Technical Design Document
Project: Band Gear Exchange Forum
Course: Database Systems and Web Programming 2024 Summer

Band-Gear-Exchange-Forum statusreport 04.08.24:

PostgreSQL db created in my local PC:

- db schema designed and created with primary and foreign keys (9 tables)

- db is populated with test data

- db inquiries and sql-views tested (couple of relevant use-cases of the service)

- first development version of the bgef-db is ready (version 4)

Next steps for db: already coded test-service use-cases as sql-views will be tested to find out the most feasible use cases for the final bgef-service.

MS Edge Copilot LLM assisted in development following standard CRUD and Fetch-services. These e2e-services are mainly targeted to test the db-schema and support as a prototyping tool in innovation of the final use-cases of BGEF-service:

Flask application: CRUD and Fetch(all and one) use-cases. Coded but not tested: model, service, controller and HTML/css-based UI for CRUD and Fetch.

Next steps with bgef-service: most feasible use-cases will be defined and prototyped. The allready coded Flask-components will be tested soon and db solution will be tested thoroughly with CRUD and Fetch functions. Based on the experiences the bgef-service use-cases and related UI will be designed for prototyping.

Documentation: documents will be updated soon: entity-diagram, db-schema, use-cases and UI, instructions. Testing: Test-log will be maintained, test data-set will be improved according to selected use-cases Configuration: config-files will be maintained and test-cycles speeded up (CI)

Band-Gear-Exchange-Forum documentation

Overview

This plan briefly describes the system’s conceptual analysis and database design. Additionally, it presents a few key use cases and the database queries they require. Finally, there is a short description of the practical implementation’s architectural solutions.

The Band Gear Exchange Forum (BGEF) is a web-based service that allows users, referred to as “clients in different roles”, to create and accept offers for exchanging band equipment.

This document also outlines the functionalities, user types, and operations of the service. The technical implementation of the application is based on the Flask web framework and SQLAlchemy, which provides an ORM interface layer for the PostgreSQL database. The user interface is implemented with HTML pages.

I**mage 1:** High level architecture of the system (will be updated here later on).

See next page, please. This page is left intentionally blank for a system picture.

Service Functions

User Registration

User Types

Registered users: Users who have created an account and logged in.

Unregistered Users: Users who browse the service without logging in.

System Functions

The system maintains, searches and sorts exchange offers
for band equipment.

Exchange Process

- **Offer Creation**: Bidders can create offers to exchange band equipment.
- **Offer Acceptance**: Bidders can accept any exchange offers.
- **Mutual Agreement**: A successful exchange requires mutual acceptance of each other's offers. Users can modify their offers to better suit the exchange operation.

Communication

Questions and answers about the band equipment are exchanged via email or
phone between the parties involved.

Financial Transactions

No money is exchanged through the service.
Any financial arrangements are made separately, outside the service, e.g.via phone or email.

User Operations via Web Browser

Bidders can perform the following operations by using their web browsers:

Create Offers: Registered Bidders can create new exchange offers.

Browse Offers: Both registered and unregistered Bidders can browse existing offers.

Accept Offers: Registered Bidders can accept offers made by others.

Modify Offers: Bidders can modify their offers to better suit the exchange deal.

Send Inquiries: Interested users can send inquiries to Bidders via email.

Users can register and list their exchange offers in various categories, such as:

- Instruments
- Amplifiers
- Microphones
- Studio Equipment
- Etc.

Offer Details

Users input the details of their exchange offers as follows:

- Define the start and end dates of the exchange offer.
- Set compensation and delivery preferences.
- Maintain the status of their exchange offers and adjust the content as needed.
- Remove offers once the exchange is completed.

System Functions

The system maintains, searches, suggests, and sorts exchange offers for band equipment. It includes:

- Creation and maintenance of band equipment lists.
- Maintenance and printing of open offers list.
- Status update of offers.

Listing and Status Management

- Lists offers by user, status, category, date, and content.
- Changes the status of offers that lead to exchanges to “completed” but does not delete them.

