ChatGPT

Data Model for Sports Betting Exchange (Pre-Game & In-Play)

Overview: Designing a sports betting exchange (with both a traditional bookmaker model and a peer-to-peer exchange model) requires a robust data model. Key entities include **Events** (sports matches/games), **Teams/Participants**, **Markets** (betting options within an event), **Outcomes** (the possible results within a market), **Odds** (the price for each outcome), **Bet Selections** (picks on specific outcomes), **Bet Slips/Tickets** (the wager placed by a user, possibly containing one or multiple selections), and **Settlement** details (resolution of bets after the event). Additionally, advanced features like **parlays/multiples**, **in-play (live) betting** data, and a special **LineLock** financial object for locking odds must be accommodated. Below is a comprehensive breakdown of the data model components and their relationships, covering both pre-game and in-play scenarios, as well as bookmaker and peer-to-peer exchange functionality.

Sports Event (Match) Entity

Event: Represents a sporting event or match on which bets are offered. An event is typically defined by the sport/league, the competing teams or players, and the scheduled date/time. For example, an NFL game or a Premier League match would be an Event in the system. Key attributes of an Event include:

- **Event ID:** Unique identifier for the event.
- **Sport/Discipline:** The sport or category (e.g. football, basketball, tennis). An event belongs to one discipline (e.g., an English Premier League match is under Football) [1] . (Sports may be organized into disciplines and leagues; e.g. *Discipline* = *Football*, *League* = *NFL* or *Premier League*, and events under those leagues.)
- **Teams/Participants:** The competitors in the event (two teams in a match, or players in a tennis match, etc.). Each team or participant entity is linked to a sport/discipline [2] . For team sports, typically two teams (home and away) are listed; for individual sports, the players' names. The model may reference a **Team** table for each participant with details (team name, home city, etc.), or simply store participant names.
- **Event Date/Time:** The scheduled start datetime of the event (and possibly time zone or location). This is crucial for pre-game betting cutoff and for displaying upcoming events.
- **Location/Venue:** (Optional) Where the event is held, if relevant.
- **Status:** The current state of the event – e.g., *scheduled*, *in-play (live)*, *suspended*, *completed*, or *cancelled*. In-play betting is enabled when status is "live", and after completion the event is closed for betting.
- **Score/Live Data:** For in-play betting, the event may carry real-time data (current score, time elapsed, period/quarter, etc.). These aren't static fields in the initial bet ticket but are updated continuously via data feeds. (This helps determine when to suspend markets or settle bets.)
- **Result/Outcome:** After completion, the final result (e.g., final score or winning team) is recorded for settlement purposes. The winning outcome of each market will be determined from this.

Relationships: An Event is linked to its sport/league. It has multiple related **Markets** (each event can offer many betting markets). It also has references to the participating Teams/Participants. Importantly, users cannot create custom events – all events are defined and managed by the system (the operator provides the list of official games on which betting is allowed).

Market and Outcome Entities

Market: A Market represents a betting **market type** within an event – essentially a specific wager category. A market offers a set of two or more mutually exclusive outcomes on which bets can be placed [3]. Common markets for major sports include: - *Moneyline* (winner of the game), - *Point Spread/Handicap* (team to win with a points handicap applied), - *Total (Over/Under)* (combined score over or under a line), - *Prop Bets* (proposition bets like specific player to score, etc.), - *Futures* (e.g., championship winner), etc.

Each Market has key attributes:
- **Market ID:** Unique identifier for the market.
- **Event ID:** Reference to the Event the market belongs to. (Each market is associated with exactly one sporting event.)
- **Market Type/Name:** The kind of bet. For example, "Match Winner (1X2)", "Point Spread", "Total Points", "First Goal Scorer", etc. This might also encode parameters (e.g., "Over/Under 2.5 goals"). Some systems use **Market Templates** to define standard market structures (like a template for 1X2 or Over/Under) that can be reused across events [4].
- **Parameters:** If applicable, any line or handicap values. For instance, a point spread market might have a handicap of +5.5 points for the underdog; a totals market might have a line of 2.5 goals. These values can be stored as part of the market definition (or the outcome, depending on modeling choice).
- **Outcomes:** A collection of Outcome entries (see below) that belong to this market. For example, a Moneyline market has typically 2 outcomes (Team A win, Team B win) in a two-team sport or 3 outcomes in a sport that allows a draw (Team A, Draw, Team B) [3]. A spread market might also have two outcomes (Team A -5.5, Team B +5.5), etc.
- **Status:** Market status (e.g., *open*, *suspended*, *closed*). During an event, markets might be suspended (temporarily locked) when something significant happens (like a red card or score in soccer) to allow oddsmakers or the system to update odds. Once the event is finished, markets are closed and settled.
- **Odds Format:** (Optional) an indicator of how odds are represented (decimal, fractional, American). The system might handle multiple formats for display, but internally odds can be stored as a decimal for consistency (e.g., 2.50, 1.80 etc.). This is more of a user preference for display, not a separate entity.

