数据库原理

The Theory of Database System

第三章 关系数据库标准语言SQL

中国矿业大学计算机学院

本讲主要内容

SQL概述

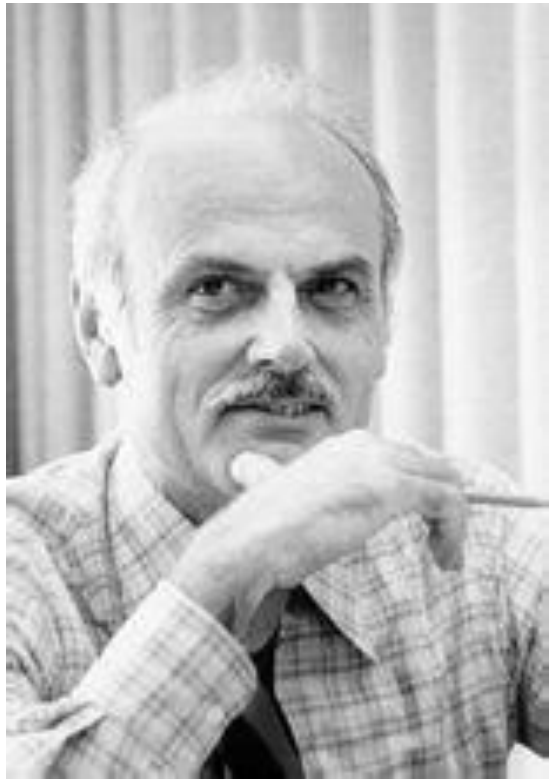
数据定义功能

3.1.1 SQL的发展

1974年由Boyce和Chamberlin提出，1975年～1979年IBM公司在System R原型系统上实现。

关系数据库的标准语言，是数据库领域中一个主流语言。

E.F.Codd



- 1970年，发表论文《大型共享数据库数据的关系模型》。
- 1981年，获图灵奖。

D.D.Chamberlin

- SQL的主要创造者之一。
- 1988年的“软件系统奖”授予了System R开发小组。

SQL的发展

- 1974年，由Boyce和Chamberlin提出SQL语言。
- 1979年，ORACLE公司和IBM公司提供了商用的SQL。
- 1986年，美国国家标准局（ANSI）采用SQL作为关系数据库管理系统的标准语言。
- 1987年，国际标准化组织（ISO）将其采纳为国际标准。

SQL标准的版本

年份	版本	
1986年	SQL86	
1989年	SQL89	
1992年	SQL92	SQL2
1999年	SQL99	SQL3
2003年	SQL:2003	
2008年	SQL:2008	
2011年	SQL:2011	
……	……	

