Database System

北京交通大学软件学院

王方石 教授

E-mail: fshwang@bjtu.edu.cn

Contents

Chapter 3-Contents

Chapter 3-1 - Objectives

- **Data types supported by SQL standard.**
- **How to define and modify the structure of Table and View using SQL**
- **Purpose of integrity enhancement feature of SQL.**
- **How to define integrity constraints using SQL.**
- **How to use the integrity enhancement feature in the CREATE and ALTER TABLE statements.**

3.1 Introduction to SQL

3.1.1 purpose and Characteristics of SQL

1. Objectives of SQL:

1）A database language should allow a user to:

- ◆create the database and relation structures;
- ◆perform basic data management tasks, such as the insertion, modification, and deletion of data from the relations;
- ◆perform both simple and complex queries.

1. Objectives of SQL:

2) Must perform these tasks with minimal user effort and command structure/syntax must be easy to learn.

3) It must be portable.

Objectives of SQL

- **SQL is a transform-oriented language with 2 major components:**
  - **A DDL for defining database structure.**
  - **A DML for retrieving and updating data.**
- **Until SQL3, SQL did not contain flow of control commands. These had to be implemented using a programming or job-control language, or interactively by the decisions of user.**

Objectives of SQL

- **Can be used by range of users including DBAs, management, application developers, and other types of end users.**

2. Characteristics of SQL

(1) SQL is a comprehensive and unified language with 3 major components:

1) A **Data Definition Language (DDL)** for defining database structure.
   - **Create, alter, and drop** the database, relation and view structures;
2) A **Data Manipulation Language (DML)** for manipulating data.
   - **insert, delete, update** and **retrieve** the data in DB.

2. Characteristics of SQL

3) A Data Control Language (DCL) DCL for security, integrity, transaction control

- Grant or Revoke the privilege to the users
- ensure entity integrity, referential integrity and user-defined integrity
- Ensure the correct schedule for concurrent transactions.

3.1.1 Characteristics of SQL

(2) SQL is relatively easy to learn:

- **It is essentially free-format.**
- **SQL Consists of standard English words.**

```
1) CREATE TABLE Staff (staffNo VARCHAR(5),
                       lName VARCHAR(15),
                       salary DECIMAL(7,2));

2) INSERT INTO Staff
   VALUES ('SG16', 'Brown', 8300);

3) SELECT staffNo, lName, salary
   FROM Staff
   WHERE salary > 10000;
```

3.1.1 Characteristics of SQL

(2) SQL is relatively easy to learn:

- **There are only 9 core command words.**

functions	command words
Data query	SELECT
Data definition	CREATE, DROP, ALTER
Data manipulation	INSERT, UPDATE, DELETE
Data control	GRANT, REVOKE

3.1.1 Characteristics of SQL

(3) SQL is **non-procedural** - you specify *what* information you require, rather than *how* to get

(4) **You could use SQL in two ways: Embedded SQL or Interactive SQL.**

(5) It must be **portable**（可移植的）.

An ISO standard now exists for SQL, making it both the **formal and *de facto*** standard language for relational databases.

3.1.2 History of SQL

- **In 1974, D. Chamberlin (IBM San Jose Laboratory) defined language called ‘Structured English Query Language’ (SEQUEL).**
- **A revised version, SEQUEL/2, was defined in 1976 but name was subsequently changed to SQL for legal reasons.**

History of SQL

- **Still pronounced ‘see-quel’, though official pronunciation is ‘S-Q-L’.**
- **IBM subsequently produced a prototype DBMS called *System R*, based on SEQUEL/2.**
- **Roots of SQL, however, are in SQUARE (Specifying Queries as Relational Expressions), which predates System R project.**

History of SQL

- **In late 70s, ORACLE appeared and was probably first commercial RDBMS based on SQL.**
- **In 1987, ANSI and ISO published an initial standard for SQL.**
- **In 1989, ISO published an addendum that defined an 'Integrity Enhancement Feature'.**
- **In 1992, first major revision to ISO standard occurred, referred to as SQL2 or SQL/92.**
- **In 1999, SQL3 was released with support for object-oriented data management.**

3.1.3 Importance of SQL

- **SQL has become part of application architectures such as IBM's Systems Application Architecture.**
- **It is strategic choice of many large and influential organizations (e.g. X/OPEN).**
- **SQL is Federal Information Processing Standard (FIPS) to which conformance is required for all sales of databases to American Government.**

Importance of SQL

- **SQL is used in other standards and even influences development of other standards as a definitional tool. Examples include:**
  - **ISO's Information Resource Directory System (IRDS) Standard**
  - **Remote Data Access (RDA) Standard.**

3.1.4 Writing SQL Commands

- **Terms**
  - **formal terms：relations, attributes, tuples,**
  - **alternative terms： tables, columns, rows**
- **SQL statement consists of *reserved words* and *user-defined words*.**
  - **Reserved words are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.**
  - **User-defined words are made up by user and represent names of various database objects such as relations, columns, views，index.**

  e.g. Table *s*， column *sno* ;

3.1.4 Writing SQL Commands

- Most components of an SQL statement are *case insensitive*, except for literal character data（'SMITH' not same with 'smith'）.
- More readable with indentation (缩进) and lineation (分行):
  - Each clause should begin on a new line.
  - The start of a clause should line up with the start of other clauses.
  - If a clause has several parts, should each appear on a separate line and be indented under the start of the clause.

