

<https://www.halvorsen.blog>



# erwin Data Modeler

Academic Edition

Hans-Petter Halvorsen

# erwin Data Modeler

- In this Tutorial we will use the erwin Data Modeler software.
- erwin Data Modeler is used to modeling, design and create ER (Entity Relationship) diagram for your Database.
- erwin Data Modeler comes in different paid versions, but you can request a free trial.
- If you are a student, you can also apply for the "Academic Edition".
- <https://www.erwin.com>



# Database Modeling and Design

Hans-Petter Halvorsen

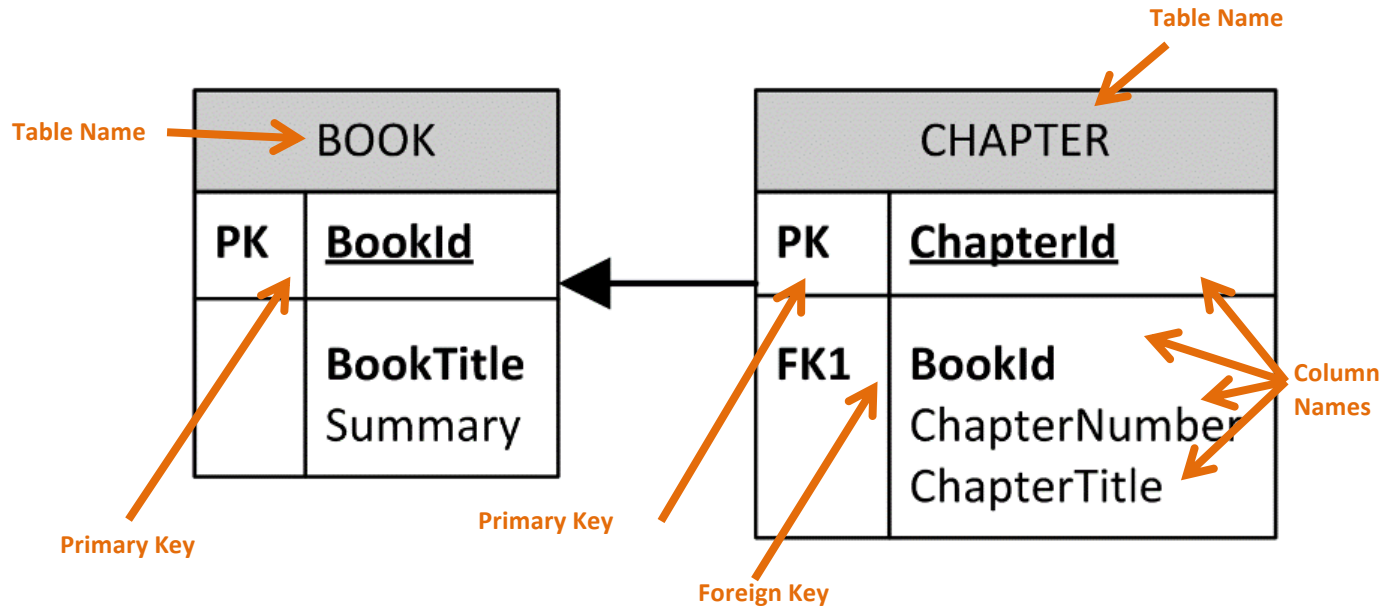
[Table of Contents](#)

# Entity Relationship (ER) Diagram

ER Diagram (Entity-Relationship Diagram)

- Used for Design and Modeling of Databases.
- Specify Tables and relationship between them (**Primary Keys** and **Foreign Keys**)

Example:



Relational Database. In a relational database all the tables have one or more relation with each other using Primary Keys (PK) and Foreign Keys (FK). Note! You can only have one PK in a table, but you may have several FK's.



# Start using erwin Data Modeler

# New Model

The screenshot displays the erwin DM software interface. The main window is titled "erwin DM" and features a menu bar with "File", "Home", "View", "Diagram", "Model", "Mart", "Actions", "Tools", and "Help". A toolbar below the menu contains various icons for file operations and navigation. On the left side, there are panels for "Model Templates", "Entity Sub-Categories", and "Model Explorer". The central area is divided into three sections: "FILE ACTIONS", "MODEL ACTIONS", and "RECENT FILES".