3.1.2 SQL的特点

(1)数据描述、操纵、控制等功能一体化

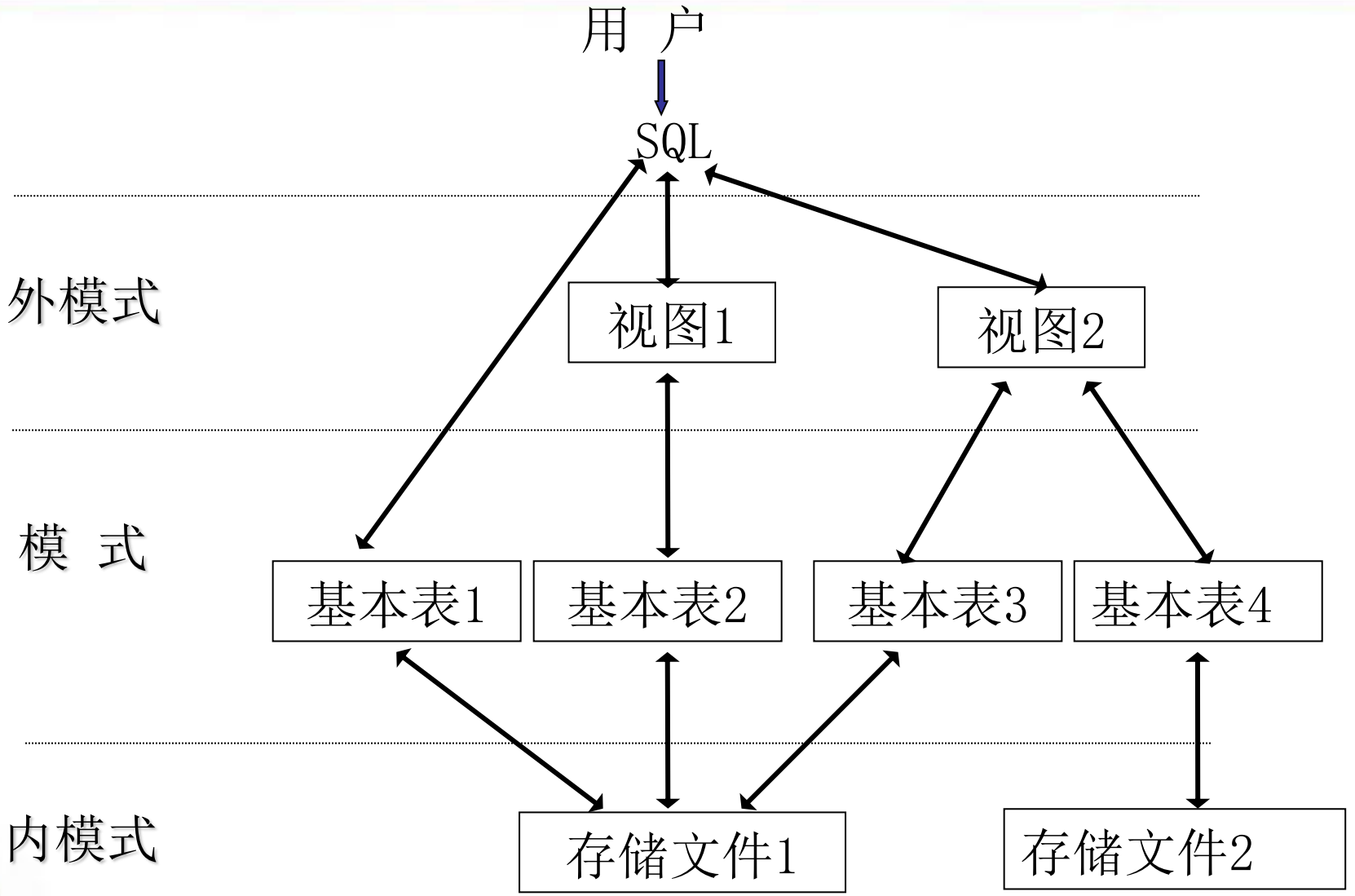
(2)两种使用方式，统一的语法结构

(3)高度非过程化

(4)语言简洁，易学易用

功　　能	动　　词
数据库查询	SELECT
数据定义	CREATE，DROP，ALTER
数据操纵	INSERT，UPDATE，DELETE
数据控制	GRANT，REVOKE

3.1.3 SQL体系结构



视图

- 视图是从一个或几个基本表（或视图）导出的表。
- 视图中不实际存储具体的数据。
- 视图是一个虚表(Virtual Table)或虚关系。

基本表

- 基本表（Base Table），是数据库中实际存在的关系。
- 基本表既包含结构的描述，也包含具体的数据。
- 基本表是一种实关系(Practical Relation)。

存储文件

- 存储文件：基本表在存储介质上的物理存储文件。
- 每个基表对应一个存储文件，一个基表还可以带一个或几个索引，存储文件和索引一起构成了关系数据库的内模式。

3.2 SQL的定义功能

SQL 的数据定义语句

操作对象	操作方式		
	创建	删除	修改
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	

基本表的定义

CREATE TABLE <表名>

（<列名> <数据类型>[<列级完整性约束条件>]

[，<列名> <数据类型>[<列级完整性约束条件>]]

……

[，<表级完整性约束条件>]

）；

[例]建立一个“学生”表，它由学号、姓名、性别、出生年份、籍贯和所在学院6个列组成，其中学号属性不能为空，并且其值是惟一的。

```
CREATE TABLE 学生
    ( 学号   CHAR(8)  NOT NULL UNIQUE,
      姓名   CHAR(8),
      性别   CHAR(2),
      出生年份   INT,
      籍贯   CHAR(8),
      学院   CHAR(15) );
```

学号	姓名	性别	出生年份	籍贯	学院

[例]建立一个“课程”表，它由课程号、课程名、学时、开课学期、课程性质5个属性组成，其中课程号不能为空，且取值唯一。

```
CREATE  TABLE  课程
    ( 课程号    CHAR(8)  NOT  NULL  UNIQUE,
      课程名    CHAR(15),
      学时      SMALLINT,
      开课学期  CHAR(4),
      课程性质  CHAR(15) );
```

常用数据类型

- 数据类型
  - 1)定长和变长字符串CHAR(n)　VARCHAR(max)
  - 2)定长和变长二进制串
    BINARY(n)　VARBINARY(max)
  - 3)整型数　INT　SMALLINT
  - 4)浮点数　FLOAT　DOUBLE
  - 5)日期型　DATE
  - 6)时间型　TIME
  - 7)时标　TIMESTAMP

完整性约束

- 常用完整性约束
  - 实体完整性

    主码约束：PRIMARY KEY
  - 参照完整性

    外码约束：FOREIGN KEY
  - 用户自定义完整性

    唯一性约束：UNIQUE

    非空值约束：NOT NULL

    ……

实体完整性约束

为学生表和课程表添加主码：

```
CREATE  TABLE  学生
    ( 学号   CHAR(8)  PRIMARY  KEY ，
      姓名   CHAR(8)，
      性别   CHAR(2)，
      出生年份   INT，
      籍贯   CHAR(8)，
      学院   CHAR(15) );
```

实体完整性约束

```
CREATE  TABLE  课程
    ( 课程号    CHAR(8)  PRIMARY  KEY,
      课程名    CHAR(15),
      学时      SMALLINT,
      开课学期  CHAR(4),
      课程性质  CHAR(15) );
```

实体完整性约束

[例]建立一个“学习”表，它由学号、课程号、成绩3个属性组成，其中学号和课程号为主关键字。

```
CREATE  TABLE  学习
    ( 学号  CHAR(8),
      课程号 CHAR(8),
      成绩   SMALLINT,
      PRIMARY  KEY (学号, 课程号));
```

主关键字的定义

1）在列出关系模式的属性时,在属性及其类型后加上保留字PRIMARY KEY；

2）在列出关系模式的所有属性后，再附加一个声明：

PRIMARY KEY （<属性1>[，<属性2>，...]）

说明：如果关键字由多个属性构成，则必须使用第二种方法。

外部关键字的定义

1）如果外部关键字只有一个属性，可以在它的属性名和类型后面直接用“REFERENCES”说明它参照了某个表的某些属性，其格式为：

REFERENCES <表名>(<属性>)