Outcome: Outcomes are the specific possible results within a market. Each Outcome is one option a user can bet on. Key attributes:
- **Outcome ID:** Unique ID for the outcome (usually unique within a market).
- **Market ID:** Reference to the parent Market.
- **Name/Description:** A description of the outcome selection. For example: "Team A to win", "Team B to win", "Draw", "Over 2.5", "Under 2.5", etc. This often includes the team name and/or the condition. In the data model, the Outcome might also carry the parameter for clarity (e.g., an Outcome for a spread could be labeled "Team A -5.5").
- **Odds (Current Price):** The latest odds being offered for this outcome. In a **bookmaker model**, this is the fixed odd set by the house. In a **P2P exchange model**, there isn't a single fixed odd – instead, there is an **order book** of offers. In an exchange, you might derive *best available odds* from the best unmatched offers. The data model for an exchange might not store odds directly on the outcome, but rather store available

offers (detailed in the exchange section below). However, for convenience, the system may display the best back/lay odds for each outcome.
- **Outcome Result:** After the event, a flag or status to mark if this outcome was the winner or not (or if it was a push/null). For example, in a Moneyline market, exactly one outcome is marked as win = true. For point spreads or totals, an outcome can be a push (tie) if the result lands exactly on the handicap line, which might result in refunds. This can be represented by an outcome status like *won/lost/push*.

Relationships: Each Market belongs to one Event, and has multiple Outcomes. Each Outcome belongs to one Market. A given outcome (like "Team A wins") could appear in multiple markets only if those markets are different contexts (e.g., "Team A wins first half" vs "Team A wins full game" – those would be separate outcomes in different markets). In general, the combination of Event + Market + Outcome fully defines a unique betting option (often called a **Selection** in betting systems [3]).

Example: For an NBA game (Event) between Lakers vs Celtics, one Market might be **Moneyline** with Outcomes: *Lakers to Win* and *Celtics to Win* (each with odds). Another Market could be **Total Points 210.5** with Outcomes: *Over 210.5* and *Under 210.5*. Each of these outcomes has odds that update over time (especially if in-play).

Bet Selection (Selection)

A **Selection** refers to a specific choice a bettor can make – essentially an Outcome on a given Event/Market **at a locked odds value at the time of betting**. In some models, “Selection” is used to denote the tuple of (Event + Market + Outcome) [3]. In the context of a bet ticket, a Selection also carries the odds that were accepted when the bet was placed (because odds might change later, but the bet is locked at the odds it was placed).

In the data model, when a user picks something to bet on, the system creates a *Bet Selection* record containing:
- **Event reference:** Which event the selection is in.
- **Market reference:** Which market within that event (e.g. Moneyline).
- **Outcome reference:** The specific outcome chosen (e.g. Team A to win).
- **Odds (Booked Odds):** The odds that were locked in for this selection at bet placement time [5] [6]. For fixed-odds bets, this is the sportsbook’s odds at that moment (fixed odds betting means the odds are locked when you place your bet [6]). For exchange bets, this would be the agreed odds after matching with a counterparty. Storing the exact odds is critical, as it determines payout.
- **Line/Handicap Value:** (If applicable) If the market has a variable line (e.g. spread or total), the specific line value should be recorded as part of the selection (e.g., +5.5 points, or over 210.5). This could be derived from the Market or Outcome definition, but recording it with the bet selection ensures clarity, especially if lines move for new bets later.
- **Stake (for this selection):** In a single bet, the stake is associated with the bet as a whole. In a multi-selection parlay, usually one stake covers all selections collectively (we detail multi-bet handling later). But for clarity, some models still break down how much of the total stake is conceptually on each leg. For straight bets, the selection stake equals the bet stake.
- **Selection Status:** Won/Lost/Pushed – after the event, this indicates whether this particular leg won or not (used in calculating parlay outcomes).

