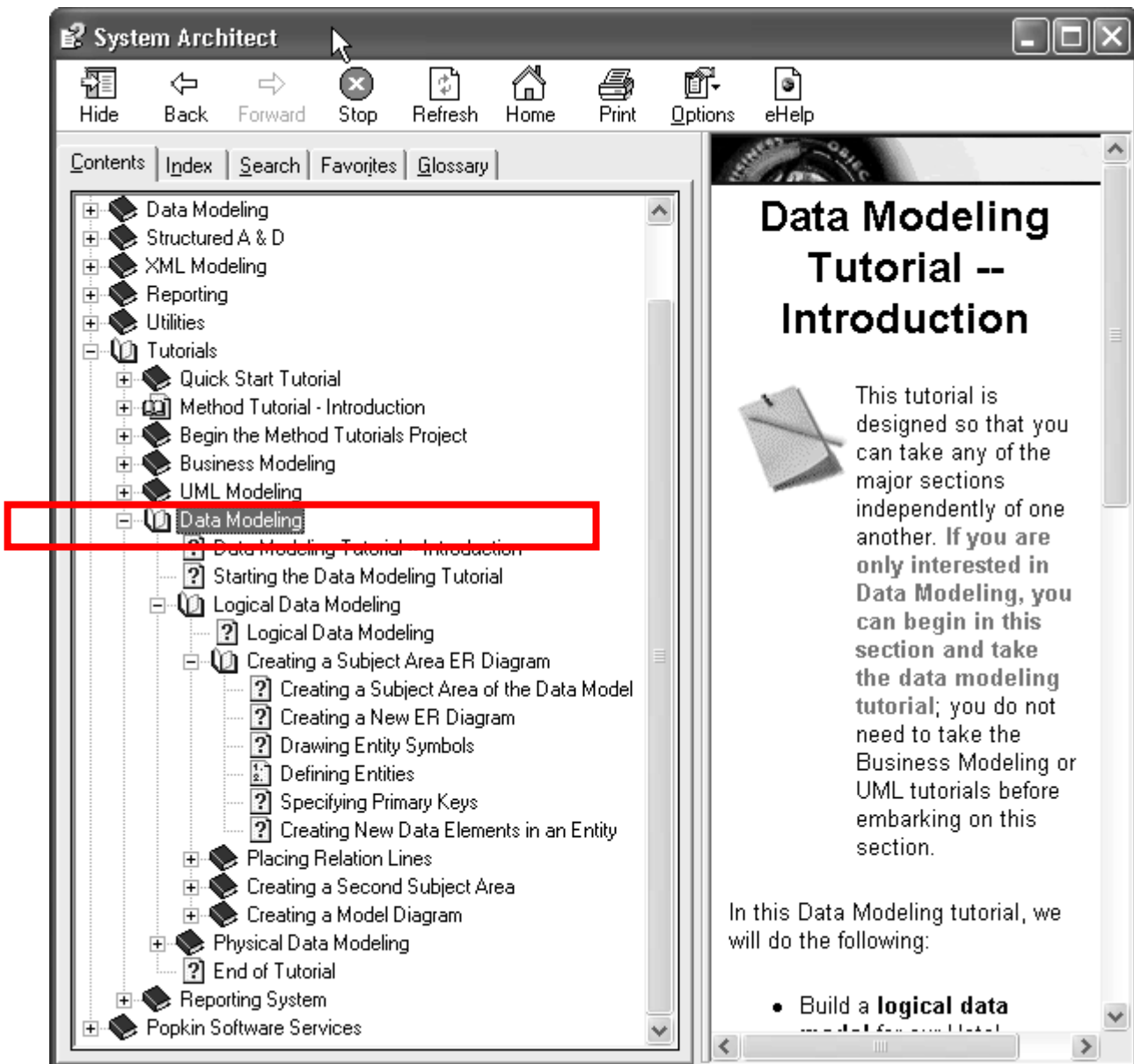


Complex ERDs – Data modelling

This material has been abstracted and slightly modified from the System Architect tutorial File

Robin Beaumont 05/01/2004 11:18



Contents

Contents	2
1. The Project: An International Hotel Business	3
2. Introduction.....	4
3. Starting System Architect and Opening the Tutorial Encyclopedia.....	4
4. Setting Preferences	5
5. Logical Data Modeling	6
6. Creating a Subject Area of the Data Model	6
7. Creating a New ER Diagram	7
8. Drawing Entity Symbols.....	7
9. Defining Entities.....	9
10. Specifying Primary Keys.....	11
11. Creating New Data Elements in an Entity.....	12
12. Relationships Between Entities	14
13. Drawing Relationship Lines	15
14. Relationship Line Types	16
15. Foreign Keys	16
16. Creating a Second Subject Area Diagram.....	17
17. Drawing the Diagram	18
18. Update Foreign Keys	20
19. Creating a Model Diagram.....	20
20. The Physical Data Model.....	22
21. Generating a Physical Data Model from an ER Diagram.....	23
22. The Physical Table	24
23. Display Options for a Table	26
24. Constraints	26
25. Generating Schema.....	26

Comments I have added are in red - RB

1. The Project: An International Hotel Business

In this tutorial, we will examine a fictitious International Hotel Chain. For argument's sake, we'll call this company Chelsea Hotels and Resorts.

Chelsea Hotels and Resorts

Chelsea is a chain of inns that has been around a number of years. The company has competed in the low-cost accommodations business segment by primarily providing rooms for cost-conscious families on vacation. The Restaurants business segment provides an on-site restaurant featuring home cooked food catering to families. The company has recently begun to expand its business, entering the resort business segment (and thus adding the word Resorts to its name), and the business travel segment. The management at Chelsea has realized that an effective Internet presence is crucial to compete in these business segments. Company management has determined that this does not simply mean designing a pretty site to introduce the company. The company's main focus is to examine business functions and processes, while making sure that the company is operating efficiently and without redundancy. They want to integrate a website into the business processes. As part of this effort, the company has determined that they want to build an effective on-line reservation system.

Chelsea's high-level business objectives can be summarized as stated:

- Create a market presence as a leader in resort travel
- Create a market presence as an ideal solution for business travel
- Achieve Sales Growth of 20 % per year for next 5 years
- Create efficient on-line Reservation System
- Integrate website with improved business processes.

This Tutorial

The SA/SE tutorial is divided into three parts; business modelling – which we will not look at, Data modelling – this tutorial and a UML tutorial which is where we will concentrate our efforts.

- In the **Data Modeling** section of the tutorial, we'll design part of the database that stores reservation information.
- In the **UML** section of the tutorial, (this consists of several documents) we'll design part of the hotel's reservation system.
- In the **Business Modeling** section of the tutorial, we will take a look at the business as it currently stands, determine problems that exist, and state Business Objectives that we would like to realize. We will then undertake the project to build a new e-Business reservation system for the hotel. We will take a look at the business process flows of the business (and build a new one), and examine the functions and organizational hierarchy of the business. (we will not be too concerned with this for the course you are on)

Through it all, we will be answering questions concerning the who, what, where, when, how, and why of the business. As we go, we will follow a work flow 'process map' that will walk us through the Zachman Framework. We'll take a look at the work flow 'process map' in a bit; first, we'll take a look at the Zachman Framework and System Architect's mapping to it.

2. Introduction

In this Data Modeling tutorial, we will do the following:

- Build a **logical data model** for our Hotel Reservation System. This data model will be one '**subject area**' of the overall system. We will add entities to the diagram, build relationships between **entities**, and add **attributes** to the entities. We will examine System Architect's underlying data dictionary and repository.
- Build a second logical data model, 'subject area' diagram (if you have taken the UML tutorial, this part has been done already).
- Build a '**model**' **diagram** for the system. This diagram will be built automatically by System Architect from the information contained in the two diagrams built above.
- Map one of our logical data models to a **physical data model**.
- Examine **tables** and **columns** in the physical data model, and the display modes, or views, that we can select for the diagram.

3. Starting System Architect and Opening the Tutorial Encyclopedia

To start System Architect:

1. Run System Architect from the **Start, Programs** menu or double-click on the **System Architect** program icon.
2. To open the **TUTORIAL** encyclopedia, select **File, Tutorial**.

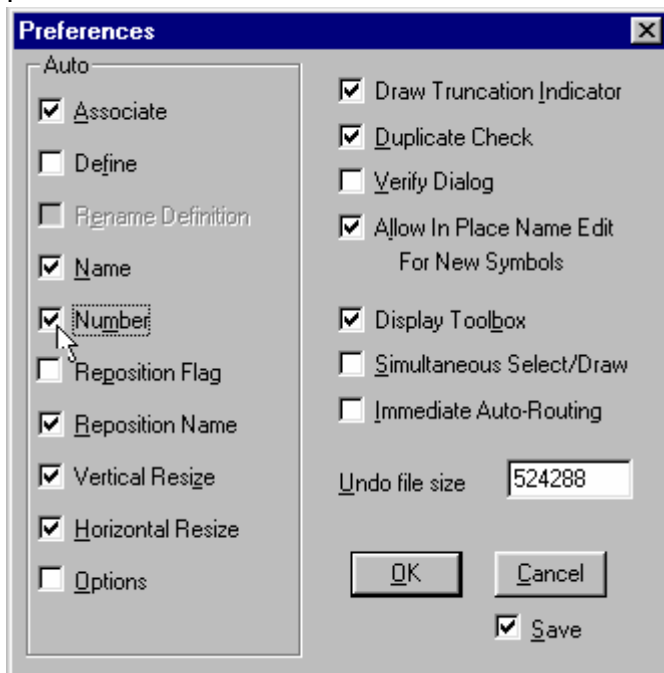
4. Setting Preferences

Now you are almost ready to begin the tutorial project. However, first you need to set your preferences for how System Architect will react to your drawing commands.

System Architect offers many optional drawing and tool behavior features. They can be turned on or off at any time with the Preferences dialog.

To set **Preferences**:

1. From the **Tools** menu, select **Preferences**. The **Preferences** dialog box appears.
2. Set the selections to match those shown in the diagram below.
3. Click **OK** to close the **Preferences** dialog

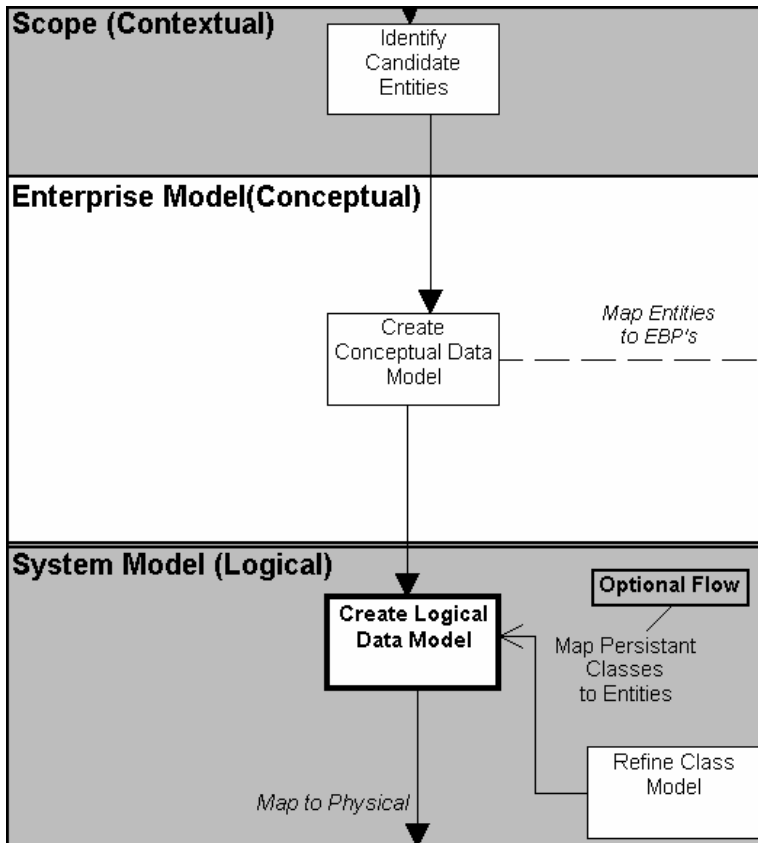


Explanation of Some Preferences		
Preferences (Auto,...)	Setting	Explanation
Associate	On	Brings up the Associative dialog whenever you draw a relationship or constraint line.
Name	On	Brings up the Add Name dialog whenever you draw a symbol on a diagram.
Define	Off	Enables you to place symbols on the diagram and define them later. This technique is useful for brainstorming.

