

Module: DB-020321-04

FINAL ASSESSMENT PROJECT

In answering the following questions, make certain you are using the terminology you desire. If you need clarification on a question, contact the instructor before submitting your answers. Your responses are due in the Assignment folder on or before next Thursday. You are not to contact other students during this assignment or receive any outside help. You may use your text, DB2, DB2 help files, DB2 Information Center, or module notes. All answers should be submitted in the same note or attachment.

1. Page 165 Question 7. Draw an ERD for this scenario. You must use the crow's foot notation. Document any assumptions you made that are not present in the scenario.

Automata Inc. produces specialty vehicles by contract. The company operates several departments, each one of which builds a particular vehicle, such as limousine, a truck, a van or an RV.

When a new vehicle is built, the department places an order with the purchasing department to request specific components. Automata's purchasing department is interested in creating a database to keep track of orders and to accelerate the process of delivering materials.

The order received by the purchasing department can contain several different items. An inventory is maintained so that the most frequently requested items are delivered almost immediately. When an order comes in, it is checked to determine whether the requested item is in inventory. If an item is not in inventory, it must be ordered from a supplier. Each item may have several suppliers.

Assumptions

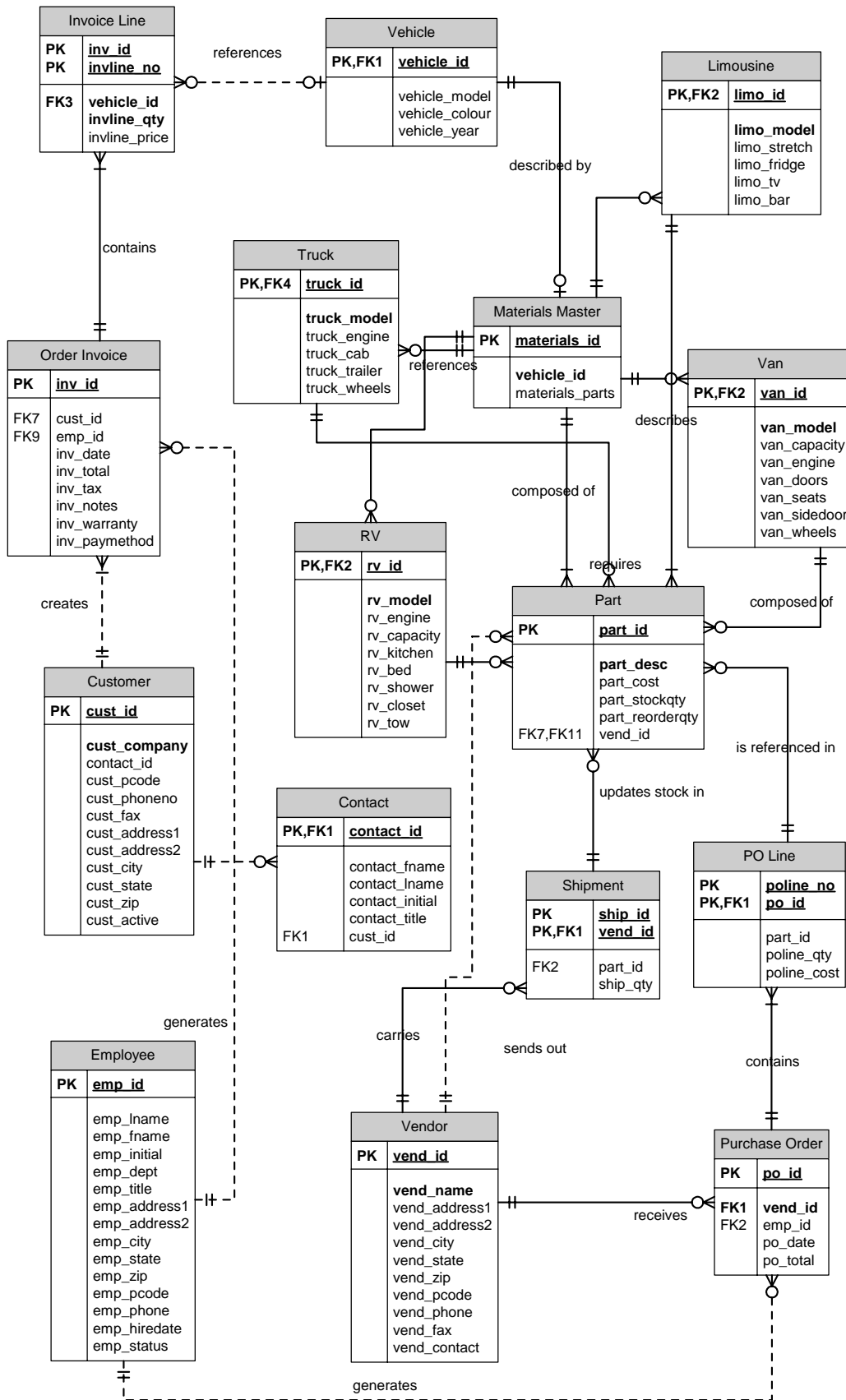
No vehicles are stocked, they are built upon receiving the contract order.

When an order is received, there is a materials master table that has all the information on the necessary parts required to build any vehicle. It basically routes all the assembly information. Ideally this should be kept in one massive table but because it was stated that there are 4 disparate departments that build 4 different types of vehicles as well as the fact that these vehicles are vastly different, I have decided to keep smaller tables that will describe the parts necessary to build the vehicle, depending on the model.

Obviously, I haven't listed all the different attributes that may exist.

When a vehicle is ordered, the materials master is checked for what types of parts are needed. Then the part inventory is queried for the necessary parts and quantities. If the quantities are not available, they are ordered. Ideally there would also be some sort of trigger mechanism to automatically order supplies once the stockqty dips below the reorderqty.