3.1.4 Writing SQL Commands

- **Use the extended form of the** Backus Naur Form **(BNF) notation to define SQL statements:**
  - **Upper-case letters represent reserved words.**
  - **Lower-case letters represent user-defined words.**
  - **| indicates a *choice* among alternatives. a | b | c**
  - **{ }Curly braces indicate a *required element*. {a}**
  - **[ ] Square brackets indicate an *optional element*. [a]**
  - **…** (e'llipsis) **indicates *optional repetition* (0 or more). {a | b} (, c . . .)**

Case：DreamHome

- **Branch** (branchNo, street, city, postcode)
- **Staff** (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
- **Property** (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo)
- **Client** (clientNo, fName, lName, telNo, prefType, maxRent)
- **PrivateOwner** (ownerNo, fName, lName, address, telNo)
- **Viewing** (clientNo, propertyNo, viewDate, comment)

Literals（常量）

- **Literals are constants used in SQL statements.**
- **All non-numeric literals must be enclosed in single quotes (e.g. ‘London’).**

select * from sc //**SQL Server**中纯数字的字符串不用单引号也可

where sno=08300012 and cno=801 能运行成功

- **All numeric literals must not be enclosed in quotes (e.g. 650.00).**

e.g. INSERT INTO Property (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo) VALUES (‘PA14’, ‘16 Holhead’, ‘Aberdeen’, ‘AB7 5SU’, ‘House’, 6, 650.00, ‘CO46’, ‘SA9’, ‘B007’);

3.1.3 ISO SQL Data Types

- **SQL Identifiers**
  - **default character set**: A . . . Z, a . . . z, 0 . . . 9, underscore (_) character.
  - **restrictions** imposed on an identifier
    - an identifier can be no longer than 128 characters (most dialects have a much lower limit than this);
    - an identifier must start with a letter ,否则用[].
    - an identifier cannot contain spaces,否则用[].

SQL Scalar Data Types

Data type	Declarations			
boolean	BOOLEAN			
character	CHAR	VARCHAR		
bit†	BIT	BIT VARYING		
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION	
datetime	DATE	TIME	TIMESTAMP	
interval	INTERVAL			
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT	

- the exact numeric value −12.345 has precision 5 and scale 3. A special case of exact numeric occurs with integers.
- BIT and BIT VARYING have been removed from the SQL:2003 standard.

11 Operations (Key Words)

- Create, Drop(schema, table,view,index) , Alter
- Select, Update, Delete, Insert
- Grant, Revoke
- Commit, Rollback

3.1.5 Integrity Enhancement Feature

（完整性增强特性）

- **Consider five types of integrity constraints:**
  - **Required data.**
  - **Domain constraints (value interval).**
  - **Entity integrity.**
  - **Referential integrity.**
  - **Enterprise constraints(user-defined integrity).**

Integrity Enhancement Feature

◆**Required Data**

position VARCHAR(10) NOT NULL

◆**Domain Constraints CHECK clause**

CHECK （searchCondition）

e.g. sex CHAR NOT NULL

CHECK (sex IN ('M', 'F'));//在SQL Server中不行，须改为：

sex CHAR CONSTRAINT Ckname

CHECK(sex='m' or sex='f')

IEF - Entity Integrity

- **Primary key of a table must contain a unique, non-null value for each row.**

e.g. Property (propertyNo, City, street, ownerNo, staffNo)

- **ISO standard supports PRIMARY KEY clause in CREATE and ALTER TABLE statements:**

  In Property table: PRIMARY KEY(propertyNo)

  In Viewing table: PRIMARY KEY(clientNo, propertyNo)

- **Can only have one PRIMARY KEY clause per table.**
- **Can still ensure uniqueness for alternate keys using UNIQUE: In Client table:**

```
telNo  CHAR(11) NOT NULL, //这个逗号和“not null”可有可无
       UNIQUE ( telNo ),
```

telNo 的“ NOT NULL”可以不写.

