EPAM Systems, RD Dep.

Конспект и раздаточный материал

CTECH.DB.01 Основы баз данных

REVISION HISTORY					
Ver.	**Description of Change**	**Author**	**Date**	**Approved**	
				Name	**Effective Date**
<1.0>	Первая версия	Святослав Куликов	<16.03.2012>		

Содержание

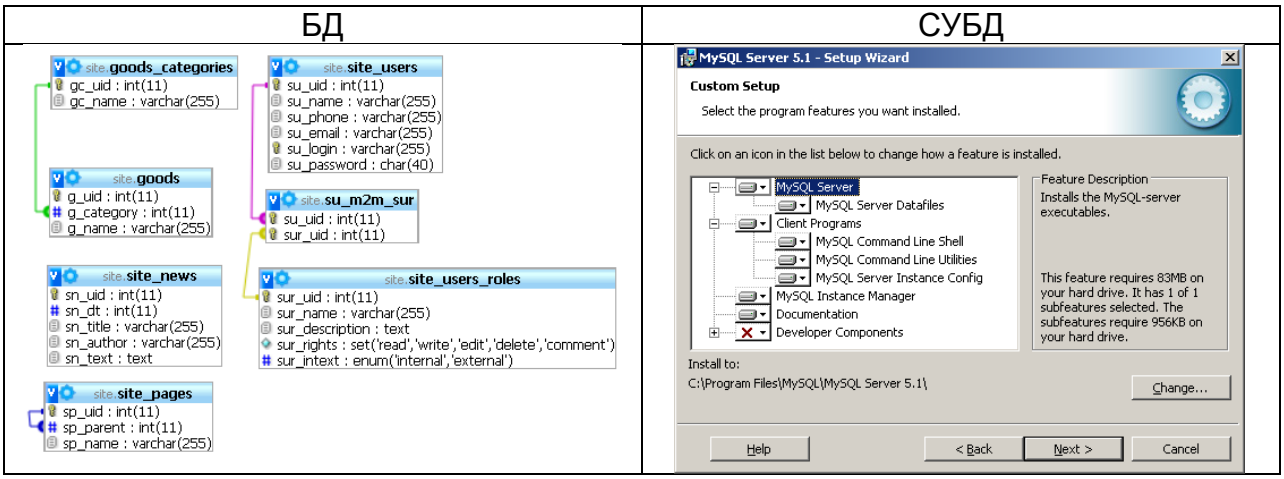
1. Данные и базы данных

Данные (data) – представление фактов и идей в формализованном виде, пригодном для передачи и обработки в некотором информационном процессе.

su_uid Уникальный идентификатор пользователя	su_name ФИО пользователя	su_phone Телефон пользователя	su_email E-mail пользователя	su_login Логин пользователя	su_password Пароль пользователя
1	Пупкин В.В.	123-45-67	pupkin@mail.ru	pupkin	7c4a8d09ca3762af61e59520943dc26494f8941b
2	John Smith	111-22-33	smith@gmail.com	smith	dd5fef9c1c1da1394d6d34b248c51be2ad740840

База данных (БД, database) – структурированный организованный набор данных, описывающих характеристики какой-либо физической или виртуальной системы.

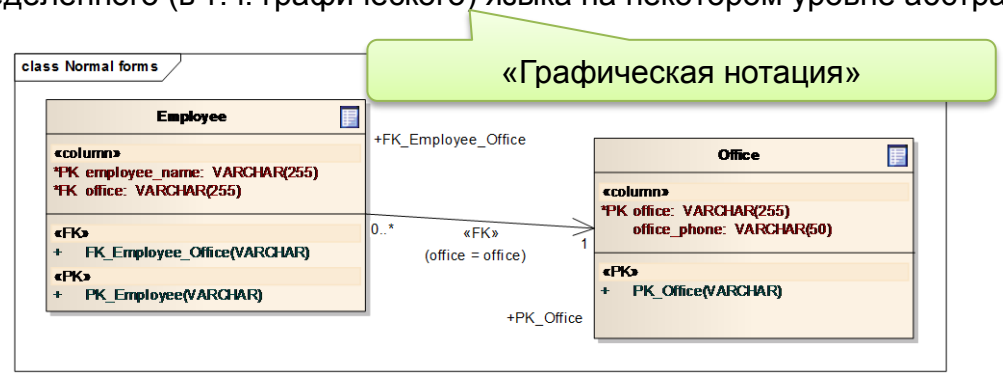
Система управления базами данных (СУБД, database management system) – программное обеспечение, предназначенное для организации и управления базами данных.