There is a Purchase Order line to facilitate the ordering of parts from the vendors. There is also another table to record shipments so how much is shipped and how much is backordered can be recorded as well as updating the inventory.



2. Write a data dictionary similar to Table 2.6 on page 77 for question 1.

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQ	PK/FK	FK	TABLE
CONTACT	contact_id	Contact ID	r(5)	X(5)	10000-99999	Y	PK		
	contact_lname	Contact Last Name	varchar(50)	X(50)		Y			
	contact_fname	Contact First Name	varchar(50)	X(50)					
	contact_initial	Contact Initial	char(5)	X(5)					
	contact_title	Contact Title	varchar(15)	X(15)					
	cust_id	Customer ID	varchar(50)	X(50)					CUSTOMER
		primary key		contact_id					
CUSTOMER	cust_id	Customer ID	char(5)	X(5)	10000-99999	Y	PK		
	cust_company	Customer Company	varchar(50)	X(50)		Y			
	contact_id	Contact ID	char(5)	X(50)					
	cust_pcode	Phone Code	char(3)	XXX					
	cust_phoneno	Phone Number	char(7)	X(7)					
	cust_fax	Customer Fax	char(7)	X(7)					
	cust_address1	Customer Address Line 1	varchar(50)	X(50)					
	cust_address2	Customer Address Line 2	varchar(50)	X(50)					
	cust_city	Customer City	varchar(50)	X(50)					
	cust_state	Customer State	char(2)	X(2)					
	cust_zip	Customer ZIP	char(10)	X(10)					
	cust_active_ind	Customer active indicator	char(1)	X					
		primary key		cust_id					
		foreign key		contact_id					
EMPLOYEE	emp_id	Employee Id	char(5)	X(5)	10000-99999	Y	PK		
	emp_lname	Employee surname	varchar(50)	X(50)		Y			
	emp_fname	Employee firstname	varchar(50)	X(50)					
	emp_initial	Employee initial	char(1)	X					
	emp_dept	Employee department	varchar(50)	X(50)					
	emp_title	Employee title	varchar(5)	X(5)					
	emp_address1	Employee Address line 1	varchar(50)	X(50)					
	emp_address2	Employee Address Line 2	varchar(50)	X(50)					
	emp_city	Employee City	varchar(50)	X(50)					
	emp_state	Employee State	char(2)	X(2)					
	emp_zip	Employee Zip	char(10)	X(10)					
	emp_pcode	Employee Phone Code	char(3)	XXX					
	emp_phone	Employee PhoneNumber	char(7)	X(7)					
	emp_hiredate	Employee Hire Date	date	mm/dd/yyyy					
	emp_status	Employee Status	char(1)	X	A/I				
		primary key		emp_id					
	ORDER_INVOICE	inv_id	Invoice Number	char(5)	X(5)	10000-99999	Y	PK	
cust_id		Customer Id	char(5)	X(5)	10000-99999	Y	FK		CUSTOMER
emp_id		Employee Id	char(5)	X(5)	10000-99999	Y	FK		EMPLOYEE
inv_date		Invoice Date	date	mm/dd/yyyy		Y			
inv_total		Invoice Total	number(9,2)	9(7),99		Y			
inv_tax		Tax Amount	number(9,2)	9(7),99		Y			
inv_warranty		Waranty indicator	char(1)	X	Y	Y			

	inv_paymethod	invoice payment method	char(2)	XX	CR/CS/CK	Y	
	inv_notes	Customer Requests	varchar(250)	X(250)		Y	
	primary key		inv_id				
	foreign key		cust_id				
	foreign key		emp_id				
INVOICE_LINE	inv_id	Invoice Number	char(5)	X(5)	10000-99999	Y	FK ORDER_INVOICE
	invline_no	Invoice Line Number	Integer	999	1-999	Y	PK
	vehicle_id	Vehicle ID	char(5)	X(5)	10000-99999		FK VEHICLE
	invline_qty	Quantity used	Integer	99	1-99		
	invline_price	Sales price of line item	number(6,2)	9999,99			
	primary key		inv_id, inv				
	foreign key		vehicle_id				
	foreign key		inv_id				
VEHICLE	vehicle_id	Vehicle ID	char(5)	X(5)	10000-99999	Y	PK
	vehicle_model	Vehicle model	varchar(50)	X(50)			
	vehicle_colour	Vehicle colour	varchar(10)	X(10)			
	vehicle_year	Vehicle year	char(4)	X(4)			
	primary key		vehicle_id				
MATERIALS_MASTER	materials_id	Materials ID	char(5)	X(5)	10000-99999	Y	PK
	vehicle_id	Vehicle ID	char(5)	X(5)	10000-99999	Y	FK VEHICLE
	materials_parts	Vehicle model	varchar(50)	X(50)			
	primary key		materials_id				
LIMO	limo_id	Limo ID	char	X	10000-99999	Y	PK
	limo_model	Limo model	varchar(50)	X(50)			
	limo_stretch	Limo stretch	char(2)	X(2)			
	limo_fridge	Limo Fridge	char(2)	X(2)			
	limo_tv	Limo TV	char(2)	X(2)			
	limo_bar	Limo bar	char(2)	X(2)			
	primary key		limo_id				
TRUCK	truck_id	Truck ID	char	X	10000-99999	Y	PK
	truck_model	Truck model	varchar(50)	X(50)			
	truck_engine	Truck engine	char(2)	X(2)			
	truck_cab	Truck cab	char(2)	X(2)			
	truck_trailer	Truck trailer	char(2)	X(2)			
	truck_wheels	Truck wheels	char(2)	X(2)			
	primary key		truck_id				
VAN	van_id	Van ID	char	X	10000-99999	Y	PK
	van_model	Van model	varchar(50)	X(50)			
	van_engine	Van engine	char(2)	X(2)			
	van_capacity	Van cab	char(2)	X(2)			
	van_doors	Van trailer	char(2)	X(2)			
	van_wheels	Van wheels	char(2)	X(2)			
	primary key		truck_id				
RV	rv_id	RV ID	char	X	10000-99999	Y	PK
	rv_model	RV model	varchar(50)	X(50)			
	rv_capacity	RV capacity	char(2)	X(2)			
	rv_kitchen	RV kitchen	char(2)	X(2)			

