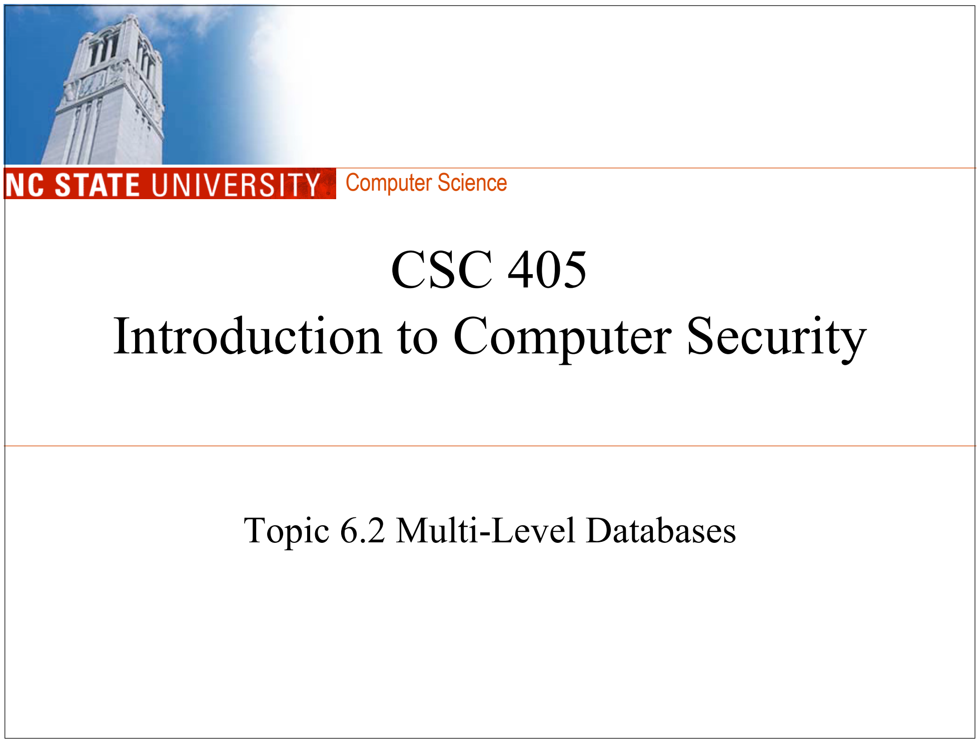
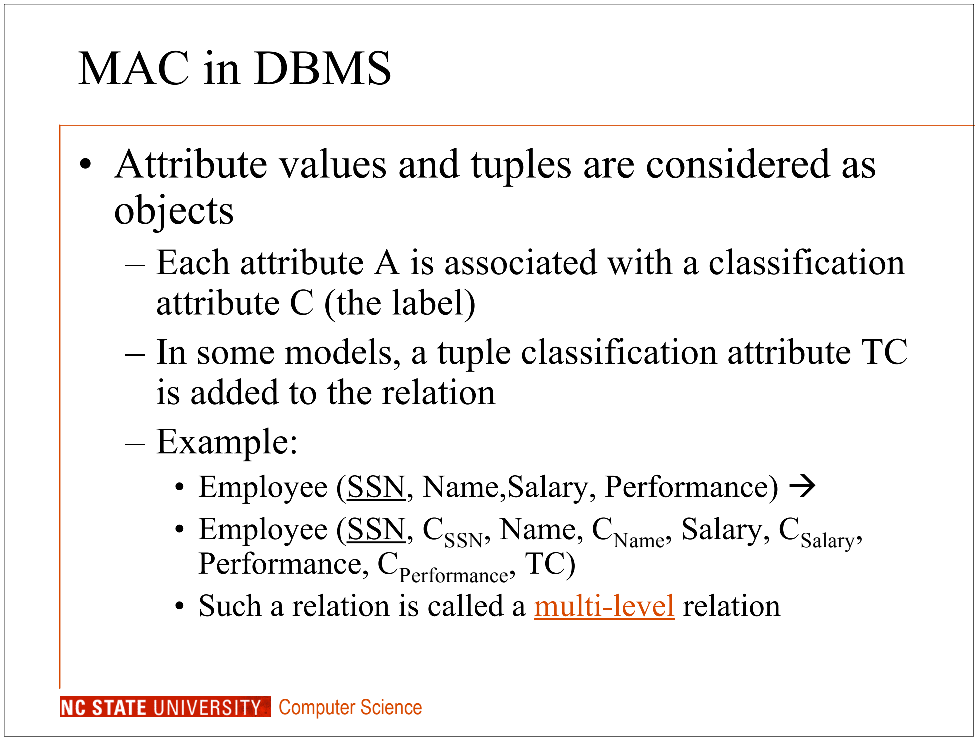


