# What personal data is in the database?

* The names of airline employees, what airline they work for, & the names of their dependents

# What limits should be placed on our users’ ability to inspect the system?

* Except for employees & dependents, users should not know what other users are in the system.
* Users should not know who fulfilled their load request

# How might our security be breached?

* We could have an SQL injection on one of the input forms
  + We’ll likely have text input forms for fulfilling load requests
  + At its worst breach could potentially result in doxxing of users (It’s my opinion that name & workplace is enough to go find the rest of a person’s information) or our database having all its information deleted, rendering our tool less useful
  + We can defend against this problem through stored procedures, which do not allow for SQL injection. We can also restrict our users to only being able to interact with our database through the stored procedures necessary to run the MyID90 program
* A user could log in directly to the database & start executing code
  + The consequences are the same as SQL injection, because both attacks are about running arbitrary SQL code.
  + Once again, we can protect against this attack through proper user permissions: Users will only be able to interact with the database through the minimum number of stored procedures needed to operate MyID90

# Integrity Constraints

## Entity Integrity

* A user cannot have less than 0 tokens
* A flight cannot have a load of less than 0
* A flight’s last load update cannot be after the current time
* To prevent invalid data in these cases, we’ll use a check statement.

## Referential Integrity

* Dependent should never refer to an employee that doesn’t exist. If an employee is removed from the database, cascade, because the dependents will no longer be able to do anything useful
* If an airline that an employee or flight depends on would be removed, reject.
* If a flight that a load request or an updates depends on is removed, cascade (This rule was made under the assumption that a flight might be cancelled)
* If an employee that a load request depends on is removed, set the reference to null, because someone might still want to fulfill the load request so that everyone else can benefit
* If an employee that an updates depends on is removed, cascade

## Business Logic

???