SQL Server中例子

```
CREATE TABLE Staff
    (staffNo VARCHAR(5) primary key,
     LName VARCHAR(15),
     TelNo VARCHAR(11) unique,
     salary DECIMAL(7,2));
```

	staffNo	LName	TelNo	salary
1	1	杨丽坤	1380138599	9600.78
2	2	辛顿	13801385500	5700.00
3	76	西蒙	13801385588	8900.56

以下语句均可以：

```
INSERT INTO staff
VALUES (76,'西蒙', '13801385588',8900.56)

INSERT INTO staff
VALUES (1,'杨丽坤', 1380138599,9600.78)

INSERT INTO staff
VALUES (2,'辛顿', 13801385500,5700)
```

IEF - Referential Integrity

- Foreign Key（FK） is a column or a set of columns that links each row in child table containing FK to the row of parent table containing matching PK.
- Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table.
- ISO standard supports definition of FKs with FOREIGN KEY clause in CREATE and ALTER TABLE: In the Staff table

FOREIGN KEY (branchNo) REFERENCES Branch

Branch (branchNo,street,city,postcode)

Staff (staffNo, fName, lName, position, branchNo, leaderNo)

FOREIGN KEY (branchNo) REFERENCES Branch

B

branchNo	street	city	postcode
B003	18 Dale Rd	Glasgow	G12
B005	22 Deer Rd	London	SW14EH
B007	16 Argyll St	Aberdeen	AB23SU

S

staffNo	….	branchNo	leaderNo
SL41	….	B005	SL31
SL21	….	B025	SL41
SA9	….		SL41
SG14	….	B003	
SG37	….	B003	SG14

IEF - Referential Integrity

- Any INSERT/UPDATE that attempts to create FK value in child table without matching candidate key value in parent is rejected.

s (sno, sname, age, sex) -- parent Relation

sc (sno, cno, grade) -- child Relation

S

sno	sname	age	sex
S1	LI	17	M
S2	SHI	19	F
S3	LIU	21	F
S4	CHEN	20	M

No !

SC

sno	cno	grade
S5	C1	78
S1	C2	86
S2	C2	77
S3	C3	89
S5	C1	80

IEF - Referential Integrity

- **Action taken【that attempts to update/delete a candidate key value in parent table with matching rows in child】is dependent on referential action specified using ON UPDATE and ON DELETE subclauses:**
  - **CASCADE**
  - **SET NULL**
  - **SET DEFAULT**
  - **NO ACTION （default option）**

IEF - Referential Integrity

CASCADE: Delete row from parent and delete matching rows in child, and so on in cascading manner.

SET NULL: Delete row from parent and set FK column(s) in child to NULL. Valid only if FK columns do not have the NOT NULL qualifier specified.

SET DEFAULT: Delete row from parent and set each component of FK in child to specified default. Valid only if FK columns have a DEFAULT value specified.

NO ACTION: Reject deletion from parent. Default.

IEF - Referential Integrity

Property(Pno, City, street, ownerNo, staffNo)

FOREIGN KEY (staffNo) REFERENCES
Staff ON DELETE SET NULL

FOREIGN KEY (ownerNo) REFERENCES
Owner ON UPDATE CASCADE

Home relation
or Parent table

IEF - Referential Integrity

CASCADE ON DELETE :

S

Sno	SN	age	sex
S1	LI	17	M
S2	SHI	19	F
S3	LIU	21	F
S4	CHEN	20	M

SC

Sno	Cno	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80

Referential Integrity: ON DELETE

CASCADE **SET NULL** **SET DEFAULT**

Branch

Bno	Location	PC
B01	…	M
B02	…	F
B03	…	F
B04	…	M

Staff

Sno	Name	Bno
S1	…	B01
S2	…	B03
S3	…	B01
S4	…	B01
S5	…	B04
S6	…	B03
S7	…	B01
S8	…	B04

E.g. Create a relation schema for electives

SC (SNO,CNO,G)

在S或C表中删除某行，会级联删除SC表中的相应行。
在SC表中删除某行，不会影响S或C表中的相应行。

```
CREATE TABLE SC
( SNO CHAR(4) NOT NULL,
  CNO  CHAR(4) NOT NULL,
  G   SMALLINT,
  PRIMARY KEY (SNO, CNO),
  FOREIGN KEY(SNO)  REFERENCES   S(SNO)
        on delete cascade,
  FOREIGN KEY(CNO)  REFERENCES   C(CNO)
        on delete cascade,
  CHECK ( (G IS NULL) OR
                (G BETWEEN 0  AND 100))
);
```

Create a relation schema for electives

SC(SNUM, CNUM, G)

```
CREATE TABLE SC
( SNUM CHAR(4) NOT NULL,
  CNUM  CHAR(4) NOT NULL,
  G   SMALLINT,
  PRIMARY KEY (SNUM, CNUM),
  FOREIGN KEY(SNUM)REFERENCES S(SNO),
  FOREIGN KEY(CNUM)REFERENCES C(CNO),
  CHECK (  (G IS NULL) OR
                (G BETWEEN 0  AND 100)  )
);
```

The name of FK does not have to be the same with that of PK, but their data type must be the same.