Database Entities

The relationships between the tables are as follows:

- **Client**: Each user can create multiple bids and bid headers in any bid genres.
- **BidGenre**: Categorizes bids and is linked to a specific bidder.
- **BidHeader**: Contains the main information about a bid, including the initial bidder and the genre of the bid.
- **BidContent**: Stores the detailed content of each bid and is linked to both the bidder and the bid header.
- **Gear**: Stores the detailed content of each bid and is linked to both the bidder and the bid header.
- **ImageMetadata**: Image and icon metadata-file can be linked to Gear, BidContent and Client tables via fk-links, which contains image definition metadata and the url to the image mediafile.
- **SoundClipMetadata**: Sound-samples can be linked to Gear and BidContent tables, which contains sound-clip definition metadata and the url to the sound-clip mediafile.

Database and Conceptual Model Documentation

Introduction

The following sections provide implementation design of each table and their relations.
The entity relation diagram will be on this page, continue next page, please.

Database Schema db_BGEF_v4.0

1.Client Table

The Client table stores information about users, who participate in the BGEF-system.

- **id**: A unique identifier for each user (Primary Key).
- **userName**: The name of the user.
- **password**: The password for the registered user's account .
- **email:** User's primary email
- **secondaryEmail:** User's secondary email
- **userIcon_Url:** Foreign Key to user's user's icon metadata file (optional)
- **role_id:** Foreign Key to User's roles in Role-table
- **blocked:** User is blocked out of the BGEF-service provisioning.
- **latestBidHeader_id:** Foreign Key to User's last active bid

```
CREATE TABLE Client
(
    id SERIAL NOT NULL PRIMARY KEY,
    userName VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(25) DEFAULT NULL,
    email VARCHAR(80) DEFAULT NULL,
    secondaryEmail VARCHAR(80) DEFAULT NULL,
    userIcon_Url VARCHAR(80) DEFAULT NULL, --user's icon metadata with icon
    role_id INT NOT NULL,
    blocked BOOLEAN DEFAULT FALSE,
    latestBidHeader_id INT,

    FOREIGN KEY(role_id) REFERENCES Role(id),
    FOREIGN KEY(userIcon_url) REFERENCES ImageMetadata(id),
    FOREIGN KEY(latestBidHeader_id) REFERENCES BidHeader(id)
  );
```

2. Role Table

- **id:** A unique identifier for each role (Primary Key).
- **roleName:** bidder, byer, admin, unregistered.
- **registeredUser:** Indicates user registration status.
- **sellerRole:** True if user have any active bids.
- **buyerRole:** True if user has bought any gear.
- **adminRole:** A boolean flag indicating if the user has administrative privileges.

```
CREATE TABLE Role
(
    id SERIAL NOT NULL PRIMARY KEY,
    roleName VARCHAR(80) NOT NULL UNIQUE,
    registeredUser BOOLEAN DEFAULT FALSE,
```

```
    sellerRole BOOLEAN DEFAULT FALSE,
    buyerRole BOOLEAN DEFAULT FALSE,
    adminRole BOOLEAN DEFAULT FALSE
  );
```

3. BidGenre Table

The BidGenre table categorizes the types of gear bids.

- **id**: A unique identifier for each bid genre (Primary Key).
- **bidGenreHeader**: A unique header for the bid genre.
- **user_id**: A foreign key referencing the Client table.
- **LastBidHeader_id:** A foreign key referencing the last active BidHeader of the BidGenre.
- **numberOfBids:** Number of active BidHeaders of the BidGenre.

```
CREATE TABLE BidGenre
(
    id SERIAL NOT NULL PRIMARY KEY,
    bidGenreHeader VARCHAR(100) NOT NULL UNIQUE,
    user_id INT,
    LastBidHeader_id INT,
    numberOfBids INT NOT NULL,

    FOREIGN KEY(user_id) REFERENCES User(id),
    FOREIGN KEY(lastBidHeader_id) REFERENCES  BidHeader(id)
  );
```

4. BidHeader Table

The BidHeader table contains information about individual bids (singles or bundles).

- **id**: A unique identifier for each bidheader (Primary Key).
- **user_id**: A foreign key referencing the Client table.
- **bidGenre_id**: A foreign key referencing the BidGenre table.
- **bidTypeCode:** Limited time offer, Flash offer, Bundle, Single...
- **bidStatus:** active, inactive, waiting to publishing, ending soon...
- **bidContent_id:** A foreign key referencing the BidContent table.
- **bidHeaderTex**t: Bidheader text. Edited by the user for marketing purposes.
- **bidExchange1_id -> bidExchange5_id:** Foreign keys referencing the BidExchange table.
- **bidStartedTime**: The timestamp when the bid started.
- **bidEndingTime**: The timestamp when the bid will end.

