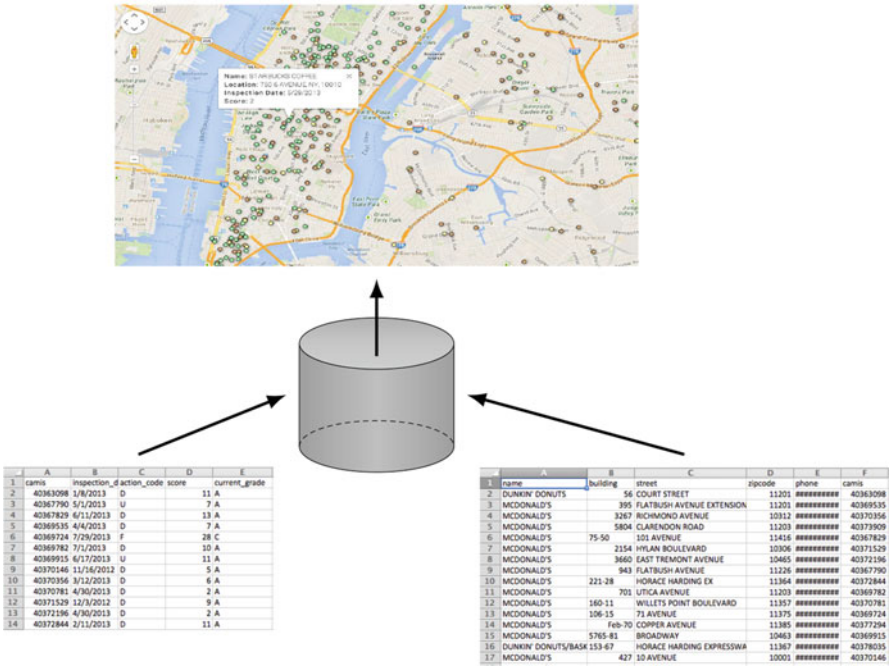

	A	B	C	D	E
1	camis	inspection_d	action_code	score	current_grade
2	40363098	1/8/2013	D	11	A
3	40367790	5/1/2013	U	7	A
4	40367829	6/11/2013	D	13	A
5	40369535	4/4/2013	D	7	A
6	40369724	7/29/2013	F	28	C
7	40369782	7/1/2013	D	10	A
8	40369915	6/17/2013	U	11	A
9	40370146	11/16/2012	D	5	A
10	40370356	3/12/2013	D	6	A
11	40370781	4/30/2013	D	2	A
12	40371529	12/3/2012	D	9	A
13	40372196	4/30/2013	D	2	A
14	40372844	2/11/2013	D	11	A

	A	B	C	D	E	F
1	name	building	street	zipcode	phone	camis
2	DUNKIN' DONUTS	56	COURT STREET	11201	##########	40363098
3	MCDONALD'S	395	FLATBUSH AVENUE EXTENSION	11201	##########	40369535
4	MCDONALD'S	3267	RICHMOND AVENUE	10312	##########	40370356
5	MCDONALD'S	5804	CLARENDON ROAD	11203	##########	40373909
6	MCDONALD'S	75-50	101 AVENUE	11416	##########	40367829
7	MCDONALD'S	2154	HYLAN BOULEVARD	10306	##########	40371529
8	MCDONALD'S	3660	EAST TREMONT AVENUE	10465	##########	40372196
9	MCDONALD'S	943	FLATBUSH AVENUE	11226	##########	40367790
10	MCDONALD'S	221-28	HORACE HARDING EX	11364	##########	40372844
11	MCDONALD'S	701	UTICA AVENUE	11203	##########	40369782
12	MCDONALD'S	160-11	WILLETS POINT BOULEVARD	11357	##########	40370781
13	MCDONALD'S	106-15	71 AVENUE	11375	##########	40369724
14	MCDONALD'S	Feb-70	COPPER AVENUE	11385	##########	40377294
15	MCDONALD'S	5765-81	BROADWAY	10463	##########	40369915
16	DUNKIN' DONUTS/BASK	153-67	HORACE HARDING EXPRESSWA	11367	##########	40378035
17	MCDONALD'S	427	10 AVENUE	10001	##########	40370146

Fig. 12.11 Web tables/fusion tables example

web tables. The fusion table's infrastructure can automatically discover the join attribute across tables and produce integration opportunities. An example is given in Fig. 12.11 which depicts two datasets contributed by two different owners, one on eateries and the other about the scores and grades given to these eateries as a result of inspection. The system would determine that the two datasets can be joined over a common attribute and provide integrated access. Although in this case both tables were contributed by users, in other cases one or both of the tables can be discovered from the web by using the techniques developed by web tables project.

12.6.2 Semantic Web and Linked Open Data

A fundamental contribution of the web is to produce a repository of *machine processable* data. Semantic!web aims to convert this data into *machine understandable* form by integrating both structured and unstructured data on the web and marking it up semantically. The original semantic web vision includes three components:

- Markup web data so that metadata is captured as annotations;
- Use ontologies for making different data collections understandable; and
- Use logic-based technologies to access both the metadata and the ontologies.

Linked Open Data (LOD) was introduced in 2006 as a clarification of this vision emphasizing the linkages among the data that is part of the semantic web. It set out guidelines for how data should be published on the web to achieve the vision of the semantic web. Thus, the semantic web is a web data integration vision realized through LOD. LOD requirements for publishing (and hence integrating) data on the web are based on four principles:

- All web resources (data) are locally identified by their URIs that serve as names;
- These names are accessible by HTTP;
- Information about web resources/entities are encoded as RDF (Resource Description Framework) triples. In other words, RDF is the semantic web data model (and we discuss it below);
- Connections among datasets are established by data links and publishers of datasets should establish these links so that more data is discoverable.

The LOD, therefore, generates a graph where the vertices are web resources and the edges are the relationships. A simplified form of "LOD graph" as of 2018 is shown in Fig. 12.12 where each vertex represents a dataset (not a web resource) categorized according to by color (e.g., publications, life sciences, social networking) and the size of each vertex represents its in-degree. At that time, LOD consisted of 1,234 datasets with 16,136 links.[5] We will come back to LOD and the LOD graph shortly.

Semantic!web consists of a number of technologies that build upon each other (Fig. 12.13). At the bottom layer, XML provides the language for writing structured web documents and exchanging them easily. On top of this is the RDF that, as noted above, establishes the data model. Although it is not necessary, if a schema over this data is specified, the RDF Schema provides the necessary primitives. Ontologies extend RDF schema with more powerful constructs to specify the relationships among web data. Finally, logic-based declarative rule languages allow applications to define their own rules.

In the remainder we discuss the technologies in the lower layers as these are the minimal requirements.

12.6.2.1 XML

The predominant encoding for web documents has been HTML (which stands for HyperText Markup Language). A web document encoded in HTML consists of *HTML elements* encapsulated by tags as discussed in Sect. 12.6.1 where we also presented approaches to discover structured data in HTML-encoded web documents and integrating them. As noted above, within the context of semantic web, the preferred representation for encoding and exchanging web documents is XML

[5] Statistics obtained from https://lod-cloud.net, which should be consulted for up-to-date statistics.

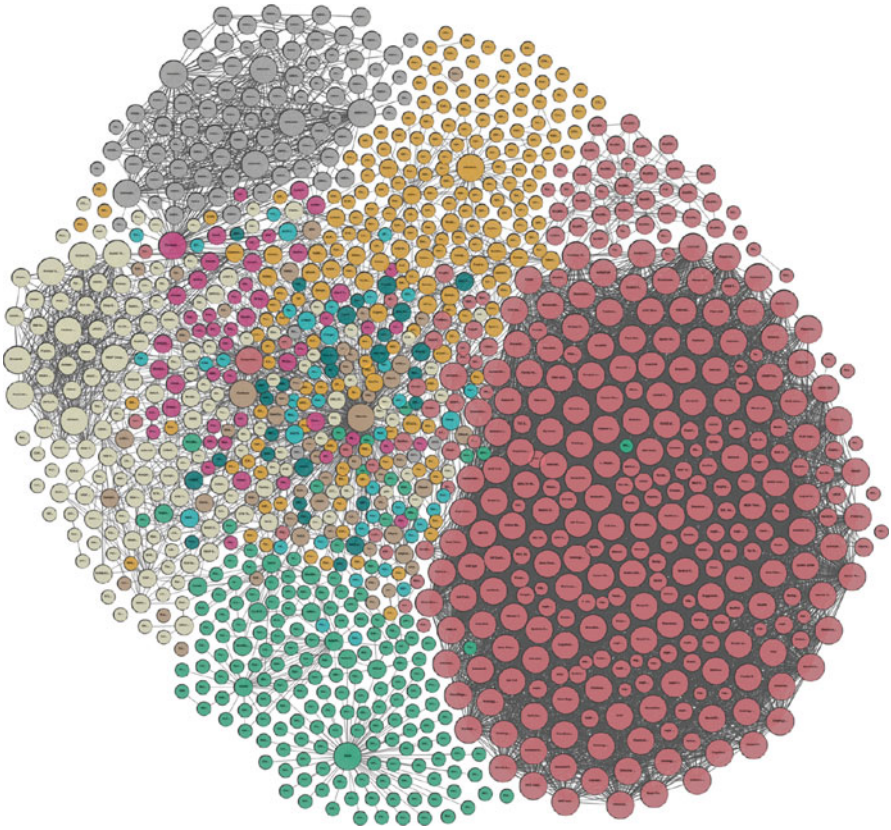

Fig. 12.12 LOD graph as of 2018

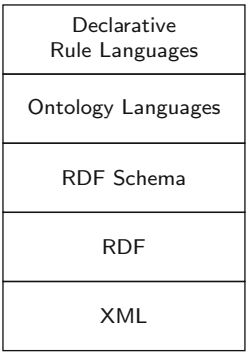

Fig. 12.13 Semantic web technologies. Simplified from [Antoniou and Plexousakis 2018]

(which stands for Extensive Markup Language) proposed by the World Wide Web Consortium (W3C).

XML tags (also called markups) divide data into pieces called *elements*, with the objective to provide more semantics to the data. Elements can be nested but they cannot be overlapped. Nesting of elements represents hierarchical relationships between them. As an example, Fig. 12.14 is the XML representation, with slight revisions, of the bibliography data that we had given earlier.

An XML document can be represented as a trie that contains a *root element*, which has zero or more nested subelements (or *child elements*), which can recursively contain subelements. For each element, there are zero or more *attributes* with atomic values assigned to them. An element also contains an optional value. Due to the textual representation of the trie, a total order, called *document order*, is defined on all elements corresponding to the order in which the first character of the elements occurs in the document.

For instance, the root element in Fig. 12.4 is `bib`, which has three child elements: two `book` and one `article`. The first `book` element has an attribute `year` with atomic value "1999", and also contains subelements (e.g., the `title` element). An element can contain a value (e.g., "`Principles of Distributed Database Systems`" for the element `title`).

Standard XML document definition is a bit more complicated: it can contain ID-IDREFs, which define references between elements in the same document or in another document. In that case, the document representation becomes a graph. However, it is quite common to use the simpler trie representation, and we'll assume the same in this section and we define it more precisely below.[6]

An XML document is modeled as an ordered, node-labeled trie $T = (V, E)$, where each node $v \in V$ corresponds to an element or attribute and is characterized by:

- a unique identifier denoted by $ID(v)$;
- a unique *kind* property, denoted as $kind(v)$, assigned from the set {`element, attribute, text`};
- a label, denoted by $label(v)$, assigned from some alphabet Σ;
- a content, denoted by $content(v)$, which is empty for nonleaf nodes and is a strong for leaf nodes.

A directed edge $e = (u, v)$ is included in E if and only if:

- $kind(u) = kind(v) =$ `element`, and v is a subelement of u; or
- $kind(u) =$ `element` $\wedge\, kind(v) =$ `attribute`, and v is an attribute of u.

Now that an XML document trie is properly defined, we can define an instance of XML data model as an ordered collection (sequence) of XML document trie nodes or atomic values. A schema may or may not be defined for an XML document, since it is a self-describing format. If a schema is defined for a collection of

[6] In addition, we omit the comment nodes, namespace nodes, and PI nodes from the model.

```
<bib>
<book year = "1999">
<author> M. Tamer Ozsu </author>
<author> Patrick Valduriez </author>
<title> Principles of Distributed ... </title>
<chapters>
<chapter>
<heading> ... </heading>
<body> ... </body>
</chapter>
...
<chapter>
<heading> ... </heading>
<body> ... </body>
</chapter>
</chapters>
<price currency= "USD"> 98.50 </price>
</book>
<article year = "2009">
<author> M. Tamer Ozsu </author>
<author> Yingying Tao </author>
<title> Mining data streams ... </title>
<venue> "CIKM" </venue>
<sections>
<section> ... </section>
...
<section> ... </section>
</sections>
</article>
<book>
<author> Anthony Bonato </author>
<title> A Course on the Web Graph </title>
<ISBN> TK5105.888.B667 </ISBN>
<chapters>
<chapter>
<heading> ... </heading>
<body> ... </body>
</chapter>
<chapter>
<heading> ... </heading>
<body> ... </body>
</chapter>
<chapter>
<heading> ... </heading>
<body> ... </body>
</chapter>
</chapters>
<publisher> AMS </publisher>
</book>
</bib>
```

Fig. 12.14 An example XML document

XML documents, then each document in this collection conforms to that schema; however, the schema allows for variations in each document, since not all elements or attributes may exist in each document. XML schemas can be defined either using the Document Type Definition (DTD) or XMLSchema. In this section, we will use a simpler schema definition that exploits the graph structure of XML documents as defined above.

An XML *schema graph* is defined as a 5-tuple $\langle \Sigma, \Psi, s, m, \rho \rangle$ where Σ is an alphabet of XML document node types, ρ is the root node type, $\Psi \subseteq \Sigma \times \Sigma$ is a set of edges between node types, $s : \Psi \rightarrow \{\text{ONCE, OPT, MULT}\}$ and $m : \Sigma \rightarrow \{\texttt{string}\}$. The semantics of this definition are as follows: An edge $\psi = (\sigma_1, \sigma_2) \in \Psi$ denotes that an item of type σ_1 may contain an item of type σ_2. $s(\psi)$ denotes the cardinality of the containment represented by this edge: If $s(\psi) = \text{ONCE}$, then an item of type σ_1 must contain exactly one item of σ_2. If $s(\psi) = \text{OPT}$, then an item of type σ_1 may or may not contain an item of type σ_2. If $s(\psi) = \text{MULT}$, then an item of type σ_1 may contain multiple items of type σ_2. $m(\sigma)$ denotes the domain of the text content of an item of type σ, represented as the set of all strings that may occur inside such an item.

Using the definition of XML data model and instances of this data model, it is now possible to define the query languages. Expressions in XML query languages take an instance of XML data as input and produce an instance of XML data as output. XPath and XQuery are two query languages proposed by the W3C. Path expressions, that we introduced earlier, are present in both query languages and are arguably the most natural way to query the hierarchical XML data. XQuery defines for more powerful constructs. Although XQuery was the subject of intense research and development efforts in the 2000s, it is not widely used any longer. It is complicated, hard to formulate by users and difficult to optimize by systems. JSON has replaced both XML and XQuery for many applications, as we discussed in Chap. 11, although XML representation remains important for the semantic web (but not XQuery).

12.6.2.2 RDF

RDF is the data model on top of XML and forms a fundamental building block of the semantic web (Fig. 12.13). Although it was originally proposed by W3C as a component of the semantic web, its use is now wider. For example, Yago and DBPedia extract facts from Wikipedia automatically and store them in RDF format to support structural queries over Wikipedia; biologists encode their experiments and results using RDF to communicate among themselves leading to RDF data collections, such as Bio2RDF (bio2rdf.org) and Uniprot RDF (dev.isb-sib.ch/projects/uniprot-rdf). Related to semantic web, LOD project builds a RDF data cloud by linking a large number of datasets, as noted earlier.

RDF models each "fact" as a set of triples (**s**ubject, **p**roperty (or **p**redicate), **o**bject), denoted as $\langle s, p, o \rangle$, where *subject* is an entity, class or blank node, a

property[7] denotes one attribute associated with one entity, and *object* is an entity, a class, a blank node, or a literal value. According to the RDF standard, an entity is denoted by a URI (Uniform Resource Identifier) that refers to a named *resource* in the environment that is being modeled. Blank nodes, by contrast, refer to anonymous resources that do not have a name.[8] Thus, each triple represents a named relationship; those involving blank nodes simply indicate that "something with the given relationship exists, without naming it."

It is appropriate at this point to briefly talk about the next layer in the semantic web technology stack (Fig. 12.13), namely the RDF Schema (RDFS). It is possible to annotate RDF data with semantic metadata using RDFS, which is also a W3C standard.[9] This annotation primarily enables reasoning over the RDF data (called *entailment*), and also impacts data organization in some cases, and the metadata can be used for semantic query optimization. We illustrate the fundamental concepts by simple examples using RDFS, which allows the definition of *classes* and *class hierarchies*. RDFS has built-in class definitions—the more important ones being rdfs:Class and rdfs:subClassOf that are used to define a class and a subclass, respectively (another one, rdfs:label is used in our query examples below). To specify that an individual resource is an element of the class, a special property, rdf:type is used.

Example 12.13 For example, if we wanted to define a class called `Movies` and two subclasses `ActionMovies` and `Dramas`, this would be accomplished in the following way:

```
Movies rdf:type rdfs:Class .
ActionMovies rdfs:subClassOf Movies .
Dramas rdfs:subClassOf Movies .
```

♦

Formally, a RDF dataset can be defined as follows. Let $\mathcal{U}$, $\mathcal{B}$, $\mathcal{L}$, and $\mathcal{V}$ denote the sets of all URIs, blank nodes, literals, and variables, respectively. A tuple $(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is an *RDF triple*. A set of RDF triples form a *RDF dataset*.

Example 12.14 An example RDF dataset is shown in Fig. 12.15 where the data comes from a number of sources as defined by the URI prefixes.

♦

[7] In literature, the terms "property" and "predicate" are used interchangeably; in this paper, we will use "property" consistently.

[8] In much of the research, blank nodes are ignored. Unless explicitly stated otherwise, we will ignore them in this paper as well.

[9] The same annotation can also be done using the ontology languages such as OWL (also a W3C standard) but we will not discuss that topic further.

Prefixes:
mdb=http://data.linkedmdb.org/resource/geo=http://sws.geonames.org/
bm=http://wifo5-03.informatik.uni-mannheim.de/bookmashup/
exvo=http://lexvo.org/id/
wp=http://en.wikipedia.org/wiki/

Subject	Property	Object
mdb: film/2014	rdfs:label	"The Shining"
mdb:film/2014	movie:initial_release_date	"1980-05-23"'
mdb:film/2014	movie:director	mdb:director/8476
mdb:film/2014	movie:actor	mdb:actor/29704
mdb:film/2014	movie:actor	mdb: actor/30013
mdb:film/2014	movie:music_contributor	mdb: music_contributor/4110
mdb:film/2014	foaf:based_near	geo:2635167
mdb:film/2014	movie:relatedBook	bm:0743424425
mdb:film/2014	movie:language	lexvo:iso639-3/eng
mdb:director/8476	movie:director_name	"Stanley Kubrick"
mdb:film/2685	movie:director	mdb:director/8476
mdb:film/2685	rdfs:label	"A Clockwork Orange"
mdb:film/424	movie:director	mdb:director/8476
mdb:film/424	rdfs:label	"Spartacus"
mdb:actor/29704	movie:actor_name	"Jack Nicholson"
mdb:film/1267	movie:actor	mdb:actor/29704
mdb:film/1267	rdfs:label	"The Last Tycoon"
mdb:film/3418	movie:actor	mdb:actor/29704
mdb:film/3418	rdfs:label	"The Passenger"
geo:2635167	gn:name	"United Kingdom"
geo:2635167	gn:population	62348447
geo:2635167	gn:wikipediaArticle	wp:United_Kingdom
bm:books/0743424425	dc:creator	bm:persons/Stephen+King
bm:books/0743424425	rev:rating	4.7
bm:books/0743424425	scom:hasOffer	bm:offers/0743424425amazonOffer
lexvo:iso639-3/eng	rdfs:label	"English"
lexvo:iso639-3/eng	lvont:usedIn	lexvo:iso3166/CA
lexvo:iso639-3/eng	lvont:usesScript	lexvo:script/Latn

Fig. 12.15 Example RDF dataset. Prefixes are used to identify the data sources

RDF data can be modeled as an RDF graph as follows. A *RDF graph* is a six-tuple $G = \langle V, L_V, f_V, E, L_E, f_E \rangle$, where

1. $V = V_c \cup V_e \cup V_l$ is a collection of vertices that correspond to all subjects and objects in RDF data, where V_c, V_e, and V_l are collections of class vertices, entity vertices, and literal vertices, respectively.
2. L_V is a collection of vertex labels.
3. A *vertex labeling function* $f_V : V \to L_V$ is an bijective function that assigns to each vertex a label. The label of a vertex $u \in V_l$ is its literal value, and the label of a vertex $u \in V_c \cup V_e$ is its corresponding URI.
4. $E = \{\overrightarrow{u_1, u_2}\}$ is a collection of directed edges that connect the corresponding subjects and objects.
5. L_E is a collection of edge labels.

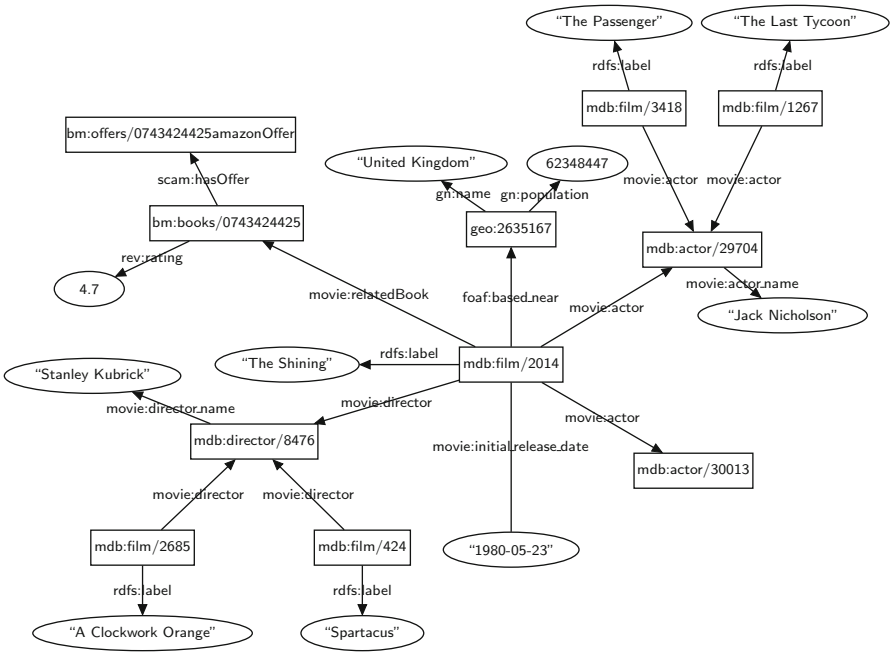

Fig. 12.16 RDF graph corresponding to the dataset in Fig. 12.15

6. An *edge labeling function* $f_E : E \rightarrow L_E$ is an bijective function that assigns to each edge a label. The label of an edge $e \in E$ is its corresponding property.

An edge $\overrightarrow{u_1, u_2}$ is an *attribute property* edge if $u_2 \in V_l$; otherwise, it is a *link* edge.

Note that RDF graph structure is different than the property graphs we discussed in Chap. 10. As you will recall, property graphs have attributes attached to vertices and edges allowing sophisticated value-based predicates to be specified in queries. In RDF graphs, the only attribute of a vertex or an edge is the vertex/edge label. What would be vertex attributes in a property graph become edges whose labels are the attribute names. Therefore, RDF graphs are simpler and more regular, but generally larger in terms of the number of vertices and edges.

Figure 12.16 shows an example of an RDF graph. The vertices that are denoted by boxes are entity or class vertices, and the others are literal vertices.

The W3C standard language for RDF is SPARQL, which can be defined as follows [Hartig 2012]. Let $\mathcal{U}, \mathcal{B}, \mathcal{L}$, and $\mathcal{V}$ denote the sets of all URIs, blank nodes, literals, and variables, respectively. A SPARQL expression is expressed recursively

1. A *triple pattern* $(\mathcal{U} \cup \mathcal{B} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{V}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L} \cup \mathcal{V})$ is a SPARQL expression,
2. (optionally) If P is a SPARQL expression, then $P\ FILTER\ R$ is also a SPARQL expression where R is a built-in SPARQL filter condition,

3. (optionally) If P_1 and P_2 are SPARQL expressions, then $P_1\ AND|OPT|OR\ P_2$ are also SPARQL expressions.

A set of triple patterns is called *basic graph pattern* (BGP) and SPARQL expressions that only contain these are called *BGP queries*. These are the subject of most of the research in SPARQL query evaluation.

Example 12.15 An example SPARQL query that finds the names of the movies directed by "Stanley Kubrick" and have a related book that has a rating greater than 4.0 is specified as follows:

```
SELECT ?name
WHERE {
?m rdfs:label ?name. ?m movie:director ?d.
?d movie:director_name "Stanley Kubrick".
?m movie:relatedBook ?b. ?b rev:rating ?r.
FILTER(?r > 4.0)
}
```

In this query, the first three lines in the **WHERE** clause form a BGP consisting of five triple patterns. All triple patterns in this example have *variables*, such as "?m", "?name" and "?r", and "?r" has a filter: **FILTER**(?r > 4.0).

♦

A SPARQL query can also be represented as a *query graph*. A *query graph* is a seven-tuple $Q = \langle V^Q, L_V^Q, E^Q, L_E^Q, f_V^Q, f_E^Q, FL\rangle$, where

1. $V^Q = V_c^Q \cup V_e^Q \cup V_l^Q \cup V_p^Q$ is a collection of vertices that correspond to all subjects and objects in a SPARQL query, where V_p^Q is a collection of variable vertices (corresponding to variables in the query expression), and V_c^Q and V_e^Q and V_l^Q are collections of class vertices, entity vertices, and literal vertices in the query graph Q, respectively.
2. E^Q is a collection of edges that correspond to properties in a SPARQL query.
3. L_V^Q is a collection of vertex labels in Q and L_E^Q is the edge labels in E^Q.
4. $f_V^Q : V^Q \rightarrow L_V^Q$ is a bijective vertex labeling function that assigns to each vertex in Q a label from L_V^Q. The label of a vertex $v \in V_p^Q$ is the variable; that of a vertex $v \in V_l^Q$ is its literal value; and that of a vertex $v \in V_c^Q \cup V_e^Q$ is its corresponding URI.
5. $f_E^Q : V^Q \rightarrow L_E^Q$ is a bijective vertex labeling function that assigns to each edge in Q a label from L_V^Q. An edge label can be a property or an edge variable.
6. FL are constraint filters.

