Database Fundamentals Part C

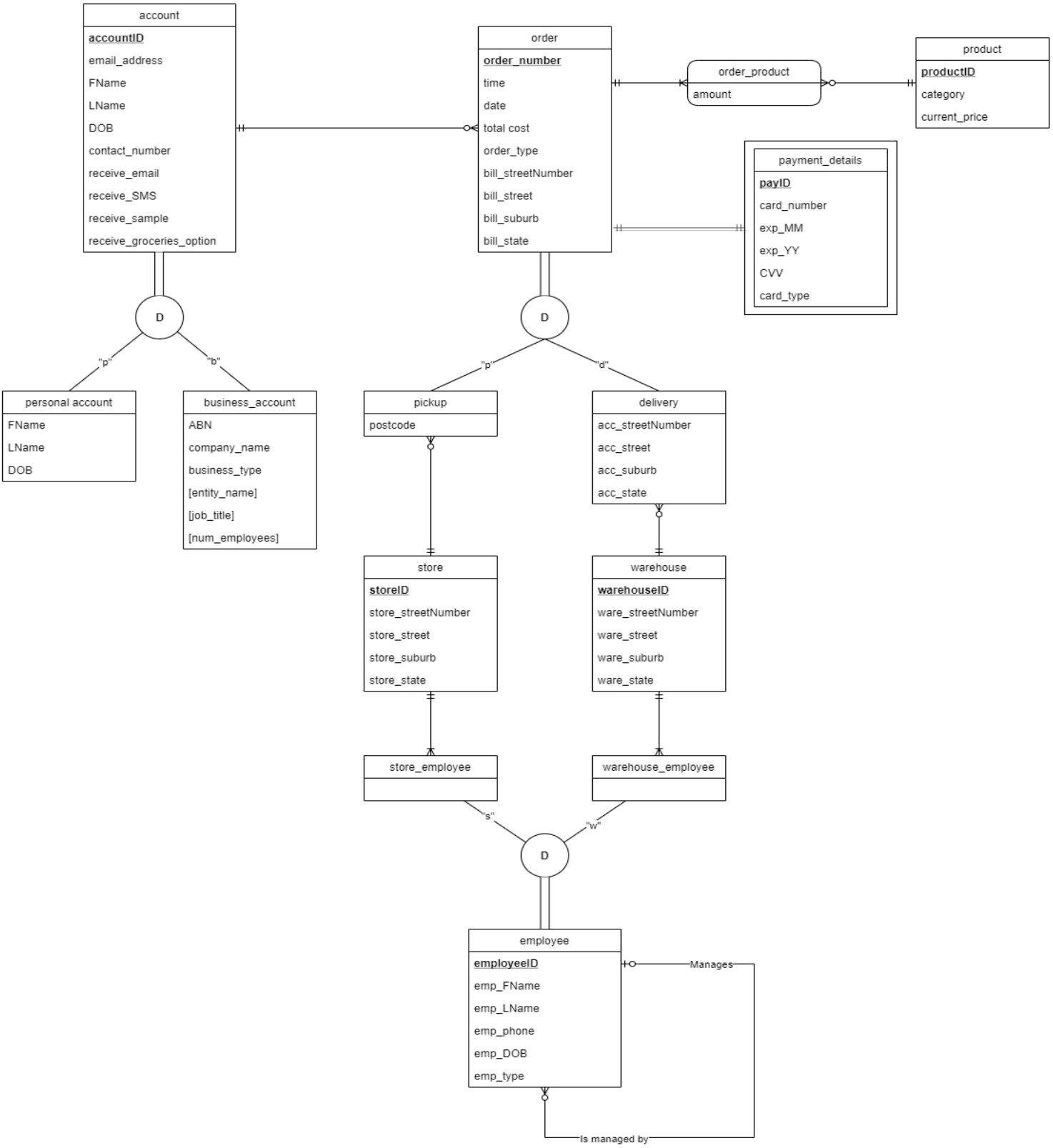
Group Name: Tut1-03-15

Tegh Bir Singh	13213929
Tristam Fryer	13553831
Yi Chun Choong	13665068

Revised Business Rules

1. To create an account the user must register email, and contact number.
2. A personal account requires a name and DOB.
3. Users are uniquely identified by their account ID.
4. Each account must be either shopping for personal use (user account) or for business purposes (business account).
5. A business account must have an ABN, company name and business type and may have an entity name, job title or a number of employees attributes.
6. Customers can set preferences regarding communication and how they receive groceries.
7. Each order has a unique order number.
8. Each order must have a time, date, and total cost.
9. Users must choose either delivery or pickup for each order.
10. Both pickup and delivery orders require a billing address.
11. For pickup, the store closest to the customer's inputted postcode will be assigned.
12. For each store, there may be many orders.
13. For delivery, the warehouse closest to the customer's delivery address will be tasked to deliver.
14. For each warehouse, there may be many delivery orders.
15. Each order requires payment details.
16. For each order there must be at least one product and, for each product, there can be many orders.
17. An order must be at least $30 for a delivery or pickup.
18. The user can specify how much of a product they want for an order.
19. Each product has a unique product ID.
20. Each Product must have a current price and a category it belongs to.
21. An employee can manage any number of employees.
22. Each employee must have personal details(name, dob, phone_no)
23. An Employee must be either a store employee or a warehouse employee.
24. Store and warehouse are managed by a number of employees.
25. All stores and warehouses must have address details
26. Each store can be uniquely identified with store ID.
27. Each warehouse can be uniquely identified with a warehouse ID.
28. Each employee can be uniquely identified with an employee ID.

Revised ERD



Note: PayId is meant to be double underlined

Relations

account (**accountID,** email_address, contact_no, receive_email, receive_SMS, receive_sample, receive_groceries_option, acc_type)

personal_account (**accountID***, FName, LName, DOB)

business_account (**accountID*,** ABN, company_name, business_type, [entity_name], [job_title], [num_employees])