The "FILE ACTIONS" section includes:

- Create New Model**: Create a new document
- Open Existing Model**: Open an existing document

The "MODEL ACTIONS" section includes:

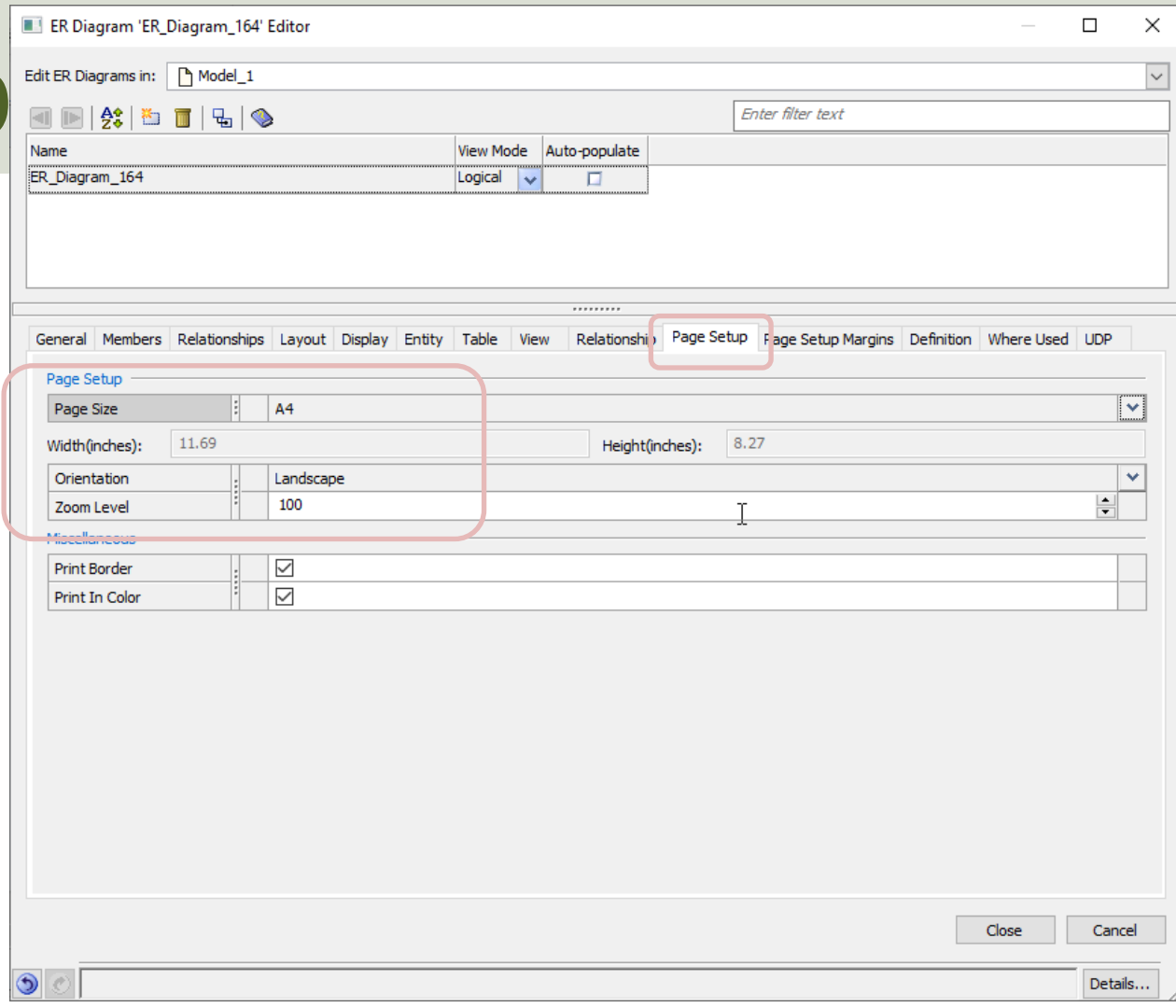
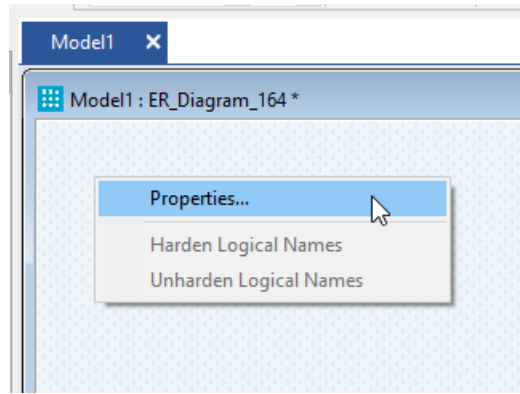
- Complete Compare**: Invoke Complete Compare
- Reverse Engineer**: Reverse Engineer from a SQL Script or Database
- erwin DM Scheduler**: Invoke erwin DM Scheduler
- Options**: Launch the Options Dialog

