**SC2207 Database**

**Lab Project 3**

REP2 Group 1:

Lim Zhi Yong

Hui Xujia

Ethan Lee Kei Shen

Huang Caihong

Chong Huai Zhi

Bhati Nancy

The ER diagram that we have chosen to use is the one provided.

**Except**:

* For **SHOP** weak entity class between **USER-ACCOUNT** and **SHOP**, to distinguish it between **SHOP**, we rename it as “**SHOPPING**”.
* For **RESTAURANT-OUTLET** entity class, we have added additional attributes:

1. Unit-no - which indicates the outlet’s unit number. (eg. 0124)
2. Floor - which indicates which floor the outlet is located on.
3. Type - the type of restaurant.
4. Name - the name of the restaurant.

Rationale:

For most cases, the unit numbers will also contain the floor number, however there are malls in which it does not, thus we included the floor attribute.

* For **SHOP** entity class, we have added additional attributes:

1. Shop-Manager-Id - the ID of the shop manager.
2. Shop-Manager-Name - the shop manager’s name.

Rationale:

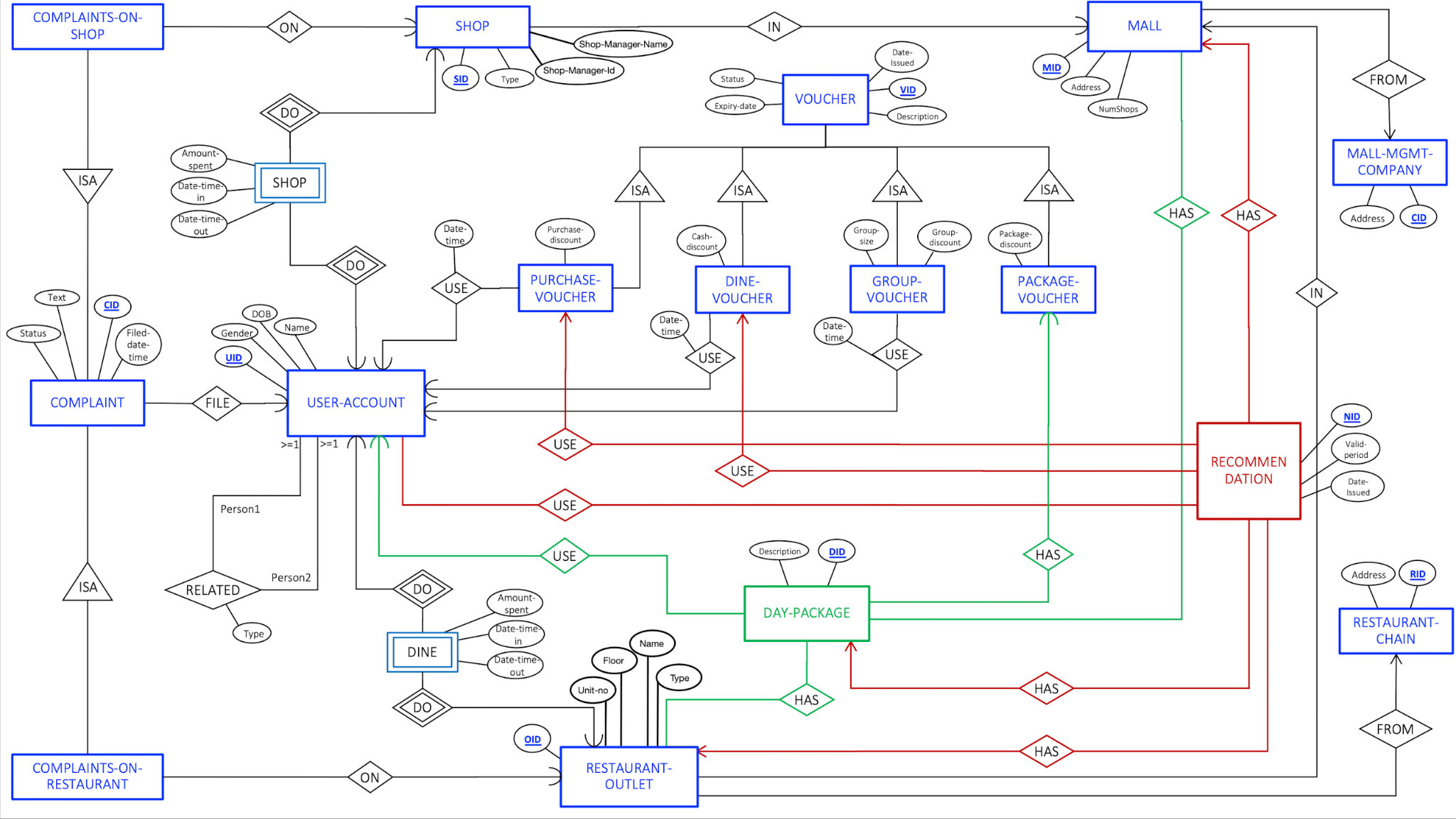
We added the attributes "Shop-Manager-ID" & "Shop-Manager-Name" to the Shop entity set because we imagine that perhaps someone wanted to keep track of who the manager of each shop is. We need a manager ID because different managers can have the same name. We also need a manager name because in reality we want to know people by their name and not just by their IDs. The Shop table is now Shop(SID, Type, MID, Shop-Manager-Name, Shop-Manager-ID). SID is the key because it determines all other attributes.

