HOW TO CRACK INVESTMENT BANKING INTERVIEWS

FOUR PARTS TO COVER MAJORLY –

1. INVESTMENT BANKING
2. SQL
3. UNIX
4. SOFTWARE TESTING

INVESTMENT BANKING

**UNDER INVESTMENT BANKING FOLLWING NEEDS TO BE COVERED –**

1. **What is Investment Banking**
2. **Terminologies used in Investment Banking**
3. **Related to Your Project**
4. **What is Investment Banking –**

Investment banking is a specific division of [banking](http://www.investopedia.com/terms/b/bank.asp) related to the creation of [capital](http://www.investopedia.com/terms/c/capital.asp) for other companies, governments and other entities. [Investment banks](http://external.investopedia.com/terms/i/investmentbank.asp) underwrite new [debt](http://www.investopedia.com/terms/d/debtsecurity.asp) and equity [securities](http://www.investopedia.com/terms/s/security.asp) for all types of [corporations](http://external.investopedia.com/terms/c/corporation.asp), aid in the sale of [securities](http://www.investopedia.com/terms/s/security.asp), and help to facilitate [mergers and acquisitions](http://external.investopedia.com/terms/m/mergersandacquisitions.asp), [reorganizations](http://external.investopedia.com/terms/r/reorganization.asp) and broker trades for both institutions and private investors. Investment banks also provide guidance to issuers regarding the issue and placement of [stock](http://www.investopedia.com/terms/s/stock.asp).

Many large investment banks are affiliated with or subsidiaries of larger banking [institutions](http://www.investopedia.com/terms/f/financialinstitution.asp), and many have become household names, the largest being Goldman Sachs, Morgan Stanley, JPMorgan Chase, Bank of America Merrill Lynch and Deutsche Bank. Broadly speaking, investment banks assist in large, complicated financial transactions. This may include advice as to how much a company is worth and how best to structure a deal if the investment banker’s client is considering an [acquisition](http://www.investopedia.com/terms/a/acquisition.asp), [merger](http://www.investopedia.com/terms/m/merger.asp) or [sale](http://www.investopedia.com/terms/s/sale.asp). It may also include the issuing of securities as a means of raising money for the client groups, and creating the documentation for the [Securities and Exchange Commission](http://www.investopedia.com/terms/s/sec.asp) necessary for a company to [go public](http://www.investopedia.com/terms/g/goingpublic.asp).

Investment banks employ [investment bankers](http://www.investopedia.com/terms/i/investmentbanker.asp) who help [corporations](http://www.investopedia.com/terms/c/corporation.asp), governments and other groups plan and manage large projects, saving their client time and money by identifying [risks](http://www.investopedia.com/terms/r/risk.asp) associated with the project before the client moves forward. In theory, investment bankers are experts in their field who have their finger on the pulse of the current investing climate, so businesses and institutions turn to investment banks for advice on how best to plan their development, as investment bankers can tailor their recommendations to the present state of economic affairs.

Essentially, investment banks serve as middlemen between a company and investors when the company wants to issue stock or [bonds](http://www.investopedia.com/terms/b/bond.asp). The investment bank assists with pricing [financial instruments](http://www.investopedia.com/terms/f/financialinstrument.asp) so as to maximize [revenue](http://www.investopedia.com/terms/r/revenue.asp) and with navigating regulatory requirements. Often, when a company holds its [initial public offering (IPO)](http://www.investopedia.com/terms/i/ipo.asp), an investment bank will buy all or much of that company’s [shares](http://www.investopedia.com/terms/s/shares.asp) directly from the company. Subsequently, as a proxy for the company holding the IPO, the investment bank will sell the shares on the [market](http://www.investopedia.com/terms/m/market.asp). This makes things much easier for the company itself, as they effectively contract out the IPO to the investment bank. Moreover, the investment bank stands to make a profit, as it will generally price its shares at a markup from the price it initially paid. Yet, in doing so the investment bank also takes on a substantial amount of [risk](http://www.investopedia.com/terms/r/risk.asp). Though experienced analysts at the investment bank use their expertise to accurately price the stock as best they can, the investment bank can lose money on the deal if it turns out they have[overvalued](http://www.investopedia.com/terms/o/overvalued.asp) the stock, as in this case they will often have to sell the stock for less than they initially paid for it.

For example, suppose that Pete’s Paints Co., a chain supplying paints and other hardware, wants to go public. Pete, the owner, gets in touch with Jose, an investment banker working for a larger investment banking firm. Pete and Jose strike a deal wherein Jose (on behalf of his firm) agrees to buy 100,000 shares of Pete’s Paints for the company’s IPO at the price of $24 per share, a price at which the investment bank’s analysts arrived after careful consideration. The investment bank pays $2.4 million for the 100,000 shares and, after filing the appropriate paperwork, begins selling the stock for $26 per share. Yet, the investment bank is unable to sell more than 20% of the shares at this price and is forced to reduce the price to $23 per share in order to sell the remaining shares. For the IPO deal with Pete’s Paints, then, the investment bank has made $2.36 million [(20,000 x $26) + (80,000 x $23) = $520,000 + $1,840,000 = $2,360,000]. In other words, Jose’s firm has lost $40,000 on the deal because it overvalued Pete’s Paints.

1. **Terminologies Used for Investment Banking**
2. **Derivatives**
3. **Forwards**
4. **Futures**
5. **Options – PUT & CALL**
6. **Corporate Actions**
7. **Swaps**
8. **Money Markets**
9. **Trading Lifecycle**
10. **Swift Messages**
11. **ADR’S and GDR’S**
12. **Derivatives -**

A derivative is a contract between two or more parties whose value is based on an agreed-upon underlying financial asset, index or security. Common underlying instruments include: bonds, commodities, currencies, interest rates, market indexes and stocks.

Futures contracts, forward contracts, options, swaps and warrants are common derivatives. A futures contract, for example, is a derivative because its value is affected by the performance of the underlying contract. Similarly, a stock option is a derivative because its value is "derived" from that of the underlying stock.

Derivatives are used for speculating and hedging purposes. Speculators seek to profit from changing prices in the underlying asset, index or security. For example, a trader may attempt to profit from an anticipated drop in an index's price by selling (or going "short") the related futures contract. Derivatives used as a hedge allow the risks associated with the underlying asset's price to be transferred between the parties involved in the contract.

For example, commodity derivatives are used by farmers and millers to provide a degree of "insurance." The farmer enters the contract to lock in an acceptable price for the commodity; the miller enters the contract to lock in a guaranteed supply of the commodity. Although both the farmer and the miller have reduced risk by hedging, both remain exposed to the risks that prices will change. For example, while the farmer locks in a specified price for the commodity, prices could rise (due to, for instance, reduced supply because of weather-related events) and the farmer will end up losing any additional income that could have been earned. Likewise, prices for the commodity could drop and the miller will have to pay more for the commodity than he otherwise would have.

Some derivatives are traded on national securities exchanges and are regulated by the U.S. Securities and Exchange Commission (SEC). Other derivatives are traded over-the-counter (OTC); these derivatives represent individually negotiated agreement between parties.

**2. Forwards –**

What is a 'Forward Contract'

A forward contract is a customized contract between two parties to buy or sell an asset at a specified price on a future date. A forward contract can be used for hedging or speculation, although its non-standardized nature makes it particularly apt for hedging. Unlike standard [futures](http://www.investopedia.com/terms/f/futures.asp) contracts, a forward contract can be customized to any commodity, amount and [delivery date](http://www.investopedia.com/terms/d/deliverydate.asp). A forward contract settlement can occur on a cash or delivery basis. Forward contracts do not trade on a centralized exchange and are therefore regarded as over-the-counter (OTC) instruments. While their [OTC](http://www.investopedia.com/terms/o/otc.asp) nature makes it easier to customize terms, the lack of a centralized clearinghouse also gives rise to a higher degree of [default risk](http://www.investopedia.com/terms/d/defaultrisk.asp). As a result, forward contracts are not as easily available to the [retail investor](http://www.investopedia.com/terms/r/retailinvestor.asp) as [futures contracts](http://www.investopedia.com/terms/f/futurescontract.asp).

