Managing Periodically Updated Data in Relational Databases: A Stochastic Modeling Approach

Avigdor Gal[*]   Jonathan Eckstein[†]

Abstract

Recent trends in information management involve the periodic transcription of data onto secondary devices in a networked environment, and the proper scheduling of these transcriptions is critical for efficient data management. To assist in the scheduling process, we are interested in modeling *data obsolescence*, that is, the reduction of consistency over time between a relation and its replica. The modeling is based on techniques from the field of stochastic processes, and provides several stochastic models for content evolution in the base relations of a database, taking referential integrity constraints into account. These models are general enough to accommodate most of the common scenarios in databases, including batch insertions and life spans both with and without memory. As an initial "proof of concept" of the applicability of our approach, we validate the insertion portion of our model framework via experiments with real data feeds. We also discuss a set of transcription protocols which make use of the proposed stochastic model.

1   Introduction and motivation

Recent developments in information management involve the transcription of data onto secondary devices in a networked environment, *e.g.*, materialized views in data warehouses and search engines, and replicas in pervasive systems. Data transcription influences the way databases define and maintain consistency. In particular, the networked environment may require periodic (rather than continuous) synchronization between the database and secondary copies, either due to paucity of resources (*e.g.*, low bandwidth or limited night windows) or to the transient characteristics of the connection. Hence, the consistency of the information in secondary copies, with respect to the transcription origin, varies over time and depends on the rate of change of the base data and on the frequency of synchronization.

Systematic approaches to the proper scheduling of transcriptions necessarily involve optimizing a trade-off between the cost of transcribing fresh information versus the cost of using obsolescent data. To do so, one must quantify, at least in probabilistic terms, this latter cost, which we call *obsolescence cost* [11]. This paper aims to provide a comprehensive stochastic framework for quantifying time-dependent data obsolescence in replicas. Suppose we are given a relation R, a start time $s \in \Re$, and some later time $f > s$. We denote the extension of a relation R at time $t \in \Re$ by $R(t)$. Starting from a known extension $R(s)$, we are interested in making probabilistic predictions about the contents of the later extension $R(f)$. We also suggest a cost model schema to quantify the difference between $R(s)$ and $R(f)$. Such tools assist in optimizing the synchronization process, as demonstrated in this paper. Our approach is based on techniques from

---

[*]Department of Management Science and Information Systems, Rutgers University, Piscataway, NJ 08854 USA, phone: (732) 445-3245, fax: (732) 445-6329, e-mail: `avigal@rci.rutgers.edu`

[†]Department of Management Science and Information Systems and RUTCOR, Rutgers University, Piscataway, NJ 08854 USA, phone: (732) 445-0510, fax: (732) 445-6329, e-mail: `jeckstei@rutcor.rutgers.edu`

the field of stochastic processes, and provides several stochastic models for content evolution in a relational database, taking referential integrity constraints into account. In particular, we make use of compound nonhomogeneous Poisson models and Markov chains; see for example [28, 29, 35]. We use Poisson processes to model the behavior of tuples entering and departing relations, allowing (nonhomogeneous) time-varying behavior — *e.g.*, more intensive activity during work hours, and less intensive activity after hours and on weekends — as well as compound (bulk) insertions, that is, the simultaneous arrival of several tuples. We use Markov chains in a general modeling approach for attribute modifications, allowing the assignment of a new value to an attribute in a tuple to depend on its current value. The approach is general enough to accommodate most of the common scenarios in databases, including batch insertions and memoryless, as well as time dependent, life spans.

As motivation, consider the following two examples:

Example 1 (Query optimization) *Query optimization relies heavily on estimating the cardinality and value distribution of relations in a database. If these statistics are outdated and inaccurate, even the best query optimizer may formulate poor execution plans. Typically, statistics are updated only periodically, usually at the discretion of the database administrator, using utilities such as DB2's RUNSTATS. Although some research has been devoted to speeding up statistics collection through sampling and wavelet approximations [14, 22], periodic updates are unavoidable in very large databases such as IBM's Net.Commerce [32], an e-business software package with roughly one hundred relations, or an SAP application, which has more than 8,000 relations and 9,000 indices. Collection of statistics becomes an even more acute problem in database federations [31], where the federation members do not always "volunteer" their statistics [27] (or their cost models for that matter [30]), and are unwilling to burden their resources with frequent statistics collection.*

In current practice, cardinality or histogram data recorded at time s are used unchanged until the next full analysis of the database at some later time $s' > s$. If a query optimization must be performed at some time $f \in (s, s')$, the optimizer simply uses the statistics gathered at time s, since the time spent recomputing them may overwhelm any benefits of the query optimization. As an alternative, we suggest using a probabilistic estimate of the necessary statistics at time f. Use of these techniques might make it possible to increase the interval between statistics-gathering scans, as will be discussed in Example 7. □

Example 2 (Replication management in distributed databases) *We now consider replication management in a distributed database. Since fully synchronous replication management, in which a user is guaranteed access to the most current data, comes at a significant computational cost, most commercial distributed database providers have adopted asynchronous replication management. That is, updates to relation replicas are performed after the original transaction has committed, in accordance with the workload of the machine on which the secondary copy is stored. Asynchronous replicas are also very common in Web applications such as search engines, where Web crawlers sample Web sites periodically, and in pervasive systems (e.g., Microsoft's Mobile Information Server[1] and Café Central[2]). In a pervasive system, a server serves many different users, each with her own unpredictable connectivity schedule and dynamically changing device capabilities. Our modeling techniques would allow client devices to reduce the rate at which they poll the server, saving both server resources and network bandwidth. We demonstrate the usefulness of stochastic modeling in this setting in Sections 3.1 and 5.6.* □

---

[1]http://www.microsoft.com/servers/miserver/
[2]http://www.comalex.com/central.htm

The novelty of this paper is in developing a formal framework for modeling content evolution in relational databases. The problem of content evolution with respect to materialized views (which may be regarded as a complex form of data transcription) in databases has already been recognized. For example, in [1], the incompleteness of data in views was noted as being a "dynamic notion since data may be constantly added/removed from the view." Yet, we believe that there has been no prior formal modeling of the evolution process.[3] Related research involves the containment property of a materialized view with respect to its base data: a few of the many references in this area include [38, 6, 19, 2, 13]. However, the temporal aspects of content evolution have not been systematically addressed in this work. In [2], for example, the containment relationships between a materialized view I and the "true" query result $\mathcal{V}(D)$, taken from a database D, can be either $I = \mathcal{V}(D)$ or $I \subseteq \mathcal{V}(D)$. The latter relationship represents a situation where the materialized view stores only a partial subset of the query result. However, taking content evolution into account, it is also possible that $I \supset \mathcal{V}(D)$, if tuples may be deleted from $\mathcal{V}(D)$ and I is periodically updated. Moreover, modifications to the base data may result in both $I \nsubseteq \mathcal{V}(D)$ and $I \nsupseteq \mathcal{V}(D)$.

Refresh policies for materialized views have been previously discussed in the literature (*e.g.*, [20] and [8]). Typically, materialized views are refreshed immediately upon updates to the base data, at query time (as in [8]), or using snapshot databases (as in [20]). The latter approach can produce obsolescent materialized views. A combination of all three approaches appears in [9]. Our methodology differs in that we do not assume an *a priori* association of a materialized view with a refresh policy, but instead design policies based on their transcription and obsolescence costs.

A preliminary attempt to describe the time dependency of updates in the context of Web management was given in [7], which suggests a simple homogeneous Poisson process to model the updating of Web pages. We suggest instead a nonhomogeneous compound Poisson model, which is far more flexible, and yet still tractable. In addition, the work in [7] supposes that transcriptions are performed at uniform time intervals, mainly because "crawlers cannot guess the best time to visit each site." We show in this paper that our model of content evolution gives rise to other, better transcription policies.

In [25], a trade-off mechanism was suggested to decide between the use of a cache or recomputation from base data by using range data, computed at the source. In this framework, an update is "pushed" to a replication site whenever updated data falls outside a predetermined interval, or whenever a query requires current data. The former requires the client and the server to be in touch continuously, in case the server needs to track down the client, which is not always realistic (either because the server does not provide such services, or because the overhead for such services undermines the cost-effectiveness of the client). The latter requirement puts the burden of deciding whether to refresh the data on the client, without providing it with any model for the evolution of the base data. We attempt to fill this gap by providing a stochastic model for content evolution, which allows a client to make judicious requests for current data. Other work in related areas (*e.g.*, [3, 5, 10]) has considered various alternatives for pushing updated data from a server to a cache on the client side. Lazy replica-update policies using replication graphs have also been discussed in, for example, [4]. This work, however, does not take the data obsolescence into account, and is primarily concerned with transaction throughput and timely updates, subject to network constraints.

As with models in general, our model is an idealized representation of a process. To be useful, we wish to make predictions based on tractable analytical calculations, rather than detailed, computationally intensive simulations. Therefore, we restrict our modeling to some of the more basic tools of applied probability theory, specifically those relating to Poisson processes and Markov

---

[3]Other research efforts involve probabilistic database systems (*e.g.*, [18]), but this work is concerned with uncertainty in the stored data, rather than data evolution.

chains. Texts such as [28, 35] contain the necessary reference material on Markov chains and Poisson processes, and specifically on nonhomogeneous Poisson processes. Poisson processes can model a world where data updates are independent from one another. In databases with widely distributed access, *e.g.*, Web interfacing databases, such an independence assumption seems plausible, as was verified in [7].

The rest of the paper is organized as follows: Section 1.1 introduces some basic notation. Section 2 provides a content evolution model for insertions and deletions, while Section 3 discusses data modifications. We shall introduce preliminary results of fitting the insertion model parameters to real data feeds in Section 4. A cost model and transcription policies that utilize it follow in Section 5, highlighting the practical impact of the model. Conclusions and topics for further research are provided in Section 6.

1.1 Notational preliminaries

In what follows, we denote the set of attributes and relations in the database by $\mathcal{B}$ and $\mathcal{R}$, respectively. Each $R \in \mathcal{R}$ consists of a set of attributes $\mathcal{A}(R) \subseteq \mathcal{B}$, and also has a *primary key* $\mathcal{K}(R)$, which is a nonempty subset of $\mathcal{A}(R)$. Each attribute $A \in \mathcal{B}$ has a *domain* $\operatorname{dom} A$, which we assume to be a finite set, and for any subset of attributes $\mathcal{A} = \{A_1, A_2, ..., A_k\}$, we let $\operatorname{dom} \mathcal{A} = \operatorname{dom} A_1 \times \operatorname{dom} A_2 \times ... \times \operatorname{dom} A_k$ denote the compound domain of $\mathcal{A}$. We denote by $r.A(t)$ the value of attribute A in tuple r at time t, and similarly use $r.\mathcal{A}(t)$ for the value of a compound attribute. For a given time t, subset of attributes $\mathcal{A} \subseteq \mathcal{A}(R)$, and value $v = \langle v_1, v_2, ..., v_k \rangle \in \operatorname{dom} \mathcal{A}$, we define $R_{\mathcal{A},v}(t) = \{r \in R(t) \mid (r.A_1(t) = v_1) \wedge (r.A_2(t) = v_2) \wedge \ldots \wedge (r.A_k(t) = v_k)\}$. We also define $\hat{R}_{\mathcal{A}}(t)$ to be the *histogram* of values of $\mathcal{A}$ at time t, that is, for each value $v \in \operatorname{dom} \mathcal{A}$, $\hat{R}_{\mathcal{A}}(t)$ associates a nonnegative integer $\hat{R}_{\mathcal{A},v}(t)$, which is the cardinality of $R_{\mathcal{A},v}(t)$.[4] This notation, and well as other symbols used throughout the paper, are also summarized in Table 1.

2 Modeling insertions and deletions

This section introduces the stochastic models for insertions and deletions. Section 2.1 discusses insertions, while deletions are discussed in section 2.2. Section 2.3 combines the effect of insertions and deletions on a relation's cardinality. We conclude with a discussion of non-exponential life spans in Section 2.4. We defer discussing model validation until Section 4.

2.1 Insertion

We use a nonhomogeneous Poisson process [28, 35] with instantaneous arrival rate $\lambda_R : \Re \to [0, \infty)$ to model the occurrence of *insertion events* into R. That is, the number of insertion events occurring in any interval $(s, f]$ is a Poisson random variable with expected value $\Lambda_R(s, f) = \int_s^f \lambda_R(t) \, dt$. A homogeneous Poisson process may be considered as the special case where $\lambda_R(t)$ is equal to a constant $\lambda_R > 0$ for all t, yielding $\Lambda_R(s, f) = \int_s^f \lambda_R(t) \, dt = \int_s^f \lambda_R \, dt = \lambda_R \cdot (f - s)$.

We now consider the interarrival time distribution of the nonhomogeneous Poisson process. We first define the nonhomogeneous exponential distribution, as follows:

---

[4]This vector can be computed exactly and efficiently using indices. Alternatively, in the absence of an index for a given attribute, statistical methods (such as "probabilistic" counting [37], sampling-based estimators [14], and wavelets [22]) can be applied.

s, f	Points in time
$R, S \in \mathcal{R}; R(t); \|R(s)\|$	Relations; R's extension at time t; its cardinality at time s.
$A \in \mathcal{B}; \mathcal{A} \subseteq \mathcal{B}; \mathrm{dom}\, A; \mathrm{dom}\, \mathcal{A}$	Attribute; compound attribute; domain of attribute; domain of compound attribute
$\mathcal{A}(R) \subseteq \mathcal{B}; \mathcal{K}(R); \mathcal{C}(R)$	Attributes of R; primary key of R; modifiable attributes of R
$r; r.A(t); r.\mathcal{A}(t)$	Tuple; value of attribute A in r at t; value of compound attribute $\mathcal{A}$ in r at t
$v \in \mathrm{dom}\, \mathcal{A}; R_{\mathcal{A},v}(t); \hat{R}_{\mathcal{A}}(t)$	Value; set of tuples with $r.A(t) = v$; histogram of $\mathcal{A}$
$b(r); d(r)$	Insertion time of r; deletion time of r
$\mathcal{N} \subset \mathcal{B}$	Set of numeric attributes
$G; G(R)$	Dependency multigraph; dependency sub-multigraph generated by R
$\lambda_R(t); \Lambda_R(s,f); B_R(s,f)$	Insertion rate (intensity); expected number of insertion events during $(s,f]$; number of insertions during $(s,f]$
$\mathrm{Exp}_s(\phi(\cdot)); L_{R,s}; L_{R,s}^{\mathrm{I}}$	Nonhomogeneous exponential distribution; interarrival time; remaining life span
$\Delta_{R,i}^+; \Delta_i^-$	Number of tuples for insertion event i; number of tuples for deletion event i
$\mu_R(t); M_R(s,f)$	Deletion rate (intensity); expected number of deletion events
$w(r, S)$	Number of tuples in S forcing deletion of r via referential integrity
$W(R, S, t)$	Random variable of $w(r,S)$ over uniform selection of $r \in R$
$W(R, t)$	Vector of $W(R,S,t)$ over $S \in G(R)$
$p_R(s,f)$	Probability that a tuple in R at time s survives through f
$\hat{p}_R(t,f)$	Survival probability through f for tuple inserted at t
$\mathrm{E}_{r \in R(s)}[\cdot]$	Expectation over uniform random selection of tuples $r \in R(s)$
$X_R(s,f)$	Number of tuples inserted into R during $(s,f]$
$Y_R(s,f)$	Number of tuples in $R(s)$ surviving through f
$Y_R^+(s,f); Y_R^-(s,f)$	Surviving tuples that were modified; surviving tuples that were not modified
$\tau_{v,s}^{R,A}; \gamma_{R,A}(t); \Gamma_{R,A}(s,f)$	Remaining time to next modification ; modification rate; expected number of modification events
$\ell_v^{R,A}$	Relative exit rate
$P_{u,v}^{R,A}(s,f); q_{u,v}^{R,A}$	Transition probability; relative transition rate
$\Delta A; \delta; \sigma^2$	Change to a value of A in a random-walk update event; expected value of change; variance of change
$C_{R,\mathrm{u}}(s,f); C_{R,\mathrm{o}}(s,f); C_R(t);$	Transription cost; obsolescence cost; total cost
$\iota_{r,A}(s,f); \iota_{R,A}(s,f); \iota_r(s,f)$	Contribution to obsolescence of: r via A; A; r
$\hat{\iota}_{R,A}^{\mathrm{M}}(s,f); \hat{\iota}_R^{\mathrm{D}}(s,f); \hat{\iota}_R^{\mathrm{I}}(s,f)$	Expected obsolescence cost due to: modification; deletion; insertion
$\hat{\iota}_{R,A,u}^{\mathrm{M}}(s,f)$	Expected obsolescence cost due to modification to the value u
$c_{u,v}^{R,A}$	Elements of a cost matrix

Table 1: List of Symbols.

Definition 1 (Nonhomogeneous exponential distribution) *Let* $\phi : \Re \to [0, \infty)$ *be a integrable function. Given some* $s \in \Re$*, a random variable* V *is said to have a* nonhomogeneous exponential *distribution (denoted by* $V \sim \mathrm{Exp}_s(\phi(\cdot))$*) if* V*'s density function is*

$$
p(\tau) = \begin{cases} \phi(s + \tau) \exp\left(-\int_0^\tau \phi(s + u)\, du \right), & \tau \geq 0 \\ 0, & \tau < 0. \end{cases}
$$

It is worth noting that if $\phi(t)$ is constant, $p(\tau)$ is just a standard exponential distribution. We shall now show that, as with homogeneous Poisson processes, the interarrival time of insertion events is distributed like an exponential random variable, $L_{R,s}$, but with a time-varying density function.

