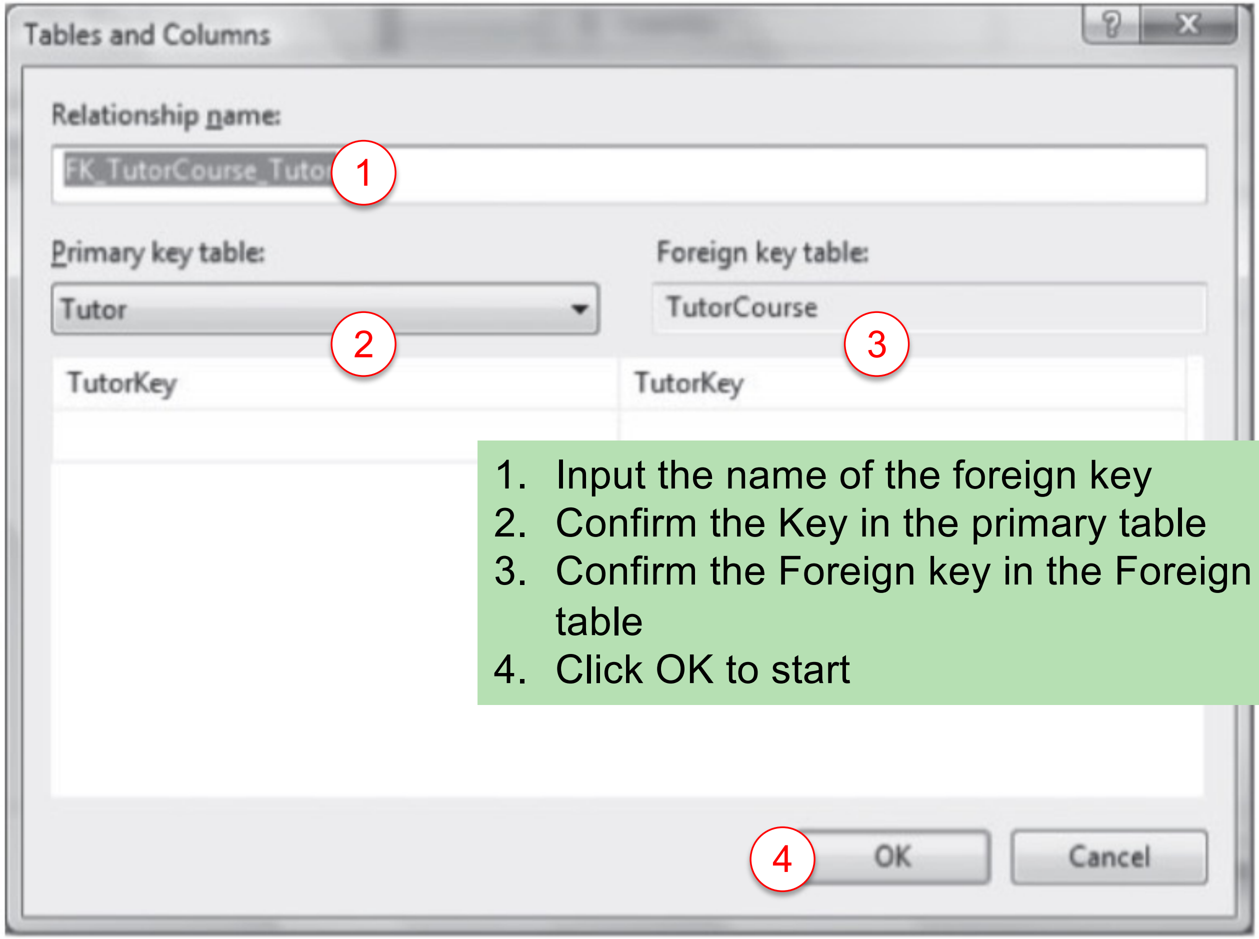


Creating Tutor – TutorCourse Rel.