1. **FUTURES**

What are 'Futures'

Futures are financial contracts obligating the buyer to purchase an [asset](http://www.investopedia.com/terms/a/asset.asp) (or the [seller](http://www.investopedia.com/terms/s/seller.asp) to sell an asset), such as a physical commodity or a [financial instrument](http://www.investopedia.com/terms/f/financialinstrument.asp), at a predetermined future date and price. Futures contracts detail the quality and quantity of the underlying asset; they are standardized to facilitate trading on a [futures exchange](http://www.investopedia.com/terms/f/futuresexchange.asp). Some futures contracts may call for [physical delivery](http://www.investopedia.com/terms/p/physicaldelivery.asp) of the asset, while others are settled in cash. The [futures markets](http://www.investopedia.com/terms/f/futuresmarket.asp) are characterized by the ability to use very high [leverage](http://www.investopedia.com/terms/l/leverage.asp) relative to [stock markets](http://www.investopedia.com/terms/s/stockmarket.asp).   
  
Futures can be used either to [hedge](http://www.investopedia.com/terms/h/hedge.asp) or to speculate on the price movement of the underlying asset. For example, a producer of corn could use futures to lock in a certain price and reduce risk (hedge). On the other hand, anybody could speculate on the price movement of corn by going long or short using futures.

1. **OPTIONS**

What is an 'Option'

An option is a financial derivative that represents a contract sold by one party (option writer) to another party (option holder). The contract offers the buyer the right, but not the obligation, to buy (call) or sell (put) a security or other [financial asset](http://www.investopedia.com/terms/f/financialasset.asp) at an agreed-upon price ([the strike price](http://www.investopedia.com/video/play/strike-price/)) during a certain period of time or on a specific date (exercise date).

Call options give the option to buy at certain price, so the buyer would want the stock to go up.

Put options give the option to sell at a certain price, so the buyer would want the stock to go down.

1. **CORPORATE ACTIONS**

When a publicly-traded company issues a [corporate action](http://www.investopedia.com/terms/c/corporateaction.asp), it is initiating a process that will bring actual change to its stock. By understanding these different types of processes and their effects, an investor can have a clearer picture of what a corporate action indicates about a company's financial affairs and how that action will influence the company's share price and performance. This knowledge, in turn, will aid the investor in determining whether to buy or sell the stock in question.

Corporate actions are typically agreed upon by a company's [board of directors](http://www.investopedia.com/terms/b/boardofdirectors.asp) and authorized by the shareholders. Some examples are [stock splits](http://www.investopedia.com/terms/s/stocksplit.asp), [dividends](http://www.investopedia.com/terms/d/dividend.asp), [mergers](http://www.investopedia.com/terms/m/merger.asp) and [acquisitions](http://www.investopedia.com/terms/a/acquisition.asp), [rights issues](http://www.investopedia.com/terms/r/rightsoffering.asp) and [spin offs](http://www.investopedia.com/terms/s/spinoff.asp). Let's take a closer look at these different examples of corporate actions.

1. **SWAPS**

What is a 'Swap'

A swap is a [derivative](http://www.investopedia.com/terms/d/derivative.asp) contract through which two parties exchange financial instruments. These instruments can be almost anything, but most swaps involve [cash flows](http://www.investopedia.com/terms/c/cashflow.asp) based on a [notional principal amount](http://www.investopedia.com/terms/n/notionalprincipalamount.asp) that both parties agree to. Usually, the [principal](http://www.investopedia.com/terms/p/principal.asp) does not change hands. Each cash flow comprises one [leg](http://www.investopedia.com/terms/l/leg.asp) of the swap. One cash flow is generally fixed, while the other is variable, that is, based on a a benchmark interest rate, [floating currency exchange rate](http://www.investopedia.com/terms/f/floatingexchangerate.asp) or index price.

The most common kind of swap is an [interest rate swap](http://www.investopedia.com/terms/i/interestrateswap.asp). Swaps do not trade on [exchanges](http://www.investopedia.com/terms/e/exchange.asp), and [retail investors](http://www.investopedia.com/terms/r/retailinvestor.asp) do not generally engage in swaps. Rather, swaps are [over-the-counter](http://www.investopedia.com/terms/o/otc.asp) contracts between businesses or financial institutions.

## What is a 'Money Market'

A money market is a segment of the financial market in which [financial instruments](http://www.investopedia.com/terms/f/financialinstrument.asp) with high liquidity and very short [maturities](http://www.investopedia.com/terms/m/maturity.asp) are traded. The money market is used by participants as a means for borrowing and lending in the short term, from several days to just under a year. Money market securities consist of [negotiable](http://www.investopedia.com/terms/n/negotiable.asp) [certificates of deposit (CDs),](http://www.investopedia.com/terms/c/certificateofdeposit.asp)bankers [acceptances](http://www.investopedia.com/terms/a/acceptance.asp), U.S. [Treasury bills](http://www.investopedia.com/terms/t/treasurybill.asp), commercial paper, [municipal notes](http://www.investopedia.com/terms/m/municipal-note.asp), [federal funds](http://www.investopedia.com/terms/f/federalfunds.asp) and [repurchase agreements (repos).](http://www.investopedia.com/terms/r/repurchaseagreement.asp)

## What is a 'Mutual Fund'

A mutual fund is an [investment vehicle](http://www.investopedia.com/terms/i/investmentvehicle.asp) that is made up of a pool of funds collected from many investors for the purpose of [investing](http://www.investopedia.com/terms/i/investing.asp) in securities such as stocks, bonds, [money market](http://www.investopedia.com/terms/m/moneymarket.asp) instruments and similar assets. Mutual funds are operated by [money managers](http://www.investopedia.com/terms/m/moneymanager.asp), who invest the fund's capital and attempt to produce [capital gains](http://www.investopedia.com/terms/c/capitalgain.asp) and income for the fund's investors. A [mutual fund's portfolio](http://www.investopedia.com/video/play/introduction-mutual-funds/) is structured and maintained to match the [investment objectives](http://www.investopedia.com/terms/i/investmentobjective.asp) stated in its [prospectus](http://www.investopedia.com/terms/p/prospectus.asp).

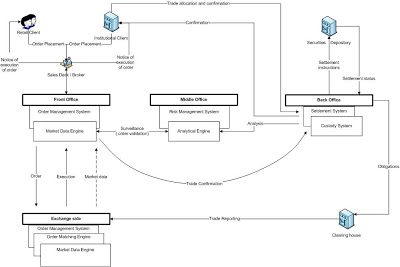
One of the main advantages of mutual funds is that they give small investors access to professionally managed, diversified portfolios of equities, bonds and other securities, which would be quite difficult (if not impossible) to create with a small amount of capital. Each shareholder participates proportionally in the gain or loss of the fund. Mutual fund units, or shares, are issued and can typically be purchased or redeemed as needed at the fund's current [net asset value (NAV)](http://www.investopedia.com/terms/n/nav.asp) per share, which is sometimes expressed as [NAVPS](http://www.investopedia.com/terms/n/navpershare.asp).

1. **TRADING LIFECYCLE –**

### Trade Life Cycle/Securities trade life cycle

Have you wondered what happens when you initiate a trade, in simple terms when you put an order to buy or sell a stock/shares in the stock market through your trading terminal.  
  
  
To understand trade life cycle we need to understand detailed steps involved in trade life cycle.  
Below mentioned are the important steps:

1. Order initiation and delivery. (Front office function)  
2. Risk management and order routing.(middle office function)  
3. Order matching and conversion into trade.(front office function)  
4. Affirmation and confirmation.(back office function)  
5. Clearing and Settlement.(back office function)



Them main objective of every trade is to get executed at the best price and settled at the least risk and less cost. Some may say trade life cycle is divided into 2 parts pre-trade activities and post trade activities, well, pre-trade activities consists of all those steps that take place before order gets executed, post trade activities are all those steps that involve order matching, order conversion to trade and entire clearing and settlement activity.

Now lets discuss every step of life cycle in detail.

Order initiation and delivery. (Front office function)

Who initiates the orders: Retail client like me and you, institutional clients like any Mutual fund company's.