2）在CREATE TABLE语句的属性列表后面增加一个或几个外部关键字说明，其格式为：

FOREIGN KEY (<属性>) REFERENCES <表名>(<属性>)

参照完整性约束

为学生选课表建立外码：

```
CREATE  TABLE  学习
    ( 学号   CHAR(8) REFERENCES 学生(学号) ,
      课程号 CHAR(8) REFERENCES 课程(课程号),
      成绩   SMALLINT,
      PRIMARY  KEY (学号, 课程号));
```

参照完整性约束

```
CREATE TABLE 学习
    ( 学号 CHAR(8) ,
      课程号 CHAR(8) ,
      成绩 SMALLINT,
      PRIMARY KEY (学号, 课程号),
    FOREIGN KEY (学号) REFERENCES 学生(学号),
    FOREIGN KEY (课程号) REFERENCES 课程(课程号));
```

用户自定义完整性约束

- 列值的约束

```
CREATE TABLE 学生
    ( 学号   CHAR(8) PRIMARY KEY ,
      姓名   CHAR(8),
      性别   CHAR(2)
          CHECK(性别= ‘男’ OR 性别= ‘女’ ),
      出生年份   INT,
      籍贯   CHAR(8),
      学院   CHAR(15) );
```

用户自定义完整性约束

- **默认值约束**

默认值是指提前给某列指定的取值。

```
CREATE TABLE 学生
    ( 学号   CHAR(8) PRIMARY KEY ,
      姓名   CHAR(8),
      性别   CHAR(2) DEFAULT '男',
      出生年份   INT,
      籍贯   CHAR(8),
      学院   CHAR(15) );
```

```
mysql> desc 学生;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| 学号         | char(8)  | NO   | PRI | NULL    |       |
| 姓名         | char(8)  | YES  |     | NULL    |       |
| 性别         | char(2)  | YES  |     | 男      |       |
| 籍贯         | char(8)  | YES  |     | NULL    |       |
| 出生年份     | int(11)  | YES  |     | NULL    |       |
| 学院         | char(15) | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

3.2.2 基本表的修改和删除

```
CREATE  TABLE  学生
(  学号          CHAR(8)        PRIMARY  KEY ,
   姓名          CHAR(8),
   性别          CHAR(2)        DEFAULT '男' ,
   出生年份      INT,
   籍贯          CHAR(8),
   学院          CHAR(15)
);
```

基本表的修改

ALTER TABLE <表名>
[ADD <新列名> <数据类型> [完整性约束]]
[DROP COLUMN<列名>|<完整性约束名>]
[ALTER COLUMN <列名> <数据类型>];

- <表名>：要修改的基本表
- ADD子句：增加新列和新的完整性约束条件
- DROP子句：删除指定的列或完整性约束条件
- ALTER子句：用于修改列名和数据类型

增加

- 增加属性列

  [例] 向学生表增加“入学日期”列，其数据类型为日期型。

  ALTER TABLE 学生 ADD 入学日期 DATE;

- 不论基本表中原来是否已有数据，新增加的列一律为空值。

```
mysql> desc 学生;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| 学号         | char(8)  | NO   | PRI | NULL    |       |
| 姓名         | char(8)  | YES  |     | NULL    |       |
| 性别         | char(2)  | YES  |     | NULL    |       |
| 出生年份     | int(11)  | YES  |     | NULL    |       |
| 籍贯         | char(8)  | YES  |     | NULL    |       |
| 学院         | char(15) | YES  |     | NULL    |       |
| 入学日期     | date     | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

增加

- 增加约束

可以是主码、外码、取值唯一等常用的列级约束。

[例]假设学生表没有指定主码，现为学生表增加主码约束。

ALTER TABLE 学生 ADD PRIMARY KEY(学号);。

[例]为姓名列增加取值唯一的约束。

ALTER TABLE 学生 ADD UNIQUE(姓名);

增加

- 增加约束

可以是主码、外码、取值唯一等常用的列级约束。

[例]为学习表SC的学号列增加一个外码约束。

```
ALTER TABLE 学习 ADD FOREIGN KEY(学号)
        REFERENCES 学生(学号);
```

说明：前提要求学生表中已经指定学号为主码。

删除

- 删除属性列

[例] 删除学生表中“入学日期”属性列。

ALTER TABLE 学生 DROP 入学日期;

注意：若一个属性被说明为NOT NULL，则不允许修改或删除。

```
mysql> alter table 学生 drop 入学日期;
Query OK, 0 rows affected (0.94 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc 学生;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| 学号         | char(8)  | NO   | PRI | NULL    |       |
| 姓名         | char(8)  | YES  |     | NULL    |       |
| 性别         | char(2)  | YES  |     | NULL    |       |
| 出生年份     | int(11)  | YES  |     | NULL    |       |
| 籍贯         | char(8)  | YES  |     | NULL    |       |
| 学院         | char(15) | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

删除

- 删除约束

【例】删除学生表中的主码约束

```
ALTER TABLE 学生 DROP PRIMARY KEY;
```

思考：如果一个表上有多个同类型的约束，如何来确定要删除的对象呢？

删除

- 删除约束

方法一：通过系统指定的名称

SHOW CREATE TABLE 学习;

```
| 学习   | CREATE TABLE `学习` (
  `学号` char(8) NOT NULL,
  `课程号` char(8) NOT NULL,
  `成绩` smallint(6) DEFAULT NULL,
  PRIMARY KEY (`学号`,`课程号`),
  KEY `课程号` (`课程号`),
  CONSTRAINT `学习_ibfk_1` FOREIGN KEY (`学号`) REFERENCES `学生` (`学号`),
  CONSTRAINT `学习_ibfk_2` FOREIGN KEY (`课程号`) REFERENCES `课程` (`课程号`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

