ICT285 Assignment 1 : 33170193 (student ID) Jin Cherng Chong

Assumption:

- The date attribute is in string “DD/MM/YYYY” format.

1a)

RESTRICT City = ‘Perth’ (CUSTOMER) → T1

PROJECT FirstName, LastName (T1) → Solution

1b)

RESTRICT Date = ‘01/08/2020’ (INVOICE) → T1

T1* T1.CustID = CUSTOMER.CustID CUSTOMER → T2

PROJECT FirstName, LastName (T2) → Solution

1c)

RESTRICT ItemName = ‘Back Scratchers’ (ITEM) → T1

PROJECT UnitPrice (T1) → Solution

1d)

RESTRICT ItemName = ‘Back Scratchers’ (ITEM) → T1

RESTRICT Quantity > 10 (INVOICE_ITEM) → T2

T2* ItemNumber.T2 = T1.ItemNumber T1 → T3

INVOICE* INVOICE.CustID = CUSTOMER.CustID CUSTOMER → T4

T4* InvoiceNumber.T4 = T3.InvoiceNumber T3 → T5

PROJECT FirstName, LastName (T5) → SOLUTION

1e)

RESTRICT FirstName = ‘Peter’ AND LastName = ‘Simpson’ (CUSTOMER)→ T1

RESTRICT Date = ‘01/08/2020’ (INVOICE) → T2

T2* CustID.T2 = T1.CustID (T1) → T3

INVOICE_ITEM* INVOICE_ITEM.ItemNumber = ITEM.ItemNumber ITEM → T4

T4* T4.InvoiceNumber = T3.InvoiceNumber (T3) → Final

PROJECT ItemName, Quantity (Final) → SOLUTION

1f)

RESTRICT FirstName = ‘Homer’ AND LastName = ‘Griffin’ (CUSTOMER) → T1

INVOICE* INVOICE.CustID = T1.CustID T1 →T2

PROJECT Date (T2) → Solution

1g)

RESTRICT ItemName = ‘Back Scratcher’ (ITEM) → T1

INVOICE_ITEM* INVOICE_ITEM.ItemNumber = T1.ItemNumber T1 → T2

T2* T2.InvoiceNumber = INVOICE.InvoiceNumber INVOICE → T3

T3* T3.CustID = CUSTOMER.CustID CUSTOMER → T4

PROJECT FirstName, LastName (T4) → Final1

RESTRICT ItemName = ‘Hair Remover’ (ITEM) → T5

INVOICE_ITEM* INVOICE_ITEM.ItemNumber = T5.ItemNumber T5 → T6

T6* T6.InvoiceNumber = INVOICE.InvoceNumber INVOICE → T7

T8* T8.CustID = CUSTOMER.CustID CUSTOMER → T8

PROJECT FirstName, LastName (T8) → Final2

Final1 OR Final2 → Solution

1h)

RESTRICT ItemName = ‘Back Scratcher’ (ITEM) → T1

INVOICE_ITEM* INVOICE_ITEM.ItemNumber = T1.ItemNumber T1 → T2

T2* T2.InvoiceNumber = INVOICE.InvoiceNumber INVOICE → T3

T3* T3.CustID = CUSTOMER.CustID CUSTOMER → T4

PROJECT FirstName, LastName (T4) → Final1

RESTRICT ItemName = ‘Hair Remover’ (ITEM) → T5

INVOICE_ITEM* INVOICE_ITEM.ItemNumber = T5.ItemNumber T5 → T6

T6* T6.InvoiceNumber = INVOICE.InvoceNumber INVOICE → T7

T8* T8.CustID = CUSTOMER.CustID CUSTOMER → T8

PROJECT FirstName, LastName (T8) → Final2

Final1 MINUS Final2 → Solution

1i)

RESTRICT ItemName = ‘Back Scratcher’ (ITEM) → T1

INVOICE_ITEM* INVOICE_ITEM.ItemNumber = T1.ItemNumber T1 → T2

T2* T2.InvoiceNumber = INVOICE.InvoiceNumber INVOICE → T3

T3* T3.CustID = CUSTOMER.CustID CUSTOMER → T4

PROJECT FirstName, LastName (T4) → Final1

RESTRICT ItemName = ‘Hair Remover’ (ITEM) → T5

INVOICE_ITEM* INVOICE_ITEM.ItemNumber = T5.ItemNumber T5 → T6

T6* T6.InvoiceNumber = INVOICE.InvoiceNumber INVOICE → T7

T7* T7.CustID = CUSTOMER.CustID CUSTOMER → T8

PROJECT FirstName, LastName (T8) → Final2

Final1 AND Final2 → Solution

Ij)

