Physical Database Design

(Chapter 8)

Physical Database Design Process

- Transformation of logical database design to a physical database design
- Requires a profound shift in topics
  - Significant knowledge of target DBMS required
  - DBA often carries out at least part of physical database design
  - Performance concerns need to be addressed
  - Goal is to produce (generate) SQL DDL to define the database objects (tables, columns, indexes, views, etc.

Skills / Knowledge Required

- Understanding of the logical database design
- Features of the target DBMS, esp. storage and indexing
- DBMS tuning options and trade-offs
- The operating system (OS) on which DBMS will run
- The hardware on which the database server will run
- Physical storage mechanisms available on the particular platform

Inputs to Physical Design

- Logical database design
- Process models, including when and how often rows are added, updated, deleted, retrieved
- Process/Entity (CRUD) Matrix
- Performance requirements
- Target DBMS
- Disk space constraints
- Development schedule
- Data retention requirements
- Data volumes and growth rate

Table Design Process

1. Each normalized relation becomes a table
   - Common exceptions are supertypes and subtypes
2. Each attribute becomes a column in a table, specifying:
   - Unique column name within the table
   - Data type with length/precision/scale as required
   - Whether values are required or not
   - Check constraints
- Primary Key constraint defined on unique identifier
- Unique constraint defined on other candidate keys
- Relationships become referential constraints

Physical Design Process (2)

6. Physical storage specifications added:
   - Tablespace assignment (file group in SQL Server)
   - Consider index organized table (IOT) options
   - Free space
   - Data compression
   - Clustering
7. Specify partitioning for very large tables
8. Alternatively, consider splitting very large tables
9. Set up any required replication
10. Add new tables and/or columns required for audit
11. Physical model can be a subset of the logical model

Physical Design Notes

- Primary key constraint components must be defined as NOT NULL
- Unique constraint components can be NULL (subject to DBMS restrictions)
- Only one primary key constraint per table, but multiple unique constraints are o.k.
- For 1:1 relationships, implement with a referential constraint and a unique constraint on the primary key in one table that was placed as a foreign key in the other table.
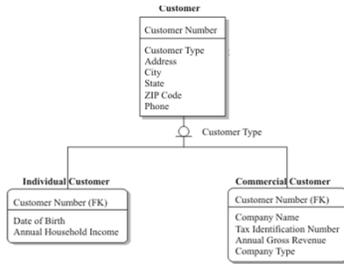
Typical Physical Diagram Differences

- Logical names shifted to all caps with underscores replacing spaces or special characters
- Data types displayed on diagram
- NULL / NOT NULL displayed on diagram
- Optionally, referential constraint names displayed on diagram (in place of verb phrases on logical diagram)
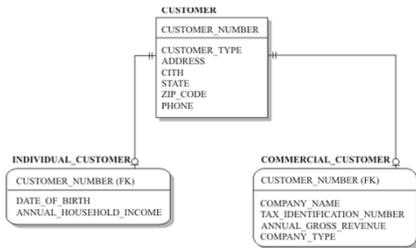- Views may be shown

Implementing Supertypes and Subtypes

- Three basic choices (subsequent slide on each):
  - Implement as is (the "three table solution")
  - Push supertype down into each subtype (the "two table solution")
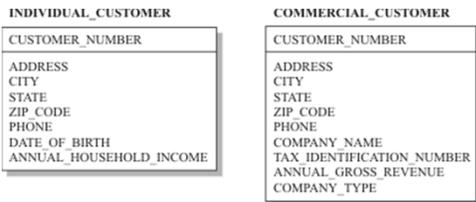 - Roll subtypes up into the supertype (the "one table solution")

Logical Model

Customer

Customer Number
Customer Type
Address
City
State
ZIP Code
Phone

Customer Type

Individual Customer

Customer Number (FK)
Date of Birth
Annual Household Income

Commercial Customer

Customer Number (FK)
Company Name
Tax Identification Number
Annual Gross Revenue
Company Type