For a fuller explanation of the Preferences dialog, select Help from the System Architect menu. You can also find this information in your System Architect User Guide or System Architect Tutorial manual. These manuals are provided in .pdf format in the **System Architect Student Version\Manuals** directory.

5. Logical Data Modeling

The entity relation diagram is a logical representation of the data structure and data relationships of the database. It focuses on what is represented in the database (not how it is represented, which is covered in the physical diagram). It includes entities to represent each person, place, event, or concept about which information will be maintained in the system, and relation lines to represent the associations between the entities.



Conceptual Data Model

In the Conceptual Data Model, you concentrate on what the entities of your system might be, and decide which ones are related to one another. More detailed information, such as the specific attributes of the entities or the complexities of the relationships are not considered appropriate for a conceptual model. A simple textual description of each entity, or perhaps the most important attribute (like the primary key), will suffice. To build a Conceptual Data Model in System Architect, you would use an ER diagram, and use the Non-specific relation line to model relationships between entities. We will not create a Conceptual Model in this tutorial.

Logical Data Model

From the conceptual model, you can develop a logical model, by adding details. For example, you replace the Non-specific relation line between entities with Identifying or Non-Identifying relationships. You model the attributes of the entities, and specify their generic data types. You can normalize your model, and specify which attributes are primary and foreign keys for the entities. You might even specify access paths for the entities - ways in which entities can be navigated to find information quickly. (Access paths

are the logical counterpart to indexes and constraints in a physical diagram.)

Within the context of the tutorial process map, we are at the stage of modeling shown in the lowest gray area of the picture below.

The completion time of this section is approximately 20 minutes.

6. Creating a Subject Area of the Data Model

In this section we will create a small data model diagram for a section of our system. A large system requires a large model. It is quite common for many different designers to work on a large system; in System Architect you can divide the model into smaller pieces, just as the project is divided into smaller pieces or responsibilities. A piece of the entire model is referred to as a "subject area data model", this model is analogous to a small area of responsibility within your project.

In this section of the tutorial, we will create it as a subject area data model; later we will create the overall model diagram. If you have worked through the UML portion of this tutorial, you have already created a subject area for part of the reservation system. This section is independent of your having worked through the UML section - we will start with building a data model in System Architect.

Scenario for Checking Into Hotel

Let's think about how a customer checks in to the hotel. We have a customer with a reservation, who we can think of at this point as a guest. The guest and his or her companions or family members approaches a sales clerk, who checks the guest and companions into a room. The guest checks into the hotel during a certain time

of the day, which correlates to a shift that is manned by a certain number of sales clerks. The guest may have a vehicle that he needs to park in the hotel parking lot. Let's model how the following entities are related:

- Guest
- Reservation
- Shift
- Room
- Vehicle

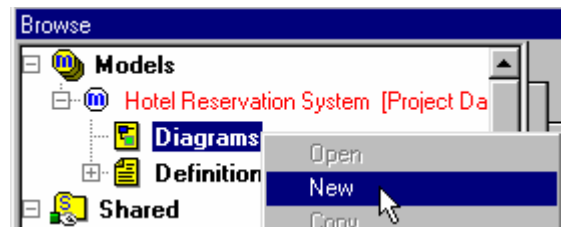
7. Creating a New ER Diagram

Let's create a subject area data model for checking into a hotel:

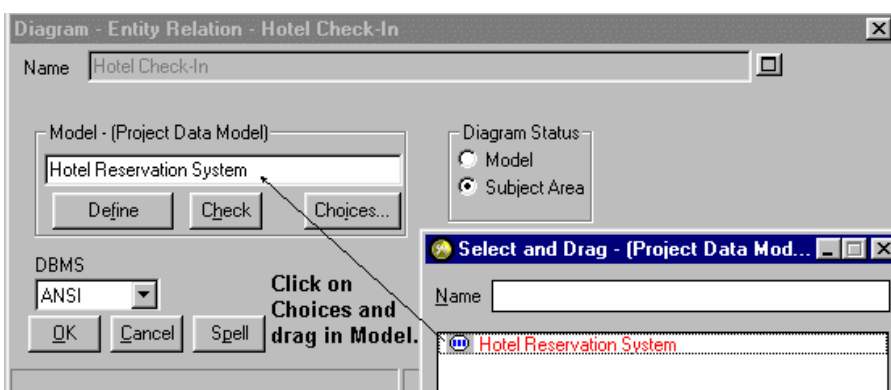
1. Select the **Data Modeling** tab in the browser.
2. From the **File** menu, select **New Diagram**.

or

in the **Data Modeling** tab of the browser, click on the **+** mark to expand **Models**, then **Hotel Reservation System**, and then right-mouse click on **Diagrams** and select **New**.



3. A dialog will open presenting you with a list of possible diagram types. Double-click on **Entity Relation**.
4. In the **New Diagram** dialog, enter the name of the diagram to be created, **Hotel Check-In**, and click **OK**. The **Diagram-Entity Relation** dialog opens. In this dialog we will specify whether this is going to be a model diagram or a subject area of a model. In either case, we must specify the model that we are representing.
5. If not already selected, toggle on **Subject Area** for the **Diagram Status** property.
6. For the **Model** property, click on the **Choices** button. The **Select and Drag** dialog will open, presenting a list of models already created in this project encyclopedia. The dialog should only display the model **Hotel Reservation System**.
7. Select and drag **Hotel Reservation System** into the **Model** property box and close the **Select and Drag** dialog (by clicking on **X** in its upper right-hand corner). (If no model existed at this point, you could type a new model name in the box and click **Define** to create it.)



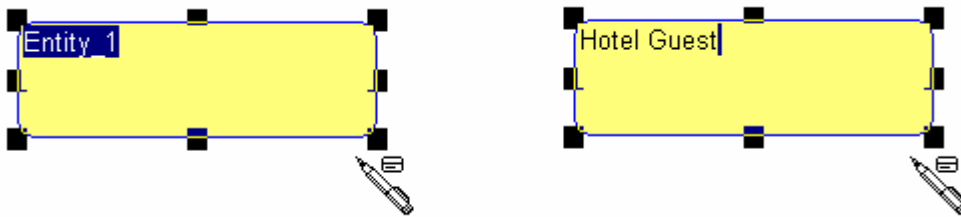
8. You do not have to select a **DBMS** at this point. You can leave it at its default, **ANSI**.
9. Click **OK** to close the dialog and create the Entity Relation diagram.

8. Drawing Entity Symbols

A Hotel Guest has a Reservation that enables them to have a Room; or "the room belongs to the Hotel Guest." The guest may own a car (or Vehicle) that they wish to park, in which case Reception needs to know the license plate number of the car, and the guest it belongs to. Let's draw the entities Hotel Guest, Reservation, Room, and Vehicle.

To place a new entity symbol:

1. Select the **Entity** symbol from the **Draw** toolbar and click on the diagram work area where you want the symbol to be placed - select a spot in the middle of the diagram workspace - leave room to add entities above it, as we will do later. The entity symbol appears with the default name, **Entity_1**, selected.
2. Overtyping the default name with **Hotel Guest** and press **Enter**.

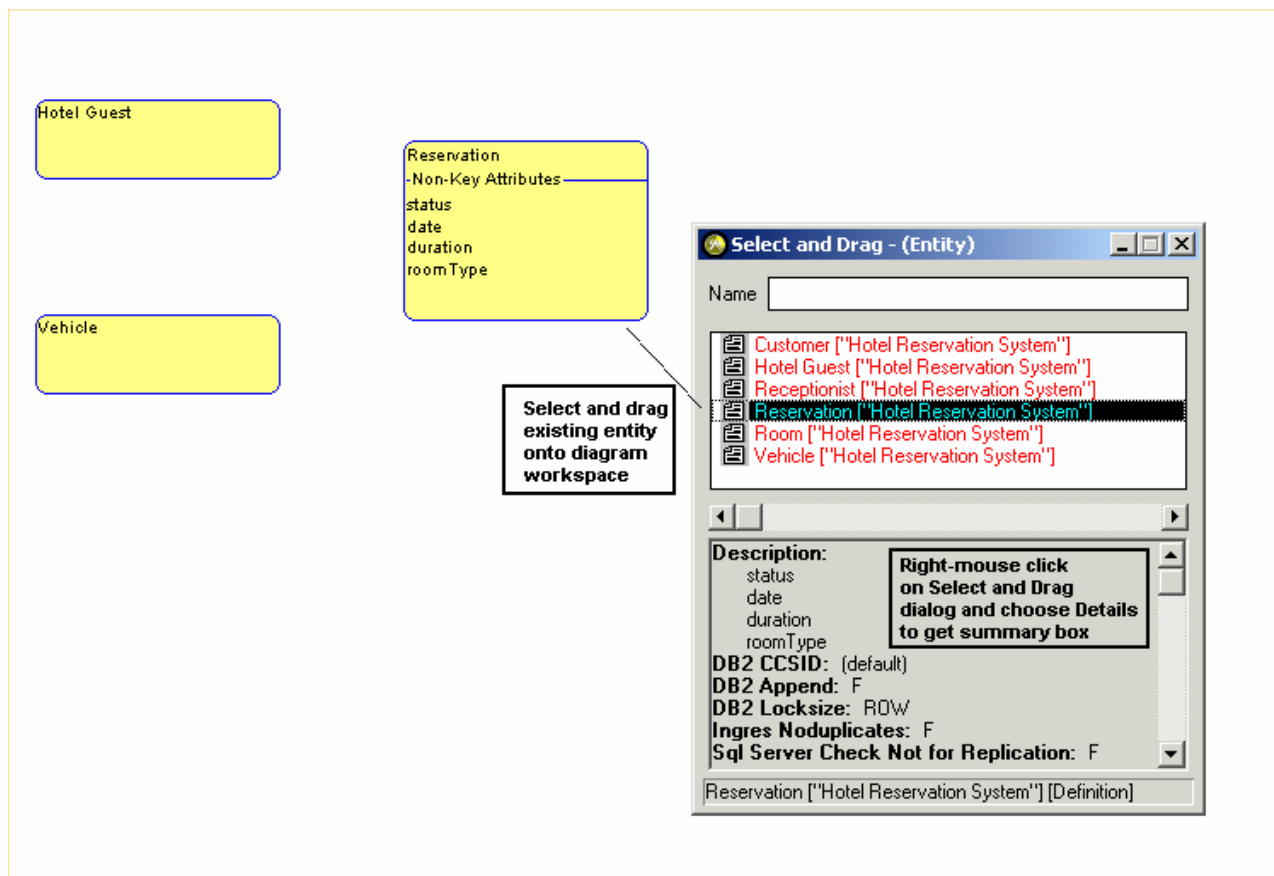


3. You should still be in draw mode - your cursor will look like a pen when you position it over the diagram workspace. Use the same procedure described in steps 1 and 2 to add another new entity, this one called **Vehicle**. Position it below **Hotel Guest**.
4. To turn the drawing cursor off, click the cursor arrow on the **Draw** toolbar or hit **Escape**.