A "New Model" dialog box is open in the center, highlighted with a red border. It contains the following fields and options:

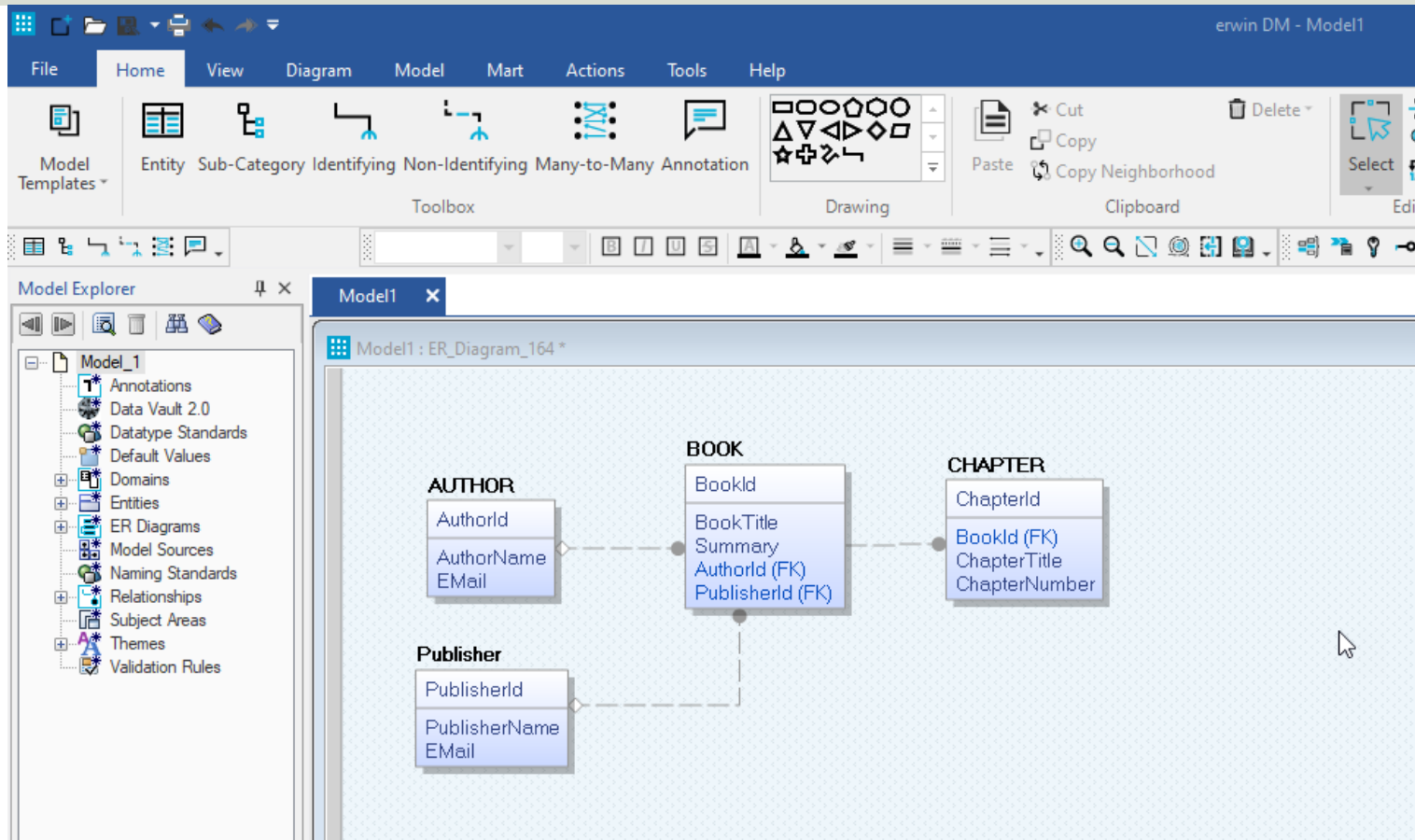
- Type**: Radio buttons for Logical, Physical, Logical/Physical (selected), and Match template.
- Target Server**:
  - Match template target server
  - Database: SQL Server (dropdown)
  - Version: 2019 (dropdown)
- Template**:<br><Default> (dropdown)
- Preserve the template binding
- Buttons: OK, Cancel

At the bottom of the interface, there is an "Action Log" panel and a taskbar with icons for "Advisories" and "Overview".

