## Characteristics of the cases

Case 1 – Vehicle rental services

unlimited vehicles available → no need to check availability
time-delay transaction → two stages handling (deposit first, rent/balance later)
computation → a bit complicated due to specific rents per durations of hires

Case 2 – Flight reservations

unnormalised table with data → no need to handle data entry and data conversion
additional requirements in text expression → a bit complicated in data model design
listed output requirements → need to write queries although rather straightforward
need to explain the objectives of the output

Case 3 – Computer laboratory booking

limited availability → has to check availability before confirming a request
matching requests with allocated → compare two lists and migrate successful requests
future enhancement → should consider how to cater for setting up priority

Case 4 – Community virus test

unnormalised table of data → no need to handle data entry and data processing
additional requirements in text expression → a bit complicated in data model design
computation → calculate age based on date of birth
determine validity based on two criteria

Case 5 – Round robin league

score recording, not match planning → no need to generate the schedule of matches
computation → complicated due to:
match scores to be converted to championship points
standing championship points be available per round
champion → to be determined manually with the support of inter-team match results

Case 6 – IT help desk

unnormalised table of data → no need to handle data entry and data processing
additional requirements in text expression → a bit complicated in data model design
free text to be formulated → need to consider how problem and problem type structured
future enhancement → possible to find solutions for listed problems

# Case 1 – Vehicle rental services

## Business needs

- record each transaction by manual data entry including name, phone, date, vehicle ID, vehicle type, and the time started only
- compute the total deposit to be paid
- on return of the vehicles, carry out manual data entry for the time returned
- compute the total rent and the balance to customer
- be able to handle queries including outstanding vehicles not yet returned, etc.

## Database design

Three tables and one logical view

### Table DEPOSIT FORM

As per paper-based deposit form, this table keeps the header information and duration of hire in hours*

| | Name | Data type | Primary Key | Foreign Key | Unique | Check | Not NULL | Collate |
|---|---|---|---|---|---|---|---|---|
| 1 | DF_PHONE | CHAR | | | | | | |
| 2 | DF_DATE | DATE | | | | | | |
| 3 | DF_NAME | CHAR | | | | | | |
| 4 | DF_TIME_S | TIME | | | | | | |
| 5 | DF_TIME_R | TIME | | | | | | |
| 6 | DF_HRS | INT | | | | | | |

### Table VEH_RENTED

As per paper-based deposit form, this table keeps each vehicle rented including vehicle identity and its vehicle type

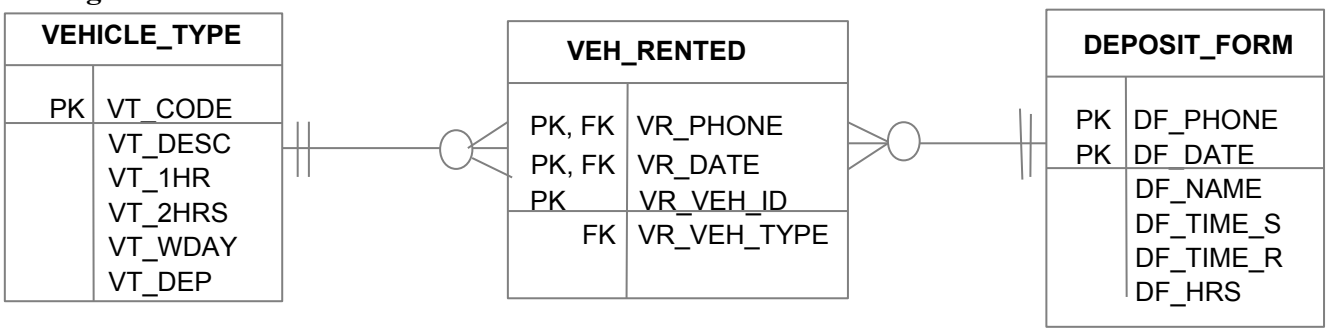
| | Name | Data type | Primary Key | Foreign Key | Unique | Check | Not NULL | Collate |
|---|---|---|---|---|---|---|---|---|
| 1 | VR_PHONE | CHAR | | | | | | |
| 2 | VR_DATE | DATE | | | | | | |
| 3 | VR_VEH_ID | CHAR | | | | | | |
| 4 | VR_VEH_TYPE | CHAR | | | | | | |

### Table VEHICLE_TYPE

Same as the Rent and Deposit legend in the paper-based deposit form, this table keeps basic information of vehicle type including rents as per hours hired and the deposit to be paid

| | Name | Data type | Primary Key | Foreign Key | Unique | Check | Not NULL | Collate |
|---|---|---|---|---|---|---|---|---|
| 1 | VT_CODE | CHAR | | | | | | |
| 2 | VT_DESC | CHAR | | | | | | |
| 3 | VT_1HR | INT | | | | | | |
| 4 | VT_2HRS | INT | | | | | | |
| 5 | VT_WDAY | INT | | | | | | |
| 6 | VT_DEP | INT | | | | | | |

## ER Diagram




* DF_HRS is a derived attribute that can be computed by running a SQL upon return of vehicles. It is justified as the frequency of use is high and the frequency of change is very low.

Initial setup: VEHICLE_TYPE: All types with descriptions, rents, and deposits

| | VT_CODE | VT_DESC | VT_1HR | VT_2HRS | VT_WDAY | VT_DEP |
|---|---|---|---|---|---|---|
| 1 | K | Tricycle for kid | 20 | 35 | 50 | 100 |
| 2 | F | 4-wheel Family Size | 50 | 70 | 90 | 200 |
| 3 | A | Bicycle for adult | 25 | 40 | 60 | 150 |

Data entry by end-users: DEPOSIT_FORM and VEH_RENTED: All information except DF_HRS

| | DF_PHONE | DF_DATE | DF_NAME | DF_TIME_S | DF_TIME_R | DF_HRS |
|---|---|---|---|---|---|---|
| 1 | 90030288 | 2022-03-07 | David Chung | 11:45 | 12:30 | 1 |
| 2 | 90632398 | 2022-03-07 | Becky Wong | 12:05 | 14:15 | 3 |
| 3 | 69828698 | 2022-03-07 | Cecil Tang | 12:20 | NULL | NULL |

| | VR_PHONE | VR_DATE | VR_VEH_ID | VR_VEH_TYPE |
|---|---|---|---|---|
| 1 | 69828698 | 2022-03-07 | B032 | A |
| 2 | 69828698 | 2022-03-07 | G629 | A |
| 3 | 69828698 | 2022-03-07 | R052 | A |
| 4 | 90632398 | 2022-03-07 | G668 | F |
| 5 | 90632398 | 2022-03-07 | B192 | A |
| 6 | 90030288 | 2022-03-07 | H352 | F |
| 7 | 90030288 | 2022-03-07 | Y638 | K |
| 8 | 90030288 | 2022-03-07 | R160 | K |

Data to be computed: DEPOSIT_FORM: DF_HRS

The following SQL is to be run upon return of vehicles:

```
UPDATE DEPOSIT_FORM
   SET DF_HRS = CAST((JulianDay(DF_TIME_R) - JulianDay(DF_TIME_S))*24 AS INTEGER) + 1;
```

Sample results:

| | DF_PHONE | DF_DATE | DF_NAME | DF_TIME_S | DF_TIME_R | DF_HRS |
|---|---|---|---|---|---|---|
| 1 | 90030288 | 2022-03-07 | David Chung | 11:45 | 12:30 | 1 |
| 2 | 90632398 | 2022-03-07 | Becky Wong | 12:05 | 14:15 | 3 |
| 3 | 69828698 | 2022-03-07 | Cecil Tang | 12:20 | NULL | NULL |