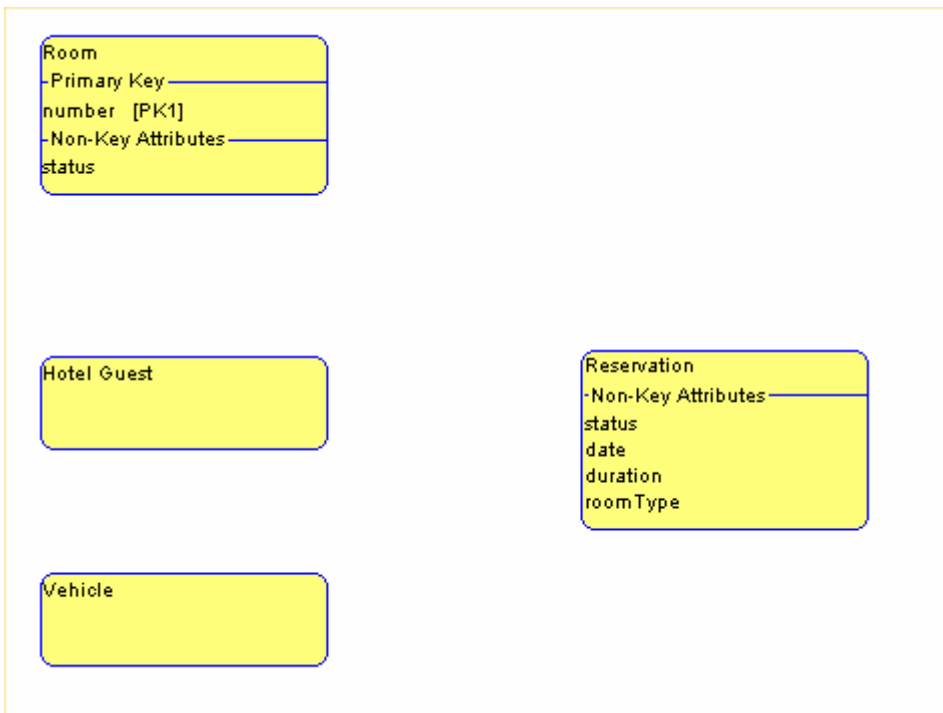
Reusing an Entity from the Project Encyclopedia

Let's reuse the Reservation entity, already established in our project encyclopedia.

1. Right-mouse click on the diagram workspace and select **Choices** from the drop-down list. You will get a list of all existing definitions of types that can be drawn on this diagram. In this case, you get a list of all entities already defined in this project.
2. Select **Reservation** and drag and drop it onto the diagram workspace, to the right of **Hotel Guest**. Notice that the entity is drawn complete with all of its already defined attributes.



7. Also select **Room** from the **Select and Drag** dialog, and drag it onto the diagram workspace above **Hotel Guest**. Notice that it too is drawn complete with all of its already defined attributes.
8. Close the **Select and Drag** dialog. Your diagram should look like the one below.



9. Defining Entities

System Architect provides an underlying data dictionary of data elements and data structures that can be used in various definition types throughout the tool (entities, tables, UML classes, business processes, etc). A data element is an elemental piece of data. It can have a data type, and various other properties. A data structure is a group of data elements. For example, you may have data elements such as Name, Street Address, City, State, and Zip Code. You might optionally combine all of these into a data structure called Address. Within an entity, an attribute is considered an instance of a data element or a data structure.

Let's define attributes for the Hotel Guest entity.

Open the **Hotel Guest** entity's definition dialog by double-clicking on the entity symbol

or

right-mouse click on the entity and choose **Edit** from the drop-down menu,

or

select the symbol and choose **Edit > Entity** from the toolbar.

This will open the **Model Object - Entity** dialog.

Note: If you want the entity definition dialog to appear automatically each time you place an entity symbol on your diagram, from the **Tools** menu, select **Preferences** and check **Auto > Define**.

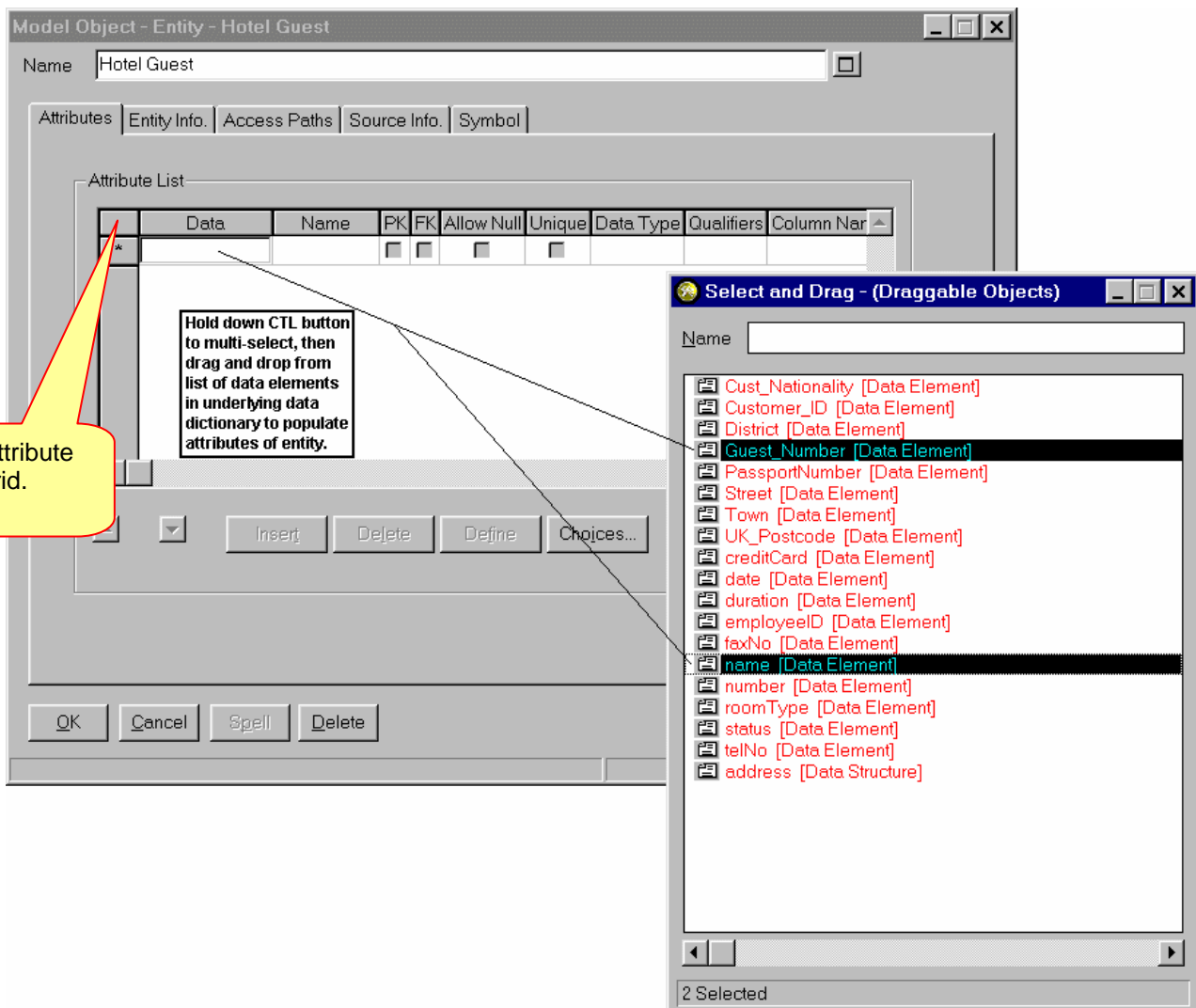
Using the Underlying Data Dictionary

Let us first examine the data elements already residing in our underlying data dictionary for this project encyclopedia, to see if we want to reuse any for this entity.

1. Click on the **Choices** button in the bottom right-hand corner of the **Attribute** list box of the **Attribute** tab to open a list of all data elements and data structures already defined in the underlying data dictionary.

Creating Data Elements: System Architect allows you to define data elements and data structures before you use them in entities. In this way you establish an underlying data dictionary that you can continue to build on as you define data in or outside of entities. To create a new data element or structure by itself, in the Browser select **Shared > Definitions > Data Element**. You may also create a new data element definition in the browser in the All Methods tab: expand the Definitions type, right-mouse click on Data Element and select New.

2. Hold down your **CTRL** key and multiple-select the *data elements* **Guest_Number** and **name**. Hold down your left mouse button and drag them into the **Data** column of the **Attributes List** grid (aim for the first cell). Two new attributes for this entity are created.



4. By default, System Architect names the *attributes* **Guest_Number** and **name**. You can change these names without affecting the names of the underlying data elements that the attributes report to.
5. Close the **Select and Drag** dialog.

As mentioned at the start of this section, the attribute is considered the *instance* of the data element within this entity. It carries instance information, such as whether or not this data is a primary key for this entity, whether or not it is a foreign key, whether or not it can be null, etc. The data element carries the overall information on this piece of data - its data type, qualifiers, default value, etc.

- Place your cursor in the **Name** cell of **Guest_Number** to highlight the whole attribute name. Type over this name to change it to **Guest#**. Notice that the data element name (in the cell to its left) does not change. **If you accidentally double click in this cell you will bring up the "Dictionary Object – Attribute – Guest" dialog box simply click on the Cancel button to close it if this happens.**

	Data	Name
1	Guest_Number	Guest#
2	Name	Name

- The name of the column which will be created from this attribute is the name of the data element with spaces and other special characters replaced by under-scores (_).
- ~~Close the definition dialogue by clicking on **Cancel**.~~ **You need to keep this dialog open for the next stage**

10. Specifying Primary Keys

The primary key of an entity is the attribute, or attributes, which uniquely identifies an instance of the record represented by the entity. To specify **Guest#** as the primary key attribute:

- Toggle on the **PK** cell adjacent to **Guest#**. This makes this attribute the primary key for this entity.
- Place your cursor back in the cell for the attribute **Guest#** and click on the **Define** button at the bottom of the **Attribute List** box. You will open the full definition dialog for this attribute. Notice that in the **Attribute Designation** group, the property **Primary Key** is toggled on. Click **OK** to close the dialog.
- Place your cursor in the **Guest_Number** cell of the **Data** column and click on the same **Define** button at the bottom of the **Attributes List** grid. You will open the full definition of the underlying data element. Notice that the properties and their values in this dialog are identical to those in the previous dialog for the instantiating attribute, except that this dialog does not contain the **Attribute Designation** group.
- Close the data element dialog.

Dictionary Object - Attribute - Guest#

Name: Guest#

General Properties | Source Info. | Access Data

Attribute Designation: PK FK Allow Null Unique

Data Designation: Data Type: character, Qualifiers: 10

Column Name: Guest_Number

Default Value:

Domain:

Effects of Changes to Data Elements and Attributes: If you change any values in the data element dialog, they are applied globally across the data model - they change for the data element, and all attributes that use the data element in all entities across the model. The same is true if you change a value in the attribute definition dialog - it will change the corresponding value in the underlying data element, and be applied globally across the model.

Adding a Data Structure as an Attribute

5. Click on the **Choices** button in the **Attributes List** grid again.
6. Right-mouse click anywhere within the **Select and Drag** dialog. This will bring up a popup menu which contains a choice called 'Details' **Select this option** and notice how the **Select and Drag** dialog windows appearance changes.
7. Select the data structure **Address**. Notice that in the **Details** window, this data structure is shown to include the data elements **Street, Town, and District**. Drag and drop it into the **Data** column of the **Attributes List** grid (aim for the next available cell). A new attribute for this entity is created.