	rv_bed	RV bed	char(2)	X(2)				
	rv_shower	RV shower	char(2)	X(2)				
	rv_closet	RV closet	char(2)	X(2)				
	rv_tow	RV tow	char(2)	X(2)				
	primary key		van_id					
PART	part_id	Part Number	char(5)	X(5)	10000-99999	Y	PK	
	part_desc	Part Description	varchar(50)	X(50)			Y	
	part_cost	Purchase cost of item	number(9,2)	9(7),99				
	part_stockqty	Quantity on hand	Integer	9999	0-9999			
	part_reorderqty	Reorder Quantity	Integer	9999	0-9999			
	primary key		part_id					
	CONSTRAINT POS_PART_STOCKQTY CHECK		part_stockqty >= 0					
	CONSTRAINT POS_REORDERQTY CHECK		part_reorderqty >= 0					
PO_LINE	po_id	Purchase Order ID	char(5)	X(5)	10000-99999	Y	FK	PURCHASE_ORDER
	poline_no	Purchase Order Number	Integer	9999	1-999	Y	PK	
	part_id	Part ID	char(5)	X(5)	10000-99999		FK	PART
	poline_qty	Purchase Order quantity	Integer	9(4)	1-99			
	poline_cost	Purchase Order cost	number(6,2)	9(6),99				
	primary key		po_id, poline_no					
	foreign key		part_id					
	foreign key		po_id					
PURCHASE_ORDER	po_id	Purchase Order ID	char(5)	X(5)	10000-99999	Y	PK	
	vend_id	Vendor ID	char(5)	X(5)	10000-99999	Y	FK	VENDOR
	emp_id	Employee ID	char(5)	X(5)	10000-99999	Y	FK	EMPLOYEE
	po_date	Purchase Order date	date	mm/dd/yyyy				
	po_total	Purchase Order total	number(9,2)	9(9),99				
	primary key		po_id					
	foreign key		vend_id					
	foreign key		emp_id					
VENDOR	vend_id	Vendor Number	char(5)	X(5)	10000-99999	Y	PK	
	vend_name	Vendor Name	varchar(50)	X(50)			Y	
	vend_address1	Vendor Address line 1	varchar(50)	X(50)				
	vend_address2	Vendor Address line 2	varchar(50)	X(50)				
	vend_city	Vendor City	varchar(50)	X(50)				
	vend_state	Vendor State	char(2)	X(2)				
	vend_zip	Vendor Zip	char(10)	X(10)				
	vend_pcode	Vendor Phone Code	char(3)	XXX				
	vend_phone	Vendor Phone Number	char(7)	X(7)				
	vend_fax	Vendor Facsimile Number	char(7)	X(7)				
	vend_contact	Contact person at Vendor	varchar(50)	X(50)				
	primary key		vend_id					
SHIPMENT	ship_id	Shipping ID	char(5)	X(5)	10000-99999	Y	PK	
	vend_id	Vendor ID	char(5)	X(5)	10000-99999	Y	FK	VENDOR
	part_id	Part ID	char(5)	X(5)	10000-99999	Y	FK	PART
	ship_qty	Shipping quantity	Integer	9999				
	primary key		ship_id,vend_id					
	foreign key		vend_id					
	foreign key		part_id					

3. Write the DDL statements for question 1. Your SQL should implement your ERD and data dictionary.

```
create database EverFail;
```

```
connect to EverFail;
```

```
create table CONTACT
```

```
(contact_id          char(5)          not null  
,contact_lname      varchar(50)   not null  
,contact_fname      varchar(50)  
,contact_initial    char(5)  
,contact_title      varchar(15)  
,cust_id            varchar(50)  
,primary key        (contact_id));
```

```
create table CUSTOMER
```

```
(cust_id            char(5)          not null  
,cust_company       varchar(50)   not null  
,contact_id        char(5)          not null  
,cust_pcode         char(3)  
,cust_phoneno      char(7)  
,cust_fax          char(7)  
,cust_address1     varchar(50)  
,cust_address2     varchar(50)  
,cust_city         varchar(50)  
,cust_state        char(2)  
,cust_zip          char(10)  
,cust_active_ind   char(1)  
,primary key       (cust_id)  
,foreign key       (contact_id) references contact on delete restrict on update  
restrict);
```

```
create table EMPLOYEE
```

```
(emp_id            char(5)          not null  
,emp_lname         varchar(50)   not null  
,emp_fname         varchar(50)   not null  
,emp_initial       char(1)  
,emp_dept          varchar(5)  
,emp_title         varchar(5)  
,emp_address1     varchar(50)  
,emp_address2     varchar(50)  
,emp_city         varchar(50)  
,emp_state        char(2)
```

```

.emp_zip          char(10)
.emp_pcode       char(3)
.emp_phone       char(7)
.emp_hiredate    date
.emp_status      char(1)
.primary key     (emp_id));

```

```

create table ORDER_INVOICE
(inv_id          char(5)          not null
,cust_id        char(5)          not null
,emp_id         char(5)          not null
,inv_date       date             not null
,inv_total      num(9,2)
,inv_tax        num(9,2)
,inv_warranty   char(1)
,inv_paymethod  char(2)
,inv_notes      varchar(250)
.primary key    (inv_id)
.foreign key    (cust_id) references customer on delete restrict on update
restrict
.foreign key    (emp_id) references employee on delete restrict on update
restrict);

