Data Requirements

- Calls must have a caller's id, a help-desk operator, date of the call, the time length, the serial number of the computer, and if relevant, the software and operating system
- The phone number should be recorded at all times so that if the help-desk does not pick up, the contact can be retrieved. A caller might call from different numbers
- The personnel id number is retrieved during the call by matching the caller name with an ***external database*** of personal details. Therefore other personnels' details do not need to be recorded.
- The equipment is checked against a register of equipments, and the type and make is stored so that equipment malfunction reports and solutions can be retrieved later.
- If relevant, the software details will be logged and the software license will be verified
- For every logged call, a problem number is generated. This will be supplied to the caller so it can be quoted on subsequent calls about the same problem
- When a problem is reported, a problem type will be selected from a list of problem types by the operator. A problem can have its problem type updated once more information is available
- Once the problem area is identified, the operator might try to solve it based on previous problems of the same type or similar equipments. If the problem can't be solved by the operator, the problem will be referred to a specialist
- A specialist will be an expert in one or more problem types. If no specialist is found for a specific problem type, then a specialist from the more general problem type will be used. The least loaded specialist should be picked.
- Once a problem is resolved, the operator/specialist will log the date and time it is resolved and record how the problem is resolved and the time taken

Important considerations

1. The personnel's employee data ***will not be stored in the database***. I.e, department, role, etc.
   This data can be retrieved by the service mentioned in the briefing by "a register of all personnel" and therefore maintaining and keeping that data is not the responsibility of our service.
   The only personal data that is going to be stored is the caller's name, so that if the register is down, or in maintenance we can still call back the employees without having to ask their names again. Also, the name details should not change often.
   Unless the employee is getting married or divorced their names would remain static from the system perspective.
2. Phone numbers are stored in the database. An employee will not necessarily call from their business phone, especially if the phone is the device they are reporting a problem on. They might also call different times with different phones.
3. The equipment details ***will be stored in the database***. I.e, the device type, software, operating system, license, etc.
   Consider that a windows laptop presented a problem and this problem was eventually resolved. Now imagine that this laptop operating system was changed to linux, and this time another problem was reported and also eventually resolved. If we are looking at a report from a past date, we need to keep information of the two operating systems used in the machine at one point in time, otherwise we might think that all the problems logged for that machine are from the most updated operating system record (linux).
4. The equipment type and make ***will be stored in the database.*** This data can be retrieved by what was mentioned in the briefing as "a register of equipment", but since the equipment type and make cannot ever change, this information will be stored in our database once it is retrieved for the first time. Additionally, consider that we want to generate a report that shows the maker of equipment with the highest number of problems. If we need to call or check the register for every one of them, this will be a scalability issue. Imagine that this company has a million devices, how many calls to this external service (register) would be necessary so that the report is generated? How long would it take? Moreover, if the register is down, or in maintenance, all the reports involving make/type would be unavailable.
5. A call can be about multiple problems, and a problem will always be about an equipment that will have a certain operating system and may have a certain software. If the call isn't about an equipment, the IT desk will log the problem with type "non equipment problem" for reporting purposes, and redirect the user to the most appropriate department.
6. An equipment can have multiple operating systems. Consider that a laptop can have a dual-boot for windows and linux, which might not be that rare in a big company.
7. A software will always have a software version in the system. If in real life the software is rolling-release (i.e, does not have a fixed version), a pretend "software-version" will be created and called "rolling-release", or "unique-release", etc.
8. A problem can be worked by more than one staff member at the same time. This is to denote they are collaborating on the problem solution.
9. The business logic for software licenses can be extremely complicated especially when we consider licenses through subscription plans. For the purposes of this work, I am going to set a table "Software License Status" that will return a boolean on the column "is_valid". This table could be cached or updated by an external service via cron task, etc.

Entities

Caller
The person (personnel) calling for help

Caller Phone
Phone number for the caller

Staff
A help-desk employee

Staff Role
The role of a help-desk employee
Eg: *operator, specialist*

Call
A call between a caller and a staff member about a problem

Call Status
In progress, missed, finished, ...

Problem
Details about the reported problem

Problem Type
Stores the different types of problems

Problem Specialist
A specialist in a certain problem type

Problem Status
Assined, pending, triaged, rejected, Reassigned etc

Problem Assignment
Assigns someone to solve the problem

Problem Solution
The final solution for the problem

Problem Call
The call in which the problem was reported

Equipment
The equipment being reported on

Equipment Type
Eg: Laptop, Desktop, Phone, Printer...

Equipment Make
Eg: HP, Apple, Microsoft...