Note on Expanding Data Structures: At this point you could choose to expand this data structure into its constituent data elements in the Attribute List grid. We will not do it here. When we map this logical data model to a physical model, and this logical entity and its attributes are mapped to a table with columns, this data structure will be automatically decomposed into an equivalent column for each of its data elements.

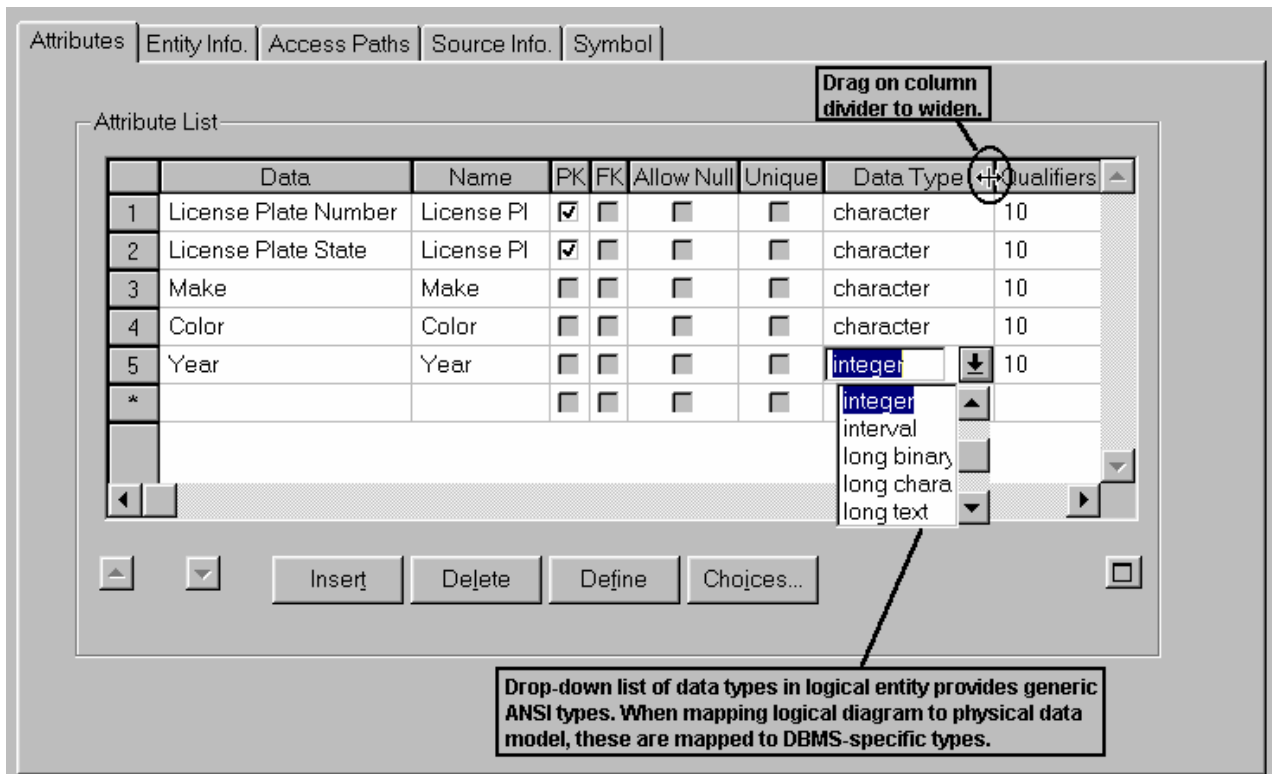
If you did wish to expand this data structure now, you would right-mouse click on address in the **Data** column cell, and select **Expand** from the drop-down list. After expanding it, a column (titled **Data Structure**) in the **Attribute List** grid (located to the far right) would detail the data structure that each resulting element was expanded from.

8. Click **OK** to close the entity definition dialog. **And this also saves your changes**

11. Creating New Data Elements in an Entity

Now let's add some new attributes to an entity, thereby also creating new data elements in our data dictionary. We will add attributes to the **Vehicle** entity.

1. Open the definition dialog for the **Vehicle** entity (double-click on the symbol or right-mouse click and select **Edit**).
2. On the **Attributes** tab, within the **Attribute List** grid, place your cursor in the first **Data** cell and enter the name for a new data element, **License_Plate_Number** and hit your **Enter** key. Notice that in the grid, default values for the attribute's **Data Type** and **Qualifiers** are entered. The default value for **Data Type** is **Character**. The default value for **Qualifiers** is **10**. What this means is that in the eventual database that will be built from this design, there will provisions for storing license plate numbers that can contain a combination of up to 10 characters. ("Character" by definition almost always means alphanumeric.)
3. Toggle on the **PK** cell for **License_Plate_Number** to make this the primary key for the **Vehicle** entity.
4. Enter another attribute called **License_Plate_State** and hit **Return**. Leave its default values for **Type** and **Qualifiers**. Toggle on the PK cell for this attribute as well - we now have a two-part primary key for this entity.
5. Enter another attribute named **Make**, and hit **Return**. Again leave the default values for **Data Type** and **Qualifiers**.
6. Enter another attribute named **Color** and hit **Return**. Leave the default values for **Data Type** and **Qualifiers**.
7. Enter one more attribute, named **Year**, and hit **Return**. This time we'll change the default values of **Data Type** and **Qualifiers**.
8. Place your cursor in the **Data Type** field for the **Year** attribute so that a drop-down arrow appears. Click on the arrow to display a list of available data types. Notice that these are generic ANSI data types - they are not DBMS specific. That is because we are in a logical entity. When we map this diagram to a physical data model, System Architect will map generic data types to DBMS-specific data types, depending on the DBMS we choose to model in the physical data model.



9. In the drop-down list of data types, find and select **Integer**.
10. Put your cursor in the **Qualifiers** cell for the **Year** attribute, and delete the default value of 10 leaving the cell blank.
11. Click **OK** to enter the attributes onto the entity symbol.

Column Name

As we model in this logical data model, System Architect is already preparing for mapping to a physical data model. If you use the bottom scroll bar in the Attribute Grid, and scroll to the right, you will notice a column called **Column Name**. Notice that the name of the data element is automatically mapped to a default column name - this is the name of the column that will be created for a corresponding table of this entity, when we map this diagram to a physical model. You may manually change the name of the column right here in the attribute grid.

You may specify the way that this mapping is done by selecting **Dictionary**, **Options**, **Column Name Options**. Please refer to the on-line help for details.

12. Relationships Between Entities

There are four types of relationship line types available in the logical data model for establishing associations between entities. The relationship may be specified as parent/child, super-sub type, or non-specific. The nature of the parent/child relationship can be further categorized as identifying or non-identifying.

Non-Specific Relationship

The non-specific relationship line is used during conceptual modeling, when you just want to relate two entities, but don't want to spend time figuring out the exact details of the relationship.

Super-Sub Relationship

The super-sub relationship line is used to model inheritance in the logical data model. For example, similar to what we covered in the UML Class diagram, you might want to model a Customer entity that has sub-entities Business Traveler and Vacationer, which inherit attributes from Customer but also have specific attributes of their own.

Identifying Relationship

An identifying relation indicates that the parent entity identifies the child entity. This means that the primary key of the parent entity is part of the primary key of the child entity. Since primary key components cannot be null the identifying relation is, by definition, a mandatory relationship; therefore, the parent cannot be specified as optional. The foreign keys that are propagated through an identifying relation line in the entity relation model are always classified as primary key components in the child entity.

Non-Identifying Relationship

A non-identifying relation indicates that the child entity does not rely on the parent entity for its identification. The child entity must, therefore, possess its own unique primary key (which may include foreign key components from other relations which are identifying). The parent can be specified as mandatory or optional when the relation is nonidentifying.

The foreign keys that are propagated through a non-identifying relation line in the entity relation model are generally classified as non-key attributes in the child entity. If the foreign key component is unified with a foreign key component of another relation, and the other relation line is specified as identifying, the identifying relation will take precedence and the resulting foreign key will be a primary key component.

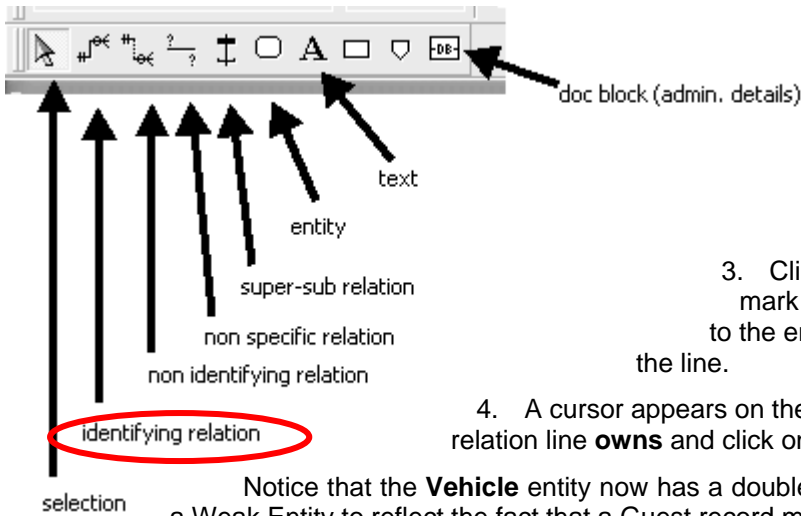
Names of Relationships

System Architect does not require names on relation lines, but it is good practice to name the relation line to indicate the relationship being represented. The relationship can generally be read as a sentence:

- The Hotel Guest owns a Vehicle.
- A Room belongs to a Hotel Guest.
- A Reservation belongs to a Hotel Guest.

13. Drawing Relationship Lines

Let's draw a relationship line between **Hotel Guest** and **Vehicle**:



1. From the **Draw** toolbar, choose the **Identifying Relation** line.

2. Click and release your left mouse button over the **Hotel Guest** entity. A + mark appears, indicating that a line is attached to the entity.

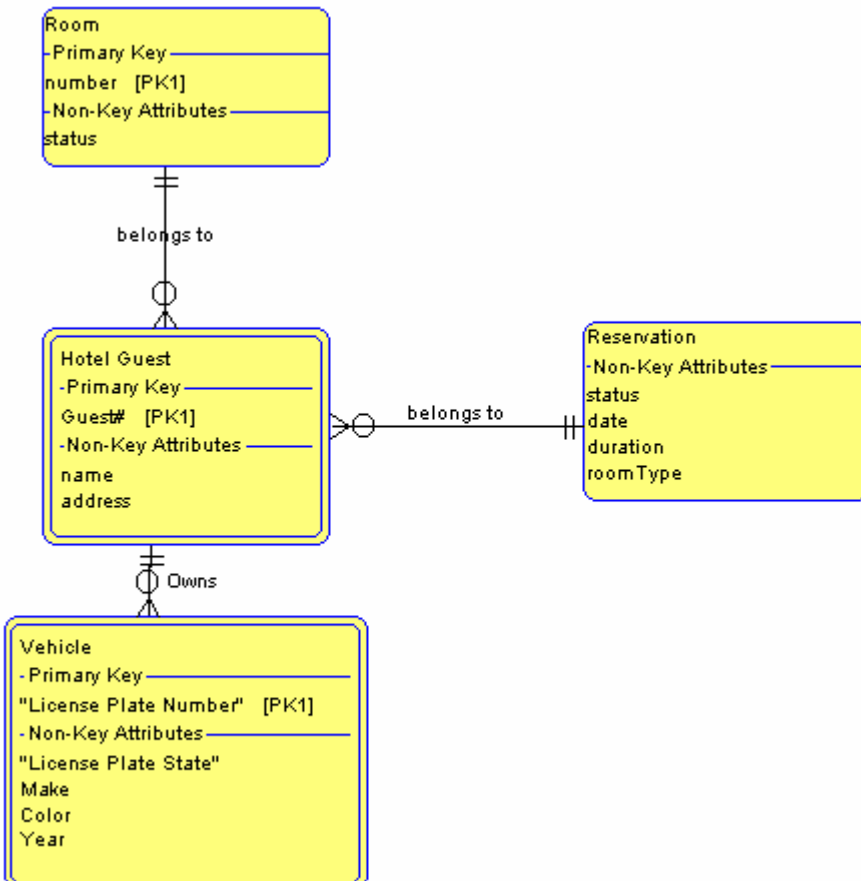
3. Click your mouse on the **Vehicle** entity. A + mark appears, indicating that the line is attached to the entity. Release the mouse button to attach the line.