The query graph for Q_1 is given in Fig. 12.17.

The semantics of SPARQL query evaluation can, therefore, be defined as subgraph matching using graph homomorphism whereby all subgraphs of an RDF graph G are found that are homomorphic to the SPARQL query graph Q.

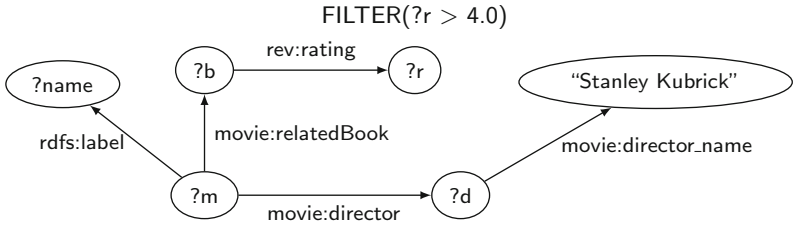

Fig. 12.17 SPARQL query graph corresponding to query Q_1

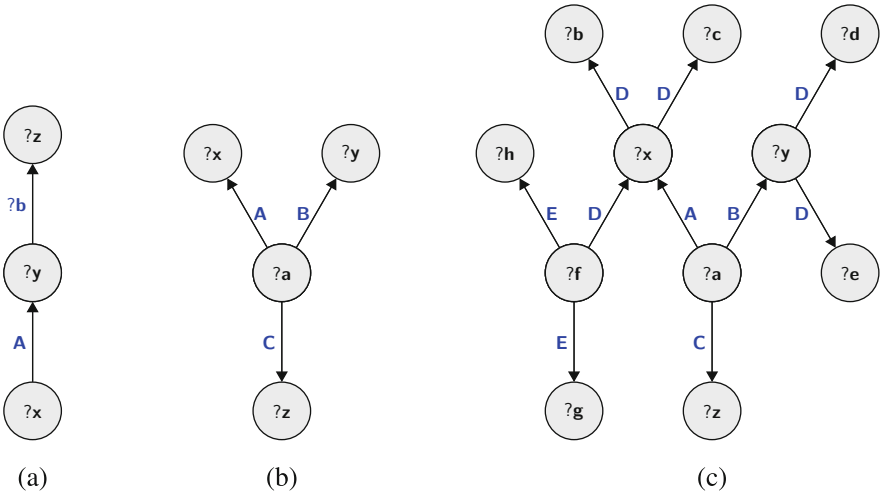

Fig. 12.18 Sample SPARQL query shapes. (**a**) Q_L. (**b**) Q_S. (**c**) Q_K

It is usual to talk about SPARQL query types based on the shape of the query graph (we will refer to these types in the following discussion). Typically, three query types are observed: (i) linear (Fig. 12.18a), where the variable in the object field of one triple pattern appears in the subject of another triple pattern (e.g., ?y in Q_L) (ii) star-shaped (Fig. 12.18b), where the variable in the object field of one triple pattern appears in the subject of multiple other triple patterns (e.g., ?a in Q_S), and (iii) snowflake-shaped (Fig. 12.18c), which is a combination of multiple star queries.

A number of RDF data management systems have been developed. These can be broadly classified into five groups: those that map the RDF data directly into a relational system, those that use a relational schema with extensive indexing (and a native storage system), those that denormalize the triples table into clustered properties, those that use column-store organization, and those that exploit the native graph pattern matching semantics of SPARQL.

Direct Relational Mapping

Direct relational mapping systems take advantage of the fact that RDF triples have a natural tabular structure. Therefore, they create a single table with three columns (Subject, Property, Object) that holds the triples (there usually are additional auxiliary tables, but we ignore them here). The SPARQL query can then be translated into SQL and executed on this table. It has been shown that SPARQL 1.0 can be full translated to SQL; whether the same is true for SPARQL 1.1 with its added features is still open. This approach aims to exploit the well-developed relational storage, query processing and optimization techniques in executing SPARQL queries. Systems such as Sesame SQL92SAIL[10] and Oracle follow this approach.

Example 12.16 Assuming that the table given in Fig. 12.15 is a relational table, the example SPARQL query in Example 12.15 can be translated to the following SQL query (where s,p,o correspond to column names: Subject, Property, Object):

```
SELECT T1.object
FROM  T AS T1, T AS T2, T AS T3,
T AS T4, T AS T5
WHERE T1.p="rdfs:label"
AND T2.p="movie:relatedBook"
AND T3.p="movie:director"
AND T4.p="rev:rating"
AND T5.p="movie:director_name"
AND T1.s=T2.s
AND T1.s=T3.s
AND T2.o=T4.s
AND T3.o=T5.s
AND T4.o > 4.0
AND T5.o="Stanley Kubrick"
```

♦

As can be seen from this example, this approach results in a high number of self-joins that are not easy to optimize. Furthermore, in large datasets, this single triples table becomes very large, further complicating query processing.

[10] Sesame is built to interact with any storage system since it implements a Storage and Inference Layer (SAIL) to interface with the particular storage system on which it sits. SQL92SAIL is the specific instantiation to work on relational systems.

Single Table Extensive Indexing

One alternative to the problems created by direct relational mapping is to develop native storage systems that allow extensive indexing of the triple table. Hexastore and RDF-3X are examples of this approach. The single table is maintained, but extensively indexed. For example, RDF-3X creates indexes for all six possible permutations of the subject, property, and object: (spo, sop,ops,ops,sop,pos). Each of these indexes is sorted lexicographically by the first column, followed by the second column, followed by the third column. These are then stored in the leaf pages of a clustered B^+-tree.

The advantage of this type of organization is that SPARQL queries can be efficiently processed regardless of where the variables occur (subject, property, object) since one of the indexes will be applicable. Furthermore, it allows for index-based query processing that eliminates some of the self-joins—they are turned into range queries over the particular index. Even when joins are required, fast merge-join can be used since each index is sorted on the first column. The obvious disadvantages are, of course, the space usage, and the overhead of updating the multiple indexes if data is dynamic.

Property Tables

Property tables approach exploits the regularity exhibited in RDF datasets where there are repeated occurrence of patterns of statements. Consequently, it stores "related" properties in the same table. The first system that proposed this approach is Jena; IBM's DB2RDF also follows the same strategy. In both of these cases, the resulting tables are mapped to a relational system and the queries are converted to SQL for execution.

Jena defines two types of property tables. The first type, which can be called *clustered property table*, group together the properties that tend to occur in the same (or similar) subjects. It defines different table structures for single-valued properties versus multivalued properties. For single-valued properties, the table contains the subject column and a number of property columns (Fig. 12.19a). The value for a given property may be null if there is no RDF triple that uses the subject and that property. Each row of the table represents a number of RDF triples—the same number as the nonnull property values. For these tables, the subject is the primary key. For multivalued properties, the table structure includes the subject and the multivalued property (Fig. 12.19b). Each row of this table represents a single RDF triple; the key of the table is the compound key (subject, property). The mapping of the single triple table to property tables is a database design problem that is done by a database administrator.

Jena also defines a *property class table* that cluster the subjects with the same *type* of property into one property table (Fig. 12.19c). In this case, all members of a class (recall our discussion of class structure within the context of RDFS) together

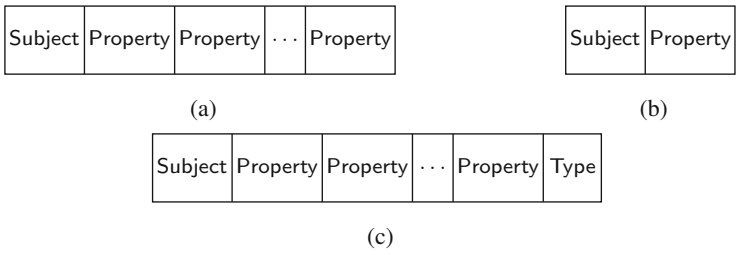

Fig. 12.19 Clustered property table design

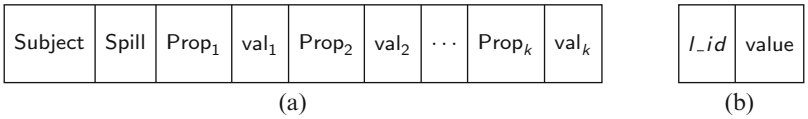

Fig. 12.20 DB2RDF table design. (**a**) DPH. (**b**) DS

in one table. The "Type" column is the value of rdf:type for each property in that row.

Example 12.17 The example dataset in Example 12.14 may be organized to create one table that includes the properties of subjects that are films, one table for properties of directors, one table for properties of actors, one table for properties of books and so on. ♦

IBM DB2RDF also follows the same strategy, but with a more dynamic table organization (Fig. 12.20). The table, called *direct primary hash* (DPH) is organized by each subject, but instead of manually identifying "similar" properties, the table accommodates k property columns, each of which can be assigned a different property in different rows. Each property column is, in fact, two columns: one that holds the property label, and the other that holds the value. If the number of properties for a given subject is greater than k, then the extra properties are spilled onto a second row and this is marked on the "spill" column. For multivalued properties, a *direct secondary hash* (DSH) table is maintained—the original property value stores a unique identifier l_id, which appears in the DS table along with the values.

The advantage of property table approach is that joins in star queries (i.e., subject-subject joins) become single table scans. Therefore, the translated query has fewer joins. The disadvantages are that in either of the two forms discussed above, there could be a significant number of null values in the tables (see the number of NULLs in Fig. 12.19), and dealing with multivalued properties requires special care. Furthermore, although star queries can be handled efficiently, this approach may not help much with other query types. Finally, when manual assignment is used, clustering "similar" properties is nontrivial and bad design decisions exacerbate the null value problem.

Binary Tables

Binary tables approach follows column-oriented database schema organization and defines a two-column table for each property containing the subject and object. This results in a set of tables each of which is ordered by the subject. This is a typical column-oriented database organization and benefits from the usual advantages of such systems such as reduced I/O due to reading only the needed properties and reduced tuple length, compression due to redundancy in the column values, etc. In addition, it avoids the null values that is experienced in property tables as well as the need for manual or automatic clustering algorithms for "similar" properties, and naturally supports multivalued properties—each becomes a separate row as in the case of Jena's DS table. Furthermore, since tables are ordered on subjects, subject-subject joins can be implemented using efficient merge-joins. The shortcomings are that the queries require more join operations some of which may be subject-object joins that are not helped by the merge-join operation. Furthermore, insertions into the tables have higher overhead since multiple tables need to be updated. It has been argued that the insertion problem can be mitigated by batch insertions, but in dynamic RDF repositories the difficulty of insertions is likely to remain a significant problem. The proliferation of the number of tables may have a negative impact on the scalability (with respect to the number of properties) of binary tables approach.

Example 12.18 For example, the binary table representation of the dataset given in Example 12.14 would create one table for each unique property—there are 18 of them. Two of these tables are shown in Fig. 12.21. ♦

Graph-Based Processing

Graph-based RDF processing approaches fundamentally implement the semantics of RDF queries as defined at the beginning of this section. In other words, they maintain the graph structure of the RDF data (using some representation such as adjacency lists), convert the SPARQL query to a query graph, and do subgraph

Subject	Object
film/2014	"The Shining"
film/2685	"A Clockwork Orange"
film/424	"Spartacus"
film/1267	"The Last Tycoon"
film/3418	"The Passenger"
iso639-3/eng	"English"

(a)

Subject	Object
film/2014	actor/29704
film/2014	actor/30013
film/1267	actor/29704
film/3418	actor/29704

(b)

Fig. 12.21 Binary table organization of properties (**a**) "rdfs:label" and (**b**) "movie:actor" from the example dataset (prefixes are removed)

matching using homomorphism to evaluate the query against the RDF graph. Systems such as gStore, and chameleon-db follow this approach.

The advantage of this approach is that it maintains the original representation of the RDF data and enforces the intended semantics of SPARQL. The disadvantage is the cost of subgraph matching—graph homomorphism is NP-complete. This raises issues with respect to the scalability of this approach to large RDF graphs; typical database techniques including indexing can be used to address this issue. In the remainder, we present the approach within the context of the gStore system to highlight the issues.

gStore uses adjacency list representation of graphs. It encodes each entity and class vertex into a fixed length bit string that captures the "neighborhood" information for each vertex and exploits this during graph matching. This results is the generation of a *data signature graph* G^*, in which each vertex corresponds to a class or an entity vertex in the RDF graph G. Specifically, G^* is induced by all entity and class vertices in the original RDF graph G together with the edges whose endpoints are either entity or class vertices. Figure 12.22a shows the data signature graph G^* that corresponds to RDF graph G in Fig. 12.16. An incoming SPARQL query is also represented as a *query graph* Q that is similarly encoded into a *query*

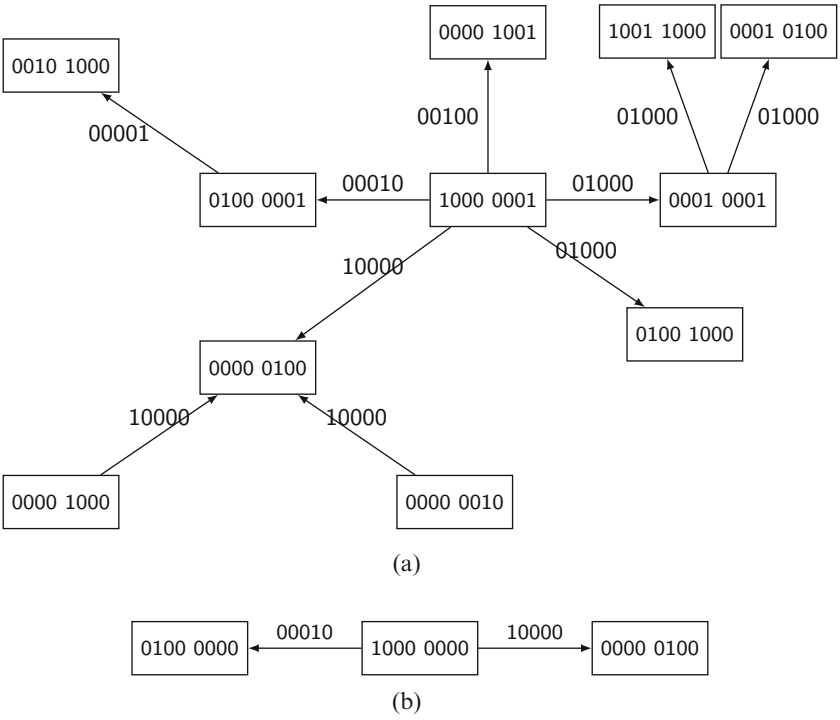

Fig. 12.22 Signature graphs. (**a**) Data signature graph G^*. (**b**) Query signature graph Q^*

signature graph Q^*. The encoding of query graph depicted in Fig. 12.17 into a query signature graph Q_2^* is shown in Fig. 12.22b.

The problem now turns into finding matches of Q^* over G^*. Although both the RDF graph and the query graph are smaller as a result of encoding, the NP-completeness of the problem remains. Therefore, gStore uses a filter-and-evaluate strategy to reduce the search space over which matching is applied. The objective is to first use a false-positive pruning strategy to find a set of candidate subgraphs (denoted as CL), and then validate these using the adjacency list to find answers (denoted as RS). Accordingly, two issues need to be addressed. First, the encoding technique should guarantee that $RS \subseteq CL$—the encoding described above provably achieves this. Second, an efficient subgraph matching algorithm is required to find matches of Q^* over G^*. For this, gStore uses an index structure called VS*-tree that is a summary graph of G^*. VS*-tree is used to efficiently process queries using a pruning strategy to reduce the search space for finding matches of Q^* over G^*.

Distributed and Federated SPARQL Execution

As RDF collections grow, scale-out solutions to scaling have been developed involving parallel and distributed processing. Many of these solutions divide an RDF graph G into several fragments and place each at a different site in a parallel/distributed system. Each site hosts a centralized RDF store of some kind. At runtime, a SPARQL query Q is decomposed into several subqueries such that each subquery can be answered locally at one site, and the results are then aggregated. Each of these papers proposes its own data partitioning strategy, and different partitioning strategies result in different query processing methods. Some of the approaches use MapReduce-based solutions where RDF triples are stored in HDFS and each triple pattern is evaluated by scanning the HDFS files followed by a MapReduce join implementation. Other approaches follow more or less the distributed/parallel query processing methodologies described in detail in various chapters of this book whereby the query is partitioned into subqueries and evaluated across the sites.

An alternative that has been proposed is to use partial query evaluation for executing distributed SPARQL queries. Partial function evaluation is a well-known programming language strategy whose basic idea is the following: given a function $f(s, d)$, where s is the known input and d is the yet unavailable input, the part of f's computation that depends only on s generates a partial answer. In this approach, data is partitioned, but queries are not—each site receives the full SPARQL query Q and executes it on the local RDF graph fragment providing data parallel computation. In this particular setting, the partial evaluation strategy is applied as follows: each site S_i treats fragment F_i as the known input in the partial evaluation stage; the unavailable input is the rest of the graph ($\overline{G} = G \setminus F_i$). There are two important issues to be addressed in this framework. The first is to compute the partial evaluation results at each site S_i given a query graph Q—in other words, addressing the graph homomorphism of Q over F_i; this is called the *local partial match* since

it finds the matches internal to fragment F_i. Since ensuring edge disjointness is not possible in vertex-disjoint partitioning, there will be *crossing edges* between graph fragments. The second task is the assembly of these local partial matches to compute crossing matches. This assembly task can be executed either on a control site or similar to distributed join.

The above approaches take a centralized RDF dataset and partition it for distributed/parallel execution. In many RDF settings, concerns arise similar to what we discussed in database integration requiring a federated solution. In the RDF world, some of the sites that host RDF data also have the capability to process SPARQL queries; these are called *SPARQL endpoints*. A typical example is LOD, where different RDF repositories are interconnected, providing a *virtually integrated distributed database*. A common technique in federated RDF environments is to precompute metadata for each individual SPARQL endpoint. The metadata can specify the capabilities of the end point or a description of the triple patterns (i.e., property) that can be answered at that endpoint, or other information that the particular algorithm uses. Based on the metadata, the original SPARQL query is decomposed into several subqueries, where each subquery is sent to its relevant SPARQL endpoints. The results of subqueries are then joined together to answer the original SPARQL query.

An alternative to precomputing metadata is to make use of SPARQL ASK queries to gather information about each endpoint and to construct the metadata on the fly. Based on the results of these queries, a SPARQL query is decomposed into subqueries and assigned to endpoints.

12.6.2.3 Navigating and Querying the LOD

LOD consists of a set of *web documents*. The starting point, therefore, is a web document with embedded RDF triples that encode web resources. The RDF triples contain *data links* to other documents that allow web documents to be interconnected to get the graph structure.

The semantics of SPARQL queries over the LOD becomes tricky. One possibility is to adopt *full web semantics* that specifies the scope of evaluating a SPARQL query expression to be all linked data. There is no known (terminating) query execution algorithm that can guarantee result completeness under this semantics. The alternative is a family of *reachability-based semantics* that define the scope of evaluating a SPARQL query in terms of the documents that can be reached: given a set of seed URIs and a reachability condition, the scope is all data along the paths of the data links from the seeds and that satisfy the reachability condition. The family is defined by different reachability conditions. In this case, there are computationally feasible algorithms.

There are three approaches to SPARQL query execution over LOD: traversal-based, index-based, and hybrid. *Traversal approaches* basically implement a reachability-based semantics: starting from seed URIs, they recursively discover relevant URIs by traversing specific data links at query execution runtime. For these

algorithms, the selection of the seed URIs is critical for performance. The advantage of traversal approaches is their simplicity (to implement) since they do not need to maintain any data structures (such as indexes). The disadvantages are the latency of query execution since these algorithms "browse" web documents, and repeated data retrieval from each document introduces significant latency. They also have limited possibility for parallelization—they can be parallelized to the same extent that crawling algorithms can.

The *index-based approaches* use an index to determine relevant URIs, thereby reducing the number of linked documents that need to be accessed. A reasonable index key is triple patterns in which case the "relevant" URIs for a given query are determined by accessing the index, and the query is evaluated over the data retrieved by accessing those URIs. In these approaches, data retrieval can be fully parallelized, which reduces the negative impact of data retrieval on query execution time. The disadvantages of the approach are the dependence on the index—both in terms of the latency that index construction introduces and in terms of the restriction the index imposes on what can be selected—and the freshness issues that result from the dynamicity of the web and the difficulty of keeping the index up-to-date.

Hybrid approaches perform a traversal-based execution using prioritized listing of URIs for look-up. The initial seeds come from a prepopulated index; new discovered URIs that are not in the index are ranked according to number of referring documents.

12.6.3 Data Quality Issues in Web Data Integration

In Chap. 7 (specifically in Sect. 7.1.5) we discussed data quality and data cleaning issues in the case of database integration (mainly data warehousing) systems. Data quality issues in web data are only more severe due to the sheer number of web data sources, the uncontrolled data entry process of web information sources, and the increased data diversity. Data quality encompasses both data consistency and veracity (authenticity and conformity of data with reality). In a data warehouse, data consistency is obtained through data cleaning, which deals with detecting and removing errors and inconsistencies from data. Data cleaning in the web context (and also in data lakes) is made difficult by the lack of schema information and the limited number of integrity constraints that can be defined without a schema.

Checking for data veracity remains a big challenge. However, if many different data sources overlap, as it is often the case with data coming from the web for instance, there will be a high-level of redundancy. It may be possible to use efficient data fusion techniques (to be discussed shortly) to detect the correct values for the same data items, and thus discover the truth.

In this section, we highlight some of the main data quality and data cleaning issues in web data and discuss current solutions for addressing them.

12.6.3.1 Cleaning Structured Web Data

Structured data represents an important category of data on the web, and they suffer from numerous data quality issues. In the following, we first summarize the techniques proposed in cleaning structured data in general. Then, we point out the unique challenges in cleaning structured data on the web.

Figure 12.23 shows a typical workflow for cleaning structured data, consisting of an optional discovery and profiling step, an error detection step, and an error repair step. To clean a dirty dataset, we often need to model various aspects of this data (metadata), e.g., schema, patterns, probability distributions, and other metadata. One way to obtain such metadata is by consulting domain experts, which is usually a costly and a time-consuming process, hence a discovery and profiling step is often used to discover these metadata automatically. Given a dirty dataset and the associated metadata, the error detection step finds part of the data that do not conform to the metadata, and declares this subset to contain errors. The errors surfaced by the error detection step can be in various forms, such as outliers, violations of integrity constraints, and duplicates. Finally, the error repair step produces data updates that are applied to the dirty dataset to remove detected errors. Since there are many uncertainties in the data cleaning process, external sources such as knowledge bases and human experts are consulted whenever possible to ensure the accuracy of the cleaning workflow.

The above process works well for structured tables that have a rich set of metadata, e.g., large schema with enough constraints to model columns and rows interactions. Also, the cleaning and error detection process works better when there are enough examples (tuples) for automatic algorithms to compare various instances to detect possible errors, and to leverage the redundancy in the data to correct these

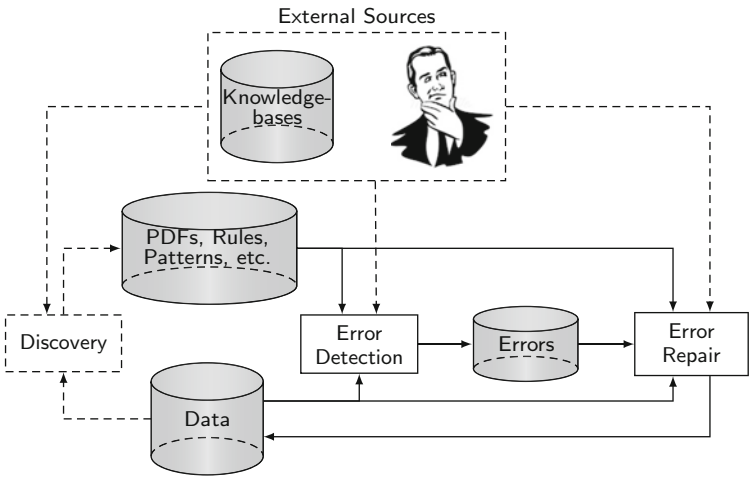

Fig. 12.23 A typical workflow for cleaning structured data

Sevilla - Jerez de la Frontera-Cádiz	1861
Córdoba - Málaga	1865.
Bobadilla - Granada	1874
Córdoba - Bélmez	1874
Osuna	La Roda

(a)

Polaco	15.04.1983	194	84
Vini	29.09.1982	N/A	N/A
Caiao	30/11/1982	N/A	N/A
Jairo	17.02.1990	N/A	N/A
Michael	20.04.1983	N/A	N/A
Ricardinho	19.11.1975	192	94

(b)

2002[12]	10.300 oz	899,500 oz
2005[13]	25.272	2.174.620 oz
2006[13]	49.354 oz	3.005.611 oz
2007[13]	48.807 oz	3.165408 oz
2008[9]	47.755 oz	3.157.837 oz
2009[2]	0.9 million oz	818.050 oz

(c)

WARRIORS@Susses Thunder	13-28	—
WARRIORS@Hampshire Thrashers	42-13	—
Essex Spartans@WARRIORS	P-P	Postponed
WARRIORS@Cambridgeshire Cats	36-44	—
East Kent Mavericks@WARRIORS	12-18	—
WARRIORS@East Kent Mavericks	15-17	—

(d)

Fig. 12.24 Data quality issues on structured web data (erroneous data is marked in red cells). Adapted from [Huang and He 2018]. (**a**) Extra dot. (**b**) Mixed dates. (**c**) Inconsistent weights. (**d**) Score placeholder

errors. However, in web tables, both of these premises are not satisfied, as most of the tables are short (few tuples) and skinny (limited number of attributes). o make matters worse, the number of web tables is far greater than the number of tables in a data warehouse. This means that manual cleaning, though relatively for a single web table, is not feasible for all structured web tables. Figure 12.24 shows some sample errors found on Wikipedia tables, and there is an estimated 300K such errors.

12.6.3.2 Web Data Fusion

A common problem that arises often in web data integration is data fusion, namely deciding what is the correct value for an item that has different representations from multiple web sources. The problem is that different web sources can provide conflicting representations, and thus making data fusion hard. There are two types of data conflicts: *uncertainty* and *contradiction*. Uncertainty is a conflict between a nonnull value and one or more null values that are used to describe the same property of a real world entity. Uncertainty is caused by missing information, usually represented by null values in a source. Contradiction is a conflict between two or more different nonnull values that represent different values of the same property of a real world entity. Contradiction is caused by different sources providing different values for the same attribute.

