Курс «Базы данных»

Тема: Физическое проектирование БД. Таблицы

Барабанщиков
Игорь Витальевич

План лекции

1. **Основные задачи** физического проектирования БД
2. **Структуры хранения данных** (таблицы):

- Традиционные таблицы
- Индекс-таблицы
- Кластеры
- Секционированные таблицы
- Внешние таблицы

Этапы проектирования БД

Физическое проектирование БД

На этапе Концептуального и Логического проектирования определяется **«Что делать?»,** на этапе *Физического* проектирования – **«Как делать?».**

Физическое проектирование БД – это описание способа реализации логической модели БД.

Физическое проектирование выполняется для конкретной СУБД:

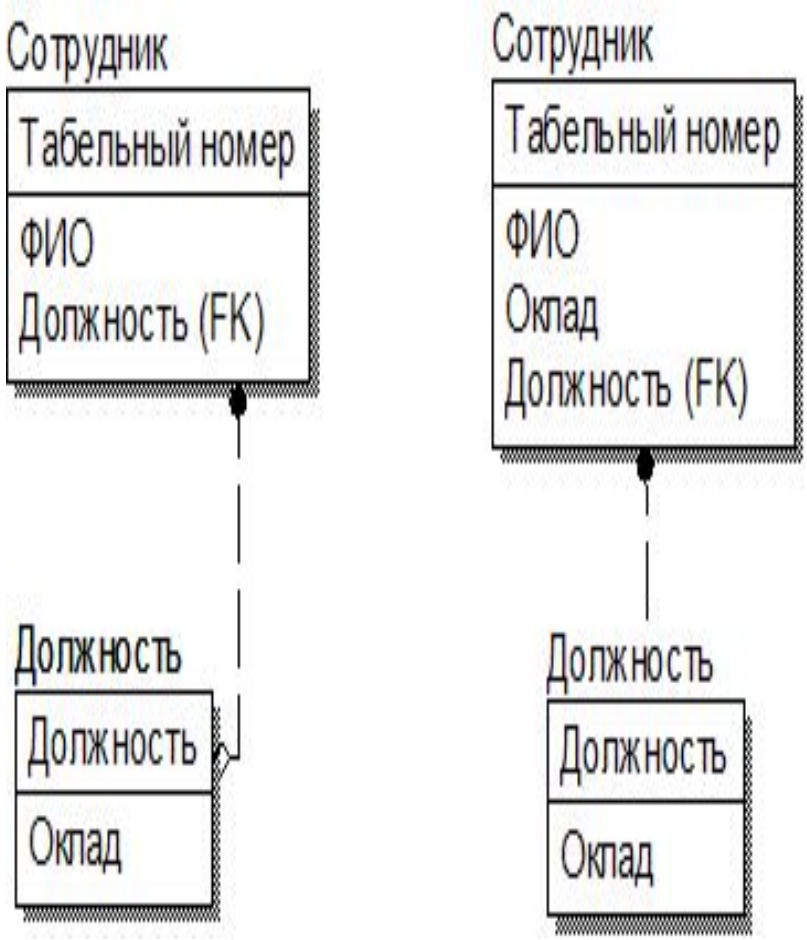
(Oracle, MS SQL Server, PostgreSQL, MySQL, DB2)

Задачи физического проектирования

- **Денормализация** БД
- **Выбор структур для хранения таблиц**
- **Выбор индексов**
- **Создание других объектов БД** (синонимы, последовательности, связи БД)
- **Проектирование транзакций**
- **Реализация бизнес-логики** (триггеры, хранимые процедуры)
- **Проектирование системы защиты** (роли и привилегии)

Денормализация отношений

- Иногда после нормализации отношений проводят их ***денормализацию***.
- Это может быть вызвано **необходимостью обеспечения более высокой скорости выполнения SQL-запросов.**



Денормализация таблиц

- В нормализованной БД одна сущность разбивается на несколько таблиц.
- Для получения исходного отношения надо выполнить **операцию соединения**.
- Операция соединения таблиц может занимать много времени, поэтому **нормализация может приводить к потере производительности БД**.

