Wild Wood Apartments - Chapter 8 - Lesson 12

Example Solutions for comparision and discussion

Building Managers

Building managers need to have the most access. I am assuming the managers enter all rent payments and maintenance requests, though it is possible to decide that tenants can enter maintenance requests.

Table name	SELECT	INSERT	UPDATE	DELETE	Constraints
Lease	X	X	X		
Building	X				
Apartment	X		X		
Tenant	X	X	X		
RentPayment	X	X	X		
MaintenanceRequest	X	X	X		
MaintenanceRequestDetails	X	X	X		

Tenants

Tenants need no direct access to the database, though it is possible that they could be granted the right to see their maintenance requests.

Owners

Owners mostly need to be able to select from tables in order to create reports. In the chart below they are given full powers over the building and apartment tables.

Table name	SELECT	INSERT	UPDATE	DELETE	Constraints
Lease	X				
Building	X	X	X		
Apartment	X	X	X		
Tenant	X				
RentPayment	X				
MaintenanceRequest	X				
MaintenanceRequestDetails	X				

One could also add a database administrator who has full permissions.

2. Create a security plan that includes authentication and authorization and general policies and procedures. Consider the use of roles, stored procedures, views, and other tools.

The security plan could vary a great deal here, but a few main points would be:

- Authentication mode: Windows, since every user can be guaranteed to have a Windows account.
- Create roles for Manager and Owner.
- Create views and stored procedures for each role to control all data access.

*3. **Documentation**: Document and define all the aspects of your plan.*

This should consist of a more full explanation of something like the above.

4. Create a preliminary threat analysis.

The response should use the threat tables and contain some discussion. The most dangerous user is certainly the manager, because they do most of the data entry and data maintenance.

Role	**Manager**
Threat	**Description**
SELECT	No real threat here
INSERT	Insertion errors in Lease, Tenant, RentPayment, MaintenanceRequest, and MaintenanceDetails
UPDATE DELETE	Update errors, even catastrophic ones, in Customer in Lease, Tenant, RentPayment, MaintenanceRequest, and MaintenanceDetails

Role	**Owner**
Threat	**Description**
SELECT	No real threat here
INSERT	Insertion errors in Building and Apartment

UPDATE DELETE	Update errors, even catastrophic ones, in Building and Apartment

There should be no public access at all. Any public access would be through malicious attack.

Role	**Public**
Threat	**Description**
SELECT	View tenant names and information, view rent information
INSERT	Could insert false or bad data
UPDATE	Could destroy data by unconstrained updates
DELETE	Could remove important data

5. Make a preliminary disaster management plan.

Because this is a corporate environment, the disaster management plan can be a bit more elaborate.

Some suggested features might be:

- Daily (or twice daily) backup (log shipping) of the local database to the corporate databases.
- The corporate level databases could be mirrored off-site in another city somewhere.
- Some sort of controls as to who has access to the manager's computer account, password controls, etc. Policies and procedures for these.
- Perhaps something on encryption of data.

The plan does not have to be full or elaborate but should include some policies about backup, access, etc. It should include some procedural lists for one or two of the tasks in the disaster plan.

6. Create a view of the data that is tailored to the needs of one of your uses.

Students can choose their own view of the data. The view should, however, reflect the "view" one of the stakeholder groups would have on the data. Here is a sample view for Managers to see Tenant and Lease information.

```
CREATEVIEW vw_TenantLease
AS
SELECT TenantLastName AS LastName
, TenantFirstName AS FirstName
, LeaseKey AS LeaseNumber
, l.ApartMentKey AS Apartment
, LeaseMonthlyRent AS MonthlyRent
, LeaseStartDate AS StartDate
, LeaseEndDate AS EndDate
,FROM Tenant t
INNERJOIN Lease l
ON t.TenantKey=l.TenantKey
INNERJOIN ApartMent a
ON l.Apartmentkey=a.ApartmentKey
WHERE Buildingkey='B1'
```

7. For extra credit, create a stored procedure that executes one of the basic activities for your database (making a rent payment, for instance, or a maintenance request).

This is difficult for students at this stage and probably should only be attempted by more advanced students. Here is a relatively simple procedure that could be done:

```
CREATEPROC usp_RentPayment
@RentPaymentKey INT,
@RentPaymentDate DATE,
@RentPaymentAmount MONEY,
@LeaseKey NCHAR(10)
AS
BEGINTRAN
BEGINTRY
INSERTINTO RentPayment(
RentPaymentKey,
RentPaymentDate,
RentPaymentAmount,
LeaseKey)
VALUES(
@RentPaymentKey,
@RentPaymentDate,
@RentPaymentAmount,
@LeaseKey)
COMMITTRAN
ENDTRY
BEGINCATCH
ROLLBACKTRAN
ENDCATCH
```