Automatic cleaning of web tables is thus particularly challenging. While cleaning techniques developed in the context of data warehouse can be applied to clean some

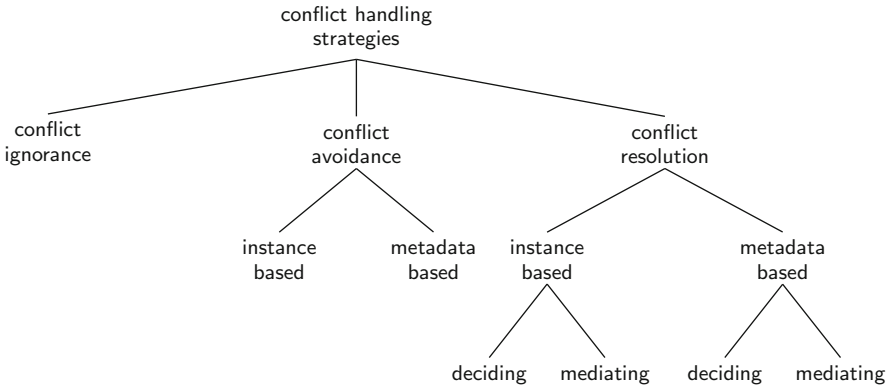

Fig. 12.25 Classification of strategies for data fusion. From [Bleiholder and Naumann 2009]

errors, cleaning web tables deserve more dedicated techniques. Auto-Detect is a recent proposal that aims at detecting such errors on web tables. Auto-Detect is a data-driven statistics-based techniques that leverage value co-occurrence statistics from large corpora for error detection. The main assumption is that if a certain value combination is extremely rare (quantified using point-wise mutual information), then it suggests a potential error. While Auto-Detect is able to detect many errors, it does not suggest data fixes. We have yet to see proposals that automatically repair errors (or even suggest fixes) in web table.

Figure 12.25 shows the classification of different data fusing strategies. *Conflict ignorance* strategies ignore the conflicts and simply pass the conflicts to the users or applications. *Conflict avoidance* strategies acknowledge the existence of conflicting representations, and apply a simple rule to take a unique decision based on either the data instance or the metadata. An example of instance based conflict avoidance strategy is to prefer nonnull values over null values. An example of metadata based conflict avoidance strategy is to prefer values from one source over values from another. *Conflict resolution* strategies resolve the conflicts, by picking a value from the already present values (deciding) or by choosing a value that does not necessarily exist among present values (mediating). An example of instance based, deciding conflict resolution strategy is to take the most frequent value. An example of instance based, mediating conflict resolution strategy is to take the average of all present values.

12.6.3.3 Web Source Quality

These basic conflict resolution strategies described above mostly rely on participating values to resolve conflicts, and they can fall short in the following three aspects. First, web sources have different qualities; data values provided by more accurate web sources are usually more accurate. However, more accurate web

sources can also provide incorrect values, therefore, an advanced resolution strategy is often needed to take source quality into consideration when predicting the correct value. Second, web sources can copy from each other, and ignoring these kinds of dependencies between web sources can cause wrong resolution decisions. For example, the majority vote strategy to resolve conflicts would be affected if some data items in a source are copied. Third, the correct value for a data item may evolve over time as well (e.g., a person's affiliation), hence, it is therefore crucial to distinguish between incorrect value and *outdated* value when evaluating source accuracies and making resolution decisions.

The building block of advanced data fusion strategies is to evaluate the trustworthiness or quality of a source. In this section, we discuss how the accuracy of a data source is modeled, and we mention how that model is extended to handle source dependencies and source freshness.

Source Accuracy

The accuracy of a source can be measured as the fraction of true values provided by a source. The accuracy of a source S is denoted by $A(S)$, which can be considered as the probability that a value provided by S is the true value. Let $V(S)$ denote the values provided by S. For each $v \in V(S)$, let $Pr(v)$ denote the probability that v is the true value. Then $A(S)$ is computed as follows:

$$A(S) = Avg_{v \in V(S)} Pr(v)$$

Consider a data item D. Let $Dom(D)$ be the domain of D, including one true value and n false values. Let S_D be the set of sources that provide a value for D, and let $S_D(v) \subseteq S_D$ be the set of sources that provide the value v for D. Let $\Phi(D)$ denote the observation of which value each $S \in S_D$ provides for D. The probability $Pr(v)$ can be computed as follows:

$$Pr(v) = Pr(v \text{ is true value} | \Phi(D)) \propto Pr(\Phi(D) | v \text{ is true value})$$

Assume that sources are independent and that the n false values are equally likely to happen, $Pr(\Phi(D)|v$ is true value) can be computed as follows:

$$Pr(\Phi(D)|v \text{ is true value}) = \prod_{S \in S_D(v)} A(S) \prod_{S \in S_D \setminus S_D(v)} \frac{1 - A(S)}{n}$$

which can be rewritten as

$$Pr(\Phi(D)|v \text{ is true value}) = \prod_{S \in S_D(v)} \frac{nA(S)}{1 - A(S)} \prod_{S \in S_D} \frac{1 - A(S)}{n}$$

Since $\prod_{S \in S_D} \frac{1-A(S)}{n}$ is the same for all values, we have

$$Pr(\Phi(D)|v \text{ is true value}) \propto \prod_{S \in S_D(v)} \frac{nA(S)}{1 - A(S)}$$

Accordingly, the *vote count* of a data source S is defined as:

$$C(S) = ln \frac{nA(S)}{1 - A(S)}$$

The *vote count* of a value v is defined as:

$$C(v) = \sum_{S \ in S_D(v)} C(S)$$

Intuitively, a source with a higher vote count is more accurate and a value with a higher vote count is more likely to be true. Combining the above analysis, the probability of each value v can be computed as follows:

$$Pr(v) = \frac{exp(C(v))}{\sum_{v_0 \ in Dom(v)} exp(C(v_0))}$$

Obviously, for a data item D, the value $v \in Dom(D)$ with the highest probability $Pr(v)$ would be selected as the true value. As we can see, the computation of the source accuracy $A(S)$ depends on the probability $Pr(v)$, and the computation of the probability $Pr(v)$ depends on the source accuracy $A(S)$. An algorithm is possible that starts with the same accuracy for every source and the same probability for every value, and iteratively computes probabilities for all sources and probabilities for all values until convergence. The convergence criteria is set to be when there is no change in source accuracies and no oscillation in decided true values.

Source Dependency

The above computation for source accuracy assumes that sources are independent. In reality, sources copy from each other, which creates dependencies. There are two intuitions for copy detection between sources. First, for a particular data item, there is only one true value, but there are usually multiple false values. Two sources sharing the same true value does not necessarily imply dependency; however, two sources sharing the same false value is typically a rare event, and thus would more likely imply source dependency. Second, a random subset of values provided by a data source would typically have similar accuracies as the full set of values provided by the data source. However, for a copier data source, the subset of values it copies may have different accuracies than the rest of the values it provides independently.

Thus, between two dependent sources where one copies another, the source whose own data values' accuracies differ significantly from the values shared with the other source is more likely to be the copier. Based on these intuitions, a Bayesian model can be developed to compute the probability of copying between two sources S_1 and S_2 given the observations Φ on all data items; this probability is then used to adjust the computation of the vote count for a value $C(v)$ to account for source dependencies.

Source Freshness

We have so far assumed that data fusion is done on a static snapshot of the data. However, in reality, data evolves over time and the true value for an item might change as well. For example, the scheduled departure time for a flight might change in different months; a person's affiliation might change over time; and the CEO of a company could also change. To capture such changes, data sources will need to update their data. In this dynamic setting, data errors occur for these several reasons: (1) the sources may provide wrong values, similar to the static setting; (2) the sources may fail to update their data at all; and (3) some sources may not update their data in time. Data fusion, in this context, aims at finding all correct values and their valid periods in the history, when the true values evolve over time. While the source quality can be capture by accuracy in the static case, the metrics for evaluating source quality are more complicated in the dynamic setting—a high-quality source should provide a new value for a data item *if and only if*, and *right after* the value becomes the true value. Three metrics can be used to capture this intuition: the *coverage* of a source measures the transitions of different data items that it captures; the *exactness* measures the percentage of transitions a source miscaptures (by providing a wrong value); and the *freshness* measures how quickly a value change is captured by a source. Again, it is possible to rely on Bayesian analysis to decide both the time and the value of each transition for a data item.

Machine learning and probabilistic models have also been used in data fusion and modeling data source quality. In particular, SLiMFast is a framework that expresses data fusion as a statistical learning problem over discriminative probabilistic models. In contrast to previous learning-based fusion approaches, SLiMFast provides quality guarantees for the fused results, and it can also incorporate available domain knowledge in the fusion process. Figure 12.26 provides the system overview of SLiMFast. The input to SLiMFast includes (1) a collection of source observations, namely the possibly conflicting values provided for different objects by different sources; (2) an optional set of labeled ground truth, namely the true values for a subset of objects; and (3) some domain knowledge about sources that users deem to be informative of the accuracies of data sources. SLiMFast takes all of these information, and compiles them into a probabilistic graphical model for holistic learning and inference. Depending on the how much ground truth data is available, SLiMFast will decide which algorithm (expectation-maximization or empirical loss minimization) to use for learning the parameters of the graphical models. The

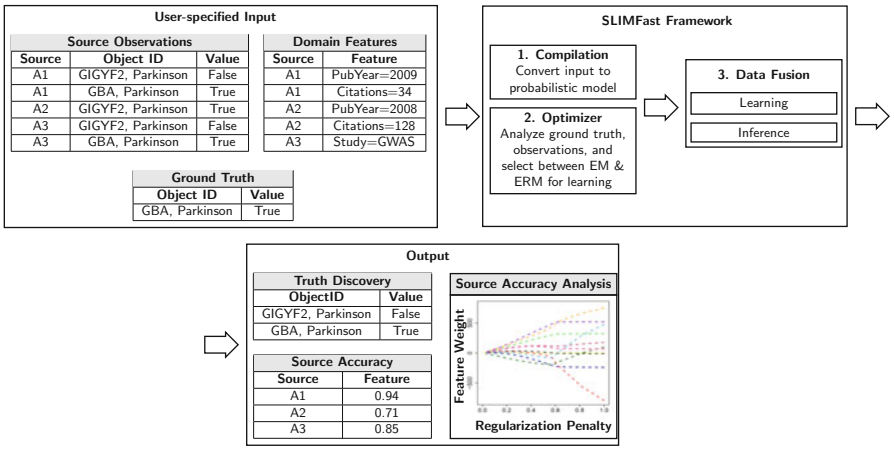

Fig. 12.26 Overview of SLiMFast. From [Rekatsinas et al. 2017]

learned model is then used for inferring both the value of objects and the source accuracies, as shown in the output.

12.7 Bibliographic Notes

There are a number of good sources on web topics, each with a slightly different focus. Abiteboul et al. [2011] focus on the use of XML and RDF for web data modeling and also contain discussions of search, and big data technologies such as MapReduce. A web data warehousing perspective is given in [Bhowmick et al. 2004]. Bonato [2008] primarily focuses on the modeling of the web as a graph and how this graph can be exploited. Early work on the web query languages and approaches are discussed in [Abiteboul et al. 1999].

A very good overview of web search issues is [Arasu et al. 2001], which we also follow in Sect. 12.2. Additionally, Lawrence and Giles [1998] provides an earlier discussion on the same topic focusing on the open web. Florescu et al. [1998] survey web search issues from a database perspective. Deep (hidden) web is the topic of [Raghavan and Garcia-Molina 2001]. Lage et al. [2002] and Hedley et al. [2004b] also discuss search over the deep web and the analysis of the results. Metasearch for accessing the deep web is discussed in [Ipeirotis and Gravano 2002, Callan and Connell 2001, Callan et al. 1999, Hedley et al. 2004a]. The metasearch-related problem of database selection is discussed by Ipeirotis and Gravano [2002] and Gravano et al. [1999] (GlOSS algorithm).

Statistics about the open web are taken from [Bharat and Broder 1998, Lawrence and Giles 1998, 1999, Gulli and Signorini 2005] and those related to the deep web are due to [Hirate et al. 2006] and [Bergman 2001].

The graph structure of the web and using graphs to model and query the web is the topic of many publications: [Kumar et al. 2000, Raghavan and Garcia-Molina 2003, Kleinberg et al. 1999] discuss web graph modeling, [Kleinberg et al. 1999, Brin and Page 1998, Kleinberg 1999] focus on graphs for search, and [Chakrabarti et al. 1998] for categorization and classification of web content. The discussion on the characteristics of the web graph and its bow-tie structure are due to Bonato [2008], Broder et al. [2000] and Kumar et al. [2000]. We did not discuss the important issues related to the management of the very large, dynamic, and volatile web graph. These are beyond the scope of this chapter, but two lines of research can be identified. The first one compresses the web graph for more efficient storage and manipulation [Adler and Mitzenmacher 2001], while the second one suggests a special representation for the web graph called S-nodes [Raghavan and Garcia-Molina 2003].

Issues on web crawling are the subject of [Cho et al. 1998, Najork and Wiener 2001] and [Page et al. 1998], the latter being the classical paper on PageRank whose revised form as discussed in this chapter is due to Langville and Meyer [2006]. Alternative crawling approaches are the subjects of [Cho and Garcia-Molina 2000] (change frequency-based), [Cho and Ntoulas 2002] (sampling-based) and [Edwards et al. 2001] (incremental). Classification techniques for evaluating relevance are discussed by [Mitchell 1997, Chakrabarti et al. 2002] (naïve Bayes), Passerini et al. [2001], Altingövde and Ulusoy [2004] (extensions of Beyesian), and by McCallum et al. [1999], Kaelbling et al. [1996] (reinforcement learning).

Web indexing is an important issue that we discussed in Sect. 12.2.2. Various text indexing methods are discussed in [Manber and Myers 1990] (suffix arrays), [Hersh 2001] [Lim et al. 2003] (inverted indexes), and [Faloutsos and Christodoulakis 1984] (signature files). Salton [1989] is probably the classical source for text processing and analysis. The challenges of building inverted indexes for the web and solutions are discussed by Arasu et al. [2001], Melnik et al. [2001], and Ribeiro-Neto and Barbosa [1998]. Related to this, ranking has been the topic of extensive research. In addition to the well-known PageRank, the HITS algorithm is due to Kleinberg [1999].

In our discussion of semistructured data approach to web querying we highlighted OEM data model and the Lorel language to expose the concepts. These are discussed in [Papakonstantinou et al. 1995] and [Abiteboul et al. 1997]. The data guides to simplify OEM are discussed in [Goldman and Widom 1997]. UnQL [Buneman et al. 1996] has similar concepts to Lorel. Our discussion of web query languages in Sect. 12.3.2 separated the languages into first and second generation; this is due to Florescu et al. [1998]. First generation languages include WebSQL [Mendelzon et al. 1997], W3QL [Konopnicki and Shmueli 1995], and WebLog [Lakshmanan et al. 1996]. The second generation languages include WebOQL [Arocena and Mendelzon 1998], and StruQL [Fernandez et al. 1997]. In the Query-Answering approach we referred to a number of systems: Mulder [Kwok et al. 2001], WebQA [Lam and Özsu 2002], Start [Katz and Lin 2002], and Tritus [Agichtein et al. 2004].

The components of the semantic web is presented by Antoniou and Plexousakis [2018]. The Linked Open Data (LOD) vision and its requirements are discussed by Bizer et al. [2018] and Berners-Lee [2006]. The topical domain separation of LOD is outlined in [Schmachtenberg et al. 2014].

Our discussion of RDF is primarily based on [Özsu 2016]. Five main approaches are described to managing RDF data: (1) direct relational mapping—Angles and Gutierrez [2008], Sequeda et al. [2014] discuss mapping SPARQL to SQL, Broekstra et al. [2002], and Chong et al. [2005] discuss Sesame SQL92SAIL and Oracle, respectively; (2) using a single table with extensive indexing (Hexastore [Weiss et al. 2008] and RDF-3X [Neumann and Weikum 2008, 2009]); (3) property tables (Jena [Wilkinson 2006]; IBM's DB2RDF [Bornea et al. 2013]); (4) binary tables (SW-Store [Abadi et al. 2009] based on the proposal by Abadi et al. [2007]) whose problems in terms of table proliferation is discussed in [Sidirourgos et al. 2008]; (5) graph-based ([Bönström et al. 2003], gStore [Zou et al. 2011, 2014], and chameleon-db [Aluç 2015]). Graph-based techniques are discussed in detail by Zou and Özsu [2017]. The distributed and cloud-based SPARQL execution is addressed in [Kaoudi and Manolescu 2015]. Three approaches are identified for SPARQL query execution over LOD [Hartig 2013a]: traversal-based [Hartig 2013b, Ladwig and Tran 2011], index-based [Umbrich et al. 2011], and hybrid [Ladwig and Tran 2010].

Cleaning structured data has been extensively studied in warehouse integration settings [Rahm and Do 2000] and [Ilyas and Chu 2015]. Expansion to a broader context, including the web, is covered in Ilyas and Chu [2019]. In our discussion of data fusion (Sect. 12.6.3.2), the separation of data conflicts into *uncertainty* and *contradiction* is due to Dong and Naumann [2009]. In the same section, the discussion of Auto-Detect is due to [Huang and He 2018] and the classification discussion (as well as Fig. 12.25) is from [Bleiholder and Naumann 2009]. The discussion of data source modeling accuracy in Sect. 12.6.3.3, its extension to handle source dependencies and source freshness are due to Dong et al. [2009b,a]. For a more comprehensive treatment on the subject of advanced data fusion, we refer readers to the tutorial [Dong and Naumann 2009] and the book [Dong and Srivastava 2015]. The SlimFAST system (see Fig. 12.26) is presented in [Rekatsinas et al. 2017] and [Koller and Friedman 2009].

One of the first systems to deal with data cleaning issues in data lakes is CLAMS [Farid et al. 2016], which allows discovering and enforcing integrity constraints over a data lake's data. CLAMS uses a graph data model, based on RDF, and a new integrity constraint formalism to capture both relational constraints and more expressive quality rules based on graph patterns as *denial constraints* [Chu et al. 2013]. CLAMS also uses Spark and parallel algorithms to enforce the constraints and detect data inconsistencies.

Exercises

Problem 12.1 How does web search differ from web querying?

Problem 12.2 (**) Consider the generic search engine architecture in Fig. 12.2. Propose an architecture for a web site with a shared-nothing cluster that implements all the components in this figure as well as web servers in an environment that will support very large sets of web documents and very large indexes, and very high numbers of web users. Define how web pages in the page directory and indexes should be partitioned and replicated. Discuss the main advantages of your architecture with respect to scalability, fault-tolerance, and performance.

Problem 12.3 (**) Consider your solution in Problem 12.2. Now consider a keyword search query from a web client to the web search engine. Propose a parallel execution strategy for the query that ranks the result web pages, with a summary of each web page.

Problem 12.4 (*) To increase locality of access and performance in different geographical regions, propose an extension of the web site architecture in Problem 12.3 with multiple sites, with web pages being replicated at all sites. Define how web pages are replicated. Define also how a user query is routed to a web site. Discuss the advantages of your architecture with respect to scalability, availability and performance.

Problem 12.5 (*) Consider your solution in Problem 12.4. Now consider a keyword search query from a web client to the web search engine. Propose a parallel execution strategy for the query that ranks the result web pages, with a summary of each web page.

Problem 12.6 (**) Consider two web data sources that we model as relations EMP1(Name, City, Phone) and EMP2(Firstname, Lastname, City). After schema integration, assume the view EMP(Firstname, Name, City, Phone) defined over EMP1 and EMP2, where each attribute in EMP comes from an attribute of EMP1 or EMP2, with EMP2. Lastname being renamed as Name. Discuss the limitations of such integration. Now consider that the two web data sources are XML. Give a corresponding definition of the XML schemas of EMP1 and EMP2. Propose an XML schema that integrates EMP1 and EMP2, and avoids the problems identified with EMP.

Appendix A
Overview of Relational DBMS

See https://cs.uwaterloo.ca/ddbs.

M. T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*,
https://doi.org/10.1007/978-3-030-26253-2

Appendix B
Centralized Query Processing

See https://cs.uwaterloo.ca/ddbs.

M. T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*,
https://doi.org/10.1007/978-3-030-26253-2

Appendix C
Transaction Processing Fundamentals

See https://cs.uwaterloo.ca/ddbs.

M. T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*,
https://doi.org/10.1007/978-3-030-26253-2

Appendix D
Review of Computer Networks

See https://cs.uwaterloo.ca/ddbs.

M. T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*,
https://doi.org/10.1007/978-3-030-26253-2

References

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, S. (2003). Aurora: a new model and architecture for data stream management. *VLDB J.*, 12(2):120–139.

Abadi, D. J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., and Zdonik, S. B. (2005). The design of the Borealis stream processing engine. In *Proc. 2nd Biennial Conf. on Innovative Data Systems Research*, pages 277–289.

Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. (2007). Scalable semantic web data management using vertical partitioning. In *Proc. 33rd Int. Conf. on Very Large Data Bases*, pages 411–422.

Abadi, D. J., Marcus, A., Madden, S., and Hollenbach, K. (2009). SW-Store: a vertically partitioned DBMS for semantic web data management. *VLDB J.*, 18(2):385–406.

Aberer, K. (2001). P-grid: A self-organizing access structure for P2P information systems. In *Proc. Int. Conf. on Cooperative Inf. Syst.*, pages 179–194.

Aberer, K. (2003). Guest editor's introduction. *ACM SIGMOD Rec.*, 32(3):21–22.

Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., and Schmidt, R. (2003a). P-grid: a self-organizing structured P2P system. *ACM SIGMOD Rec.*, 32 (3):29–33.

Aberer, K., Cudré-Mauroux, P., and Hauswirth, M. (2003b). Start making sense: The chatty web approach for global semantic agreements. *J. Web Semantics*, 1(1):89–114.

Abiteboul, S., Quass, D., McHugh, J., Widom, J., and Wiener, J. (1997). The Lorel query language for semistructured data. *Int. J. Digit. Libr.*, 1(1):68–88.

Abiteboul, S., Buneman, P., and Suciu, D. (1999). *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann.

Abiteboul, S., Manolescu, I., Rigaux, P., Rousset, M.-C., and Senellart, P. (2011). *Web Data Management*. Cambridge University Press.

Abou-Rjeili, A. and Karypis, G. (2006). Multilevel algorithms for partitioning power-law graphs. In *Proc. 20th IEEE Int. Parallel & Distributed Processing Symp.*, pages 124–124.

Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A., and Rasin, A. (2009). HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. *Proc. VLDB Endowment*, 2(1):922–933.

Adali, S., Candan, K. S., Papakonstantinou, Y., and Subrahmanian, V. S. (1996a). Query caching and optimization in distributed mediator systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 137–148.

M. T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*,
https://doi.org/10.1007/978-3-030-26253-2

Adali, S., Candan, K. S., Papakonstantinou, Y., and Subrahmanian, V. S. (1996b). Query caching and optimization in distributed mediator systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 137–148.

Adamic, L. and Huberman, B. (2000). The nature of markets in the world wide web. *Quart. J. Electron. Comm.*, 1:5–12.

Adiba, M. (1981). Derived relations: A unified mechanism for views, snapshots and distributed data. In *Proc. 7th Int. Conf. on Very Data Bases*, pages 293–305.

Adiba, M. and Lindsay, B. (1980). Database snapshots. In *Proc. 6th Int. Conf. on Very Data Bases*, pages 86–91.

Adler, M. and Mitzenmacher, M. (2001). Towards compressing web graphs. In *Proc. Data Compression Conf.*, pages 203–212.

Aggarwal, C. C., editor. (2007). *Data Streams: Models and Algorithms*. Springer.

Agichtein, E., Lawrence, S., and Gravano, L. (2004). Learning to find answers to questions on the web. *ACM Trans. Internet Tech.*, 4(3):129—162.

Agrawal, D. and Sengupta, S. (1993). Modular synchronization in distributed, multiversion databases: Version control and concurrency control. *IEEE Trans. Knowl. and Data Eng.*, 5 (1):126 –137.

Agrawal, D., Das, S., and El Abbadi, A. (2012). *Data Management in the Cloud: Challenges and Opportunities*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Agrawal, S., Narasayya, V., and Yang, B. (2004). Integrating vertical and horizontal partitioning into automated physical database design. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*.

Akal, F., Böhm, K., and Schek, H.-J. (2002). Olap query evaluation in a database cluster: A performance study on intra-query parallelism. In *Proc. 6th East European Conf. Advances in Databases and Information Systems*, pages 218–231.

Akal, F., Türker, C., Schek, H.-J., Breitbart, Y., Grabs, T., and Veen, L. (2005). Fine-grained replication and scheduling with freshness and correctness guarantees. In *Proc. 31st Int. Conf. on Very Large Data Bases*, pages 565–576.

Akbarinia, R. and Martins, V. (2007). Data management in the APPA system. *J. Grid Comp.*, 5 (3):303–317.

Akbarinia, R., Martins, V., Pacitti, E., and Valduriez, P. (2006). Design and implementation of Atlas P2P architecture. In Baldoni, R., Cortese, G., and Davide, F., editors, *Global Data Management*, pages 98–123. IOS Press.

Akbarinia, R., Pacitti, E., and Valduriez, P. (2007a). Processing top-k queries in distributed hash tables. In *Proc. 13th Int. Euro-Par Conf.*, pages 489–502.

Akbarinia, R., Pacitti, E., and Valduriez, P. (2007b). Query processing in P2P systems. Technical Report 6112, INRIA, Rennes, France.

Akbarinia, R., Pacitti, E., and Valduriez, P. (2007c). Best position algorithms for top-k queries. In *Proc. 33rd Int. Conf. on Very Large Data Bases*, pages 495–506.

Akbarinia, R., Pacitti, E., and Valduriez, P. (2007d). Data currency in replicated dhts. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 211–222.

Akidau, T., Balikov, A., Bekiroglu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., and Whittle, S. (2013). MillWheel: Fault-tolerant stream processing at internet scale. *Proc. VLDB Endowment*, 6(11):1033–1044.

Alagiannis, I., Borovica, R., Branco, M., Idreos, S., and Ailamaki, A. (2012). NoDB: efficient query execution on raw data files. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 241–252.

Alagiannis, I., Idreos, S., and Ailamaki, A. (2014). H2O: A hands-free adaptive store. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1103–1114.

