Anfragegeneralisierung zur komprimierten Repräsentation von Indexen semantischer Caches auf mobilen Endgeräten

Hagen Höpfner* und Eike Schallehn
{hoepfner|eike}@iti.cs.uni-magdeburg.de
Institut für Technischen und Betriebliche Informationssysteme
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
Postfach 4120, 39016 Magdeburg, Deutschland

Abstract: Mobile Endgeräte sind in erster Linie – verglichen mit etablierten Desktop-Systemen – durch Restriktionen gekennzeichnet. Diese reichen von beschränkten Eingabemöglichkeiten über teure und langsame Datennetzanbindungen bis hin zur stark eingeschränkten Speicherkapazität. Insbesondere die zahlreichen Nachteile von Funknetzwerken werden häufig durch das lokale Zwischenspeichern von Daten auf den Mobilgeräten reflektiert. Offensichtlich besteht hierbei ein Konflikt zwischen der Cachegröße und der verfügbaren Speicherkapazität. In diesem Papier werden erste Ideen zur Ausnutzung semantischer Ähnlichkeiten von Anfragen zur Komprimierung von Cacheeinträgen durch Anfragegeneralisierung vorgestellt.

1 Einleitung und Motivation

Mobile Informationssysteme unterliegen zahlreichen Beschränkungen, welche in erster Linie aus ihren jeweiligen Einsatzgebieten und den verwendeten Geräten resultieren. Funknetzwerke sind zwar mittlerweile fast flächendeckend verfügbar, die Datenübertragung ist jedoch im Allgemeinen vergleichsweise teuer und langsam. Aus diesem Grund wird der Zugriff auf im Netzwerk verfügbaren Informationen häufig durch Zwischenspeichertechniken unterstützt. Generell gilt: Daten, die nicht über das Netzwerk bezogen werden müssen, sind kostengünstiger. Eine weitere Einschränkung ist, dass Mobilgeräte in der Regel leichtgewichtig sind. Das heißt, dass sie nur beschränkt Daten speichern und verarbeiten können. Offensichtlich tritt hier ein Konflikt auf. Aus Sicht der Datenübertragungskosten wäre es am Besten, alle einmal übertragenen Daten lokal zwischenzuspeichern und neue Anfragen vorrangig lokal zu beantworten. Aufgrund des beschränkten Speicherplatzes auf den Mobilgeräten ist diese Strategie aber nur kurz realisierbar.

In dem vorliegenden Papier stellen wir erste Ideen zu einer kompakten Darstellung der Daten im Zwischenspeicher vor. Dabei werden semantische Überlappungen bzw. Nachbarschaftsbeziehungen unterschiedlicher Anfragen genutzt und mittels Generalisierung die Anzahl der im Zwischenspeicher gehaltenen Referenzanfragen reduziert. Nach der Diskussion verwandter Arbeiten in Abschnitt 2 wird in Abschnitt 3 kurz die gewählte Struktur

*Gefördert durch die DFG unter der Fördernummer SA 782/3-2.

des lokalen Caches erläutert, und darauf aufbauend in Abschnitt 4 ein Algorithmus vorgestellt, der beim Einfügen von neuen Cacheeinträgen deren „Kompatibilität“ mit bereits lokal verfügbaren Anfrageergebnissen prüft.

Zur Abgrenzung sei darauf hingewiesen, dass wir hier kein detailliertes Kostenmodell vorstellen. Vielmehr wird ein Verfahren vorgestellt, welches ausschließlich das Ziel hat, möglichst viele Anfragen lokal zu beantworten und gleichzeitig die dazu notwendigen Metadaten zu reduzieren.

2 Verwandte Arbeiten

Sematische Caches werden in zahlreichen Forschungsgebieten und Anwendungsbereichen mit dem Ziel der Minimierung von Antwortzeiten und Übertragungskosten eingesetzt. Dies schließt klassische Client/Server-Datenbanken [DFJ+96, KB96], heterogene Multi-Datenbank Systeme [GG97, GG99], Web-Datenbanken [LC98, LC01, KSGH03] sowie die hier betrachteten mobilen Informationssysteme [LC99, RD00] ein.

