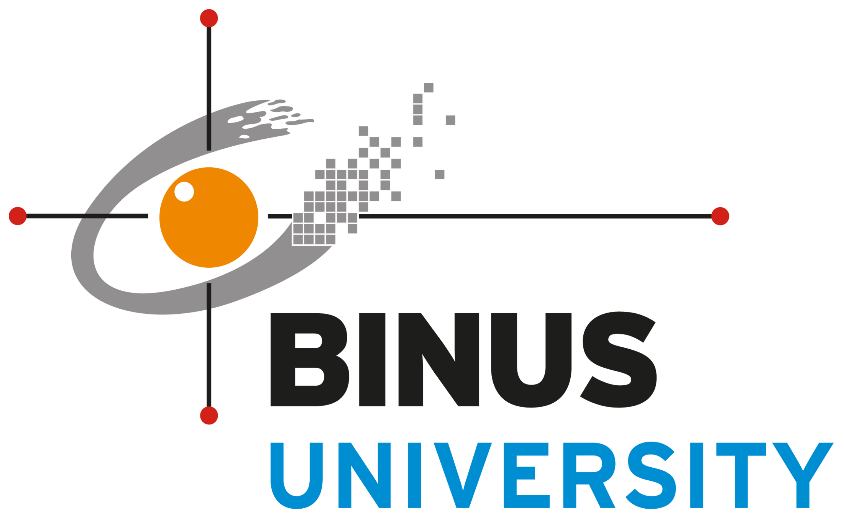
**ISYS6172 - Database Design**



**Group 8:**

Edward Rezzky (2301868346) – Head

Michael Albert Purnama (2301877716) – Vice Head 1

Yoga Danuarta (2301847354) – Vice Head 2

Andre Panji (2301909346)

Muhammad Alif Nadhirrahman (2301933414)

Putra Willianto (2301893764)

Reinaldo Adrianus (2301847700)

Valentino Widjaya (2301864700)

**Case**: JBA – Toko Pelelangan

**Name of Database**: YukBid

**Bina Nusantara University**

**2021**

**Store Profile**

YukBid is an auction-based buying and selling application that can be used easily on various platforms in Indonesia. Yukbid is here to facilitate human activities in conducting auction activities where there is often news about online auction fraud in Indonesia. With YukBid, convenience comes in conducting the auction process legally, safely, and reliably without the slightest element of fraud. Often, we think that auctions are only done to get items that are categorized as expensive items, but YukBid is here to satisfy everyone's curiosity about the auction process. The goods being auctioned here are quite diverse and have been categorized according to the type of goods.

**Mission Statement**

The reason YukBid is here is to bring a new experience to the interactive digital world of auctions. It's not just about selling and buying stuff; we're bringing you an experience that has never been tried before by many. To support this, we will manage various data stored in YukBid using a good Database System. The system will assist us in storing data about auction items, info for each account, info on each transaction, track record of each account, and so on. The data will be stored to minimize the occurrence of crashes or bugs in this application.

**Mission Objective**

To manage (insert, update and delete) data on Staff

To manage (insert, update and delete) data on Seller

To manage (insert, update and delete) data on Bidder

To manage (insert, update and delete) data on Lelang

To manage (insert, update and delete) data in Category

To manage (insert, update, and delete) data on Items

To display data search results on Staff

To display data search results on Seller

To display data search results on the Bidder

To display data search results on Lelang

To display data search results in Category

To display data search results on Item

To view the status of an Item

To view the status of the Lelang

To see Seller or Bidder status

To make a Staff report

To create an Items report

To create an Lelang report

**System Boundary**

Chart, diagram, box and whisker chart

Description automatically generated

**User Views for YukBid Database System**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data** | **Director** | **Manager** | **Supervisor** | **Admin** | **Database Admin** | **Software Developer** |
| **Item** | **X** | **X** |  | **X** | **X** |  |
| **Staff** | **X** | **X** | **X** | **X** | **X** | **X** |
| **Item's Owner** | **X** | **X** | **X** | **X** | **X** |  |
| **Auctioneer** | **X** | **X** | **X** | **X** | **X** |  |
| **Item Viewing** |  |  |  | **X** | **X** |  |

**Interviews Questions:**

Top:

1. What is the estimated budget needed for the YukBid Application itself?

2. Is a high level of security required?

3. Who can access this database?

4. Which platforms (Website/Mobile) do you think are suitable for this Yukbid application?

5. What kind of flow do you want?

Middle:

1. What are the attributes needed in the formation of this database?

2. What software and design tools will be used?

3. What are the steps to be taken in an analytical project?

4. What programming language will be used?

5. What is the Software Development Life Cycle?

6. What approach will you use for this database project?

Bottoms:

1. Are there features or services that are difficult to understand?

2. In your opinion, for the YukBid application itself, what features should be added or improved?

3. What systems can be put in place to make auctions easier?

4. What kind of goods would you like to buy/sell?

5. Which offline or online auction can be more effective?

**Fact Finding Technique**

Observation:

With this technique, we can directly participate in auction activities on YukBid so that we can immediately see and experience the application that we have developed. After we use the application, we can see the advantages and disadvantages of our application.

Research:

With this technique, we did some research on the internet related to buying and selling using auctions. We are looking for several sources related to how a good auction system is and what kind of auction people want and what kind of goods people are looking for.

Questionnaires:

We distributed questionnaires to users regarding the YukBid application. We asked some questions about the system used in this YukBid application. With this technique, we can also collect the wishes of many customers in a short time and with little cost**.**

**CONCEPTUAL DATABASE DESIGN**

**Step 1.1 Identify Entity Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Entity Name* | *Description* | *Aliases* | *Occurrence* |  |
| Staff | General term of describing all staff employed by YukBid. | Staff | Each member of staff works at one branch. |  |
|  |
| Seller | General term of describing all user who want to open an auction in YukBid. | Seller | Each auction has a unique AuctionID and will not be the same as any other auction. Each user can open multiple auctions. The auction can be seen and followed by all users. |  |
|  |
| Bidder | General term of describing all user who want to bid in YukBid auction. | Bidder | Each user can participate in the auction that has been opened. Each user can also bid multiple times, but the bid submitted must be higher than the last bid recorded. The user will be declared a winner if successfully reaches the highest bid. |  |
|  |
| Item | General term of describing all the items to be auctioned at YukBid auction. | Item | Each item to be auctioned must match the description and title given. Items that are going to be auctioned are not allowed to be recalled. Sharp objects, drugs, alcoholic beverages, and other dangerous goods will not be allowed to enter the auction. |  |
|  |
| Category | General term of describing all the item category in YukBid auction. | Category | Each item will have one category, it will not have multiple categories. Each category can be related to other categories. Users can search for items by category. |  |
|  |
| Lelang | General term of describing all the transaction detail in YukBid auction. | Lelang | Every transaction will be recorded by the system. Transactions can be made if the bidder is successful in winning the auction. Transactions will be recorded based on the time when the bidder wins the auction. |  |
|  |

