Ch 9: Physical Model Design

Mike Lynott
Adjunct Prof, CS
Boise State University

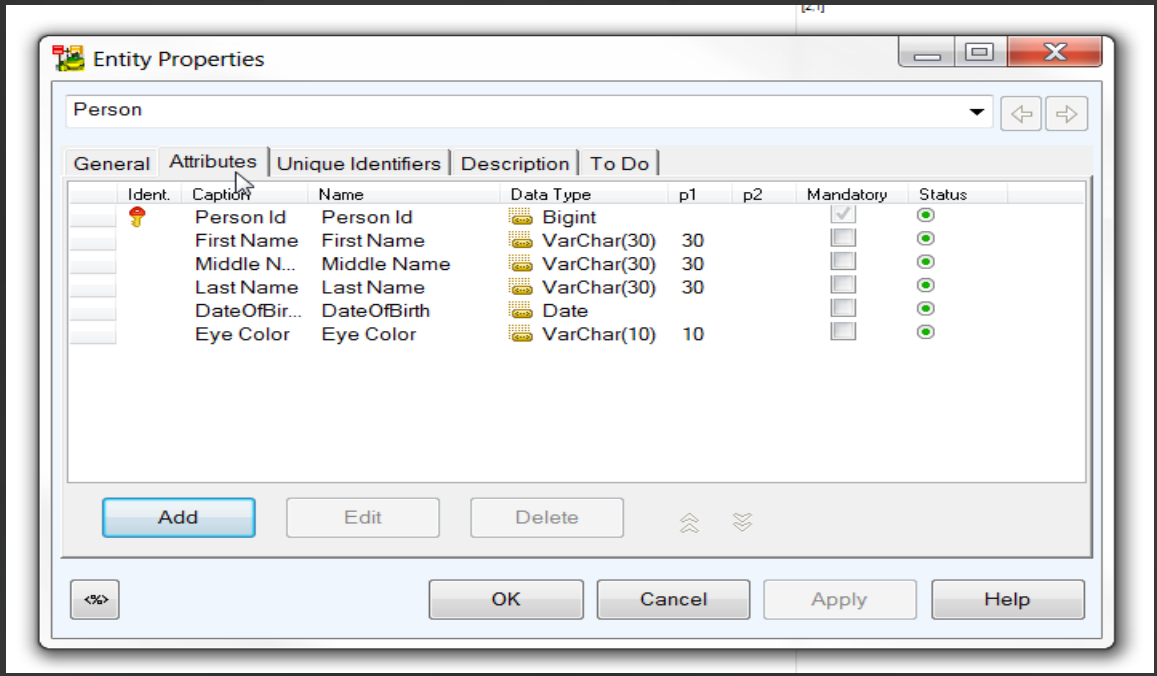
Agenda

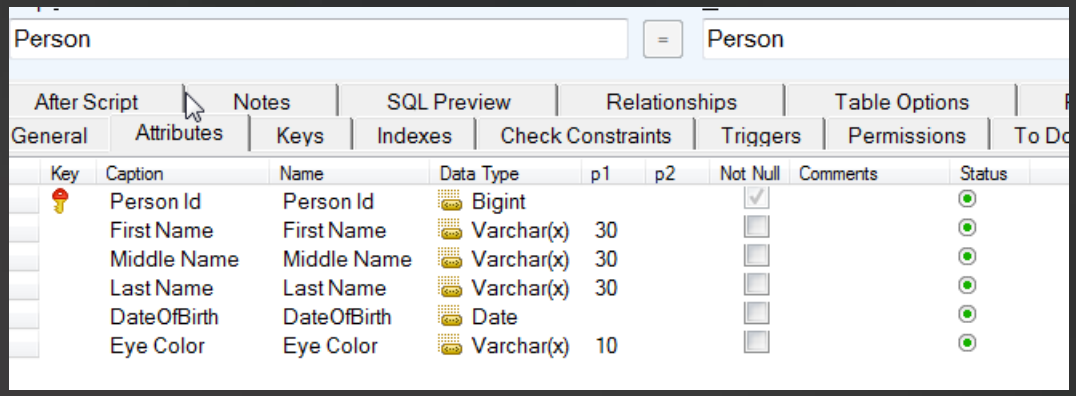
- Entities
- Attributes
  - Audit fields
- Relationships
  - XOR / Arc'd relationships
- Subtypes
  - Option 1
  - Option 2
- Reference data

Entity->Table

- Generally, each entity is translated into a table.
  - Except supertype/subtype sets (which we'll deal with below)
  - And, in many cases, some "type" entities may be combined into a single table

Example Logical->Physical Model





Example Physical Model->SQL DDL

Person | Person

After Script | Notes | SQL Preview | Relationships | Table Options

General | Attributes | Keys | Indexes | Check Constraints | Triggers | Permissions | To Do