- **bidEditedTime**: The timestamp when the bid was last edited.

```
CREATE TABLE BidHeader
(
    id SERIAL NOT NULL PRIMARY KEY,
    user_id INT,
    bidGenre_id INT,
    bidTypeCode INT,
    bidStatus INT,
    bidContent_id INT,
    bidHeaderText VARCHAR(100) NOT NULL UNIQUE,
    bidExchange1_id INT,
    bidExchange2_id INT,
    bidExchange3_id INT,
    bidExchange4_id INT,
    bidExchange5_id INT,
    bidStartedTime TIMESTAMP NOT NULL,
    bidEndingTime TIMESTAMP NOT NULL,
    hdrEditedTime TIMESTAMP NOT NULL,

    FOREIGN KEY(user_id) REFERENCES User(id),
    FOREIGN KEY(bidGenre_id) REFERENCES BidGenre(id),
    FOREIGN KEY(bidContent_id) REFERENCES BidContent(id),
    FOREIGN KEY(bidExchange1_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange2_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange3_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange4_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange5_id) REFERENCES BidExchange(id)
    );
```

5. BidContent Table

The BidContent table stores the content of individual bids.

- **id**: A unique identifier for each bid content (Primary Key).
- **bidHeader_id:** A foreign key referencing the BidHeader table.
- **bidContent**: Descriptive text or summary of the content of the bid.
- **gear1_id -> gear10_id:** Foreign keys referencing the Gear table.
- **conEditedTime**: The timestamp when the bid was last edited.
  Optional:
- **bundleImage_id:** A foreign key referencing to the bundle image metadata file.
  Optional:
- **bundleSound_id:** A foreign key referencing to the bundle sound sample metadata file.

```
CREATE TABLE BidContent
(
    id SERIAL NOT NULL PRIMARY KEY,
    bidHeader_id INT,
    bidContent VARCHAR(500) NOT NULL,
    gear1_id INT,
    gear2_id INT,
    gear3_id INT,
    gear4_id INT,
    gear5_id INT,
```

```
    gear6_id INT,
    gear7_id INT,
    gear8_id INT,
    gear9_id INT,
    gear10_id INT,
    conEditedTime TIMESTAMP NOT NULL,
    bundleImage_id INT, -- Link to bundle image metadata file
    bundleSound_id INT, -- Links to bundle sound sample metadata file

    FOREIGN KEY(bidHeader_id) REFERENCES BidHeader(id),
    FOREIGN KEY(bundleImage_id) REFERENCES ImageMetadata(id),
    FOREIGN KEY(bundleSound_id) REFERENCES SoundClipMetadata(id),
    FOREIGN KEY(gear1_id) REFERENCES Gear(id),
    FOREIGN KEY(gear2_id) REFERENCES Gear(id),
    FOREIGN KEY(gear3_id) REFERENCES Gear(id),
    FOREIGN KEY(gear4_id) REFERENCES Gear(id),
    FOREIGN KEY(gear5_id) REFERENCES Gear(id),
    FOREIGN KEY(gear6_id) REFERENCES Gear(id),
    FOREIGN KEY(gear7_id) REFERENCES Gear(id),
    FOREIGN KEY(gear8_id) REFERENCES Gear(id),
    FOREIGN KEY(gear9_id) REFERENCES Gear(id),
    FOREIGN KEY(gear10_id) REFERENCES Gear(id)
);
```

6. Gear Table

- **id**: Unique system identifier for the equipment.
- **gearBrand: e.g. VOX, Roland, Sovtek, Marshall, Yamaha, Fender, Gibson, Gretch**
- **Gear Code**: Unique manufacturing-code for the equipment.
- **gearStatus:** e.g. reserved, free to offer, broken, badly scratched, used, mint
- **bidHeader_id:** A foreign key referencing the BidHeader table.
- **gearName**: Name and type details of the equipment, such as model identifier and year of manufacturing.
- **gearDetails**: Detailed description of the equipment including, size, weight, condition, color, sound-description, etc.
- **gearStory** (optional)**:** Life-story of a Vintage-instrument, e.g. if famous musicians have used it or similar gear.
- **amountOffered:** Offered number of gear or gear.
- **approxValue:** The approximated current value of the gear in euros.

