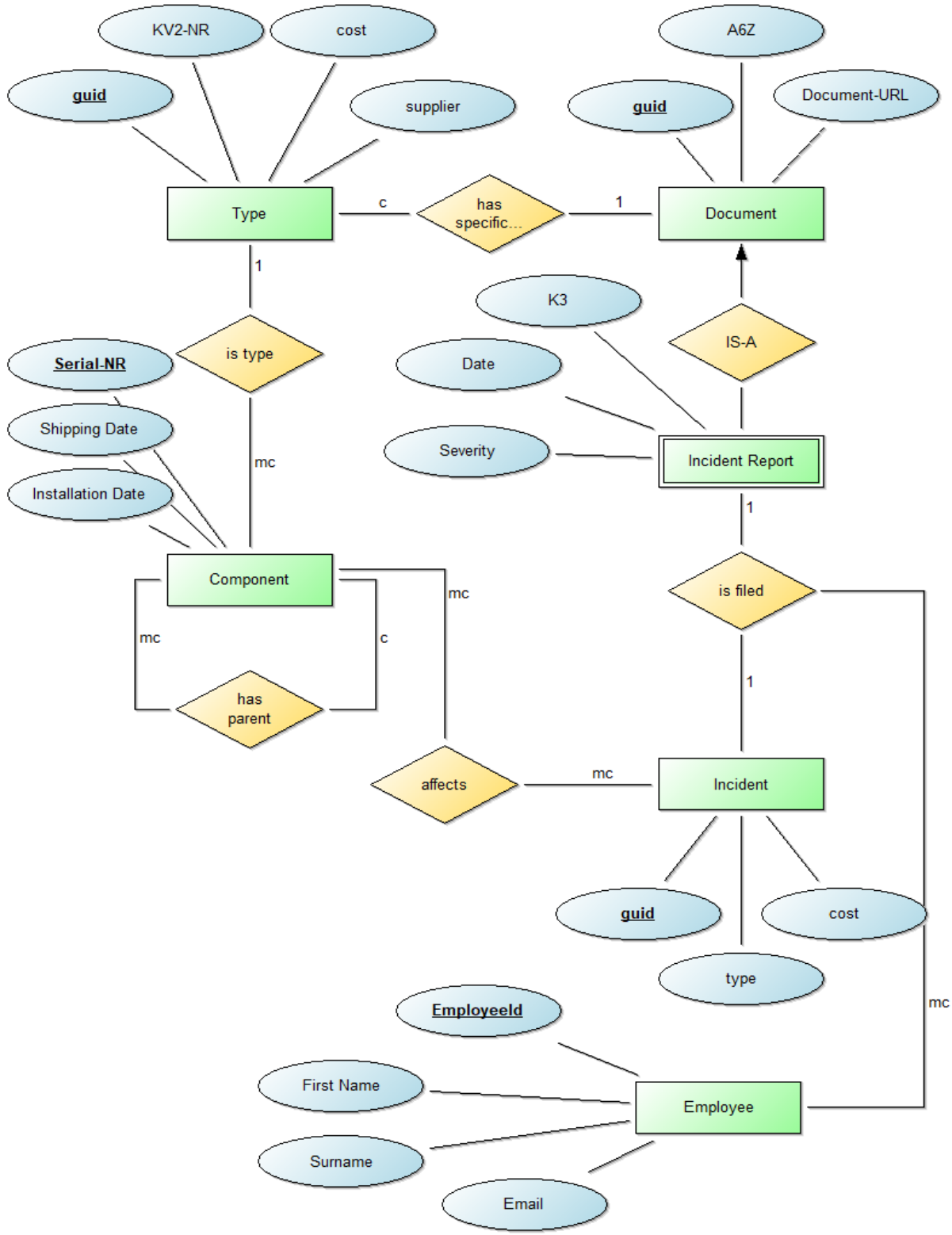
Meilenstein 1

Anforderungsanalyse

Ein Unternehmen fertigt Produkte aus zugekauften Einzelteilen. Jedes dieser Teile entspricht einem Typen, zu welchen eine alphanumerische KV2-Nummer, der Lieferant und die Stückkosten bekannt sind. Jeder Typ hat genau eine Spezifikation. Die Spezifikationen sind Dokumente, zu welchen eine alphanumerische AZ6-Nummer und eine URL, wo das Dokument hinterlegt ist, bekannt ist. Zu den Einzelteilen wird der Typ gespeichert, sowie eine eindeutige Seriennummer, das Lieferdatum und das Einbaudatum. Auch wird als Hierarchie gespeichert, wenn ein Einzelteil in einem anderem verbaut wird. Wenn Vorfälle geschehen, werden diese gespeichert, mit der Vorfalls-Art, den verursachten Kosten und den betroffenen Einzelteilen. Mitarbeiter, zu denen eine Mitarbeiternummer, der Name und eine E-Mail bekannt sind, können zu jedem Vorfall genau einen Vorfalls-bericht verfassen. Dies sind Dokumente, bei denen zusätzlich eine K3-Nummer, das Vorfalls Datum und Schwere des Vorfalls gespeichert werden. Es wird gespeichert, welcher Mitarbeiter welchen Bericht zu welchem Vorfall verfasst hat.

ER-Diagram


KV2-NR
cost
guid
supplier
Type
c
has specific...
1
Document
A6Z
guid
Document-URL
1
is type
Serial-NR
Shipping Date
Installation Date
mc
K3
Date
Severity
IS-A
Incident Report
1
Component
mc
mc
c
has parent
is filed
1
affects
mc
Incident
guid
cost
type
mc
EmployeeId
First Name
Surname
Email
Employee

Meilenstein 2

Relationenschema

Document(**guid**, az6, document-url) PK: guid

Type(**guid**, *specification*, kv2, supplier, cost) PK: guid

type.specicifaction ◊ document.guid

Component(**serialNr**, *parent*, *type*, ,) PK: serialNrn

component.parent ◊ component.serialNr

component.type ◊ type.guid

