THE iSCHOOL
Syracuse University

IST 659: Data Admin Concepts and Database Management

Final Project Report

KEY TRACKER

Team name: APT Key Trackers 659

Rutu Waghela
Nehal Taya
Eunbi Kim
Elizabeth Myers

Professor:
Aaron Miller

INTRODUCTION

Problem Statement

The Rescue Mission provides emergency housing, meals, and services to the unhoused in Syracuse, NY. This company has its current key tracking software on a PC from the mid-90s, called Alpha Four database. As this tracking system is an old version, we found several problems below.

First, there are 700 distinct keys they are managing. The Rescue Mission has 11 housings, an administration building, 27 stores, and donation centers with 700 distinct keys. In other words, the management team including 400 employees should manage these assets and keys. However, as the team is a small business, it is difficult for them to track these keys efficiently.

Also, the facility management team should change the records of each key as even the conditions or other options are changed. The team has received many reports of loss of transaction records from their staff. It can be an enormous loss for the management team because it may cause to increase their unnecessary expenses to buy or maintain the key.

Proposed Solution

We expect the management team provides a high-quality tracking service to their staff with this new database system.

First, the system will allow the team to record, store, and track thousands of key transactions in the database. They can assign categories such as creation, assignment, return, and loss or destruction to each key, and put a key holder's information into the database. Also, they can record which key works in which lock and check the blank needed and codes used for creating keys.

Second, the database system will improve user experiences for recording and inventory management. The property team can make a record with a few inputs and a click button. And they can also store data in the same space. The possibility of losing the data will decrease.

Third, the database facilitates the accessibility of records. The team can find the status of keys in a few minutes as they receive a request from their staff. It is also possible for them to provide accurate key information to the customers, so the level of customer satisfaction will increase.

BUSINESS QUESTIONS

Users

The main target of the database system is facilities or HR. The property management team will have access to the transaction records of property keys easier than before. They also should record and track which key is connected to which property and the status of property keys such as newly created, destroyed/fixed, or lost/found. We concentrate on only administration teams who will access the database in terms of convenience, high accuracy, and data security.

Requirements

The new key tracking database has several requirements as below.

1. Key Transactions

This function presents the status of each key such as creation, assignment, return, and loss or destruction of keys. Also, with this function, the staff member can record a history of registering keys for each asset (deadbolt, interior door, exterior door, desk, cabinet, padlock, elevator, alarm panel, and vehicle).

2. Documentation

In this function, we can track what keys work in what locks. Also, we can find the blank needed and code used for creating keys. These records can be documented by displaying the list view in the application.

3. Inventory Management

The inventory management function will allow the staff member in the facility management team to check the key inventory.

Functions

Based on the above requirements, we defined the functions in the new key-tracking database system.

Function 1: Viewing what keys are assigned to an employee.
Use case: HR looks up the employee to verify what keys should be returned when off-boarding staff.

Function 2: Viewing all employees that are assigned a particular key.
Use case: When rekeying a door lock, Facilities can look up everyone who has access to that lock to make sure they get the new key.

Function 3: Track key data (storage hook, bitting, blank type, keys on hand)
Use case: The information facilities need to make keys (blank, bitting, stamp) needs to be easily available.

Function 4: Track asset data
Use case: Allows Facilities to look up an asset (room, door, desk, vehicle) to find the keys that operate it.

DATA REQUIREMENTS

Before we prepare for this new key tracking database system, we approached by dividing the three main parts such as keys, keyholders, and building/assets.

Key Database Data Requirements

In this section, we will use the terms: a connection table and an additional information table.

A connection table is for connecting more than two tables. And an additional information table has the data for detailed categories of a particular table.

The reasons of creating an additional information table and a connection table are the followings.

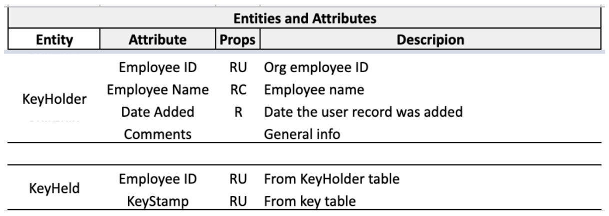
1. We do not want to make the data structure complex.
2. We want to project from losing the data in the database system.

