

1

3

0



The concept of view

• A view is a "virtual" table

• the content (tuples) is defined by means of an SQL query on the database

• the content of the view depends on the content of the other tables present in the database

• the content is not memorized physically in the database

• it is recalculated every time the view is used by executing the query that defines it

• A view is an object of the database

• it can be used in queries as if it were a table

• If the query refers to a view, it has to be reformulated by the DBMS before execution

• This operation is carried out automatically

• the references to the view are substituted by its definition

2

Example n.1: definition of the view

Definition of the view small suppliers

the suppliers that have fewer than 3 employees are considered "small suppliers"

The view "small suppliers"

contains the code, name, number of employees and city of the suppliers that have fewer than 3 employees.

Name of the views

CREATEVIEW SMALL_SUPPLIERS AS

SELECT SId, SName, #Employees, City
FROM S
WHERE #Employees<3;

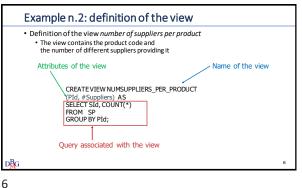
Query associated with the view

• View the code, name, employee number and city of "small suppliers" in London
• The query can be answered without using views

SELECT *
FROM S
WHERE #Employees<3 AND City='London';
• The query can be answered using the view defined previously

SELECT *
FROM SMALL_SUPPLIERS
WHERE City='London';
• The view SMALL_SUPPLIERS is used like a table

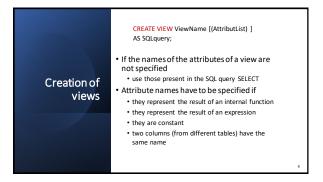
4 5



Advantages of views • Simplification of queries by breaking down a complex query into subqueries associated with the views Security management • it is possible to introduce different privacy protection mechanisms for each user or group access authorization is associated with the view · each user, or group, accesses the database only via views that are appropriate for the operation they are authorized to carry out · Database mainteinance and evolution • If a database is restructured, it is possible to change the views it is not necessary to re-formulate the queries written before the restructuring and present in the applications that have already been developed

7

9



DROP VIEW ViewName; · Cancelling a table that a view refers to can have various effects Cancelling · automatic elimination of the associated views views • automatic invalidation of the associated views · prohibition to execute the operation of cancelling the • the effect depends on the specific DBMS

8

Updating views • It is possible to update the data in a view only for some types of views Only views in which a single row of each table corresponds to a single row of the view can be updated (Standard SQL-92) univocal correspondence between the tuple of the view and the tuple of the table on which it is defined it is possibile to propagate without ambiguity the changes made to the view to each table on which it is defined It is not possible to update a view which in the outermost block of the query that defines it: - does not contain the primary key of the table on which it is defined · contains joins that represent one-to-many or many-to-many matches contains aggregated functions contains the DISTINCT keyword Some non-updatable views can become updatable by modifying the SQL expression associated with the view • it may be necessary to reduce the information content of the view

Example n.1 View SUPPLIER CITY CREATE VIEW SUPPLIER_CITY AS SELECT SId, City

Example n.1: insertion

• Insertion in SUPPLIER_CITY of

('S10', 'Rome')

• corresponds to the insertion in S of

('S10',NULL,NULL,Rome')

• the attributes SName, #Employees have to admit the value NULL

Example n.1: deletion

• Deletion from SUPPLIER_CITY of

('S1', 'London')

• corresponds to the deletion from S of

('S1', 'Smith',20',London')

• identification of the tuple to delete is enabled by the primary key

12 13

• update in SUPPLIER_CITY of

('S1,' 'London') to ('S1,' 'Milan')

• update in S of

('S1,' 'Smith',20,'London') to ('S1,' 'Smith',20,'Milan')

• identification of the tuple to be updated is enabled by the primary key

Example n.1: updating

• The view SUPPLIER_CITY can be updated

• each tuple of the view corresponds to a single tuple of table S

• the changes carried out on the view can be propagated to the table on which it is defined

14 15

Example n.2

• View NUMEMPLOYEE_CITY

CREATE VIEW NUMEMPLOYEE_CITY AS
SELECT DISTINCT #Employees, City
FROM S;

DMG

16

Esempio n.2: insertion

• Insertion in NUMEMPLOYEE_CITY of

(40, 'Rome')

• it is impossible to insert in S

(NULL, NULL, 40, 'Rome')

• the value of the primary key is missing

16 17

• Deletion from NUMEMPLOYEE_CITY of (20, 'London') • several tuples are associated with the pair (20, 'London') • Which tuple has to be deleted from S?

