**ПР№13 – Построение информационных систем**

**Ответить на вопрос исходя из текста + дополнительных источников, пересказ**

**Информационная система: как ее сделать?**

Вопросительный знак стоит не даром. Это действительно проблема, которую приходится решать в большинстве современных предприятий, независимо от того, какого рода бизнесом они занимаются. Термин "информационная система" относится к классу программных продуктов, облегчающих, или "автоматизирующих" ведение бизнеса. Система называется "информационной", если она поддерживает информационную поддержку бизнеса (любой деловой человек знает, сколько сил и средств уходит на поддержание необходимой информации). Соответствующая программа называется "системой", если она (в некоторых случаях последовательно, а иногда параллельно) выполняет более одной функции (одним из распространенных примеров являются информационные системы, поддерживающие ведение складского хозяйства: они отслеживают поступление товаров на склад, отпуск товаров покупателю, а также контролируют наличие на складе необходимого количества каждого продукта).

В большинстве случаев для создания собственной информационной системы невозможно обойтись без использования баз данных. Чем отличается "база данных" от какого бы то ни было склада данных, поддерживаемого в файлах операционной системы? Основное отличие состоит в том, что набором данных, входящих в состав базы данных управляет специальная системная программа, обычно называемая "системой управления базами данных (СУБД)", которая обладает знаниями по поводу связи между разнородными данными. Например, в случае складской системы, СУБД, управляющая соответствующей базой данных, должна знать, что для всех единиц любого товара, перечисленных в общей складской ведомости, должно быть проставлено правильное число в документе, регулирующем поступление товаров на склад.

Такого типа свойства называются целостностью базы данных. При создании базы данных информационной системы разработчик сообщает СУБД, какого рода ограничения целостности система должна поддерживать в базе данных, а далее ответственность берет на себя СУБД, без требования вмешательства прикладной программы. Обычно механизм обеспечения целостности баз данных интегрируется с механизмом управления транзакциями - последовательностью операций модификации базы данных, воспринимаемыми СУБД как одна атомарная операция.

Второй важной особенностью СУБД является обеспечение выполнения так называемых "незапланированных (ad hoc)" запросов к базе данных. Представим себе, что при проектировании информационной системы, предназначенной для автоматизации управления складом, было запланировано выполнение запросов о наличии на складе товаров, операций модификации данных при отпуске товаров со склада и их получении, а впоследствии понадобилась информация об общем объеме поставок от данного поставщика. При отсутствии СУБД понадобилась бы переделка информационной системы. Однако СУБД, обладая достаточными знаниями о предметной области (например, о структуре и смысле данных складской информационной системы), может обеспечить (и реально обеспечивает) универсальный язык запросов (обычно, язык SQL), позволяющий сформулировать произвольный запрос на выборку информации из соответствующей базы данных. Такой запрос может быть в любой момент подан с терминала (без участия информационной системы) или встроен в одну из прикладных программ, входящих в информационную систему.

Наконец, еще одной важной особенностью большинства современных СУБД является обеспечение так называемого "режима мультидоступа". Сегодня развитые компьютерные архитектуры обычно относятся к одной из двух категорий (или к их комбинации): информационно-вычислительный сервер (mainfraim) с более чем одним подключенным к нему терминалом или локальная или распределенная информационно-вычислительная сеть серверов и клиентских рабочих станций, обеспечивающая совместное использование ресурсов. В любом случае каждый из потенциальных пользователей может захотеть (и иметь на это право) в любой момент времени воспользоваться услугами информационной системы.

Соответственно, информационная система должна иметь возможность параллельно (или квази-параллельно - главное, чтобы у конечного пользователя не возникали ощущения чувствительной задержки ответа) выполнить операции, задаваемые несколькими пользователями одновременно (или очень близко во времени). При этом такое "параллельное" выполнение является корректным, т.е. результат получается таким, как если бы несколько параллельных транзакций выполнялось последовательно (в литературе по базам данных используется термин "сериализованное выполнение пользовательских транзакций"). Подавляющее большинство современных развитых СУБД поддерживает эту возможность, избавляя разработчиков информационных систем от необходимости заботиться об обеспечении режима мультидоступа. Базовые знания в области систем управления базами данных кажутся необходимыми для проектировщиков и разработчиков информационных систем (независимо от того, какая конкретная СУБД будет использоваться).