These 2 new attributes cause an additional FD Shop-Manager-ID -> Shop-Manager-Name to appear in the resulting Shop table, which then causes the table to not be in 3NF because the LHS does not contain a key and the RHS is not in a key.

Also note that Shop-Manager-ID cannot determine SID because a manager can be a manager of more than one shop.

In reality, a principled decision would be to add a separate Manager entity set to not cause any 3NF violation. However, we deliberately did not do this because we want to show 3NF decomposition in this lab submission.

The changes have been reflected in the ER diagram below:



Entities:

| **USER-ACCOUNT(UID, Gender, DOB, Name)** |
| --- |
| Keys: UID |
| Primary Key: UID |
| FDs:   * UID → {Gender, DOB, Name} |
| In 3NF: Yes |

In order to execute 3NF decomposition, we introduced a few new attributes: Shop-Manager-ID, Shop-Manager-Name.

Although we know that creating an entity called “Shop-Manager”, and a many(shop)-to-one(manager) relationship “Manage”, is a better way to do that. But we are regarding these two attributes as a part of the entity shop just because we want to demonstrate 3NF decomposition.

| **SHOP(SID, Type, MID, Shop-Manager-ID, Shop-Manager-Name )** |
| --- |
| Keys: SID |
| Primary Key: SID |
| FDs:   * SID → {Type, MID, Shop–Manager-ID, Shop-Manager-Name} * Shop–Manager-ID → Shop-Manager-Name |
| In 3NF: No. In the FD “Shop–Manager-ID → Shop-Manager-Name  ”, the LHS does not contain a key and the RHS is not in a key.  Step 1: Derive minimal basis of set of FDs   * Each RHS of every FD should only have 1 attribute   + SID → {Type, MID, Shop–Manager-ID, Shop-Manager-Name} will become:     - SID → Type, SID → MID, SID → Shop–Manager-ID, SID → Shop-Manager-Name   + Shop–Manager-ID → Shop-Manager-Name keeps the same:     - * Shop–Manager-ID → Shop-Manager-Name * Remove Redundant FDs   + Check SID → Type     - Compute closures for the LHS:       * {SID}+ = {SID, MID, Shop–Manager-ID, Shop-Manager-Name}       * Not redundant   + Check SID → MID     - Compute closures for the LHS:       * {SID}+ = {SID, Type, Shop–Manager-ID, Shop-Manager-Name}       * Not redundant   + Check SID → Shop–Manager-ID     - Compute closures for the LHS:       * {SID}+ = {SID, Type, MID, Shop-Manager-Name}       * Not redundant   + Check SID → Shop-Manager-Name     - Compute closures for the LHS:       * {SID}+ = {SID, Type, MID, Shop–Manager-ID, Shop-Manager-Name}       * Redundant   + Check Shop–Manager-ID → Shop-Manager-Name     - Compute closures for the LHS:       * {Shop–Manager-ID}+ = {Shop–Manager-ID, Shop-Manager-Name}       * Not redundant   + Result:     - SID → Type, SID → MID, SID → Shop–Manager-ID     - Shop–Manager-ID → Shop-Manager-Name   Step 2: Combine FDs with the same LHS   * SID → {Type, MID, Shop–Manager-ID} * Shop–Manager-ID → Shop-Manager-Name   Step 3: Create a table for each remaining FD   * R1(SID, Type, MID, Shop–Manager-ID) * R2(Shop–Manager-ID, Shop-Manager-Name)   Step 4: If none of the tables contain a key of the original table, create a table that contains it   * R1 already contains the original primary key SID   Step 5: Remove redundant tables   * There is no redundant table to remove.   Step 6: Final result:   * **Shop-A(SID,Type, MID, Shop-Manager-ID)** * **Shop-B(Shop-Manager-ID, Shop-Manager-Name)** |