Clients keep a close watch on the stock market and build a perception on the movement of market. They also try to find investment opportunities so that they can build position in the market. Positions are the result of trade that are executed in the market.Clients place orders with brokers through telephone, fax , online trading and hand held devices. Orders can be placed either market orders or limit orders, market orders means order to buy or sell is placed at the market price of the share/equity/stock that the investor/client wants to buy/sell whereas limit orders means order placed to buy or sell at a price that investor/client wants to buy/sell.

When Broker receives these orders, he/she records these orders carefully so that there is no ambiguity/mistakes in processing. Institutional investor/ fund manager at this stage would not have decided on the allocation of the funds so he/she will just contact sales desk and place the order so that later they can allocate their investments to the mutual funds(irrespective whether it is buy/sell).

Risk management and order routing.(middle office function)  
  
  
As we know that getting trades settled lies with the broker, if any client makes any trade default, then the same has to be made good by the broker to the clearing corporation by broker.  
  
  
When orders are accepted and sent to exchange these orders go through various risk management checks institutions and retails clients. Although the risk management checks are more for retail investors , the underlying assumptions is that they are less creditworthy also due to online trading the client has become faceless so the risk has increased more.Its not same for institutional investors because they have a large balance sheet compared to the size of orders they want to place. They also maintain collateral with the members they push their trades through. Their trades are hence subjected to fewer risk management checks than retail clients.

Below are the steps how risk management is done in case of retail transactions:

* Client logs into the trading portal provide credential and places orders.
* The broker validates that the order is coming from a reliable source.
* Lets say the client places buy order, in this case the broker places query to verify whether the client has sufficient balance(margin money) and passes the order to the exchange, in case the client does not have sufficient margin then order is rejected. If client has the margin money then the order is accepted and margin money is reduced from the available margin so that client is aware of the real time margin available to him for trade, also to make sure that he/she does not place order more than margin money available so later on the broker need not make good on behalf of client to the exchange.
* Lets say the client places sell order, in this case the broker places query to verify whether the client has sufficient stocks to place the order in case of there are no sufficient stocks then the order stays rejected, if there are sufficient stocks available then the order is accepted and stocks are blocked for sale and remaining stocks are shown as balance available for sell.
* Once the above risk management check passes then the order is passed to the exchange.
* On receipt of the order, the exchange immediately sends an order confirmation to the broker's trading system.
* Depending upon the order terms and the actual prices prevailing in the market, the order could get executed immediately or remain pending in the order book of the exchange.

A margin is an amount that clearing corporations levy on the brokers for maintaining positions on the exchange. The amount of margin levied is proportional to the exposure and risk the broker is carrying. Since positions may belong to a broker’s clients, it is the broker’s responsibility to recover margins from clients. To protect the market from defaulters, clearing corporations levy margins on the date of the trade.

Order matching and conversion into trade.(front office function)  
  
  
Below are the steps:

* All the orders are collated and sent to the exchange for execution, exchange tries to allot the shares in the best price available to the investors.
* Broker has the record of all the orders that were received from whom , at what time, against which stock, type of order and quantity. Broker maintains these records against client ID.
* Brokers are in real time conversation with exchange so that they have details of how many orders are still pending and and how many have been executed in the exchange.
* Once the order is executed it turns into trade and exchange sends sends notification of the trade to the broker. The broker in turn communicates these trades to the client either immediately or end of day.
* Official communication from broker is done to the client through contract note, which contains details of the trade executed along with taxes being charged and commission being charged by the broker and other institutions like clearing corporation, custodian etc...

Affirmation and confirmation.(back office function)

Every institution engages the services of an agency called a custodian to assist them in clearing and settlement activities. Institutions specialize in taking positions and holding. To outsource the activity of getting their trades settled and to protect themselves and their shareholder’s interests, they hire a local custodian in the country where they trade. When they trade in multiple countries, they also have a global custodian who ensures that settlements are taking place seamlessly in local markets using local custodians.

As discussed earlier, while giving the orders for the purchase/sale of a particular security, the fund manager may just be in a hurry to build a position. He may be managing multiple funds or portfolios. At the time of giving the orders, the fund managers may not really have a fund in mind in which to allocate the shares. So to make more profit and avoid unfavourable market conditions he/she places the order.  
  
The broker accepts this order for execution. On successful execution, the broker sends the trade confirmations to the institution. The fund manager at the institution during the day makes up his mind about how many shares have to be allocated to which fund and by evening sends these details to broker. Brokers does a cross verification whether all the alocation details match the trade details and  then prepares the contract notes in the names of the funds in which the fund manager has requested allocation.  
  
Along with the broker, the institution also has to send details to custodian for the orders it has given to the broker. The institution provides allocation details to the custodian as well. It also provides the name of the securities, the price range, and the quantity of shares ordered. This prepares the custodian, who is updated about the information expected to be received from the broker. The custodian also knows the commission structure the broker is expected to charge the institution and the other fees and statutory levies.

On receipt of the trade details, the custodian sends an affirmation to the broker indicating that the trades have been received and are being reviewed.Trades are validated to check the following:

* Whether trade has happened on the desired security.
* Whether the trade is correct i.e buy or sell.
* The price at which rate it was executed.
* Whether the charges are as per the agreement.

For this verification process the custodian normally runs a software such as TLM for recon process.In case the trade details do not match, the custodian rejects the trade, and the trades shift to the broker’s books. It is then the broker’s decision whether to keep the trade (and face the associated price risk) or square it at the prevailing market prices.

Clearing and Settlement.(back office function)  
  
  
As we know that there are hundreds and thousands of trades being executed everyday and all these trades needs to be cleared and settled. Normally all these trades gets settled in T+2 days, which means the trade will gets allotted to the investor to his/her demat account in 2 days from trade date.   
  
  
The clearing corporation has obligation to every investor in form of

* Funds (for all buy transactions and also to those transactions that are not squared for the sale positions).
* Securities(for all the sale transactions done)

Clearing corporation calculates and informs the members of what their obligations are on the funds side (cash) and on the securities side. These obligations are net obligations with respect to the clearing corporation. Lets say broker purchased 1000 shares of reliance for client A  and sold 600 shares of reliance for client B which means the obligation of the clearing corporation to the broker is only for 400 shares.

Clearing members are expected to open clearing accounts with certain banks specified by the clearing corporation as clearing banks. They are also expected to open clearing accounts with the depository. They are expected to keep a ready balance for their fund obligations in the bank account and similarly maintain stock balances in their clearing demat account.

Once the clearing corporation informs all members of their obligations, it is the responsibility of the clearing members to ensure that they make available their obligations (shares and money) in the clearing corporation’s account. Once these obligations are done the balance of payments takes place and all the investors will have their stocks/financial instruments/shares in their demat account if a buy trade is executed and cash will be credited to their demat accounts if sale trade is executed. On every end of day basis the clearing corporation generates various reports that need to be circulated to exchanges and custodians.

1. SWIFT MESSAGES –

What is SWIFT?

SWIFT stands for the Society for Worldwide Interbank Financial Telecommunications. It is a messaging network that financial institutions use to securely transmit information and instructions through a standardized system of codes.

SWIFT assigns each financial organization a unique code that has either eight characters or 11 characters. The code is called interchangeably the bank identifier code (BIC), SWIFT code, SWIFT ID, or ISO 9362 code. (See related: [What's the difference between an IBAN and a swift code](http://www.investopedia.com/ask/answers/100214/whats-difference-between-iban-and-swift-code.asp)?) To understand how the code is assigned, let’s look at Italian bank UniCredit Banca, headquartered in Milan. It has the 8-character SWIFT code UNCRITMM.

* First four characters: the institute code (UNCR for UniCredit Banca)
* Next two characters: the country code (IT for the country Italy)
* Next two characters: the location/city code (MM for Milan)
* Last three characters: optional, but organizations use it to assign codes to individual branches. (The UniCredit Banca branch in Venice may use the code UNCRITMMZZZ.)

Assume a customer of a [Bank of America](http://www.investopedia.com/markets/stocks/bac/) branch in New York wants to send money to his friend who banks at the UniCredit Banca branch in Venice. The New Yorker can walk into his Bank of America branch with his friend’s account number and UnicaCredit Banca’s unique SWIFT code for its Venice branch. Bank of America will send a payment transfer SWIFT message to the UniCredit Banca branch over the secure SWIFT network. Once Unicredit Banca receives the SWIFT message about the incoming payment, it will clear and credit the money to the Italian friend’s account.

