Please provide your thoughts on following topics:

* Share your experience or conduct a research on complex business rules that require triggers, stored procedures, functions, and packages.
* Do you implement business rules in your database applications instead of using triggers, stored procedures/functions?  If yes, how do you decide which way you want to implement a business rule?
* What is SQL injection and its potential security impacts,and how to prevent it?
* Share your experience on using object-relational features in database projects if you or your colleagues have used them.  The features may be RDBMS-dependent features.
* Discuss pros and cons using object-relational features.  Are object-relational features popular in practice?  Why?
* Share any challenges for developing databases and database applications based on your experience or your database project.

I do not have any current experience in implementing complex business rules with triggers, stored procedures, functions and packages. Some possible examples of business rules (I am using Amazon as an example) is to ensure that a product is in stock before an order is created. Or ensuring that a shipping address is accessible by Amazon shipping before placing an order could be a business rule. Computing the estimated shipping time can be done using a stored procedure based on the current product location and product destination.

For my own database project there will be some business rules that will need to be put in place. My database project is a database system for a hockey league. I have a few business rules in place to ensure that a certain entity exists before it can be created. One example is that a player must exist before a player injury report can be created. Another example is some limitations on the data that can be passed to the database, for example the rink size must be either ‘standard’, ‘olympic’, or ‘half-sheet’, captain status must either be ‘assistant’ or ‘captain’, etc. I have not used stored functions yet, so all of my business rules are built using triggers currently.

SQL injection is a security attack against a database where the attacker enters SQL code into a web form box to gain access to database resources maliciously. Injecting malicious data can have unexpected consequences for the database behavior. the best way to prevent SQL injection attacks is to control the type and number of characters accepted by input, ensure access control over sensitive tables and columns, and to use security software such as AppScan, Netsparker, Fortify Software, etc. to identify application vulnerabilities.

I do not have any experience with object-relational features in my professional, education or personal life. Object relational features allow for expanded data types to be inserted to the database, such as nchar, varchar2, date, bfile, etc. Collection items such as varray and table can be added as well. Other things that can be added are user-defined data types, objects with attributes and member methods, LOB’s such as BLOB, CLOB, etc. While this can allow for greater table flexibility, it can also increase data complexity, and take the relation out of 1NF, 2NF or 3NF, depending on the data type inserted.

Some challenges that I observed working on the database project was ensuring the relationships between each of the relations, managing the migration of foreign keys, applying appropriate constraints to my relations, and ensuring that the data was inserted in the proper sequence. This was a lot of trial and error, but getting things wrong and seeing error messages helped me a lot to learn further how to create databases in a way that makes sense and will work.