| **SHOPPING(SID, UID, Amount-spent, Date-time-in, Date-time-out)** |
| --- |
| Keys: {SID, UID, Date-time-in} , {SID, UID, Date-time-out} |
| Primary Key: {SID, UID, Date-time-in} |
| FDs:   * {SID, UID, Date-time-in} → {Amount-spent, Date-time-out} * {SID, UID, Date-time-out} → {Amount-spent, Date-time-in} |
| In 3NF: Yes |

| **DINING(UID, OID, Amount-spent, Date-time-in, Date-time-out)** |
| --- |
| Keys: {UID + OID + Date-time-in}, {UID + OID + Date-time-out} |
| Primary Key: {UID + OID + Date-time-in} |
| FDs:   * {UID, SID, Date-time-in} → {Amount-spent, Date-time-out} * {UID, SID, Date-time-out} → {Amount-spent, Date-time-in} |
| In 3NF: Yes |

| **MALL(MID, Address, NumShops, CID)** |
| --- |
| Keys: MID, Address |
| Primary Keys: MID |
| FDs:   * MID → {Address, NumShops, CID} * Address → {MID, NumShops, CID} |
| In 3NF: Yes |

| **MALL-MGMT-COMPANY(CID, Address)** |
| --- |
| Keys: CID |
| Primary Keys: CID |
| FDs:   * CID → Address |
| In 3NF: Yes |

In order to execute 3NF decomposition, we introduced a few new attributes: Unit-no, Type, Name, Floor

| **RESTAURANT-OUTLET(OID, Unit-no, MID, RID, Type, Name, Floor)** |
| --- |
| Keys: OID, {MID, Unit-no} |
| Primary Keys: OID |
| FDs:   * OID → {Unit-no, MID, RID, Type, Name, Floor} * {MID, Unit-no} → {OID, RID, Type, Name, Floor} * Unit-no → Floor |
| In 3NF: No. In the FD “Unit-no → Floor”, the LHS does not contain a key and the RHS is not in a key.  Step 1: Derive minimal basis of set of FDs   * Each RHS of every FD should only have 1 attribute   + OID → {Unit-no, MID, RID, Type, Name, Floor} will become:     - OID → Unit-no, OID → MID, OID → RID, OID → Type, OID → Name, OID → Floor   + {MID, Unit-no} → {OID, RID, Type, Name, Floor} will become:     - {MID, Unit-no} → OID, {MID, Unit-no} → RID, {MID, Unit-no} → Type, {MID, Unit-no} → Name, {MID, Unit-no} → Floor   + Unit-no → Floor keeps the same:     - Unit-no → Floor * Remove redundant FDs   + Check OID → Unit-no     - Compute closures for the LHS:       * {OID}+ = {OID, MID, RID, Type, Name, Floor}       * Not redundant   + Check OID → MID     - Compute closures for the LHS:       * {OID}+ = {OID, Unit-no, RID, Type, Name, Floor}       * Not redundant   + Check OID → RID     - Compute closures for the LHS:       * {OID}+ = {OID, Unit-no, MID, RID, Type, Name, Floor}       * Redundant   + Check OID → Type     - Compute closures for the LHS:       * {OID}+ = {OID, Unit-no, MID, RID, Type, Name, Floor}       * Redundant   + Check OID → Name     - Compute closures for the LHS:       * {OID}+ = {OID, Unit-no, MID, RID, Type, Name, Floor}       * Redundant   + Check OID → Floor     - Compute closures for the LHS:       * {OID}+ = {OID, Unit-no, MID, RID, Type, Name, Floor}       * Redundant   + Check {MID, Unit-no} → OID     - Compute closures for the LHS:       * {MID, Unit-no}+ = {Unit-no, MID, RID, Type, Name, Floor}       * Not redundant   + Check {MID, Unit-no} → RID     - Compute closures for the LHS:       * {MID, Unit-no}+ = {OID, Unit-no, MID, Type, Name, Floor}       * Not redundant   + Check {MID, Unit-no} → Type     - Compute closures for the LHS:       * {MID, Unit-no}+ = {OID, Unit-no, MID, RID, Name, Floor}       * Not redundant   + Check {MID, Unit-no} → Name     - Compute closures for the LHS:       * {MID, Unit-no}+ = {OID, Unit-no, MID, RID, Type, Floor}       * Not redundant   + Check {MID, Unit-no} → Floor     - Compute closures for the LHS:       * {MID, Unit-no}+ = {OID, Unit-no, MID, RID, Type, Name, Floor}       * Redundant   + Check Unit-no → Floor     - Compute closures for the LHS:       * {Unit-no}+ = {Unit-no}       * Not redundant   + Result:     - OID → Unit-no, OID → MID     - {MID, Unit-no} → OID, {MID, Unit-no} → RID, {MID, Unit-no} → Type, {MID, Unit-no} → Name     - Unit-no → Floor   Step 2: Combine FDs with the same LHS   * OID → {Unit-no, MID} * {MID, Unit-no} → {OID, RID, Type, Name} * Unit-no → Floor   Step 3: Create a table for each remaining FD   * R1(OID, Unit-no, MID) * R2(OID, Unit-no, MID, RID, Type, Name) * R3(Unit-no, Floor)   Step 4: If none of the tables contain a key of the original table, create a table that contains it   * R1/R2 already contains the original primary key A   Step 5: Remove redundant tables   * R1 is subset of R2, thus remove R1   Step 6: Final result:   * **RESTAURANT-OUTLET-A(OID, Unit-no, MID, RID, Type, Name)** * **RESTAURANT-OUTLET-B(Unit-no, Floor)** |