Alamoudi, A. A., Grover, R., Carey, M. J., and Borkar, V. R. (2015). External data access and indexing in AsterixDB. In *Proc. 24th ACM Int. Conf. on Information and Knowledge Management*, pages 3–12.

Albutiu, M.-C., Kemper, A., and Neumann, T. (2012). Massively parallel sort-merge joins in main memory multi-core database systems. *Proc. VLDB Endowment*, 5(10):1064–1075.

Allard, T., Hébrail, G., Masseglia, F., and Pacitti, E. (2015). Chiaroscuro: Transparency and privacy for massive personal time-series clustering. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 779–794.

Alomari, M., Cahill, M., Fekete, A., and Rohm, U. (2008). The cost of serializability on platforms that use snapshot isolation. In *Proc. 24th Int. Conf. on Data Engineering*, pages 576 –585.

Alomari, M., Fekete, A., and Rohm, U. (2009). A robust technique to ensure serializable executions with snapshot isolation DBMS. In *Proc. 25th Int. Conf. on Data Engineering*, pages 341–352.

Alsberg, P. A. and Day, J. D. (1976). A principle for resilient sharing of distributed resources. In *Proc. 2nd Int. Conf. on Software Engineering*, pages 562–570.

Alsubaiee, S., Altowim, Y., Altwaijry, H., Behm, A., Borkar, V. R., Bu, Y., Carey, M. J., Cetindil, I., Cheelangi, M., Faraaz, K., Gabrielova, E., Grover, R., Heilbron, Z., Kim, Y., Li, C., Li, G., Ok, J. M., Onose, N., Pirzadeh, P., Tsotras, V. J., Vernica, R., Wen, J., and Westmann, T. (2014). AsterixDB: A scalable, open source DBMS. *Proc. VLDB Endowment*, 7(14):1905–1916.

Altingövde, I. S. and Ulusoy, Ö. (2004). Exploiting interclass rules for focused crawling. *IEEE Intelligent Systems*, 19(6):66–73.

Aluç, G. (2015). *Workload Matters: A Robust Approach to Physical RDF Database Design*. PhD thesis, University of Waterloo.

Alvarez, V., Schuhknecht, F. M., Dittrich, J., and Richter, S. (2014). Main memory adaptive indexing for multi-core systems. In *Proc. 10th Workshop on Data Management on New Hardware*, pages 3:1—-3:10.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proc. Spring Joint Computer Conf.*, pages 483–485.

Amsaleg, L., Franklin, M. J., Tomasic, A., and Urhan, T. (1996). Scrambling query plans to cope with unexpected delays. In *Proc. 4th Int. Conf. on Parallel and Distributed Information Systems*, pages 208–219.

Andreev, K. and Racke, H. (2006). Balanced graph partitioning. *Theor. Comp. Sci.*, 39(6):929–939.

Angles, R. and Gutierrez, C. (2008). The expressive power of SPARQL. In *Proc. 7th Int. Semantic Web Conf.*, pages 114–129.

Antoniou, G. and Plexousakis, D. (2018). Semantic web. In Liu, L. and Özsu, M. T., editors, *Encyclopedia of Database Systems*, pages 3425–3429. Springer New York, New York, NY.

Apache. (2016). Apache Giraph. http://giraph.apache.org. Last accessed June 2019.

Apers, P., van den Berg, C., Flokstra, J., Grefen, P., Kersten, M., and Wilschut, A. (1992). Prisma/DB: a parallel main-memory relational DBMS. *IEEE Trans. Knowl. and Data Eng.*, 4:541–554.

Apers, P. M. G. (1981). Redundant allocation of relations in a communication network. In *Proc. 5th Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 245–258.

Arasu, A. and Widom, J. (2004). A denotational semantics for continuous queries over streams and relations. *ACM SIGMOD Rec.*, 33(3):6–11.

Arasu, A., Cho, J., Garcia-Molina, H., Paepcke, A., and Raghavan, S. (2001). Searching the web. *ACM Trans. Internet Tech.*, 1(1):2–43.

Arasu, A., Babu, S., and Widom, J. (2006). The CQL continuous query language: Semantic foundations and query execution. *VLDB J.*, 15(2):121–142.

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., and Zaharia, M. (2015). Spark SQL: Relational data processing in Spark. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1383–1394.

Arocena, G. and Mendelzon, A. (1998). WebOQL: Restructuring documents, databases and webs. In *Proc. 14th Int. Conf. on Data Engineering*, pages 24–33.

Asad, O. and Kemme, B. (2016). Adaptcache: Adaptive data partitioning and migration for distributed object caches. In *Proc. ACM/IFIP/USENIX 17th Int. Middleware Conf.*, pages 7:1–7:13.

Aspnes, J. and Shah, G. (2003). Skip graphs. In *Proc. 14th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 384–393.

Avnur, R. and Hellerstein, J. (2000). Eddies: Continuously adaptive query processing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 261–272.

Ayad, A. and Naughton, J. (2004). Static optimization of conjunctive queries with sliding windows over unbounded streaming information sources. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 419–430.

Azar, Y., Broder, A. Z., Karlin, A. R., and Upfal, E. (1999). Balanced allocations. *SIAM J. on Comput.*, 29(1):180–200.

Babb, E. (1979). Implementing a relational database by means of specialized hardware. *ACM Trans. Database Syst.*, 4(1):1–29.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 1–16.

Balazinska, M., Kwon, Y., Kuchta, N., and Lee, D. (2007). Moirae: History-enhanced monitoring. In *Proc. 3rd Biennial Conf. on Innovative Data Systems Research*, pages 375–386.

Balke, W.-T., Nejdl, W., Siberski, W., and Thaden, U. (2005). Progressive distributed top-k retrieval in peer-to-peer networks. In *Proc. 21st Int. Conf. on Data Engineering*, pages 174–185.

Bancilhon, F. and Spyratos, N. (1981). Update semantics of relational views. *ACM Trans. Database Syst.*, 6(4):557–575.

Barbara, D., Garcia-Molina, H., and Spauster, A. (1986). Policies for dynamic vote reassignment. In *Proc. 6th IEEE Int. Conf. on Distributed Computing Systems*, pages 37–44.

Barbara, D., Molina, H. G., and Spauster, A. (1989). Increasing availability under mutual exclusion constraints with dynamic voting reassignment. *ACM Trans. Comp. Syst.*, 7(4):394–426.

Barthels, C., Loesing, S., Alonso, G., and Kossmann, D. (2015). Rack-scale in-memory join processing using RDMA. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1463–1475.

Batini, C. and Lenzirini, M. (1984). A methodology for data schema integration in entity-relationship model. *IEEE Trans. Softw. Eng.*, SE-10(6):650–654.

Batini, C., Lenzirini, M., and Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364.

Beeri, C., Bernstein, P. A., and Goodman, N. (1989). A model for concurrency in nested transaction systems. *J. ACM*, 36(2):230–269.

Bell, D. and Grimson, J. (1992). *Distributed Database Systems*. Addison Wesley. Reading.

Bell, D. and Lapuda, L. (1976). Secure computer systems: Unified exposition and Multics interpretation. Technical Report MTR-2997 Rev.1, MITRE Corp, Bedford, MA.

Berenson, H., Bernstein, P., Gray, J., Melton, J., O'Neil, E., and O'Neil, P. (1995). A critique of ansi sql isolation levels. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1–10.

Bergamaschi, S. (2001). Semantic integration of heterogeneous information sources. *Data & Knowl. Eng.*, 36(3):215–249.

Bergman, M. K. (2001). The deep web: Surfacing hidden value. *J. Electronic Publishing*, 7(1).

Bergsten, B., Couprie, M., and Valduriez, P. (1991). Prototyping DBS3, a shared-memory parallel database system. In *Proc. Int. Conf. on Parallel and Distributed Information Systems*, pages 226–234.

Bergsten, B., Couprie, M., and Valduriez, P. (1993). Overview of parallel architectures for databases. *The Comp. J.*, 36(8):734–739.

Berkholz, C., Keppeler, J., and Schweikardt, N. (2017). Answering conjunctive queries under updates. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 303–318.

Berlin, J. and Motro, A. (2001). Autoplex: Automated discovery of content for virtual databases. In *Proc. Int. Conf. on Cooperative Inf. Syst.*, pages 108–122.

Berners-Lee, T. (2006). Linked data. Accessible at https://www.w3.org/DesignIssues/LinkedData.html. Last accessed June 2019.

Bernstein, P. and Blaustein, B. (1982). Fast methods for testing quantified relational calculus assertions. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 39–50.

Bernstein, P. and Melnik, S. (2007). Model management: 2.0: Manipulating richer mappings. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1–12.

Bernstein, P., Blaustein, B., and Clarke, E. M. (1980a). Fast maintenance of semantic integrity assertions using redundant aggregate data. In *Proc. 6th Int. Conf. on Very Data Bases*, pages 126–136.

Bernstein, P., Shipman, P., and Rothnie, J. B. (1980b). Concurrency control in a system for distributed databases (SDD-1). *ACM Trans. Database Syst.*, 5(1):18–51.

Bernstein, P. A. and Chiu, D. M. (1981). Using semi-joins to solve relational queries. *J. ACM*, 28 (1):25–40.

Bernstein, P. A. and Goodman, N. (1981). Concurrency control in distributed database systems. *ACM Comput. Surv.*, 13(2):185–222.

Bernstein, P. A. and Goodman, N. (1983). Multiversion concurrency control — theory and algorithms. *ACM Trans. Database Syst.*, 8(4):465–483.

Bernstein, P. A. and Goodman, N. (1984). An algorithm for concurrency control and recovery in replicated distributed databases. *ACM Trans. Database Syst.*, 9(4):596–615.

Bernstein, P. A. and Newcomer, E. (1997). *Principles of Transaction Processing for the Systems Professional*. Morgan Kaufmann.

Bernstein, P. A., Goodman, N., Wong, E., Reeve, C. L., and Jr, J. B. R. (1981). Query processing in a system for distributed databases (SDD-1). *ACM Trans. Database Syst.*, 6(4):602–625.

Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison Wesley.

Bernstein, P. A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., and Zaihrayeu, I. (2002). Data management for peer-to-peer computing : A vision. In *Proc. 5th Int. Workshop on the World Wide Web and Databases*, pages 89–94.

Bernstein, P. A., Fekete, A., Guo, H., Ramakrishnan, R., and Tamma, P. (2006). Relexed concurrency serializability for middle-tier caching and replication. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 599–610.

Beyer, K. S., Ercegovac, V., Krishnamurthy, R., Raghavan, S., Rao, J., Reiss, F., Shekita, E. J., Simmen, D. E., Tata, S., Vaithyanathan, S., and Zhu, H. (2009). Towards a scalable enterprise content analytics platform. *Q. Bull. IEEE TC on Data Eng.*, 32(1):28–35.

Bharat, K. and Broder, A. (1998). A technique for measuring the relative size and overlap of public web search engines. *Comp. Networks and ISDN Syst.*, 30:379 – 388. (Proc. 7th Int. World Wide Web Conf.).

Bhowmick, S. S., Madria, S. K., and Ng, W. K. (2004). *Web Data Management*. Springer.

Bifet, A., Gavaldà, R., Holmes, G., and Pfahringer, B. (2018). *Machine Learning for Data Streams: with Practical Examples in MOA*. MIT Press.

Binnig, C., Hildenbrand, S., Färber, F., Kossmann, D., Lee, J., and May, N. (2014). Distributed snapshot isolation: global transactions pay globally, local transactons pay locally. *VLDB J.*, 23:987–1011.

Biscondi, N., Brunie, L., Flory, A., and Kosch, H. (1996). Encapsulation of intra-operation parallelism in a parallel match operator. In *Proc. ACPC Conf.*, volume 1127 of *Lecture Notes in Computer Science*, pages 124–135.

Bitton, D., Boral, H., DeWitt, D. J., and Wilkinson, W. K. (1983). Parallel algorithms for the execution of relational database operations. *ACM Trans. Database Syst.*, 8(3):324–353.

Bitton, D., DeWitt, D. J., Hsiao, D. K., and Menon, J. (1984). A taxonomy of parallel sorting. *ACM Comput. Surv.*, 16(3):287–318.

Bizer, C., Vidal, M.-E., and Skaf-Molli, H. (2018). Linked open data. In Liu, L. and Özsu, M. T., editors, *Encyclopedia of Database Systems*, pages 2096–2101. Springer New York, New York, NY.

Blanas, S., Patel, J. M., Ercegovac, V., Rao, J., Shekita, E. J., and Tian, Y. (2010). A comparison of join algorithms for log processing in MapReduce. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 975–986.

Blaustein, B. (1981). *Enforcing Database Assertions: Techniques and Applications*. PhD thesis, Harvard University, Cambridge, Mass.

Bleiholder, J. and Naumann, F. (2009). Data fusion. *ACM Comput. Surv.*, 41(1):1:1–1:41.

Bonato, A. (2008). *A Course on the Web Graph*. American Mathematical Society.

Bondiombouy, C. and Valduriez, P. (2016). Query processing in multistore systems: an overview. *Int. J. Cloud Computing*, 5(4):309–346.

Bondiombouy, C., Kolev, B., Levchenko, O., and Valduriez, P. (2016). Multistore big data integration with CloudMdsQL. *Trans. Large-Scale Data- and Knowledge-Centered Syst.*, 28: 48–74.

Bonifati, A., Summa, G., Pacitti, E., and Draidi, F. (2014). Query reformulation in PDMS based on social relevance. *Trans. Large-Scale Data- and Knowledge-Centered Syst.*, 13:59–90.

Bonnet, P., Gehrke, J., and Seshadri, P. (2001). Towards sensor database systems. In *Proc. 2nd Int. Conf. on Mobile Data Management*, pages 3–14.

Bönström, V., Hinze, A., and Schweppe, H. (2003). Storing RDF as a graph. In *Proc. 1st Latin American Web Congress*, pages 27 – 36.

Boral, H. and DeWitt, D. (1983). Database machines: An idea whose time has passed? A critique of the future of database machines. In *Proc. 3rd Int. Workshop on Database Machines*, pages 166–187.

Boral, H., Alexander, W., Clay, L., Copeland, G., Danforth, S., Franklin, M., Hart, B., Smith, M., and Valduriez, P. (1990). Prototyping bubba, a highly parallel database system. *IEEE Trans. Knowl. and Data Eng.*, 2(1):4–24.

Borkar, D., Mayuram, R., Sangudi, G., and Carey, M. J. (2016). Have your data and query it too: From key-value caching to big data management. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 239–251.

Bornea, M. A., Dolby, J., Kementsietsidis, A., Srinivas, K., Dantressangle, P., Udrea, O., and Bhattacharjee, B. (2013). Building an efficient RDF store over a relational database. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 121–132.

Borr, A. (1988). High performance SQL through low-level system integration. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 342–349.

Bouganim, L., Florescu, D., and Valduriez, P. (1996). Dynamic load balancing in hierarchical parallel database systems. In *Proc. 22th Int. Conf. on Very Large Data Bases*, pages 436–447.

Bouganim, L., Florescu, D., and Valduriez, P. (1999). Multi-join query execution with skew in NUMA multiprocessors. *Distrib. Parall. Databases*, 7(1). in press.

Breitbart, Y. and Korth, H. F. (1997). Replication and consistency: Being lazy helps sometimes. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 173–184.

Breitbart, Y. and Silberschatz, A. (1988). Multidatabase update issues. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 135–142.

Breitbart, Y., Olson, P. L., and Thompson, G. R. (1986). Database integration in a distributed heterogeneous database system. In *Proc. 2nd Int. Conf. on Data Engineering*, pages 301–310.

Brewer, E., Ying, L., Greenfield, L., Cypher, R., and T'so, T. (2016). Disks for data centers. Technical report, Google.

Brewer, E. A. (2000). Towards robust distributed systems (abstract). In *Proc. ACM SIGACT-SIGOPS 19th Symp. on the Principles of Distributed Computing*, page 7.

Bright, M. W., Hurson, A. R., and Pakzad, S. H. (1994). Automated resolution of semantic heterogeneity in multidatabases. *ACM Trans. Database Syst.*, 19(2):212–253.

Brill, D., Templeton, M., and Yu, C. (1984). Distributed query processing strategies in MERMAID: A front-end to data management systems. In *Proc. 1st Int. Conf. on Data Engineering*, pages 211–218.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Comp. Netw.*, 30(1-7):107 – 117.

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000). Graph structure in the web. *Comp. Netw.*, 33(1-6):309–320.

Broekstra, J., Kampman, A., and van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying RDF and RDF schema. In *Proc. 1st Int. Semantic Web Conf.*, pages 54–68.

Bu, Y., Howe, B., Balazinska, M., and Ernst, M. D. (2010). HaLoop: efficient iterative data processing on large clusters. *Proc. VLDB Endowment*, 3(1):285–296.

Bu, Y., Howe, B., Balazinska, M., and Ernst, M. D. (2012). The HaLoop approach to large-scale iterative data analysis. *VLDB J.*, 21(2):169–190.

Bu, Y., Borkar, V. R., Jia, J., Carey, M. J., and Condie, T. (2014). Pregelix: Bigger graph analytics on a dataflow engine. *Proc. VLDB Endowment*, 8(2):161–172.

Bugiotti, F., Bursztyn, D., Deutsch, A., Ileana, I., and Manolescu, I. (2015). Invisible glue: Scalable self-tunning multi-stores. In *Proc. 7th Biennial Conf. on Innovative Data Systems Research*.

Buneman, P., Davidson, S., Hillebrand, G. G., and Suciu, D. (1996). A query language and optimization techniques for unstructured data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 505–516.

Cahill, M. J., Röhm, U., and Fekete, A. D. (2009). Serializable isolation for snapshot databases. *ACM Trans. Database Syst.*, 34(4):Article 20.

Calì, A. and Calvanese, D. (2002). Optimized querying of integrated data over the web. In *Engineering Information Systems in the Internet Context*, pages 285–301.

Callan, J. P. and Connell, M. E. (2001). Query-based sampling of text databases. *ACM Trans. Information Syst.*, 19(2):97–130.

Callan, J. P., Connell, M. E., and Du, A. (1999). Automatic discovery of language models for text databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 479–490.

Cammert, M., Krämer, J., Seeger, B., and S.Vaupel. (2006). An approach to adaptive memory management in data stream systems. In *Proc. 22nd Int. Conf. on Data Engineering*, page 137.

Canaday, R. H., Harrisson, R. D., Ivie, E. L., Rydery, J. L., and Wehr, L. A. (1974). A back-end computer for data base management. *Commun. ACM*, 17(10):575–582.

Cao, P. and Wang, Z. (2004). Query processing issues in image (multimedia) databases. In *Proc. ACM SIGACT-SIGOPS 23rd Symp. on the Principles of Distributed Computing*, pages 206–215.

Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., and Tzoumas, K. (2015). Apache Flink™: Stream and batch processing in a single engine. *Q. Bull. IEEE TC on Data Eng.*, 38 (4):28–38.

Carey, M. and Lu, H. (1986). Load balancing in a locally distributed database system. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 108–119.

Castano, S. and Antonellis, V. D. (1999). A schema analysis and reconciliation tool environment for heterogeneous databases. In *Proc. 3rd Int. Conf. on Database Eng. and Applications*, pages 53 – 62.

Castano, S., Fugini, M. G., Martella, G., and Samarati, P. (1995). *Database Security*. Addison Wesley.

Castro, M. and Liskov, B. (1999). Practical byzantine fault tolerance. In *Proc. 3rd USENIX Symp. on Operating System Design and Implementation*, pages 173–186.

Cellary, W., Gelenbe, E., and Morzy, T. (1988). *Concurrency Control in Distributed Database Systems*. North-Holland.

Ceri, S. and Owicki, S. (1982). On the use of optimistic methods for concurrency control in distributed databases. In *Proc. 6th Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 117–130.

Ceri, S. and Pelagatti, G. (1983). Correctness of query execution strategies in distributed databases. *ACM Trans. Database Syst.*, 8(4):577–607.

Ceri, S. and Pernici, B. (1985). DATAID–D: Methodology for distributed database design. In Albano, V. d. A. and di Leva, A., editors, *Computer-Aided Database Design*, pages 157–183. North-Holland.

Ceri, S. and Widom, J. (1993). Managing semantic heterogeneity with production rules and persistent queues. In *Proc. 19th Int. Conf. on Very Large Data Bases*, pages 108–119.

Ceri, S., Martella, G., and Pelagatti, G. (1982a). Optimal file allocation in a computer network: A solution method based on the knapsack problem. *Comp. Netw.*, 6:345–357.

Ceri, S., Negri, M., and Pelagatti, G. (1982b). Horizontal data partitioning in database design. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 128–136.

Ceri, S., Navathe, S. B., and Wiederhold, G. (1983). Distribution design of logical database schemes. *IEEE Trans. Softw. Eng.*, SE-9(4):487–503.

Ceri, S., Gottlob, G., and Pelagatti, G. (1986). Taxonomy and formal properties of distributed joins. *Inf. Syst.*, 11(1):25–40.

Ceri, S., Pernici, B., and Wiederhold, G. (1987). Distributed database design methodologies. *Proc. IEEE*, 75(5):533–546.

Chairunnanda, P., Daudjee, K., and Özsu, M. T. (2014). ConfluxDB: multi-master replication for partitioned snapshot isolation databases. *Proc. VLDB Endowment*, 7(11):947–958.

Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M. (2002). Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27.

Chakrabarti, S., Dom, B., and Indyk, P. (1998). Enhanced hypertext classification using hyperlinks. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 307 – 318.

Chamberlin, D. (2018). *SQL++ For SQL Users: A Tutorial*. CouchBase Inc.

Chamberlin, D., Gray, J., and Traiger, I. (1975). Views, authorization and locking in a relational database system. In *Proc. National Computer Conf*, pages 425–430.

Chambers, C., Raniwala, A., Perry, F., Adams, S., Henry, R. R., Bradshaw, R., and Weizenbaum, N. (2010). FlumeJava: easy, efficient data-parallel pipelines. In *Proc. ACM SIGPLAN 2010 Conf. on Programming Language Design and Implementation*, pages 363–375.

Chandra, T. D., Griesemer, R., and Redstone, J. (2007). Paxos made live: An engineering perspective. In *Proc. ACM SIGACT-SIGOPS 26th Symp. on the Principles of Distributed Computing*, pages 398–407.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., and Shah, M. A. (2003). TelegraphCQ: Continuous dataflow processing for an uncertain world. In *Proc. 1st Biennial Conf. on Innovative Data Systems Research*.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. (2008). Bigtable: A distributed storage system for structured data. *ACM Trans. Comp. Syst.*, 26(2):Article 4.

Chang, S. K. and Liu, A. C. (1982). File allocation in a distributed database. *Int. J. Comput. Inf. Sci*, 11(5):325–340.

Chattopadhyay, B., Lin, L., Liu, W., Mittal, S., Aragonda, P., Lychagina, V., Kwon, Y., and Wong, M. (2011). Tenzing: A SQL implementation on the MapReduce framework. *Proc. VLDB Endowment*, 4(12):1318–1327.

Chaudhuri, S., Ganjam, K., Ganti, V., and Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 313–324.

Chen, R., Shi, J., Chen, Y., and Chen, H. (2015). PowerLyra: Differentiated graph computation and partitioning on skewed graphs. In *Proc. 10th ACM SIGOPS/EuroSys European Conf. on Comp. Syst.*, pages 1:1–1:15.

Chiu, D. M. and Ho, Y. C. (1980). A methodology for interpreting tree queries into optimal semi-join expressions. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 169–178.

Cho, J. and Garcia-Molina, H. (2000). The evolution of the web and implications for an incremental crawler. In *Proc. 26th Int. Conf. on Very Large Data Bases*.

Cho, J. and Ntoulas, A. (2002). Effective change detection using sampling. In *Proc. 28th Int. Conf. on Very Large Data Bases*.

Cho, J., Garcia-Molina, H., and Page, L. (1998). Efficient crawling through URL ordering. *Comp. Networks and ISDN Syst.*, 30(1-7):161–172.

Chockler, G., Keidar, I., and Vitenberg, R. (2001). Group communication specifications: a comprehensive study. *ACM Comput. Surv.*, 33(4):427–469.

Chong, E., Das, S., Eadon, G., and Srinivasan, J. (2005). An efficient SQL-based RDF querying scheme. In *Proc. 31st Int. Conf. on Very Large Data Bases*, pages 1216–1227.

Chu, W. W. (1969). Optimal file allocation in a multiple computer system. *IEEE Trans. Comput.*, C-18(10):885–889.

Chu, W. W. (1973). Optimal file allocation in a computer network. In Abramson, N. and Kuo, F. F., editors, *Computer Communication Networks*, pages 82–94.

Chu, W. W. (1976). Performance of file directory systems for data bases in star and distributed networks. In *Proc. National Computer Conf.*, volume 45, pages 577–587.

Chu, W. W. and Nahouraii, E. E. (1975). File directory design considerations for distributed databases. In *Proc. 1st Int. Conf. on Very Data Bases*, pages 543–545.

Chu, X., Ilyas, I. F., and Papotti, P. (2013). Discovering Denial Constraints. *Proc. VLDB Endowment*, 6(13):1498–1509.

Chundi, P., Rosenkrantz, D. J., and Ravi, S. S. (1996). Deferred updates and data placement in distributed databases. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 469–476.

Civelek, F. N., Dogac, A., and Spaccapietra, S. (1988). An expert system approach to view definition and integration. In *Proc. 7th Int'l. Conf. on Entity-Relationship Approach*, pages 229–249.

Cohen, J. (2009). Graph twiddling in a MapReduce world. *Computing in Science & Engineering*, 11(4):29–41.

Cole, R. L. and Graefe, G. (1994). Optimization of dynamic query evaluation plans. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 150–160.

Coletta, R., Castanier, E., Valduriez, P., Frisch, C., Ngo, D., and Bellahsene, Z. (2012). Public data integration with websmatch. In *Proc. Int. Workshop on Open Data*, pages 5–12.

Copeland, G., Alexander, W., Boughter, E., and Keller, T. (1988). Data placement in bubba. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 99–108.

Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J. J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., Hsieh, W., Kanthak, S., Kogan, E., Li, H., Lloyd, A., Melnik, S., Mwaura, D., Nagle, D., Quinlan, S., Rao, R., Rolig, L., Saito, Y., Szymaniak, M., Taylor, C., Wang, R., and Woodford, D. (2013). Spanner: Google's globally distributed database. *ACM Trans. Database Syst.*, 31(3):8:1–8:22.

Crainiceanu, A., Linga, P., Gehrke, J., and Shanmugasundaram, J. (2004). Querying peer-to-peer networks using p-trees. In *Proc. 7th Int. Workshop on the World Wide Web and Databases*, pages 25–30.

