

SQL

Other Features and Commands

Davood Rafiei

Original material Copyright 2001-2022
(some material from textbooks and other instructors)



Outline

- Update commands
- Views: more details
- NULL values
- Constraints



Updates - Insert

- *emp(sin, name, phone, city)*

- **INSERT**

- **INSERT INTO** *emp*
VALUES ('999', 'John Doe',
'444-555', 'Edmonton')
- **INSERT INTO** *emp(sin, name, city)*
VALUES ('999', 'Jim Gray', 'Ottawa')

Same as:

```
INSERT INTO emp  
VALUES ('999', 'Jim Gray', NULL, 'Ottawa')
```



Updates – Bulk Insert

- *emp* (*sin*, name, phone, city)
Edmonton_phonebook (name, phone)
 - **INSERT INTO** *Edmonton_phonebook*
SELECT name, phone
FROM emp
WHERE city = 'Edmonton'



Updates – Delete & Update

- $R(a_1, \dots, a_n)$

- DELETE

DELETE FROM R
WHERE *condition*

- UPDATE

UPDATE R
SET $a_i = v_i, \dots, a_k = v_k$
WHERE *condition*



Update Examples

customer(cname, street, city)

deposit(accno, cname, bname, balance)

- Insert a new customer record for John Smith who lives on 345 Jasper Avenue, Edmonton.

```
INSERT INTO customer
VALUES ('John Smith', '345 Jasper Avenue', 'Edmonton')
```

- Delete all customers who have less than \$1,000 in their accounts.

```
DELETE FROM customer
WHERE      cname IN (SELECT      cname
                     FROM        deposit
                     WHERE        balance < 1000)
```



Update Examples

customer(cname, street, city)

deposit(accno, cname, bname, balance)

- Increase by 5% the balance of every customer who lives in Edmonton and has a balance of more than \$5,000.

```
UPDATE      deposit
SET         balance = balance*1.05
WHERE       balance > 5000
AND         cname IN
              (SELECT  cname
               FROM      customer
               WHERE    city = 'Edmonton')
```



View Definition Examples

customer(cname, street, city)

- Create a view of customers who live in Jasper and name it **jasper_customers**.

```
CREATE VIEW jasper_customers
AS SELECT      *
      FROM      customer
      WHERE      city = 'Jasper'
```

- List the names of all customers in Jasper.

```
SELECT cname
FROM jasper_customers
```

- In queries, a view is exactly like a base table.



View Definition Examples

customer(cname, street, city)

deposit(accno, cname, bname, balance)

- Create a view (called **cust_info**) which gives for each customer the name, city, the number of deposit accounts owned and the total balance.

```
CREATE VIEW      cust_info(Name, city, Num, Total)
AS   SELECT    c.cname, city, COUNT(accno),
                SUM(balance)
FROM           deposit d, customer c
WHERE          d.cname=c.cname
GROUP BY      c.cname, city
```



View Definition Examples

customer(cname, street, city)

deposit(accno, cname, bname, balance)

- Create a view (called **deposit_holders**) which includes the name and the city of every deposit account holder.

```
CREATE VIEW    deposit_holders (Name, city)
AS SELECT    distinct c.cname, city
FROM         deposit d, customer c
WHERE        d.cname=c.cname
```



Views: A Summary

- View = query \cong table
- A derived table whose definition, not the table itself, is stored.
- Provides a degree of data independence.
- Queried like a table.
- Updated under some conditions.