Equipment Model
imac 2012, windows surface 2017 etc

Equipment Operating System
The equipment operational system

Operating System
IOS, android, windows, linux...

Operating System Version
Windows XP, iOS 10, android 11

Equipment Software
An equipment software

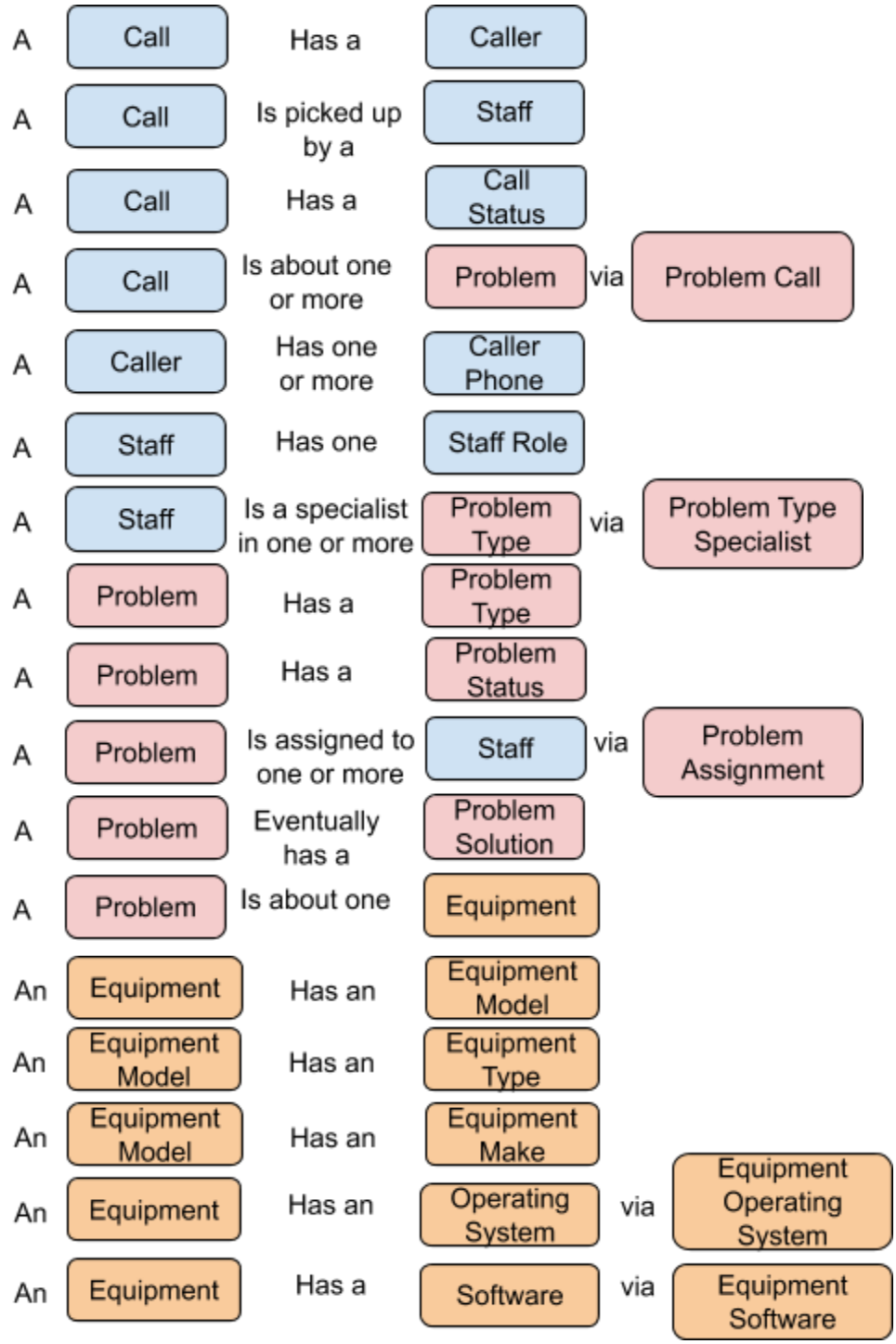
Software
A software

Software Version
Acrobat 10, photoshop 9, office 10, etc.

