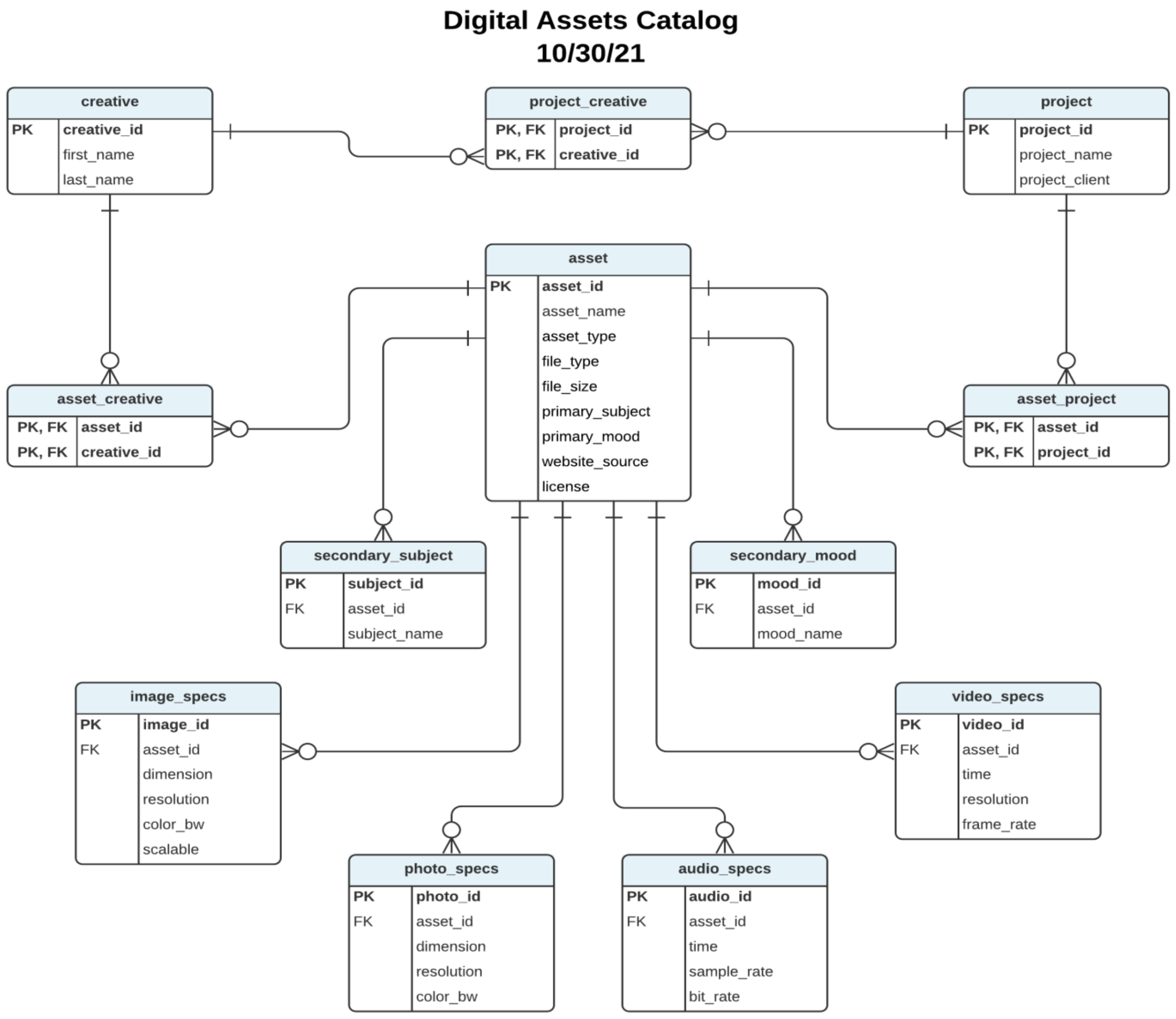

Asset Database Project

A Stepwise Approach

October 2021

Renee Raven

Table of Contents

Scenario

Our client runs a small web design business and often uses open source and creative common digital assets (photos, graphics, audio, video). Each asset may be used in any number of projects. She would like a database to catalog the items she has collected and/or used in projects.

Currently she has over 5,000 assets distributed in five folders: BW Photos, Color Photos, Graphics, Audio, and Video. Her collection continues to grow. She saves the new items in the appropriate folder and titles them with a unique name. However, the number of items, and the lack of a system to search for the items, often leads to trouble finding specific assets.

Constraints & Considerations

- Each asset may have multiple creatives (creators/authors)
- Each asset may be used for any number of projects
- Each project may use any number of assets
- Each creative may produce any number of assets
- Each asset has a primary subject
- Each asset may have any number of secondary assets
- Each asset has a primary mood
- Each asset may have any number of secondary moods
- Each project has 1 and only 1 project owner

Additionally, our client requested the ability to search the database of her collection of digital assets by:

- Name
- Creative(s)
- Type
- Source
- Project(s)
- File size
- Subject(s)
- Mood(s)
- File type
- License
- Resolution
- Dimensions
- Bit rate
- Project owner

Identify Entities

- asset
- creative
- project
- secondary subjects
- secondary moods
- photo specs
- image specs
- audio specs
- video specs

Preliminary List of Attributes with Entities

- asset
    - asset_id
    - asset_name
    - creative(s)
    - asset_type (category - photo, graphic, audio, video)
    - file_type (category - multiple options)
    - file_size
    - primary_subject
    - primary_mood
    - asset_source
    - license_type
    - project(s)
- creative
    - creative_id
    - creative_name
    - project(s)
    - asset(s)
- project
    - project_id
    - project_name
    - asset(s)
    - creative(s)
    - owner
- subjects
    - asset_id
    - subject
- moods