**Step 1.2 Identify Relationship Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity Name | Multiplicity | Relationship | Multiplicity | Entity Name |
| Seller | 1..1  1…1 | Sell  Provide | 1..\*  1..\* | Item  Lelang |
| Bidder | 0..\* | Bid | 1..1 | Lelang |
| Staff | 0..\* | Manage | 1..1 | Lelang |
| Item | 1..1 | Has | 0..\* | Category |

**Step 1.3 Identify and Associate Attributes with Entity or Relationship Types**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Entity Name | Attributes | Description | Data Type & Length | Nulls | Multi-valued |
| Bidder | BidderID | Uniquely identifies a bidder | 6 variable characters (BIDxxx) | No | No |
| BidderName | Name of bidder | 255 characters | No | No |
| BidderPhone | Phone number of bidders | 12 variable characters | No | No |
| BidderEmail | Email address of bidder | 255 variable characters | No | No |
| BidderAddress | Address of bidder | 255 variable characters | No | No |
| BidPrice | Price of bidder to bid an item | Integer | No | No |
| Seller | SellerID | Uniquely identifies a seller | 6 variable characters (SELxxx) | No | No |
| SellerName | Name of seller | 255 characters | No | No |
| SellerPhone | Phone number of sellers | 12 variable characters | No | No |
| SellerEmail | Email address of seller | 255 variable characters | No | No |
| SellerAddress | Address of seller | 255 variable characters | No | No |
| Staff | StaffID | Uniquely identifies of admin or staff | 6 variable characters (ADMxxx) | No | No |
| StaffName | Name of admin or staff | 255 characters | No | No |
| StaffPhone | Phone number of admin or staff | 12 variable characters | Yes | No |
| StaffEmail | Email address of admin or staff | 255 variable characters | No | No |
| StaffAddress | Address of admin or staff | 255 variable characters | Yes | No |
| Category | CategoryID | Uniquely identifies of category | 6 variable characters (CATxxx) | No | No |
| CategoryName | Name of category | 255 characters | No | No |
| Item | ItemID | Uniquely identifies of item | 6 variable characters (ITMxxx) | No | No |
| CategoryID | Uniquely identifies of category (refers from Category table) | 6 variable characters (CATxxx) | No | No |
| ItemName | Name of item | 255 variable characters | No | No |
| Description | Description of item | 255 characters | No | No |
| StartItemPrice | Start price for bidders to bid an item | Integer | No | No |
| BuyNowPrice | Buy now price for bidders to bid an item | Integer | No | No |
| Lelang | LelangID | Uniquely identifies of transaction | 6 variable characters (TRSxxx) | No | No |
| StaffID | Uniquely identifies of admin or staff (refers from Admin table) | 6 variable characters (STFxxx) | No | No |
| BidderID | Uniquely identifies a bidder (refers from Bidder table) | 6 variable characters (BIDxxx) | No | No |
| SellerID | Uniquely identifies a seller (refers from Seller table) | 6 variable characters (SELxxx) | No | No |
| Date | Date of transaction | Date | No | No |

**Step 1.4 Determine attribute domains**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Name** | **Attributes** | **Data Type** | **Length** |
| Bidder | BidderID | Variable Characters (BIDxxx) | 6 |
| BidderName | Characters | 255 |
| BidderPhone | Variable Characters | 12 |
| BidderEmail | Variable Characters | 255 |
| BidderAddress | Variable Characters | 255 |
| BidPrice | Integer | 35 |
| Seller | SellerID | Variable Characters (SELxxx) | 6 |
| SellerName | Characters | 255 |
| SellerPhone | Variable Characters | 12 |
| SellerEmail | Variable Characters | 255 |
| SellerAddress | Variable Characters | 255 |
| Staff | StaffID | Variable Characters (STFxxx) | 6 |
| StaffName | Characters | 255 |
| StaffPhone | Variable Characters | 12 |
| StaffEmail | Variable Characters | 255 |
| StaffAddress | Variable Characters | 255 |
| Category | CategoryID | Variable Characters (CATxxx) | 6 |
| CategoryName | Characters | 255 |
| Item | ItemID | Variable Characters (ITMxxx) | 6 |
| CategoryID | Variable Characters (CATxxx) | 6 |
| ItemName | Variable Characters | 255 |
| Description | Characters | 255 |
| StartItemPrice | Integer | 35 |
| BuyNowPrice | Integer | 35 |
| Lelang | LelangID | Variable Characters (TRSxxx) | 6 |
| AdminID | Variable Characters (STFxxx) | 6 |
| BidderID | Variable Characters (BIDxxx) | 6 |
| SellerID | Variable Characters (SELxxx) | 6 |
| Date | Date | - |

