



## Introduction

- The destruction of database is generally caused by the following factors:
  - System failure
  - Inconsistency caused by concurrent access
  - Man-caused destruction (intentionally or accidentally)
  - The data inputted is incorrect, the updating transaction didn't obey the rule of consistency preservation
- In above factors, 1 and 2 should be resolved by recovery mechanism of DBMS (Chapter 4); 3 belongs to database security; 4 belongs to integrity constraints



#### Security of Database

- Protect databases not be accessed illegally.
  - View and query rewriting

  - General user
    - User with resource privilege
       DBA
  - > Identification and authentication of users
    - Password

    - Special articles, such as key, IC card, etc.
       Personal features, such as fingerprint, signature, etc.

  - Authorization
    GRANT CONNECT TO JOHN IDENTIFIED BY xyzabc;
    GRANT SELECT ON TABLE S TO UI WITH GRANT OPTION;
  - > Role

  - Audit trail
    - AUDIT SELECT, INSERT, DELETE, UPDATE ON emp WHENEVER SUCCESSFUL;



#### **Security of Statistical Database**

- In many situation, the statistical data is public while the detailed individual data is secret.  $\,$
- Public statistical database
- But some detailed individual data can be derived from public statistical data
- ➤ How prevent this leak? --- not a easy thing

NAME	SEX	DEPENDENTS	OCCUPATION	SALARY
Wang	M	2	Programmer	120
Chang	F	2	Manager	240
Chen	F	0	Programmer	140
Li	F	2	Engineer	160
Liu	M	2	Clerk	110
Zhu	F	1	Teacher	80
Zhao	M	0	Professor	180
Sun	M	1	Teacher	110
Xu	F	2	Programmer	130
Ma	F	1	Programmer	150



## 🛛 🙀 Individual Tracker

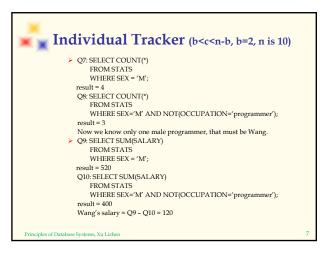
 Suppose we know Wang is a male programmer, and salary in STATS is secret but other information is public, we can get wang's salary from public data.

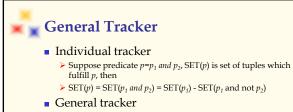
```
➤ Q1: SELECT COUNT(*)
     FROM STATS
     WHERE SEX='M' AND OCCUPATION='programmer'
  result = 1
 Q2: SELECT SUM(SALARY)
     FROM STATS
     WHERE SEX='M' AND OCCUPATION='programmer';
 result = 120
```



#### Individual Tracker (c>b, b=2)

Q3: SELECT COUNT(\*) FROM STATS; result = 10 Q4: SELECT COUNT(\*) FROM STATS WHERE NOT(SEX='M' AND OCCUPATION='programmer'); result = 9 Now we know only one male programmer, that must be Wang. Q5: SELECT SUM(SALARY) FROM STATS; result = 1420Q4: SELECT SUM(SALARY) WHERE NOT(SEX='M' AND OCCUPATION='programmer'); result = 1300Wang's salary = Q5 - Q6 = 120





It is a predicate T which fulfill:  $2b \leq |\operatorname{SET}(T)| \leq (\text{n-2b}), \, b \leq n/4$ 

➤ Suppose a tuple *R* can be limited uniquely by predicate *p*, that is  $SET(p) = \{R\}$ , then  $SET(p) = SET(p \text{ or } T) \ \underline{\cup} \ SET(p \text{ or not } T) - SET(T) - SET(not \ T)$ 

U means union without eliminating repeated tuples.



#### Integrity Constraints

- An IC describes conditions that every *legal* instance of a relation must satisfy.
  - ▶ Inserts/deletes/updates that violate IC's are disallowed.
  - Can be used to ensure application semantics (e.g., sid is a key), or prevent inconsistencies (e.g., sname has to be a string, age must be < 200)



## Types of Integrity Constraints

- Static constraints: constraints to database state
  - Inherent constraints (data model), such as 1NF
  - > Implicit constraints: implied in data schema, indicated by DDL generally. Such as domain constraints, primary key constraints, foreign key constraints.
    - Domain constraints: Field values must be of right type. Always enforced.
  - Explicit constraints or general constraints
- Dynamic constraints: constraints while database transferring from one state to another. Can be combined with trigger.