As powerful as SWIFT is, keep in mind that it is only a messaging system—SWIFT does not hold any funds or securities, nor does it manage client accounts.

1. ADR’S AND GDR’S -

Investing in foreign stocks should be part of any investor's portfolio. Not only does it diversify your holdings, it offers plenty of opportunities to [profit from trends and developments outside your home country](http://www.investopedia.com/articles/mutualfund/05/intlfunds.asp).

The U.S. currently accounts for nearly half of the world's total stock market value, but that's likely to decline in the years ahead as more investors look to emerging markets such as China and India.

Below is an overview of the easiest ways to invest in foreign stocks, whether you live in the U.S. or any other country.

DRs (Depositary Receipts)

A depositary receipt is a negotiable financial instrument issued by a bank to represent a foreign company's publicly traded securities. The depositary receipt trades [on a local stock exchange](http://www.investopedia.com/articles/basics/04/092404.asp), such as the New York Stock Exchange (NYSE) in the U.S., but represents an interest in a company that is headquartered outside of the United States. A depositary receipt traded in Germany would represent a non-German company.

A DR can be sponsored or unsponsored. The sponsored variety is issued with the knowledge and cooperation of the underlying foreign company. With this cooperation, a sponsored DR usually lets the owners have the same rights normally given to the stockholders in the home country, such as [voting rights](http://www.investopedia.com/terms/v/votingright.asp). An unsponsored DR may not have the same voting rights.

ADRs (American Depositary Receipt)

An American Depositary Receipt (ADR) is a negotiable certificate issued by a U.S. bank representing a specified number of shares in a foreign (i.e. non-U.S.) stock that is traded on a U.S. exchange. ADRs are denominated in U.S. dollars, with the underlying security held by a U.S. financial institution overseas. ADRs help to reduce administration and duty costs that would otherwise be levied on each transaction.

It’s important to note that an ADR must be sponsored by the underlying corporation. If not, the security is likely to be traded [over the counter](http://www.investopedia.com/terms/o/otc.asp), which is considered more risky because there are fewer regulatory requirements.

In most cases, a foreign firm trading OTC will have a five letter ticker. For instance, European Aeronautic Defence and Space Company N.V. trades under the U.S. ticker “EADSF.” In contrast, an ADR may trade with the common three [ticker symbol on the NYSE](http://www.investopedia.com/articles/basics/09/translating-ticker-talk.asp). Swiss-based ABB Ltd. trades with the symbol “ABB” on the NYSE.

Overall, there are four levels of ADRs. The chart below explains the difference between sponsored and unsponsored ADRs, as well as different levels of sponsored ADRs and U.S. Securities and Exchange Commission (SEC) reporting requirements.

You can see that the unsponsored variety trades over the counter, which leaves more work to the investor to determine if the voting rights are the same. These shares may also have much less liquidity. To determine which level a company trades at, an investor should check the SEC site (sec.gov) to see which forms have been filed. Checking on the website of a depository bank or ADR.com can also provide insight into a company's filing status.

GDRs (Global Depositary Receipt)

Beyond the ADR, there is a second category of DR. A Global Depositary Receipt (GDR) represents a bank certificate issued in more than one country for shares in a foreign company. The shares are held by a foreign branch of an international bank. The shares trade as domestic shares, but are offered for sale globally through the various bank branches. The term GDR is used throughout the globe and designates any foreign firm that trades on an exchange outside its home country.

1. SQL –
2. Joins –

## SQL INNER JOIN Keyword

The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.

### SQL INNER JOIN Syntax

SELECT *column\_name(s)*  
FROM *table1*  
INNER JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

or:

SELECT *column\_name(s)*  
FROM *table1*  
JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

PS! INNER JOIN is the same as JOIN.



In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CustomerID | CustomerName | ContactName | Address | City | PostalCode | Country |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |

And a selection from the "Orders" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrderID | CustomerID | EmployeeID | OrderDate | ShipperID |
| 10308 | 2 | 7 | 1996-09-18 | 3 |
| 10309 | 37 | 3 | 1996-09-19 | 1 |
| 10310 | 77 | 8 | 1996-09-20 | 2 |

## SQL INNER JOIN Example

The following SQL statement will return all customers with orders:

### Example

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
INNER JOIN Orders  
ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName

## QL LEFT JOIN Keyword

The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

### SQL LEFT JOIN Syntax

SELECT *column\_name(s)*  
FROM *table1*  
LEFT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

or:

SELECT *column\_name(s)*  
FROM *table1*  
LEFT OUTER JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

PS! In some databases LEFT JOIN is called LEFT OUTER JOIN.



## Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CustomerID | CustomerName | ContactName | Address | City | PostalCode | Country |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |

And a selection from the "Orders" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrderID | CustomerID | EmployeeID | OrderDate | ShipperID |
| 10308 | 2 | 7 | 1996-09-18 | 3 |
| 10309 | 37 | 3 | 1996-09-19 | 1 |
| 10310 | 77 | 8 | 1996-09-20 | 2 |

## SQL LEFT JOIN Example

The following SQL statement will return all customers, and any orders they might have:

### Example

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders  
ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;

**Note:** The LEFT JOIN keyword returns all the rows from the left table (Customers), even if there are no matches in the right table (Orders).

## SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

### SQL RIGHT JOIN Syntax

SELECT *column\_name(s)*  
FROM *table1*  
RIGHT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

or:

SELECT *column\_name(s)*  
FROM *table1*  
RIGHT OUTER JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

PS! In some databases RIGHT JOIN is called RIGHT OUTER JOIN.



## Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Orders" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrderID | CustomerID | EmployeeID | OrderDate | ShipperID |
| 10308 | 2 | 7 | 1996-09-18 | 3 |
| 10309 | 37 | 3 | 1996-09-19 | 1 |
| 10310 | 77 | 8 | 1996-09-20 | 2 |

And a selection from the "Employees" table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EmployeeID | LastName | FirstName | BirthDate | Photo | Notes |
| 1 | Davolio | Nancy | 12/8/1968 | EmpID1.pic | Education includes a BA in psychology..... |
| 2 | Fuller | Andrew | 2/19/1952 | EmpID2.pic | Andrew received his BTS commercial and.... |
| 3 | Leverling | Janet | 8/30/1963 | EmpID3.pic | Janet has a BS degree in chemistry.... |

## SQL RIGHT JOIN Example

The following SQL statement will return all employees, and any orders they have placed:

### Example

SELECT Orders.OrderID, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees  
ON Orders.EmployeeID=Employees.EmployeeID  
ORDER BY Orders.OrderID;

**Note:** The RIGHT JOIN keyword returns all the rows from the right table (Employees), even if there are no matches in the left table (Orders).

## SQL FULL OUTER JOIN Keyword

The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).

The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.

### SQL FULL OUTER JOIN Syntax

SELECT *column\_name(s)*  
FROM *table1*  
FULL OUTER JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;



## Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CustomerID | CustomerName | ContactName | Address | City | PostalCode | Country |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |

And a selection from the "Orders" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OrderID | CustomerID | EmployeeID | OrderDate | ShipperID |
| 10308 | 2 | 7 | 1996-09-18 | 3 |
| 10309 | 37 | 3 | 1996-09-19 | 1 |
| 10310 | 77 | 8 | 1996-09-20 | 2 |

## SQL FULL OUTER JOIN Example

The following SQL statement selects all customers, and all orders:

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders  
ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;

A selection from the result set may look like this:

|  |  |
| --- | --- |
| CustomerName | OrderID |
| Alfreds Futterkiste |  |
| Ana Trujillo Emparedados y helados | 10308 |
| Antonio Moreno Taquería | 10365 |
|  | 10382 |
|  | 10351 |

**Note:** The FULL OUTER JOIN keyword returns all the rows from the left table (Customers), and all the rows from the right table (Orders). If there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

# Write a SQL query that will return the maximum value from the “Numbers” column, without using a SQL aggregate like MAX or MIN

