



SQL Fundamentals

Обзор – День #1

Организационные вопросы

Введение в SQL

1. Извлечение информации из таблиц
2. Формирование вывода запроса
3. Агрегатные функции
4. Соединение таблиц

Обзор – День #2

5. Вложенные запросы
6. Соотнесенные запросы
7. Изменение содержимого таблиц
8. Создание таблиц
9. Поддержка целостности данных

Введение

1. Организационные вопросы
2. Что такое реляционная БД
3. Стандарт ANSI и Oracle SQL
4. Интерактивный и встраиваемый SQL
5. Способы доступа к БД
6. Знакомство с **Toad**

Доступ к удаленным станциям

5

SQL Fundamentals, Performance Lab

Нажать: windows+R
Ввести: mstsc

Вести
компьютер:
rdp.pflb.ru:5
6920

Login\pass:
UserX\UserS
QLX, где X
это номер
от 1 до 15

В Toad for
Oracle
выбрать
learn/learn

Классификации СУБД по модели данных

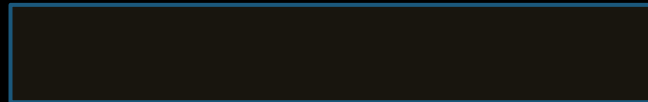
6

SQL Fundamentals, Performance Lab

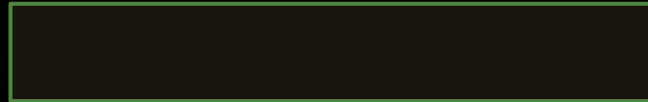
Иерархические



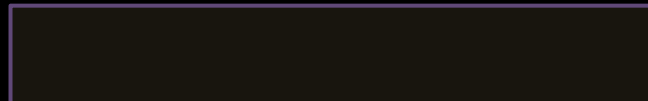
Сетевые



Реляционные



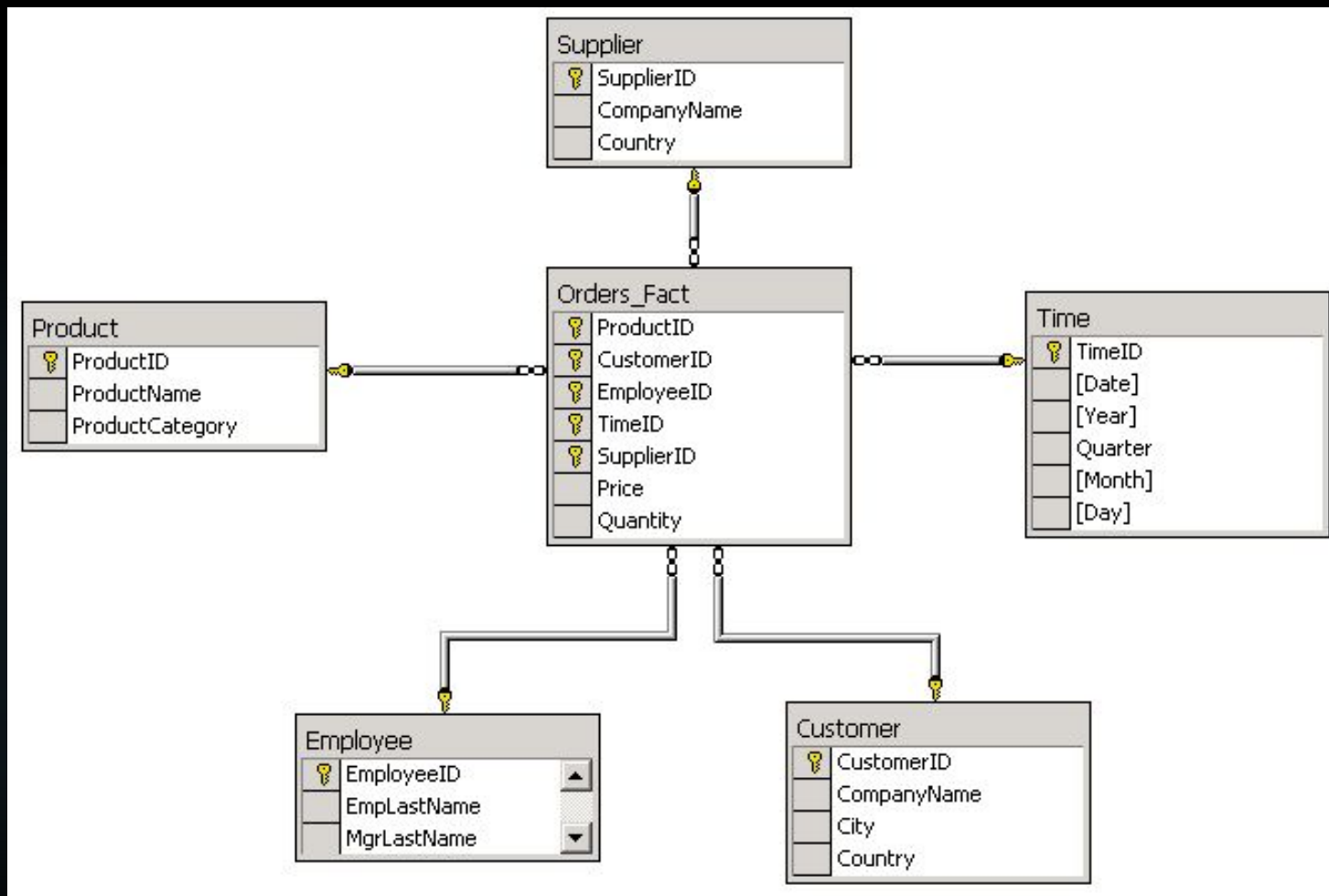
Объектно-ориентированные



Реляционная БД

7

SQL Fundamentals, Performance Lab



Стандарт ANSI и Oracle SQL

8

Тип данных	Значение
INTEGER	Представляет целочисленные значения длиной в 4 байта в диапазоне от -2^{32} до $2^{32} - 1$. INT - сокращенная форма от INTEGER.
REAL	Применяется для представления значений с плавающей точкой. Диапазон положительных значений простирается приблизительно от $2,23E-308$ до $-1,18E-38$. Также может быть представлено и нулевое значение.
FLOAT[(p)]	Подобно типу REAL, представляет значения с плавающей точкой [(p)]. Аргумент p определяет точность. При значении $p < 25$ представляемые значения имеют одинарную точность (требуют 4 байта для хранения), а при значении $p \geq 25$ - двойную точность (требуют 8 байтов для хранения).
CHAR[(n)]	Применяется для представления строк фиксированной длины, состоящих из n однобайтовых символов. Максимальное значение n равно 8000. Если n явно не указано, то его значение полагается равным 1.
VARCHAR[(n)]	Используется для представления строки однобайтовых символов переменной длины ($0 < n < 8\,000$). В отличие от типа данных CHAR, количество байтов для хранения значений типа данных VARCHAR равно их действительной длине.
TIMESTAMP	Для каждой базы данных система содержит счетчик, значение которого увеличивается всякий раз, когда вставляется или обновляется любая строка, содержащая ячейку типа TIMESTAMP, и присваивает этой ячейке данное значение.
DATETIME	Применяется для хранения даты и времени в виде целочисленных значений длиной в 4 и 2 байта соответственно. Составляющая даты значений типа DATETIME хранится в диапазоне от 01/01/1753 до