A Selection is usually not a standalone top-level entity the user sees, but part of a **Bet Slip**. However, it's useful in the model to encapsulate the choice and odds. In code or database terms, you might have a BetSelection join table connecting Bets to Outcomes (including the odds and line at placement).

Bet Slip / Betting Ticket Entity

The **Bet Slip** (or betting ticket, wager) represents the actual bet placed by a user. This is the record of a financial transaction where the user risks some amount (stake) on one or multiple selections with the goal of winning if the predictions are correct. Understanding a bet slip's components is crucial – it contains all details of the wager [7]. Key attributes of a Bet (ticket) include:

- **Bet ID / Ticket ID:** A unique identifier for the bet slip. This might be shown as a reference number or QR code on a ticket [8], useful for users to track or for customer support to lookup.
- **User ID:** The account that placed the bet (for an exchange, there could be two users ultimately matched on opposite sides, but each will have their own bet record for their side).
- **Selections:** One or multiple selections that this bet comprises. A bet can be:
- **Single Bet:** one selection. (The simplest case – win or lose based on one outcome.)
- **Parlay/Accumulator:** multiple selections combined into one bet, where *all* must win for the bet to pay out. The odds are compounded (multiplied) to form a higher potential payout, but a higher risk since one loss means the whole bet loses. The Bet record would list multiple selection references.
- **System/Round-Robin Bets:** multiple selections forming several combinations. For example, a 3-selection round-robin might be three doubles and one treble. These are essentially a collection of smaller bets; they can be represented either as multiple Bet records or one parent Bet with sub-bet details. (Advanced, but the data model can accommodate by linking sub-bets or by a structure to indicate combination bets.)
- **Bet Type:** A field to indicate if the bet is a single, parlay, system, etc. [9]. This helps in both UI and settlement logic.
- **Stake (Wager Amount):** The amount of money risked on this bet [10]. For a single bet, this is the amount on that one selection. For a parlay, this is the total wager that covers all legs collectively. (For system bets, stake might be per combination or total – can be more complex.)
- **Odds/Payout Calculation:** For single bets, the odds of the selection determine the payout. For parlays, the effective odd is the product of all selections' odds (if using decimal odds, multiplication gives the parlay decimal odd). The data model might either store the *final odd* for parlays or simply store each selection's odds and calculate on the fly.
- **Potential Payout:** The amount that would be returned if the bet wins, including profit and sometimes stake. This is typically *Stake* × *Odds* (for decimal odds) or computed accordingly [11]. Many systems will display this on the slip for the user's reference. It can be derived and not necessarily stored, but it might be cached for quick display.
- **Date/Time Placed:** Timestamp when the bet was placed [12]. This is important for audit, for determining if a bet was placed before event start (or in-play), etc. The slip might also show the event date/time [12] for context, but the event time is in the event entity.
- **Bet Status:** Current state of the bet – e.g. *pending (open)*, *won*, *lost*, *cancelled*, *void*.
- Pending means the event(s) haven't finished yet (or are in progress).
- Won/Lost are final outcomes after settlement.
- Cancelled/Void might occur if an event is cancelled or a selection is voided (in which case rules might reduce a parlay or refund).

- In live betting, a bet might also be in a temporary state like *"bet acceptance pending"* if there's a slight delay to confirm odds. (Many sportsbooks have a delay for in-play bets – if the odds change or a critical event happens in those seconds, the bet may be rejected or require re-confirmation.)
- **Settlement Details:** When settled, the bet record will have:
- **Result:** Win/Lose/Push (push = tie or void selection resulting in refund or partial payout). For parlays, if one leg pushes, typically the parlay reverts to the next lower number of legs. The result field can capture if the bet is fully won, lost, or partially won (in system bets cases).
- **Payout Amount:** How much was paid out to the user (profit plus returned stake if any) when settled. For a win, this is stake * odds (or the accumulated formula). For a loss, 0 (stake lost). For a push, typically the stake returned (so payout = stake).
- **Settled Time:** Timestamp when the bet was settled (i.e., when results were confirmed and the bet closed).
- **Bookmaker/Exchange info:** If the model supports both exchange and house bets, we might have a flag or reference indicating whether this bet was placed *"Against House"* (bookmaker) or *"Exchange"*. In an exchange, technically the user's bet is matched with one or more counterparty offers – but from the user's perspective, they have a normal bet slip. The system might internally mark exchange bets differently for accounting (and commission calculation on wins, etc.). For a bookmaker bet, the "bookmaker" is the house itself. (If needed, **Bookmaker ID** could even be stored if the platform aggregates odds from multiple books, but assuming a single operator system.)