4. A cursor appears on the relation line, specifying a name. Name the relation line **owns** and click on your left mouse button.

Notice that the **Vehicle** entity now has a double line around it - it has been transformed to a Weak Entity to reflect the fact that a Guest record must exist for a Vehicle to exist.

If you select **Tools > Preferences**, and toggle on the **Auto-Define** choice, the **Relationship Editor** will appear automatically each time you place a relation line on a diagram. By default, this choice is toggled off -- in this exercise we will first draw the relationship lines between entities, and then go back and define them.

5. You should still be in draw mode -- your cursor should still resemble a pen. Using the procedure for drawing lines described in steps 2 and 3 above, draw a relationship between **Room** and **Hotel Guest**. Name the line **belongs to**.



Notice that the **Hotel Guest** entity now has a double line around it - it has been transformed to a Weak Entity to reflect the fact that a **Room** must exist for a **Hotel Guest** to exist.

6. You should still be in draw mode. Using the procedure for drawing lines described in steps 2 and 3 above, draw a line between **Reservation** and **Hotel Guest** and name it **belongs to**.

14. Relationship Line Types

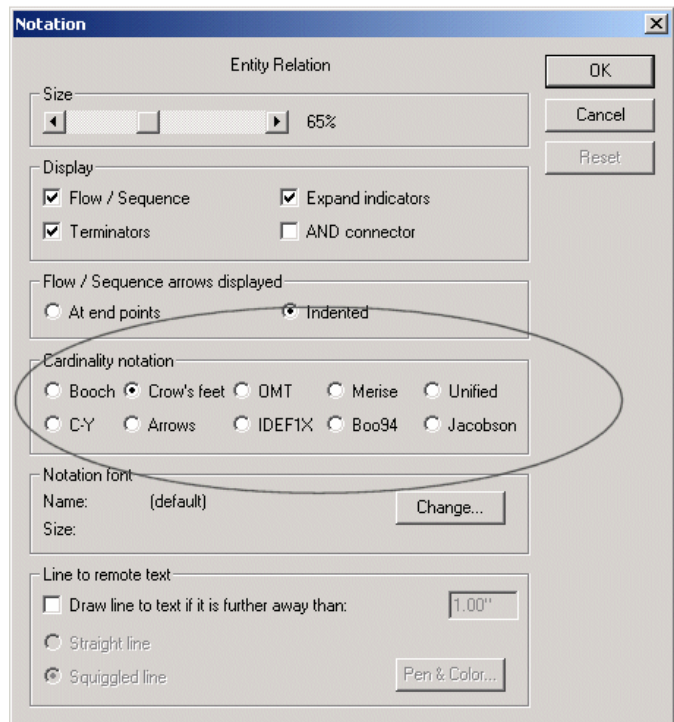
System Architect supports a number of types of relationship lines. To see the variety, select from the menu bar **Format, Diagram Format, Notation**.

The most widely-used notations in data modeling are Crow's feet and IDEF1X. Whichever notation you choose will be used for all relation lines in all ERDs in the encyclopedia. All the examples in this tutorial use Crow's feet.

15. Foreign Keys

In System Architect, you have complete control over when foreign keys will be propagated from the parent to the child entity. At any time, without deleting relationship lines, you can remove foreign keys.

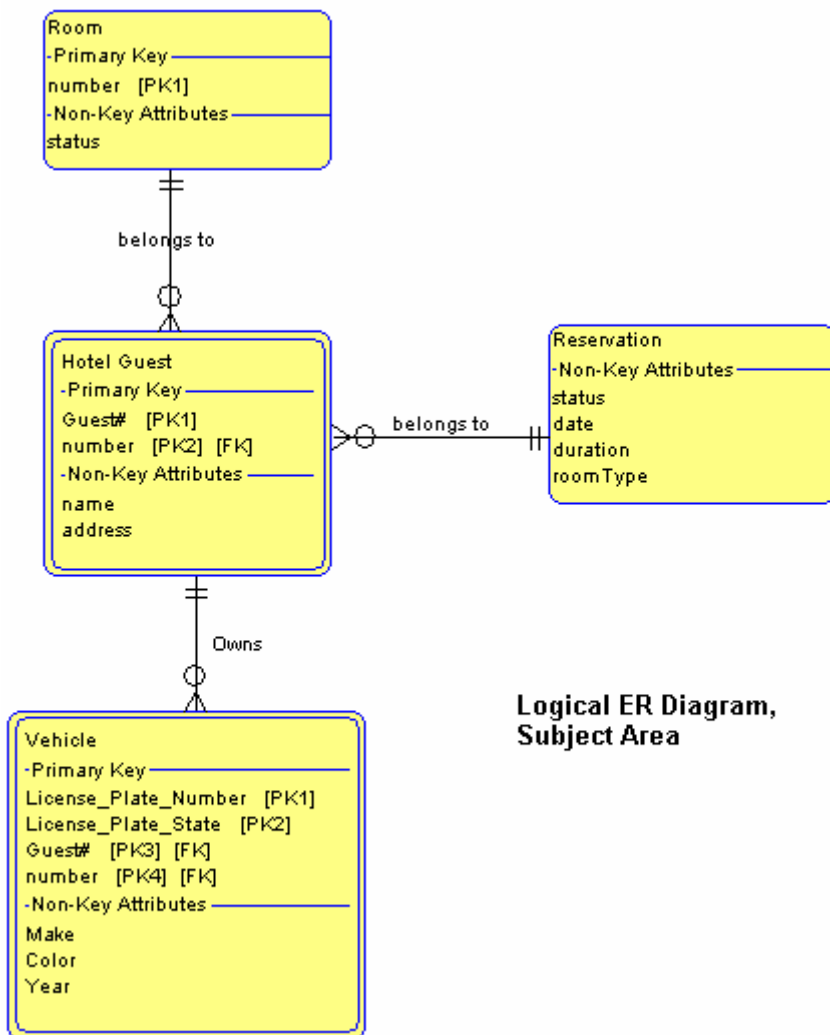
1. Select **Dictionary, Update FKs**. System Architect will examine the data model, and establish foreign keys in child entities based on the relationship types between entities. This process is also referred to as "migrating foreign keys."



Once you have done this you need to close the details box that appears telling you what SA/SE has done.

2. Select **Dictionary > Remove FKs**. System Architect will examine the data model, and remove all foreign keys in child entities based on the relationship types between entities. Foreign keys that are primary in the child entity will be change from PK/FK back to PK. Once you have done this you need to close the details box that appears telling you what SA/SE has done.
3. Select **Dictionary > Update FKs** again.

After you propagate the foreign keys, the logical data model should look like the one pictured below.



Logical ER Diagram, Subject Area

Migrating Keys

4. Open the definition dialog for the **Owns** identifying relationship line between **Hotel Guest** and **Vehicle** (double-click on it or right-mouse click on it and select **Edit**).

In the definition dialog that opens, notice that on the **Parent Entity** side of the relationship, the **Parent identifies child** box is toggled on. This is an identifying relationship. The primary keys of the parent entity (**Guest# and number**) are migrated through this relationship into the child entity. They will be

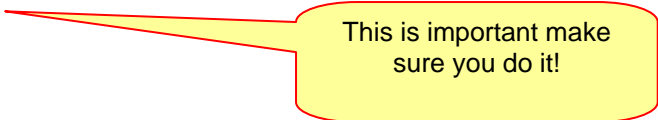
'foreign' keys in the child entity **Vehicle**. The primary key **number** was migrated into **Hotel Guest** from its relationship to **Room**.

5. Click **OK** to close the **Owns** relationship definition dialog.

Display Options

System Architect provides you with a number of display options for the logical ER diagram. Let's examine them.

1. Right-mouse click on the entity **Hotel Guest** and select **Display Mode** from the drop-down list. You are presented with a dialog that gives you a number of choices for display.
2. Within the **Display View** group, select **Key Based** and click **OK**. Notice that only the primary and foreign keys are displayed on the entity symbols.
3. Right-mouse click on an entity (any entity) again and choose **Display Mode**. Change the display mode to **Fully Attributed** and click **OK**.
4. Select **File, Save Diagram** to save the diagram.
5. Select **File, Close Diagram** to close the diagram.



This is important make sure you do it!

16. Creating a Second Subject Area Diagram

The steps on this page are optional if you have already created a second subject area diagram by performing the steps of the section "[Mapping a Class Diagram to ER Model](#)." If you have not worked through [that](#) chapter, then [carry on](#). (Otherwise, jump to the next section, [Creating a Model Diagram](#).)

Create the Subject Area Diagram

1. Select the **Data Modeling** tab in the browser.
2. From the **File** menu, select **New Diagram**. A dialog will open presenting you with a list of possible diagram types. Double-click on **Entity Relation**.
3. In the **New Diagram** dialog, enter the name of the diagram to be created, **Reservation System**, and click **OK**. The **Diagram-Entity Relation** dialog opens. In this dialog we will specify whether this is going to be a model diagram or a subject area of a model. In either case, we must specify the model that we are representing.
4. For the **Diagram Status** property, toggle on **Subject Area** (it should be toggled on already since this is the default).
5. For the **Model** property, click on the **Choices** button. The **Select and Drag** dialog will open, presenting a list of models already created in this project encyclopedia. You should see the model **Hotel Reservation System**.
6. Select and drag **Hotel Reservation System** into the **Model** property box and close the **Select and Drag** dialog (by clicking on the **X** in its upper right-hand corner). (If no model existed at this point, you could type a new model name in the box and click **Define** to create it.)
7. Leave **DBMS** at its default value, **ANSI**. (**Note:** the selections presented in this list are dependent on the **Database Properties** we have selected for you in the **Customize Method Support** dialog. To add different **DBMS**'s to this list, you would select **Tools > Customize Method Support** and reopen the encyclopedia.) **The Student Version of SA doesn't allow you to make these changes. You may only do so with the commercial release of SA.**
8. Click **OK** to close the dialog and create the **Entity Relation** diagram.

