(Introduction – Title slide)
Welcome to this demonstration of IBM InfoSphere Data Architect. InfoSphere Data Architect is a collaborative data design solution to discover, model, relate, and standardize diverse and distributed data assets.

In this demo, we'll use InfoSphere Data Architect to:
- Import and Forward engineer changes from a logical model to the database
- Publish a web-based report of the physical model for auditing and team review
- Reverse engineer a model from an existing database, then update, compare and sync the physical model with the original source database

(Forward engineering)
Let's start by exploring the forward engineering process. We'll import an existing logical data model from another system, make some changes to the model to reflect new requirements, and transform the logical model to a physical data model. From that physical model, we'll generate and save the database-specific DDL deployment script, and generate a Web report that our DBA and other team members can review. After the DDL is signed off and approved, the DBA can deploy the changed schema to the target database.

[Start demo]
When we first start InfoSphere Data Architect, we see what is called a "Perspective." InfoSphere Data Architect is built upon open-source Eclipse technology, which provides a basic workbench into which other tools, such as Data Architect, are plugged in for extended integration. Perspectives provide a grouping of *views* that are associated with the particular tools that are installed in this Eclipse instance. In this case, to work with Data Architect, we need to switch to the "Data" perspective.

We can customize this Data Perspective by resizing, adding, or removing the views. Let's close some tabbed views that we don't need for the demo. Don't worry; it's easy to get them back by selecting Window from the top menu bar and selecting whichever views you want back, or Reset perspective if you want the default Data perspective back.

The Data Project Explorer on the left is where all the data development and modeling objects reside in associated *projects*. For data modeling, we must first create a Data Design Project to store our data model objects. To do this, we right click in the Data Project Explorer and choose New, Project. (Using the right click action in Eclipse is frequently used to see what actions are available for use.) The wizard guides us through the process of creating a new Data Design Project

From this point, we could create a new logical data model. However since we have one already that we want to modify, we'll use a wizard to import what was previously created using another modeling tool by right clicking in the Data Project Explorer and choosing Import. We can import various model formats created by different modeling tools, such as CA AllFusion ERwin, Cognos, IBM Rational Rose, Sybase, Microsoft Excel, and others. Let's import a CA ERwin Data Model. First we choose the input file and then

the target project folder to store the imported model. Data Architect supports importing both logical and physical models, or, as we do here, can detect automatically what type of model we have. We choose the import options we want and the model is created.

By expanding the new data design project, we see the appropriate folders for storing model metadata and diagrams.

Now let's take a look at information about the GOSALESCT logical model that we imported by double-clicking on it. In the Data Model Editor we can first add some relevant documentation about the model.

Remember, a model is simply metadata. To visualize and work with a model, we need to work with the visual representation of the model, which is a diagram. Notice that diagrams are often too large to be seen in the diagram editor in their entirety. The outline view here gives us a visual map of where we are in the context of the complete diagram. By clicking on the outline, we can move directly to that location in the diagram.

The first thing we want to do with this model is add a new customer state tax entity, which we can do by simply dragging and dropping a new entity to the diagram, giving it a name, and adding the appropriate attributes, including the primary key, the customer state. To add more details about this key, we use the context-sensitive properties view below.

The second attribute is the customer tax rate, which is defined as a decimal. Note that with Data Architect, it is possible to use *domain models* to constrain the values and pattern of business level data types. So for example, we could have defined a domain model for tax rate to ensure no values outside an appropriate range are entered, may be between 0 and 10. A Domain model could also be used to specify when data is private and requires masking, such as social security numbers or credit card data. To enforce naming standards, you could also use a glossary model.

To create a foreign key relationship from the parent CUSTOMER STATE TAX table to the child CUSTOMER table, we simply point, click, and drag, and enter the name 'Taxes' for this relationship..

The next thing we want to do is delete an attribute we don't need any more, CUSTOMER INFO, from the customer entity. Note that the only delete option is to "Delete from Model" which means the attribute will be deleted from both the diagram *and* the data model. For entities, there is an additional option to "Delete from Diagram" which means it is only removed from the diagram but not the data model. This can be handy when different people want to focus and work on different parts of the model.

Finally, because working with diagram can make it difficult to read at times, let's right click in the model view and select arrange all to arrange the model and then save it. Then we can double-click on the diagram tab to maximize the diagram view to see our new model and then double-click again to return to its normal size.

Now, we're going to create a physical data model from the logical model we just created. We use the physical model to deploy the schema to a target database platform. With Data Architect we can create multiple physical models from a single logical model. Data Architect supports many database platforms, including all of the DB2 platforms, Informix, Oracle, SQL Server, My SQL and others. Here, we create a physical data model for DB2 for z/OS.

We can see the GOSALESCT database model is now visible in the Data Project Explorer, where we see that all the entities and attributes are now tables and columns in the schema. By default, underscore is used as a separator in names – you can change this default using the Data Architect Window preference for data management naming standards.

Next, let's generate DDL for the new schema. Because our z/OS system is under strict control, we won't deploy automatically on that server, although we certainly have that option. Instead, we'll generate and save a DDL script into the data model project SQL scripts folder so that it can be checked in a library for version control, shared with others on the team, reviewed and approved before execution by the DBA.