SELECT Numbers

FROM Compare

WHERE Numbers NOT IN (

SELECT Smaller.Numbers

FROM Compare AS Larger

JOIN Compare AS Smaller ON Smaller.Numbers < Larger.Numbers

)

select TOP 1 -- select the very top entry in result set

Numbers

from

Compare

order by

Numbers DESC

# In SQL, what’s the difference between the having clause and the group by statement?

In SQL, the having clause and the group by statement work together when using aggregate functions like SUM, AVG, MAX, etc. This is best illustrated by an example. Suppose we have a table called emp\_bonus as shown below. Note that the table hasmultiple entries for employees A and B – which means that both employees A and B have received multiple bonuses.

|  |
| --- |
| emp\_bonus |
| |  |  | | --- | --- | | Employee | Bonus | | A | 1000 | | B | 2000 | | A | 500 | | C | 700 | | B | 1250 | |

If we want to calculate the total bonus amount that each employee has received, then we would write a SQL statement like this:

|  |
| --- |
| select employee, sum(bonus) from emp\_bonus group by employee; |

## he Group By Clause

In the SQL statement above, you can see that we use the "group by" clause with the employee column. The group by clause allows us to find the sum of the bonuses for eachemployee – because each employee is treated as his or her very own group. Using the ‘group by’ in combination with the ‘sum(bonus)’ statement will give us the sum of all the bonuses for employees A, B, and C.

[Subscribe to our newsletter for more free interview questions.](http://www.programmerinterview.com/index.php/popup/)

Running the SQL above would return this:

|  |  |
| --- | --- |
| **Employee** | **Sum(Bonus)** |
| A | 1500 |
| B | 3250 |
| C | 700 |

Now, suppose we wanted to find the employees who received more than $1,000 in bonuses for the year of 2012 – this is assuming of course that the emp\_bonus table contains bonuses only for the year of 2012. This is when we need to use the HAVINGclause to add the additional check to see if the sum of bonuses is greater than $1,000, and this is what the SQL look like:

|  |
| --- |
| GOOD SQL:  select employee, sum(bonus) from emp\_bonus  group by employee having sum(bonus) > 1000; |

And the result of running the SQL above would be this:

|  |  |
| --- | --- |
| **Employee** | **Sum(Bonus)** |
| A | 1500 |
| B | 3250 |

## Difference between having clause and group by statement

So, from the example above, we can see that the group by clause is used to group column(s) so that aggregates (like SUM, MAX, etc) can be used to find the necessary information. The having clause is used with the group by clause when comparisons need to be made with those aggregate functions – like to see if the SUM is greater than 1,000, as in our example above. So, the having clause and group by statements are not really alternatives to each other – but they are used alongside one another!

# What is a self join? Explain it with an example

Let’s illustrate the need for a self join with an example. Suppose we have the following table – that is called employee. The employee table has 2 columns – one for the employee name (called employee\_name), and one for the employee location (called employee\_location):

|  |
| --- |
| employee |
| |  |  | | --- | --- | | employee\_name | employee\_location | | Joe | New York | | Sunil | India | | Alex | Russia | | Albert | Canada | | Jack | New York | |

Now, suppose we want to find out which employees are from the same location as the employee named Joe. In this example, that location would be New York. Let’s assume – for the sake of our example – that we can not just directly search the table for people who live in New York with a simple query like this (maybe because we don’t want to hardcode the city name) in the SQL query:

SELECT employee\_name

FROM employee

WHERE employee\_location = "New York"

So, instead of a query like that what we could do is write a nested SQL query (basically a query within another query – which is more commonly called a subquery) like this:

|  |
| --- |
| SELECT employee\_name  FROM employee  WHERE employee\_location in  ( SELECT employee\_location  FROM employee  WHERE employee\_name = "Joe") |

A subquery is inefficient

Using a subquery for such a simple question is inefficient. Is there a more efficient and elegant solution to this problem?

t turns out that there is a more efficient solution – we can use something called a self join. A self join is basically when a table is joined to itself. The way you should visualize a self join for a given table is by imagining that a join is performed between two identical copies of that table. And that is exactly why it is called a self join – because of the fact that it’s just the same table being joined to anothercopy of itself rather than being joined with a different table.

## How does a self join work

Before we come up with a solution for this problem using a self join, we should go over some concepts so that you can fully understand how a self join works. This will also make the SQL in our self join tutorial a lot easier to understand, which you will see further below.

## A self join must have aliases

In a self join we are joining the same table to itself by essentially creating two copies of that table. But, how do we distinguish between the two different copies of the table – because there is only one table name after all? Well, when we do a self join, the table names absolutely must use aliases otherwise the column names would be ambiguous. In other words, we would not know which of the two copies of the table’s columns is being referenced without using an alias for each copy of the table. If you don’t already know what an alias is, it’s simply another name given to a table – think of an alias as a nickname – and that nickname is then used in the SQL query to reference the table. Because we need two copies of the employee table, we will just use the aliases e1 and e2 for the employee table when we do a self join.

## Self join predicate

As with any join there must be a condition upon which a self join is performed – we can not just arbitrarily say “do a self join”, without specifying some condition. That condition will be our join predicate. If you need a refresher on join predicates (or just joins in general) then check this link out: [Inner vs. Outer joins](http://www.programmerinterview.com/sql/inner-vs-outer-joins.php).

Now, let’s come up with a solution to the original problem using a self join instead of a subquery. This will help illustrate how exactly a self join works. The key question that we must ask ourselves is what should our join predicate be in this example? Well, we want to find all the employees who have the same location as Joe.

Because we want to match between our two tables (both of which are the same table – employee – aliased as e1 and e2) on location our join predicate should clearly be “WHERE e1.employee\_location = e2.employee\_location”. But is that enough to give us what we want? No, it’s not, because we also want to filter the rows returned since we only want people who are from the same location as Joe.

So, how can we filter the rows returned so that only people from Joe’s location are returned? Well, what we can do is simply add a condition on one of the tables (e2 in our example) so that it only returns the row where the name is Joe. Then, the other table (e1) will match up all the names that have the same location in e2, because of our join predicate – which is “WHERE e1.employee\_location = e2.employee\_location”. We will then just select the names from e1, and not e2 because e2 will only have Joe’s name. If that’s confusing then keep reading further to understand more about how the query will work.

So, the self join query that we come up with looks like this:

## Self Join SQL Example

|  |
| --- |
| SELECT e1.employee\_name  FROM employee e1, employee e2  WHERE e1.employee\_location = e2.employee\_location  AND e2.employee\_name="Joe"; |

This query will return the names Joe and Jack – since Jack is the only other person who lives in New York like Joe.

Generally, queries that refer to the same table can be greatly simplified by re-writing the queries as self joins. And, there is definitely a performance benefit for this as well.

## What does a self join look like?

It will help tremendously to actually visualize the actual results of a self join internally. Remember that a self join is just like any other join, where the two tables are merged into one temporary table. First off, you should visualize that we have two separate copies of the employee table, which are given aliases of e1 and e2. These copies would simply look like this – note that we shortened the column names from employee\_name and employee\_location to just Name and Location for convenience:

|  |  |
| --- | --- |
| e1 | e2 |
| |  |  | | --- | --- | | Name | Location | | Joe | New York | | Sunil | India | | Alex | Russia | | Albert | Canada | | Jack | New York | | |  |  | | --- | --- | | Name | Location | | Joe | New York | | Sunil | India | | Alex | Russia | | Albert | Canada | | Jack | New York | |

And the final results of running the self join query above – the actual joined table – would look like this:

|  |  |  |  |
| --- | --- | --- | --- |
| e1.employee\_name | e1.employee\_location | e2.employee\_name | e2.employee\_location |
| Joe | New York | Joe | New York |
| Jack | New York | Joe | New York |

Self joins manager employee example

The most commonly used example for self joins is the classic employee manager table. The table is called Employee, but holds all employees – including their managers. Every employee has an ID, and there is also a column for the manager ID. So, for example, let’s say we have a table that looks like this – and we call it Employee:

|  |  |  |
| --- | --- | --- |
| EmployeeID | Name | ManagerID |
| 1 | Sam | 10 |
| 2 | Harry | 4 |
| 4 | Manager | NULL |
| 10 | AnotherManager | NULL |

Notice that in the table above there are two managers, conveniently named “Manager” and “AnotherManager”. And, those managers don’t have managers of their own – as noted by the NULL value in their Manager column.