Виды денормализации

- **Восходящая** – перенос некоторой информации из подчиненного отношения в родительское.
- **Нисходящая** – информация переносится из родительского отношения в подчиненное.
- **Разбиение одного отношения на два** – когда таблица имеет много полей и некоторые из них (большие по размеру) редко используются, их можно выделить в отдельную таблицу.

Решение проблем денормализации

- Денормализация таблиц может привести к аномалиям обновления данных.
- В случае денормализации таблицы **надо принимать дополнительные меры для обеспечения целостности данных:**

  - триггеры

  - хранимые процедуры

Структуры хранения таблиц в БД Oracle

- **Традиционные таблицы** (Heap organized table)
- **Индекс-таблицы** (Index organized table - IOT)
- **Кластеры:**

  - хэш-кластеры

  - индекс-кластеры
- **Секционированные таблицы** – разбиение больших таблиц на несколько единиц (секций).
- **Внешние таблицы** – доступ к данным, хранящимся вне БД.
- **Вложенные таблицы** – дочерние таблицы.
- **Объектные таблицы** – создаются на основе объектного типа.

Традиционные таблицы

- Представляют собой «обычные» таблицы БД.
- **Данные в них распределяются подобно тому, как они распределяются в куче**.
- При добавлении данных для них используется первое обнаруженное в сегменте подходящее место.
- При удалении данных из такой таблицы, место, которое они занимали, становится доступным для повторного использования.
- Отсюда и название **Heap (куча) – это область, которая используется произвольным образом**.

Heap-organized table

Таблица, организованная в виде кучи, - это **неупорядоченный набор строк**.

Блок данных **heap-таблицы** содержит строки в *неупорядоченном* виде:

50, Shipping, 121, 1500

120, Treasury, , 1700

70, Public Relations, 204, 2700

30, Purchasing, 114, 1700

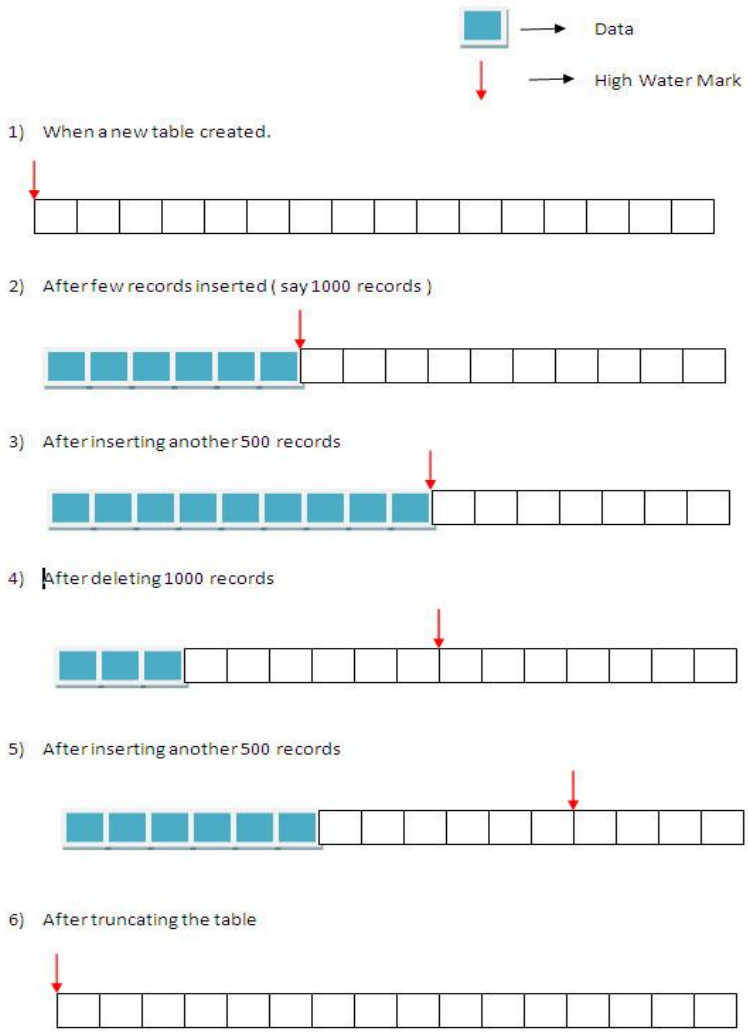
130, Corporate Tax, , 1700

10, Administration, 200, 1700

110, Accounting, 205, 1700

Полный просмотр таблицы

