# Реферат на тему:

# SGML. SGML - структури. DTD

# SGML

SGML - Міжнародний Стандарт Всесвітньої Організації Стандартів - ISO 8879-1986: Information processing--Text and office systems--Standard Generalized Markup Language (SGML). SGML - міжнародний стандарт для визначення системно-незалежного методу маркапа. Для представлення тексту в електронному виді. Незважаючи на всю складність і глобальність поставленої мети, SGML є вкрай простим по ідеї стандартом. Ця ідея може бути представлена так:

Створити чітку і формальну систему, що визначає єдиний, головний метод для виконання дій; і, у той же час, залишити можливість для будь-яких інших методів.

Перш ніж перейти до формального опису стандарту, буде представлений неформальний опис, що фактично є перекладом стандарту.

## Неформальне введення в SGML

SGML - стандарт, у формальному змісті цього слова, - створений і погоджений міжнародним науковим співтовариством під егідою Міжнародної Організації Стандартизації (ISO). Його формальна, повна назва - ISO 8879 Information processing - Text and office systems - Standard Generalized Markup Language (SGML), перше видання було опубліковано 15.10.1986р., перше виправлення було опубліковано 01.07.1988р. Він доступний у найближчому відділенні організації стандартизації (яке зажадає за нього тонни грошей).

Визначити неформально, чим насправді є SGML, досить важко. Це мова, що може бути використана, щоб побудувати інфраструктуру для обміну і довгострокового збереження інформації. Використовуючи аналогію, її можна визначити як "SGML і мистецтво збереження й обробки інформації - Розуміння значення Інформації". Таким чином, це спосіб життя після того, як ми зрозуміли, що інформацію, що ми створюємо, живе своїм власним життям і може вмерти, якщо ми не доглядаємо за нею і не годуємо її правильним образом. У древні часи для знищення інформації необхідно було спалити одну з деяких значних бібліотек. Мало того, тебе б ще і внесли в історію за це. Сьогодні досить усього лише установити нову версію якого-небудь програмного продукту, перемінити принтер, скористатися патентованим методом для компресії даних, і т.п., щоб знищити в багато разів більше інформації. А підручники по історії навіть не намагаються відбивати той факт, що попереднє покоління технічних засобів було останніми, хто бачив ці дані.

SGML дотепер розглядається як формат для представлення інформації для публікації тексту друкованих документів, або як засіб представлення тих же документів на CD-ROM і т.д. Публікація – була початкова ціль стандарту, однак незабаром стало ясним, що він має набагато великий потенціал (у зв'язку з чим навіть народилася нова розшифровка акронима SGML - Sounds Great, Maybe Later (Звучить чудово, може бути, пізніше)).

За межами публікаційної індустрія SGML розглядається як можливість врятувати інформацію, що генерує людство і записує в тлінних, закритих, не документованих форматах. Наприклад, за час, що потрібно для створення словника якої-небудь мови, комп'ютерна індустрія переживе принаймні дві значні революції. Індустрія, де "три секунди - великий проміжок часу", допомагає будувати й обслуговувати об'єкти, час життя яких - кілька мільярдів секунд (3 мільярди секунд - ледве більше 95 років). Сюди входять підвищення цін на енергоносії, міста, закони, "культурна спадщина", стандарти.

Однак останні кілька сотень мільйонів секунд має значення тільки конкретний продукт, "сумісність", і використання комп'ютерів для імітування друкованого й іншого видів інформації. Цей етап пройде, і люди, нічого не здатні поробити зі своєю старою інформацією, зрозуміють, що ті божевільні провісники 80-х і 90-х були праві.

Багато людей цілком щасливі виконувати ту саму роботу знову і знову - роботу, що інші робили раніш, а ще інші будуть робити після них - навіть тоді, коли немає ніякої зовнішньої необхідності в цьому. Величезні зграї менеджерів щасливі втрачати стільки ж доларів, скільки губиться секунд при такій схемі. Цілі нації побудовані на працевлаштуванні людей, що тасують папери туди сюди. Безумовно, один стандарт не може зупинити подібну розтрату. Але він може зробити життя легшим тим, хто відмовляється брати участь у цьому величезному псуванні ресурсів.

