**Реляционная база данных** - совокупность отношений (таблиц), содержащих всю информацию, которая должна храниться в БД. Основана на реляционной модели данных.

Реляционная модель данных включает следующие компоненты:

* Структурный аспект - данные в базе данных представляют собой набор отношений (таблиц).
* Аспект целостности — отношения (таблицы) отвечают определенным условиям целостности.

1. *Структурная целостность –* Отсутствие дубликатов кортежей, наличие первичного ключа, отсутствие упорядоченности кортежей.
2. *Ссылочная целостность –* Выполняется одно из двух правил:

* При удалении кортежа из родительского отношения, удаляются все связанные кортежи подчиненного отношения.
* При удалении кортежа из родительского отношения, на месте ключа в связанных отношениях ставится NULL.

1. *Семантическая целостность –* обеспечивается разработчиком с помощью средств СУБД.

Constraints: Тип данных, диапазон, уникальность.

NOT NULL, UNIQUE, PRIMARY KEY, CHECK, FOREYGN KEY.

* Аспект обработки (манипулирования) — РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).

определение множества операций над данными.

**Типы данных**

* **Целые числа.**INTEGER, SMALLINT, BIGINT
* **Десятичные числа.**Числа, имеющие дробную часть, но которые вычис­ляются точно. NUMERIC, DECIMAL.
* **Числа с плавающей точкой.**Величины, которые можно вычислить при­близительно, такие как вес или расстояние. Числа с плавающей точкой могут представлять более широкий диапазон значений, чем десятичные числа, но при работе с ними возможны погрешности округления. FLOAT, REAL
* **Строки символов постоянной длины. Char** –строка фиксированной длины, если длина меньше дополняется пробелами. Работает быстрее чем varchar.
* **Строки символов переменной длины.**Varchar – строка переменной длины, игнорирует конечные пробелы. Хранит дополнительно один байт(Длина строки).
* **Денежные величины.**Во многих СУБД поддерживается тип данных MONEY или CURRENCY. Однако в стандарте SQL такие типы данных не определяются.
* **Дата и время.** Стандарт SQL включает детальную спецификацию типов данных date, TIME, TIMESTAMP и INTERVAL, а также поддержку часовых поясов и точ­ного времени (например, десятые или сотые доли секунды).
* **Логические величины.**Одни СУБД, в частности Microsoft SQL Server, явно поддерживают логические значения (true или false), а другие разрешают выполнять в инструкциях SQL логические операции (сравнение, логическое И/ИЛИ и другие) над данными.
* **Длинные символьные объекты.**В стандарт SQL:1999 добавлен тип данных CLOB, который обеспечивает хранение больших символьных строк. Обычно СУБД ограничивает применение таких столбцов в интерактивных запросах и поисках.
* **Большие бинарные объекты.** В стандарт SQL:1999 добавлен тип данных BLOB, который поддерживает хранение неструктурированных последо­вательностей байтов переменной длины.
* **Национальные символы. Nvarchar, nchar –** хранит строку в Unicode. Символ char занимает 8 бит, nchar – 16 бит.

Для сравнения BLOB и CLOB в БД Oracle есть функция DBMS\_LOB.COMPARE(lob\_1, lob\_2, amount, offset\_1, offset\_2)

**Нормальная форма –** требование, предъявляемое к структуре таблиц, для устранения из базы избыточных функциональных зависимостей.

Нормализация – процесс преобразования отношений базы данных к виду, отвечающему нормальным формам (переход от одной нормальной формы к следующей). Позволяет привести БД к минимальной избыточности.

**1НФ –** все значения атомарны(отсутствуют массивы и списки).

**2НФ –** 1НФ + каждый неключевой атрибут функционально полно зависит от первичного ключа.

**3НФ –** 2НФ + отсутствие транзитивных зависимостей, когда один неключевой аттрибут зависит от другого неключевого.

**НФ Бойса-Кодда –** 3НФ + ключевые аттрибуты не зависят от неключевых.

**Денормализация –** намеренное преведение структуры БД в состояние, не соответствующее критериям нормализации, обычно с целью ускорения операций чтения.

**Потенциальный ключ –** подиножество аттрибутов отношения, которое уникально идентифицирует каждую запись в таблице. Удовлетворяет след. Требованиям – уникальность, минимальность (если из ключа убрать аттрибут, он утратит уникальность).

Из множетсва всех потенциальных ключей выбирается **первичный ключ** в качестве основного (используемого по умолчанию). Остальные ключи называют альтернативными.

Первичный ключ накладывает на поле ограничения NOT NULL, и UNIQUE. Также для каждого первичного ключа в таблице создается индекс.

**Индекс –** объект базы данных создаваемый с целью улучшения поиска записей в бд.

Индекс содержит ключи, построенные из одного или нескольких столбцов в таблице или представлении. Для индексов на диске эти ключи хранятся в виде структуры сбалансированного дерева, которая поддерживает быстрый поиск строк по значениям ключей.

Существует еще два способа хранения индекса:

* Hash – используется в основном для точного соответствия ( = ) и не работает со сравнениями (≥, ≤)
* B-tree – это наиболее распространенный способ хранения данных, обеспечивающий эффективную работу с дисковой памятью.

**Кластерные индексы -** При кластеризованном индексе строки физически хранятся на диске в том же порядке, что и индекс.Таблица может иметь только 1 кластерный индекс.

**Некластерные индексы** – они не перестраивают физическую структуру таблицы, а лишь организуют ссылки на соответствующие строки.