1. Click then drag column TutorKey
2. Drop it to TutorCourse table

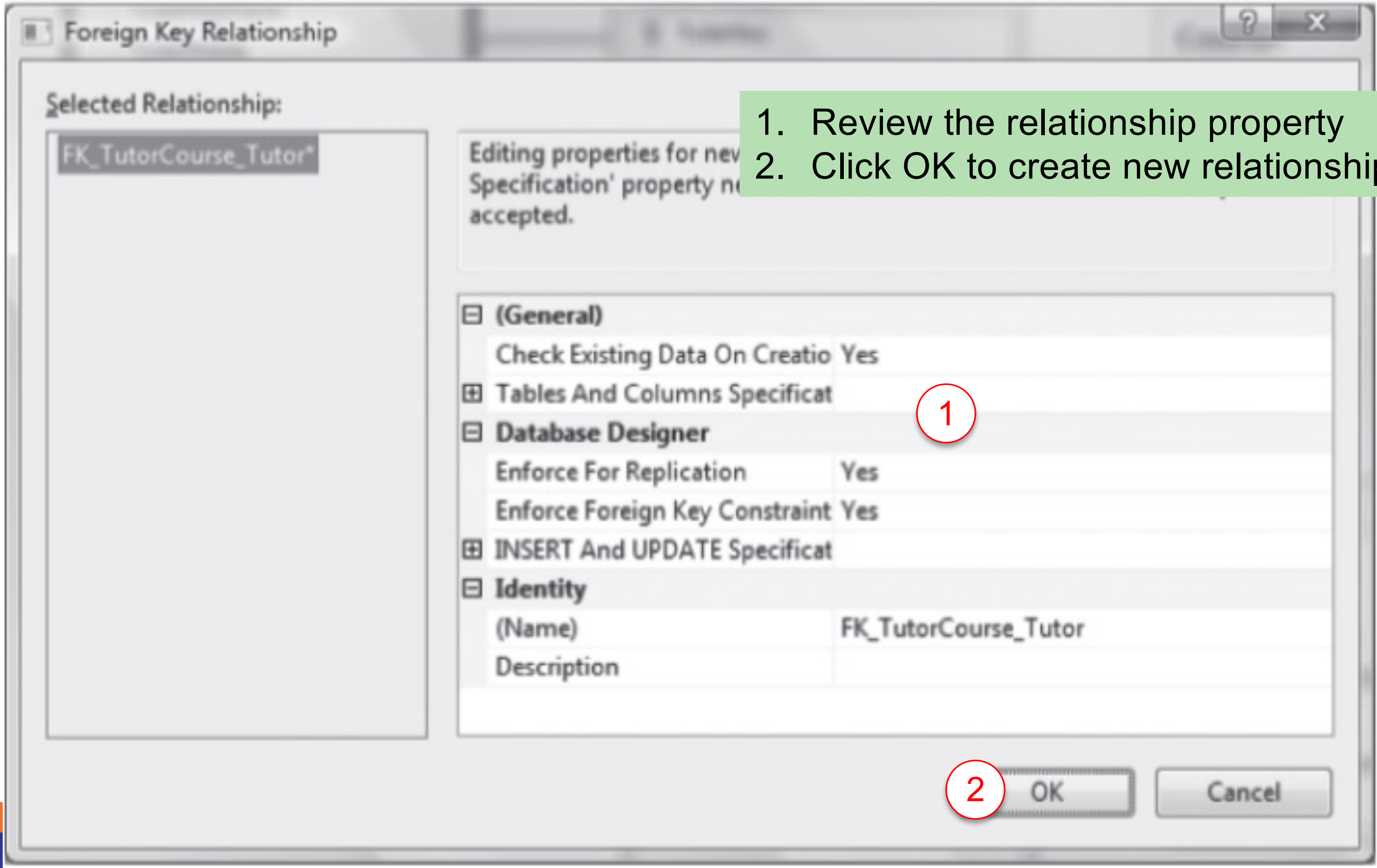
Creating Tutor – TutorCourse Rel.