| **RESTAURANT-CHAIN(RID, Address)** |
| --- |
| Keys: RID |
| Primary Keys: RID |
| FDs:   * RID → Address |
| In 3NF: Yes |

| **DAY-PACKAGE (DID, Description,VID, UID)** |
| --- |
| Keys: DID |
| Primary Keys: DID |
| FDs:   * DID → {Description, VID, UID} |
| In 3NF: Yes |

| **COMPLAINT (CID,Text,Status,Filed-date-time,UID)** |
| --- |
| Keys: CID |
| Primary Key: CID |
| FDs:   * CID → {Text,Status,Filed-date-time,UID} |
| In 3NF: Yes |

| **COMPLAINTS-ON-RESTAURANT (CID, OID)** |
| --- |
| Keys: CID |
| Primary Key: CID |
| FDs:   * CID → OID |
| In 3NF: Yes |

| **COMPLAINTS-ON-SHOP(CID, SID)** |
| --- |
| Keys: CID |
| Primary Key: CID |
| FDs:   * CID → SID |
| In 3NF: Yes |

| **VOUCHER(VID, Date-issued, Description, Status, Expiry-date)** |
| --- |
| Keys: VID |
| Primary Key: VID |
| FDs:   * VID → {Date-Issued, Description, Status, Expiry-date} |
| In 3NF: Yes |

| **PURCHASE-VOUCHER(VID, Purchase-discount, UID, Date-time)** |
| --- |
| Keys: VID |
| Primary Key: VID |
| FDs:   * VID → Purchase-discount, UID, Date-time |
| In 3NF: Yes |

| **DINE-VOUCHER(VID, Cash-discount, UID, Date-time)** |
| --- |
| Keys: VID |
| Primary Key: VID |
| FDs:   * VID → {Cash-discount, UID, Date-time} |
| In 3NF: Yes |

| **GROUP-VOUCHER(VID, Group-size, Group-discount, UID, Date-time)** |
| --- |
| Keys: VID |
| Primary Key: VID |
| FDs:   * VID → {Group-size, Group-discount, UID, Date-time} |
| In 3NF: Yes |

| **PACKAGE-VOUCHER(VID, Package-discount, UID, Date-time)** |
| --- |
| Keys: VID |
| Primary Key: VID |
| FDs:   * VID → {Package-discount, UID, Date-time} |
| In 3NF: Yes |

| **RECOMMENDATION(NID, Valid-period, Date-issued, MID, Dine-voucher, Purchase-voucher, DID, OID)** |
| --- |
| Keys: NID |
| Primary Key: NID |
| FDs:   * NID → {Valid-period, Date-issued, MID, Dine-voucher, Purchase-voucher, DID, OID} |
| In 3NF: Yes |

Relationship:

| **USER-RELATED-USER(UID1, UID2, Type)** |
| --- |
| Keys: {UI1, UI2} |
| Primary Key: {UI1, UI2} |
| FDs:   * {UI1, UI2} → Type |
| In 3NF: Yes |

| **USER-USE-RECOMMENDATION(UID, NID)** |
| --- |
| Keys: {UID, NID} |
| Primary Key: {UID, NID} |
| FDs:   * {UID, NID} → {UID, NID} (trivial) |
| In 3NF: Yes |

| **DAY-PACKAGE-HAS-MALL(DID, MID)** |
| --- |
| Keys: {DID, MID} |
| Primary Key: {DID, MID} |
| FDs:   * {DID, MID} → {DID, MID} (trivial) |
| In 3NF: Yes |

| **DAY-PACKAGE-HAS-RESTAURANT-OUTLET(DID, OID)** |
| --- |
| Keys: {DID, OID} |
| Primary Key: {DID, OID} |
| FDs:   * {DID, OID} → {DID, OID} (trivial) |
| In 3NF: Yes |