1. Key

Entities and Attributes			
Entity	**Attribute**	**Props**	**Description**
Key	KeyStamp	RU	Code stamped on key
	Blank		Key blank needed to cut key
	Bitting	RU	Machine bitting to cut key
	Description		General info
	Hook	RU	Where made key is hung
	Made	D	Number made - generated by log
	Found	D	Number found - generated by log
	Returned	D	Number returned - generated by log
	Issued	D	Number issued - generated by log
	Destroyed	D	Number destroyed - generated by log
	Lost	D	Number lost - generated by log
	Available	D	Number on hand - Creat+Fnd+Rtrn-Issue-Dstr-Lost
KeyLog	Event ID	RU	System generated
	KeyStamp	R	Code stamped on key
	EventType	R	Made, found, returned, issued, etc
	event_date	R	Date
	Quantity	R	Number of keys
	KeyHolder	R	Keyholder assigned
	Authorized by	R	Facilities staff assigning key
Blank	Code	RU	Code stamped on blank key
KeyClass	Type	RU	Master, not master
Hook	ID	RU	cabinet & hook # (A1-400, B1-350, C1-120, D1-80)

- Key: Key Specification
  - Columns: Key stamp(id), Status (created, found, destroyed, etc.), Brief Description
- KeyLog: Key Registration Record
  - Columns: Event (Registration) id, date, key holder, etc.
- Blank, Hook, KeyClass: Additional Information

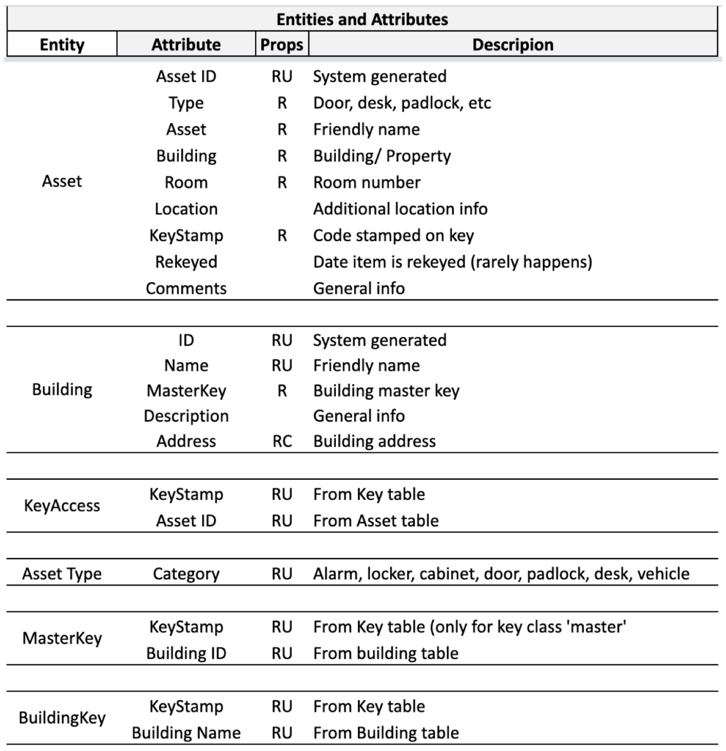
2. Keyholder



Entities and Attributes			
Entity	Attribute	Props	Descripion
KeyHolder	Employee ID	RU	Org employee ID
	Employee Name	RC	Employee name
	Date Added	R	Date the user record was added
	Comments		General info
KeyHeld	Employee ID	RU	From KeyHolder table
	KeyStamp	RU	From key table

- KeyHolder: The information who holds the key
  - Columns: Employee id, and date the user record was added, brief comments
- KeyHeld: A connection table between KeyHolder and Keys table.
  - Columns: Employee id (from the KeyHolder table) and KeyStemp (from the Keys table)