SGML - це ідея, філософія, мова для вираження глобальних ідей для сьогодення і майбутнього. А якщо він має ці якості, він обов'язково буде використовуватися і тими, хто має набагато більш обмежене бачення. І тут функцію коміксів, мультиків і MTV виконує HTML. Як і багато читачів коміксів, більшість фанів HTML не мають уяви, відкіля він узявся. Однак деякі бачать зв'язок і задумуються над більш глобальними схемами.

На жаль, у даний момент немає яких-небудь дійсно гарних "підручників" по SGML. Ніде прочитати про головні принципи, способи реалізації їх і принципам побудови закінченої системи і способам забезпечення довговічного використання інформації. Однак багато приватних питань освітлені в численних "вступних" і "практичних" описах.

Якщо ви шукаєте підручник, придатний до використання в умовах чи інституту, чи університету, буде виданий подібний наприкінці цього чи початку наступного року. Я упевнений, що SGML буде залишатися не при справах, поки його основні ідеї не будуть зрозумілі всіма програмістами і дослідниками в області інформатики. Однак відсутність інформації, книг і програмних засобів в академічних установах - саме там, де зародилися багато сучасні реально використовуваних засобів, такі, як C, Unix, Internet, WWW, і т.д. - є однією з основних причин гальмування SGML.

## Введення в SGML

Дозволимо собі ще раз повторити, що SGML є міжнародним стандартом для представлення розміченого електронного тексту. Більш точно, SGML є *метамовою*. Це значить, що він призначений для опису мови, у даному випадку, мови маркапа тексту. Перш ніж йти далі, дамо більш розширені і формальні описи основних понять.

Історично, слово *маркап* (розмітка) використовувалося для опису спеціальних позначок, що призначалися для чи машиніста складача і що описують, як повинний бути видрукуваний чи набраний конкретний фрагмент тексту. Приклади включають хвилясте підкреслення для жирного шрифту, спеціальні знаки на початку абзацу, що повинний бути видрукуваний конкретним шрифтом і т.д. У міру того, як форматування і печатка тексту стала автоматизованої, значення терміна маркап було розширено для позначення всіх типів спеціальних *маркап-кодів*, що вставляються в електронний текст для керування процесом чи форматування печатки, чи іншої обробки.

Узагальнюючи цей зміст, ми визначаємо *маркап*, чи *розмітку*, як будь-який спосіб вказівки явного способу інтерпретації тексту. На тривіальному рівні, весь друкований текст розмічений у цьому змісті: пунктуація, використання заголовних букв, розташування літер на сторінці, навіть пробіли між словами можуть вважатися як деякий вид маркапа, призначення якого полягає в тому, щоб допомогти людині визначити де закінчується одне слово і починається інше, чи як інтерпретувати більш глобальні структури - такі, як заголовки, підлеглий пропозиції в головній пропозиції, уся головна пропозиція, абзаци і т.д. Маркап тексту в принципі, також як і стенографія, - процес вказівки явно того, яким чином повинно інтерпретуватися зміст тексту.

Під мовою маркапа розуміється сукупність угод про маркап, використовуваних для розмітки тексту. Мова маркапа повинна визначати:

* Синтаксис маркапа:
* який маркап є припустимим
* який маркап необхідний
* як маркап відрізняється від тексту
* Семантика маркапа:
* що маркап означає.

SGML дозволяє визначити три перші, синтаксичні обмеження на маркап. Для визначення семантики використовується додатковий стандарт, наприклад HTML, TEI, CES чи іншої.

Схема маркапа, що вводиться SGML (далі просто SGML) має три основних відмінності від "класичних" схем:

* Орієнтація на дескриптивний маркап,
* Об‘єктно-оріентована модель,
* Незалежність від конкретного фізичного представлення тексту.