**Step 1.5 Determine Candidate, Primary, and Alternate Key Attributes**

**Seller:**

Candidate Key= SellerId, SellerPhone, SellerEmail

Primary Key= SellerId

Alternate Key= SellerPhone, SellerEmail

**Lelang:**

Candidate Key= LelangId

Primary Key= LelangId

Alternate Key= -

**Bidder:**

Candidate Key= BidderId, BidderPhone, BidderEmail

Primary Key= BidderId

Alternate Key= BidderPhone, BidderEmail

**Staff:**

Candidate Key= StaffId, StaffPhone, StaffEmail

Primary Key= StaffId

Alternate Key= StaffPhone, StaffEmail

**Item:**

Candidate Key=ItemId, ItemName, Description

Primary Key=ItemId

Alternate Key= ItemName, Description

**Category:**

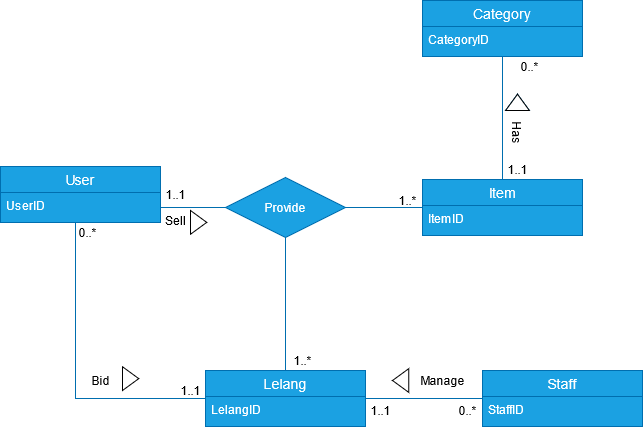
Candidate Key=CategoryId, CategoryName

Primary Key=CategoryId

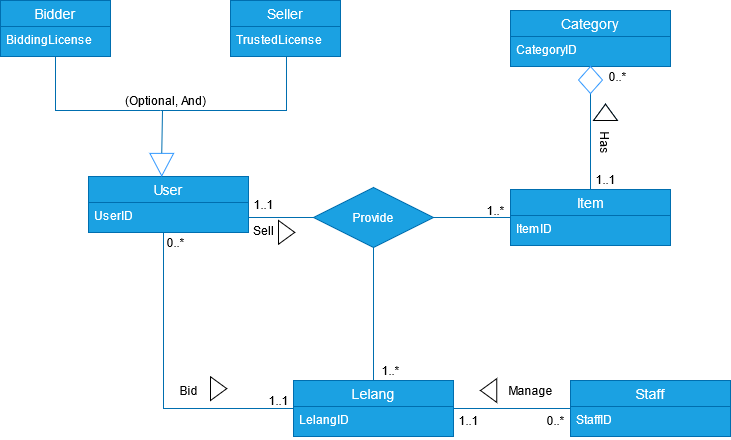
Alternate Key= CategoryName

**Step 1.6 Consider Use of Enhanced Modeling Concepts (optional step)**

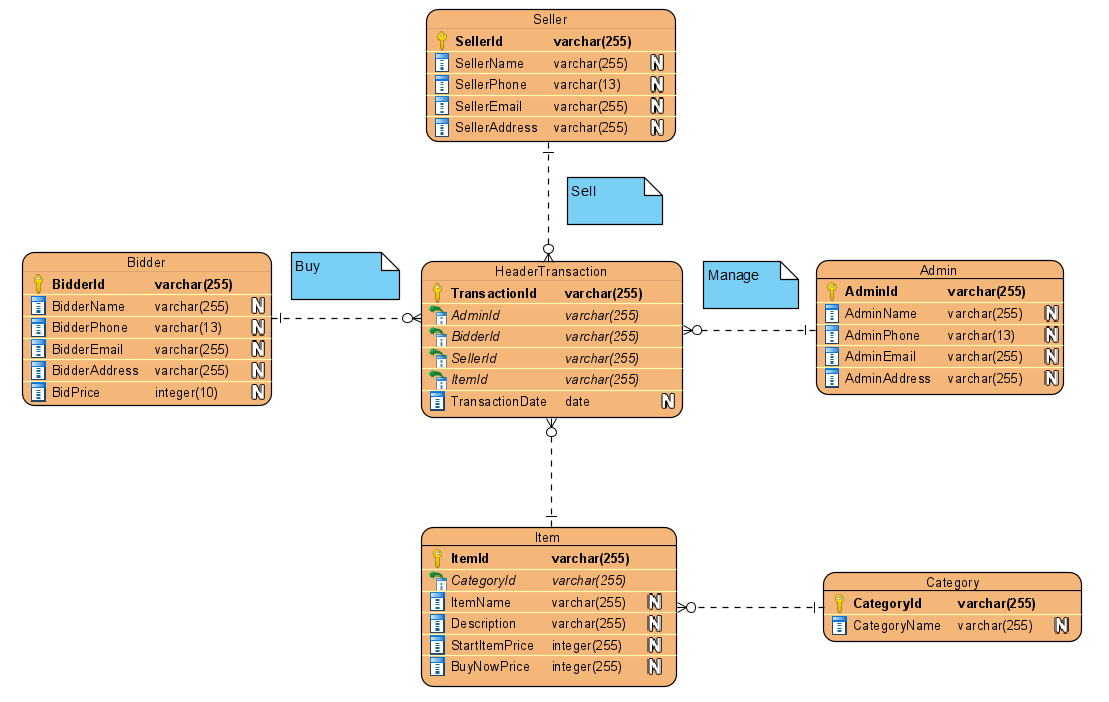
Enhanced Relationship Model



Enhanced Entity Relationship

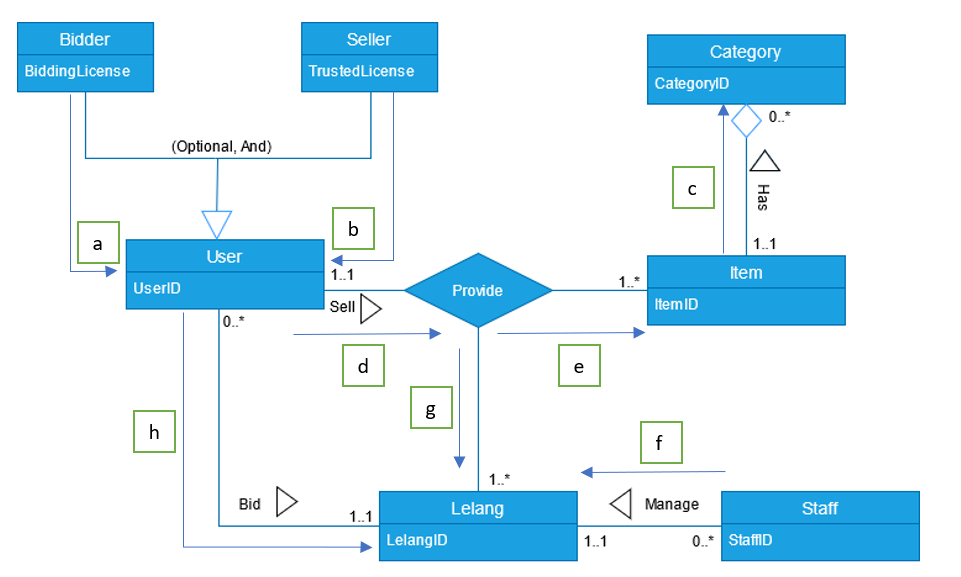


**Step 1.7   Check Model for Redundancy**



There’s no redundant in Entity Relationship Diagram.

**Step 1.8   Validate Conceptual Model Against User Transactions**



**Step 1.9   Review Conceptual Data Model with User**

At the conceptual stage of the data model, datas are needed to build a model relationship. There are many steps involved in designing a data model and other documentation. The table is also checked for data anomalies and redundancy, if something is not appropriate, it is necessary to change and repeat the previous stages. The result of the conceptual database is a data model that has entities, attributes, relationships, and problem domains. The data model is also implemented on a user who performs a transaction.