**Logical View VEH_RENTED_EXT**

This logical view is to extend VEH_RENTED that keeping deposit, duration of hire, rent thereon.

```
CREATE VIEW VEH_RENTED_EXT AS
  SELECT VR_PHONE, VR_DATE, VR_VEH_ID, VR_VEH_TYPE, VT_DEP AS VR_DEP, DF_HRS, VT_WDAY AS VR_RENT
    FROM VEH_RENTED, VEHICLE_TYPE, DEPOSIT_FORM
   WHERE VR_VEH_TYPE = VT_CODE AND VR_PHONE = DF_PHONE AND VR_DATE = DF_DATE AND DF_HRS > 2
UNION
  SELECT VR_PHONE, VR_DATE, VR_VEH_ID, VR_VEH_TYPE, VT_DEP AS VR_DEP, DF_HRS, VT_2HRS AS VR_RENT
    FROM VEH_RENTED, VEHICLE_TYPE, DEPOSIT_FORM
   WHERE VR_VEH_TYPE = VT_CODE AND VR_PHONE = DF_PHONE AND VR_DATE = DF_DATE AND DF_HRS = 2
UNION
  SELECT VR_PHONE, VR_DATE, VR_VEH_ID, VR_VEH_TYPE, VT_DEP AS VR_DEP, DF_HRS, VT_1HR AS VR_RENT
    FROM VEH_RENTED, VEHICLE_TYPE, DEPOSIT_FORM
   WHERE VR_VEH_TYPE = VT_CODE AND VR_PHONE = DF_PHONE AND VR_DATE = DF_DATE AND DF_HRS = 1
UNION
  SELECT VR_PHONE, VR_DATE, VR_VEH_ID, VR_VEH_TYPE, VT_DEP AS VR_DEP, 0 AS DF_HRS, 0 AS VR_RENT
    FROM VEH_RENTED, VEHICLE_TYPE, DEPOSIT_FORM
   WHERE VR_VEH_TYPE = VT_CODE AND VR_PHONE = DF_PHONE AND VR_DATE = DF_DATE AND DF_HRS IS NULL;
```

Sample results:

| | VR_PHONE | VR_DATE | VR_VEH_ | VR_VEH_ | VR_DEP | DF_HRS | VR_RENT |
|---|---|---|---|---|---|---|---|
| 1 | 69828698 | 2022-03-07 | B032 | A | 150 | 0 | 0 |
| 2 | 69828698 | 2022-03-07 | G629 | A | 150 | 0 | 0 |
| 3 | 69828698 | 2022-03-07 | R052 | A | 150 | 0 | 0 |
| 4 | 90030288 | 2022-03-07 | H352 | F | 200 | 1 | 50 |
| 5 | 90030288 | 2022-03-07 | R160 | K | 100 | 1 | 20 |
| 6 | 90030288 | 2022-03-07 | Y638 | K | 100 | 1 | 20 |
| 7 | 90632398 | 2022-03-07 | B192 | A | 150 | 3 | 60 |
| 8 | 90632398 | 2022-03-07 | G668 | F | 200 | 3 | 90 |

**Test results**

(1) Deposit to be paid by selected transaction:

```
SELECT VR_PHONE, VR_DATE, SUM(VR_DEP)
  FROM VEH_RENTED_EXT
 WHERE VR_PHONE = ?
   AND VR_DATE = ?
 GROUP BY VR_PHONE, VR_DATE;
```

Sample results:

| | VR_PHONE | VR_DATE | SUM(VR_DEP) |
|---|---|---|---|
| 1 | 90030288 | 2022-03-07 | 400 |

(2) Rent of a transaction and Balance to customer:

```
SELECT VR_PHONE, VR_DATE, SUM(VR_DEP) as DEPOSIT, SUM(VR_RENT) as RENT,
       (SUM(VR_DEP) - SUM(VR_RENT)) as BALANCE
  FROM VEH_RENTED_EXT
 WHERE VR_PHONE = ?
   AND VR_DATE = ?
 GROUP BY VR_PHONE, VR_DATE;
```

Sample results:

| | VR_PHONE | VR_DATE | DEPOSIT | RENT | BALANCE |
|---|---|---|---|---|---|
| 1 | 90632398 | 2022-03-07 | 350 | 150 | 200 |

(3) Outstanding vehicles not yet returned:

```
SELECT VR_VEH_ID, VT_DESC
  FROM VEH_RENTED_EXT, VEHICLE_TYPE
 WHERE VR_VEH_TYPE = VT_CODE
   AND DF_HRS = 0;
```

Sample results:

| | VR_VEH_ID | VT_DESC |
|---|---|---|
| 1 | B032 | Bicycle for adult |
| 2 | G629 | Bicycle for adult |
| 3 | R052 | Bicycle for adult |

**Assumptions and limitations**

- the rent and deposit are set as current, no historical data can be retrieved
- the duration of hire is computed as per Deposit Form basis but not per vehicle basis

**Conclusion**

This design meets the minimium requirements. Should an inventory of vehicles is required, there should have additional tables to handle the inventory and to cater for storing history of hires.

## Case 2 – Flight reservations

### Business needs

- handle special requests in line with flight reservations of passengers
- support report generations on various aspects and cater for further reporting requirements

### Database design

Six tables and two logical views are created. Four tables are master which include CABIN, FLIGHT, PASSENGER, SPECREQ. Two tables are transactional which include RESERVATION, SRL_LINE.

**Table CABIN**

Basic information of cabin code and description

| | CAB_CODE | CAB_CLASS |
|---|---|---|
| 1 | B | Business |
| 2 | F | First |
| 3 | Y | Economy |

CABIN

**Table FLIGHT**

Basic information of flight

| | FLT_NO | FLT_FROM | FLT_TO |
|---|---|---|---|
| 1 | CFX83 | Bangkok | Hong Kong |
| 2 | CFX82 | Hong Kong | Bangkok |
| 3 | CFX37 | Manila | Hong Kong |
| 4 | CFX36 | Hong Kong | Manila |

FLIGHT

**Table PASSENGER**

Basic information of passenger

| | PX_NO | PX_NAME |
|---|---|---|
| 1 | DL935347 | Davis, Ludd |
| 2 | PE045262 | Pang, Esther |
| 3 | YH830168 | Yeung, Helen |
| 4 | SK602439 | Smiths, Kelvin |
| 5 | WA346528 | Wong, Albert |
| 6 | TR364582 | Trevors, Ray |

PASSENGER

**Table SPECREQ**

Basic information of special request

| | SR_CODE | SR_DESC |
|---|---|---|
| 1 | LS11 | Over-sized luggage |
| 2 | WC03 | Use of wheel chairs |
| 3 | MV08 | Vegetarian meal |
| 4 | MC04 | Child meal |

SPECREQ

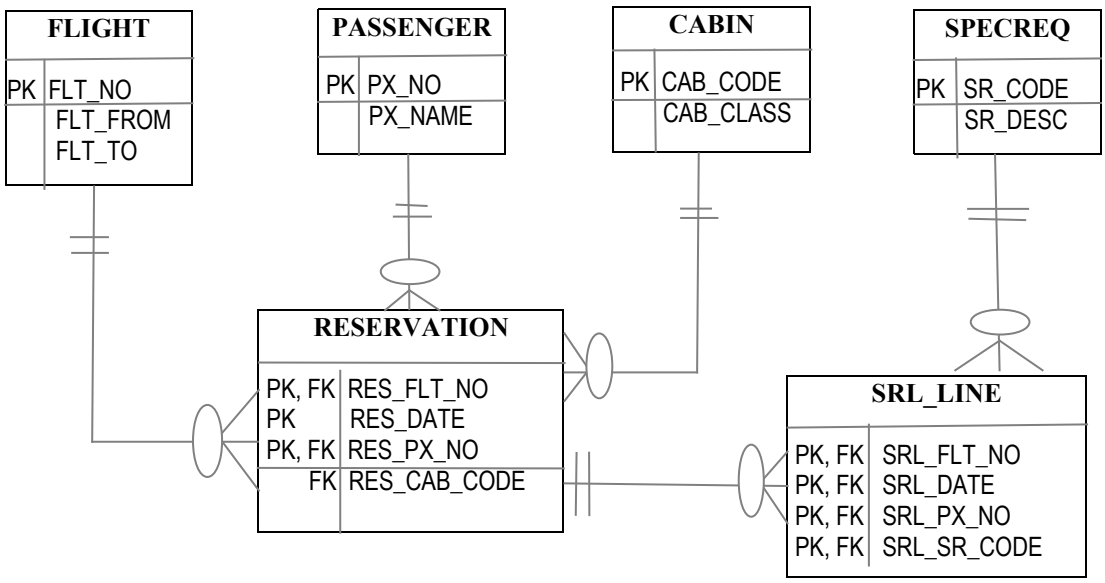
**Table RESERVATION**

As a transactional entity, it keeps all flight reservation entries.