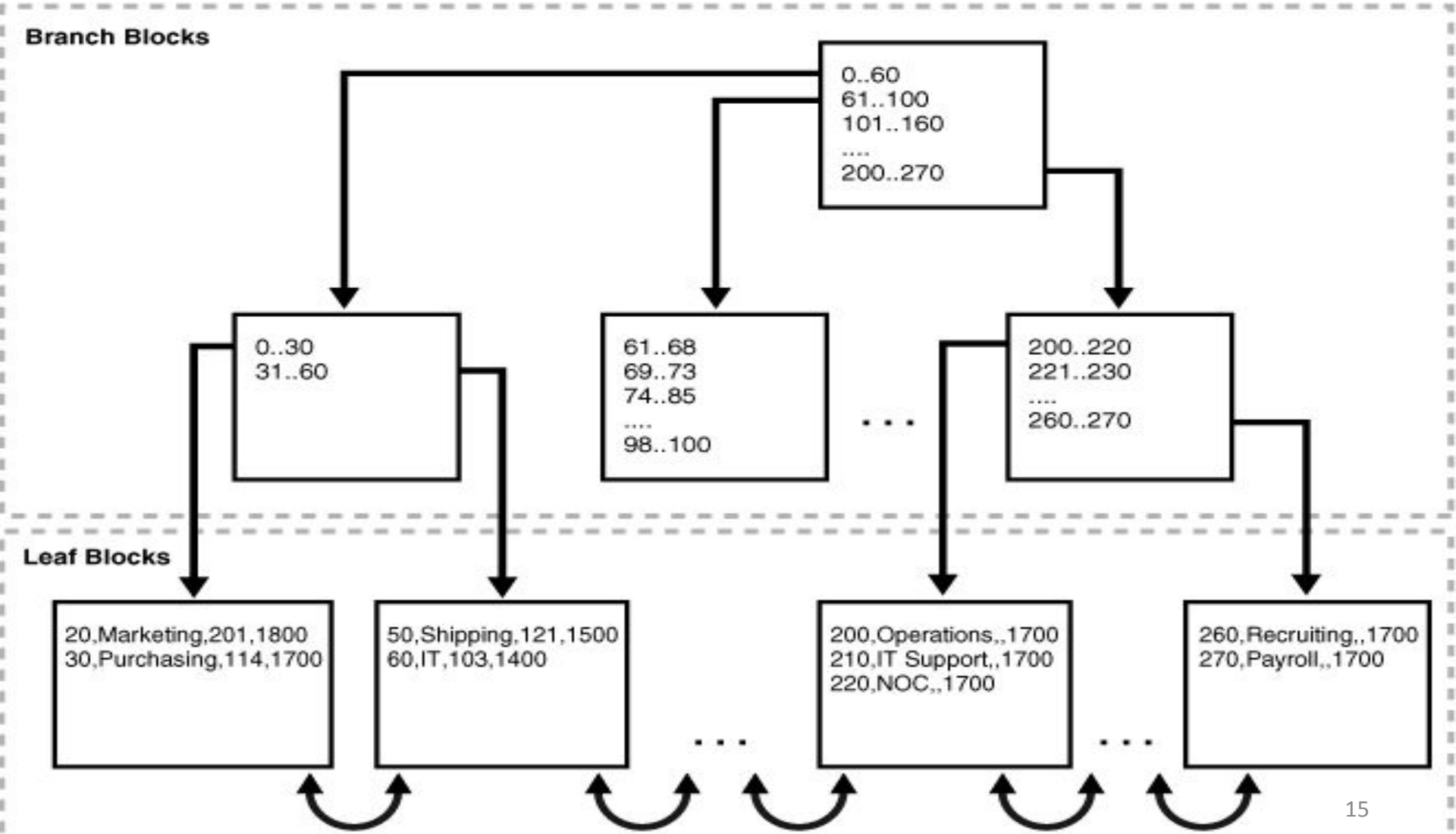
- Если **heap-organized table** не имеет индекса, то при поиске строки СУБД должна выполнять полный просмотр таблицы.
- При выполнении **полного просмотра** таблицы СУБД выполняет чтение ВСЕХ блоков сегмента ДО отметки HWM, включая те, которые не содержат