Key	Caption	Name	Data Type	p1	p2	Not Null	Comments	Status
	Person Id	Person Id	Bigint			✓		
	First Name	First Name	Varchar(x)	30				
	Middle Name	Middle Name	Varchar(x)	30				
	Last Name	Last Name	Varchar(x)	30				
	DateOfBirth	DateOfBirth	Date					
	Eye Color	Eye Color	Varchar(x)	10				

Caption

Person

General | Attributes | Keys | Indexes | Check Constraints

After Script | Notes | SQL Preview | Relations

```
1
2   -- Table Person
3
4   CREATE TABLE Person
5   (
6     Person Id Bigint NOT NULL,
7     First Name Varchar(30),
8     Middle Name Varchar(30),
9     Last Name Varchar(30),
10    DateOfBirth Date,
11    Eye Color Varchar(10)
12  )
13  ;
14
15  ALTER TABLE Person ADD PRIMARY KEY (Person Id)
16  ;
```

In TDM

- Physical Model still shows “Entity” (not table)
- You can tell which modeler you’re in because the number of options for a physical model includes a SQL Preview tab—and many others

Attributes

- Each logical attribute->column
- For each target db option (brand + version) there is a mapping from logical datatype to physical datatype.
- The human designer needs to review each attribute/column to insure that the translated datatype is correct.

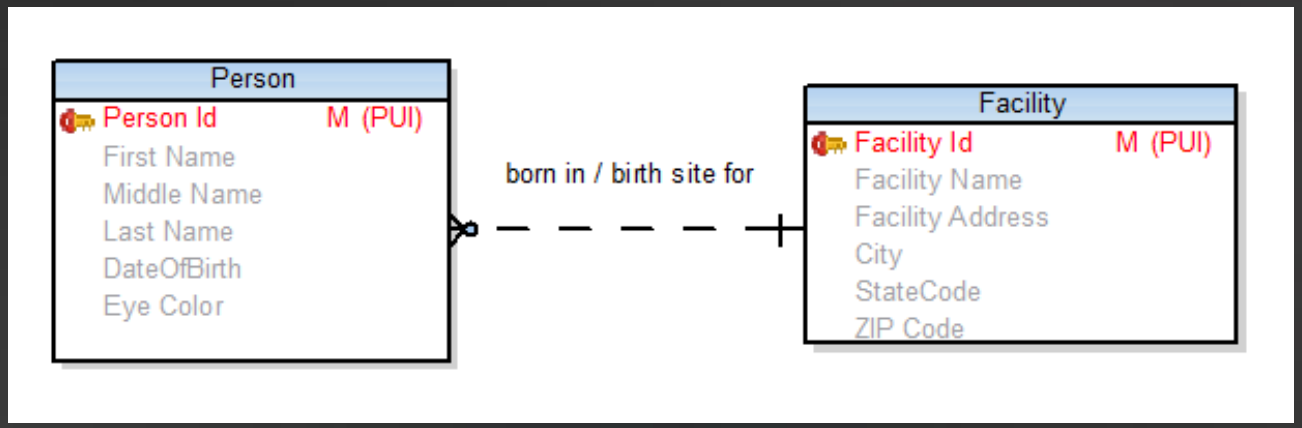
“Roll-your-own” Conversion

- In some tools, it is possible to change the logical->physical mappings.
- In TDM, see Data Type Conversion Settings