Lemma 1 *At any time* s*, the amount of time* $L_{R,s}$ *to the next insertion event is distributed like* $\mathrm{Exp}_s(\lambda_R(\cdot))$*. The probability of an insertion event occurring during* $(s,f]$ *is* $\mathrm{P}\{L_{R,s} < f - s\} = 1 - e^{-\Lambda_R(s,f)}$*.*

Proof. Let $\{N(t), t \geq 0\}$ be a nonhomogeneous Poisson process with intensity function $\lambda_R(t)$, which implies $P\{N(f) - N(s) = 0\} = e^{-\Lambda_R(s,f)}$. Now, the chance that no new tuple was inserted during $(s, f]$ is the same as the chance that the process $N(\cdot)$ has no arrivals during $(s, f]$, that is, $e^{-\Lambda_R(s,f)}$. The chance that a new tuple was inserted during $(s, f]$ is just the complement of the chance of no arrivals, namely,

$$P\{L_{R,s} < f - s\} = \{N(f) - N(s) \geq 1\} = 1 - P\{N(f) - N(s) = 0\} = 1 - e^{-\Lambda_R(s,f)}.$$

Taking the derivative of this expression with respect to f and making a change of variables, the probability density of the time until the next insertion from time s is $p(\tau) = \lambda_R(s + \tau)e^{-\Lambda_R(s,s+\tau)}$. Thus, $L_{R,s} \sim \mathrm{Exp}_s(\lambda_R(\cdot))$. ∎

At insertion event i, a random number of tuples $\Delta_{R,i}^+$ are inserted, allowing us to model bulk insertions. A *bulk insertion* is the simultaneous arrival of multiple tuples, and may occur because the tuples are related, or because of limitations in the implementation of the server. For example, e-mail servers may process an input stream periodically, resulting in bulk updates of a mailbox. Assuming that the $\{\Delta_{R,i}^+\}$ are independent and identically distributed (IID), then the stochastic process $\{B_R(t), t \geq 0\}$ representing the cumulative number of insertions through time t is a *compound Poisson* process (*e.g.*, [29], pp. 87-88). We let $B_R(s, f)$ denote the number of insertions falling into the interval $(s, f]$. The expected number of inserted tuples during $(s, f]$ may be computed via $\mathrm{E}[B_R(s, f)] = \int_s^f \lambda_R(t) \, \mathrm{E}[\Delta_R^+] \, dt = \mathrm{E}[\Delta_R^+] \int_s^f \lambda_R(t)dt = \mathrm{E}[\Delta_R^+] \Lambda_R(s, f)$. Here, Δ_R^+ represents a generic random variable distributed like the $\{\Delta_{R,i}^+\}$.

We now consider three simple cases of this model:

General nonhomogeneous Poisson process: Assume that $\mathrm{E}[\Delta_R^+] = 1$. The expected number of insertions simplifies to $\mathrm{E}[B_R(s, f)] = \mathrm{E}[\Delta_R^+] \Lambda_R(s, f) = 1 \cdot \Lambda_R(s, f) = \Lambda_R(s, f)$.

Homogeneous Poisson process: Assume once more that $\mathrm{E}[\Delta_R^+] = 1$. Assume further that $\lambda_R(t)$ is a constant function, that is, $\lambda_R(t) = \lambda_R$ for all times t. In this case, as shown above, $\Lambda_R(s, f)$ takes on the simple form of $\lambda_R \cdot (f - s)$. Thus, $\mathrm{E}[B_R(s, f)] = \Lambda_R(s, f) = \lambda_R \cdot (f - s)$. The interarrival times are distributed as $\mathrm{Exp}(\lambda_R)$, the exponential distribution with parameter λ_R.

Recurrent piecewise-constant Poisson process: A simple kind of nonhomogeneous Poisson process can be built out of homogeneous Poisson processes that repeat in a cyclic pattern. Given some length of time T, such as one day or one week, suppose that the arrival rate function $\lambda_R(t)$ of the recurrent Poisson process repeats every T time units, that is, $\lambda_R(t) = \lambda_R(t - T\lfloor t/T \rfloor)$ for all t. Furthermore, the interval $[0, T)$ is partitioned into a finite number of subsets $J_1, \ldots, J_K$, with $\lambda_R(t)$ constant throughout each J_k, $k = 1, \ldots, K$. Finally, each J_k is in turn composed of a finite number of half-open intervals of the form $[s, f)$. For instance, T might be one day, with $K = 24$ and $J_1 = [0\!:\!00, 1\!:\!00), J_2 = [1\!:\!00, 2\!:\!00), \ldots, J_{24} = [23\!:\!00, 0\!:\!00)$. As another simple example, T might be one week, and $K = 2$. The subset J_1 would consist of a firm's normal hours of operation, say $[9\!:\!00, 18\!:\!00)$ for each weekday, and $J_2 = [0, T)\backslash J_1$ would denote all "off-hour" times. Formalisms like those of [24] could also be used to describe such processes in a more structured way. We term this class of Poisson processes to be *recurrent piecewise-constant* — abbreviated *RPC*.

It is worth noting that, in client-server environments, the insertion model should typically be formed from the client's point of view. Therefore, if the server keeps a database from which many clients transcribe data, the modeling of insertions for a given client should only include the part of the database the client actually transcribes. Therefore, if a "road warrior" is interested only in

6

new orders for the 08904 zip code area, the insertion model for that client should concentrate on that zip code, ignoring the arrival orders from other areas.

2.1.1   The complexity of computing $\Lambda_R(s, f)$

$\Lambda_R(s, f)$, the Poisson expected value, is computed by integrating the model parameter $\lambda_R(t)$ over the interval $[s, f]$. Standard numerical methods allow rapid approximation of this definite integral even if no closed formula is known for the indefinite integral. However, the complexity of this calculation depends on the information-theoretic properties of $\lambda_R(t)$ [36, Section 1].

For our purposes, however, simple models of $\lambda_R(t)$ are likely to suffice. For example, if $\lambda_R(t)$ is a polynomial of degree $d \geq 0$, the integration can be performed in $O(d + 1)$ time. Consider next a piecewise-polynomial Poisson process: the time line is divided into intervals such that, in each time interval, $\lambda_R(t)$ can be written as a polynomial. The complexity of calculating $\Lambda_R(s, f)$ in this case is $O(n(d + 1))$, where n is the number of segments in the time interval $(s, f]$, and d is the highest degree of the n polynomials.

Further suppose that the piecewise-polynomial process is recurrent in a similar manner to the RPC process, that is, given some fixed time interval T, $\lambda_R(t) = \lambda_R(t - T \lfloor t/T \rfloor)$ for all t. Note that the RPC Poisson process is the special case of this model in which $d = 0$. If there are c segments in the interval $[0, T]$, then the complexity of calculating $\Lambda_R(s, f)$ becomes $O(c(d + 1))$, regardless of the length of the interval $[s, f]$. This reduction occurs because, for all intervals of the form $[kT, (k+1)T] \subseteq [s, f]$ for which k is an integer, the integral $\int_{kT}^{(k+1)T} \lambda_R(t)\, dt$ is equal to $\int_0^T \lambda_R(t)\, dt$, which only needs to be calculated once.

In Section 4, we demonstrate the usefulness of the RPC model for one specific application. We hypothesize that a recurrent piecewise-polynomial process of modest degree (for example, $d=3$) will be sufficient to model most systems we are likely to encounter, and so the complexity of computing $\Lambda_R(s, f)$ should be very manageable.

2.2   Deletion

We allow for two distinct deletion mechanisms. First, we assume individual tuples have their own intrinsic stochastic life spans. Second, we assume that tuples are deleted to satisfy referential integrity constraints when tuples in other relations are deleted. These two mechanisms are combined in a tuple's overall probability of being deleted. Let R and S be two relations such that $\mathcal{K}(S)$ is a foreign key of S in R. We refer to S as a *primary relation* of R. Consider the directed multigraph G whose vertices consist of all relations R in the database, and whose edges are of the form $\langle R, S \rangle$, where S is a primary relation of R. The number of edges $\langle R, S \rangle$ is the number of foreign keys of S in R for which integrity constraints are enforced. We assume that $G(R)$ has no directed cycles. Let $G(R)$ denote the subgraph of G consisting of R and all directed paths starting at R. We denote the vertices of this subgraph by $S(R)$.

Example 3 (Referential integrity constraints in Net.Commerce) *IBM's Net.Commerce is supported by a DB2 database with about a hundred relations interrelated through foreign keys. For demonstration purposes, consider a sample of seven relations in the Net.Commerce database. Figure 1 is a pictorial representation of the multigraph G of these seven relations. The* **MERCHANT** *relation provides data about merchant profiles, the* **SCALE** *and* **DISCCALC** *relations are for computing price discounts, the* **CATEGORY** *and* **CGRYREL** *relations assist in categorizing products, and the* **ORDERS** *and* **SHIPTO** *relations contain information about orders. The six relations,* **SCALE, DISCCALC, CATEGORY, CGRYREL, ORDERS,** *and* **SHIPTO** *have a foreign key to the* **MERCHANT** *relation, through*
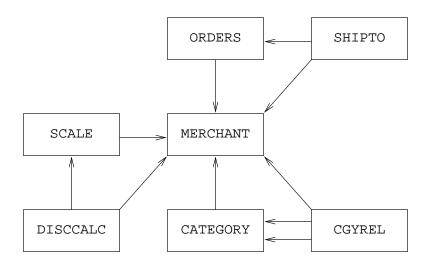
Figure 1: A partial multigraph of the case study.

MERCHANT's primary key (MERFNBR). Integrity constraints are enforced between the SCALE relation and the MERCHANT relation, as long as SCALE.SCLMENBR (the foreign key to MERCHANT.MERFNBR) does not have the value NULL. That is, unless a NULL value is assigned to the MERCHANT.MERFNBR attribute, a deletion of a tuple in MERCHANT results in a deletion of all tuples in SCALE such that SCALE.SCLMENBR = MERCHANT.MERFNBR. DISCCALC has a foreign key to the SCALE relation, through SCALE's primary key (SCLRFNBR). There are two attributes of CGRYREL that serve as foreign keys to the CATEGORY relation, through CATEGORY's primary key (CGRFNBR). Finally, SHIPTO contains shipment information of each product in an order, and therefore it has a foreign key to ORDERS through its primary key (ORFNBR). □

   With regard to intrinsic deletions within a relation, we assume that each tuple $r \in R(s)$ has a stochastic remaining life span $L_{R,s}^{\mathrm{I}}$. This random variable is identically distributed for each $r \in R(s)$, and is independent of the remaining life span of any other tuple and of r's age at time s (see Section 2.4 for a discussion of tuples with a non-memoryless life span). Specifically, we will assume that the chance of $r \in R(t)$ being deleted in the time interval $[t, t+\Delta t]$ approaches $\mu_R(t)\Delta t$ as $\Delta t \to 0$, for some function $\mu_R : \Re \to [0, \infty)$. We define $M_R(s,f) = \int_s^f \mu_R(t)\, dt$.

Lemma 2 $L_{R,s}^{\mathrm{I}} \sim \mathrm{Exp}_s(\mu_R(\cdot))$. *The probability that a tuple $r \in R(s)$ is deleted by time f, given that no corresponding tuple in $S(R)\backslash\{R\}$ is deleted, is $\mathrm{P}\{L_{R,s}^{\mathrm{I}} < f - s\} = 1 - e^{-M_R(s,f)}$.*

Proof. Let $r \in R(s)$ be a randomly chosen tuple, and assume that no corresponding tuple to r in $S(R)\backslash\{R\}$ is deleted. The proof is identical to that of Lemma 1, replacing $\lambda_R(t)$ with $\mu_R(t)$ and $\Lambda_R(s,f)$ by $M_R(s,f)$. ∎

2.2.1 Deletion and referential integrity

For any $r \in R(s)$ and any relation $S \in S(R)$, we define $w(r, S)$ to be the number of tuples in S whose deletion would force deletion of r in order to maintain referential integrity. This value can be between 0 and the number of paths from R to S in $G(R)$. For example, if $r \in R = $ CGRYREL of Figure 1, then $0 \le w(r, \mathtt{CATEGORY}) \le 2$ and $0 \le w(r, \mathtt{MERCHANT}) \le 3$. For completeness, we define

$w(r, R) = 1$. Each tuple in S has an independent remaining lifetime distributed as $\mathrm{Exp}_s(\mu_S(\cdot))$, and if any of the $w(r, S)$ tuples corresponding to r is deleted, then r must be immediately deleted, to maintain referential integrity constraints. We use $p_R(s, f)$ to denote the probability that a randomly chosen tuple in $R(s)$ survives until time f.

Lemma 3 $p_R(s, f) = \mathrm{E}_{r \in R(s)}\Big[\exp\Big(-\sum_{S \in S(R)} w(r, S) M_S(s, f)\Big)\Big]$, where $\mathrm{E}_{r \in R(s)}[\cdot]$ denotes expectation over random selection of tuples in $R(s)$.

Proof. Considering all $S \in S(R)$, and using the well-known fact that if $L_i \sim \mathrm{Exp}_s(\mu_i(\cdot))$ for $i = 1, \ldots, k$ are independent, then

$$\min\{L_1, \ldots, L_k\} \sim \mathrm{Exp}_s\left(\sum_{i=0}^{k} \mu_i(\cdot)\right),\tag{1}$$

we conclude that the remaining lifetime of r (denoted $L_{R,s}$) has a nonhomogeneous exponential distribution with intensity function $\sum_{S \in S(R)} w(r, S) \mu_S(\cdot)$. The probability of a given tuple $r \in R(s)$ surviving through time f is thus

$$\exp\left(-\int_s^f \left(\sum_{S \in S(R)} w(r, S) \mu_S(t)\right) dt\right) = \exp\left(-\sum_{S \in S(R)} w(r, S) M_S(s, f)\right),$$

and the probability that a randomly chosen tuple in $R(s)$ survives until time f is therefore

$$p_R(s, f) = \mathrm{E}_{r \in R(s)}\left[\exp\left(-\sum_{S \in S(R)} w(r, S) M_S(s, f)\right)\right].\tag{2}$$

∎

The complexity analysis of integrating $\mu_S(t)$ over time to obtain $M_S(s, f)$ is similar to that of Section 2.1.1. However, the computation required by Lemma 3 may be prohibitive, in the most general case, because it requires knowing the empirical distribution of the $w(r, S)$ over all $r \in R(s)$ for all $S \in S(R)$. This empirical distribution can be computed accurately by computing for each tuple, upon insertion, the number of tuples in any $S \in S(R)$ with a comparable foreign key, using either histograms or by directly querying the database. Maintaining this information requires $O(|R(s)||S(R)|)$ space. This complexity can be reduced using a manageably-sized sample from $R(s)$. Our initial analysis of real-world applications, however, indicates that in many cases, $w(r, S)$ takes on a much simpler form, in which $w(r, S)$ is identical for all $r \in R(s)$. We term such a typical relationship between R and $S \in S(R)$ a *fixed multiplicity*, as defined next:

Definition 2 *The pair* $\langle R, S \rangle$, *where* $S \in S(R)$, *has* fixed multiplicity *if* $w(r, S)$ *is identical for all tuples in* R. *In this case, we denote its common value by* $w(R, S)$. □

Example 4 (Fixed multiplicies in Net.Commerce) *Consider the example multigraph of Figure 1. Both* `DISCALC` *and* `SCALE` *reference* `MERCHANT`. *It is clear that the discount calculation of a product (as stored in* `DISCALC`*) cannot reference a different merchant than* `SCALE`. *The only exception is when the foreign key in* `SCALE` *is assigned with a null value. If this is the case, however, there is only a single tuple in* `MERCHANT` *whose deletion requires the deletion of a tuple in* `DISCALC`. *Thus,*

for any tuple $r \in$ DISCALC, $w(r, \texttt{SCALE}) = w(r, \texttt{MERCHANT}) = 1$ and therefore $\langle\texttt{DISCALC}, \texttt{SCALE}\rangle$ and $\langle\texttt{DISCALC}, \texttt{MERCHANT}\rangle$ both have fixed multiplicity of 1. Now consider CGRYREL. Since each tuple in CGRYREL describes the relationship between a category and a subcategory, it is clear that its two foreign keys to CATEGORY must always have distinct values. Thus, $\langle\texttt{CGRYREL}, \texttt{CATEGORY}\rangle$ has a fixed multiplicity, and $w(\texttt{CGRYREL}, \texttt{CATEGORY}) = 2$. □

As the following lemma shows, fixed multiplicities permit great simplification in computing $p_R(s, f)$.