**Table SRL_LINE**

As a composite entity between RESERVATION and SPECREQ, it keeps special request detail line.

### ER Diagram



**Logical View R_BOOK**

This logical view is created for to join core tables with reservation

```
CREATE VIEW R_BOOK AS
  SELECT RES_FLT_NO, RES_DATE, FLT_FROM, FLT_TO, RES_PX_NO, PX_NAME, RES_CAB_CODE,
         CAB_CLASS
    FROM FLIGHT, PASSENGER, CABIN, RESERVATION
   WHERE FLT_NO = RES_FLT_NO AND PX_NO = RES_PX_NO AND CAB_CODE = RES_CAB_CODE;
```

Sample results:

| | RES_FLT_ | RES_DATE | FLT_FROM | FLT_TO | RES_PX_NC | PX_NAME | RES_CAB | CAB_CLA |
|---|---|---|---|---|---|---|---|---|
| 1 | CFX36 | 2022-04-05 | Hong Kong | Manila | TR364582 | Trevors, Ray | Y | Economy |
| 2 | CFX36 | 2022-04-05 | Hong Kong | Manila | WA346528 | Wong, Albert | F | First |
| 3 | CFX36 | 2022-04-05 | Hong Kong | Manila | SK602439 | Smiths, Kelvin | B | Business |
| 4 | CFX37 | 2022-04-05 | Manila | Hong Kong | WA346528 | Wong, Albert | B | Business |
| 5 | CFX37 | 2022-04-05 | Manila | Hong Kong | YH830168 | Yeung, Helen | Y | Economy |
| 6 | CFX36 | 2022-04-06 | Hong Kong | Manila | WA346528 | Wong, Albert | Y | Economy |
| 7 | CFX83 | 2022-04-05 | Bangkok | Hong Kong | PE045262 | Pang, Esther | B | Business |
| 8 | CFX82 | 2022-04-05 | Hong Kong | Bangkok | PE045262 | Pang, Esther | B | Business |
| 9 | CFX82 | 2022-04-06 | Hong Kong | Bangkok | DL935347 | Davis, Ludd | Y | Economy |

**Logical View SR_BOOK**

This logical view is created for to join special request with reservation

```
CREATE VIEW SR_BOOK AS
  SELECT RES_FLT_NO, RES_DATE, RES_PX_NO, PX_NAME, SR_CODE, SR_DESC
    FROM RESERVATION, SRL_LINE, SPECREQ, PASSENGER
   WHERE (SRL_FLT_NO||SRL_DATE||SRL_PX_NO) = (RES_FLT_NO||RES_DATE||RES_PX_NO)
     AND SRL_SR_CODE = SR_CODE AND RES_PX_NO = PX_NO;
```

Sample results:

| | RES_FLT_N | RES_DATE | RES_PX_NC | PX_NAME | SR_CODE | SR_DESC |
|---|---|---|---|---|---|---|
| 1 | CFX36 | 2022-04-05 | WA346528 | Wong, Albert | MV08 | Vegetarian meal |
| 2 | CFX36 | 2022-04-05 | WA346528 | Wong, Albert | LS11 | Over-sized luggage |
| 3 | CFX36 | 2022-04-05 | SK602439 | Smiths, Kelvin | WC03 | Use of wheel chairs |
| 4 | CFX36 | 2022-04-05 | SK602439 | Smiths, Kelvin | MV08 | Vegetarian meal |
| 5 | CFX37 | 2022-04-05 | WA346528 | Wong, Albert | LS11 | Over-sized luggage |
| 6 | CFX37 | 2022-04-05 | WA346528 | Wong, Albert | MV08 | Vegetarian meal |
| 7 | CFX36 | 2022-04-06 | WA346528 | Wong, Albert | MV08 | Vegetarian meal |
| 8 | CFX82 | 2022-04-06 | DL935347 | Davis, Ludd | WC03 | Use of wheel chairs |

**Test results**

(1) List all reservations by passenger number, flight number, date in order to identify frequent flyers:

```
SELECT RES_PX_NO, PX_NAME, RES_FLT_NO, RES_DATE, CAB_CLASS
  FROM R_BOOK
 ORDER BY RES_PX_NO, RES_FLT_NO, RES_DATE;
```

Sample results:

| | RES_PX_NO | PX_NAME | RES_FLT_NO | RES_DATE | CAB_CLASS |
|---|---|---|---|---|---|
| 1 | DL935347 | Davis, Ludd | CFX82 | 2022-04-06 | Economy |
| 2 | PE045262 | Pang, Esther | CFX82 | 2022-04-05 | Business |
| 3 | PE045262 | Pang, Esther | CFX83 | 2022-04-05 | Business |
| 4 | SK602439 | Smiths, Kelvin | CFX36 | 2022-04-05 | Business |
| 5 | TR364582 | Trevors, Ray | CFX36 | 2022-04-05 | Economy |
| 6 | WA346528 | Wong, Albert | CFX36 | 2022-04-05 | First |
| 7 | WA346528 | Wong, Albert | CFX36 | 2022-04-06 | Economy |
| 8 | WA346528 | Wong, Albert | CFX37 | 2022-04-05 | Business |
| 9 | YH830168 | Yeung, Helen | CFX37 | 2022-04-05 | Economy |

*WA346528 Albert Wong is frequent flyer.*

(2) List reservations and special requests of selected passenger in order to identify one's need:

```
SELECT RES_PX_NO, PX_NAME, RES_FLT_NO, RES_DATE, SR_DESC
  FROM SR_BOOK
 WHERE RES_PX_NO = ?;
```

Sample results:

| | RES_PX_NO | PX_NAME | RES_FLT_NO | RES_DATE | SR_DESC |
|---|---|---|---|---|---|
| 1 | WA346528 | Wong, Albert | CFX36 | 2022-04-05 | Over-sized luggage |
| 2 | WA346528 | Wong, Albert | CFX36 | 2022-04-05 | Vegetarian meal |
| 3 | WA346528 | Wong, Albert | CFX37 | 2022-04-05 | Over-sized luggage |
| 4 | WA346528 | Wong, Albert | CFX37 | 2022-04-05 | Vegetarian meal |
| 5 | WA346528 | Wong, Albert | CFX36 | 2022-04-06 | Vegetarian meal |

*WA346528 Albert Wong needs vegetarian meal.*

(3) List passenger counts by flight number and date in order to review popularity of flights:

```
SELECT RES_FLT_NO, RES_DATE, COUNT(RES_PX_NO) as 'Number of passengers'
  FROM R_BOOK
 GROUP BY RES_FLT_NO, RES_DATE
 ORDER BY RES_FLT_NO, RES_DATE;
```

Sample results:

| | RES_FLT_NO | RES_DATE | Number of passengers |
|---|---|---|---|
| 1 | CFX36 | 2022-04-05 | 3 |
| 2 | CFX36 | 2022-04-06 | 1 |
| 3 | CFX37 | 2022-04-05 | 2 |
| 4 | CFX82 | 2022-04-05 | 1 |
| 5 | CFX82 | 2022-04-06 | 1 |
| 6 | CFX83 | 2022-04-05 | 1 |

*CFX36 is popular flight on daily basis.*

(4) List the counts of special requests in order to consider it as standard service:

```
SELECT SR_CODE, SR_DESC, COUNT(SR_CODE) as 'Number of requests'
  FROM SR_BOOK
 GROUP BY SR_CODE, SR_DESC
 ORDER BY SR_CODE, SR_DESC;
```

Sample results:

| | SR_CODE | SR_DESC | Number of requests |
|---|---|---|---|
| 1 | LS11 | Over-sized luggage | 2 |
| 2 | MV08 | Vegetarian meal | 4 |
| 3 | WC03 | Use of wheel chairs | 2 |

*Vegetarian meal can be set as standard service.*

**Assumptions and limitations**

- if there is a limit of special requests per cabin class, an attribute in CABIN table can be added to enforce this constraint.