Индекс-таблицы

- Эти таблицы имеют структуру **B*Tree индекса**.
- Это накладывает определенный **физический порядок** на сами строки.
- Если в традиционных таблицах данные размещаются там, где они могут поместиться, то в индекс-таблицах данные сохраняются в определенном (отсортированном) порядке в соответствии с первичным ключом.
- **Используются с таблицами, которые редко обновляются.**
- **Для поиска данных по первичному ключу требуется меньше операций Ввода\Вывода.**

Индекс-таблицы



Примеры создания таблиц

```
CREATE TABLE emp
(id number(9) PRIMARY KEY,
 name varchar2(50)
 )
```

Обычная таблица (Heap):

- Создается сегмент данных для таблицы
- Создается сегмент данных для индекса
- Данные в таблице не упорядочены.

```
CREATE TABLE emp
(id number(9) PRIMARY KEY,
 name varchar2(50)
 )
ORGANIZATION INDEX
```

Индекс таблица (IOT):

- Создается один сегмент данных
- Данные отсортированы по ключу

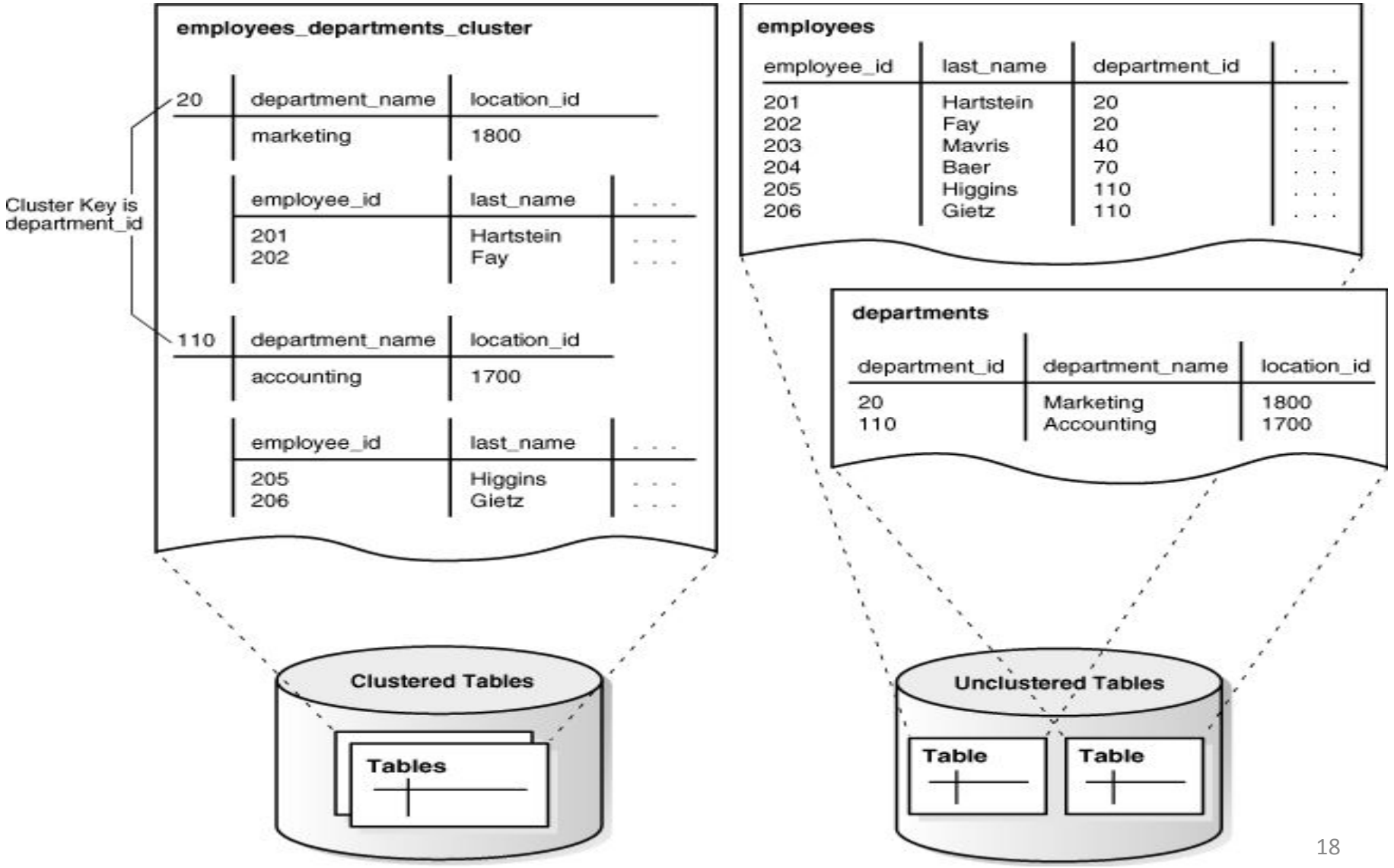
Кластерные таблицы

- **Кластеры** – это группы, состоящие из одной или более таблиц, которые *физически* хранятся в *одинаковых* блоках БД.
- **Все строки кластера, в которых используют одно и то же значение ключа кластера, находятся физически рядом друг с другом.**
- Данные как бы «кластеризуются» (собираются) вокруг значений ключа кластера.
- **Ключ кластера создается:**

  - с помощью B*Tree индекса (индексный кластер)

  - хешированием (хеш-кластер)

Кластерные таблицы



Секционирование

Секционирование – это способность БД разбивать большие таблицы и индексы на меньшие, более управляемые части.

Схемы секционирования таблиц:

- Основанное на диапазонах
- Основанное на случайном выборе (хеш)
- Основанное на списке
- Составное (гибридное)

Схемы секционирования индексов:

- Локально разделенный индекс
- Глобально разделенный индекс

Секционированные таблицы

Table Partition EVEN_CHANNELS

PROD_ID	CUST_ID	TIME_ID	CHANNEL_ID	PROMO_ID	QUANTITY_SOLD	AMOUNT_SOLD
116	11393	05-JUN-99	2	999	1	12.18
118	133	06-JUN-01	2	999	1	17.12
133	9450	01-DEC-00	2	999	1	31.28
30	170	23-FEB-01	2	999	1	8.8
24	11899	26-JUN-99	4	999	1	43.04
45	9491	28-AUG-98	4	350	1	47.45

Table Partition ODD_CHANNELS

PROD_ID	CUST_ID	TIME_ID	CHANNEL_ID	PROMO_ID	QUANTITY_SOLD	AMOUNT_SOLD
40	100530	30-NOV-98	9	33	1	44.99
36	4523	27-JAN-99	3	999	1	53.89
125	9417	04-FEB-98	3	999	1	16.86
35	2606	17-FEB-00	3	999	1	54.94