Cranor, C., Johnson, T., Spatscheck, O., and Shkapenyuk, V. (2003). Gigascope: High performance network monitoring with an SQL interface. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 647–651.

Crespo, A. and Garcia-Molina, H. (2002). Routing indices for peer-to-peer systems. In *Proc. 22nd IEEE Int. Conf. on Distributed Computing Systems*, pages 23–33.

Cuenca-Acuna, F., Peery, C., Martin, R., and Nguyen, T. (2003). PlanetP: using gossiping to build content addressable peer-to-peer information sharing communities. In *IEEE Int. Symp. on High Performance Distributed Computing*, pages 236–249.

Curino, C., Jones, E., Zhang, Y., and Madden, S. (2010). Schism: a workload-driven approach to database replication and partitioning. *Proc. VLDB Endowment*, 3(1):48–57.

Curino, C., Jones, E. P. C., Madden, S., and Balakrishnan, H. (2011). Workload-aware database monitoring and consolidation. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 313–324.

Cusumano, M. A. (2010). Cloud computing and SaaS as new computing platforms. *Commun. ACM*, 53(4):27–29.

Dasgupta, S., Coakley, K., and Gupta, A. (2016). Analytics-driven data ingestion and derivation in the AWESOME polystore. In *Proc. 2016 IEEE Int. Conf. on Big Data*, pages 2555–2564.

Daswani, N., Garcia-Molina, H., and Yang, B. (2003). Open problems in data-sharing peer-to-peer systems. In *Proc. 9th Int. Conf. on Database Theory*, pages 1–15.

Daudjee, K. and Salem, K. (2004). Lazy database replication with ordering guarantees. In *Proc. 20th Int. Conf. on Data Engineering*, pages 424–435.

Daudjee, K. and Salem, K. (2006). Lazy database replication with snapshot isolation. In *Proc. 32nd Int. Conf. on Very Large Data Bases*, pages 715–726.

Davenport, R. A. (1981). Design of distributed data base systems. *Comp. J.*, 24(1):31–41.

Davidson, S. B. (1984). Optimism and consistency in partitioned distributed database systems. *ACM Trans. Database Syst.*, 9(3):456–481.

Davidson, S. B., Garcia-Molina, H., and Skeen, D. (1985). Consistency in partitioned networks. *ACM Comput. Surv.*, 17(3):341–370.

Dawson, J. L. (1980). A user demand model for distributed database design. In *Digest of Papers – COMPCON*, pages 211–216.

Dayal, U. and Bernstein, P. (1978). On the updatability of relational views. In *Proc. 4th Int. Conf. on Very Data Bases*, pages 368–377.

Dayal, U. and Hwang, H. (1984). View definition and generalization for database integration in MULTIBASE: A system for heterogeneous distributed database. *IEEE Trans. Softw. Eng.*, SE-10(6):628–644.

Dean, J. and Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. In *Proc. 6th USENIX Symp. on Operating System Design and Implementation*, pages 137–149.

Dean, J. and Ghemawat, S. (2010). MapReduce: a flexible data processing tool. *Commun. ACM*, 53(1):72–77.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. In *Proc. 21st ACM Symp. on Operating System Principles*, pages 205–220.

Demers, A. J., Greene, D. H., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H. E., Swinehart, D. C., and Terry, D. B. (1987). Epidemic algorithms for replicated database maintenance. In *Proc. ACM SIGACT-SIGOPS 6th Symp. on the Principles of Distributed Computing*, pages 1–12.

Deshpande, A. and Gupta, A. (2018). *Principles of Graph Data Management and Analytics*. ACM Books. Forthcoming.

Devine, R. (1993). Design and implementation of DDH: A distributed dynamic hashing algorithm. In *Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms*, pages 101–114.

Dewitt, D. and Stonebraker, M. (2009). MapReduce: A major step backwards. https://homes.cs.washington.edu/~billhowe/mapreduce_a_major_step_backwards.html.

DeWitt, D., Naughton, J., Schneider, D., and Seshadri, S. (1992). Practical skew handling in parallel joins. In *Proc. 22th Int. Conf. on Very Large Data Bases*, pages 27–40.

DeWitt, D. J. and Gerber, R. (1985). Multi processor hash-based join algorithms. In *Proc. 11th Int. Conf. on Very Large Data Bases*, pages 151–164.

DeWitt, D. J. and Gray, J. (1992). Parallel database systems: The future of high performance database systems. *Commun. ACM*, 35(6):85–98.

DeWitt, D. J., Katz, R., Olken, F., Shapiro, L., Stonebraker, M., and Wood, D. (1984). Implementation techniques for main memory database systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1–8.

DeWitt, D. J., Gerber, R. H., Graek, G., Heytens, M. L., Kumar, K. B., and Muralikrishna, M. (1986). Gamma: A high performance dataflow database machine. In *Proc. 12th Int. Conf. on Very Large Data Bases*, pages 228–237.

DeWitt, D. J., Paulson, E., Robinson, E., Naughton, J., Royalty, J., Shankar, S., and Krioukov, A. (2008). Clustera: an integrated computation and data management system. *Proc. VLDB Endowment*, 1:28–41.

DeWitt, D. J., Halverson, A., Nehme, R. V., Shankar, S., Aguilar-Saborit, J., Avanes, A., Flasza, M., and Gramling, J. (2013). Split query processing in Polybase. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1255–1266.

Dhamankar, R., Lee, Y., Doan, A., Halevy, A. Y., and Domingos, P. (2004). iMAP: Discovering complex mappings between database schemas. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 383–394.

Ding, L. and Rundensteiner, E. (2004). Evaluating window joins over punctuated streams. In *Proc. 13th ACM Int. Conf. on Information and Knowledge Management*, pages 98–107.

Ding, L., Mehta, N., Rundensteiner, E., and Heineman, G. (2004). Joining punctuated streams. In *Advances in Database Technology, Proc. 9th Int. Conf. on Extending Database Technology*, pages 587–604.

Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., and Wang, J. (2018). Untangling blockchain: A data processing view of blockchain systems. *IEEE Trans. Knowl. and Data Eng.*, 30(7):1366–1385.

Do, H. and Rahm, E. (2002). COMA: a system for flexible combination of schema matching approaches. In *Proc. 28th Int. Conf. on Very Large Data Bases*, pages 610–621.

Doan, A. and Halevy, A. Y. (2005). Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94.

Doan, A., Domingos, P., and Halevy, A. Y. (2001). Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 509–520.

Doan, A., Domingos, P., and Halevy, A. (2003a). Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3):279–301.

Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., and Halevy, A. (2003b). Learning to match ontologies on the semantic web. *VLDB J.*, 12(4):303–319.

Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of Data Integration*. Morgan Kaufmann.

Dogac, A., Kalinichenko, L., Özsu, M. T., and Sheth, A., editors. (1998). *Advances in Workflow Systems and Interoperability*. Springer.

Dong, X. L. and Naumann, F. (2009). Data fusion: resolving data conflicts for integration. *Proc. VLDB Endowment*, 2(2):1654–1655.

Dong, X. L. and Srivastava, D. (2015). *Big Data Integration*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Dong, X. L., Berti-Equille, L., and Srivastava, D. (2009a). Truth discovery and copying detection in a dynamic world. *Proc. VLDB Endowment*, 2(1):562–573.

Dong, X. L., Berti-Equille, L., and Srivastava, D. (2009b). Integrating conflicting data: the role of source dependence. *Proc. VLDB Endowment*, 2(1):550–561.

Dowdy, L. W. and Foster, D. V. (1982). Comparative models of the file assignment problem. *ACM Comput. Surv.*, 14(2):287–313.

Du, W., Krishnamurthy, R., and Shan, M. (1992). Query optimization in a heterogeneous DBMS. In *Proc. 18th Int. Conf. on Very Large Data Bases*, pages 277–291.

Du, W., Shan, M., and Dayal, U. (1995). Reducing multidatabase query response time by tree balancing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 293–303.

Duggan, J., Elmore, A. J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., Madden, S., Maier, D., Mattson, T., and Zdonik, S. B. (2015). The BigDAWG polystore system. *ACM SIGMOD Rec.*, 44(2):11–16.

Duschka, O. M. and Genesereth, M. R. (1997). Answering recursive queries using views. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 109–116.

Eager, D. L. and Sevcik, K. C. (1983). Achieving robustness in distributed database systems. *ACM Trans. Database Syst.*, 8(3):354–381.

Edwards, J., McCurley, K., and Tomlin, J. (2001). An adaptive model for optimizing performance of an incremental web crawler. In *Proc. 10th Int. World Wide Web Conf.*

El Abbadi, A., Skeen, D., and Cristian, F. (1985). An efficient, fault–tolerant protocol for replicated data management. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 215–229.

Elbushra, M. M. and Lindström, J. (2015). Causal consistent databases. *Open Journal of Databases*, 2(1):17–35.

Elmagarmid, A., Rusinkiewicz, M., and Sheth, A., editors. (1999). *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann.

Elmagarmid, A. K. (1986). A survey of distributed deadlock detection algorithms. *ACM SIGMOD Rec.*, 15(3):37–45.

Elmagarmid, A. K., editor. (1992). *Transaction Models for Advanced Database Applications*. Morgan Kaufmann.

Elmagarmid, A. K., Soundararajan, N., and Liu, M. T. (1988). A distributed deadlock detection and resolution algorithm and its correctness proof. *IEEE Trans. Softw. Eng.*, 14(10):1443–1452.

Elmasri, R., Larson, J., and Navathe, S. B. (1987). Integration algorithms for database and logical database design. Technical report, Honeywell Corporate Research Center, Golden Valley, Minn.

Elmore, A. J., Arora, V., Taft, R., Pavlo, A., Agrawal, D., and El Abbadi, A. (2015). Squall: Fine-grained live reconfiguration for partitioned main memory databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 299–313.

Elseidy, M., Elguindy, A., Vitorovic, A., and Koch, C. (2014). Scalable and adaptive online joins. *Proc. VLDB Endowment*, 7(6):441–452.

Embley, D. W., Jackman, D., and Xu, L. (2001). Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Proc. Workshop on Information Integration on the Web*, pages 110–117.

Embley, D. W., Jackman, D., and Xu, L. (2002). Attribute match discovery in information integration: exploiting multiple facets of metadata. *Journal of the Brazilian Computing Society*, 8(2):32–43.

Epstein, R., Stonebraker, M., and Wong, E. (1978). Query processing in a distributed relational database system. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 169–180.

Eswaran, K. P. (1974). Placement of records in a file and file allocation in a computer network. In *Information Processing '74*, pages 304–307.

Etzion, O. and Niblett, P. (2010). *Event Processing in Action*. Manning.

Evrendilek, C., Dogac, A., Nural, S., and Ozcan, F. (1997). Multidatabase query optimization. *Distrib. Parall. Databases*, 5(1):77–114.

Eyal, I., Gencer, A. E., Sirer, E. G., and van Renesse, R. (2016). Bitcoin-ng: A scalable blockchain protocol. In *Proc. 13th USENIX Symp. on Networked Systems Design & Implementation*, pages 45–59.

Fagin, R. (2002). Combining fuzzy information: an overview. *ACM SIGMOD Rec.*, 31(2):109–118.

Fagin, R., Lotem, J., and Naor, M. (2003). Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66(4):614–656.

Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005). Data exchange: semantics and query answering. *Theor. Comp. Sci.*, 336(1):89–124.

Faleiro, J. M. and Abadi, D. J. (2015). Rethinking serializable multiversion concurrency control. *Proc. VLDB Endowment*, 8(11):1190–1201.

Faloutsos, C. and Christodoulakis, S. (1984). Signature files: an access method for documents and its analytical performance evaluation. *ACM Trans. Information Syst.*, 2(4):267–288.

Farid, M. H., Roatis, A., Ilyas, I. F., Hoffmann, H., and Chu, X. (2016). CLAMS: bringing quality to data lakes. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 2089–2092.

Farrag, A. A. and Özsu, M. T. (1989). Using semantic knowledge of transactions to increase concurrency. *ACM Trans. Database Syst.*, 14(4):503–525.

Fekete, A., Lynch, N., Merritt, M., and Weihl, W. (1987a). Nested transactions and read/write locking. Technical Memo MIT/LCS/TM–324, Massachusetts Institute of Technology, Cambridge, Mass.

Fekete, A., Lynch, N., Merritt, M., and Weihl, W. (1987b). Nested transactions, conflict-based locking, and dynamic atomicity. Technical Memo MIT/LCS/TM–340, Massachusetts Institute of Technology, Cambridge, Mass.

Fekete, A., Lynch, N., Merritt, M., and Weihl, W. (1989). Commutativity-based locking for nested transactions. Technical Memo MIT/LCS/TM-370b, Massachusetts Institute of Technology, Cambridge, Mass.

Fernandez, M., Florescu, D., and Levy, A. (1997). A query language for a web-site management system. *ACM SIGMOD Rec.*, 26(3):4–11.

Fernandez, R. C., Migliavacca, M., Kalyvianaki, E., and Pietzuch, P. (2013). Integrating scale out and fault tolerance in stream processing using operator state management. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 725–736.

Fernández-Moctezuma, R., Tufte, K., and Li, J. (2009). Inter-operator feedback in data stream management systems via punctuation. In *Proc. 4th Biennial Conf. on Innovative Data Systems Research*.

Ferraiolo, D. and Kuhn, R. (1992). Role-based access control. In *Proc. National Computer Conf.*, pages 554–563.

Fisher, M. K. and Hochbaum, D. S. (1980). Database location in computer networks. *J. ACM*, 27 (4):718–735.

Fisher, P. S., Hollist, P., and Slonim, J. (1980). A design methodology for distributed data bases. In *Digest of Papers – COMPCON*, pages 199–202.

Florentin, J. J. (1974). Consistency auditing of databases. *Comp. J.*, 17(1):52–58.

Florescu, D., Levy, A., and Mendelzon, A. (1998). Database techniques for the World-Wide Web: a survey. *ACM SIGMOD Rec.*, 27(3):59–74.

Friedman, M., Levy, A. Y., and Millstein, T. D. (1999). Navigational plans for data integration. In *Proc. 16th National Conf. on Artificial Intelligence and 11th Innovative Applications of Artificial Intelligence Conf.*, pages 67–73.

Fu, Y., Ong, K. W., Papakonstantinou, Y., and Zamora, E. (2014). FORWARD: data-centric UIs using declarative templates that efficiently wrap third-party JavaScript components. *Proc. VLDB Endowment*, 7(13):1649–1652.

Furtado, C., Lima, A. A. B., Pacitti, E., Valduriez, P., and Mattoso, M. (2008). Adaptive hybrid partitioning for OLAP query processing in a database cluster. *Int. Journal of High Performance Computing and Networking*, 5(4):251–262.

Fushimi, S., Kitsuregawa, M., and Tanaka, H. (1986). An overview of the system software of a parallel relational database machine GRACE. In *Proc. 12th Int. Conf. on Very Large Data Bases*, pages 209–219.

Gadepally, V., Chen, P., Duggan, J., Elmore, A. J., Haynes, B., Kepner, J., Madden, S., Mattson, T., and Stonebraker, M. (2016). The BigDAWG polystore system and architecture. In *Proc. IEEE High Performance Extreme Computing Conf.*, pages 1–6.

Galhardas, H., Florescu, D., Shasha, D., Simon, E., and Saita, C.-A. (2001). Declarative data cleaning: Language, model, and algorithms. In *Proc. 27th Int. Conf. on Very Large Data Bases*, pages 371–380.

Gançarski, S., Naacke, H., Pacitti, E., and Valduriez, P. (2007). The leganet system: Freshness-aware transaction routing in a database cluster. *Inf. Syst.*, 32(7):320–343.

Ganesan, P., Yang, B., and Garcia-Molina, H. (2004). One torus to rule them all: Multidimensional queries in P2P systems. In *Proc. 7th Int. Workshop on the World Wide Web and Databases*, pages 19–24.

Gankidi, V. R., Teletia, N., Patel, J. M., Halverson, A., and DeWitt, D. J. (2014). Indexing HDFS data in PDW: splitting the data from the index. *Proc. VLDB Endowment*, 7(13):1520–1528.

Garcia-Molina, H. (1982). Elections in distributed computing systems. *IEEE Trans. Comput.*, C-31(1):48–59.

Garcia-Molina, H. (1983). Using semantic knowledge for transaction processing in a distributed database. *ACM Trans. Database Syst.*, 8(2):186–213.

Garcia-Molina, H. and Salem, K. (1987). Sagas. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 249–259.

Garcia-Molina, H. and Wiederhold, G. (1982). Read–only transactions in a distributed database. *ACM Trans. Database Syst.*, 7(2):209–234.

Garcia-Molina, H., Gawlick, D., Klein, J., Kleissner, K., and Salem, K. (1990). Coordinating multi-transaction activities. Technical Report CS-TR-247-90, Department of Computer Science, Princeton University.

Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J. D., Vassalos, V., and Widom, J. (1997). The TSIMMIS approach to mediation: Data models and languages. *J. Intell. Information Syst.*, 8(2):117–132.

Garofalakis, M. N. and Ioannidis, Y. E. (1996). Multi-dimensional resource scheduling for parallel queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 365–376.

Gavish, B. and Pirkul, H. (1986). Computer and database location in distributed computer systems. *IEEE Trans. Comput.*, C-35(7):583–590.

Gedik, B. (2014). Partitioning functions for stateful data parallelism in stream processing. *VLDB J.*, 23:517–539.

Georgakopoulos, D., Hornick, M., and Sheth, A. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distrib. Parall. Databases*, 3: 119–153.

Ghemawat, S., Gobioff, H., and Leung, S. (2003). The Google file system. In *Proc. 19th ACM Symp. on Operating System Principles*, pages 29–43.

Ghoting, A., Krishnamurthy, R., Pednault, E. P. D., Reinwald, B., Sindhwani, V., Tatikonda, S., Tian, Y., and Vaithyanathan, S. (2011). SystemML: Declarative machine learning on MapReduce. In *Proc. 27th Int. Conf. on Data Engineering*, pages 231–242.

Gifford, D. K. (1979). Weighted voting for replicated data. In *Proc. 7th ACM Symp. on Operating System Principles*, pages 50–159.

Gilbert, S. and Lynch, N. A. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59.

Glasbergen, B., Abebe, M., Daudjee, K., Foggo, S., and Pacaci. (2018). Apollo: Learning query correlations for predictive caching in geo-distributed systems. In *Proc. 21st Int. Conf. on Extending Database Technology*, pages 253–264.

Golab, L. and Özsu, M. T. (2003). Processing sliding window multi-joins in continuous queries over data streams. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 500–511.

Golab, L. and Özsu, M. T. (2010). *Data Stream Systems*. Synthesis Lectures on Data Management. Morgan & Claypool.

Goldman, K. J. (1987). Data replication in nested transaction systems. Technical Report MIT/LCS/TR-390, Massachusetts Institute of Technology, Cambridge, Mass.

Goldman, R. and Widom, J. (1997). Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proc. 23th Int. Conf. on Very Large Data Bases*, pages 436–445.

Gonzalez, J. E., Low, Y., Gu, H., Bickson, D., and Guestrin, C. (2012). PowerGraph: Distributed graph-parallel computation on natural graphs. In *Proc. 10th USENIX Symp. on Operating System Design and Implementation*, pages 17–30.

Gonzalez, J. E., Xin, R. S., Dave, A., Crankshaw, D., Franklin, M. J., and Stoica, I. (2014). GraphX: graph processing in a distributed dataflow framework graph processing in a distributed dataflow framework. In *Proc. 11th USENIX Symp. on Operating System Design and Implementation*, pages 599–613.

Goodman, J. R. and Woest, P. J. (1988). The Wisconsin multicube: A new large-scale cache-coherent multiprocessor. Technical Report TR766, University of Wisconsin-Madison.

Gounaris, A., Paton, N. W., Fernandes, A. A. A., and Sakellariou, R. (2002). Adaptive query processing: A survey. In *Proc. British National Conf. on Databases*, pages 11–25.

Graefe, G. (1990). Encapsulation of parallelism in the Volcano query processing systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 102–111.

Graefe, G. (1993). Query evaluation techniques for large databases. *ACM Comput. Surv.*, 25(2): 73–170.

Graefe, G. (1994). Volcano - an extensible and parallel query evaluation system. *IEEE Trans. Knowl. and Data Eng.*, 6(1):120–135.

Graefe, G. and Kuno, H. (2010a). Self-selecting, self-tuning, incrementally optimized indexes. In *Proc. 13th Int. Conf. on Extending Database Technology*, pages 371–381.

Graefe, G. and Kuno, H. (2010b). Adaptive indexing for relational keys. In *Proc. Workshops of 26th Int. Conf. on Data Engineering*, pages 69–74.

Graefe, G., Idreos, S., Kuno, H., and Manegold, S. (2010). Benchmarking adaptive indexing. In *Proc. TPC Technology Conference on Performance Evaluation, Measurement and Characterization of Complex Systems*, pages 169–184.

Graefe, G., Halim, F., Idreos, S., Kuno, H., and Manegold, S. (2012). Concurrency control for adaptive indexing. *Proc. VLDB Endowment*, 5(7):656–667.

Graefe, G., Halim, F., Idreos, S., Kuno, H. A., Manegold, S., and Seeger, B. (2014). Transactional support for adaptive indexing. *VLDB J.*, 23(2):303–328.

Grapa, E. and Belford, G. G. (1977). Some theorems to aid in solving the file allocation problem. *Commun. ACM*, 20(11):878–882.

Gravano, L., Garcia-Molina, H., and Tomasic, A. (1999). Gloss: Text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264.

Gray, J. (1979). Notes on database operating systems. In Bayer, R., Graham, R., and Seegmüller, G., editors, *Operating Systems – An Advanced Course*, pages 393–481. Springer, New York.

Gray, J. and Lamport, L. (2006). Consensus on transaction commit. *ACM Trans. Database Syst.*, 31(1):133–160.

Gray, J. and Reuter, A. (1993). *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann.

Gray, J., Helland, P., O'Neil, P. E., and Shasha, D. (1996). The dangers of replication and a solution. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 173–182.

Gray, J. N., McJones, P., Blasgen, M., Lindsay, B., Lorie, R., Price, T., Putzolu, F., and Traiger, I. (1981). The recovery manager of the System R database manager. *ACM Comput. Surv.*, 13(2): 223–242.

Grefen, P. and Widom, J. (1997). Protocols for integrity constraint checking in federated databases. *Distrib. Parall. Databases*, 5(4):327–355.

Griffiths, P. P. and Wade, B. W. (1976). An authorization mechanism for a relational database system. *ACM Trans. Database Syst.*, 1(3):242–255.

Grossman, R. L. and Gu, Y. (2009). On the varieties of clouds for data intensive computing. *Q. Bull. IEEE TC on Data Eng.*, 32(1):44–50.

Guha, S. and McGregor, A. (2006). Approximate quantiles and the order of the stream. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 273–279.

Gulisano, V., Jiménez-Peris, R., Patino-Martinez, M., and Valduriez, P. (2010). StreamCloud: A large scale data streaming system. In *Proc. 30th IEEE Int. Conf. on Distributed Computing Systems*.

Gulisano, V., Jiménez-Peris, R., Patino-Martinez, M., and Valduriez, P. (2012). StreamCloud: An elastic and scalable data streaming system. *IEEE Trans. Parall. Dist. Sys.*, 23(12):2351–2365.

Gulli, A. and Signorini, A. (2005). The indexable web is more than 11.5 billion pages. In *Proc. 14th Int. World Wide Web Conf.*, pages 902–903.

Gummadi, P. K., Gummadi, R., Gribble, S. D., Ratnasamy, S., Shenker, S., and Stoica, I. (2003). The impact of DHT routing geometry on resilience and proximity. In *Proc. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 381–394.

Güntzer, U., Kießling, W., and Balke, W.-T. (2000). Optimizing multi-feature queries for image databases. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 419–428.

Gupta, A. and Mumick, I. S., editors. (1999). *Materialized Views: Techniques, Implementations, and Applications*. M.I.T. Press.

Gupta, A., Mumick, I. S., and Subrahmanian, V. S. (1993). Maintaining views incrementally. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 157–166.

Gupta, A., Jagadish, H., and Mumick, I. S. (1996). Data integration using self-maintainable views. In *Advances in Database Technology, Proc. 5th Int. Conf. on Extending Database Technology*, pages 140–144.

Gupta, A., Agrawal, D., and El Abbadi, A. (2003). Approximate range selection queries in peer-to-peer systems. In *Proc. 1st Biennial Conf. on Innovative Data Systems Research*, pages 141–151.

Haas, L. (2007). Beauty and the beast: The theory and practice of information integration. In *Proc. 11th Int. Conf. on Database Theory*, pages 28–43.

Haas, L., Kossmann, D., Wimmers, E., and Yang, J. (1997a). Optimizing queries across diverse data sources. In *Proc. 23th Int. Conf. on Very Large Data Bases*, pages 276–285.

Haas, L. M., Kossmann, D., Wimmers, E. L., and Yang, J. (1997b). Optimizing queries across diverse data sources. In *Proc. 23th Int. Conf. on Very Large Data Bases*, pages 276–285.

Haas, P. and Hellerstein, J. (1999a). Ripple joins for online aggregation. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 287–298.

Haas, P. J. and Hellerstein, J. M. (1999b). Ripple joins for online aggregation. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 287–298.

Hacigümüs, H., Sankaranarayanan, J., Tatemura, J., LeFevre, J., and Polyzotis, N. (2013). Odyssey: A multi-store system for evolutionary analytics. *Proc. VLDB Endowment*, 6(11): 1180–1181.

Haderle, C. M. D., Lindsay, B., Pirahesh, H., and Schwarz, P. (1992). Aries: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Trans. Database Syst.*, 17(1):94–162.

Hadzilacos, V. (1988). A theory of reliability in database systems. *J. ACM*, 35(1):121–145.

Halevy, A., Rajaraman, A., and Ordille, J. (2006). Data integration: the teenage years. In *Proc. 32nd Int. Conf. on Very Large Data Bases*, pages 9–16.

Halevy, A. Y. (2001). Answering queries using views: A survey. *VLDB J.*, 10(4):270–294.

Halevy, A. Y., Etzioni, O., Doan, A., Ives, Z. G., Madhavan, J., McDowell, L., and Tatarinov, I. (2003). Crossing the structure chasm. In *Proc. 1st Biennial Conf. on Innovative Data Systems Research*.

Halici, U. and Dogac, A. (1989). Concurrency control in distributed databases through time intervals and short-term locks. *IEEE Trans. Softw. Eng.*, 15(8):994–995.