Ці три особливості описуються більш докладно далі

### Орієнтація на дескриптивний маркап

SGML по визначенню припускає використання дескриптивної схеми маркапа. Дані, необхідні для якої-небудь конкретної обробки документа (наприклад форматування), чітко відокремлюються від дескриптивного маркапа, що міститься в документі. Звичайно вони зібрані за межами документа у виді відповідних алгоритмів і програм. Існують стандарти, що визначають їхній (DSSSL, HyTime).

При використанні дескриптивного маркапа один і той же документ, як вже відзначалося, може оброблятися різними способами за допомогою різних програм, кожна з який приділяє увагу тим частинам документа, що є важливими для даного методу обробки. SGML, як стандарт, підтримує і робить реально реалізованим такий принцип роботи. Наприклад, програма аналізу змісту тексту (приміром, система автоматичного анотування) може цілком ігнорувати виноски і примітки, що зустрічаються в тексті, у той час як форматуюча програма може витягати їх з тексту і збирати всі разом, щоб потім вивести наприкінці розділу. Різні методи обробки можуть застосовуватися для тих самих частин документа. Одна програма може робити вибірку з документа всіх особистих імен і назв місць і створювати по них базу даних, у той час як інша, для тих же елементів документа, виконує їхнє форматування таким чином, щоб вони виділялися з навколишнього тексту.

### Об‘ектно-орієнтовна модель

SGML вводить поняття *класу*, чи *типу*, *документа* і спосіб його завдання й опису - Document Type Definition (DTD). Документи вважаються визначеного *типу*, точно також, як і будь-які інші об'єкти, оброблювані комп'ютером. *Тип* документа формально визначається його складовими частинами і їхньою структурою. Звіт, наприклад, визначається як назва і, можливо, автор, за яких випливає реферат і послідовність з одного чи більш абзаців. Текст, що не має назви, відповідно до цього формального визначення, формально не є звітом, також як і послідовність абзаців, з яких випливає реферат, незважаючи на подібність на звіт з погляду читача-людини.

Якщо тип документа відомий, спеціальна програма (*парсер*) може бути використана для перевірки того, що вміст документа відповідає його типу - всі його необхідні частини присутні і випливають у потрібному порядку. Більш важливо, що різні документи того самого типу можуть оброблятися однаковим способом. Можуть бути написані програми, що використовують інформацію, задану у визначеному типу документа і здатні завдяки цьому робити більш розумні і значимі дії.

### Незалежність від низькорівневих представлення даних

Базовою метою, що ставилася при розробці SGML, була задача забезпечення перенесення документів між різними програмними й апаратними платформами без втрати інформації. Завдяки попереднім двом особливостям ця вимога досягається на абстрактному рівні. Розглянута тут можливість забезпечує цю вимогу на фізичному рівні, за допомогою визначення значення послідовностей байтів, з яких складається документ. SGML надає механізм загального призначення для заміни рядків, тобто простий машинно-незалежний спосіб для вказівки того, що конкретна послідовність символів у документі повинна бути замінена на іншу під час обробки документа. Одне з застосувань цієї можливості - забезпечення однаковості термінології й інших параметрів документів; інше, більш важливе - забезпечення можливості переносу тексту, що містить нестандартні і т.п. символи між різними платформами, що мають різні набори символів, можливості по їхньому відображенню і т.д. Це досягається за допомогою заміни нестандартних символів на рядки, що задають їхні коди. Послідовності символів, визначені для описаної вище заміни називаються entities ("поняття").

# SGML-структури

## Текстуальна структура

Текст не є недиференційованою послідовністю слів, а тим більше байт. Для різних цілей він може бути розбитий на безліч різних елементів різних типів і розмірів. Прозаїчний текст може бути розбитий на розділи, глави, абзаци і пропозиції. Віршований - на cantos, строфи і рядки. Надрукований текст, залишаючись чи прозою віршем і не залежно від цього, розбивається на томи, брошури і сторінки.