Der Unterschied zu klassichen Zwischenspeichertechniken besteht darin, dass die Anfragen dazu benutzt werden, um die entsprechenden Ergebnisse lokal (in unserem Szenario auf dem mobilen Endgerät) zu referenzieren. Nach [RD99] hängt die Effizienz eines derartigen Caches davon ab, wie stark der semantische Zusammenhang zwischen Anfragen ist und wie effizient der Cache verwaltet wird. Der Begriff Effizienz wir in unserer Arbeit als „möglichst wenig Meta-Informationen“ interpretiert. Daher ist der Ausgangspunkt die Idee der semantischen Regionen [DFJ+96], die besagt, dass die Daten eines Caches derart strukturiert sind, dass sie nichtüberlappende Regionen im Datenraum bilden. Auf diesem Vorschlag aufbauend existieren zwei erweiterte Ansätze.

Bei der in [RD99] vorgeschlagenen semantischen Clusterung, werden ähnliche Anfragen zu Clustern zusammengefasst. Wenn eine Anfrage Q komplett oder zum Teil durch eine bereits gecachte Anfrage C beantwortet werden kann, wird Q in das selbe Cluster wie C aufgenommen womit semantisch ähnliche Anfragen dicht beieinander gespeichert werden. Hierzu wird eine zusätzliche Cluster-Indexstruktur eingeführt. Da unser Ziel eine Reduktion der Metainformationen ist, repräsentieren wir die mit dem Clusterindex vergleichbaren Metainformationen implizit in der notwendigen Indexstruktur des Caches.

Die zweite zentrale Erweiterung der semantischen Regionen diskutiert [LLS99]. Durch die Cache-Partitionierung wird versucht, die häufig benutzten von den weniger oft verwendeten Daten zu trennen. Dazu werden die semantischen Regionen weiter in Sub-Fragmente zerlegt. Es werden also zusätzliche Metainformationen eingefügt, die später durch die Ersetzungsstrategie genutzt werden können, um diejenigen Daten zu entfernen, die am seltensten benutzt wurden. Im Gegensatz zu unserem Verfahren stellt die Partitionierung eine Spezialisierung dar. Die Cacheeinträge werden verfeinert und nicht generalisiert. Aus Sicht der Speichereffizienz wäre sicherlich ein „Mittelweg“ zu empfehlen. Eine diesbezügliche Evaluation wird in Folgearbeiten betrachtet werden.

3 Stuktur des Zwischenspeichers

Der in diesem Papier diskutierte Ansatz basiert auf dem trie-basierten Cache, welcher detailliert in [HS03c] eingeführt wurde. Die Anfrageergebnisse werden als Relationen lokal gespeichert und durch eine Indexstruktur, welche die Anfragen in Form eines Tries [Fre59] repräsentiert, referenziert. Die hierzu verwendete kalkülbasierte Anfragesprache wurde ebenfalls in [HS03c] eingeführt und in [HS03b] derart erweitert, dass kontextbasierte Anfragen als Konjunktionen der folgenden Prädikate repräsentiert werden:

Relationenprädikate r repräsentieren die Attribute welche in das Ergebnis einfließen sollen. Sie sind folglich mit den Parametern des Projektionsoperators der Relationenalgebra vergleichbar.
Beispiel: `address(name,fname)`

Verbundprädikate j repräsentieren Verbünde von Relationen und die dabei zu verwendenden Verbundattribute. Sie sind folglich mit den Parametern des Verbundoperators der Relationenalgebra vergleichbar.
Beispiel: `address.name=phone.last_name`

Selektionsprädikate s repräsentieren die Bedingungen, die erfüllt sein müssen, damit ein Tupel in die Ergebnisrelation aufgenommen wird. Sie sind folglich mit den Parametern der Selektionsoperation der Relationenalgebra vergleichbar. Eine Selektion $\sigma_F(r(R))$ mit der Selektionsbedingung F wird als Selektionsprädikat $r(R).F$ beschrieben. F ist hierbei auf Konstantenselektion beschränkt und muss der Konvention `attribut`θ`constant` genügen. Desweiteren wird gefordert, dass $\theta \in \{\leq, <, =, \neq, \geq, >\}$ gilt und dass logische Verknüpfungen in separate Selektionsprädikate zerlegt werden.
Beispiel: `address.fname = 'George'`

Kontextprädikate c repräsentieren Informationen über den Kontext unter dem die Datenbankanfrage gestellt wurde (z.B. aktuelle Lokation). Auf dem Datenbankserver können sie genutzt werden, um die Anfrage nur auf vordefinierten personalisierte Schemata (Fragmente) auszuführen (vgl. [HS03a]). Auf dem mobilen Endgerät können sie u.A. in die Ersetzungsstrategie des Zwischenspeichers einfließen [1].

