


Dimension Modeling for Data Warehouse

Lesson 4: Logical Model

June 13, 2014

Proprietary and Confidential

- 1 -


Speed. Agility. Integration.

Lesson Objectives

- On completion of this lesson, you will be able to:
- Define logical model
 - List features of a logical model
 - Name the transformations required to be done while converting a conceptual model into a logical model
 - Identify activities involved in those transformations
 - Name the types of attributes which do not get converted into a single column in the logical model



4-1: Introduction

Introduction to Logical Model

- A logical model is produced from a set of well-defined transformations of the conceptual data model.
- The logical data model reflects business information requirements without considering performance.
- If the database is ported to another DBMS supporting a similar structure, the logical data model can still be used as a baseline for the new physical data model.

June 15, 2018

Proprietary and Confidential

- 3 -

IGATE
Ignite Anytime Intelligence

The primary purpose of logical data modeling is to document the business information structures, processes, rules, and relationships by a single view.

The logical data model helps to address the following:

- 1) Validation of the functional application model against business requirements
- 2) The product and implementation independent requirements for the physical database design (Physical Data Modeling)
- 3) Clear and unique identification of all business entities in the system along with their relations.

Note:

Without the logical data model, the stored business information is described by a functional model or conceptual model. Without the logical data model, there is no single view of all data, and data normalization is impossible.

In this case, the physical data model has to be designed from a functional model. This will potentially cause performance problems, and data inconsistency and redundancies, which can result in an inefficient physical design.

4.2: Features of a Logical Model

Characteristics of a Logical Model

- Logical model works in an iterative manner.
- Its design is independent of database.
- It includes all entities and relationships among them.
- All attributes for each entity are specified.
- The primary key for each entity is specified.
- Foreign keys (keys identifying the relationship between different entities) are specified.

June 15, 2018

Proprietary and Confidential

• 4 •

IGATE
Ignited Agility. Imagined.

4-3: Requisite Transformations

Transformation Required

While converting a conceptual model into a logical model, following transformations are required to be performed:

Transformations

Table Specification

Column Specification

Primary Key Specification

Foreign Key Specification

June 15, 2018

Proprietary and Confidential

+ 5 +

IGATE

Ignited Agility. Imagined.

Page 04-5

4.3: Table Specification

Activities in Table Specification

- In general, each entity class in the conceptual data model becomes a table in the logical data model and is given a name that corresponds to that of the source entity class

June 15, 2018

Proprietary and Confidential

• 6 •

IGATE

Ignited Agility. Imagined.

4.3: Column Specification

Activities in Column Specification

- In general, each attribute in the conceptual data model becomes a column in the logical data model and should be given a name that corresponds to that of the corresponding attribute.

June 15, 2018

Proprietary and Confidential

~ 7 ~

IGATE
Ignited Agility. Imagined.

4.3: Primary Key Specification

Activities in Primary Key Specification

➤ Primary Key Specification

Perform the following activities::

- Identify the primary key and unique key.
- Remove derivable objects.
- Create primary keys.
- Test them as foreign keys for related tables.
- Introduce a surrogate key, if needed.
- Establish the relationship as one-one or one-to-many.

June 15, 2018

Proprietary and Confidential

+ 8 -

IGATE
Speed Agility Innovation

Activities in Primary Key Specification:

Existing columns are assessed for the primary keys; and if required, surrogate keys are introduced.

To access data in a relational database, we need to be able to locate specific rows of a table by specifying values for their primary key column or columns.

In particular:

- We must be able to unambiguously specify the row that corresponds to a particular real-world entity instance. When a payment for an account arrives, we need to be able to retrieve the single relevant row in the **Account** table by specifying the Account Number that was supplied with the payment.
- Relationships are implemented using foreign keys, which must each point to one row only. Imagine the problems if we had an insurance policy that referred to customer number “12345” but found two or more rows with that value in the **Customer** table.