**LOGICAL DATABASE DESIGN**

**Step 2.1 Derive Relations for Logical Data Model**

1. **Strong Entity Types**

Bidder (BidderId[PK], BidderName, BidderPhone, BidderEmail, BidderAddress, BidPrice)

Seller (SellerId[PK], SellerName, SellerPhone, SellerEmail, SellerAddress)

Staff (StaffId[PK], StaffName, StaffPhone, StaffEmail, StaffAddress)

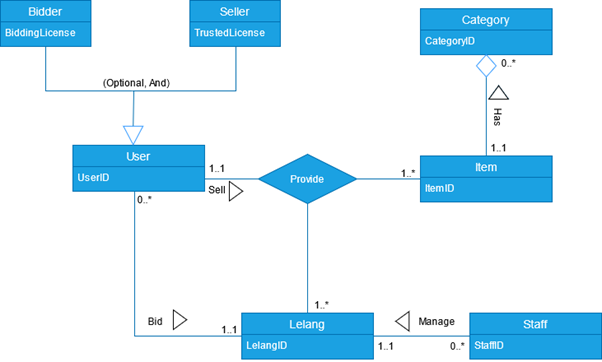
Category (CategoryId[PK], CategoryName)

1. **Weak Entity Types**

Lelang(LelangId[PK], AdminId[FK], BidderId[FK], SellerId[FK],  ItemId[FK], TransactionDate)

Item (ItemId[PK], CategoryId[FK], ItemName, Description, StartItemPrice, BuyNowPrice)

1. **One-to-many (1:\*) Binary Relationship Types**



Lelang 1…\* --------------------- Provide

Item     1…\* --------------------- Provide

1. **One-to-one (1:1) Binary Relationship Types**

***Mandatory* Participation on *One* Side of a 1:1 Relationship**

 Item     1...1  -------------------- Category 0…\*

 User     1...1  -------------------- Provide

 Lelang  1...1 -------------------- User 0…\*

 Lelang  1...1 -------------------- Staff 0…\*

1. **One-to-one (1:1) Recursive Relationships**

We do not have recursive 1-1 relationship.

1. **Superclass/Subclass Relationship Types**

**Option 2 – Optional, Nondisjoint**

User (UserID)

Primary Key UserID

UserDetails (UserID, BiddingLicense, TrustedLicense)

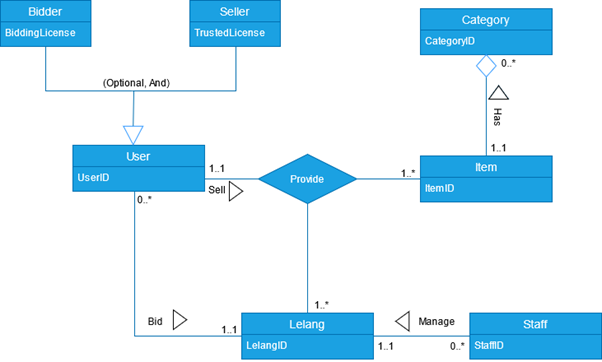
Primary Key UserID

Foreign Key UserID references User (UserID)

**7. Many-to-Many (\*:\*) Binary Relationship Types**

We don’t have many-to-many \*:\* binary relationship types.

**8.    Complex Relationship Types**

****

**Provide** (ProvideId, UserId, ItemId, LelangId, RegistrationDate)

**Primary Key** ProvideId

**Foreign Key** UserId, ItemId, LelangId

**9.     Multi-Valued Attributes**

In the ER Model, an admin is assumed to have more than 1 telephone number to contact the seller or bidder. Therefore, multi-valued attributes will be formed.

  StaffPhone (StaffPhoneId, StaffPhone)

Primary Key StaffPhoneId

**Step 2.2 Validate Relations Using Normalization**

**Normalization**

**UNF:**

Lelang (LelangId, LelangDate, BidderId, BidderName, BidderPhone, BidderEmail, BidderAddress, BidPrice, SellerId, SellerName, SellerPhone, SellerEmail, SellerAddress, StaffId, StaffName, StaffPhoneId, StaffEmail, StaffAddress, {ItemId, ItemName, Description, StartItemPrice, BuyNowPrice, CategoryId, CategoryName})

Repeated data: {}

**1NF:**

Lelang (**LelangId[PK]**, LelangDate, BidderId, BidderName, BidderPhone, BidderEmail, BidderAddress, BidPrice, SellerId, SellerName, SellerPhone, SellerEmail, SellerAddress, StaffId, StaffName, StaffPhoneId, StaffEmail, StaffAddress)

Item (**ItemId[PK]**, *LelangId[FK]*, ItemName, Description, StartItemPrice, BuyNowPrice, CategoryId, CategoryName)

Bold = PK

Italic = FK

**2NF:**

Lelang (**LelangId[PK]**, LelangDate, BidderId, BidderName, BidderPhone, BidderEmail, BidderAddress, BidPrice, SellerId, SellerName, SellerPhone, SellerEmail, SellerAddress, StaffId, StaffName, StaffPhoneId, StaffEmail, StaffAddress)

Item (**ItemId[PK]**, *StaffId[FK]*, ItemName, Description, StartItemPrice, BuyNowPrice, CategoryId, CategoryName)

**3NF:**

Lelang (**LelangId[PK]**, *BidderId[FK]*, *SellerId[FK]*, *StaffId[FK]*, *ItemId[FK]*, LelangDate)

Item (**ItemId[PK]**, *CategoryId[FK]*, ItemName, Description, StartItemPrice, BuyNowPrice)

Bidder (**BidderId[PK]**, BidderName, BidderPhone, BidderEmail, BidderAddress, BidPrice)