**Conclusion**

This design supports basic business needs. It can also support additional requirements such as saving passengers' special requests in the PASSENGER table, etc.

## Case 3 – Computer laboratory booking

### Business needs

- record booking by manual data entry including Lab, Date, Time slot only
- check availability and set status as 'No' when computer laboratory is unavailable
- list requests of requestors to review usage

### Database design

Five tables and one logical view are created. Three tables require initial setup as below figures. REQUEST is the staging table that keeps requests being processed.

### Table ALLOCATED

Entries of successful requests and scheduled lectures

### Table COMP_LAB

Basic information of computer laboratory

| | CL_ID | CL_CAP | CL_REMARK |
|---|---|---|---|
| 1 | KMC0608 | 40 | Nil |
| 2 | KMC0609 | 40 | Colour laser printer |
| 3 | KMC2306 | 40 | Nil |
| 4 | KMC2307 | 40 | 3D printer |
| 5 | KMC2308 | 42 | Nil |

COMP_LAB

### Table REQUEST

Request entries issued by requestors
Successful requests will be migrated to ALLOCATED

### Table REQUESTOR

Basic information of requestor

| | REQ_ID | REQ_NAME |
|---|---|---|
| 1 | 52763984 | Tommy Siu |
| 2 | 63325849 | Kevin Chau |
| 3 | 65273348 | Grace Luk |
| 4 | 90030688 | Howard Chu |
| 5 | ADMIN | College |

REQUESTOR

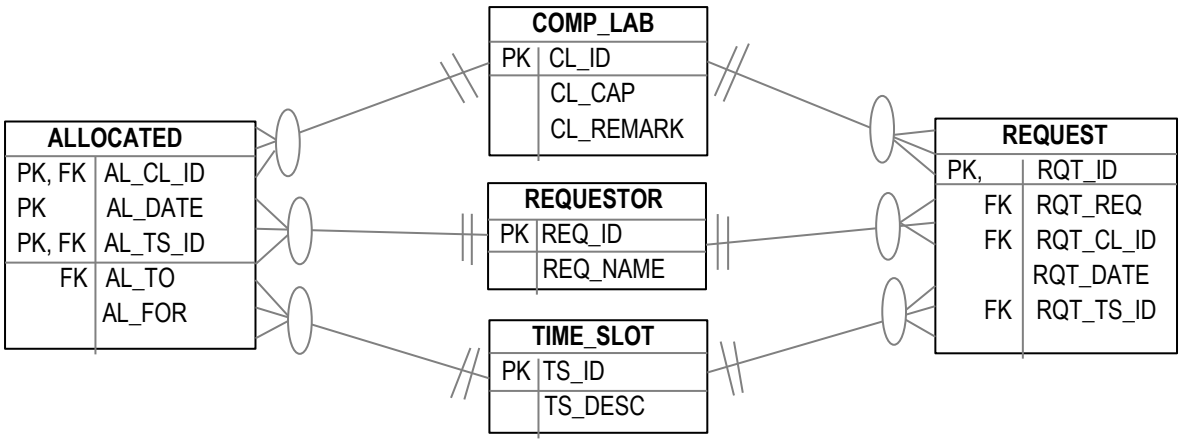
### Table TIME_SLOT

Time slot label and time interval

| | TS_ID | TS_DESC |
|---|---|---|
| 1 | A | 08:30-10:00 |
| 2 | B | 10:00-11:30 |
| 3 | C | 11:30-13:00 |
| 4 | D | 13:00-14:30 |
| 5 | E | 14:30-16:00 |
| 6 | F | 16:00-17:30 |
| 7 | G | 17:30-19:00 |

TIME_SLOT

### ER Diagram




| | | |
|---|---|---|
| Initial setup: | COMP_LAB: | Basic information of computer laboratories |
| | REQUESTOR: | The only entry "ADMIN" of the College |
| | TIME_SLOT: | Time slot labels and time intervals |

| | | |
|---|---|---|
| Data entry by end-users: | ALLOCATED: | Entries for scheduled lectures |
| | REQUESTOR: | Register requestor information |
| | REQUEST: | Computer Laboratory, Date, Time slot |

Data to be computed: Nil

**Logical View MATCHING**

This logical view is created to set M_STS after matching requests with allocated time slots.

```
CREATE VIEW MATCHING AS
  SELECT RQT_CL_ID AS M_CL_ID, RQT_DATE AS M_DATE, RQT_TS_ID AS M_TS_ID,
         RQT_REQ AS M_TO, RQT_ID AS M_FOR, 'No' AS M_STS
    FROM REQUEST, ALLOCATED
   WHERE (RQT_CL_ID || RQT_DATE || RQT_TS_ID) = (AL_CL_ID || AL_DATE || AL_TS_ID)
  UNION
  SELECT RQT_CL_ID AS M_CL_ID, RQT_DATE AS M_DATE, RQT_TS_ID AS M_TS_ID,
         RQT_REQ AS M_TO, RQT_ID AS M_FOR, 'Yes' AS M_STS
    FROM REQUEST
   WHERE (RQT_CL_ID || RQT_DATE || RQT_TS_ID) NOT IN
         (SELECT (AL_CL_ID || AL_DATE || AL_TS_ID) FROM ALLOCATED);
```

Sample results:

| | M_CL_ID | M_DATE | M_TS_ID | M_TO | M_FOR | M_STS |
|---|---|---|---|---|---|---|
| 1 | KMC0608 | 2022-03-08 | F | 52763984 | R105 | No |
| 2 | KMC0609 | 2022-03-08 | D | 63325849 | R102 | Yes |
| 3 | KMC0609 | 2022-03-08 | E | 63325849 | R103 | Yes |
| 4 | KMC2307 | 2022-03-07 | C | 90030688 | R001 | No |
| 5 | KMC2307 | 2022-03-08 | D | 52763984 | R101 | Yes |
| 6 | KMC2307 | 2022-03-08 | E | 90030688 | R104 | Yes |

**Batch processing of requests**

1. Requests are marked with 'Yes' or 'No' in M_STS of MATCHING when submitted by requestors

2. On regular basis (e.g. every 2 hours), requests with 'Yes' will be appended to ALLOCATED using

```
INSERT INTO ALLOCATED
       SELECT M_CL_ID as AL_CL_ID, M_DATE as AL_DATE, M_TS_ID as AL_TS_ID,
              M_TO as AL_TO, M_FOR as AL_FOR
         FROM MATCHING
        WHERE M_STS = 'Yes';
```

3. Then, successful requests are deleted from the REQUEST table using

```
DELETE FROM REQUEST
 WHERE RQT_ID IN
       (SELECT AL_FOR FROM ALLOCATED);
```

After the above steps, ALLOCATED has all the successful requests and scheduled lectures. REQUEST has unsuccessful requests only and ready for new entry submissions.