Halim, F., Idreos, S., Karras, P., and Yap, R. H. C. (2012). Stochastic database cracking: Towards robust adaptive indexing in main-memory column-stores. *Proc. VLDB Endowment*, 5(6):502–513.

Hammad, M., Aref, W., and Elmagarmid, A. (2003a). Stream window join: Tracking moving objects in sensor-network databases. In *Proc. 15th Int. Conf. on Scientific and Statistical Database Management*, pages 75–84.

Hammad, M., Aref, W., Franklin, M., Mokbel, M., and Elmagarmid, A. (2003b). Efficient execution of sliding window queries over data streams. Technical Report CSD TR 03-035, Purdue University.

Hammad, M., Mokbel, M., Ali, M., Aref, W., Catlin, A., Elmagarmid, A., Eltabakh, M., Elfeky, M., Ghanem, T., Gwadera, R., Ilyas, I., Marzouk, M., and Xiong, X. (2004). Nile: a query processing engine for data streams. In *Proc. 20th Int. Conf. on Data Engineering*, page 851.

Hammad, M., Aref, W., and Elmagarmid, A. (2005). Optimizing in-order execution of continuous queries over streamed sensor data. In *Proc. 17th Int. Conf. on Scientific and Statistical Database Management*, pages 143–146.

Hammer, M. and Niamir, B. (1979). A heuristic approach to attribute partitioning. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 93–101.

Hammer, M. and Shipman, D. W. (1980). Reliability mechanisms for SDD-1: A system for distributed databases. *ACM Trans. Database Syst.*, 5(4):431–466.

Han, M. (2015). On improving distributed Pregel-like graph processing systems. Master's thesis, University of Waterloo, David R. Cheriton School of Computer Science.

Han, M. and Daudjee, K. (2015). Giraph unchained: Barrierless asynchronous parallel execution in Pregel-like graph processing systems. *Proc. VLDB Endowment*, 8(9):950–961.

Härder, T. and Reuter, A. (1983). Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317.

Hartig, O. (2012). SPARQL for a web of linked data: Semantics and computability. In *Proc. 9th Extended Semantic Web Conf.*, pages 8–23.

Hartig, O. (2013a). An overview on execution strategies for linked data queries. *Datenbank-Spektrum*, 13(2):89–99.

Hartig, O. (2013b). SQUIN: a traversal based query execution system for the web of linked data. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1081–1084.

Harvey, N. J. A., Jones, M. B., Saroiu, S., Theimer, M., and Wolman, A. (2003). SkipNet: A scalable overlay network with practical locality properties. In *Proc. 4th USENIX Symp. on Internet Tech. and Systems*.

He, B., Chang, K. C.-C., and Han, J. (2004). Mining complex matchings across web query interfaces. In *Proc. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 3–10.

He, Q. and Ling, T. W. (2006). An ontology-based approach to the integration of entity-relationship schemas. *Data & Knowl. Eng.*, 58(3):299–326.

Hedley, Y. L., Younas, M., James, A., and Sanderson, M. (2004a). A two-phase sampling technique for information extraction from hidden web databases. In *WIDM04*, pages 1–8.

Hedley, Y.-L., Younas, M., James, A. E., and Sanderson, M. (2004b). Query-related data extraction of hidden web documents. In *Proc. 27th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 558–559.

Heinze, T., Pappalardo, V., Jerzak, Z., and Fetzer, C. (2014). Auto-scaling techniques for elastic data stream processing. In *Proc. 8th Int. Conf. Distributed Event-Based Systems*, pages 318–321.

Heinze, T., Roediger, L., Meister, A., Ji, Y., Jerzak, Z., and Fetzer, C. (2015). Online parameter optimization for elastic data stream processing. In *Proc. 6th ACM Symp. on Cloud Computing*, pages 276–287.

Helal, A. A., Heddaya, A. A., and Bhargava, B. B. (1997). *Replication Techniques in Distributed Systems*. Kluwer Academic Publishers.

Hellerstein, J. M., Haas, P., and Wang, H. (1997). Online aggregation. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 171–182.

Hellerstein, J. M., Franklin, M. J., Chandrasekaran, S., Deshpande, A., Hildrum, K., Madden, S., Raman, V., and Shah, M. A. (2000). Adaptive query processing: Technology in evolution. *Q. Bull. IEEE TC on Data Eng.*, 23(2):7–18.

Herlihy, M. (1987). Concurrency versus availability: Atomicity mechanisms for replicated data. *ACM Trans. Comp. Syst.*, 5(3):249–274.

Herman, D. and Verjus, J. P. (1979). An algorithm for maintaining the consistency of multiple copies. In *Proc. 1st IEEE Int. Conf. on Distributed Computing Systems*, pages 625–631.

Hersh, W. (2001). Managing gigabytes - compressing and indexing documents and images (second edition). *Inf. Retr.*, 4(1):79–80.

Hevner, A. R. and Schneider, G. M. (1980). An integrated design system for distributed database networks. In *Digest of Papers - COMPCON*, pages 459–465.

Hirate, Y., Kato, S., and Yamana, H. (2006). Web structure in 2005. In *Proc. 4th Int. Workshop on Algorithms and Models for the Web-Graph*, pages 36 – 46.

Hoffer, H. A. and Severance, D. G. (1975). The use of cluster analysis in physical data base design. In *Proc. 1st Int. Conf. on Very Data Bases*, pages 69–86.

Hoffer, J. A. (1975). *A Clustering Approach to the Generation of Subfiles for the Design of a Computer Data Base*. PhD thesis, Department of Operations Research, Cornell University, Ithaca, N.Y.

Hoffman, J. L. (1977). *Model Methods for Computer Security and Privacy*. Prentice-Hall.

Holze, M. and Ritter, N. (2008). Autonomic databases: Detection of workload shifts with n-gram-models. In *Proc. 12th East European Conf. Advances in Databases and Information Systems*, pages 127–142.

Hong, W. (1992). Exploiting inter-operation parallelism in XPRS. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 19–28.

Hong, W. and Stonebraker, M. (1993). Optimization of parallel query execution plans in XPRS. *Distrib. Parall. Databases*, 1(1):9–32.

Hoque, I. and Gupta, I. (2013). LFGraph: simple and fast distributed graph analytics. In *Proc. 1st ACM SIGOPS Conf. on Timely Results in Operating Syst.*, pages 9:1–9:17.

Hortonworks. (2014). White paper: A modern data architecture with Apache Hadoop: the journey to the data lake. Technical report, Hortonworks. Last accessed August 2018.

Hsiao, H. I. and DeWitt, D. (1991). A performance study of three high-availability data replication strategies. In *Proc. Int. Conf. on Parallel and Distributed Information Systems*, pages 18–28.

Huang, Z. and He, Y. (2018). Auto-detect: Data-driven error detection in tables. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1377–1392.

Huebsch, R., Hellerstein, J., Lanham, N., Loo, B. T., Shenker, S., and Stoica, I. (2003). Querying the internet with pier. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 321–332.

Hull, R. (1997). Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 51–61.

Hwang, J., Balazinska, M., Rasin, A., Cetintemel, U., Stonebraker, M., and Zdonik, S. (2005). High-availability algorithms for distributed stream processing. In *Proc. 21st Int. Conf. on Data Engineering*, pages 779–790.

Idreos, S. (2010). *Database Cracking: Towards Auto-tuning Database Kernels*. PhD thesis, University of Amsterdam.

Idreos, S., Kersten, M. L., and Manegold, S. (2007a). Updating a cracked database. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 413–424.

Idreos, S., Kersten, M. L., and Manegold, S. (2007b). Database cracking. In *Proc. 3rd Biennial Conf. on Innovative Data Systems Research*, pages 68–78.

Idreos, S., Kersten, M. L., and Manegold, S. (2009). Self-organizing tuple reconstruction in column-stores. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 297–308.

Idreos, S., Alagiannis, I., Johnson, R., and Ailamaki, A. (2011). Here are my data files. here are my queries. where are my results? In *Proc. 5th Biennial Conf. on Innovative Data Systems Research*, pages 57–68.

Ilyas, I. and Chu, X. (2019). *Principles of Data Cleaning*. ACM Books.

Ilyas, I. F. and Chu, X. (2015). Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393.

Ilyas, I. F., Beskales, G., and Soliman, M. A. (2008). A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4):1–58.

Ioannidis, Y. and Wong, E. (1987). Query optimization by simulated annealing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 9–22.

Ipeirotis, P. G. and Gravano, L. (2002). Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proc. 28th Int. Conf. on Very Large Data Bases*, pages 394–405.

Irani, K. B. and Khabbaz, N. G. (1982). A methodology for the design of communication networks and the distribution of data in distributed computer systems. *IEEE Trans. Comput.*, C-31(5): 419–434.

Isloor, S. and Marsland, T. (1980). The deadlock problem : An overview. *Computer*, 13(9):58–78.

Ito, J., Narula, N., and Ali, R. (2017). The blockchain will do to the financial system what the internet did to media. Accessible at https://hbr.org/2017/03/the-blockchain-will-do-to-banks-and-law-firms-what-the-internet-did-to-media/. Last accessed February 2019.

Jagadish, H. V., Ooi, B. C., and Vu, Q. H. (2005). BATON: A balanced tree structure for peer-to-peer networks. In *Proc. 31st Int. Conf. on Very Large Data Bases*, pages 661–672.

Jagadish, H. V., Ooi, B. C., Tan, K.-L., Vu, Q. H., and Zhang, R. (2006). Speeding up search in peer-to-peer networks with a multi-way tree structure. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1–12.

Jajodia, S. and Mutchler, D. (1987). Dynamic voting. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 227–238.

Jajodia, S. and Sandhu, R. S. (1991). Towards a multilevel secure relational data model. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 50–59.

Jajodia, S., Atluri, V., Keefe, T. F., McCollum, C. D., and Mukkamala, R. (2001). Multilevel security transaction processing. *J. Computer Security*, 9(3):165–195.

Jhingran, A. D., Mattos, N., and Pirahesh, H. (2002). Information integration: A research agenda. *IBM Systems J.*, 41(4):555–562.

Jimenez-Peris, R. and Patiño Martinez, M. (2011). System and method for highly scalable decentralized and low contention transactional processing. US Patent 9,760,597 B2, EU Patent 2780832.

Jiménez-Peris, R., Patiño-Martínez, M., and Alonso, G. (2002). Non-intrusive, parallel recovery of replicated data. In *Proc. 21st Symp. on Reliable Distributed Systems*, pages 150–159.

Jiménez-Peris, R., Patiño-Martínez, M., Kemme, B., and Alonso, G. (2002). Improving the scalability of fault-tolerant database clusters. In *Proc. 22nd IEEE Int. Conf. on Distributed Computing Systems*, pages 477–484.

Jiménez-Peris, R., Patiño-Martínez, M., Alonso, G., and Kemme, B. (2003). Are quorums an alternative for data replication? *ACM Trans. Database Syst.*, 28(3):257–294.

Johnson, T., Muthukrishnan, S., Shkapenyuk, V., and Spatscheck, O. (2005). A heartbeat mechanism and its application in Gigascope. In *Proc. 31st Int. Conf. on Very Large Data Bases*, pages 1079–1088.

Johnson, T., Muthukrishnan, S. M., Shkapenyuk, V., and Spatscheck, O. (2008). Query-aware partitioning for monitoring massive network data streams. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1135–1146.

Kaelbling, L. P., Littman, M. L., and Moore, A. P. (1996). Reinforcement learning: A survey. *J. Autom. Reasoning*, 4:237–285.

Kalogeraki, V., Gunopulos, D., and Zeinalipour-Yazti, D. (2002). A local search mechanism for peer-to-peer networks. In *Proc. 11th Int. Conf. on Information and Knowledge Management*, pages 300–307.

Kambayashi, Y., Yoshikawa, M., and Yajima, S. (1982). Query processing for distributed databases using generalized semi–joins. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 151–160.

Kang, J., Naughton, J., and Viglas, S. (2003). Evaluating window joins over unbounded streams. In *Proc. 19th Int. Conf. on Data Engineering*, pages 341–352.

Kaoudi, Z. and Manolescu, I. (2015). RDF in the clouds: A survey. *VLDB J.*, 24:67–91.

Kara, A., Ngo, H. Q., Nikolic, M., Olteanu, D., and Zhang, H. (2019). Counting triangles under updates in worst-case optimal time. In *Proc. 22nd Int. Conf. on Database Theory*, pages 1:1–1:18.

Karlapalem, K. and Navathe, S. B. (1994). Materialization of redesigned distributed relational databases. Technical Report HKUST-CS94-14, Hong Kong University of Science and Technology, Department of Computer Science.

Karlapalem, K., Navathe, S. B., and Ammar, M. (1996). Optimal redesign policies to support dynamic processing of applications on a distributed relational database system. *Inf. Syst.*, 21 (4):353–367.

Karypis, G. and Kumar, V. (1995). Multilevel graph partitioning schemes. In *Proc. 1995 Int. Conf. on Parallel Processing*, pages 113–122.

Kashyap, V. and Sheth, A. P. (1996). Semantic and schematic similarities between database objects: A context-based approach. *VLDB J.*, 5(4):276–304.

Katz, B. and Lin, J. (2002). Annotating the world wide web using natural language. In *Proc. 2nd Workshop on NLP and XML*, pages 1–8.

Kazerouni, L. and Karlapalem, K. (1997). Stepwise redesign of distributed relational databases. Technical Report HKUST-CS97-12, Hong Kong University of Science and Technology, Department of Computer Science.

Keeton, K., Patterson, D., and Hellerstein, J. M. (1998). A case for intelligent disks (idisks). *ACM SIGMOD Rec.*, 27(3):42–52.

Keller, A. M. (1982). Update to relational databases through views involving joins. In *Proc. 2nd Int. Conf. on Databases: Improving Usability and Responsiveness*, pages 363–384.

Kementsietsidis, A., Arenas, M., and Miller, R. J. (2003). Managing data mappings in the hyperion project. In *Proc. 19th Int. Conf. on Data Engineering*, pages 732–734.

Kemme, B. and Alonso, G. (2000a). A new approach to developing and implementing eager database replication protocols. *ACM Trans. Database Syst.*, 25(3):333–379.

Kemme, B. and Alonso, G. (2000b). Don't be lazy, be consistent: Postgres-R, a new way to implement database replication. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 134–143.

Kemme, B., Bartoli, A., and Babaoglu, O. (2001). Online reconfiguration in replicated databases based on group communication. In *Proc. Int. Conf. on Dependable Systems and Networks*, pages 117–130.

Kemme, B., Peris, R. J., and Patino-Martinez, M. (2010). *Database Replication*. Morgan & Claypool.

Kemper, A. and Neumann, T. (2011). HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *Proc. 27th Int. Conf. on Data Engineering*, pages 195–206.

Kermarrec, A.-M. and van Steen, M. (2007). Gossiping in distributed systems. *Operating Systems Rev.*, 41(5):2–7.

Kermarrec, A.-M., Rowstron, A., Shapiro, M., and Druschel, P. (2001). The icecube approach to the reconciliation of diverging replicas. In *Proc. ACM SIGACT-SIGOPS 20th Symp. on the Principles of Distributed Computing*, pages 210–218.

Khayyat, Z., Awara, K., Alonazi, A., Jamjoom, H., Williams, D., and Kalnis, P. (2013). Mizan: A system for dynamic load balancing in large-scale graph processing. In *Proc. 8th ACM SIGOPS/EuroSys European Conf. on Comp. Syst.*, pages 169–182.

Khoshafian, S. and Valduriez, P. (1987). Sharing persistence and object-orientation: A database perspective. In *Int. Workshop on Database Programming Languages*, pages 181–205.

Kim, W. and Seo, J. (1991). Classifying schematic and data heterogeneity in multidatabase systems. *Computer*, 24(12):12–18.

Kirsch, J. and Amir, Y. (2008). Paxos for system builders: An overview. In *Proc. 2nd Workshop on Large-Scale Distributed Systems and Middleware*, pages 3:1–3:6.

Kitsuregawa, M. and Ogawa, Y. (1990). Bucket spreading parallel hash: A new, robust, parallel hash join method for data skew in the super database computer. In *Proc. 16th Int. Conf. on Very Large Data Bases*, pages 210–221.

Kitsuregawa, M., Tanaka, H., and Moto-Oka, T. (1983). Application of hash to data base machine and its architecture. *New Generation Computing*, 1(1):63–74.

Kiveris, R., Lattanzi, S., Mirrokni, V., Rastogi, V., and Vassilvitskii, S. (2014). Connected components in MapReduce and beyond. In *Proc. 5th ACM Symp. on Cloud Computing*, pages 18:1–18:13.

Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999). The web as a graph: Measurements, models, and methods. In *Proc. 5th Annual Int. Conf. Computing and Combinatorics*, pages 1–17.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5): 604–632.

Knapp, E. (1987). Deadlock detection in distributed databases. *ACM Comput. Surv.*, 19(4):303–328.

Knuth, D. E. (1973). *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley.

Koch, C. (2001). *Data Integration against Multiple Evolving Autonomous Schemata*. Ph.D. thesis, Technical University of Vienna.

Koch, C. (2010). Incremental query evaluation in a ring of databases. In *Proc. 29th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 87–98.

Koch, C., Ahmad, Y., Kennedy, O., Nikolic, M., Nötzli, A., Lupei, D., and Shaikhha, A. (2014). DBToaster: higher-order delta processing for dynamic, frequently fresh views. *VLDB J.*, 23(2): 253–278.

Kohler, W. H. (1981). A survey of techniques for synchronization and recovery in decentralized computer systems. *ACM Comput. Surv.*, 13(2):149–183.

Kolev, B., Bondiombouy, C., Valduriez, P., Jiménez-Peris, R., Pau, R., and Pereira, J. (2016a). The cloudmdsql multistore system. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 2113–2116.

Kolev, B., Valduriez, P., Bondiombouy, C., Jiménez-Peris, R., Pau, R., and Pereira, J. (2016b). CloudMdsQL: querying heterogeneous cloud data stores with a common language. *Distrib. Parall. Databases*, 34(4):463–503.

Kolev, B., Levchenko, O., Pacitti, E., Valduriez, P., Vilaça, R., Gonçalves, R. C., Jiménez-Peris, R., and Kranas, P. (2018). Parallel polyglot query processing on heterogeneous cloud data stores with LeanXcale. In *Proc. 2018 IEEE Int. Conf. on Big Data*, pages 1757–1766.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Kollias, J. G. and Hatzopoulos, M. (1981). Criteria to aid in solving the problem of allocating copies of a file in a computer network. *Comp. J.*, 24(1):29–30.

Konopnicki, D. and Shmueli, O. (1995). W3QS: A query system for the World Wide Web. In *Proc. 21th Int. Conf. on Very Large Data Bases*, pages 54–65.

Kossmann, D. (2000). The state of the art in distributed query processing. *ACM Comput. Surv.*, 32 (4):422–469.

Krishnamurthy, R., Litwin, W., and Kent, W. (1991). Language features for interoperability of databases with schematic discrepancies. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 40–49.

Kshemkalyani, A. and Singhal, M. (1994). On characterization and correctness of distributed deadlocks. *J. Parall. and Distrib. Comput.*, 22(1):44–59.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B. (2000). Oceanstore: an architecture for global-scale persistent storage. In *ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 190–201.

Kulkarni, S., Bhagat, N., Fu, M., Kedigehalli, V., Kellogg, C., Mittal, S., Patel, J. M., Ramasamy, K., and Taneja, S. (2015). Twitter heron: Stream processing at scale. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 239–250.

Kumar, A. and Segev, A. (1993). Cost and availability tradeoffs in replicated data concurrency control. *ACM Trans. Database Syst.*, 18(1):102–131.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., and Upfal, E. (2000). The Web as a graph. In *Proc. 19th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 1–10.

Kumar, V., editor. (1996). *Performance of Concurrency Control Mechanisms in Centralized Database Systems*. Prentice-Hall.

Kung, H. and Robinson, J. (1981). On optimistic methods for concurrency control. *ACM Trans. Database Syst.*, 6(2):213–226.

Kwok, C. C. T., Etzioni, O., and Weld, D. S. (2001). Scaling question answering to the web. In *Proc. 10th Int. World Wide Web Conf.*, pages 150–161.

Ladwig, G. and Tran, T. (2010). Linked data query processing strategies. In *Proc. 9th Int. Semantic Web Conf.*, pages 453–469.

Ladwig, G. and Tran, T. (2011). SIHJoin: Querying remote and local linked data. In *Proc. 8th Extended Semantic Web Conf.*, pages 139–153.

Lage, J. P., da Silva, A. S., Golgher, P. B., and Laender, A. H. F. (2002). Collecting hidden web pages for data extraction. In *Proc. 4th Int. Workshop on Web Information and Data Management*, pages 69–75.

Lakshmanan, L. V. S., Sadri, F., and Subramanian, I. N. (1996). A declarative language for querying and restructuring the web. In *Proc. 6th Int. Workshop on Research Issues on Data Eng.*, pages 12–21.

Lam, S. S. and Özsu, M. T. (2002). Querying web data – the WebQA approach. In *Proc. 3rd Int. Conf. on Web Information Systems Eng.*, pages 139–148.

Lamport, L. (1998). The part-time parliament. *ACM Trans. Comp. Syst.*, 16(2):133–169.

Lamport, L. (2001). Paxos made simple. *ACM SIGACT News*, 32(4):51–58.

Lampson, B. and Sturgis, H. (1976). Crash recovery in distributed data storage system. Technical report, Xerox Palo Alto Research Center, Palo Alto, Calif.

Landers, T. and Rosenberg, R. L. (1982). An overview of multibase. In Schneider, H.-J., editor, *Distributed Data Bases*, pages 153–184. North-Holland, Amsterdam.

Langville, A. N. and Meyer, C. D. (2006). *Google's PageRank and Beyond*. Princeton University Press.

Lanzelotte, R., Valduriez, P., Zaït, M., and Ziane, M. (1994). Industrial-strength parallel query optimization: issues and lessons. *Inf. Syst.*, 19(4):311–330.

Larriba-Pey, J. L., Martínez-Bazán, N., and Domínguez-Sal, D. (2014). Introduction to graph databases. In Koubarakis, M., Stamou, G., Stoilos, G., Horrocks, I., Kolaitis, P., Lausen, G., and Weikum, G., editors, *Reasoning Web: Reasoning on the Web in the Big Data Era*, pages 171–194. Springer.

Larson, P.-Å., Blanas, S., Diaconu, C., Freedman, C., Patel, J. M., and Zwilling, M. (2011). High-performance concurrency control mechanisms for main-memory databases. *Proc. VLDB Endowment*, 5(4):298–309.

Law, Y.-N., Wang, H., and Zaniolo, C. (2004). Query languages and data models for database sequences and data streams. In *Proc. 30th Int. Conf. on Very Large Data Bases*, pages 492–503.

Lawrence, S. and Giles, C. L. (1998). Searching the world wide web. *Science*, 280(5360):98–100.

Lawrence, S. and Giles, C. L. (1999). Accessibility of information on the web. *Nature*, 400(6740): 107–9.

Lee, K.-H., Lee, Y.-J., Choi, H., Chung, Y. D., and Moon, B. (2012). Parallel data processing with mapreduce: A survey. *ACM SIGMOD Rec.*, 40(4):11–20.

LeFevre, J., Sankaranarayanan, J., Hacigumus, H., Tatemura, J., Polyzotis, N., and Carey, M. J. (2014). MISO: Souping up big data query processing with a multistore system. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1591–1602.

Leis, V., Boncz, P. A., Kemper, A., and Neumann, T. (2014). Morsel-driven parallelism: a NUMA-aware query evaluation framework for the many-core age. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 743–754.

Lenoski, D., Laudon, J., Gharachorloo, K., Weber, W. D., Gupta, A., Henessy, J., Horowitz, M., and Lam, M. S. (1992). The Stanford Dash multiprocessor. *Computer*, 25(3):63–79.

Lenzerini, M. (2002). Data integration: a theoretical perspective. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 233–246.

Levandoski, J. J., Larson, P. Å., and Stoica, R. (2013). Identifying hot and cold data in main-memory databases. In *Proc. 29th Int. Conf. on Data Engineering*, pages 26–37.

Levin, K. D. and Morgan, H. L. (1975). Optimizing distributed data bases: A framework for research. In *Proc. National Computer Conf*, pages 473–478.

Levy, A. Y., Mendelzon, A. O., Sagiv, Y., and Srivastava, D. (1995). Answering queries using views. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 95–104.

Levy, A. Y., Rajaraman, A., and Ordille, J. J. (1996a). The world wide web as a collection of views: Query processing in the information manifold. In *Proc. Workshop on Materialized Views: Techniques and Applications*, pages 43–55.

Levy, A. Y., Rajaraman, A., and Ordille, J. J. (1996b). Querying heterogeneous information sources using source descriptions. In *Proc. 22th Int. Conf. on Very Large Data Bases*, pages 251–262.

Li, F., Ooi, B. C., Özsu, M. T., and Wu, S. (2014). Distributed data management using MapReduce. *ACM Comput. Surv.*, 46(3):Article No. 31.

Li, H.-G., Chen, S., Tatemura, J., Agrawal, D., Candan, K. S., and Hsiung, W.-P. (2006). Safety guarantee of continuous join queries over punctuated data streams. In *Proc. 32nd Int. Conf. on Very Large Data Bases*, pages 19–30.

Li, J., Maier, D., Tufte, K., Papadimos, V., and Tucker, P. a. (2005). Semantics and evaluation techniques for window aggregates in data streams. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 311–322.

Li, W.-S. and Clifton, C. (2000). Semint: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowl. Eng.*, 33(1):49–84.

Li, W.-S., Clifton, C., and Liu, S.-Y. (2000). Database integration using neural networks: Implementation and experiences. *Knowl. and Information Syst.*, 2(1):73–96.

Lim, L., Wang, M., Padmanabhan, S., Vitter, J. S., and Agarwal, R. (2003). Dynamic maintenance of web indexes using landmarks. In *Proc. 12th Int. World Wide Web Conf.*, pages 102–111.

Lima, A., Mattoso, M., and Valduriez, P. (2004). OLAP query processing in a database cluster. In *Proc. 10th Int. Euro-Par Conf.*, pages 355–362.

Lin, Q., Chang, P., Chen, G., Ooi, B. C., Tan, K., and Wang, Z. (2016). Towards a Non-2PC transaction management in distributed database systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1659–1674.

Lin, Y., Kemme, B., Patiño Martínez, M., and Jiménez-Peris, R. (2005). Middleware based data replication providing snapshot isolation. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 419–430.

Litwin, W., Neimat, M.-A., and Schneider, D. A. (1993). LH* – linear hashing for distributed files. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 327–336.