Интерактивный и встраиваемый SQL

- **Интерактивный SQL**

- Непосредственно в БД используется;
- После ввода команды, она сразу выполнится;

- **Встраиваемый SQL**

- Команды помещённых внутри программ на других языках программирования;

Подразделы SQL

10

SQL Fundamentals, Performance Lab

- DDL (Язык Определения Данных)
- DML (Язык Манипулирования Данными)
- DCD (Язык Управления Данными)

Способы доступа к БД

11

SQL Fundamentals, Performance Lab

- Способ доступа
 - Файл серверные
 - Клиент серверные
 - Встраиваемые

Знакомство с Toad

12

SQL Fundamentals, Performance Lab

- Поддерживает: Oracle Database, Microsoft SQL Server, Adaptive Server Enterprise, DB2, MySQL , Hadoop, MongoDB, SimpleDB, Apache Cassandra и Windows Azure
- TOAD содержит три основных компонента: Database Browser, SQL Editor и PL/SQL Procedure Editor и SQL Modeller.
- Официально работает на 32-битных Windows-платформах: Windows 95, 98, NT, 2000, XP и Vista.
- <https://www.toadworld.com/>

Извлечение информации

- Оператор **SELECT**
- Ключевое слово **DISTINCT**
- Ключевое слово **WHERE**
- Столбцы упорядочены, строки – нет
- Ключевое слово **ORDER BY**
- Что такое предикат
- Условия в предикатах: сравнения, логические (**=, >, <, =>, <=, <>, and, or, not**)
- Ключевые слова **IN, BETWEEN, LIKE**
- **NULL**-значения
- Псевдостолбцы **ROWID, LEVEL, ROWNUM**

Задание #1

Написать запрос, выводящий...

1. дату заказа, номер заказа и его стоимость для всех заказов (всех строк таблицы **Orders**)
2. всех продавцов с номером, большим 1005 (таблица **Sales**)
3. все заказы, оформленные в промежутке с '10/03/1990' по '10/04/1990' (таблица **Orders**)
4. в алфавитном порядке всех покупателей из города 'SanJose' (таблица **Clients**)
5. заказы с номером 3000, 3004 или 3005 (таблица **Orders**)
6. всех покупателей, проживающих в городах, не начинающихся с R, чей рейтинг больше 130 (таблица **Clients**)

Формирование вывода

- Именованние столбцов **AS**
- Использование констант
- Использование скалярных функций
- Функции **LENGTH, UPPER, LOWER, INITCAP, CONCAT, SUBSTR, INSTR**
- Функции **ROUND, TRUNC, REMAINDER, MOD**
- Функции **NVL, NULLIF, DECODE, CASE**
- Таблица **DUAL**
- Вложенные функции

Задание #2

Написать запрос, выводящий...

1. стоимости (**amt**) всех заказов, округленные до целого
2. имя и город проживания продавцов, причем если город не задан (т.е. **null**), выводить «не задан» вместо **null**
3. один столбец со строками вида "Рейтинг покупателя **cname** равен **rating** (**высокий**, **средний** или **низкий**)" для всех записей таблицы **Clients**, при этом имя покупателя должно быть выведено большими буквами. В конце строки в скобках указать «**высокий**» - если рейтинг выше 200, «**средний**» - если равен 200 и «**низкий**» – во всех остальных случаях.

Агрегатные функции

- Функции **COUNT**, **SUM**, **AVG**, **MAX**, **MIN**
- Различия *****, **DISTINCT**, **ALL** в агрегатных функциях
- Вложенные агрегатные функции
- Ключевое слово **GROUP BY**
- Ключевое слово **HAVING**

GROUP BY

- В конструкции GROUP BY можно указать сколько угодно выражений, в том числе – названия столбцов;
- Обычно GROUP BY используется для указания столбцов, значения в которых должны быть одинаковыми для того, чтобы соответствующие строки вошли в группу;
- Если в теле запроса есть GROUP BY, то в селект-листе этого запроса допускается использовать только:
 - Выражения, использованные в GROUP BY;
 - Агрегатные функции;
 - Константы;
- Выражения из GROUP BY необязательно подставлять в селект-лист.

Порядок обработки операторов

19

Логический порядок

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Синтаксический порядок

1. SELECT [DISTINCT | ALL]{*
| [<выражение для столбца> [[AS] <псевдоним>]] [...]}
1. FROM <имя таблицы> [[AS] <псевдоним>] [...]
2. [WHERE <предикат>]
3. [[GROUP BY <список столбцов>]
4. [HAVING <условие на агрегатные значения>]]
5. [ORDER BY <список столбцов>]

ORDER BY совместно с GROUP BY

- Если в теле запроса есть GROUP BY, то в конструкции ORDER BY можно указывать только:
 - Выражения, указанные в GROUP BY;
 - Выражения, указанные в селект-листе (можно использовать порядковый номер столбца, его имя или алиас);
 - Агрегатные функции (независимо от того, присутствуют ли они где-то еще в теле запроса);
- Если в теле запроса есть GROUP BY, то в конструкции ORDER BY нельзя указывать:
 - Столбцы таблицы, которые не присутствуют в GROUP BY.

Задание #3

Написать запрос, выводящий...

1. общее количество сделанных заказов
2. среднюю длину имени покупателя
3. дату и максимальную стоимость заказа за эту дату, отсортированные по убыванию максимальной стоимости заказа
4. названия городов, в которых суммарный рейтинг покупателей превышает 100

