Physical Design of Relational Databases

Chapter 8

Physical Database Design

- After completion of logical database design, the next phase is the design of one or more physical databases.
  - Most projects require only one physical design
  - Software vendors often “port” to several RDBMSs
  - Some organizations use both relational and NoSQL
  - May be a subset of the physical design
- In larger organizations, there may be a handoff between a a data modeler and a database administrator (DBA):
  - Nature of the handoff varies based on personalities and the organization’s culture
  - Most common is a collaboration

Designing Tables

- *Table:* a grouping of columns of data that pertains to a single particular class of people, things, events or ideas in an enterprise. (Tables represent entities).
- Tables are the primary unit of storage in relational databases. (Relational databases are table centric.)
- If sufficient time and energy was put into logical design, the physical design is mostly a mapping exercise.
  - Most data modeling tools provide automation for logical to physical mapping

Table Design Process

1. Map Entities to Tables
   - Generally, each entity becomes a table
   - Supertypes and subtypes a common exception
2. Map Attributes to Columns
   - Each attribute becomes a column
   - Attribute names converted to column names
   - Columns should be atomic, but don't overdo it

Table Design Process

2. Map Attributes to Columns (continued)

- Each column requires definition of:
  - Data Type (covered in Class 1)
  - Optionality: NULL or NOT NULL
    - Most RDBMS implementations default to NULL
    - Primary key components must be NOT NULL
  - Optionally, columns can have CHECK constraints defined
    - Defined as a simple

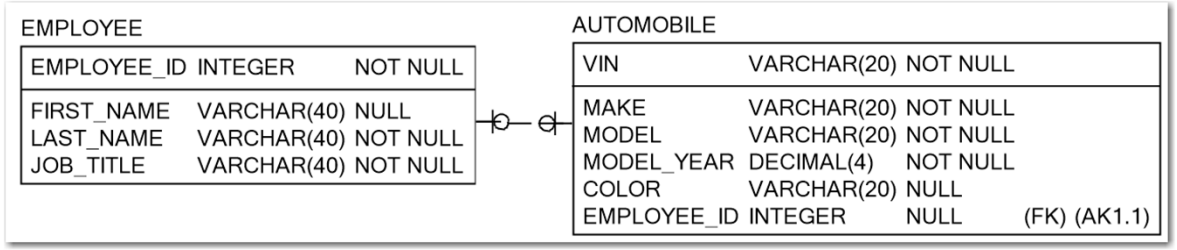
Table Design Process

3. Define the primary key
   - Only one per table, based on unique identifier in logical model
   - Components must be NOT NULL
4. Define any unique identifiers
   - Multiple per table permitted
     - Based on key candidates that were not selected as the primary key
   - NULLS permitted (in most RDBMS products)

Table Design Process

5. Define referential constraints
   - Based on relationships in the logical model
   - Most relationships are one-to-many
   - For supertypes and subtypes (always one-to-one), it is best to use the same primary key for both tables.
   - For other one-to-one relationships, be sure to define a foreign key in just one of the two tables.
     - Define a unique constraint on the foreign key column(s)
     - Defining the constraint in both directions allows for inconsistencies.

1:1 Relationship Definition (1)




```
CREATE TABLE EMPLOYEE
(    EMPLOYEE_ID          INTEGER NOT NULL,
     FIRST_NAME           VARCHAR(40) NULL,
     LAST_NAME            VARCHAR(40) NOT NULL,
     JOB_TITLE            VARCHAR(40) NOT NULL
);

ALTER TABLE EMPLOYEE
ADD CONSTRAINT PK_EMPLOYEE PRIMARY KEY
(EMPLOYEE_ID);
```

1:1 Relationship Definition (2)

```
CREATE TABLE AUTOMOBILE
(
    VIN                     VARCHAR(20) NOT NULL,
    MAKE                    VARCHAR(20) NOT NULL,
    MODEL                   VARCHAR(20) NOT NULL,
    MODEL_YEAR              DECIMAL(4) NOT NULL,
    COLOR                   VARCHAR(20) NULL,
    EMPLOYEE_ID             INTEGER NULL
);

ALTER TABLE AUTOMOBILE
ADD CONSTRAINT PK_AUTOMOBILE PRIMARY KEY (VIN);
```