Структурні елементи подібного типу найчастіше використовуються для посилання на конкретні місця в тексті ("третя пропозиція другого параграфа глави 10", "песня 10, рядок 1234", "сторінка 412" і т.д.), хоча можуть використовуватися і по прямому призначенню - як позначення деякої структурно- і завершеною по змісту одиниці тексту, наприклад для аналітичних цілей (" чи відрізняється середня довжина пропозиції в розділі 2 у порівнянні з розділом 5?", "скільки абзаців розділяють кожне зустрінуте слово `природа' ?", "скільки сторінок у документі?"). Інші структурні елементи є чисто аналітичними, у тім змісті, що вони характеризують деяку частину тексту. У драматичному тексті мова якого-небудь персонажа може вважатися елементом одного типу, а вказівки для чи сцени опису дій - як інший тип. Подібний аналіз менш корисний для посилання на конкретне місце в тексті ("93-я мова Гораціо в акті 2"), чим для проведення порівнянь лексики одного персонажа в порівнянні з іншим і т.п.

У прозаїчним тексті подібним же чином можна виділяти пряму і непряму мову, стильові особливості тексту (оповідання, полеміка, коментар, аргументація і т.д.), цитати різних авторів і так далі. А для деяких типів аналізу (найбільше для критичного розбору), фізичного представлення друкованого чи рукописного тексту також може мати значення: парадоксально, але може виявитися необхідним використовувати дескриптивний маркап для опису процедурного.

Описані текстуальні структури перетинаються і накладаються один на одного, утворити складні і непередбачені комбінації. Таким чином, вимагаються як способи забезпечення можливості роботи з декількома ієрархіями, так і методи забезпечення надійності такого маркапа і виключення небажаного взаємного впливу. SGML забезпечує ці можливості на різних рівнях - як на найпростішому, коли обов'язок стежити за несуперечністю перетинання структур лягати на користувача, так і на самому верхньому рівні, забезпечуючи можливість застосування різних схем маркапа для того самого документа, тобто, формально, допускають що документ може включати декілька типів документів.

## SGML-структури

У цьому параграфі описуються чітка і проста ідея схеми маркапа, використовувана в SGML для ідентифікації структурних елементів у тексті. Також описуються методи, що надає SGML для визначення правил, що описують, які комбінації цих елементів є припустимими для документа.

### Елементи

*Елемент* - термін, використовуваний у SGML для позначення якої-небудь структурної одиниці тексту. Різним типам елементів даються різні імена, але SGML не надає ніякого способу визначення змісту і значення конкретного типу елемента, крім завдання його *відносин* з іншими елементами. Таким чином, єдине, що можна сказати про елемент (наприклад) <blort> - це те, чи можуть екземпляри цього елемента зустрічатися усередині елементів типу <farble>, і те, чи може він сам бути декомпозован на елементи типу <blortette>. Варто підкреслити, що SGML ніяким образом не визначає семантику елементів; по стандарті вона залежить від додатка.Створення SGML-сумісного набору тегів і завдання їхнього змісту саме і є тією задачею, що вирішують додаткові стандарти, такі як HTML і XML, XSL; саме вони задають конкретну схему маркапа й у частині синтаксису роблять це в термінах SGML. При визначенні елемента йому необхідно привласнити ім'я, що відбиває його призначення; таке ім'я називається *generic identifier* (*GI*).

У тексті (*document instance*, *екземплярі документа*) кожен елемент повинний бути явно позначений деяким чином. Стандарт надає безліч можливостей зробити це, найбільше часто використовуваним є вставка тега на початку елемента (start-tag) і іншого наприкінці (end-tag). Старт- і енд- теги використовуються щоб узяти в дужки текст, що входить в елемент, точно так само, як різні дужки і лапки в звичайній пунктуації. Наприклад, цитата може бути позначена в такий спосіб:

... Rosalind's remarks <quote>This is the silliest stuff

that ere I heard of!</quote> clearly indicate ...

Як видно з цього приклада, старт-тег має форму <gi>, де кутові дужки задають початок і кінець самого старт-тега, а gi - ідентифікатор тега. Енд-тег відрізняється від старт-тега тільки тим, що перед ідентифікатором ставиться коса риса: </gi>.

