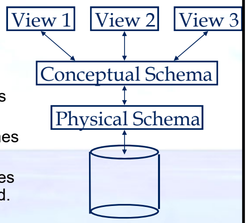
# CpSc 4620/6620: Database Management Systems (DBMS) (TEXNH Approach)

## Views, Constraints, Triggers and Index

James Wang

## Database Architecture

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



2

## Example: Course Database

- **Conceptual schema:**
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- **Physical schema:**
  - Relations stored as unordered files.
  - Index on first column of Students.
- **External Schema (View):**
  - *Course_info(cid:string,enrollment:integer)*

3

## What are views in a database?

- **A view in a database is created by applying a relational algebra expression to the base relations (tables).**
- **A view is a virtual relation (table) that is derived from one or many base relations. Therefore, no physical table is stored in the database for a view. The actual data are stored in base relations.**
- **Updates applied to the base relations are immediately visible to the view.**
- **Updates to values in the view may also be reflected in the base relations.**

4

## Why views?

- **Views can subset the data contained in a table.**
- **Views can join multiple tables into a single virtual table to simplify the queries.**
- **Views can act as aggregated tables, where aggregated data (sum, average etc.) are calculated and presented as part of the data.**
- **Views can hide the complexity of data.**
- **Views do not incur any extra storage overhead**
- **Depending on the SQL engine used, views can provide extra security.**
- **Views control how a table or tables are exposed to outer world.**

5

## Read-only & updatable views

- **If a DBMS system is able to determine the reverse mapping from the view schema to the schema of the underlying base tables, then the view is updatable.**
- **INSERT, UPDATE, and DELETE operations can be performed on updatable views.**
- **Read-only views do not support such operations because the DBMS is not able to map the changes to the underlying base tables.**
  - Materialized view is a pre-populated, non-virtual and read-only view used in data warehousing.

6

## Views in MySQL

- **Views (including updatable views) are implemented in MySQL Server 5.0, and are available in MYSQL binary releases from 5.0.1 and up.**
- **Metadata about views can be obtained from the INFORMATION_SCHEMA.VIEWS table and by using the SHOW CREATE VIEW statement.**
- **View Operations:**
  - Creating or altering views with CREATE VIEW or ALTER VIEW
  - Destroying views with DROP VIEW

7

## Create a view

```
CREATE [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

- **Example:**

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM t;
mysql> SELECT * FROM v;
    +------+-------+-------+
    | qty  | price | value |
    +------+-------+-------+
    |    3 |    50 |   150 |
    +------+-------+-------+
```

**http://dev.mysql.com/doc/refman/5.0/en/create-view.html**

8

## Restrictions in MySQL view creation

- **The SELECT statement cannot contain a subquery in the FROM clause.**
- **The SELECT statement cannot refer to system or user variables.**
- **Any table or view referred to in the definition must exist. However, after a view has been created, it is possible to drop a table or view that the definition refers to. In this case, use of the view results in an error.**

9

## Restrictions in MySQL view creation (cont.)

- **The definition cannot refer to a TEMPORARY table, and you cannot create a TEMPORARY view.**
- **The tables named in the view definition must already exist.**
- **You cannot associate a trigger with a view.**
- **ORDER BY is allowed in a view definition, but it is ignored if you select from a view using a statement that has its own ORDER BY.**

10

## Alter a view

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

- **Example:**

```
mysql> ALTER VIEW v AS SELECT qty, price FROM t;
mysql> SELECT * FROM v;
    +------+-------+
    | qty  | price |
    +------+-------+
    |    3 |    50 |
    +------+-------+
```

**http://dev.mysql.com/doc/refman/5.0/en/alter-view.html**

11

## Drop a view

```
DROP VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```

- **Example:**

```
mysql> DROP VIEW v IF EXISTS;
```

- **DROP VIEW removes one or more views.**
- **You must have the DROP privilege for each view.**
- **If any of the views named in the argument list do not exist, MySQL returns an error indicating by name which non-existing views it was unable to drop, but it also drops all of the views in the list that do exist.**
- **RESTRICT and CASCADE, if given, are parsed and ignored in MySQL.**

**http://dev.mysql.com/doc/refman/5.0/en/alter-view.html**

12

## Show the view create statement

SHOW CREATE VIEW view_name

- **This statement shows a CREATE VIEW statement that creates the given view.**

mysql> SHOW CREATE VIEW v;