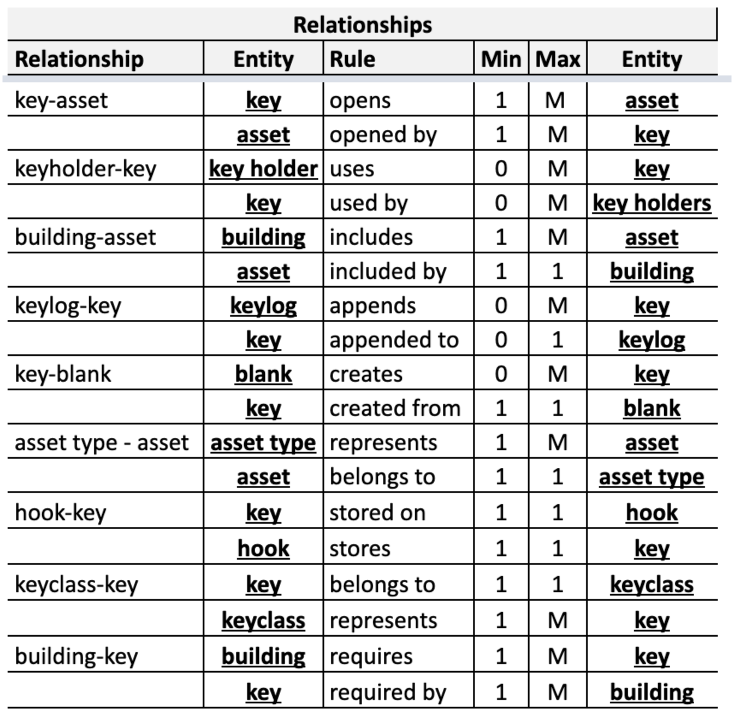
3. Building



Entities and Attributes			
Entity	Attribute	Props	Descripion
Asset	Asset ID	RU	System generated
	Type	R	Door, desk, padlock, etc
	Asset	R	Friendly name
	Building	R	Building/ Property
	Room	R	Room number
	Location		Additional location info
	KeyStamp	R	Code stamped on key
	Rekeyed		Date item is rekeyed (rarely happens)
	Comments		General info
Building	ID	RU	System generated
	Name	RU	Friendly name
	MasterKey	R	Building master key
	Description		General info
	Address	RC	Building address
KeyAccess	KeyStamp	RU	From Key table
	Asset ID	RU	From Asset table
Asset Type	Category	RU	Alarm, locker, cabinet, door, padlock, desk, vehicle
MasterKey	KeyStamp	RU	From Key table (only for key class 'master'
	Building ID	RU	From building table
BuildingKey	KeyStamp	RU	From Key table
	Building Name	RU	From Building table

- Building: The information of each building
  - Columns: Related master key, building name and address
- Asset: The information of each asset
  - Columns: building, room, location, asset type, etc
- KeyAccess, BuildingKeys: Connection between Key and Asset/Building
- Asset Type: Additional Information of Assets
- Master Key: Managing for only master key

We also defined the required relationships as the below table.

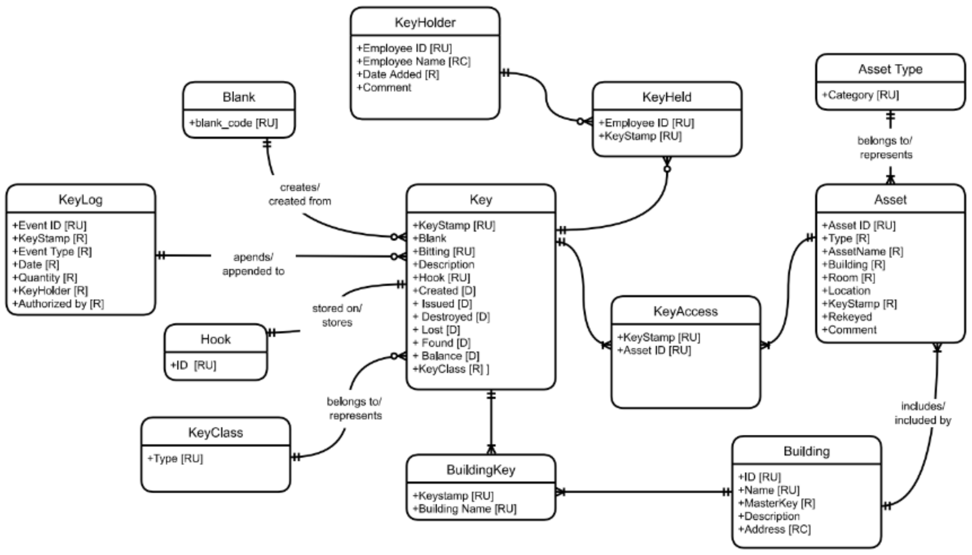


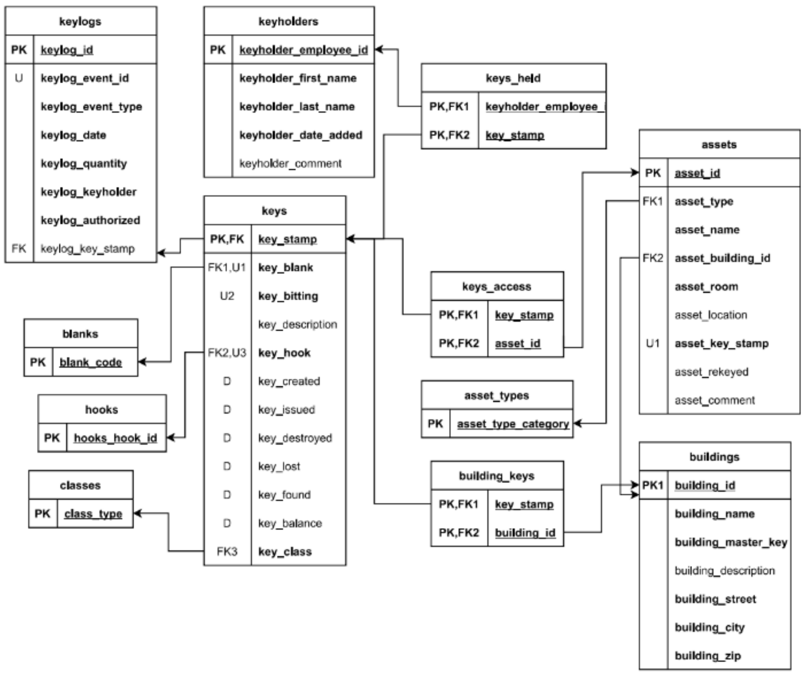
Relationships					
Relationship	**Entity**	**Rule**	**Min**	**Max**	**Entity**
key-asset	**key**	opens	1	M	**asset**
	asset	opened by	1	M	**key**
keyholder-key	**key holder**	uses	0	M	**key**
	key	used by	0	M	**key holders**
building-asset	**building**	includes	1	M	**asset**
	asset	included by	1	1	**building**
keylog-key	**keylog**	appends	0	M	**key**
	key	appended to	0	1	**keylog**
key-blank	**blank**	creates	0	M	**key**
	key	created from	1	1	**blank**
asset type - asset	**asset type**	represents	1	M	**asset**
	asset	belongs to	1	1	**asset type**
hook-key	**key**	stored on	1	1	**hook**
	hook	stores	1	1	**key**
keyclass-key	**key**	belongs to	1	1	**keyclass**
	keyclass	represents	1	M	**key**
building-key	**building**	requires	1	M	**key**
	key	required by	1	M	**building**