Sample results:

| | M_CL_ID | M_DATE | M_TS_ID | M_TO | M_FOR | M_STS |
|---|---|---|---|---|---|---|
| 1 | KMC0608 | 2022-03-08 | F | 52763984 | R105 | No |
| 2 | KMC2307 | 2022-03-07 | C | 90030688 | R001 | No |

**Test results**

(1) List requests that cannot be fulfilled:

```
SELECT M_FOR as RQT_ID, M_TO, REQ_NAME, M_CL_ID, M_DATE, TS_DESC
  FROM MATCHING, REQUESTOR, TIME_SLOT
 WHERE M_FOR = RQT_ID AND M_TS_ID = TS_ID AND M_TO = REQ_ID AND M_STS = 'No';
```

Sample results:

| | RQT_ID | M_TO | REQ_NAME | M_CL_ID | M_DATE | TS_DESC |
|---|---|---|---|---|---|---|
| 1 | R105 | 52763984 | Tommy Siu | KMC0608 | 2022-03-08 | 16:00-17:30 |
| 2 | R001 | 90030688 | Howard Chu | KMC2307 | 2022-03-07 | 11:30-13:00 |

(2) List the usage by users:

```
SELECT AL_TO, REQ_NAME, COUNT(AL_TS_ID) as 'Usage'
  FROM ALLOCATED, REQUESTOR
 WHERE AL_TO = REQ_ID
 GROUP BY AL_TO, REQ_NAME
 ORDER BY AL_TO, REQ_NAME;
```

Sample results:

| | AL_TO | REQ_NAME | Usage |
|---|---|---|---|
| 1 | 52763984 | Tommy Siu | 2 |
| 2 | 63325849 | Kevin Chau | 3 |
| 3 | 65273348 | Grace Luk | 1 |
| 4 | 90030688 | Howard Chu | 2 |
| 5 | ADMIN | College | 8 |

(3) List the usage by computer laboratory:

```
SELECT AL_CL_ID, COUNT(AL_TS_ID) as 'Usage'
  FROM ALLOCATED
 GROUP BY AL_CL_ID
 ORDER BY AL_CL_ID;
```

Sample results:

| | AL_CL_ID | Usage |
|---|---|---|
| 1 | KMC0608 | 2 |
| 2 | KMC0609 | 2 |
| 3 | KMC2306 | 2 |
| 4 | KMC2307 | 8 |
| 5 | KMC2308 | 2 |

**Assumptions and limitations**

- the time interval (e.g. every 2 hours) to run batch processing of requests is effective
- the allocated time slots are supposed be used by the users without unattended

**Conclusion**

With the presence of the REQUESTOR table, this design caters for future enhancement in the area of priority booking. Once the mechanism of determining priority is in force, the implementation can be applied by changing existing SQLs or creating new SQLs.

## Case 4 – Community virus test

### Business needs

- record virus tests and determine validity as per the age of the citizen and the test type
- generate reports on various aspects including citizens, test centres, and test types

### Database design

Four tables and two logical views are created. Three master tables are basic and TEST_RESULT is the records of virus tests. Logical views are generated to calculate age and determine validity.

**Table CITIZEN**
Basic information of citizen

| | CZ_NO | CZ_NAME | CZ_DOB |
|---|---|---|---|
| 1 | LEC40 | Echo C LUDWIG | 1940-09-11 |
| 2 | SKH65 | SHEUNG Kin Hong | 1950-10-14 |
| 3 | NSK29 | NG Sai King | 1981-04-03 |
| 4 | CFS27 | CHAN Fong Sum | 1996-02-07 |
| 5 | JHE38 | Henry E JONES | 2002-07-19 |
| 6 | NVT93 | Vincy T NEIL | 2015-03-11 |
| 7 | TSL12 | TAI Sai Lung | 2018-08-26 |
| 8 | FCT18 | FONG Chun Tai | 2020-02-02 |
| 9 | TNH07 | TAI Nin Han | 2021-08-08 |

CITIZEN

**Table TEST_CENTRE**
Basic information of test centre

| | TC_CODE | TC_NAME |
|---|---|---|
| 1 | E06 | Evergreen Clinic |
| 2 | N21 | Norstorm Hospital |
| 3 | S17 | Southern Polyclinics |

TEST_CENTRE

**Table TEST_RESULT**
Details of a virus test

| | TR_DATE | TR_CZ_NO | TR_CC | TR_TTYPE | TR_RES |
|---|---|---|---|---|---|
| 1 | 2022-02-25 | JHE38 | E06 | PCR | Negative |
| 2 | 2022-02-27 | CFS27 | E06 | PCR | Negative |
| 3 | 2022-02-28 | SKH65 | N21 | RAT | Negative |
| 4 | 2022-03-02 | TNH07 | E06 | RAT | Negative |
| 5 | 2022-03-05 | NSK29 | E06 | RAT | Positive |
| 6 | 2022-03-05 | NSK29 | N21 | PCR | Negative |
| 7 | 2022-03-06 | TSL12 | E06 | IBN | Negative |
| 8 | 2022-03-08 | FCT18 | N21 | IBN | Negative |
| 9 | 2022-03-11 | SKH65 | S17 | PCR | Negative |
| 10 | 2022-03-11 | TNH07 | S17 | IBB | Negative |

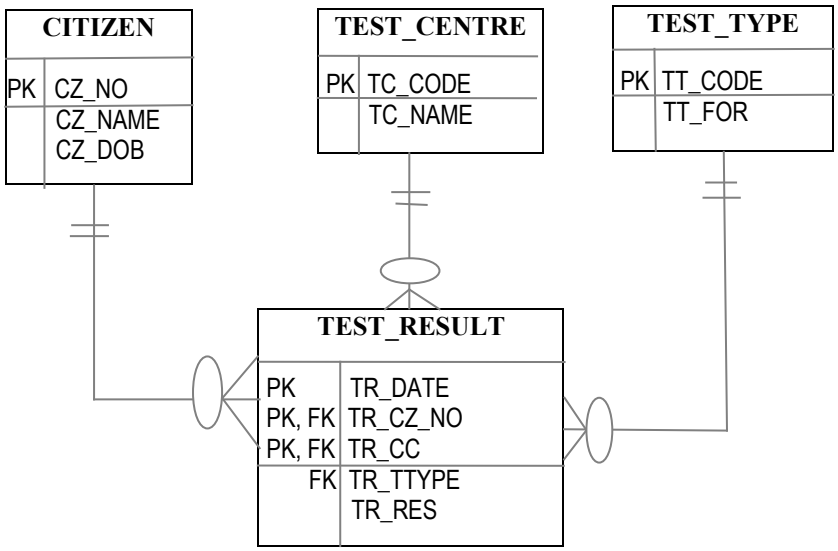
TEST_RESULT

**Table TEST_TYPE**
Basic information of test type including the suitability by age

| | TT_CODE | TT_FOR |
|---|---|---|
| 1 | IBB | 2 or below |
| 2 | IBN | 2 or below |
| 3 | PCR | 3 or above |
| 4 | RAT | 3 or above |

TEST_TYPE

### ER Diagram



**Logical View TR_EXT**

This logical view is created for joining all tables together and calculate the age of each citizen record.

```
CREATE VIEW TR_EXT AS
  SELECT TR_DATE, TR_CZ_NO, CZ_NAME, CZ_DOB, CAST((JulianDay(TR_DATE) -
         JulianDay(CZ_DOB))/365.25 AS INTEGER) as TR_AGE, TR_CC, TC_NAME, TR_TTYPE,
         TT_FOR, TR_RES
    FROM CITIZEN, TEST_CENTRE, TEST_RESULT, TEST_TYPE
   WHERE TR_CZ_NO = CZ_NO AND TR_CC = TC_CODE AND TR_TTYPE = TT_CODE;
```

Sample results:

| | TR_DATI | TR_CZ_N( | CZ_NAME | CZ_DOB | TR_AGE | TR_CC | TC_NAME | TR_TTYPI | TT_FOR | TR_RES |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2022-02-25 | JHE38 | Henry E JONES | 2002-07-19 | 19 | E06 | Evergreen Clinic | PCR | 3 or above | Negative |
| 2 | 2022-02-27 | CFS27 | CHAN Fong Sum | 1996-02-07 | 26 | E06 | Evergreen Clinic | PCR | 3 or above | Negative |
| 3 | 2022-02-28 | SKH65 | SHEUNG Kin Hong | 1950-10-14 | 71 | N21 | Norstorm Hospital | RAT | 3 or above | Negative |
| 4 | 2022-03-02 | TNH07 | TAI Nin Han | 2021-08-08 | 0 | E06 | Evergreen Clinic | RAT | 3 or above | Negative |
| 5 | 2022-03-05 | NSK29 | NG Sai King | 1981-04-03 | 40 | E06 | Evergreen Clinic | RAT | 3 or above | Positive |
| 6 | 2022-03-05 | NSK29 | NG Sai King | 1981-04-03 | 40 | N21 | Norstorm Hospital | PCR | 3 or above | Negative |
| 7 | 2022-03-06 | TSL12 | TAI Sai Lung | 2018-08-26 | 3 | E06 | Evergreen Clinic | IBN | 2 or below | Negative |
| 8 | 2022-03-08 | FCT18 | FONG Chun Tai | 2020-02-02 | 2 | N21 | Norstorm Hospital | IBN | 2 or below | Negative |
| 9 | 2022-03-11 | SKH65 | SHEUNG Kin Hong | 1950-10-14 | 71 | S17 | Southern Polyclinics | PCR | 3 or above | Negative |
| 10 | 2022-03-11 | TNH07 | TAI Nin Han | 2021-08-08 | 0 | S17 | Southern Polyclinics | IBB | 2 or below | Negative |