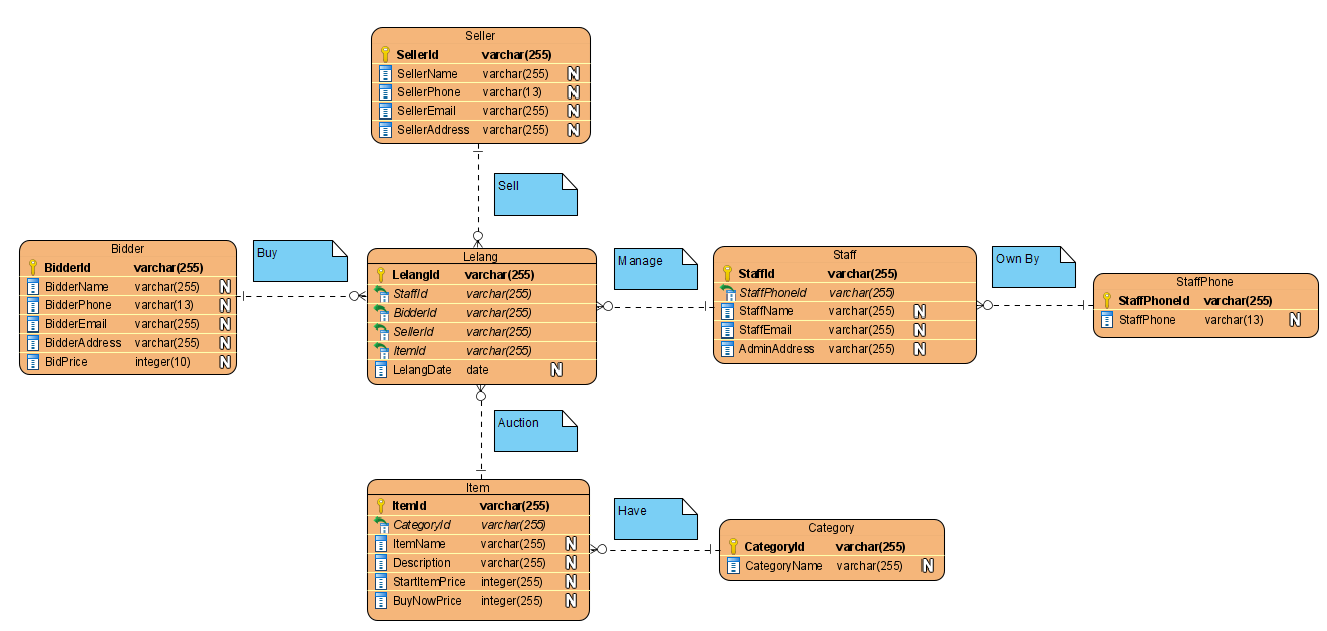
Seller (**SellerId[PK]**, SellerName, SellerPhone, SellerEmail, SellerAddress)

Staff (**StaffId[PK]**, *StaffPhoneId[FK]*, StaffName, StaffEmail, StaffAddress)

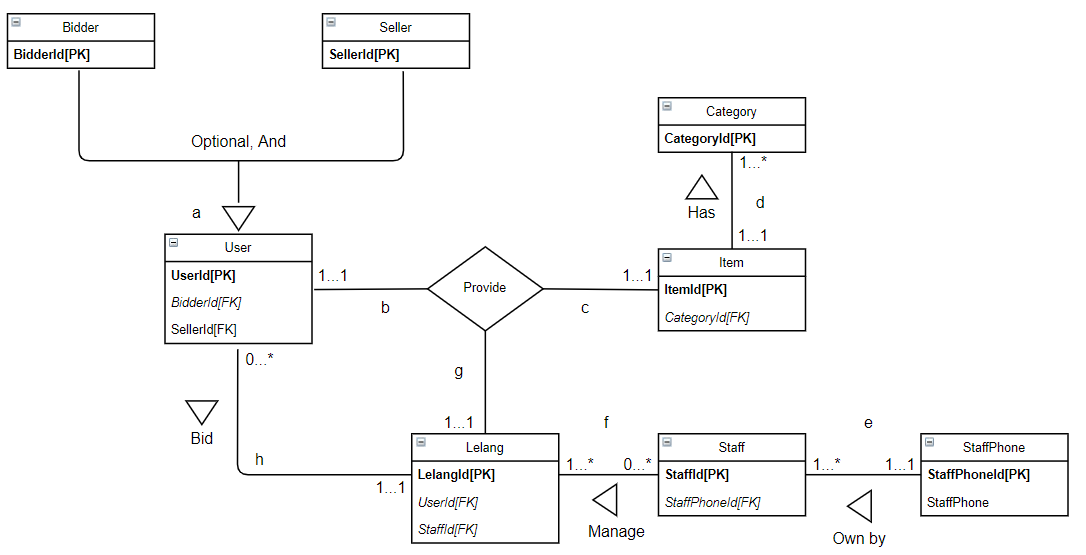
StaffPhone (**StaffPhoneId[PK]**, StaffPhone)

Category (**CategoryId[PK],** CategoryName)

**Entity Relationship Diagram**



**Step 2.3 Validate Relations Against User Transactions**



Relationship validation against user transactions does not have problems. All stages of the transaction are in order. Attributes and ER Model also don’t have a problem.

Flow in user transactions:

a. Users are divided into two parts, namely bidders and sellers.

b. Each user has its own role. The bidder can conduct the auction while the seller will auction the item.

c. Items to be auctioned will be selected by the user.

d. Each item has its own category.

e. The staff will assist in the auction process where there are many staff telephone numbers.

f. Staff who assist the auction will confirm user data, goods, and other needs.

g. The user will make a transaction and it has done.

**Step 2.4 Check Integrity Constraints**

**Bidder** (BidderId, BidderName, BidderPhone, BidderEmail, BidderAddress, BidPrice)

PRIMARY KEY: BidderId

FOREIGN KEY: -

ALTERNATE KEY: BidderName, BidderPhone

**Seller** (SellerId, SellerName, SellerPhone, SelerEmail, SellerAddress)

PRIMARY KEY: SellerId

FOREIGN KEY: -

ALTERNATE KEY: SellerName, SellerPhone

**Staff** (StaffId, StaffPhoneId, StaffName, StaffEmail, StaffAddress)

PRIMARY KEY: StaffId

FOREIGN KEY: StaffPhoneId REFERENCES StaffPhone(StaffPhoneId) ON UPDATE CASCADE ON DELETE CASCADE

ALTERNATE KEY: StaffName

**Category** (CategoryId, CategoryName)

PRIMARY KEY: CategoryId

FOREIGN KEY: -

ALTERNATE KEY: CategoryName

**Item** (ItemId, CategoryId, ItemName, Description, StartItemPrice, BuyNowPrice)

PRIMARY KEY: ItemId

FOREIGN KEY: CategoryId REFERENCES Category(CategoryID) ON UPDATE CASCADE ON DELETE CASCADE

ALTERNATE KEY: ItemName

**Lelang** (LelangId, StaffId, BidderId, SellerId, Date)

PRIMARY KEY: LelangId

FOREIGN KEY: StaffId REFERENCES Staff(StaffId) ON UPDATE CASCADE ON DELETE CASCADE

FOREIGN KEY: BidderId REFERENCES Bidder(BidderId) ON UPDATE CASCADE ON DELETE CASCADE

FOREIGN KEY: SellerId REFERENCES Seller(SellerId) ON UPDATE CASCADE ON DELETE CASCADE