INVOICE* INVOICE.CustID = CUSTOMER.CustID CUSTOMER → T1

T1 LEFT OUTER JOIN T1.InvoiceNumber = INVOICE_ITEM.InvoiceNumber INVOICE_ITEM → T2

ITEM LEFT OUTER JOIN ITEM.ItemNumber = T2.InvoiceNumber T2 → T3

PROJECT FirstName, LastName, ItemNumber (T3) → Final1

PROJECT ItemNumber (ITEM) → Final2

Final1 DIVIDEBY Final2 → Solution

Assume every single workID must have a TransactionID. Any work that doesn't have a transactionID will be considered

2a)

SELECT WORKID, TITLE, COPY, MEDIUM, DESCRIPTION, FIRSTNAME || ' ' || LASTNAME AS FULLNAME

FROM dtoohey.WORK, dtoohey.ARTIST

WHERE WORK.ARTISTID = ARTIST.ARTISTID

AND DESCRIPTION LIKE '%Surrealist%';

	WORKID	TITLE	COPY	MEDIUM	DESCRIPTION	FULLNAME	
1	521	The Tilled Field	788/1000	High Quality Limited Print	Early Surrealist style	Joan	Miro
2	522	La Lecon de Ski	353/500	High Quality Limited Print	Surrealist style	Joan	Miro

2b)

SELECT WORK.WORKID, TITLE, COPY, MEDIUM, DESCRIPTION, FIRSTNAME || ' ' || LASTNAME AS FULLNAME, ACQUISITIONPRICE, ASKINGPRICE

FROM dtoohey.TRANS, dtoohey.WORK, dtoohey.ARTIST

WHERE TRANS.WORKID = WORK.WORKID

AND WORK.ARTISTID = ARTIST.ARTISTID

AND ASKINGPRICE > 400

AND DATESOLD IS NULL;

	WORKID	TITLE	COPY	MEDIUM	DESCRIPTION	FULLNAME		ACQUISITIONPRICE	ASKINGPRICE
1	588	Universal Field	114/500	High Quality Limited Print	Northwest School Abstract Expressionist style	Mark	Tobey	250	500
2	565	Farmer's Market #2	268/500	High Quality Limited Print	Northwest School Abstract Expressionist style	Mark	Tobey	250	500
3	596	Surf and Bird	366/500	High Quality Limited Print	Northwest School Expressionist style	Morris	Graves	250	500
4	595	Surf and Bird	365/500	High Quality Limited Print	Northwest School Expressionist style	Morris	Graves	250	500
5	594	Surf and Bird	362/500	High Quality Limited Print	Northwest School Expressionist style	Morris	Graves	250	500
6	593	Surf and Bird	Unique	Gouache	26.5 x 29.75 in. - Signed	Morris	Graves	25000	50000
7	578	Mid-Century Hibernation	362/500	High Quality Limited Print	Northwest School Expressionist style	Morris	Graves	250	500

2c)

```
SELECT TITLE
FROM dtoohey.WORK
GROUP BY TITLE
HAVING COUNT(TITLE) = 2;
```

	TITLE
1	Farmer's Market #2
2	The Fiddler

2d)

```
SELECT FIRSTNAME || ' ' || LASTNAME AS FULLNAME, DATEDECEASED - DATEOFBIRTH AS AGEOFDEATH
FROM dtoohey.ARTIST
WHERE DATEDECEASED IS NOT NULL;
```

	FULLNAME		AGEOFDEATH
1	Joan	Miro	90
2	Wassily	Kandinsky	78
3	Paul	Klee	61
4	Henri	Matisse	85
5	Marc	Chagall	98
6	John Singer	Sargent	69
7	Mark	Tobey	86
8	Paul	Horiuchi	93
9	Morris	Graves	81
10	Bloxham Smythe	Julio	21

2e)

```
SELECT FIRSTNAME || ' ' || LASTNAME As FullName, count(*) As NumberOfArt
FROM dtoohey.WORK, dtoohey.ARTIST
WHERE WORK.ARTISTID = ARTIST.ARTISTID
GROUP BY FIRSTNAME || ' ' || LASTNAME
ORDER BY NumberOfArt asc;
```