2. Модели баз данных

2.1. Понятие модели базы данных

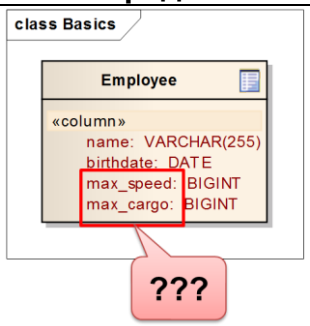
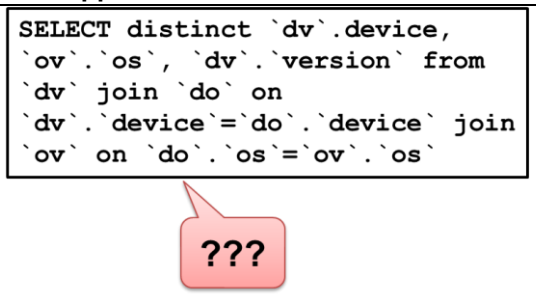
Модель базы данных (database model) – описание базы данных с помощью определённого (в т.ч. графического) языка на некотором уровне абстракции.



Уровни моделирования («уровни абстракции»)

Уровень	В чём суть	Пример «из жизни»
Инфологический	Описание предметной области, нет привязки к СУБД, предназначено «для людей».	«Данные сотрудников будем хранить в личных делах».
Даталогический	Модель предметной области в привязке к СУБД определённого вида или к конкретной СУБД.	«Личное дело каждого сотрудника представляет собой два документа – листок по учёту кадров и биографию».
Физический	Таблицы, связи, индексы, методы хранения, настройки производительности, безопасности и т.п.	«Документы личного дела будут распечатаны на листах А4, сшиты в папку и спрятаны в сейф в углу кабинета начальника».

Модель базы данных формируется на любом из уровней с учётом следующих требований:

Адекватность предметной области	Удобство использования
	

2.2. Инфологические модели

На инфологическом уровне приводится описание предметной области без привязки к конкретной СУБД и в форме, предназначенной для удобного восприятия человеком.

Описывать модель здесь можно…

«Словами» (просто в виде списков)

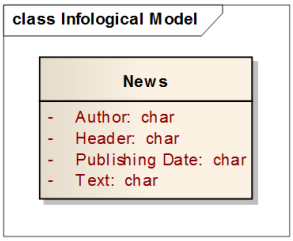
Новости:
- Заголовок
- Автор
- Дата публикации
- Текст

«Табличками» (в Word или подобных редакторах)

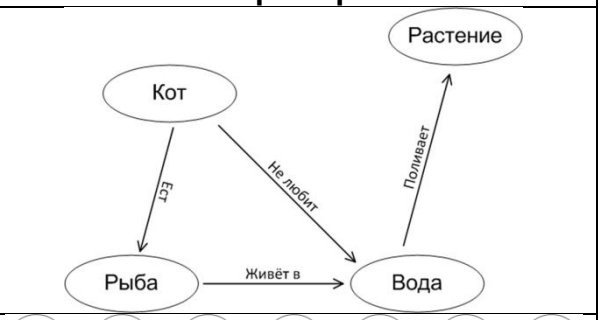
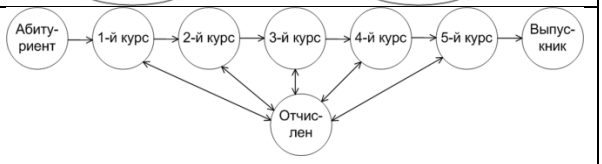
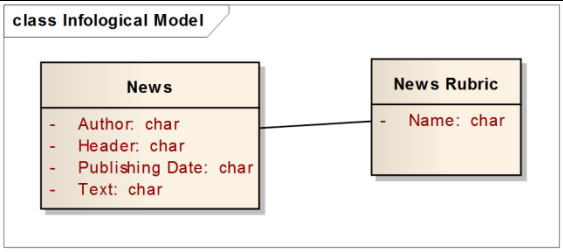
Новости

Заголовок	Автор	Дата публикации	Текст
Текст	*Текст*	*Дата*	*Текст*

Специальными графическими нотациями (UML и т.п.)



Виды инфологических моделей

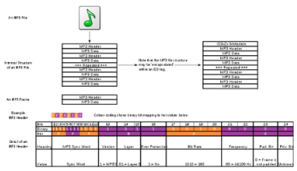
Вид	Пояснение	Пример
Семантическая	Часто строится с использованием семантических сетей, показывает взаимосвязь объектов, понятий и т.п.	
Графовая	Полностью основана на теории графов, достаточно эффективна лишь в специфических предметных областях.	
«Сущность-связь»	Идеально подходит для моделирования реляционных баз данных. Основана на понятиях «сущности», «атрибута», «связи».	

2.3. Даталогические модели

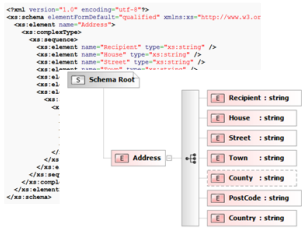
На даталогическом уровне модель предметной области представляется в привязке к СУБД определённого вида или к конкретной СУБД и описывает способ организации данных безотносительно их физического размещения.

Описывать модель здесь можно…

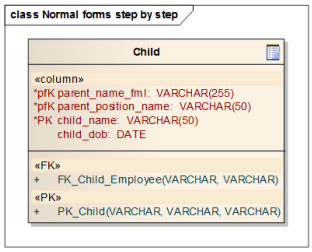
Спецификациями форматов данных



Собственными средствами формата (XSD в XML)



Специальными графическими нотациями (UML и т.п.)



