SELECT Person.name FROM Person JOIN Planet on Person.birthplace = Planet.name WHERE NOT Planet.destroyed;

Towards a Concurrent Implementation of Keyword Search Over Relational Databases

by

Richard J.I. Drake

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science (M.Sc.)

in

The Faculty of Science

Computer Science

University of Ontario Institute of Technology

Supervisor: Dr. Ken Q. Pu

December 2013

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Background (2 days)

Literature search on:

- DBExplore
- XRank
- BANKS
- ...

Chapter 2

A Tale of Two Data Models

The term "data model" refers to a notation for describing data and/or information. It consists of the data structure, operations that may be performed on the data, as well as constraints placed on the data [GUW09].

In this chapter we provide a formal definition of the relational data model, discuss its merits, its shortcomings, and contrast it to the document data model. Contrary to the relational model, the document model permits fast and flexible keyword search without requiring explicit domain knowledge of the data. In addition, we demonstrate the feasibility of encoding a relational model into a document model in a lossless manner.

2.1 Relational Model

In its most basic form, the relational data model is built upon sets and tuples. Each of these sets consist of a set of finite possible values. Tuples are constructed from these sets to form relations.

Definition 1 (Named Tuple). A named tuple t is an instance of a relation r, consisting of values corresponding to the attributes of r. For example,

$$t = \{\text{name} : \text{“Jack Bauer”}, \text{age} : 39\}$$

We denote the attributes of t as $\text{ATTR}[t] = \{\text{name}, \text{age}\}$ The values are $t[\text{name}] =$ "Jack Bauer", and $t[\text{age}] = 39$.

Definition 2 (Relation). A relation r is a set of named tuples, $r = \{t_1, t_2, \ldots, t_n\}$, such that all the named tuples share the same attributes.

$$\forall t, t' \in r, \text{ATTR}[t] = \text{ATTR}[t']$$

For example,

$$r = \left\{ \begin{array}{l} \{\text{name} : \text{"Jack Bauer"}, \text{age} : 39\}, \\ \{\text{name} : \text{"Bruce Wayne"}, \text{age} : 39\}, \\ \{\text{name} : \text{"Clark Kent"}, \text{age} : 45\} \end{array} \right\}$$

Relations are typically represented as tables.

name	age
"Jack Bauer"	39
"Bruce Wayne"	39
"Clark Kent"	45

Table 2.1: Person table

Definition 3 (Keys). Keys are constraints imposed on relations. A key constraint K on a relation r is a subset of $\text{ATTR}[r]$ which may uniquely identify a tuple. Formally, we say r satisfies the key constraint K, denoted as $r \vDash K$, subject to

$$\forall t, t' \in r, t \neq t' \implies t[K] \neq t'[K]$$

For example, in Table 2.1, the relation satisfies the key constraint $\{\text{name}\}$, but not $\{\text{age}\}$.

Definition 4 (Foreign Keys). A foreign key constraint applies to two relations, r_1, r_2. It asserts that values of certain attributes of r_1 must appear as values of some corresponding attributes of r_2. A foreign key constraint is written as

$$\theta = r_1(a_1, a_2, \ldots, a_k) \rightarrow r_2(b_1, b_2, \ldots, b_k)$$

where $a_i \subseteq \text{ATTR}[r_1]$ and $b_i \subseteq \text{ATTR}[r_2]$. We say (r_1, r_2) satisfies θ, denoted as $(r_1, r_2) \models \theta$, if

$$\forall t \in r_1, \exists t' \in r_2 \mid t[a_1, a_2, \ldots, a_k] = t'[b_1, b_2, \ldots, b_k]$$

Example 1. Suppose we have a relation Superhero(name, superpower). We can impose a FK constraint of

$$\text{Superhero(name)} \rightarrow \text{Person(name)}$$

Definition 5 (Relational Database). A relational database, d, is a named collection of relations (as defined by Definition 2, keys (as defined by Definition 3), and foreign key constraints (as defined by Definition 4.

We use $\textsc{Name}[d]$ to denote the name of d, $\textsc{Rel}[d]$ the list of relations in d, $\textsc{Key}[d]$ the list of key constraints of d, and $\text{FK}[d]$ the list of foreign key constraints of d.

2.1.1 Schema Group

Definition 6 (Schema Graph). If we view relations as vertices, and foreign key constraints as edges, a database d can be viewed as a *schema graph* G, formally defined as

$$\begin{aligned} \text{vertices} : V(G) &= \text{REL}[d] \\ \text{edges} : E(G) &= \text{FK}[d] \end{aligned}$$

Example 2. Given the following schema