Eine kontextbasierte Anfrage $CBQ = \{c_1 \wedge \cdots \wedge c_q \wedge r_1 \wedge \cdots \wedge r_m \wedge j_1 \wedge \cdots \wedge j_n \wedge s_1 \wedge \cdots \wedge s_o\}$ wird als Sequenz von Prädikaten $\langle c_1, \ldots, c_q, r_1, \ldots, r_m, j_1, \ldots, j_n, s_1, \ldots, s_o \rangle$ repräsentiert, wobei $\forall i, k \in 1 \ldots q, i < k \Rightarrow c_i \triangleleft c_k$, $\forall i, k \in 1 \ldots m, i < k \Rightarrow r_i \triangleleft r_k$, $\forall i, k \in 1 \ldots n, i < k \Rightarrow j_i \triangleleft j_k$ und $\forall i, k \in 1 \ldots o, i < k \Rightarrow s_i \triangleleft s_k$ gilt.

Hierbei steht $\triangleleft$ für lexikographisch kleiner. Offensichtlich ist diese Anfragesprache nicht streng relational vollständig. Beispielsweise ist kein Vereinigungsoperator definiert. Für zukünftige Arbeiten wird aber die relationale Vollständigkeit angestrebt. Daneben werden aufgrund des Kalkülcharakters der Sprache keine Aggregatfunktionen unterstützt.

[1] Wird in diesem Papier aber vorerst nicht betrachtet.

Die trie-basierte Indexstruktur repräsentiert nun derartige kontextbasierte Anfragen als Pfade eines Tries (von der Wurzel zu den Blättern). Die Blätter bilden hierbei die jeweiligen Anfrageergebnisse. Somit kann durch Traversierung des Tries in Kombination mit entsprechenden Prädikatvergleichen effizient festgestellt werden, ob neue Anfragen komplett, partiell oder garnicht lokal beantwortet werden können. Für eine genaue Beschreibung dieses Vorgehens sei auf [HS03c] verwiesen.

4 Anfragegeneralisierung

Ziel der Generalisierung von Anfragen ist es, die Anzahl der Index-Einträge zu reduzieren und dennoch möglichst genausoviele Anfragen lokal beantworten zu können. Das heisst, dass die folgende Situation betracht werden muss:

- Eine Anfrage Q konnte lokal nicht beantwortet werden und wurde an den Server geschickt.
- Das Ergebnis dieser Anfrage soll in den Zwischenspeicher eingetragen werden.
- Es ist bereits eine Anfrage P im Zwischenspeicher vorhanden, die „ähnliche“ Daten abdeckt.

Wir gehen zuerst davon aus, dass Q eine Anfrage ist, die vollständig durch den Server beantwortet werden musste und keine Kompensationsanfrage. Folglich muss Q separat in den Index eingetragen werden. Wie bereits eingangs erwähnt, handelt es sich bei diesem Papier um erste Ideen. Daher wird vorerst angenommen, dass sich Q und P nur in den Selektionsprädikaten unterscheiden. Für P und Q gilt also:

- $Q = \{c_1 \wedge \cdots \wedge c_q \wedge r_1 \wedge \cdots \wedge r_m \wedge j_1 \wedge \cdots \wedge j_n \wedge s_1^1 \wedge \cdots \wedge s_o^1\}$ und
- $P = \{c_1 \wedge \cdots \wedge c_q \wedge r_1 \wedge \cdots \wedge r_m \wedge j_1 \wedge \cdots \wedge j_n \wedge s_1^2 \wedge \cdots \wedge s_q^2\}$.