View Updates

- Consider the following inserts:
 - INSERT INTO jasper_customers VALUES ('John Smith', '111-87 Ave', 'Jasper')
 - INSERT INTO jasper_customers VALUES ('Joe Smith', '112-99 Ave', 'Edmonton')
 - Is 'Joe Smith' in jasper_customers?
 - To prevent this, add **WITH CHECK OPTION** to the view definition.
- What can we say about the following insert:
 - INSERT INTO cust_info VALUES ('Jim Carey', 'Calgary', 2, 30000)



View Update - Example

Insert into deposit_holders values ('John', 'Edmonton')

Insert into deposit_holders values ('Mary', 'Calgary')

<i>cname</i>	<i>city</i>
Joe	Edmonton
John	Edmonton

deposit_holders

<i>accno</i>	<i>cname</i>	<i>bname</i>	<i>balanace</i>
1123	John	Main St	2000
1144	Joe	Whyte Ave	2200
1126	John	Pine St	1100

deposit

<i>cname</i>	<i>street</i>	<i>city</i>
Joe	111-22 St	Edmonton
Mary	98-88 Ave	Calgary
John	33-34 St	Edmonton

customer



Updatable Views in SQL/92

- A view defined on a single table is updatable if the view attributes contain the primary key or some other candidate key.
- Views defined on multiple tables using joins are generally not updatable.
- Views defined using aggregate functions are not updatable.
- **Basic idea**: each row and each column in an updatable view must correspond to a distinct row and column in a base table

SQLite only supports read-only views.



Left Outer Join

R

A	B
a1	b1
a2	b4
a5	b7

S

B	C
b1	c1
b2	c3
b4	c6

select A, B, C from R **left outer join** S **using** (B);

A	B	C
a1	b1	c1
a2	b4	c6
a5	b7	null



Right Outer Join

R

A	B
a1	b1
a2	b4
a5	b7

S

B	C
b1	c1
b2	c3
b4	c6

Select * from R **right outer join** S **using** (B);

What if R and S have more than one join columns?

Can list them all in the using clause.

A	B	C
a1	b1	c1
a2	b4	c6
null	b2	c3



Full Outer Join

R

A	B
a1	b1
a2	b4
a5	b7

S

B	C
b1	c1
b2	c3
b4	c6

Select * from R **full outer join** S **using** (B);

A	B	C
a1	b1	c1
a2	b4	c6
a5	b7	null
null	b2	c3

Implementation in SQLite

R

A	B
a1	b1
a2	b4
a5	b7

S

B	C
b1	c1
b2	c3
b4	c6

Select A, B, C from R **left outer join** S **using** (B) union
select A, B, C from S **left outer join** R **using** (B)

A	B	C
a1	b1	c1
a2	b4	c6
a5	b7	null
null	b2	c3

SQLite only supports left outer join



Alternative Syntax

Select * from R left outer join S on R.B=S.B;

- With the new syntax,
 - (1) join columns can have different names
 - (2) more general conditions are possible
✓ (e.g. R.B<S.B)
 - (3) two copies of column B

A	B	B	C
a1	b1	b1	c1
a2	b4	b4	c6
a5	b7	null	null



Unknown Value - NULL

- Useful when we don't know a column value
 - very common in practice.
- Simple usage in queries:
 - ✓ ... WHERE phone IS NULL
 - ✓ ... WHERE phone IS NOT NULL

SELECT cname **FROM** CUSTOMER **WHERE** city **IS NOT NULL**
- Complication:

SELECT cname **FROM** CUSTOMER **WHERE** city = 'Edmonton'

 - What happens if city for some customers is NULL?



NULL (Cont.)

- The predicate `city='Edmonton'` evaluates to UNKNOWN when city is NULL.
 - ✓ the customer name will not be printed.
- What if the WHERE clause consists of several predicates?
 - E.g. `city='Edmonton' OR street LIKE '100%'`
 - use three-valued logic: values TRUE, FALSE, UNKNOWN.
- NULL is much different from a constant!




Set Operations

- Operations: UNION, INTERSECT, EXCEPT
- Duplicates:
 - By default, duplicates are removed from the result of a set operation.
 - To keep duplicates, use UNION ALL (Oracle doesn't accept INTERSECT ALL or MINUS ALL!)