1:1 Relationship Definition (3)

```
ALTER TABLE AUTOMOBILE
ADD CONSTRAINT UQ_AUTOMOBILE_EMPLOYEE_ID
    UNIQUE(EMPLOYEE_ID);

ALTER TABLE AUTOMOBILE
ADD CONSTRAINT FK_AUTOMOBILE_EMPLOYEE_ID
    FOREIGN KEY (EMPLOYEE_ID)
    REFERENCES EMPLOYEE (EMPLOYEE_ID);
```

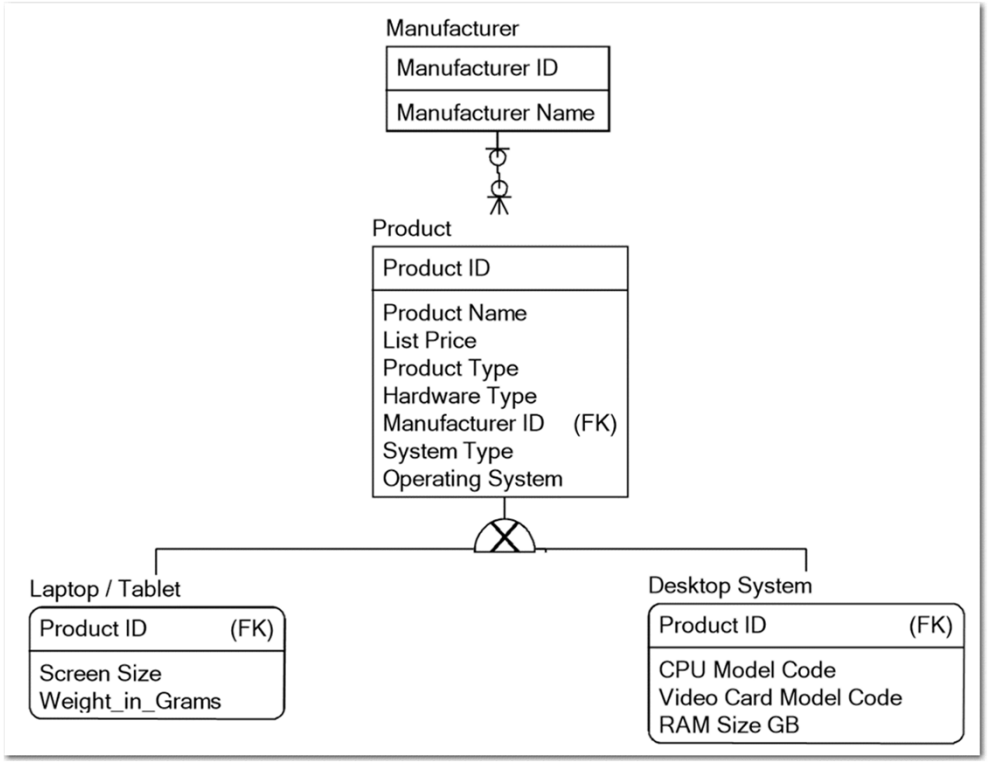
Table Design Process

6. Partition large tables
   - "Large" usually means over 1-2 gigabytes
   - *Partitioning* is a DBMS feature that permits a table to be broken into multiple physical components, each stored in separate data files, in a manner that is transparent to the database users.
   - Advantages of partitioning:
     - Partitions can be backed up and recovered independently.
     - Queries can eliminate partitions that do not apply.
     - Parallel processing is possible.
     - Partitions can be dropped nearly instantly with very little overhead. (DROP PARTITION similar to TRUNCATE)
   - Partitioning (AKA sharding) is also available in many NoSQL implementations

