10.3 Document types

This section has been set as optional by your instructor.

Document types

Structured data is stored as a fixed set of named data elements, organized in groups. A type is explicitly declared for each element. Each group has the same number of elements with the same names and types. Ex: Spreadsheets and relational tables contain structured data.

Semistructured data is similar to structured data, except each group may have a different number of elements with different names. Types are not explicitly declared. Instead, elements are stored as characters, and type is inferred from the data. Ex: `<Temperature>98.6</Temperature>` represents an element 98.6 named Temperature with a decimal type.

Unstructured data is stored as elements embedded in a continuous string of characters or bits. Element names and types are not declared. Ex: A statistical report contains numerous data elements, but individual elements are not explicitly separated or named. Ex: A bitmap image may contain images of people, but sophisticated algorithms are necessary to identify each individual.

Semistructured data is stored in a ***document*** as text in a flexible format, such as XML or JSON. Most relational databases support XML or JSON document types. Each value of a document type is a complete XML or JSON document and may contain many elements.

Table 10.3.1: Document type support.

	XML	JSON
MySQL	-	✓
Oracle Database	✓	-
PostgreSQL	✓	✓
SQL Server	✓	-
DB2	✓	-
Informix	-	-

Feedback?

PARTICIPATION ACTIVITY 10.3.1: Structured, semistructured, and unstructured data.

Match the data category to the example.

Unstructured	The Library of Congress scans all books and documents in the collection. Scanned images are converted to characters with optical character recognition software. Each book is associated with a title and author. Although title and author are structured data, unstructured data is contained in the long character strings scanned from books and documents.	Correct
Structured	A file contains 80,000 lines of data. Each line is separated by a new line character and contains 12 data values separated by commas. A second file contains descriptions of each of the 12 values in a line. Each line of the file corresponds to a row of a table. Each value corresponds to a column. Column names and types can be inferred from the information in the second file.	Correct
	Bloomberg News provides a financial information service. The service streams financial	Correct

Semistructured	data as a series of records. Each record contains current price and identifier for a financial instrument, such as a stock or bond. Additional information in the record depends on the type of the financial instrument, as ~~described in a reference manual~~. The reference manual indicates name and type of all data in the records. Since each record may contain different values, the data is semistructured.

Reset

Feedback?

XML format

XML stands for eXtensible Markup Language and uses tags instead of columns. A ***tag*** is a name enclosed in angle brackets < >. An ***XML element*** consists of a start tag, data, and an end tag. The end tag is the start tag with a forward slash / before the name. Ex: `<Department>Accounting</Department>` is an element with tag name 'Department' and data 'Accounting'.

XML elements can be nested by embedding one pair of start and end tags within another. An XML document must have a ***root*** element that contains all other elements.

XML documents have an optional first line, called the ***declaration***, that specifies document processing information such as the XML version and character encoding. Ex: `<?xml version = "2.0" encoding = "UTF-8"?>` indicates the document is formatted with XML version 2.0 and the UTF-8 Unicode encoding.

PARTICIPATION ACTIVITY 10.3.2: XML format.

1 2 3 4 5 6 7 2x speed

```
<?xml version = "2.0" encoding = "UTF-16"?>
<Customer>
   <Name>Maria Rodriguez</Name>
   <Vehicle>
      <Make>Ford</Make>
      <Model>F-150</Model>
      <Year>2008</Year>
   </Vehicle>
   <Vehicle>
      <Make>Toyota</Make>
```

```
        <Model>Camry</Model>
        <Year>2019</Year>
    </Vehicle>
    <Budget></Budget>
    <PreviousCustomer>true</PreviousCustomer>
    <FamilyMember>Jose</FamilyMember>
    <FamilyMember>Felicia</FamilyMember>
    <FamilyMember>Isabella</FamilyMember>
    <Notes>Shopping for a new sports car. Interested in leasing.</Notes>
</Customer>
```

Customer is a repeat customer with three family members.