```
+------+----------------------------------------------------+
| View | Create View                                        |
+------+----------------------------------------------------+
| v    | CREATE VIEW `test`.`v` AS select 1 AS `a`,2 AS `b` |
+------+----------------------------------------------------+
```

- **Use of SHOW CREATE VIEW requires the SHOW VIEW privilege and the SELECT privilege for the view in question.**
- **You can also obtain information about view objects from INFORMATION_SCHEMA, which contains a VIEWS table.**

13

## Constraints

- **SQL provides variety of techniques to express constraints to ensure the integrity and consistency of data.**
- **In DBMS, constraints and their associated "active" elements include:**
  - Key constraints.
  - Referential constraints.
  - Constraints on attributes or tuples.
  - Interrelation constraints.
  - Triggers.

14

## Unique key and Primary key

- **In a database table, keys are special attributes used to identify rows or represent relationships.**
- **Unique key and primary key:**
  - A unique key or primary key is a single column or a set of columns that uniquely identify each row in a table.
  - No two distinct rows in a table can have the same value (or combination of values) in those columns.
  - A database table may have arbitrarily many unique keys but at most one primary key.

15

## Unique key and Primary key (cont.)

- **Differences between unique key and primary key.**
  - A primary key is a special case of unique keys.
  - A primary key is automatically enforced by NOT NULL constaint.
  - The NOT NULL constraint is not implicitly enforced for a unique key.
  - The relational model does not distinguish between primary keys and other kinds of keys.
  - Primary keys were added to the SQL standard mainly as a convenience to the application programmer.

16

## Define Primary Key

```
ALTER TABLE <table identifier>
    ADD [ CONSTRAINT <constraint identifier> ]
    PRIMARY KEY ( <column expression> {, <column expression>}... )

CREATE TABLE table_name (
   id  INT,
   col2 VARCHAR(20),
   ...
   CONSTRAINT tab_pk PRIMARY KEY(id),
   ...
)

CREATE TABLE table_name (
   id  INT  PRIMARY KEY,
   col2 VARCHAR(20),
   ...
)
```

- **Note: some DBMS systems require explicitly specify NOT NULL constraint for a primary key**

17

## Define  Unique Key

```
ALTER TABLE <table identifier>
    ADD [ CONSTRAINT <constraint identifier> ]
    UNIQUE ( <column expression> {, <column expression>}... )

CREATE TABLE table_name (
   id   INT,
   col2     VARCHAR(20),
   key1  SMALLINT,
   ...
   CONSTRAINT key_uniqe UNIQUE(key1),
   ...
)

CREATE TABLE table_name (
   id  INT  PRIMARY KEY,
   col2 VARCHAR(20),
   ...
   key1  SMALLINT UNIQUE,
   ...
)
```

18

## Foreign Keys

- The foreign key identifies a column or a set of columns in one (referencing) table that refers to a column or set of columns in another (referenced) table. We call it a referential constraint between two tables.
- The columns in the referenced table must be a primary key or unique key.
- The values in one row of the referencing columns must occur in a single row in the referenced table.
- Most of the time, a foreign key reflects a one to many relationship.

19

## Define Foreign Key

```
ALTER TABLE <table identifier>
  ADD [ CONSTRAINT <constraint identifier> ]
    FOREIGN KEY ( <column expression> {, <column expression>}... )
    REFERENCES <table identifier> [ ( <column expression> {, <column
    expression>}... ) ]
    [ ON UPDATE <referential action> ]
    [ ON DELETE <referential action> ]
```

**Examples:**

```
CREATE TABLE table_name (
  id    INT  PRIMARY KEY,
  col2  VARCHAR(20),
  col3  INT,
  ...
  CONSTRAINT col3_fk FOREIGN KEY(col3)
    REFERENCES other_table(key_col) ON DELETE CASCADE,
  ... )

CREATE TABLE table_name (
  id    INT  PRIMARY KEY,
  col2  VARCHAR(20),
  col3  INT FOREIGN KEY REFERENCES other_table(column_name),
  ... )
```

20

## Referential Actions

- **CASCADE:** A foreign key with a cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted.
- **RESTRICT:** A row in the referenced table cannot be updated or deleted if dependent rows still exist. In that case, no data change is even attempted.
- **NO ACTION:** The UPDATE or DELETE SQL statement is executed on the referenced table only if none of the referential relationships is violated. This is different from the RESTRICT, which does not allow UPDATE or DELETE in any case.
- **SET NULL or SET DEFAULT:** The foreign key values in the referencing row are set to NULL or DEFAULT VALUE when the referenced row is updated or deleted.

21

## Artificial Key

- An artificial key is used by the DBMS system to identify an instance of a relation. It is not derived from any application data in the database.
- Using an artificial key as the primary key insulates the database relationships from changes in data values or database design (making your database more agile) and guarantees uniqueness.
- An artificial key is also called a surrogate key.

22

## Advantages of using an artificial key

- Database applications won't lose their "handle" on the row because the data changes;
- Many database systems do not support cascading updates of keys across foreign key constraints. This results in difficulty in modifying the primary key data. Using an artificial key may avoid the problem.
- Artificial keys are composed of a compact data type, such as four-byte integers. This allows the database to query faster than using primary keys consisting of multiple columns or a long attribute, such as an email address.

23

## Constraints on Attributes and Tuples

- When we create a database table, we can declare two kinds of constraints:
  - A constraint on a single attribute.
  - A constraint on a tuple as a whole.
- Not-Null constraints.
- CHECK constraints (not supported in MySQL yet):