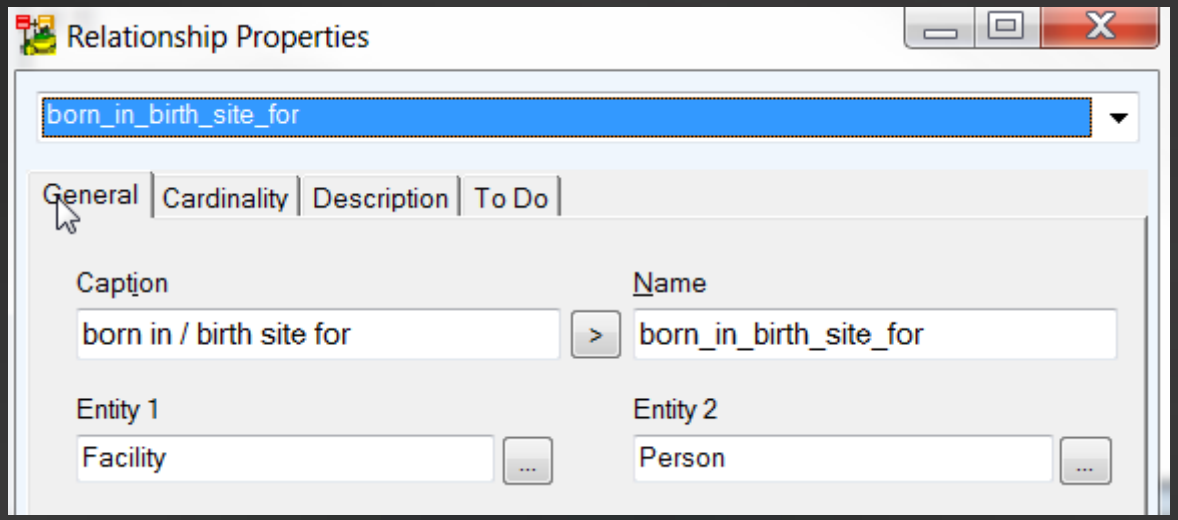
Relationships

- In general, a relationship is translated as follows:
  - A Foreign Key column is added to the entity/table on the crow's foot side of the relationship.
  - A Foreign Key constraint is created on the same entity/table, with the name from the relationship's "name" field.