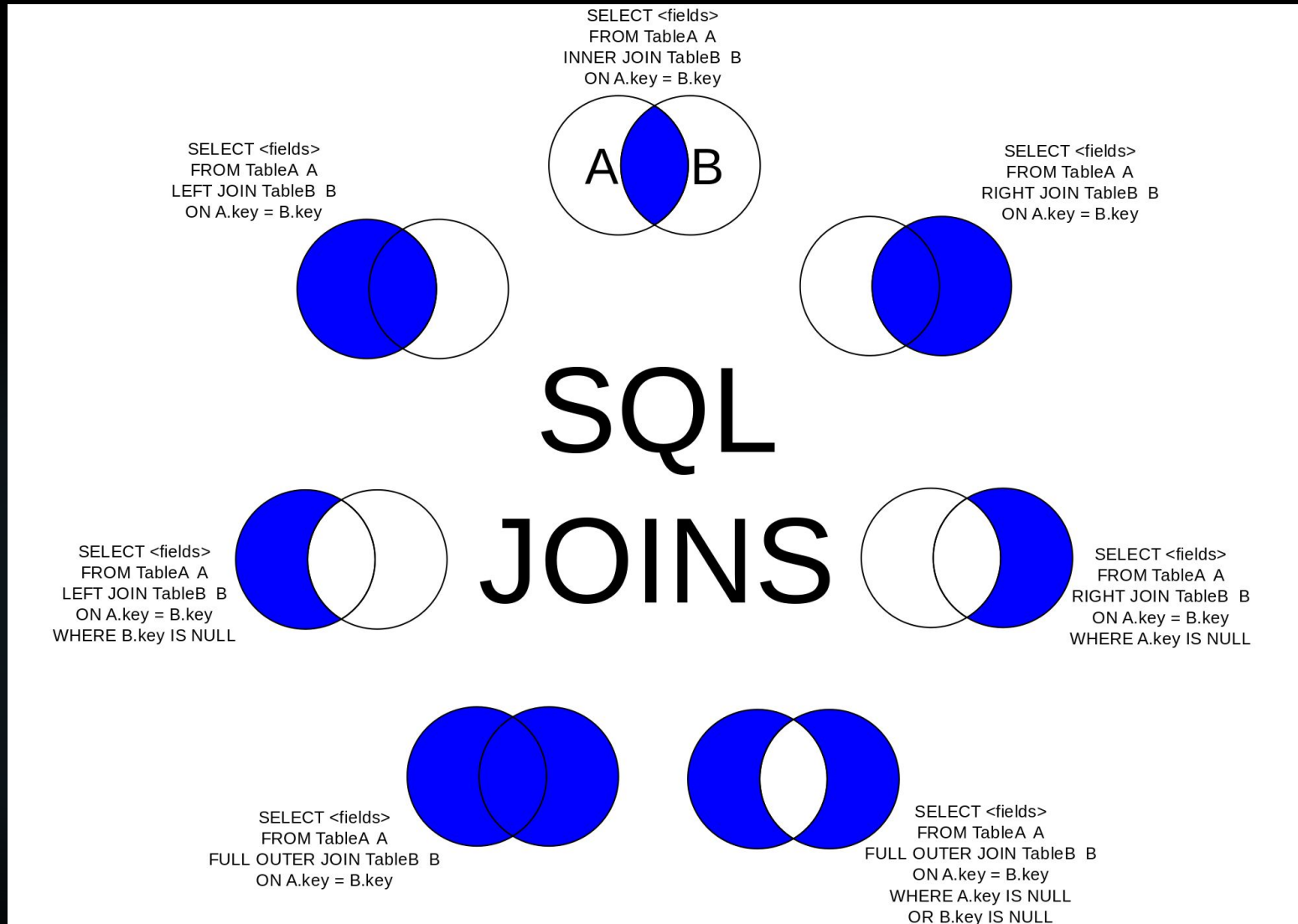
Соединение таблиц

- Разница между **INNER** и **OUTER JOIN**
- Разница между **LEFT OUTER**, **RIGHT OUTER** и **FULL OUTER JOIN**
- **NATURAL JOIN**
- Ключевое слово **USING**
- Ключевое слово **CROSS JOIN**
- Соединение более двух таблиц
- «Самосоединение» таблиц

Различный виды операции JOIN

23

SQL Fundamentals, Performance Lab



Задание #4

Написать запрос, выводящий...

1. имена продавцов и соответствующие им имена клиентов, в том числе для продавцов без клиентов
2. имена и номера продавцов, имеющих клиентов в Лондоне (**London**), Москве (**Moscow**) или Сочи (**Sochi**)
3. строки, показывающие всю информацию о тех продавцах и их клиентах , которые живут в одном и том же городе
4. имя продавца и сумму всех совершенных его клиентами заказов, отсортированные по убыванию суммы

Вложенные запросы

Что такое подзапрос

Принцип работы вложенного запроса

Single- и multirow подзапросы

Использование **IN** в подзапросах

Ключевые слова **ALL, ANY, SOME**

Операции над множествами: **UNION, INTERSECT, MINUS**

Операции над множествами: **ORDER BY**

Использование агрегатных функций в подзапросах

Запросы с **WITH**

Задание #5

Написать запрос, выводящий...

1. все заказы, оформленные продавцами из Лондона (**London**) без использования join
2. всех продавцов, имеющих комиссию выше средней
3. общее количество продавцов и покупателей из каждого города, отсортированное по убыванию количества
4. 2, 3, и 4 строки таблицы **Sales**, отсортированной по возрастанию имени продавца

Соотнесенные запросы

Принцип работы соотнесенного запроса

Ключевое слово **EXISTS**

Скалярные подзапросы

Принцип работы соотнесенного подзапроса

1. Выбрать строку из таблицы, именованной во внешнем запросе. Это будет текущая строка-кандидат.
2. Сохранить значения из этой строки-кандидата в псевдониме с именем в предложении FROM внешнего запроса.
3. Выполнить подзапрос. Везде, где псевдоним, заданный для внешнего запроса, найден, использовать значение текущей строки-кандидата. (Использование значения из строки-кандидата внешнего запроса в подзапросе называется внешней ссылкой.)
4. Оценить предикат внешнего запроса на основе результатов подзапроса, выполняемого в шаге 3. Он определяет, выбирается ли строка-кандидат для вывода.
5. Повторить процедуру для следующей строки-кандидата таблицы, и так далее, пока все строки таблицы не будут проверены.

Скалярные подзапросы

- Это single-row subquery, в которой только 1 столбец;
- Скалярные подзапросы обязательно заключаются в скобки;
- Могут быть соотнесенными;
- Могут быть использованы почти в любом месте, где может быть использовано выражение, за исключением:
 - В конструкции **GROUP BY**
 - В прочих случаях (не исследуются в рамках этого курса)

Задание #6

Написать запрос, выводящий...