```
CREATE TABLE MovieStar (
        name char(30) PRIMARY KEY,
        gender char(1),
        CHECK(gender='F' or name NOT LIKE 'Ms. %')
);
```

24

## Assertions

- **The most powerful active elements in SQL are not associated with particular tuples or components of tuples.**
- **These elements, called “triggers” and “assertions”, are part of the database schema, on a par with tables.**
  - An assertion is a boolean-valued SQL expression that must be true at all times.
  - A trigger is a series of actions that are associated with certain events, such as insertions into a particular table. These actions will be performed whenever these events arises.

## Creating Assertions

```
CREATE ASSERTION <assertion-name>
CHECK(<condition>)
```

- **It is very easy to create an assertion for an SQL programmer.**
- **However, it is very hard for the DBMS system to implement the assertion.**
- **MySQL does not support assertion yet.**

## Triggers

- **Triggers are sometimes called event-condition-action (ECA) rules.**
  - Triggers are only activated when certain events, specified by the database programmer, occur. These events include insert, delete, update and transaction end.
  - When triggering event happens, the trigger tests a condition to determine whether to execute the actions associated with the trigger.
  - If the condition of the trigger is satisfied, the action associated with the trigger is performed.

## Triggers in MySQL

- **Support for triggers is included beginning with MySQL 5.0.2.**

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.06 sec)
```

- **MySQL triggers are activated by SQL statements *only*. They are not activated by changes in tables made by APIs that do not transmit SQL statements to the MySQL Server.**

## Create Trigger

```
CREATE
  [DEFINER = { user | CURRENT_USER }]
  TRIGGER trigger_name trigger_time trigger_event
  ON tbl_name FOR EACH ROW trigger_stmt
```

- **trigger_time is the trigger action time. It can be BEFORE or AFTER to indicate that the trigger activates before or after the statement that activated it.**
- **trigger_event indicates the kind of statement that activates the trigger.**

## Trigger Events and Statements

- **The trigger_event can be one of the following:**
  - **INSERT:** The trigger is activated whenever a new row is inserted into the table; for example, through INSERT, LOAD DATA, and REPLACE statements.
  - **UPDATE:** The trigger is activated whenever a row is modified; for example, through UPDATE statements.
  - **DELETE:** The trigger is activated whenever a row is deleted from the table; for example, through DELETE and REPLACE statements. However, DROP TABLE and TRUNCATE statements on the table do not activate this trigger, because they do not use DELETE.
- ***trigger_stmt* is the statement to execute when the trigger activates. If you want to execute multiple statements, use the BEGIN ... END compound statement construct.**

## Example

```
CREATE TABLE test1(a1 INT);
CREATE TABLE test2(a2 INT);
CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
CREATE TABLE test4(
  a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  b4 INT DEFAULT 0
);

DELIMITER |
CREATE TRIGGER testref BEFORE INSERT ON test1
  FOR EACH ROW BEGIN
    INSERT INTO test2 SET a2 = NEW.a1;
    DELETE FROM test3 WHERE a3 = NEW.a1;
    UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
  END;
|
DELIMITER ;
```

31

## Drop Trigger

DROP TRIGGER [IF EXISTS] [*schema_name*.]*trigger_name*

- **This statement drops a trigger. The schema (database) name is optional. If the schema is omitted, the trigger is dropped from the default schema.**
- **Use IF EXISTS to prevent an error from occurring for a trigger that does not exist.**

32

## OLD and NEW keywords

- **The OLD and NEW keywords enable you to access columns in the rows affected by a trigger. (OLD and NEW are not case sensitive.)**
  - In an INSERT trigger, only NEW.col_name can be used; there is no old row.
  - In a DELETE trigger, only OLD.col_name can be used; there is no new row.
  - In an UPDATE trigger, you can use OLD.col_name to refer to the columns of a row before it is updated and NEW.col_name to refer to the columns of the row after it is updated.