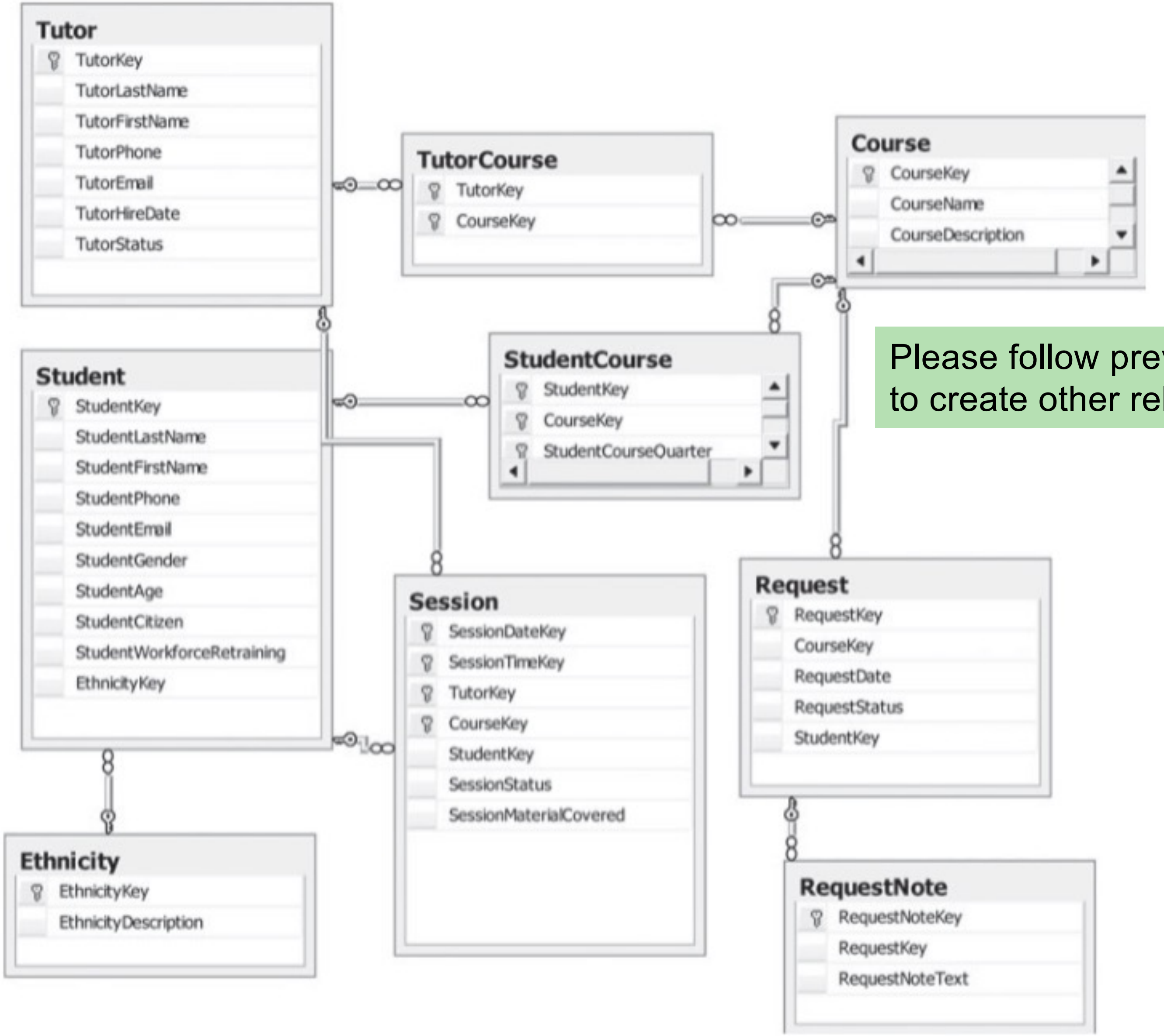


1. Input the name of the foreign key
2. Confirm the Key in the primary table
3. Confirm the Foreign key in the Foreign key table
4. Click OK to start

Creating Tutor – TutorCourse Rel.

1. Review the relationship property
2. Click OK to create new relationship





Please follow previous steps to create other relationships.

Referential Integrity

- This exists when every foreign key relates to an existing primary key
  - There are no orphan records in child tables that have no references in a parent table
- Enforcing referential integrity with Ms. SQL
  - In the properties for a relation, there is a property “Enforce Foreign Key Constraint”
  - The default value is “Yes”

Referential Integrity Example

```
create table category(
catId int primary key,
catName varchar(50));

create table product(
prodId int primary key,
prodName varchar(50),
prodPrice float,
catId int foreign key references category(catId));

insert into category values(1, 'TV');
insert into product values(1, 'LG TV', 100, 1);
insert into product values(2, 'Vaio', 200, 2);
delete from category where catId = 1;
```