Lemma 4 *If $\langle R, S\rangle$ has fixed multiplicity for all $S \in S(R)$, $p_R(s, f) = \exp(-\widetilde{M}_R(s, f))$, where $\widetilde{M}_R(s, f) = \int_s^f \tilde{\mu}_R(t)\, dt$ and $\tilde{\mu}_R(t) = \sum_{S \in S(R)} w(R, S)\mu_S(t)$.*

Proof.

$$
\begin{aligned}
p_R(s, f) &= \mathrm{E}_{r \in R(s)}\left[\exp\left(-\sum_{S \in S(R)} w(r, S)M_S(s, f)\right)\right] \\
&= \mathrm{E}_{r \in R(s)}\left[\exp\left(-\int_s^f \left(\sum_{S \in S(R)} w(r, S)\mu_S(t)\right) dt\right)\right] \\
&= \mathrm{E}_{r \in R(s)}\left[\exp\left(-\int_s^f \left(\sum_{S \in S(R)} w(R, S)\mu_S(t)\right) dt\right)\right] \\
&= \exp\left(-\sum_{S \in S(R)} \left(w(R, S)\int_s^f \mu_S(t)\, dt\right)\right) \\
&= \exp\left(-\sum_{S \in S(R)} w(R, S)M_S(s, f)\right).
\end{aligned}
$$

∎

Since $w(R, S)$ is fixed and constant over time, no additional statistics need to be collected for it. As a final note, it is worth noting that in certain situations, another alternative may also be available. Let $\{N_R(t), t \geq 0\}$ be a nonhomogeneous Poisson process with intensity function $\hat{\mu}_R(t)$, modeling the occurrence of *deletion events* in R. At deletion event i, a random number Δ_i^- tuples are deleted from R. Generally speaking, this kind of model cannot be accurate, since it ignores that each deletion causes a reduction in the number of remaining tuples, and thus presumably a change in the spacing of subsequent deletion events. However, it may be reasonably accurate for large databases with either a stable or steadily growing number of tuples, or whenever the time interval $(s, f]$ is sufficiently small. Statistical analysis of the database log would be required to say whether the model is applicable. If the model is valid, then the stochastic process $\{D_R(t), t \geq 0\}$ representing the cumulative number of deletions through time t, can be taken to be a compound Poisson process. The expected number of deleted tuples during $(s, f]$ may be computed via

$$
\mathrm{E}[D_R(t)] = \int_s^f \mu_R(t)\, \mathrm{E}\!\left[\Delta^-\right] dt = M_R(s, f)\, \mathrm{E}\!\left[\Delta^-\right],
$$

where Δ^- is a generic random variable distributed like the $\{\Delta_i^-\}$.

10

2.3 Tuple survival: the combined effect of insertions and deletions

Some tuples inserted during $(s, f]$ may be deleted by time f. Let the random variable $X_R(s, f)$ denote the number of tuples inserted during the interval $(s, f]$ that survive through time f. Consider any tuple inserted into R at time $t \in (s, f]$, and denote its chance of surviving through time f by $\hat{p}_R(t, f)$. For any $S \in S(R)$ and $t \in (s, f]$, let $W(R, S, t)$ be a random variable denoting the value of $w(r, S)$, given that r was inserted into R at time t. Let $W(R, t)$ denote the random vector, of length $|S(R)|$, formed by concatenating the $W(R, S, t)$ for all $S \in S(R)$.

Lemma 5 $\hat{p}_R(t, f) = \mathrm{E}_{W(R,t)}\left[\exp\left(-\sum_{S \in S(R)} W(R, S, t) M_S(t, f)\right)\right]$. *When $\langle R, S \rangle$ has fixed multiplicity for all $S \in S(R)$, then $\hat{p}_R(t, f) = p_R(t, f) = \exp(-\widetilde{M}_R(t, f))$.*

Proof. Let $L_{R,t}$ denote the lifetime of a tuple inserted into R at time t. Similarly to the proof of Lemma 3, we know that $L_{R,t} \sim \mathrm{Exp}_t(\sum_{S \in S(R)} W(R, S, t) \mu_S(\cdot))$. The probability such a tuple survives through time f is the random quantity

$$\exp\left(-\int_t^f \left(\sum_{S \in S(R)} W(R, S, t) \mu_S(\tau)\right) d\tau\right) = \exp\left(-\sum_{S \in S(R)} W(R, S, t) M_S(t, f)\right).$$

Considering all the possible elements of the vector $W(R, t)$, we then obtain

$$\hat{p}_R(t, f) = \mathrm{E}_{W(R,t)}\left[\exp\left(-\sum_{S \in S(R)} W(R, S, t) M_S(t, f)\right)\right],$$

Assume now that $\langle R, S \rangle$ has fixed multiplicity for all $S \in S(R)$. Consequently, we replace $W(R, S, t)$ with $w(R, S)$. Drawing on the proof of the previous lemma,

$$\hat{p}_R(t, f) = \mathrm{E}_{W(R,t)}\left[\exp\left(-\sum_{S \in S(R)} w(R, S) M_S(t, f)\right)\right]$$

$$= \exp\left(-\int_t^f \left(\sum_{S \in S(R)} w(R, S) \mu_S(\tau)\right) d\tau\right)$$

$$= p_R(t, f).$$

$\blacksquare$

The following proposition establishes the formula for the expected value of $X_R(s, f)$.

Proposition 1 $\mathrm{E}[X_R(s, f)] = \widetilde{\Lambda}_R(s, f) \mathrm{E}[\Delta_R^+]$, *where* $\widetilde{\Lambda}_R(s, f) = \int_s^f \lambda_R(t) \hat{p}_R(t, f) \, dt$. *In the simple case where each insertion involves exactly one tuple, $X_R(s, f) \sim \mathrm{Poisson}(\widetilde{\Lambda}_R(s, f))$.*

Proof. Let N be the number of insertion events in $(s, f]$, and let their times be $\{T_1, T_2, \dots, T_N\}$. Suppose that $N = n$ and that insertion event i happens at time $t_i \in (s, f]$. Event i inserts a random number of tuples $\Delta_{R,i}^+$, each of which has probability $\hat{p}_R(t_i, f)$ of surviving through time f. Therefore, the expected number of tuples surviving through f from insertion event i is $\mathrm{E}[\Delta_R^+] \hat{p}_R(t_i, f)$. Consequently,

$$\mathrm{E}[X_R(s, f) \mid N = n, T_1 = t_1, T_2 = t_2, \dots, T_n = t_n] = \mathrm{E}[\Delta_R^+] \sum_{i=1}^n \hat{p}_R(t_i, f).$$

Next, we recall, given that $N = n$, that the times T_i of the insertion events are distributed like n independent random variables with probability density function $\lambda_R(t)/\Lambda_R(s, f)$ on the interval $(s, f]$. Thus,

$$\mathrm{E}\big[X_R(s, f) \mid N = n\big] = \mathrm{E}_{T_1, \dots, T_n}\left[\mathrm{E}\big[\Delta_R^+\big] \sum_{i=1}^{n} \hat{p}_R(T_i, f)\right]$$

$$= \mathrm{E}\big[\Delta_R^+\big] \sum_{i=1}^{n} \left(\int_s^f \hat{p}_R(t, f) \frac{\lambda_R(t)}{\Lambda_R(s, f)}\, dt\right)$$

$$= n\left(\frac{\mathrm{E}\big[\Delta_R^+\big] \widetilde{\Lambda}_R(s, f)}{\Lambda_R(s, f)}\right).$$

Finally, removing the conditioning on $N = n$, we obtain

$$\mathrm{E}[X_R(s, f)] = \mathrm{E}_N\left[N\left(\frac{\mathrm{E}\big[\Delta_R^+\big] \widetilde{\Lambda}_R(s, f)}{\Lambda_R(s, f)}\right)\right]$$

$$= \Lambda_R(s, f)\left(\frac{\mathrm{E}\big[\Delta_R^+\big] \widetilde{\Lambda}_R(s, f)}{\Lambda_R(s, f)}\right)$$

$$= \mathrm{E}\big[\Delta_R^+\big] \widetilde{\Lambda}_R(s, f).$$

In the case that Δ_R^+ is always 1, we may use the notion of a *filtered* Poisson process: if we consider only tuples that manage to survive until time f, the chance of a single insertion in time interval $[t, t+\Delta t]$ no longer has the limiting value $\lambda_R(t)\Delta t$, but instead $\lambda_R(t)\hat{p}_R(t, f)\Delta t$. Therefore, the insertion of surviving tuples can be viewed as a nonhomogeneous Poisson process with intensity function $\lambda_R(t)\hat{p}_R(t, f)$ over the time interval $(s, f]$, so $X_R(s, f) \sim \mathrm{Poisson}(\widetilde{\Lambda}_R(s, f))$. ∎

In the general case, the computation of $\widetilde{\Lambda}_R(s, f)$ will require approximation by numerical integration techniques; the complexity of this calculation will depend on the information-theoretic properties of $\lambda_R(\cdot)$ and the $\mu_S(\cdot)$, $S \in S(R)$, but is unlikely to be burdensome if these functions are reasonably smoothly-varying. In one important special case, however, the complexity of computing $\widetilde{\Lambda}_R(s, f)$ is essentially the same as that of calculating $\Lambda_R(s, f)$: suppose that for some constants $\alpha(R, S)$, $S \in S(R)$, one has that $\mu_S(t) = \alpha(R, S)\lambda_R(t)$ for all t. That is, the general insertion and deletion activity level of the relations in $S(R)$ all vary proportionally to some common fluctuation pattern. In this case, we have $\widetilde{\mu}_R(t) = \alpha(R)\lambda_R(t)$ and $\widetilde{M}_R(s, f) = \alpha(R)\Lambda_R(s, f)$ for all t, s, f, where $\alpha(R) = \sum_{S \in S(R)} \alpha(R, S)$. Making a substitution $u(t) = \Lambda_R(t, f)$, we have:

$$\widetilde{\Lambda}_R(s, f) = \int_s^f \lambda_R(t) \exp(-\alpha(R)\Lambda_R(t, f))\, dt$$

$$= \int_s^f \left(\frac{-d\Lambda_R(t, f)}{dt}\right) \exp(-\alpha(R)\Lambda_R(t, f))\, dt$$

$$= \int_s^f -\exp(-\alpha(R)u(t))\, du(t)$$

$$= -\int_{u(s)}^{u(f)} e^{-\alpha(R)u}\, du$$

$$= \frac{1}{\alpha(R)}\left(1 - e^{-\alpha(R)\Lambda(s, f)}\right),$$

so $\widetilde{\Lambda}_R(s, f)$ can be calculated directly from $\Lambda(s, f)$.

We define the random variable $Y_R(s, f)$ to be the number of tuples in $R(s)$ that survive through time f.

Proposition 2 $\mathrm{E}[Y_R(s, f)] = p_R(s, f) |R(s)|$ *and* $\mathrm{E}[|R(f)|] = p_R(s, f) |R(s)| + \widetilde{\Lambda}_R(s, f) \mathrm{E}[\Delta_R^+]$.

Proof. Each tuple in $R(s)$ has a survival probability of $p_R(s, f)$, which yields that $\mathrm{E}[Y_R(s, f)] = p_R(s, f) |R(s)|$. By the definitions of $Y_R(s, f)$ and $X_R(s, f)$, one has that

$$|R(f)| = Y_R(s, f) + X_R(s, f),$$

so therefore

$$\mathrm{E}[|R(f)|] = \mathrm{E}[Y_R(s, f)] + \mathrm{E}[X_R(s, f)] = p_R(s, f) |R(s)| + \widetilde{\Lambda}_R(s, f) \mathrm{E}[\Delta_R{}^+].$$

$\blacksquare$

In cases where deletions may also be accurately modeled as a compound Poisson process, we have

$$\begin{aligned}
\mathrm{E}[|R(f)|] &= \mathrm{E}[|R(s)|] + B_R(s, f) - D_R(s, f) \\
&= |R(s)| + \Lambda_R(s, f) \mathrm{E}[\Delta^+] - M_R(s, f) \mathrm{E}[\Delta^-].
\end{aligned}$$

Example 5 (The homogeneous case) *Assume that* $\mathrm{E}[\Delta_R^+] = 1$, *that* $\langle R, S \rangle$ *has fixed multiplicity for all* $S \in S(R)$, *and furthermore* $\lambda_R(t)$ *and* $\mu_S(t)$, *for all* $S \in S(R)$, *are constant functions, that is,* $\lambda_R(t) = \lambda_R$ *for all times* t *and* $\mu_S(t) = \mu_S$ *for all* $S \in S(R)$ *and times* t. *Then* $\Lambda_R(s, f) = \lambda_R \cdot (f - s)$ *and* $M_R(s, f) = \mu_R \cdot (f - s)$. *Thus, letting* $\tilde{\mu}_R = \sum_{S \in S(R)} w(R, s) \mu_S$,

$$\widetilde{\Lambda}_R(s, f) = \int_s^f \lambda_R e^{-\tilde{\mu}_R(t-s)} \, dt = \frac{\lambda_R}{\tilde{\mu}_R} \left(1 - e^{-\tilde{\mu}_R(f-s)} \right),$$

and assuming $\Delta_{R,i}^+ = 1$ *for all* $i > 0$,

$$\mathrm{E}[|R(f)|] = |R(s)| \, e^{-\tilde{\mu}_R(f-s)} + \frac{\lambda_R}{\tilde{\mu}_R} \left(1 - e^{-\tilde{\mu}_R(f-s)} \right) = \frac{\lambda_R}{\tilde{\mu}_R} + e^{-\tilde{\mu}_R(f-s)} \left(|R(s)| - \frac{\lambda_R}{\tilde{\mu}_R} \right).$$

$\square$

2.4 Tuples with non-exponential life spans

We now consider the possibility that tuples in R have a stochastic life span L_R^I that is not memoryless, but rather has some general cumulative distribution function G_R. For example, if tuples in R correspond to pieces of work in process on a production floor, the likelihood of deletion might rise the longer the tuple has been in existence. Let us consider a single relation, and thus no referential integrity constraints. For any tuple r, let $b(r)$ denote the time it was created. We next establish the expected cardinality of R at time f.

Proposition 3 *In the case that tuples in* R *have lifetimes with a general cumulative distribution function* G_R,

$$\mathrm{E}[|R(f)|] = |R(s)| \, \mathrm{E}_{r \in R(s)} \left[\frac{1 - G_R(f - b(r))}{1 - G_R(s - b(r))} \right] + \mathrm{E}[\Delta_R^+] \int_s^f \lambda_R(t) \, (1 - G_R(f - t)) \, dt. \quad (3)$$

Proof. Let L_R^{I} denote a generic random variable with cumulative distribution G_R. The probability of $r \in R(s)$ surviving throughout $(s, f]$ is then

$$\mathrm{P}\{L_R^{\mathrm{I}} \geq f - b(r) \mid L_R^{\mathrm{I}} \geq s - b(r)\} = \frac{1 - G_R(f - b(r))}{1 - G_R(s - b(r))},$$

and therefore the expected number of tuples in $R(s)$ that survive through time f is

$$\mathrm{E}[Y_R(s, f)] = \sum_{r \in R(s)} \left(\frac{1 - G_R(f - b(r))}{1 - G_R(s - b(r))} \right) = |R(s)| \, \mathrm{E}_{r \in R(s)} \left[\frac{1 - G_R(f - b(r))}{1 - G_R(s - b(r))} \right].$$

We now consider a tuple r inserted at some time $t \in (s, f]$. The probability that such a tuple survives through time f is simply $\hat{p}_R(t, f) = 1 - G_R(f - t)$. By reasoning similar to the proof of Lemma 5,

$$\mathrm{E}[X_R(s, f]] = \mathrm{E}\big[\Delta_R^+\big] \, \widetilde{\Lambda}_R(s, f) = \mathrm{E}\big[\Delta_R^+\big] \int_s^f \lambda_R(t) \, (1 - G_R(f - t)) \, dt.$$

The conclusion then follows from $\mathrm{E}[|R(f)|] = \mathrm{E}[Y_R(s, f)] + \mathrm{E}[X_R(s, f)]$ ∎

It is worth noting that, as opposed to the memoryless case presented above, the calculation of $\mathrm{E}[Y_R(s, f)]$ requires remembering the commit times $b(r)$ of all tuples $r \in R(s)$, or equivalently the ages of all such tuples. Of course, for large relations R, a reasonable approximation could be obtained by using a manageably-sized sample to estimate

$$\mathrm{E}_{r \in R(s)} \left[\frac{1 - G_R(f - b(r))}{1 - G_R(s - b(r))} \right].$$

It is likely that the integral in (3) will require general numerical integration, depending on the exact form of G_R.

2.5 Summary

In this section, we have provided a model for the insertion and deletion of tuples in a relational database. The immediate benefit of this model is the computation of the expected relation cardinality ($\mathrm{E}[|R(f)|]$), given an initial cardinality and insertion and tuple life span parameters. Relation cardinality has proven to be an important property in many database tools, including query optimization and database tuning. Reasonable assumptions regarding constant multiplicity allow, once appropriate statistics have been gathered, a rapid computation of cardinalities in this framework. Section 4 elaborates on statistics gathering and model validation.

A note regarding tuples with non-exponential life spans is now warranted. For the case of a single relation, non-exponential life spans add only a moderate amount of complexity to our model, namely the requirement to store at least an approximation of the distribution of tuples ages in $R(s)$. For multiple relations with referential integrity constraints, however, the complexity of dealing with general tuple life spans is much greater. First, to estimate the cardinality of $R(f)$, we must keep (approximate) tuple age distributions for all relations in $S(R)$. Second, because the tuple life span distributions of some of the members of $S(R)$ are not memoryless, we cannot combine them with a simple relation like (1). Furthermore, in attempting to find the distribution of the remaining life span of a particular tuple $r \in R(s)$, it may become necessary to consider the issue of the correlation of ages of tuples in $R(s)$ with the ages of the corresponding tuples in other relations of $S(R)$. Because of these complications, we defer further consideration of non-exponential tuple life spans to future research.

3 Modeling data modification

This section describes various ways to model the modification of the contents of tuples. We start with a general approach, using Markov chains, followed by several special cases where the amount of computation can be greatly reduced.

3.1 Content-dependent updates

In this section, we model the modification of the contents of tuples as a finite-state continuous-time Markov chain, thus assuming dependence on tuples' previous contents. For each relation R, we allow for some (possibly empty) subset $\mathcal{C}(R) \subset \mathcal{A}(R)$ of its attributes to be subject to change over the lifetime of a tuple. We do not permit primary key fields to be modified, that is, $\mathcal{C}(R) \cap \mathcal{K}(R) = \emptyset$.