Виды даталогических моделей (кратко, подробно будет позже)

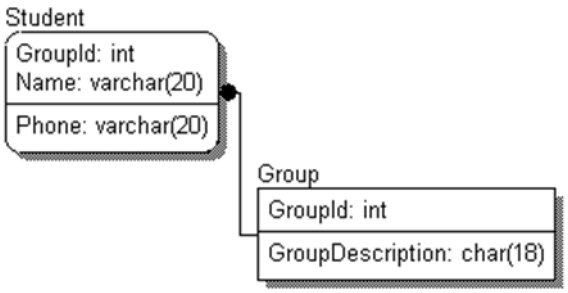
Модели	Пояснение
Документальные (архивы)	Обработка однотипных или разнотипных файлов по некоторым правилам
Фактографические (картотеки)	Картотека («плоская таблица») в любом её виде
Теоретико-графовые	Сети и графы с соответствующими правилами обработки
Теоретико-множественные	Например, **реляционная** («сущность-связь») – о ней речь пойдёт позже
Объектно-ориентированные	См. ООП в программировании ☺
Основанные на инвертированных файлах	Редкий случай. Лучший пример такого в жизни – «облако тегов».

2.4. Физические модели

Физический уровень описывает конкретные таблицы, связи, индексы, мето-

ды хранения, настройки производительности, безопасности и т.п.

Описывать модель здесь можно… как угодно. Чаще всего выбирают такой же способ представления, как и у даталогической модели. Из-за разнообразия предметных областей здесь используется всё – от чертежей и схем, до специальных нотаций (IDEF0).



3. Виды баз данных

Виды баз данных базируются на даталогических моделях и представлены следующим списком:

- Картотеки.
- Сетевые базы данных.
- Иерархические базы данных.
- Реляционные базы данных.
- Многомерные базы данных.
- Объектно-ориентированные базы данных.
- Дедуктивные базы данных.
- NoSQL базы данных.

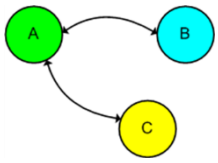
Картотека (card index) – упорядоченное (по алфавиту, дате и т.п.) собрание данных в виде записей («карт»), каждая из которых предоставляет сведения о каком-то объекте базы данных. Современный аналог картотек – «табличка» в Excel.

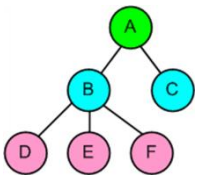
	A	B	C
	Автор	**Книга**	**Страниц**
	Азимов А.	Основание	154
	Азимов А.	Основание и империя	102
	Азимов А.	Второе основание	104

В **сетевых** (network) БД каждый элемент может быть связан с другим, а **иерархические** (hierarcy) БД построены на основе той или иной иерархической структуры данных (например, на основе дерева).

Типичные примеры «иерархических БД»:

- Файловая система.
- Реестр Windows.
- LDAP и Active Directory.



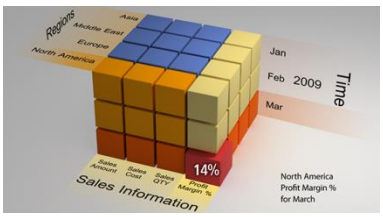


Реляционные (relational) БД основаны на теоретико-множественной реляционной даталогической модели (предложена доктором Эдгаром Коддом в 1970 году). Все данные представлены в виде (связанных между собой) таблиц, разбитых на строки и столбцы.

u_uid	u_login	u_password	u_fio	u_role
1	John	61409aa1fd47d4a5332de23cbf59a36f	John Smith	1
2	Joe	3a368818b7341d48660e8dd6c5a77dbe	Joe Black	2

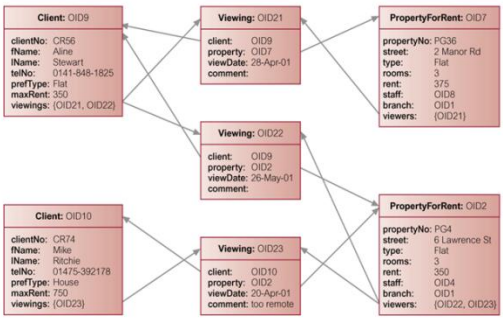
ame	r_rights
Admin	
User	

Многомерные БД (OLAP, online analytical processing) предназначены для обработки данных из различных источников и временных данных. Могут строиться на основе реляционных БД или на основе своих, более сложных хранилищ.



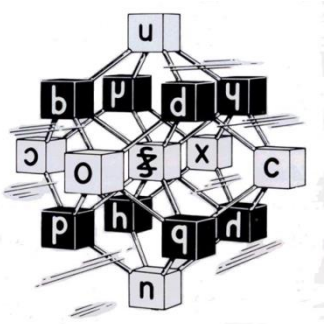
В **объектно-ориентированных** (object-oriented) БД данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями.