### Моделі змісту

Елемент може бути порожній, тобто взагалі не мати вмісту, чи містити простий текст. Однак звичайно елементи одного типу будуть вкладені (будуть цілком міститися) усередині елементів іншого типу і т.д. Таким чином, текст має деревоподібну структуру, у якій елементи рекурсивно входять один в іншій:

Element::=EMPTY | Char+ | Element+

Щоб проілюструвати це, розглянемо вкрай просту структурну модель: припустимо, що в межах антології ми хочемо ідентифікувати тільки вірші, а саме, їхні назви і строфи і рядки, з яких вони складаються. У термінах SGML, тип нашого документа - антологія і вона складається з послідовності віршів. Кожен стих являє собою заголовок, за яким випливає одна чи більш строфа. Строфа складається з послідовності рядків. Цілком розмічений, текст буде виглядати в такий спосіб:

<anthology>

<poem><title>The SICK ROSE</title>

<stanza>

<line>O Rose thou art sick.</line>

<line>The invisible worm,</line>

<line>That flies in the night</line>

<line>In the howling storm:</line>

</stanza>

<stanza>

<line>Has found out thy bed</line>

<line>Of crimson joy:</line>

<line>And his dark secret love</line>

<line>Does thy life destroy.</line>

</stanza>

</poem>

<!-- more poems go here -->

</anthology>

Пробіли і розриви рядків додані винятково для зручності читання, вони не мають особливого значення для SGML. Рядок

<!-- more poems go here -->

є SGML-коментарем.

У цьому прикладі не робиться ніяких припущень із приводу обмежень накладаються на елементи, наприклад такі, чи може назва розташовуватися не перед першою строфою чи вірша про тім, чи можуть існувати рядка, що не входять у строфу. Саме тому маркап такий багатослівний: у ньому початок і кінець кожного елемента явно позначені, оскільки не задано правил про тім, де і коли може розташовуватися деякий елемент. Однак на практиці можна сформулювати правила, що дозволяють скоротити необхідний обсяг маркапа. Наприклад, у нашій спрощеній моделі вірша, ми можемо задати наступні обмеження:

1. У межах антології містяться вірші і нічого більш.
2. Вірш завжди має єдиний елемент-заголовок, що передує першій строфі і не утримуючих інших елементів.
3. Крім заголовка, вірш містить тільки строфи.
4. Строфи складаються тільки з рядків і будь-який рядок міститься в строфі.
5. За строфою може випливати тільки інша чи строфа кінець вірша.
6. За рядком випливає тільки інший чи рядок початок нової строфи

З цих обмежень можна вивести деякі наслідки:

* З обмежень 5 і 6 випливає, що немає необхідності явно позначати кінці строф і рядків.
* З обмеження 2 випливає, що не потрібно позначати кінець назви - він задається неявно початком першої строфи.
* Аналогічно, з обмежень 1 і 3 випливає, що не обов'язково позначати кінець вірша: оскільки вірші не можуть міститися усередині віршів, але повинні міститися усередині антологій, кінець вірша мається на увазі початком іншого або завершенням антології.

Застосовуючи ці спрощення, ми може розмітити той же вірш у такий спосіб:

<anthology>

<poem><title>The SICK ROSE

<stanza>

<line>O Rose thou art sick.

<line>The invisible worm,

<line>That flies in the night

<line>In the howling storm:

<stanza>

<line>Has found out thy bed

<line>Of crimson joy:

<line>And his dark secret love

<line>Does thy life destroy.

<!-- more poems go here -->

</anthology>

Можливість завдання правил, що визначають, які елементи можуть бути вкладені в інші елементи - одна з найважливіших характеристик SGML. Фактично, SGML дозволяє задати граматику мови маркапа.