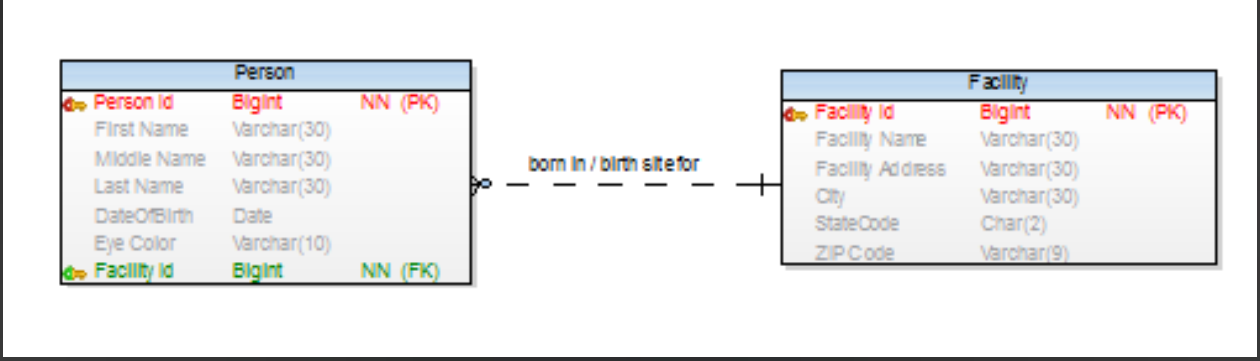
Example Logical Relationship



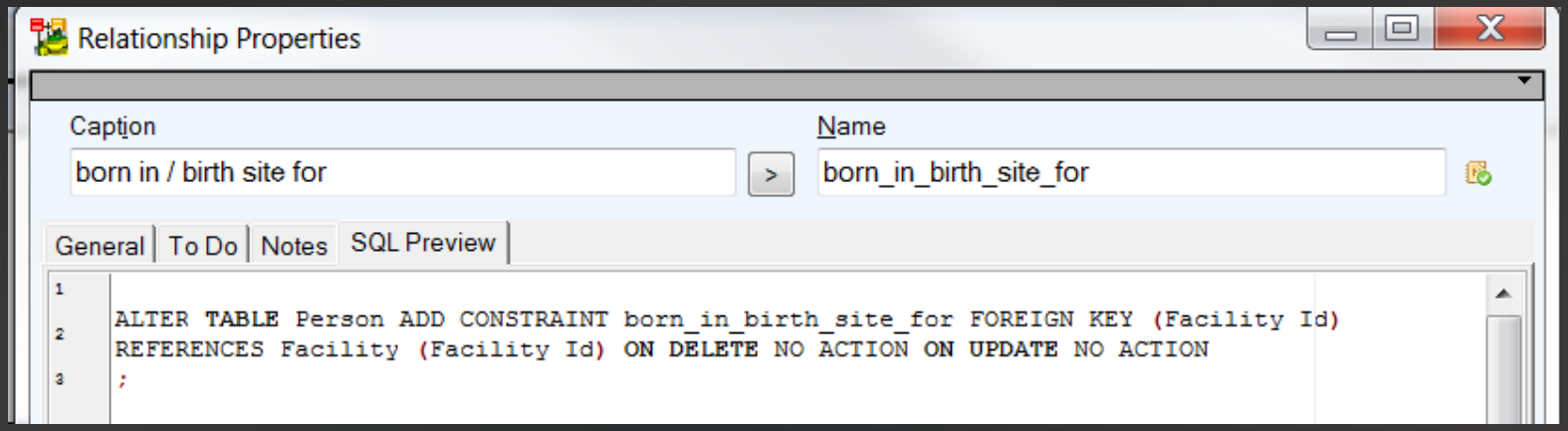
Logical Relationship details



Physical Relationship



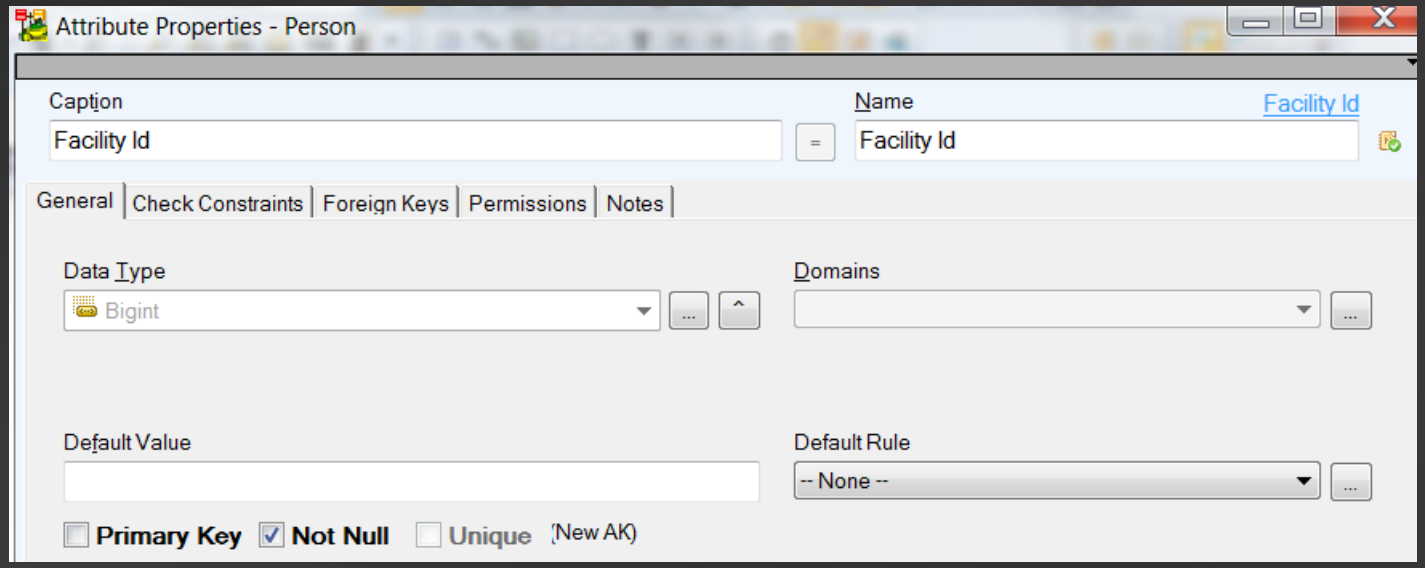
Physical relationship->SQL DDL



“Cascade” Options

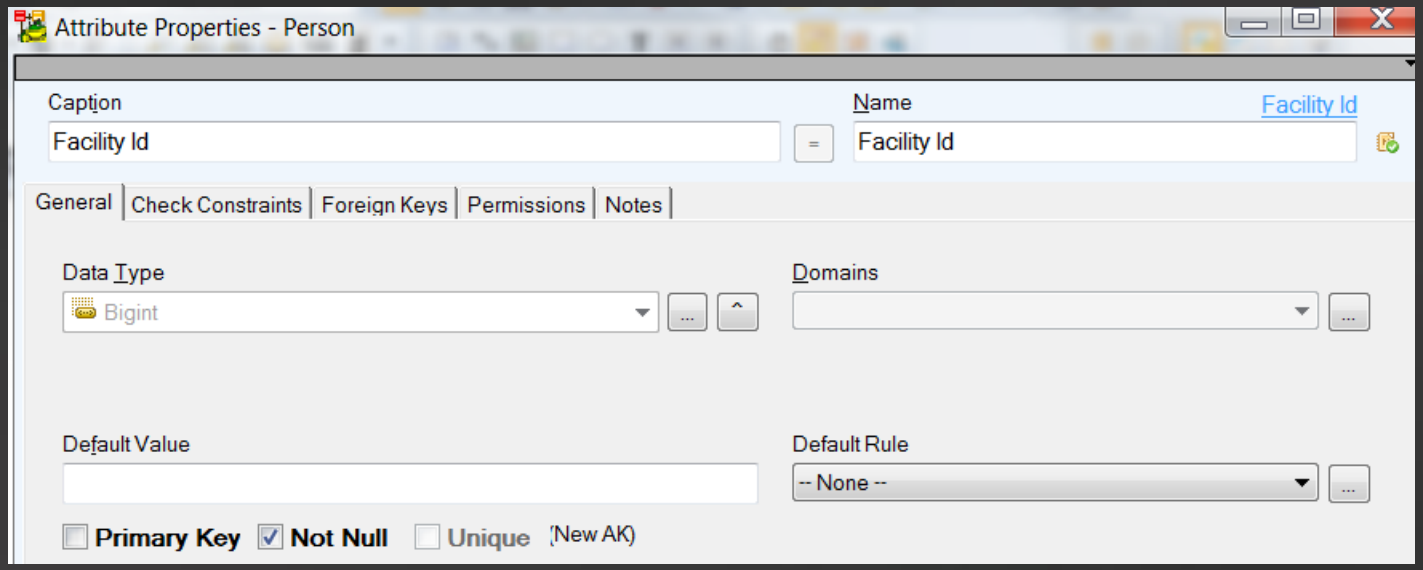
- On Delete
  - None
  - Restrict: don’t allow a deletion if, for any Facility, there is a related Person.
  - Cascade: if a Facility is deleted, delete all related Person records
  - Set NULL: If a Facility is deleted, the Facility Id of all related Person records is to be set to NULL
  - Set Default: If a Facility is deleted, the Facility Id of all related Person records is to be set to the default value.