- Key – Asset (1:M Relationship)
  - A key opens one or many assets.
  - An asset is opened by one key.
- Key Holder – Key (0:M Relationship)
  - A key holder uses many key. (or a key holder may not have a key.)
  - A key is used by many keyholders.
- Building – Asset (1:M Relationship)
  - A building includes one or many assets.
  - An asset is included by one building. (1:1 Relationship)
- Key Log – Key (0:M Relationship)
  - A building includes many assets or it doesn't have any key.
  - A key does not have any key log or it has one log. (0:1 Relationship)
- Blank – Key (0:M Relationship)
  - A blank creates many keys or it doesn't have any key.
  - A key is created from one blank. (1:1 Relationship)
- Asset– Asset Type (1:M Relationship)
  - An asset type represents one or more asset.
  - An asset belongs to one asset type.
- Hook – Key (1:1 Relationship)
  - A hook stores one key.
  - A key is stored on one hook.
- Key Class – Keys (1:M Relationship)
  - A key class represents one or more keys.
  - A key belongs to one key class. (1:1 Relationship)
- Building – Key (1:M Relationship)
  - A building requires one or more keys.
  - A key is required by one or more buildings.

Conceptual and Logical Data Model

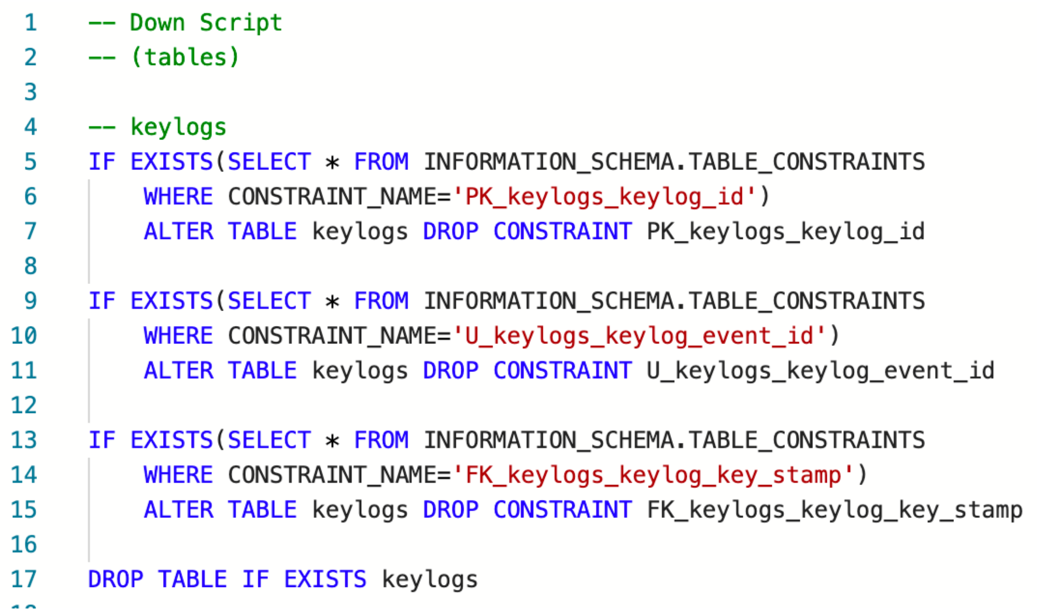
We created conceptual and logical data model diagrams based on the above data requirements (entities, columns, and relationships) as below.