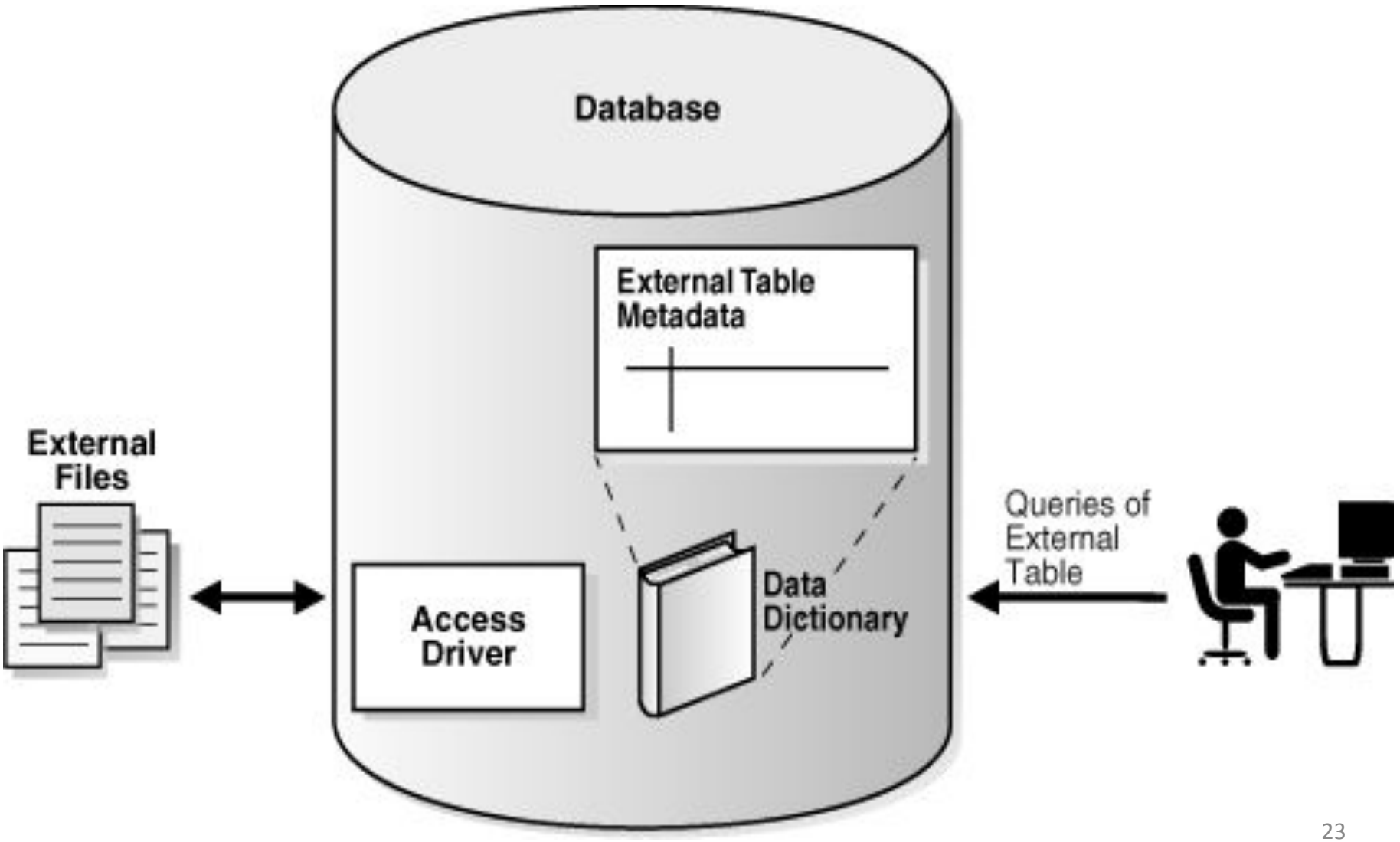
Секционированные таблицы

```
CREATE TABLE list_sales
( prod_id NUMBER(6) ,
  cust_id   NUMBER ,
  time_id   DATE ,
  channel_id CHAR(1) ,
  promo_id NUMBER(6) ,
  quantity_sold NUMBER(3) ,
  amount_sold NUMBER(10,2) )
PARTITION BY LIST (channel_id)
  (PARTITION even_channels VALUES (2,4),
   PARTITION odd_channels VALUES (3,9) );
```

Внешние таблицы

- Используются для таблиц **данные, которых хранятся за пределами БД Oracle**.
- Позволяют выполнять выборку данных из плоских файлов, из файлов с разделителями, из позиционных файлов фиксированной ширины.
- **Эти таблицы нельзя изменять**, к ним можно только направлять запросы.
- **Внешние таблицы предназначены для загрузки данных в БД.**

Внешние таблицы



Пример внешней таблицы

```
CREATE  TABLE  emp_external
   (emp_id   NUMBER(7),
    ename     VARCHAR2(20),
    hiredate  DATE )
ORGANIZATION  EXTERNAL
  ( type  oracle_loader
    default  directory  data_dir
    access  parameters
    ( fields  terminated by  ';'  )
    location ('emp.dat')
   )
```

Итоги

- При разработке эффективных приложений важным этапом является *физическое проектирование* БД.
- **СУБД Oracle предоставляет богатые возможности для выбора физических структур хранения таблиц.**