NC STATE UNIVERSITY Computer Science

CSC 405
Introduction to Computer Security

Topic 6.2 Multi-Level Databases



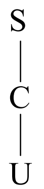
MAC in DBMS

- Attribute values and tuples are considered as objects
  - Each attribute A is associated with a classification attribute C (the label)
  - In some models, a tuple classification attribute TC is added to the relation
  - Example:
    - Employee (SSN, Name,Salary, Performance) →
    - Employee (SSN, C_{SSN}, Name, C_{Name}, Salary, C_{Salary}, Performance, $C_{Performance}$, TC)
    - Such a relation is called a multi-level relation

NC STATE UNIVERSITY Computer Science

Employee

SSN	C_S	Name	C_N	Salary	C_S	Performance	C_P	TC
111111111	U	Smith	U	40000	C	Fair	S	S
222222222	C	Brown	C	80000	S	Good	C	S



Employee (What class C users' see)

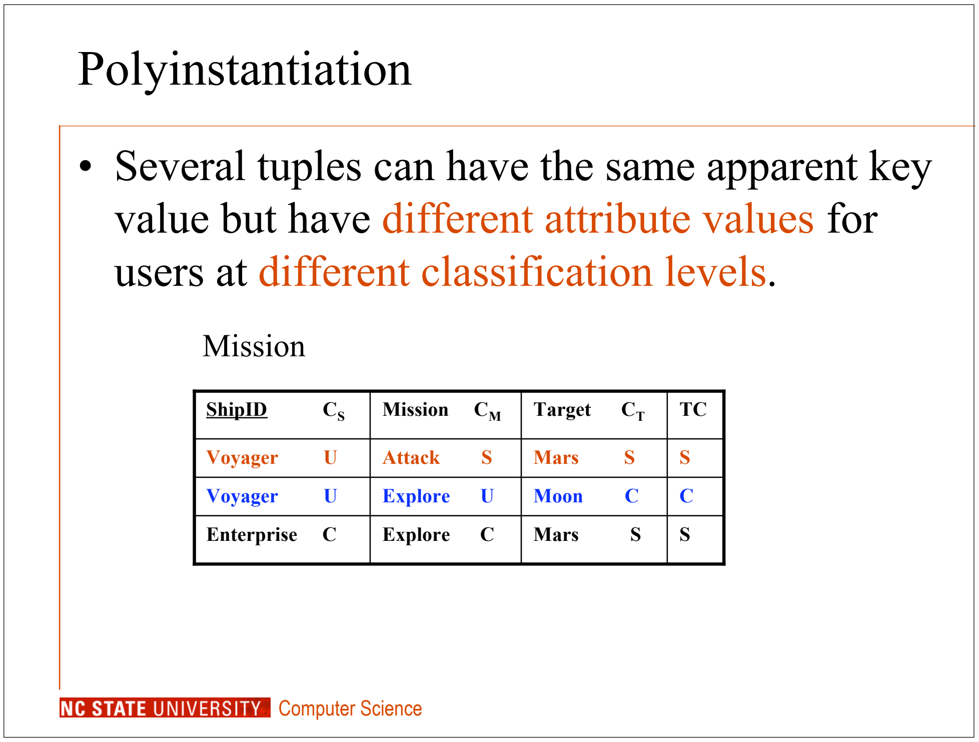
SSN	C_S	Name	C_N	Salary	C_S	Performance	C_P	TC
111111111	U	Smith	U	40000	C	Null	C	C
222222222	C	Brown	C	Null	C	Good	C	C

Employee (What class U users' see)

SSN	C_S	Name	C_N	Salary	C_S	Performance	C_P	TC
111111111	U	Smith	U	Null	U	Null	U	U

MAC in DBMS (Cont'd)

- Employee (SSN, C_{SSN}, Name, C_{Name}, BDate, C_{BDate}, Salary, C_{Salary}, TC)
- Primary key:
  - The set of attributes that can uniquely identify each tuple.
- Apparent key:
  - The set of attributes that would have formed the primary key in a regular (single-level) relation.