Captions

1. The declaration is optional. This document is formatted in XML version 2.0 and the UTF-16 character encoding.
2. The root element is required. This document describes a customer.
3. Customer name is Maria Rodriguez.
4. Customer owns a 2008 Ford F-150.
5. Customer also owns a 2019 Toyota Camry.
6. Customer's budget is unknown.
7. Customer is a repeat customer with three family members.

Feedback?

XML tags are similar to relational column names but have several advantages:

- **Readable**. Tag names and embedded data are legible in an XML document and easy to understand.
- **Flexible**. Tags can be easily added or dropped as elements are inserted into or removed from a document. Flexibility is important for semistructured data, such as user click sequences on a website.
- **Hierarchical**. Hierarchical data is easily represented by nesting elements within elements. In comparison, the relational representation of hierarchical data requires multiple tables, foreign keys, and referential integrity rules.

The primary disadvantage of XML is document size. Each element appears between a pair of tags, and each pair is repeated for every value. Ex: A ten-character tag repeated over a million elements requires 20 million bytes (10 characters × 2 tags per element × 1,000,000 elements). By comparison, only a few bytes of overhead are necessary for each relational column.

PARTICIPATION ACTIVITY 10.3.3: XML format.

Refer to this XML:

```
<Library>
  <Book>
    <Title>For Whom the Bell Tolls</Title>
    <Author>Ernest Hemingway</Author>
    <Publisher>Random House</Publisher>
    <Copyright>1940</Copyright>
  <___A___>
  <Book>
    <Title>The Map of Knowledge</Title>
    <Author>Violet Moller</Author>
    <Publisher>Doubelday</Publisher>
    <Copyright>2019</Copyright>
  </Book>
  <Movie>
    <Title>La La Land</Title>
    <Director>Damien Chazelle</Director>
    <Producer>Fred Berger</Producer>
    <Producer>Jordan Horowitz</Producer>
    <Producer>Gary Gilbert</Producer>
    <Producer>Mark Platt</Producer>
    <___B___>
      <Budget>$30,000,000</Budget>
      <Revenue>$446,100,000</Revenue>
    </Financial>
    <Copyright>2016</Copyright>
    <RunTime>128 minutes</RunTime>
  </Movie>
___C___
```

1) What is A?

/Book

Check Show answer

Correct

/Book

This end tag matches the start tag <Book>. End tags have a / prior to the tag name.

2) What is B?

Financial

Check Show answer

Correct

Financial

This start tag matches the end tag </Financial>. The start tag is the end tag without a /.

3) What is C?

</Library>

Check Show answer

Correct

</Library>

This tag ends the XML document. Since the document starts with the <Library> tag, the end tag is </Library>.

4) How many levels are in the XML hierarchy?

4

Correct

4 or Four

Check Show answer

<Budget> and <Revenue> are the deepest tags in the hierarchy. The hierarchy of tags is <Library> - <Movie> - <Financial> - <Budget>, or four levels.

Feedback?

JSON format

JSON stands for JavaScript Object Notation and is commonly pronounced 'JAY-sun'. JSON format is similar to XML but more compact.

A ***JSON element*** consists of a name and associated data, written as `"name":data`. Ex: The JSON element `"Department":"Accounting"` is equivalent to the XML element `<Department>Accounting</Department>`. The element name is not repeated, reducing document size compared to XML.

Unlike XML, JSON elements have a type. The data following an element name must be one of six types:

- String — a series of characters enclosed in double quotes
- Number — a series of digits with an optional decimal point
- Boolean — the strings `true` or `false`
- Null — the string `null`
- Array — multiple data values enclosed in brackets. Ex: `["Arabic","English","Spanish"]`.
- Object — multiple elements enclosed in braces. Ex: `{"Employee":"Sam Snead","Salary":55000}`.

Hierarchical data is represented in JSON by nesting elements, as in XML. Ex: First and Last elements are nested in a Name element, and Name and Salary elements are nested in an Employee element in the following JSON: `"Employee":{"Name":{"First":"Sam", "Last":"Snead"},"Salary":55000}`.