Liu, B., Zhu, Y., and Rundensteiner, E. (2006). Run-time operator state spilling for memory intensive long running queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 347–358.

Livny, M., Khoshafian, S., and Boral, H. (1987). Multi-disk management. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 69–77.

Lohman, G., Mohan, C., Haas, L., Daniels, D., Lindsay, B., Selinger, P., and Wilms, P. (). Query processing in R*. pages 31–47.

Lomet, D., Feket, A., Wang, R., and Ward, P. (2012). Multi-version concurrency via timestamp range conflict management. In *Proc. 28th Int. Conf. on Data Engineering*, pages 714–725.

Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., and Guestrin, C. (2010). GraphLab: new framework for parallel machine learning. In *Proc. 26th Conf. on Uncertainty in Artificial Intelligence*, pages 340–349.

Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., and Hellerstein, J. M. (2012). Distributed graphlab: A framework for machine learning in the cloud. *Proc. VLDB Endowment*, 5(8):716–727.

Lu, H., Shan, M.-C., and Tan, K.-L. (1991). Optimization of multi-way join queries for parallel execution. In *Proc. 17th Int. Conf. on Very Large Data Bases*, pages 549–560.

Lu, H., Ooi, B., and Goh, C. (1992). On global multidatabase query optimization. *ACM SIGMOD Rec.*, 21(4):6–11.

Lu, H., Ooi, B., and Goh, C. (1993). Multidatabase query optimization: Issues and solutions. In *Proc. 3rd Int. Workshop on Res. Issues in Data Eng*, pages 137–143.

Lugowski, A., Alber, D., Buluç, A., Gilbert, J. R., Reinhardt, S., Teng, Y., and Waranis, A. (2012). A flexible open-source toolbox for scalable complex graph analysis. In *Proc. 2012 SIAM Int. Conf. on Data Mining*, pages 930–941.

Lumsdaine, A., Gregor, D., Hendrickson, B., and Berry, J. (2007). Challenges in parallel graph processing. *Parallel Processing Letters*, 17(01):5–20.

Lunt, T. F. and Fernández, E. B. (1990). Database security. *ACM SIGMOD Rec.*, 19(4):90–97.

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. In *Proc. 16th Annual Int. Conf. on Supercomputing*, pages 84–95.

Lynch, N. (1983a). Multilevel atomicity: A new correctness criterion for database concurrency control. *ACM Trans. Database Syst.*, 8(4):484–502.

Lynch, N. (1983b). Concurrency control for resilient nested transactions. In *Proc. 2nd ACM SIGACT–SIGMOD Symp. on Principles of Database Systems*, pages 166–181.

Lynch, N. and Merritt, M. (1986). Introduction to the theory of nested transactions. Technical Report MIT/LCS/TR-367, Massachusetts Institute of Technology, Cambridge, Mass.

Lynch, N., Merritt, M., Weihl, W. E., and Fekete, A. (1993). *Atomic Transactions in Concurrent Distributed Systems*. Morgan Kaufmann.

Mackert, L. and Lohman, G. (1986a). R* optimizer validation and perfromance evaluation for distributed queries. In *Proc. 12th Int. Conf. on Very Large Data Bases*, pages 149–159.

Mackert, L. F. and Lohman, G. (1986b). R* optimizer validation and performance evaluation for local queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 84–95.

Madden, S. and Franklin, M. J. (2002). Fjording the stream: An architecture for queries over streaming sensor data. In *Proc. 18th Int. Conf. on Data Engineering*, pages 555–566.

Madden, S., Shah, M., Hellerstein, J., and Raman, V. (2002a). Continuously adaptive continuous queries over streams. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 49–60.

Madden, S., Shah, M. A., Hellerstein, J. M., and Raman, V. (2002b). Continuously adaptive continuous queries over streams. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 49–60.

Madhavan, J., Bernstein, P., and Rahm, E. (2001). Generic schema matching with Cupid. In *Proc. 27th Int. Conf. on Very Large Data Bases*, pages 49–58.

Mahmoud, . A. and Riordon, J. S. (1976). Optimal allocation of resources in distributed information networks. *ACM Trans. Database Syst.*, 1(1):66–78.

Maiyya, S., Zakhary, V., Agrawal, D., and El Abbadi, A. (2018). Database and distributed computing fundamentals for scalable, fault-tolerant, and consistent maintenance of blockchains. *Proc. VLDB Endowment*, 11(12):2098–2101.

Malewicz, G., Austern, M. H., Bik, A. J. C., Dehnert, J. C., Horn, I., Leiser, N., and Czajkowski, G. (2010). Pregel: a system for large-scale graph processing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 135–146.

Manber, U. and Myers, G. (1990). Suffix arrays: a new method for on-line string searches. In *Proc. 1st Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 319–327.

Manegold, S., Boncz, P. A., and Kersten, M. L. (2002). Optimizing main-memory join on modern hardware. *IEEE Trans. Knowl. and Data Eng.*, 14(4):709–730.

Manolescu, I., Florescu, D., and Kossmann, D. (2001). Answering XML queries on heterogeneous data sources. In *Proc. 27th Int. Conf. on Very Large Data Bases*, pages 241–250.

Martins, V. and Pacitti, E. (2006). Dynamic and distributed reconciliation in p2p-dht networks. In *uropean Conf. on Parallel Computing (Euro-Par)*, pages 337–349.

Martins, V., Akbarinia, R., Pacitti, E., and Valduriez, P. (2006a). Reconciliation in the APPA P2P system. In *Proc. IEEE Int. Conf. on Parallel and Distributed Systems*, pages 401–410.

Martins, V., Pacitti, E., and Valduriez, P. (2006b). Survey of data replication in P2P systems. Technical Report 6083, INRIA, Rennes, France.

Martins, V., Pacitti, E., Dick, M. E., and Jimenez-Peris, R. (2008). Scalable and topology-aware reconciliation on P2P networks. *Distrib. Parall. Databases*, 24(1–3):1–43.

McBrien, P. and Poulovassilis, A. (2003). Defining peer-to-peer data integration using both as view rules. In *Proc. 1st Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing*, pages 91–107.

McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (1999). A machine learning approach to building domain-specific search engines. In *Proc. 16th Int. Joint Conf. on AI*.

McCann, R., AlShebli, B., Le, Q., Nguyen, H., Vu, L., and Doan, A. (2005). Mapping maintenance for data integration systems. In *Proc. 31st Int. Conf. on Very Large Data Bases*, pages 1018–1029.

McCormick, W. T., Schweitzer, P. J., and White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique. *Oper. Res.*, 20(5):993–1009.

McCune, R. R., Weninger, T., and Madey, G. (2015). Thinking like a vertex: A survey of vertex-centric frameworks for large-scale distributed graph processing. *ACM Comput. Surv.*, 48(2): 25:1–25:39.

Mehta, M. and DeWitt, D. (1995). Managing intra-operator parallelism in parallel database systems. In *Proc. 21th Int. Conf. on Very Large Data Bases*.

Melnik, S., Raghavan, S., Yang, B., and Garcia-Molina, H. (2001). Building a distributed full-text index for the web. In *Proc. 10th Int. World Wide Web Conf.*, pages 396–406.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proc. 18th Int. Conf. on Data Engineering*, pages 117–128.

Menasce, D. A. and Muntz, R. R. (1979). Locking and deadlock detection in distributed databases. *IEEE Trans. Softw. Eng.*, SE-5(3):195–202.

Mendelzon, A. O., Mihaila, G. A., and Milo, T. (1997). Querying the World Wide Web. *Int. J. Digit. Libr.*, 1(1):54–67.

Meng, W., Yu, C., Kim, W., Wang, G., Phan, T., and Dao, S. (1993). Construction of relational front-end for object-oriented database systems. In *Proc. 9th Int. Conf. on Data Engineering*, pages 476–483.

Milán-Franco, J. M., Jiménez-Peris, R., Patiño-Martínez, M., and Kemme, B. (2004). Adaptive middleware for data replication. In *Proc. ACM/IFIP/USENIX 5th Int. Middleware Conf.*, pages 175–194.

Miller, R. J., Haas, L. M., and Hernández, M. A. (2000). Schema mapping as query discovery. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 77–88.

Miller, R. J., Hernández, M. A., Haas, L. M., Yan, L., Ho, C. T. H., Fagin, R., and Popa, L. (2001). The Clio project: Managing heterogeneity. *ACM SIGMOD Rec.*, 31(1):78–83.

Milo, T. and Zohar, S. (1998). Using schema matching to simplify heterogeneous data translation. In *Proc. 24th Int. Conf. on Very Large Data Bases*, pages 122–133.

Minoura, T. and Wiederhold, G. (1982). Resilient extended true-copy token scheme for a distributed database system. *IEEE Trans. Softw. Eng.*, SE-8(3):173–189.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Mitzenmacher, M. (2001). The power of two choices in randomized load balancing. *IEEE Trans. Parall. Dist. Sys.*, 12(10):1094–1104.

Mohan, C. (1979). Data base design in the distributed environment. Working Paper WP-7902, Department of Computer Sciences, University of Texas at Austin.

Mohan, C. and Lindsay, B. (1983). Efficient commit protocols for the tree of processes model of distributed transactions. In *Proc. ACM SIGACT-SIGOPS 2nd Symp. on the Principles of Distributed Computing*, pages 76–88.

Mohan, C. and Yeh, R. T. (1978). *Distributed Data Base Systems: A Framework for Data Base Design. In Distributed Data Bases, Infotech State-of-the-Art Report*. Infotech.

Mohan, C., Lindsay, B., and Obermarck, R. (1986). Transaction management in the r* distributed database management system. *ACM Trans. Database Syst.*, 11(4):378–396.

Morgan, H. L. and Levin, K. D. (1977). Optimal program and data location in computer networks. *Commun. ACM*, 20(5):315–322.

Moss, E. (1985). *Nested Transactions*. M.I.T. Press.

Muthukrishnan, S. (2005). *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science. NOW Publishers.

Naacke, H., Tomasic, A., and Valduriez, P. (1999). Validating mediator cost models with DISCO. *Networking and Information Systems Journal*, 2(5):639–663.

Najork, M. and Wiener, J. L. (2001). Breadth-first crawling yields high-quality pages. In *Proc. 10th Int. World Wide Web Conf.*, pages 114–118.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Accessible at https://bitcoin.org/bitcoin.pdf/. Last accessed February 2019.

Nasir, M. A. U., Morales, G. D. F., García-Soriano, D., Kourtellis, N., and Serafini, M. (2015). The power of both choices: Practical load balancing for distributed stream processing engines. In *Proc. 31st Int. Conf. on Data Engineering*, pages 137–148.

Nasir, M. A. U., Morales, G. D. F., Kourtellis, N., and Serafini, M. (2016). When two choices are not enough: Balancing at scale in distributed stream processing. In *Proc. 32nd Int. Conf. on Data Engineering*, pages 589–600.

Naumann, F., Ho, C.-T., Tian, X., Haas, L. M., and Megiddo, N. (2002). Attribute classification using feature analysis. In *Proc. 18th Int. Conf. on Data Engineering*, page 271.

Navathe, S. B., Ceri, S., Wiederhold, G., and Dou, J. (1984). Vertical partitioning of algorithms for database design. *ACM Trans. Database Syst.*, 9(4):680–710.

Nejdl, W., Siberski, W., and Sintek, M. (2003). Design issues and challenges for rdf- and schema-based peer-to-peer systems. *ACM SIGMOD Rec.*, 32(3):41–46.

Nepal, S. and Ramakrishna, M. (1999). Query processing issues in image (multimedia) databases. In *Proc. 15th Int. Conf. on Data Engineering*, pages 22–29.

Neumann, T. and Weikum, G. (2008). RDF-3X: a RISC-style engine for RDF. *Proc. VLDB Endowment*, 1(1):647–659.

Neumann, T. and Weikum, G. (2009). The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113.

Newman, M. E. J., Watts, D. J., and Strogatz, S. H. (2002). Random graph models of social networks. In *(Sackler NAS Colloquium) Self-Organized Complexity in the Physical, Biological, and Social Sciences*, pages 2566–2573. National Academy of Sciences.

Niamir, B. (1978). Attribute partitioning in a self–adaptive relational database system. Technical Report 192, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Mass.

Nicolas, J. M. (1982). Logic for improving integrity checking in relational data bases. *Acta Informatica*, 18:227–253.

Nikolic, M. and Olteanu, D. (2018). Incremental view maintenance with triple lock factorization benefits. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 365–380.

Novakovic, S., Daglis, A., Bugnion, E., Falsafi, B., and Grot, B. (2014). Scale-out NUMA. In *Architectural Support for Programming Languages and Operating Systems, ASPLOS*, pages 3–18.

Obermack, R. (1982). Distributed deadlock detection algorithm. *ACM Trans. Database Syst.*, 7 (2):187–208.

Okcan, A. and Riedewald, M. (2011). Processing theta-joins using MapReduce. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 949–960.

Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. (2008). Pig latin: a not-so-foreign language for data processing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1099–1110.

Ong, K. W., Papakonstantinou, Y., and Vernoux, R. (2014). The SQL++ semi-structured data model and query language: A capabilities survey of SQL-on-Hadoop, NoSQL and NewSQL databases. CoRR/abs/1405.3631.

Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *Proc. USENIX 2014 Annual Technical Conf.*, pages 305–320.

Ooi, B., Shu, Y., and Tan, K.-L. (2003). Relational data sharing in peer-based data management systems. *ACM SIGMOD Rec.*, 32(3):59–64.

Ouksel, A. M. and Sheth, A. P. (1999). Semantic interoperability in global information systems: A brief introduction to the research area and the special section. *ACM SIGMOD Rec.*, 28(1): 5–12.

Özsoyoglu, Z. M. and Zhou, N. (1987). Distributed query processing in broadcasting local area networks. In *Proc. 20th Hawaii Int. Conf. on System Sciences*, pages 419–429.

Özsu, M. T. (2016). A survey of RDF data management systems. *Front. Comput. Sci.*, 10(3): 418–432.

Pacaci, A. and Özsu, M. T. (2018). Distribution-aware stream partitioning for distributed stream processing systems. In *Proc. 5th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond*, pages 6:1–6:10.

Pacitti, E. and Simon, E. (2000). Update propagation strategies to improve freshness in lazy master replicated databases. *VLDB J.*, 8(3-4):305–318.

Pacitti, E., Simon, E., and de Melo, R. (1998). Improving data freshness in lazy master schemes. In *Proc. 18th IEEE Int. Conf. on Distributed Computing Systems*, pages 164–171.

Pacitti, E., Minet, P., and Simon, E. (1999). Fast algorithms for maintaining replica consistency in lazy master replicated databases. In *Proc. 25th Int. Conf. on Very Large Data Bases*, pages 126–137.

Pacitti, E., Coulon, C., Valduriez, P., and Özsu, M. T. (2005). Preventive replication in a database cluster. *Distrib. Parall. Databases*, 18(3):223–251.

Pacitti, E., Valduriez, P., and Mattoso, M. (2007). Grid data management: open problems and new issues. *Journal of Grid Computing*, 5(3):273–281.

Pacitti, E., Akbarinia, R., and Dick, M. E. (2012). *P2P Techniques for Decentralized Applications*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University.

Palopoli, L. (2003). Experiences using DIKE, a system for supporting cooperative information system and data warehouse design. *Inf. Syst.*, 28(7):835–865.

Palopoli, L., Saccà, D., and Ursino, D. (1998). Semi-automatic semantic discovery of properties from database schemas. In *Proc. 2nd Int. Conf. on Database Eng. and Applications*, pages 244–253.

Palopoli, L., Saccà, D., Terracina, G., and Ursino, D. (1999). A unified graph-based framework for deriving nominal interscheme properties, type conflicts and object cluster similarities. In *Proc. Int. Conf. on Cooperative Inf. Syst.*, pages 34–45.

Palopoli, L., Saccà, D., Terracina, G., and Ursino, D. (2003). Uniform techniques for deriving similarities of objects and subschemes in heterogeneous databases. *IEEE Trans. Knowl. and Data Eng.*, 15(2):271–294.

Papadimitriou, C. H. (1986). *The Theory of Concurrency Control*. Computer Science Press.

Papakonstantinou, Y., Garcia-Molina, H., and Widom, J. (1995). Object exchange across heterogeneous information sources. In *Proc. 11th Int. Conf. on Data Engineering*, pages 251–260.

Pape, C. L., Gançarski, S., and Valduriez, P. (2004). Refresco: Improving query performance through freshness control in a database cluster. In *Proc. Confederated Int. Conf. DOA, CoopIS and ODBASE*, Lecture Notes in Computer Science 3290, pages 174–193.

Paris, J. F. (1986). Voting with witnesses: A consistency scheme for replicated files. In *Proc. 6th IEEE Int. Conf. on Distributed Computing Systems*, pages 606–612.

Pasetto, D. and Akhriev, A. (2011). A comparative study of parallel sort algorithms. In *Proc. 26th ACM SIGPLAN Conf. on Object-Oriented Programming Systems, Languages & Applications*, pages 203–204.

Passerini, A., Frasconi, P., and Soda, G. (2001). Evaluation methods for focused crawling. In *Proc. 7th Congress of the Italian Association for Artificial Intelligence*, pages 33–39.

Pasupuleti, P. and Purra, B. S. (2015). *Data Lake Development with Big Data*. Packt Books.

Patiño-Martínez, M., Jiménez-Peris, R., Kemme, B., and Alonso, G. (2000). Scalable replication in database clusters. In *Proc. 14th Int. Symp. on Distributed Computing*, pages 315–329.

Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D. J., Madden, S., and Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 165–178.

Pavlo, A., Curino, C., and Zdonik, S. B. (2012). Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 61–72.

Perez-Sorrosal, F., Vuckovic, J., Patiño-Martínez, M., and Jiménez-Peris, R. (2006). Highly available long running transactions and activities for J2EE. In *Proc. 26th IEEE Int. Conf. on Distributed Computing Systems*, page 2.

Petraki, E., Idreos, S., and Manegold, S. (2015). Holistic indexing in main-memory column-stores. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1153–1166.

Pike, R., Dorward, S., Griesemer, R., and Quinlan, S. (2005). Interpreting the data: Parallel analysis with sawzall. *Sci. Program.*, 13(4):277–298.

Pirahesh, H., Mohan, C., Cheng, J. M., Liu, T. S., and Selinger, P. G. (1990). Parallelism in rdbms : Architectural issues and design. In *Proc. 2nd Int. Symp. on Databases in Distributed and Parallel Systems*, pages 4–29.

Pirk, H., Manegold, S., and Kersten, M. (2014). Waste not ... efficient co-processing of relational data. In *Proc. 30th Int. Conf. on Data Engineering*, pages 508–519.

Plattner, C. and Alonso, G. (2004). Ganymed: Scalable replication for transactional web applications. In *Proc. ACM/IFIP/USENIX 5th Int. Middleware Conf.*, pages 155–174.

Plugge, E., Membrey, P., and Hawkins, T. (2010). *The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing*. Apress.

Popa, L., Velegrakis, Y., Miller, R. J., Hernandez, M. A., and Fagin, R. (2002). Translating web data. In *Proc. 28th Int. Conf. on Very Large Data Bases*.

Porto, F., Laber, E. S., and Valduriez, P. (2003). Cherry picking: A semantic query processing strategy for the evaluation of expensive predicates. In *Proc. Brazilian Symposium on Databases*, pages 356–370.

Ports, D. R. K. and Grittner, K. (2012). Serializable snapshot isolation in postgresql. *Proc. VLDB Endowment*, 5(12):1850–1861.

Pottinger, R. and Levy, A. Y. (2000). A scalable algorithm for answering queries using views. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 484–495.

Pu, C. (1988). Superdatabases for composition of heterogeneous databases. In *Proc. 4th Int. Conf. on Data Engineering*, pages 548–555.

Pu, C. and Leff, A. (1991). Replica control in distributed systems: An asynchronous approach. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 377–386.

Qian, Z., He, Y., Su, C., Wu, Z., Zhu, H., Zhang, T., Zhou, L., Yu, Y., and Zhang, Z. (2013). Timestream: Reliable stream computation in the cloud. In *Proc. 8th ACM SIGOPS/EuroSys European Conf. on Comp. Syst.*, pages 1–14.

Qin, L., Yu, J. X., Chang, L., Cheng, H., Zhang, C., and Lin, X. (2014). Scalable big graph processing in mapreduce. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 827–838.

Quamar, A., Kumar, K. A., and Deshpande, A. (2013). Sword: Scalable workload-aware data placement for transactional workloads. In *Proc. 16th Int. Conf. on Extending Database Technology*, pages 430–441.

Raghavan, S. and Garcia-Molina, H. (2001). Crawling the hidden web. In *Proc. 27th Int. Conf. on Very Large Data Bases*.

Raghavan, S. and Garcia-Molina, H. (2003). Representing web graphs. In *Proc. 19th Int. Conf. on Data Engineering*, pages 405–416.

Rahal, A., Zhu, Q., and Larson, P.-Å. (2004). Evolutionary techniques for updating query cost models in a dynamic multidatabase environment. *VLDB J.*, 13(2):162–176.

Rahimi, S. (1987). Reference architecture for distributed database management systems. In *Proc. 3th Int. Conf. on Data Engineering*. Tutorial Notes.

Rahimi, S. K. and Haug, F. S. (2010). *Distributed Database Management Systems – A Practical Approach*. Wiley.

Rahm, E. and Bernstein, P. a. (2001). A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350.

Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *Q. Bull. IEEE TC on Data Eng.*, 23(4):3–13.

Rahm, E. and Marek, R. (1995). Dynamic multi-resource load balancing in parallel database systems. In *Proc. 21th Int. Conf. on Very Large Data Bases*, pages 395–406.

Ramabhadran, S., Ratnasamy, S., Hellerstein, J. M., and Shenker, S. (2004). Brief announcement: prefix hash tree. In *Proc. ACM SIGACT-SIGOPS 23rd Symp. on the Principles of Distributed Computing*, page 368.

Ramamoorthy, C. V. and Wah, B. W. (1983). The isomorphism of simple file allocation. *IEEE Trans. Comput.*, 32:221–223.

Ramamritham, K. and Pu, C. (1995). A formal characterization of epsilon serializability. *IEEE Trans. Knowl. and Data Eng.*, 7(6):997–1007.

Raman, V. and Hellerstein, J. M. (2001). Potter's wheel: An interactive data cleaning system. In *Proc. 27th Int. Conf. on Very Large Data Bases*, pages 381–390.

Raman, V., Deshpande, A., and Hellerstein, J. M. (2003). Using state modules for adaptive query processing. In *Proc. 19th Int. Conf. on Data Engineering*, pages 353–365.

Rao, J., Zhang, C., Megiddo, N., and Lohman, G. (2002). Automating physical database design in a parallel database. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*.

Rastogi, V., Machanavajjhala, A., Chitnis, L., and Sarma, A. D. (2013). Finding connected components in map-reduce in logarithmic rounds. In *Proc. 29th Int. Conf. on Data Engineering*, pages 50–61.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. (2001). A scalable content-addressable network. *ACM SIGCOMM Computer Communication Review*, 31(4):161–172.

Ray, I., Mancini, L. V., Jajodia, S., and Bertino, E. (2000). Asep: A secure and flexible commit protocol for mls distributed database systems. *IEEE Trans. Knowl. and Data Eng.*, 12(6):880–899.

Redmond, E. and Wilson, J. R. (2012). *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. The Pragmatic Programmers.

Reed, D. P. (1978). *Naming and Synchronization in a Decentralized Computer System*. PhD thesis, MIT.

Reiss, F. and Hellerstein, J. (2005). Data triage: an adaptive architecture for load shedding in telegraphCQ. In *Proc. 21st Int. Conf. on Data Engineering*, pages 155–156.

Rekatsinas, T., Joglekar, M., Garcia-Molina, H., Parameswaran, A. G., and Ré, C. (2017). SLiMFast: Guaranteed results for data fusion and source reliability. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 1399–1414.

Revilak, S., O'Neil, P. E., and O'Neil, E. J. (2011). Precisely serializable snapshot isolation (PSSI). In *Proc. 27th Int. Conf. on Data Engineering*, pages 482–493.

Ribeiro-Neto, B. A. and Barbosa, R. A. (1998). Query performance for tightly coupled distributed digital libraries. In *Proc. 3rd ACM Int. Conf. on Digital Libraries*, pages 182–190.

Richter, S., Quiané-Ruiz, J.-A., Schuh, S., and Dittrich, J. (2013). Towards zero-overhead static and adaptive indexing in Hadoop. *VLDB J.*, 23(3):469–494.

Ritter, J. (2001). Why Gnutella can't scale, no, really. http://www.darkridge.com/~jpr5/doc/gnutella.html. Last accessed June 2019.

Rivera-Vega, P., Varadarajan, R., and Navathe, S. B. (1990). Scheduling data redistribution in distributed databases. In *Proc. Int. Conf. on Data Eng*, pages 166–173.

Rjaibi, W. (2004). An introduction to multilevel secure relational database management systems. In *Proc. Conf. of the IBM Centre for Advanced Studies on Collaborative Research*, pages 232–241.

Robinson, I., Webber, J., and Eifrem, E. (2015). *Graph Databases*. O'Reilly, 2 edition.

Röhm, U., Böhm, K., and Schek, H.-J. (2000). OLAP query routing and physical design in a database cluster. In *Advances in Database Technology, Proc. 7th Int. Conf. on Extending Database Technology*, pages 254–268.

Röhm, U., Böhm, K., and Schek, H.-J. (2001). Cache-aware query routing in a cluster of databases. In *Proc. 17th Int. Conf. on Data Engineering*, pages 641–650.

Röhm, U., Böhm, K., Schek, H.-J., and Schuldt, H. (2002). FAS - A freshness-sensitive coordination middleware for a cluster of OLAP components. In *Proc. 28th Int. Conf. on Very Large Data Bases*, pages 754–765.

Roitman, H. and Gal, A. (2006). Ontobuilder: Fully automatic extraction and consolidation of ontologies from web sources using sequence semantics. In *Proc. EDBT Workshops*, volume 4254 of *LNCS*, pages 573–576.

Roth, M. and Schwartz, P. (1997). Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. 23th Int. Conf. on Very Large Data Bases*, pages 266–275.

Roth, M. T., Ozcan, F., and Haas, L. M. (1999). Cost models do matter: Providing cost information for diverse data sources in a federated system. In *Proc. 25th Int. Conf. on Very Large Data Bases*, pages 599–610.

Rothermel, K. and Mohan, C. (1989). Aries/nt: A recovery method based on write-ahead logging for nested transactions. In *Proc. 15th Int. Conf. on Very Large Data Bases*, pages 337–346.