```

```

create table INVOICE_LINE
(inv_id          char(5)          not null
,invline_no     Integer          not null
,vehicle_id     char(5)          not null
,invline_qty    Integer          not null
,invline_price  num(6,2)
.primary key    (inv_id, invline_no)
.foreign key    (vehicle_id) references vehicle on delete restrict on update
restrict
.foreign key    (inv_id) references ORDER_INVOICE on delete restrict on update
restrict);

```

```

create table VEHICLE
(vehicle_id      char(5)          not null
,vehicle_model  varchar(50)
,vehicle_colour varchar(10)
,vehicle_year   char(4)
.primary key    (vehicle_id));

```

```

create table MATERIALS_MASTER
(materials_id    char(5)          not null
,vehicle_id     char(5)          not null

```

```
,material_parts      char(5)
,primary key         (materials_id));
```

```
creat table LIMO
```

```
(limo_id             char
,limo_model          varchar(50)  not null
,limo_stretch        char(2)
,limo_fridge         char(2)
,limo_tv             char(2)
,limo_bar            char(2)
,primary key         (limo_id));
```

```
creat table TRUCK
```

```
(truck_id           char
,truck_model        varchar(50)  not null
,truck_engine       char(2)
,truck_cab          char(2)
,truck_trailer      char(2)
,truck_wheels       char(2)
,primary key        (truck_id));
```

```
creat table VAN
```

```
(van_id            char
,van_model         varchar(50)  not null
,van_engine        char(2)
,van_doors         char(2)
,van_seats         char(2)
,van_capacity      char(2)
,van_wheels        char(2)
,primary key       (van_id));
```

```
creat table RV
```

```
(rv_id            char
,rv_model         varchar(50)  not null
,rv_capacity      char(2)
,rv_kitchen       char(2)
,rv_bed           char(2)
,rv_shower        char(2)
,rv_closet        char(2)
,rv_tow           char(2)
,primary key      (van_id));
```

```
create table PART
```

```
(part_id          char(5)      not null
,part_desc        varchar(50)  not null
,part_cost        num(9,2)
```

```

,part_stockqty      Integer
,part_reorderqty    Integer
,primary key        (part_id))
,CONSTRAINT POS_PART_STOCKQTY CHECK (part_stockqty >= 0)
,CONSTRAINT POS_REORDERQTY CHECK (part_reorderqty >= 0);

```

```

create table PO_LINE

```

```

(po_id              char(5)      not null
,po_line_no         Integer      not null
,part_id            char(5)      not null
,po_line_qty        Integer      not null
,po_line_cost       num(6,2)
,primary key        (po_id, po_line_no)
,foreign key        (part_id) references PART on delete restrict on update
restrict)
,foreign key        (po_id) references PURCHASE_ORDER on delete restrict on
update
restrict);

```

```

create table PURCHASE_ORDER

```

```

(po_id              char(5)      not null
,vend_id            char(5)      not null
,emp_id             char(5)      not null
,po_date            date         not null
,po_total           num(9,2)
,primary key        (po_id)
,foreign key        (vend_id) references VENDOR on delete restrict on update
restrict
,foreign key        (emp_id) references EMPLOYEE on delete restrict on update
restrict);

```

```

create table VENDOR

```

```

(vend_id           char(5)      not null
,vend_name          varchar(50)  not null
,vend_address1     varchar(50)
,vend_address2     varchar(50)
,vend_city          varchar(50)
,vend_state         char(2)
,vend_zip           char(10)
,vend_pcode         char(3)
,vend_phone         char(7)
,vend_fax           char(7)
,vend_contact       varchar(50)
,primary key        (vend_id));

```

```

create table SHIPMENT

```

```

(ship_id      char(5)      not null
,vend_id     char(5)      not null
,part_id     char(5)      not null
,ship_qty    integer
,primary key (ship_id,vend_id)
,foreign key (vend_id) references VENDOR on delete restrict on update
restrict
,foreign key (part_id) references PART on delete restrict on update
restrict);

```

4. Assume that we have a set of attributes {A, B, C, D, E, F, G} and the functional dependencies:

```

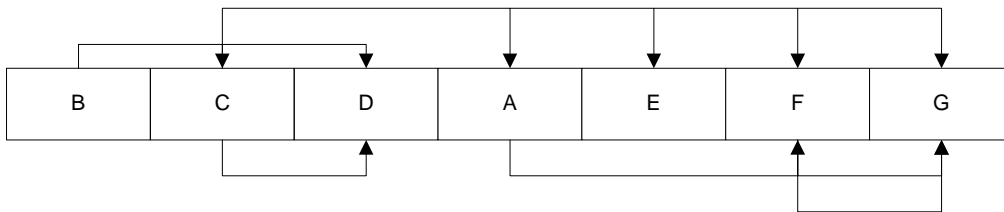
BCD --> A
BC  --> E
A   --> F
F   --> G
C   --> D
A   --> G

```

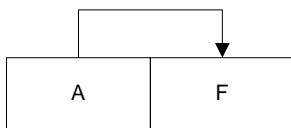
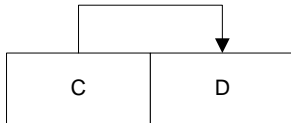
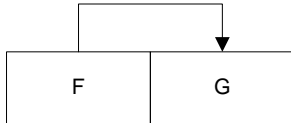
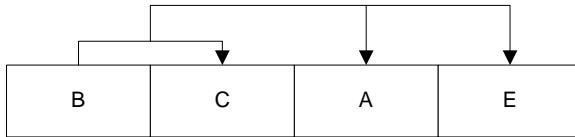
where --> means “determines.”