Now, given the table above, how can we return results that will show each employee’s name, and his/her manager’s name in nicely arranged results – with the employee in one column and his/her manager’s name in the other column. Well, it turns out we can use a self join to do this. Try to come up with the SQL on your own before reading our answer.

Self join manager employee answer

In order to come up with a correct answer for this problem, our goal should be to perform a self join that will have both the employee information and manager information in one row. First off, since we are doing a self join, it helps to visualize the one table as two tables – let’s give them aliases of e1 and e2. Now, with that in mind, we want the employee’s information on one side of the joined table and the manager’s information on the other side of the joined table. So, let’s just say that we want e1 to hold the employee information and e2 to hold the corresponding manager’s information. What should our join predicate be in that case?

Well, the join predicate should look like “ON e1.ManagerID = e2.EmployeeID” – this basically says that we should join the two tables (a self join) based on the condition that the manager ID in e1 is equal to the employee ID in e2. In other words, an employee’s manager in e1 should have the manager’s information in e2. An illustration will help clarify this. Suppose we use that predicate and just select everything after we join the tables. So, our SQL would look like this:

SELECT \*

FROM Employee e1

INNER JOIN Employee e2

ON e1.ManagerID = e2.EmployeeID

The results of running the query above would look like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| e1.EmployeeID | e1.Name | e1.ManagerID | e2.EmployeeID | e2.Name | e2.ManagerID |
| 1 | Sam | 10 | 10 | AnotherManager | NULL |
| 2 | Harry | 4 | 4 | Manager | NULL |

Note that there are only 2 rows returned – this is because an inner join is performed, which means that only when there is a match between employee ID’s and manager ID’s will there be a result returned. And since there are 2 people without managers (who have a manager ID of NULL), they will not be returned as part of table e1, because no employees have a matching ID of NULL.

Now, remember that we only want to return the names of the employee and corresponding manager as a pair. So, we can fine-tune the SQL as follows:

SELECT e1.Name, e2.Name

FROM Employee e1

INNER JOIN Employee e2

ON e1.ManagerID = e2.EmployeeID

Running the SQL above would return:

Sam AnotherManager

Harry Manager

Suppose that you are given the following simple database table called Employee that has 2 columns named Employee ID and Salary:

|  |  |
| --- | --- |
| Employee | |
| Employee ID | Salary |
| 3 | 200 |
| 4 | 800 |
| 7 | 450 |

Write a SQL query to get the second highest salary from the table above. Also write a query to find the nth highest salary in SQL, where n can be any number.

olution to finding the 2nd highest salary in SQL

Now, here is what the SQL will look like:

SELECT MAX(Salary) FROM Employee

WHERE Salary NOT IN (SELECT MAX(Salary) FROM Employee )

Running the SQL above would return us “450”, which is of course the 2nd highest salary in the Employee table.

SELECT TOP 1 Salary

FROM (

SELECT DISTINCT TOP N Salary

FROM Employee

ORDER BY Salary DESC

) AS Emp

ORDER BY Salary

In SQL, what is the default sort order of the Order By clause?

By default, the order by statement will sort in ascending order if no order (whether ascending or descending) is explicitly specified. This means that because the default sort order is ascending, the values will be sorted starting from the “smallest” value to the largest. This is true in all major RDBMS’s – including MySQL, Oracle, Microsoft SQL Server, Teradata, SAP, and others.

|  |
| --- |
| In SQL, what’s the difference between the having clause and the where clause? |
|  |
| The difference between the having and where clause is best illustrated by an example. Suppose we have a table called emp\_bonus as shown below. Note that the table has multiple entries for employees A and B.   |  | | --- | | emp\_bonus | | |  |  | | --- | --- | | **Employee** | **Bonus** | | A | 1000 | | B | 2000 | | A | 500 | | C | 700 | | B | 1250 | |   If we want to calculate the total bonus that each employee received, then we would write a SQL statement like this:   |  | | --- | | select employee, sum(bonus) from emp\_bonus group by employee; | |

The Group By Clause

In the SQL statement above, you can see that we use the "group by" clause with the employee column. What the group by clause does is allow us to find the sum of the bonuses for each employee. Using the ‘group by’ in combination with the ‘sum(bonus)’ statement will give us the sum of all the bonuses for employees A, B, and C.

Running the SQL above would return this:

|  |  |
| --- | --- |
| Employee | Sum(Bonus) |
| A | 1500 |
| B | 3250 |
| C | 700 |

Now, suppose we wanted to find the employees who received more than $1,000 in bonuses for the year of 2007. You might think that we could write a query like this:

|  |
| --- |
| BAD SQL:  select employee, sum(bonus) from emp\_bonus  group by employee where sum(bonus) > 1000; |

The WHERE clause does not work with aggregates like SUM

The SQL above will not work, because the where clause doesn’t work with aggregates – like sum, avg, max, etc.. Instead, what we will need to use is the having clause. The having clause was added to sql just so we could compare aggregates to other values – just how the ‘where’ clause can be used with non-aggregates. Now, the correct sql will look like this:

|  |
| --- |
| GOOD SQL:  select employee, sum(bonus) from emp\_bonus  group by employee having sum(bonus) > 1000; |

Difference between having and where clause

So we can see that the difference between the having and where clause in sql is that the where clause can not be used with aggregates, but the having clause can. One way to think of it is that the having clause is an additional filter to the where clause.

In SQL, what are the differences between primary, foreign, and unique keys?

The one thing that primary, unique, and foreign keys all have in common is the fact that each type of key can consist of more than just one column from a given table. In other words, foreign, primary, and unique keys are not restricted to having just one column from a given table – each type of key can cover multiple columns. So, that is one feature that all the different types of keys share – they can each be comprised of more than just one column, which is something that many people in software are not aware of.

Of course, the database programmer is the one who will actually define which columns are covered by a foreign, primary, or unique key. That is one similarity all those keys share, but there are also some major differences that exist between primary, unique, and foreign keys. We will go over those differences in this article. But first, we want to give a thorough explanation of why foreign keys are necessary in some situations.

## What is the point of having a foreign key?

Foreign keys are used to reference unique columns in another table. So, for example, a foreign key can be defined on one table A, and it can reference some unique column(s) in another table B. Why would you want a foreign key? Well, whenever it makes sense to have a relationship between columns in two different tables.

## An example of when a foreign key is necessary

|  |
| --- |
|  |

Suppose that we have an Employee table and an Employee Salary table. Also assume that every employee has a unique ID. The Employee table could be said to have the ‘master list’ of all Employee ID’s in the company. But, if we want to store employees salaries in another table, then do we want to recreate the entire master list of employee ID’s in the Employee Salary table as well? No – we don’t want to do that because it’s inefficient. It would make a lot more sense to just define a relationship between an Employee ID column in the Employee Salary table and the “master” Employee ID column in the Employee table – one where the Employee Salary table can just reference the employee ID in the Employee table. This way, whenever someone’s employee ID is updated in the Employee table, it will also automatically get updated in the Employee Salary table. Sounds good right? So now, nobody has to manually update the employee ID’s in the Employee Salary table every time the ID is update in the master list inside the Employee table. And, if an employee is removed from the Employee table, he/she will also automatically be removed (by the RDBMS) from the Employee Salary table – of course all of this behavior has to be defined by the database programmer, but hopefully you get the point.

## Foreign keys and referential integrity

Foreign keys have a lot to do with the concept of referential integrity. What we discussed in the previous paragraph are some of the principles behind referential integrity. You can and should read a more in depth article on that concept here: [Referential integrity explained](http://www.programmerinterview.com/index.php/database-sql/what-is-referential-integrity/).

## Can a table have multiple unique, foreign, and/or primary keys?

A table can have multiple unique and foreign keys. However, a table can have only one primary key.

## Can a unique key have NULL values? Can a primary key have NULL values?

Unique key columns are allowed to hold NULL values. The values in a primary key column, however, can never be NULL.

## Can a foreign key reference a non-primary key?

Yes, a foreign key can actually reference a key that is not the primary key of a table. But, a foreign key must reference a unique key.

