Дисциплина: Система управления базами данных

Преподаватель: Бектемесов Жоламан Мақтағалиұлы, PhD, и.о. доцента

Лекция 1. Введение в базы данных

База данных представляет собой набор взаимосвязанных данных. Например, *телефонный справочник* – это база данных имен, телефонных номеров и адресов всех людей, проживающих в определенной местности. Будучи, несомненно, широко и часто используемой базой данных, телефонный справочник имеет ряд **недостатков**:

• Поиск конкретного телефонного номера занимает много времени, особенно если в телефонном справочнике очень много записей.

• Телефонный справочник проиндексирован только по именам/фамилиям, поэтому для поиска абонента по определенному адресу такая база данных практически не используется.

• С момента публикации телефонного справочника актуальность представленной в нем информации постепенно снижается, поскольку люди уезжают и приезжают из этой местности, меняют свои телефонные номера или переезжают по другому адресу в той же местности.

Недостатками, присущими телефонным справочникам, обладает любая другая некомпьютеризированная система хранения данных, например, *регистратура поликлиники*, хранящая медицинские карты пациентов.

Из-за громоздкости бумажных баз данных одними из первых компьютерных приложений были разработаны ***системы баз данных*** (*database systems*) – средства компьютеризированного хранения и извлечения данных. Поскольку система БД хранит информацию в электронном виде, а не на бумаге, она может быстрее извлекать данные, индексировать их разными способами и поставлять своим пользователям наиболее актуальную информацию.

В первых системах БД информация хранилась на магнитных лентах. Количество лент во много раз превышало количество устройств считывания, поэтому техническим специалистам приходилось постоянно менять ленты, чтобы предоставить ту или иную запрашиваемую информацию. Хотя эти системы БД и были существенным усовершенствованием по сравнению с бумажными БД, им было еще далеко до возможностей современных технологий.

Нереляционные системы баз данных

Первые несколько десятилетий данные в компьютеризированных системах БД хранились и представлялись по-разному.

1) В ***иерархической системе баз данных*** (*hierarchical database system*) данные представляются в виде одной или нескольких древовидных структур.