So we require that a primary key be *unique*.

- A very simple way of meeting all of the requirements is to invent a new column for each table, specifically to serve as its primary key, and to assign a different system-generated value to each row, and, by extension, to the corresponding entity instance. We refer to such a column as a **surrogate key**, which is typically named by appending “ID” (or, less often, “Number” or “No”) to the table name. Familiar examples are customer IDs, employee IDs, and account numbers allocated by the system.

4.3: Foreign Key Specification

Foreign Key Specification

➤ Foreign Key Specification:

Perform the following activities:

— Identify the foreign key and establish relationships.

— Specify the types of relationships

- One to Many
- One to One
- Many to Many

June 15, 2016

Proprietary and Confidential

— 9 —

IGATE

Ignited Agility. Imagined.

Foreign keys are our means of implementing one-to-many and occasionally one-to-one relationships.

4.4: Types of Relationships

One-to-many and Many-to-one Relationships

➤ An Employee works only for one dept

➤ A Dept has many employees

The EMPLOYEE table:

EMPNO	WORKDEPT	JOB
000010	A00	President
000020	B01	Manager
000120	A00	Clerk
000130	C01	Analyst
000030	C01	Manager
000140	C01	Analyst
000170	D11	Designer

June 15, 2016

Proprietary and Confidential

< 10 >

IGATE
Speed Agility Innovation

An employee can work in only one department; this relationship is single-valued for employees. On the other hand, one department can have many employees; this relationship is multi-valued for departments. The relationship between employees (single-valued) and departments (multi-valued) is a one-to-many relationship.

To define tables for each one-to-many and each many-to-one relationship:

- Group all the relationships for which the "many" side of the relationship is the same entity.
- Define a single table for all the relationships in the group.

One-to-Many Relationship Implementation

When moving from a conceptual to a logical data model, however, we work from a diagram to tables and apply the following rule

A one-to-many relationship is supported in a relational database by holding the primary key of the table representing the entity class at the "one" end of the relationship as a foreign key in the table representing the entity class at the "many" end of the relationship.

In the logical data model, therefore, we create, in the table representing the entity class at the "many" end of the relationship, a copy of the primary key of the entity class at the "one" end of the relationship.

4.4: Types of Relationships

One-to-One Relationship Implementation

➤ A one-to-one relationship can be supported in a relational database by implementing both entity classes as tables, then using the same primary key for both.

The DEPARTMENT table:

DEPTNO	DEPTNAME	MGRPHO	ADMDEPT
D01	Administration Support	000070	D01

The EMPLOYEE table:

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Daniel	Smith	D01	Clerk

June 15, 2016

Proprietary and Confidential

> 11 <

IGATE
Speeding Agility. Inspiring Innovation.

One-to-one relationships are single-valued in both directions. A manager manages one department; a department has only one manager. The questions, "Who is the manager of Department C01?", and "What department does Sally Kwan manage?" both have single answers. The relationship can be assigned to either the DEPARTMENT table or the EMPLOYEE table. Because all departments have managers, but not all employees are managers, it is most logical to add the manager to the DEPARTMENT table, as shown in the following example.

The above table shows the representation of a one-to-one relationship. You can have more than one table describing the attributes of the same set of entities. For example, the EMPLOYEE table shows the number of the department to which an employee is assigned, and the DEPARTMENT table shows which manager is assigned to each department number. To retrieve both sets of attributes simultaneously, you can join the two tables on the matching columns, as shown in the following example. The values in WORKDEPT and DEPTNO represent the same entity, and represent a *join path* between the DEPARTMENT and EMPLOYEE tables.

When you retrieve information about an entity from more than one table, ensure that equal values represent the same entity. The connecting columns can have different names (like WORKDEPT and DEPTNO in the previous example), or they can have the same name (like the columns called DEPTNO in the department and project tables).