 D_{G}^{B}

22

• Update in NUMEMPLOYEE_CITY of

(20, 'London') to (30, 'Rome')

• Several tuples are associated with the pair (20, 'London')

• Which tuple has to be updated in S?

18 19

• The view NUMEMPLOYEE_CITY cannot be updated • the primary key of table S is not present in the view • the insertion of new tuples in the view cannot be propagated to S • some tuples of the view correspond to several tuples in the table S • the association between the tuples in the view and the tuples in the table is ambiguous • it is not possible to propagate the changes carried out on the tuples of the view to the tuples of the table on which it is defined

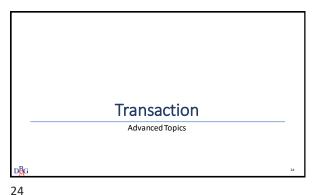
CREATE VIEW SUPPLIER_LONDON AS SELECT * FROM S WHERE City='London';

• The view is non-updatable
• it does not explicitly select the primary key of table S
• It is sufficient to replace the symbol "*" with the name of the attributes

20 21

Example 4: non-updatable view CREATE VIEW BEST_SUPPLIER (SId, SName) AS SELECT DISTINCT SId, SName FROM. S, SP WHERE S.SId = SP.SId AND Qty >100; • The view is not updatable • there is a join • the DISTINCT keyword is present

Example n.4: changed view CREATE VIEW BEST_SUPPLIER (SId, SName) AS SELECT SId, SName FROM S WHERE SId IN (SELECT SId FROM SP WHERE Qty>100); • The view is updatable • the join was removed using the IN operator • the keyword DISTINCT is no longer necessary It is not always possible to rewrite the query to make the view updatable



 A transaction is necessary when several users can simultaneous access the same data • It provides efficient mechanisms to · manage competing access to data recovery after a malfunction It is a logical unit of work, which cannot be further broken down a sequence of data modification operations (SQL statements) that takes the database from one consistent state to another consistent state Transaction there is no need to maintain consistency in intermediate states A system that makes a mechanism available for definining and executing transactions is called a transactional system The DBMS contains architecture blocks that offer transaction management services

25

Beginning a transaction • To define the beginning of a transaction, the SQL language uses the instruction START TRANSACTION • Usually the instruction to begin a transaction is omitted the beginning is implicit for the first SQL instruction of the programme that accesses the database · the first SQL instruction following the instruction ending the previous transaction

Ending a transaction $\bullet \ \ \text{The SQL language has instructions for defining the end of} \\$ a transaction Transaction successful COMMIT [WORK] the action associated with the instruction is called *commit* Transaction failed • the action associated with the instruction is called *abort*

27 26

Commit · Action executed when a transaction ends with success • The database is in a new (final) correct state • The changes to the data executed by the transaction become permanent • visibile to other users

Rollback • Action executed when a transaction ends because of an error • for example, an error in application • All the operations modifying the data executed during the transaction are "cancelled" • The database returns to the state prior to the beginning of the transaction • the data is visible again to the other users

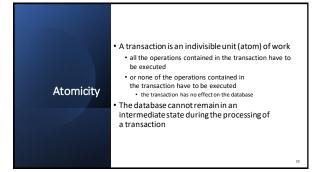
Example • Transfer the sum of 100 from current account number IT92X0108201004300000322229 • to current account number IT32L0201601002410000278976 START TRANSACTION; UPDATE Account SET Balance= Balance - 100 WHERE IBAN='IT92X0108201004300000322229'; UPDATE Account SET Balance = Balance + 100 WHERE IBAN= 'IT32L0201601002410000278976'; COMMIT; D_{G}^{B}

Properties of transactions • The principal properties of transactions are Atomicity Consistency Isolation Durability • They are summarized by the English acronym ACID