3.2 Data Definition statements

SQL DDL allows database objects such as schemas, tables, views, and indexes to be created and destroyed.

DB objects	OPERATION		
	CREATE	DROP	ALTER
Schema	CREATE SCHEMA	DROP SCHEMA	
Table	CREATE TABLE	DROP TABLE	ALTER TABLE
View	CREATE VIEW	DROP VIEW	
Index	CREATE INDEX	DROP INDEX	

Case

Estates Agency-Dream of Home

- **Branch** (branchNo, street, city, postcode)
- **Staff** (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
- **Property** (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo)
- **Client** (clientNo, fName, lName, telNo, prefType, maxRent)
- **PrivateOwner** (ownerNo, fName, lName, address, telNo)
- **Viewing** (clientNo, propertyNo, viewDate, comment)

(1) CREATE / DROP SCHEMA(模式)

CREATE SCHEMA [Name | AUTHORIZATION CreatorId]

e.g. CREATE SCHEMA student AUTHORIZATION wang;

DROP SCHEMA Name [RESTRICT | CASCADE]

- **With RESTRICT (default), schema must be empty or operation fails.**
- **With CASCADE, operation cascades to drop all objects associated with schema in the order defined above. If any of these drop operations fail, DROP SCHEMA fails.**

(2) CREATE TABLE

The basic syntax:

```
CREATE TABLE TableName
{(colName dataType [NOT NULL] [UNIQUE]
[DEFAULT defaultOption]
[CHECK searchCondition] [,...]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listOfColumns),] […,]}
{[FOREIGN KEY (listOfFKColumns)
  REFERENCES ParentTableName [(listOfCKColumns)],
  [ON UPDATE referentialAction]
  [ON DELETE referentialAction ]] [,…]}
 {[CHECK (searchCondition)] [,…] })
```

(2) CREATE TABLE

- Creates a table with one or more columns of the specified ***dataType***.
- With **NOT NULL**, system rejects any attempt to insert a null in the column.
- Can specify **a DEFAULT value** for the column.
- **Primary keys** should always be specified as **NOT NULL**.
- **FOREIGN KEY** clause specifies FK along with the referential action

Create a relation schema for students

S (SNO,SN,age,sex)

```
CREATE TABLE S
    ( SNO CHAR(4) NOT NULL,
      SN CHAR(8) NOT NULL,
       AGE SMALLINT,
       SEX CHAR(1),
      PRIMARY KEY (SNO)
    );
```

Not null 不是必须的

```
CREATE TABLE S
    ( SNO CHAR(4) PRIMARY KEY,
      SN CHAR(8) NOT NULL,
       AGE SMALLINT,
       SEX CHAR(1)
    );
```

Create a relation schema for courses C(CNO,CN,T, CREDIT)

```
CREATE TABLE C
    ( CNO CHAR(4) NOT NULL,
      CN CHAR(8) NOT NULL,
       T  CHAR(10),
      CREDIT  SMALLINT,
      PRIMARY KEY (CNO)
     );
```

Not null 不是必须的

```
CREATE TABLE C
    ( CNO CHAR(4) PRIMARY KEY,
      CN CHAR(8) NOT NULL,
       T  CHAR(10),
      CREDIT  SMALLINT
     );
```

Create a relation schema for electives

SC(SNO,CNO,G)

```
CREATE TABLE SC
 ( SNO CHAR(4) NOT NULL,   primary key
   CNO  CHAR(4) NOT NULL,  primary key
   G   SMALLINT,
   PRIMARY KEY (SNO, CNO),
   FOREIGN KEY(SNO)REFERENCES S(SNO)
         on delete cascade,
   FOREIGN KEY(CNO)REFERENCES C(CNO)
         on delete cascade,
   CHECK ((G IS NULL) OR (G BETWEEN 0
         AND 100))
);
```

Wrong !

Create a relation schema for electives

SC(SNUM, CNUM, G)

```
CREATE TABLE SC
 ( SNUM CHAR(4) NOT NULL,
   CNUM  CHAR(4) NOT NULL,
    G   SMALLINT,
    PRIMARY KEY (SNUM, CNUM),
    FOREIGN KEY(SNUM)REFERENCES S (SNO),
    FOREIGN KEY(CNUM)REFERENCES C (CNO),
    CHECK ((G IS NULL) OR (G BETWEEN 0
          AND 100))
 );
```

Foreign key name may not be the same with that of its primary key. But the type must be the same.