**Logical View TR_VALIDITY**

This logical view is created for determining the validity of a virus test.

```
CREATE VIEW TR_VALIDITY AS
  SELECT *, 'Yes' AS TR_VALID
    FROM TR_EXT
   WHERE TR_AGE >= 3 AND TT_FOR = '3 or above'
  UNION
  SELECT *, 'Yes' AS TR_VALID
    FROM TR_EXT
   WHERE TR_AGE <= 2 AND TT_FOR = '2 or below'
  UNION
  SELECT *, 'No' AS TR_VALID
    FROM TR_EXT
   WHERE TR_AGE >= 3 AND TT_FOR = '2 or below'
  UNION
  SELECT *, 'No' AS TR_VALID
    FROM TR_EXT
   WHERE TR_AGE <= 2 AND TT_FOR = '3 or above';
```

Sample results:

| | TR_DATI | TR_CZ_N( | CZ_NAME | CZ_DOB | TR_AGE | TR_CC | TC_NAME | TR_TTYPI | TT_FOR | TR_RES | TR_VALII |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2022-02-25 | JHE38 | Henry E JONES | 2002-07-19 | 19 | E06 | Evergreen Clinic | PCR | 3 or above | Negative | Yes |
| 2 | 2022-02-27 | CFS27 | CHAN Fong Sum | 1996-02-07 | 26 | E06 | Evergreen Clinic | PCR | 3 or above | Negative | Yes |
| 3 | 2022-02-28 | SKH65 | SHEUNG Kin Hong | 1950-10-14 | 71 | N21 | Norstorm Hospital | RAT | 3 or above | Negative | Yes |
| 4 | 2022-03-02 | TNH07 | TAI Nin Han | 2021-08-08 | 0 | E06 | Evergreen Clinic | RAT | 3 or above | Negative | No |
| 5 | 2022-03-05 | NSK29 | NG Sai King | 1981-04-03 | 40 | E06 | Evergreen Clinic | RAT | 3 or above | Positive | Yes |
| 6 | 2022-03-05 | NSK29 | NG Sai King | 1981-04-03 | 40 | N21 | Norstorm Hospital | PCR | 3 or above | Negative | Yes |
| 7 | 2022-03-06 | TSL12 | TAI Sai Lung | 2018-08-26 | 3 | E06 | Evergreen Clinic | IBN | 2 or below | Negative | No |
| 8 | 2022-03-08 | FCT18 | FONG Chun Tai | 2020-02-02 | 2 | N21 | Norstorm Hospital | IBN | 2 or below | Negative | Yes |
| 9 | 2022-03-11 | SKH65 | SHEUNG Kin Hong | 1950-10-14 | 71 | S17 | Southern Polyclinics | PCR | 3 or above | Negative | Yes |
| 10 | 2022-03-11 | TNH07 | TAI Nin Han | 2021-08-08 | 0 | S17 | Southern Polyclinics | IBB | 2 or below | Negative | Yes |

**Test results**

(1) List citizens who have not yet taken any valid tests:

```
SELECT CZ_NO, CZ_NAME, CZ_DOB, CAST((JulianDay(date()) -
       JulianDay(CZ_DOB))/365.25 as INTEGER) as 'Age'
  FROM CITIZEN
 WHERE CZ_NO NOT IN
       (SELECT DISTINCT TR_CZ_NO FROM TR_VALIDITY WHERE TR_VALID='Yes');
```

Sample results:

| | CZ_NO | CZ_NAME | CZ_DOB | Age |
|---|---|---|---|---|
| 1 | LEC40 | Echo C LUDWIG | 1940-09-11 | 81 |
| 2 | NVT93 | Vincy T NEIL | 2015-03-11 | 7 |
| 3 | TSL12 | TAI Sai Lung | 2018-08-26 | 3 |

(2) Count citizens who have taken valid tests and being negative:

```
SELECT TR_CZ_NO, CZ_NAME, CZ_DOB, TR_AGE, COUNT(TR_CZ_NO)
  FROM TR_VALIDITY
 WHERE TR_VALID='Yes' AND TR_RES='Negative'
 GROUP BY TR_CZ_NO, CZ_NAME, CZ_DOB, TR_AGE
 ORDER BY TR_CZ_NO, CZ_NAME, CZ_DOB, TR_AGE;
```

Sample results:

| | TR_CZ_NO | CZ_NAME | CZ_DOB | TR_AGE | COUNT(TR_CZ_NO) |
|---|---|---|---|---|---|
| 1 | CFS27 | CHAN Fong Sum | 1996-02-07 | 26 | 1 |
| 2 | FCT18 | FONG Chun Tai | 2020-02-02 | 2 | 1 |
| 3 | JHE38 | Henry E JONES | 2002-07-19 | 19 | 1 |
| 4 | NSK29 | NG Sai King | 1981-04-03 | 40 | 1 |
| 5 | SKH65 | SHEUNG Kin Hong | 1950-10-14 | 71 | 2 |
| 6 | TNH07 | TAI Nin Han | 2021-08-08 | 0 | 1 |

(3) Count test centres where invalid tests have been taken:

```
SELECT TR_CC, TC_NAME, COUNT(TR_VALID)
  FROM TR_VALIDITY
 WHERE TR_VALID='No'
 GROUP BY TR_CC, TC_NAME
 ORDER BY TR_CC, TC_NAME;
```

Sample results:

| | TR_CC | TC_NAME | COUNT(TR_VALID) |
|---|---|---|---|
| 1 | E06 | Evergreen Clinic | 2 |

**Assumptions and limitations**

- the virus test result is either positive or negative

**Conclusion**

This design supports stated requirements. If required, SQLs can be prepared to show virus test records within a stated date range.

## Case 5 – Round robin league

### Business needs

- record match results by manual data entry including Round, Teams, Scores only
- show standing championship points per team after each round
- list match results per team
- be able to handle championship leagues that composed of N teams

### Database design

Three tables and one logical view

Table name: TEAM  WITHOUT ROWID

| | Name | Data type | Primary Key | Foreign Key | Unique | Check | Not NULL |
|---|---|---|---|---|---|---|---|
| 1 | TEAM_ID | CHAR | | | | | |
| 2 | TEAM_NAME | CHAR | | | | | |
| 3 | TEAM_MANAGER | CHAR | | | | | |

Table name: ROUND  WITHOUT ROWID

| | Name | Data type | Primary Key | Foreign Key | Unique | Check | Not NULL |
|---|---|---|---|---|---|---|---|
| 1 | RND_ID | INT | | | | | |
| 2 | RND_DATE | DATE | | | | | |

Table name: MATCH  WITHOUT ROWID

| | Name | Data type | Primary Key | Foreign Key | Unique | Check | Not NULL | Collate |
|---|---|---|---|---|---|---|---|---|
| 1 | MATCH_ID | CHAR | | | | | | |
| 2 | MATCH_RND | INT | | | | | | |
| 3 | MATCH_T_W | CHAR | | | | | | |
| 4 | MATCH_T_L | CHAR | | | | | | |
| 5 | MATCH_T1 | CHAR | | | | | | |
| 6 | MATCH_T1_RES | INT | | | | | | |
| 7 | MATCH_T2 | CHAR | | | | | | |
| 8 | MATCH_T2_RES | INT | | | | | | |