1. имена и номера всех продавцов, которые имеют не менее одного заказчика
2. всех продавцов, имеющих комиссию не ниже средней по своему городу
3. [подзапросом] номера и имена всех продавцов, имеющих в своем городе заказчиков, которых они не обслуживают
4. [join'ом] номера и имена всех продавцов, имеющих в своем городе заказчиков, которых они не обслуживают
5. номера всех заказов, а также разницу между стоимостью каждого заказа и средней стоимостью заказа за тот же день

Изменение содержимого таблиц

Операторы **INSERT**, **UPDATE**, **DELETE**

Использование подзапросов при изменении содержимого

Понятие транзакции, ключевые слова **COMMIT**, **ROLLBACK**

Типы запросов: **DML**, **DDL**, **TCL**

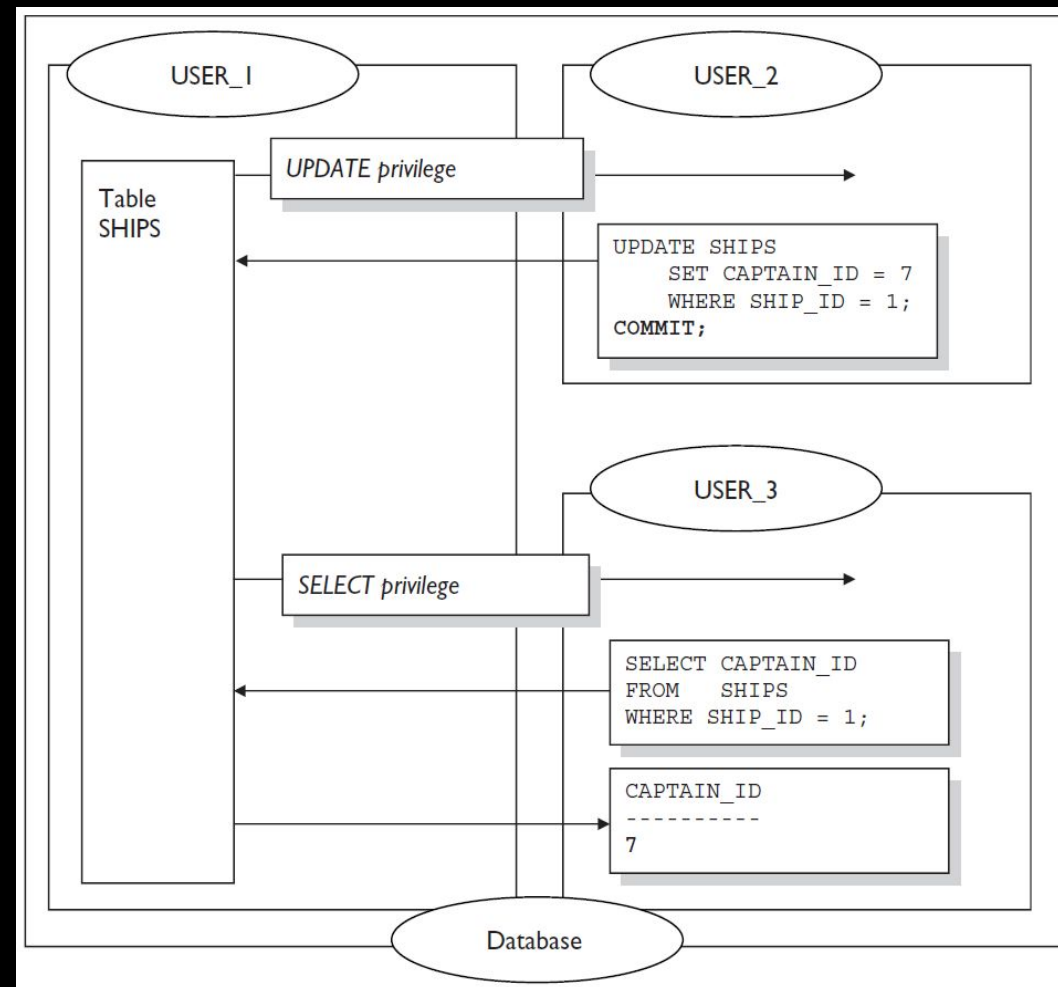
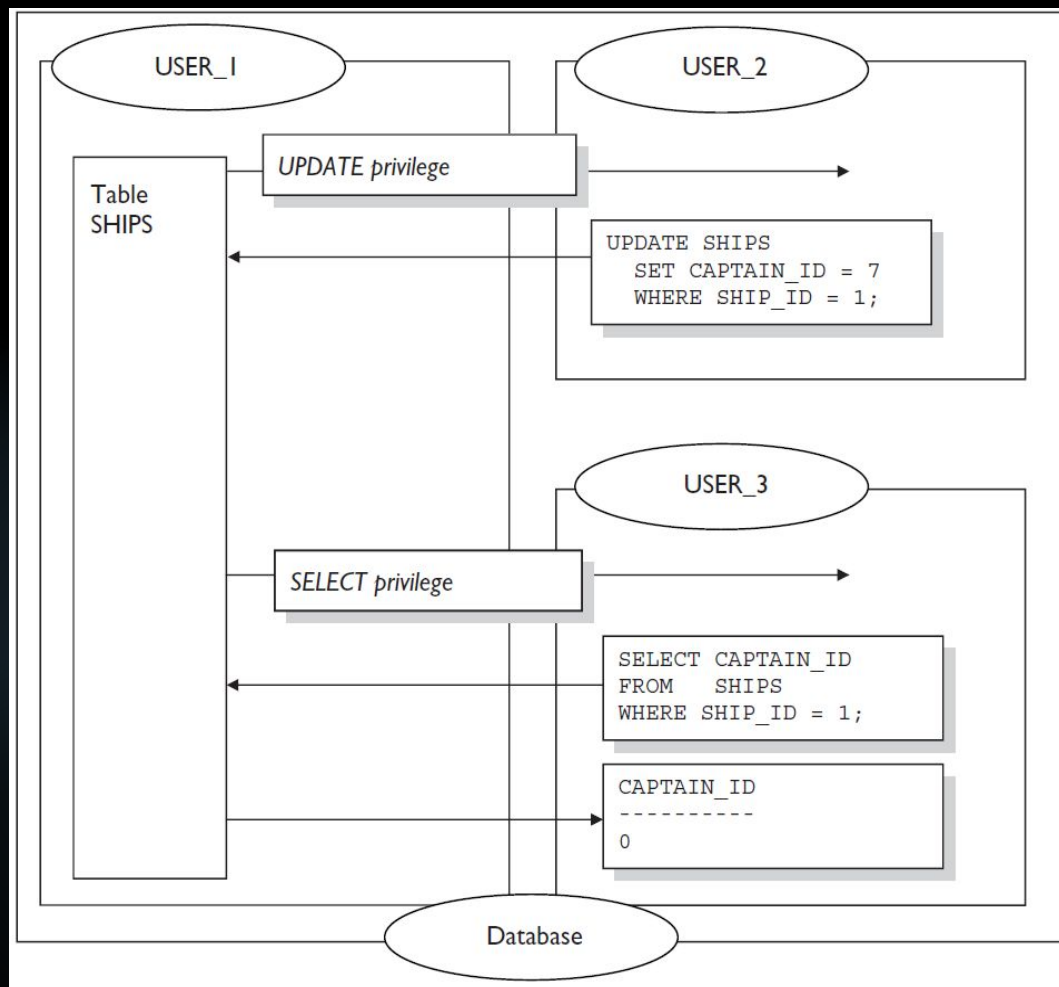
Порядок выполнения INSERT

1. Проверка, что таблица, указанная в **INTO**, существует
2. Проверка, что колонки указанные в **INSERT**, существуют в таблице
3. Выражения, указанные в списке **VALUES**, вычисляются
4. Проверяется совместимость типов выражений в **VALUES** и типов соответствующих колонок таблицы
5. Выражения в **VALUES** проверяются на соответствие имеющимся в таблице ограничениям целостности (они будут рассмотрены позднее)

ACID

- Atomicity (Атомарность)
- Consistency (Согласованность)
- Isolation (Изолированность)
- Durability (Надежность)

Многопользовательские коммиты



Задание #7

Написать запрос, который...