XML is an early document type, developed as internet use proliferated in the 1990s. JSON became popular about ten years later, replacing XML in applications when document size was important. JSON is commonly used to transmit data over the internet — the compact format reduces transmission time and improves application performance.

Terminology

*The **name** of a JSON element is commonly called a key and the **data** is commonly called a value. This section uses the terms name and data to avoid confusion with*

the primary key of a table and the value in a cell of a table.

PARTICIPATION ACTIVITY | 10.3.4: JSON format.

1 2 3 4 5 6 2x speed

```
{
   "Customer":
   {
      "Name": "Maria Rodriguez",
      "Vehicle"
      [
            { "Make": "Ford", "Model": "F-150", "Year": 2008 },
            { "Make": "Toyota", "Model": "Camry", "Year": 2019 }
      ],
      "Budget": null,
      "PreviousCustomer": true,
      "FamilyMember": [ "Jose", "Felicia", "Isabella" ],
      "Notes": "Shopping for a new sports car. Interested in leasing."
   }
}
```

Customer is a repeat customer with three family members.

Captions

1. The JSON document is equivalent to the XML document in prior animation.
2. Customer data is a JSON object.
3. Customer name is Maria Rodriguez.
4. Customer owns two vehicles, represented as an array of objects.
5. Customer's budget is unknown.
6. Customer is a repeat customer with three family members.

Feedback?

PARTICIPATION ACTIVITY | 10.3.5: JSON format.

Refer to following JSON document:

```
[
  { "Name": "oreo",
    "Type": "cookie",
    "Flavors": ["chocolate", "vanilla"],
    "Favorite": false,
    "Created": 1912
  },
  { "Name": "snickers",
    "Type": "candy bar",
    "Flavors": ["chocolate", "peanuts", "caramel", "nougat"],
    "Favorite": true,
    "Created": 1930
  },
  { "Name": "malt",
    "Type": "frozen dairy",
    "Flavors": ["vanilla", "chocolate", "strawberry"],
    "Favorite": false,
    "Created": 1922
  }
]
```

1) What type of data is created by the outer brackets in the JSON document?

- (•) array
- ( ) boolean
- ( ) object

Correct

The JSON document produces an array of three desserts because the dessert data is surrounded by brackets.

2) How many objects are created by the JSON document?

- ( ) 1
- (•) 3
- ( ) 4

Correct

JSON objects are wrapped in braces. The three dessert objects are elements of a single array.

3) What is the data type of `Favorite`?

- (•) boolean
- ( ) object
- ( ) string

Correct

`Favorite` only contains true and false data.

4) What is the data type of `Created`?

- (•) number
- ( ) object

Correct

`Created` only contains digits.

○ string

Feedback?

XML and JSON types

Most relational databases support XML or JSON types, but capabilities and internal format vary greatly.

MySQL supports the JSON type. A JSON value is stored as an internal binary format rather than a character string. The format is optimized so that, given an element name or array index, MySQL can quickly find element data. If the document were stored as a character string, the database would have to read and parse the entire document to find an element, which is relatively slow.

MySQL supports comparisons of JSON values with the < > and = operators but not BETWEEN and IN. In addition, MySQL provides roughly 30 functions that manipulate JSON values. Ex:

- ***JSON_ARRAY()*** formats string or numeric data as a JSON array.
- ***JSON_DEPTH()*** determines the maximum number of levels in a JSON document hierarchy.
- ***JSON_EXTRACT()*** returns data from a JSON document.
- ***JSON_OBJECT()*** converts element names and data to a JSON document.
- ***JSON_PRETTY()*** prints a JSON document in a format that is easy to read, with one element per line.

Like document type implementations in most databases, MySQL checks JSON values for correct syntax and does not store invalid documents.

PARTICIPATION ACTIVITY 10.3.6: MySQL JSON type.