order (**order_number,** accountID*, time, date, total_cost, order_type, bill_streetNumber, bill_street, bill_suburb, bill_state)

FK (accountID) reference ACCOUNT

pickup (**order_number***, storeID*, postcode)

FK (storeID) references STORE

delivery (**order_number***, warehouseID*, acc_streetNumber, acc_street, acc_suburb, acc_state)

FK (warehouseID) references WAREHOUSE

store (**storeID**, store_streetNumber, store_street, store_suburb, store_state)

warehouse (**warehouseID**, ware_streetNumber, ware_street, ware_suburb, ware_state)

employee (**employeeID**, emp_FName, emp_LName, emp_phone, emp_DOB, managerID*,emp_type)

FK (managerID) references employee

store_employee (**employeeID***, storeID*)

FK (employeeID) references EMPLOYEE

FK (storeID) references STORE

warehouse_employee (**employeeID*,** warehouseID*)

FK (employeeID) references EMPLOYEE

FK (warehouseID) references WAREHOUSE

product (**productID,** category, current_price)

order_product (**order_number*, productID***, amount)

FK (order_number) references ORDER

FK (productID) references PRODUCT

payment_details (**order_number*, payID(partial identifier)**, card_number, exp_MM, exp_YY, CVV, card_type)

FK (order_number) references ORDER

List of Functional Dependencies related to Business Rules

4.1 Account

FD1: accountID → email_address, contact_no, receive_email, receive_SMS, receive_sample, receive_groceries_option, acc_type

BR1: To create an account the user must register email, and contact number.

BR3: Users are uniquely identified by their account ID.

BR6: Customers can set preferences regarding communication and how they receive groceries.

FD2: accountID → FName, LName, DOB

BR2: A personal account requires a name and DOB.

BR3: Users are uniquely identified by their account ID.

FD3: accountID → ABN, company_name, business_type, [entity_name], [job_title], [num_employees]

BR3: Users are uniquely identified by their account ID.

BR5: A business account must have an ABN, company name and business type and may have an entity name, job title or a number of employees attributes.

4.2 Order

FD4: Order_number → time, date, total_cost, order_type, bill_streetNumber, bill_street, bill_suburb, bill_state

BR7: Each order has a unique order number.

BR8: Each order must have a time, date, and total cost.

BR9: Users must choose either delivery or pickup for each order.

BR10: Both pickup and delivery orders require a billing address.

FD5: Order_number → postcode

BR7: Each order has a unique order number.

BR11: For pickup, the store closest to the customer's inputted postcode will be assigned.