Factor this set of attributes into relations that are in 3NF. Are all of your tables in BCNF? Why or why not.

If we convert the above to a dependency diagram, it should like:



After factoring to 3NF, we have:



It is currently in BCNF because it meets 3NF requirements (no transitive or partial dependencies) and every determinant is a candidate key.

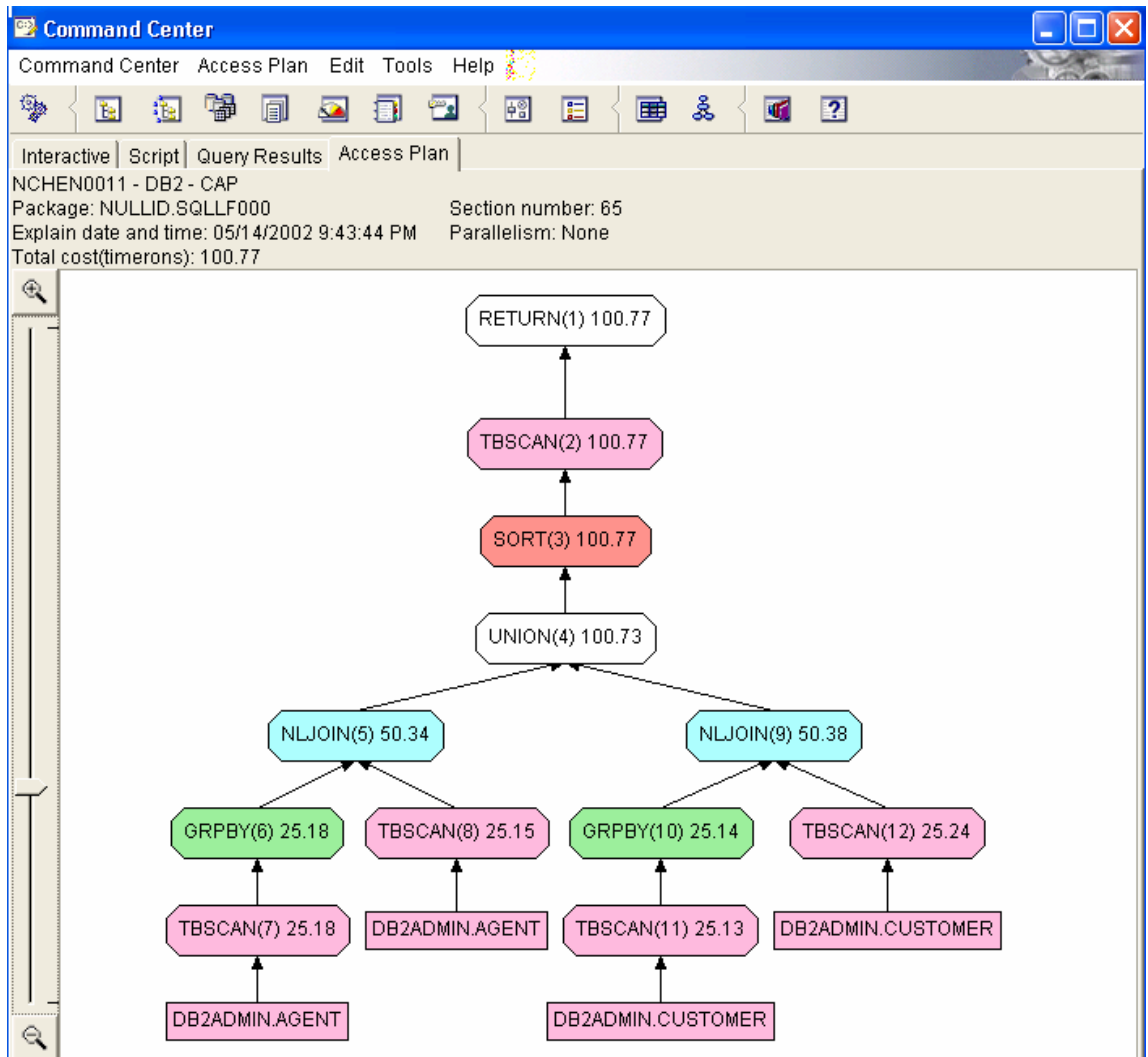
- Use the Command Center to issue the command **runstats on table db2admin.agent;**
Next issue the command **runstats on table db2admin.customer;**

You may need to substitute your ID for db2admin. Generate an access plan for the following SQL query:

```

select city from db2admin.customer where discnt >=all
  (select discnt from db2admin.customer where city = 'Duluth')
union
select city from db2admin.agent where percent >any
  (select percent from db2admin.agent where city like 'N%');
  
```

Take a screen shot of the access plan generated. Paste the plan in a Word document. Explain how the query physically executes using this access plan. What is the cost of this query in timerons?



The cost in timerons is 100.77

A table scan is performed on the CUSTOMER table at TBSCAN(12) for disct >= and the result(s) are stored in a temp table

A table scan is performed on the CUSTOMER table at TBSCAN(11) for city = 'Duluth' and the result is stored in temp table and then grouped by GRPBY(10)

A Nested Loop Join (NLJOIN(9)) is performed on the two tables (50.38 timerons)