 $D_{G}^{B}G$

31

30



• The execution of a transaction has to take the database · from an initial state of consistency (correct) • to a final state of consistency • Correctness is verified by integrity constraints defined on the database Consistency · When there is a violation of the integrity constraint the system intervenes · to abort the transaction · or to modify the state of the database by eliminating the violation of the constraint

32 33

 The execution of a transaction is independent from the simultaneous execution of other The effects of a transaction are not visible by other transactions until the transaction is terminated **Isolation** • the visibility of unstable intermediate states is avoided an intermediate state can be aborted by a subsequent rollback in case of a rollkback, it would be necessary to rollback other transactions that have observed the intermediate state (domino

 The effect of a transaction that has executed a commit is memorized permanently the changes to the data carried out by a transaction ending successfully are permanent after a commit Durability It guarantees the reliability of the operations of data modification the DBMS provides mechanisms for recovery to the correct state of the database after a malfunction has occurred

34 35

Access control Advanced topics D_{G}^{B}

Data security • Protection of data from unauthorized readers alteration or destruction $\bullet \ \ The \ DBMS \ provides \ protection \ tools \ which \ are \ defined \ by$ the database administrator (DBA) • Security control verifies that users are authorized to execute the operations they request • Security is guaranteed through a set of constraints • specificied by the DBA in an appropriate language • memorized in the data dictionary system

Resources

- Any component of the database scheme is a resource
 - table
 - view
 - · attribute in a table or view
 - domain
- procedure
- Resources are protected by the definition of access privileges

38

36

Access privileges

- Describe access rights to system resources
- $\bullet \ \mathsf{SQL} \ \mathsf{provides} \ \mathsf{very} \ \mathsf{flexible} \ \mathsf{access} \ \mathsf{control} \ \mathsf{mechanisms} \ \mathsf{for} \ \mathsf{specifying}$
 - the resources users can access
 - the resources that have to remain private

39

37

Privileges: characteristics

- Each privilege is characterized by the following information
 - · the resource it refers to

• the type of privilege describes the action allowed on the resource • the user granting the privilege • the user receiving the privilege \bullet the faculty to transmit the privilege to other users Types of privilege SELECT • enables the insertion of a · enables using the resource in a new object in the resource query valid for tables and views valid for tables and views • REFERENCES • UPDATE enables referring to a resource in the definition of a table scheme enables updating the value of be associated with tables and attributes valid for tables, views and attributes • DELETE USAGE · enables removal of objects from enables use of the resource (e.g. a new type of data) in the definition of new schemes the resource · valid for tables and views

40 41

Resource creator privileges Resource creator System administrator • When a resource is created, the • The system administrator (user system grants all privileges over system) possesses all privileges that resource to the over all the resources user who created it Only the resource creator has the privilege to eliminate a resource (DROP) and modify a scheme (ALTER) the privilege to eliminate and modify a resource cannot be g ranted to any other user

Management of privileges in SQL

Privileges are granted or revoked using SQL instructions

GRANT

grants privileges over a resource to one or more users

REVOKE

revokes privileges granted to one or more users

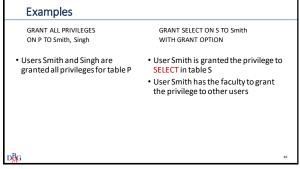
42 43



GRANT PrivilegeList ON ResourceName
TO UserList
[WITH GRANT OPTION]

• PrivilegeList
• specifies the list of privileges
• ALL PRIVILEGES
• Keyword for identifying all privileges
• ResourceName
• specifies the resource for which the privilege is granted
• UserList
• Specifies the users who are granted the privilege
• WITH GRANT OPTION
• faculty to transfer the privilege to other users

44 45



REVOKE PrivilegeList ON ResourceName FROM UserList [RESTRICT [CASCADE]

• Can remove
• all the privileges that have been granted
• a subset of privileges granted

• RESTRICT
• the command must not be executed if revoking the user's privileges entails revoking other privileges.
• bample: the user has received the privilege with the GRANT Orticl and has propagated the privileges to other users
• default value

• CASCADE
• revokes also all the privileges which have been propagated • generates a chain reaction
• for each privilege revoked
• all granted privileges are revoked in a cascade
• all database elements which have been created exploiting these privileges are removed