**Table TEAM**

Basic information of a team

**Table ROUND**

Round number and date only

**Table MATCH**

MATCH_ID is key, MATCH_RND is round number

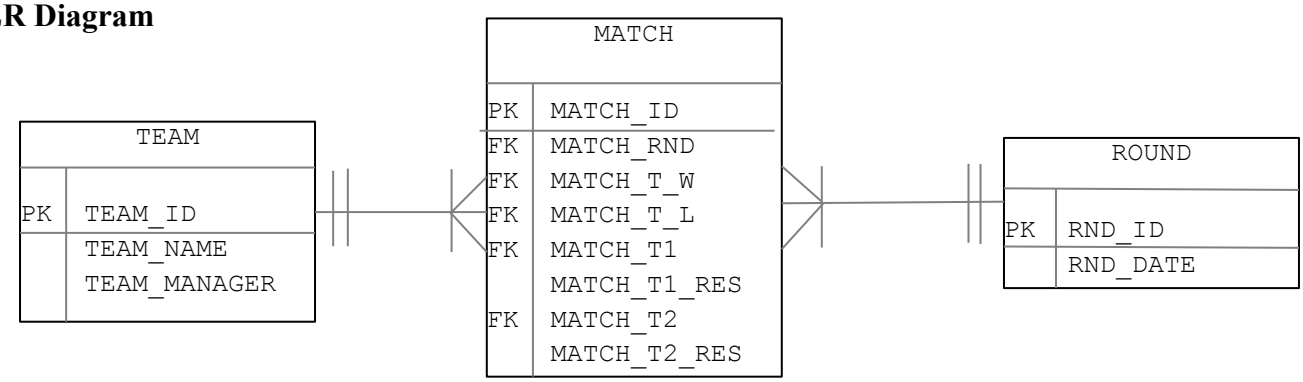
MATCH_T_W is winner, MATCH_T_L is loser

*These two attributes are derived by running a SQL at any time to ensure accuracy.*

*They are justified because the frequency of use is high and the frequency of change is very low.*

MATCH_T1 and MATCH_T2 are matching teams, the order is not significant

### ER Diagram




| | | |
|---|---|---|
| Initial setup: | TEAM: | TEAM_ID, TEAM_NAME, TEAM_MANAGER |
| | ROUND: | RND_ID, RND_DATE |
| Data entry by end-users: | MATCH: | MATCH_ID, MATCH_RND, MATCH_T1, MATCH_T2, MATCH_T1_RES, MATCH_T2_RES |

Data to be computed: MATCH: MATCH_T_W, MATCH_T_L

```
update MATCH
   set MATCH_T_W = MATCH_T1,
       MATCH_T_L = MATCH_T2
 where MATCH_T1_RES > MATCH_T2_RES;
update MATCH
   set MATCH_T_W = MATCH_T2,
       MATCH_T_L = MATCH_T1
 where MATCH_T1_RES < MATCH_T2_RES;
```

*This SQL can be run at any time and can be re-run.*

**Logical View POINT_PER_MATCH**

This logical view is created for keeping rounds, teams, points that shows standing championship points.

```
CREATE VIEW POINT_PER_MATCH AS
            SELECT MATCH_RND, MATCH_T_W AS TEAM, '1' AS POINT FROM MATCH
              UNION
            SELECT MATCH_RND, MATCH_T_L AS TEAM, '0' AS POINT FROM MATCH;
```

**Test results**

(1) List standing championship points by team:

```
SELECT TEAM, SUM(POINT)
  FROM POINT_PER_MATCH
 GROUP BY TEAM
 ORDER BY TEAM;
```

| | TEAM | SUM(POINT) |
|---|---|---|
| 1 | A | 5 |
| 2 | B | 3 |
| 3 | C | 3 |
| 4 | D | 3 |
| 5 | E | 4 |
| 6 | F | 1 |
| 7 | G | 5 |
| 8 | H | 4 |

(2) List match results of selected team:

```
SELECT MATCH_RND as ROUND,
       MATCH_T1 as TEAM_1,
       MATCH_T1_RES as T1_SCORE,
       MATCH_T2 as TEAM_2,
       MATCH_T2_RES as T2_SCORE,
       MATCH_T_W as WINNER
  FROM MATCH
 WHERE MATCH_T1 = ?
    OR MATCH_T2 = ?;
```

| | ROUND | TEAM 1 | T1 SCORE | TEAM 2 | T2 SCORE | WINNER |
|---|---|---|---|---|---|---|
| 1 | 1 | A | 3 | B | 2 | A |
| 2 | 2 | A | 3 | E | 1 | A |
| 3 | 3 | A | 2 | C | 3 | C |
| 4 | 4 | A | 3 | D | 0 | A |
| 5 | 5 | A | 3 | F | 1 | A |
| 6 | 6 | A | 2 | G | 3 | G |
| 7 | 7 | A | 3 | H | 2 | A |

Sample output for Team A

| | ROUND | TEAM 1 | T1 SCORE | TEAM 2 | T2 SCORE | WINNER |
|---|---|---|---|---|---|---|
| 1 | 1 | G | 3 | H | 0 | G |
| 2 | 2 | C | 2 | G | 3 | G |
| 3 | 3 | E | 3 | G | 1 | E |
| 4 | 4 | F | 2 | G | 3 | G |
| 5 | 5 | D | 3 | G | 2 | D |
| 6 | 6 | A | 2 | G | 3 | G |
| 7 | 7 | D | 1 | G | 3 | G |

Sample output for Team G

**Assumptions and limitations**

- the schedule of matches per round has been planned in advance
- the champion is determined manually by looking for the team scored highest points from (1); if two or more teams scored the same points, the match results listed in (2) can show their inter-team match scores; this example shows Team G won Team A at Round 6, so Team G is champion

**Conclusion**

This design supports championship leagues of any number of teams. However, this design cannot generate the schedule of matches per round.

## Case 6 – IT help desk

### Business needs

- record service requests including requestors, problems, solutions, and handling helpers
- generate reports on usage, performance, problems, and solutions
- be able to support future enhancements

### Database design

Four tables and one logical view are created. Three master tables keep basic information. SERV_REQ table keeps details of service requests.

### Table PROBLEM

Basic information of problem and problem type

| | PROB_CODE | PROB_TYPE | PROB_DESC |
|---|---|---|---|
| 1 | D01 | Desktop | desktop hang-up |
| 2 | D02 | Desktop | no display |
| 3 | N01 | Notebook | battery failure |
| 4 | N81 | Network | cannot connect network printer |
| 5 | N91 | Network | cannot access Internet |
| 6 | P01 | Printer | paper jam |
| 7 | P02 | Printer | toner low |
| 8 | P03 | Printer | no printout |
| 9 | S01 | Security | cannot sign-on |

PROBLEM

### Table SOLUTION

Basic information of solution

| | SOL_CODE | SOL_DESC |
|---|---|---|
| 1 | B225 | Replace notebook battery |
| 2 | N174 | Re-install network cable |
| 3 | N554 | Re-create network printer icon |
| 4 | P001 | Power reset |
| 5 | P005 | Remove jam paper |
| 6 | R125 | Reset password |
| 7 | R326 | Replace printer toner |
| 8 | Z999 | No action, problem disappeared |

SOLUTION

### Table STAFF

Basic information of staff

| | STAFF_EXT | STAFF_NAME | STAFF_DEPT |
|---|---|---|---|
| 1 | 2984 | Ivan Lam | IT |
| 2 | 2985 | Bill Shek | IT |
| 3 | 2986 | Rose Wong | IT |
| 4 | 5026 | Wu Xiaoyu | Sales |
| 5 | 5318 | Nelson Ng | Sales |
| 6 | 6293 | Doris Luk | R & D |
| 7 | 7602 | Steve Tam | Finance |