A table scan (TBSCAN(8)) is performed on the AGENT table for percent >

And the result(s) are stored in a temp table

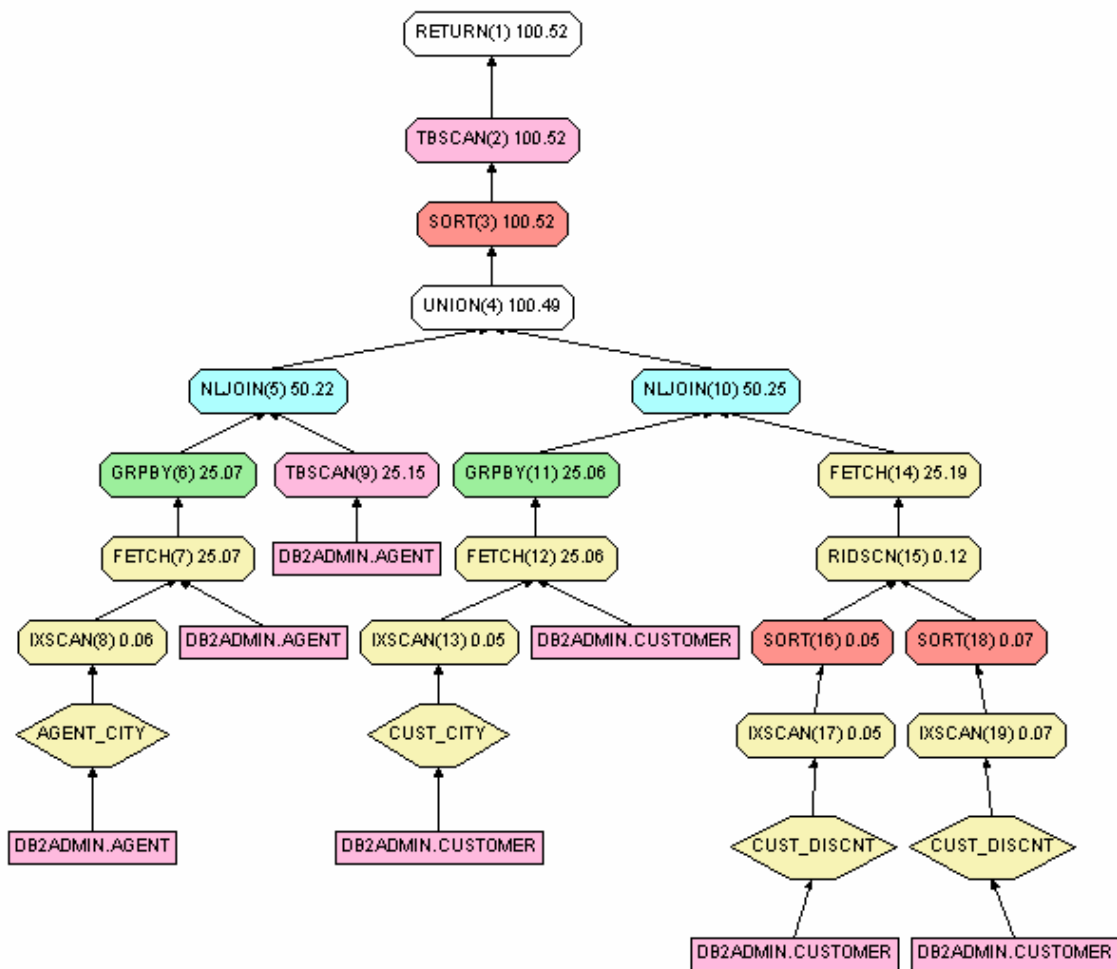
A table scan (TBSCAN(7)) is performed on the AGENT table for city beginning with the letter 'N' and is stored in a temp table

A Nested Loop Join (NLJOIN(5)) is performed on the two tables

Union(4) concatenates the rows from the output tables from NLJOIN(5) and NLJOIN(9). Sort(3) sorts the output by city. TBSCAN(2) scans the final output table and RETURN(1) sends the displayed output to the user.

Next create the following indices, `cust_city` on `customer (city)`, `agent_city` on `agent(city)` and `cust_discnt` on `customer (discnt)`. Generate another access plan for the above query.

```
CREATE INDEX CUST_CITY
  ON CUSTOMER(CITY);
CREATE INDEX AGENT_CITY
  ON AGENT(CITY);
CREATE INDEX CUST_DISCNT
  ON CUSTOMER(DISCNT);
```



Explain this new access plan.

Index Scan (IXSCAN(19)) and (IXSCAN(17)) is performed on index `CUST_DISCNT` for `discnt >=` and the result(s) are stored in a temp table and is sorted by `SORT(18)` and `SORT(16)` respectively. Fetch(14) fetches the discount column from Row Index Scan (RIDSCN(15))

Index Scan (IXSCAN(13)) is performed on index CUST_CITY for city = 'Duluth' and the city column is fetched by FETCH(12) and then grouped by city. Nested Loop Join NLJOIN(10) is performed on the CUST_DISCNT and CUST_CITY index scan results.

A table scan (TBSCAN(9)) is performed on the AGENT table for percent values. A index scan (IXSCAN(8)) is performed on the AGENT_CITY index for city beginning with the letter 'N' and is stored in a temp table. FETCH(7) and GRPBY(6) is performed on the output.

Nested Loop Join NLJOIN(5) is performed on the percent and city values from TBSCAN(9) and IXSCAN(8)

Union(4) concatenates the rows from the output tables from NLJOIN(5) and NLJOIN(10). Sort(3) sorts the output by city. TBSCAN(2) scans the final output table and RETURN(1) sends the displayed output to the user.

Probably the cost in timerons was not significantly different. Do you think there would ever be a situation with the above query where any or all of the indices would be justified? Explain.

A nested loop join is likely to be more efficient if there is an index on the join-predicate columns of the inner table. If the tables are large and the table is accessed frequently, then the indexes are justified.

- 6. Write the SQL query that will return the cid values for customers who place at least one order, but only through agent a04. On the same line with each cid, you query should list the total dollar amount of orders placed by that customer. Submit your query along with the DB2 output.**

```
select o.cid, sum(dollars) as Total_Dollar_Amount
      from ord_order_line o
      inner join order p on p.cid = o.cid
      where p.aid = 'a04'
group by o.cid
```

CID TOTAL_DOLLAR_AMOUNT

c001 9690.50

1 record(s) selected.