Эта технология напоминает объектно-ориентированное программирование (ООП) в применении к БД. СУБД для работы с такими технологиями бывают объектными и объектно-реляционными.



Дедуктивная (deductive) БД состоит из двух частей: экстенциональной (содержащей факты) и интенциональной (содержащей правила для логического вывода новых фактов).

Основным отличием дедуктивной СУБД от реляционной является то, что правила интенциональной части БД и запросы пользователей могут содержать рекурсию.

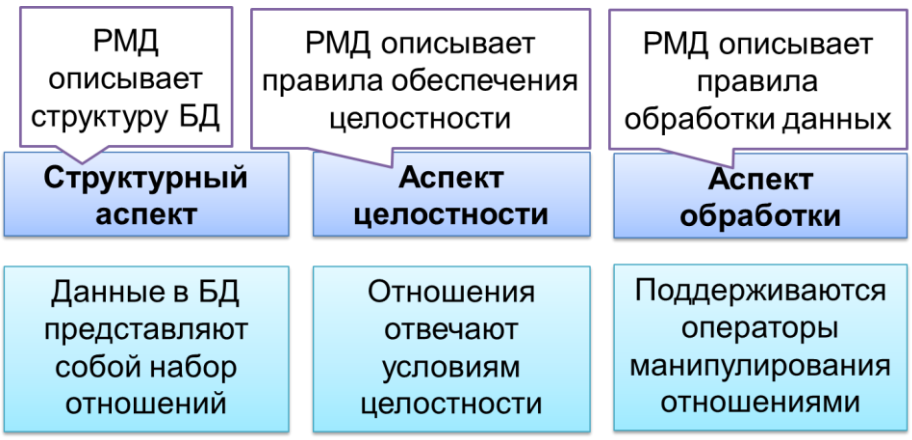


NoSQL (not only SQL) БД реализуют ряд подходов, имеющих существенные отличия от используемых в реляционных СУБД. Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и т.д.

4. Реляционная модель данных

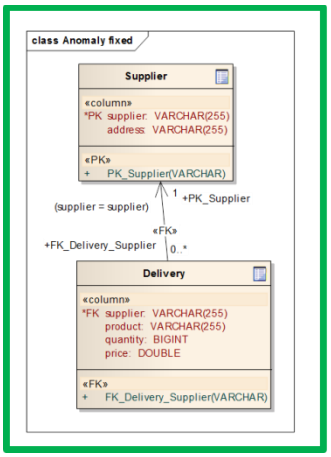
4.1. Определение

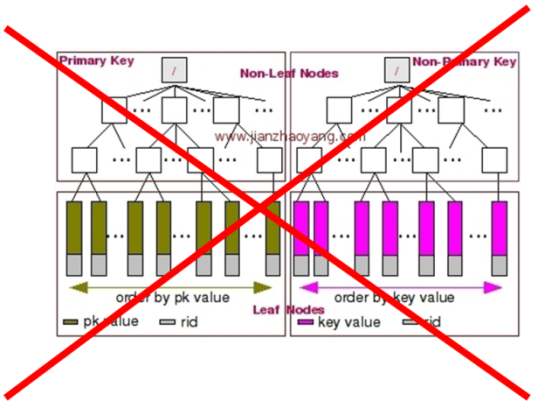
Реляционная модель данных (relational data model) – математическая теория, описывающая структурный аспект, аспект целостности и аспект обработки данных в реляционных базах данных.



4.2. Основные факты о реляционной модели данных

РМД является **логической**, т.е. отношения являются логическими (абстрактными), а не физическими (хранимыми) структурами.





Все данные представлены явным заданием значений атрибутов в кортежах отношений, т.е. нет никаких указателей (адресов), связывающих одно значение с другим.

su_uid Уникальный идентификатор пользователя	su_name ФИО пользователя	su_phone Телефон пользователя	su_email E-mail пользователя	su_login Логин пользователя	su_password Пароль пользователя
1	Пупкин В.В.	123-45-67	pupkin@mail.ru	pupkin	7c4a8d09ca3762af61e59520943dc26494f8941b
2	John Smith	111-22-33	smith@gmail.com	smith	dd5fef9c1c1da1394d6d34b248c51be2ad740840

РМД поддерживает как декларативное, так и процедурное программирование.

```
CREATE TABLE IF NOT EXISTS `news` (
  `n_uid` int(11) NOT NULL
AUTO_INCREMENT,
  `n_parent` int(11) NOT NULL,
  `n_dt` int(11) NOT NULL,
  `n_header` text NOT NULL,
  `n_text` text NOT NULL,
  PRIMARY KEY (`n_uid`),
  KEY `n_parent` (`n_parent`),
  KEY `n_dt` (`n_dt`)
) ENGINE=InnoDB  DEFAULT CI
AUTO_INCREMENT=100002 ;
```