# Page Setup

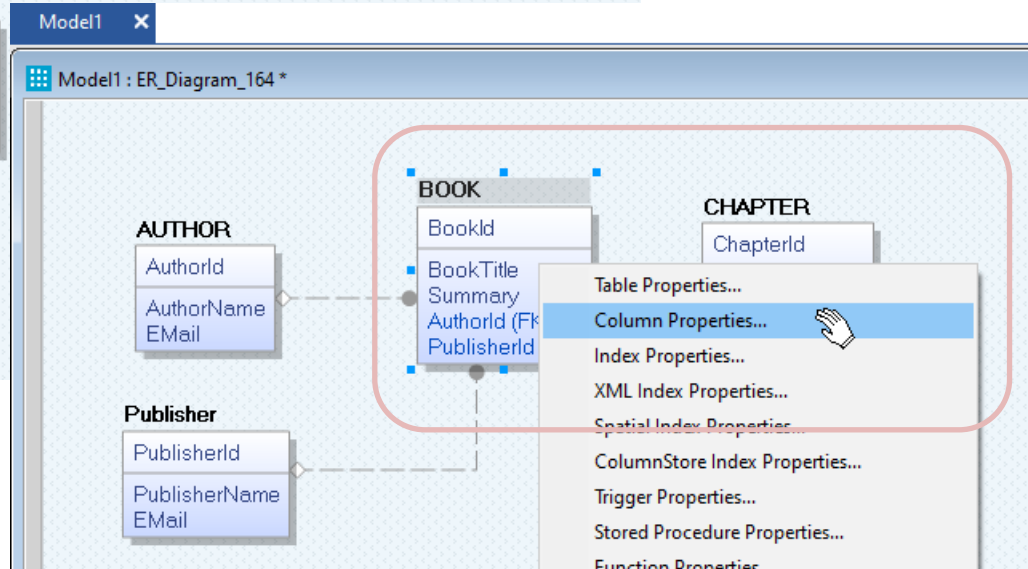
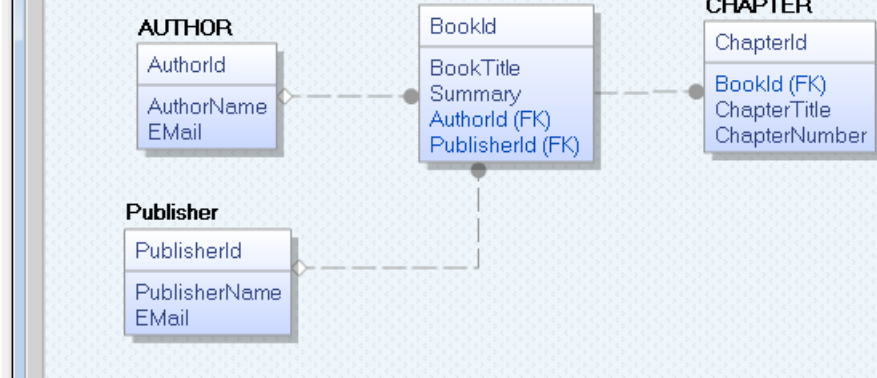
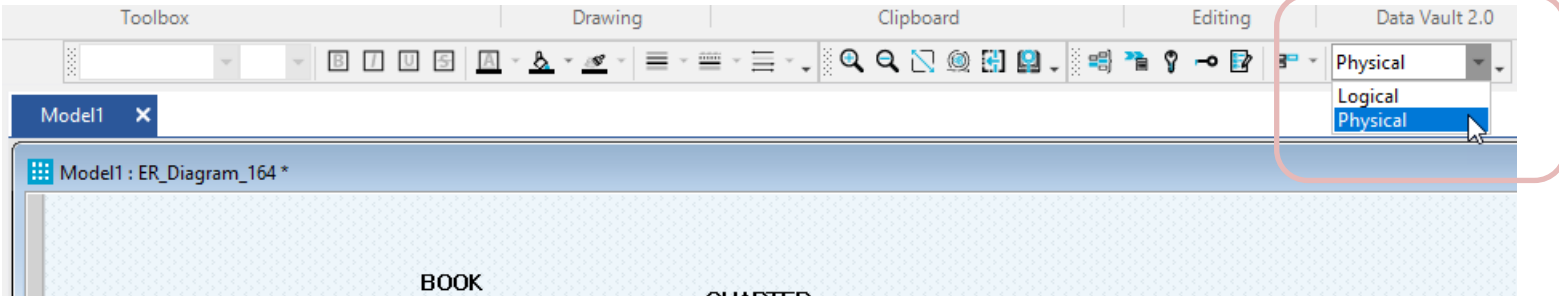