Перш ніж розглянути ці правила більш докладно, приведемо конкретні приклади тих можливих дій, які можна виконати з розробленим нами документом-антологією. Проста програма, що індексує, може витягати деякі текстові елементи, наприклад, назви, щоб скласти зміст антології; чи створити список усіх слів (wordlist), використовуваних у віршах. Проста форматуюча програма може вставляти порожні рядки між строфами, можливо нумеруючи останні. Різні частини віршів можуть бути представлені різним форматуванням, наприклад, заголовки - жирним шрифтом. Більш складна програма, що аналізує, може простежити використання розділових знаків стосовно розділів на строфи і метричний малюнок вірша. І звичайно ж, вищенаведений текст може бути перенесений з одного комп'ютера на іншій і оброблений будь-якою чи людиною будь-якою програмою, що розуміють, який зміст мають його теги, без необхідності в різних трансформаціях і перетвореннях, необхідних при переносі файлів різних proprietary текстових процесорів.

# DTD

## Визначення типу і структури документа: DTD

Формальний запис правил, таких, що були неформально описані вище, є першою стадією визначення структури SGML документа, а саме стадією розробки специфікації типу документа (Document Type Definition, DTD). Розробляючи DTD, дизайнер може задати тільки загальну ідею документа, не описуючі конкретностей і обмежень, або навпаки, явно задати їхній усі - у залежності від призначення типу документа; як уже відзначалося, така воля була однієї з головних цілей розробки SGML. Баланс між двома цими формами досягається при компромісі між легкістю проходження простим правилам і складністю обробки реальних текстів. Один зі складних випадків - обробка вже існуючих текстів. Розроблювач найчастіше має тільки загальне поняття про призначення оригінальних текстів і тільки базові ідеї про їхню структуру. У таких випадках найчастіше задаються самі загальні текстові елементи і найбільш вільні правила з взаємної композиції. З іншого боку, при розробки нових текстів, призначення яких точно відомо, розумно задати безліч обмежень, домагаючись тим гарантованої правильності і без проблемною подальшої обробки. Прикладом, де важливий саме такий підхід, може бути підготовка інформації для різних баз даних і знань. Навіть у випадку існуючого тексту може бути корисно задати обмежуючі правила які описують деяку гіпотезу про структуру документа - це принаймні допоможе перевірити адекватність цієї гіпотези. Важливо розуміти, що будь-яка специфікація типу документа - це інтерпретація тексту тільки з якоїсь однієї сторони. Не може бути єдиної DTD, придатної на усі випадки життя, що є "абсолютною істиною", крім, можливо, DTD, тавтологічно визначальної, що "текст є текст". Однак звичайно від такий DTD мало реальної користі. Таким чином, для різних цілей можуть знадобитися різні DTD, хоча, безумовно, невелика їхня кількість буде покривати найбільш важливі і часто використовувані області.

В даний час SGML найбільше широко застосовується в областях, де схожість структури документів - найбільш важливий критерій. Наприклад, при розробці технічної документації, має велику важливість, щоб розділи і підрозділи були правильно вкладені, перехресні посилання були коректно дозволені і т.д. У таких ситуаціях документ розглядається як напівфабрикат, що перевіряється на відповідність визначеному набору правил. Однак як відзначалося раніше, навіть для більш вільних текстів має сенс явно задати різні правила, що стосуються змісти. Цим самим оброблювач позбавить себе від тривіальних помилок, що можуть проявитися тільки при наступній обробці і полегшить собі сам процес розмітки.

### Проста DTD

DTD у SGML записується як набір операторів декларації, використовуючи простий синтаксис, визначений у стандарті. Для нашої простої моделі антології, ці декларації будуть виглядати в такий спосіб:

<!ELEMENT anthology - - (poem+)>

<!ELEMENT poem - O (title?, stanza+)>

<!ELEMENT title - O (#PCDATA) >

<!ELEMENT stanza - O (line+) >

<!ELEMENT line O O (#PCDATA) >

Ці п'ять рядків приклад формальної декларації SGML-елементів. Декларація, також як і елемент, полягає в кутові дужки; перший символ після відкриваючої дужки повинний бути знак оклику, за яким одне з невеликого набору ключових слів SGML, що визначає, який об'єкт декларується. Усі п'ять декларацій вище того самого типу: вони починаються з ключового слова ELEMENT, що показує, що визначається елемент. Кожна декларація складається з трьох частин: ім'я тега (GI) чи група імен, два символи, які описують правила мінімізації, і модель змісту. Кожна їхньої цієї частин більш докладно описується далі. Компоненти декларації розділяються символом space, яким є послідовність пробілів, табуляцій і кінців рядків.