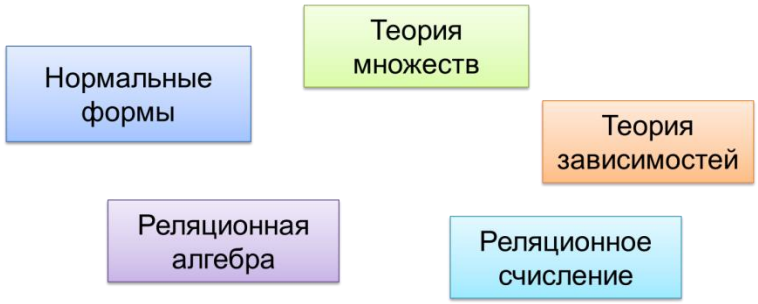
```
DELIMITER //
CREATE TRIGGER `upd_date_ai` AFTER INSERT ON
`news`
 FOR EACH ROW BEGIN
       DECLARE old_last_date int;
       SET old_last_date = (SELECT `nr_last_date` from
`news_rubrics` where `nr_uid`=NEW.`n_parent`);
       IF old_last_date < NEW.`n_dt` THEN
          UPDATE `news_rubrics` SET `nr_last_date` =
NEW.`n_dt` where `nr_uid`=NEW.`n_parent`;
       END IF;
    END
//
```

4.3. Достоинства реляционной модели данных

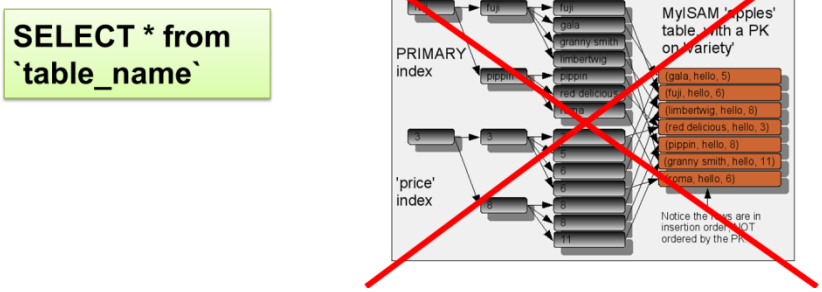
Простота (единственной информационной конструкцией является отношение).

su_uid Уникальный идентификатор пользователя	su_name ФИО пользователя	su_phone Телефон пользователя	su_email E-mail пользователя	su_login Логин пользователя	su_password Пароль пользователя
1	Пупкин В.В.	123-45-67	pupkin@mail.ru	pupkin	7c4a8d09ca3762af61e59520943dc26494f8941b
2	John Smith	111-22-33	smith@gmail.com	smith	dd5fef9c1c1da1394d6d34b248c51be2ad740840

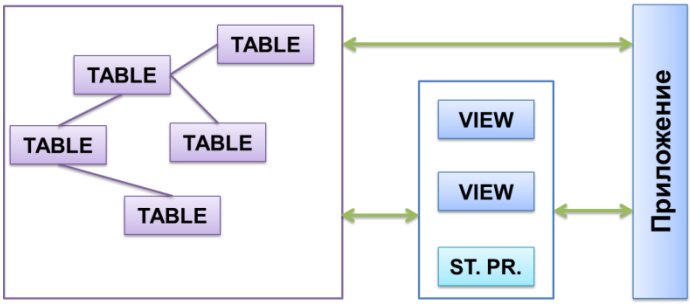
При проектировании применяются строгие правила, базирующие на математическом аппарате.



РМД обеспечивает полную независимость данных (приложениям «не нужно знать» ничего о внутренних форматах хранения, способах обработки данных и т.п.)

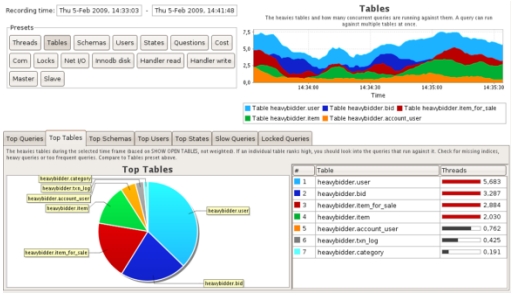


Изменения в структуре БД могут быть скрыты от внешних приложений (в т.ч. с использованием т.н. «представлений» (view) и хранимых подпрограмм (stored routines)).