1 2 3 4 2x speed

```
CREATE TABLE Library (
   Code VARCHAR(10),
   Book JSON
);

INSERT INTO Library
VALUES (103, JSON_OBJECT('Title', 'War and Peace')),
       (192, JSON_OBJECT('Title', 'Tom Sawyer', 'Author', 'Mark Twain', 'Pages', 145))
```

```
SELECT Code, Book
FROM Library;
```

Result

Code	Book
103	{"Title": "War and Peace"}

192 {"Title": "Tom Sawyer", "Author": "Mark Twain", "Pages": 14

```
SELECT JSON_EXTRACT(Book, '$.Title') AS Title
FROM Library
WHERE JSON_EXTRACT(Book, '$.Pages') > 100;
```

Result

Title
"Tom Sawyer"

The SELECT statement uses JSON_EXTRACT() to extract data from the Book column. Only "Tom S has a Pages element.

Captions

1. The Library table contains a Book column with JSON type.
2. JSON_OBJECT() converts element names and data to a JSON document, stored in the internal MySQL format.
3. The SELECT statement retrieves JSON documents. Each document appears on one line.
4. The SELECT statement uses JSON_EXTRACT() to extract data from the Book column. Only "Tom Sawyer" has a Pages element.

Feedback?

PARTICIPATION ACTIVITY 10.3.7: XML and JSON types.

1) Functions that manipulate XML and JSON values are similar in most relational databases.

- True
- False

Correct

Although databases that support XML and JSON types provide functions to manipulate documents, function syntax and capabilities varies greatly.

2) JSON documents can be stored as a VARCHAR type in MySQL.

- True
- False

Correct

JSON documents are strings of characters and therefore can be stored as VARCHAR. However VARCHAR does not provide the optimized JSON storage and does not automatically validate JSON syntax. Furthermore, JSON functions cannot be applied to VARCHAR columns.

3) When a document is inserted to a column with XML or JSON type, databases generate an

Correct

Although database implementations of XML and JSON types vary greatly, all databases check syntax when a document is stored.

error if the syntax is incorrect.

- (●) True
- ( ) False

4) In MySQL, tables can be joined on columns with type JSON.

- (●) True
- ( ) False

Correct

MySQL supports comparison operators < > = on JSON values. Tables can be joined on any columns that support comparison operators. Join performance depends on the speed of the comparison, however, which may be slow for long documents.

Feedback?

CHALLENGE ACTIVITY 10.3.1: Document types.

379958.2369558.qx3zqy7

Jump to level 1

1 2 3 4 5

Consider the MySQL table and SELECT query.

Traveler

ID	Name	Country
712	Dan Hart	{"Name": "Cuba", "Code": 192, "IndepYear": 1868}
610	Ani Holt	{"Name": "Belgium", "Code": 56, "ContinentCode": "EU"}
980	Aya Reid	{"Name": "Cambodia", "Population": 16250000}
124	Val Dunn	{"Name": "Mozambique", "Code": 508, "IndepYear": 1975}

```
SELECT JSON_EXTRACT(Country, '$.Name')
FROM Traveler
WHERE JSON_EXTRACT(Country, '$.Code') > 290;
```

What columns appear in the query result?

- [x] Name
- [ ] CountryName
- [ ] Country

What data appears in the quer

- [ ] "Cuba"
- [ ] "Belgium"
- [ ] "Cambodia"
- [x] "Mozambique"

1 2 3 4 **5**

Check Next **Done**. Click any level to practice more. Completion is preserved.

✔ Expected:
Name
"Mozambique"

`SELECT JSON_EXTRACT(Country, '$.Name')` extracts `Name` data from the JSON col
Therefore, the column of the Result table will be Name.

`WHERE JSON_EXTRACT(Country, '$.Code') > 290;` selects the Countries whose C

Cambodia does not have a Code element, so is not selected. Cuba has a Code element not selected. Belgium has a Code element of 56 <= 290, so is not selected. Mozambique has a Code element of 508, so is selected.

The Result table of the SELECT query is:

Name
Mozambique

Feedback?

Exploring further:

- MySQL JSON type
- MySQL JSON functions
- Oracle Database XML type
- PostgreSQL XML type
- PostgreSQL JSON type

How was this section? **Provide feedback**