Example 3.1 - CREATE TABLE

```
CREATE TABLE Property  (
   propertyNo    Char(10)   NOT NULL, ….
   rooms    int    NOT NULL   DEFAULT 4,
   rent    int          NOT NULL   DEFAULT 600,
   ownerNo     Char(10)       NOT NULL,
   staffNo       Char(10)
                Constraint StaffNotHandlingTooMuch ….
```

(see next slide)

```
Constraint StaffNotHandlingTooMuch
    CHECK ( NOT EXISTS(SELECT staffNo
                       FROM Property
                       GROUP BY staffNo
                       HAVING COUNT(*) > 100)),
branchNo Char(10)  NOT NULL,
PRIMARY KEY (propertyNo),
FOREIGN KEY (staffNo) REFERENCES Staff
    ON DELETE SET NULL ON UPDATE CASCADE,
FOREIGN KEY (ownerNo) REFERENCES Owner
    ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (branchNo) REFERENCES Branch
    ON DELETE NO ACTION ON UPDATE CASCADE
);
```

(3) ALTER TABLE

ALTER TABLE TableName

[**ADD [COLUMN]** columnName dataType [**NOT NULL**]
[**UNIQUE**] [**DEFAULT** defaultOption]
[**CHECK** (searchCondition)]]

[**DROP** [**COLUMN**] columnName [**RESTRICT** | **CASCADE**]]

[**ADD** [**CONSTRAINT** [ConstraintName]]
tableConstraintDefinition]

[**DROP CONSTRAINT** ConstraintName [**RESTRICT** | **CASCADE**]]

[**ALTER** [**COLUMN**] **SET DEFAULT** defaultOption]

[**ALTER** [**COLUMN**] **DROP DEFAULT**]

(3) ALTER TABLE

- **Add a new column to a table.**
- **Drop a column from a table.**
- **Add a new table constraint.**
- **Drop a table constraint.**
- **Set a default for a column.**
- **Drop a default for a column.**

Examples 3.2

- /*add a column named ‘addr’in table S，then change its length from 20 to 25, then drop it*/

SQL Server:

```
use test
alter table s  add addr char(20);  //add后不能加column
alter table s alter column addr char(25);
alter table s drop column addr;
```

DB2:

- alter table s **add** addr char(20);
- alter table s **alter column** addr **set data type** char(25);
- alter table s **drop column** addr;

Example 3.3 ALTER TABLE

Change Staff table by removing default of 'Assistant' for position column and setting default for sex column to female ('F').

```
ALTER TABLE Staff
        ALTER position DROP DEFAULT;

ALTER TABLE Staff
        ALTER sex SET DEFAULT 'F';
```

SQL Server 中，用【 sp_help 表名 】查看某列上的默认值

约束名，然后， ALTER TABLE 表名 drop constraint 约束名

Examples 3.4

- /*set a default value 18 for column age in table S*/

  SQL Server:

  alter table s add **constraint** age_default

  default 18 **for** age ;

  DB2:

  alter table s alter column age set default 18;

  alter table s alter column age drop default;

Examples 3.5

- **set a default value ‘male’ for column sex in table S**

SQL Server :

```
alter table s
    add constraint sex_default default 'm' for sex;
```

DB2:

```
alter table s alter column sex set default  'm';
```

- **set column sname as alternate key (not primary key) in table S*/**

SQL Server and DB2:

```
alter table s
     add constraint uni_sname unique(sname)
```

Example 3.6 ALTER TABLE

- **Remove the constraint from Property that staff not allowed to handle more than 100 properties at time.**

```
ALTER TABLE Property
    DROP CONSTRAINT StaffNotHandlingTooMuch;
```

- Change the Client table by adding a new column representing the preferred number of rooms.

```
ALTER TABLE Client ADD prefNoRooms int;
```

SQL Server 不加column，MySQL可以加column

Examples 3.7

- **/*add a constraint in table S :male students' age should be younger than 23 and female students' age should be younger than 21 */**

SQL Server and DB2:

- **alter table s**

  add constraint sex_age

  check((sex='f' and age<21) or

  (sex='m' and age<23))

- **alter table s**

  drop constraint sex_age

(4) DROP TABLE

DROP TABLE TableName
[RESTRICT | CASCADE]

e.g. DROP TABLE Property ;

- **Removes named table and all rows within it.**
- **With RESTRICT (default), if any other objects depend for their existence on continued existence of this table, SQL does not allow request.**
- **With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).**

(5) Indexes

- *Index* is a data structure that allows the DBMS to locate particular records in a file more quickly and thereby speed response to user queries.
- Index is a data structure used to **speed access** to tuples of a relation, given values of one or more attributes.
- Index could be a **hash** table, but in a DBMS it is always a balanced search tree with giant nodes (a full disk page) called a *B^+tree*.

(5) Indexes

Format:

CREATE [UNIQUE] [CLUSTER] INDEX
<IndexName> **ON**
<TableName>**(**<ColumnName>**[**<ASC|DESC>**][,[…]);**

- **ASC –Ascending order(default);**
- **DESC---Descending order**
- **UNIQUE**：uniqueness of the indexed column will be enforced by the DBMS.
- **CLUSTER**：聚簇索引
  - The order of **cluster index** is the same with that of the physical storage order of logical records in the data file,
  - but the order of **non-cluster index** is independent of the order of physical storage of logical records in the data file.