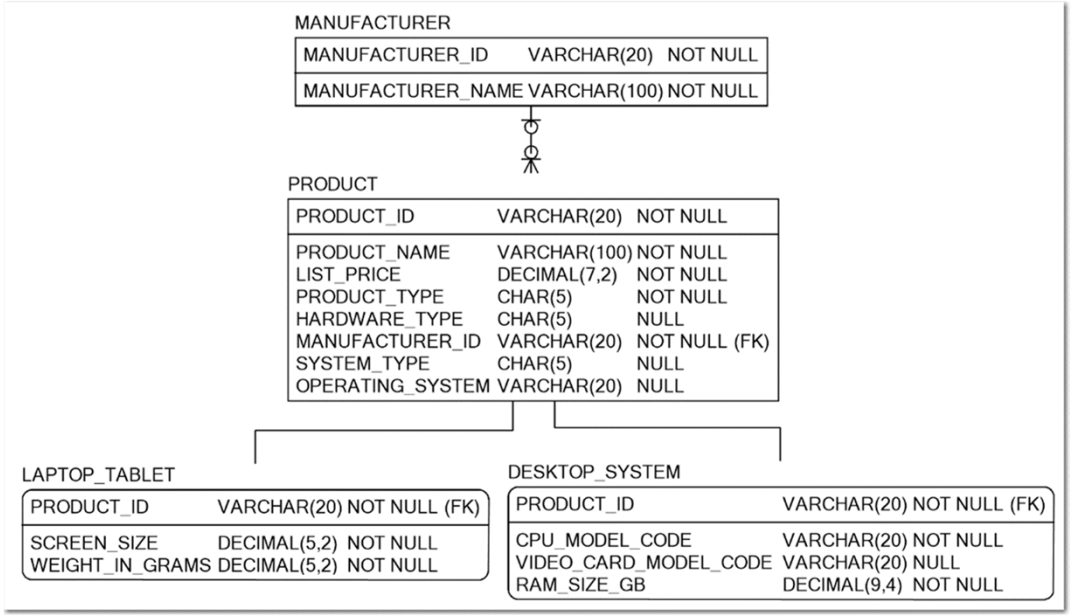
Implementing Subtypes: 3 Options

- As Is (the “three table” solution)
  - When subtypes have many common attributes and many distinct attributes
- Each Subtype as a Discrete Table (the “two table” solution)
  - When there are few common attributes among subtypes
- Collapse Subtypes into Supertype Table (the “one table” solution)
  - When there are few distinct attributes among subtypes

Logical Model with Subtypes



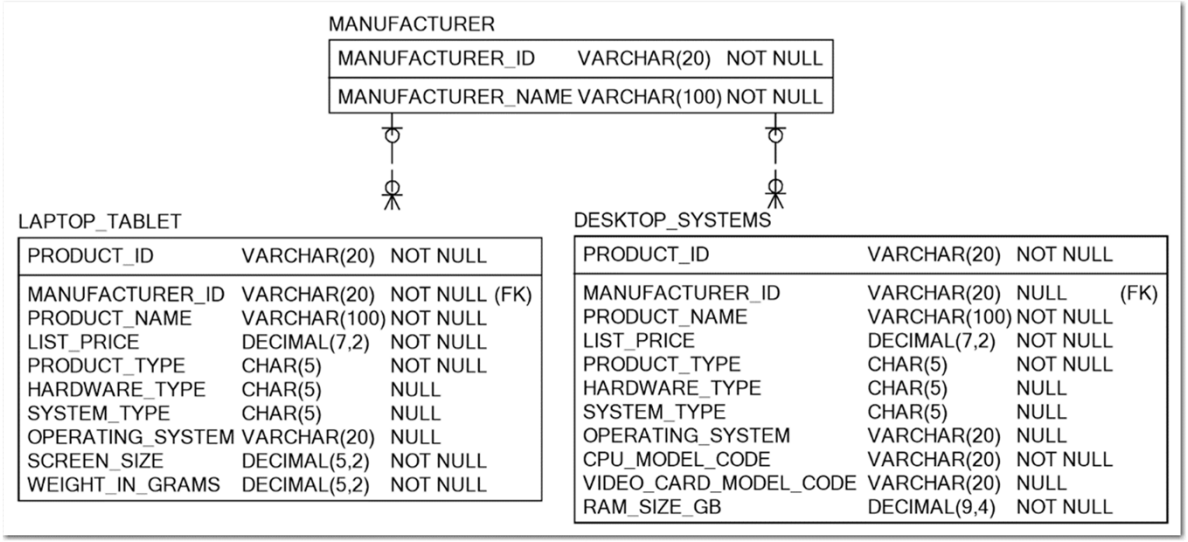
Implementing Subtypes As Is



Most appropriate when either:

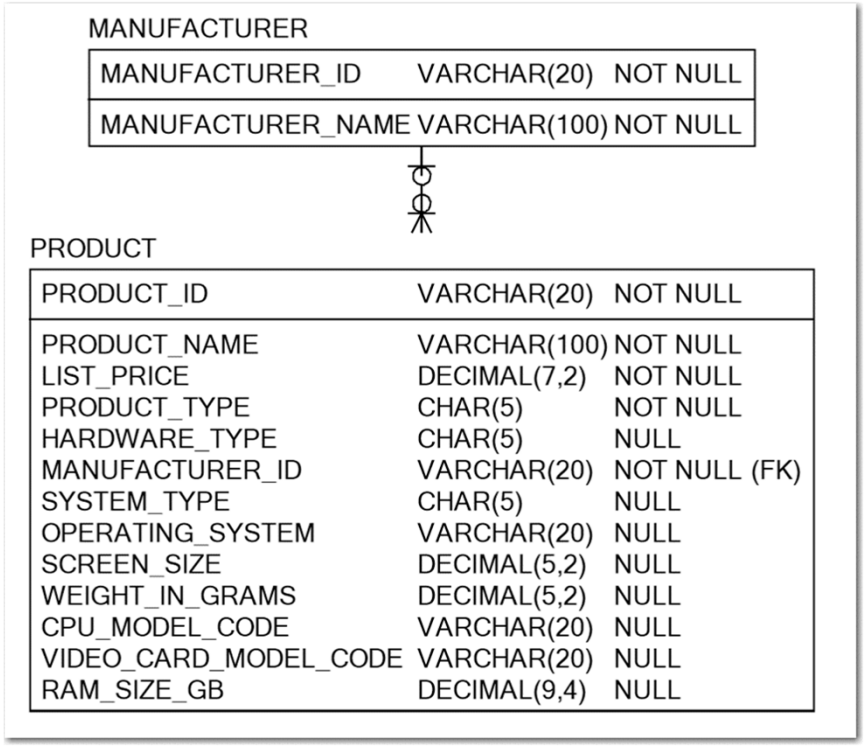
- There are many attributes that are specific to one of the subtypes
- One or more subtypes are involved in relationships that are specific to that subtype

Implementing Each Subtype as a Discrete Table



- Most appropriate where subtypes share few common attributes or relationships

Collapse Subtypes into the Supertype Table



MANUFACTURER

MANUFACTURER_ID	VARCHAR(20)	NOT NULL
MANUFACTURER_NAME	VARCHAR(100)	NOT NULL

PRODUCT

PRODUCT_ID	VARCHAR(20)	NOT NULL
PRODUCT_NAME	VARCHAR(100)	NOT NULL
LIST_PRICE	DECIMAL(7,2)	NOT NULL
PRODUCT_TYPE	CHAR(5)	NOT NULL
HARDWARE_TYPE	CHAR(5)	NULL
MANUFACTURER_ID	VARCHAR(20)	NOT NULL (FK)
SYSTEM_TYPE	CHAR(5)	NULL
OPERATING_SYSTEM	VARCHAR(20)	NULL
SCREEN_SIZE	DECIMAL(5,2)	NULL
WEIGHT_IN_GRAMS	DECIMAL(5,2)	NULL
CPU_MODEL_CODE	VARCHAR(20)	NULL
VIDEO_CARD_MODEL_CODE	VARCHAR(20)	NULL
RAM_SIZE_GB	DECIMAL(9,4)	NULL

- Most attributes must be defined as NULL
- Most appropriate when subtypes have fewer relationships and distinct attributes

Naming Conventions

- Help promote consistency in the names of database objects
- Every organization should develop a standard set of naming conventions that is published and enforced
  - Variations likely required for different DBMS products
- The conventions offered here are suggestions based on current industry best practices.