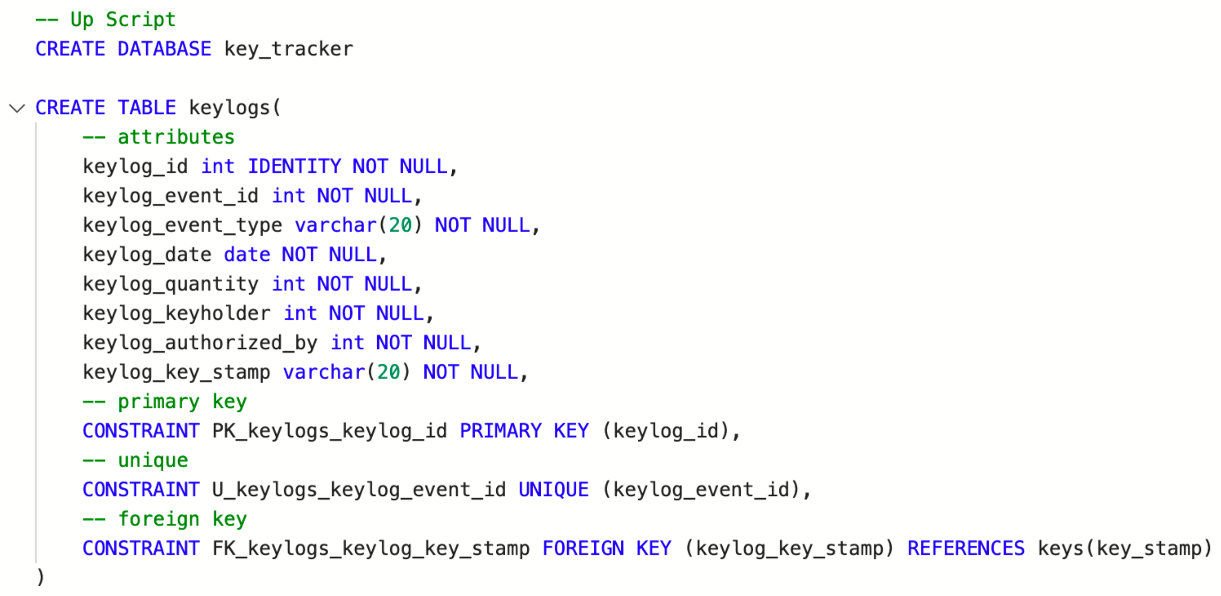
SQL CODES

1. Up and Down Scripts
We wrote SQL down and up scripts to create the key tracker database for keylogs and keyholders by creating tables and constraints.



```sql
-- Down Script
-- (tables)

-- keylogs
IF EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    WHERE CONSTRAINT_NAME='PK_keylogs_keylog_id')
    ALTER TABLE keylogs DROP CONSTRAINT PK_keylogs_keylog_id

IF EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    WHERE CONSTRAINT_NAME='U_keylogs_keylog_event_id')
    ALTER TABLE keylogs DROP CONSTRAINT U_keylogs_keylog_event_id

IF EXISTS(SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
    WHERE CONSTRAINT_NAME='FK_keylogs_keylog_key_stamp')
    ALTER TABLE keylogs DROP CONSTRAINT FK_keylogs_keylog_key_stamp

DROP TABLE IF EXISTS keylogs
```

- **Down Script:** We added the SQL codes for dropping constraints and tables



```sql
-- Up Script
CREATE DATABASE key_tracker

CREATE TABLE keylogs(
    -- attributes
    keylog_id int IDENTITY NOT NULL,
    keylog_event_id int NOT NULL,
    keylog_event_type varchar(20) NOT NULL,
    keylog_date date NOT NULL,
    keylog_quantity int NOT NULL,
    keylog_keyholder int NOT NULL,
    keylog_authorized_by int NOT NULL,
    keylog_key_stamp varchar(20) NOT NULL,
    -- primary key
    CONSTRAINT PK_keylogs_keylog_id PRIMARY KEY (keylog_id),
    -- unique
    CONSTRAINT U_keylogs_keylog_event_id UNIQUE (keylog_event_id),
    -- foreign key
    CONSTRAINT FK_keylogs_keylog_key_stamp FOREIGN KEY (keylog_key_stamp) REFERENCES keys(key_stamp)
)
```

- **Up Script:** We created tables and related constraints such as primary keys, foreign keys, and unique keys by following the data requirements and logic diagrams.

2. Add Values

After creating the tables, we put the values into each table using the below codes.

```sql
--keyholders table data insert
insert into keyholders (keyholder_employee_id, keyholder_first_name, keyholder_last_name, keyholder_date_added, keyholder_comment)
values
(616514,Elizabeth,Myers,"2022-12-04",'Signs out keys for IT Dept'),
(354654,Todd,Olden,"2022-12-04"," "),
(418651,Tyler,Reddo,"2022-12-04"," "),
(156678,Rick,Kent,"2022-12-04"," "),
(321562,Belinda,Sayer,"2022-12-04","CHRO"),
(746987,Suzie,Quaker,"2022-12-04"," "),
(439461,Frosty,Paws,"2022-12-04"," "),
(433351,Izzy,Girl,"2022-12-04"," "),
(596348,Pandora,DeGrey,"2022-12-04"," "),
(941683,Iggy,Quaker,"2022-12-04",'Limited access, loses keys');
```

3. Additional Functions

After creating the tables, we put the values into each table using the below codes. First, we defined functions which have to put into the application and wrote codes.