FD6: Order_number → acc_streetNumber, acc_street, acc_suburb, acc_state

BR7: Each order has a unique order number.

BR13: For delivery, the warehouse closest to the customer's delivery address will be tasked to deliver.

4.3 Product

FD7: productID → category, current_price

BR19: Each product has a unique product ID.

BR20: Each Product must have a current price and a category it belongs to.

4.4 Order_product

FD8: order_number, productID → amount

BR7: Each order has a unique order number.

BR9: The user can specify how much of a product they want for an order.

BR16: For each order, there must be at least one product and, for each product, there can be many orders.

BR19: Each product has a unique product ID.

4.5 Payment_details

FD9: payID → card_number, exp_MM, exp_YY, CVV, card_type

BR15: Each order requires payment details.

4.6 Store

FD10: storeID → store_streetNumber, store_street, store_suburb, store_state

BR25: All stores and warehouses must have address details

BR26: Each store can be uniquely identified with a store ID.

4.7 Warehouse

FD11: warehouseID → ware_streetNumber, ware_street, ware_suburb, ware_state

BR25: All stores and warehouses must have address details

BR27: Each warehouse can be uniquely identified with a warehouse ID.

4.8 Employee

FD12: employeeID → emp_FName, emp_LName, emp_phone, emp_DOB, emp_type

BR22: Each employee must have personal details(name, dob, phone_no)

BR23: An Employee must be either a store employee or a warehouse employee.

BR28: Each employee can be uniquely identified with an employee ID.

FD13: employeeID → *managerID

BR21: An employee can manage any number of employees.

Normalisation

Notes:

- If a relation is in 2NF then it must also be in 1NF. If a relation is 3NF then it must also be in 2NF. Due to this, we will first check if all relations are in 2NF(as we have already checked all relations are in 1NF) and then 3NF.

-As all relations were normalised, all the relations have remained the same.

5.1 Account

All attributes are atomic and there are no derived attributes so the relation must be at least 2NF considering partial dependency is not possible, and as the primary key has only one element (“accountID”). As there is only a single determinant used in each functional dependency, “accountID” as a transitive dependency is not possible.

Therefore, Account is in 3NF.

Relation: account (**accountID,** email_address, contact_no, receive_email, receive_SMS, receive_sample, receive_groceries_option, acc_type)

FD1: accountID → email_address, contact_no, receive_email, receive_SMS, receive_sample, receive_groceries_option, acc_type

FD2: accountID → FName, LName, DOB

FD3: accountID → ABN, company_name, business_type, [entity_name], [job_title], [num_employees]

5.2 Order

All attributes are atomic and there are no derived attributes so the relation must be in at least 1NF. There is no partial dependency as the primary key consists of only one attribute. There is no transitive dependency as the primary key is the determinant for all the functional dependencies.

Therefore, Order is in 3NF.

Relation: order (**order_number,** accountID*, time, date, total_cost, order_type, bill_streetNumber, bill_street, bill_suburb, bill_state)

FD4: Order_number → time, date, total_cost, order_type, bill_streetNumber, bill_street, bill_suburb, bill_state

FD5: Order_number → postcode

FD6: Order_number → acc_streetNumber, acc_street, acc_suburb, acc_state

5.3 Product

All attributes are atomic and there are no derived attributes so the relation must be in at least 1NF. No partial dependency exists due to the key containing only one element. No transitive dependency is possible due to there only being one functional dependency.

Therefore, Product is in 3NF.

Relation: product (**productID,** category, current_price)

FD7: productID → category, current_price

5.4 Order_Product

All attributes are atomic and there are no derived attributes so the relation must be in at least 1NF. Since both attributes are required for FD8, partial dependency cannot occur. Notice that a functional dependency such as productID->amount is not possible as multiple orders can have the same product with different amounts. Similarly, order_number -> amount is not possible as each order can have many products with different amounts.

Transitive dependency is not possible as there is only one functional dependency.

Therefore, Order_product is in 3NF.

Relation: order_product (**order_number*, productID***, amount)

FD8: order_number, productID → amount

5.5 Payment_details

All attributes are atomic and there are no derived attributes so the relation must be in at least 1NF. Partial dependency and transitive dependency are not possible due to there being only one attribute in the primary key and one functional dependency.