**User** (UserId, SellerId, BidderId)

PRIMARY KEY: UserId

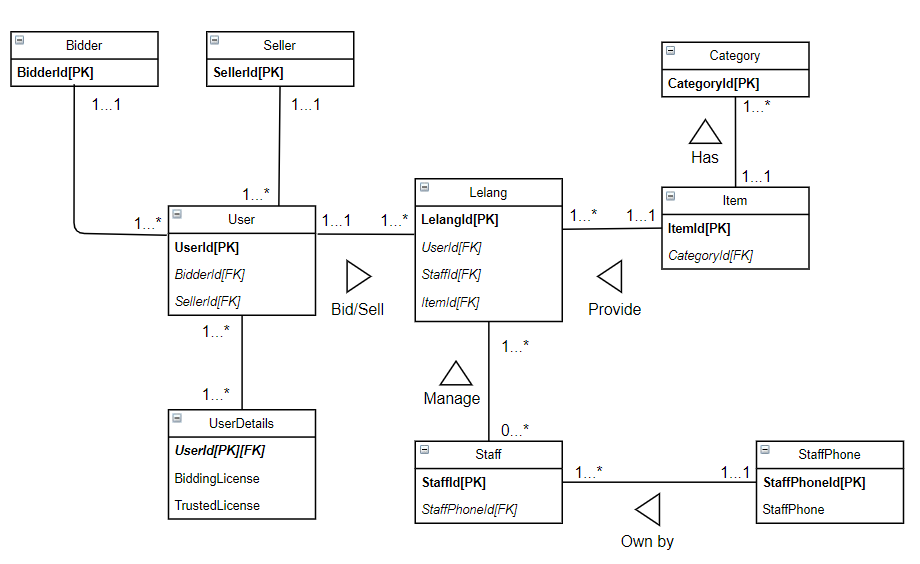
FOREIGN KEY: SellerId REFERENCES Seller(SellerId) ON UPDATE CASCADE ON DELETE CASCADE

FOREIGN KEY: BidderId REFERENCES Bidder(BidderId) ON UPDATE CASCADE ON DELETE CASCADE

**StaffPhone** (StaffPhoneId, StaffPhone)

PRIMARY KEY: StaffPhoneId

**Step 2.5 Review Logical Data Model with User**



The database is designed for e-commerce that is engaged in auctions. e-commerce will facilitate all auction activities that occur in the application. Users can open auctions or participate in existing auctions. Each item will be grouped by category. Staff will supervise all running activities. The following is the workflow of the database:

Workflow for bid transaction with user:

a. Users who want to bid items are needed to search a category of item.

b. If the user gets the item, then staff will maintain and help to provide the bid process.

c. Bid process contains a transaction with the bidder and seller.

d. A transaction can be said to be complete when the bidder reaches the highest agreement with the auctioneer.

Workflow for registering item:

a. Users who want to open auctions are needed to prepare items.

b. The item will be registered after the staff checks the items.

c. Users set the time how long the auction is running.

d. A transaction can be said to be complete when the bidder reaches the highest agreement with the auctioneer.

**Step 2.6 Merge Logical Data Models into Global Model (optional step)**

**Bidder** (BidderId, BidderName, BidderPhone, BidderEmail, BidderAddress, BidPrice)

PRIMARY KEY: BidderId

FOREIGN KEY: -

ALTERNATE KEY: BidderName, BidderPhone

**Seller** (SellerId, SellerName, SellerPhone, SelerEmail, SellerAddress)

PRIMARY KEY: SellerId

FOREIGN KEY: -

ALTERNATE KEY: SellerName, SellerPhone

**Staff** (StaffId, StaffPhoneId, StaffName, StaffEmail, StaffAddress)

PRIMARY KEY: StaffId

FOREIGN KEY: StaffPhoneId REFERENCES StaffPhone(StaffPhoneId) ON UPDATE CASCADE ON DELETE CASCADE

ALTERNATE KEY: StaffName

**Category** (CategoryId, CategoryName)

PRIMARY KEY: CategoryId

FOREIGN KEY: -

ALTERNATE KEY: CategoryName

**Item** (ItemId, CategoryId, ItemName, Description, StartItemPrice, BuyNowPrice)

PRIMARY KEY: ItemId

FOREIGN KEY: CategoryId REFERENCES Category(CategoryID) ON UPDATE CASCADE ON DELETE CASCADE

ALTERNATE KEY: ItemName

**Lelang** (LelangId, StaffId, BidderId, SellerId, Date)

PRIMARY KEY: LelangId

FOREIGN KEY: StaffId REFERENCES Staff(StaffId) ON UPDATE CASCADE ON DELETE CASCADE

FOREIGN KEY: BidderId REFERENCES Bidder(BidderId) ON UPDATE CASCADE ON DELETE CASCADE

FOREIGN KEY: SellerId REFERENCES Seller(SellerId) ON UPDATE CASCADE ON DELETE CASCADE

**User** (UserId, SellerId, BidderId)

PRIMARY KEY: UserId

FOREIGN KEY: SellerId REFERENCES Seller(SellerId) ON UPDATE CASCADE ON DELETE CASCADE

FOREIGN KEY: BidderId REFERENCES Bidder(BidderId) ON UPDATE CASCADE ON DELETE CASCADE