Default example



Where is the default value found?

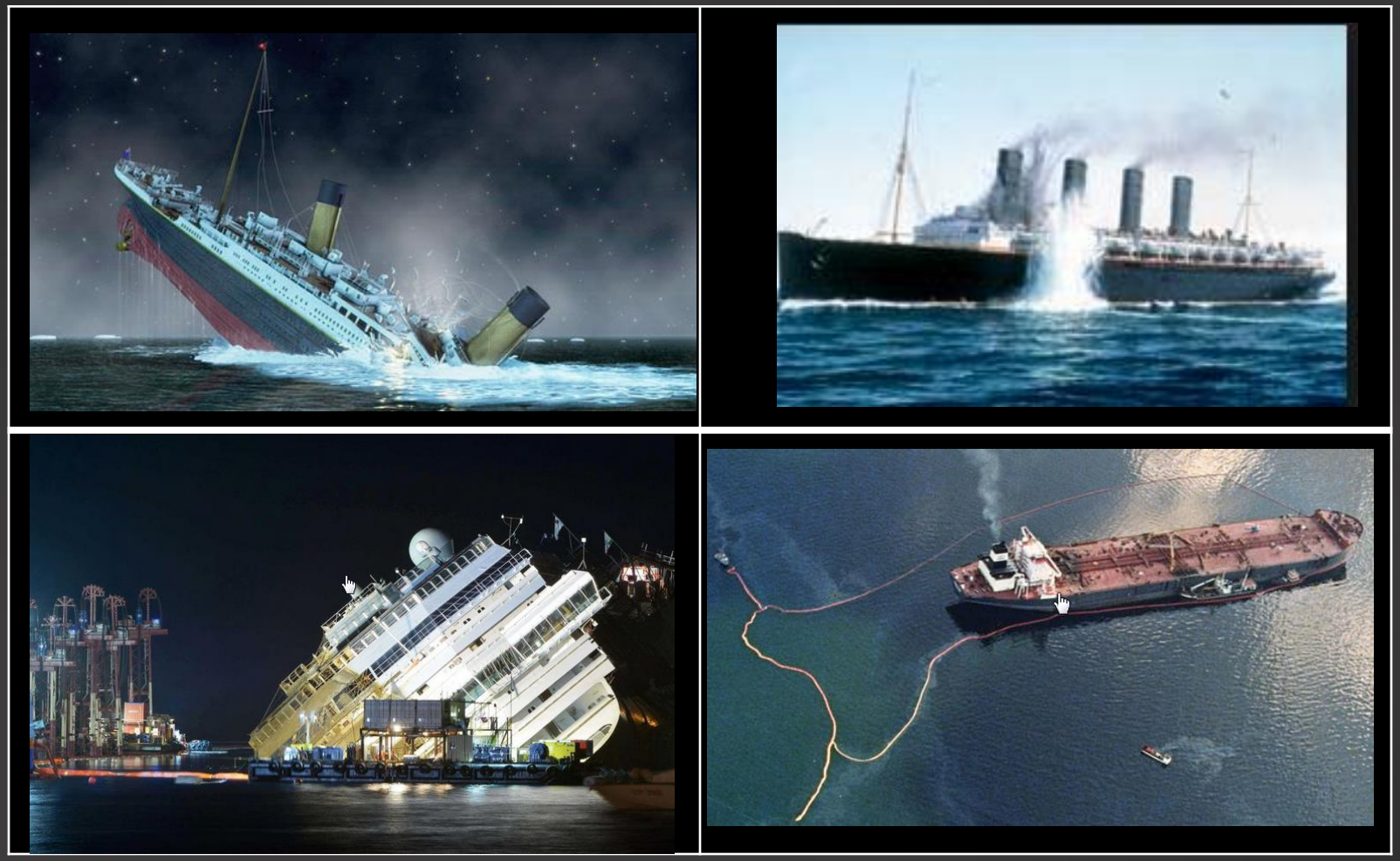
- Note that the default option in the Alter Table statement is detailed on the Person.Facility_Id