STAFF

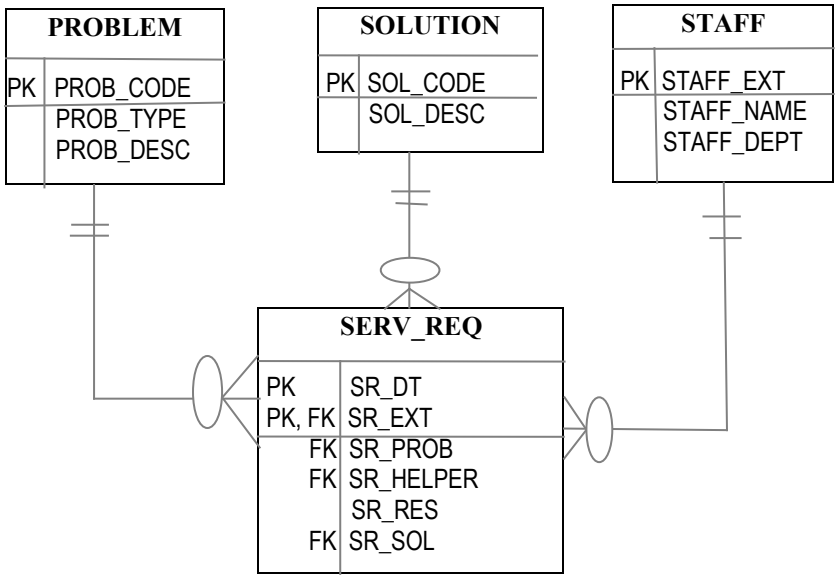
### Table SERV_REQ

Details of each service request

| | SR_DT | SR_EXT | SR_PROB | SR_HELPEI | SR_RES | SR_SOL |
|---|---|---|---|---|---|---|
| 1 | 2022-03-14 12:05 | 5318 | S01 | 2986 | 2022-03-14 14:25 | R125 |
| 2 | 2022-03-15 09:35 | 7602 | P01 | 2985 | 2022-03-15 09:40 | P005 |
| 3 | 2022-03-15 10:40 | 7602 | P01 | 2984 | 2022-03-15 11:15 | R326 |
| 4 | 2022-03-16 16:10 | 6293 | D02 | 2985 | 2022-03-16 16:30 | P001 |
| 5 | 2022-03-18 17:25 | 5026 | S01 | 2984 | 2022-03-21 10:05 | R125 |
| 6 | 2022-03-21 11:50 | 5318 | P02 | 2986 | 2022-03-21 12:25 | R326 |
| 7 | 2022-03-21 15:15 | 6293 | D01 | 2985 | NULL | NULL |

SERV_REQ

### ER Diagram



**Logical View SERVREQ_EXT**

This logical view is created for joining related tables and calculate the duration of each service request.

```
CREATE VIEW SERVREQ_EXT AS
  SELECT SR_DT, SR_EXT, STAFF_NAME, STAFF_DEPT, SR_PROB, PROB_TYPE, PROB_DESC,
     SR_HELPER, SR_RES, CAST ((JulianDay(SR_RES) - JulianDay(SR_DT)) * 24 AS INTEGER)
     AS SR_DUR, SR_SOL, SOL_DESC
    FROM PROBLEM, SERV_REQ, SOLUTION, STAFF
   WHERE SR_PROB = PROB_CODE AND SR_SOL = SOL_CODE AND SR_EXT = STAFF_EXT;
```

Sample results:

| | SR_DT | SR_EXT | STAFF_NAM | STAFF_DE | SR_PROB | PROB_TY | PROB_DESC | SR_HELPE | SR_RES | SR_DUR | SR_SOL | SOL_DESC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2022-03-14 12:05 | 5318 | Nelson Ng | Sales | S01 | Security | cannot sign-on | 2986 | 2022-03-14 14:25 | 2 | R125 | Reset password |
| 2 | 2022-03-15 09:35 | 7602 | Steve Tam | Finance | P01 | Printer | paper jam | 2985 | 2022-03-15 09:40 | 0 | P005 | Remove jam paper |
| 3 | 2022-03-15 10:40 | 7602 | Steve Tam | Finance | P01 | Printer | paper jam | 2984 | 2022-03-15 11:15 | 0 | R326 | Replace printer toner |
| 4 | 2022-03-16 16:10 | 6293 | Doris Luk | R & D | D02 | Desktop | no display | 2985 | 2022-03-16 16:30 | 0 | P001 | Power reset |
| 5 | 2022-03-18 17:25 | 5026 | Wu Xiaoyu | Sales | S01 | Security | cannot sign-on | 2984 | 2022-03-21 10:05 | 64 | R125 | Reset password |
| 6 | 2022-03-21 11:50 | 5318 | Nelson Ng | Sales | P02 | Printer | toner low | 2986 | 2022-03-21 12:25 | 0 | R326 | Replace printer toner |

Note: Service requests that have not been resolved are not shown here.

**Test results**

(1) List service requests by requestor and problem type:

```
SELECT SR_EXT, STAFF_NAME, STAFF_DEPT, PROB_TYPE, PROB_DESC, SR_DUR, SOL_DESC
  FROM SERVREQ_EXT
 ORDER BY SR_EXT, PROB_TYPE;
```

Sample results:

| | SR_EXT | STAFF_NAM | STAFF_DE | PROB_TY | PROB_DESC | SR_DUR | SOL_DESC |
|---|---|---|---|---|---|---|---|
| 1 | 5026 | Wu Xiaoyu | Sales | Security | cannot sign-on | 64 | Reset password |
| 2 | 5318 | Nelson Ng | Sales | Printer | toner low | 0 | Replace printer toner |
| 3 | 5318 | Nelson Ng | Sales | Security | cannot sign-on | 2 | Reset password |
| 4 | 6293 | Doris Luk | R & D | Desktop | no display | 0 | Power reset |
| 5 | 7602 | Steve Tam | Finance | Printer | paper jam | 0 | Replace printer toner |
| 6 | 7602 | Steve Tam | Finance | Printer | paper jam | 0 | Remove jam paper |

*The frequent requestors are Nelson Ng and Steve Tam.*
*The most frequent problem type is printer.*

(2) Count service requests handled by helper and show average duration of handling a request:

```
SELECT SR_HELPER, STAFF.STAFF_NAME, COUNT(SR_DT), AVG(SR_DUR)
  FROM SERVREQ_EXT, STAFF
 WHERE SR_HELPER = STAFF.STAFF_EXT
 GROUP BY SR_HELPER, STAFF.STAFF_NAME
 ORDER BY SR_HELPER, STAFF.STAFF_NAME;
```

Sample results:

| | SR_HELPER | STAFF_NAME | COUNT(SR_DT) | AVG(SR_DUR) |
|---|---|---|---|---|
| 1 | 2984 | Ivan Lam | 2 | 32 |
| 2 | 2985 | Bill Shek | 2 | 0 |
| 3 | 2986 | Rose Wong | 2 | 1 |

*Ivan Lam should explain why the average duration being 32 hours.*

(3) List solutions by problem types:

```
SELECT DISTINCT PROB_TYPE, PROB_DESC, SOL_DESC
  FROM SERVREQ_EXT
 ORDER BY PROB_TYPE, PROB_DESC;
```

Sample results:

| | PROB_TYPE | PROB_DESC | SOL_DESC |
|---|---|---|---|
| 1 | Desktop | no display | Power reset |
| 2 | Printer | paper jam | Replace printer toner |
| 3 | Printer | paper jam | Remove jam paper |
| 4 | Printer | toner low | Replace printer toner |
| 5 | Security | cannot sign-on | Reset password |

*Printer problems are frequent and complicated.*

(4) List problems by solutions:

```
SELECT DISTINCT SOL_DESC, PROB_TYPE, PROB_DESC
  FROM SERVREQ_EXT
 ORDER BY SOL_DESC;
```

Sample results:

| | SOL_DESC | PROB_TYPE | PROB_DESC |
|---|---|---|---|
| 1 | Power reset | Desktop | no display |
| 2 | Remove jam paper | Printer | paper jam |
| 3 | Replace printer toner | Printer | paper jam |
| 4 | Replace printer toner | Printer | toner low |
| 5 | Reset password | Security | cannot sign-on |

*Reset and replace components are frequently used.*

**Assumptions and limitations**

- the service requests related to the use of software applications are not handled by IT help desk
- the service requests are simple and resolved at the first visit/attempt by the IT helper

**Conclusion**

This design supports future enhancements in pairing problems and solutions. One may list all solutions by a specific problem type to narrow down the actions to be taken.