	FULLNAME		NUMBEROFART
1	Paul	Klee	1
2	Wassily	Kandinsky	2
3	Henri	Matisse	2
4	Joan	Miro	2
5	Marc	Chagall	3
6	John Singer	Sargent	4
7	Paul	Horiuchi	4
8	Mark	Tobey	9
9	Morris	Graves	9

2f)

```
SELECT WORK.WORKID, TITLE, FIRSTNAME || ' ' || LASTNAME AS FULLNAME
FROM dtoohey.WORK, dtoohey.TRANS, dtoohey.artist
WHERE TRANS.WORKID = WORK.WORKID
AND WORK.ARTISTID = ARTIST.ARTISTID
AND SALESPRICE > ACQUISITIONPRICE
AND SALESPRICE >
   (SELECT AVG(SALESPRICE)
   FROM dtoohey.TRANS);
```

	WORKID	TITLE	FULLNAME	
1	500	Memories IV	Paul	Horiuchi
2	500	Memories IV	Paul	Horiuchi
3	548	Night Bird	Morris	Graves
4	561	Sunflower	Morris	Graves
5	570	Untitled Number 1	Mark	Tobey
6	571	Yellow Covers Blue	Paul	Horiuchi

2g)

SELECT extract(YEAR FROM DATESOLD) as YEAR, SUM(SALESPRICE) as YearSale, count(extract(YEAR FROM DATESOLD)) AS NUMSOLD

FROM dtoohey.TRANS

Group by extract(YEAR FROM DATESOLD)

HAVING extract(YEAR FROM DATESOLD) IS NOT NULL;

SQL | All Rows Fetched: 6 in 0.028 seconds

	YEAR	YEARSALE	NUMSOLD
1	2009	160575	10
2	2010	11950	5
3	2007	43000	2
4	2011	475	1
5	2008	32050	8
6	2012	350	1

2h)

SELECT ARTISTID, Count(ArtistId) AS workSold

FROM dtoohey.WORK, dtoohey.TRANS

WHERE TRANS.WORKID = WORK.WORKID

```
GROUP BY ARTISTID
ORDER BY Count(ArtistId) DESC
FETCH FIRST 1 ROWS ONLY;
```

	ARTISTID	WORKSOLD
1	17	9

2J)

```
SELECT FIRSTNAME || ' ' || LASTNAME AS FULLNAME
FROM dtoohey.CUSTOMER
WHERE NOT EXISTS
   (SELECT *
    FROM dtoohey.ARTIST
    WHERE NOT EXISTS
      (SELECT *
      FROM dtoohey.CUSTOMER_ARTIST_INT
      WHERE CUSTOMER_ARTIST_INT.CUSTOMERID = CUSTOMER.CUSTOMERID
      AND CUSTOMER_ARTIST_INT.ARTISTID = ARTIST.ARTISTID));
```

SQL | All Rows Fetched: 1 in 0.022 seconds

	FULLNAME
1	David Smith

3a)

```
CREATE TABLE PRESENTER (
   PresenterNo NUMBER(9),
   PresenterName VARCHAR2(20) NOT NULL,
```

```
  Biography VARCHAR2(20) NOT NULL,
  InstitutionName VARCHAR2(35) NOT NULL,
  CONSTRAINT PresenterPK PRIMARY KEY(PresenterNo)
);
```

```
Name            Null?    Type
--------------- -------- ------------
PRESENTERNO     NOT NULL NUMBER(9)
PRESENTERNAME   NOT NULL VARCHAR2(20)
BIOGRAPHY       NOT NULL VARCHAR2(20)
INSTITUTIONNAME NOT NULL VARCHAR2(35)
```

3b)

```
CREATE TABLE LECTURE (
  LectureNo NUMBER(15),
  LectureName VARCHAR2(20) NOT NULL,
  Description VARCHAR2(50) NOT NULL,
  Theme VARCHAR2(40) NOT NULL,
  Capacity NUMBER(3) NOT NULL,
  DateAndTime DATE NOT NULL,
  PresenterNo NUMBER(9),
  CONSTRAINT LecturePK PRIMARY KEY(LectureNo),
  CONSTRAINT LecturePresenterFK FOREIGN KEY(PresenterNo)
  REFERENCES PRESENTER(PresenterNo)
  ON DELETE CASCADE);
```

```
Name        Null?    Type
----------- -------- ------------
LECTURENO   NOT NULL NUMBER(15)
LECTURENAME NOT NULL VARCHAR2(20)
DESCRIPTION NOT NULL VARCHAR2(50)
THEME       NOT NULL VARCHAR2(40)
CAPACITY    NOT NULL NUMBER(3)
DATEANDTIME NOT NULL DATE
PRESENTERNO          NUMBER(9)
```