Attribute values may change at time instants called *transition events*, which are the transition times of the Markov chain. We assume that the spacing of transition events is memoryless with respect to the age of a tuple (although it may depend on the time and the current value of the attribute, as demonstrated below). For any attribute A, tuple r, time s, and value $v \in \operatorname{dom} A$ with $r.A(s) = v$, the time remaining until the next transition event for $r.A$ is a random variable $\tau_{v,s}^{R,A}$ with the distribution $\operatorname{Exp}_s(\ell_v^{R,A} \gamma_{R,A}(\cdot))$, where $\gamma_{R,A} : \Re \to [0, \infty)$ is a function giving the general instantaneous rate of change for the attribute, and $\ell_v^{R,A}$ is a nonnegative scalar which we call the *relative exit rate* of v. We define $\Gamma_{R,A}(s, f) = \int_s^f \gamma_{R,A}(t)\, dt$. When a transition event occurs from state $u \in \operatorname{dom} A$, attribute A changes to $v \in \operatorname{dom} A$ with probability $P_{u,v}^{R,A}$.

Suppose $\mathcal{A} = \{A_1, A_2, ..., A_k\} \subseteq \mathcal{R}$ is an independently varying set of attributes, and $v = \langle v_1, v_2, ..., v_k \rangle \in \operatorname{dom} \mathcal{A}$ is a compound value. Then the time until the next transition event for $r.\mathcal{A}$ is $\tau_{v,s}^{R,\mathcal{A}} = \min\{\tau_{v_1,s}^{R,A_1}, \dots, \tau_{v_k,s}^{R,A_k}\}$. As a rule, we will assume that the modification processes for the attributes of a relation are independent, so $\tau_{v,s}^{R,\mathcal{A}} \sim \operatorname{Exp}_s(\sum_{i=1}^k \ell_{v_i}^{R,A_i} \gamma_{A_i,R}(\cdot))$. When the functions γ_{R,A_i} are identical for $i = 1, \dots, k$, we define $\gamma_{R,\mathcal{A}} = \gamma_{R,A_i}$ and $\ell_v^{R,\mathcal{A}} = \sum_{i=1}^k \ell_{v_i}^{R,A_i}$, so $\tau_{v,s}^{R,\mathcal{A}} \sim \operatorname{Exp}_s(\ell_v^{R,\mathcal{A}} \gamma_{R,\mathcal{A}}(\cdot))$. To justify the assumption of independence, we note that coordinated modifications among attributes can be modeled by replacing the coordinated attributes with a single compound attribute (this technique requires that the attributes have identical $\gamma_{R,\mathcal{A}}(\cdot)$ functions, which is reasonable if they change in a coordinated way).

Under these assumptions, let $\overline{\mathcal{C}}(R)$ denote a partition of $\mathcal{C}(R)$ into subsets $\mathcal{A}$ such that any two attributes $A_1, A_2 \in \mathcal{C}(R)$ vary dependently iff they are in the same $\mathcal{A} \in \overline{\mathcal{C}}(R)$.

Example 6 (First alteration time) *For a relation R and time s, we define $\Upsilon_{R,s}$ to be the amount of time until the next change in R, be it a tuple insertion, a tuple deletion, or an attribute modification. Also, for any $S \in S(R)$, let $D(R, S, s)$ denote the number of tuples in $S(s)$ whose deletion would force the deletion of some tuple in $R(s)$ (and therefore $D(R, R, s) = |R(s)|$). The following proposition establishes the distribution of $\Upsilon_{R,s}$.*

Proposition 4 $\Upsilon_{R,s} \sim \operatorname{Exp}_s(\zeta_R(\cdot))$, *where*

$$\zeta_R(t) = \lambda_R(t) \;+\; \sum_{S \in S(R)} D(R, S, s) \mu_S(t) \;+\; \sum_{\mathcal{A} \in \overline{\mathcal{C}}(R)} h(R, A, s) \gamma_{R,\mathcal{A}}(t)$$

and $h(R, A, s) = \sum_{v \in \operatorname{dom} \mathcal{A}} \hat{R}_{\mathcal{A},v}(s) \ell_v^{R,A}$. The probability of any alteration to R in the time interval $(s, f]$ is $1 - e^{-Z_R(s,f)}$, where

$$Z_R(s, f) = \Lambda_R(s, f) \;+\; \sum_{S \in S(R)} D(R, S, s) M_S(t) \;+\; \sum_{\mathcal{A} \in \overline{\mathcal{C}}(R)} h(R, A, s) \Gamma_{R,\mathcal{A}}(s, f)$$

Proof. Let $\Upsilon^{\mathrm{I}}_{R,s}$, $\Upsilon^{\mathrm{M}}_{R,s}$, and $\Upsilon^{\mathrm{D}}_{R,s}$ be the times until the next insertion, modification, and deletion in R, respectively. From Section 2, we have that $\Upsilon^{\mathrm{I}}_{R,s} \sim \mathrm{Exp}_s(\lambda_R(\cdot))$. Now, for each $S \in S(R)$, there are $D(R, S, s)$ tuples whose deletion would cause a deletion in R. The time until deletion of any such $r \in S \in S(R)$ is distributed like $\mathrm{Exp}_s(\mu_S(\cdot))$. The deletion processes for all these tuples are independent across all of $S(R)$, so we can use (1) to conclude that

$$\Upsilon^{\mathrm{D}}_{R,s} \sim \mathrm{Exp}_s\left(\sum_{S \in S(R)} D(R, S, s)\mu_S(\cdot) \right).$$

From the preceding discussion, we have

$$\Upsilon^{\mathrm{M}}_{R,s} = \tau^{\mathcal{C}(R),R}_{v,s} \sim \mathrm{Exp}_s\left(\sum_{\mathcal{A} \in \overline{\mathcal{C}}(R)} \ell^{R,\mathcal{A}}_{r.\mathcal{A}(s)}\gamma_{R,\mathcal{A}}(\cdot) \right).$$

Since $\Upsilon_{R,s} = \min\{\Upsilon^{\mathrm{I}}_{R,s}, \Upsilon^{\mathrm{M}}_{R,s}, \Upsilon^{\mathrm{D}}_{R,s}\}$, we therefore have, again using independence and (1), that $\Upsilon_{R,s} \sim \mathrm{Exp}_s(\zeta_R(\cdot))$, where

$$\zeta_R(t) = \lambda_R(t) \;+\; \sum_{S \in S(R)} D(R, S, s)\mu_S(t) \;+\; \sum_{r \in R(s)}\left[\sum_{\mathcal{A} \in \overline{\mathcal{C}}(R)} \ell^{R,\mathcal{A}}_{r.\mathcal{A}(s)}\gamma_{R,\mathcal{A}}(t) \right]$$

$$= \lambda_R(t) \;+\; \sum_{S \in S(R)} D(R, S, s)\mu_S(t) \;+\; \sum_{\mathcal{A} \in \overline{\mathcal{C}}(R)} h(R, A, s)\gamma_{R,\mathcal{A}}(t).$$

Integrating over $(s, f]$ results in

$$Z_R(s, f) = \int_s^f \zeta_R(t)\, dt$$

$$= \Lambda_R(s, f) \;+\; \sum_{S \in S(R)} D(R, S, s)M_S(t) \;+\; \sum_{\mathcal{A} \in \overline{\mathcal{C}}(R)} h(R, A, s)\gamma_{R,\mathcal{A}}(s, f).$$

Therefore, the probability of any alteration to R in the time interval $(s, f]$ is

$$\mathrm{P}\left\{\Upsilon_{R,s} < f - s\right\} = 1 - e^{-Z_R(s,f)}$$

$\blacksquare$

Example 6 continued (First alteration transcription policy) *Suppose the user wishes to refresh her replica of relation R whenever the probability that it contains any inaccuracy exceeds some threshold π, a tactic we call the* first alteration policy. *Then, a refresh is required at time f if $1 - e^{-Z_R(s,f)} > \pi$.* $\square$

Given $\mathcal{A}$, we describe the transition process for the value $r.\mathcal{A}$ over time by probabilities

$$P^{R,\mathcal{A}}_{u,v}(s, f) = \mathrm{P}\left\{r.\mathcal{A}(f)=v \mid r.\mathcal{A}(s)=u\right\},$$

for any two values $u, v \in \mathrm{dom}\, \mathcal{A}$ and times $s < f$. Under the assumption of independence,

$$P^{R,\mathcal{A}}_{u,v}(s, f) = \prod_{i=1}^k P^{R,A_i}_{u_i,v_i}(s, f). \tag{4}$$

Given any simple attribute A, we define $q_{u,v}^{R,A}$, the *relative transition rate* from u to v, by

$$q_{u,v}^{R,A} = \ell_u^{R,A} P_{u,v}^{R,A}.$$

Given a set of attributes $\mathcal{A}$ with identical $\gamma_{R,A}(\cdot)$ functions, the compound transition rate $q_{u,v}^{R,\mathcal{A}}$ may be computed via

$$q_{u,v}^{R,\mathcal{A}} = \ell_u^{R,\mathcal{A}} P_{u,v}^{R,\mathcal{A}} = \left(\sum_{i=1}^k \ell_{u_i}^{R,A_i} \right) \left(\prod_{i=1}^k P_{u_i,v_i}^{R,A_i} \right). \tag{5}$$

Let $Q^{R,A}$ be the matrix of $q_{u,v}^{R,A}$, where $q_{u,u}^{R,A} = -\ell_u^{R,A}$.

Proposition 5 *The matrix $P^{R,A}(s,f)$ of elements $P_{u,v}^{R,A}(s,f)$ is given by the matrix exponential formula*

$$P^{R,A}(s,f) = \exp\!\left(\Gamma_{R,A}(s,f)\, Q^{R,A} \right) = \sum_{n=0}^\infty \frac{\Gamma_{R,A}(s,f)^n}{n!} \left(Q^{R,A} \right)^n. \tag{6}$$

Proof. Consider a continuous-time Markov chain on the same state space $\operatorname{dom} A$, and with the same instantaneous transition probabilities $P_{u,v}^{R,A}$, where $u,v \in \operatorname{dom} A$. However, in the new chain, the holding time in each state v is simply a homogeneous exponential random variable with arrival rate $\ell_v^{R,A}$. We call this system the *linear-time* chain, to distinguish it from the original chain. Define $\overline{P}_{u,v}^{A,R}(t)$ to be the chance that the linear-time chain is in state v at time t, given that it is in state u at time 0. Standard results for finite-state continuous time Markov chains imply that

$$\overline{P}_{u,v}^{R,A}(t) = \exp\!\left(t\, Q^{R,A} \right) = \sum_{n=0}^\infty \frac{t^n}{n!} \left(Q^{R,A} \right)^n.$$

By a transformation of the time variable, we then assert that

$$P_{u,v}^{R,A}(s,f) = \overline{P}_{u,v}^{R,A}(\Gamma_{R,A}(s,f)),$$

from which the result follows. ∎

Example 7 (Query optimization, revisited) *As the following proposition shows, our model can be used to estimate the histogram of a relation R at time f. A query optimizer running at time f could use expected histograms, calculated in this manner, instead of the old histograms $\hat{R}_A(s)$.*

Proposition 6 *Assume that $w(r,S)$, for all $S \in S(R)$, is independent of the attribute values $r.A(s)$ for all $A \in \mathcal{C}(R)$. Let $\hat{\omega}_u^{R,A}(t)$ denote the probability that $r.A(t) = u$, given that r is inserted into R at time t. Then, for all $v \in \operatorname{dom} A$,*

$$\mathrm{E}\!\left[\hat{R}_{A,v}(f) \right] = p_R(s,f) \sum_{u \in \operatorname{dom} A} \hat{R}_{A,u}(s) P_{u,v}^{R,A}(s,f)$$

$$+ \quad \mathrm{E}[\Delta_R^+] \sum_{u \in \operatorname{dom} A} \left(\int_s^f \hat{\omega}_u^{R,A}(t) \hat{p}_R(s,f) \lambda_R(t) P_{u,v}^{R,A}(t,f)\, dt \right). \tag{7}$$

Proof. We first compute the expected number of surviving tuples r whose values $r.A$ migrate to v. Given a value $u \in \mathrm{dom}\, A$, there are $\hat{R}_{A,u}(s)$ tuples at time s such that $r.A(s) = u$. Using the previous results, the expected number of these tuples surviving through time f is $\hat{R}_{A,u}(s)p_R(s,f)$, and the probability of each surviving tuple r having $r.A(f) = v$ is $P_{u,v}^{R,A}(s,f)$. Using the independence assumption and summing over all $u \in \mathrm{dom}\, A$, one has that the expected numbers of tuples in $R(s)$ that survive through f and have $r.A(f) = v$ is

$$\sum_{u \in \mathrm{dom}\, A} \hat{R}_{A,u}(s)p_R(s,f)P_{u,v}^{R,A}(s,f) = p_R(s,f) \sum_{u \in \mathrm{dom}\, A} \hat{R}_{A,u}(s)P_{u,v}^{R,A}(s,f).$$

We next consider newly inserted tuples. Recall that $\hat{\omega}_u^{R,A}(t)$ denotes the probability that $r.A(t) = u$, given that r is inserted into R at time t. Suppose that an insertion occurs at time $t \in (s,f]$. The expected number of tuples r created at this insertion that both survive until f and have $r.A(f) = v$ is

$$\hat{p}_R(t,f) \sum_{u \in \mathrm{dom}\, A} \omega_u^{R,A}(t)P_{u,v}^{R,A}(t,f).$$

By logic similar to Proposition 5, one may then conclude that the expected number of newly-inserted tuples that survive through time f and have $r.A(f) = v$ is

$$\mathrm{E}\big[\Delta_R^+\big] \sum_{u \in \mathrm{dom}\, A} \left(\int_s^f \hat{\omega}_u^{R,A}(t)\hat{p}_R(s,f)\lambda_R(t)P_{u,v}^{R,A}(t,f)\, dt \right).$$

The result follows by adding the last two expressions. ∎

Example 7 continued *We next consider whether the complexity of calculating (7) is preferable to recomputing the histogram vector $\hat{R}_A(f)$. This topic is quite involved and depends heavily on the specific structure of the database (e.g., the availability of indices) and the specific application (e.g., the concentration of values in a small subset of an attribute's domain). In what follows, we lay out some qualitative considerations in deciding whether calculating (7) would be more efficient than recalculating $\hat{R}_A(f)$ "from scratch." Experimentation with real-world application is left for further research.*

Generally speaking, direct computation of the histogram of an attribute A (in the absence of an index for A) can be done by either scanning all tuples (although sampling may also be used) or scanning a modification log to capture changes to the prior histogram vector $\hat{R}_A(s)$ during $(s,f]$. Therefore, the recomputation can be performed in $\mathrm{O}(\min\{|R(f)|, T(s,f)\})$ time, where $T(s,f)$ denotes the total number of updates during $(s,f]$. Whenever $|R(f)|$ and $T(s,f)$ are both large — i.e., the database is large and the transaction load is high — the straightforward techniques will be relatively unattractive. As for the estimation technique, it will probably work best when $|\mathrm{dom}\, A|$ is small (for example, for a binary attribute) or whenever the subset of actually utilized values in the domain is small. In addition, commercial databases recompute the entire histogram as a single, atomic task. Formula (7), on the other hand, can be performed on a subset of the attribute values. For example, in the case of exact matching (say, a condition of the form WHERE $A = v$*), it is sufficient to compute $\hat{R}_{A,v}(f)$, rather than the full $\hat{R}_A(f)$ vector. Finally, it is worth noting that the computing the expected value of $\hat{R}_{A,v}(f)$ via (7) does not require locking R, while a full histogram recomputation may involve extended periods of locking.* □

We next consider the number of tuples in $R(s)$ that have survived through time f without being modified, which we denote $Y_R^-(s, f)$. The expectation of this random variable is

$$\mathrm{E}\big[Y_R^-(s, f)\big] = p_R(s, f) \sum_{v \in \mathrm{dom}\, \mathcal{A}} \hat{R}_{\mathcal{A}, v}(s) P_{v,v}^{R, \mathcal{A}}(s, f)$$

We let $Y_R^+(s, f) = Y_R(s, f) - Y_R^-(s, f)$ denote the number of tuples in $R(s)$ that have survived through time f and were modified; it follows from the linearity of the $\mathrm{E}[\cdot]$ operator that

$$\mathrm{E}\big[Y_R^+(s, f)\big] = \mathrm{E}[Y_R(s, f)] - \mathrm{E}\big[Y_R^-(s, f)\big].$$

3.1.1 Complexity analysis of content-dependent updates

In practice, as with computing a scalar exponential, only a limited number of terms will be needed to compute the sum (6) to machine precision. It is worth noting that efficient means of calculating (6) are a major topic in the field of computational probability.

In the case of a compound attribute $\mathcal{A} = \{A_1, A_2, ..., A_k\}$ with independently varying components, it will be computationally more efficient to first calculate the individual transition probability matrices $P^{A_i, R}(s, f)$ via (6), and then calculate the joint probability matrix $P_{u,v}^{R, \mathcal{A}}(s, f)$ using (4), rather than first finding the joint exit rate matrix $Q^{R, \mathcal{A}}$ via (5) and then applying (6). The former approach would involve repeated multiplications of square matrices of size $|\mathrm{dom}\, A_i|$, for $i = 1, \ldots, k$, resulting in a computational complexity of $\mathrm{O}(\sum_{i=1}^k n_i |\mathrm{dom}\, A_i|^\nu)$, where n_i is the number of iterations needed to compute the sum (6) to machine precision, and the complexity of multiplying two $n \times n$ matrices is $\mathrm{O}(n^\nu)$.[5] The latter would involve multiplying square matrices of size $\prod_{i=1}^k |\mathrm{dom}\, A_i|$, resulting in the considerably worse complexity of $\mathrm{O}(n(\prod_{i=1}^k |\mathrm{dom}\, A_i|)^\nu)$, where n is the number of iterations needed to obtain the desired precision.