All the information on a typical betting slip given to a user would be drawn from these fields – the selections (with event and outcome names), the odds, the stake, the potential payout, the date/time, and some ticket ID [13] [14]. For example, a slip might show: *Selection: Team A to win vs Team B (Moneyline) @ 2.5 odds; Stake: $100; Potential Payout: $250; Bet Type: Single; Placed on 2025-11-13 21:30.* All these correspond to the data model elements described.

Bet Settlement Process

Settlement is the process of grading the bet as a win or loss (or push) after the event outcome is known, and then crediting or debiting the user's balance accordingly. In the model, settlement isn't a separate entity but a phase where certain fields in the Bet (and possibly BetSelection) are updated. However, understanding it is key to the data design:

- Once an Event is completed and official results are in, the system will mark the winning Outcome(s) for each market. For example, in a Match Winner market, the outcome corresponding to the actual winner team is marked as the winner. In a totals market, if the total score is above the line, "Over" is the winning outcome, etc.
- The system then looks at all Bet Selections on that event's markets:
- If a selection's outcome matches the winning outcome, that selection is a win.
- If it doesn't match, it's a loss. If the outcome is void/push (e.g., a tie against the spread), that selection is void (neither win nor loss; typically stake returned for that leg).
- For each Bet (ticket), determine the overall result:
- A single bet is won if its sole selection won, lost if it lost, or push if voided.
- A parlay is won if **all** selections won; if any selection lost, the whole parlay loses; if one selection is voided (and none lose), the parlay payout is usually recalculated without that leg (effectively reducing the parlay length).

- A system bet (e.g., trebles, doubles combinations) will have some combinations win and others lose – these are effectively multiple smaller bets. The model might break these out or compute aggregate outcome.
- Payout calculation: For each winning bet, the payout = stake * (product of odds of winning selections) (plus returning the stake depending on convention). This amount is credited to the user's balance.
- The Bet record is updated with **status = settled** and result = win/loss/push, along with the payout field.

It's critical that the system maintain an **accurate record of bets, odds, and results** for transparency and dispute resolution [15]. Bet settlement is the "moment of truth" – it's when the bet is formally declared a win or loss and winnings or losses are tallied and paid out [16]. The data model should ensure that for every bet you can trace what event and outcome it was tied to and what the result was, to facilitate audits or resolving any customer disputes.

Note: Settlement timing for in-play bets might be immediate at game end, whereas some special markets (like player award futures) settle much later. The model might include a notion of **settlement status per selection** especially if a bet has components that settle at different times (though in sports it's usually all at once when the game is over, except in a parlay spanning multiple events).

Pre-Game vs. In-Play Betting Considerations

The data model for the most part remains the same for pre-match and live betting, but there are a few additional considerations for in-play:

- **Odds Updates:** In-play odds change rapidly as the game progresses. The system will be receiving new odds from traders or an automated feed. The data model needs to allow odds on an Outcome to be updated frequently. If using a relational DB, one might store odds history or at least current odds. In an exchange, new user orders might come in constantly.
- **Bet Delay & Confirmation:** Although not a schema element, it's worth noting that live bets often go through a brief "pending" state where the system confirms the odds didn't move while the bet was being placed. In the model, a Bet could have a status like "pending acceptance" for those few seconds, then become "confirmed". If odds moved or a critical event happened, the bet might be rejected (or repriced) – if rejected, the Bet may be marked cancelled or never recorded as confirmed.
- **Event State Data:** As mentioned, the Event entity might carry some live state (score, game clock) primarily for the business logic to know when to suspend markets or void certain bets (e.g., if a player who a prop was on doesn't play, etc., those could be void). The model might not store every moment's data, but could have fields like *current_period*, *time_remaining*, *team_a_score*, *team_b_score* in the Event for quick reference, or this could be entirely handled in memory with feed data.
- **Suspensions and Market Closure:** In-play, markets (particularly those not covering full-game, like next team to score) may open and close during the event. For example, a "Next Goal" market closes once a goal is scored and then may reopen for the following goal. The model should allow Market status changes and possibly have validity times (open from X to Y).
- **Live Specific Markets:** Some markets exist only in-play (like "Next point winner" in tennis, or various micro-bets). These would just be additional Market and Outcomes tied to the Event, with perhaps a short lifespan. The data model doesn't need a fundamentally different structure, but the **Event Part** concept mentioned in the reference can be useful [4] – e.g., markets could be scoped to a part of