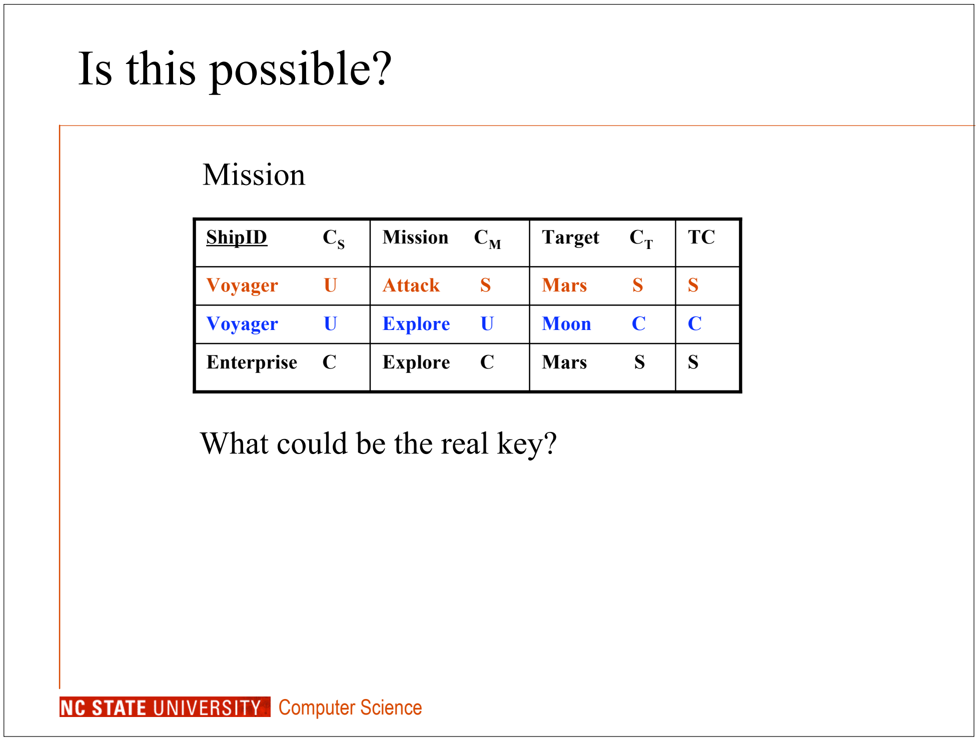
Polyinstantiation

- Several tuples can have the same apparent key value but have different attribute values for users at different classification levels.

Mission

ShipID	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Attack	S	Mars	S	S
Voyager	U	Explore	U	Moon	C	C
Enterprise	C	Explore	C	Mars	S	S

NC STATE UNIVERSITY Computer Science



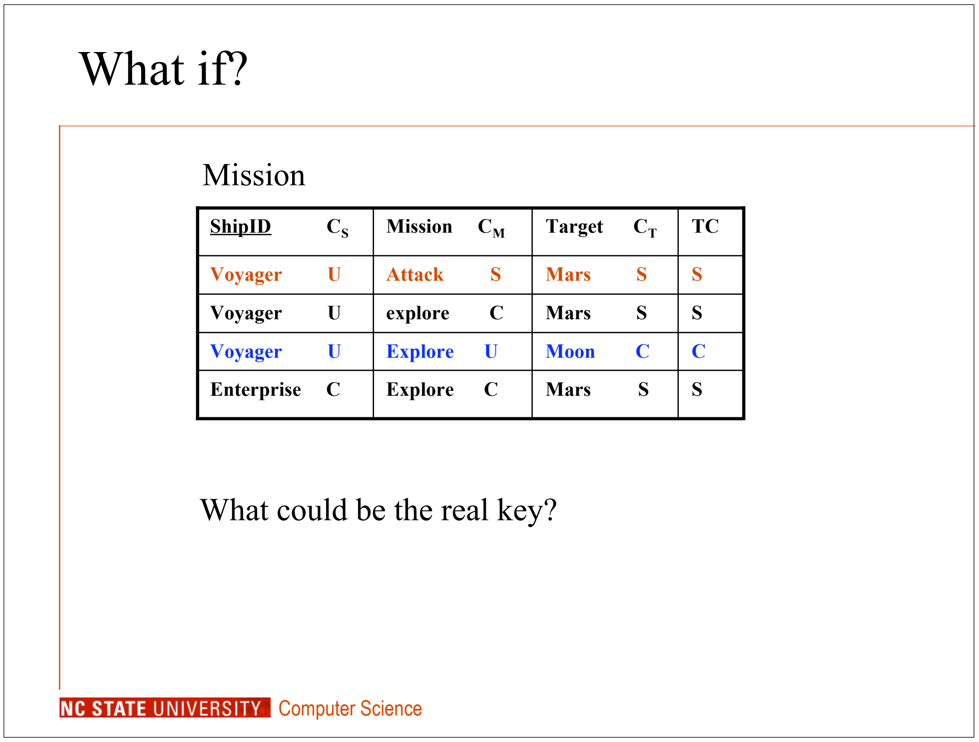
Is this possible?

Mission

ShipID	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Attack	S	Mars	S	S
Voyager	U	Explore	U	Moon	C	C
Enterprise	C	Explore	C	Mars	S	S

What could be the real key?