(5) Indexes

- **CREATE UNIQUE INDEX** StaffNoInd
  ON Staff (staffNo **DESC**);

- **CREATE UNIQUE INDEX** PropertyNoInd
  ON Property (propertyNo);

- **CREATE INDEX** RentInd
  ON Property (city, rent **DESC**);

Basic Concepts

- Indexing mechanisms is used to speed up access to desired data, just like an index in a book.
- The file containing the **logical records** is called the ***data file,*** the file containing the **index records** is called the ***index file***.
- An **index file** consists of index records (called **index entries**) with two fields: **Search Key** and **pointer** as the following form:

search-key	pointer

- **Search Key** is an attribute or a set of attributes used to look up records in a file.
- The **pointer** is the address of the logical record containing the Search key value in the file **.**

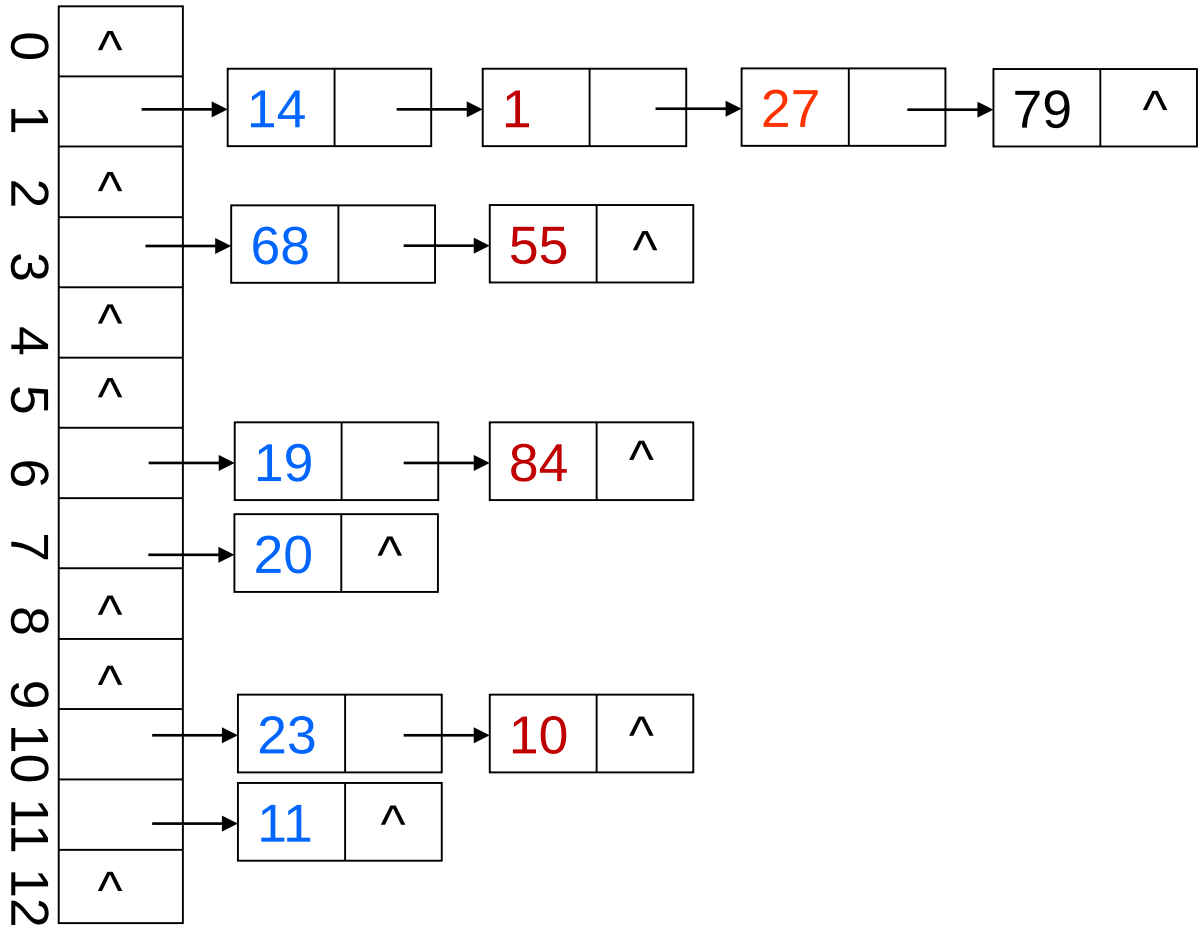
Basic Concepts

- The index records in the index file are ordered according to the ***indexing field***, which is usually a single attribute.
- Index files are typically much smaller than the data file.
- Three basic kinds of indices:
  - **Ordered indices:** search keys are stored in sorted order
  - **Hash indices:** search keys are distributed uniformly across “buckets” using a “hash function”.
  - **B^+trees**