## Can a foreign key contain null values?

Yes, a foreign key can hold NULL values. Because foreign keys can reference unique, non-primary keys – which can hold NULL values – this means that foreign keys can themselves hold NULL values as well.

## Some other differences between foreign, primary, and unique keys

While unique and primary keys both enforce uniqueness on the column(s) of one table, foreign keys define a relationship between two tables. A foreign key identifies a column or group of columns in one (referencing) table that refers to a column or group of columns in another (referenced) table – in our example above, the Employee table is the referenced table and the Employee Salary table is the referencing table.

DIFFERENCE BETN UNION AND UNION ALL –

The difference between Union and Union all is that Union all will not eliminate duplicate rows, instead it just pulls all rows from all tables fitting your query specifics and combines them into a table. A UNION statement effectively does a SELECT DISTINCT on the results

DIFFERENCE BETN DELETE TRUNCATE AND DROP

DROP and TRUNCATE are DDL commands, whereas DELETE is a DML command. Therefore DELETE operations can be rolled back

(undone), while DROP and TRUNCATE operations cannot be rolled back.

### DROP

The DROP command removes a table from the database. All the tables' rows, indexes and privileges will also be removed. No DML triggers will be fired. The operation cannot be rolled back.

SQL> DROP TABLE emp;

Table dropped.

SQL> SELECT \* FROM emp;

SELECT \* FROM emp

\*

ERROR at line 1:

ORA-00942: table or view does not exist

DELETE -

The DELETE command is used to remove rows from a table. A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed. After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it. Note that this operation will cause all DELETE triggers on the table to fire.

SQL> SELECT COUNT(\*) FROM emp;

COUNT(\*)

----------

14

SQL> DELETE FROM emp WHERE job = 'CLERK';

4 rows deleted.

SQL> COMMIT;

Commit complete.

SQL> SELECT COUNT(\*) FROM emp;

COUNT(\*)

----------

10

### TRUNCATE

TRUNCATE removes all rows from a table. The operation cannot be rolled back and no triggers will be fired. As such, TRUCATE is faster and doesn't use as much undo space as a DELETE.

SQL> TRUNCATE TABLE emp;

Table truncated.

SQL> SELECT COUNT(\*) FROM emp;

COUNT(\*)

----------

0

CROSS JOIN -

The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from the two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to True or where the join-condition is absent from the statement.

Syntax:

The basic syntax of CARTESIAN JOIN or CROSS JOIN is as follows:

SELECT table1.column1, table2.column2...

FROM table1, table2 [, table3 ]

CONSTRAINTS IN SQL –

In SQL, we have the following constraints:

* NOT NULL - Indicates that a column cannot store NULL value
* UNIQUE - Ensures that each row for a column must have a unique value
* PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
* FOREIGN KEY - Ensure the referential integrity of the data in one table to match values in another table
* CHECK - Ensures that the value in a column meets a specific condition
* DEFAULT - Specifies a default value for a column

UNIX

USEFULL COMMANDS

|  |  |
| --- | --- |
| Command | Description |
| cat | Display File Contents |
| cd | Changes Directory to dirname |
| chgrp | change file group |
| chmod | Changing Permissions |
| cp | Copy source file into destination |
| file | Determine file type |
| find | Find files |
| grep | Search files for regular expressions. |
| head | Display first few lines of a file |
| ln | Create softlink on oldname |
| ls | Display information about file type. |
| mkdir | Create a new directory dirname |
| more | Display data in paginated form. |
| mv | Move (Rename) a oldname to newname. |
| pwd | Print current working directory. |
| rm | Remove (Delete) filename |
| rmdir | Delete an existing directory provided it is empty. |
| tail | Prints last few lines in a file. |
| touch | Update access and modification time of a file. |

Manipulating data

The contents of files can be compared and altered with the following commands.

|  |  |
| --- | --- |
| Command | Description |
| awk | Pattern scanning and processing language |
| cmp | Compare the contents of two files |
| comm | Compare sorted data |
| cut | Cut out selected fields of each line of a file |
| diff | Differential file comparator |
| expand | Expand tabs to spaces |
| join | Join files on some common field |
| perl | Data manipulation language |
| sed | Stream text editor |
| sort | Sort file data |
| split | Split file into smaller files |
| tr | Translate characters |
| uniq | Report repeated lines in a file |
| wc | Count words, lines, and characters |
| vi | Opens vi text editor |
| vim | Opens vim text editor |
| fmt | Simple text formatter |
| spell | Check text for spelling error |
| ispell | Check text for spelling error |
| ispell | Check text for spelling error |
| emacs | GNU project Emacs |
| ex, edit | Line editor |
| emacs | GNU project Emacs |
| emacs | GNU project Emacs |

Compressed Files

Files may be compressed to save space. Compressed files can be created and examined −

|  |  |
| --- | --- |
| Command | Description |
| compress | Compress files |
| gunzip | Uncompress gzipped files |
| gzip | GNU alternative compression method |
| uncompress | Uncompress files |
| unzip | List, test and extract compressed files in a ZIP archive |
| zcat | Cat a compressed file |
| zcmp | Compare compressed files |
| zdiff | Compare compressed files |
| zmore | File perusal filter for crt viewing of compressed text |

Getting Information

Various Unix manuals and documentation are available on-line. The following Shell commands give information −

|  |  |
| --- | --- |
| Command | Description |
| apropos | Locate commands by keyword lookup |
| info | Displays command information pages online |
| man | Displays manual pages online |
| whatis | Search the whatis database for complete words. |
| yelp | GNOME help viewer |

Network Communication

These following commands are used to send and receive files from a local UNIX hosts to the remote host around the world.

|  |  |
| --- | --- |
| Command | Description |
| ftp | File transfer program |
| rcp | Remote file copy |
| rlogin | Remote login to a UNIX host |
| rsh | Remote shell |
| tftp | Trivial file transfer program |
| telnet | Make terminal connection to another host |
| ssh | Secure shell terminal or command connection |
| scp | Secure shell remote file copy |
| sftp | secure shell file transfer program |

Some of these commands may be restricted at your computer for security reasons.

Messages between Users

The UNIX systems support on-screen messages to other users and world-wide electronic mail −

|  |  |
| --- | --- |
| Command | Description |
| evolution | GUI mail handling tool on Linux |
| mail | Simple send or read mail program |
| mesg | Permit or deny messages |
| parcel | Send files to another user |
| pine | Vdu-based mail utility |
| talk | Talk to another user |
| write | Write message to another user |

Programming Utilities

The following programming tools and languages are available based on what you have installed on your Unix.

|  |  |
| --- | --- |
| Command | Description |
| dbx | Sun debugger |
| gdb | GNU debugger |
| make | Maintain program groups and compile programs. |
| nm | Print program's name list |
| size | Print program's sizes |
| strip | Remove symbol table and relocation bits |
| cb | C program beautifier |
| cc | ANSI C compiler for Suns SPARC systems |
| ctrace | C program debugger |
| gcc | GNU ANSI C Compiler |
| indent | Indent and format C program source |
| bc | Interactive arithmetic language processor |
| gcl | GNU Common Lisp |
| perl | General purpose language |
| php | Web page embedded language |
| py | Python language interpreter |
| asp | Web page embedded language |
| CC | C++ compiler for Suns SPARC systems |
| g++ | GNU C++ Compiler |
| javac | JAVA compiler |
| appletvieweir | JAVA applet viewer |
| netbeans | Java integrated development environment on Linux |
| sqlplus | Run the Oracle SQL interpreter |
| sqlldr | Run the Oracle SQL data loader |
| mysql | Run the mysql SQL interpreter |