1. делает все города проживания клиентов, начинающиеся с S, написанными большими буквами, затем откатывает изменения
2. добавляет нового продавца (некоторые из полей можно оставить пустыми), причем новый продавец должен стать видимым для остальных слушателей курса
3. удаляет всех продавцов, у которых не указан город проживания и чьи имена длиннее восьми символов

Создание таблиц

Правила именования объектов БД

Основные типы данных: **NUMBER**, **CHAR**, **VARCHAR2**, **DATE**

Оператор **CREATE**, оператор **DESC**

Неявные коммиты

Использование подзапросов для создания таблиц

Операторы **ALTER**, **DROP**

Создание и модификация представлений (**VIEW**)

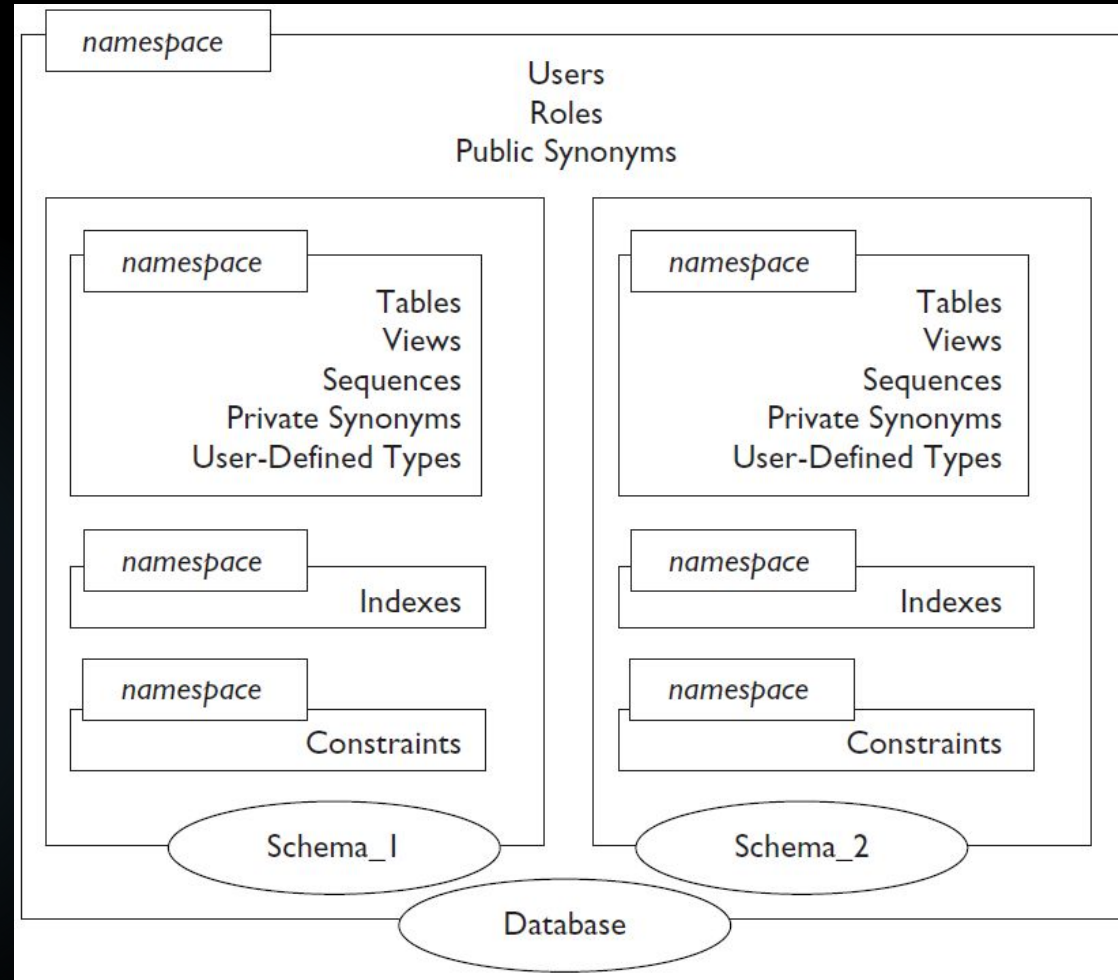
Правила именования объектов БД

1. Длина имени – хотя бы один символ, но не больше 30 символов;
2. Первый символ в имени должен быть буквой (латинского алфавита);
3. После первого символа могут идти любые буквы, цифры, а также знаки \$, _, # в любом порядке (никаких других символов в имени не допускается);
4. Имена не могут быть зарезервированными словами языка SQL (такими как **SELECT**, **CREATE** и т.д.)

Пространства имен

38

SQL Fundamentals, Performance Lab



NUMBER(n, m)

- Синоним **DECIMAL(n, m)**
- Параметры **n** и **m** – опциональные
- **n** $\in [1, 38]$ – максимальное количество значащих цифр. Значение по умолчанию = 38.
- **m** $\in [-84, 127]$ – количество цифр справа от запятой. Значение по умолчанию = 0.
- $x \in \text{NUMBER} \Rightarrow |x| \in [1*10^{-130}, 1*10^{126})$

NUMBER(n, m)

Datatype	Value Entered	Value Stored As
NUMBER	4.56	4.56
NUMBER(2)	4.56	5
NUMBER(5,2)	4.56	4.56
NUMBER(5,2)	4.5678	4.57
NUMBER(3,2)	10.56	Nothing is stored. Instead, displays error code ORA-01438: “value larger than specified precision allowed for this column”. The reason: the value has a precision of 4 (1,0,5,6—four digits), but NUMBER here is declared with a precision of 3.
NUMBER(5,-2)	1056.34	1100

CHAR(n)

- Строка фиксированной длины, состоящая из букв и цифр;
- **n** $\in [1, 2000]$ – длина строки. Необязательный параметр. Значение для n по умолчанию = 1;
- В столбцах типа **CHAR(3)** значения из двух символов (например, 'r2') будут храниться как 'r2 ', т.е. дополняться пробелами справа (до длины **n**)

DATE

- Хранит информацию о дате и времени
- Примеры форматирования дат (функции **TO_CHAR**, **TO_DATE**)

TIMESTAMP(n)

- Расширение **DATE** – хранит также доли секунд
- **n** $\in [1, 9]$ – количество цифр после запятой (точность).
Значение по умолчанию = 9.