ALTER TABLE 学习 DROP FOREIGN KEY 学习_ibfk_1;

删除

方法二：

- 给约束取名称

```
ALTER TABLE 学习 ADD CONSTRAINT fk_sno
FOREIGN KEY(学号) REFERENCES 学生(学号);
```

- 通过约束名删除

```
ALTER TABLE 学习 DROP FOREIGN KEY fk_sno;
```

修改

- 修改属性列的类型

[例] 将入学年份的数据类型改为半字长整数。

ALTER TABLE 学生 MODIFY 出生年份 SMALLINT;

- 注：修改原有的列定义有可能会破坏已有数据

修改

- 修改属性列的类型

【例】假设简历表中已经存在一列编号，类型为INT，将编号的数据类型改为自动增长型。

```
ALTER TABLE 简历 MODIFY 编号 AUTO_INCREMENT;
```

- 注：如果简历表原先有数据，那么修改后会从下一条记录开始自动增长。

删除基本表

语法：

DROP TABLE <表名>[RESTRICT|CASCADE];

- RESTRICT：受限删除
- CASCADE: 级联删除

[例] 删除学生表

DROP TABLE S;

3.2.3 索引的建立与删除

搜索码

零件名	存放位置
	1层1区3-4-17
……	……

搜索码：用于在文件中查找记录的属性或属性集。

索引文件

索引文件

091501	
091502	
091503	
091504	
091505	
091506	
…	…

磁盘文件

091505	徐颖	女	江苏	1997	外文
091503	张小飞	男	江西	1996	计算机
…	…	…	…	…	…
091501	王英	女	河北	1997	计算机
091502	王小梅	女	江苏	2000	信电
…	…	…	…	…	…
091504	孙志鹏	男	海南	1998	计算机
091506	钱易蒙	男	河北	2000	信电
…	…	…	…	…	…