NC STATE UNIVERSITY Computer Science



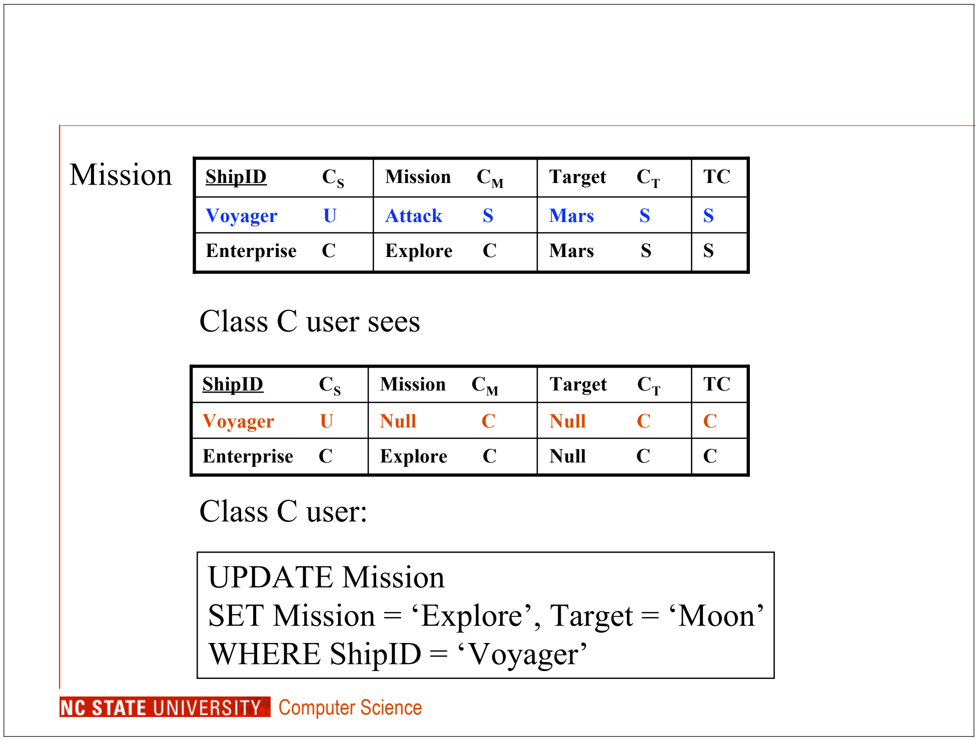
What if?

Mission

<u>ShipID</u>	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Attack	S	Mars	S	S
Voyager	U	explore	C	Mars	S	S
Voyager	U	Explore	U	Moon	C	C
Enterprise	C	Explore	C	Mars	S	S

What could be the real key?

NC STATE UNIVERSITY Computer Science



Mission

<u>ShipID</u>	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Attack	S	Mars	S	S
Enterprise	C	Explore	C	Mars	S	S

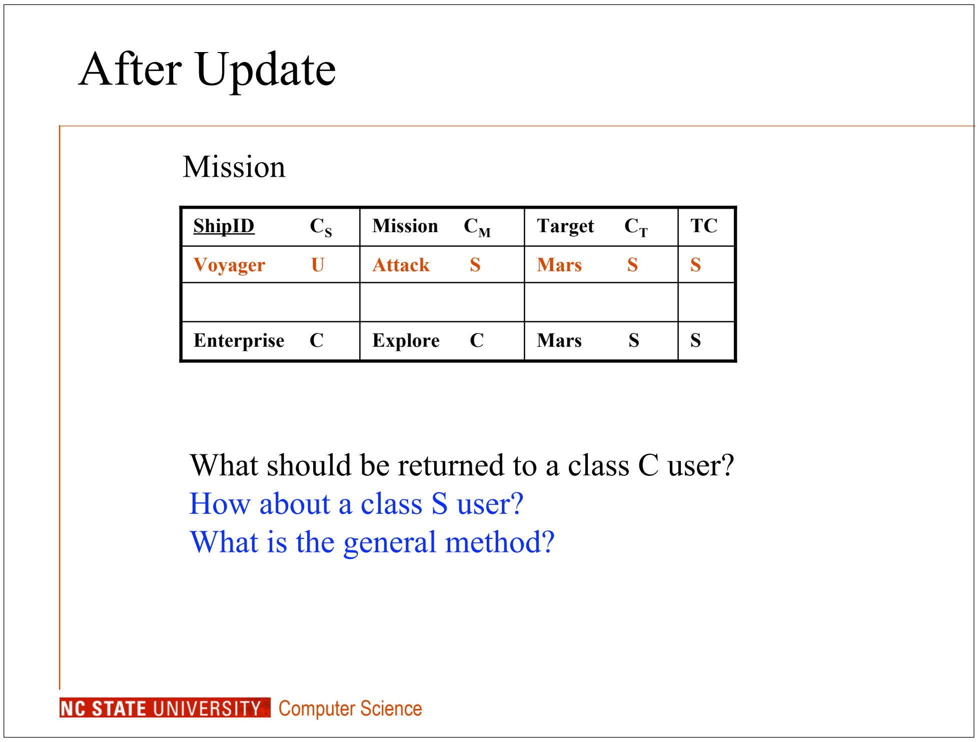
Class C user sees

<u>ShipID</u>	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Null	C	Null	C	C
Enterprise	C	Explore	C	Null	C	C