# Example





# Set Data Types



# Set Data Types

The screenshot shows the 'SQL Server Table 'AUTHOR' Column 'AuthorId' Editor' dialog box. The table being edited is 'AUTHOR'. The columns and their properties are as follows:

Physical Name	Domain Parent	Physical Data Type	Primary Key	Foreign Key	Physical Only	PII	Comments
AuthorId	?_default_	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Author...	?_default_	varchar(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Email	?_default_	varchar(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

The 'Physical Data Type' column is highlighted with a red box. The dialog also shows tabs for 'General', 'SQL Server', 'Constraint', 'Link', 'Indexes', 'Style', 'Comment', 'Where Used', 'UDP', 'History', 'Notes', and 'Extended'. The 'General' tab is currently selected, showing 'Domain Parent' and 'Physical Name' fields.

# Show Data Types in Diagram

The screenshot shows the ER Diagram Editor interface. On the left, an ER diagram is displayed with three entities: AUTHOR, PUBLISHER, and BOOK. The AUTHOR entity has attributes: AuthorId: int, AuthorName: varchar(100), and Email: varchar(100). The PUBLISHER entity has attributes: PublisherId: int, PublisherName: varchar(100), and Email: varchar(100). The BOOK entity has attributes: BookId: int, BookTitle: varchar, Summary: varchar, AuthorId: int (FK), and PublisherId: int (FK). A red box highlights the AUTHOR entity in the diagram.

On the right, the 'Properties' tab is active, showing the 'Table Display Options' section. A red box highlights the 'Show Attributes/Columns as Grid' checkbox, which is currently unchecked. Other options include 'Display Column Data Type' (checked), 'Display Column Domain' (unchecked), 'Display Column Null Option' (unchecked), 'Display Physical Primary Key (PK) Designator' (unchecked), 'Display Physical Foreign Key (FK) Designator' (checked), 'Display Physical Alternate Key (AK) Designator' (unchecked), 'Display Owner' (unchecked), 'Display Owner For Non Current Users' (unchecked), 'Display Ungenerated Tables' (checked), and 'Display Ungenerated Indexes' (unchecked).

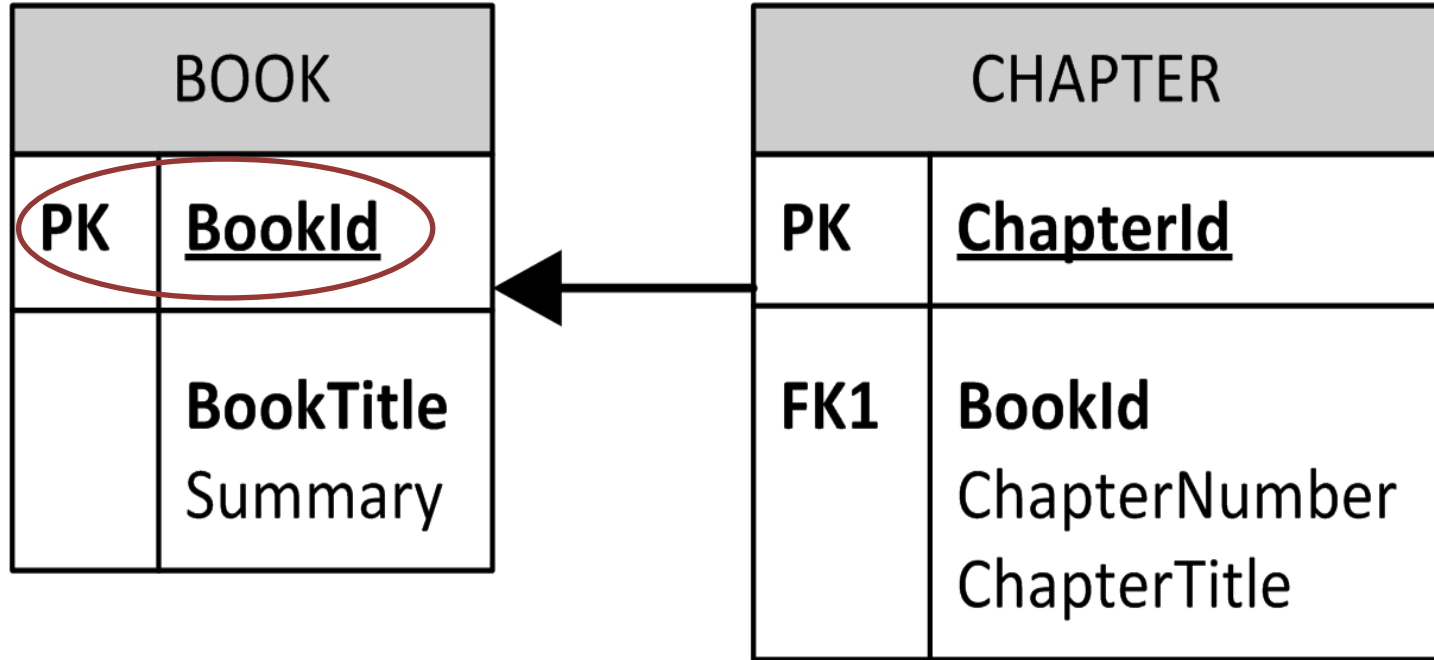
At the bottom left, a 'Properties...' context menu is open, showing options: 'Harden Physical Names' and 'Unharden Physical Names'.