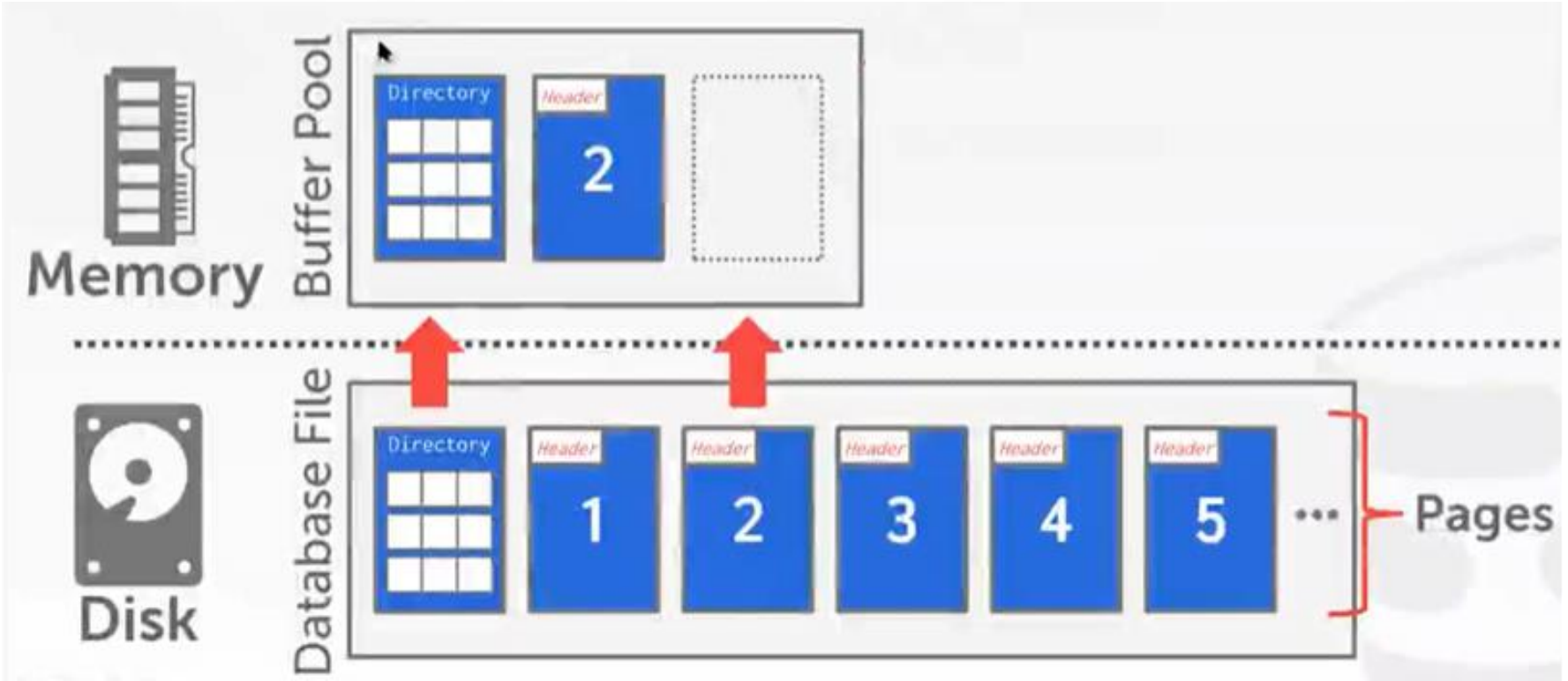
索引的概念

索引是根据关系（表）中某些字段的值，按照一定结构存放的文件。

搜索码

B-树、B^+树、散列桶……

存储文件

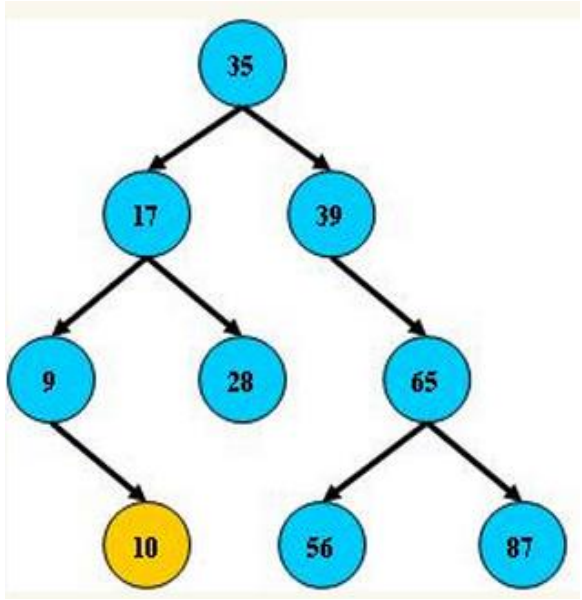


索引的作用

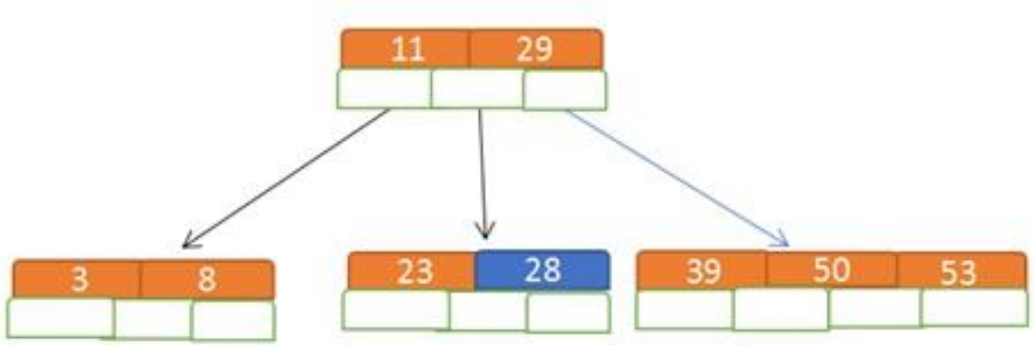
