ACID

Una **transacción**, en SQL, es un conjunto de instrucciones que tienen un propósito definido, y por tanto, se entienden como una unidad conjunta de trabajo sobre la base de datos.

Por ejemplo, en una base de datos de un centro comercial, la transacción *vender el artículo A al consumidor C*, es una unidad única de trabajo que, muy probablemente, estará formada al menos por las sentencias encargadas de eliminar A del conjunto de artículos disponibles para venta, y registrar que C ha comprado A (además de otras posibles).

A partir del propio concepto de transacción, es intuitivo que, en una base de datos sólida, se deberían cumplir una serie de objetivos o requisitos funcionales:

- **La transacción debería ejecutarse completamente, o, en caso de fallar, debería no producir modificaciones en la base de datos.** En ningún caso debería realizarse "a medias". En el ejemplo, jamás debería darse el caso de que se elimine A del conjunto de artículos disponibles para venta, pero no se registre su venta a C. Es deseable que la transacción tenga éxito, pero en caso de fallar, la opción "menos mala" es que no se modifique la base de datos en absoluto.

  Esta idea se llama **atomicidad**: las transacciones, o se ejecutan correctamente, o dejan intacta la base de datos, como si nunca se hubieran ejecutado.

- **La transacción debería producir el efecto deseado.** Por ejemplo, sería inadecuado que la transacción simplemente registrase que se ha vendido A a C, pero no eliminase A del conjunto de artículos a la venta. Esto depende del programador de las transacciones, ya que un sistema gestor de bases de datos no sabe cuál es la lógica de negocio, y por tanto, no puede manejar esta cuestión por sí mismo. Pese a todo como se indica en [1], sí nos puede ayudar con herramientas como las restricciones de integridad.

  A esto lo llamamos **consistencia**: la instancia de la base de datos después de ejecutar la transacción debe ser válida (coherente con la instancia previa a la transacción).

- **La transacción debería trabajar de forma aislada.** Es decir, a la transacción no debería afectarle que otras transacciones se estén ejecutando al mismo tiempo sobre la base de datos. Por ejemplo, no sería adecuado que dos transacciones vendieran el mismo artículo al mismo tiempo. Una de ellas debería ver que el artículo no está vendido, y la otra debería observar la instancia resultante de la primera transacción (y, por tanto, no poder realizar la venta).

  Este concepto es el de **aislamiento** (*isolation*): la transacción debe percibir que ninguna otra transacción está accediendo a la base de datos a la vez.

- **La transacción, una vez ejecutada, debería ser permanente.** Sería un fallo que se vendiese el artículo, pero la venta desapareciese de la base de datos posteriormente.

  En este caso, hablamos de **durabilidad**: los cambios deben permanecer.

En conjunto, estas cuatro propiedades se conocen con el acrónimo **ACID**. Una base de datos sólida debe asegurarse de que se cumplen las cuatro, y en ello juega un papel muy relevante la elección del sistema gestor de bases de datos, pero también el uso que hace el programador del mismo, especialmente en el caso de la consistencia.

[1] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database system concepts*. McGraw-Hill, New York, 6. ed., internat. ed. edition, 2011.