the game (1st half, 2nd half, etc.). If modeling that, an Outcome or Market might have a field indicating which period it applies to, or the Event Part as a separate entity (First Half as an event segment). This is an optional extension for complex live betting coverage.

In summary, pre-game vs in-play mostly affects how odds are managed and how quickly the data updates, rather than the static structure. The same event->market->outcome->bet chain is used, just with more frequent updates and the event status toggling to live.

Peer-to-Peer Exchange Model vs Bookmaker Model

Our data model must support both a traditional **bookmaker** (house-driven) model and a **peer-to-peer exchange** model:

- **Bookmaker Model:** The sportsbook (house) offers odds on each outcome. The **Odds field** on each Outcome is set by the house. When a user places a bet, the counterparty is implicitly the house. The **Bet** record will indicate a contract between user and house. The house's profit comes from the built-in "vig" (odds margin). The data model elements we described (Event, Market, Outcome, Bet) suffice; there isn't a need for additional entities to represent the other side of the bet, since the house is not explicitly modeled as a user (though the operator will track liability per outcome behind the scenes).
- The bookmaker model may include additional data like **limit management** (max stake allowed on certain bets) or **odds history**, but these can be auxiliary. A **Risk Management** module might adjust odds or reject bets that exceed certain liability – those aspects connect to the data (like sum of stakes on each outcome), but can be computed from Bet records per outcome.

- **Exchange Model:** In a betting exchange, users bet against each other by proposing odds and stakes. The exchange operator doesn't risk its own money on bets, but usually takes a commission on winners. This requires a few **additional entities** or fields in the model:

- **Bet Offer / Order:** When a user wants to place a bet on an exchange, if their desired odds are not immediately available, the system creates an **offer** (order) that sits in the market. An Offer includes:
    - *Offer ID*, *User ID*, *Selection (Event/Market/Outcome)*, *Offered Odds*, *Stake Amount*, *Side (Back or Lay)*, *Remaining unmatched stake*, *Timestamp*, *Status (open/partially_matched/matched/cancelled)*.
    - "Back" means the user is betting *on* the outcome to happen (like a normal bet). "Lay" means the user is betting *against* that outcome (essentially acting as the bookmaker for that selection). In exchange terminology, every bet has two sides: one backing, one laying.
    - Multiple Offers can exist at different odds for the same outcome, creating an order book. For example, one user might offer to back Team A at odds 2.0 for $100, another might offer at 2.1 for $50, etc. Conversely, on the lay side, users propose odds to lay.
- **Matching Engine:** This is the logic (not a single table) that matches opposing offers. When a user submits a bet request:
    - If they chose an existing odds (say they accept the best available offer), the system matches them immediately with one or more counter-offers. If fully matched, a **Bet** record is created for each user. If partially matched, the remainder becomes an Offer.
    - If they posted a new offer (e.g., requesting better odds than currently available), it goes into the order book and waits for someone else to take it.

- **Matched Bet / Trade:** In some models, once matched, the result could simply be recorded as two Bet records (one for each user with opposite sides). Each Bet references the selection and odds, etc., as usual, but perhaps with a flag that it's an exchange bet and who the counterparty was. Alternatively, one could have a separate entity like **MatchedPair** linking the two users' positions. However, since the outcomes are binary (if one side wins, the other loses), it's enough to record individual bets for each user for accounting, as long as the linkage can be traced (for example, by a common match ID or linking both to the same offer match).
- **Liquidity Pool:** Not an entity per se, but note that in an exchange, the available odds at any time for an outcome come from the unmatched offers. The *best back odds* is the highest odds among lay offers from others (since as a bettor you want the highest payout), and *best lay odds* is the lowest odds someone is willing to back (since as a layer you want the lowest risk odds). These are dynamic. The model might not explicitly store "best odds" because it's derived from offers, but it might cache it for quick retrieval in a field on the Market/Outcome (updated whenever offers change). For example, Outcome could have fields *best_back_odds* and *best_lay_odds* for display.