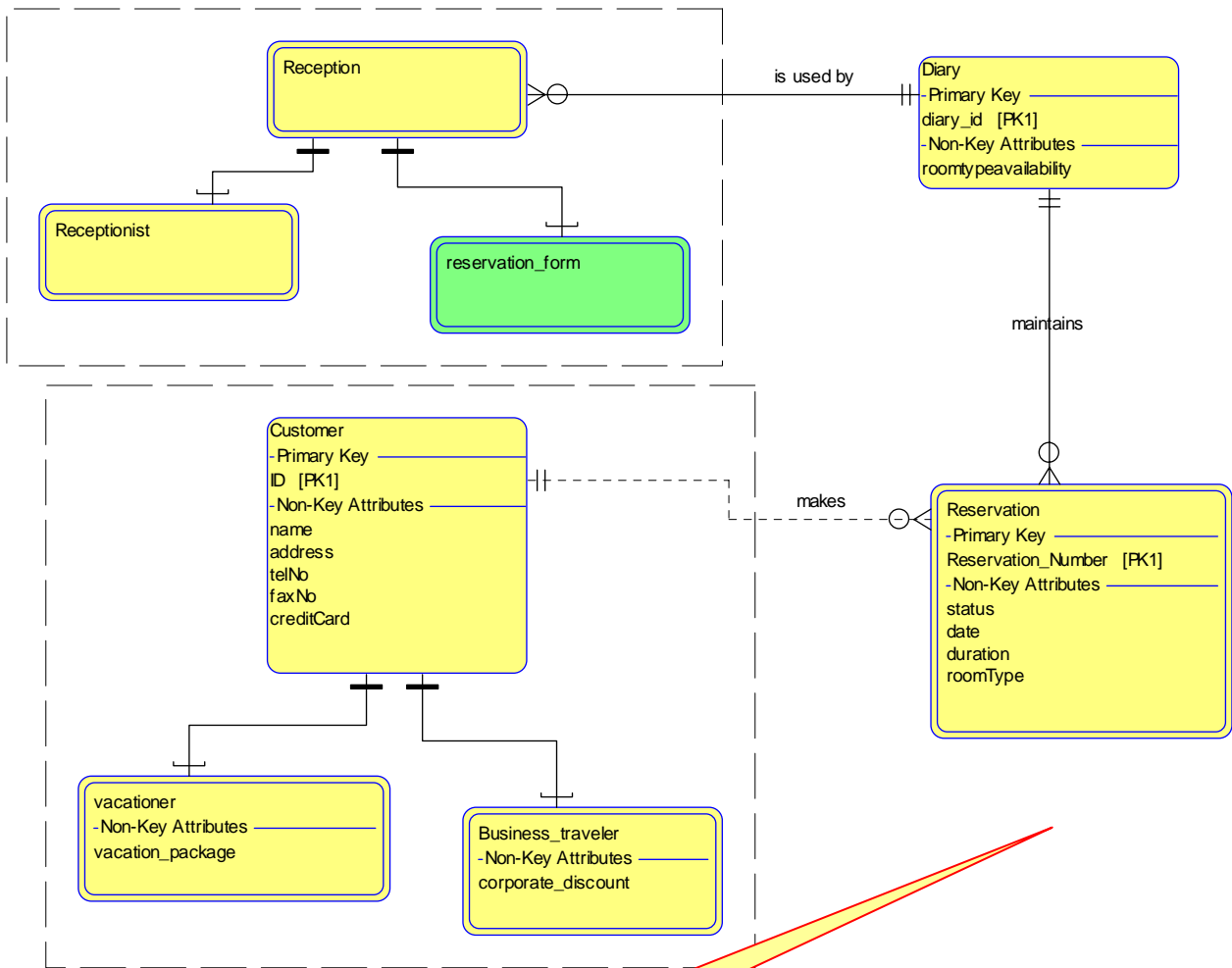
17. Drawing the Diagram

1. Right-mouse click on the diagram workspace and select **Choices**. Drag and drop onto the diagram workspace the entities **Reservation** and **Customer**.
2. Create new entities **Reception**, **Receptionist**, **Reservation_Form**, **Diary**, **Business Traveler**, and **Vacationer**, referring to the diagram below for placement of the entities.
3. Select the **Super-sub Relation** line, and draw a line **from Customer to Business Traveler**, and another line **from Customer to Vacationer**.
4. Select the **Super-sub Relation** line, and draw a line **from Reception to Receptionist**, and another line **from Reception to Reservation_Form**.
5. Select the **Non-Identifying Relation** line, and draw a line **from Customer to Reservation**. Name the line **makes**.
6. Select the **Identifying Relation** line, and draw a line **from Reservation to Diary**. Name the line **maintains**. and draw a line **from Diary to Reservation** Name the line **maintains**.
7. Select the **Identifying Relation** line, and draw a line **from Diary to Reception**. Name the line **is used by**.
8. Add the following attributes to the following entities (typing them into the attribute grid and pressing **Enter** after each one):

Entity	Add the Following Attribute(s)
Diary	Diary_ID (and toggle on PK) Room_Type_Availability
Reservation	Reservation_Number (and toggle on PK)
Business Traveler	Corporate_Discount
Vacationer	Vacation_Package_Discount
Receptionist	employeeID (and toggle on PK)

Foreign keys in a subject area diagram reflect the entity's position in the model diagram. You will notice that even though we have not propagated foreign keys for this subject model, the Reservation entity reflects the foreign keys from its relationship in the existing subject area model, Hotel Check In

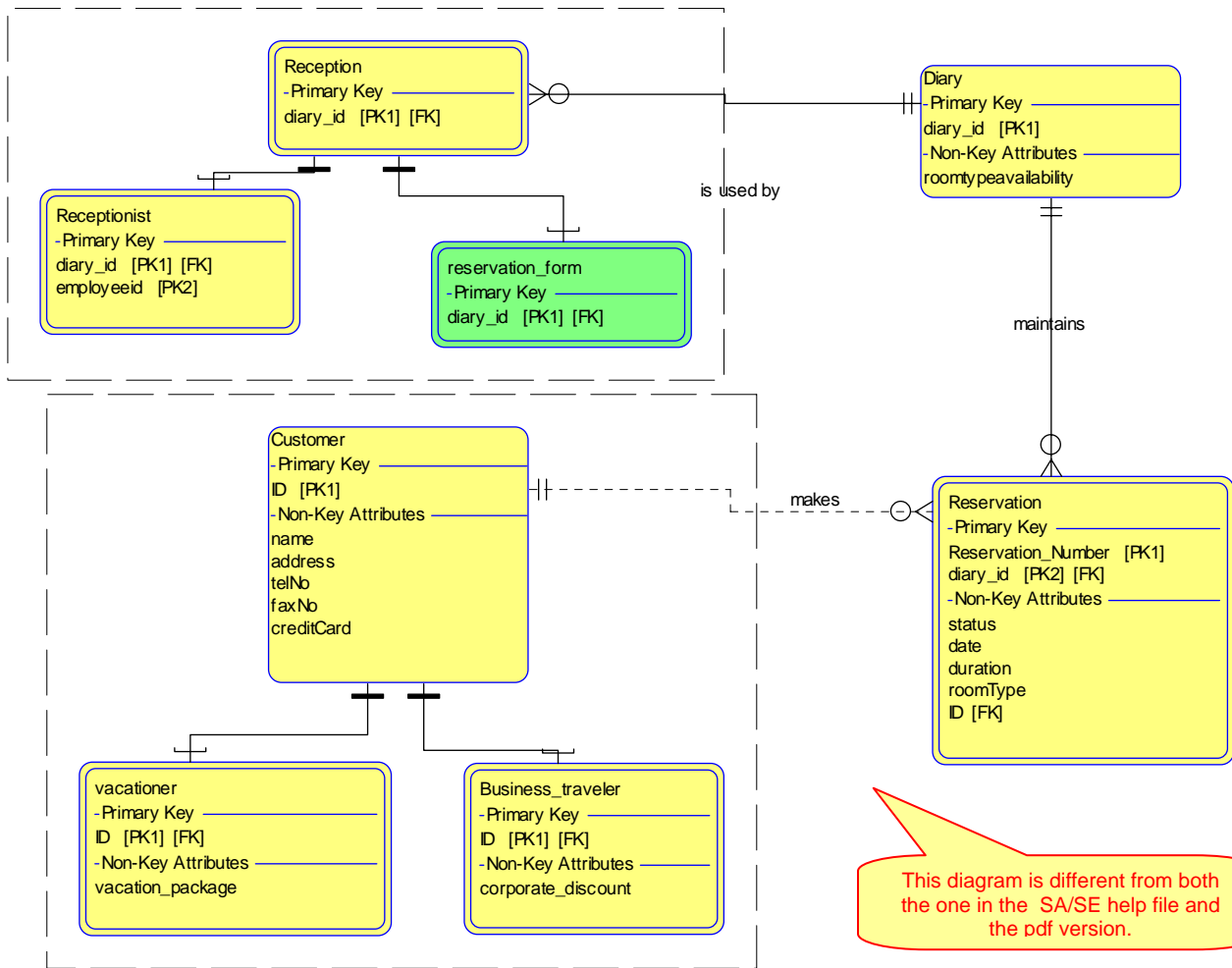
Your diagram should look like the one **below**



This diagram is different from both the one in the SA/SE help file and the pdf version.

18. Update Foreign Keys

1. Select **Dictionary, Update FKs**. System Architect will examine the data model, and establish foreign keys in child entities based on the relationship types between entities. This is also referred to as migrating foreign keys.



2. Save the diagram by selecting **File, Save Diagram**.

The completion time of this section is approximately 10 minutes.

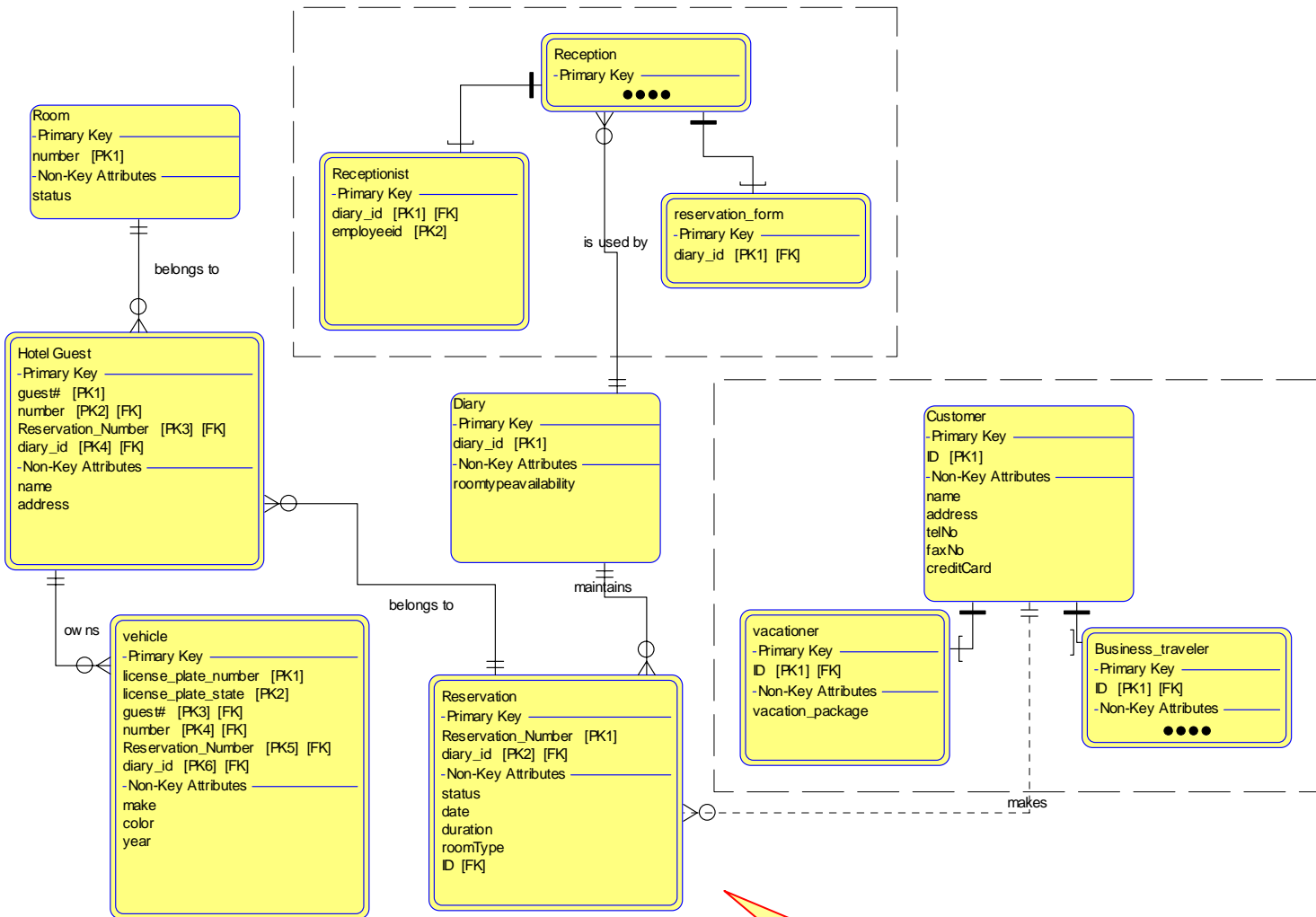
19. Creating a Model Diagram

We have thus far created two subject area diagrams for the **Hotel Reservation System Model**. Now let's bring them together into one model diagram. The model definition **Hotel Reservation System** already exists, but we have not yet created a diagram to represent it.