Class C user:

```
UPDATE Mission
SET Mission = ‘Explore’, Target = ‘Moon’
WHERE ShipID = ‘Voyager’
```

NC STATE UNIVERSITY Computer Science



After Update

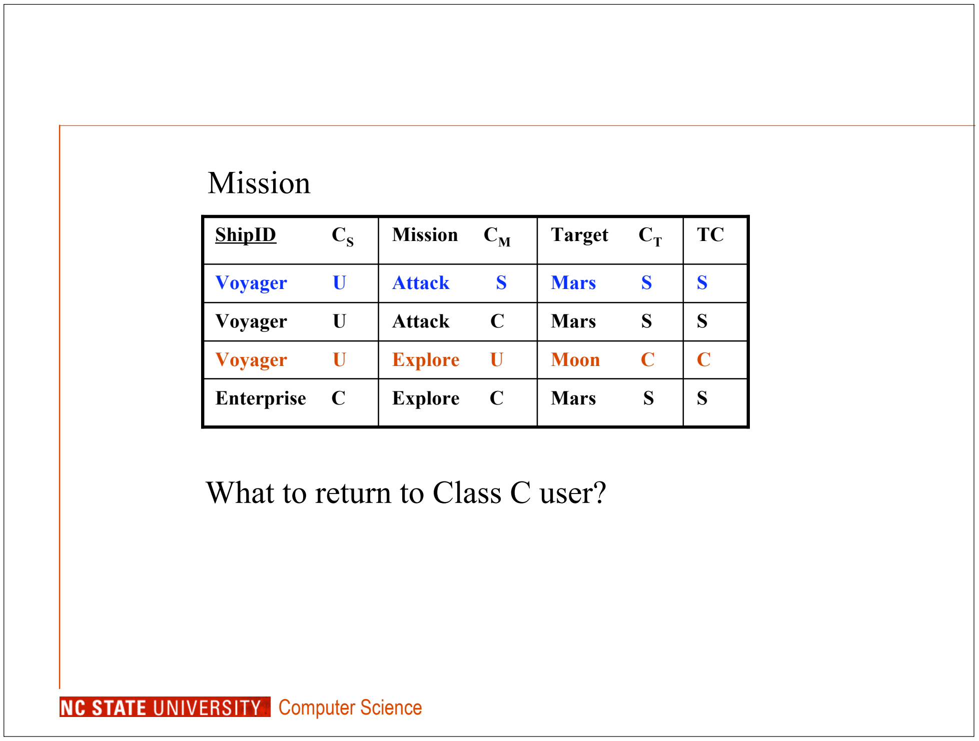
Mission

<u>ShipID</u>	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Attack	S	Mars	S	S
Enterprise	C	Explore	C	Mars	S	S

What should be returned to a class C user?

How about a class S user?

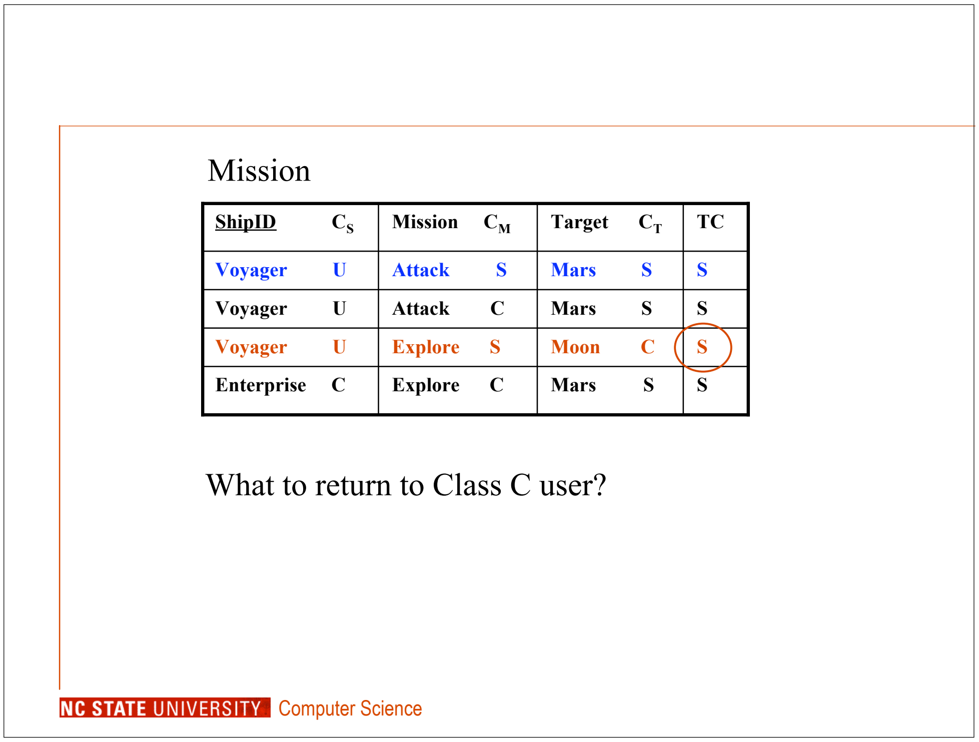
What is the general method?