“On Update” clause

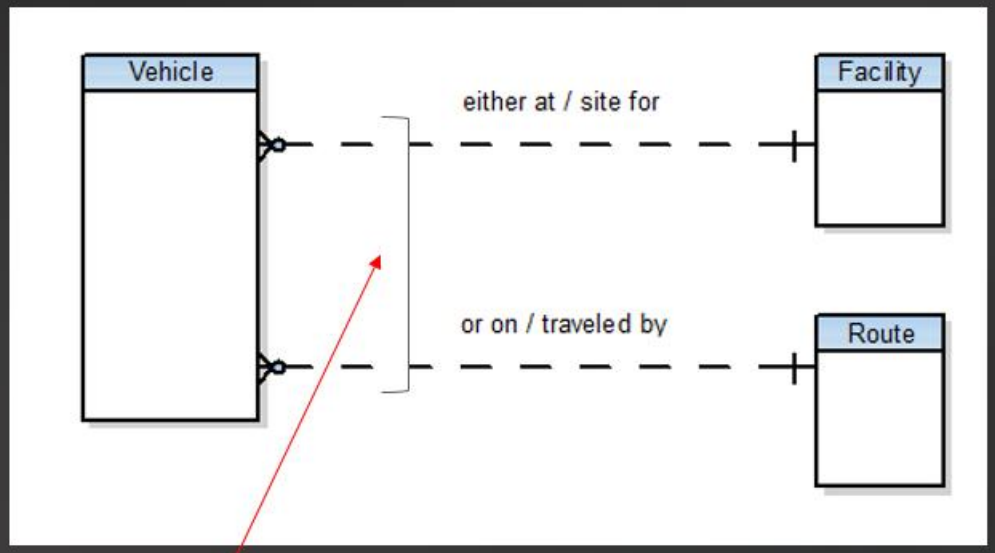
- DON’T EVEN THINK ABOUT IT!
- If someone near you suggests that you need to allow a table’s Primary Key to be updateable, please report this looming disaster to the nearest responsible adult.
- The default “On Update NULL” should be the only option specified on any Foreign Key constraint specified for your database.

Images for DBs using “On Update”