Function 1: Helps us view the assigned key of a particular employee

```sql
-- FUNCTIONS
/*
1.  Requirement: Looking up employee to see what keys they were assigned.
a.  Use case: HR can look up employee to see what keys needs to be returned when offboarding staff.
*/

SELECT keys_held.keyholder_employee_id, keys.key_stamp
FROM keys_held
LEFT JOIN keys ON keys_held.key_stamp =keys.key_stamp
-- WHERE keys_held.keyholder_employee_id = [input] or key_held.keyholder_last_name LIKE '%[input]%' or key_held.keyholder_first_name '%[input]%'
ORDER BY keys_held.keyholder_employee_id;
```

Function 2: Helps us view all the employees that have been assigned a key

```sql
/*
2.  Requirement: Looking up key to see all employees that have it.
a.  Use case: When rekeying a door lock, Facilities can look up everyone
who has access to that lock to make sure they get the new key.
*/
SELECT keyholders.keyholder_employee_id, concat(keyholders.keyholder_first_name, ' ', keyholders.keyholder_last_name) as employee_name
FROM keyholders
RIGHT JOIN keys_held ON keyholders.keyholder_employee_id =keys_held.keyholder_employee_id;
-- WHERE keys.key_stamp ='';
```

Function 3: Helps us track all the key data

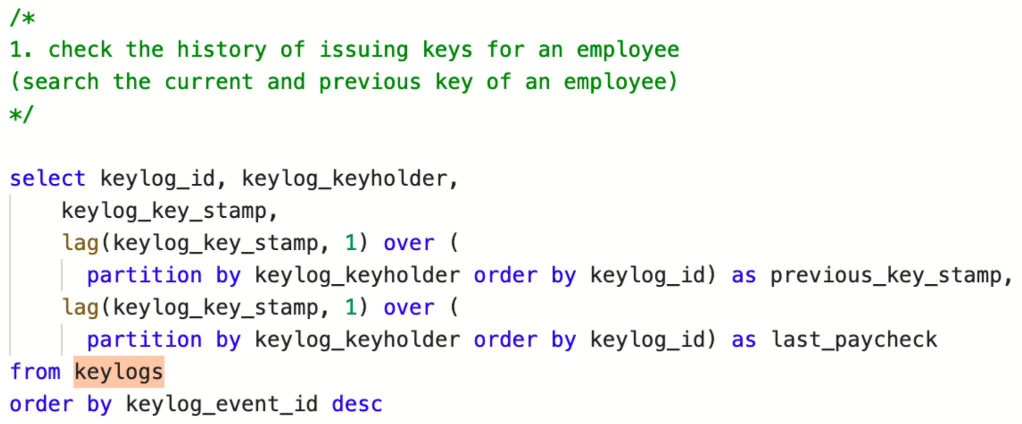
```sql
/*
3.  Requirement: Track key data – storage hook, bitting (machine configuration to cut new key),
blank needed to make new key, how many keys are on hand.
a.  Use case: When Facilities needs to make a key, they have the information needed
(blank, bitting, stamp) at their fingertips.
*/
SELECT key_stamp,key_bitting,key_hook,key_blank,key_balance AS keys_on_hand
FROM keys;
```

Function 4: Helps us track the assists and also the key-building data

```
/*
4.  Requirement: Track asset data
a.  Use case: Allows Facilities to look up an asset (room, door, desk, vehicle)
to find the key information for the asset.
*/

SELECT assets.asset_id,assets.asset_type,assets.asset_name,keys.key_bitting,keys.key_stamp,keys.key_hook,keys.key_class
FROM assets
LEFT JOIN keys ON keys.key_stamp = assets.asset_key_stamp;
-- WHERE assets.asset_id = ;
```

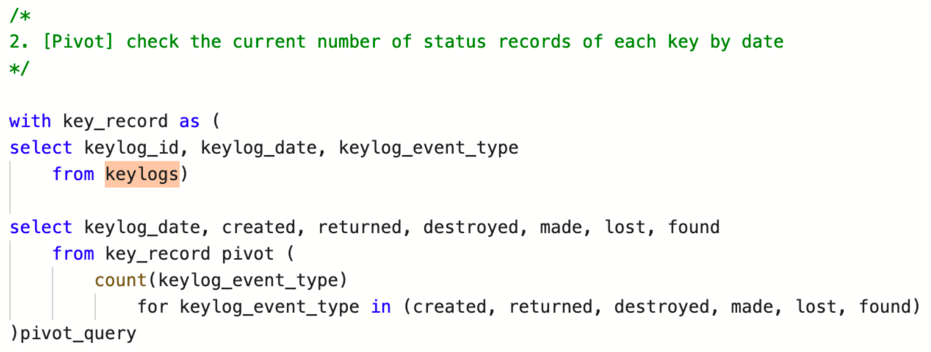
Also, we added additional window functions using a pivot method and Value Window Functions.