Hash table, function = mod (key/13)

Key set={19,14,23,1,68,20,84,27,55,11,10,79}

using separate chaining to solve collisions



B+ Tree of Order 4

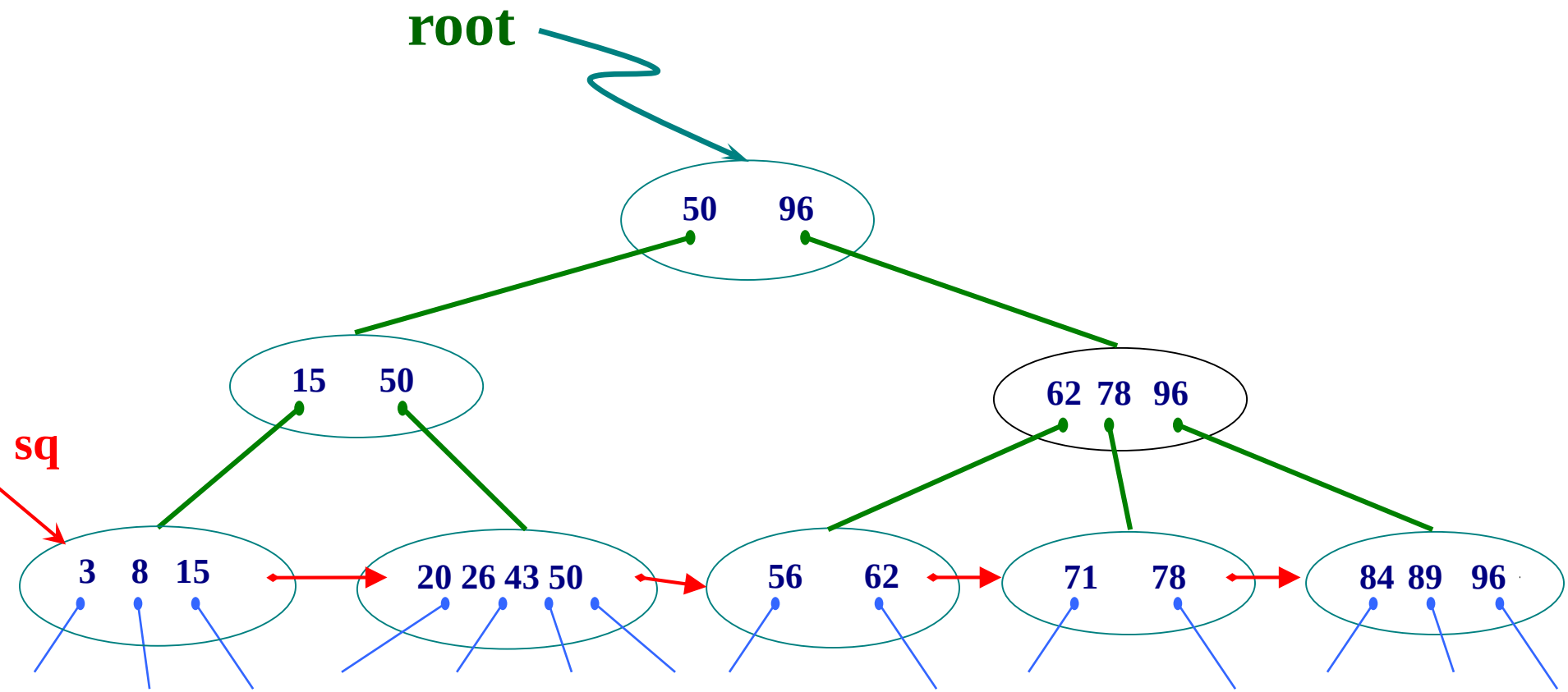


Table *Stud* is about the student information,
Create the index file on *grade* for Table *Stud*.

Table Stud

Record No.	sno	name	sex	birthday	grade
1	03083101	王文	male	09-20-81	510
2	02083106	李道	male	01-15-83	502
3	02083103	邓轩斌	male	11-28-78	524
4	02093108	王瑞姝	female	01-01-84	522
5	02093105	刘光辉	male	11-06-80	526

Index file on grade of Table Stud

Key (grade)	Record No.
502	2
510	1
522	4
524	3
526	5

Difference between sort and index

（1）Sorting will generate a new table file.

Indexing only generates index file, keep the data file unchanged.

（2）Sorting changes the order of records in original table.

Indexing will not change the order of records in original table.

（3）The new table generated by sorting can be used independently.

Index file can not be used independently. It can be used only when original table is opened.