#### Generic Identifier

Перша частина декларації задає GI елемента, наприклад title, poem і т.д. Можливо одночасно визначити кілька елементів; ця можливість описується нижче.

#### Правила мінімізації

Друга частина декларації визначає те, що називається *правилами мінімізації* елемента. Правила мінімізації визначають, чи є необхідним явна вказівка старт- і енд-тегів при позначці елемента і записуються за допомогою пари символів, що відносяться до старт- і енд-тегу відповідно і розділених space символом. У будь-якому випадку, повинний бути заданий символ `-' (мінус), якщо тег обов'язковий, і `O' (omissible), якщо тег може бути опущений.

Таким чином, кожен елемент, крім <line> повинний мати старт-тег. Тільки елемент <anthology> повинний мати енд-тег.

#### Модель змісту

Третя частина декларації, укладена в дужки, називається *моделлю вмісту* (content model) елемента, оскільки вона описує елементи, що можуть міститися в даному. Вміст описується в термінах інших елементах і спеціальних зарезервованих словах. Існує кілька таких слів, з яких найбільше часто зустрічається #PCDATA, що є присутнім і в прикладі. Це абревіатура для послідовності символів, і вона означає, що елемент, описаний з її допомогою, може містити будь-як припустимі символи, тобто власне вміст тексту. Якщо розглядати специфікацію структури як дерево відносини *елемент-вміст* (мал. 2), що має одну кореневу вершину (у нашому випадку, <anthology>) і проміжні вершини (<poem>, <stanza>, ...), то листами цього дерева майже у всіх випадках будуть #PCDATA.

*Рис. 2 Дерево відносини елемент-вміст*

У нашому випадку в такий спосіб задані <title> і <line>. Оскільки їхня модель умісту містить тільки #PCDATA, те вони не можуть містити інших вкладених елементів.

#### Регулярні оператори декларації

При завданні моделі вмісту, а в деяких випадках і при вказівці імен тегов, що декларуються, використовуються *оператори*, подібні з операторами, використовуваними для завдання *регулярних виражень*. Далі описуються їхні функціональні групи.

##### 1. Оператори повторень

Декларація елемента <stanza> у прикладі вище вказує, що строфа складається з однієї чи більш рядка. Оператор повторення `+' (плюс) указує цей факт. Синтаксис SGML містить три операторів повторення:

Таблиця 7 Оператори повторення SGML

|  |  |
| --- | --- |
| Символ оператора повторень | Зміст оператора |
| ? | Попередній елемент опціонален, тобто може повторюватися 0 чи 1 раз |
| \* | Попередній елемент може повторюватися 0 і більш раз |
| + | Попередній елемент може повторюватися 1 і більш раз |

Як приклад, декларація для <poem> задає, що даний елемент не може мати більш одного <title>, але може не мати взагалі; і те, що він повинний містити принаймні один елемент <stanza> і може містити трохи.

##### 2. Оператори угруповання

Модель умісту (title?, stanza+) містить більш одного елемента і тому додатково необхідно вказати порядок їхньої появи. Цей порядок визначається оператором угруповання `,' (кома), що вказується між компонентами. Існує три можливих оператора угруповання:

Таблиця 8 Оператори групування SGML

|  |  |
| --- | --- |
| Символ оператора угруповання | Зміст оператора |
| , | Компоненти повинні випливати в порядку, заданому в моделі вмісту |
| & | Обидва компоненти повинні бути присутнім, але можуть випливати в будь-якому порядку |
| | | Тільки один з компонентів може з'явиться усередині описуваного елемента |