XOR Relationships

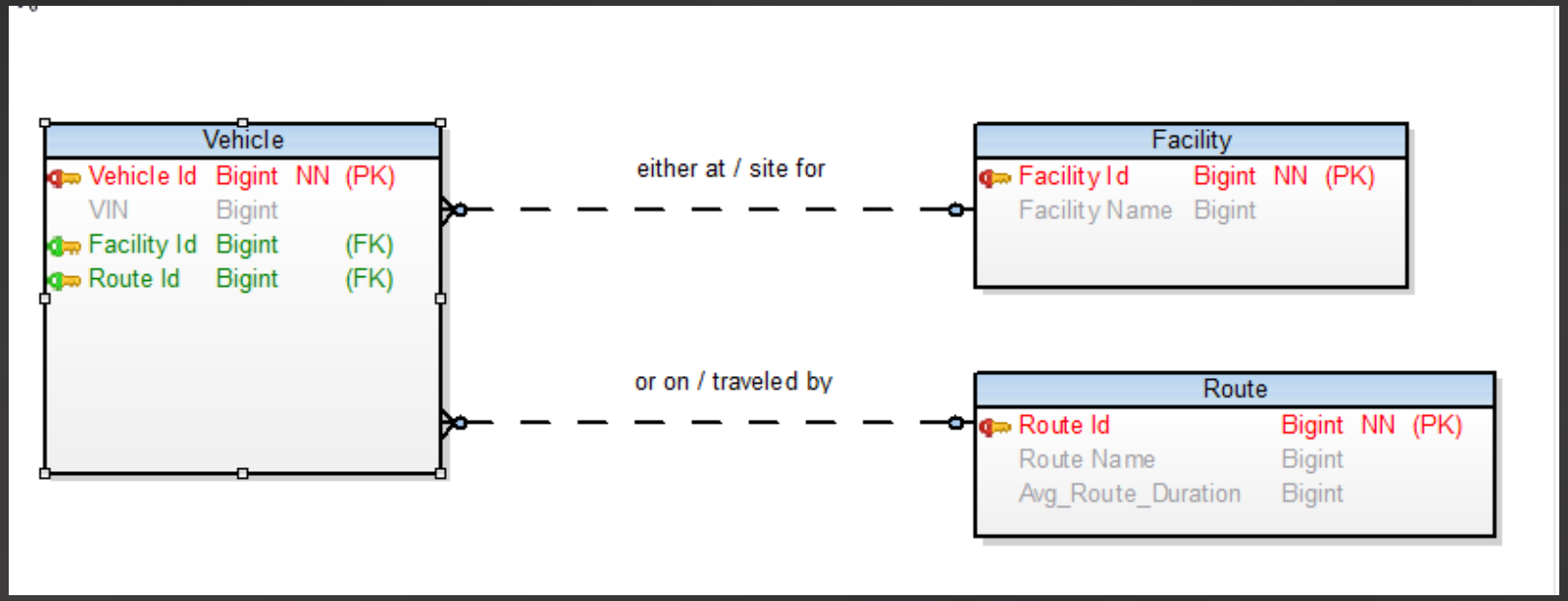
Reminder: Logical Model



Added using
PPT graphics

XOR Relationships

Option 1: two foreign keys



XOR Relationships

Opt 1 (cont.)

- Add a check constraint:

  NOT (Facility_Id IS NOT NULL and Route_Id IS NOT NULL)

  AND

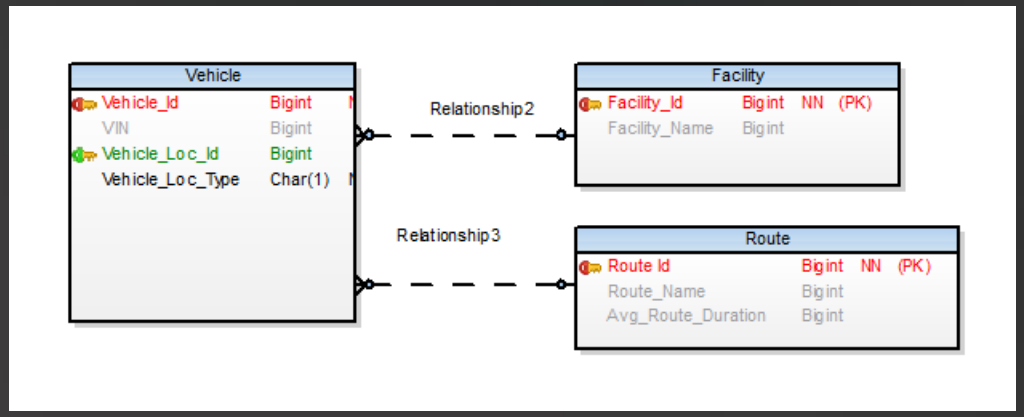
  NOT(Facility_Id IS NULL AND Route_Id IS NULL)

- The logic of a check constraint is a bit tortuous, but it means “you can’t have both FK columns populated, and you can’t have both FK columns empty.” The only acceptable condition is one FK column is NULL and one is NOT NULL.

XOR Relationships

Option 2: One pseudo-FK column

- The “natural” FK columns are replaced by a single column
- There is a “type” column to indicate which table the Vehicle_Loc_Id column points to.



XOR Relationships

Option 2: (cont)

- In the final design, we can't have a "real" FK, because we would end up with an unsupportable (i.e., WRONG) syntax, shown on the next page.

The problem area

- -- Create relationships section ----------
- ALTER TABLE Vehicle ADD CONSTRAINT Relationship2 FOREIGN KEY (Vehicle_Loc_Id) REFERENCES Facility (Facility_Id) ON DELETE NO ACTION ON UPDATE NO ACTION;
- ALTER TABLE Vehicle ADD CONSTRAINT Relationship3 FOREIGN KEY (Vehicle_Loc_Id) REFERENCES Route (Route Id) ON DELETE NO ACTION ON UPDATE NO ACTION;

XOR Relationships

Option 2 (cont 2)

- So if we drop the FK relationships how do we confirm data integrity?
- Add a trigger (on insert and update of Vehicle) that checks the Vehicle_Loc_Type column, and confirms that the value presented in Vehicle_Loc_Id is found in one of the related tables (Facility or Route).
- We are replacing the “behind-the-scenes” validation with an explicit one.