Преобразование типов

Некоторые правила:

- Неявное преобразование в целевой тип при **INSERT** и **UPDATE**, вызове функций
- Операции над типом **NUMERIC** производятся с максимально возможной точностью
- При сравнении **CHAR*** vs **NUMERIC** значение **CHAR*** преобразуется в **NUMERIC**
- При сравнении **CHAR*** vs **DATE** значение **CHAR*** преобразуется в **DATE**
- В операциях присваивания выражение справа от = преобразуется в тип переменной слева от =

	CHAR	VARCHAR2	NCHAR	NVARCHAR2	DATE	NUMBER
CHAR		X	X	X	X	X
VARCHAR2	X		X	X	X	X
NCHAR	X	X		X	X	X
NVARCHAR2	X	X	X		X	X
DATE	X	X	X	X		--
NUMBER	X	X	X	X	--	

Функции конвертации типов

- `TO_NUMBER(e1[, format_model[, nls_params]])`
- Для `CHAR*`: `TO_CHAR(c)`
- Для `NUMBER`: `TO_CHAR(n[, format_model[, nls_params]])`
- Для `DATE`: `TO_CHAR(d[, format_model[, nls_params]])`
- `TO_DATE(c[, format_model[, nls_params]])`
- `TO_TIMESTAMP(c[, format_model[, nls_params]])`

Задание #8

Написать запрос, который...

1. создает новую таблицу, содержащую только те строки из таблицы клиентов, в каждой из которых нет значений **null**
2. создает представление на основе таблицы из п.1, включающее только столбцы **cname** и **city**
3. удаляет таблицу из п.1
4. пробует запросить содержимое из представления из п.2
5. выполнить п.1, затем снова п.4

Поддержка целостности данных

Ограничения (constraints): **UNIQUE**, **NOT NULL**, **CHECK**

Вспомним про свойство транзакций «Consistency»

Просмотр ограничений таблицы

Изменение ограничений

Первичные и внешние ключи

Inline- и out-of-line-синтаксис описания ограничений

Создание последовательностей (**SEQUENCE**)

Поддержка целостности данных

Ограничения (constraints): **UNIQUE**, **NOT NULL**, **CHECK**

Просмотр ограничений таблицы

Изменение ограничений

Первичные и внешние ключи

Inline- и out-of-line-синтаксис описания ограничений

СИНТАКСИС

	CREATE TABLE	ALTER TABLE
In-line unnamed	<pre>CREATE TABLE table_name (column_name datatype inline_constraint, ...);</pre>	<pre>ALTER TABLE table_name ADD MODIFY (column_name ... inline_constraint, ...);</pre>
In-line named	<pre>CREATE TABLE table_name (column_name datatype CONSTRAINT constraint_name inline_constraint, ...);</pre>	<pre>ALTER TABLE table_name ADD MODIFY (column_name ... CONSTRAINT constraint_name inline_constraint, ...);</pre>
Out-of-line	<pre>CREATE TABLE table_name (column_name datatype, ..., CONSTRAINT constraint_name outOfLine_constraint, ...);</pre>	<pre>ALTER TABLE table_name ADD MODIFY (CONSTRAINT constraint_name outOfLine_constraint, ...);</pre>

СИНТАКСИС

Type	<i>inline_constraint</i>	<i>outOfLine_constraint</i>
PRIMARY KEY	PRIMARY KEY	PRIMARY KEY (<i>column_list</i>)
FOREIGN KEY	REFERENCES <i>table_name</i> (<i>column_list</i>)	FOREIGN KEY (<i>column_list</i>) REFERENCES <i>table_name</i> (<i>column_list</i>)
UNIQUE	UNIQUE	UNIQUE (<i>column_list</i>)
CHECK	CHECK (<i>expression</i>)	CHECK (<i>expression</i>)
NOT NULL	NOT NULL	(*** Not Applicable ***)

UNIQUE

- Может быть применено к одному или более столбцам;
- Допускаются значения null;
- Создание – in-line либо out-of-line синтаксис;
- `ALTER TABLE table_name DROP UNIQUE (column1, column2, ...)`

NOT NULL

- Может быть применено только к одному столбцу;
- Не допускает значения null в столбце;
- Создание – только in-line синтаксис;
- ALTER TABLE table_name MODIFY column_name NOT NULL;
- ALTER TABLE table_name MODIFY column_name NULL;

CHECK

- Позволяет использовать сложные выражения для описания нетривиальных правил для добавляемых строк, например:
 - Хотя бы одна из n колонок is not null;
 - Значения в столбце должны быть больше x;
 - Значения в столбце должны быть из некоего предопределенного набора;
- Создание – in-line либо out-of-line синтаксис;
- ALTER TABLE table_name DROP CONSTRAINT name;
- ALTER TABLE table_name RENAME CONSTRAINT name TO new_name;

PRIMARY KEY

- Один или более столбцов, однозначно идентифицирующий каждую строку в таблице;
- В любой таблице может быть не более одного первичного ключа;
- PRIMARY KEY = UNIQUE + NOT NULL;
- Создание – in-line либо out-of-line синтаксис;
- ALTER TABLE table_name DROP CONSTRAINT name;
- ALTER TABLE table_name RENAME CONSTRAINT name TO new_name;

FOREIGN KEY

- Применяется к одному или нескольким столбцам в таблице;
- Поддерживает ссылочную целостность БД – гарантирует, что для каждого столбца (столбцов) таблицы существует соответствующий столбец (столбцы) в другой таблице;
- FOREIGN KEY одной таблицы может ссылаться только на PRIMARY KEY (или UNIQUE) другой таблицы.

SEQUENCE

- Объект в БД, который используется (в основном) для генерации значений первичных ключей;
- NEXTVAL – переводит сиквенс на следующее значение и возвращает это (новое) значение;
- Сиквенс передвинется на следующее значение даже в случае, если использовавший его запрос не выполнится (из-за ошибки);
- CURRVAL – возвращает текущее значение сиквенса. CURRVAL нельзя вызывать, если еще ни разу (в текущей сессии) не вызывался NEXTVAL.

