УДК 004.624

СРАВНЕНИЕ УСТОЙЧИВОСТИ ФОРМАТОВ ДАННЫХ К ПОВРЕЖДЕНИЯМ ПРИ МЕЖСИСТЕМНОМ ВЗАИМОДЕЙСТВИИ

С. В. ЛУКАШЕВИЧ, В. Л. ШАБАЛОВ, В. В. ЛУКАШЕВИЧ
(Представлено: А. А. СКУКОВСКАЯ)

В статье рассматриваются вопросы надёжности различных форматов данных – CSV, JSON, XML и Excel – при частичной потере или повреждении информации в процессе передачи между системами. Оцениваются такие параметры, как восстанавливаемость данных, чувствительность к синтаксическим ошибкам и возможность частичного чтения.

Ключевые слова: *форматы данных, надёжность, повреждение данных, CSV, JSON, XML, Excel, межсистемный обмен.*

Введение. При интеграции различных программных систем данные часто передаются через промежуточные форматы, такие как CSV, JSON, XML или файлы Excel. Однако на практике каналы передачи могут быть ненадёжными: возможны обрывы соединения, ошибки записи на диск, несовместимость кодировок или человеческий фактор. В таких условиях важно понимать, насколько устойчив выбранный формат к частичным повреждениям и способен ли он сохранить хотя бы часть полезной информации.

Основная часть. *CSV (Comma-Separated Values).* CSV – это простой текстовый формат, в котором каждая строка представляет собой запись, а поля внутри строки разделяются запятыми (или другими символами, например, точкой с запятой). Формат не имеет строгой стандартизации, хотя существует рекомендация RFC 4180.

Особенности структуры:

– Нет встроенной информации о типах данных (всё – строки).

– Заголовок (если есть) не обязателен.

– Специальные символы (запятые, кавычки, переносы строк) должны экранироваться с помощью двойных кавычек.

Устойчивость к повреждениям:

– Низкая. Даже незначительное повреждение (например, пропущенная кавычка или лишняя запятая) может нарушить выравнивание колонок во всех последующих строках.

– Отсутствие метаданных не позволяет парсеру понять, где заканчивается одна запись и начинается другая.

– При обрыве передачи невозможно определить, завершена ли последняя строка.

Пример уязвимости. Если в поле «Комментарий» содержится текст "Важно, срочно!", но кавычки не экранированы, парсер интерпретирует запятую как разделитель полей, что приведёт к смещению данных.

JSON (JavaScript Object Notation). JSON – это лёгкий текстовый формат обмена данными, основанный на синтаксисе объектов JavaScript. Он поддерживает вложенные структуры, массивы, строки, числа, логические значения и null.

Особенности структуры:

– Строгий синтаксис: обязательные двойные кавычки для ключей и строк, запятые между элементами, правильное вложение фигурных и квадратных скобок.

– Не поддерживает комментарии.

– Часто используется в веб-API и конфигурационных файлах.

Устойчивость к повреждениям:

– Очень низкая для классического JSON. Любая синтаксическая ошибка (например, отсутствие запятой или скобки) делает весь документ недействительным – парсер не сможет прочитать ни одного элемента.

– Однако существует альтернатива – JSONL (JSON Lines), где каждый объект записан в отдельной строке. В этом случае повреждение одной строки не влияет на остальные, что значительно повышает надёжность.

Пример уязвимости. Если при передаче обрывается последняя скобка (}), стандартный JSON-парсер выдаст ошибку и не вернёт никаких данных, даже если 99% документа корректны.

XML (eXtensible Markup Language). XML – это разметочный язык, предназначенный для хранения и передачи структурированных данных. Он поддерживает иерархию, атрибуты, пространства имён и валидацию через схемы (XSD).

Особенности структуры:

– Данные заключаются в открывающие и закрывающие теги (<name>Иван</name>).

– Поддерживает вложенность и самодокументируемость.

– Может включать инструкции по обработке и ссылки на схемы.

Устойчивость к повреждениям:

– Средняя. XML требует корректного завершения всех тегов и соблюдения иерархии. Однако многие парсеры (особенно SAX-типа) могут работать в режиме «восстановления после ошибки» (error recovery), пропуская повреждённые фрагменты.

– Если повреждён только один узел (например, не закрыт тег <note>), парсер может продолжить обработку других частей документа.

– Тем не менее, повреждение корневого элемента или пролога (<?xml ...?>) часто делает файл полностью непригодным.

Пример устойчивости. При потере одной записи в большом XML-файле с журналом событий система может извлечь остальные записи, если используется потоковый парсер.

Excel (.xlsx). Формат .xlsx (начиная с Microsoft Office 2007) основан на стандарте Office Open XML (ECMA-376). Файл представляет собой ZIP-архив, содержащий множество XML-файлов, описывающих листы, стили, связи и метаданные.

Особенности структуры:

– Модульная архитектура: каждый лист хранится отдельно (/xl/worksheets/sheet1.xml).

– Поддержка формул, форматирования, изображений и макросов (в .xlsm).

– Встроенные механизмы контроля целостности (например, резервные копии в некоторых версиях Excel).

Устойчивость к повреждениям:

– Высокая по сравнению с текстовыми форматами. Благодаря архивной структуре, повреждение одного компонента (например, одного листа) не всегда делает недоступным весь документ.

– Современные программы (Excel, LibreOffice, библиотеки Python openpyxl, pandas) могут восстанавливать часть данных даже из частично повреждённых файлов.

– Однако если повреждён корневой файл архива ([Content_Types].xml) или архив не может быть распакован, восстановление становится невозможным.

Пример устойчивости. Если при передаче повреждён XML-файл с третьим листом, Excel может открыть первые два листа без ошибок, предупредив пользователя о проблеме.

Анализ показал, что устойчивость форматов данных к повреждениям варьируется в широких пределах. CSV и JSON обладают низкой надёжностью из-за отсутствия механизмов локализации ошибок и строгой зависимости от целостности всего документа. XML предлагает средний уровень устойчивости при использовании потоковых парсеров. Наиболее надёжным в условиях частичной потери данных оказывается формат Excel (.xlsx) благодаря своей модульной архитектуре и встроенной избыточности.

При проектировании систем межсистемного взаимодействия рекомендуется учитывать не только удобство обработки, но и потенциальные риски повреждения данных. В критически важных сценариях следует применять форматы с поддержкой частичного чтения, использовать контрольные суммы и предусматривать механизмы восстановления.

Заключение. Устойчивость формата данных к повреждениям – важный, но часто недооцениваемый фактор при проектировании систем обмена информацией. Текстовые форматы, такие как CSV и JSON, просты в использовании, но крайне уязвимы к синтаксическим ошибкам. В то же время структурированные и модульные форматы, включая XML и Excel, демонстрируют большую гибкость при частичной потере данных. При разработке надёжных систем следует выбирать форматы с поддержкой частичного чтения, использовать механизмы валидации и предусматривать резервные пути восстановления информации.

ЛИТЕРАТУРА

1. McKinney, W. Python for Data Analysis. – O’Reilly Media, 2022.
2. Bray, T. (Ed.). The JavaScript Object Notation (JSON) Data Interchange Format. – RFC 8259, 2017.
3. Microsoft. Excel Specifications and Limits. – Документация Microsoft, 2023.
4. PostgreSQL Global Development Group. PostgreSQL Documentation. – https://www.postgresql.org/docs/