Additional Function 1: Check the current and previous key stamp of each employee (History Tracking)



```
/*
1. check the history of issuing keys for an employee
(search the current and previous key of an employee)
*/

select keylog_id, keylog_keyholder,
    keylog_key_stamp,
    lag(keylog_key_stamp, 1) over (
      partition by keylog_keyholder order by keylog_id) as previous_key_stamp,
    lag(keylog_key_stamp, 1) over (
      partition by keylog_keyholder order by keylog_id) as last_paycheck
from keylogs
order by keylog_event_id desc
```

Additional Function 2: Check the current number of status records by date



```
/*
2. [Pivot] check the current number of status records of each key by date
*/

with key_record as (
select keylog_id, keylog_date, keylog_event_type
    from keylogs)

select keylog_date, created, returned, destroyed, made, lost, found
    from key_record pivot (
        count(keylog_event_type)
            for keylog_event_type in (created, returned, destroyed, made, lost, found)
)pivot_query
```

4. Stored Procedure and Trigger

In the final stage, we created a stored procedure and a trigger to update(or insert) a new comment of an employee and a master key of a building.

Stored Procedure: Update a new comment for an employee, if an employee already exists leave as is, if not update and insert new employee with a new comment.

```sql
/* stored procedure */
-- update a new comment for an employee, if an employee already exists leave as is, if not update and insert new employee
drop procedure if exists p_upsert_keyholder_comment
GO
create procedure p_upsert_keyholder_comment (
    @keyholder_employee_id int,
    @new_comment varchar(500)
) as begin
    if exists(select * from keyholders where keyholder_employee_id = @keyholder_employee_id) begin
        update keyholders set keyholder_comment = @new_comment
            where keyholder_employee_id = @keyholder_employee_id
    end
    else begin
        insert into keyholders (keyholder_employee_id, keyholder_comment)
            values (@keyholder_employee_id, @new_comment)
    end
end
GO

select * from keyholders
exec dbo.p_upsert_keyholder_comment 156678, 'GM'
select * from keyholders
```

<u>Trigger</u>: Update a master key of a building

```sql
-- trigger to update a master key of a building
GO
CREATE TRIGGER t_update_master_key_1 ON buildings
AFTER UPDATE, INSERT AS
BEGIN
    declare @new_master_key varchar(10)
    if exists (select i.building_master_key from inserted i) begin
    update buildings set building_master_key = building_master_key + @new_master_key
    where building_id in (select i.building_id from inserted i)
END

update buildings set building_master_key = 'ABMK' where building_id = 2

select * from buildings
```

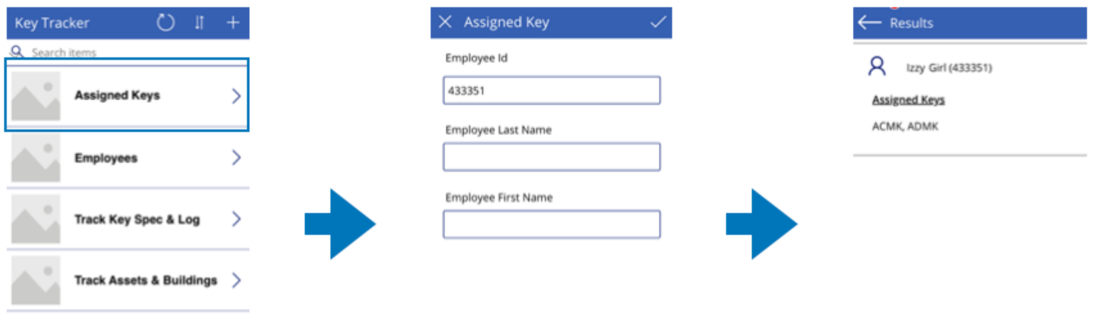
APPLICATIONS

Following are the snapshots of the working PowerApps with the functions which we created in the SQL.

Main Page & Function 1: Helps us view the assigned key of a particular employee

- Screen 1: The application displays the list of functions.
  - Assigned keys: Function 1
  - Employees: Function 2
  - Track Key Spec & Log: Function 3
  - Track Assets & Buildings: Function 4
- Screen 2: When a user clicks the '**Assigned Keys**' button, the application shows three input sections to search the keys and the employee who currently has the key.
- Screen 3: The result is shown as below. We can check the name and id of an employee, and which key the employee has.

Function 1 - Viewing What Keys Are Assigned to an Employee.



433351

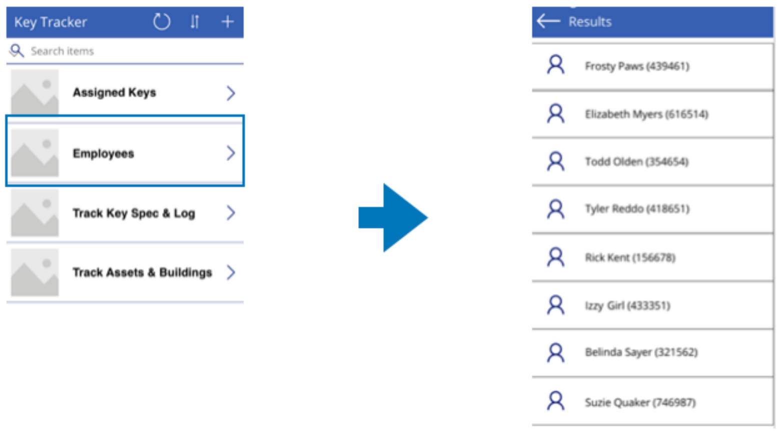
Izzy Girl (433351)

ACMK, ADMK

Function 2: Helps us view all the employees that have been assigned a key

- Screen 1: The application displays the list of functions.
- Screen 2: When a user clicks the '**Employees'** button, the application shows the list of all employees who have currently one or more keys. (The employee who does not have a key will not be displayed on this screen.)