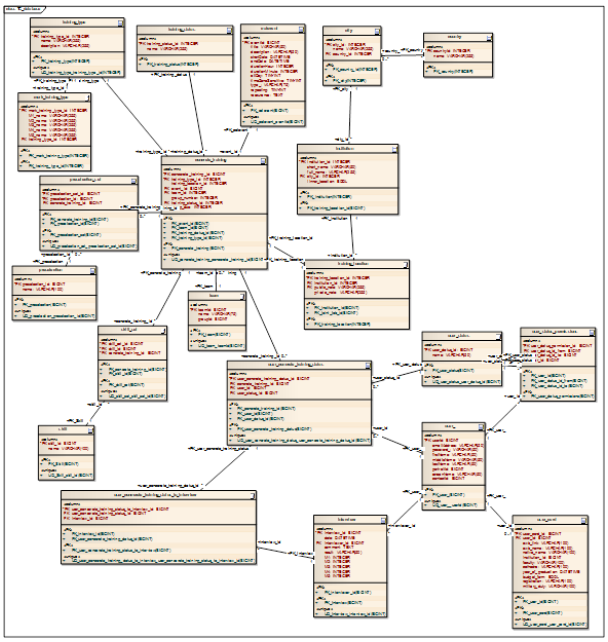


4.4. Недостатки реляционной модели данных

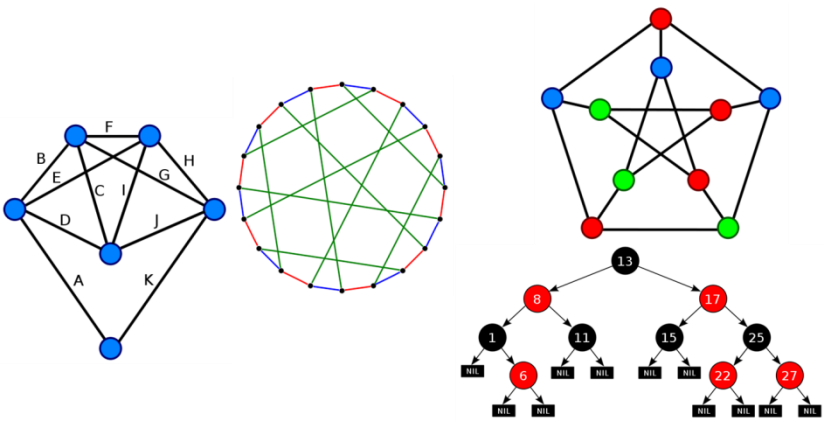
Относительно низкая скорость доступа к данным и использование большого объёма внешней памяти.



Трудность понимания структуры данных из-за появления большого количества таблиц в результате логического проектирования.

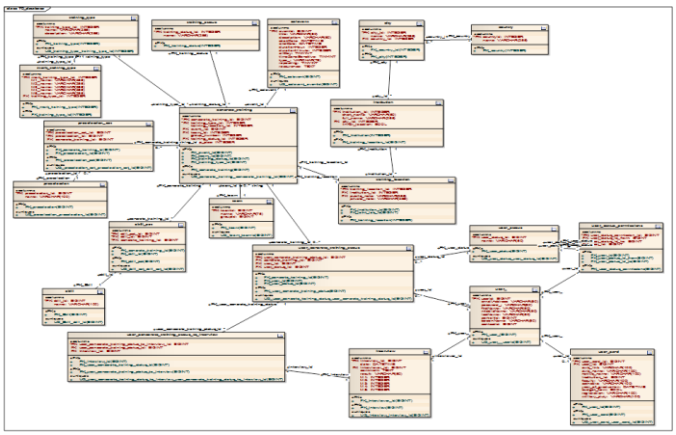


Невозможность или крайне высокая сложность представления в виде таблиц некоторых предметных областей.



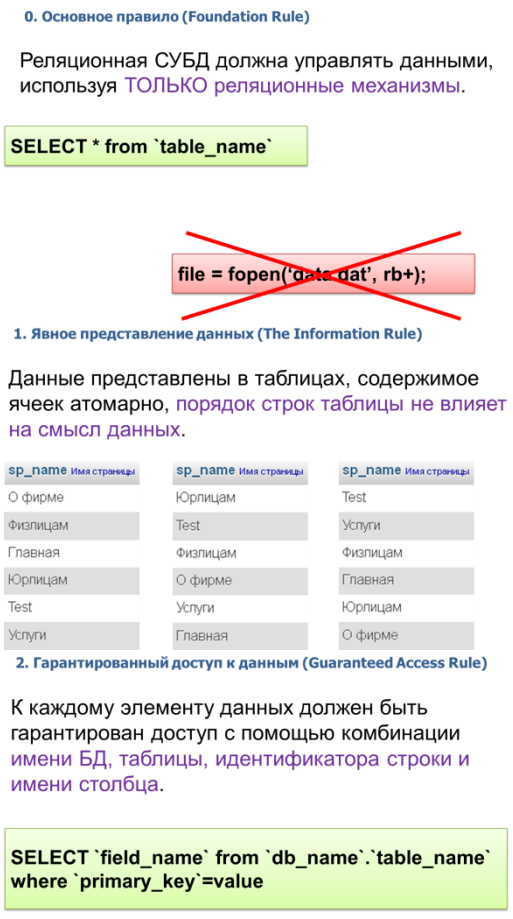
4.5. Реляционные базы данных

Реляционные базы данных (relational databases) – базы данных, основанные на только что рассмотренной реляционной модели данных.



5. Реляционные системы управления базами данных

Любая реляционная СУБД (это – самый распространённый вид СУБД) должна удовлетворять определённым «12-ти правилам Кодда». На практике некоторые СУБД отступают от тех или иных правил, но лишь по особой необходимости.



0. Основное правило (Foundation Rule)

Реляционная СУБД должна управлять данными, используя ТОЛЬКО реляционные механизмы.

1. Явное представление данных (The Information Rule)

Данные представлены в таблицах, содержимое ячеек атомарно, порядок строк таблицы не влияет на смысл данных.