3c)

INSERT INTO PRESENTER(PresenterNo, PresenterName, Biography, InstitutionName)

VALUES (1, 'Jin Chong', 'A Kmart employee', 'Murdoch University');

	PRESENTERNO	PRESENTERNAME	BIOGRAPHY	INSTITUTIONNAME
1	1	Jin Chong	A Kmart employee	Murdoch University

3d)

ALTER TABLE LECTURE

ADD VenueName VARCHAR2(10);

```
Name        Null?    Type
----------- -------- ------------
LECTURENO   NOT NULL NUMBER(15)
LECTURENAME NOT NULL VARCHAR2(20)
DESCRIPTION NOT NULL VARCHAR2(50)
THEME       NOT NULL VARCHAR2(40)
CAPACITY    NOT NULL NUMBER(3)
DATEANDTIME NOT NULL DATE
PRESENTERNO          NUMBER(9)
VENUENAME            VARCHAR2(10)
```

ALTER TABLE LECTURE

ADD CONSTRAINT VenueNameCheck

CHECK (VenueName IN('Building A','Building B','Building C'));

Proof Constraint is added:

SELECT * FROM User_constraints

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	SEARCH_CONDITION_VC	R_OWNER	R_CONSTRAINT_NAM
1	V33170193	LECTUREPRESENTERFK	R	LECTURE	(null)	(null)	V33170193	PRESENTERPK
2	V33170193	VENUENAMECHECK	C	LECTURE	VenueName IN('Building A','Building B','Building C')	VenueName IN('Building A','Building B','Building C')	(null)	(null)

```
INSERT INTO LECTURE(LectureNo, LectureName, Description, Theme, Capacity, DateAndTime, PresenterNo, VenueName)
VALUES (1, 'Maths Exam Tip', 'Exam about math', 'Math', 40, To_Date('01-09-2019 10:00:00', 'DD-MM-YYYY HH24:MI:SS'), 1, 'Jin House');
```

Script Output × | Query Result ×

Task completed in 0.096 seconds

```
*Cause:
*Action:
Name         Null?    Type
----------- -------- ------------
LECTURENO    NOT NULL NUMBER(15)
LECTURENAME  NOT NULL VARCHAR2(20)
DESCRIPTION  NOT NULL VARCHAR2(50)
THEME        NOT NULL VARCHAR2(40)
CAPACITY     NOT NULL NUMBER(3)
DATEANDTIME  NOT NULL DATE
PRESENTERNO           NUMBER(9)
VENUENAME             VARCHAR2(10)

Table LECTURE altered.


Error starting at line : 153 in command -
INSERT INTO LECTURE(LectureNo, LectureName, Description, Theme, Capacity, DateAndTime, PresenterNo, VenueName)
VALUES (1, 'Maths Exam Tip', 'Exam about math', 'Math', 40, To_Date('01-09-2019 10:00:00', 'DD-MM-YYYY HH24:MI:SS'), 1, 'Jin House')
Error report -
ORA-02290: check constraint (V33170193.VENUENAMECHECK) violated
```

3e)

	LECTURENO	LECTURENAME	DESCRIPTION	THEME	CAPACITY	DATEANDTIME	PRESENTERNO	VENUENAME
1	1	Maths Exam Tip	Exam about math	Math	50	01/SEP/19	1	Building A
2	2	English Exam Tip	Exam about English	English	20	02/SEP/19	1	Building B

UPDATE lecture

SET Capacity = Capacity + 10;

	LECTURENO	LECTURENAME	DESCRIPTION	THEME	CAPACITY	DATEANDTIME	PRESENTERNO	VENUENAME
1	1	Maths Exam Tip	Exam about math	Math	60	01/SEP/19	1	Building A
2	2	English Exam Tip	Exam about English	English	30	02/SEP/19	1	Building B

4a)

Assumption:

- A patient (patient ID) can't have the same surgery (Item Number) more than once from same doctor (provider number)

Before any problems can be identified the primary key for the relation needs to be identified. We assume the current primary key for the relation is:

PatientID, Item Number, Provider Number

Firstly, anomalies are problems that arise when changes are made to relations with redundant data. In the current design, Insertion anomalies would arise when we want to add another Item in the relation. For example, the addition of a new Item: Medium (Item description) A016 (Item number) to the relation would result in null values for the other attributes of the compound primary key. So even though the attribute Item number is allocated a value A016, the other attributes in the key such as Patient ID and provider number are null. Therefore, the entity integrity constraint is broken. The entity integrity constraint specifies that the primary key value can't be null.

Another potential problem that may arise with the current design is an update anomaly. For example, an update to patient 437 (patient ID) date of birth from 4/08/1989 to 4/07/1989 in one record would not update all the other instances of the same data as well. This leaves potential inconsistencies with the data where a single person would have two different date of births, which doesn't make sense.

A third possible problem with current design is that it allows deletion anomalies to arise. A deletion anomaly is where the deletion of other attributes causes certain needed attributes to be lost. In this relation, a deletion to attribute A013 (Item Number) by provider S55768 will cause information

about patient Bilstein (Patient Name) to be lost as well. Since having a primary key attribute being null is unacceptable

4b)

Direct dependencies (original design)-

Provider Number → Doctor Name (partial functional dependencies)

Patient ID → Patient Name, Patient DOB (partial functional dependencies)

Item Number → Item Description, Fee (partial functional dependencies)

Patient ID, Item Number, Provider Number (primary key) → Consult date

The current design is in first normal form meaning there is no repeating groups or nesting in the relation. Also, there are partial functional dependencies as illustrated above. So, for example, Doctor name, which is a non-primary key, is determined by provider number, which is only a part of primary key. Therefore, through the progress of normalisation we convert the first normal form design to second normal form design to remove some of the modification anomalies identified.

Relation1 (Provider Number, Doctor Name)

Relation2 (Patient ID, Patient Name, Patient DOB)

Relation3 (Item Number, Item Description, Fee)

Relation4 (**Patient ID, Item Number, Provider Number**, Consult Date)

Legend → Primary key (underlined), **Foreign key** (bold)

The current design is now in at least second normal form meaning there are no partial function dependencies. However, given there are no transitive functional dependencies as well, the design is really in third normal form. A transitive functional dependency is where a non-key attribute is

determined by another non-key attribute. Since the new design is in third normal form the problems associated with the previous design are resolved.

Firstly, with the new design there is no possibility that insertion anomalies can arise. For example, the addition of a new item: Medium (Item description) A016 (Item number) to the relation3 would not result in the primary key attribute having a null value since relation3 has only a one attribute primary key. So, adding a new item to relation3 would automatically result in the primary key being not null.

Secondly, there is now no possibility that update anomalies can arise. For example, an update to patient 437 (patient ID) date of birth from 4/08/1989 to 4/07/1989 in one record will now update all instances of the data. The reason why is because relation3 will only have one instance (record) for each patient, thus any updates to patient 437 will apply to essentially all the instances due to the design.

Thirdly, the new design also addresses the issue of deletion anomalies arising. In the scenario where attribute A013 (Item Number) is deleted in relation3 the only other attributes deleted are the ones directly related to A013; so, Item description and fee would be deleted. The patient Bilstein information would not be deleted as it is in a separate relation.

The set of relations for the new design also supports the lossless join property and dependency preserving. The lossless join property allows the set of relations to be joined back together to get the original relation. So, for relation1, relation2 and relation3 the primary key links to the foreign key in relation4. In addition, the new design preserves all the functional dependencies found in the original relation. Both the lossless join property and dependency preserving allow for a good design.

5)

Assume

- Assume customer can pay full amount of bill only
- Assume customer does not want to pay in mix payments (i.e half through cash, and half paypal)
- Assume a WaterMeter for an address gets replaced every time a new customer is assigned to it
- Assume a bill is created immediately after a new customer is assigned a WaterMeter
- Assume WaterMeter can have no customers assigned to it
- Assume MeterReaders have to read at least one WaterMeter per month. Failure do so will get them fired. If the MeterReader was to go on vacation they would no longer be employed since they are employed as a casual

Drip Drip Water Company ERD