Der Einfachheit halber werden im Folgenden nur noch die Selektionsprädikate betrachtet. Der Generalisierungalgorithmus optimiert zuerst die einzelnen Anfragen. Aufgrund der konjunktiven Verknüpfung der Prädikate wird somit erreicht, dass maximal zwei positive[2] Prädikate das gleiche Attribut der gleichen Relation verwenden. Beispielsweise wäre `t1.a>100 ∧ t1.a<150 ∧ t1.a<200` eine im Sinne des verwendeten Kalküls korrekte Anfrage, würde aber in diesem ersten Schritt auf `t1.a>100 ∧ t1.a<150` reduziert werden. Die somit entstandene Anfrage sei Q'.

Um darüber zu entscheiden, wie die Ergebnisse der Anfragen in Beziehung stehen, können wir nur die Prädikate auf Ihre Beziehung hin analysieren. Hierzu bezeichnen wir mit $Q_s(t1.a)$ und $P_s(t1.a)$ die Selektionsprädikate der Anfragen P und Q, die sich auf das Attribut $t1.a$ beziehen. Dabei beschreiben die konjunktiv verknüpften Prädikate jeweils

[2] $\theta \in \{\leq, <, =, \geq, >\}$

einen Wertebereich $range(Q_s(t1.a)) = [min(Q_s(t1.a)), max(Q_s(t1.a))] \subseteq dom(t1.a)$ mit einer auf $dom(t1.a)$ definierten Halbordnung.

Wir betrachten als Beziehungen zwischen von Prädikaten beschriebenen Wertebereichen die Adjazenz, die Inklusion mit dem Spezialfall der Identität, die Disjunktheit sowie die Überlappung. Betrachten wir dies zunächst für den Fall, dass ausschließlich Prädikate zu einem Attribut angegeben sind. Decken Selektionsprädikate einen gemeinsamen Wertebereich oder angrenzende Wertebereichebereiche ab, können sie generalisiert werden. Hierbei können aufgrund der Struktur des Caches nur die folgenden Fälle auftreten:

Adjazenz

Die Wertebereiche von $Q_s(t1.a)$ und $P_s(t1.a)$ grenzen aneinander an. Beispiel: $P = \{\cdots \wedge \texttt{t1.a=150}\}$; $Q' = \{\cdots \wedge \texttt{t1.a>150} \wedge \texttt{t1.a<200}\}$
Die Ergebnisse von P und Q' werden vereinigt, P wird entfernt und die generalisierte Anfrage (hier $G(P, Q') = \{\ldots \texttt{t1.a} \geq \texttt{150} \wedge \texttt{t1.a<200}\}$) wird neu eingetragen.

Disjunktheit

Die von $Q_s(t1.a)$ und $P_s(t1.a)$ beschriebenen Wertebereiche sind disjunkt und nicht angrenzend. Beispiel: $P = \{\cdots \wedge \texttt{t1.a=150}\}$; $Q' = \{\cdots \wedge \texttt{t1.a>170} \wedge \texttt{t1.a<200}\}$
Eine Generalisierung wäre zwar möglich, die zusätzlich erfassten unbekannten Daten (hier $150 \leq t1.a \leq 170$) müssten aber entsprechend markiert werden. Vorerst wird daher Q' zusätzlich in den Zwischenspeicher eingetragen.

War Q eine Kompensationsanfrage, d.h., sie wurde erzeugt, da die ursprüngliche Anfrage K nicht vollständig durch P lokal beantwortet werden konnte, muss untersucht werden, ob $P \cup Q'$ eingetragen und P entfernt wird. Laut Definition gilt für die Ergebnisse $r(Q')$, $r(K)$ und $r(P)$ der Anfragen Q', K und P: $r(P) \cup r(Q') = r(K)$. Daher kann nur der folgenden Fall auftreten:

Inklusion

Der von $P_s(t1.a)$ beschriebene Wertebereich ist komplett in $range(K_s(t1.a))$ enthalten. Beispiel: $P = \{\cdots \wedge \texttt{t1.a=150}\}$; $K = \{\cdots \wedge \texttt{t1.a>100} \wedge \texttt{t1.a<200}\}$
P kann also aus dem Zwischenspeicher entfernt werden und K wird neu eingefügt. (In diesem Fall wäre $Q' = \{\cdots \wedge \texttt{t1.a>100} \wedge \texttt{t1.a<200} \wedge \texttt{t1.a} \neq \texttt{150}\}$)