Examples REVOKE UPDATE ON P FROM White REVOKE SELECT ON S FROM Red CASCADE • User White's privilege to **UPDATE** User Red's privilege to SELECT table S is revoked table P is revoked • the command is not executed if it • User Red had received the privile entails revoking the privilege of ge through GRANTOPTION other users if Red has propagated the privilege to other users, the privilege is rev oked in cascade • if Red has created a view using the SELECT privilege, the view is removed

Concept of role • The role is an access profile • Defined by its set of privileges · Each user has a defined role • it enjoys the privileges associated with that role Advantages access control is more flexible a user can have different roles at different times • it simplifies administration an access profile need not be defined at the moment of its activation · it is easy to define new user profiles

48 49



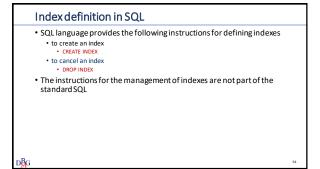


51

Physical data organization • In a relational DBMS the data are represented as collections of records memorized in one or more files • the physical organization of the data in a file influences the time required to access the information • each physical data organization makes some operations efficient and others cumbersome • There is no physical data organization that is efficient for any type of data reading and writing

Indexes • Indexes are the accessory physical structures provided by the relational DBMS to improve the efficiency of data access operations indexes are realized using different types of physical structures trees hash tables • The instructions for managing the indexes are not part of the

52 53



CREATE INDEX NomeIndice ON NomeTabella (ElencoAttributi) • The order in which the attributes appear in AttributeList is important • the order of the index keys is first on the basis of the first attribute in AttributeList
equal in value to the first attribute on the values of the second attribute **CREATE INDEX** • and so on, in order, until the last attribute Use the minimum number of attributes, usually one

54 55

Example

• Creation of an index on the attribute Residence of the table **EMPLOYEE**

> CREATE INDEX ResidenceIndex ON EMPLOYEE (Residence)

- The index is jointly defined on the two attributes
- The index keys are ordered
 - first on the basis of the value of the attribute Surname
 - of equal value to the attribute Surname, on the value of the attribute Name

56 57



DROP INDEX NomeIndice

- Eliminate the index with the name IndexName
- This command is used when
 - the index is no longer utilized
 - the improvement in performance is
 - reduced reduction in response time for the
 - slowing down of updates due to index maintenance

Physical design Advanced Topics

Physical design: input data

- · Logical scheme of the database
- · Characteristics of the chosen DBMS
 - physically available options
 - physical memory structures
 indexes
- Data volumes
 - · cardinality of tables
 - · cardinality and distribution of the attribute values domain
- Estimate of application load
 - · most important queries and their frequency
 - most important updating operations and their frequency
- response time requirements for important queries/updates

Physical design: result

- Physical scheme of the database
 - physical organization of tables
- Memorization and operating parameters
 - \bullet Initial file sizes, expansion possibilities, free space at outset, \ldots

 $D_{G}^{B}G$ 60

Procedure

- Physical design is carried out empirically, using a trial and error approach
 - there are no reference methodologies
- $\bullet \ Characterization \ of the \ application \ load$
 - · for each important query it is necessary to define
 - access relationships
 - attributes to be viewed
 - · attributes involved in selections/joins
 - degree of selectivity of selection conditions
 - for each important update it is necessary to define type of update (Insertion, cancellation, modification)
 - relation to any attributes involved
 - · degree of selectivity of selection conditions

61

Procedure: choices to be made

- Choices to be made
- physical structuring of the files containing the tables
- choice of attributes to index
 - driven by estimating applicative load and data volume
 - definition of type for each index
 - e.g. hash or B-tree
- any variations of the scheme
 - horizontal partitioning in the secondary memory
 - denormalization of tables
 - used in data warehouses

Tuning

- If the result is not satisfactory
 - Tuning, adding and removing indexes
- This is a procedure guided by the availability of tools that enable
 - verification of the execution plan adopted by the chosen DBMS
 - the execution plan defines the sequence of activities carried out by the DBMS to execute a query
 - data access methods
 join methods
 - assess the execution cost of various alternatives

63