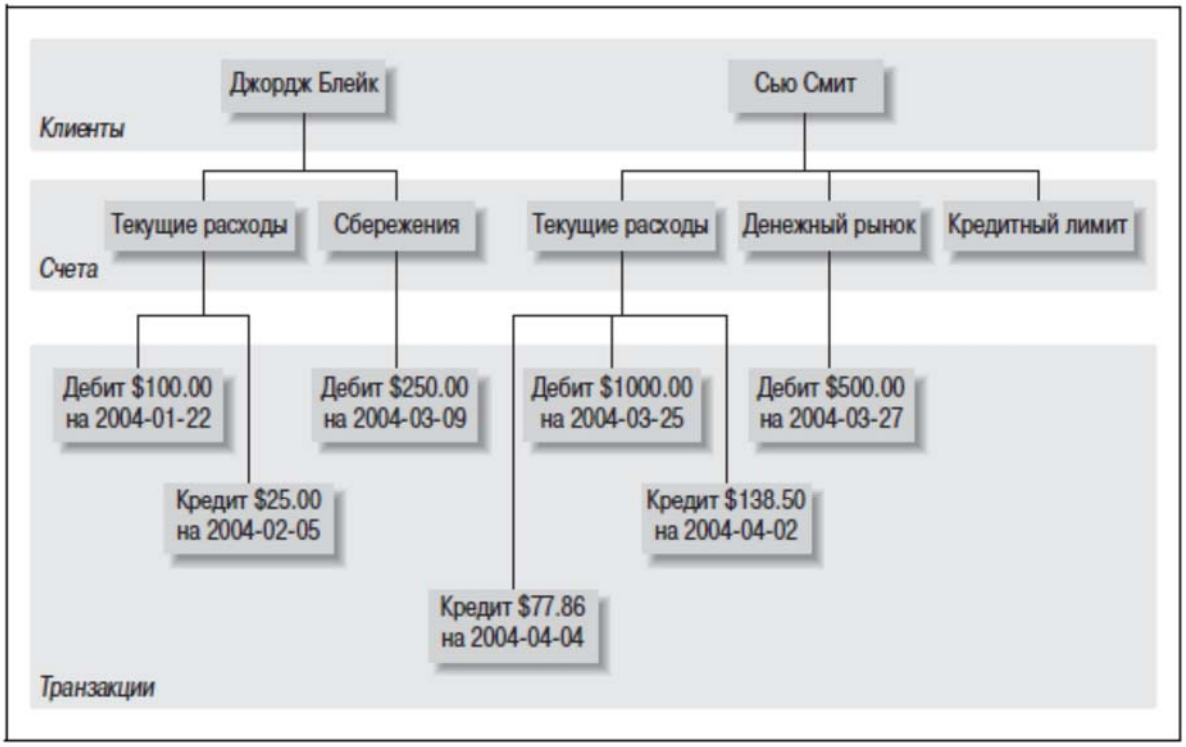


Рисунок 1 – Иерархическое представление информации по счетам

Иерархическая система базы данных предоставляет средства для нахождения дерева конкретного клиента и последующего обхода этого дерева в поисках нужных счетов и/или транзакций. У каждого узла дерева может быть ни одного или один родитель и ни одного, один или много дочерних узлов. Такую конфигурацию называют ***иерархией с одним родителем*** (*single-parent hierarchy*).

2) ***Сетевая база данных*** (*network database system*), представляет собой наборы записей и наборы связей, определяющих отношения между разными записями.