Задание #9

(все названия каждый придумывает самостоятельно, т.е. у каждого должен быть свой объект БД)

1. Создать новую таблицу (таблица **C**), содержащую только уникальные строки таблицы **Clients**
2. Создать новую таблицу (таблица **S**), содержащую только уникальные строки таблицы **Sales**
3. Добавить PRIMARY KEY на столбец **snum** таблицы **S** (при необходимости удалить повторяющиеся **snum**ы и null-значения)
4. Добавить FOREIGN KEY на столбец **snum** таблицы **C** (при необходимости из **C** удалить строки с такими **snum**, которых нет в **S**)

Индексы

Неявное создание

Явное создание

Модификация и удаление

Индексы

Неявное создание

Явное создание

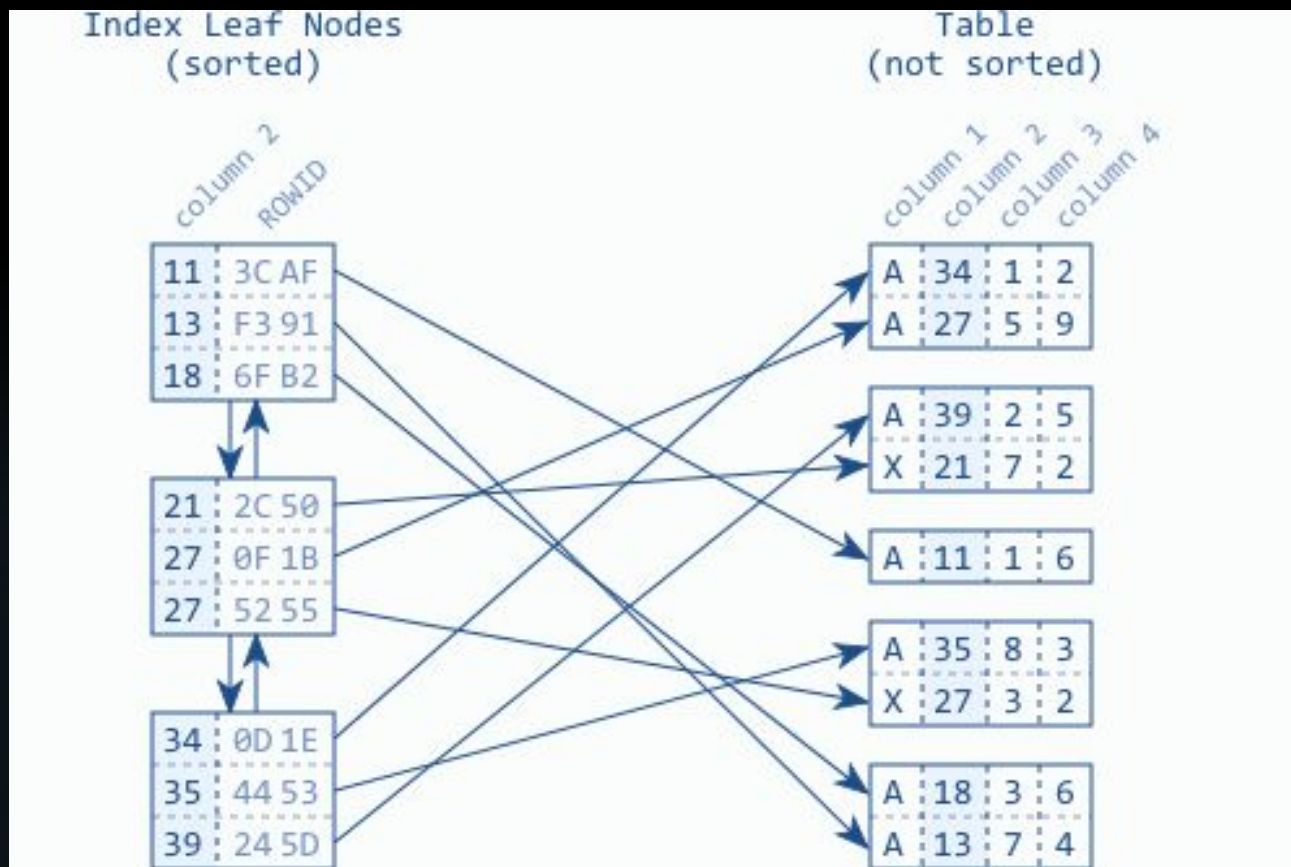
Модификация и удаление

Понятия **SELECTIVITY, CARDINALITY**

Индексы

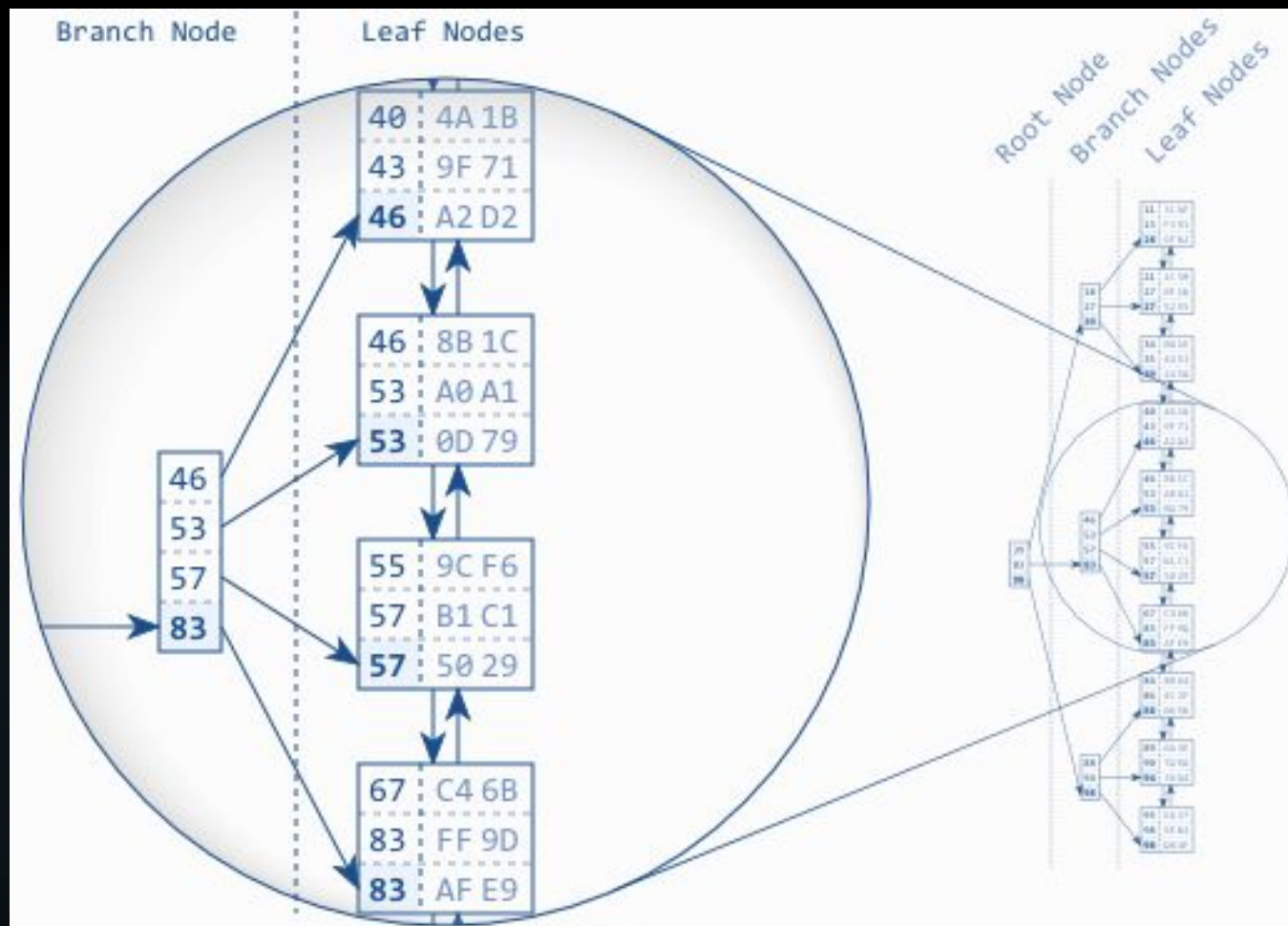
59

SQL Fundamentals, Performance Lab



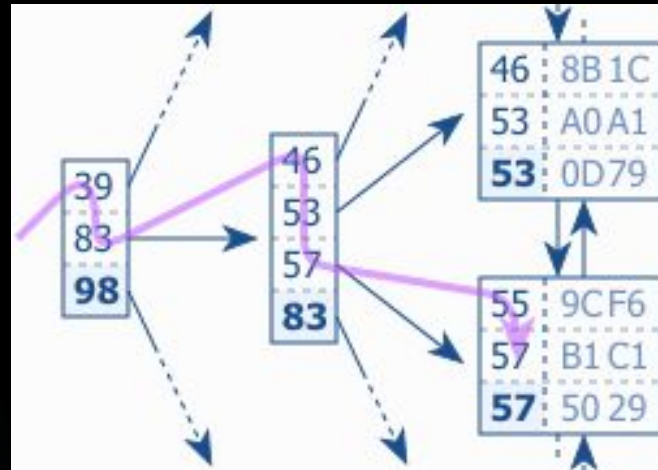
Индексы

60



Индексы

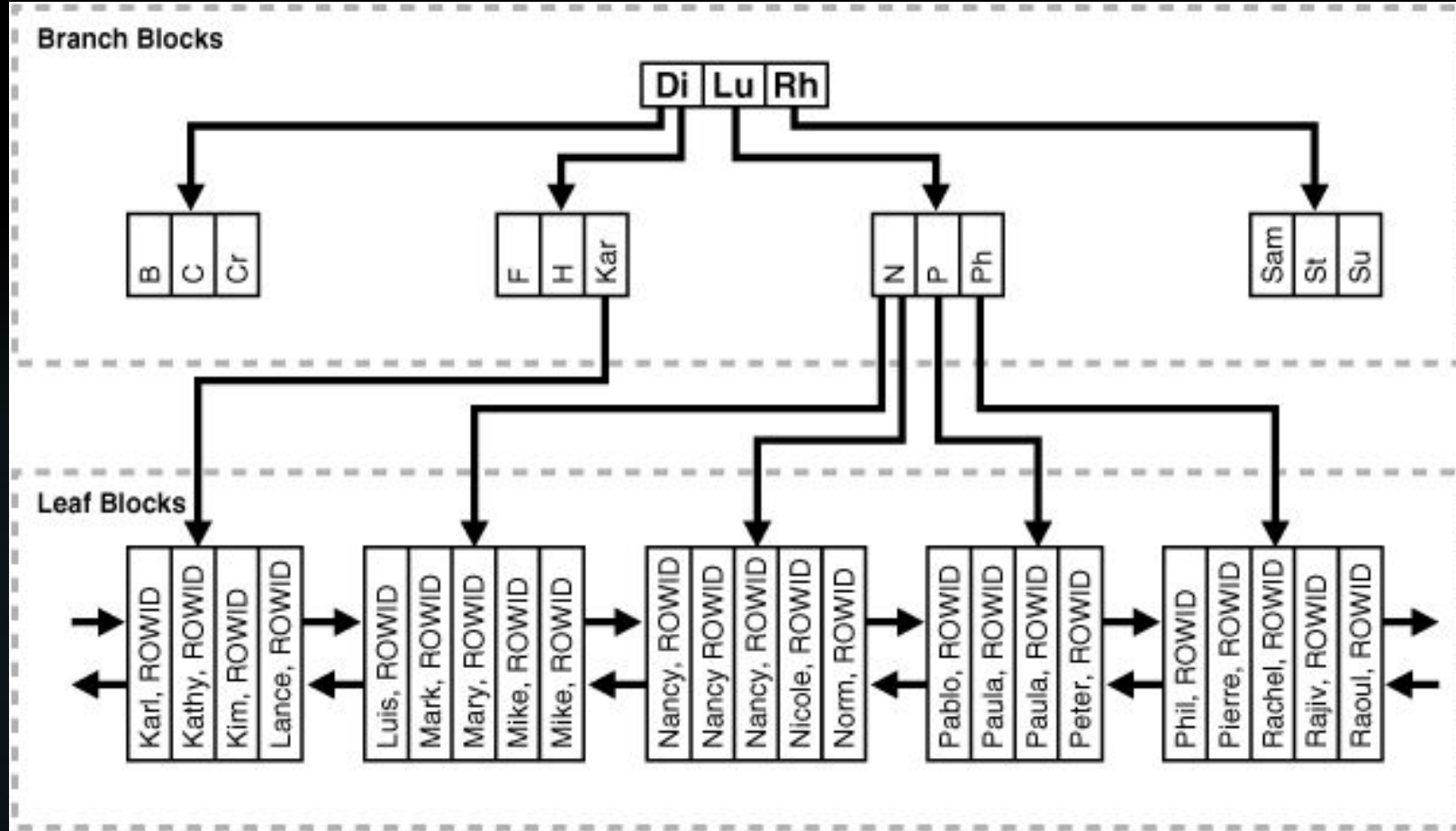
- Пример поиска по индексу: отбираем строки со значением 57 в индексированном столбце



B+ tree index

62

SQL Fundamentals, Performance Lab



Преимущества B+ tree index

- Высота дерева фиксированная \Rightarrow поиск любой записи в индексе занимает примерно одинаковое время
- Автоматически балансируются
- Ускоряют большинство запросов, включая запросы с предикатами, содержащими равенство либо диапазон значений
- Быстрые операции **UPDATE, INSERT, DELETE**
- Обеспечивают приемлемую производительность для таблиц больших и малых объемов
- Скорость поиска записей в индексе не падает с ростом числа записей (строк в таблице)

Composite Index

VENDOR_PARTS		
VEND ID	PART NO	UNIT COST
1012	10-440	.25
1012	10-441	.39
1012	457	4.95
1010	10-440	.27
1010	457	5.10
1220	08-300	1.33
1012	08-300	1.19
1292	457	5.28

Concatenated Index
(index with multiple columns)

Задание #10

65

Дополнительная информация

Теория:

1. <http://www.sql-tutorial.ru/>
2. <http://www.firststeps.ru/sql/oracle/oracle1.html>
3. <https://professorweb.ru/my/sql-server/2012/level1/>

Практика:

1. <http://sql-ex.ru/>

Книги:

1. Steve O'Hearn - OCA Oracle Database SQL Certified Expert Exam Guide