**Естественный ключ –** первичный ключ, состоящих из информационных полей таблицы.

**Суррогатный ключ –** дополнительное поле, добавленное в таблицу, чтобы служить первичным ключом. Генерируется исскуственно.(Гарантированная уникальность).

**Внешний ключ –** определяет связь поля с другой таблицей, ссылается на первичный ключ другой таблицы.

С помощью **Каскадных ограничений ссылочной целостности**  можно определять действия, которые будут предприниматься при попытке удаления или обновления ключа, на который указывает внешний ключ другой таблицы.

Могут быть определены следующие действия:

* NO ACTION – формирует ошибку и откатывает операции удаления или обновления в родительской таблице.
* CASCADE – соответствующие строки обновляются, либо удаляются вслед за столбцами родительской таблицы.
* SET NULL – значениям внешнего ключа присваиваются значения NULL при удалении или изменении строки родительской таблицы.
* SET DEFAULT – значения внешнего ключа при удалении или изменении строки родительской таблицы устанавливаются в значение по умолчанию.

**Транзакция –** последовательность действий, выполняющаяся как единое целое, и переводящая базу данных из одного согласованного состояния в другое.

**ACID** описывает требования к транзакционной системе:

**Atomicity (Атомарность) –** транзакция либо выполняется полностью, либо не выполняется совсем.

**Consistency (Согласованность) –** данные до транзакции должны быть согласованы с данными после выполнения транзакции.

**Isolation (Изолированность) –** параллельно выполняющиеся транзакции не должны оказывать влияния друг на друга.

**Durability (долговечность) –** данные должны быть сохранены независимо от проблем на нижних уровнях.

Уровни изоляции транзакции

**Чтение неподтверждённых данных (read uncommitted)**

Самый низкий уровень изоляции. Можно читать незафиксированные изменения других транзакций, но запись идет строго последовательно.

Исключает потерянное обновление – проблема, когда две транзакции одновременно пишут значение в одну ячейку, при этом одно из значений теряется.

**Чтение подтверждённых данных (read committed)**

Транзакция видит только зафиксированные изменения других транзакций.

Исключает грязное чтение – когда читаются данные, изменяемые другой транзакцией, но затем транзакция откатывается, и изменения исчезают.

**Повторяемое чтение (repeatable read)**

Транзакция не видит измененные или удаленные записи другой транзакции. Она блокирует изменение данных другой транзакцией. При многократной выборке транзакция получает один и тот же результат.

Исключает неповторяемое чтение – когда несколько раз читаются данные, в данный момент изменяемые другой транзакцией. В итоге каждый раз результат может быть разным.

**Сериализуемый (serializable)**

Транзакции полностью изолируются друг от друга, фактически выполняясь последовательно.

Исключает проблему фантомов – когда однна транзакция выбирает несколько строк по каким-то критериям. Другая между выборками вставляет либо удаляет сстроки или изменяет столбцы некоторых строк, используемых в критериях выборки.

В результате получается что одни и те же выборки первой транзакции дают разные множества строк.

Уровни изоляции реализуюся с помощью блокировок. Блокировкой называется временное ограничение на выполнение некоторых операций обработки данных. Происходит блокирование доступа к объекту.

**Блокировки по области действия (гранулярности):**

* Строчная блокировка - действуют только на одну строку таблицы, не ограничивая манипуляции над другими строками.
* Гранулярная блокировка – действует на всю таблицу (или страницу) и все строки.
* Предикатная блокировка – действует на область, ограниченную предикатами (отдельные поля, объекты).

Есть 2 вида блокировок **по строгости**:

* Разделяемая блокировка (s-блокировка  Shared lock). Резервирует ресурс только для чтения. Несколько транзакций могут одновременно накладывать эту блокировку. Другие транзакции в этот момент не могут изменять данный ресурс.
* Монопольная блокировка (x-блокировка exclusive lock). Резервирует ресурс на изменение. Может быть установлена на ресурс только при отсутствии других блокировок. На ресурс может быть установлена только одна монопольная блокировка.

**Блокировка по логике реализации**

* Пессиместичная блокировка – блокирует модифицируемые записи полностью, во время действия другие сессии не могут изменять данные.
* Оптимистичная блокировка – не ограничивает модифифкацию данных сторонними сессиями. Она предполагает, что модификации обычно не конфликтуют. Для определения конфликтов использует номер версии либо временную метку. Перед модификацией мы считываем версию. Перед записью мы проверяем, не изменилась ли версия. Если нет, изменения фиксируются, и версия обновляется. Если с момента считывания версия изменилась, то транзакция откатывается, либо применяются различные схемы разрешения коллизий.

Взаимоблокировка(deadlock) – это проблема конкурентного доступа, в которой транзакции блокируют друг друга. В частности, первая транзакция блокирует объект базы данных, доступ к которому хочет получить другая транзакция, и наоборот. (В общем, взаимоблокировка может быть вызвана несколькими транзакциями, которые создают цикл зависимостей.)

Система БД обрабатывает взаимную блокировку выполняя откат одной из транзакций. Вы можете повлиять на то, какая транзакция будет выбрана системой в качестве "жертвы" взаимоблокировки, присвоив в инструкции SET параметру DEADLOCK\_PRIORITY один из 21 (от -10 до 10) разных уровней приоритета взаимоблокировки.

HAVING применяется для фильтрации строк после группировки, фильтрует группы по значениям результата агрегатных функций.

Агрегатной функцией в языке SQL называется функция, возвращающая какое-либо одну строку по набору из входных строк.