Table Naming Conventions

- Base table names based on entity names
  - Descriptive, yet concise
- Strive for unique table names across entire enterprise
  - Except when table is a duplicate copy
- Consistently use of *either* singular or plural names
  - Most data modelers prefer singular names
- Avoid technical jargon such as “table” and “file”
- Use only uppercase names with underscores to separate words
- Consistently use abbreviations
- Avoid names that limit table contents

Column Naming Conventions

- Should be based on attribute name
- Must be unique within table; should be unique within organization
- Use only uppercase letters with underscores between words
- Prefixing column names with entity names is a controversial issue
- Use abbreviations consistently
- Most experts prefer foreign key columns to have names identical to the corresponding primary key columns

Constraint Naming Conventions

- Since constraint names appear in error messages (in most RDBMSs), meaningful names are most helpful
- Suggested Format is **TNAME_TYPE_CNAME** where:
  - **TNAME** is the name of the table on which the constraint is defined
  - **TYPE** is the type of constraint (PK, FK, UQ, CK)
  - **CNAME** is the name of the column on which the constraint is defined

Index Naming Conventions

- Primary key constraint indexes automatically created by DBMS; usually named the same as the constraint
- Suggested Convention **TNAME_TYPE_CNAME** where:
  - **TNAME** is the table name
  - **TYPE** is the index type (UX, IX)
  - **CNAME** is the name of the indexed column(s), abbreviated as necessary

View Naming Conventions

- Table, Synonym, and View names come from the same *namespace,* so a view name must be unique among all table, synonym and view names
- Suggested Convention:
  - End view names with "_VW"
  - Include name of most significant base table
  - Name should describe purpose of view
  - Use only standard abbreviations

Integrating Business Rules and Data Integrity

- Business rules determine how organization operates and utilizes its data
- *Data Integrity*: the process of ensuring that data is protected and stays intact through defined constraints placed on the data
- Database constraints are enforced no matter how the database is accessed

Constraint Types

- Slides follow for each type:
  - NOT NULL constraints
  - Primary key constraints
  - Referential (foreign key) constraints
  - Unique constraints
  - CHECK constraints
  - Data types, precision, and scale
  - Triggers

NOT NULL Constraints

- Restrict the use of NULL values
- Required for primary key columns
- Watch out for defaults (Default is usually NULL)
- Example:

```
PRODUCT_NUMBER       VARCHAR(10)    NOT NULL,
PRODUCT_MANAGER_ID   NUMBER(6)      NULL
```

Primary Key Constraints

- Table may have **only** one
- Column(s) must have NOT NULL constraint
- DBMS will automatically create a unique index
- Example:

```
ALTER TABLE INVOICE_LINE_ITEM
  ADD CONSTRAINT INVOICE_LI_PK_INV_PROD_NOS
      PRIMARY KEY (INVOICE_NUMBER,
                   PRODUCT_NUMBER);
```

Referential Integrity

- The glue that binds the tables together
- Primary key column(s) on the “one” side; Foreign key column(s) on the “many” side
- Joins generally done using primary and foreign key columns
- Defined integrity is always there no matter how table is accessed

Referential Constraints

- Ensure that foreign key values in child table always have matching primary key values in the parent table
- Defined on the child table
- ON DELETE CASCADE option (if supported) allows parent delete to cascade to the children
- Example:

```
ALTER TABLE INVOICE_LINE_ITEM
   ADD CONSTRAINT INVOICE_LI_FK_INVOICE_NUMBER
       FOREIGN KEY (INVOICE_NUMBER)
       REFERENCES INVOICE (INVOICE_NUMBER);
```

Unique Constraints

- Ensure that no two rows have duplicate values in the column(s) named in the constraint
- Columns may allow NULL values
- Index automatically created (usually with same name as the constraint)
- Example:

```
ALTER TABLE ORDER
   ADD CONSTRAINT ORDER_UK_PO_NUMBER
       UNIQUE (PO_NUMBER);
```

