Topic: 1.6.2 Data integrity

What is Data Integrity?

Data Integrity defines a quality of data, which guarantees the data is complete and has a whole structure. Data integrity is most often talked about with regard to data residing in databases, and referred to as database integrity as well. Data integrity is preserved only if and when the data is satisfying all the business rules and other important rules. These rules might be how each piece of data is related to each other, validity of dates, lineage, etc. According to data architecture principles, functions such as data transformation, data storage, meta-data storage and lineage storage must guarantee the integrity of data. That means, data integrity should be maintained during transfer, storage and retrieval.

If data integrity is preserved, the data can be considered consistent and can be given the assurance to be certified and reconciled. In terms of data integrity in databases (database integrity), in order to guarantee that integrity is preserved, you have to ensure that the data becomes an accurate reflection of the universe it is modeled after. In other words, it must make sure that the data stored in the database corresponds exactly to the real world details it is modeled after.

Validation and Verification

When data is input to a computer system it is only valuable data if it is correct. If the data is in error in any way then no amount of care in the programming will make up for the erroneous data and the results produced can be expected to be unreliable. There are three types of error that can occur with the data on entry. The first is that the data, while reasonable, is wrong. If your birthday is written down on a data capture form as 18th of November 1983, it will (except in very rare cases) be wrong. It can be typed into the computer with the utmost care as "181183", it can be checked by the computer to make sure that is a sensible date, and will then be accepted as your date of birth despite the fact that it is wrong. There is no reason for the computer to imagine that it may be wrong, quite simply when you filled out the original form you made a mistake. The second type of error is when the operator typing in the data hits the wrong key and types in "181193", or the equivalent. In this case an error has been made that should be able to be spotted if a suitable check is made on the input. **This type of data checking is called a verification check**. The third type of error is when something is typed in which simply is not sensible. If the computer knows that there are only 12 months in a year then it will know that "181383" must be wrong because it is not sensible to be born in the thirteenth month. **Checks on the sensibility of the data are called validation checks.**

Faulty data:

There is very little that can be done about faulty data except to let the owner of the data check it visually on a regular basis. The personal information kept on the school administration system about you and your family may well be printed off at regular intervals so that your parents can check to ensure that the stored information is still correct.

Topic: 1.6.2 Data integrity

Verification:

Verification means tallying the input data with the original data to make sure that there have been no transcription errors. The standard way to do this is to input the data twice to the computer system. The computer then checks the two sets of data (which should be the same) and if there is a difference between the two sets of data the computer knows that one of the inputs is wrong. It won't know which one is wrong but it can now ask the operator to check that particular input.

Validation:

The first thing is to clear out a common misinterpretation of validation. Specifically, the use of parity bits to check data. This is NOT validation. Parity bits and echoing back are techniques that are used to check that data has been transmitted properly within a computer system (e.g. from the disk drive to the processor), validation checks are used to check the input of data to the system in the first place.

Validation is a check on DATA INPUT to the system by comparing the data input with a set of rules that the computer has been told the data must follow. If the data does not match up with the rules then there must be an error. There are many different types of validation checks that can be used to check input in different applications:

1. Range check. A mathematics exam is out of 100. A simple validation rule that the computer can apply to any data that is input is that the mark must be between 0 and 100 inclusive. Consequently, a mark of 101 would be rejected by this check as being outside the acceptable range.

2. Character check. A person's name will consist of letters of the alphabet and sometimes a hyphen or apostrophe. This rule can be applied to input of a person's name so that "dav2d" will immediately be rejected as unacceptable.

3. Format check. A particular application is set up to accept a national insurance number. Each person has a unique national insurance number, but they all have the same format of characters, 2 letters followed by 6 digits followed by a single letter. If the computer knows this rule then it knows what the format of a NI number is and would reject "ABC12345Z" because it is in the wrong format, it breaks the rule.

4. Length check. A NI number has 9 characters, if more or fewer than 9 characters are keyed in then the data cannot be accurate.

5. Existence check. A bar code is read at a supermarket check-out till. The code is sent to the main computer which will search for that code on the stock file. As the stock file contains details of all items held in stock, if it is not there then the item cannot exist, which it obviously does, therefore the code must have been wrongly read.

6. Check digit. When the code is read on the item at the supermarket, it consists of numbers. One number is special; it is called the check digit. If the other numbers have some arithmetic done to them using a

simple algorithm the answer should be this special digit. When the code is read at the check-out till, if the arithmetic does not give the check digit it must have been read wrongly, it is at this point that the beeping sound would normally be heard if everything is alright.

Error checking and correcting

When you send data across the internet or even from your USB to a computer, you are sending millions upon millions of ones and zeros. What would happen if one of them got corrupted? Think of this situation: You are buying a new game from an online retailer and put £40 into the payment box. You click on send and the number 40 is sent to your bank stored in a byte: 00101000. Now imagine if the second most significant bit got corrupted on its way to the bank, and the bank received the following: 01101000. You'd be paying £104 for that game! Error Checking and Correction stops things like this happening. There are many ways to detect and correct corrupted data, we are going to learn two.

a) Parity: All data is transmitted as bits (0s and 1s). The Number of 1s in a byte must always be either an odd number or an even number. If two devices that are communicating decide that there will always be an odd number of 1s, then if a byte is received that has an even number of 1s, an error must have occurred. E.g. the byte 01011000 has 3 ones in it. 3 is an odd number, so it fits the rule that it must have an odd number of ones. When it is sent there is an error in transmission so that the first bit is received as a one. So, the byte received is 11011000. This has 4 ones in it, which is an even number, so there must be an error. The receiving device would ask for it to be sent again.

Notes:

- If two mistakes are made in the same byte they will cancel each other out and the faulty data will be accepted. This problem can be overcome, and in the same way, a clever way of correcting error mistakes can be implemented. This method is not part of this course.
- Earlier in this course it was said that a byte was the number of bits necessary to hold a character code. Specifically, an ASCII character uses 8 bits in a byte, giving 256 different characters. This is not true because one of the bits has to be reserved for a parity bit, the bit that can change to make sure that the number of ones is always odd. This means that there are 128 different characters possible.
- The implication in all the above work is that odd parity is always used. Even parity can equally well be used, whichever has been agreed between the two devices.
- Parity is not only used during data transfer between devices, but even when data is transferred between different parts of the CPU.

Topic: 1.6.2 Data integrity

Parity Blocks and Parallel Parity

It is an error-checking technique involving the comparison of a transmitted block check character with one calculated by the receiving device. Parallel parity is based on regular parity. Parallel parity can detect whether or not there was an error, furthermore it can detect which bit has flipped. This method is implemented on a block of data which is made of sum words; the parity bit is then added to the columns and rows.

Here's an example:

1	1	1	0	1	0	1	1
1	1	0	0	0	1	1	0
0	0	1	0	0	1	1	1
0	0	0	0	1	0	1	0

b) Check Sum: Data will normally be sent from one place to another as a block of bytes rather than as individual bytes. The computer can add numbers together without any trouble, so another checking procedure is to add all the bytes together that are being sent in the block of data. The carry, out of the byte, is not taken into account, so the answer is an 8 bit number, just like the bytes. This answer is calculated before the data is sent, and then calculated again when it is received, and if there are no errors in the transmission, the two answers will match.