Function 2 - Viewing All Employees That Are Assigned a Key



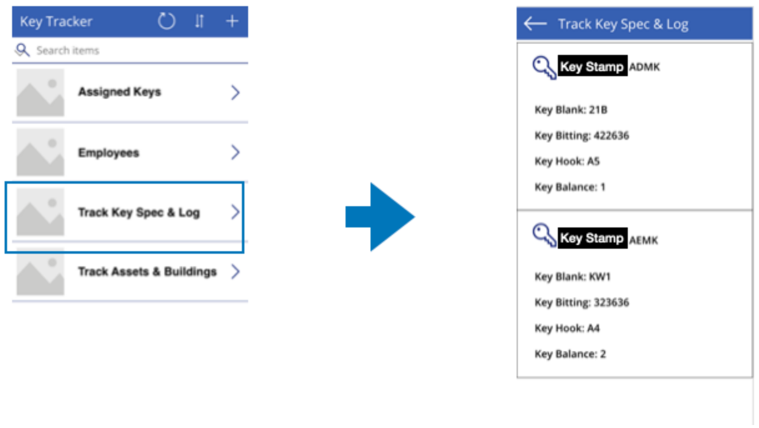
Frosty Paws (439461)
Elizabeth Myers (616514)
Todd Olden (354654)
Tyler Reddo (418651)
Rick Kent (156678)
Izzy Girl (433351)
Belinda Sayer (321562)
Suzie Quaker (746987)

*** This function shows employees who has an assigned key.**

Function 3: Helps us track all the key data

- Screen 1: The application displays the list of functions.
- Screen 2: When a user clicks the '**Track Key Spec & Log** button, the application shows the specification of each key which is recorded in the database system.
  - It includes 1) key stamp, 2) key blank, 3) key bitting, 4) key hook, and 5) key balance (how many keys are registered in the database system.)

Function 3 - Track Key Data (Storage Hook, Bitting, Blank, Keys on Hand)



Key Stamp ADMK
Key Blank: 21B
Key Bitting: 422636
Key Hook: A5
Key Balance: 1

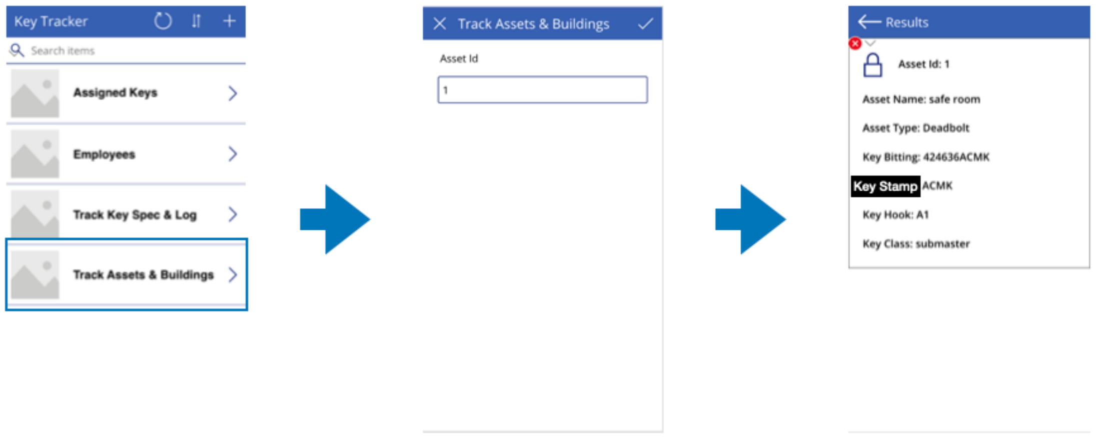
Key Stamp AEMK
Key Blank: KW1
Key Bitting: 323636
Key Hook: A4
Key Balance: 2

Function 4: Helps us track the assists and also the key-building data

- Screen 1: The application displays the list of functions.
- Screen 2: When a user clicks the '**Track Assets & Buildings'** button, the application shows the input section for searching with asset id.

- Screen 3: It shows the result with the information: the id, name, and type of an asset, the bitting, stamp, hook, and class information of a key that is related to the asset.

Function 4 - Track Assets & Building Data



Asset Id
1

Asset Id: 1
Asset Name: safe room
Asset Type: Deadbolt
Key Bitting: 424636ACMK
Key Stamp ACMK
Key Hook: A1
Key Class: submaster