- 7. You have a new agent eferguso with aid of a07. eferguso needs to be able to update the qty attribute of the Order_Line table. They also need to be able to SELECT and INSERT any rows in Order that were placed through him. You may need to work with the CAP database to limit eferguso's view of the database so he can not view orders placed through other agents. eferguso should be able to INSERT rows into the Order_Line table. List all SQL statements you need.**

```

INSERT INTO AGENT VALUES ('a07', 'Eferguso', 'AnyCity', 1);

GRANT CONNECT ON DATABASE TO USER Eferguso;

GRANT UPDATE (QTY) ON TABLE ORD_LINE TO USER Eferguso;

CREATE VIEW Eferguso_View as
Select *
        from Order
        where aid = 'a07'

GRANT SELECT ON TABLE Eferguso_View TO USER Eferguso;
GRANT INSERT ON TABLE Eferguso_View TO USER Eferguso;

GRANT INSERT ON TABLE ORD_LINE TO USER Eferguso;

```

8. A centralized multi-user database has a log file and was backed up yesterday. The hard drive on which the database is stored crashes while being used. The disk was not mirrored. Explain how this database should be recovered. Be specific.

Assumptions:

Everything is lost and hard disk is not recoverable

Procedure:

1. Replace hard disk (preferably with some fault tolerance this time, eg. RAID10)
2. From other server's Server Manager, remove the Old server from the domain.
3. Install Windows 2000 Server, any application, network settings, and users. Use the same server name and IP address.
4. Install any Windows 2000 services packs
5. Perform a SQL Server installation
6. Install any SQL Server service packs
7. Install Backup Software
8. Install any Backup Software service packs
9. Restore from backup tape database and transaction log full backup, apply differential backups (if necessary) to database backup.
10. Make sure you have the following scripts available
 - o Gen_sys_info.log contains the output of sp_config which allows the resetting of all configurations options in the event the server needs to be rebuilt.
 - o Gen_create_dev.log contains the information to recreate any database devices.
 - o Gen_create_db.log contains the scripts to recreate all databases with the same segmentation as originally designed.
 - o Gen_create_login.log contains the scripts to recreate all logins

- Gen_create_user_xx.log contains the script to recreate all database users.
11. Reboot the Server
 12. Run the device creation script generated from the script Gen_create_dev.log
 13. Run the database creation script generated from the sp_help_revdatabase procedure with a for load option Gen_create_db.log
 14. Restore the master database
 15. If master fails to restore cleanly
 - A. Rebuild master
 - B. Turn on allow updates
 - C. BCP in the file from syslogins (Gen_create_login.log)
 - D. Turn off allow updates
 - E. Do NOT attempt another restore of master and do NOT restore msdb
 - F. Run script to recreate all dump devices
 - G. Run script to recreate all tasks with the exception of replication tasks
 - H. Reset configuration options (Gen_sys_info.log)
 16. Restore the msdb database
 17. Restore each of the user databases
 18. Apply Transaction Logs (DBMS will ROLLBACK according to whether it is using deferred write or write-through recovery procedure)
 19. Verify that all databases and all objects have been created properly.
 20. Verify that all operators, tasks, and alerts have been set properly
 21. Database is up and running

Reference : <http://www.mssqlserver.com>

9. Distinguish between fat clients and thin clients in a client/server centralized database environment. Give an example description of each.

Fat-Client:

The main portion of the processing resides on the client computer. Because the bulk of the processing is done on the client side, more client resources are necessary in order to accommodate resource hungry applications. At first, this seems to be a negative feature but the current PCs are powerful and this works quite effectively over modern day networks. The main benefit of moving the bulk of the processing to the client is that server resources are available for more critical processes.

Thin-Client:

The main part of the processing is offloaded from the client machine. Maintaining the client application is made much easier because changes and upgrades can be done to the application on the server and subsequently downloaded by client computers upon accessing the application. Furthermore, by moving the majority of the processing off the client, less client resources such as memory or processor speed is required. Therefore high end personal computers can be replaced with relatively low end computers or even windows terminals. This is usually achieved by running a Visual Basic or Java Application.

In client server architecture, there are basically 4 components, data collection, data manipulation, application graphics and presentation. In the thin client model, only the presentation function resides on the client side whereas in the fat client model, only the data collection resides on the server side.

In essence:

Thin Client

Server Side

Client Side

Data Collection → Data Manipulation → Application Graphics → Network → Presentation

Fat Client

Server Side

Client Side

Data Collection → Network → Data Manipulation → Application Graphics → Presentation

Data collection involves moving the data from external sources into a database. Data manipulation involves the sorting and arranging of the data that is in the database to an optimized and manageable form for the application. Application graphics involve creating the layout of all the different graphical elements used to display the manipulated data. Presentation is the actual display of graphics through the client application.

10. Give an example business scenario of a candidate for OLAP. Describe the system architecture you would need for doing OLAP for your scenario.