# Best Practice

- **Tables:** Use upper case and singular form in table names – not plural, e.g., “STUDENT” (not “students”)
- **Columns:** Use Pascal notation, e.g., “StudentId”
- **Primary Key:**
  - If the table name is “COURSE”, name the Primary Key column “CourseId”, etc.
  - “Always” use Integer and Identity(1,1) for Primary Keys. Use UNIQUE constraint for other columns that needs to be unique, e.g., “RoomNumber”
- Specify **Required** Columns (NOT NULL) – i.e., which columns that need to have data or not
- Standardize on few/these **Data Types:** *int, float, varchar(x), datetime, bit*
- Use English for table and column names
- Avoid abbreviations! (Use “RoomNumber” – not “RoomNo”, “RoomNr”, ...)

# Identity(1,1)

Typically, you want to use Integer and Identity(1,1) on the Primary Keys



# Identity(1,1)

SQL Server Table 'BOOK' Column 'BookId' Editor

Table: BOOK

Physical Name	Domain Parent	Physical Data Type	Primary Key	Physical Only	Foreign Key
BookId	??_default_	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
BookName	??_default_	varchar(50)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

General SQL Server Constraint Link Indexes Style Comment Where Used UDP History Notes Extended Notes

Datatype

Physical Datatype: integer  
Null Option: Not Null

Options

Row GUID:   
Sparse:   
Is Sparse Column Set:

Identity

Generated Identity:   
Starting Value: 1  
Increment By: 1  
Is Not For Replication:

Collation

XML

Populate All Rows With Default Value:

Expression

Enter SQL text here...

Persisted:

Close Cancel

SQL Server Schema Generation Preview

```
CREATE TABLE BOOK
(
    BookId          integer NOT NULL IDENTITY ( 1,1
),
    BookName       varchar(50) NULL ,
    CONSTRAINT XPKBOOK PRIMARY KEY CLUSTERED (BookId ASC)
)
go
```

Table Filter: 1/1

Generate... Close

# Unique Constraint

Typically, you want to use a UNIQUE constraint for other columns that needs to be unique, e.g., “RoomNumber”

The screenshot displays the SQL Server Enterprise Manager interface. The top window shows the 'AUTHOR' table with columns: AuthorId (int, Primary Key), Author... (varchar(100)), and EMail (varchar(100)). The bottom window shows the 'XPKAUTHOR' index, which is a Primary Key (PK) index on the 'AUTHOR' table. The 'Index Membership' tab is selected, showing the index 'XPKAUTHOR' and its membership in the 'XPKAUTHOR' index. A red circle highlights the 'Index Membership' tab and the 'XPKAUTHOR' index entry. Another red circle highlights the 'Index Editor' window, which is open for the 'XPKAUTHOR' index. A blue arrow points from the 'Index Editor' window to the 'Index Membership' tab.

Physical Name	Domain Parent	Physical Data Type	Primary Key	Foreign Key	Physical Only	PII	Comments
AuthorId	?_default_	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Author...	?_default_	varchar(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
EMail	?_default_	varchar(100)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Physical Name	AK ID	Disabled	Is Unique	Physical Only	General
XPKAUTHOR	PK	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Ind	Type
XPKAUTHOR	PK



# Create Table Script



# Database Design & Implementation

## Recommended Steps:

1. Database Modelling/Design using erwin Data Modeler
2. Generate SQL Table Script using erwin Data Modeler  
(you might need to adjust/improve it in order to make it more robust)
3. Create Tables in SQL Server using the SQL Script generated by erwin Data Modeler
4. Create Stored Procedures, View, Triggers, etc. inside SQL Server if needed.