Page 04-11

4.4: Types of Relationships

Many-to-Many Relationship

➤ An Employee works in Multiple projects

➤ A project has multiple employees

EMPNO	PROJNO
000030	IF1000
000030	IF2000
000130	IF1000
000140	IF2000
000250	AD3112

IGATE

June 15, 2016


Proprietary and Confidential

> 12 <

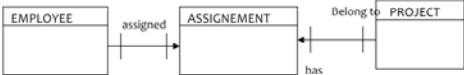
A relationship that is multi-valued in both directions is a many-to-many relationship. An employee can work on more than one project, and a project can have more than one employee. The questions "What does Dolores Quintana work on?", and "Who works on project IF1000?" both yield multiple answers. A many-to-many relationship can be expressed in a table with a column for each entity ("employees" and "projects").

4.4: Types of Relationships

Resolving Many-to-Many Relationship



➤ Resolved Many-To-Many Relationship



(Association)

➤ Data model should not contain an unresolved many-to-many relationship

June 15, 2016

Proprietary and Confidential

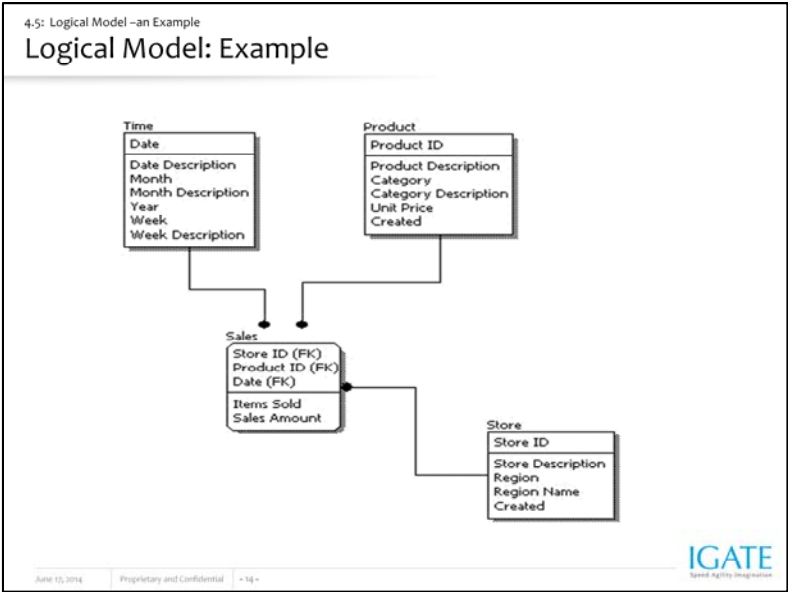
> 13 <

IGATE

Ignited Agility. Imagined Success.

Many-to-many relationships cannot be used in the data model because they cannot be represented by the relational model. Therefore, many-to-many relationships must be resolved early in the modelling process. The strategy for resolving many-to-many relationship is to replace the relationship with an association entity and then relate the two original entities to the association entity

Page 04-13



4.6: Logical Model


Logical Data Design

Feature	Conceptual	Logical	Physical
Entity Names	✓	✓	
Entity Relationships	✓	✓	
Attributes		✓	
Primary Keys		✓	
Foreign Keys		✓	
Table Names			
Column Names			
Column Data Types			

June 15, 2016

Proprietary and Confidential

+ 15 +



IGATE
Ignited Agility. Imagined.

Summary

➤ In this lesson, you have learnt that:

- A logical model is produced from a set of well-defined transformations of the conceptual data model.
- While converting a conceptual model into a logical model, some transformations, such as table and column specifications, are required to be performed.
- Except a few, each attribute of a conceptual model gets converted into a column in logical model.
- Additional columns are needed to support maintenance or operations-related data.



Add the notes here.

Review Question

- Question 1: The design of Logical Model is dependent on database.
 - True/False
- Question 2: Surrogate Keys are introduced in the Conceptual Model.
 - True/False