Misc Commands

These commands list or alter information about the system −

td>groups

|  |  |
| --- | --- |
| Command | Description |
| chfn | Change your finger information |
| chgrp | Change the group ownership of a file |
| chown | Change owner |
| date | Print the date |
| determin | Automatically find terminal type |
| du | Print amount of disk usage |
| echo | Echo arguments to the standard options |
| exit | Quit the system |
| finger | Print information about logged-in users |
| groupadd | Create a user group |
| Show group memberships |  |
| homequota | Show quota and file usage |
| iostat | Report I/O statistics |
| kill | Send a signal to a process |
| last | Show last logins of users |
| logout | log off UNIX |
| lun | List user names or login ID |
| netstat | Show network status |
| passwd | Change user password |
| passwd | Change your login password |
| printenv | Display value of a shell variable |
| ps | Display the status of current processes |
| ps | Print process status statistics |
| quota -v | Display disk usage and limits |
| reset | Reset terminal mode |
| script | Keep script of terminal session |
| script | Save the output of a command or process |
| setenv | Set environment variables |
| stty | Set terminal options |
| time | Time a command |
| top | Display all system processes |
| tset | Set terminal mode |
| tty | Print current terminal name |
| umask | Show the permissions that are given to view files by default |
| uname | Display name of the current system |
| uptime | Get the system up time |
| useradd | Create a user account |
| users | Print names of logged in users |
| vmstat | Report virtual memory statistics |
| w | Show what logged in users are doing |
| who | List logged in users |

HOW TO FIND A STRING IN A FILE –

grep -rnw '/path/to/somewhere/' -e "pattern"

* -r or -R is recursive,
* -n is line number, and
* -w stands match the whole word.
* -l (lower-case L) can be added to just give the file name of matching files.
* Along with these, --exclude or --include parameter could be used for efficient searching. Something like below:

grep --include=\\*.{c,h} -rnw '/path/to/somewhere/' -e "pattern"

This will only search through the files which have .c or .h extensions. Similarly a sample use of --exclude:

grep --exclude=\*.o -rnw '/path/to/somewhere/' -e "pattern"

Above will exclude searching all the files ending with .o extension. Just like exclude file it's possible to exclude/include directories through --exclude-dir and --include-dir parameter; for example, the following shows how to integrate --exclude-dir:

grep --exclude-dir={dir1,dir2,\*.dst} -rnw '/path/to/somewhere/' -e "pattern"

HOW TO REPLACE A STRING IN A FILE –

sed is **s**tream **ed**itor, but can edit files directly too, with the following:

sed -i -e 's/foo/bar/g' filename

s is used to replace the found expression "foo" with "bar" and g used to replace any found matches.

-i option is used to edit in place on filename.

-e option indicates a command to run.

# In Unix, how do I get the line, word, or character count of a document?

In [Unix](https://kb.iu.edu/d/agat), to get the line, word, or character count of a document, use the wc command. At the Unix [shell](https://kb.iu.edu/d/agvf) prompt, enter:

wc filename

Replace filename with the file or files for which you want information. For each file, wc will output three numbers. The first is the line count, the second is the word count, and the third is the character count. For example, if you entered wc .login, the output would be something similar to the following:

38 135 847 .login

To narrow the focus of your query, use one or more of the following wc options:

| Option | Entities counted |
| --- | --- |
| -c | bytes |
| -l | lines |
| -m | characters |
| -w | words |

**Note:** In some versions of wc, the -m option will not be available or -c will report characters. However, in most cases, the values for -c and -m are equal.

For example, to find out how many bytes are in the .login file, you could enter:

wc -c .login

You may also pipe standard output into wc to determine the size of a stream. For example, to find out how many files are in a directory, enter:

/bin/ls -l | wc -l

For more information about wc, read its [man](https://kb.iu.edu/d/afjm) page. To do this, at the Unix prompt, enter:

man wc

# UNIX BASIC COMMANDS: LS - LIST

The **ls** command lists all files in the directory that match the *name*. If name is left blank, it will list all of the files in the directory.

## SYNTAX

The syntax for the **ls** command is:

ls [options] [names]

## OPTIONS

|  |  |
| --- | --- |
| Option | Description |
| -a | Displays all files. |
| -b | Displays nonprinting characters in octal. |
| -c | Displays files by file timestamp. |
| -C | Displays files in a columnar format (default) |
| -d | Displays only directories. |
| -f | Interprets each *name* as a directory, not a file. |
| -F | Flags filenames. |
| -g | Displays the long format listing, but exclude the owner name. |
| -i | Displays the inode for each file. |
| -l | Displays the long format listing. |
| -L | Displays the file or directory referenced by a symbolic link. |
| -m | Displays the names as a comma-separated list. |
| -n | Displays the long format listing, with GID and UID numbers. |
| -o | Displays the long format listing, but excludes group name. |
| -p | Displays directories with **/** |
| -q | Displays all nonprinting characters as **?** |
| -r | Displays files in reverse order. |
| -R | Displays subdirectories as well. |
| -t | Displays newest files first. (based on timestamp) |
| -u | Displays files by the file access time. |
| -x | Displays files as rows across the screen. |
| -1 | Displays each entry on a line. |

## EXAMPLE

ls -la

SCHEDULING JOBS IN UNIX

You may want to schedule some programs to run at later time or want them to run on a regular, repeating schedule. Examples of when you might want to do this are:

* If you have a program that uses a lot of the computer’s resources, you may want it to run at time when you will not be using the computer.
* To run system maintenance tasks, such as making backups or scanning the filesystem for viruses or other security concerns.

Unix has a facility for running scheduled tasks called **cron**, but users do run **cron** directly. It is always running in the background to run scheduled commands at the appropriate times. We call system programs, such as **cron**, that run in the backgroud *daemons*.

Note

The output from any scheduled commands, is by default sent to the user in an e-mail message. If this is not desired, then when scheduling the command, redirect the output from the command to a file or to the system provided trash can (/dev/null).

If you receive an e-mail message from the cron daemon, you might try using **mutt**, to read the message.

The programs that users use to schedule programs are **crontab** and **at**.

## crontab

SYNOPSIS

**crontab** [-u user] [file]

**crontab** [-u user] -l | -r [-i] | -e

DESCRIPTION

Crontab is the program used to install, deinstall or list the tables used to drive the cron daemon for running commands on a repeating schedule.

The first usage is to install a schedule of commands from a file or from standard input. If no options are specified, it reads from standard input.

-u user

Specify which user’s schedule to adjust (only for root). The default is for your own account.

-l

List the currently scheduled commands

-r

Remove the currently scheduled commands

-i

Prompt the user to confirm removal of the scheduled commands

-e

Edit the list of scheduled commands. After saving the temporary file and exiting from the editor, the commands entered in the file are scheduled. The default editor used is **vi**. If you wish to use another editor, such as **nano**, set your EDITOR environment variable:

export EDITOR='nano'

**crontab** **file format**

* Each scheduled command should be on one line of the file.
* The file should contain 6 fields separated by spaces.
* The first five field specify when the command should run (minute, hour, day, month, weekday). A wild-card (**\***) may used to specify every instance.[[1]](http://faculty.salina.k-state.edu/tim/unix_sg/advanced/schedule.html#f1) A specific time value, a list of times or a range of times may be used for any field.
  + The hour is in 24 hour time format (0-23).
  + The valid values for weekday are 0 to 7. Sunday is both 0 and 7. Monday to Saturday take the values 1 to 6.
* The last field specifies the command to run along with any arguments or redirection of the output. For some commands, it may be advisable to list the absolute path name of command rather than relying

## at

SYNOPSIS

**at** -r job | -l | *TIME*

DESCRIPTION

The **at** command is used to schedule a command to run one time.

-r job

Remove scheduled jobs, identified by their job number. This is the same as running atrm job

-l

List the scheduled jobs

TIME

*TIME* is a very flexible specification of when the command should run. See the text book for examples. **at** then reads from standard input for list of commands to run at this time. Type the EOF character (Cntrl-d) when finished entering commands.

Footnotes

|  |  |
| --- | --- |
| [[1]](http://faculty.salina.k-state.edu/tim/unix_sg/advanced/schedule.html#id2) | Give some though to where you might want to use a wildcard (**\***) character. A wildcard in the *day* field, means that that the day of the month is not important – other criteria such *weekday* will determine when the program runs. However, a wildcard in the *minute* field would mean that for the specified *hour*, the command runs once for every minute (60 times). |

PROCESS COMMAND -

The ps command

Type the following ps command to display all running process:  
# ps aux | less  
Where,

* -A: select all processes
* a: select all processes on a terminal, including those of other users
* x: select processes without controlling ttys

Task: see every process on the system

# ps -A  
# ps -e

Task: See every process except those running as root

# ps -U root -u root -N

Task: See process run by user vivek

# ps -u vivek