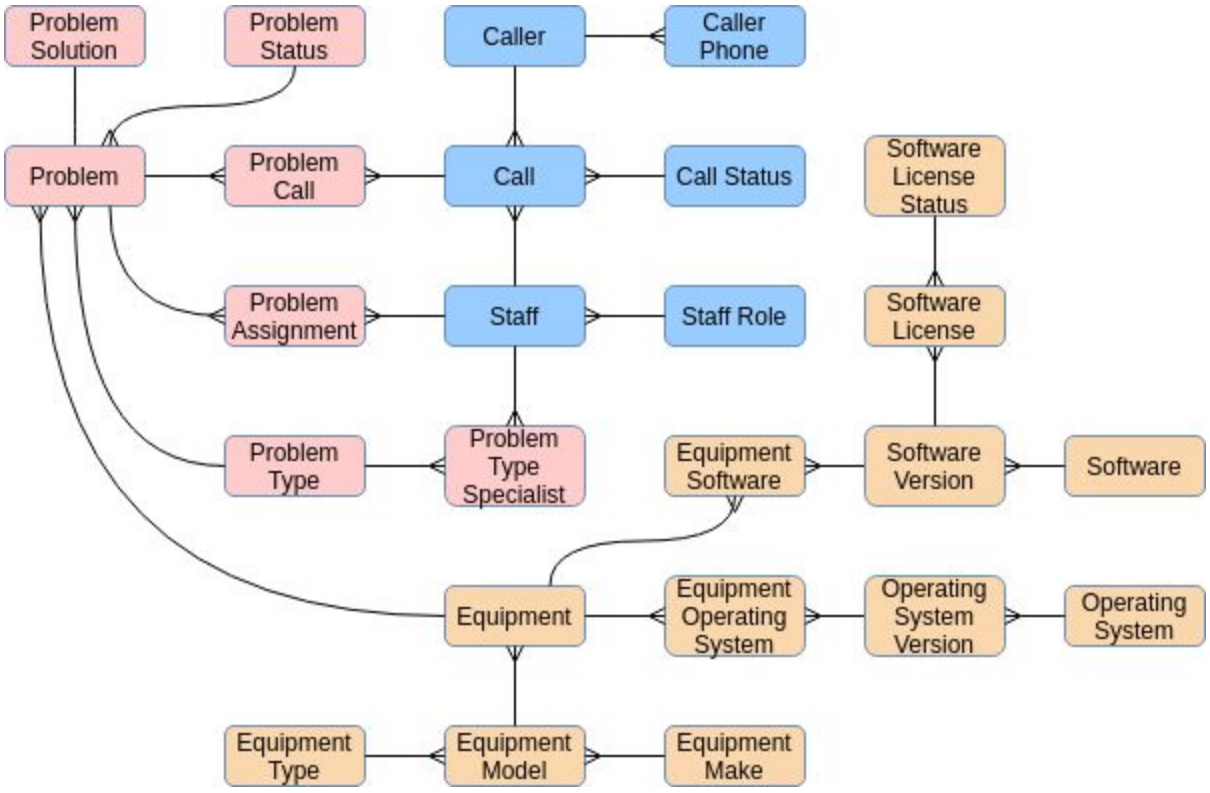
Software License
A valid or invalid software license

Software License Status
Being used by, expiration date etc

Possible Relationships



Entity Relationship Diagram



Relational Schema Tabular Format (and constraints)

Note: Entities that have names with more than one word such as “Caller Phone” were previously described with a space between words for readability. In the database this space will be removed. This means that in this tabular format you will see “CallerPhone” instead of “Caller Phone”.

Caller

Column Name	***Data Type***	***Constraints***
id	CHAR(10)*	PrimaryKey
first_name	VARCHAR2(30)	
last_name	VARCHAR2(30)	NOT NULL

- The “id” column is the value returned by the service that matches names against caller’s id. The assumption is that this value will be a fixed length CHAR field, but it could be changed to whatever pattern the matching system returns.

CallerPhone

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
phone_number	CHAR(12)	NOT NULL, UNIQUE
caller_id	CHAR(10)	Foreign Key References Caller(id) NOT NULL

CallStatus

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(30)	NOT NULL, UNIQUE

StaffRole

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(30)	NOT NULL, UNIQUE

Staff

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
first_name	VARCHAR2(30)	
last_name	VARCHAR2(30)	NOT NULL
staff_role_id	NUMBER(8,0)	Foreign Key References StaffRole(id) NOT NULL

Call

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
caller_id	CHAR(10)	Foreign Key References Caller(id) NOT NULL
call_status_id	NUMBER(8,0)	Foreign Key References CallStatus(id) NOT NULL
staff_id	NUMBER(8,0)	Foreign Key References Staff(id)
start	TIMESTAMP ‘YYYY-MM-DD HH24:MI:SS’	NOT NULL
end	TIMESTAMP ‘YYYY-MM-DD HH24:MI:SS’	