3.2 Simplified modification models

We next introduce several possible simplifications of the general Markov chain case. To do so, we start by differentiating numeric domains from non-numeric domains. Certain database attributes $A \in \mathcal{A}$, such as prices and order quantities, represent numbers, and numeric operations such as addition are meaningful for these attributes. For such attributes, one can easily define a distance function between two attribute values, as we shall see below. We call the domains $\mathrm{dom}\, A$ of such attributes *numeric domains*, and denote the set of all attributes with numeric domains by $\mathcal{N} \subset \mathcal{B}$. All other attributes and domains are considered *non-numeric*.[6] It is worth noting that not all numeric data necessarily constitute a numeric domain. Consider, for example, a customer relation R whose primary key is a customer number. Although the customer number consists of numeric symbols, it is essentially an arbitrary identification string for which arithmetic operations like addition and subtraction are not intrinsically meaningful for the database application. We consider such attributes to be non-numeric.

3.2.1 Domain lumping

To make our data modification model more computationally tractable, it may be appropriate, in many cases, to simplify the Markov chain state space for an attribute A so that it is much smaller

---

[5]$\nu = 3$ for the standard method and $\nu = \log_2 7$ for Strassen's and related methods.

[6]Distance metrics can also be defined for complex data types such as images. We leave the handling of such cases to further research.

than dom A. Suppose, for example, that A is a 64-character string representing a street address. Restricting to 96 printable characters, A may assume on the order of $96^{64} \approx 10^{126}$ possible values. It is obviously unnecessary, inappropriate, and intractable to work with a Markov chain with such an astronomical number of states.

One possible remedy for such situations is referred to as *lumping* in the Markov chain literature [17]. In our terminology, suppose we can partition dom A into a collection of sets $\{V\}_{V \in \mathcal{V}}$ with the property that $|\mathcal{V}| \ll |\text{dom } A|$ and

$$\forall \, U, V \in \mathcal{V}, \, \forall \, u, u' \in U \quad \sum_{v \in V} q_{u,v}^{R,A} = \sum_{v \in V} q_{u',v}^{R,A}.$$

Then, one can model the transitions between the "lumps" $V \in \mathcal{V}$ as a much smaller Markov chain whose set of states is $\mathcal{V}$, with the transition rate from $U \in \mathcal{V}$ to $V \in \mathcal{V}$ being given by the common value of $\sum_{v \in V} q_{u,v}^{R,A}$, $u \in U$. If we are interested only in which lump the attribute is in, rather than its precise value, this smaller chain will suffice. Using lumping, the complexity of the computation is directly dependent on the number of lumps. We now give a few simple examples:

Example 8 (Lumping into a binary domain) *Consider the street address example just discussed. Fortunately, if an address has changed since time s, the database user is unlikely to be concerned with how different it is from the address at time s, but simply whether it is different. Thus, instead of modeling the full domain $\text{dom } A$, we can represent the domain via the simple binary set $\{0, 1\}$, where 0 indicates that the address has not changed since time s, and 1 indicates that it has. We assume that the exit rates $q_{v,r.A(s)}^{R,A}$ from all other addresses $v \in \text{dom } A$ back to the original value $r.A(s)$ all have the identical value θ'. In this case, one has $P_{0,1}^{R,A} = P_{1,0}^{R,A} = 1$, and the behavior of the attribute is fully captured by the exit rates $\ell_0^{R,A} = q_{0,1}^{R,A}$ and $\ell_1^{R,A} = q_{1,0}^{R,A}$. We will abbreviate these quantities by θ and θ', respectively.*

Using standard results for a two-state continuous-time Markov chain [35, Section VI.3.3], we conclude that

$$P_{0,0}^{R,A}(s, f) = \frac{\theta' + \theta e^{-(\theta+\theta')\Gamma_{R,A}(s,f)}}{\theta + \theta'} \tag{8}$$

$$P_{0,1}^{R,A}(s, f) = \frac{\theta - \theta e^{-(\theta+\theta')\Gamma_{R,A}(s,f)}}{\theta + \theta'}. \tag{9}$$

$\square$

Example 9 (Web crawling) *As an even simpler special case, consider a Web crawler (e.g., [26, 15, 7]). Such a crawler needs to visit Web pages upon change to re-process their content, possibly for the use of a search engine. Recalling Example 8, one may define a boolean attribute* `Modified` *in a relation that collects information on Web pages.* `Modified` *is set to* `True` *once the page has changed, and back to* `False` *once the Web crawler has visit the page. Therefore, once a page has been modified to* `True`*, it cannot be modified back to* `False` *before the next visit of the Web crawler. In the analysis of Example 8, one can set $\theta' = 0$, resulting in $P_{0,0}^{A,R}(s, f) = e^{-\theta\Gamma_{R,A}(s,f)}$ and $P_{0,1}^{A,R}(s, f) = 1 - e^{-\theta\Gamma_{R,A}(s,f)}$.* $\square$

3.2.2 Random walks

Like large non-numeric domains, many numeric domains may also be cumbersome to model directly via Markov chain techniques. For example, a 32-bit integer attribute can, in theory, take $2^{32} \approx$

4×10^9 distinct values, and it would be virtually impossible to directly form, much less exponentiate, a full transition rate matrix for a Markov chain of this size.

Fortunately, it is likely that such attributes will have "structured" value transition patterns that can be modeled, or at least closely approximated, in a tractable way. As an example, we consider here a random walk model for numeric attributes.

In this case, we still suppose that the attribute A is modified only at transition event times that are distributed as described above. Letting t_i denote the time of transition event i, with $t_0 = s$, we suppose that at transition event i, the value of attribute A is modified according to

$$r.A(t_i) = r.A(t_{i-1}) + \Delta A_i,$$

where ΔA_i is a random variable. We suppose that the random variables $\{\Delta A_i\}$ are IID, that is, they are independent and share a common distribution with mean δ and variance σ^2. Defining

$$\Delta A(s, f) = \sum_{i:t_i \in (s,f]} \Delta A_i,$$

we obtain that $\{\Delta A(s, f), f \geq s\}$ is a nonhomogeneous compound Poisson process, and $r.A(f) = r.A(s) + \Delta A(s, f)$. From standard results for compound Poisson processes, we then obtain for each tuple $r \in R(s)$ that $\mathrm{E}[r.A(f)] = r.A(s) + \Gamma_{R,A}(s, f)\delta$.

It should be stressed that such a model must ultimately be only an approximation, since a random walk model of this kind would, strictly speaking, require an infinite number of possible states, while $\mathrm{dom}\, A$ is necessarily finite for any real database. However, we still expect it to be accurate and useful in many situations, such as when $r.A(s)$ and $\mathrm{E}[r.A(f)]$ are both far from largest and smallest possible values in $\mathrm{dom}\, A$.

3.2.3 Content-independent overwrites

Consider the simple case in which $P_{u,v}^{R,A}(s, f)$ is independent of u once a transition event has occurred. Let $\mathcal{A} \subseteq \mathcal{C}(R)$ be a set of attributes A with identical $\gamma_{R,A}$ functions, and let $\Gamma_{R,\mathcal{A}}(s, t) = \Gamma_{R,A}(s, t)$ for any $A \in \mathcal{A}$. We define a probability distribution $\omega_{R,\mathcal{A}}$ over $\mathrm{dom}\,\mathcal{A}$, and assume that at each transition event, a new value for $\mathcal{A}$ is selected at random from this distribution, without regard to the prior value of $r.\mathcal{A}$. It is thus possible that a transition event will leave $r.\mathcal{A}$ unchanged, since the value selected may be the same one already stored in r. For any tuple $r \in R(s) \cap R(f)$ and $u \in \mathrm{dom}\,\mathcal{A}$, we thus compute the probability $P_{u,u}^{R,\mathcal{A}}(s, f)$ that the value of $r.\mathcal{A}$ remains unchanged at u at time f to be

$$
\begin{aligned}
P_{u,u}^{R,\mathcal{A}}(s, f) &= \mathrm{P}\{\tau_u^{R,\mathcal{A}} > f - s\} + \mathrm{P}\{\tau_u^{R,\mathcal{A}} \leq f - s\}\,\omega_{R,\mathcal{A}}(u) \\
&= e^{-\ell_u^{R,\mathcal{A}}\Gamma_{R,\mathcal{A}}(s,f)} + \left(1 - e^{-\ell_u^{R,\mathcal{A}}\Gamma_{R,\mathcal{A}}(s,f)}\right)\omega_{R,\mathcal{A}}(u) \\
&= e^{-\ell_u^{R,\mathcal{A}}\Gamma_{R,\mathcal{A}}(s,f)}\left(1 - \omega_{R,\mathcal{A}}(u)\right) + \omega_{R,\mathcal{A}}(u)
\end{aligned}
$$

For $u, v \in \mathrm{dom}\,\mathcal{A}$ such that $u \neq v$, we also compute the probability $P_{u,v}^{R,\mathcal{A}}(s, f)$ that $r.\mathcal{A}$ changes from u to v in $[s, f)$ to be

$$
\begin{aligned}
P_{u,v}^{R,\mathcal{A}}(s, f) &= \mathrm{P}\{\tau_u^{R,\mathcal{A}} \leq f - s\}\,\omega_{R,\mathcal{A}}(v) \\
&= \left(1 - e^{-\ell_u^{R,\mathcal{A}}\Gamma_{R,\mathcal{A}}(s,f)}\right)\omega_{R,\mathcal{A}}(v).
\end{aligned}
$$

Content-independent overwrites are a special case of the Markov chain model discussed above. To apply the general model formulae when content-independent updates are present, each $\ell_v^{R,A}$ is multiplied by $1 - \omega_{R,A}(v)$ and $P_{u,v}^{R,A} = \omega_{R,A}(v)/(1 - \omega_{R,A}(u))$ for all $u, v \in \mathrm{dom}\, A$, $u \neq v$.

3.3   Summary

In this section we have introduced a general Markov-chain model for data modification, and discussed three simplified models that allows tractable computation. Using these models, one can compute, in probabilistic terms, value histograms at time f, given a known initial set of value histograms at time $s < f$. Such a model could be useful in query optimization, whenever the continual gathering of statistics becomes impossible due to either heavy system loads or structural constraints (*e.g.*, federations of databases with autonomous DBMSs).

Generally speaking, computing the transition matrix for an attribute A involves repeated multiplications of square matrices of size $|\text{dom } A|$, resulting in a computational complexity of $\text{O}(n|\text{dom } A|^\nu)$, where n is the number of iterations needed to compute the sum (6) to machine precision. While n is usually small, $|\text{dom } A|$ may be very large, as demonstrated in Section 3.2.1 and Section 3.2.2. Methods such as domain lumping would require $\text{O}(nX^\nu)$ time, where $X \ll |\text{dom } A|$.[7] As for random walks and independent updates, both methods no longer require repeated matrix multiplications, but rather the computation of $\Gamma_{R,A}(s,f)$. The complexity of calculating $\Gamma_{R,A}(s,f)$ is similar to that for $\Lambda_R(s,f)$ in Section 2.1.1.

4   Insertion model verification

It is well-known that Poisson processes model a world where data updates are independent from one another. While in databases with widely distributed access, *e.g.*, incoming e-mails, postings to newsgroups, or posting of orders from independent customers, such an independence assumption seems plausible, we still need to validate the model against real data. In this section we shall present some initial experiments as a "proof of concept." These experiments deal only with the insertion component of the model. Further experiments, including modification and deletion operations, will be reported in future work.

Our data set is taken from postings to the DBWORLD electronic bulletin board. The data were collected over more than seven months and consists of about 750 insertions, from November 9th, 2000 through May 14th, 2001. Figure 2 illustrates a data set with 580 insertions during the interval [2000/11/9:00:00:00, 2001/3/31:00:00:00). We used the Figure 2 data as a *training set*, *i.e.*, it serves as our basis for parameter estimation. Later, in order to test the model, we applied these parameters to a separate *testing set* covering the period [2001/3/31:00:00:00, 2001/5/15:00:00:00). In the experiments described below, we tried fitting the training data with two insertion-only models, namely a homogeneous Poisson process and an RPC Poisson process (see Section 2.1). For each of these two models, we have applied two variations, either as a compound or as a non-compound model. In the experiments described below, we have used the Kolmogorov-Smirnov goodness of fit test (see for example [16, Section 7.7]). For completeness, we first overview the principles of this statistical test.

The Kolmogorov-Smirnov test evaluates the likelihood of a *null hypothesis* that a given sample may have been drawn from some hypothesized distribution. If the null hypothesis is true, and sample set has indeed been drawn from the hypothesized distribution, then the empirical cumulative distribution of the sample should be close to its theoretical counterpart. If the sample cumulative distribution is too far from the hypothesized distribution at any point, that suggests that the sample comes from a different distribution. Formally, suppose that the theoretical distribution is $F(x)$, and we have n sample values $x_1, ..., x_n$ in nondecreasing order. We define an empirical cumulative

---

[7]Here, n may also be affected by the change of domain.

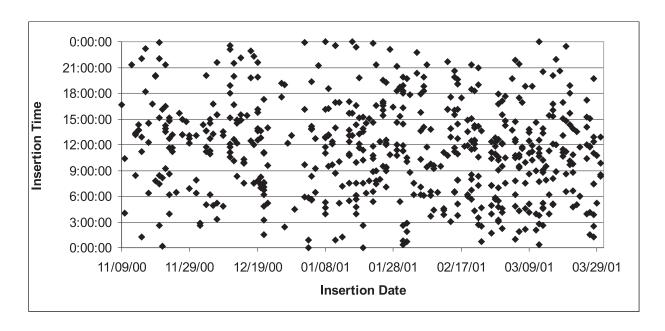Figure 2: Training data set.

distribution $F_n(x)$ via

$$
F_n(x) = \begin{cases} 0, & \text{if } x < x_1 \\ \frac{k}{n}, & \text{if } x_k \le x < x_{k+1} \\ 1, & \text{if } x > x_n, \end{cases}
$$

and then compute $D_n = \sup_{k=1,\dots,n}\{|F_n(x_k) - F(x_k)|\}$. For large n, given a significance level α, the test measures D_n against $X(\alpha)/\sqrt{n}$, where $X(\alpha)$ is a factor depending on the *significance level* α at which we reject the null hypothesis. For example, $X(0.05) = 1.36$ and $X(0.1) = 1.22$. The value of α is the probability of a "false negative," that is, the chance that the null hypothesis might be rejected when it is actually true. Larger values of α make the test harder to pass.

4.1 Fitting the homogeneous Poisson process

Based on the training set, we computed the parameter for a homogeneous Poisson process by averaging the 580 interarrival times, an unbiased estimator of the Poisson process parameter. The average interarrival time was computed to be 5:15:19, and thus $\lambda = 4.57$ per day. Figure 3(a) provides a pictorial comparison of the cumulative distribution functions of the interarrival times with their theoretical counterpart. We applied the Kolmogorov-Smirnov test to the distribution of interarrival times, comparing it with an exponential distribution with a parameter of $\lambda = 4.57$. The outcome of the test is $D_n = 0.106$, which means we can reject the null hypothesis at any reasonable level of confidence $\alpha \ge 0.005$ (for $\alpha = 0.005$, the rejection threshold is 0.0718 for $n = 580$). In all likelihood, then, the data are not derived from a homogeneous Poisson process.

Next, we have applied a compound homogeneous Poisson model. Our rationale in this case is that DBWORLD is a moderated list, and the moderators sometimes work on postings in batches. These batches are sometimes posted to the group in tightly-spaced clusters. For all practical purposes, we treat each such cluster as a single batch insertion event. To construct the model, any two insertions occurring within less than one minute from one another were considered to
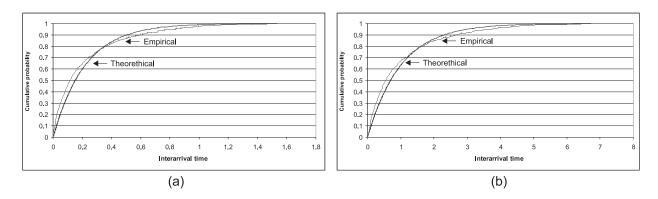
23

Figure 3: A comparison of a theoretical and empirical distribution functions for the homogeneous Poisson process model (a) and the compound homogeneous Poisson process model (b).

	Workdays	**Saturday**	**Sunday**
[0:00, 3:00)	2.40		
[3:00, 6:00)	5.96		
[6:00, 9:00)	6.04		
[9:00, 18:00)	7.50	1.50	1.15
[18:00, 21:00)	3.03		
[21:00, 24:00)	2.41		

Table 2: Average λ levels for the recurrent piecewise-constant Poisson model.

be a single event occurring at the insertion time of the first arrival. For example, on November 14, 2000, we had three arrivals, one at 13:43:19, and two more at 13:43:23. All three arrivals are considered to occur at the same insertion arrival event, with an insertion time of 13:43:19. Using the compound variation, the data set now has 557 insertion events. The revised average interarrival time is now 5:28:20, and thus $\lambda = 4.39$ per day. Figure 3(b) provides a pictorial comparison of the cumulative distribution functions of the interarrival times, assuming a compound model, with their theoretical counterpart. We have applied the Kolmogorov-Smirnov test to the distribution of interarrival times, comparing it with an exponential distribution with a parameter of $\lambda = 4.39$. The outcome was somewhat better than before. $D_n = 0.094$, which means we can still reject the null hypothesis at any level of confidence $\alpha \geq 0.005$ (for $\alpha = 0.005$, the rejection threshold is 0.0733 for $n = 557$). Although the compound variant of the model fits the data better, it is still not statistically plausible.

