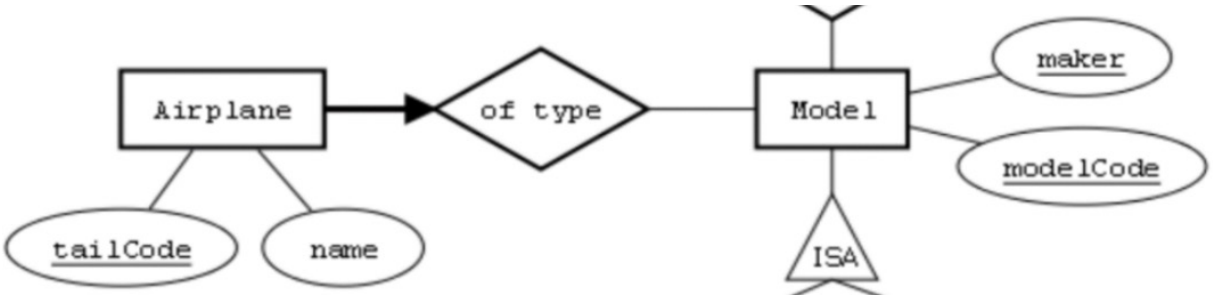
Isys2120 Lab13

A(i)

- Explain

- This is a composite attribute: the attribute name is made from two separate components, one called given, the other called family

A(ii)



- State whether the diagram allows for a situation where airplane with tailCode ‘QXS213’ named ‘Spirit of Sydney’ is of type ‘373’ from maker Boeing, and also the airplane with tailcode ‘VGY961’ named ‘HappyDays’ is of type ‘373’ from maker ‘Airbus’. Explain the specific features of the diagram which either allows this, or prevents this situation?
- The diagram allows this. Having the two airplanes is allowed, since the diagram shows tailCode as primary key (by underlining), and these two have different tailCode values. The diagram says that every airplane ‘has type’ of exactly one model (by thick arrow from Airplane to of-type), and that is true in this case. The diagram allows two models with same modelCode and different maker, because the primary key for Model is the composite of (maker and modelCode), shown by underlining both attributes.

A(iii)

```
CREATE TABLE Engineer(
    emp_Id VARCHAR(5),
    name_given VARCHAR(20),
    name_family VARCHAR(20)
    PRIMARY KEY emp_Id);
CREATE TABLE Airplane (
     tailCode VARCHAR(6),
      name VARCHAR(10),
      maker VARCHAR(10) NOT NULL,
      modelCode VARCHAR(10) NOT NULL,
      PRIMARY KEY (tailCode),
      FOREIGN KEY (maker, modelCode) REFERENCES Model(maker,
modelCode));
CREATE TABLE Model (
     maker VARCHAR(10),
      modelCode VARCHAR(10),
      PRIMARY KEY (maker, modelCode));
CREATE TABLE Airliner(
     maker VARCHAR(10),
     modelCode VARCHAR(10),
     maxPassengers INTEGER,
     FOREIGN KEY (maker, modelCode) REFERENCES Model(maker, modelCode));
 CREATE TABLE CargoFreighter(
     maker VARCHAR(10),
     modelCode VARCHAR(10),
     maxCargo INTEGER,
     FOREIGN KEY (maker, modelCode) REFERENCES Model(maker, modelCode));
CREATE TABLE Certified(
     emp_id VARCHAR(5),
     maker VARCHAR(10),
     modelCode VARCHAR(10),
     PRIMARY KEY (emp_id, maker, modelCode),
     FOREIGN KEY emp_id REFERENCES Engineer(emp_id),
     FOREIGN KEY (maker, modelCode) REFERENCES Model(maker, modelCode));
```

B(i)

- Find the SId and SDate of shipments that contain a part whose PName is ‘WashingMachine’

SELECT Sid, SDate

FROM Shipment NATURAL JOIN (Contains NATURAL JOIN Part)

WHERE Pname = ‘WashingMachine’;

// Many other solutions possible, including with subquery, or putting the join condition is where clause

B(ii)

- produce a report showing the various Customers who have made orders, and for each CId, it gives the number of orders made by that customer.