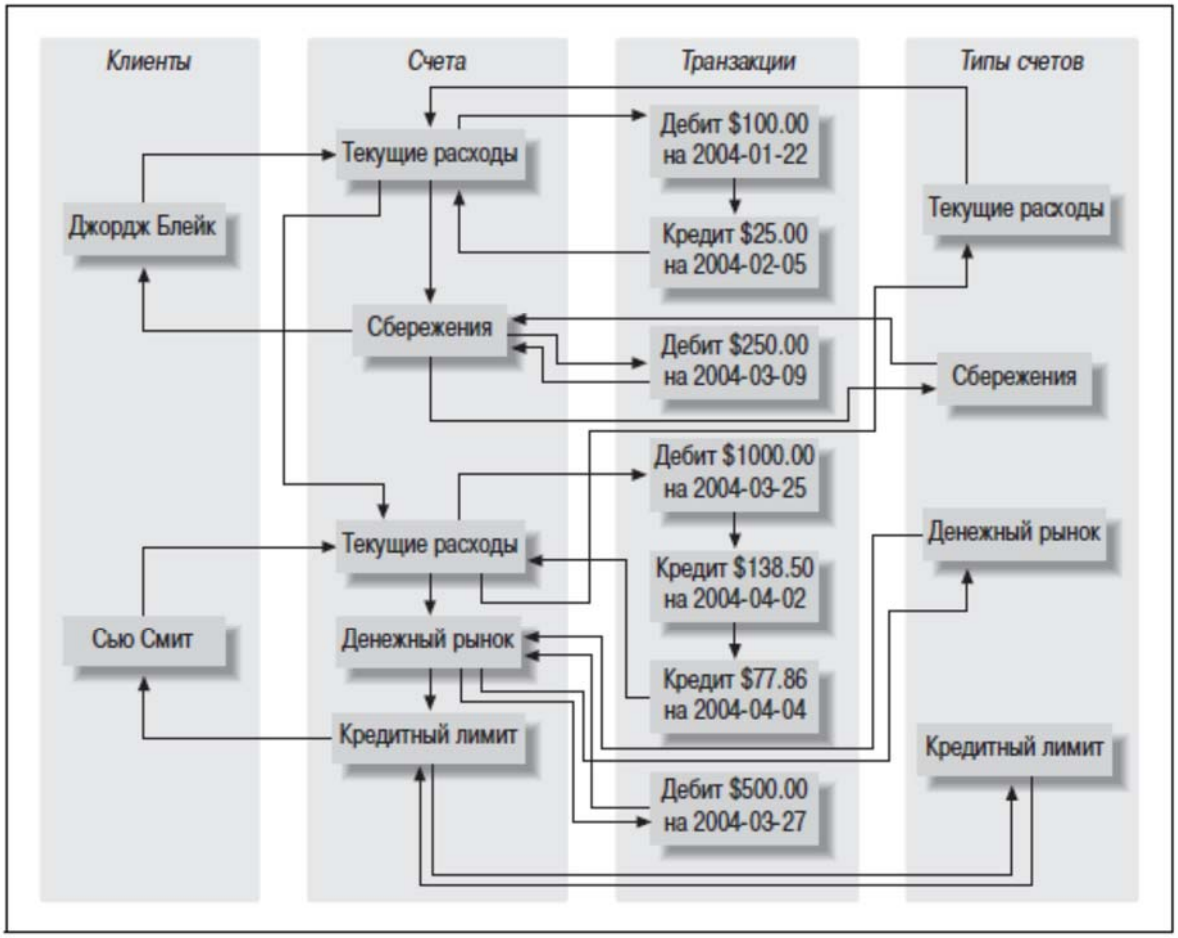


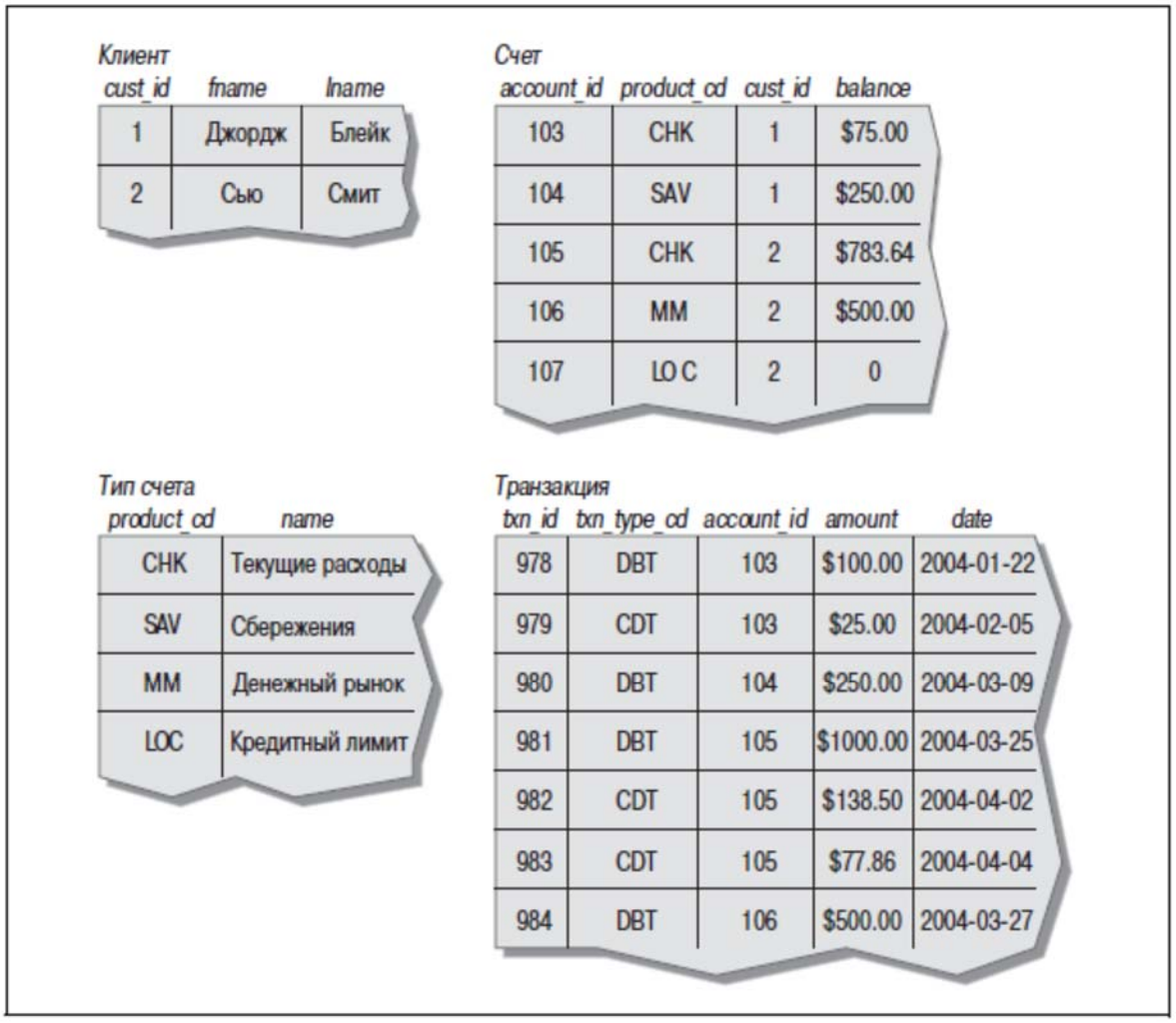
Рисунок 2 – Сетевое представление информации по счетам

Интересной особенностью сетевых баз данных является то, что доступ к определенным записям может быть осуществлен из нескольких мест (например, доступ к записям **Счета** может быть получен и через записи **Клиенты**, и через записи **Типы счетов**). Поэтому такую систему баз данных также называют ***иерархией с несколькими родителями*** (*multiparent hierarchy*).