- **Comments:** Bidder's preferencess and other relevant details.
- **user_id**: A foreign key referencing the Client table.g. for user's equipment listing
- **comments:** User preferences concerning e.g. delivery and payment**.**
- **editedTime**: The timestamp when the gear attributes were last edited.
- **gearImage_id:** Foreign Key to gear image metadata file (optional)
- **soundClip_id:** Foreign Key to gear sound sample metadata file (optional)

```
CREATE TABLE Gear
(
  id SERIAL NOT NULL PRIMARY KEY,
  gearBrand VARCHAR(30) NOT NULL,
  gearCode INT,
  gearStatus INT,
  bidHeader_id INT,
  gearName VARCHAR(100) NOT NULL,
  gearDetails TEXT,
  gearStory TEXT,
  amountOffered INT NOT NULL,
  approxValue FLOAT default 0.00,
  user_id INT,
  comments TEXT,
  editedTime TIMESTAMP NOT NULL,
  gearImage_id INT,  -- Links to gear image metadata file (optional)
  soundClip_id INT, -- Links to gear sound sample metadata file (optional)

  FOREIGN KEY(user_id) REFERENCES User(id),
  FOREIGN KEY(bidHeader_id) REFERENCES BidHeader(id),
  FOREIGN KEY(gearImage_id) REFERENCES ImageMetadata(id),
  FOREIGN KEY(soundClip_id) REFERENCES SoundClipMetadata(id));
```

7. BidExchange Table

The BidExchange table manages mutual agreements between bidders.

- **id**: A unique identifier for each bid exchange (Primary Key).
- **salesBidder_id**: A foreign key referencing the Client table (user).
- **salesBidHeader_id:** A foreign key referencing the `BidHeader` table.
- **purchaseBidder_id**: A foreign key referencing the Client table (user).
- **purchaseBidHeader_id:** A foreign key referencing the BidHeader table.
- **agreedDeal**: A boolean flag indicating, if the deal is mutually agreed upon.
- **dealTimestamp**: The timestamp when the deal was made.

```
CREATE TABLE BidExchange --This table will allow bidders to exchange equipment
based on their bids.
(
    id SERIAL NOT NULL PRIMARY KEY,
    salesBidder_id INT NOT NULL, -- ID of the initial bidder (sales-bidder)
    salesBidHeader_id INT NOT NULL, -- ID of the sales offer header
    purchaseBidder_id INT NOT NULL, -- ID of the bidder making the purchase offer
    purchaseBidHeader_id INT NOT NULL, -- ID of the purchase offer header
    agreedDeal BOOLEAN DEFAULT FALSE, -- Is the the deal mutually agreed upon
    dealTimestamp TIMESTAMP NOT NULL, -- Timestamp when the deal was made

    FOREIGN KEY(salesBidder_id) REFERENCES User(id),
    FOREIGN KEY(salesBidHeader_id) REFERENCES BidHeader(id),
    FOREIGN KEY(purchaseBidder_id) REFERENCES User(id),
    FOREIGN KEY(purchaseBidHeader_id) REFERENCES BidHeader(id)
);
```

8. ImageMetadata Table

- **ImageMetadata**: Image and icon metadata-file can be linked to Gear, BidContent and Client tables via fk-links, which contains image definition metadata and the url to the image mediafile.

```
CREATE TABLE ImageMetadata

(

  id SERIAL NOT NULL PRIMARY KEY,

  imageName VARCHAR(100) NOT NULL,

  AxSize VARCHAR(10) NOT NULL,

  resolution VARCHAR(20) NOT NULL,

  image_url VARCHAR(255) NOT NULL -- Link to the actual image file

);
```

9. SoundClipMetadata Table

- **SoundClipMetadata**: Sound-samples can be linked to Gear and BidContent tables, which contains sound-clip definition metadata and the url to the sound-clip mediafile.

```
CREATE TABLE SoundClipMetadata

(

  id SERIAL NOT NULL PRIMARY KEY,

  clipName VARCHAR(100) NOT NULL,

  format VARCHAR(10) NOT NULL, -- e.g., '.wav', '.mp3', etc.

  quality VARCHAR(30) NOT NULL, -- e.g., '24-bit WAV', '196 kbps MP3', etc.

  clipLengthInSeconds FLOAT NOT NULL,
```

```
    isStereo BOOLEAN NOT NULL,
    sample_url VARCHAR(255) NOT NULL -- Link to the actual sound sample file
);
```