2. Гарантированный доступ к данным (Guaranteed Access Rule)

К каждому элементу данных должен быть гарантирован доступ с помощью комбинации имени БД, таблицы, идентификатора строки и имени столбца.

3. Обработка неизвестных значений (Treatment of Null Values)

Неизвестные значения NULL должны поддерживаться для всех типов данных при выполнении любых операций.

Например, для числовых данных NULL'ы не должны рассматриваться как нули, а для символьных – как пустые строки.

```
SELECT * FROM `a` RIGHT JOIN `b`
ON `a`.`name` = `b`.`name`
```

id	name	id	name
NULL	NULL	1	Nathan
1	John	2	John
NULL	NULL	3	Bob
3	Jack	4	Jack

4. Доступ к описанию БД в терминах РМД (Active On-Line Catalog Based on the Relational Model)

С помощью стандартных средств производится не только управление самими данными, но и описание их структур (таблиц, связей и т.п.)

```
CREATE TABLE IF NOT EXISTS `news` (
  `n_uid` int(11) NOT NULL AUTO_INCREMENT,
  `n_parent` int(11) NOT NULL,
  `n_dt` int(11) NOT NULL,
  `n_header` text NOT NULL,
  `n_text` text NOT NULL,
  PRIMARY KEY (`n_uid`),
  KEY `n_parent` (`n_parent`),
  KEY `n_dt` (`n_dt`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=100002 ;
```

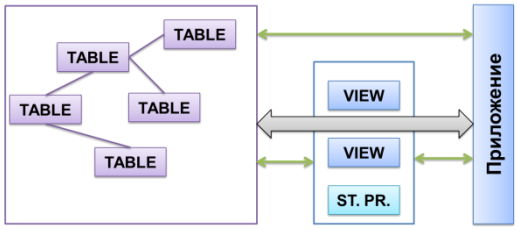
5. Полнота подмножества языка (Comprehensive Data Sublanguage Rule)

Реляционная СУБД должна поддерживать язык, который:

- имеет линейный синтаксис (может быть записан простым текстом);
- может использоваться как интерактивно, так и в прикладных программах;
- поддерживает все необходимые реляционные операции.

6. Возможность модификации представлений (View Updating Rule)

Представления (view) должны поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы.



7. Наличие высокоуровневых операций управления данными (High Level Data Manipulation)

Операции вставки, модификации и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но по отношению к любому множеству строк.

```
DELETE from `db_name`.`table_name` where
`primary_key`>=999
```

3. Обработка неизвестных значений (Treatment of Null Values)

Неизвестные значения NULL должны поддерживаться для всех типов данных при выполнении любых операций. Например, для числовых данных NULL'ы не должны рассматриваться как нули, а для символьных – как пустые строки.

4. Доступ к описанию БД в терминах РМД (Active On-Line Catalog Based on the Relational Model)

С помощью стандартных средств производится не только управление самими данными, но и описание их структур (таблиц, связей и т.п.)

5. Полнота подмножества языка (Comprehensive Data Sublanguage Rule)

Реляционная СУБД должна поддерживать язык, который: имеет линейный синтаксис (может быть записан простым текстом); может использоваться как интерактивно, так и в прикладных программах; поддерживает все необходимые реляционные операции.

6. Возможность модификации представлений (View Updating Rule)

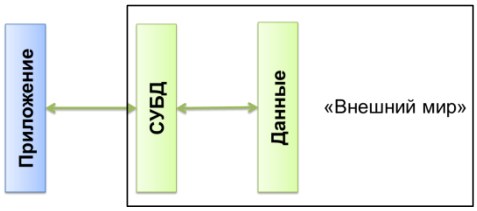
Представления (view) должны поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы.

7. Наличие высокоуровневых операций управления данными (High Level Data Manipulation)

Операции вставки, модификации и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но по отношению к любому множеству строк.

8. Физическая независимость данных (Physical Data Independence)

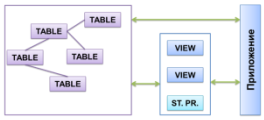
Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.



9. Логическая независимость данных (Logical Data Independence)

Представление данных в приложении не должно зависеть от структуры реляционных таблиц.

Если в процессе нормализации структура БД меняется, представление (view) должно обеспечить для приложений доступ к данным так, словно изменений не было.



10. Независимость контроля целостности (Integrity Independence)

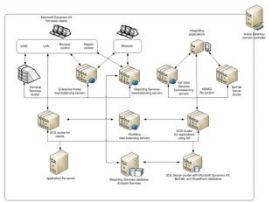
Вся информация, необходимая для поддержания целостности, должна находиться в словаре данных.

Язык для работы с данными должен выполнять проверку входных данных и автоматически поддерживать целостность данных.

```
ALTER TABLE `news`
  ADD CONSTRAINT `news_ibfk_1` FOREIGN KEY (`n_parent`)
REFERENCES `news_rubrics` (`nr_uid`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

11. Дистрибутивная независимость (Distribution Independence)

База данных может быть распределённой, может находиться на нескольких компьютерах, переноситься с одного компьютера на другой, и это не должно оказывать влияния на приложения.



12. Согласование языковых уровней (The Non-Subversion Rule)

Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

```
ALTER TABLE `news`
  ADD CONSTRAINT `news_ibfk_1` FOREIGN KEY (`n_parent`)
REFERENCES `news_rubrics` (`nr_uid`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

8. Физическая независимость данных (Physical Data Independence)

Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.