A model diagram may not automatically display objects you have recently drawn on one of the subject area diagrams related to it. From the **Dictionary** menu select **Refresh**. All entities and relationships on all related subject area models will be added to the model.

1. Select the **Data Modeling** tab in the browser.
2. From the **File** menu, select **New Diagram**. A dialog will open presenting you with a list of possible diagram types. Double-click on **Entity Relation**.
3. In the **New Diagram** dialog, enter the name of the diagram to be created, **Hotel Reservation System**, and click **OK**. The **Diagram-Entity Relation** dialog opens.

- For the **Diagram Status** property, toggle on **Model**.
- For the **Model** property, click on the **Choices** button. The **Select and Drag** dialog will open, presenting a list of models already created in this project encyclopedia. Select and drag **Hotel Reservation System** into the **Model** property box and close the **Select and Drag** dialog (by clicking on the **X** in its upper right-hand corner). Leave **DBMS** at its default value, **ANSI**.
- Click **OK** to close the dialog and create the Entity Relation diagram. The diagram will be created and all entities and relationships from affiliated subject area diagrams will be automatically drawn on the model. Note that there are entities and relationships on the diagram that you did not include on either of your subject area diagrams. These entities and relationships were already in the model of this encyclopedia.



Do not worry too much if yours does not look exactly like this one – I have moved the entities and relation lines around to make it much clearer.

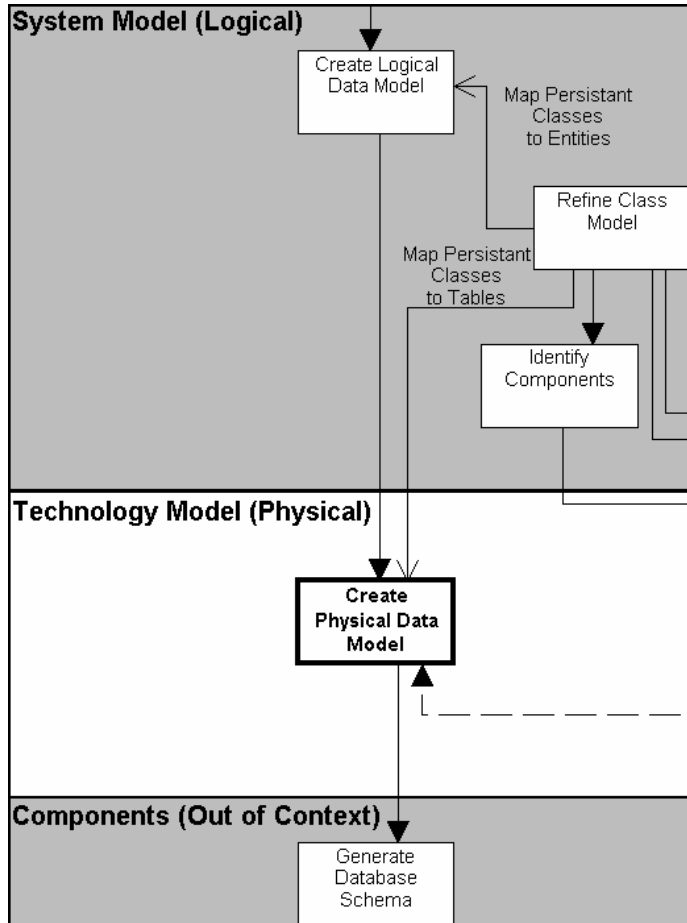
Again this is not the same as the one in the SA/SE help file but more similar to the one in the PDF SA/SE tutorial file. (p212)

The following pages up to the end of this document provide information about how the model is converted to an actual database and or program – I do not expect you to know this in detail only be aware of the process. RB

20. The Physical Data Model

There are two main reasons for separating logical and physical database design:

- To enable modeling of normalized and denormalized designs.
- To enable modeling of distributed databases.



Database Denormalization

When designing a database system, it is often advantageous to adhere to the rules of data normalization during the logical design stage. This assures that data redundancy is minimized. More often than not, however, the implemented database is not normalized for a number of reasons; for example, a denormalized database may make data retrieval faster.

Only if you separate the logical and physical design stages, can the logical design be normalized while the physical data model is denormalized.

Database Distribution

Distributed systems are logically one database system, but physically multiple database systems. Separation of logical and physical models allows distributed databases to be effectively modeled and reported on.

In System Architect, you can create a Project Data Model comprised of multiple subject area ER diagrams, each of which represents a different database in the distributed system. Then you can generate a physical model from each entity relation subject area diagram.

In this Section

In this section you will automatically build a physical diagram from the logical model built in the previous chapter. In terms of the tutorial scenario map that we are following, we will perform the process indicated in the white section below:

The completion time of this section is approximately 10 minutes.

21. Generating a Physical Data Model from an ER Diagram

To create a physical representation of one of the logical diagrams created in the previous chapter, perform the following steps:

1. Open the logical model diagram **Hotel Check-In**. If it is not open, in the **Data Modeling** tab of the browser, expand **Models >Hotel Reservation System >Diagrams >Entity Relation**. Double-click on **Hotel Check-In**, or drag it onto diagram workspace.
2. Select **Dictionary >Create Data Model >Physical Data Model**. The **Create Physical Data Model Diagram** dialog opens.

Super/Sub Resolution Method

Within this dialog, you may specify how you want to map super-sub relationships in the logical ER diagram to physical tables -- the concept of super-sub relationships and the inheritance from the super to the sub does not exist in physical modeling.

The choices are:

Separate Tables: creates a separate physical table for every entity in a super-sub grouping. Attributes for each entity are mapped to columns in each respective table.

Merge Supertype to Subtype: creates tables for all sub-entities, does not create a table for the super-entity. Each table created for a sub-entity contains columns mapped from the attributes of the corresponding sub-entity, plus attributes 'handed down' from the super-entity.

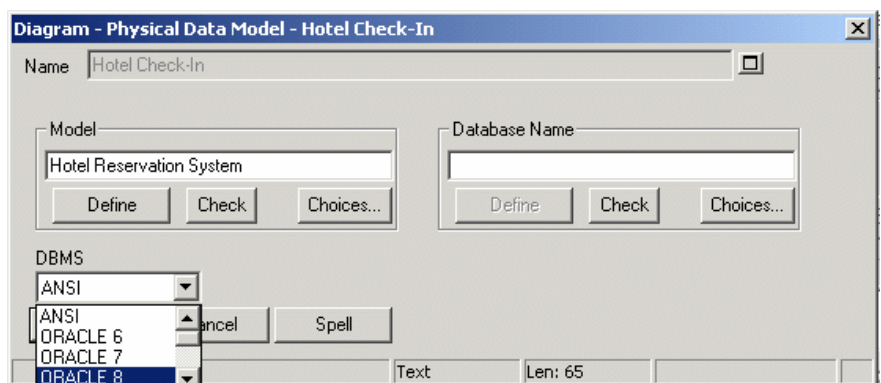
Merge Subtype to Supertype: creates one table for the whole super-sub entity structure. Attributes from all super- and sub-entities are mapped to columns in the one table created.

Prompt for Each Super/Sub Group: System Architect will prompt you with a dialog enabling you to choose the mapping for each Super/Sub group as the transformation of logical to physical diagram takes place.

Name Mapping

In the Create Physical Data Model Diagram dialog you globally choose how the case (upper, lower, or retain) of names of Tables, Columns and Constraints are mapped from the names of their corresponding Entities, Attributes, Relationships and Indexes.

3. Leave the default choice for all choices on the **Create Physical Data Model Diagram** dialog (**Separate Tables** and **Retain Case**), and click **OK**.
4. System Architect will begin the mapping. A running log detailing the mapping will appear on the screen, under the title **Generate Physical Data Model**.
5. System Architect presents you with a dialog in which you must select the **Model**, **DBMS** and **Database Name** for the physical diagram. Leave **Model** at its default, **Hotel Reservation System**. Leave the **Database Name** empty. From the **DBMS** drop-down list choose **Oracle 8**. Click **OK**.



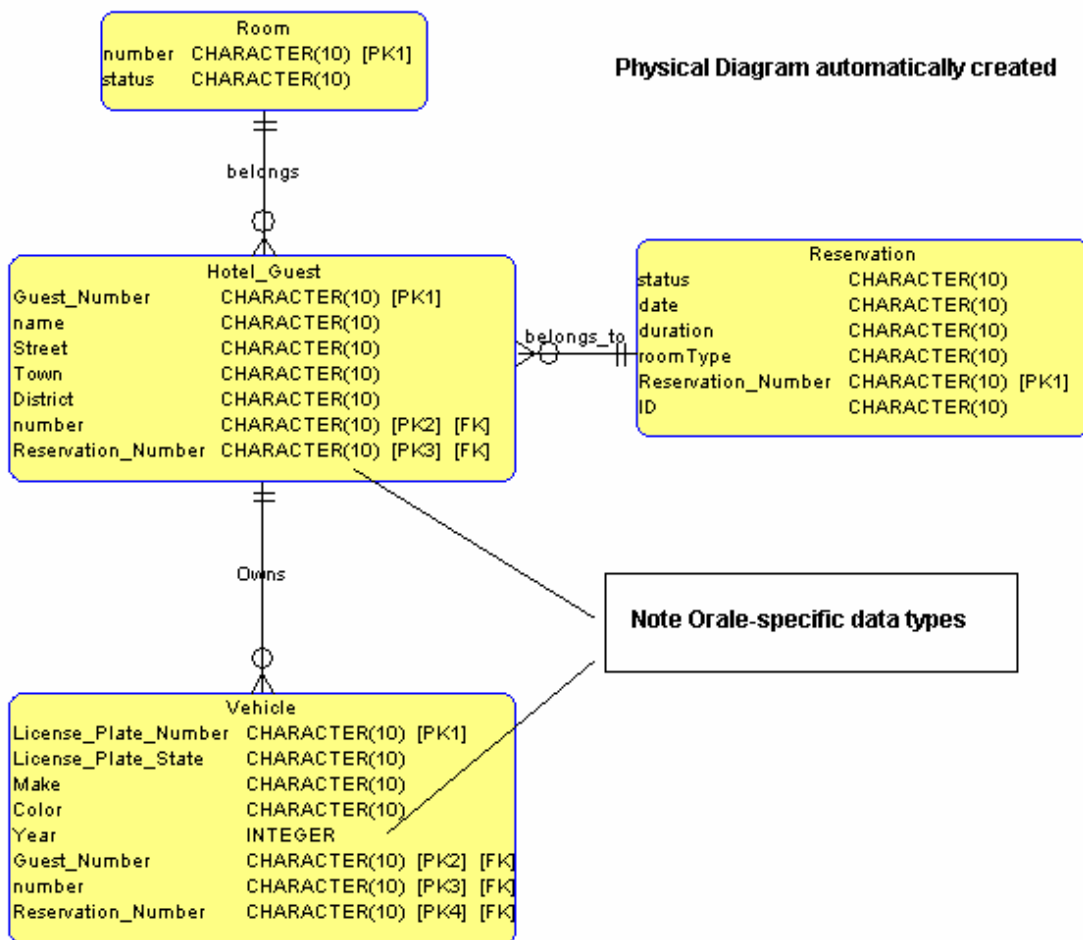
The number of DBMS's listed depends on the DBMS's chosen in the **System Architect Property Configuration** dialog. The list of DBMS's can be changed at any time during a project by selecting **Tools >Customize Method Support >Encyclopedia Configuration**, selecting **DBMS's** from the **Target Databases** list, and reopening your encyclopedia. *The Student Version doesn't allow you to make changes to the*

Encyclopedia Configuration. You may only do so with the Commercial release of System Architect.