Appendix:

Database creation script:

```
CREATE TABLE Client
(
    id SERIAL NOT NULL PRIMARY KEY,
    userName VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(25) DEFAULT NULL,
    email VARCHAR(80) DEFAULT NULL,
    secondaryEmail VARCHAR(80) DEFAULT NULL,
    userIcon_Url VARCHAR(80) DEFAULT NULL,  -- Link to user's user's icon metadata file
    role_id INT NOT NULL,
    blocked BOOLEAN DEFAULT FALSE,
    latestBidHeader_id INT,

    FOREIGN KEY(role_id) REFERENCES Role(id),
    FOREIGN KEY(userIcon_url) REFERENCES ImageMetadata(id),
    FOREIGN KEY(latestBidHeader_id) REFERENCES BidHeader(id)
  );

CREATE TABLE Role
(
    id SERIAL NOT NULL PRIMARY KEY,
    roleName VARCHAR(80) NOT NULL UNIQUE,
    registeredUser BOOLEAN DEFAULT FALSE,
    sellerRole BOOLEAN DEFAULT FALSE,
    buyerRole BOOLEAN DEFAULT FALSE,
    adminRole BOOLEAN DEFAULT FALSE
  );

CREATE TABLE BidGenre
(
    id SERIAL NOT NULL PRIMARY KEY,
    bidGenreHeader VARCHAR(100) NOT NULL UNIQUE,
    user_id INT,
    LastBidHeader_id INT,
    numberOfBids INT NOT NULL,

    FOREIGN KEY(user_id) REFERENCES User(id),
```

```
    FOREIGN KEY(lastBidHeader_id) REFERENCES  BidHeader(id)
  );

CREATE TABLE BidHeader
(
    id SERIAL NOT NULL PRIMARY KEY,
    user_id INT,
    bidGenre_id INT,
    bidTypeCode INT,
    bidStatus INT,
    bidContent_id INT,
    bidHeaderText VARCHAR(100) NOT NULL UNIQUE,
    bidExchange1_id INT,
    bidExchange2_id INT,
    bidExchange3_id INT,
    bidExchange4_id INT,
    bidExchange5_id INT,
    bidStartedTime TIMESTAMP NOT NULL,
    bidEndingTime TIMESTAMP NOT NULL,
    hdrEditedTime TIMESTAMP NOT NULL,

    FOREIGN KEY(user_id) REFERENCES User(id),
    FOREIGN KEY(bidGenre_id) REFERENCES BidGenre(id),
    FOREIGN KEY(bidContent_id) REFERENCES BidContent(id),
    FOREIGN KEY(bidExchange1_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange2_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange3_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange4_id) REFERENCES BidExchange(id),
    FOREIGN KEY(bidExchange5_id) REFERENCES BidExchange(id)
    );


CREATE TABLE BidContent
(
    id SERIAL NOT NULL PRIMARY KEY,
    bidHeader_id INT,
    bidContent VARCHAR(500) NOT NULL,
    gear1_id INT,
    gear2_id INT,
    gear3_id INT,
    gear4_id INT,
    gear5_id INT,
    gear6_id INT,
    gear7_id INT,
    gear8_id INT,
```

```
    gear9_id INT,
    gear10_id INT,
    conEditedTime TIMESTAMP NOT NULL,
    bundleImage_id INT, -- Link to bundle image metadata file
    bundleSound_id INT, -- Links to bundle sound sample metadata file

    FOREIGN KEY(bidHeader_id) REFERENCES BidHeader(id),
    FOREIGN KEY(bundleImage_id) REFERENCES ImageMetadata(id),
    FOREIGN KEY(bundleSound_id) REFERENCES SoundClipMetadata(id),
    FOREIGN KEY(gear1_id) REFERENCES Gear(id),
    FOREIGN KEY(gear2_id) REFERENCES Gear(id),
    FOREIGN KEY(gear3_id) REFERENCES Gear(id),
    FOREIGN KEY(gear4_id) REFERENCES Gear(id),
    FOREIGN KEY(gear5_id) REFERENCES Gear(id),
    FOREIGN KEY(gear6_id) REFERENCES Gear(id),
    FOREIGN KEY(gear7_id) REFERENCES Gear(id),