Error: Because it is referencing to the value which doesn't exist in the PK of category

Error: Because category 1 is being used by product

Referential Integrity

Action	Effect of Enforcing Referential Integrity
INSERT	You must enter data into the parent (primary key) table before you can enter data into a child(foreign key) table. For example: You must enter the Customer information before entering the Sale information.
UPDATE	**1.** You cannot change the primary key value for any record in the parent table without also changing the related foreign key. This creates a dilemma because both must be changed simultaneously. You can either suspend referential integrity while making the update or use cascading updates (see the following point). **2.** You can only update or change a foreign key in a child table to one that has a matching value in a parent or primary key table.
DELETE	You cannot delete a row in a primary key table unless all related records are first deleted in the foreign key table. Example: You can't delete an order unless all the order details for that order are first deleted.

Referential Integrity

- In the properties of a relationship, we can set referential integrity actions as
  - Cascade update
  - Cascade delete
- Cascade update
  - A value in the PK of the primary table is updated
  - The corresponding value in the FK of the foreign key table is also updated automatically
- Cascade delete
  - A row in the PK table is deleted
  - All related rows in the child table will also be deleted

Cascading options

```
drop table product;
drop table category;

create table category(
catId int primary key,
catName varchar(50));

create table product(
prodId int primary key,
prodName varchar(50),
prodPrice float,
catId int foreign key references category(catId)
on delete cascade on update cascade,
);

insert into category values(1, 'TV');
insert into product values(1, 'LG TV', 100, 1);

delete from category where catId = 1;
```

Making use of cascading options

This will also delete the products belonging to this category 1

Inserting sample data into tables

- Sample data should be
  - Complete to test business rules
  - Varied enough to represent a variety of situations
  - Contain some exceptions and possibly even errors to test how your DB handles them

Input Course Data

CourseKey	CourseName	CourseDescription
ITC110	Beginning Programming	Programming using C#
ITC220	Introduction to Database	Overview of database design and topics
ITC255	Systems Analysis	Systems analysis and design
MAT107	Applied Math	Applied math for computers
ENG211	Technical Writing	Technical writing for information technology
WEB110	Beginning Web Page Design	Basic xhtml
ITC226	Database Administration	SQL Server administration

Right click on the Course table then select corresponding menu to open the table to input

Input Data Using SQL

- You can also input data using SQL Command

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);

INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Activity: Input Ethnicity Data

EthnicityKey	EthnicityDescription
Caucasian	White, European origin
Asian	Chinese, Japanese, Korean, Southeast Asian
AfrAmer	African American or of African origin
Hispanic	Mexican, Central or South American, Caribbean
Pacific	Pacific islander
Mideast	Arabic or Persian
Other	Other or not disclosed

Activity: Input Tutor

TutorKey	TutorLastName	TutorFirstName	TutorPhone	TutorEmail	TutorHireDate	TutorStatus
980010000	Roberts	Martha	2065551467	mroberts@yahoo.com	1/6/2010	Active
980010001	Brown	Susan	2065553528	Sb4@hotmail.com	2/1/2009	Active
980010002	Foster	Daniel	2065553490	Foster32@aol.com	2/12/2009	Active
980010003	Anderson	Nathan	3065556320	Null	3/2/2009	Inactive
980010004	Lewis	Ginger	2065552985	ginger@hotmail.com	3/15/2009	Active

Activity: Input TutorCourse Data

TutorKey	CourseKey
980010002	ITC255
980010002	ENG211
980010004	MAT107
980010000	WEB110
980010001	ITC220
980010001	WEB110
980010003	ITC110

Activity: Input Student Data

StudentKey	StudentLastName	StudentFirstName	StudentEmail	StudentPhone
990001000	Peterson	Laura	Null	2065559318
990001002	Carter	Shannon	Shannon@Carter.Org	2065554301
990001003	Martinez	Sandy	sandym@gmail.com	2065551158
990001004	Nguyen	Lu	lstar@yahoo.com	2065552938
990001005	Zukof	Mark	Null	Null
990001006	Taylor	Patty	P147@marketplace.com	2065552076
990001007	Thomas	Lawrence	Null	Null
980001008	Bradbury	Ron	rbradbury@mars.org	2065557296
980001009	Carlos	Juan	Carlos23@hotmail.com	2065559134
009001010	Min	Ly	lymin@hotmail.com	2065552789