- The physical diagram is created, and the **Generate Physical Data Model** running log dialog completes.

Note: at this writing, the shipping version of System Architect does *not* display a 'Diagram Created' message in the log dialog; when the diagram is completed, the log dialog simply shows a blinking cursor. In addition, the physical diagram created will be off to the right on the diagram workspace, and not necessarily be visible on the screen. To view the diagram, perform the step below.

- Close the running log by clicking **x** in its upper right-hand corner. Select **View, Used Area** to view the entire physical diagram.
- Select **File, Save Diagram** to save the diagram.



22. The Physical Table

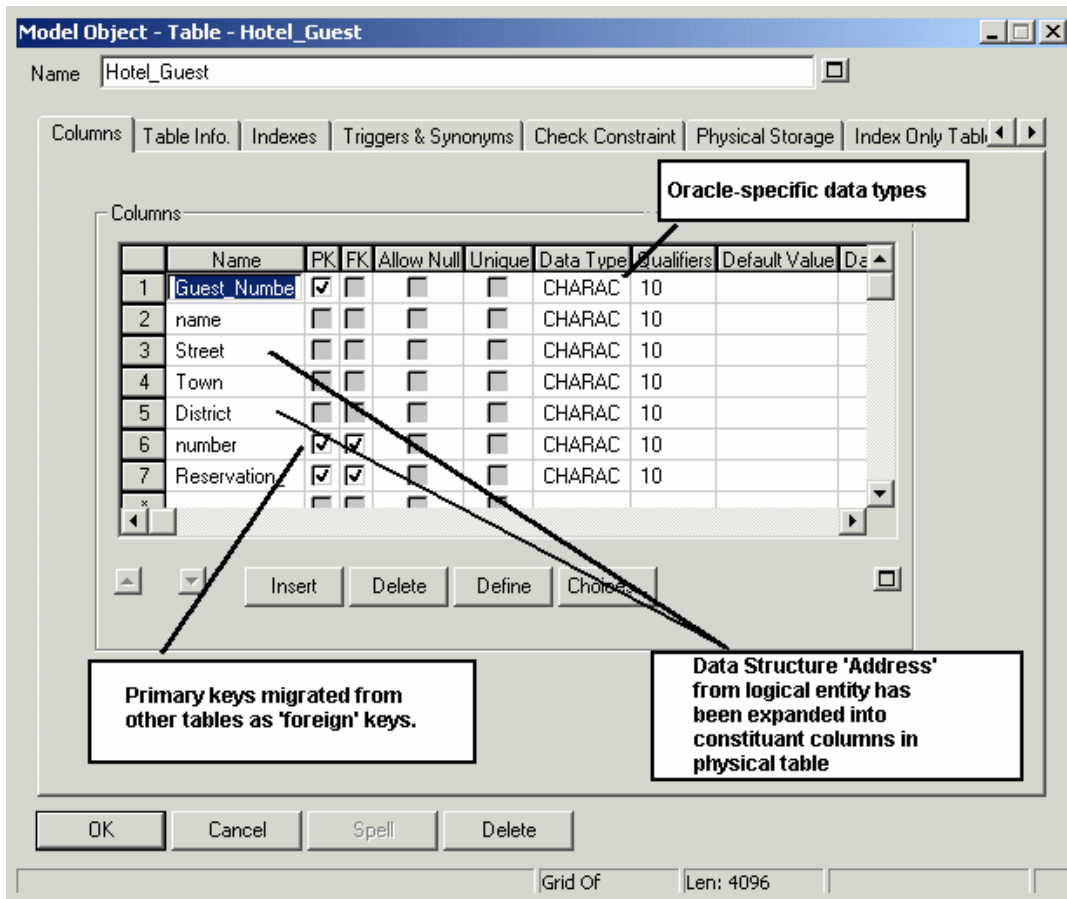
The table in the physical model represents the table that will be implemented in the database. It contains specification of the table's columns (not the rows, obviously, since the rows correlate to the actual data itself, which is stored in the database and has nothing to do with the design).

Let's examine a table in our physical diagram.

- Select the table **Hotel_Guest**. In the logical ER model there was an embedded space in its name - '**Hotel Guest**'. Since Oracle does not deal with embedded spaces in names of tables or columns,

System Architect has added an underscore to the name - you specified this in the **Create Physical Data Model Diagram** dialog by leaving the "_" for Special Characters choice toggled on for tables.

2. Select the table **Hotel_Guest**. Open its definition by double-clicking on it or by clicking on it with the right mouse button, and selecting **Edit**.
3. Notice in the **Column** grid of the table's definition dialog, that the columns have been mapped over from corresponding attributes in the corresponding entity of the logical data model. Notice also that the data types are all upper case - the generic data types in the logical model have been mapped to Oracle-specific types.



4. Notice that **Hotel Guest** has columns **Street**, **Town**, and **District**. If you remember in the logical model, we added the data structure **Address** to the logical entity for **Hotel Guest**. The data structure was composed of data elements **Street**, **Town**, and **District**. The data structure has now been expanded into relevant columns in the physical table.
5. Notice that this table contains a number of new primary keys - **number** - that have been migrated from other tables as foreign keys (both the **PK** and **FK** cells are checked for each).
6. Notice the column **Guest Number** - remember that in the logical entity, the name of the logical data element was **Guest Number** (with an embedded space), and the attribute name was **Guest#**. The data element has been mapped over.
7. Select **Guest Number** (put your cursor in its name box) and click the **Define** button at the bottom of the **Column** grid. You open the full definition dialog for this column.
8. Select the **Source** and **Key Info**. tab. In this tab, you can see the source attribute and data element for this column. It is read-only information. You may click the **Define** button to view full information on either source item.
9. Click **OK** or **Cancel** to close all dialogs.

23. Display Options for a Table

System Architect provides you with a number of display options for the physical diagram. Let's examine them.

1. Right-mouse click on the table **Hotel_Guest** and select **Display Mode** from the drop-down list. You are presented with a dialog that gives you a number of choices.
2. Select **Key** and click **OK**. Notice that only key columns are displayed on the table symbols on the physical diagram.
3. Right-mouse click on a table (any table) again and choose **Display Mode**. Change the display mode back to **Columns** and click **OK**.

The completion time of this section is approximately 2 minutes.

24. Constraints

A constraint in the physical data model is the equivalent of a relationship line in an ER diagram. The constraint line in the physical diagram is actually a visual representation of the foreign key constraints in the database represented by the diagram.

The constraint can be specified only as identifying or non-identifying. An identifying constraint is one in which an instance of the child table can only exist if there is a parent table. The primary keys in the parent table will be primary and foreign keys in the child table.

A non-identifying constraint indicates that an instance of the child table can exist even without a parent table. The foreign keys may be non-key columns in the child table.

1. Open the definition dialog for the **owns** constraint line between **Hotel_Guest** and **Vehicle** (double-click on it or right-mouse click on it and select **Edit**).
2. Notice that you can change whether this line is identifying or non-identifying (child optional) through the dialog.
3. Click **Cancel** or **OK** to close the dialog.

The completion time of this section is approximately 3 minutes.

25. Generating Schema

The Student Version of System Architect does not allow you to Generate Schema. You may only do this with System Architect's commercial version.

You can generate schema from a physical diagram for your implementation DBMS. System Architect provides a choice of generation to a DDL file, or directly to the DBMS through ODBC for most DBMS's.

Let's generate schema for Oracle 8 for our data model.

1. Open the physical model diagram **Hotel Check-In** (if it is not still open, in the **Data Modeling** tab of the browser, expand **Models, Hotel Reservation System, Diagrams, Physical Data Model**, and then double-click on **Hotel Check-In** (or drag it onto diagram workspace).
2. Select **Tools, DB Schema Generation**. The **System Architect Schema Generation** dialog opens.
3. In the **System Architect Schema Generation** dialog, click the **Select** menu - notice all tables are selected. Click **Cancel**.
4. In the **System Architect Schema Generation** dialog, use standard Windows techniques (hold down **SHIFT** key) to select all of the properties to be generated in the **Output(s)** list.
5. Select the **Generate** button. The Oracle 8 DDL will be generated and output to a file. The file will be presented within Notepad on the screen. You may open this file in Oracle 8 to view the database.

```
tmp.sql - Notepad
File Edit Search Help
/* SQL Product = ORACLE 8 */
/* Environment Indexes = CLUSTERED unique indexes */
/* Environment Drop Tables = Generate DROP */
/* Environment Drop Indices = Generate DROP */
/* Environment Drop Constraints = Generate DROP */
/* Environment Drop Procedures = Generate DROP */
/* Environment Drop Sequences = Generate DROP */
/* Environment Drop Synonyms = Generate DROP */
/* Environment Drop Views = Generate DROP */
DROP INDEX Hotel_Guest_X1;
DROP INDEX Hotel_Guest_X2;
DROP INDEX Vehicle_X3;
ALTER TABLE Hotel_Guest DROP
    CONSTRAINT belongs_to;
ALTER TABLE Hotel_Guest DROP
    CONSTRAINT belongs_to;
ALTER TABLE Vehicle DROP
    CONSTRAINT owns;
DROP VIEW Hotel_Guest_UW;
DROP INDEX Hotel_Guest_PK;
ALTER TABLE Hotel_Guest DROP
    PRIMARY KEY CASCADE;
DROP TABLE Hotel_Guest CASCADE CONSTRAINTS;
DROP VIEW Reservation_UW;
DROP INDEX Reservation_PK;
ALTER TABLE Reservation DROP
    PRIMARY KEY CASCADE;
DROP TABLE Reservation CASCADE CONSTRAINTS;
DROP VIEW Room_UW;
DROP INDEX Room_PK;
ALTER TABLE Room DROP
    PRIMARY KEY CASCADE;
DROP TABLE Room CASCADE CONSTRAINTS;
DROP VIEW Vehicle_UW;
DROP INDEX Vehicle_PK;
ALTER TABLE Vehicle DROP
    PRIMARY KEY CASCADE;
DROP TABLE Vehicle CASCADE CONSTRAINTS;
CREATE TABLE Hotel_Guest(
    Guest_Number          CHARACTER(10) NOT NULL,
    name                  CHARACTER(40) NOT NULL,
    Street                CHARACTER(25) NOT NULL,
    Town                 CHARACTER(25) NOT NULL,
```

Remember to save your work.

END OF TUTORIAL