Mission

<u>ShipID</u>	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Attack	S	Mars	S	S
Voyager	U	Attack	C	Mars	S	S
Voyager	U	Explore	U	Moon	C	C
Enterprise	C	Explore	C	Mars	S	S

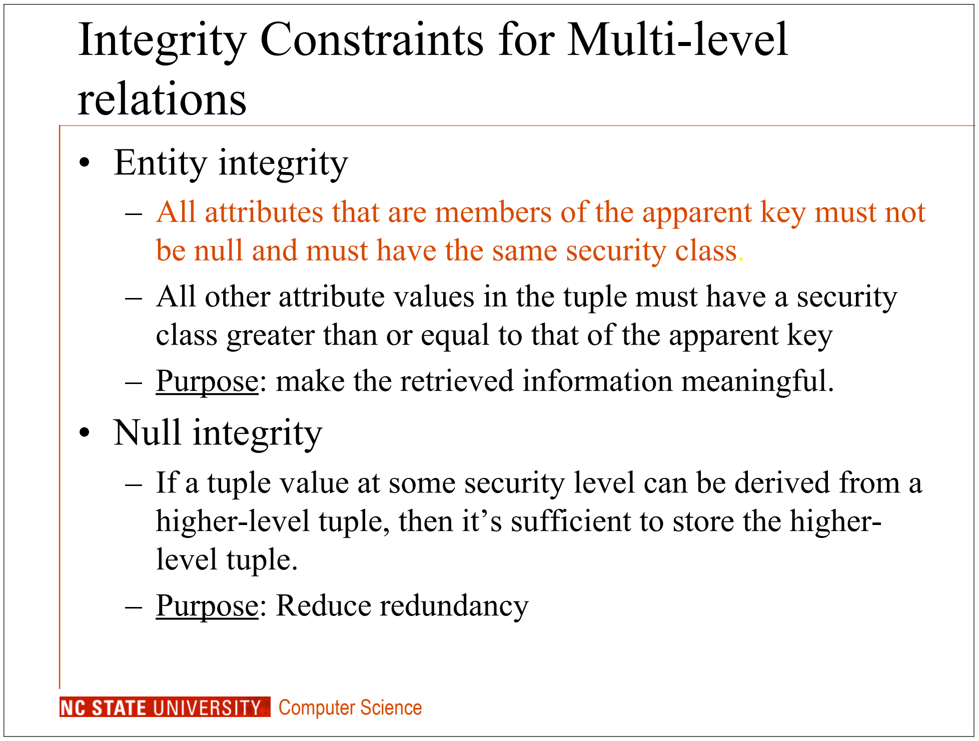
What to return to Class C user?