Roubini, N. (2018). Testimony for the hearing of the US senate committee on banking, housing and community affairs on exploring the cryptocurrency and blockchain ecosystem. Accessible at https://www.banking.senate.gov/imo/media/doc/Roubini%20Testimony%2010-11-18.pdf/. Last accessed February 2019.

Roy, A., Mihailovic, I., and Zwaenepoel, W. (2013). X-stream: edge-centric graph processing using streaming partitions. In *Proc. 24th ACM Symp. on Operating System Principles*, pages 472–488.

Ryvkina, E., Maskey, A., Adams, I., Sandler, B., Fuchs, C., Cherniack, M., and Zdonik, S. (2006). Revision processing in a stream processing engine: A high-level design. In *Proc. 22nd Int. Conf. on Data Engineering*, page 141.

Sacca, D. and Wiederhold, G. (1985). Database partitioning in a cluster of processors. *ACM Trans. Database Syst.*, 10(1):29–56.

Sacco, M. S. and Yao, S. B. (1982). Query optimization in distributed data base systems. In Yovits, M., editor, *Advances in Computers*, volume 21, pages 225–273.

Saito, Y. and Shapiro, M. (2005). Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81.

Sakr, S., Liu, A., and Fayoumi, A. G. (2013). The family of MapReduce and large-scale data processing systems. *ACM Comput. Surv.*, 46(1):11:1–11:44.

Salihoglu, S. and Widom, J. (2013). GPS: a graph processing system. In *Proc. 25th Int. Conf. on Scientific and Statistical Database Management*, pages 22:1–22:12.

Salihoglu, S. and Widom, J. (2014). Optimizing graph algorithms on Pregel-like systems. *Proc. VLDB Endowment*, 7(7):577–588.

Salton, G. (1989). *Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2):38–47.

Schenkel, R., Weikum, G., Weißenberg, N., and Wu, X. (2000). Federated transaction management with snapshot isolation. In Saake, G., Schwarz, K., and Türker, C., editors, *Transactions and Database Dynamics*, pages 1–25. Springer.

Schmachtenberg, M., Bizer, C., and Paulheim, H. (2014). Adoption of best data practices in different topical domains. In *Proc. 13th Int. Semantic Web Conf.*, pages 245–260.

Schmidt, C. and Parashar, M. (2004). Enabling flexible queries with guarantees in P2P systems. *IEEE Internet Computing*, 8(3):19–26.

Schreiber, F. (1977). A framework for distributed database systems. In *Proc. Int. Computing Symposium*, pages 475–482.

Schuhknecht, F. M., Jindal, A., and Dittrich, J. (2013). The uncracked pieces in database cracking. *Proc. VLDB Endowment*, 7(2):97–108.

Selinger, P. G. and Adiba, M. (1980). Access path selection in distributed data base management systems. In *Proc. First Int. Conf. on Data Bases*, pages 204–215.

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. (1979). Access path selection in a relational database management system. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 23–34.

Sequeda, J. F., Arenas, M., and Miranker, D. P. (2014). OBDA: query rewriting or materialization? in practice, both! In *Proc. 13th Int. Semantic Web Conf.*, pages 535—551.

Shah, M. A., Hellerstein, J. M., Chandrasekaran, S., and Franklin, M. J. (2003). Flux: An adaptive partitioning operator for continuous query systems. In *Proc. 19th Int. Conf. on Data Engineering*, pages 25–36.

Shao, B., Wang, H., and Li, Y. (2013). Trinity: a distributed graph engine on a memory cloud. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 505–516.

Shatdal, A. and Naughton, J. F. (1993). Using shared virtual memory for parallel join processing. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 119–128.

Shatdal, A., Kant, C., and Naughton, J. F. (1994). Cache conscious algorithms for relational query processing. In *Proc. 20th Int. Conf. on Very Large Data Bases*, pages 510–521.

Shekita, E. J., Young, H. C., and Tan, K. L. (1993). Multi-join optimization for symmetric multiprocessor. In *Proc. 19th Int. Conf. on Very Large Data Bases*, pages 479–492.

Sheth, A., Larson, J., Cornellio, A., and Navathe, S. B. (1988a). A tool for integrating conceptual schemas and user views. In *Proc. 4th Int. Conf. on Data Engineering*, pages 176–183.

Sheth, A., Larson, J., and Watkins, E. (1988b). Tailor, a tool for updating views. In *Advances in Database Technology, Proc. 1st Int. Conf. on Extending Database Technology*, pages 190–213.

Sheth, A. P. and Kashyap, V. (1992). So far (schematically) yet so near (semantically). In *Proc. IFIP WG 2.6 Database Semantics Conf. on Interoperable Database Systems*, pages 283–312.

Sheth, A. P. and Larson, J. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236.

Shute, J., Vingralek, R., Samwel, B., Handy, B., Whipkey, C., Rollins, E., Oancea, M., Littlefield, K., Menestrina, D., Ellner, S., Cieslewicz, J., Rae, I., Stancescu, T., and Apte, H. (2013). F1: A distributed SQL database that scales. *Proc. VLDB Endowment*, 6(11):1068–1079.

Sidell, J., Aoki, P. M., Sah, A., Staelin, C., Stonebraker, M., and Yu, A. (1996). Data replication in Mariposa. In *Proc. 12th Int. Conf. on Data Eng*, pages 485–494.

Sidirourgos, L., Goncalves, R., Kersten, M., Nes, N., and Manegold, S. (2008). Column-store support for RDF data management: not all swans are white. *Proc. VLDB Endowment*, 1(2): 1553–1563.

Silberschatz, A., Korth, H., and Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill, 7 edition.

Simitsis, A., Wilkinson, K., Castellanos, M., and Dayal, U. (2009). QoX-driven ETL design: reducing the cost of ETL consulting engagements. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 953–960.

Simitsis, A., Wilkinson, K., Castellanos, M., and Dayal, U. (2012). Optimizing analytic data flows for multiple execution engines. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 829–840.

Simon, E. and Valduriez, P. (1984). Design and implementation of an extendible integrity subsystem. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 9–17.

Simon, E. and Valduriez, P. (1986). Integrity control in distributed database systems. In *Proc. 19th Hawaii Int. Conf. on System Sciences*, pages 622–632.

Simon, E. and Valduriez, P. (1987). Design and analysis of a relational integrity subsystem. Technical Report DB-015-87, Microelectronics and Computer Corporation, Austin, Tex.

Singhal, M. (1989). Deadlock detection in distributed systems. *Computer*, 22(11):37–48.

Skarra, A. (1989). Concurrency control for cooperating transactions in an object-oriented database. In *Proc. ACM SIGPLAN Workshop on Object-Based Concurrent Programming*, pages 145–147.

Skarra, A., Zdonik, S., and Reiss, S. (1986). An object server for an object-oriented database system. In *Proc. of the 1st Int. Workshop on Object-Oriented Database Systems*, pages 196–204.

Skeen, D. (1981). Nonblocking commit protocols. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 133–142.

Skeen, D. (1982a). A quorum-based commit protocol. In *Proc. 6th Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 69–80.

Skeen, D. (1982b). *Crash Recovery in a Distributed Database Management System*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, Calif.

Skeen, D. and Stonebraker, M. (1983). A formal model of crash recovery in a distributed system. *IEEE Trans. Softw. Eng.*, SE-9(3):219–228.

Skeen, D. and Wright, D. (1984). Increasing availability in partitioned networks. In *Proc. 3rd ACM SIGACT–SIGMOD Symp. on Principles of Database Systems*, pages 290–299.

Somani, A., Choy, D., and Kleewein, J. C. (2002). Bringing together content and data management systems: Challenges and opportunities. *IBM Systems J.*, 41(4):686–696.

Sousa, A., Oliveira, R., Moura, F., and Pedone, F. (2001). Partial replication in the database state machine. In *Proc. IEEE Int. Symp. Network Computing and Applications*, pages 298–309.

Srivastava, U. and Widom, J. (2004a). Flexible time management in data stream systems. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, pages 263–274.

Srivastava, U. and Widom, J. (2004b). Memory-limited execution of windowed stream joins. In *Proc. 30th Int. Conf. on Very Large Data Bases*, pages 324–335.

Stanoi, I., Agrawal, D., and El Abbadi, A. (1998). Using broadcast primitives in replicated databases. In *Proc. 8th IEEE Int. Conf. on Distributed Computing Systems*, pages 148–155.

Stöhr, T., Märtens, H., and Rahm, E. (2000). Multi-dimensional database allocation for parallel data warehouses. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 273–284.

Stoica, I., Morris, R., Karger, D., Kaashoek, M., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 149–160.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M., Dabek, F., and Balakrishnan, H. (2003). Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32.

Stonebraker, M. (1975). Implementation of integrity constraints and views by query modification. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 65–78.

Stonebraker, M. (1981). Operating system support for database management. *Commun. ACM*, 24 (7):412–418.

Stonebraker, M. (1986). The case for shared nothing. *Q. Bull. IEEE TC on Data Eng.*, 9(1):4–9.

Stonebraker, M. and Neuhold, E. (1977). A distributed database version of INGRES. In *Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 9–36.

Stonebraker, M., Abadi, D. J., DeWitt, D. J., Madden, S., Paulson, E., Pavlo, A., and Rasin, A. (2010). MapReduce and parallel DBMSs: friends or foes? *Commun. ACM*, 53(1):64–71.

Strauch, C. (2011). *NoSQL Databases*. Stuttgart Media University.

Sullivan, M. and Heybey, A. (1998). Tribeca: A system for managing large databases of network traffic. In *Proc. USENIX 1998 Annual Technical Conf.*

Swami, A. (1989). Optimization of large join queries: combining heuristics and combinatorial techniques. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 367–376.

Taft, R., Mansour, E., Serafini, M., Duggan, J., Elmore, A. J., Aboulnaga, A., Pavlo, A., and Stonebraker, M. (2014). E-Store: Fine-Grained Elastic Partitioning for Distributed Transaction Processing. *Proc. VLDB Endowment*, 8(3):245–256.

Taft, R., El-Sayed, N., Serafini, M., Lu, Y., Aboulnaga, A., Stonebraker, M., Mayerhofer, R., and Andrade, F. (2018). P-store: An elastic database system with predictive provisioning. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 205–219.

Tandem. (1987). NonStop SQL – a distributed high-performance, high-availability implementation of sql. In *Proc. Int. Workshop on High Performance Transaction Systems*, pages 60–104.

Tandem. (1988). A benchmark of NonStop SQL on the debit credit transaction. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 337–341.

Tanenbaum, A. S. and van Renesse, R. (1988). Voting with ghosts. In *Proc. 8th IEEE Int. Conf. on Distributed Computing Systems*, pages 456–461.

Tang, W., Zhao, X., Rafique, W., Qi, L., Dou, W., and Ni, Q. (2019). An offloading method using decentralized P2P-enabled mobile edge servers in edge computing. *Journal of Systems Architecture – Embedded Systems Design*, 94:1–13.

Tatarinov, I., Ives, Z. G., Madhavan, J., Halevy, A. Y., Suciu, D., Dalvi, N. N., Dong, X., Kadiyska, Y., Miklau, G., and Mork, P. (2003). The piazza peer data management project. *ACM SIGMOD Rec.*, 32(3):47–52.

Tatbul, N., Cetintemel, U., Zdonik, S., Cherniack, M., and Stonebraker, M. (2003). Load shedding in a data stream manager. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 309–320.

Thiran, P., Hainaut, J.-L., Houben, G.-J., and Benslimane, D. (2006). Wrapper-based evolution of legacy information systems. *ACM Trans. Softw. Eng. and Meth.*, 15(4):329–359.

Thomas, R. H. (1979). A majority consensus approach to concurrency control for multiple copy databases. *ACM Trans. Database Syst.*, 4(2):180–209.

Thomasian, A. (1996). *Database Concurrency Control: Methods, Performance, and Analysis*. Kluwer Academic Publishers.

Thomson, A. and Abadi, D. J. (2010). The case for determinism in database systems. *Proc. VLDB Endowment*, 3(1):70–80.

Thuraisingham, B. (2001). Secure distributed database systems. *Information Security Technical Report*, 6(2).

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., and Murthy, R. (2009). Hive: a warehousing solution over a map-reduce framework. *Proc. VLDB Endowment*, 2(2):1626–1629.

Tian, F. and DeWitt, D. J. (2003). Tuple routing strategies for distributed eddies. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 333–344.

Tian, Y., Balmin, A., Corsten, S. A., Tatikonda, S., and McPherson, J. (2013). From "think like a vertex" to "think like a graph". *Proc. VLDB Endowment*, 7(3):193–204.

Tian, Y., Özcan, F., Zou, T., Goncalves, R., and Pirahesh, H. (2016). Building a hybrid warehouse: Efficient joins between data stored in HDFS and enterprise warehouse. *ACM Trans. Database Syst.*, 41(4):21:1–21:38.

Tomasic, A., Raschid, L., and Valduriez, P. (1996). Scaling heterogeneous databases and the design of disco. In *Proc. 16th IEEE Int. Conf. on Distributed Computing Systems*, pages 449–457.

Tomasic, A., Amouroux, R., Bonnet, P., Kapitskaia, O., Naacke, H., and Raschid, L. (1997). The distributed information search component (DISCO) and the world-wide web – prototype demonstration. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 546–548.

Tomasic, A., Raschid, L., and Valduriez, P. (1998). Scaling access to distributed heterogeneous data sources with Disco. In *IEEE Trans. Knowl. and Data Eng*. in press.

Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S., and Ryaboy, D. (2014). Storm@twitter. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 147–156.

Traiger, I. L., Gray, J., Galtieri, C. A., and Lindsay, B. G. (1982). Transactions and recovery in distributed database systems. *ACM Trans. Database Syst.*, 7(3):323–342.

Triantafillou, P. and Taylor, D. J. (1995). The location-based paradigm for replication: Achieving efficiency and availability in distributed systems. *IEEE Trans. Softw. Eng.*, 21(1):1–18.

Tu, S., Zheng, W., Kohler, E., Liskov, B., and Madden, S. (2013). Speedy transactions in multicore in-memory databases. In *Proc. 24th ACM Symp. on Operating System Principles*, pages 18–32.

Tucker, P., Maier, D., Sheard, T., and Faragas, L. (2003). Exploiting punctuation semantics in continuous data streams. *IEEE Trans. Knowl. and Data Eng.*, 15(3):555–568.

Ugander, J. and Backstrom, L. (2013). Balanced label propagation for partitioning massive graphs. In *Proc. 6th ACM Int. Conf. Web Search and Data Mining*, pages 507–516.

Ullman, J. (1997). Information integration using logical views. In *Proc. 6th Int. Conf. on Database Theory*, pages 19–40.

Ullman, J. D. (1982). *Principles of Database Systems*. Computer Science Press, 2nd edition.

Ulusoy, Ö. (2007). Research issues in peer-to-peer data management. In *Proc. 22nd Int. Symp. on Computer and Information Science*, pages 1–8.

Umbrich, J., Hose, K., Karnstedt, M., Harth, A., and Polleres, A. (2011). Comparing data summaries for processing live queries over linked data. *World Wide Web J.*, 14(5-6):495–544.

Urhan, T. and Franklin, M. (2000). XJoin: A reactively-scheduled pipelined join operator. *Q. Bull. IEEE TC on Data Eng.*, 23:27.

Urhan, T., Franklin, M. J., and Amsaleg, L. (1998). Cost based query scrambling for initial delays. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 130–141.

Valduriez, P. (1982). Semi-join algorithms for distributed database machines. In Schneider, J.-J., editor, *Distributed Data Bases*. pages 23–37.

Valduriez, P. (1993). Parallel database systems: Open problems and new issues. *Distrib. Parall. Databases*, 1:137–16.

Valduriez, P. and Gardarin, G. (1984). Join and semi-join algorithms for a multi processor database machine. *ACM Trans. Database Syst.*, 9(1):133–161.

Valduriez, P. and Pacitti, E. (2004). Data management in large-scale P2P systems. In *Proc. 6th Int. Conf. High Performance Comp. for Computational Sci.*, pages 104–118.

Valiant, L. G. (1990). A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111.

van Hee, K. (2002). *Workflow Management*. M.I.T. Press.

Van Renesse, R. and Altinbuken, D. (2015). Paxos made moderately complex. *ACM Comput. Surv.*, 47(3):42:1–42:36.

Varadarajan, R., Rivera-Vega, P., and Navathe, S. B. (1989). Data redistribution scheduling in fully connected networks. In *Proc. 27th Annual Allerton Conf. on Communication, Control, and Computing*.

Velegrakis, Y., Miller, R. J., and Popa, L. (2004). Preserving mapping consistency under schema changes. *VLDB J.*, 13(3):274–293.

Verhofstadt, J. S. (1978). Recovery techniques for database systems. *ACM Comput. Surv.*, 10(2): 168–195.

Verma, S., Leslie, L. M., Shin, Y., and Gupta, I. (2017). An experimental comparison of partitioning strategies in distributed graph processing. *Proc. VLDB Endowment*, 10(5):493–504.

Vermeer, M. (1997). *Semantic Interoperability for Legacy Databases*. Ph.D. thesis, Department of Computer Science, University of Twente, Enschede, Netherlands.

Viglas, S., Naughton, J., and Burger, J. (2003). Maximizing the output rate of multi-join queries over streaming information sources. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 285–296.

Voulgaris, S., Jelasity, M., and van Steen, M. (2003). A robust and scalable peer-to-peer gossiping protocol. In *Agents and Peer-to-Peer Computing, Second Int. Workshop, (AP2PC)*, pages 47–58.

Vu, Q. H., Lupu, M., and Ooi, B. C. (2009). *Peer-to-Peer Computing: Principles and Applications*. Springer.

Wah, B. W. and Lien, Y. N. (1985). Design of distributed databases on local computer systems. *IEEE Trans. Softw. Eng.*, SE-11(7):609–619.

Walton, C., Dale, A., and Jenevin, R. (1991). A taxonomy and performance model of data skew effects in parallel joins. In *Proc. 17th Int. Conf. on Very Large Data Bases*, pages 537–548.

Wang, G., Xie, W., Demers, A. J., and Gehrke, J. (2013). Asynchronous large-scale graph processing made easy. In *Proc. 6th Biennial Conf. on Innovative Data Systems Research*.

Wang, H., Zaniolo, C., and Luo, R. (2003). Atlas: A small but complete SQL extension for data mining and data streams. In *Proc. 29th Int. Conf. on Very Large Data Bases*, pages 1113–1116.

Wang, L., Xiao, Y., Shao, B., and Wang, H. (2014). How to partition a billion-node graph. In *Proc. 30th Int. Conf. on Data Engineering*, pages 568–579.

Wang, W., Li, J., Zhang, D., and Guo, L. (2004). Processing sliding window join aggregate in continuous queries over data streams. In *Proc. 8th East European Conf. Advances in Databases and Information Systems*, pages 348–363.

Weikum, G. and Vossen, G. (2001). *Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control*. Morgan Kaufmann.

Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D. E., and Maltzahn, C. (2006). Ceph: A scalable, high-performance distributed file system. In *Proc. 7th USENIX Symp. on Operating System Design and Implementation*, pages 307–320.

Weiss, C., Karras, P., and Bernstein, A. (2008). Hexastore: sextuple indexing for semantic web data management. *Proc. VLDB Endowment*, 1(1):1008–1019.

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *Computer*, 25 (3):38–49.

Wiesmann, M., Schiper, A., Pedone, F., Kemme, B., and Alonso, G. (2000). Database replication techniques: A three parameter classification. In *Proc. 28th Symp. on Reliable Distributed Systems*, pages 206–215.

Wilkinson, K. (2006). Jena property table implementation. Technical Report HPL-2006-140, HP Laboratories Palo Alto.

Wilms, P. F. and Lindsay, B. G. (1981). A database authorization mechanism supporting individual and group authorization. Research Report RJ 3137, IBM Almaden Research Laboratory, San Jose, Calif.

Wilschut, A. and Apers, P. (1991). Dataflow query execution in a parallel main-memory environment. In *Proc. 1st Int. Conf. on Parallel and Distributed Information Systems*, pages 68–77.

Wilson, B. and Navathe, S. B. (1986). An analytical framework for the redesign of distributed databases. In *Proc. 6th Advanced Database Symposium*, pages 77–83.

Wolfson, O. (1987). The overhead of locking (and commit) protocols in distributed databases. *ACM Trans. Database Syst.*, 12(3):453–471.

Wong, E. (1977). Retrieving dispersed data from SDD-1. In *Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks*, pages 217–235.

Wong, E. and Youssefi, K. (1976). Decomposition: A strategy for query processing. *ACM Trans. Database Syst.*, 1(3):223–241.

Wright, D. D. (1983). Managing distributed databases in partitioned networks. Technical Report TR83-572, Department of Computer Science, Cornell University, Ithaca, N.Y.

Wu, E., Diao, Y., and Rizvi, S. (2006). High-performance complex event processing over streams. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 407–418.

Wu, K.-L., Yu, P. S., and Pu, C. (1997). Divergence control algorithms for epsilon serializability. *IEEE Trans. Knowl. and Data Eng.*, 9(2):262–274.

Wu, S., Yu, G., Yu, Y., Ou, Z., Yang, X., and Gu, Y. (2005). A deadline-sensitive approach for real-time processing of sliding windows. In *Proc. 6th Int. Conf. on Web-Age Information Management*, pages 566–577.

Xing, Y., Hwang, J.-H., Çetintemel, U., and Zdonik, S. (2006). Providing resiliency to load variations in distributed stream processing. In *Proc. 32nd Int. Conf. on Very Large Data Bases*, pages 775–786.

Yan, D., Cheng, J., Lu, Y., and Ng, W. (2014). Blogel: A block-centric framework for distributed computation on real-world graphs. *Proc. VLDB Endowment*, 7(14):1981–1992.

Yan, D., Bu, Y., Tian, Y., and Deshpande, A. (2017). Big graph analytics platforms. *Foundations and Trends in Databases*, 7(1-2):1–195.

Yan, L. L. (1997). Towards efficient and scalable mediation: The AURORA approach. In *Proc. IBM CASCON Conference*, pages 15–29.

Yan, L.-L., Özsu, M. T., and Liu, L. (1997). Accessing heterogeneous data through homogenization and integration mediators. In *Proc. Int. Conf. on Cooperative Inf. Syst.*, pages 130–139.

Yan, L. L., Miller, R. J., Haas, L. M., and Fagin, R. (2001). Data-driven understanding and refinement of schema mappings. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 485–496.

Yang, B. and Garcia-Molina, H. (2002). Improving search in peer-to-peer networks. In *Proc. 22nd IEEE Int. Conf. on Distributed Computing Systems*, pages 5–14.

Yang, X., Lee, M.-L., and Ling, T. W. (2003). Resolving structural conflicts in the integration of XML schemas: A semantic approach. In *Proc. 22nd Int. Conf. on Conceptual Modeling*, pages 520–533.

Yao, S. B., Waddle, V., and Housel, B. (1982). View modeling and integration using the functional data model. *IEEE Trans. Softw. Eng.*, SE-8(6):544–554.

Yu, C. and Meng, W. (1998). *Principles of Query Processing for Advanced Database Applictions*. Morgan Kaufmann.

Zaharia, M. (2016). *An Architecture for Fast and General Data Processing on Large Clusters*. ACM Books.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proc. 2nd USENIX Workshop on Hot Topics in Cloud Computing*, pages 10–10.

Zaharia, M., Das, T., an dTimothy Hunter, H. L., Shenker, S., and Stoica, I. (2013). Discretized streams: Fault-tolerant streaming computation at scale. In *Proc. 24th ACM Symp. on Operating System Principles*, pages 423–438.

Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A. D., and Kubiatowicz, J. (2004). Tapestry: A resilient global-scale overlay for service deployment. *IEEE J. Selected Areas in Comm.*, 22 (1):41–53.

Zhu, M. and Risch, T. (2011). Querying combined cloud-based and relational databases. In *Proc. 2011 Int. Conf. on Cloud and Service Comp.*, pages 330–335.

Zhu, Q. (1995). *Estimating Local Cost Parameters for Global Query Optimization in a Multidatabase System*. Ph.D. thesis, Department of Computer Science, University of Waterloo, Waterloo, Canada.

Zhu, Q. and Larson, P.-Å. (1994). A query sampling method of estimating local cost parameters in a multidatabase system. In *Proc. 10th Int. Conf. on Data Engineering*, pages 144–153.

Zhu, Q. and Larson, P. A. (1996a). Global query processing and optimization in the CORDS multidatabase system. In *Proc. Int. Conf. on Parallel and Distributed Computing Systems*, pages 640–647.

Zhu, Q. and Larson, P. A. (1996b). Developing regression cost models for multidatabase systems. In *Proc. 4th Int. Conf. on Parallel and Distributed Information Systems*, pages 220–231.

Zhu, Q. and Larson, P. A. (1998). Solving local cost estimation problem for global query optimization in multidatabase systems. *Distrib. Parall. Databases*, 6(4):373–420.

Zhu, Q., Sun, Y., and Motheramgari, S. (2000). Developing cost models with qualitative variables for dynamic multidatabase environments. In *Proc. 16th Int. Conf. on Data Engineering*, pages 413–424.

Zhu, Q., Motheramgari, S., and Sun, Y. (2003). Cost estimation for queries experiencing multiple contention states in dynamic multidatabase environments. *Knowledge and Information Systems*, 5(1):26–49.

Zhu, Y., Rundensteiner, E., and Heineman, G. (2004). Dynamic plan migration for continuous queries over data streams. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 431–442.

Zhu, Y., Zhang, H., Qin, L., and Cheng, H. (2017). Efficient MapReduce algorithms for triangle listing in billion-scale graphs. *Distrib. Parall. Databases*, 35(2):149–176.

Ziane, M., Zaït, M., and Borla-Salamet, P. (1993). Parallel query processing with zigzag trees. *VLDB J.*, 2(3):277–301.

Zilio, D. C. (1998). *Physical Database Design Decision Algorithms and Concurrent Reorganization for Parallel Database Systems*. PhD thesis, University of Toronto.

Zou, L. and Özsu, M. T. (2017). Graph-based RDF data management. *Data Science and Engineering*, 2(1):56–70.

Zou, L., Mo, J., Chen, L., Özsu, M. T., and Zhao, D. (2011). gStore: answering SPARQL queries via subgraph matching. *Proc. VLDB Endowment*, 4(8):482–493.

Zou, L., Özsu, M. T., Chen, L., Shen, X., Huang, R., and Zhao, D. (2014). gStore: A graph-based SPARQL query engine. *VLDB J.*, 23(4):565–590.

Index

M. T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*,
https://doi.org/10.1007/978-3-030-26253-2