# Forward Engineering Schema

The screenshot displays the SQL Server Enterprise Architect interface. The 'Actions' ribbon is active, and the 'Forward Engineer Schema' button is highlighted with a red circle. A tooltip for this button is visible, showing the text 'Schema Forward Engineer Schema Generation'. The main workspace shows an ER diagram titled 'Model1 : ER\_Diagram\_164 \*' with the following tables and attributes:

- AUTHOR**: AuthorName: varchar(100), AuthorId: int, Email: varchar(100)
- BOOK**: BookId: int, BookTitle: varchar(100), Summary: varchar(100), PublisherId: int (FK), AuthorName: varchar(100) (FK)
- CHAPTER**: ChapterId: int, BookId: int (FK), ChapterNumber: varchar(100), ChapterTitle: varchar(100)
- PUBLISHER**: PublisherId: int, PublisherName: varchar(100), Email: varchar(100)

The diagram shows relationships: AUTHOR to BOOK (one-to-many), PUBLISHER to BOOK (one-to-many), and BOOK to CHAPTER (one-to-many).

### Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default Schema Generation Open... Save

SQL Server 2019 Schema Generation

Table

- Pre-Script
- Post-Script
- Create
  - Table
  - Statistics
  - Procedure
  - Function
  - Partitions
- Table Validation
  - Create
  - Alter
- Physical Storage
- Drop
  - Table
  - Statistics
  - Procedure
  - Function

Database Template: SqlServer2019.fet Browse

### Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default Schema Generation Open... Save Save As... Delete

SQL Server 2019 Schema Generation

Column

- Validation
  - Create
  - Alter
- Physical Order
- Default
  - Create
  - Alter
- User Datatype

Forward Engineer Schema Generation Wizard

### Schema Generation Options

This page allows the user to change the Forward Engineer Schema Generation Options.

Option Set: Default Schema Generation Open... Save Save As... Delete

SQL Server 2019 Schema Generation

Referential Integrity

- Generate Column Level Constraint
- Primary Key (PK)
  - CREATE/PK
  - ALTER/PK
- Foreign Key (FK)
  - ON DELETE
  - ON UPDATE
  - Statement Format
    - CREATE/FK
    - ALTER/PK
- UNIQUE(AK)
  - CREATE/AK
  - ALTER/AK

Database Template: SqlServer2019.fet Browse... Edit... Reset

< Back Next > Generate OK Cancel Help

# Table Script

Forward Engineer Schema Generation Wizard

**Schema Generation Preview**  
This page provides a preview of the Forward Engineer Schema Generation.

Overview  
Option Selection  
Summary  
Owner Override  
Table Filter  
**Preview**

1  
2 CREATE TABLE AUTHOR  
3 (  
4 AuthorId int IDENTITY ( 1,1 ) NOT NULL ,  
5 AuthorName varchar(100) NULL ,  
6 EMail varchar(100) NULL ,  
7 PRIMARY KEY CLUSTERED (AuthorId ASC),  
8 UNIQUE (AuthorName ASC)  
9 )  
10 go  
11  
12 CREATE TABLE PUBLISHER  
13 (  
14 PublisherId int IDENTITY ( 1,1 ) NOT NULL ,  
15 PublisherName varchar(100) NULL ,  
16 EMail varchar(100) NULL ,  
17 PRIMARY KEY CLUSTERED (PublisherId ASC),  
18 UNIQUE (PublisherName ASC)  
19 )  
20 go  
21  
22 CREATE TABLE BOOK  
23 (  
24 BookId int IDENTITY ( 1,1 ) NOT NULL ,  
25 BookTitle varchar(100) NULL ,  
26 Summary varchar(100) NULL ,  
27 PublisherId int FOREIGN KEY REFERENCES PUBLISHER (PublisherId) ON DELETE CASCADE ON UPDATE CASCADE )  
28 )  
29 go

< Back Next > Generate OK Cancel Help



# SQL Server

# Create Tables in SQL Server

The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the server hierarchy for 'XPS15HPH\SQLEXPRESS (SQL Server 13.0.1742 - sa)'. The main window shows a SQL query with three CREATE TABLE statements. The Messages pane at the bottom indicates that the command was completed successfully.

```
CREATE TABLE AUTHOR
(
    AuthorId          int IDENTITY ( 1,1 ) NOT NULL ,
    AuthorName        varchar(100) NULL ,
    Email              varchar(100) NULL ,
    PRIMARY KEY CLUSTERED (AuthorId ASC),
    UNIQUE (AuthorName ASC)
)
go

CREATE TABLE PUBLISHER
(
    PublisherId       int IDENTITY ( 1,1 ) NOT NULL ,
    PublisherName     varchar(100) NULL ,
    Email              varchar(100) NULL ,
    PRIMARY KEY CLUSTERED (PublisherId ASC),
    UNIQUE (PublisherName ASC)
)
go

CREATE TABLE BOOK
(
    BookId            int IDENTITY ( 1,1 ) NOT NULL ,
    BookTitle         varchar(100) NULL ,
    Summary           varchar(100) NULL ,
)
```

Messages  
Command(s) completed successfully.