**StaffPhone** (StaffPhoneId, StaffPhone)

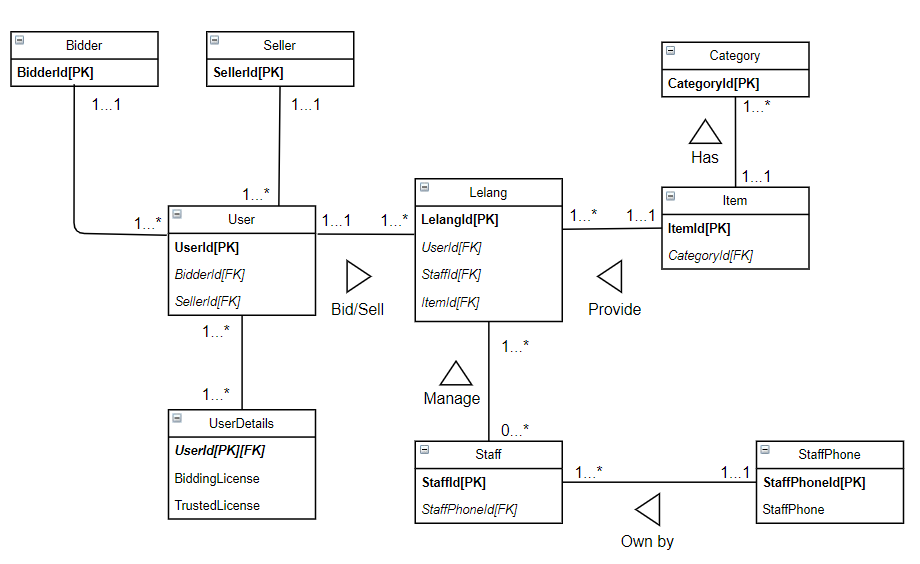
PRIMARY KEY: StaffPhoneId

**UserDetails** (UserId, BiddingLicense, TrustedLicense)

PRIMARY KEY: UserId

FOREIGN KEY: UserId REFERENCES User(UserId) ON UPDATE CASCADE ON DELETE CASCADE

**Global Relation Diagram for *YukBid***



**Step 2.7   Check for Future Growth**

This logical data model can continue to run and develop over time. Judging from the data model, logical results are non-rigid and changeable. What this means is that the logical data model supports possible future developments. It has long durability and does not require rework in creating a data model. This will make it easier for YukBid to continue to develop the database and maintain user transactions. Changes to terms and conditions depend on the user. So, in conclusion, the YukBid logical data model can survive and develop for the future.

**PHYSICAL DATABASE DESIGN**

**Step 4** Design File Organizations and Indexes

In this step, it is necessary to manage file organization and grouping files based on certain indexes that function to improve performance.

**Step 4.1** Analyze Transactions

This step is useful to see how big the transaction we are doing. Transactions will affect performance. Then it will be seen what transactions are often carried out at a certain time and become the main transaction.

There are two things to do, namely the transaction/relation cross-reference matrix to analyze transactions and the transaction usage map to determine frequently used relationships.

**Cross-referencing Transactions and Relations**

Sample bid transaction with user:

A. Users who want to bid items are needed to search a category of item.

B. If the user gets the item, then staff will maintain and help to provide the bid process.

C. Bid process contains a transaction with the bidder and seller. D. A transaction can be said to be complete when the bidder reaches the highest agreement with the auctioneer.

Sample for registering item:

E. Users who want to open auctions are needed to prepare items.

F. The item will be registered after the staff checks the items.

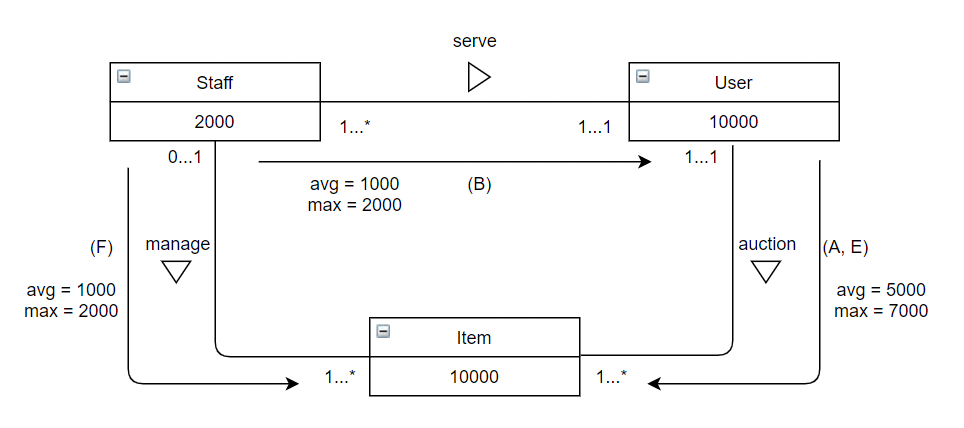
G. Users set the time how long the auction is running. H. A transaction can be said to be complete when the bidder reaches the highest agreement with the auctioneer.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Transaction Relation | A  I R U D | B  I R U D | C  I R U D | D  I R U D | E  I R U D | F  I R U D | G  I R U D | H  I R U D |
| Staff | X | X X | X | X | X | X X X | X X | X X |
| StaffPhone |  |  |  |  |  |  |  |  |
| Lelang |  |  |  |  |  |  |  |  |
| Item | X | X | X | X X | X X | X X | X | X |
| Category |  |  |  |  |  |  |  |  |
| User | X X X |  | X X X | X | X |  | X X | X |
| UserDetails |  |  |  |  |  |  |  |  |
| Seller |  |  | X X | X | X |  | X X | X |
| Bidder | X X X |  | X X X | X |  |  |  | X |

I = Insert; R = Read; U = Update; D = Delete

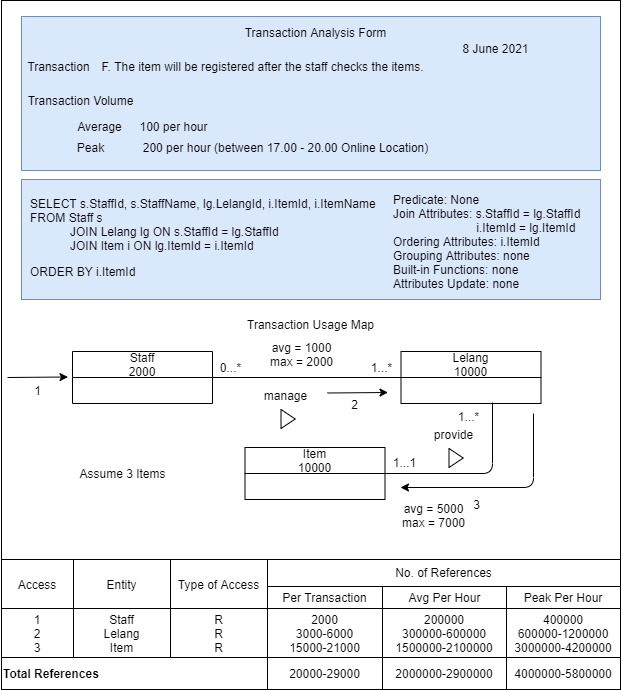
At this stage of cross-referencing transactions and relations, sample transactions are made from the bidding process and registering items. Then the transaction is entered into a table that contains relationships between entities and each entity can insert, read, update, and delete. The staff takes an important role where at every step he can read updates and deletes. Users, sellers, and bidders can insert, read, and update at certain steps.

**Relations Transaction Usage Map**



The Usage Map represents what entities are frequently used and appear. There are 3 entities, namely staff, users, and items where each entity has its quantity and the average and maximum number for each relationship. For example, the relationship between staff and users has an average of 1000 and a maximum of 2000 staff who will serve each existing user.