9. Логическая независимость данных (Logical Data Independence)

Представление данных в приложении не должно зависеть от структуры реляционных таблиц. Если в процессе нормализации структура БД меняется, представление (view) должно обеспечить для приложений доступ к данным так, словно изменений не было.

10. Независимость контроля целостности (Integrity Independence)

Вся информация, необходимая для поддержания целостности, должна находиться в словаре данных. Язык для работы с данными должен выполнять проверку входных данных и автоматически поддерживать целостность данных.

11. Дистрибутивная независимость (Distribution Independence)

База данных может быть распределённой, может находиться на нескольких компьютерах, переноситься с одного компьютера на другой, и это не должно оказывать влияния на приложения.

12. Согласование языковых уровней (The Non-Subversion Rule)

Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

6. История развития баз данных

1-й этап – «большие машины»

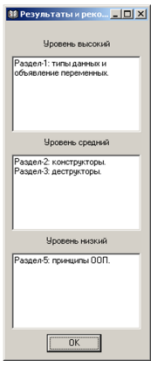
Первый этап развития СУБД связан с организацией баз данных на больших машинах типа IBM 360/370, ЕС-ЭВМ и т.п.

Базы данных хранились во внешней памяти центральной ЭВМ, пользователями этих баз данных были задачи, запускаемые в основном в пакетном режиме.

2-й этап – «персональные компьютеры»

Эпоха «быдлокодинга» – появилось множество «СУБД-образных» программ, написанных «на скорую руку».

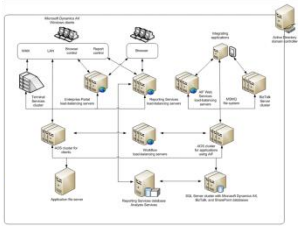
В большинстве своём эти программы были примитивны, решали узкий круг задач, были ориентированы на однопользовательский режим.



3-й этап – «распределённые базы данных»

Остро встала задача согласованности данных, логически связанных друг с другом, но хранящихся и обрабатывающихся в разных местах.

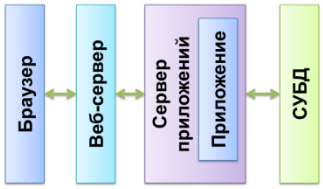
Распределённые СУБД сохранили все преимущества «настольных» СУБД и в то же время позволили организовать параллельную обработку информации и поддержку целостности БД.



4-й этап – «многоуровневая архитектура»

Отпадает необходимость использования специализированного клиентского программного обеспечения (всё берут на себя сервера приложений и сами веб-приложения).

Клиентской частью для доступа к данным выступают браузеры и их аналоги.



1-й этап – «большие машины»

Первый этап развития СУБД связан с организацией баз данных на больших машинах типа IBM 360/370, ЕС-ЭВМ и т.п.

Базы данных хранились во внешней памяти центральной ЭВМ, пользователями этих баз данных были задачи, запускаемые в основном в пакетном режиме.

2-й этап – «персональные компьютеры»

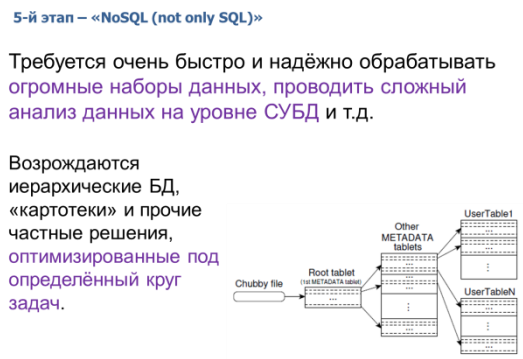
Эпоха «быдлокодинга» – появилось множество «СУБД-образных» программ, написанных «на скорую руку». В большинстве своём эти программы были примитивны, решали узкий круг задач, были ориентированы на однопользовательский режим.

3-й этап – «распределённые базы данных»

Остро встала задача согласованности данных, логически связанных друг с другом, но хранящихся и обрабатывающихся в разных местах. Распределённые СУБД сохранили все преимущества «настольных» СУБД и в то же время позволили организовать параллельную обработку информации и поддержку целостности БД.

4-й этап – «многоуровневая архитектура»

Отпадает необходимость использования специализированного клиентского программного обеспечения (всё берут на себя сервера приложений и сами веб-приложения). Клиентской частью для доступа к данным выступают браузеры и их аналоги.



5-й этап – «NoSQL (not only SQL)»

Требуется очень быстро и надёжно обрабатывать огромные наборы данных, проводить сложный анализ данных на уровне СУБД и т.д. Возрождаются иерархические БД, «картотеки» и прочие частные решения, оптимизированные под определённый круг задач.