Mission

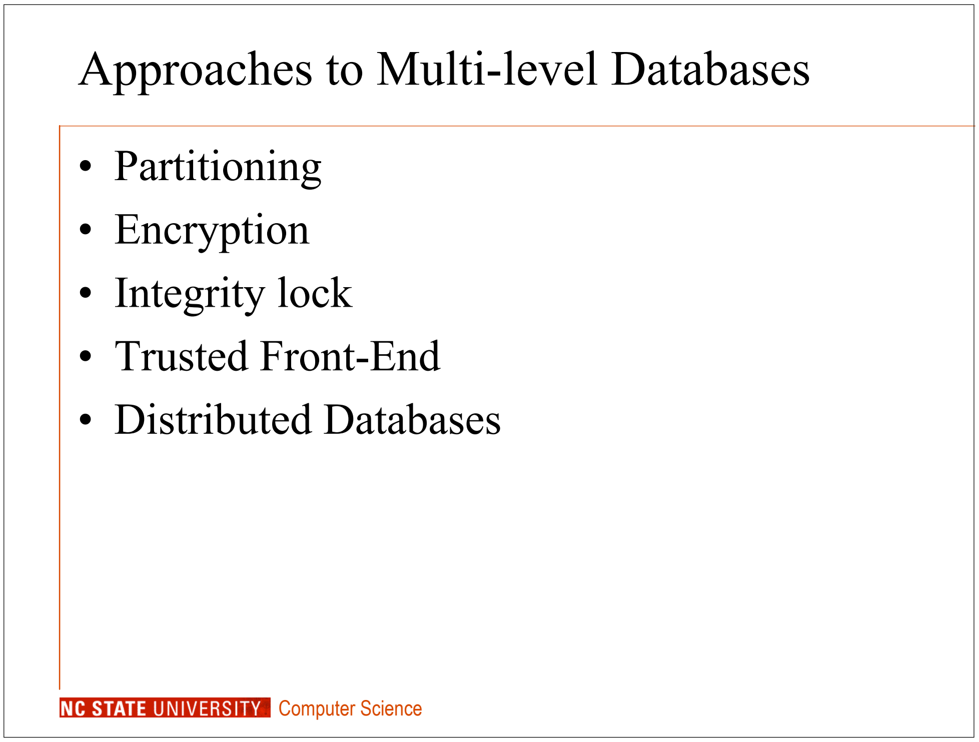
ShipID	C_S	Mission	C_M	Target	C_T	TC
Voyager	U	Attack	S	Mars	S	S
Voyager	U	Attack	C	Mars	S	S
Voyager	U	Explore	S	Moon	C	S
Enterprise	C	Explore	C	Mars	S	S

What to return to Class C user?



Integrity Constraints for Multi-level relations

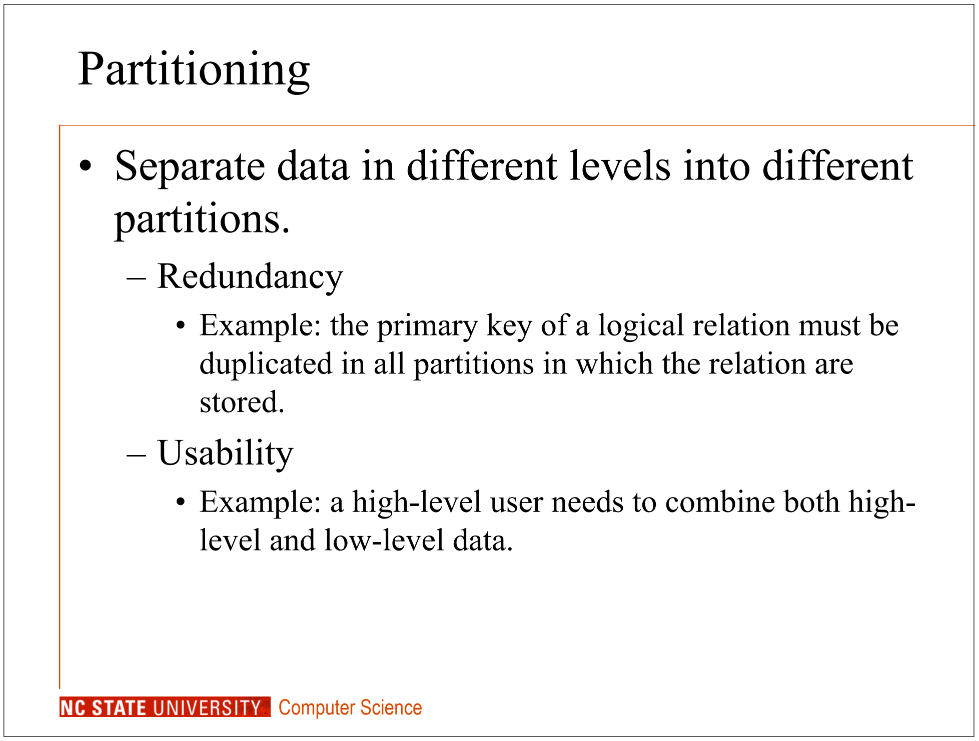
- Entity integrity
  - All attributes that are members of the apparent key must not be null and must have the same security class.
  - All other attribute values in the tuple must have a security class greater than or equal to that of the apparent key
  - Purpose: make the retrieved information meaningful.
- Null integrity
  - If a tuple value at some security level can be derived from a higher-level tuple, then it's sufficient to store the higher-level tuple.
  - Purpose: Reduce redundancy



Approaches to Multi-level Databases

- Partitioning
- Encryption
- Integrity lock
- Trusted Front-End
- Distributed Databases

NC STATE UNIVERSITY Computer Science



Partitioning

- Separate data in different levels into different partitions.
  - Redundancy
    - Example: the primary key of a logical relation must be duplicated in all partitions in which the relation are stored.
  - Usability
    - Example: a high-level user needs to combine both high-level and low-level data.