Aufgrund der in [HS03c] angegebenen Beschränkungen tritt der Fall, dass sich die Ergebnisse von P und Q' überlappen nicht ein. Dazu wäre es notwendig, Kombinationen aus Filter- und Kompensationsanfragen zuzulassen. Beispiel: $P = \{\cdots \wedge \texttt{t1.a>100}\}$; $K = \{\cdots \wedge \texttt{t1.a>90} \wedge \texttt{t1.a<200}\}$; $P' = filter(P, K) = \{\cdots \wedge \texttt{t1.a} \leq \texttt{200}\}$; $Q' = compensation(P, K) = \{\cdots \wedge \texttt{t1.a>90} \wedge \texttt{t1.a} \leq \texttt{100}\}$. Für zukünftige Arbeiten ist aber auch eine Unterstützung derartiger Kombinationen geplant. Für die Generalisierung beim Eintragen neuer Anfragen in den Zwischenspeicher bedeutet das:

Überlappung

Die Wertebereiche von $P_s(t1.a)$ und $K_s(t1.a)$ sind überlappend. Folglich kann analog zur Adjazenz die Vereinigung von K und P in den Zwischenspeicher aufgenommen und P entfernt werden.

Die bisherigen Betrachtungen bezogen sich jeweils auf von einzelnen Prädikaten beschriebene Wertebereiche. Der folgende Algorithmus gibt an, wie davon ausgehend die Generaliserung zweier Anfragen entsteht.

Algorithmus 1: *Generalisierung von zwei Anfragen P und Q*

Eingabe: Anfragen P und Q
Ausgabe: $G(P, Q)$

$attributes :=$ in P und Q in Selektionsprädikaten verwendete Attribute
$G_s(P, Q) := \emptyset$ // Selektionsprädikate der generalisierten Anfrage
for $a \in attributes$
    if $range(P_s(a))$ und $range(Q_s(a))$ sind adjazent
        $G_s(P, Q) := G_s(P, Q) \cup \{$Prädikate der gemeinsamen Bereichsgrenzen$\}$
    else if $range(P_s(a))$ beinhaltet $range(Q_s(a))$
        $G_s(P, Q) := G_s(P, Q) \cup P_s(a)$
    else if $range(Q_s(a))$ beinhaltet $range(P_s(a))$
        $G_s(P, Q) := G_s(P, Q) \cup Q_s(a)$
    else if $range(Q_s(a))$ und $range(Q_s(a))$ sind überlappend
        $G_s(P, Q) := G_s(P, Q) \cup \{$Prädikate der gemeinsamen Bereichsgrenzen$\}$
    else if $range(Q_s(a))$ und $range(Q_s(a))$ sind disjunkt
        $G_s(P, Q) := \emptyset$
        break
if $G_s(P, Q) \neq \emptyset$
    erzeuge $G(P, Q)$ aus $G_s(P, Q)$

---

5 Zusammenfassung und Ausblick

In diesem Papier wurden erste Ideen präsentiert, wie beim semantischen Cachen auf mobilen Endgeräten die dort vorherrschenden Beschränkungen (insbesondere die Speicherbeschränkungen) besser reflektiert werden können. Dazu wird versucht, neue zwischenzuspeichernde Anfragen und bereits zwischengespeicherte Anfragen zu generalisieren und die generellere Anfrage statt der beiden spezielleren zu cachen.

Das Papier stellt den Ausgangspunkt für zahlreiche Folgearbeiten dar. Zum einen ist geplant, die generellen Einschränkungen des in [HS03c] vorgestellten Caching-Ansatzes zu

reduzieren was direkten Enfluss auf die Möglichkeiten der Generalisierung hat. Zum anderen müssen die in der hier vorliegenden Arbeit getroffenen Einschränkungen (insbesondere die ausschließliche Betrachtung der Selektionsprädikate) aufgeweicht werden. Daneben müssen die Fragen:

- *„Wann macht ist es überhaupt sinnvoll, Daten auf einem Mobilgerät zwischenzuspeichern?“* und
- *„Wie kann sichergestellt werden, dass der Cache korrekte Daten enthält?“*

untersucht werden. Wie bereits in Abschnitt 2 diskutiert wurde, muss untersucht werden, inwieweit Spezialisierung und Generalisierung kombiniert werden können, um eine optimale Speichereffizienz zu gewährleisten.