Итак, современные СУБД предназначены именно для разработчиков и последующих пользователей информационных систем. Однако достаточно ли этих базовых программных средств? Достаточно часто проблемой является проектирование и разработка информационной системы даже в том случае, если система опирается на какую-либо современную СУБД. Приходится решать задачи трех видов: проектирование и разработка логической структуры самой информационной системы как набора программ; проектирование лежащей в основе общего проекта информационной системы базы данных; проектирование и разработка интерфейсных подсистем, как тех, которые относятся к взаимодействиям информационной системы с конечным пользователем, так и тех, которые связывают прикладные программы с СУБД. Если проектируемая информационная система достаточно сложна (а это обычное явление), то решение этих задач "вручную", без привлечения программных инструментальных средств, как правило, превышает человеческие возможности.

К нашему с вами счастью о нас (по крайней мере, частично) позаботились софтверные компании, которые производят средства, служащие инструментом при проектировании и разработке информационных систем. Конечно, прежде всего это системы класса CASE (Computer Added Software Enginering), ориентированные на поддержку разработки информационных систем. Наиболее развитые CASE-системы позволяют автоматизировать процесс проектирования и разработки прикладной системы, поддерживая полную документацию (возможно, с разными версиями) обо всем этом процессе.

Может быть, наиболее важно то, что такие системы существенно помогают создавать схему базы данных, лежащей в основе проекта информационной системы. CASE-системы позволяют естественно (и достаточно просто) пройти путь от интуитивного представления структуры и поведения нужной предметной области до формализованного представления в терминах языка SQL. Такие возможности CASE-систем может оценить каждый, кому приходилось вручную проектировать схему достаточно сложной базы данных.

Другой класс программных средств (часто интегрированных с CASE-системами) составляют программные системы языков четвертого поколения (4GL). Это, может быть, чересчур громкое название означает лишь то, что такие, как правило, интерпретируемые языки предоставляют пользователю более или менее удобные средства для формирования интерфейса с конечным пользователем (например, в виде меню или форм), обеспечивают сравнительно простые возможности для взаимодействия с системой управления базами данных, а также предоставляют (обычно, достаточно примитивные) средства программирования. Основным достоинством языков четвертого поколения является то, что они обеспечивают возможность так называемого "быстрого прототипирования приложений (rapid prototyping)".

Это означает то, что при использовании 4GL можно действительно быстро соорудить работающий прототип будущей системы, обеспечивающий требуемый интерфейс с конечным пользователем и взаимодействующий с макетом базы данных (а возможно, и с реальной базой данных, если она к этому времени подготовлена). Приходится говорить про "прототип", поскольку большинство современных систем 4GL не обеспечивают той эффективности прикладных систем, какую дают обычные языки программирования (как теперь принято выражаться, 2GL или 3GL). Вместе с тем, стоит заметить, что уже существует масса реально используемых информационных систем, разработанных исключительно на том или ином 4GL. Кто знает, что будет завтра, но тенденция к увеличению использования 4GL очевидна.

Наконец, некоторое количество замечаний по поводу методологии проектирования и разработки информационных систем. Для профессиональных программистов постоянной проблемой являлся разрыв между данными и программой. При наличии сложно структурированной информации проектирование схемы базы данных является не менее сложной задачей, чем написание собственно прикладной системы. Очевидно, что всегда хуже иметь две сложные задачи, чем одна, даже если в совокупности ее сложность превосходит сложность каждой их первых задач в отдельности. Решение этой проблемы предлагает объектно-ориентированный подход.

Если говорить совсем коротко, суть этого подхода состоит в том, что проектируются не данные и программы в отдельности, а объекты, сочетающие в себе и данные, и программы, информационно и функционально характеризующие соответствующие сущности предметной области. Подход полезен как с методологической точки зрения (исчезают две разнородные характеристики предметной области - данные и программы объединяются в объекты), так и с точки зрения техники проектирования и разработки программных систем (вместо двух технически не связанных, но логически переплетенных веток образуется один надежный ствол).

Заметим, что в последнее время подавляющее большинство CASE-систем и 4GL если не ориентируется, то обращает внимание на объектно-ориентированный подход. Более того, стали появляться методики по объектно-ориентированному использованию средств автоматизированного проектирования и разработки информационных систем, которые (средства) исходно для этого не предназначались.

 Коротко отвечая на вопрос, заданный в заглавии, скажем, что сегодня создателю информационной системы, как минимум, нужно иметь базовые знания о современных СУБД, ориентироваться в мире инструментальных средств разработки программных систем и иметь представления об объектно-ориентированном подходе к проектированию и разработке программ. Конечно, при этом неплохо еще уметь программировать :-).