User Interaction: In practice, a user on an exchange sees a interface similar to an order book (not unlike a stock exchange), or a simplified view of best odds to back or lay. But underlying, the data model is recording offers and matches.

Differences in Bet Records: In an exchange, when User A backs Team A for $100 at 2.5 against User B (who effectively lays Team A), the system might record: - Bet for User A: Selection = Team A win, Odds 2.5, Stake $100, Payout potential $250 (profit $150 plus return $100 stake if win). If Team A wins, this bet is settled as win (User A gets $250). If Team A loses, this bet loses (User A loses $100). - Bet for User B: What is User B betting on? Effectively, User B is saying Team A will **not** win (i.e., Team B wins or draw). This could be modeled as a Lay bet. Often, exchanges don't explicitly list the "not win" outcome as a separate stored outcome (except in two-outcome events where laying one side is equivalent to backing the other). For clarity, we might record User B's bet as: Selection = Team A *to not win* (or simply a Lay on Team A), Odds 2.5, Stake $150 *liability* for a $100 lay. (In lay betting, the stake concept flips: if you lay at 2.5 odds with a backer staking $100, the layer's liability is $150 because that's what they pay if they lose the lay.) The data model might store **liability** for lay bets instead of stake, or store both: lay stake vs payout. - Both bets would be linked by a match or common reference. Both will settle opposite (if Team A wins, A's bet wins, B's bet loses $150; if Team A loses, A's loses $100, B's wins $100 profit). - The exchange typically charges commission on net winnings of a bet (e.g., 5%). That might introduce a **Commission** field or a separate transaction. In the data model, perhaps simply note commission percentage somewhere (by user or by bet).

In summary, to support the exchange, **BetOffer** (for unmatched bets) and possibly a way to mark **Bet** records with side/back-lay is needed. The rest of the model (Event, Market, Outcome) remains the same. A betting exchange essentially uses the same event and outcome structure; it just changes how odds are sourced and how bets are matched.

Importantly, **users cannot create arbitrary new markets/events** in this model – they can only propose odds on existing markets that the system defines. (In some theoretical exchanges, users might request new bet propositions, but as per requirements, we assume all markets are predefined in the system and users choose from those. No custom user-created events or markets are allowed, which simplifies the model.)

Advanced Bet Types and Features

Modern sportsbooks and exchanges offer advanced betting options beyond straight singles. Our data model is designed to accommodate these:

- **Parlays/Accumulators:** As discussed, the Bet entity linking multiple selections covers this. Each selection is still an Event/Market/Outcome. The *bet type* field can indicate “Parlay” with, say, 4 legs. No fundamental change in the schema is needed, just the ability for a Bet to reference multiple selections (one-to-many relationship). The payout calculation for a parlay is multiplicative of the leg odds (or using formula for American odds, etc.), but that’s a computation detail. Some systems choose to store the parlay odds or payout once calculated.
- **Round Robins / System Bets:** These are essentially multiple bets packaged together. For instance, a 3-leg round robin of doubles consists of 3 choose 2 = 3 separate two-leg parlay bets. One way to model this is to have a parent Bet (for the whole round robin) and child Bet entries for each combination, or a Bet table that can handle grouping. Another approach: treat each combination as an independent bet with a reference to a common group ID. For simplicity, one could flatten them into separate Bet records. The model should at least allow linking them or storing the system bet parameters (e.g., "system 2/3" meaning 2-leg combos out of 3 selections). The reference from earlier shows generating combination bets in code [17] [18], indicating multiple Bet objects are created for the system.
- **Prop Bets and Special Markets:** These are just additional market types (like player-specific or statistic-specific markets). The model supports them by having flexible Market definitions. For example, a market “Player to Score” could have outcomes for each player (that’s a case where a market might have many outcomes, not just 2 or 3). Our Outcome table would handle that list of players as outcomes. Another example: “Exact Score” market could have dozens of outcomes (each possible scoreline). There’s no structural change, just more outcome entries.
- **Same-Game Parlay / Bet Builder:** This feature allows combining correlated selections from the same event (e.g., Team A to win AND Over 200 points in the same game). From a data perspective, this is still a parlay, but the system needs to ensure it calculates odds appropriately (often via a special algorithm since outcomes are not independent). The model might include a marker or separate handling for such correlated parlays (some systems treat it as a special market of its own called a "BetBuilder market" or so). However, one can treat it simply as a parlay with a flag "SameGameParlay = true" for awareness. The underlying pieces (selections) are still normal outcomes.
- **Cash Out:** Cash out is a feature where a user can settle their bet before the event ends for a certain value. Data-model wise, if a bet is cashed out, it could be marked as *cashed out* (a final state similar to settled) and have a payout equal to the cashout amount (which is typically less than full winnings but more than zero). The bet is then effectively closed. The model might need a field for **cashed_out_flag** and the cashout value. This might also link to a new transaction. Since the user specifically didn’t ask, we just note it as a possible field in an extended model.
- **Bonuses/Free Bets:** If needed, a bet might be marked if it used a free bet stake (so accounting knows not to return stake, etc.). This could be a boolean on Bet (freeBet=true) or a separate voucher entity reference.
- **Result Overrides and Adjustments:** In rare cases, events can be overturned or voided after the fact. The model should allow updating a settlement (with audit trails), but that’s more of an operational detail than a structure concern.