4.2 Fitting the RPC Poisson process

Next, we tried fitting the data to an RPC model. Examining the data, we chose a cycle of one week. Within each week, we used the same pattern for each weekday, with one interval for work hours (9:00-18:00), plus five additional three-hour intervals for "off hours". We treated Saturday and Sunday each as one long interval. Table 2 shows the arrival rate parameters for each segment of the RPC Poisson model, calculated in much the same manner as the for the homogeneous Poisson model.
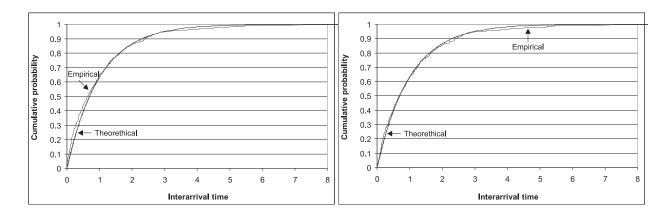
Figure 4: A comparison of a theoretical and empirical distribution functions of U for the RPC Poisson model (a) and the compound RPC Poisson model (b).

The specific methodology for structuring the RPC Poisson model is beyond the scope of this paper and can range from *ad hoc* "look and feel" crafting (as practiced here) to more established formal processes for statistically segmenting, filtering, and aggregating intervals [34, 33]. It is worth noting, however, that from experimenting with different methods, we have found that the model is not sensitive to slight changes in the interval definitions. Also, the model we selected has only 8 segments, and thus only 8 parameters, so there is little danger of "overfitting" the training data set, which has over 500 observations.

Next, we attempted to statistically validate the RPC model. To this end, we use the following lemma:

Lemma 6 *Given a nonhomogeneous Poisson process with arrival intensity $\lambda(t)$, the random variable $U_s = \int_s^{s+L_{R,s}} \lambda(t)\,dt$ is of the distribution* $\mathrm{Exp}(1)$.

Proof. Let $f_s(t) = \Lambda(s, s+t)$, which is a monotonically nondecreasing function. From Lemma 1, $\mathrm{P}\{L_{R,s} < t\} = 1 - e^{f_s(t)}$ for all $t \geq 0$. We have $U_s = f_s(L_{R,s})$. By applying the monotonic function f_s to both sides of the inequality $L_{R,s} < t$, one has that $\mathrm{P}\{f_s(L_{R,s}) < f_s(t)\} = \mathrm{P}\{L_{R,s} < t\} = 1 - e^{f_s(t)}$ for all $t \geq 0$. Substituting in the definitions of U_s and $u = f_s(t)$, one then obtains $\mathrm{P}\{U_s < u\} = 1 - e^{-u}$ for all $u \geq 0$, and therefore $U_s \sim \mathrm{Exp}(1)$. ∎

Thus, given an instantaneous arrival rate $\lambda(t)$, and a sequence of observed arrival events $\{t_n\}_{n=0}^N$, we compute the set of values $u_n = \int_{t_{n-1}}^{t_n} \lambda(t)\,dt$, $n = 1, \dots, N$, and perform a Kolmogorov-Smirnov test of them versus the unit exponential distribution.

Figure 4(a) provides a comparison of the theoretical and empirical cumulative distribution of the random variable U. We applied the Kolmogorov-Smirnov test to U, comparing it with an exponential distribution with $\lambda = 1$, based on Lemma 6. The outcome of the test is $D_n = 0.080$, which is better than either homogeneous model, but is still rejected at any reasonable level of significance (recall that for $\alpha = 0.005$, the rejection threshold is again 0.0718 for $n = 580$).

Finally, we evaluated a compound version of the RPC model, combining successive postings separated by less than one minute. We kept the same segmentation as in Table 2, but recalculated the arrival intensities in each segment, as shown in Table 3.

Next we recalculated the sample of the random variable U for the compound RPC Poisson model, and applied the Kolmogorov-Smirnov test. In this case, we have $D_n = 0.050$, which cannot

	Workdays	Saturday	Sunday
$[0{:}00, 3{:}00)$	2.40		
$[3{:}00, 6{:}00)$	5.96		
$[6{:}00, 9{:}00)$	5.59		
$[9{:}00, 18{:}00)$	7.11	1.45	1.15
$[18{:}00, 21{:}00)$	3.03		
$[21{:}00, 24{:}00)$	2.33		

Table 3: Average λ levels for the compound RPC Poisson model.

Model	D_n	Rejection level
Homogeneous	0.106	< 0.005
Homogeneous+compound	0.094	< 0.005
RPC	0.080	< 0.005
RPC+compound	0.050	> 0.100

Table 4: Goodness of fit of the four models.

be rejected at any reasonable confidence level through $\alpha = 0.10$ (for $\alpha = 0.10$, the rejection threshold is 0.0517 for $n = 557$). Figure 4(b) shows the theoretical and empirical distributions of U in this case.

As a final confirmation of the applicability of the compound RPC Poisson model, we attempted to validate the assumption that the number of postings in successive insertion events are independent and identically distributed (IID). In the sample, 536 insertion events were of size 1, 19 were of size 2, and 2 were of size 3. Thus, we approximate the random variable Δ_R^+ as having a $536/557 \approx .962$ probability of being 1, a $19/557 \approx .034$ probability of being 2, and a $2/557 \approx .004$ probability of being 3. Validating that the observed insertion batch sizes $\Delta_{R,i}^+$ appear to be independently drawn from this distribution is somewhat delicate, since they nearly always take the value 1. To compensate, we performed our test on the *runs* in the sample, that is, the number of consecutive insertion events of size 1 between insertions of size 2 or 3. Our sample contains 21 runs, ranging from 0 to 112. If the insertion batch sizes $\{\Delta_{R,i}^+\}$ are independent with the distribution Δ_R^+, then the length of a run should be a geometric random variable with parameter $536/557 \approx .962$. We tested this hypothesis via a Kolmogorov-Smirnov test, as shown in Figure 5. The D_n statistic is 0.207, which is within the $\alpha = 0.1$ acceptance level for a sample of size $n = 21$ (although the divergence of the theoretical and empirical curves in Figure 5 is more visually pronounced than in the prior figures, it should be remembered that the sample is far smaller). Thus, the assumption that the insertion batch sizes $\{\Delta_{R,i}^+\}$ are IID is plausible.

Table 4 compares the goodness-of-fit of the four models to the test data. For each of the models, we have specified the KS test result (D_n) and the level at which one can reject the null hypothesis. The higher the level of confidence is, the better the fit is. The RPC compound Poisson model models best the data set, accepting the null hypothesis at any level up to 0.1 (which practically means that the model can fit to the data well). The main conclusion from these experiments is that the simple model of homogeneous Poisson process is limited to the modeling of a restricted class of applications (one of which was suggested in [7]). Therefore, there is a need for a more elaborate model, as suggested in this paper, to capture a broader range of update behaviors. A
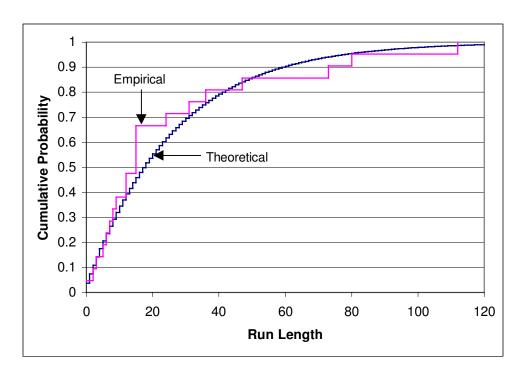
Figure 5: Empirical and theoretical distributions for number of single arrivals between multiple arrivals, compound RCP Poisson model.

nonhomogeneous model consisting of just 8 segments per week, as we have constructed, seems to model the arrivals significantly better than the homogeneous approach.

5   Content evolution cost model

We now develop a cost model suitable for transcription-scheduling applications such as those described in Example 2. The question is how often to generate a remote replica of a relation R. We have suggested one such policy in Example 6. In this section, we shall introduce two more policies and show an empirical comparison based on the data introduced in Section 4.

A transcription policy aims to minimize the combined cost of *transcription cost* and *obsolescence cost* [11]. The former includes the cost of connecting to a network and the cost of transcribing the data, and may depend on the time at which the transcription is performed (*e.g.*, as a function of network congestion), and the length of connection needed to perform the transcription. The obsolescence cost captures the cost of using obsolescent data, and is basically a function of the amount of time that has passed since the last transcription.

In what follows, let the set $\{b_i, e_i\}_{i=1}^{\infty}$ represents an infinite sequence of connectivity periods between a client and a server. During session i, the client data is synchronized with the state of the server at time b_i, the information becoming available at the client at time e_i. At the next session, beginning at time b_{i+1}, the client is updated with all the information arriving at the server during the interval $(b_i, b_{i+1}]$, which becomes usable at time e_{i+1}, and so forth. We define $b_0 = e_0 = 0$, and require that $0 < b_1 \leq e_1 < b_2 \leq e_2 < \ldots$.

Let $C_{R,\mathrm{u}}(s, f)$ denote the cost of performing a transcription of R starting at time f, given that the last update was started at time s. Let $C_{R,\mathrm{o}}(s, f)$, to be described in more detail later, denote

the obsolescence cost through time f attributable to tuples inserted into R at the server during the time interval $(s, f]$. Then the total cost $C_R(t)$ through time t is

$$C_R(t) = \sum_{i:b_i \leq t} \Big(\alpha C_{R,\mathrm{u}}(b_{i-1}, b_i) + (1 - \alpha) C_{R,\mathrm{o}}(b_{i-1}, b_i) \Big) + (1 - \alpha) C_{R,\mathrm{o}}(b_{i^*(t)}, t), \qquad (10)$$

where $i^*(t) = \max \{ i \mid b_i \leq t \}$ and α serves as the ratio of importance a user puts on the transcription cost versus the obsolescence cost. Traditionally, $\alpha = 0$, and therefore $C_R(t)$ is minimized for $C_{R,\mathrm{o}}(b_{i-1}, b_i) = 0$, $\forall b_i < t$, allowing the use of current data only. In this section we shall look into another, more realistic approach, where data currency is sacrificed (up to a level defined by the user through α) for the sake of reducing the transcription cost. Ideally, one would want to choose the sequence $\{b_i, e_i\}_{i=1}^{\infty}$ of connectivity periods, subject to any constraints on their durations $e_i - b_i$, to minimize $C_R(t)$ over some time horizon t. One may also consider the asymptotic problem of minimizing the average cost over time, $\lim_{t \to \infty} C_R(t)/t$. We note that the presence of α is not strictly required, as its effects could be subsumed into the definitions of the $C_{R,\mathrm{u}}$ and $C_{R,\mathrm{o}}$ functions, especially if both are expressed in natural monetary units. However, we retain α in order to demonstrate some of the parametric properties of our model.

In general, modeling transcription and obsolescence costs may be difficult and application-dependent. They may be difficult to quantify and difficult to convert to a common set of units, such as dollars or seconds. Some subjective estimation may be needed, especially for the obsolescence costs. However, we maintain that, rather than avoiding the subject altogether, it is best to try construct these cost models and then use them, perhaps parametrically, to evaluate transcription policies. Any transcription policy implicitly makes some trade-off between consuming network resources and incurring obsolescence, so it is best to try quantify the trade-off and see if a better policy exists. In particular, one should try to avoid policies that are clearly *dominated*, meaning that there is another policy with the same or lower transcription cost, and strictly lower obsolescence, or *vice versa*. Below, for purposes of illustration, we will give one simple, plausible way in which the cost functions may be constructed; alternatives are left to future research.

5.1 Transcription costing example

In determining the transcription cost, one may use existing research into costs of distributed query execution strategies. Typically, (*e.g.*, [21]) the transcription time can be computed as some function of the CPU and I/O time for writing the new tuples onto the client and the cost of transmitting the tuples over a network. There is also some fixed setup time to establish the connection, which can be substantial. For purposes of example, suppose that

$$
\begin{aligned}
C_{R,\mathrm{u}}(s, f) &= c + \beta \cdot \big(X_R(s, f) + Y_R^+(s, f) + |R(s)| - Y_R(s, f) \big) \\
&= c + \beta \cdot \big(X_R(s, f) + |R(s)| - Y_R^-(s, f) \big)
\end{aligned}
$$

Here, $c \geq 0$ denotes the fixed setup cost, $\beta \geq 0$, $X_R(s, f)$ denotes the number of tuples inserted during the interval $(s, f]$ that survive through time f, $Y_R^+(s, f)$ is the number of tuples that survive but are modified, by time f, and $|R(s)| - Y_R(s, f)$ is the number of deleted tuples. For the latter, it may suffice to transmit only the primary key of each deleted tuple, incurring a unit cost of less than β. For sake of simplicity, however, we use the same cost factor β for deletion, insertion, and modification. We note that, under this assumption,

$$\sum_{i:b_i \leq t} C_{R,\mathrm{u}}(b_{i-1}, b_i) = n(t)c + \beta |R(s)| + \beta \sum_{i:b_i \leq t} \big(X_R(b_{i-1}, b_i) - Y_R^-(b_{i-1}, b_i) \big),$$

28

where $n(t)$ is the number of transcriptions in the interval $[0, t]$. For the special case that there are no deletions or modifications, $\beta |R(s)| + \beta \left(X_R(s, f) - Y_R^-(s, f) \right) = \beta B(s, f)$ and

$$\sum_{i:b_i \leq t} C_{R,\text{u}}(b_{i-1}, b_i) = n(T)c + \beta B(0, b_{i^*(T)}).$$

For large t, one would expect the $\beta B(0, b_{i^*(t)})$ term to be roughly comparable across most reasonable polices, whereas the $n(t)c$ term may vary widely for any value of t. It is worth noting that c and β could be generalized to vary with time or other factors. For example, due to network congestion, certain times of day may have higher unit transcription costs than others. Also, transcribing via airline-seat telephone costs substantially more than connecting via a cellular phone. For simplicity, we have refrained from discussing such variations in the transcription cost.

5.2   Obsolescence costing example

We next turn our attention to the obsolescence cost, which is clearly a function of the update time of tuples and the time they were transcribed to the client. Intuitively, the shorter the time between the update of a tuple and its transcription to the client, the better off the client would be. As a basis for the obsolescence cost, we suggest a criterion that takes into account user preferences, as well as the content evolution parameters. For any relation R, times $s < f$, and tuple $r \in R(s) \cup R(f)$, let $b(r)$ and $d(r)$ denote the time r was inserted into and deleted from R, respectively. We let $\iota_r(s, f)$ be some function denoting the contribution of tuple r to the obsolescence cost over $(s, f]$; we will give some more specific example forms of this function later. We then make the following definition:

Definition 3 *The total* obsolescence cost *of a relation R over the time interval $(s, f]$ (annotated $C_{R,\text{o}}(s, f)$) is defined to be $C_{R,\text{o}}(s, f) \triangleq \sum_{r \in R(s) \cup R(f)} \iota_r(s, f)$* □

Our principal concern is with the *expected* obsolescence cost, that is, the expected value of $C_{R,\text{o}}(s, f)$,

$$\text{E}[C_{R,\text{o}}(s, f)] = \text{E}\left[\sum_{r \in R(s) \cup R(f)} \iota_r(s, f) \right].$$

To compute $\text{E}[C_{R,\text{o}}(s, f)]$, we note that

$$\text{E}[C_{R,\text{o}}(s, f)] = \text{E}\left[\sum_{r \in R(s) \cap R(f)} \iota_r(s, f) \right] + \text{E}\left[\sum_{r \in R(s) \setminus R(f)} \iota_r(s, f) \right] + \text{E}\left[\sum_{r \in R(f) \setminus R(s)} \iota_r(s, f) \right].$$

The three terms in the last expression represent potentially modified tuples, deleted tuples, and inserted tuples, respectively. We denote these three terms by $\hat{\iota}_R^{\text{M}}(s, f)$, $\hat{\iota}_R^{\text{D}}(s, f)$, and $\hat{\iota}_R^{\text{I}}(s, f)$, respectively, whence

$$\text{E}[C_{R,\text{o}}(s, f)] = \hat{\iota}_R^{\text{M}}(s, f) + \hat{\iota}_R^{\text{D}}(s, f) + \hat{\iota}_R^{\text{I}}(s, f).$$

5.3 Obsolescence for insertions

We will now consider a specific metric for computing the obsolescence stemming from insertions in $(s, f]$, as follows:

$$\iota_r^{\mathrm{I}}(s, f) = \begin{cases} g^{\mathrm{I}}(s, f, b(r)) & s < b(r) \leq f < d(r) \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where $g^{\mathrm{I}}(s, f, t)$ is some application-dependent function representing the level of importance a user assigns, over the interval $(s, f]$, to a tuple arriving at a time t. For example, in an e-mail transcription application, a user may attach greater importance to messages arriving during official work hours, and a lesser measure of importance to non-work hours (since no one expects her to be available at those times). Thus, one might define

$$g^{\mathrm{I}}(s, f, t) = \int_t^f a(\tau) \, d\tau, \quad \text{where } a(\tau) = \begin{cases} a_1, & \text{if } \tau \text{ is during work hours} \\ a_2, & \text{if } \tau \text{ is after hours,} \end{cases} \tag{12}$$

and $a_1 \geq a_2$. For $a_1 = a_2 = 1$, $g^{\mathrm{I}}(s, f, t)$ takes a form resembling the age of a local element in [7]. More complex forms of $g^{\mathrm{I}}(s, f, t)$ are certainly possible. In this simple case, we refer to a_1/a_2 as the *preference ratio*.