    FOREIGN KEY(gear8_id) REFERENCES Gear(id),
    FOREIGN KEY(gear9_id) REFERENCES Gear(id),
    FOREIGN KEY(gear10_id) REFERENCES Gear(id)
);

CREATE TABLE Gear
(
  id SERIAL NOT NULL PRIMARY KEY,
  gearBrand VARCHAR(30) NOT NULL,
  gearCode INT,
  gearStatus INT,
  bidHeader_id INT,
  gearName VARCHAR(100) NOT NULL,
  gearDetails TEXT,
  gearStory TEXT,
  amountOffered INT NOT NULL,
  approxValue FLOAT default 0.00,
  user_id INT,
  comments TEXT,
  editedTime TIMESTAMP NOT NULL,
  gearImage_id INT,  -- Links to gear image metadata file
  soundClip_id INT, -- Links to gear sound sample metadata file

  FOREIGN KEY(user_id) REFERENCES User(id),
  FOREIGN KEY(bidHeader_id) REFERENCES BidHeader(id),
  FOREIGN KEY(gearImage_id) REFERENCES ImageMetadata(id),
  FOREIGN KEY(soundClip_id) REFERENCES SoundClipMetadata(id)
);
```

```
CREATE TABLE BidExchange --This table will allow bidders to exchange equipment based on
their bids.
(
    id SERIAL NOT NULL PRIMARY KEY,
    salesBidder_id INT NOT NULL, -- ID of the initial bidder (sales-bidder)
    salesBidHeader_id INT NOT NULL, -- ID of the sales offer header
    purchaseBidder_id INT NOT NULL, -- ID of the bidder making the purchase offer
    purchaseBidHeader_id INT NOT NULL, -- ID of the purchase offer header
    agreedDeal BOOLEAN DEFAULT FALSE, -- Indicates if the deal is mutually agreed upon
    dealTimestamp TIMESTAMP NOT NULL, -- Timestamp when the deal was made

    FOREIGN KEY(salesBidder_id) REFERENCES User(id),
    FOREIGN KEY(salesBidHeader_id) REFERENCES BidHeader(id),
    FOREIGN KEY(purchaseBidder_id) REFERENCES User(id)
    FOREIGN KEY(purchaseBidHeader_id) REFERENCES BidHeader(id),
);

CREATE TABLE ImageMetadata
(
    id SERIAL NOT NULL PRIMARY KEY,
    imageName VARCHAR(100) NOT NULL,
    AxSize VARCHAR(10) NOT NULL,
    resolution VARCHAR(20) NOT NULL,
    image_url VARCHAR(255) NOT NULL -- Link to the actual image file
);

CREATE TABLE SoundClipMetadata
(
    id SERIAL NOT NULL PRIMARY KEY,
    clipName VARCHAR(100) NOT NULL,
    format VARCHAR(10) NOT NULL, -- e.g., '.wav', '.mp3', etc.
    quality VARCHAR(30) NOT NULL, -- e.g., '24-bit WAV', '196 kbps MP3', etc.
    clipLengthInSeconds FLOAT NOT NULL,
    isStereo BOOLEAN NOT NULL,
    sample_url VARCHAR(255) NOT NULL -- Link to the actual sound sample file
);
-- SOME VIEWS FOR TESTING:
CREATE VIEW BidGenreList AS
SELECT bg.id, bg.bidGenreHeader, COUNT(bc.id) as count, MAX(bc.conEditedTime) AS latest
FROM db_bgefv4_schemav2.BidGenre bg
LEFT JOIN db_bgefv4_schemav2.BidHeader bhr ON (bg.id = bhr.bidGenre_id)
LEFT JOIN db_bgefv4_schemav2.BidContent bc ON (bhr.id = bc.bidHeader_id)
GROUP by bg.id, bg.bidGenreHeader;

CREATE VIEW BidList AS
```

```
SELECT bh.id, bh.user_id, bh.bidGenre_id, bh. bidTypeCode, bh.bidStatus, bh.bidHeaderText,
COUNT(*) AS count, MIN(bc.coneditedTime) AS edited,
MAX(bc.coneditedTime) AS latest
FROM db_bgefv4_schemav2.BidHeader bh
LEFT JOIN db_bgefv4_schemav2.BidContent bc ON (bh.id = bc.bidHeader_id)
GROUP BY bh.id, bh.bidHeaderText, bh.user_id, bh.bidGenre_id
ORDER BY MAX(bc.coneditedTime) DESC;
```