LineLock – Odds Locking Feature

The **LineLock** is a special financial instrument in this system: it allows a user to **lock in a specific odds/line for a future bet by paying a fee**. It's akin to buying an option – the user secures the price now and has the right to place the actual bet at those odds later, regardless of market movements. We will model LineLock as its own entity:

LineLock Entity:
- **LineLock ID:** Unique identifier for the lock transaction.
- **User ID:** The user who purchased the lock.
- **Event/Market/Outcome reference:** The specific selection for which odds are locked. For example, *Outcome = Team A to Win* in *Event X*. This ties the lock to the same identifiers as a regular selection so we know what it's for.
- **Locked Odds (Strike Price):** The odds that are locked in for the user [19]. For example, maybe Team A is currently 2.0 odds; by paying a fee, the user locks that price. Even if the odds later drop to 1.8, the user can still bet at 2.0. This is essentially the "strike" odd. The format could be decimal odds or the native system format.
- **Lock Fee:** The amount paid for this lock. This is often charged upfront (like a premium). It could be a flat fee or a percentage of potential stake; business rules define it. The model records it as a transaction (likely separate in a transactions ledger but also stored here for reference).
- **Max Stake or Bet Amount Allowed:** The lock might only guarantee the odds for a certain maximum stake. For instance, the user can lock Team A @ 2.0 for up to $100 stake. If they want to bet more, the excess might not be guaranteed at that price. We'd store the max stake that the lock covers.
- **Expiration Time:** The lock isn't indefinite; it expires after a certain period or by a deadline (especially for pre-game, likely until game start, or shorter in live). For example, the lock could be valid for 60 seconds in live betting (common in some exchange features) or perhaps a longer window pre-game (minutes or hours) depending on product design. We record the datetime until which the user can exercise the lock. After this, the LineLock becomes void if not used.
- **Lock Status:** *Active* (lock is currently holding the price open), *Used* (the user exercised it and placed a bet using this lock), or *Expired* (not used in time), *Cancelled* (if the event is cancelled or user refunded). If used or expired, it's no longer active. If used, we may link it to the bet that was placed.
- **Associated Bet ID:** (Optional) If the user exercised the lock and placed a bet, we can store a reference to the Bet ID that consumed this lock. This ties the chain: from paying the fee to actually placing the bet. The odds on that bet should match the locked odds.

With this model, the workflow is: the user sees an attractive odd and buys a LineLock to freeze it. The system creates a LineLock record with current odds, and possibly behind the scenes it might ensure the liquidity or house liability to honor that odd is reserved. The user later decides to place the bet – when they do, the system checks their active LineLock for that selection, verifies it's still valid, then creates a Bet at the locked odds (even if current odds have changed) and marks the LineLock as used. If the user never bets in time, the LineLock expires worthless (like an expired option).