Using the properties of nonhomogeneous Poisson processes, we calculate

$$\begin{aligned} \hat{\iota}_R^{\mathrm{I}}(s, f) &= \mathrm{E}\left[\sum_{r \in R(f) \backslash R(s)} \iota_r(s, f) \right] \\ &= \mathrm{E}[X_R(s, f)] \cdot \mathrm{E}\big[f(s, f, b(r)) \mid s < b(r) \leq f < d(r) \big] \\ &= \widetilde{\Lambda}_R(s, f) \, \mathrm{E}\big[\Delta_R^+\big] \int_s^f \frac{\lambda_R(t')}{\widetilde{\Lambda}_R(s, f)} g^{\mathrm{I}}(s, f, t') dt' \\ &= \mathrm{E}\big[\Delta_R^+\big] \int_s^f \lambda_R(t') g^{\mathrm{I}}(s, f, t') dt'. \end{aligned}$$

Example 10 (Transcription policies using the expected obsolescence cost) *Consider the insertion-only data set of Section 4. Figure 6 compares two transcription policies for the week $[2001/4/2{:}0{:}00, 2001/4/8{:}0{:}00)$. The transcription policy in Figure 6(a) (referred to below as the uniform synchronization point — USP — policy) was suggested in [7]. According to this policy, the intervals $(s, f]$ are always of the same size. The decision regarding the interval size $f - s$ may be either arbitrary (e.g., once a day) or may depend on λ, the Poisson model parameter (in which case a homogeneous Poisson process is implicitly assumed). The policy may be expressed as $f = s + M/\lambda$ for some multiplier $M > 0$. According to this policy with $M = 1$ (as suggested in [7]), and $\lambda = 4.57$ per day as computed from the training data. Therefore, one would refresh the database every 5:15:19. Figure 6(a) shows the transcription times resulting from the USP policy.*

Consider now another transcription policy, dubbed the threshold *policy. With this policy, given that the last connection started at time s, we transcribe at time f if the expected obsolescence cost from insertions ($\hat{\iota}_R^{\mathrm{I}}(s, f)$) exceeds Π, where Π is a threshold that measures the user's tolerance to obsolescent data. In comparing the two policies, one can compute Π, given M, as follows. Consider the homogeneous case where $\mathrm{E}\big[\Delta_R^+\big] = 1$ and $\lambda_R(t) = \lambda_R$ for all t. Assume further that $a_1 = a_2 = 1$ for all t. In this case,*

$$\hat{\iota}_R^{\mathrm{I}}(s, f) = \int_s^f \lambda_R \cdot (f - t) dt = \lambda_R \int_s^f (f - t) dt = \frac{1}{2} \lambda_R \cdot (f - s)^2$$
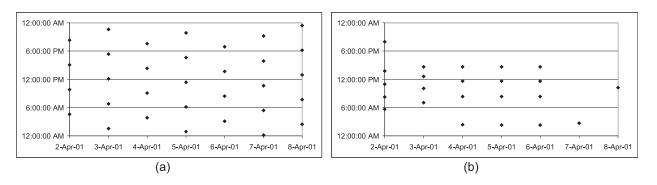
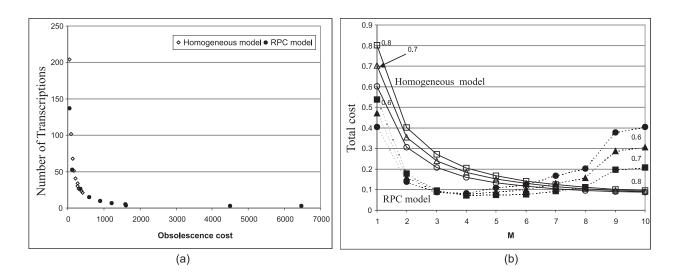Figure 6: Transcription times for the USP and RPC/threshold policies.

Figure 7: Threshold policy, homogeneous vs. RPC.

Setting $f = s + M/\lambda_R$ and $\Pi = \hat{\imath}_R^I(s, f)$, one has that

$$\Pi = (1/2)\lambda_R \cdot (f - s)^2 = (1/2)\lambda_R \cdot (M/\lambda_R)^2 = M^2/2\lambda_R.$$

Figure 6(b) shows the transcription times using the RPC arrival model (see Section 4.1) and the threshold policy with $\Pi = 0.109$ (obtained by setting $M = 1$ and $\lambda = 4.57$ per day, and letting $\Pi = M^2/2\lambda$). It is worth noting that transcriptions are more frequent when the λ intensity is higher and less frequent whenever the arrival rate is expected to be more sluggish.

We have performed experiments comparing the performance of the threshold policy for the homogeneous Poisson model (equivalent to the USP policy) and the RPC Poisson model. Figure 7 shows representative results, with costs computed over the testing set. Figure 7(a) displays the obsolescence cost and the number of transcriptions for various M values, with a preference ratio $a_2/a_1 = 4$. For all M values, there is no dominant model. For example, for $M = 1$, the RPC model has a slightly higher obsolescence cost (43.02 versus 42.35, a 1.6% increase) and a significantly lower number of transcriptions (137 versus 204, a 32.8% decrease).

Figure 7(b) provides a comparison of combined normalized obsolescence and transcription costs for both insertion models and $\alpha \in \{0.6, 0.7, 0.8\}$ (still assuming a 4:1 preference ratio). Solid lines represent results related with the homogeneous Poisson model, while dotted lines represent results
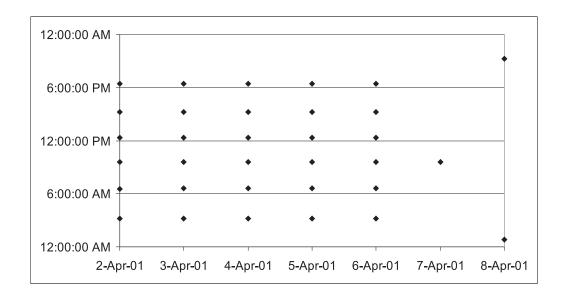
Figure 8: Transcription schedule based on the first alteration policy.

related with the RPC Poisson model. Generally speaking, the RPC model performs better for small M values ($M \leq 7$), while the homogeneous model performs better for the largest M values ($M \geq 8$).

□

Example 11 (Comparison of USP, threshold and FA policies) *Once again with the data from Section 4, we consider one more transcription policy, the first alteration (FA) policy derived from the analysis of Example 6. Since there are no deletions or modifications, $Z_R(s, f)$ simplifies to $\Lambda_R(s, f)$. We choose π in the FA policy to be a function of M such that the transcription intervals agree with the USP policy in the case of the homogeneous model. Figure 9 compares the performance of all three transcription policies: USP, threshold, and FA, for a 4:1 preference ratio and $\alpha = 0.8$, using the testing data set to compute the costs. For $M = 1$, the threshold policy and the FA policy perform similarly, where the FA policy performs slightly better than the Threshold policy. Both policies outperform the USP policy. The threshold policy is best for $M \in \{2 \ldots 8\}$. For all $M > 8$, the USP policy is best. The best policy for this choice of a_1/a_2 and α is threshold with $M = 6$, followed closely by FA with $M \in \{5, 6\}$. We have conducted our experiments with various α values and our conclusion is that the Threshold model is preferred over the USP model for larger α, that is, the more the user is willing to sacrifice currency for the sake of reducing transcription cost.*

□

5.4   Obsolescence for deletions

In a similar manner to Section 5.3, we will consider the following metric for computing the obsolescence stemming from deletions in $(s, f]$. We compute $\iota_r^{\mathrm{D}}(s, f)$ via

$$\iota_r^{\mathrm{D}}(s, f) = \begin{cases} g^{\mathrm{D}}(s, f, d(r)) & b(r) \leq s < d(r) \leq f \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where $g^{\mathrm{D}}(s, f, t)$ is some application-dependent function, possibly similar to $g^{\mathrm{I}}(s, f, t)$ above.
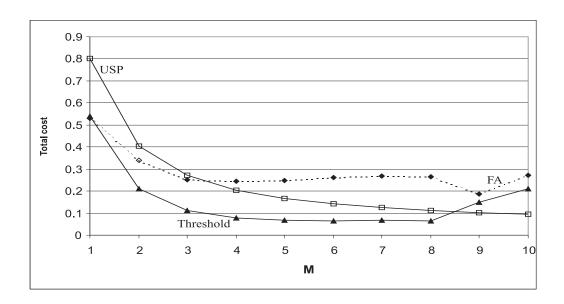
32

Figure 9: Comparison of three policies, for a 4:1 preference ratio and $\alpha = 0.5$.

Using the properties of nonhomogeneous Poisson processes, we calculate

$$
\begin{aligned}
\hat{\iota}_R^{\mathrm{D}}(s,f) &= \mathrm{E}\left[\sum_{r \in R(s) \setminus R(f)} \iota_r(s,f)\right] \\
&= (|R(s)| - \mathrm{E}[Y_R(s,f)])\,\mathrm{E}\big[g^{\mathrm{D}}(s,f,d(r)) \mid b(r) \le s < d(r) \le f\big] \\
&= (|R(s)| - p_R(s,f)\,|R(s)|)\,\mathrm{E}\big[g^{\mathrm{D}}(s,f,d(r)) \mid b(r) \le s < d(r) \le f\big] \\
&= |R(s)|\,(1 - p_R(s,f))\,\mathrm{E}\big[g^{\mathrm{D}}(s,f,d(r)) \mid b(r) \le s < d(r) \le f\big]
\end{aligned}
$$

In the case $\langle R, S \rangle$ has fixed multiplicity for all $S \in S(R)$, $p_R(s,f) = \exp(-\widetilde{M}_R(s,f))$, where $\widetilde{M}_R(s,f) = \int_s^f \tilde{\mu}_R(t)\,dt$ and $\tilde{\mu}_R(t) = \sum_{S \in S(R)} w(R,S)\mu_S(t)$. Therefore,

$$
\begin{aligned}
\hat{\iota}_{R,A}^{\mathrm{D}}(s,f) &= |R(s)|\left(1 - \exp(-\widetilde{M}_R(s,f))\right) \int_s^f \frac{\tilde{\mu}_R(t')}{\widetilde{M}_R(s,f)} g^{\mathrm{D}}(s,f,t')dt' \\
&= |R(s)|\left(\frac{1 - \exp(-\widetilde{M}_R(s,f))}{\widetilde{M}_R(s,f)}\right) \int_s^f \tilde{\mu}_R(t') \cdot g^{\mathrm{D}}(s,f,t')dt'
\end{aligned}
$$

5.5 Obsolescence for modification

We now consider obsolescence costs relating to modifications. While, in some applications, a user may be primarily concerned with how many tuples were modified during $[s,f)$, we believe that a more general, attribute-based framework is warranted here, taking into account exactly how each tuple was changed. Therefore, we define $\iota_{r,A}(s,f)$ to be some function denoting the contribution of attribute $A \in \mathcal{A}(R)$ in tuple r to the obsolescence cost over $(s,f]$ and assume that

$$
\iota_r(s,f) = \sum_{A \in \mathcal{A}(R)} \iota_{r,A}(s,f)
$$

33

Therefore,

$$
\hat{\iota}_R^{\mathrm{M}}(s, f) = \mathrm{E}\left[\sum_{r \in R(s) \cap R(f)} \iota_r(s, f)\right]
$$

$$
= \mathrm{E}\left[\sum_{A \in \mathcal{A}(R)} \sum_{r \in R(s) \cap R(f)} \iota_{r,A}(s, f)\right]
$$

$$
= \sum_{A \in \mathcal{A}(R)} \hat{\iota}_{R,A}^{\mathrm{M}}(s, f)
$$

where $\hat{\iota}_{R,A}^{\mathrm{M}}(s, f)$ is the expected obsolescence cost due to modifications to A during $(s, f]$. Assuming that attributes not in $\mathcal{C}(R)$ incur zero modification cost, the last sum may be taken over $\mathcal{C}(R)$ instead of $\mathcal{A}(R)$.

We start the section by introducing the notion of distance metric and provide two models of $\iota_{r,A}(s, f)$, for numeric and non-numeric domains. We then provide an explicit description of $\hat{\iota}_{R,A}^{\mathrm{M}}$, based on distance metrics.

5.5.1 General distance metrics

Let $c_{u,v}^{R,A}$, where $u, v \in \mathrm{dom}\, A$ denote the elements of a matrix of costs for an attribute A. We declare that if $r.A(s) = u$ and $r.A(f) = v$, then $\iota_{r,A}(s, f) = c_{u,v}^{R,A}$, or equivalently,

$$
\iota_{r,A}(s, f) = c_{r.A(s),r.A(f)}^{R,A}.
$$

Consequently, we require that $c_{u,u}^{R,A} = 0$ for all $u \in \mathrm{dom}\, A$, so that an unchanged attribute field yields a cost of zero.

A squared-error metric for numeric domains: For numeric domains, that is, $A \in \mathcal{N}$, we propose a squared-error metric, as is standard in statistical regression models. In this case, we let

$$
\iota_{r,A}(s, f) = c_{r.A(s),r.A(f)}^{R,A} = k_{R,A}(s)(r.A(f) - r.A(s))^2,
$$

where $k_{R,A}(s)$ is a user-specified scaling factor. A typical choice for the scaling factor would be the reciprocal $1/\left(\mathrm{Var}_{r \in R(s)}[r.A(s)]\right)$ of the variance of attribute A in R at time s,

$$
\mathrm{Var}_{r \in R(s)}[r.A(s)] = \mathrm{E}_{r \in R(s)}\left[\left(r.A(s) - \mathrm{E}_{r \in R(s)}[r.A(s)]\right)^2\right]
$$

$$
= \mathrm{E}_{r \in R(s)}\left[r.A(s)^2\right] - \mathrm{E}_{r \in R(s)}[r.A(s)]^2
$$

$$
= \frac{1}{|R(s)|}\left(\sum_{v \in \mathrm{dom}\, A} v^2 \hat{R}_{A,v}(s)\right) - \left(\frac{1}{|R(s)|}\sum_{v \in \mathrm{dom}\, A} v \hat{R}_{A,v}(s)\right)^2.
$$

Other choices for the scaling factor $k_{R,A}(s)$ are also possible. In any case, we may calculate the expected alteration cost for attribute A in tuple r via

$$
\mathrm{E}[\iota_{r,A}(s, f)] = \mathrm{E}\left[k_{R,A}(s)(r.A(f) - r.A(s))^2\right]
$$

$$
= k_{R,A}(s)\,\mathrm{E}\left[r.A(f)^2 - 2\,r.A(f)r.A(s) + r.A(s)^2\right]
$$

$$
= k_{R,A}(s)\left(\mathrm{E}\left[r.A(f)^2\right] - 2\,r.A(s)\,\mathrm{E}[r.A(f)] + r.A(s)^2\right). \tag{14}
$$

A general metric for non-numeric domains: For non-numeric domains, it may not be possible or meaningful to compute the difference of $r.A(s)$ and $r.A(f)$. In such cases, we shall use a general cost matrix $[c_{u,v}^{R,A}]_{u,v \in \text{dom } A}$ and compute

$$
\begin{aligned}
\mathrm{E}[\iota_{r,A}(s,f)] &= \sum_{v \in \text{dom } A} \left(P_{r.A(s),v}^{R,A}(s,f) \right) \left(c_{r.A(s),v}^{R,A} \right) \\
&= \sum_{\substack{v \in \text{dom } A \\ v \neq r.A(s)}} \left(P_{r.A(s),v}^{R,A}(s,f) \right) \left(c_{r.A(s),v}^{R,A} \right).
\end{aligned}
$$

For domains that have no particular structure, a typical choice might be $c_{u,v}^{R,A} = 1$ whenever $u \neq v$. In this case, the expected cost calculation simplifies to

$$
\begin{aligned}
\mathrm{E}[\iota_{r,A}(s,f)] &= \mathrm{P}\{r.A(f) \neq r.A(s)\} \\
&= 1 - P_{r.A(s),r.A(s)}^{A,R}(s,f).
\end{aligned}
$$

We are now ready to consider the calculation of $\hat{\iota}_{R,A}^{\mathrm{M}}(s,f)$.

5.5.2 The expected modification cost

We next consider computing the expected modification cost $\hat{\iota}_{R,A}^{\mathrm{M}}(s,f)$. To do so, we partition the tuples r in $R(s) \cap R(f)$ according to their initial value $r.A(s)$ of the attribute A. Consider the subset $R_{A,u}(s) \cap R(f)$ of all $r \in R(s) \cap R(f)$ that have $r.A(s) = u$. Since all such tuples are indistinguishable from the point of view of the modification process for (R, A), their $\iota_{r,A}(s,f)$ random variables will be identically distributed. The number of tuples $r \in R(s)$ with $r.A(s) = u$ is, by definition, $\hat{R}_{A,u}(s)$. The number $|R_{A,u}(s) \cap R(f)|$ that are also in $r.A(f)$ is a random variable whose expectation, by the independence of the deletion and modification processes, must be $p_R(s,f)\hat{R}_{A,u}(s)$. Using standard results for sums of random numbers of IID random variables, we conclude that

$$
\begin{aligned}
\hat{\iota}_{R,A}^{\mathrm{M}}(s,f) &= \mathrm{E}\left[\sum_{r \in R(s) \cap R(f)} \iota_{r,A}(s,f) \right] \\
&= \sum_{u \in \text{dom } A} \left(p_R(s,f)\hat{R}_{A,u}(s) \right) \mathrm{E}\left[\iota_{r,A}(s,f) \mid r.A(s) = u \right] \\
&= p_R(s,f) \sum_{\substack{u \in \text{dom } A \\ \hat{R}_{A,u}(s) > 0}} \hat{R}_{A,u}(s) \, \mathrm{E}\left[\iota_{r,A}(s,f) \mid r.A(s) = u \right] \\
&= p_R(s,f) \sum_{\substack{u \in \text{dom } A \\ \hat{R}_{A,u}(s) > 0}} \hat{R}_{A,u}(s) \hat{\iota}_{R,A,u}^{\mathrm{M}}(s,f),
\end{aligned}
$$

where we define $\hat{\iota}_{R,A,u}^{\mathrm{M}}(s,f) = \mathrm{E}\left[\iota_{r,A}(s,f) \mid r.A(s) = u \right]$. We now address the calculation of the $\hat{\iota}_{R,A,u}^{\mathrm{M}}(s,f)$.