This concludes the forward engineering process, in which we imported a logical model, changed the logical model to reflect new business requirements, created a physical data model for a target database, and then generated and saved the DDL deployment script.

(Transition)

Next, we'll show how to create a Web report of the physical model's schema that can be a useful accompaniment with the DDL script for review.

(**Web publishing**)

Data Architect provides a variety of reporting capabilities including the ability to create customized reports using BIRT, an eclipse-based Business Intelligent Reporting Tool. For the purpose of providing audit documentation and for team review, we can create a web-based report. To do this, we select the schema and then choose Data and then Publish.

The overview page has links to the root page of each model that was published. Click on it to open the report.

On the Data Model page for this logical model, the left side shows the objects and diagrams associated with this schema. Let's look at the diagram of the model. From this diagram view, we can click on the table or columns to drill down the details for that element.

Next, we'll show how to generate a model by reverse engineering an existing database. **(Transition)**

(**Reverse Engineer**)
Let's use Data Architect to reverse engineer from an existing database. We'll create a physical data model from an existing Oracle database, then update that physical model to reflect changing business requirements. Next, we'll use compare and sync to generate the appropriate DDL for the schema changes and then directly deploy the changes to the database.

First, let's create a new physical data model from a live connection to the Oracle database in the Data Source Explorer. The Data Source Explorer provides a list of configured database connection profiles where we can perform various operations on the objects associated with the databases. . To create the model, we simply drag and drop the Oracle XE database to the data model folder in our data design project. The content of the database is analyzed and the physical data model is created along with the associated metadata.

We can make our changes to that model from the diagram or directly from the Data Project Explorer, as we show here. Let's change the length of the PRODUCT IMAGE column in the PRODUCT table, which we can do in the context-sensitive Properties view for that column. Then, delete the GROSS MARGIN column by right clicking on it and choosing Delete. Finally, to add a new column we right click on the table, choose add data object and then column. Add the new column name, PRODUCT COMMENT, and then use the properties view to specify the data type.

(**Compare and sync**)

Now, let's compare and do a bi-directional sync between the changed model and the original Oracle database. By bidirectional, we mean:
1. We can generate the DDL to update a database, and
2. We can change the data model to reflect the source database.

From the data project explorer, right click on the GOSALES schema of the physical model we just created and compare with the original source which opens up the structural compare view. The view indicates that there are differences in the PRODUCT table between the updated physical model on the left and the original Oracle database on the right.

By drilling down on the table, we can see the differences in detail. As we would expect, we can see that the new column PRODUCT COMMENT exists in the physical model and does not exist in the Oracle database. Click on the right arrow to copy the change over to the Oracle database view on the right. (Note that we aren't actually deploying anything just yet.). Another difference is the length of the data type for the PRODUCT IMAGE column. As a reminder, you can use the Properties views to change the data type

for the database, the model, or both. We'll copy that change over as well. Finally, we see that the GROSS MARGIN column does not exist in the model because we deleted it from the model earlier. Let's say we decide that a change to the physical model is incorrect, we can copy over the missing column from the database to make sure that the physical model and the database stay in sync.

On the Toolbar, Data Architect lets us do impact analysis for the objects on either side of the comparison and to generate DDL for changes that we made to objects on the left as well as on the right. Since our database is on the right side, we'll generate DDL for the right. We have the capability to run the DDL on the server now, but let's save the DDL and manually execute the DDL script later. This is a best practice so that the DDL can be reviewed and approved by the DBA as well as giving application developers a chance to modify any applications that have dependencies on that change.

[Short transition]

The DBA has approved the DDL and now the changes can be deployed from the Data Project Explorer, which has a folder just for SQL scripts. Right click on the script and choose Run SQL, then specify the target database to deploy the changes. A status message in the SQL Results view indicates the DDL has been run successfully on the Oracle server.

We can use the Data Source Explorer to directly view the changes we specified during compare and sync: the new PRODUCT_COMMENT column and the changed PRODUCT_IMAGE column.

Another great feature of the Data Source Explorer is the ability to see sample data from the data, which can help data architects understand the data characteristics in real time. For example, we can check the PRODUCT table and confirm that data for the PRODUCT IMAGE column is extended correctly and default value is correctly assigned to the new PRODUCT COMMENT column.

This concludes the reverse engineering process. To summarize, we:

- Created a new physical data model from a database
- Changed the model
- Compared the changed model with the source database to generate the delta DDL necessary to sync the model and the database.
- Executed the DDL back on the database server to deploy the changes we made in the model
- Used the Data Source Explorer to see the schema changes on the database server and view sample data from the changed table.

(Transition)
(**Conclusion & IDA Integration**)

InfoSphere Data Architect provides rich and powerful capabilities for collaborative data design and modeling. But what really sets Data Architect apart is its integration with Rational software development and design tools, InfoSphere information integration tools, and integrated data management tools such as Data Studio Developer and Administrator, facilitating alignment and collaboration across a variety of architectural, administrative, and development roles.

(Transition)
[Resources slide]
Try InfoSphere Data Architect today by clicking on the download link shown here!