帮助数据库引擎使用最小的资源，最高效的找到需要的数据。

索引文件的物理结构

平衡二叉树

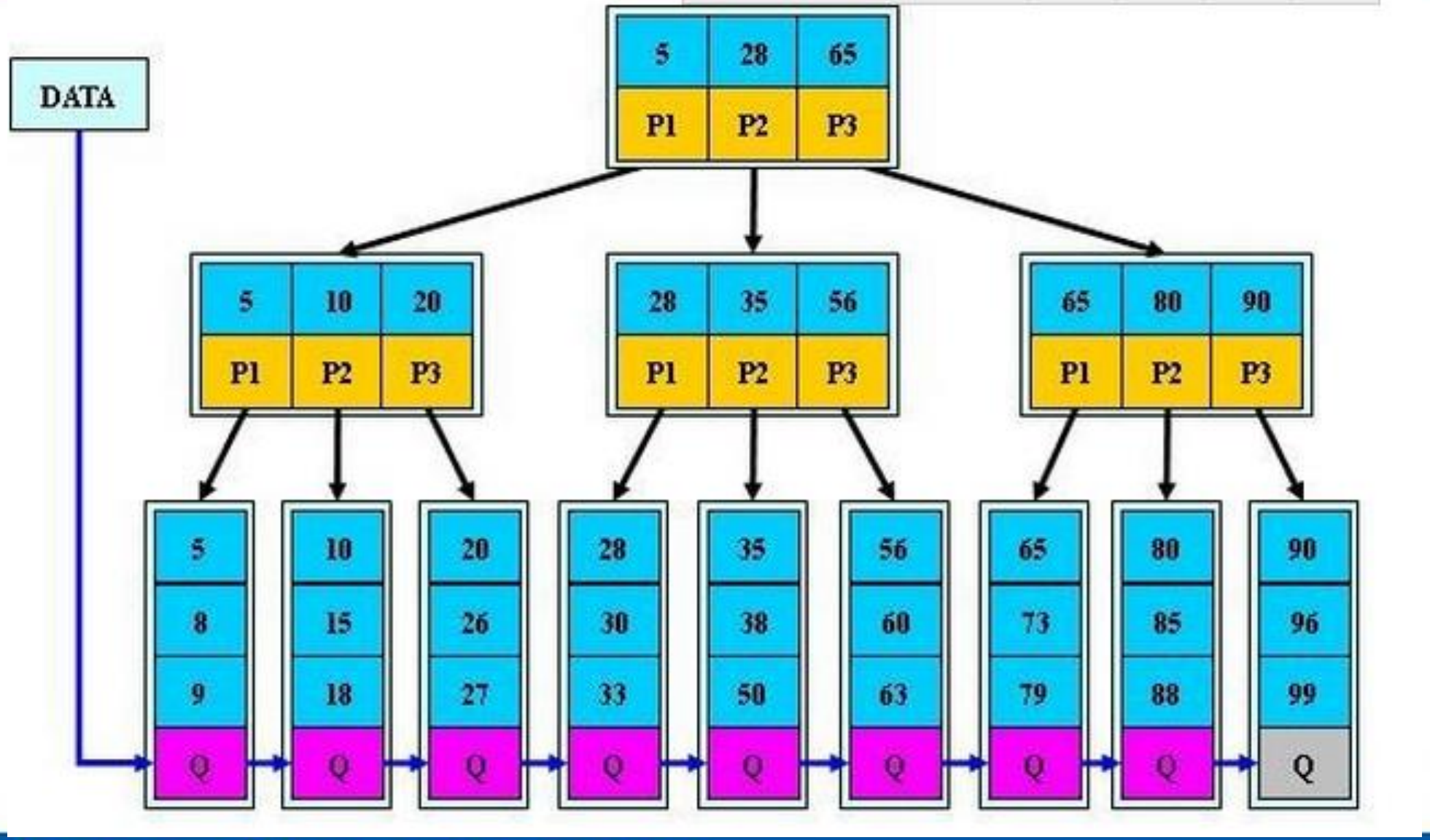


B-树

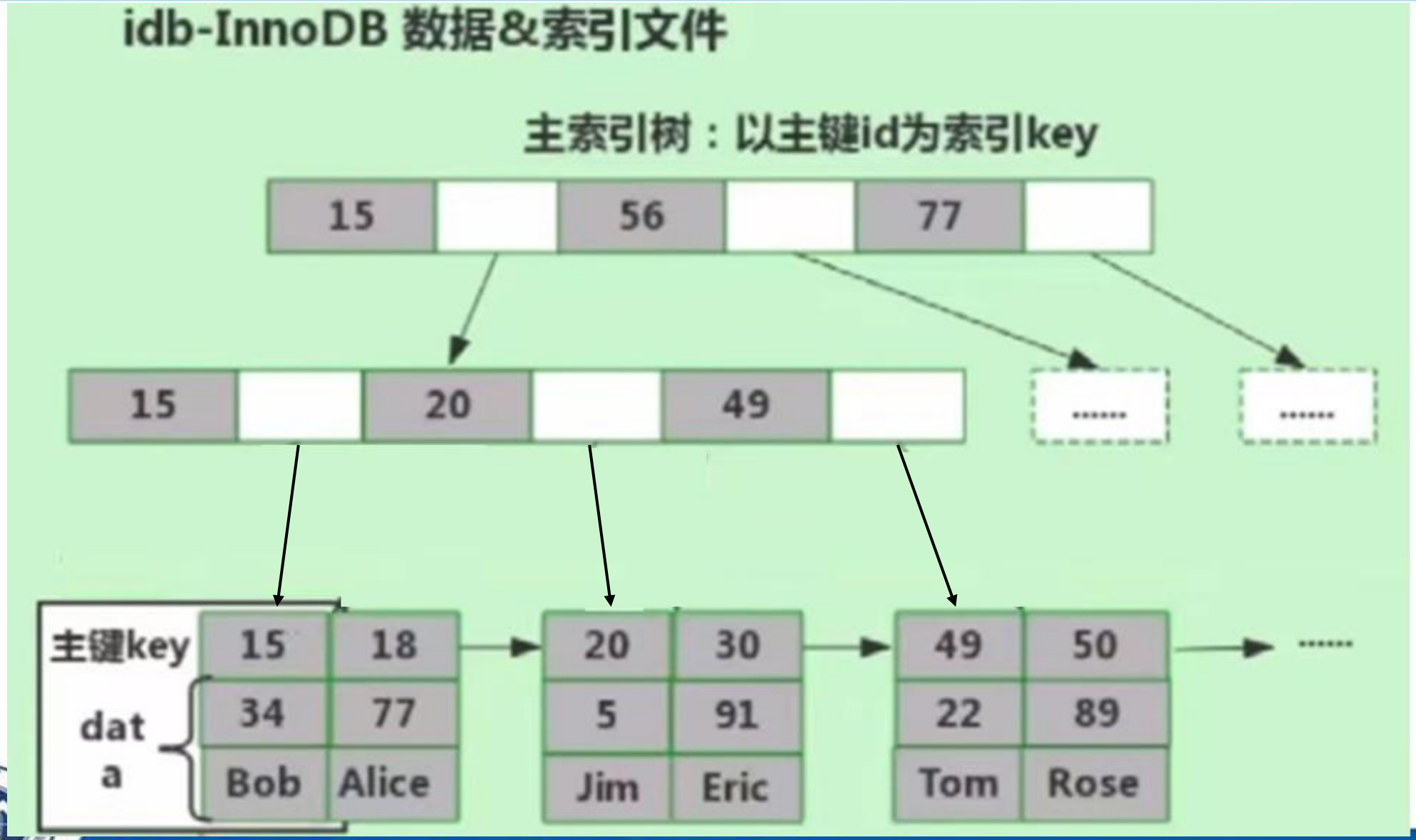


索引文件的物理结构

B+树