SELECT CId, COUNT(DISTINCT Oid)

FROM Order

GROUP BY Cid;

// If you also want to show customers who made no orders, with zero as the number of orders, you need to use OUTER JOIN of Customer and Order table

B(iii)

- find the SId of whichever shipment has the most recent SDate, among all shipments for customer 56 (if several shipments for this customer have equally recent SDate, the calculation should report on all of them)

```
SELECT SId
FROM Shipment
WHERE Cid = 56 AND SDate = (SELECT MAX(SDate)
                            FROM Shipment
                            WHERE Cid = 56);
```

B(iv)

- achieve the following security goals: User Kelly can access information about the SId and SDate of any shipment whose SDate is after January 1 2022; Kelly should not have access to information about other shipments; Kelly should not have access to the Cid column of the shipments data.

```
CREATE VIEW ForKelly AS
SELECT SId, SDate FROM Shipment WHERE Sdate > '01-01-2022';
GRANT SELECT ON ForKelly TO Kelly;
```

B(v)

- Give an information request in English, whose answer is calculated by $\pi_{PName}\left(\sigma_{PaidStatus='Y'}(Order \bowtie Part)\right)$

Find the PName of the parts which appear in an order whose PaidStatus is ‘Y’

C(i)

- Calculate the attribute closure flightCode+ Show your working.

Answer: (flightCode destination)

Working:

Start with flightCode

Because *flightCode --> destination*, get flightCode, destination

Nothing more can be added (no other fds with lhs contained in this list)

- *Crew(flightCode,destination,date,tailCode, acName,airline,emp_id,name,title)*
- *flightCode --> destination*
- *flightCode, date --> tailCode*
- *tailCode --> acName, airline*
- *emp_id --> name, title*

C(ii)

- Calculate the attribute closure (flightCode, date)+ Show your working.

Answer: flightCode date destination tailCode acName airline

Working:

Start with (flightCode date)

Because *flightCode --> destination*, get (flightCode date destination)

Because *flightCode, date --> tailCode* get (flightCode date destination tailCode)

Because *tailCode --> acName, airline*

get (flightCode date destination tailCode acName airline)

Nothing more can be added (no more fds with lhs contained in this list)

- *Crew(flightCode,destination,date,tailCode, acName,airline,emp_id,name,title)*
- *flightCode --> destination*
- *flightCode, date --> tailCode*
- *tailCode --> acName, airline*
- *emp_id --> name, title*

C(iii)

- State whether the relation Crew is in BCNF. Show your working.
- Crew is not in BCNF
- Working: We calculated the attribute closure of lhs of *flightCode --> destination and this closure* is not the full list of attributes; that is, lhs of this fd is not a superkey
- Thus, the relation is not in BCNF

- *Crew(flightCode,destination,date,tailCode, acName,airline,emp_id,name,title)*
- *flightCode --> destination*
- *flightCode, date --> tailCode*
- *tailCode --> acName, airline*
- *emp_id --> name, title*

C(iv)

- Give a candidate key for the relation Crew. Justify your claim that this is a candidate key.
- Answer: (flightCode, date, emp_id) is candidate key
- Working: calculate attribute closure (flightCode, date, emp_id)+
- First steps are like for C(ii), get to flightCode date emp_id destination tailCode acName airline
- Then use *emp_id --> name, title* to expand list to flightCode date emp_id destination tailCode acName airline name title
- This list contains all the attributes, so (flightCode date emp_id) is superkey
- Check attribute closure of each subset of this, none gives all attributes of the relation; thus this is candidate key

- *Crew(flightCode,destination,date,tailCode, acName,airline,emp_id,name,title)*
- *flightCode --> destination*
- *flightCode, date --> tailCode*
- *tailCode --> acName, airline*
- *emp_id --> name, title*

C(v)

- Give a decomposition of the relation Crew, into two or more relations, with the properties that the decomposition is both lossless-join and dependency- preserving. Explain how you know that the decomposition has these properties.
- There are several solutions, depending on which fd you use to do the decomposition
- Eg use *flightCode, date --> tailCode*
- This would give Crew1(flightCode, date, tailCode) and Crew2(flightCode, destination, date, acName, airline, emp_id, name, title)
    - Crew1 is the columns in the splitting fd; Crew2 is all the columns except for rhs of splitting fd
- Use of an fd to split tells us that the decomposition is lossless-join;
- for Crew1 we still have the original fd *flightCode, date --> tailCode*
- And for Crew2 we still have *flightCode --> destination*, *tailCode --> acName, airline, emp_id --> name, title*
- So all original fds are found in one or other decomposed relation; thus decomposition is dependency preserving

- *Crew(flightCode,destination,date,tailCode, acName,airline,emp_id,name,title)*
- *flightCode --> destination*
- *flightCode, date --> tailCode*
- *tailCode --> acName, airline*
- *emp_id --> name, title*