For a non-numeric domain, we have from Section 5.5.1 that

$$
\hat{\iota}_{R,A,u}^{\mathrm{M}}(s,f) = \sum_{v \in \text{dom } A} \left(P_{u,v}^{R,A}(s,f) \right) \left(c_{u,v}^{R,A} \right),
$$

and in the simple case of $c_{u,v}^{R,A} = 1$ whenever $u \neq v$,

$$\hat{\imath}_{R,A,u}^{\mathrm{M}}(s,f) = 1 - P_{u,u}^{R,A}(s,f).$$

In any case, $P_{u,v}^{R,A}(s,f)$ and $P_{u,u}^{R,A}(s,f)$ may be computed using the results of Section 3.

For a numeric domain, we have from (14) that

$$\hat{\imath}_{R,A,u}^{\mathrm{M}}(s,f) = k_{R,A}(s)\left(\mathrm{E}\left[(r.A(f))^2 \mid r.A(s){=}u\right] - 2\,u\,\mathrm{E}\left[r.A(f) \mid r.A(s){=}u\right] + u^2\right)$$

$$= k_{R,A}(s)\left(\left(\sum_{v\in\mathrm{dom}\,A}(v^2 - 2uv)P_{u,v}^{R,A}(s,f)\right) + u^2\right).$$

In cases where a random walk approximation applies, however, the situation simplifies considerably, as demonstrated in the following proposition.

Proposition 7 *When a random walk model with mean δ and variance σ^2 accurately describes modifications to a numeric attribute A, $\hat{\imath}_{R,A,u}^{\mathrm{M}}(s,f) \approx k_{R,A}(s)\,\Gamma_{R,A}(s,f)\,\left(\sigma^2 + 2\,\Gamma_{R,A}(s,f)\delta^2\right).$*

Proof. In this case, we note that the random variable $r.A(f) - r.A(s)$ is identical to $\Delta A(s,f)$ (using the notation of section 3.2.2), and is independent of $r.A(s)$. The number N of modification events in $(s,f]$ has a Poisson distribution with mean $\Gamma_{R,A}(s,f)$, and hence variance $\Gamma_{R,A}(s,f)^2$. Therefore we have, for any $u \in \mathrm{dom}\,A$,

$$\hat{\imath}_{R,A,u}^{\mathrm{M}}(s,f) \approx k_{R,A}(s)\,\mathrm{E}\left[(\Delta A(s,f))^2\right]$$

$$= k_{R,A}(s)\left(\mathrm{Var}[\Delta A(s,f)] + \mathrm{E}[\Delta A(s,f)]^2\right)$$

$$= k_{R,A}(s)\left(\mathrm{E}[N]\,\sigma^2 + \delta^2\,\mathrm{Var}[N] + \mathrm{E}[N]^2\delta^2\right)$$

$$= k_{R,A}(s)\,\Gamma_{R,A}(s,f)\,\left(\sigma^2 + 2\,\Gamma_{R,A}(s,f)\delta^2\right).$$

$\blacksquare$

5.6   Example: the use of the cost model in Web crawling

The following example concludes the introduction of the cost function. We show how, by using the cost model, one can generate an optimal transcription policy for Web crawling.

Example 12 (Web Monitoring) *WebSQL [23] is a Web monitoring tool which uses a virtual database schema to query the structural properties of Web documents. The database schema consists of two relations, `Document` with six attributes, namely `url, title, text, type, length,` and `modif`, and `Anchor` with four attributes, namely `base, label, href`, and `context`. Each tuple in `Anchor` indicates that document `base` contains a link to document `href`. Consider the following query (taken from `http://www.cs.toronto.edu/~websql/`), which identifies locally reachable documents that contain some hyperlink to a compressed Postscript File:*

```
SELECT d.url, d.modif
FROM Document d SUCH THAT ''http://www.OtherDoc.html'' ->->* d,
     Anchor a SUCH THAT base = d
WHERE filename(a.href) CONTAINS ''.ps.Z'';
```

(We refrain from dwelling on the language specification;he interested reader is referred to the cited Web site.) Assume that the cost of performing the query at time t is $\sum_{d \in D(t)} \psi_d$, where $D(t)$ represents the set of scanned documents and ψ_d is a random variable representing the size of document d in bytes. Assuming the $\{\psi_d\}$ are IID, the expected cost of performing the query at time t is thus

$$\mathrm{E}\left[\sum_{d \in D(t)} \psi_d\right] = \mathrm{E}[|D(t)|]\,\mathrm{E}[\psi],$$

where ψ is a generic random variable distributed like the $\{\psi_d\}$.

A modification to a document is identified using changes to the `modif` attribute of the Document relation. For brevity in what follows, we let $R = $ `Document` and $A = $ `modif`. We assign the following costs to changes in A:

- $g^{\mathrm{D}}(s,f,t) = 0$ *for all $s < t < f$, that is, the user has no interest in being notified of deleted documents.*

- *For all $s < t < f$ and $u,v \in \mathrm{dom}\,A$, $u \neq v$, $c_{u,v}^{R,A} = g^{\mathrm{I}}(s,f,t) = \mathrm{E}[\psi]$, where $c_{R,A}^{\mathrm{M}}$ is the cost for a modified document. For all other attribute $A' \neq A$, $c_{u,v}^{R,A'} = 0$ for all $u,v \in \mathrm{dom}\,A'$.*

Suppose that a query was performed at time s, scanning the set of documents $D(s)$, and returning the set of documents $B(s)$, where $|B(s)| \leq |D(s)|$. A user is interested in refreshing the query result without overloading system resources, thus balancing the cost of refreshing the query results against the cost of using partial or obsolescent data. This trade-off can be captured by the following policy: refresh the query at time f, after performing it at time s iff

$$\mathrm{E}\left[\sum_{d \in D(f)} \psi_d\right] < \mathrm{E}[C_{R,\mathrm{o}}(s,f)]$$

Thus, an equivalent conditions is

$$\mathrm{E}[|D(f)|]\,\mathrm{E}[\psi] < \sum_{A' \in \mathcal{A}(R)} \hat{\iota}_{R,A'}^{\mathrm{M}}(s,f) + \hat{\iota}_R^{\mathrm{D}}(s,f) + \hat{\iota}_R^{\mathrm{I}}(s,f)$$
$$= \hat{\iota}_{R,A}^{\mathrm{M}}(s,f) + \hat{\iota}_R^{\mathrm{I}}(s,f),$$

or

$$\left(p_R(s,f)\,|D(s)| + \widetilde{\Lambda}_R(s,f)\,\mathrm{E}\big[\Delta_R^+\big]\right)\mathrm{E}[\psi]$$

$$< \left(p_R(s,f)\sum_{u \in \mathrm{dom}\,A} \hat{R}_{A,u}(s)(1 - P_{u,u}^{A,R}(s,f)) + \widetilde{\Lambda}_R(s,f)\,\mathrm{E}\big[\Delta_R^+\big]\,p^{\mathrm{I}}\right)\mathrm{E}[\psi],$$

where p^{I} is the probability of a newly-inserted document being relevant to the query. Cancelling the factor of $\mathrm{E}[\psi]$, another equivalent condition is

$$p_R(s,f)\,|D(s)| + \widetilde{\Lambda}_R(s,f)\,\mathrm{E}\big[\Delta_R^+\big] < p_R(s,f)\sum_{u \in \mathrm{dom}\,A} \hat{R}_{A,u}(s)(1 - P_{u,u}^{A,R}(s,f)) + \widetilde{\Lambda}_R(s,f)\,\mathrm{E}\big[\Delta_R^+\big]\,p^{\mathrm{I}},$$

which is independent of the expected document size. Further assume that $P_{u,u}^{RA}(s,f) = P_{*,*}^{R,A}(s,f)$ is independent of u. Then the refresh condition can be expressed as

$$p_R(s,f)D(s) + \widetilde{\Lambda}_R(s,f)\,\mathrm{E}\big[\Delta_R^+\big] < p_R(s,f)\,|B(s)|\,(1 - P_{*,*}^{A,R}(s,f)) + \widetilde{\Lambda}_R(s,f)\,\mathrm{E}\big[\Delta_R^+\big]\,p^{\mathrm{I}}$$

$\square$

6 Conclusion and topics for future research

This paper represents a first step in a new research area, the stochastic estimation of the consistency of transcribed data over time. We have also suggested one possible technique for assigning a cost to the differences between two relation extensions, including a means of computing the expected value of this cost under our stochastic model. We have discussed a number of potential applications relating managing replicas, query management, and Web crawling. We have also examined several strategies for refreshing replicas, although other strategies are certainly possible.

As an illustration of the low client-side computational demands of the insertion-only transcription application of our model, a Java-based demo, based on the transcription policies described in [12] and in this paper, can be accessed at `http://rbs.rutgers.edu:6677/`. The demo compares the performance of various policies using data that exist at a backend mSQL database.

We hope to extend our work to the case where the materialized views are not simple replications, but are produced by SQL queries that involve selections, projections, natural joins, and certain types of aggregations. This work will involve a *propagation algebra* for tracing the base data changes through a series of relational operators.

This development should make it possible to apply the theory to the management of more complex queries than presented here. In particular, it will facilitate a possible approach to managing general materialized view obsolescence on a query-by-query basis, taking into account current user preferences for query accuracy and speed. The refresh rate of materialized views in a periodically-updated data source (such as a data warehouse) can be defined in terms of data obsolescence, which in turn can be stochastically estimated using our model for content evolution. In this case, we advocate a three-way cost model for query optimization [11], in which the query optimizer evaluates various query plans using three complementary factors, namely *generation cost*, *transmission cost*, and *obsolescence cost*. The first two factors take on a conventional interpretation and the obsolescence cost of a query represents a penalty for basing the query result on possibly obsolescent materialized views. A query plan using only selection from a local materialized view, for example, might have lower generation and transmission costs, but a higher obsolescence cost, than a plan fetching complete base relations from an extranet and then processing them through a series of join operations. Our model, when combined with additional techniques to propagate updates through relational operators, can be used as a basis for estimating the obsolescence cost. However, developing the propagation algebra may require some enrichment of our basic model, in particular the introduction of dependency between the deletion and modification processes.

We foresee several additional future research directions. One direction involves the design of efficient algorithms for the numerical computations required by our model. As it stands so far, the most demanding computations required are general numerical integration and the matrix exponentiation formula (6). With regard to integration, we note that, in practice, the nonhomogeneous Poisson arrival rate functions $\lambda_R(\cdot)$, $\mu_R(\cdot)$, and $\gamma_{R,A}(\cdot)$ will most likely be chosen to be periodic piecewise low-order polynomials, as suggested in Section 4. In such cases, many of the integrals needed by the model could be performed in closed form within each time period.

Further calibration and verification of the models in real situations is also needed. So far, we have demonstrated that the insertion model has plausible applications, but this work needs to be extended to the deletion and modification models. Furthermore, the insertion model may need to be generalized to handle situations where there is "burstiness" or autocorrelation in the interarrival times that may require more involved techniques than simply combining very closely spaced arrivals.

Another future research direction involves applying the model to real-life settings such as managing a data warehouse. While the model is quite flexible, a methodology is still needed for structuring

Markov chains and estimating the stochastic model's parameters. Finally, in order to calibrate the cost model, the issue of measuring user tolerance for data obsolescence should be considered.

Acknowledgments

References

[1] S. Abiteboul. On views and XML. In *Proceedings of the 1999 ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Sys. (PODS)*, pages 1–9, 1999.

[2] S. Abiteboul and O.M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the 1998 ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Sys. (PODS)*, 1998.

[3] R. Alonso, D. Barbara, and H. Garcia-Molina. Data caching issues in information retrieval system. *ACM Transactions on Database Systems*, 15(3):359–384, September 1990.

[4] T.A. Anderson, Y. Breitbart, H.F. Korth, and A. Wool. Replication, consistency, and practicality: Are these mutually exclusive? In Laura M. Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 484–495. ACM Press, 1998.

[5] M.J. Carey, M.J. Franklin, M. Livny, and E.J. Shekita. Data caching tradeoffs in client-server dbms architectures. In J. Clifford and R. King, editors, *Proceedings of the 1991 ACM SIGMOD International Conference on Management of Data, Denver, Colorado, May 29-31, 1991*, pages 357–366. ACM Press, 1991.

[6] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proceedings of the 11th International Conference on Data Engineering*, pages 190–200, Taipei, Taiwan, 1995.

[7] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the 2000 ACM-SIGMOD conference on Management of Data*, pages 117–128, Dallas, Texas, May 2000.

[8] L.S. Colby, T. Griffin, L. Libkin, I.S. Mumick, and H. Trickey. Algorithms for deferred view maintenance. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 469–480. ACM Press, 1996.

[9] L.S. Colby, A. Kawaguchi, D.F. Lieuwen, I.S. Mumick, and K.A. Ross. Supporting multiple view maintenance policies. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 405–416. ACM Press, 1997.

[10] A. Delis and N. Roussopoulos. Techniques for update handling in the enhanced client-server dbms. *TKDE*, 10(3):458–476, 1998.

[11] A. Gal. Obsolescent materialized views in query processing of enterprise information systems. In *Proc. Eighth International Conference on Information and Knowledge Management (CIKM'99)*, pages 367–374, Kansas City, MI, 1999.

[12] A. Gal and J. Eckstein. Scheduling of data transcription in periodically connected databases. Technical Report 25-2001, Rutgers University, RUTCOR, Rutgers Center for Operations Research, February 2001.

[13] S. Grumbach and L. Tininini. On the content of materialized aggregate views. In *Proceedings of the 2000 ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Sys. (PODS)*, pages 47–57, 2000.

[14] P. Haas, J. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proceedings of the International Conference on VLDB*, pages 311–322, 1995.

[15] A. Heydon and M. Najork. Mercator: A scalable, extensible, Web crawler. In *Proceedings of the Eighth World-Wide Web Conference*, pages 219–229, 1999.

[16] R.V. Hogg and E.A. Tanis. *Probability and Statistical Inference*. MacMillan, New York, second edition, 1983.

[17] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Van Nostrand, 1960.

[18] L.V.S. Lakshmanan, N. Leone, R. Ross, and V.S. Subrahmanian. Probview: A flexible probabilistic database system. *TODS*, 22(3):419–469, 1997.

[19] A. Levy, A.O. Mendelzon, and Y. Sagiv. Answering queries using views. In *Proceedings of the 1998 ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Sys. (PODS)*, pages 95–104, San Jose, May 1995.

[20] B.G. Lindsay, L.M. Haas, C. Mohan, H. Pirahesh, and P.F. Wilms. A snapshot differential refresh algorithm. In Carlo Zaniolo, editor, *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 28-30, 1986*, pages 53–60. ACM Press, 1986.

[21] G. Lohman, C. Mohan, L. Haas, D. Daniels, B. Lindsay, P. Selinger, and P. Wilms. Query processing in R^*. In W. Kim, D. Reiner, and D. Batroy, editors, *Query Processing in Database Systems*. Springer-Verlag, NY, 1985.

[22] Y. Mattias, J.S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM-SIGMOD conference on Management of Data*, pages 448–459, 1998.

[23] A.O. Mendelzon and T. Milo. Formal models of the Web. In *Proceedings of the 1997 ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Sys. (PODS)*, pages 134–143, 1997.

[24] M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *Proc. First International Conference on Information and Knowledge Management*, 1992.

[25] C. Olston and J. Widom. Offering a precision-perfomrance tradeoff for aggregation queries over replicated data. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 144–155. Morgan Kaufmann, 2000.

[26] B. Pinkerton. Finding what people want: Experiences with the Web crawler. In *Proceedings of the Second World-Wide Web Conference*, 1994.

[27] L. Raschid, M.E. Vidal, and V. Zadoroshny. A strategy for query optimization for wide are applications. Technical report, University of Maryland, 2001.

[28] S. Ross. *Introduction to Probability Models*. Academic Press, 1980.

[29] S. Ross. *Stochastic Processes*. Wiley, second edition, 1995.

[30] M.T. Roth, M. Arya, L.M. Hass, M.J. Carey, W.F. Codey, R. Fagin, P.M. Schwartz, J. Tomas II, and E.L. Wimmers. The Garlic project. In *SIGMOD 1996, Proceedings ACM SIGMOD International Conference on Management of Data*. ACM Press, 1996.

[31] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[32] S. Shurety. *E-Business With Net.Commerce*. Prentice Hall, 1998.

[33] Z.G. Stoumbos. Economic statistical design of adaptive individuals control schemes for monitoring the process mean and variance. *to appear in Nonlinear Analysis*, 2002.

[34] Z.G. Stoumbos and M.R. Reynolds Jr. Control charts applying a sequential test at fixed sampling intervals. *Journal of Quality Technology*, 29:21–40, 1997.

[35] H.M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 1994.

[36] J.F. Traub and A.G. Werschulz. *Complexity and Information*. Cambridge University Press, 1998.

[37] K.-Y. Whang, B.T. Vander Zanden, and H.M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems (TODS)*, 15(2):208–229, 1990.

[38] H.Z. Yang and P.A. Larson. Query transformation for PSJ-queries. In *Proceedings of the 13th International VLDB Conference*, pages 245–254, 1987.