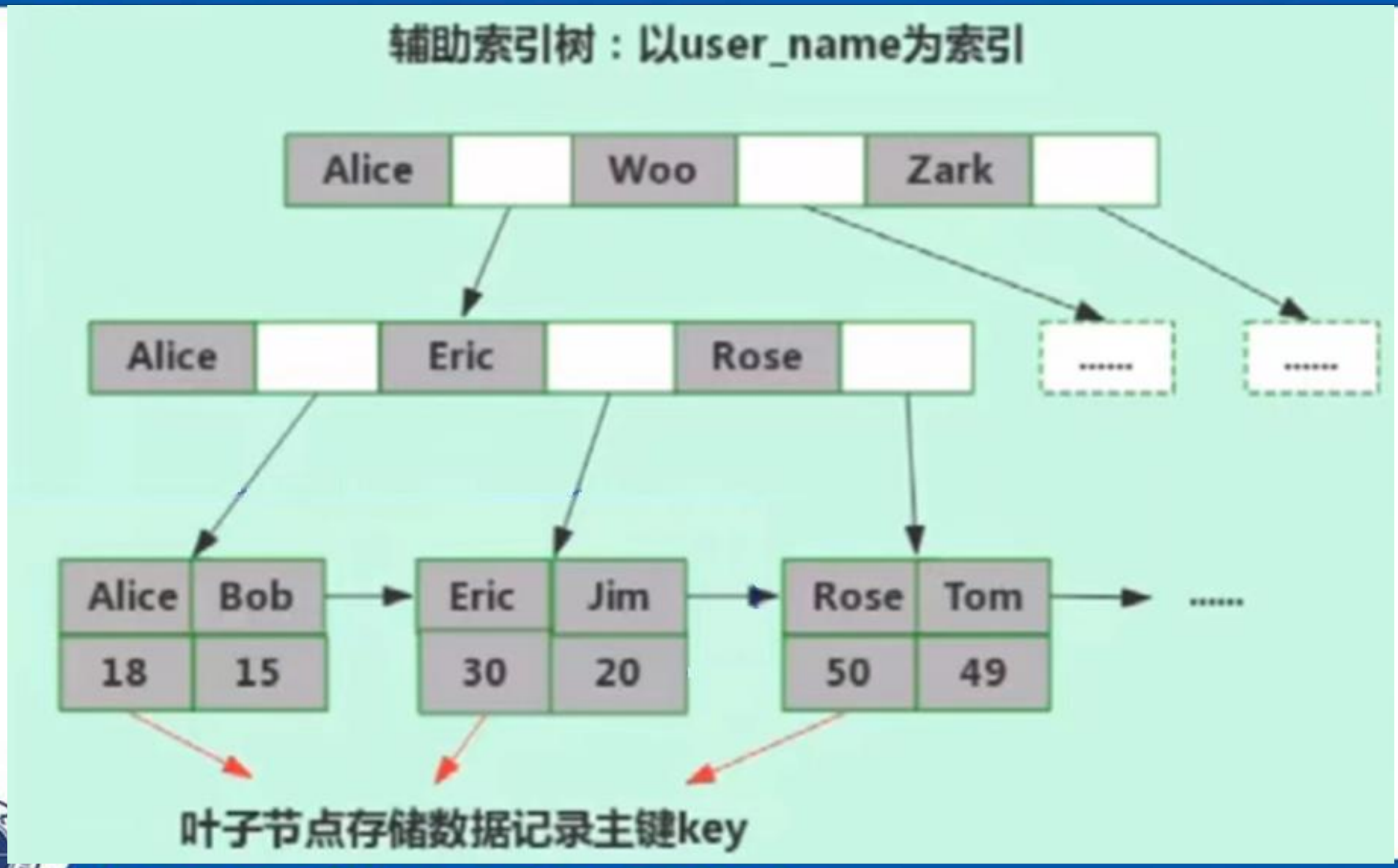
B+树索引文件


idb-InnoDB 数据&索引文件
主索引树：以主键id为索引key
15
56
77
15
20
49
.....
.....
主键key
15
18
20
30
49
50
.....
data
34
77
5
91
22
89
Bob
Alice
Jim
Eric
Tom
Rose