排序与索引的主要区别

（1）排序要生成新表文件；

索引只生成索引文件。

（2）排序生成的新表记录重新调整改变了，

索引不改变原表中记录号。

（3）排序生成的新表可以单独使用，

索引文件不能单独使用（只有在表文件打开时才能使用）

排序与索引的概念

数据库表或自由表都可排序和索引，排序是从表记录中依某字段或字段表达式的值，按升序或降序重新组织排列记录先后顺序、并生成排序成功的新表文件的操作，排序操作后只有打开了排序成功后的新表文件，排序才有效；

索引是依表中某字段或字段表达式值，进行表中记录先后顺序重排的操作（升、降序），不过索引不生成新表文件，只产生索引文件，且不破坏原来表中的记录号次序，即保持记录号不变，只是调整了记录次序，索引文件因为不存储数据，只保存记录指针，所以索引文件比较小。

索引的概念

索引（INDEX）：对表中的记录进行逻辑排序。即另外形成一个索引关键表达式值与记录号之间的对照表，这个对照表就是索引文件。

索引文件是一个二维表，其中仅有二列数据：关键字值和记录的物理位置。关键字值是包含有字段的排序规则表达式，记录的物理位置指向关键字值在表中所在的物理位置。

注意：索引并未改变表中记录的物理位置，仅仅改变了表中记录的逻辑排序。但是，当用户将建立好的索引文件打开以后，记录的显示顺序或读取处理记录的顺序将会按照索引文件排列的记录顺序进行。这样大大提高了记录的检索速度。

可以为一个表同时建立多个索引文件，每个索引文件表示处理记录的不同顺序。

索引

SQL Server提供了两类索引,群集的和非群集的,两者都是B树型索引,由于群集索引的数据是以群集索引顺序有效的,所以一个表只能有一个群集索引,而每个表可以有249个非群集索引, 索引可以包含1...16个属性,但总索引的宽度不大于255个字节.

采用索引可以:

*按索引关键字保持记录的先后次序

*根据一个或多个列中的有序值来在一个表中搜索

*SQL Server可自主建立索引

*可以控制每个记录只输入一次(实体完整性,只用于唯一索引)

*加快查询

建立与删除索引

- 建立索引是加快查询速度的有效手段
- 建立索引
  - DBMS自动建立
    - PRIMARY KEY
    - UNIQUE
  - DBA或表的属主（即建立表的人）根据需要建立
- 维护索引
  - DBMS自动完成
- 使用索引
  - DBMS自动选择是否使用索引以及使用哪些索引

建立索引

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**聚簇索引**

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**。

聚簇索引

建立聚簇索引后，基表中数据也需要按指定的聚簇属性值的升序或降序存放。也即聚簇索引的索引项顺序与表中记录的物理顺序一致。

- 数据表中元组的物理次序与索引次序相同
- 索引项包含指向物理页的指针
- 每个表最多只能建立一个聚簇索引

聚簇索引（续）

- 聚簇索引的**用途**：对于某些类型的查询，可以**提高查询效率**
- 聚簇索引的**适用范围**
  - 很少对基表进行增删操作
  - 很少对其中的索引列进行修改操作

Difference between cluster index and non-cluster index

- The order of **cluster index** is the same with that of the physical storage order of logical records in the data file,
- but the order of **non-cluster index** is independent of the order of physical storage of logical records in the data file.

create cluster index *myindex* on table S (sno)

Index file

key	addr
s1	1
s2	2
s3	0

S	sno	sn	age	sex
0	s3	zhou	20	F
1	s1	wang	18	F
2	s2	li	19	M

physical storage order

of original Data file

key	addr
s1	0
s2	1
s3	2

	sno	sn	age	sex
0	s1	wang	18	F
1	s2	li	19	M
2	s3	zhou	20	F

Physical storage order of data file after creating cluster index

CREATE CLUSTER INDEX *Stusname* ON S (sn);

Index file

key	addr
li	2
wang	1
zhou	0

key	addr
li	0
wang	1
zhou	2

S	sno	sn	age	sex
0	s3	zhou	20	F
1	s1	wang	18	F
2	s2	li	19	M

physical storage order of original Data file

sno	sn	age	sex
s2	li	19	M
s1	wang	18	F
s3	zhou	20	F

Physical storage order of data file after creating cluster index

Example ---- **unique** Index

1. **create a unique index on ascending sno for Table Student.**

   CREATE UNIQUE INDEX Stusno ON Student(Sno)；

2. **create a unique index on ascending cno for Table Course.**

   CREATE UNIQUE INDEX Coucno ON Course(Cno)；

3. **create the unique index on ascending sno and descending cno in Table SC.**

   CREATE UNIQUE INDEX SCno ON SC(Sno ASC，Cno DESC)；

   CREATE UNIQUE INDEX SCno ON SC(Sno) **Wrong!**

(6) Drop Index

DROP INDEX <index name>;

– Delete the description of this index from data dictionary.

Example 3.7

Delete the index *Stusname* in Table Student.

DROP INDEX Stusname on Student;