ProblemCall

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
call_id	NUMBER(8,0)	Foreign Key References Call(id) NOT NULL
problem_id	NUMBER(8,0)	Foreign Key References Problem(id) NOT NULL

Problem

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
problem_status_id	NUMBER(8,0)	Foreign Key References ProblemStatus(id) NOT NULL
problem_type_id	NUMBER(8,0)	Foreign Key References ProblemType(id) NOT NULL
equipment_id	NUMBER(8,0)	Foreign Key References Equipment(id)
description	VARCHAR2(2000)*	NOT NULL
datetime_created	TIMESTAMP 'YYYY-MM-DD HH24:MI:SS'	NOT NULL

* I considered having description as an unlimited text field, but this can be prone to bugs such as someone copy-pasting or hitting a key-stroke multiple types by mistake and then ending up with a strange record. Apparently DBA's recommendation is to not use it as they are hard to manage. 2000 characters seem to be a good compromise to summarise a problem as described by the caller.

ProblemSolution

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
problem_id	NUMBER(8,0)	Foreign Key References Problem(id) NOT NULL
description	VARCHAR2(2000)	NOT NULL
datetime_created	TIMESTAMP 'YYYY-MM-DD HH24:MI:SS'	NOT NULL

ProblemStatus

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(30)	NOT NULL, UNIQUE

ProblemAssignment

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
problem_id	NUMBER(8,0)	Foreign Key References Problem(id) NOT NULL
staff_id	NUMBER(8,0)	Foreign Key References Staff(id) NOT NULL
datetime_created	TIMESTAMP 'YYYY-MM-DD HH24:MI:SS'	NOT NULL

ProblemType

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(30)	NOT NULL, UNIQUE

ProblemTypeSpecialist

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
problem_type_id	NUMBER(8,0)	Foreign Key References ProblemType(id) NOT NULL
staff_id	NUMBER(8,0)	Foreign Key References Staff(id) NOT NULL

Equipment

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
Identifier	VARCHAR2(200)	NOT NULL
equipment_model_id	NUMBER(8,0)	Foreign Key References EquipmentModel(id) NOT NULL

- identifier is the serial number of the equipment, or some other identifier.

EquipmentModel

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(100)	NOT NULL
release_date	DATE	NOT NULL
equipment_type_id	NUMBER(8,0)	Foreign Key References EquipmentType(id) NOT NULL
equipment_make_id	NUMBER(8,0)	Foreign Key References EquipmentMake(id) NOT NULL

- Additional constraint: UNIQUE TOGETHER (equipment_type_id, equipment_make_id, name)

EquipmentMake

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(30)	NOT NULL, UNIQUE

EquipmentType

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(30)	NOT NULL, UNIQUE

EquipmentOperatingSystem

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
equipment_id	NUMBER(8,0)	Foreign Key References Equipment(id) NOT NULL
operating_system_version_id	NUMBER(8,0)	Foreign Key References OperatingSystemVersion(id) NOT NULL

OperatingSystemVersion

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(100)	NOT NULL
release_date	DATE	NOT NULL
operating_system_id	NUMBER(8,0)	Foreign Key References OperatingSystem(id) NOT NULL

- Additional constraint: UNIQUE TOGETHER (operating_system_id, name)

OperatingSystem

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(100)	NOT NULL, UNIQUE

EquipmentSoftware

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
software_version_id	NUMBER(8,0)	Foreign Key References SoftwareVersion(id) NOT NULL
equipment_id	NUMBER(8,0)	Foreign Key References Equipment(id) NOT NULL

SoftwareVersion

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(100)	NOT NULL
release_date	DATE	NOT NULL
software_id	NUMBER(8,0)	Foreign Key References Software(id) NOT NULL

- Additional constraint: UNIQUE TOGETHER (software_id, name)

Software

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
name	VARCHAR2(100)	NOT NULL, UNIQUE

SoftwareLicense

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
identifier	VARCHAR2(1000)	NOT NULL
software_version_id	NUMBER(8,0)	Foreign Key References SoftwareVersion(id) NOT NULL

SoftwareLicenseStatus

Column Name	***Data Type***	***Constraints***
id	NUMBER(8,0)	PrimaryKey
is_valid	CHAR(1)	NOT NULL
software_license_id	NUMBER(8,0)	Foreign Key References SoftwareLicense(id) NOT NULL

- Note: Oracle does not support boolean fields, and also does not provide recommendations for how to substitute it for a supported data type. Therefore `is_valid` will be either “Y” (yes) or “N” (no) and a check constraint will be created to make sure those are the only two possible values.