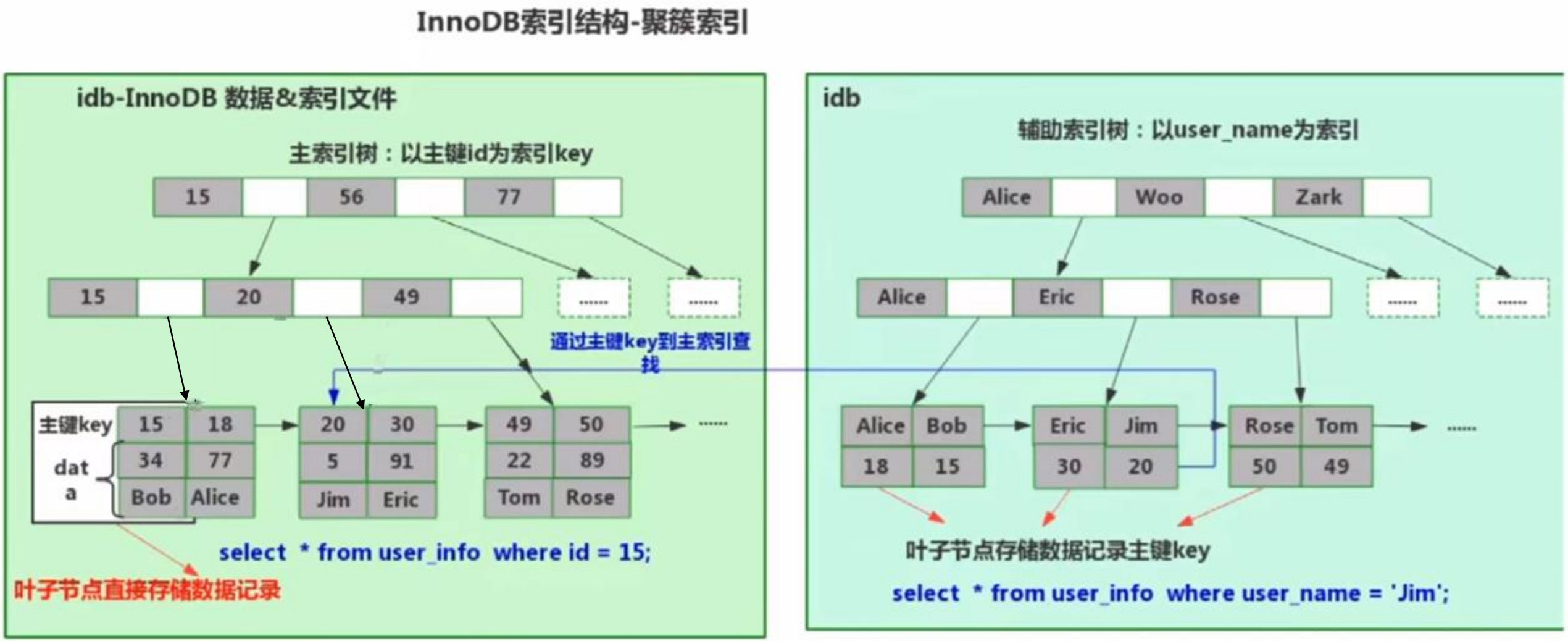
索引的分类

- 聚集索引
  - 表中的记录按照某个搜索码指定的顺序排序，那么该搜索码对应的索引称为聚集索引。聚集索引也称为主索引。
  - **每个表只能有一个聚集索引。**
- 非聚集索引
  - 搜索码指定的顺序与表中记录的物理顺序不同的索引称为非聚集索引。
  - 每个表中可以有多个非聚集索引。

非聚集索引


辅助索引树：以user_name为索引
Alice
Woo
Zark
Alice
Eric
Rose
.....
.....
Alice
Bob
18
15
Eric
Jim
30
20
Rose
Tom
50
49
.....
叶子节点存储数据记录主键key

聚簇索引与非聚簇索引


InnoDB索引结构-聚簇索引
idb-InnoDB 数据&索引文件
主索引树：以主键id为索引key
15
56
77
15
20
49
.....
.....
通过主键key到主索引查找
主键key
dat a
15
18
34
77
Bob
Alice
20
30
5
91
Jim
Eric
49
50
22
89
Tom
Rose
.....
select * from user_info where id = 15;
叶子节点直接存储数据记录
idb
辅助索引树：以user_name为索引
Alice
Woo
Zark
Alice
Eric
Rose
.....
.....
Alice
Bob
18
15
Eric
Jim
30
20
Rose
Tom
50
49
.....
叶子节点存储数据记录主键key
select * from user_info where user_name = 'Jim';

索引的说明

说明：

- 索引中的搜索码取值可以是唯一，也可以不唯一，即多行记录共享同一键值。
- 在创建 PRIMARY KEY 约束时，如果该表不存在聚集索引且未指定唯一非聚集索引，则将自动对一列或多列创建唯一聚集索引。
- 在创建 UNIQUE 约束时，默认情况下将创建唯一非聚集索引，以便强制 UNIQUE 约束。如果不存在该表的聚集索引，则可以指定唯一聚集索引。

索引的定义

- 语句格式

  CREATE [UNIQUE] [CLUSTER] INDEX <索引名>　　ON <表名>(<列名>[<次序>][,<列名>[<次序>]]…);

  - 用<表名>指定要建索引的基本表名字
  - 索引可以建立在该表的一列或多列上，各列名之间用逗号分隔
  - 用<次序>指定索引值的排列次序，升序：ASC，降序：DESC。缺省值：ASC
  - UNIQUE表明此索引的每一个索引值只对应唯一的数据记录
  - CLUSTER表示要建立的索引是聚集索引

索引的定义

[例]为学生-课程数据库中的学生，课程和学习三个表建立索引。其中学生表按学号升序建唯一索引，课程表按课程号升序建唯一索引，学习表按学号升序和课程号降序建唯一索引。

CREATE UNIQUE INDEX Stusno ON 学生 (学号);

CREATE UNIQUE INDEX Coucno ON 课程 (课程号);

CREATE UNIQUE INDEX SCno ON 学习(学号 ASC，课程号 DESC);

索引的定义

- 唯一值索引
  - 对于已含重复值的属性列不能建UNIQUE索引
  - 对某个列建立UNIQUE索引后，插入新记录时DBMS会自动检查新记录在该列上是否取了重复值。这相当于增加了一个UNIQUE约束。

索引的定义

- 聚集索引
  - 建立聚集索引后，基表中数据会按指定的属性值升序或降序存放。

例：在学生表的姓名列上建立一个聚集索引，而且学生表中的记录将按照姓名升序存放。

```
CREATE CLUSTER INDEX Stusname ON
 学生(姓名 ASC);
```

索引的删除

ALTER TABLE 表名 DROP INDEX <索引名>;

- 删除索引时，系统会从数据字典中删去有关该索引的描述。

[例] 删除学生表的Stusname索引。

ALTER TABLE 学生 DROP INDEX Stusname;

使用索引的技巧

- 对于常用的小型表来说，使用索引不会使性能有任何提高。
- 不要在memo、note型字段或者大型字段上创建索引。
- 不要对经常需要更新或修改的字段创建索引。
- 索引列中有较多不同的数据时索引会使性能有极大的提高。
- 当查询要返回的数据很少时，索引可以优化查询。
- 索引可以提高数据的返回速度，但是它使得数据的更新操作变慢。