33

## What is an index?

- **A database index is a sequential file (or similar data structure) that improves the speed of operations in a table.**
- **Indexes can be created based on one or more columns.**
- **Indexes can make a SQL query more efficient.**
  - The disk space required to store the index is typically less than the storage of the table.
  - It is possible to store indexes into memory from tables that would not fit into it.
  - Not all indexes are good for all queries.

34

## Clustered Index vs. Non-Clustered Index

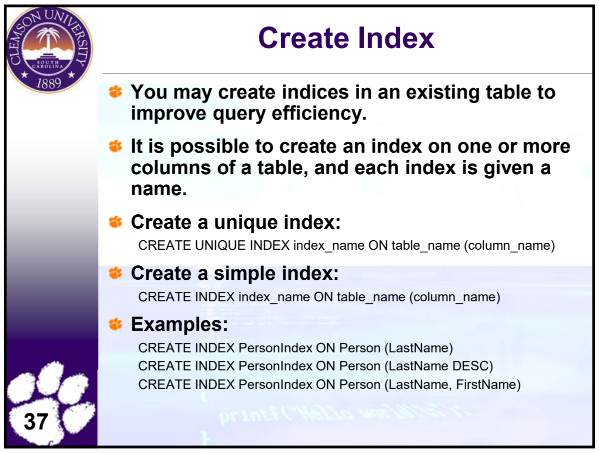
- **Index architectures can be classified as clustered or non-clustered.**
- **A clustered index forces the relational table physically ordered by the order of the index. Because a table can only be stored in one order physically only one clustered index may be created on a given database table.**
- **A non-clustered index does not care how records are stored physically.**
- **Clustered indexes can greatly increase access speed, but usually only where the data is accessed sequentially in the same or reverse order of the clustered index, or when a range of items are selected.**
- **Primary index vs. secondary index.**

35

## Dense index vs. sparse index

- **A dense index in databases is a sequential file with pairs of keys and pointers for every record in the data file.**
  - Every key in this file is associated with a particular pointer to a record in the sorted data file.
  - In clustered indexes with duplicate keys the dense index may point to the first record with that key.
- **A sparse index in databases is a sequential file with index pointers pointing to the block of the sorted data file.**
  - In clustered indexes with duplicate keys the sparse index may point to the lowest search key in each block.

36



## Create Index

- You may create indices in an existing table to improve query efficiency.
- It is possible to create an index on one or more columns of a table, and each index is given a name.
- Create a unique index:

```
CREATE UNIQUE INDEX index_name ON table_name (column_name)
```

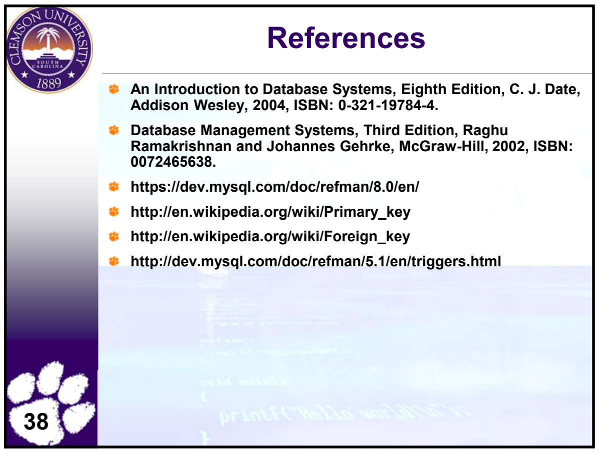
- Create a simple index:

```
CREATE INDEX index_name ON table_name (column_name)
```

- Examples:

```
CREATE INDEX PersonIndex ON Person (LastName)
CREATE INDEX PersonIndex ON Person (LastName DESC)
CREATE INDEX PersonIndex ON Person (LastName, FirstName)
```

37



## References

- An Introduction to Database Systems, Eighth Edition, C. J. Date, Addison Wesley, 2004, ISBN: 0-321-19784-4.
- Database Management Systems, Third Edition, Raghu Ramakrishnan and Johannes Gehrke, McGraw-Hill, 2002, ISBN: 0072465638.
- https://dev.mysql.com/doc/refman/8.0/en/
- http://en.wikipedia.org/wiki/Primary_key
- http://en.wikipedia.org/wiki/Foreign_key
- http://dev.mysql.com/doc/refman/5.1/en/triggers.html

38