Literaturverzeichnis

[DFJ+96] Shaul Dar, Michael J. Franklin, Björn Þór Jónsson, Divesh Srivastava, and Michael Tan. Semantic Data Caching and Replacement. In *VLDB'96, Proceedings of the 22th International Conference on Very Large Data Bases*, pages 330–341, Mumbai (Bombay), India, September 3–6 1996. Morgan Kaufmann.

[Fre59] Edward Fredkin. *Trie memory*. Information Memorandum, Bolt Beranek and NewMan Inc., Cambridge, MA, 1959.

[GG97] Parke Godfrey and Jarek Gryz. Semantic Query Caching for Heterogeneous Databases. In *Intelligent Access to Heterogeneous Information, Proceedings of the 4th Workshop KRDB-97, Athens, Greece*, volume 8 of *CEUR Workshop Proceedings*, pages 6.1–6.6, August 30 1997.

[GG99] Parke Godfrey and Jarek Gryz. Answering Queries by Semantic Caches. In *Database and Expert Systems Applications, 10th International Conference, DEXA '99, Florence, Italy, Proceedings*, volume 1677 of *LNCS*, pages 485 – 498. Springer, August 30 - September 3 1999.

[HS03a] Hagen Höpfner and Kai-Uwe Sattler. Semantic Replication in Mobile Federated Information Systems. In Anne James, Stefan Conrad, and Wilhelm Hasselbring, editors, *Proceedings of the Fifth International Workshop on Engineering Federated Information Systems (EFIS), Coventry, UK 17th - 18th July, 2003*, pages 36–41. Akademische Verlagsgesellschaft Aka GmbH, Berlin, July 2003.

[HS03b] Hagen Höpfner and Kai-Uwe Sattler. SMoS: A Scalable Mobility Server. In Anne James and Muhammad Younas, editors, *Poster Proceedings of the Twentieth British National Conference on Databases (BNCOD20), Coventry, UK 15th - 17th July, 2003*, pages 49–52. School of Mathematical and Informational Sciences; Coventry University, July 2003.

[HS03c] Hagen Höpfner and Kai-Uwe Sattler. Towards Trie-Based Query Caching in Mobile DBS. In B. König-Ries, M. Klein, and P. Obreiter, editors, *Post-Proceedings of the Workshop Scalability, Persistence, Transactions - Database Mechanisms for Mobile Applications*, Lecture Notes in Informatics (LNI), pages 106–121, 2003.

[KB96] Arthur M. Keller and Julie Basu. A Predicate-based Caching Scheme for Client-Server Database Architectures. *VLDB Journal: Very Large Data Bases*, 5(1):35–47, January 1996.

[KSGH03] Marcel Karnstedt, Kai-Uwe Sattler, Ingolf Geist, and Hagen Höpfner. Semantic Caching in Ontology-based Mediator Systems. In Robert Tolksdorf and Rainer Eckstein, editors, *Proceedings of Berliner XML Tage 2003, Berlin, 13.-15. Oktober 2003, 3rd International Workshop of the GI Working Group „Web und Datenbanken“*, pages 155–169, October 2003.

[LC98] Dongwon Lee and Wesley W. Chu. Conjunctive Point Predicate-based Semantic Caching for Wrappers in Web Databases. In *CIKM'98 Workshop on Web Information and Data Management (WIDM'98), Washington, DC, USA*, November 6 1998.

[LC99] Dongwon Lee and Wesley W. Chu. Semantic Caching via Query Matching for Web Sources. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA*, pages 77–85. ACM, November 2–6 1999.

[LC01] Dongwon Lee and Wesley W. Chu. Towards Intelligent Semantic Caching for Web Sources. *Journal of Intelligent Information Systems*, 17(1):23–45, November 2001.

[LLS99] Ken. C. K. Lee, H. V. Leong, and Antonio Si. Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(2):28–36, 1999.

[RD99] Qun Ren and Margaret H. Dunham. Using Clustering for Effective Management of a Semantic Cache in Mobile Computing. In *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access, August 20, 1999, Seattle, WA, USA*, pages 94–101. ACM, 1999.

[RD00] Qun Ren and Margaret H. Dunham. Using Semantic Caching to Manage Location Dependent Data in Mobile Computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM-00)*, pages 210–242, New York, August 6–11 2000. ACM.