IncidentReport(***guid***, k3, date, severity) PK: guid

IncidentReport.guid ◊ document.guid

Employee(**employeeId**, firstname, surname , email) PK: employeeId

Incident(**guid**, type, cost) PK: guid

IsFiled(***reportGuid***, ***employeeId***, ***incidentGuid***)

TS: reportGuid, employeeId, incidentGuid

isFiled. reportGuid ◊ incidentReport.guid

isFiled. employeeId ◊ employee.employeeId

isFiled. incidentGuid ◊ incident.guid

Affects(***componentGuid***, ***incidentGuid***) TK: componentGuid, incidentGuid

affects.componentGuid ◊ component.guid

affects.incidentGuid ◊ incident.guid

Meilenstein 4

Java

Das Java Programm erfüllt zwei Funktionen, das Generieren der Testdaten und das Importieren in die Datenbank. Die beiden Funktionen sind zwar in der Ausführung verbunden da, um manche Daten zu generieren andere Tabellen schon befüllt sein müssen, sind von der Programmierung allerdings unabhängig.

Daten Generieren

Um die Daten zu generieren, wird erst der Text einer CSV-Datei generiert und in einem StringBuilder gespeichert, welcher als eine Operation in die Entsprechende Datei geschrieben wird.

Daten Importieren

Zur Verbindung mit der Oracle Datenbank verwendet mein Programm OJDBC11, welcher über Maven importiert wird. Alle Tabellen werden über eine Methode befüllt, welche so generalisiert wurde, dass der Tabellenname, die Spaltennamen und die Spalten-Datentypen als Parameter übergeben werden können. Die einzelnen Datensätze werden aus einer CSV-Datei gelesen und mittels Prepared-Statement als Batch abgearbeitet.

PHP

Die PHP-Anwendung basiert auf der Beispielimplementierung, wobei die Datenbankzugriffe für bessere Übersicht auf mehrere Dateien aufgeteilt wurden. Ein paar Elemente wurden mit CSS angepasst, aber der Großteil der Anwendung ist pures HTML.