Как иерархические, так и сетевые системы баз данных все еще используются, хотя преимущественно в мире мэйнфреймов. Кроме того, иерархические системы БД возродились в службах каталогов, таких как Active Directory компании Microsoft и Directory Server компании Netscape, а также с появлением XML (Extensible Markup Language).

Реляционная модель

В 1970 году сотрудник исследовательской лаборатории IBM доктор Е.Ф. Кодд опубликовал статью под названием «A Relational Model of Data for Large Shared Data Banks», в которой предложил представлять данные как наборы **таблиц**. Вместо указателей для навигации по взаимосвязанным сущностям он предложил использовать избыточные данные, связывающие записи разных таблиц.



Клиент

cust_id	fname	lname
1	Джордж	Блейк
2	Сью	Смит

Счет

account_id	product_cd	cust_id	balance
103	CHK	1	$75.00
104	SAV	1	$250.00
105	CHK	2	$783.64
106	MM	2	$500.00
107	LOC	2	0

Тип счета

product_cd	name
CHK	Текущие расходы
SAV	Сбережения
MM	Денежный рынок
LOC	Кредитный лимит

Транзакция

txn_id	txn_type_cd	account_id	amount	date
978	DBT	103	$100.00	2004-01-22
979	CDT	103	$25.00	2004-02-05
980	DBT	104	$250.00	2004-03-09
981	DBT	105	$1000.00	2004-03-25
982	CDT	105	$138.50	2004-04-02
983	CDT	105	$77.86	2004-04-04
984	DBT	106	$500.00	2004-03-27

Рисунок 3 – Реляционное представление информации по счетам

Таким образом, в ***реляционной базе данных*** сами данные и отношения между ними представляются в виде набора таблиц. Максимально возможное количество столбцов в таблице отличается для разных серверов, но обычно это достаточно большое число, и с ним нет проблем (**Microsoft SQL Server**, например, допускает до 1024 столбцов в таблице). Число строк в таблице – это больше вопрос физических возможностей (т.е. определяется доступным дисковым пространством), чем ограничений серверов БД.

Каждая таблица реляционной базы данных включает информацию, уникально идентифицирующую строку этой таблицы (***первичный ключ*** (*primary key*)), а также дополнительные данные, необходимые для полного описания сущности. Например, в таблице **Клиент** первичным ключом является столбец *cust_id*, по которому можно уникально идентифицировать каждого клиента банка. Никогда никаким двум клиентам не будет присвоен одинаковый идентификатор, и этой информации достаточно, чтобы обнаружить данные Джорджа Блейка или Сью Смит в таблице **Клиент**.

Первичный ключ – один или более столбцов, которые можно использовать как уникальный идентификатор для каждой строки таблицы. Строки в таблицах также называют ***записями***.

Некоторые из таблиц также содержат информацию, используемую для навигации к другой таблице. Например, в таблице **Счет** есть столбец *cust_id*, содержащий уникальный идентификатор клиента, открывшего счет, и столбец *product_cd*, содержащий уникальный идентификатор типа счета, которому будет соответствовать счет. Эти столбцы называют ***внешними ключами*** (*foreign keys*).

Внешний ключ – один или более столбцов, которые можно совместно использовать для идентификации одной строки другой таблицы.

Может показаться излишним хранить одни и те же данные в нескольких местах, но реляционная модель использует избыточность данных очень четко. Например, если таблица **Счет** включает столбец для уникального идентификатора клиента, открывшего счет, это правильно, а если включены также его имя и фамилия, то это неправильно. Например, если клиент изменяет имя, нужна уверенность, что его имя хранится только в одном месте базы данных. В противном случае данные могут быть изменены в одном месте, но не изменены в другом, что приведет к их недостоверности. Правильное решение – хранить эту информацию в таблице **Клиент**. В другие таблицы следует включить только *cust_id*. Также неправильно располагать в одном столбце несколько элементов данных, например, в столбец *name* помещать имя и фамилию человека или в столбце *address* указывать улицу, город, страну и почтовый индекс.

Процесс улучшения структуры базы данных с целью обеспечения хранения всех независимых элементов данных только в одном месте (за исключением внешних ключей) называют ***нормализацией*** (*normalization*).

Литература:

1. Алан Бьюли. Изучаем SQL: пер. с англ. – СПб-М.: Символ, O’Reilly, 2007. – 310 с.