The data is continued in the next slide (horizontally)

Activity: Input Student Data

StudentGender	StudentAge	StudentCitizen	StudentWorkerRetraining	EthnicityKey
F	23	True	False	Caucasian
F	32	True	True	AfrAmer
F	18	True	False	Hispanic
M	19	False	False	Asian
Null	Null	Null	Null	Null
F	42	True	True	Caucasian
M	24	True	False	Caucasian
M	53	True	True	Caucasian
M	25	False	False	Hispanic
F	20	False	False	Asian

Activity: Input StudentCourse Data

StudentKey	CourseKey	StudentCourseQuarter
990001000	ITC220	Fall09
990001000	ITC110	Fall09
990001000	WEB110	Fall09
990001002	ITC220	Fall09
990001002	ITC110	Fall09
990001004	MAT107	Fall09
990001004	WEB110	Fall09
990001007	ITC110	Fall09
980001009	ITC110	Fall09
980001009	ITC220	Fall09
980001009	MAT107	Fall09

The data is continued in the next slide (vertically)

Activity: Input StudentCourse Data

990001002	ENG211	Winter10
990001002	ITC255	Winter10
990001003	ENG211	Winter10
990001003	ITC255	Winter10
990001005	MAT107	Winter10
009001010	MAT107	Winter10
009001010	ITC255	Winter10
009001010	ENG211	Winter10
990001000	ITC255	Winter10
990001000	MAT107	Winter10

Activity: Input Session Data

SessionDateKey	SessionTimeKey	TutorKey	CourseKey	StudentKey	SessionStatus	SessionMaterialCovered
10/20/2009	14:00	980010001	WEB110	990001000	C	CSS
10/20/2009	13:00	980010003	ITC110	990001000	C	For next loop
11/20/2009	10:30	980010001	ITC220	990001002	C	Relations
11/5/2009	10:00	980010001	ITC220	Null	NS	Null
11/10/2009	13:00	980010004	MAT107	990001004	C	Binary Numbers
11/10/2009	14:00	980010001	WEB110	990001000	C	Web Forms
1/15/2010	9:30	980010002	ITC255	990001000	C	Use Cases
1/20/2010	11:00	980010002	ENG211	990001003	C	Document structure
1/22/20120	14:00	980010004	MAT107	990001005	NS	Null
2/5/2010	10:30	980010002	ITC255	990001000	C	Feasibility
2/10/2010	13:30	980010004	MAT107	Null	Null	Null
2/10/2010	14:00	980010004	MAT107	Null	Null	Null
2/13/2010	10:00	980010002	ITC255	Null	Null	Null
2/14/2010	11:00	980010002	ENG211	Null	Null	Null

Activity: Input Request Data

RequestKey	RequestDate	CourseKey	RequestStatus	StudentKey
1001	1/5/2010	ITC226	Active	009001010

Activity: Input RequestNote Data

RequestNoteKey	RequestID	RequestNoteText
1/6/2010 2:00 PM	1001	Only offered once a year and not a lot of requests for this class
1/10/2010 10:00 AM	1001	No students available, because a capstone class would have to get someone off campus

Documentation

- In many ways the database is self documenting
- System tables keep the “meta” information tables and other objects.
- But it is useful to keep a separate data dictionary
  - – Describes all tables, columns, data types, and constraints

Things We Have Done

- Translated our logical design into physical design
- Created a DB in SQL Server
- Created tables
- Assigned data types to columns
- Determined which columns should allow nulls and which should not
- Set primary keys
- Created a database diagram
- Created relationships among the tables
- Entered sample data into those tables

Vocabulary

Match the definitions to the vocabulary words:

1. Data types	—	a.	An extended language set that includes non-Latin characters
2. Database transactions	—	b.	A missing or unknown value for a column in a table
3. Null	—	c.	Every action in a database
4. Physical design	—	d.	Where every foreign key refers to an existing primary key in a related table
5. Referential integrity	—	e.	Database design adapted to the features and limits of a particular RDBMS
6. Unicode	—	f.	Column specifications that refer to what kind of data can be stored in a column

References

- Cogner, S., 2012. *Hands-on Database: An Introduction to Database Design and Development*. Prentice Hall.