Query executed successfully. | XPS15HPH\SQLEXPRESS (13.0 RTM) | sa (55) | BOOKS | 00:00:00 | 0 rows



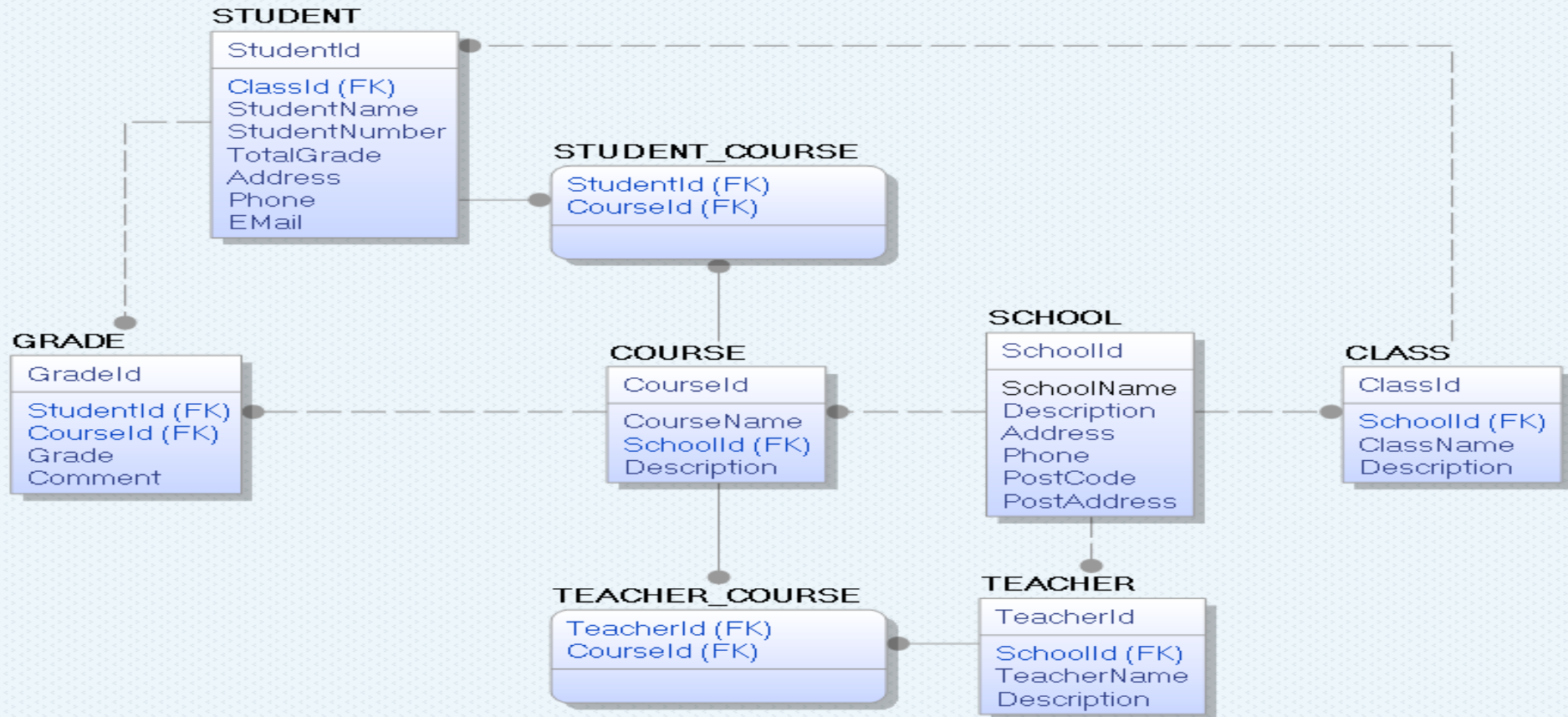
# Summary

# Summary

1. Database Modelling/Design using erwin Data Modeler
2. Generate SQL Table Script using erwin Data Modeler (you might need to adjust/improve it in order to make it more robust)
3. Create Tables in SQL Server using the SQL Script generated by erwin Data Modeler
4. Create Stored Procedures, View, Triggers, etc. inside SQL Server if needed.



# Are you able to make this diagram?



# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

