Web 2.0 has provisioned development of many data intensive applications, using different frameworks. These frameworks are used to Object Relational Mapping systems. This paper studies the use of one such framework, i.e Rails, as it is one of the languages widely used in database systems and operations, and investigates the database concurrency control mechanisms particularly feral mechanisms for maintaining the integrity, data corruption due to the limitations of the database systems because of the concurrency errors. Different aspects of the applications such as the total models, validations, transactions, associations were also taken into consideration.

Rails divides the architecture in two parts: One where data modeling and schema description is described, and the other where the presentation and business logic is written. Rails has a MVC (Model-View-Controller) architecture. Thus, the paper focuses on the model layer, where database querying, scheme and persistence functionality is handled. This model is known as Active Record. The architecture of the Rails is such that it processes all the requests concurrently. This concurrency is defined by four main mechanisms: Transactions, Locking, Validation, Associations, of which Validation and Association are integral part of feral concurrency control mechanism of Rails. Transactions contain the business logic. Validations are used to mention checks on the attributes in a model. Associations mention the relations i.e, referential integrity among different Active Records. Out of these all mechanisms, Validations and Associations are the primary form of the concurrency controls.

Tests are performed upon these mechanisms, using the invariant confluence theory. The invariant confluence theory defines required conditions for deciding if the invariants preserve under the coordination free and concurrent transactions. The principle ensures that the transactions can run simultaneously, and a correct state can be produced as a result of these transactions. Validations defined by the system, i.e built in validations, work well under concurrent operations. Most of the user defined validations work well under concurrent operations, however, there are few validations which could be of concern for the concurrent execution. When quantifying the anomalies of the feral mechanism, we can see that as we scale up the number of concurrent servers running, we can see the unique validations and association validations are broken. Around 86.9 % of the tests are passed, while the remaining 13.1 % fails, under the principle of invariance confluence theory.

Strong Points:
1). There is a significant reduction in inconsistency, because of the validations, due to less workload.
2). Due to the implementation of the schemes in different databases, along with the PostgreSQL, a bug was pointed out in PostgreSQL pertaining to Serializable Snapshot Isolation.

3). The paper provides a good brief summary of different ORM frameworks, comparing it with Rails feral mechanism and also of the shortcomings in today's database, with respect to the ORM systems, which can help database designers to overcome some business problems.

Weak Points:
1). Validations are prone to data corruption because of the anomalies of weak isolation
2). Integrity is deteriorated, compared to other database systems.
3). Even though the usefulness of these feral mechanisms, there is limited support for implementing them in a Rails application.

Question:
1). Feral is understood as wild, does the paper use the feral in a way that the concurrency control is out of the control of the database system?