## Database Modification

• If  $\alpha$  is foreign key in  $r_2$  which references to K1 in  $r_1$ , the following tests must be made in order to preserve the following referential integrity constraint:

 $\Pi_{\alpha}(r_2) \subseteq \Pi_{K1}(r_1)$ 

- **Insert.** If a tuple  $t_2$  is inserted into  $r_2$ , the system must ensure that there is a tuple  $t_1$  in  $r_1$  such that  $t_1[K_1] = t_2[\alpha]$ . That is  $t_2[\alpha] \in \prod_{K1} (r_1)$
- **Delete.** If a tuple,  $t_1$  is deleted from  $r_1$ , the system must compute the set of tuples in  $r_2$  that reference  $t_1$ :

 $\sigma_{\alpha = f1[K1]}(r_2)$ 

If this set is not empty, either the delete command is rejected as an error, or the tuples that reference  $t_1$  must themselves be deleted (cascading deletions are possible).



#### Database Modification (Cont.)

- Update. There are two cases:
  - ▶ If a tuple  $t_2$  is updated in relation  $r_2$  and the update modifies values for foreign key  $\alpha$ , then a test similar to the insert case is made. Let  $t_2$ ' denote the new value of tuple  $t_2$ . The system must ensure that

 $t_2'[\alpha]\in\prod_{\mathrm{K1}}(r_1)$ 

If a tuple  $t_1$  is updated in  $r_1$ , and the update modifies values for the primary key  $(K_1)$ , then a test similar to the delete case is made. The system must compute

 $\sigma_{\alpha=t1[K1]}\left(r_{2}\right)$ 

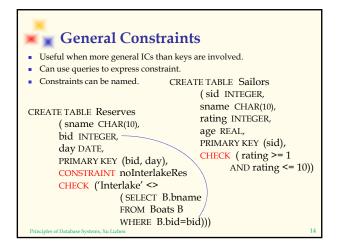
using the old value of  $t_1$  (the value before the update is applied). If this set is not empty, the update may be rejected as an error, or the update may be cascaded to the tuples in the set, or the tuples in the set may be deleted



#### Definition of Integrity Constraints

- Indicated with procedure
  - ➤ Let application programs responsible for the checking of integrity constrains.
- Indicated with ASSERTION
  - Defined with assertion specification language, and checked by DBMS automatically
    - > ASSERT balanceCons ON account: balance>=0;
- Indicated with CHECK clause in base table definition, and checked by DBMS automatically

rinciples of Database Systems, Xu Lizhen





#### **Constraints Over Multiple Relations**

CREATE TABLE Sailors
( sid INTEGER, sname CHAR(10), rating INTEGER, age REAL, PRIMARY KEY (sid), CHECK
( (SELECT COUNT (S.sid) FROM Sailors S) + (SELECT COUNT (B.bid) FROM Boats B) < 100)

- Awkward and wrong!
- If Sailors is empty, the number of Boats tuples can be anything!

Principles of Database Systems, Xu Lizhe



#### Assertion

 ASSERTION is the right solution; not associated with either table.

CREATE ASSERTION smallClub CHECK ( (SELECT COUNT (S.sid) FROM Sailors S) +(SELECT COUNT (B.bid) FROM Boats B) < 100 )

Principles of Database Systems, Xu Lizhen

16



#### **Triggers**

- Trigger: procedure that starts automatically if specified changes occur to the DBMS
- Three parts:
  - ➤ Event (activates the trigger)
  - > Condition (tests whether the triggers should run)
  - > Action (what happens if the trigger runs)
- Active database rules (ECA rules)

Principles of Database Systems, Xu Lizhen



## 🗽 Triggers: Example (SQL:1999)

CREATE TRIGGER youngSailorUpdate AFTER INSERT ON SAILORS REFERENCING NEW TABLE NewSailors FOR EACH STATEMENT INSERT

> INTO YoungSailors(sid, name, age, rating) SELECT sid, name, age, rating FROM NewSailors N WHERE N.age <= 18

18



# Execution of Rules

- Immediate execution √
- Deferred execution
- Decoupled or detached mode
- Cascading trigger
  - ➤ Control nested execution of rules
  - ➤ Prevent nontermination
    - ➤ Triggering graph
    - ➤ Specify the upper limit of cascading times
  - ➤ So triggers should be used reasonably



# Implementation of ECA

- Loosely coupling
- Tightly coupling (DB2, Oracle, etc.)
- Nested method

The rules are nested into transaction and executed by DBMS as a part of the transaction.

- ➤ Grafting method
- ➤ Query modification method