Therefore, Payment_details is in 3NF.

Relation: payment_details (**order_number***, **payID(partial identifier)**, card_number, exp_MM, exp_YY, CVV, card_type)

FD9: payID → card_number, exp_MM, exp_YY, CVV, card_type

5.6 Store

All attributes are atomic and there are no derived attributes so the relation must be in at least 1NF. Partial dependency and transitive dependency are not possible due to there being only one attribute in the primary key and one functional dependency.

Therefore, Store is in 3NF.

Relation: store (storeID, store_streetNumber, store_street, store_suburb, store_state)

FD10: storeID → store_streetNumber, store_street, store_suburb, store_state

5.7 Warehouse

All attributes are atomic and there are no derived attributes so the relation must be in at least 1NF. Partial dependency and transitive dependency are not possible due to there only one attribute in the primary key and one functional dependency.

Therefore, Warehouse is in 3NF.

Relation: warehouse (warehouseID, ware_streetNumber, ware_street, ware_suburb, ware_state)

FD11: warehouseID → ware_streetNumber, ware_street, ware_suburb, ware_state

5.8 Employee

All attributes are atomic and there are no derived attributes so the relation must be in at least 1NF. Partial dependency is not possible as there is only one attribute in the primary key. Though there are two functional dependencies a transitive dependency is not possible as the only determinant is the primary key.

Therefore, Employee is in 3NF.

Relation: employee (employeeID, emp_FName, emp_LName, emp_phone, emp_DOB, managerID*, emp_type)

FD12: employeeID → emp_FName, emp_LName, emp_phone, emp_DOB, emp_type
FD13: employeeID → *managerID

Reflection

Database Fundamentals has allowed this group to further our professional development. Through our own personal studies as well as the lectures we have learnt how to correctly organise a database into a format that is easy to understand and access. Studying this subject has granted us a deeper knowledge about creating databases, using methods such as an entity-relationship diagram to visualise the connections entities have with each other and normalisation to allow the data and each entity to be more organised and accessible. We would like to reflect on our prior knowledge, what we have taken away from the subject and how we will use this going forwards.

Before this subject we knew very little about databases and the complexity required to implement them correctly. None of us had done a subject previously that covered the topics in Database Fundamentals so we all held very little knowledge on database design.

From the lecture content, the helpful tutorials, and our own studies, we managed to understand the difficult process of creating an effective and efficient database. We deepened our knowledge by investigating the purpose of each step and how they compound to create a better design.

From the subject's contents, we have learnt the following designs

- Business Rules
    - Used to gather requirements from the shareholders
- ERDs
    - Used to display the relationships between entities
- Relations
    - Used to visualise the data, showing relevant attributes
- Normalisation
    - Used to identify functional dependencies and therefore data redundancy

The lectures and tutorials promoted the theoretical knowledge used in database design, although the assignment has evolved our practical knowledge. Each assessment has required us to find database solutions to real world problems, sculpting us into better critical thinkers, which directly improves our skills as engineers.

The assignment helped us realise where we lacked knowledge in designing ERD. deal with particular cases. We learned how to write functional dependency, and relations for subtypes which we struggled with. We learned that derived attributes do not necessarily have to be derived from one entity, but multiple entities.

We learned to make connections with other subjects such as Data Systems in which we had to make a class diagram. We realised that a class diagram is essentially an ERD but with

functions being used added to the ERD. This helped us realise the notation for an ERD could also be used for class view diagrams.

Collectively, from tutor reviews and internal discussion, our group believes our design has a 10/10 ranking. We have applied every design procedure learnt this semester and ensured each detail was reviewed before final implementation, including notation, designs and format. We took in feedback from our previous parts and improved upon any areas that were brought to our attention.

Possible improvements:

- For the assignment we believe we can better model the functionality of Woolworths. For instance to keep things simple we did not add a weak entity called pricing history in which would have recorded the price of each product at different dates. However we believe we have met the goal and requirements for this assignment.

Summarily, Database Fundamentals has been an entertaining and educational subject. We would only suggest more time be allocated to tutorials or less be asked to cover during the session as it often feels rushed and questions are hard to get to. Overall, we believe it has been a valuable experience as engineering students and will benefit our learning careers going forward.