Implement Subtypes As Is

CUSTOMER

CUSTOMER_NUMBER
CUSTOMER_TYPE
ADDRESS
CITH
STATE
ZIP_CODE
PHONE

INDIVIDUAL_CUSTOMER

CUSTOMER_NUMBER (FK)
DATE_OF_BIRTH
ANNUAL_HOUSEHOLD_INCOME

COMMERCIAL_CUSTOMER

CUSTOMER_NUMBER (FK)
COMPANY_NAME
TAX_IDENTIFICATION_NUMBER
ANNUAL_GROSS_REVENUE
COMPANY_TYPE

Push Supertype Into Subtypes

INDIVIDUAL_CUSTOMER

CUSTOMER_NUMBER
ADDRESS
CITY
STATE
ZIP_CODE
PHONE
DATE_OF_BIRTH
ANNUAL_HOUSEHOLD_INCOME

COMMERCIAL_CUSTOMER

CUSTOMER_NUMBER
ADDRESS
CITY
STATE
ZIP_CODE
PHONE
COMPANY_NAME
TAX_IDENTIFICATION_NUMBER
ANNUAL_GROSS_REVENUE
COMPANY_TYPE

Roll Subtypes Into Supertype

CUSTOMER

CUSTOMER_NUMBER
CUSTOMER_TYPE
ADDRESS
CITY
STATE
ZIP_CODE
PHONE
COMPANY_NAME
TAX_IDENTIFICATION_NUMBER
ANNUAL_GROSS_REVENUE
COMPANY_TYPE
DATE_OF_BIRTH
ANNUAL_HOUSEHOLD_INCOME

Table Naming Conventions

- Table names based on entity names
- Table names unique across entire organization
- Consistency of singular vs. plural names
- Do not use names like "table" or "file"
- Best to use only uppercase letters and underscores
- Best to use abbreviations only when necessary
- Avoid limiting words such as WEST_SALES

Column Naming Conventions

- Column names based on attribute names
- Column names unique within the table
- Best to use only uppercase letters and underscores
- Prefixing column names with entity names is controversial
- Best to use abbreviations only when necessary
- Foreign key column names same as primary key columns except when role names are required

Constraint Naming Convention

- Important because constraint names can appear in DBMS error messages
- Suggested convention: TNAME_TYPE_CNAME where:
  - TNAME is the name of the table
  - TYPE is:
    - PK for primary key constraints
    - FK for foreign key constraints
    - UQ for unique constraints
    - CK for check constraints
  - CNAME is the most important column name

Index Naming Convention

- Most DBMSs permit indexes for primary key / unique constraints to be pre-defined, so you can specify name
- Suggested convention: TNAME_TYPE_CNAME where:
  - TNAME is the name of the table being indexed
  - TYPE is the type of index:
    - UX for unique indexes
    - IX for non-unique indexes
  - CNAME is the name of the most important column

View Naming Conventions

- Must be unique among all tables, views and synonyms in the same schema
- Suggested convention:
  - End names with a suffix such as _VW
  - Include the name of the most table
  - Attempt to describe the purpose or contents of the view
  - Add any abbreviations used to the standard list

18

6

Implement Business Rules as Constraints

- NOT NULL constraints
- Primary key constraints
- Referential (foreign key) constraints
- Unique constraints
- Check constraints
- Data types, length, precision/scale
- Triggers

Adding Indexes for Performance

- How a b-tree (balanced tree) index works:
  http://mattfleming.com/node/192

- How a bit map index works

Index Guidelines

- Remember that RDBMSs automatically create indexes for primary key and unique constraints
- Indexes on foreign key columns can dramatically improve join performance
- If a query selects only columns from a single index, table row fetches are not necessary
- Consider indexes on columns that are frequently referenced in WHERE clauses
- Indexes on long VARCHAR columns are seldom useful
- Indexes cannot be used to find NULL values

Index Guidelines (2)

- The larger the table, the less you want table scans
- Indexes on frequently updated columns can be trouble
- For relatively small tables, table scans are just fine
- For tables with short rows that are most often accessed using primary keys, consider an index organized table
- Consider the performance consequences before defining more than two or three indexes on a table
- For B-tree, index selectivity needs to be high (0.8 – 1.0)
- For low selectivity and relatively few values, consider a bitmap index

Designing Views

- View Restrictions:
  - For views referencing multiple tables, any insert, update or delete can only reference columns from one table
  - Inserts are impossible when required (NOT NULL) columns are left out unless they have DEFAULT values
  - Calculated and derived columns in views cannot be updated
  - View access requires privileges (just as table access does)
  - DBMSs vary somewhat in view support and restrictions

Advantages of Views

- In some RDBMSs, view access performs better than table access (stored procedures may be better still)
- Views may be tailored to user department needs
- Views can provide alternative representations (transformations) of the data
- Views insulate users from some table/column changes
- Views simplify access by hiding complex joins and calculations
- Views can omit rows and columns that users don't need to see (a good security tool)
- Views can reestablish supertypes and subtypes that were no implemented