OLAP scenario:

A nation wide computer retailer with presence wishes to have answers to the following questions:

- *What are the top products whose number of sold pieces in the months chosen by the user compared to the respective month ago has increased mostly?*
- *What is the distribution of the number of sold products over the different states for products and years chosen by the user?*
- *Which are the top customers who bought products in the last three years of a minimum total amount chosen by the user and who show the lowest standard deviation of the totals in these years?*
- *What is the total sales volume for computer products for each month of the past year*
- *If Christmas has the highest sales: drill down to the month of December. What is the total sales volume for computer products for each day of December. What is the total sales volume by computer product for December.*
- *What is the correlation between Personal Home Computer and Laser printer sales and Inkjet Printer sales: eg. 25:1*

This is obviously a customer in desperate need of multidimensional analysis tools. The customer's current database environment can be described by the following:

- 1) They already have a centralized client/server RDBMS
- 2) All reporting and decision making is done at the head office where the database resides

- 3) The offices are connected via high speed Wide Area Network connections
- 4) The current database is approximately 150GB
- 5) All of the sales staff have recently been upgraded to the latest computers

OLAP architecture consists of three primary modules: the GUI (graphical user interface, OLAP Analytical Processing Logic and OLAP data processing logic. It is possible to put all three components into one system but this configuration is costly and not recommended. The reason being is that a very powerful computer is necessary to house and run efficiently all the OLAP functionality. This may be required if the data analysts are power users that really need to have high speed performance and quicker processing. The drawback is that if there are more than one analyst, they would each have to have their own private copy of the data that needs to be synchronized with everyone else to make sure that they are reporting on the same information. This will obviously become an IT support nightmare.

A more common approach can be illustrated below by the OLAP architecture diagram in Figure 13.7 from the text.

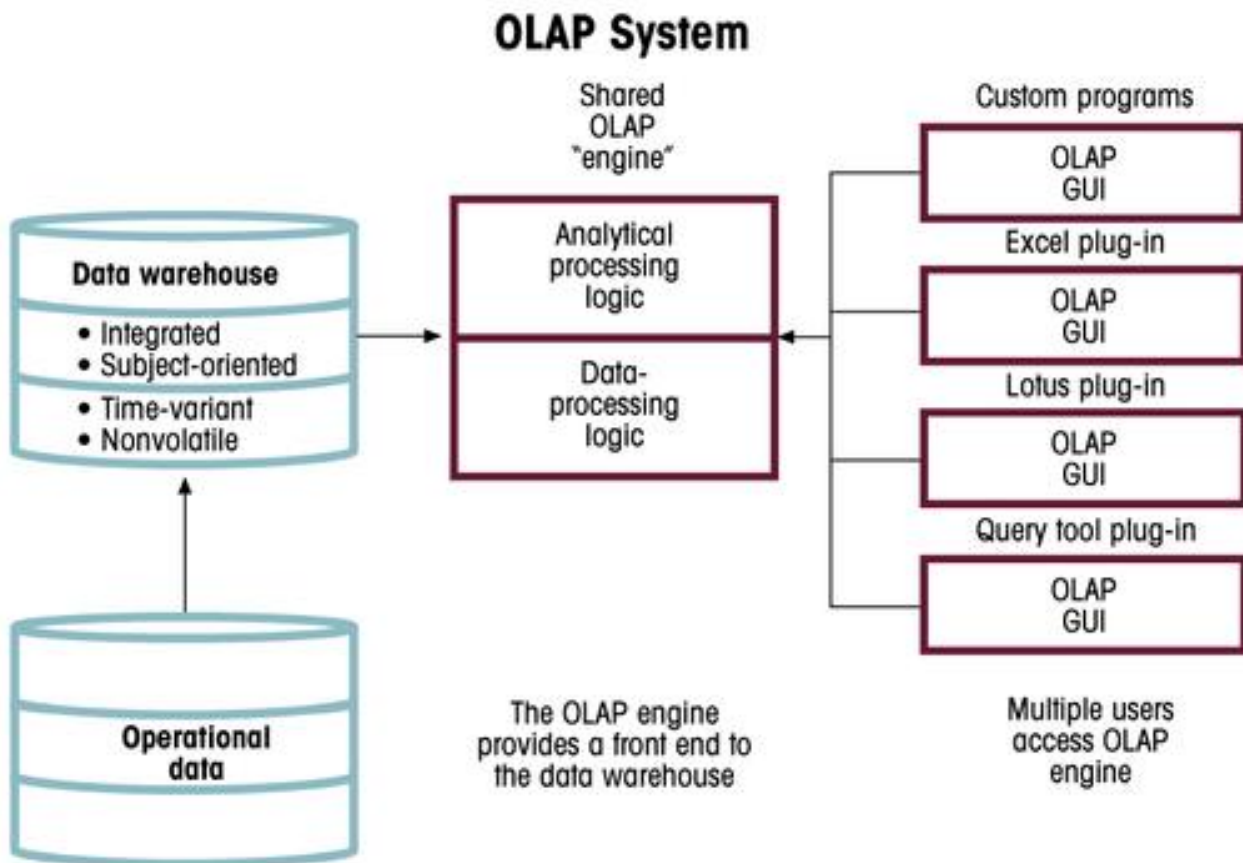


FIGURE 13.7 OLAP SERVER ARRANGEMENT

Here, only the OLAP GUI is moved to the client workstations. The OLAP engine, consisting of the Analytical Processing Logic and the Data Processing Logic would reside on a shared server. This configuration depicts the data warehouse and OLAP components as different systems. The data warehouse extracts, filters and transforms the operational data into data warehouse data for further analysis by the OLAP server which basically provides the front end component in this arrangement. Other options include having the OLAP engine directly access the Operational Data and then have it stored in a multidimensional structure for further processing. And to increase performance, small local data marts can be provided to the analysts for more specialized reporting such as a sales analyst who only needs to work with sales data.

This particular customer should utilize a Relational OLAP system rather than a Multidimensional OLAP system. Some of the information that they wish to gain from the system are very standard and some are more ad-hoc in nature. In a MOLAP system, every dimension must be pre-defined and reconfiguration is required in order to accommodate new dimensions. ROLAP is much more flexible because it allows ad-hoc manipulation. The customer's database has already exceeded 150GB. MOLAP systems generally work better with smaller databases smaller than 50GB. The reason being is that multidimensional design increases the storage requirements many times because of the redundancy. ROLAP in general is much easier to maintain from an IT perspective.