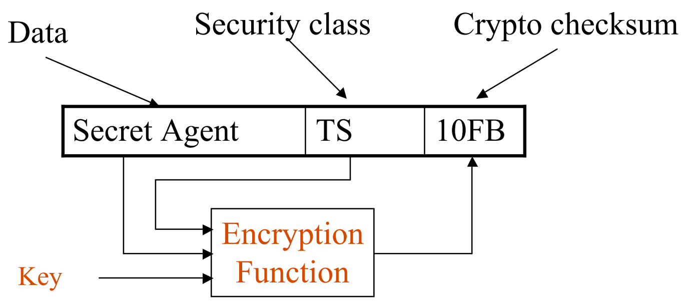
NC STATE UNIVERSITY Computer Science

Encryption

- Encrypt the sensitive data at each level with a key unique to that level.
  - Known plaintext attack
    - Example:
      - Party attribute is encrypted.
      - Alice knows party="Democrat" for Bob; she can compare the ciphertext of Bob's party attribute with other tuples
    - Reason: Limited set of plaintexts.
  - Authentication
    - Example:
      - Replace one ciphertext with another
  - Above problems can be partially avoided with multiple keys.
  - Unable to use DBMS functionalities for encrypted data.
    - Query optimization, indexes, etc.

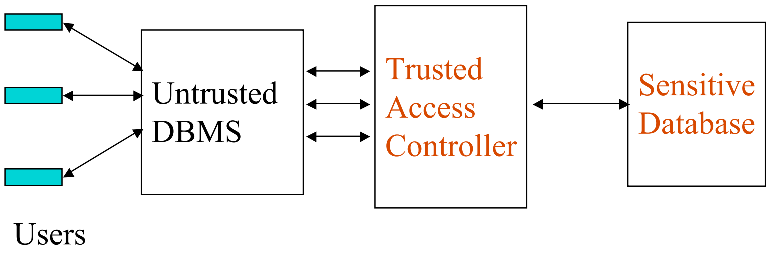
Integrity Lock

- Provide integrity and limited access for a database.



- Any unauthorized changes to data items can be detected.
- Access to data items is based on the security labels.

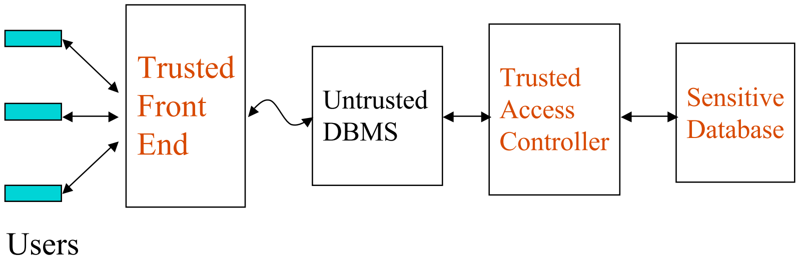
Integrity Lock DBMS



- Problems
  - Efficiency
    - Data expansion
    - Processing time required for generating, modifying, and verifying integrity locks
  - Security
    - Untrusted DBMS sees all data passing through it.

Trusted Front End

- Trusted Front End
  - User authentication
  - Access control
  - Verification
  - Essentially a reference monitor

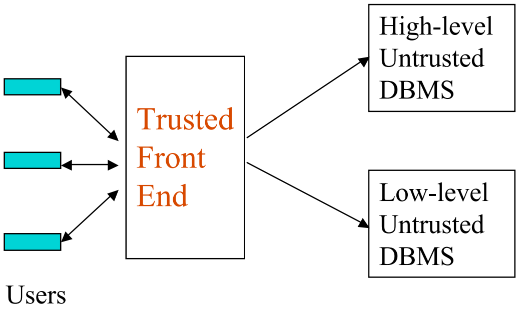


Trusted Front End (Cont'd)

- Commutative Filters
  - Processes that interfaces to both the user and the DBMS.
  - Reformat the query by putting in more conditions to filter out unnecessary records.
  - Example:
    - Retrieve NAME where ((Occup= Physicist) ^ (City =WashDC))
      From all records R
    - After reformatting
    - Retrieve NAME where ((Occup= Physicist) ^ (City =WashDC))
      From all records R where
      (Name-level (R) <= User-level) ^
      (Occup-level (R) <= User-level) ^
      (City-level (R) <= User-level)

Distributed Databases

- Store data items at different level in different physical databases
- Trusted front-end translates each query into single-level queries and send to different databases
- Trusted front-end combines results and returns to the user.


High-level
Untrusted
DBMS
Trusted
Front
End
Low-level
Untrusted
DBMS
Users