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

3.1 Introduction to SQL

3.1.1 purpose and Characteristics of SQL

1. Objectives of SQL:

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.

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 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.**

  e.g. Table *s*, column *sno* ;

3.1.2 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.

3.1.2 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, branchNo)
- **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’).
- 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;
    - an identifier cannot contain spaces.

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.

3.1.4 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）

sex CHAR NOT NULL

CHECK (sex IN (‘M’, ‘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,
UNIQUE ( telNo ),
```

telNo must be declared as NOT NULL before UNIQUE clause.

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.**
- **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 DEFAULT**
  - **SET NULL**
  - **NO ACTION**

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)

```
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, branchNo)
- **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)
    );
```

```
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)
      );
```

```
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);
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';
```

Example 3.4 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;
```

Examples 3.5

- **/*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 **.**

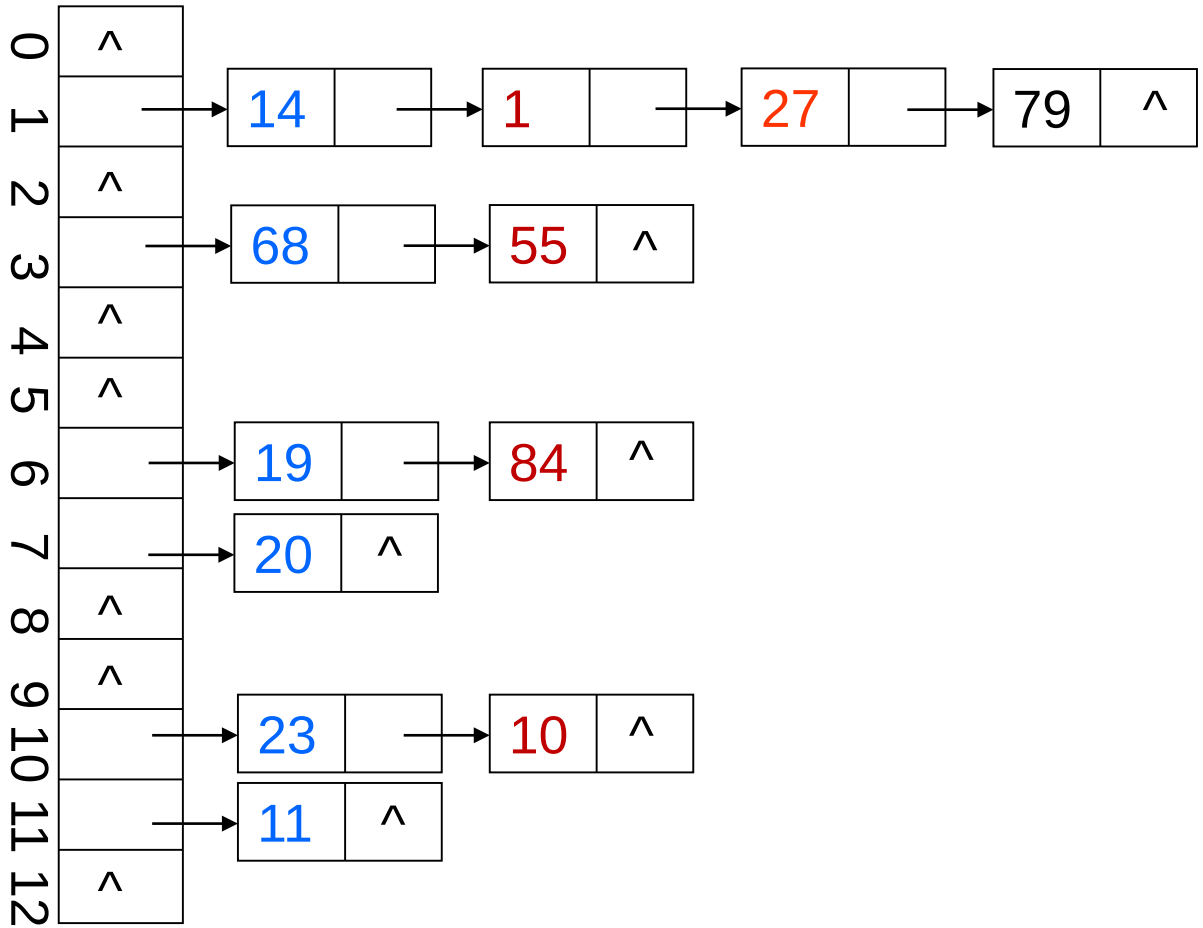
Basic Concepts

- 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

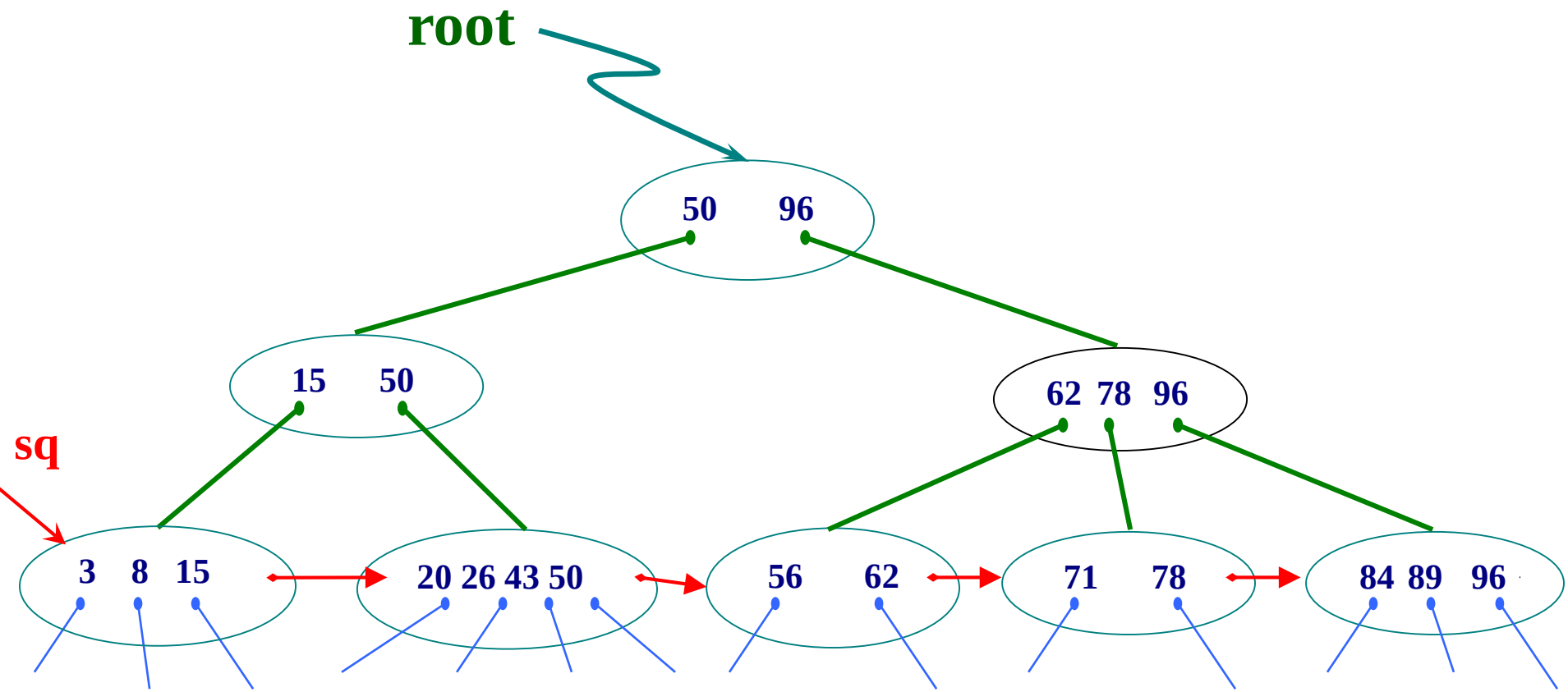


Table *Stud* is about the student information,
Create the index file on *grade* for Table *Stud*.

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.

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;