    - asset_id
    - mood
- photo_specs
    - photo_id
    - asset_id
    - dimensions
    - resolution
    - color or bw (category - color or bw)
- image_specs
    - image_id
    - asset_id
    - dimensions
    - resolution
    - color or bw (category - color or bw)
    - scalable (category - yes or no)
- audio_specs
    - audio_id
    - asset_id
    - time
    - sample_rate
    - bit_rate
- video_specs
    - video_id
    - asset_id
    - time
    - resolution
    - frame_rate

Check Atomicity of Attributes

The asset, creative, and project tables contain attributes where multiple values are possible. Since not every asset is used in a project and not every project is attached to a creative, we can’t make a table to connect the three tables.

We need to create 3 associative entity tables built with a combination of foreign keys acting as a primary key and then remove the non-atomic attributes from the asset, creative, and project tables.

New associative tables to add:

- asset_creative
    - asset_id
    - creative_id
- asset_project

    - asset_id
    - project_id
- creative_project
    - creative_id
    - project_id

Identifiers / Keys

- asset PK asset_id
- creative PK creative_id
- project PK project_id
- project_creative PK, FK project_id, PK, FK creative_id
- asset_project PK, FK asset_id, PK, FK project_id
- asset_creative PK, FK asset_id, PK, FK creative_id
- secondary_subjects PK subject_id, FK asset_id
- secondary_moods PK mood_id, FK asset_id
- photo_specs PK photo_id, FK asset_id
- image_specs PK image_id, FK asset_id
- audio_specs PK audio_id, FK asset_id
- video_specs PK video_id, FK asset_id

Relationships

- asset --> asset_project; 1 asset can belong to multiple asset_project instances -->1:N
- asset_project --> asset; 1 asset_project always includes 1 and only 1 asset --> M:1
- asset --> asset_creative; 1 asset can belong to multiple asset_creative instances -->1:N
- asset_creative --> asset; 1 asset_creative always includes 1 and only 1 asset --> M:1
- creative --> asset_creative; 1 creative can belong to multiple asset_creative instances -->1:N
- asset_creative --> creative; 1 asset_creative always includes 1 and only 1 creative --> M:1
- creative --> project_creative; 1 creative can belong to multiple asset_project instances -->1:N
- project_creative --> creative; 1 project_creative always includes 1 and only 1 creative --> M:1
- project--> asset_project; 1 project can belong to multiple asset_project instances -->1:N
- asset_project --> project; 1 asset_project always includes 1 and only 1 project --> M:1
- project --> project_creative; 1 project can belong to multiple project_creative instances -->1:N
- project_creative --> project; 1 project_creative always includes 1 and only 1 project --> M:1
- asset --> secondary_subject; 1 asset can belong to multiple secondary_subject instances -->1:N
- secondary_subject --> asset; 1 secondary_subject always includes 1 and only 1 asset --> M:1
- asset --> secondary_mood; 1 asset can belong to multiple secondary_ mood instances -->1:N
- secondary_ mood --> asset; 1 secondary_ mood always includes 1 and only 1 asset --> M:1
- asset --> image_specs; 1 asset can belong to multiple image_specs instances -->1:N

- image_specs --> asset; 1 image_specs always includes 1 and only 1 asset --> M:1
- asset --> photo_specs; 1 asset can belong to multiple photo_specs instances -->1:N
- photo_specs --> asset; 1 photo_specs always includes 1 and only 1 asset --> M:1
- asset --> audio_specs; 1 asset can belong to multiple audio_specs instances -->1:N
- audio_specs --> asset; 1 audio_specs always includes 1 and only 1 asset --> M:1
- asset --> video_specs; 1 asset can belong to multiple video_specs instances -->1:N
- video_specs --> asset; 1 video_specs always includes 1 and only 1 asset --> M:1

Normalize

The tables are in 3NF.

Each table meets the criteria of 1NF. Every attribute is atomic and single-valued, meaning there are no repeating groups of columns in an entity, and each table has an identified primary key.

Each table meets the criteria for 2NF. The associative tables asset_creative, asset_project, and project_creative each have composite primary keys that define the unique instance of that table. There are no non-key attributes to judge dependencies. The rest of the tables don't have composite primary keys, so they are already in 2NF.

Finally, all tables meet the criteria for 3NF because all values in non-primary key columns are determined by the primary key and there are no transitive dependencies on non-primary key columns.

Assign Char Types to Revised Attributes

- asset
  - asset_id
  - asset_name
  - asset_type (category - photo, gaphic, audio, video)
  - file_type (category - multiple options)
  - file_size
  - primary_subject
  - primary_mood
  - website_source
  - license
- creative
  - creative_id
  - first_name
  - last_name

- project
    - project_id
    - project_name
    - project_client
- project_creative
    - project_id
    - creative_id
- asset_project
    - asset_id
    - project_id
- asset_creative
    - asset_id
    - creative_id
- secondary_subject
    - asset_id
    - subject_id
    - subject_name
- secondary_mood
    - asset_id
    - mood_id
    - mood_name
- photo_specs
    - photo_id
    - asset_id
    - dimensions
    - resolution
    - color_bw
- image_specs
    - image_id
    - asset_id
    - dimensions
    - resolution
    - color_bw
    - scalable
- audio_specs
    - audio_id
    - asset_id
    - time
    - sample_rate
    - bit_rate
- video_specs
    - video_id
    - asset_id
    - time

    - resolution
    - frame_rate

DDL

DML

SQL