1 Integrity Constraints

Changes to a database may result in loss of **data consistency**. The **semantics** of a database are decided by database designers.

2 Specific Integrity Constraints

The Entity Relationship model already has some constructs for integrity contraints. **key declarations** state that some attributes depend on other attributes in a relation. More precisely,

> in relation schema R (a set of attributes), $K \subseteq R$ is a **superkey** of R if $K \rightarrow R$ (K **determines** R), meaning that whenever 2 tuples in a relation over R are equal on the key attributes, the tuples are equal on all attributes.

Relationship cardinalities impose constraints on the number of participants, e.g. (one-to-many).

3 General Integrity Constraints

A **functional dependency** is a a generalization of a key.

> In relation schema R, if A and B are **attribute sets** $A \rightarrow B$ (A determines B) is a functional dependency, meaning that whenever 2 tuples in a relation over R are equal on attributes A, the tuples are equal on attributes B.

The main difference from superkeys is that B $\neq$ R - B are not all the attributes of R.

A **domain constraint** defines a set of legal values for an attribute, more specificly than the basic types present in SQL. It can work like a typedef, e.g.:

```
CREATE DOMAIN Dollars NUMERIC(12,2)
```

Generic checks are also possible, e.g.:

```
CREATE DOMAIN Age INT CONSTRAINT AgeCheck CHECK(VALUE > 0)
```

Of course, any predicate can be checked, and you can effectively create a new domain with a **set membership** check, e.g.:

```
CREATE DOMAIN operation VARCHAR(10) CONSTRAINT operationValue
       CHECK(VALUE IN('consumer', 'producer'))
```

or

```
CREATE DOMAIN operation VARCHAR(10) CONSTRAINT operationValue
       CHECK(VALUE IN(SELECT name FROM operations))
```

4 Referential Integrity

referential integrity is the consistency of **foreign keys**, i.e. primary keys of other relations. Referential integrity ensures that there are no **dangling tuples**, tuples with foreign keys that references non-existant tuples in another relation. We must also ensure that the foreign keys of **weak entity sets** actually reference a primary key in the **owner** entity set.

The referential integrity constraint can be formulated as follows:

> If P is the primary key of r(R) (relation r on schema R) and $F \subseteq S$, then F is a foreign key if for each tuple in s, the attributes F are equal to a primary key in r.

Referential integrity enforcement is activated in SQL by using the REFERENCES clause in a CREATE TABLE command, e.g. CREATE TABLE (... FOREIGN KEY(attribute) REFERENCES table).

The DBS enforce referential integrity by doing the following checks:

When a new tuple is INSERTed into s, or s is UPDATEd, check for corresponding primary key in r when inserting tuples with foreign key.

When a tuple is DELETEd from s, or s is UPDATEd, check that there are no corresponding foreign keys in s, i.e. no dangling tuples.

If a violation occurs, the operation could be **canceled**, the user could be **informed**, **updates could be triggered** to correct the error. A **trigger** is a sideeffect to updates, and can be created with the following command:

```
CREATE TRIGGER name AFTER UPDATE ON table action
```

An **assertion** is a predicate that the database always must satisfy. In SQL, assertions are created with the command 'CREATE ASSERTION assertion-name CHECK predicate'. The assertion is initially checked for validity. Any future update to the database is only allowed if it does not violate the assertion. Assertions entails a large overhead, and are not implemented in all DBSs, e.g. mySQL.