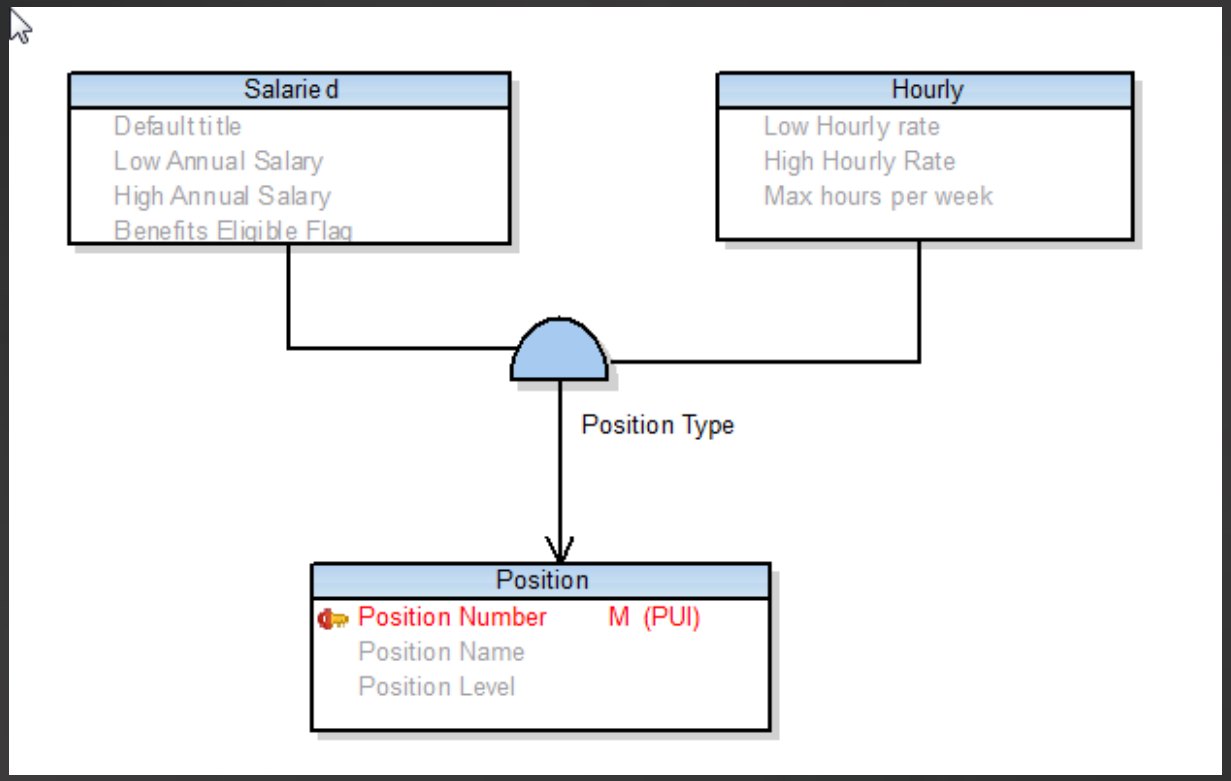
Which design is better?

- There are passionate advocates for both.
- Option 1 uses the “native” database checks, and is often used if there are only two XOR relationships.
- Option 2 may be selected if there are more than two XOR relationships—but this is very rare
- In either case, there is an option that is seductive, BUT…

We can play “Let’s pretend”

- You can pretend that facilities and routes are the same thing!
- That way you don’t have to worry about this pesky XOR design choice.
- This is NOT a good idea (refer to the ships’ photos above for my prediction.)

Subtypes
Example



Subtype design options
after the late Steven J Gould

- Splitters: Every “leaf” subtype in a subtype tree becomes a table
- Joiners: A subtype tree becomes one table and many views

- These terms are not industry buzzwords. They were used by Gould to identify paleontologists.

The “Splitters” Design

- Two tables
  - Salaried Position
  - Hourly Position
- Each table would include its subtype’s attributes and relationships, AND the attributes and relationships of its supertype.
- This might create XOR relationships that didn’t appear in the original model!

The “Joiners” Design

- One table
  - Position, which holds all the attributes of the original Position PLUS all the attributes and relationships of the two subtypes.
  - Any mandatory attributes of the subtypes would become optional in the resulting table.
  - Andy mandatory relationships of the subtypes would become optional in the resulting table.
- Two views
  - One for each subtype.

“Joiners” Design (2)

- One variant of the Joiners Design
  - The Position table exists, but is not accessible by any role
  - All operations are performed through the views—with appropriate triggers to enforce optionality etc.
- Another variant of the Joiners Design
  - The Position table exists, but is inaccessible
  - The two views exist for read only
  - Persistent Stored Modules (PSM) are used to manipulate the table (insert/update/delete.)

Comments on Design Choices

- These choices are rarely reviewed and debated.
- The default Joiners design is probably most common.
- Dislike of structures-other-than-tables often guides these design choices.
- Very infrequently, maintainability is discussed, though Total-Cost-of-Ownership (TCO) would suggest otherwise.
- Most of the time, performance ("assumed" performance) is considered paramount.