CHECK Constraints

- Enforce business rules that restrict a column to a list or range of values or some condition that can be verified using a simple expression
- Written as conditional statements that must be true for an insert of update to succeed
- Cannot compare to values in other tables or other rows in the same table
- Example:

```
ALTER TABLE INVOICE_LINE_ITEM
  ADD CONSTRAINT INVOICE_CK_SALE_UNIT_PRICE
      CHECK (SALE_UNIT_PRICE >= 0);
```

Data Types: Length, Precision, and Scale

- Data Type automatically constrains the data to values that match the data type
- Length applies to character data types
  - Constrains the length of maximum size of the data
- Precision and Scale apply to some numeric data types
  - Precision constrains the maximum size of the data value
  - Scale determines the location of an assumed decimal point

Data Types

- Fixed length Character (CHAR, NCHAR)
- Variable length Character (VARCHAR, NVARCHAR)
- Integer Numeric (INTEGER, NUMBER, DECIMAL)
- Floating Point Numeric (FLOAT, REAL)
- Date/Time: (DATE, TIME, TIMESTAMP)
- Long Character (CLOB, NCLOB)
- Long Binary (BLOB)

Data Types (Cont.)

- Logical (BOOLEAN)
- Object Types (BFILE, COLLECTION, ROWTYPE)
- Store dates instead of ages
- Use numeric types only for columns used in arithmetic calculations

Case of Data

- Data can be in upper, lower or mixed case
  - Some DBMSs (e.g. MS SQL Server, Sybase) can be set to case insensitive mode
  - MySQL is case sensitive except on Windows
- Consistent storage and entry methods essential
- SQL functions can convert case (UPPER, LOWER, INITCAP)
  - However, without function indexes (Oracle 9i), they preclude use of an index when searching

Triggers

- *Trigger:* a unit of program code that executes automatically based on an event in the data base
- Must be written in a language supported by the RDBMS
- Can be used to enforce business rules too complex to include in CHECK constraints
  - If trigger raises an error condition, the insert or update that caused it to fire fails and the SQL statement is rolled back
  - Not always easy to decide between trigger code and application program logic to enforce constraints
    - Triggers not easily circumvented
    - Triggers can be a performance problem

Designing Views

- Views can be thought of as virtual tables
- Views are stored queries
  - When referenced in SQL, the SQL engine merely substitutes the view name with the stored query (similar to macro expansion)
- Views do not contain any data, but they can still cause performance problems
- Like all database objects, views must be managed

View Defintion (Example)

```
CREATE VIEW DESKTOP_PRODUCT_VW AS
 SELECT A.PRODUCT_ID, A.PRODUCT_NAME,
        A.LIST_PRICE, A.PRODUCT_TYPE,
        A.HARDWARE_TYPE, A.MANUFACTURER_ID,
        A.SYSTEM_TYPE, A.OPERATING_SYSTEM,
        B.CPU_MODEL_CODE,
        B.VIDEO_CARD_MODEL_CODE,
        B.RAM_SIZE_GB
  FROM  PRODUCT A
   JOIN DESKTOP_SYSTEM
     ON A.PRODUCT_ID = B.PRODUCT_ID;
```

Advantages of Views

- In some RDBMS products, views offer performance advantages due to compiled SQL
- May be tailored to individual needs
- Insulate users from changes to object names
- Can simplify data usage by hiding complex joins and calculations
- Can assist security by restricting access to some rows and/or columns

View Restrictions

- For views containing joins, any DML statement must reference only one table
- Inserts are not possible into views that have omitted any required columns
- Calculated and derived columns in views cannot be updated
- Privileges are required on views as well as the base tables they reference
- Some RDBMS products have other restrictions

A Tune About Backups

YESTERDAY

Yesterday, all those backups seemed a waste of pay.
Now my database has gone away.
Oh I believe in yesterday.

Suddenly, there's not half the files there used to be,
And there's a milestone hanging over me
The system crashed so suddenly.

I pushed something wrong
What it was I could not say.
Now all my data's gone
and I long for yesterday-ay-ay-ay.

Yesterday, the need for back-ups seemed so far away.
I knew my data was all here to stay,
Now I believe in yesterday.