R

A	B
a1	5
a1	4
a5	25

select A from R union all
select A from R where B < 10



A	B
a1	5
a1	4
a5	25
a1	5
a1	4

Constraints in SQL/92

- Already seen: **primary key** and **foreign key** constraints.
- **NOT NULL**
 - specifies that an attribute cannot contain null values
 - should be specified for all primary keys (if it is not default)
- **UNIQUE**
 - specifies the alternate keys
- **Domain Constraints**
- **CHECK Constraints**



Example

```
CREATE TABLE branch (  
    bname                CHAR(15) UNIQUE,  
    address              VARCHAR(20) ,  
    city                 CHAR(9) NOT NULL,  
    assets               DECIMAL(10,2)  
                        DEFAULT 0.00)
```



User-Defined Domains

- Example

```
CREATE DOMAIN Gender AS CHAR(1)
```

- Add some domain constraints

```
CREATE DOMAIN Gender AS CHAR(1)
```

```
CHECK (VALUE = 'M' OR VALUE = 'F')
```

```
CREATE DOMAIN Gender AS CHAR(1)
```

```
CHECK (VALUE IN ('M', 'F'))
```

```
CREATE TABLE TEMP ( ..., sex Gender, ...)
```



Domain Constraints

- specifies the condition that each row has to satisfy
- Example:

```
CREATE TABLE branch
  (bname      CHAR(15) NOT NULL,
   address    VARCHAR(20) ,
   city       CHAR(9) ,
   assets     DECIMAL(10,2) DEFAULT 0.00
             CHECK (assets >= 0) );
```



Tuple Constraints

- Checked every time a tuple is inserted or updated
 - violations are rejected.

- Example

```
CREATE TABLE deposit
( accno      CHAR(9) NOT NULL,
  cname      VARCHAR(15) ,
  bname      VARCHAR(15) ,
  balance    DECIMAL(10,2) DEFAULT 0.00,
  CHECK (cname = 'Bill Clinton' OR
         balance > 100000));
```

Need the comma before a tuple constraint.



Assertions

- Global constraints of the form
 - **CREATE ASSERTION** name **CHECK** (condition)
- Example: No branch can have a customer who has more than 2 loans over \$100,000.

```
CREATE ASSERTION deposit CHECK  
  (NOT EXISTS  
    (SELECT  bname, cname  
      FROM    loan  
      WHERE   amount > 100000  
      GROUP BY   bname, cname  
      HAVING  COUNT (*) > 2) ) ;
```

SQLite supports triggers not assertions



Triggers in SQLite

```
CREATE TRIGGER [IF NOT EXISTS] trigger_name  
    [BEFORE | AFTER | INSTEAD OF]  
    [INSERT | UPDATE | DELETE]  
    ON table_name  
    [WHEN condition]  
BEGIN  
    statements;  
END;
```



Trigger Example

customer(cname, street, city)

addresslog(cname, ctime, ostreet, ocity, nstreet, ncity)

```
CREATE TRIGGER log-addresses
  AFTER UPDATE ON customer
  WHEN old.street <> new.street OR old.city<>new.city
BEGIN
  INSERT INTO addresslog VALUES
    (old.cname, datetime('now'), old.street, old.city,
     new.street, new.city);
END;
```

For actions **INSERT** and **DELETE**,
the only available references are *new* and *old* respectively.



Triggers for Updating Views

customer(cname, street, city)

- SQLite does not support updatable views
- Triggers can be used instead

```
CREATE TRIGGER update_jasper_customers
INSTEAD OF INSERT ON jasper_customers
WHEN new.city = 'Jasper'
BEGIN
    INSERT INTO customer VALUES
        (new.cname, new.street, new.city);
END;
```



Summary

- Covered
 - Basic Queries, nested queries and aggregate Queries
 - Update commands, constraints, Nulls, Views
- Will Cover
 - Embedded SQL
- Note some of the syntactic differences between SQL/92 and SQLite.