This concept has precedent: for example, BetConnect (an exchange platform) introduced a feature where bettors can **lock in odds for 60 seconds** to protect against price movements [19]. In that case, no fee was mentioned for BetConnect's timer (it's more of a user experience feature), but one could monetize it by charging a fee. Our model's **LineLock Fee** covers that monetization. Another hypothetical example: a

sportsbook might allow you to pay $5 to lock a parlay's odds for an hour while you finalize your bet – ensuring if odds drift you keep the original price.

From a data standpoint, a LineLock is separate from a Bet because it's not a bet itself (no stake on an outcome placed, just a right to place later). However, it's closely related; consider it a *derivative object* linked to a future bet. Financially, it might also be logged in a transactions ledger (fee collected).

Putting it Together – Summary of Entities and Relationships

To summarize the core data model components in a structured way (covering both exchange and sportsbook aspects):

- **Sport/Discipline** – e.g., Football, Basketball, etc. (Mostly hierarchical classification, with relationship to leagues and teams.)
- **League/Tournament** – groups events (e.g., NBA, English Premier League).
- **Team/Participant** – entities for teams or individual players, linked to a sport (and possibly leagues).
- **Event (Match)** – identified by sport/league, has date/time, teams/participants, status, and links to one or more betting markets [1].
- **Market** – a betting market within an event (e.g. Moneyline, Spread, Totals, Props), characterized by a set of outcomes [3]. Related to one event.
- **Outcome** – a possible outcome within a market (e.g. Team A wins, Team B wins, Draw), each with odds (if fixed-odds) or associated offers (if exchange). Outcomes relate to one market [3].
- **BetOffer (Exchange only)** – represents an unmatched bet offer from a user (to back or lay an outcome at certain odds and stake). Linked to an outcome (or selection) and a user.
- **Bet (Bet Slip / Ticket)** – a placed bet by a user. Contains one or multiple selections (outcomes chosen with locked odds) [3]. Linked to the user, and if exchange, possibly references the offers matched. Key fields: stake, type (single/parlay/etc) [9], status, payout, timestamps, etc. [8] [12].
- **BetSelection** – join entity between Bet and Outcome, capturing the specific selection details (event/market/outcome, odds at placement [5], maybe line value, selection result after settlement).
- **LineLock** – an optional entity for the odds-lock feature, linking a user to an outcome with a guaranteed odd for a time, for a fee (like an option contract) [19]. It gets exercised or expires.
- **Settlement Data:** (Not a separate table, but conceptually) The results of events propagate to bets. Could also consider a **Result** entity that records final scores and winning outcomes per event, which could be used to update bets.

Everything above ensures that for *big market sports* (football/soccer, basketball, etc.) both **pre-game** and **in-play** betting is supported. The model is flexible for different bet types and even novel features like odds locking. It's aligned with how real betting systems are structured: for example, the concept of event/market/outcome aligns with industry standards (sometimes called **Event/Market/Selection model** in many sportsbook platforms, or **fixture/market/outcome**). We've also incorporated exchange-specific needs by adding an order layer.

By researching industry practices and common models, we arrived at a comprehensive schema. The domain objects identified (Match, Market, Outcome, Selection, Bet) are standard in betting software [20], and the additional constructs (exchange orders, LineLock) address the extended functionality. This data

model should serve as a solid foundation for implementing a sports betting exchange platform with both peer-to-peer betting and traditional bookmaker features.

Sources: The model draws on known sportsbook domain definitions [20] and real-world betting slip components [21] [14], as well as industry commentary on bet settlement [15] [16]. The LineLock feature is inspired by recent innovations allowing odds to be temporarily fixed [19]. All these elements together represent best practices for a sports betting exchange data model.

---

[1] [2] [3] [4] [17] [18] [20] Risk Models
https://alexandrugris.github.io/statistics/2018/08/15/risk.html

[5] [7] [8] [9] [10] [11] [12] [13] [14] [21] How to Read a Betting Slip - VSiN
https://vsin.com/how-to-bet/how-to-read-a-betting-slip/

[6] Betting Basics: What are Fixed Odds? | Cloudbet Blog
https://www.cloudbet.com/en/blog/posts/what-are-fixed-odds

[15] [16] Why Are Betting Settlements Important For Your Sportsbook Operation?
https://www.lsports.eu/blog/what-are-betting-settlements/

[19] BetConnect Show you how to Match Bet on Football | BetConnect
https://info.betconnect.com/blog/matched-betting-on-football-betconnect-show-you-how/