**Transaction Analysis Form**



In the form of transaction analysis, one of the transactions made in the cross-reference transaction is displayed. There is a transaction volume with an average of 100 transactions per hour and the peak is 200 transactions per hour in certain hours. Then displayed queries from existing transactions to select staff, auctions, and items. Displayed indication of the predicate none because there is no condition. The join attributes are StaffId and ItemId. The sort attribute is ItemId in ascending order. For attribute grouping, built-in functions, and attribute updates are none. Then a map of the use of transactions from existing transactions is also displayed. Consists of 3 entities with access type are read. It also displays the amount per transaction, average, hourly peak, and total references.

**Step 4.2: Choose File Organizations**

The selection of a file organization is useful for increasing the efficiency of entity relationships. We are targeting a SQL Server DBMS for our database case. We also choose a **sequential file organization** type with a pile file method and a sorted file when the data comes directly. This type is fast and efficient to handle large amounts of data in the YukBid database with a simple design. Sequential files can record every auction transaction properly and can be used to report auction data statistics on YukBid.

Diagram, table

Description automatically generated

Reference <https://static.javatpoint.com/dbms/images/dbms-sequential-file-organization_4.png>

**Step 4.3: Choose Indexes**

For index selection, we choose a **clustering index** based on attributes that are frequently used and appear in transactions. Clustering is more often used in join operations. As an example:

CREATE UNIQUE INDEX LelangIdIndex ON Lelang(LelangId);

Create an index with the name LelangIdIndex on the Lelang table with the LelangId attribute.

Implementation is:

SELECT l.LelangId, s.StaffId, s.StaffName

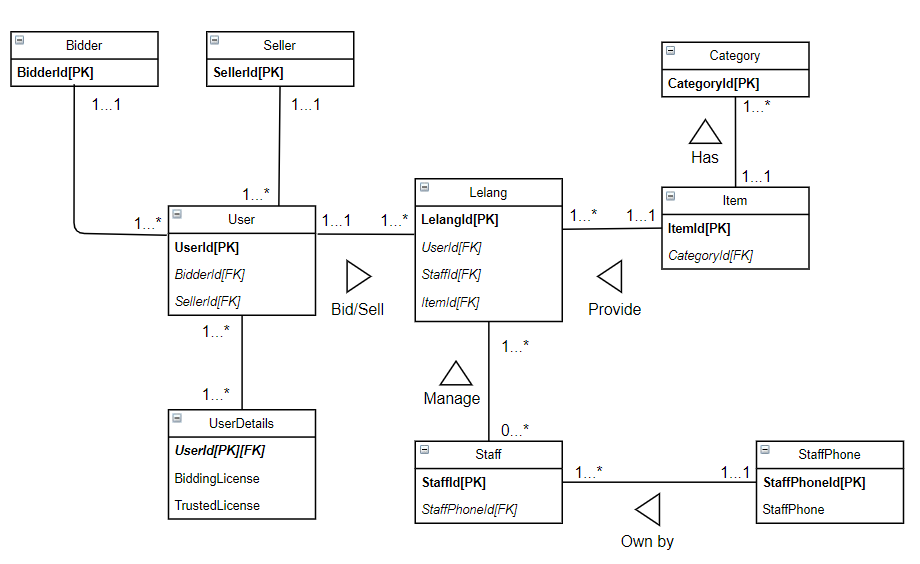
FROM Lelang l

JOIN Staff s ON l.StaffId = s.StaffId

WHERE l.LelangId = 'LI001'

The Lelang and Staff tables will be clustered where there is a join condition.

**MONITORING AND TUNING THE OPERATIONAL SYSTEM**



**Denormalization**

**Step 7.1 Combining 1:1 Relationships**

We don’t have 1:1 relationship.

**Step 7.2 Duplicating Non-key Attributes in 1:\* Relationships to Reduce Joins**

Duplicating Item table to Lelang table

**Lelang**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LelangId | StaffId | BidderId | SellerId | Date | ItemName |
| TRS001 | STF001 | BID001 | SEL001 | 6/14/2021 | PlayStation 5 |

**Step 7.3 Duplicating FK Attributes in 1:\* Relationship to Reduce Joins**

Duplicating ItemId on Lelang table to Staff table

**Staff**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| StaffId | StaffPhoneId | StaffName | StaffEmail | StaffAddress | ItemId |
| STF001 | SPH001 | Andy Deny | andy@gmail.com | Jl U, Kemanggisan | ITM001 |

**Step 7.4 Duplicating Attributes in \*:\* Relationships to Reduce Joins**

We don’t have \*:\* relationship.

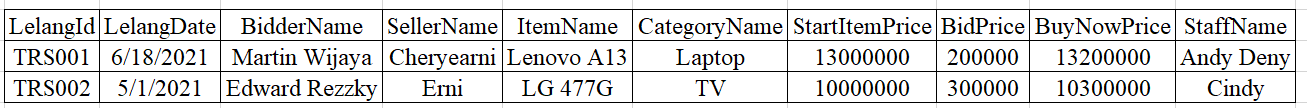
**Step 7.5 Introducing Repeating Groups**

**Staff**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| StaffId | StaffPhoneId | StaffName | StaffEmail | StaffAddress | StaffPhone1 | StaffPhone2 |
| STF001 | SPH001 | Andy Deny | andy@gmail.com | Jl U, Kemanggisan | 0822-2038-3219 | 0898-3476-6312 |
| STF002 | SPH002 | Cindy | cindy@gmail.com | JL A, Kemanggisan |  | 0818-7364-8766 |

**Step 7.6 Creating Extract Tables**

LelangTransaction Table



**Step 7.7 Partitioning Relations**

We are partitioning based on LelangTransaction table.

Table

Description automatically generated

We choose a horizontal partitioning to dividing by month of auction transaction.

**Step 8 Monitor & Tune Operational System**

In monitoring and tuning the operational system, we will improve several components so that the system has fast performance. Some of the components that are considered are transaction throughput which affects the number of transactions, response time affects transaction time, and the disk storage required to store the database. Then other things that can improve performance are seen based on system resources starting from large main memory, CPU to control tasks, Disk I / O for storing and retrieving data, and a good network. YukBid has a new requirement to improve database performance. The new requirement is that the system can store images of items that are sold and will be auctioned off by bidders, users and bidders can communicate in one social media or in one application to facilitate auctions, and sellers can see the real auction process -time.

**The Result of YukBid Denormalization**

Diagram, schematic

Description automatically generated

**References**

Thomas Connolly and Carolyn Begg. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management. 06. Pearson Education. USA. ISBN: 978-1-292-06118-4.